Open Standards for Embedded Compute and Vision Acceleration

June 2024

Neil Trevett
VP Developer Ecosystems, NVIDIA
President, Khronos Group

Khronos Connects Software to Silicon

Close to 200 members worldwide
Any organization is welcome to join

Liaisons: Cooperation with industry associations and organizations

Founded in 2000
~ 200 Members | ~ 40% US, 30% Europe, 30% Asia

KHRONOS GROUP

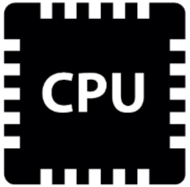
Open, royalty-free interoperability standards to harness the power of GPU, XR and multiprocessor hardware

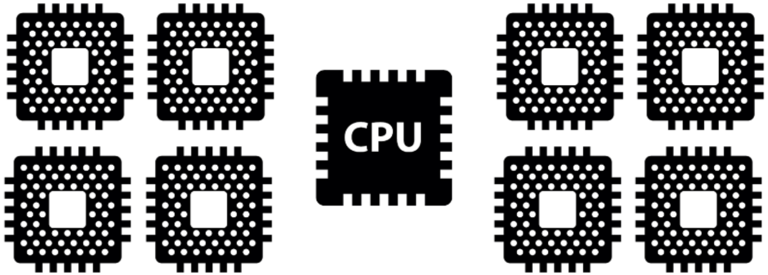
3D graphics, augmented and virtual reality, parallel programming, inferencing and vision acceleration

Non-profit, member-driven standards organization, open to any company

Proven multi-company governance and Intellectual Property Rights Framework

The Need for Parallel Processing





Single Processor

Simple to program *but* may not provide enough performance

especially

as Moore's Law frequency/power scaling is slowing

Multi-Processor

Additional processors can process expanded workloads *but a*dds complexity to system design and programming:

(i) Divide workload into kernel programs for distribution across available processors

(ii) Synchronize use of compute and memory resources

(iii) Communicate intermediate data and results

Open standard APIs and languages can help manage this complexity

What is an Open Interoperability Standard?

Open Standards

INTEROPERABILITY is precisely specified COMMUNICATION

E.g., software to hardware, client to server

OPEN standard specifications are created through multi-company cooperation under an agreed IP framework

Open standard specifications PLUS conformance tests enable MULTIPLE CONSISTENT IMPLEMENTATIONS to meet the needs of diverse markets, price points, and use cases

Open Source

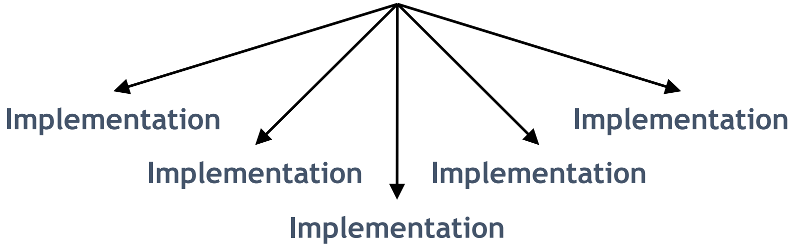
Open-source projects are created through multi-company cooperation and software effort via a contribution license

Design governance ranges from narrow to broad

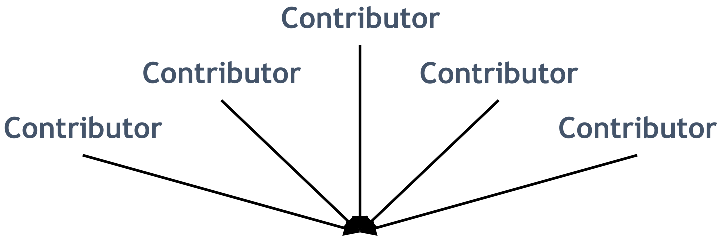
Depending on project's history and purpose

Open *standards* often use open *source* to share the development effort for sample implementations, tools, samples, conformance tests, validators, etc.

Open Standard = Shared Specification



Often used for HARDWARE APIs to enable competition between diverse implementations without fragmentation



Open Source = Shared Implementation

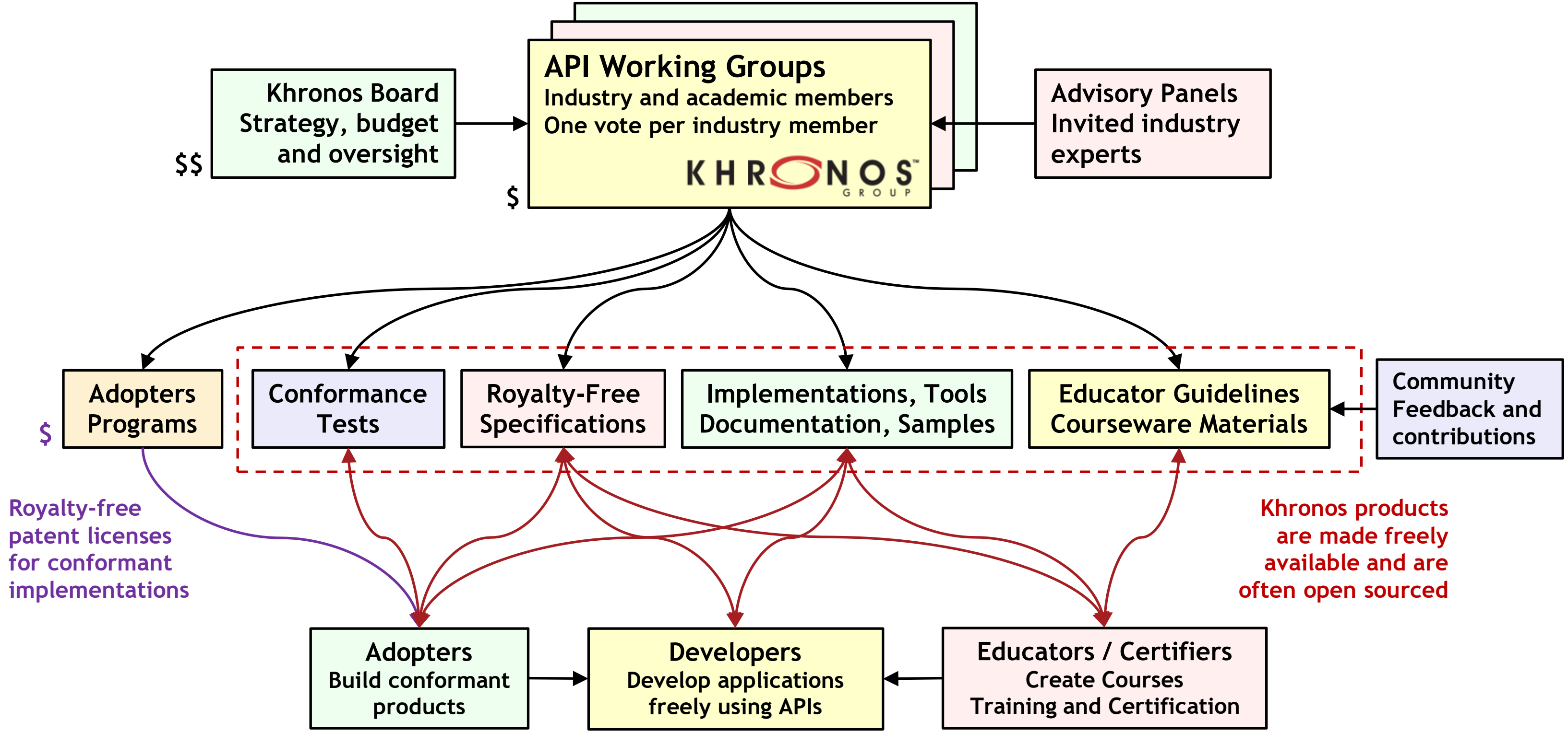
Often used for SOFTWARE libraries and languages to share effort AND gain consistency through a single implementation

Benefits of Open Interoperability Standards

- **Proven solutions - often available royalty free**
  - Leveraging significant industry effort and industry expertise
- **Benefits for hardware and software developers**
  - Cross-platform application portability and reusability
  - Industry-wide ecosystem of tools and libraries
- **Benefits for embedded markets**
  - Decoupled software and hardware for streamlined development, integration and safety certification
  - Cross-generation reusability and field upgradability

Why Open Standards?	**Expand Commercial Opportunity** Network effect of compatible products & services	**Reduce Costs** Share design effort and drive increased volume
	Avoid Market Friction Reduce fragmentation and confusion	**Speed Time to Market** Leverage proven functionality and testing
When?	**When Technologies are Proven** Avoid R&D by standards committee	**Consensus Need** Downsides of no available standard widely obvious
How?	**Multi-company Governance to Build Trust** Avoid single-company control or dependency	**Well-defined IP Rights Policy** Royalty-free standards drive wide adoption
	Innovation through Flexible Extensibility Extensions meet timely customer & market needs	**Innovation through Careful Abstraction** Freedom to innovate implementation details

Khronos Cooperative Framework



Khronos Active Standards

3D Graphics
Desktop, Mobile, Web

WebGL · ANARI · OpenGL ES · OpenGL · Vulkan

3D Asset
Authoring/Delivery

glTF · COLLADA · KTX

Portable XR
Augmented and Virtual Reality

OpenXR

Parallel Computation
Vision, Camera, Inferencing, Machine Learning

OpenCL · Kamaros · OpenVX · NNEF · SYCL · SPIR

Vulkan SC · OpenGL SC · SYCL SC — **Safety Critical APIs**

Khronos standards most relevant to compute, embedded, vision and safety critical markets

Khronos Compute Acceleration Standards

Higher-level Languages and APIs
Streamlined development and performance portability

SYCL SC / SYCL
Single source C++ programming with compute acceleration

NNEF
Neural Network Exchange Format Trained Networks

OpenVX
Graph-based vision and inferencing acceleration

PyTorch, gstreamer, OpenCV, FFmpeg
Third party vision, streaming and inferencing libraries

