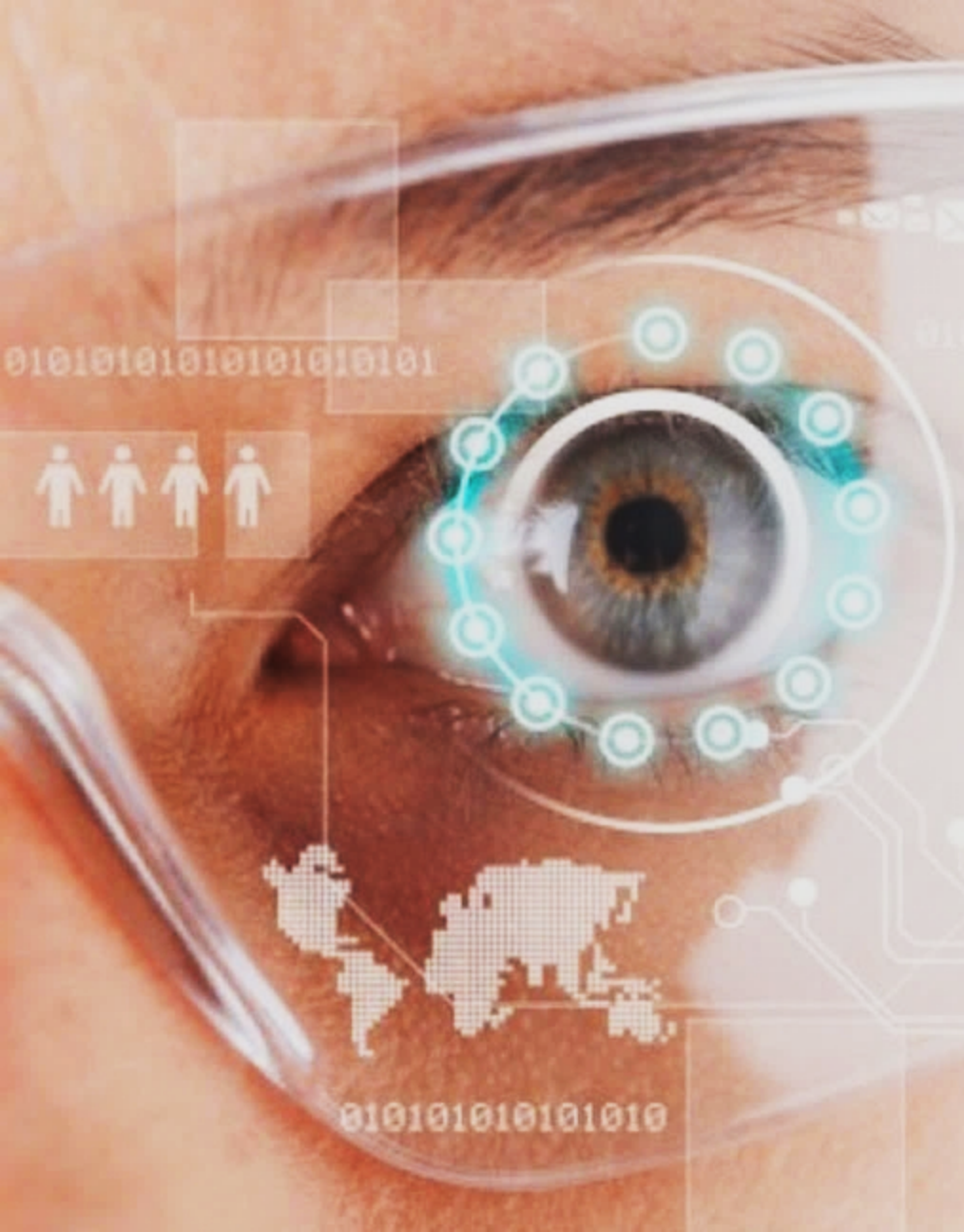


KHRONOS GROUP

# APIs for Accelerating Embedded Vision and Inferencing

## Industry Overview of Options and Trade-offs

Neil Trevett
Khronos President
NVIDIA VP Developer Ecosystems
ntrevett@nvidia.com | @neilt3d

embeddedworld2020
Exhibition&Conference
... it´s a smarter world

KHRONOS GROUP

# Connecting Software to Silicon


Khronos is 20 years old this year!

>150 Members ~ 40% US, 30% Europe, 30% Asia

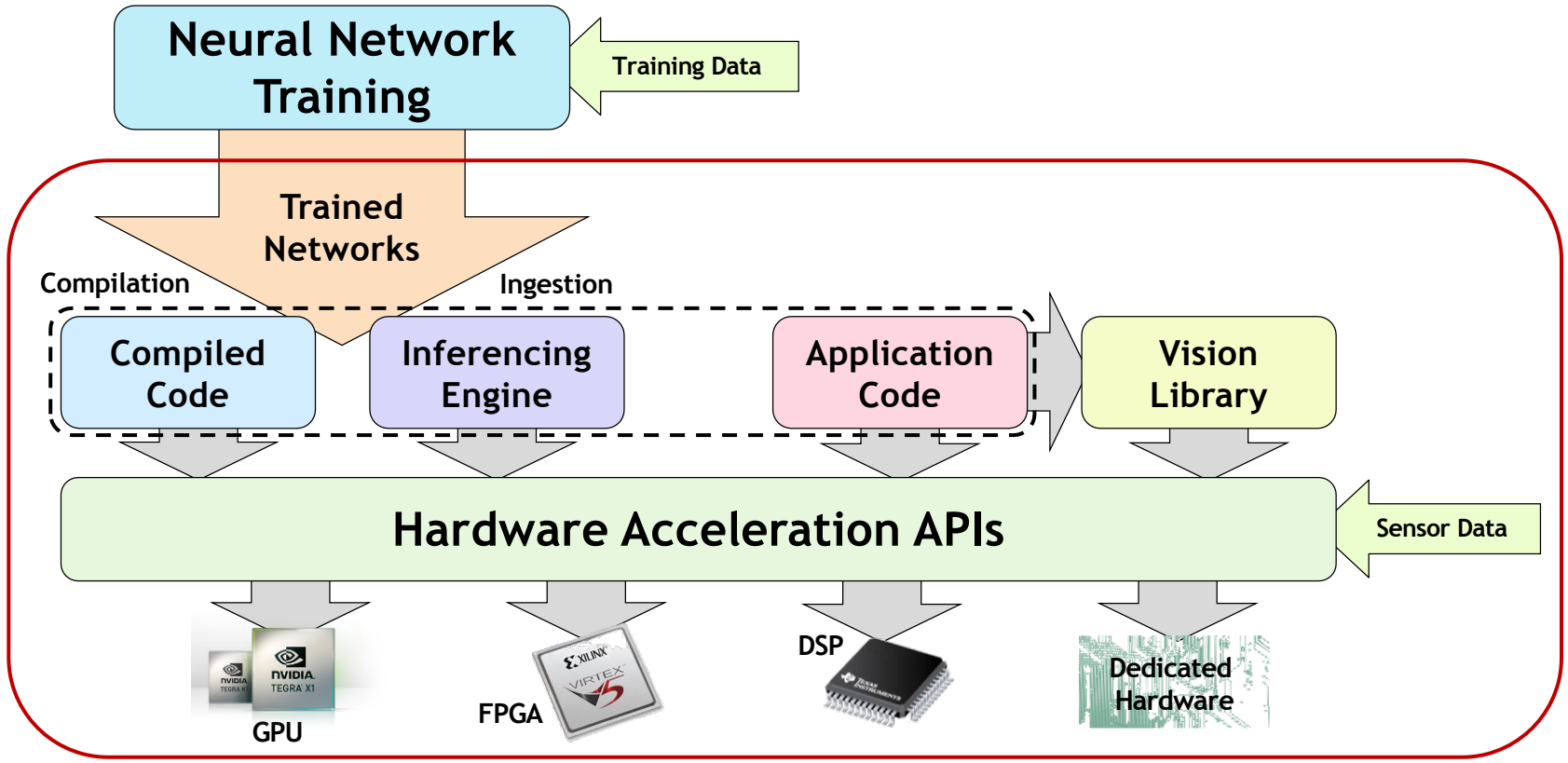
**Open, non-profit, member-driven industry consortium creating advanced, royalty-free interoperability standards for 3D graphics, augmented and virtual reality, parallel programming, vision acceleration and machine learning**