Lower-level Languages and APIs
Explicit hardware control

Applications, libraries, and higher-level languages and APIs may use lower-level Khronos standards to access hardware acceleration

Vulkan SC / Vulkan
GPU rendering + compute acceleration

GPU

Shaders ⇦ SPIR ⇨ Kernels
Intermediate Representation (IR) language compiler target supporting parallel execution and graphics

OpenCL
Heterogeneous compute acceleration

FPGA | CPU | GPU | DSP
Custom HW | AI/Tensor HW

Multiple programming abstractions to meet the needs of diverse software stack architectures

OpenCL - Low-level Parallel Programing

Programming and Runtime Framework for Application Acceleration

Offload compute-intensive kernels onto parallel heterogeneous processors

CPUs, GPUs, DSPs, FPGAs, Tensor Processors

OpenCL C or C++ kernel languages

Platform Layer API

Query, select and initialize compute devices

Runtime API

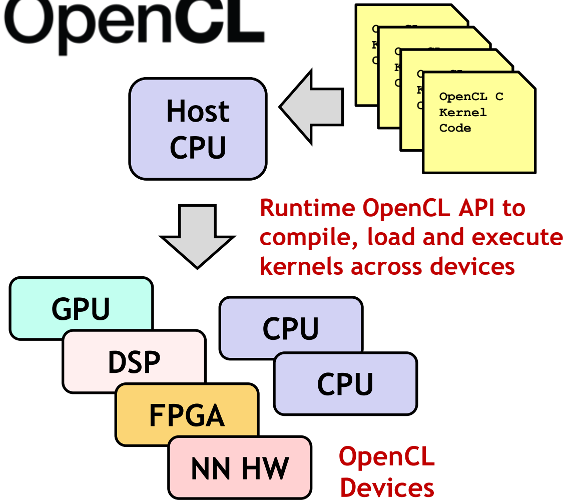
Build and execute kernels programs on multiple devices

Explicit Application Control

Which programs execute on what device

Where data is stored in memories in the system

When programs are run, and what operations are dependent on earlier operations



Complements GPU-only APIs

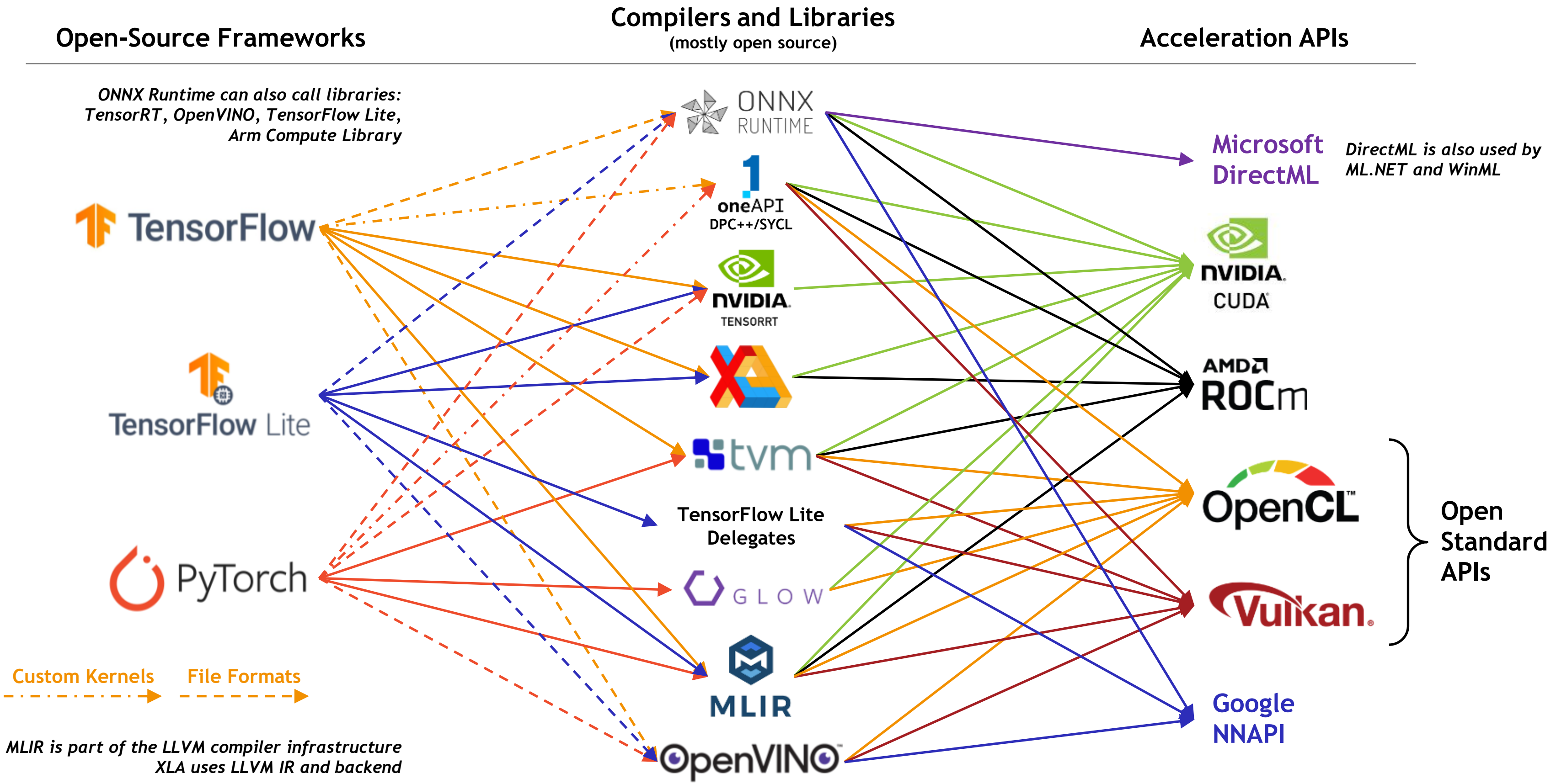
Simpler programming model

Relatively lightweight run-time

More language flexibility, e.g., pointers

Rigorously defined numeric precision

Machine Learning Acceleration APIs



OpenCL 3.0 Adoption and Roadmap

- **Regular (roughly) quarterly releases with new unified specification format**
  - 3.0.16 released in April 2024 with External Memory and Semaphores finalized
- **Considerable open-source activity**
  - Mesa Rusticl for Linux
  - clang/LLVM compilation front-ends
  - Layered implementations clspv and Ancle over Vulkan, OpenCLon12 over DX12
- **Emerging acceptance of OpenCL as compute layer over Vulkan**
  - Especially for Machine Learning
- **Active extension pipeline - driven by mobile, embedded and desktop markets**
  - Recordable Command Buffers, Unified Shared Memory, Cooperative Matrix and other ML primitives

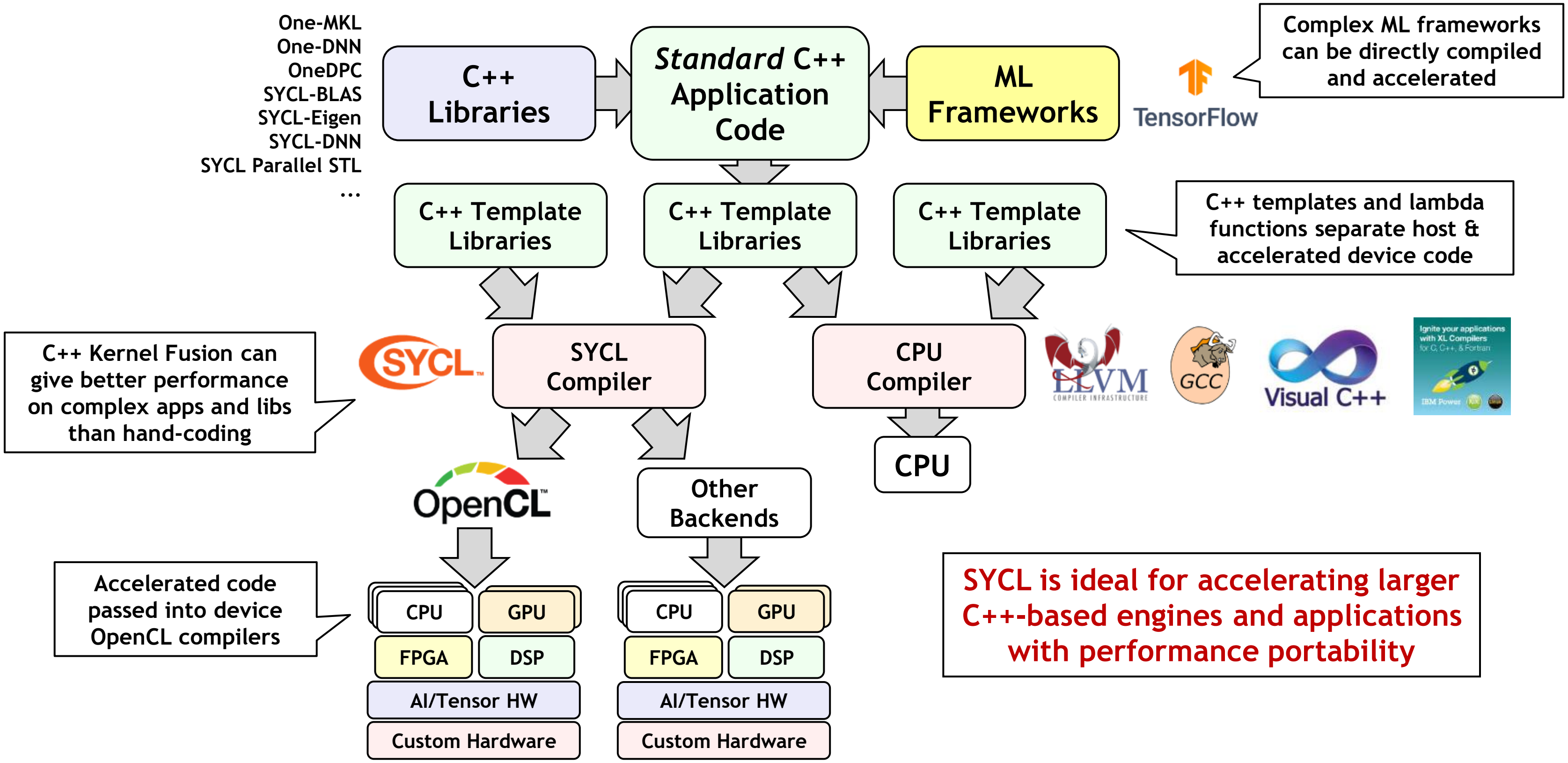
OpenCL 3.0 growing adoption

https://www.khronos.org/conformance/adopters/conformant-products/opencl

✓ Shipping OpenCL 3.0 Implementations

Adopters of previous OpenCL Versions

SYCL Single-Source C++ Parallel Programming



SYCL Timeline

SYCL enables Khronos to influence ISO C++ to (eventually) support heterogeneous compute

SYCL, OpenCL and SPIR-V, as open industry standards, enable flexible integration and deployment of multiple acceleration technologies

2011

- C++11
- OpenCL 1.2
  OpenCL C Kernel Language

2015

- C++14
- SYCL 1.2
  C++11 Single source programming
- OpenCL 2.1
  SPIR-V in Core

2017

- C++17
- SYCL 1.2.1
  C++11 Single source programming
- OpenCL 2.2

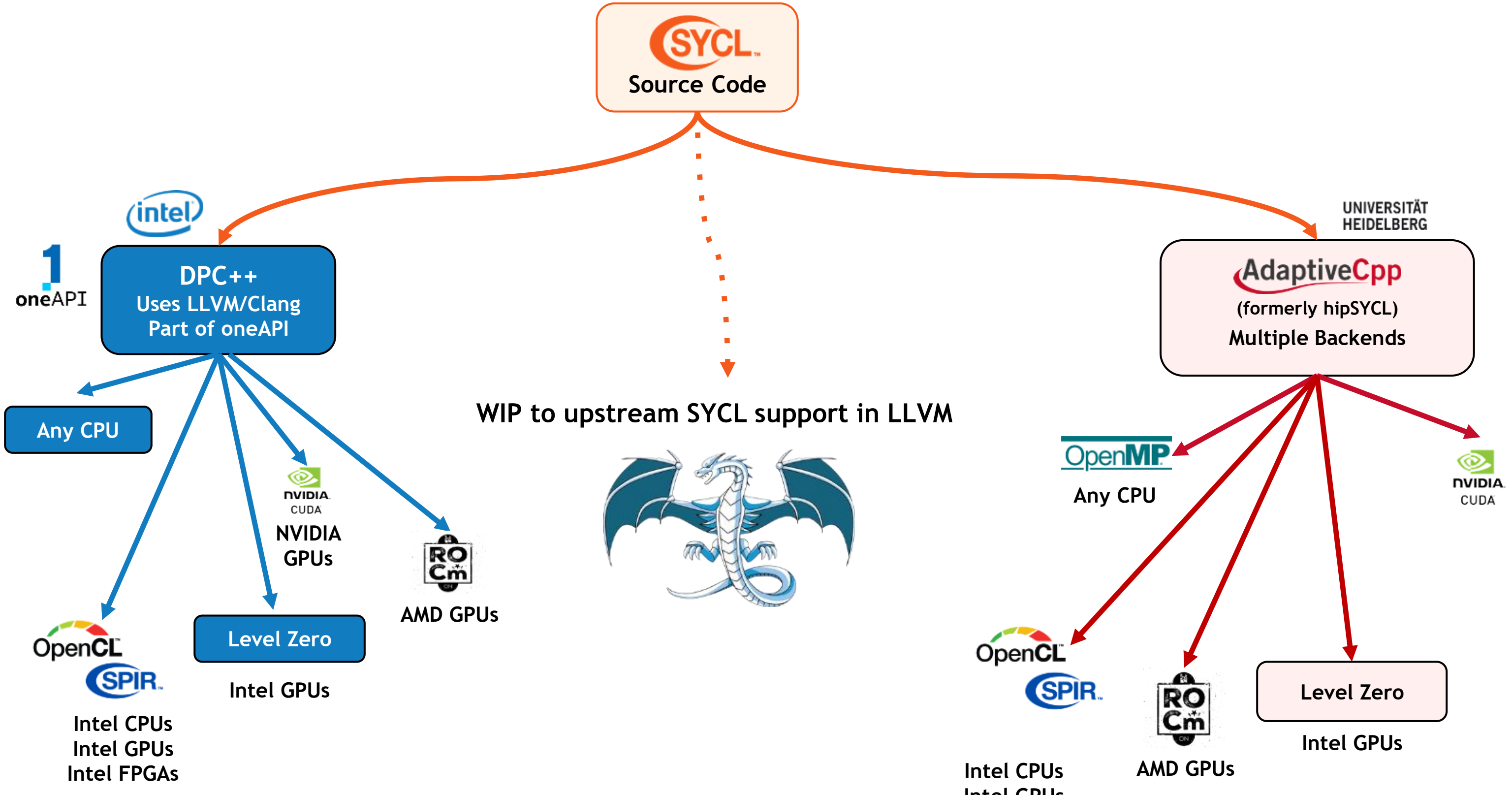
2020

- C++20
- SYCL 2020
  C++17 Single source programming
  Many HPC features
  Multiple backend options
- OpenCL 3.0

202X

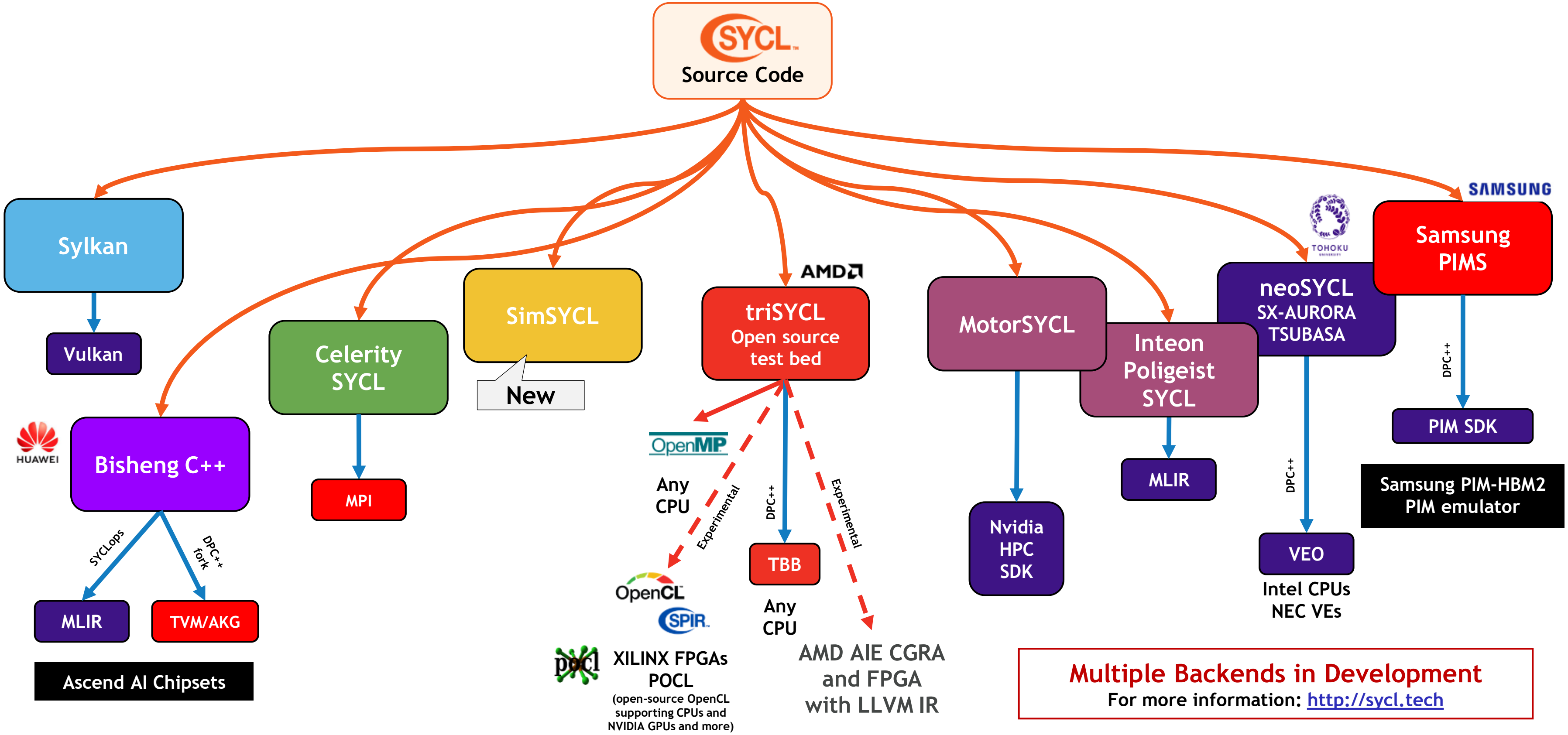
- C++23
- SYCL 202X
  C++XX Single source programming