# Embedded Vision and Inferencing Acceleration

Networks trained on high-end desktop and cloud systems

Apps link to compiled code or inferencing library

Diverse Embedded Hardware (GPUs, DSPs, FPGAs)



# Khronos Active Initiatives

## 3D Graphics
Desktop, Mobile, Web
Embedded and Safety Critical

Vulkan
OpenGL ES
OpenGL
WebGL
EGL
OpenGL SC
Vulkan SC

## 3D Assets
Authoring
and Delivery

glTF
COLLADA
3D Commerce

## Portable XR
Augmented and
Virtual Reality

OpenXR

## Parallel Computation
Vision, Inferencing,
Machine Learning

SPIR
SYCL
OpenCL
NNEF
OpenVX

KHRONOS SAFETY CRITICAL ADVISORY FORUM | SC

**Guidelines for creating APIs to streamline system safety certification**

## Exploratory Groups

Heterogeneous Communications between offload compute devices

Making High-Level Languages more effective at acceleration offload

Rendering for scientific visualization and data analytics

KHRONOS GROUP

# Khronos Compute Acceleration Standards

**Higher-level APIs**
Streamlined development and performance portability

SYCL™
Single source C++ programming with compute acceleration

OpenVX™
Graph-based vision and inferencing acceleration

Import

NNEF™
Trained Neural Network exchange format

**Lower-level APIs**
Direct Hardware Control

Vulkan®
GPU rendering + compute acceleration

SPIR™
Intermediate Representation (IR) supporting parallel execution and graphics

OpenCL
Heterogeneous compute acceleration

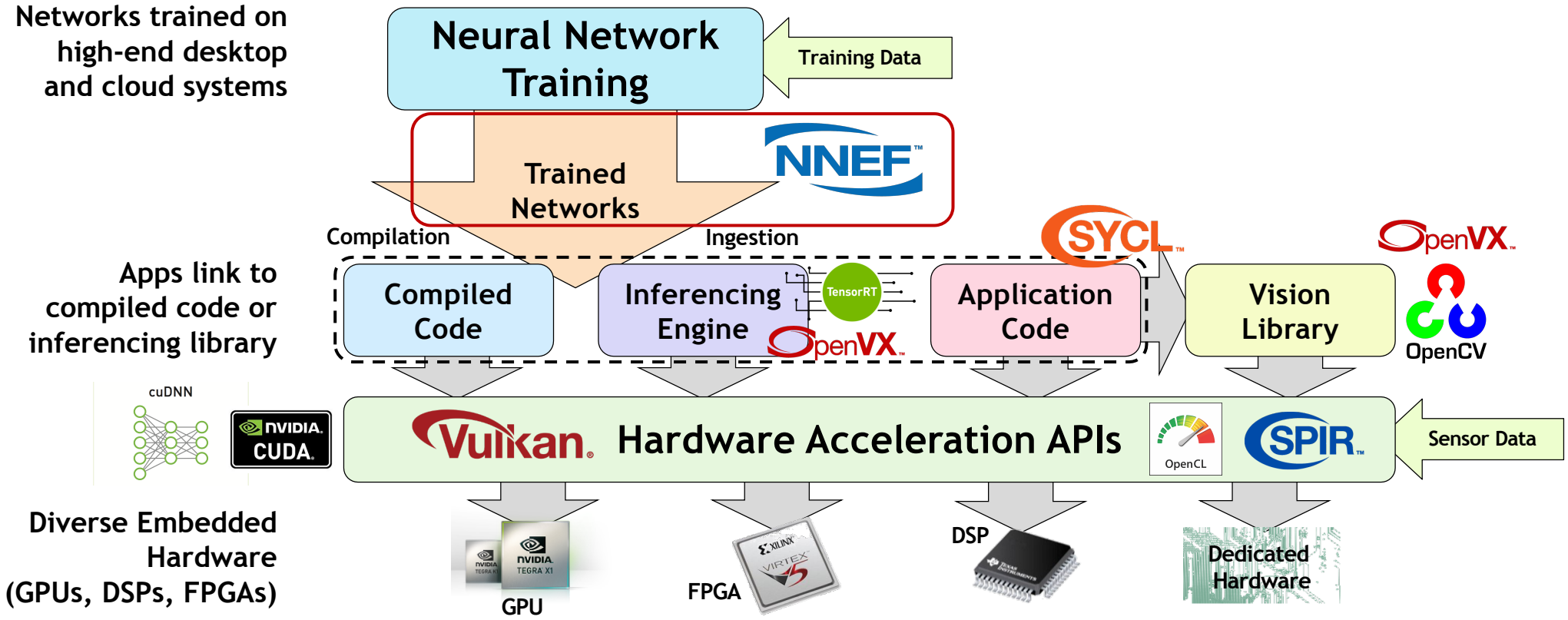
GPU

CPU
GPU
FPGA
DSP
AI/Tensor HW
Custom Hardware

Increasing industry interest in parallel compute acceleration to combat the 'End of Moore's Law'