SYCL Implementations in Development



Logos are the property of their respective owners

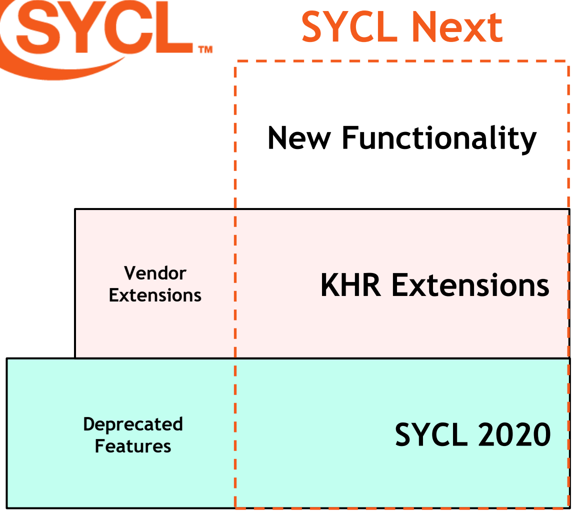
SYCL Experimental Development



Multiple Backends in Development
For more information: http://sycl.tech

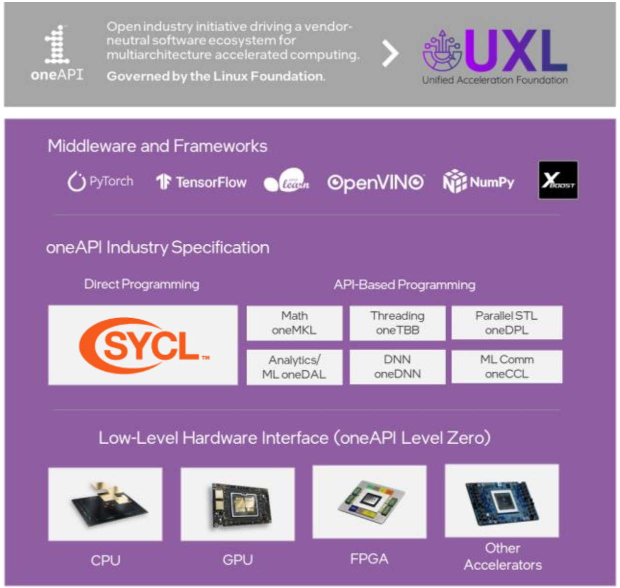
SYCL Next

- **Strategic path to incrementally release new features as KHR extensions**
  - Complete with tests and implementations
- **Key priorities are:**
  - Syntax improvements
  - Queue event performance
  - Task graphs
  - Compile-time properties
  - Hierarchical parallelism
- **Seeking feedback on priority features!**
  - https://community.khronos.org/c/sycl/
  - https://registry.khronos.org/SYCL/



Intel oneAPI DPC++ and UXL Foundation

- **Intel oneAPI DPC++ is conformant with SYCL 2020 Specification**
  - Unified Shared Memory, Parallel Reductions, Work Group Algorithms, Class Template Argument Deductions, Simplification of Accessors, Expanded Interoperability, and more
- **UXL Unified Acceleration Foundation**
  - Accelerated computing open ecosystem
  - Tools and Libraries
  - Compilers and development tools
  - APIs and specifications
- **Khronos and UXL have just announced a liaison**



OpenVX Cross-Vendor Vision and Inferencing

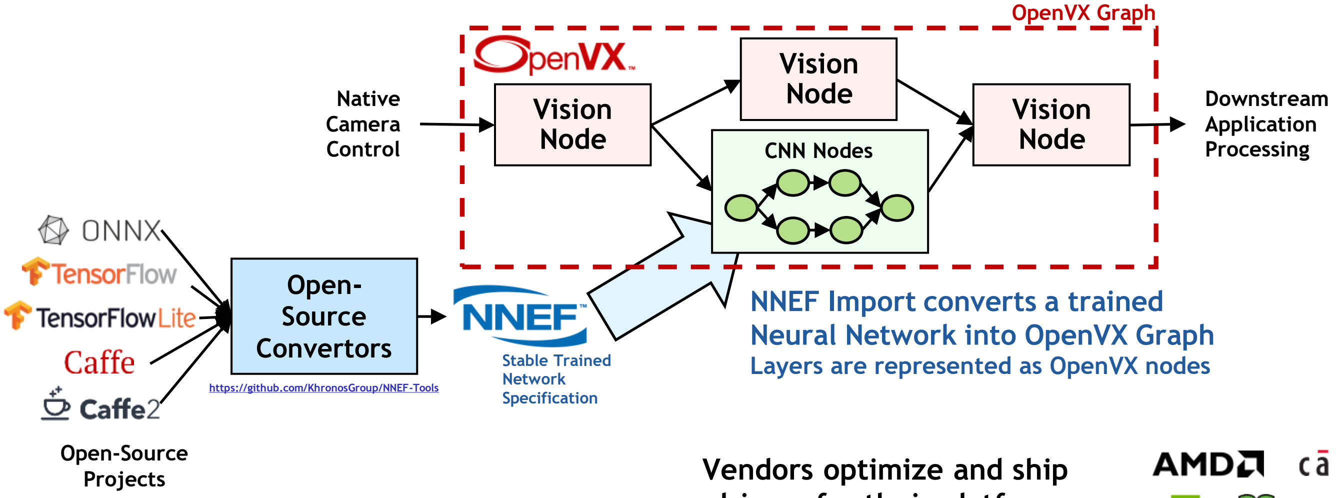
High-level graph-based abstraction for portable, efficient vision processing

Optimized OpenVX drivers created, optimized and shipped by processor vendors

Implementable on almost any hardware or processor with performance portability

Graph can contain vision processing and NN nodes for global optimization

Run-time graph execution need very little host CPU interaction



NNEF Import converts a trained Neural Network into OpenVX Graph

Layers are represented as OpenVX nodes

Vendors optimize and ship drivers for their platform

Full list of conformant OpenVX implementations here:
https://www.khronos.org/conformance/adopters/conformant-products/openvx

OpenVX Roadmap

OpenVX Node Functionality

- Base Features
  - Graph Framework + Data Objects
- Vision Nodes
- Enhanced Vision Nodes
- NN Nodes
- NNEF Import Nodes
- Extensions

OpenVX 1.X grew Node functionality over time

OpenVX 1.3.X Feature Sets to reduce fragmentation

- Vision Feature Set
- NN Feature Set
- NNEF Import Feature Set
- Enhanced Vision Feature Set
- Binary Image Support Feature set

OpenVX 1.3.X implementations include Base Features plus at least one of Vision/NN/NNEF feature sets

OpenVX 2.0 Core + Optional Extensions

Focus on the OpenVX Graph Framework
Flexible pipelined data flow through target hardware
Seamless custom kernel support
Extensions e.g., for vision/radar/lidar processing

OpenVX Core

- Base Features
  - Graph Framework + Data Objects
- Essential Extensions
- Custom/Target Nodes

Optional

- Vision Nodes
- Enhanced Vision Nodes
- NN Nodes
- NNEF Import Nodes
- Extensions

OpenVX 2.0 implementations need to include just OpenVX Core plus customer or selected Nodes

Growing Need for APIs for Functional Safety

Safety-critical APIs reduce system-level certification effort where functional safety is paramount

1) Streamlined to reduce documentation and testing surface area
2) Deterministic behavior to simplify system design and testing
3) Unambiguous and comprehensive fault handling

ISO 26262

DO-178C

1990s - Avionics

2010s - Automotive

2020s - Autonomous vehicles and devices, avionics, medical, industrial, and energy

Khronos has 20 years experience in adapting safety-critical API from proven mainstream APIs

OpenGL ES 2003 → OpenGL SC 2005

OpenVX 2017 Safety Critical Extension

Vulkan 2020 (V1.3) → Vulkan SC 2022

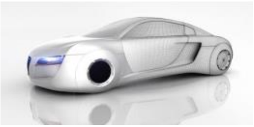
SYCL SC 2023 Working Group Formed

The Need for a Camera System API Standard

Increasing Sensor Diversity
Including camera arrays and depth sensors such as Lidar

Multiple Sensors Per System
Synchronization and coordination become essential

The cost and time to integrate and utilize sensors in embedded systems has become a major constraint on innovation and efficiency in the embedded vision market



Increasing Sensor Processing Demands
Including inferencing. Sensor outputs need to be flexibly and efficiently generated and streamed into acceleration processors

Proprietary APIs Hinder Innovation
Vendor-specific APIs to control cameras, sensors and close-to-sensor ISPs prevent access of full camera capabilities

Kamaros™

Embedded Camera System API - In Development

Open, cross-vendor, royalty-free open standard for camera, sensor and ISP control in embedded, mobile, industrial, XR, automotive, and scientific markets