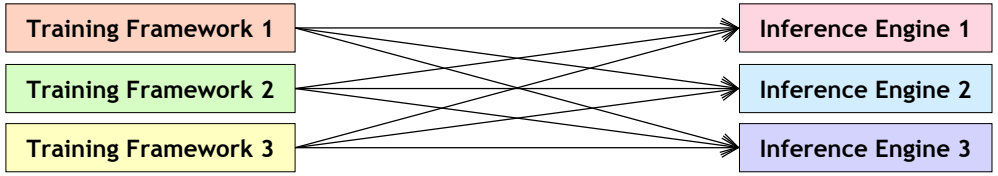
KHRONOS GROUP

# Embedded Vision and Inferencing Acceleration



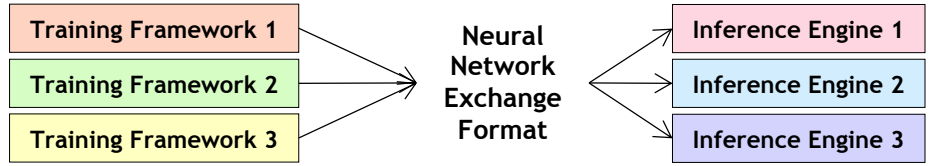
# Neural Network Exchange Formats

**Before - Training and Inferencing Fragmentation**




Every Inferencing Engine needs a custom importer from every Framework

**After - NN Training and Inferencing Interoperability**




Common Optimization and processing tools

## Two Neural Network Exchange Format Initiatives

| NNEF | ONNX |
|---|---|
| Embedded Inferencing Import | Training Interchange |
| Defined Specification | Open Source Project |
| Multi-company Governance at Khronos | Initiated by Facebook & Microsoft |
| Stability for hardware deployment | Software stack flexibility |

**ONNX and NNEF are Complementary**

**ONNX moves quickly to track authoring framework updates**
**NNEF provides a stable bridge from training into edge inferencing engines**

# NNEF and ONNX Industry Support

## NNEF V1.0 released in August 2018

After positive industry feedback on Provisional Specification.
Maintenance update issued in September 2019
Extensions to V1.0 released for expanded functionality

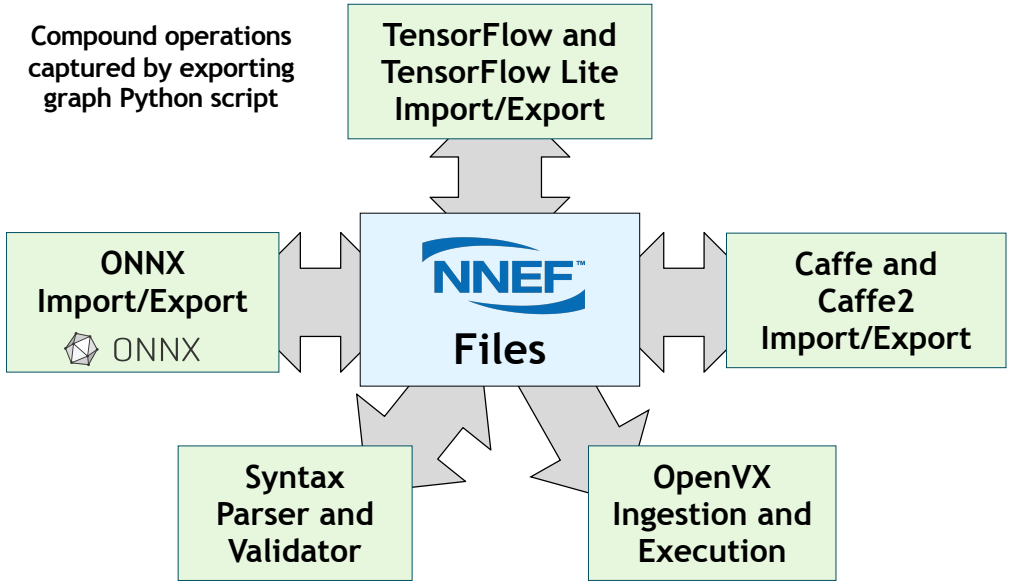

NNEF Working Group Participants

## ONNX 1.6 Released in September 2019

Introduced support for Quantization
ONNX Runtime being integrated with GPU inferencing engines such as NVIDIA TensorRT


ONNX Supporters

# NNEF Tools Ecosystem

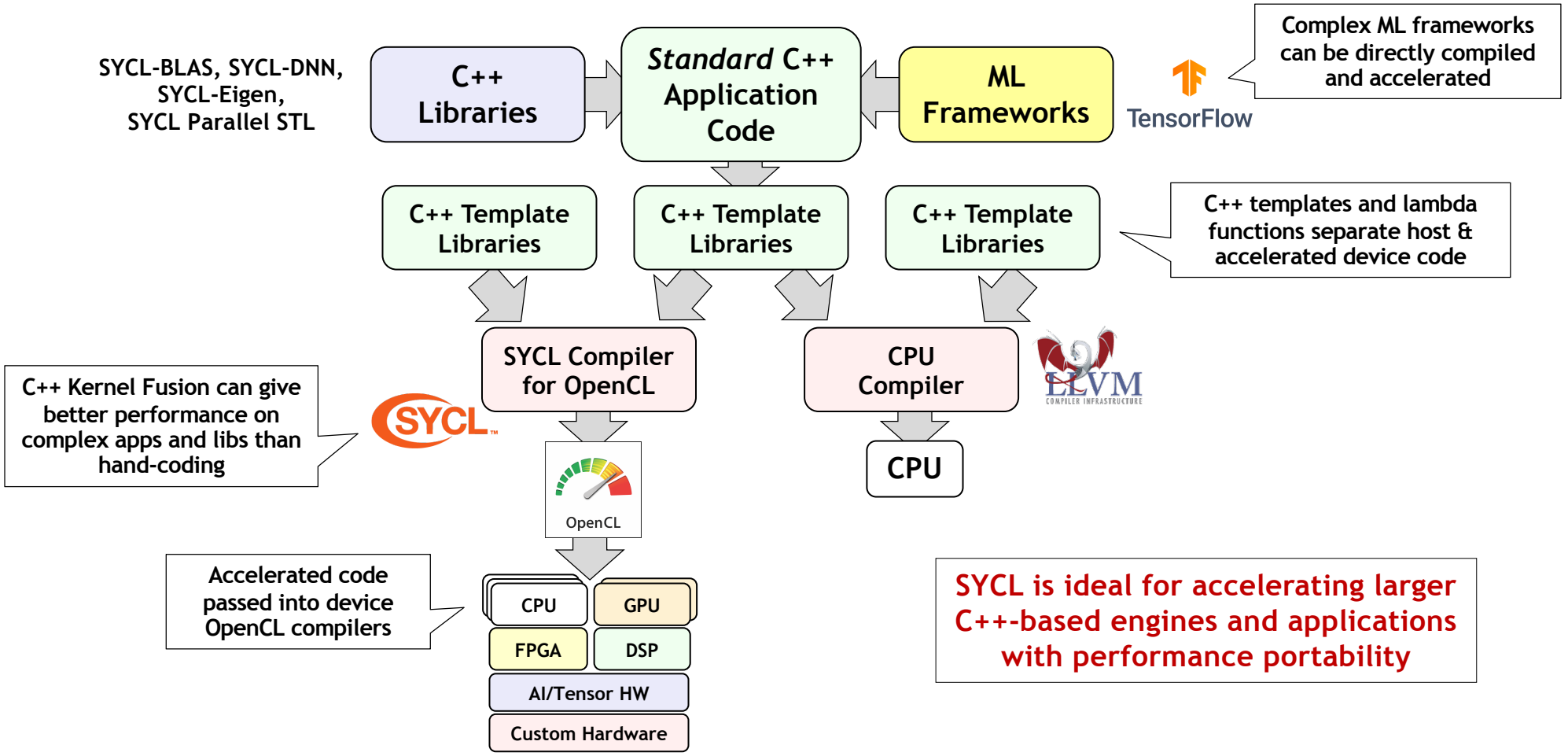



NNEF open source projects hosted on Khronos
NNEF GitHub repository under Apache 2.0
https://github.com/KhronosGroup/NNEF-Tools

**NNEF Model Zoo**
Now available on GitHub. Useful for checking that ingested NNEF produces acceptable results on target system

**NNEF adopts a rigorous approach to design lifecycle**
Especially important for safety-critical or mission-critical applications in automotive, industrial and infrastructure markets

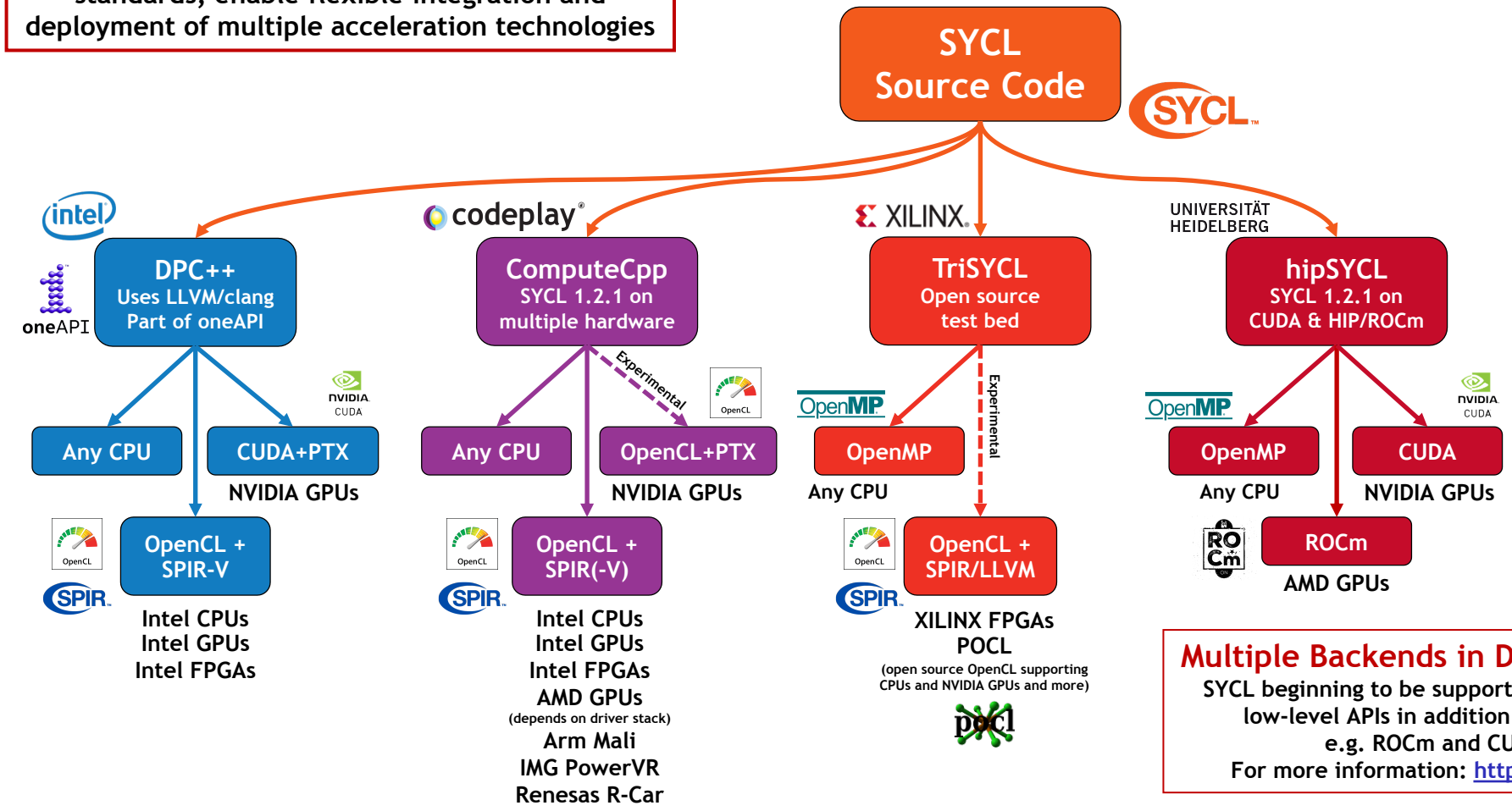
# SYCL Single Source C++ Parallel Programming



# SYCL Implementations

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

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




**Multiple Backends in Development**

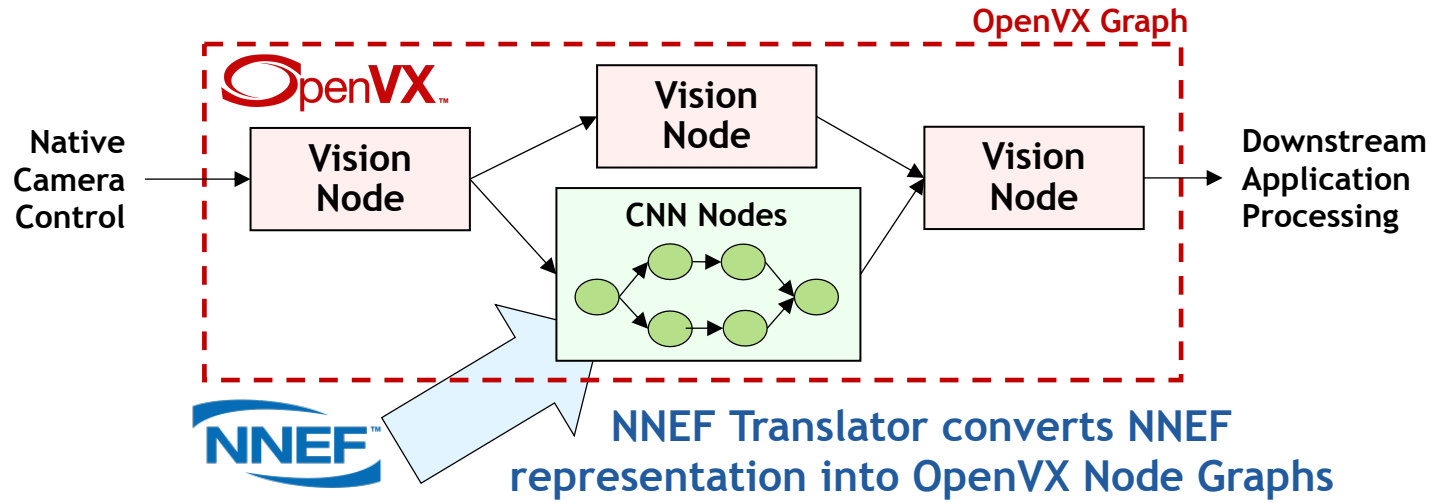
SYCL beginning to be supported on multiple low-level APIs in addition to OpenCL
e.g. ROCm and CUDA
For more information: http://sycl.tech

# OpenVX Cross-Vendor Inferencing

## OpenVX

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

Graph can contain vision processing and NN nodes - enables global optimizations
Optimized OpenVX drivers created and shipped by processor vendors
Implementable on almost any hardware or processor with performance portability
Run-time graph execution need very little host CPU interaction




**Performance comparable to hand-optimized, non-portable code**

Real, complex applications on real, complex hardware
Much lower development effort than hand-optimized code

AMD cadence ELVEES intel Imagination QUALCOMM NVIDIA Texas Instruments SYNOPSYS socionext VeriSilicon

Hardware Implementations

# OpenVX 1.3 Released October 2019


## Functionality Consolidation into Core 1.3

Neural Net Extension, NNEF Kernel Import, Safety Critical etc.