Benefits

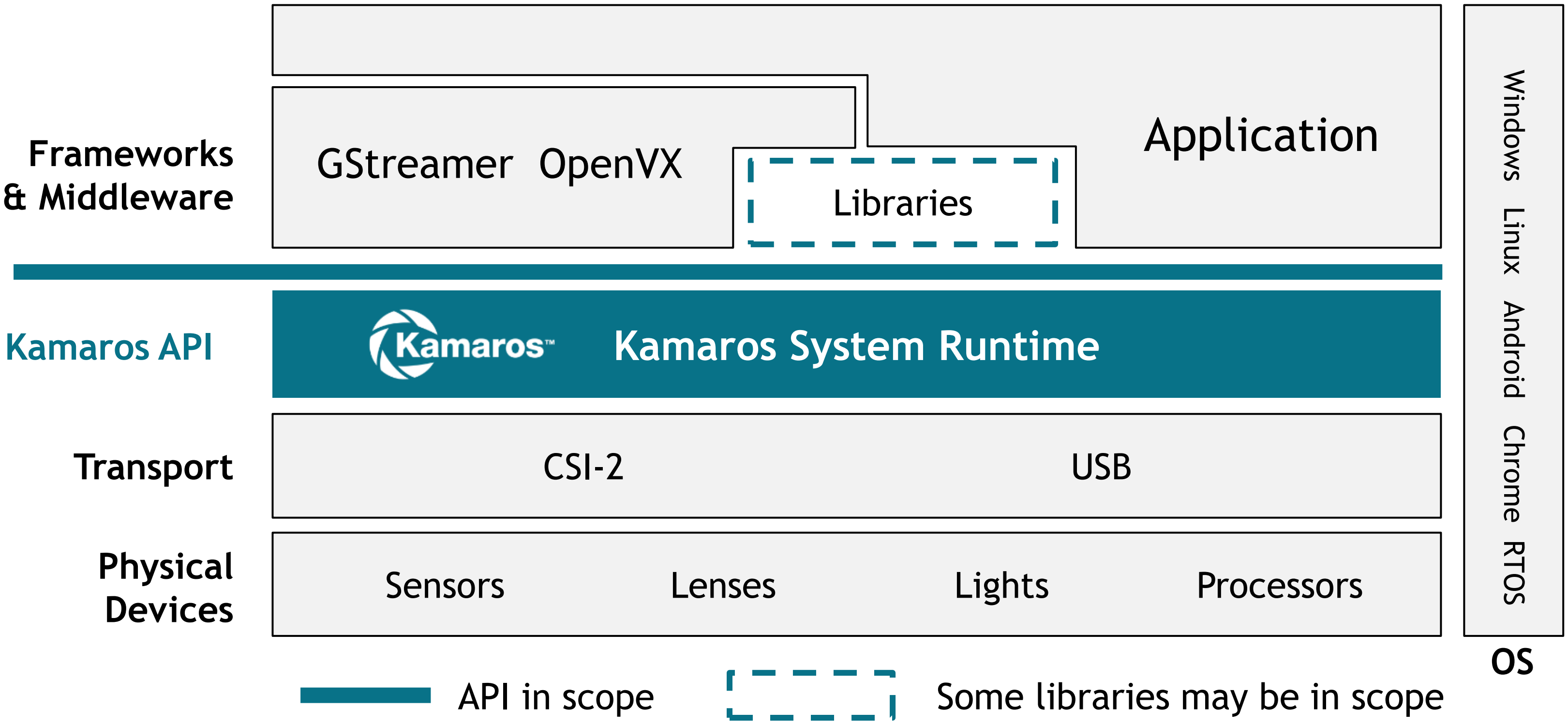
Portability of camera/sensor code for easier system integration of new sensors

Preservation of application code across multiple generations of cameras and sensors

Sophisticated control over sensor stream generation for effective downstream processing

An effective camera API abstraction will enable camera and sensor vendors to expose hardware capabilities without disclosing proprietary implementation details while gaining access to a larger ecosystem of libraries and applications

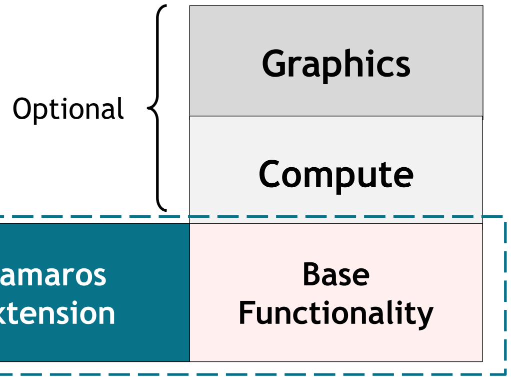
Typical Kamaros Software Stack



Names of transport layers, framework and operating systems are illustrative examples

Kamaros and Vulkan

- **Kamaros is re-using Vulkan design elements**
  - Saves time in redesigning recurring elements
  - Queues, buffers, synchronization etc.
- **Deploy as a Vulkan extension or standalone API**
  - Standalone API can be implemented without a GPU
- **Vulkan design is well-proven**
  - Low-level, explicit hardware access
  - Seamless interop with compute and graphics functionality
- **Leverages Vulkan Ecosystem**
  - SDK tooling including layers and loaders
  - Use relevant parts of Vulkan CTS
  - Developer familiarity

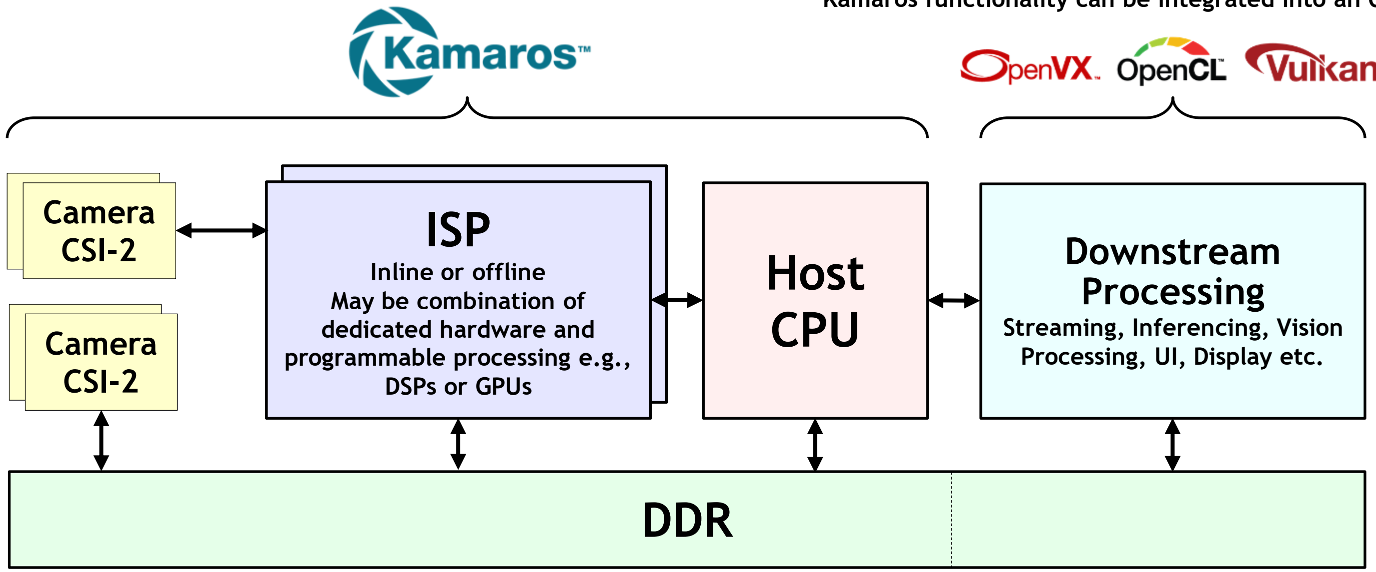


Typical Kamaros System Implementation

Can use any API - for example Khronos open standards

If Kamaros is implemented as Vulkan extension, can be single Vulkan runtime on ISP and Downstream Processing

Kamaros functionality can be integrated into an OpenVX Node



Kamaros Portable Application Structure

1. Discover Camera

Select Camera via Kamaros Loader
Data provided on physical or geographic location, sensor and lens etc.

Query Pipeline Templates for selected Camera
Select Pipeline that has Controls and capability limits that satisfy use case

2. Discover Pipeline

Request Pipeline Configuration
Controls set in priority order

Best Attempt Pipeline Configuration
Achievable Control Values, Defaults and Constraints

Kamaros™

3. Create Pipeline

Create Pipeline with initial Controls
Allocate any needed resources e.g., buffers

4. Capture Frames

Sets Controls per Frame
Synchronize for Frame availability

Process Returned Frame
Per frame metadata provides applied settings and errors

Stop Pipeline
Free resources

Application

Pipeline Templates enable flexible and scalable exposure of camera capabilities

Simpler Templates provide application portability

Customized Templates provide detailed hardware control and can even be vendor-supplied

KHRONOS® GROUP

Kamaros, Khronos and EMVA Cooperation

KHRONOS GROUP

Working Group
Design decisions and execution

Drafts →
← Feedback

Advisory Panel
Design input and feedback

Other Khronos Working Groups and Advisory Panels

EMVA members are invited to join Kamaros Advisory Panel at no charge

emva – european machine vision association

GenICam, layered libraries, and other EMVA machine vision initiatives may leverage Kamaros

Khronos / EMVA have a Liaison Agreement for ongoing coordination and joint membership privileges for designated liaisons

Get Involved!

Khronos is developing a growing family of open, royalty-free API standards relevant to embedded and safety-critical markets

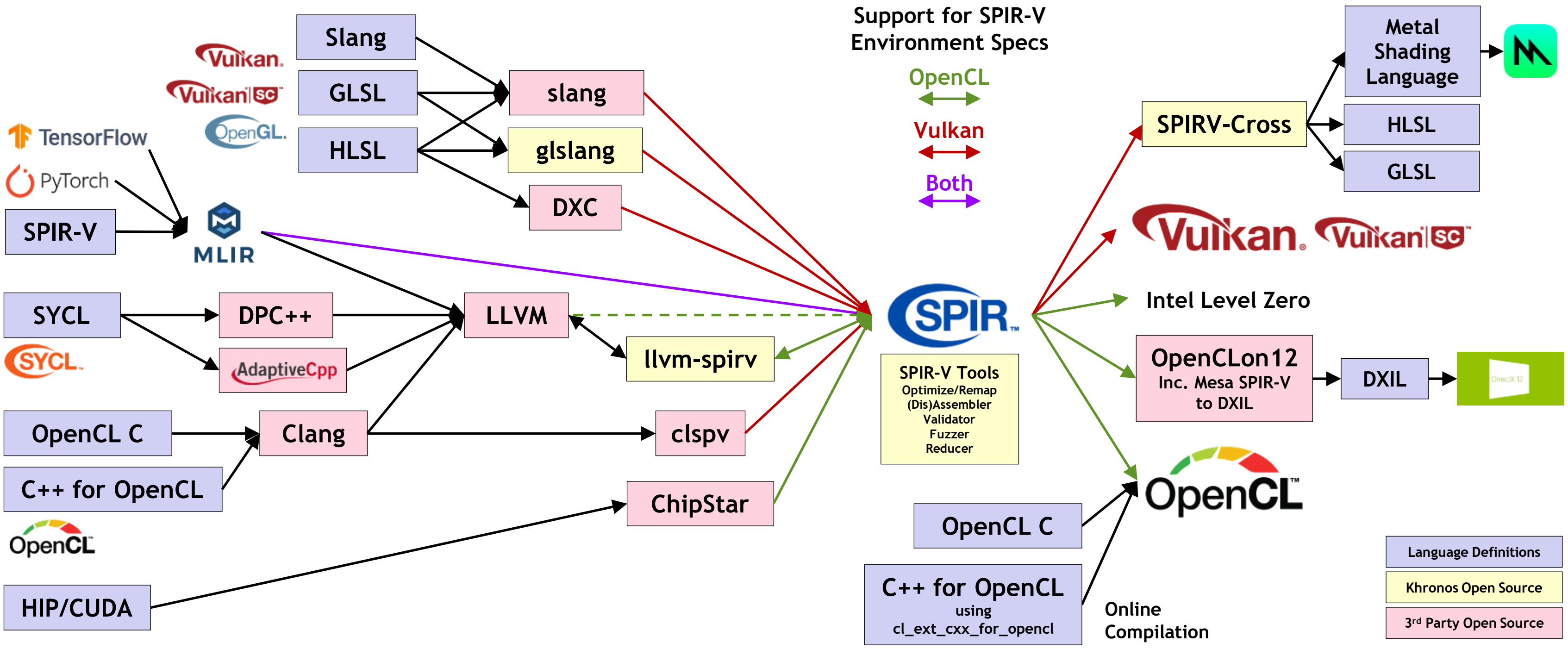
Any company is welcome to join Khronos to influence standards development
https://www.khronos.org/members/ or email memberservices@khronosgroup.org

More information on any Khronos API
https://www.khronos.org/

Khronos members can participate in the Kamaros Camera Working Group
EMVA Members can join the Kamaros Advisory panel
https://www.khronos.org/kamaros

Background and Archive

SPIR-V Ecosystem



MLIR is part of the LLVM compiler infrastructure

API Layering

Enabled by increasingly robust of open-source compiler ecosystem leveraging SPIR-V

Layers Over	Vulkan	OpenGL	OpenCL	OpenGL ES	DX12	DX9-11
Vulkan		Zink	clspv + clvk Ancle RustiCL/Zink	GLOVE Angle	vkd3d-Proton vkd3d	DXVK WineD3D
OpenGL	gfx-rs Ashes			Angle		WineD3D
DX12	Dozen gfx-rs	Microsoft 'GLOn12'	Microsoft 'CLOn12'			Microsoft D3D11On12
DX9-11	gfx-rs Ashes			Angle		
Metal	MoltenVK gfx-rs			MoltenGL Angle		

ROWS
Benefit Platforms by enabling content without additional kernel level drivers

COLUMNS
Benefit ISVs by providing application deployment flexibility and fighting fragmentation by making an API available across multiple platforms even if no native drivers available

Layered OpenCL Implementations

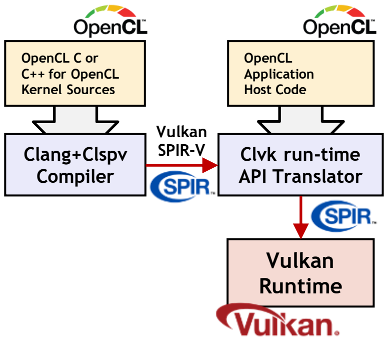
clspv + clvk

OpenCL over Vulkan
Google

clspv open-source OpenCL kernel to Vulkan SPIR-V compiler - tracks top-of-tree LLVM and Clang - not a fork

clvk - prototype open-source OpenCL to Vulkan run-time API translator

Used by shipping apps and engines on Android e.g., Adobe Premiere Rush video editor - 200K lines of OpenCL C kernel code



clspv + Ancle

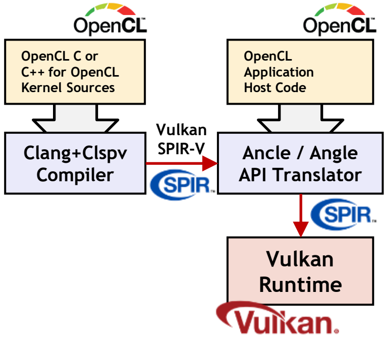
OpenCL over Vulkan
Samsung

Integrates clspv and OpenCL runtime into Angle code base

Samsung Motivation
"OpenCL is widely used and deployed and is making a comeback thanks to ML"

"OpenCL is a favored high-level (front-end) compute language! Easier to write than Vulkan"

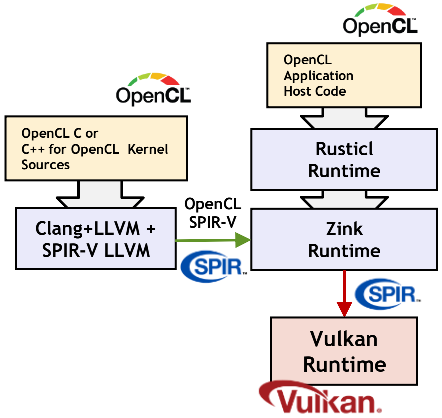
Ancle makes OpenCL a first-class citizen in Android by relying on Vulkan as its Native Driver"



Rusticl over Zink

OpenCL over Vulkan
Mesa

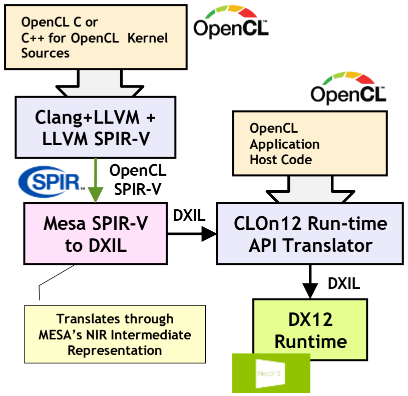
The Zink Gallium driver emits Vulkan API calls and now supports OpenCL Kernels



OpenCLOn12

OpenCL over DX12
Microsoft

GPU-accelerated OpenCL on any DX12 PC and Cloud instance (x86 or Arm)



Apps, Libraries and Engines using OpenCL

**Pervasive, cross-vendor, open standard for
low-level heterogeneous parallel programming**

https://en.wikipedia.org/wiki/List_of_OpenCL_applications

Desktop Creative Apps

Adobe, otoy, acdsee, DASSAULT SYSTEMES, Modo, SONY, SideFX, darktable, AUTODESK, CHAOSGROUP, Vegas Pro, Capture One, SILHOUETTE, CyberLink, ptc, GIMP, ArcSoft, RADEON PRORENDER, Blackmagicdesign, REALFLOW, LuxCoreRender OPEN SOURCE PHYSICALLY BASED RENDERER

Parallel Languages

SYCL, OpenACC DIRECTIVES FOR ACCELERATORS, aparapi, PyOpenCL

Machine Learning Libraries and Frameworks

Intel OpenVINO, Intel clDNN, Xiaomi MACE Mobile AI Compute Engine, SYCL-DNN, Caffe, NNAPI, Acuity ML/AI Software Foundation, VeriSilicon, Qualcomm Neural Processing SDK for AI, Synopsis MetaWare EV, TI DL Library (TIDL), Arm Compute Library, 飞桨 PaddlePaddle, MNN Mobile Neural Network

Molecular Modelling Libraries

siremol.org, FOLDING @HOME, ForceBalance, OpenMM, CHARMM Chemistry at HARvard Macromolecular Mechanics, GROMACS FAST. FLEXIBLE. FREE.

Machine Learning Compilers

MLIR, GLOW, tvm, plaidML

Vision, Imaging and Video Libraries

FAST, OpenVX, Halide, VisionCpp, Metashape, OpenCV, FFmpeg

Math and Physics Libraries

BULLET PHYSICS LIBRARY, GNU Octave, Wolfram Mathematica, ArrayFire C, C++, Fortran, Matlab

Linear Algebra Libraries

SYCL-BLAS, ViennaCL, CLBlast

Executing OpenCL Programs

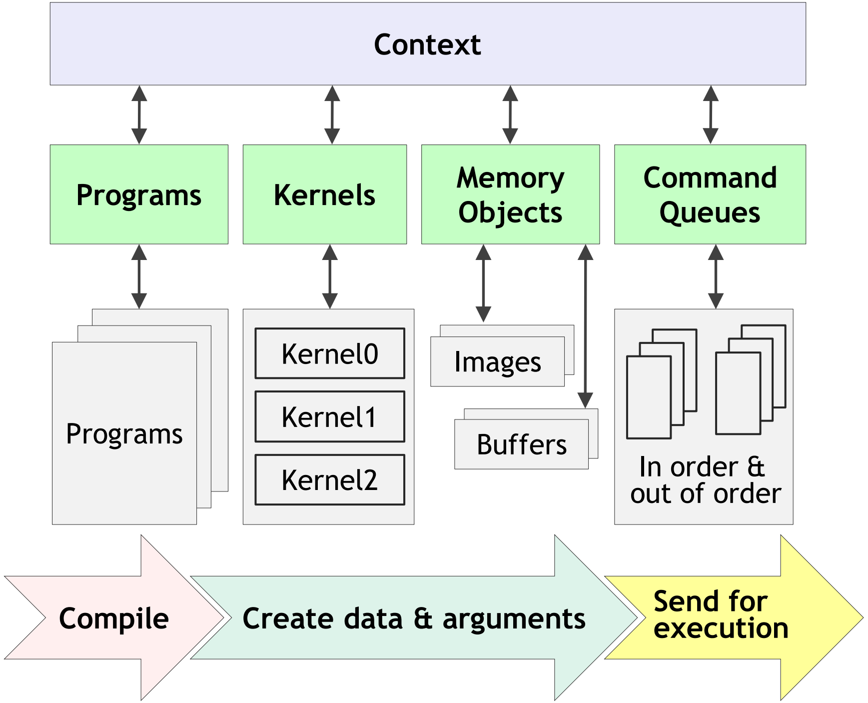
A ***kernel*** program is the basic unit of executable code (similar to a C function)