## Deployment Flexibility through Feature Sets

Conformant Implementations ship one or more complete feature sets
Enables market-focused Implementations
- Baseline Graph Infrastructure (enables other Feature Sets)
- Default Vision Functions
- Enhanced Vision Functions (introduced in OpenVX 1.2)
- Neural Network Inferencing (including tensor objects)
- NNEF Kernel import (including tensor objects)
- Binary Images
- Safety Critical (reduced features for easier safety certification)

https://www.khronos.org/registry/OpenVX/specs/1.3/html/OpenVX_Specification_1_3.html

## Open Source Conformance Test Suite

https://github.com/KhronosGroup/OpenVX-cts/tree/openvx_1.3

# OpenVX Open Source Implementation & Apps

## Open Source OpenVX Tutorial and Code Samples

https://github.com/rgiduthuri/openvx_tutorial
https://github.com/KhronosGroup/openvx-samples


## Open Source OpenVX 1.3 for Raspberry Pi

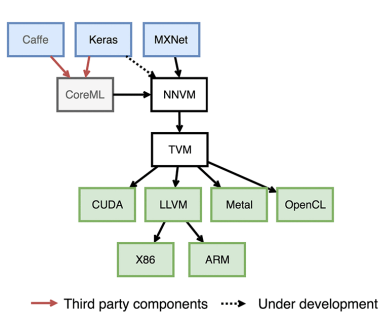
**Raspberry Pi 3 Model B with Raspbian OS**
**Memory access optimization via tiling/chaining**
**Highly optimized kernels on multimedia instruction set**
**Automatic parallelization for multicore CPUs and GPUs**
**Automatic merging of common kernel sequences**
https://github.com/KhronosGroup/OpenVX-sample-impl/tree/openvx_1.3

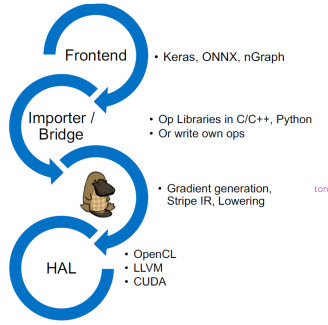
# OpenVX and OpenCV are Complementary

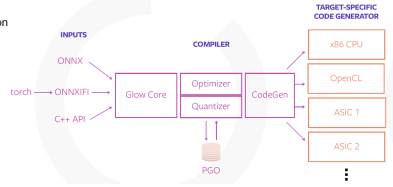
| | OpenCV | OpenVX |
|---|---|---|
| **Implementation** | Community driven open source library | Callable API implemented, optimized and shipped by hardware vendors |
| **Conformance** | Extensive OpenCV Test Suite but no formal Adopters program | Implementations must pass Khronos Conformance Test Suite to use trademark |
| **Scope** | 100s of imaging and vision functions<br>Multiple camera APIs/interfaces | Tight focus on dozens of core hardware accelerated functions plus extensions and accelerated custom nodes.<br>Uses external camera drivers |
| **Inferencing** | Deep Neural Network module to construct networks from layers for forward pass computations only.<br>Import from ONNX, TensorFlow, Torch, Caffe | Neural Network layers and operations represented directly in the OpenVX Graph.<br>NNEF direct import, ONNX through NNEF convertor |
| **Acceleration** | OpenCV 3.0 Transparent API (or T-API) enables function offload to OpenCL devices  | Implementation free to use any underlying API such as OpenCL. Can use OpenCL for Custom Nodes  |
| **Efficiency** | OpenCV 4.0 G-API graph model for some filters, arithmetic/binary operations, and well-defined geometrical transformations | Graph-based execution of all Nodes.<br>Optimizable computation and data transfer |
| **IP Protection** | None. Source code licensed under BSD.<br>Some modules require royalties/licensing | Protected under Khronos IP Framework - Khronos members agree not to assert patents against API when used in Conformant implementations |

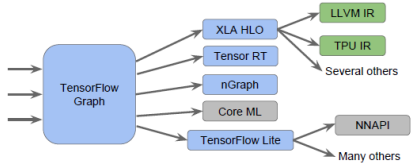
# Primary Machine Learning Compilers

| | tvm.ai | plaidML | Glow | XLA |
|---|---|---|---|---|
| | amazon | intel | facebook | Google |
| **Import Formats** | Caffe, Keras, MXNet, ONNX | TensorFlow Graph, MXNet, PaddlePaddle, Keras, ONNX | PyTorch, ONNX | TensorFlow Graph, PyTorch |
| **Front-end / IR** | NNVM / Relay IR | nGraph / Stripe IR | Glow Core / Glow IR | XLA HLO, MLIR |
| **Output** | OpenCL, LLVM, CUDA, Metal | OpenCL, LLVM, CUDA | OpenCL LLVM | LLVM, TPU IR, XLA IR TensorFlow Lite / NNAPI (inc. HW accel) |



# ML Compiler Steps

**Embedded NN Compilers**
CEVA Deep Neural Network (CDNN)
Cadence Xtensa Neural Network Compiler (XNNC)

| | tvm.ai / amazon | plaidML / intel | Glow / facebook | XLA / Google |
|---|---|---|---|---|
| Import Formats | Caffe, Keras, MXNet, ONNX | TensorFlow Graph, MXNet, PaddlePaddle, Keras, ONNX | PyTorch, ONNX | TensorFlow Graph, PyTorch |
| Front-end / IR | NNVM / Relay IR | nGraph / Stripe IR | Glow Core / Glow IR | XLA HLO, MLIR |
| Output | OpenCL, LLVM, CUDA, Metal | OpenCL, LLVM, CUDA | OpenCL, LLVM | LLVM, TPU IR, XLA IR, TensorFlow Lite / NNAPI (inc. HW accel) |