An OpenCL ***program*** is a collection of kernels and functions

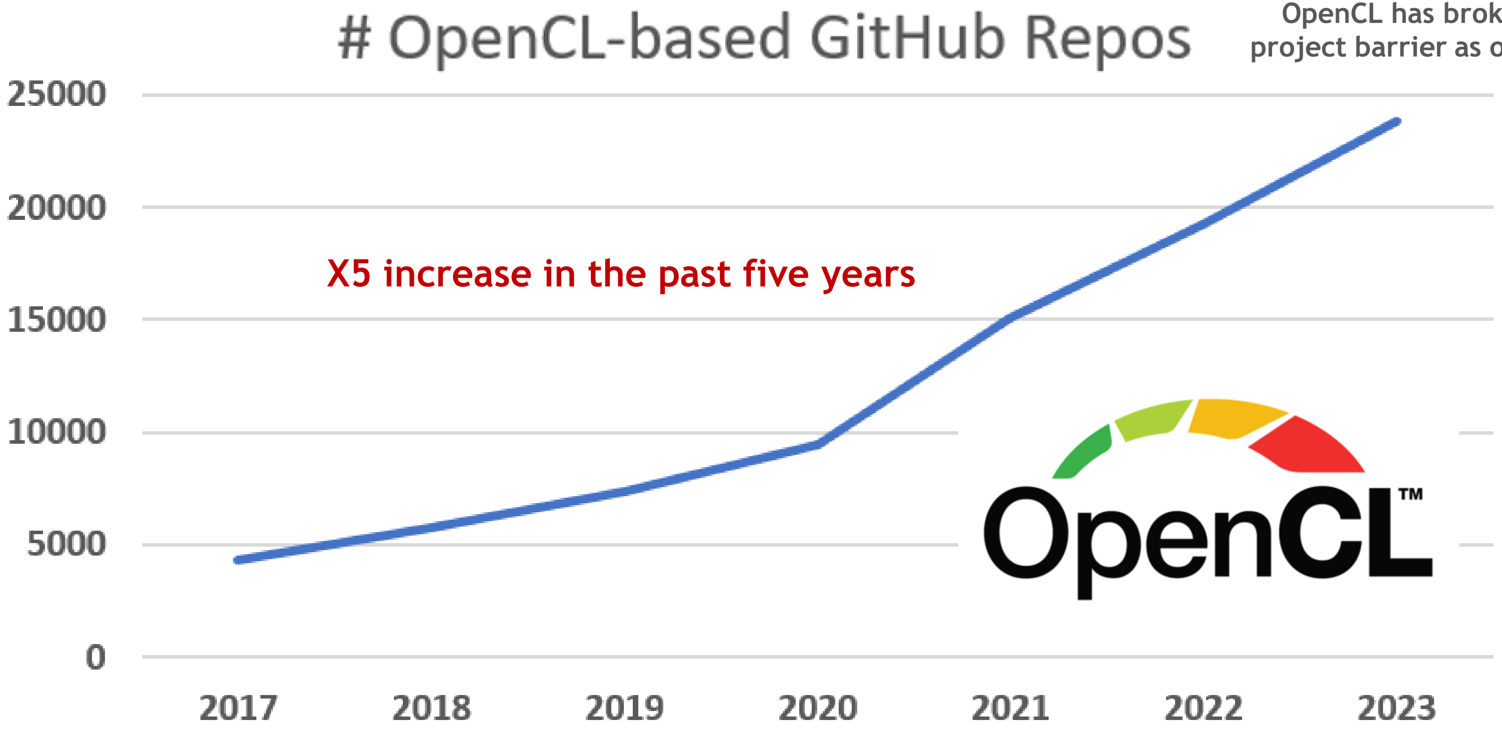
An OpenCL ***command queue*** is used by the host application to send kernels and data transfer functions to a device for execution.

By *enqueueing* commands into a command queue, kernels and data transfer functions may execute asynchronously and in parallel with application host code

As an open standard, OpenCL is a well proven design, available from many silicon vendors with an extensive ecosystem of available tools, compilers, libraries and educational materials



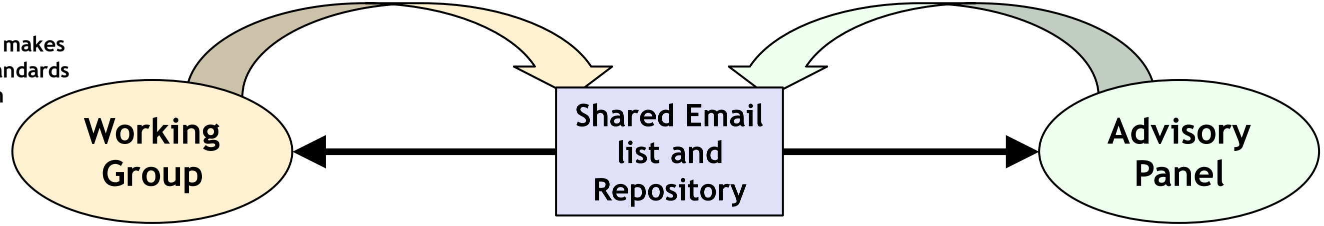
OpenCL Open-Source Project Momentum



OpenCL has broken the 25K project barrier as of March 2024

OpenCL Advisory Panel

Working Group makes decisions on standards evolution



Hosted by Khronos.
Under Khronos NDA

Khronos Members

Any company can join.
Membership Fee.
Covered by NDA *and* IP Framework

Working groups can share draft specifications and accept detailed design contributions as Panel Members are covered by IP Framework

Panel Members

Invited industry experts.
$0 Cost.
Covered by NDA *and* IP Framework

Please reach out to opencl-chair@lists.khronos.org if you wish to apply

Developers - Please Give Us Feedback!

- **Give us your feedback on the OpenCL spec GitHub**
  - What could be added to the OpenCL ecosystem to make you more productive?
  - What API and Language features do you most need?
  - https://github.com/KhronosGroup/OpenCL-Docs
- **Please download and run the GPUinfo OpenCL Hardware Capability Viewer**
  - https://opencl.gpuinfo.org/download.php
- **Consider applying to join the OpenCL Advisory Panel!**
  - Email opencl-chair@lists.khronos.org

OpenCL Resources

- OpenCL Home Page
  - https://www.khronos.org/opencl/
- OpenCL Registry for OpenCL core and extension specifications
  - https://www.khronos.org/registry/OpenCL/
- C++ for OpenCL Documentation
  - https://github.com/KhronosGroup/Khronosdotorg/blob/master/api/opencl/assets/CXX_for_OpenCL.pdf
- OpenCL SDK
  - https://github.com/KhronosGroup/OpenCL-SDK
- OpenCL Guide
  - https://github.com/KhronosGroup/OpenCL-Guide
- OpenCL Specification Source
  - https://github.com/KhronosGroup/OpenCL-Docs
- OpenCL Conformant Products
  - https://www.khronos.org/conformance/adopters/conformant-products/opencl
- GPUinfo.org Hardware Database
  - https://www.gpuinfo.org/
- Layered OpenCL implementations - clspv/clvk and OpenCLon12
  - https://github.com/google/clspv
  - https://github.com/kpet/clvk
  - https://github.com/microsoft/OpenCLOn12

SYCL Developer Resources

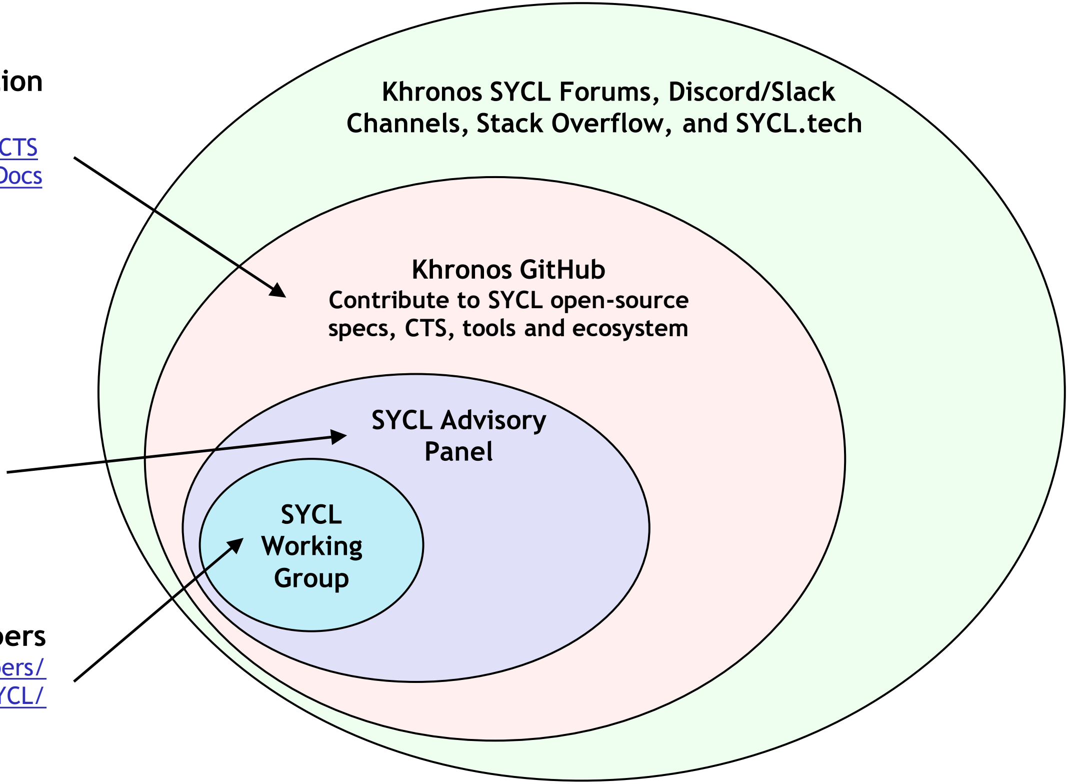
- **I need to learn SYCL**
  - The book
  - Attend a tutorial
  - SYCL Academy: https://github.com/codeplaysoftware/syclacademy
- **I know SYCL, and need more information about an API**
  - SYCL Reference https://www.khronos.org/sycl/reference
- **I need to know the ins-and-outs of an API**
  - SYCL Spec (it's quite readable!) https://registry.khronos.org/SYCL/
- **I still need help!**
  - Forums:
    - https://community.khronos.org/c/sycl/
    - https://stackoverflow.com/*questions*/*tagged*/sycl
  - SYCL.tech: https://sycl.tech/
  - Khronos Discord: https://www.khr.io/khrdiscord
  - Ask your implementor

Get Involved!