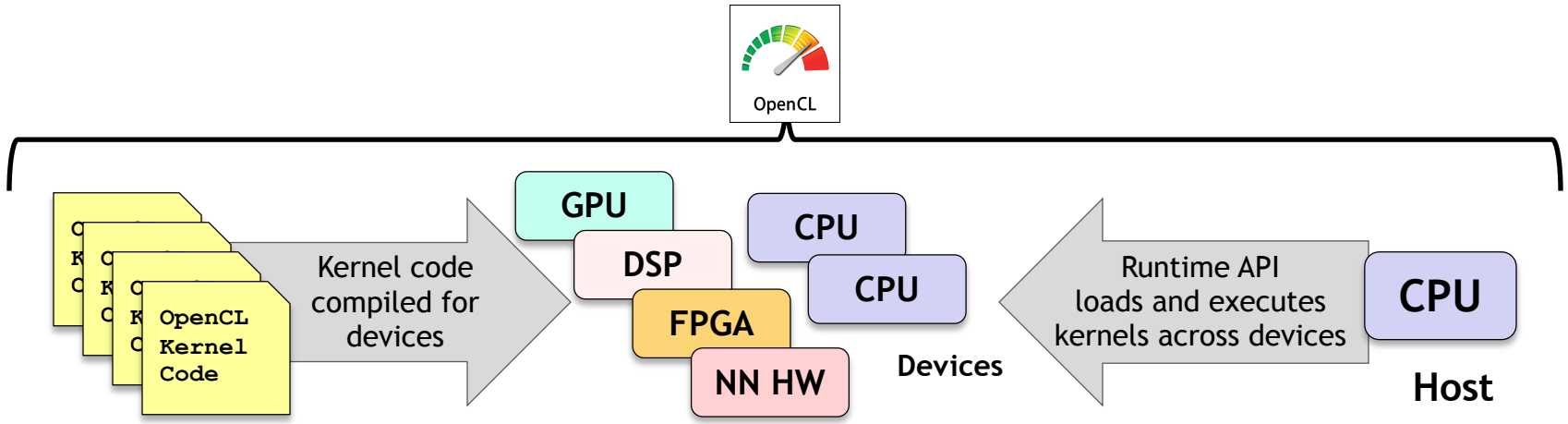
## Consistent Steps

1. Import Trained Network Description
2. Apply graph-level optimizations e.g. node fusion, node lowering and memory tiling
3. Decompose to primitive instructions and emit programs for accelerated run-times

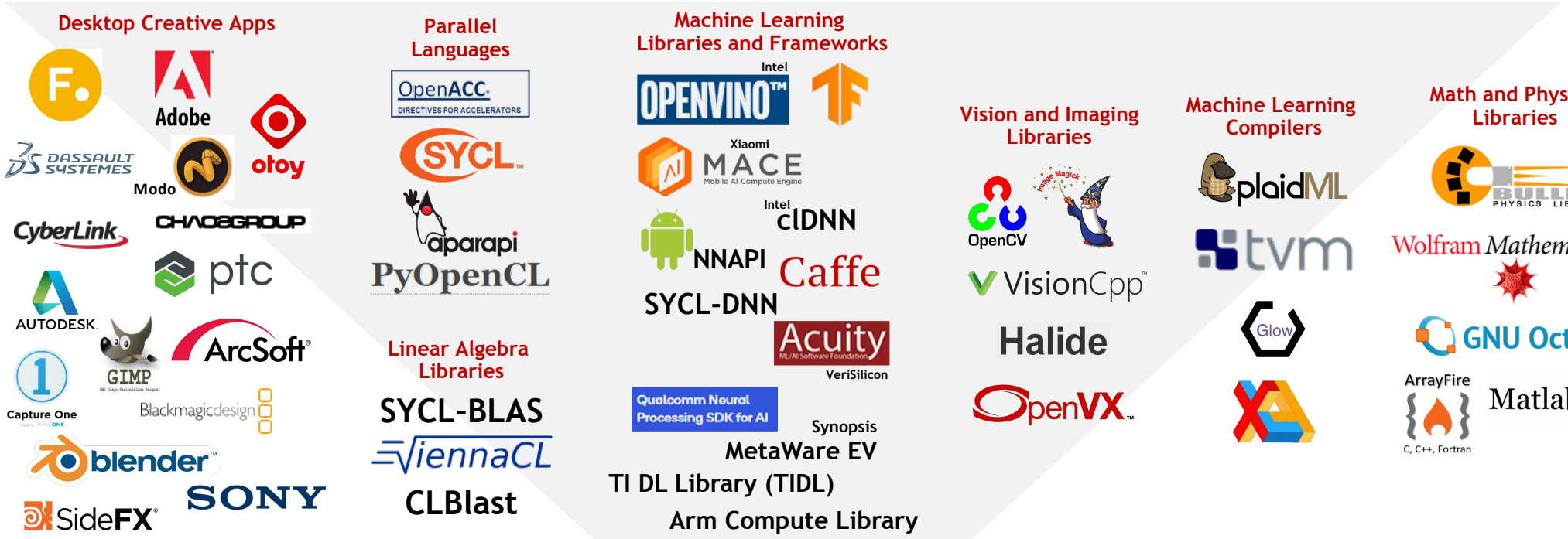
**Fast progress but still area of intense research**
If compiler optimizations are effective - hardware accelerator APIs can stay ‘simple’ and won’t need complex metacommands (combined primitive commands) like DirectML

# OpenCL - Low-level Parallel Programing

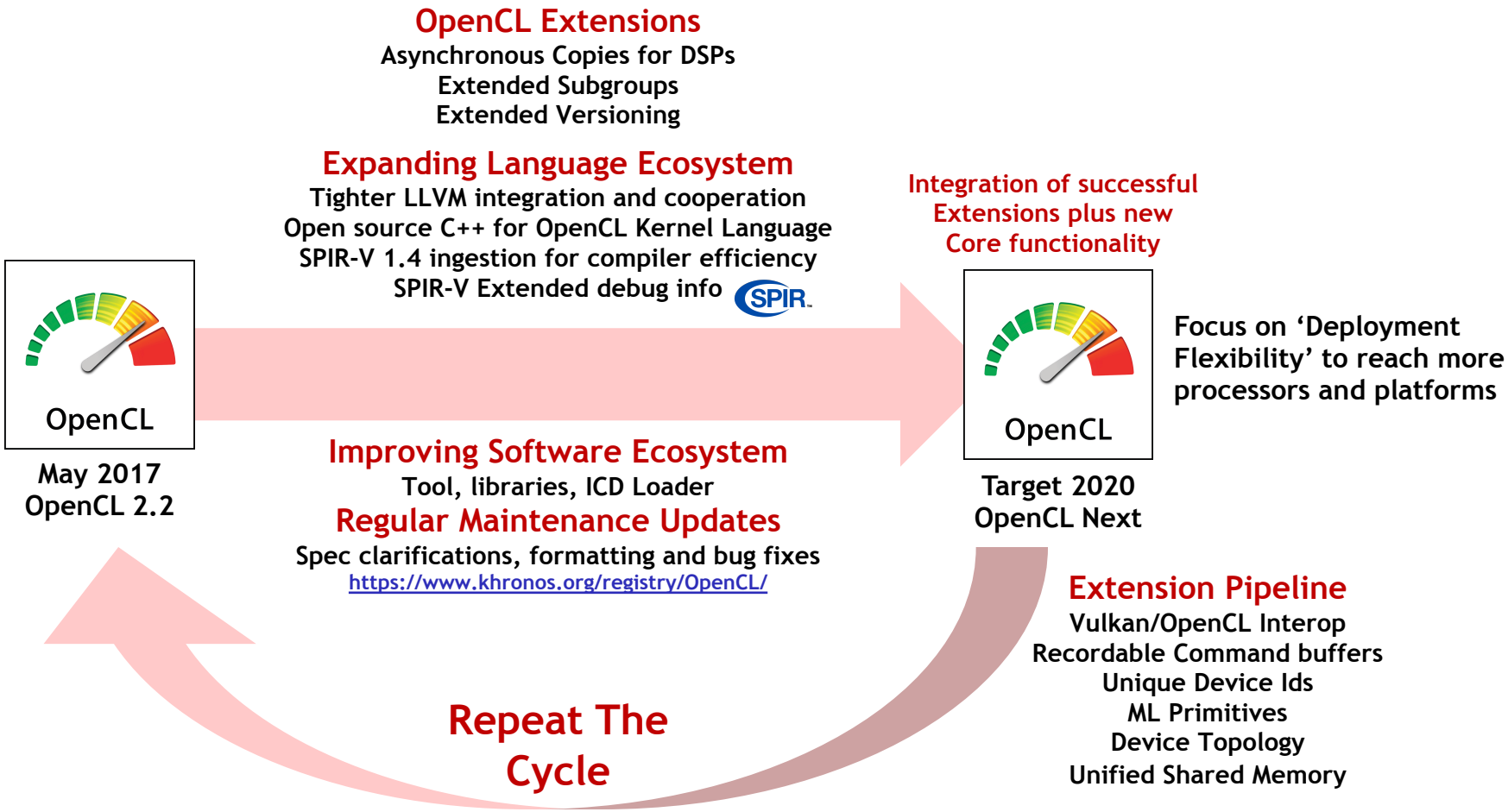
- **Low-level programming of heterogeneous parallel compute resources**
  - One code tree can be executed on CPUs, GPUs, DSPs, FPGAs, Tensor Processors …
- **OpenCL C or C++ language to write kernel programs to execute on any compute device**
  - Platform Layer API - to query, select and initialize compute devices
  - Runtime API - to build and execute kernels programs on multiple devices
- **The programmer gets to control:**
  - What programs execute on what device
  - Where data is stored in various speed and size memories in the system
  - When programs are run, and what operations are dependent on earlier operations



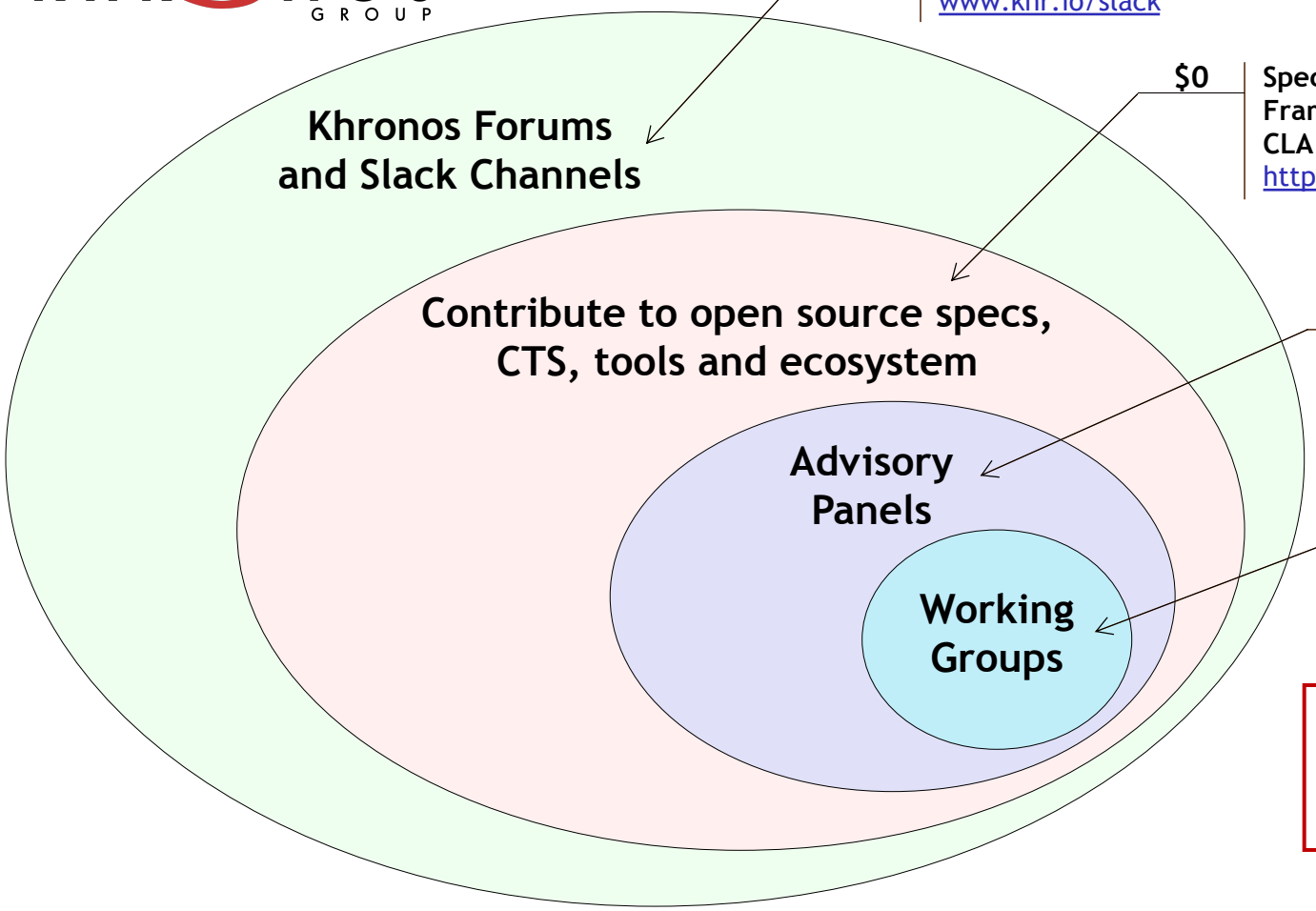
# OpenCL is Widely Deployed and Used



# OpenCL Evolution



# Engaging with the Khronos Ecosystem




**$0** — Open to all!
https://community.khronos.org/
www.khr.io/slack

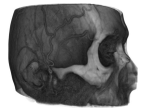
**$0** — Spec fixes and suggestions made under the Khronos IP Framework. Open source contributions under repo's CLA - typically Apache 2.0
https://github.com/KhronosGroup

**$0** — Invited Advisors under the Khronos NDA and IP Framework can comment and contribute to requirements and draft specifications
https://www.khronos.org/advisors/

**$** — Khronos members under Khronos NDA and IP Framework participate and vote in working group meetings. Starts at $3.5K/yr.
https://www.khronos.org/members/