Public contributions to Specification and Conformance Tests
https://github.com/KhronosGroup/SYCL-CTS
https://github.com/KhronosGroup/SYCL-Docs

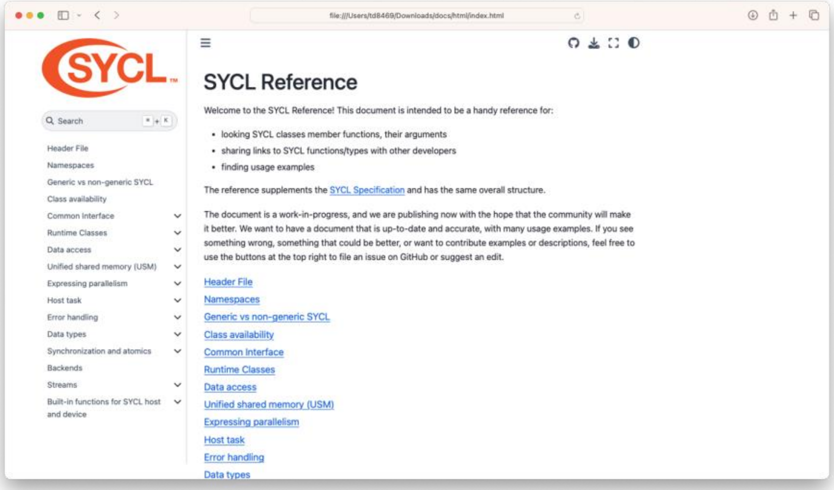
Join as an Invited Expert (no cost, sign Khronos NDA)
https://www.khronos.org/advisors/

Join as a Khronos members
https://www.khronos.org/members/
https://www.khronos.org/registry/SYCL/



SYCL Reference

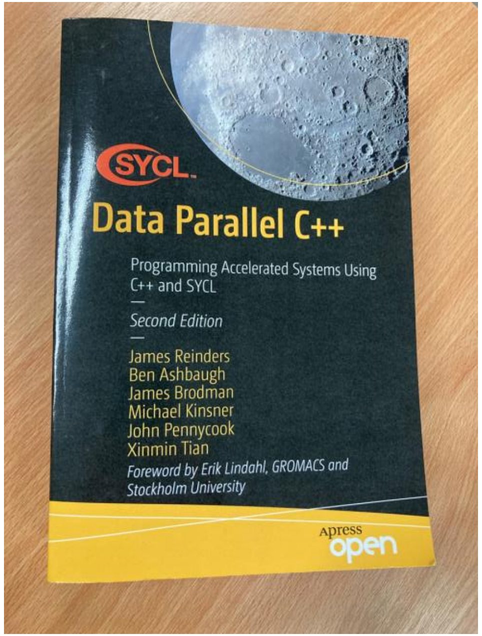
- New resource to support SYCL developers
- Inspired by cppreference.com
- Short descriptions of SYCL 2020 API
- Specification remains the canonical document
- https://www.khronos.org/sycl/reference



The SYCL Book (second edition)

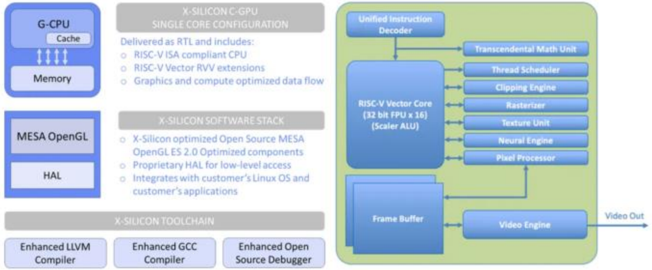
- **New edition up to date with SYCL 2020**
  - Published Oct 4th, 2023
- **Source code repository of examples**
- **Freely open online, or available in paperback**

  https://link.springer.com/book/10.1007/978-1-4842-9691-2



Recent AI/RISC-V Use of Khronos Standards

- **SiM.ai Chip Startup Raises $70 Million to Quicken AI on Cars and Robots**
  - https://www.msn.com/en-us/money/other/chip-startup-raises-70-million-to-quicken-ai-on-cars-and-robots/ar-BB1l48bI
  - SiMa.ai is one of a growing number of startups trying to perfect hardware for a future where AI is mainstream. The startup has enlisted more than 50 customers for its first chip, which mainly targeted computer vision, and is now working on a second generation. The new chip is scheduled for release in the first quarter of next year. SiMa.ai's products support various types of open standards including Linux and OpenCL
- **Axelera Uses oneAPI Construction Kit to Rapidly Enable Open Standards Programming for the Metis AIPU**
  - https://www.edge-ai-vision.com/2024/04/axelera-uses-oneapi-construction-kit-to-rapidly-enable-open-standards-programming-for-the-metis-aipu/
  - At Axelera, we therefore believe that the answer to the question of how to best bushwhack through the accelerator jungle is to embrace open standards, such as OpenCL and SYCL. OpenCL and SYCL are open standards defined by the Khronos Group. They define an application programming interface (API) for interacting with all kinds of devices as well as programming languages for implementing compute kernels to run on these devices.
- **New RISC-V microprocessor can run CPU, GPU, and NPU workloads simultaneously leveraging Khronos OpenGL**
  - https://www.tomshardware.com/pc-components/cpus/former-silicon-valley-vets-create-risc-v-microprocessor-that-can-run-cpu-gpu-and-npu-workloads-simultaneously

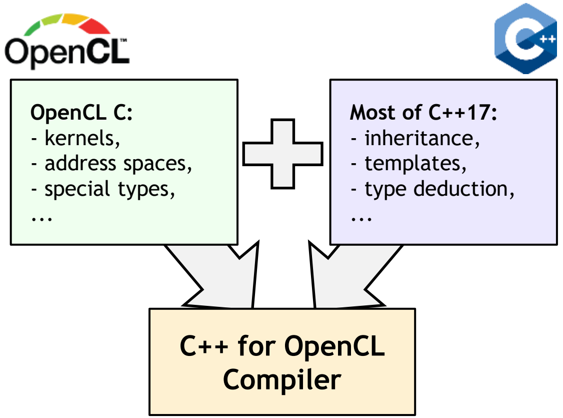


C++ for OpenCL

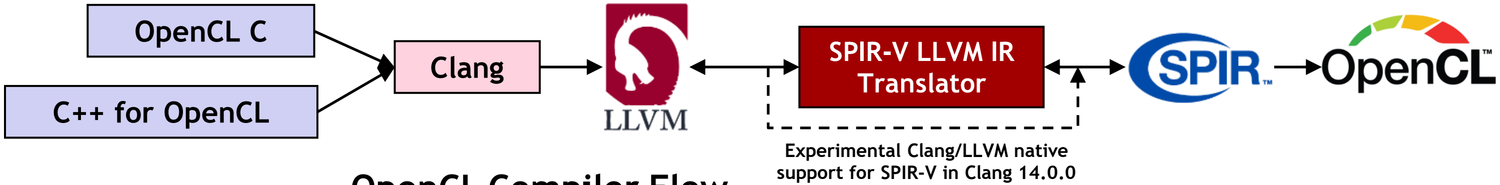
Open-Source Compiler Front-end
Replaces the OpenCL C++ kernel language spec
Official release published in OpenCL-Docs repo

Enables full OpenCL C and most C++17 capabilities
OpenCL C code is valid and fully compatible
Enables gradual transition to C++ for existing apps

Supported in Clang since release 9.0
Generates SPIR-V 1.0 plus SPIR-V 1.2 where necessary
Online compilation via cl_ext_cxx_for_opencl extension



Check it out in
Compiler Explorer



OpenCL Compiler Flow

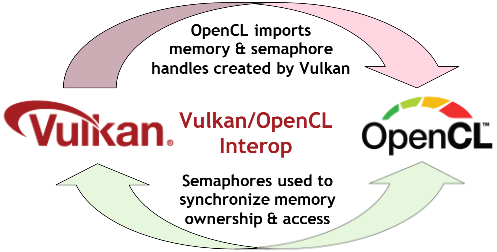
OpenCL Specification Releases and Roadmap

OpenCL 3.0.16 shipped on April 4th, 2024

Continues the regular release cadence for new functionality and bug fixes

External memory objects and semaphores for external sharing and Interop finalized

Kernel Clock extension provisional release



OpenCL Extension Pipeline

Provisional, EXT and Vendor extensions – candidates for final ratification

We are listening to your input!

- Support C++ for OpenCL (EXT)
- Command Buffer Record/Replay (provisional)
- Unified Shared Memory
- Floating Point Atomics
- Required Subgroup Size
- Generalized Image from buffer
- Image Tiling Controls
- YUV Multi-planar Images
- Cross-workgroup Barriers
- Cooperative Matrices
- Timeline Semaphores
- 32 and 64-length vectors
- Indirect Dispatch
- ML Operations

KHRONOS GROUP