**Any member *or* non-member can propose a new Khronos initiative**

# Safety Critical GPU API Evolution



OpenGL SC 1.0 - 2005
Fixed function graphics safety critical subset

OpenGL SC 2.0 - April 2016
Programmable Shaders safety critical subset

New Generation Safety Critical APIs for Graphics, Compute and Display

OpenGL ES 1.0 - 2003
Fixed function graphics

OpenGL ES 2.0 - 2007
Programmable Shaders

Vulkan 1.0 - 2016
Explicit Graphics and Compute

OpenCL

Potential OpenCL SC work will leverage the deployment flexibility of 'OpenCL Next' to minimize API surface area

Rendering Compute Display

**Industry Need**
for GPU Acceleration APIs designed to ease system safety certification is increasing
ISO 26262 / ASIL-D

ISO 26262

**Vulkan is Compelling Starting Point for SC GPU API Design**
Widely adopted, royalty-free open standard
Low-level explicit API with smaller surface area than OpenGL
Not burdened by debug functionality
Very little internal state
Well-defined thread behavior

**Clearly Definable Design Goals to Adapt Vulkan for SC**
Reduce driver size and complexity
-> Offline pipeline creation, no dynamic display resolutions
Deterministic Behavior
-> No ignored parameters, static memory management, eliminate undefined behaviors
Robust Error Handling
-> Error callbacks so app can respond, Fatal error callbacks for fast recovery initiation
C API - MISRA C Compliance

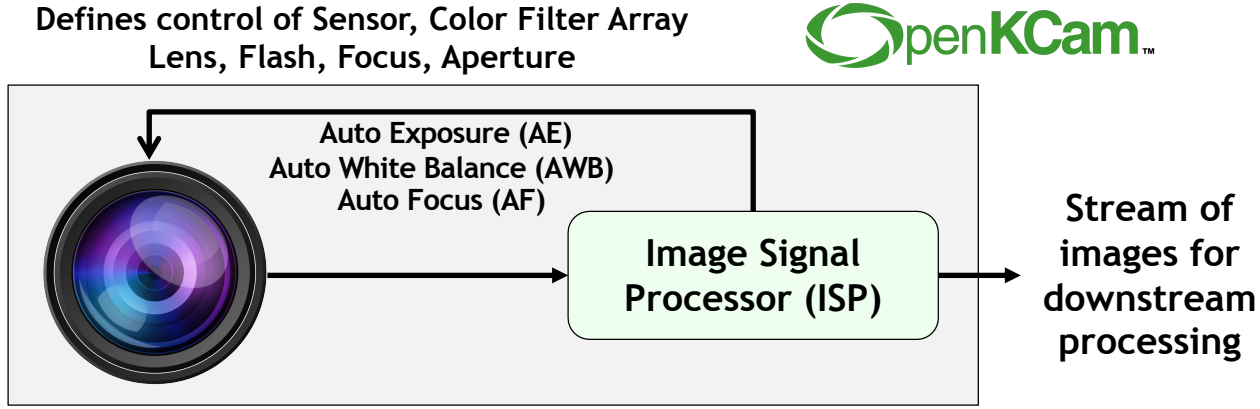
COREAVI
Beta Safety Critical Vulkan Demo: Booth 4-630b

Khronos Vulkan SC Working Group started work in February 2019

# Need for New Camera Control API Standard?

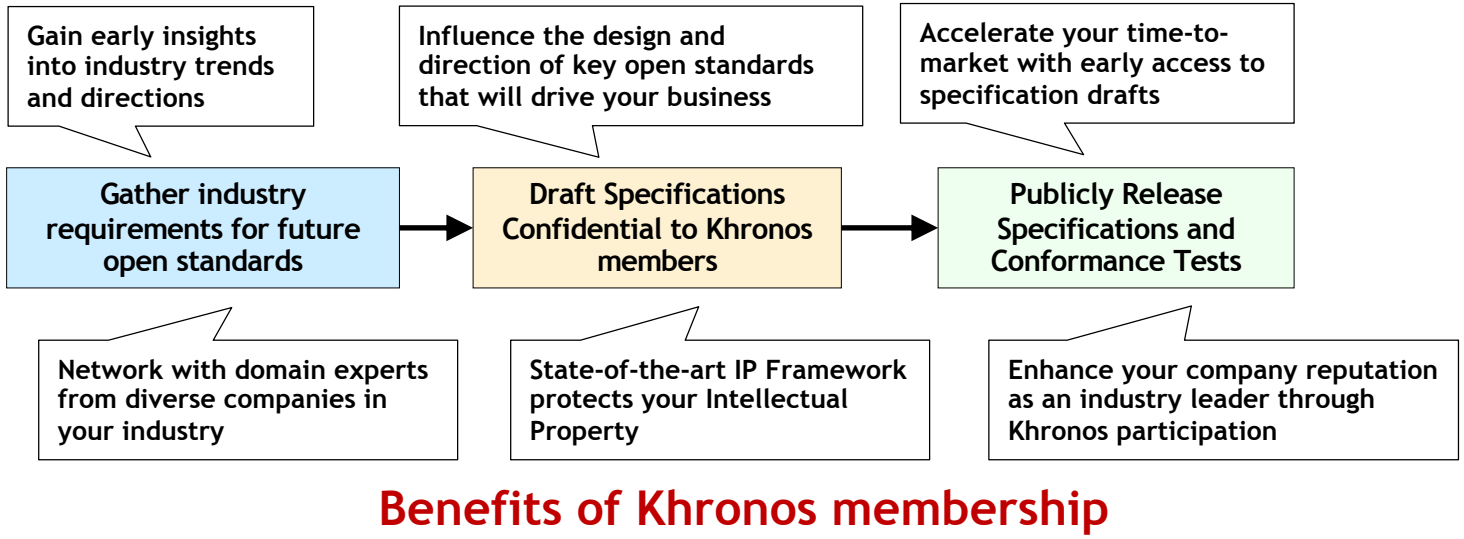
- **Khronos suspended work on OpenKCam standard several years ago**
  - Mobile market went proprietary - but embedded market has different needs
- **OpenKCAM was aiming at advanced control of ISP and camera with cross-platform portability**
  - Generate sophisticated image stream for advanced imaging & vision apps
  - Portable access to growing sensor diversity: e.g. depth sensors and sensor arrays
  - Cross sensor synch: e.g. synch of multiple camera and MEMS sensors
  - Advanced, high-frequency per-frame burst control of camera/sensor: e.g. ROI
  - Multiple input, output re-circulating streams with RAW, Bayer or YUV Processing

**Emerging potential liaison opportunity for cooperation – maybe over OpenKCam and GeniCam?**
**Work together to integrate next generation camera control and acceleration APIs?**



# Thank You!

- **Khronos is creating cutting-edge royalty-free open standards**
  - For 3D, compute, vision, inferencing acceleration
- **Information on Khronos Standards: www.khronos.org**
- **Any company is welcome to join Khronos: https://www.khronos.org/members/**
- **Neil Trevett: ntrevett@nvidia.com | @neilt3d**




**Benefits of Khronos membership**