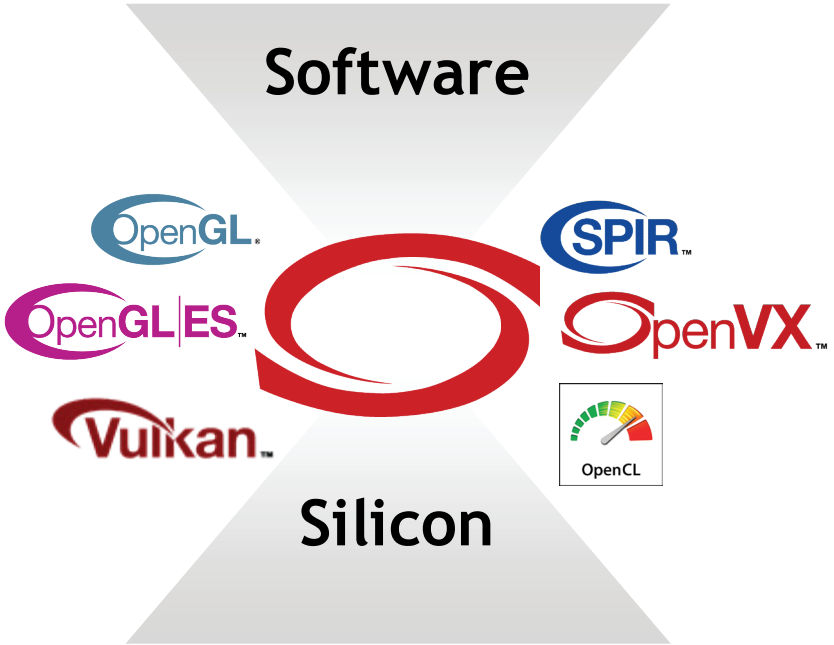
The Vision API Landscape

ARC Processor Summit Silicon Valley

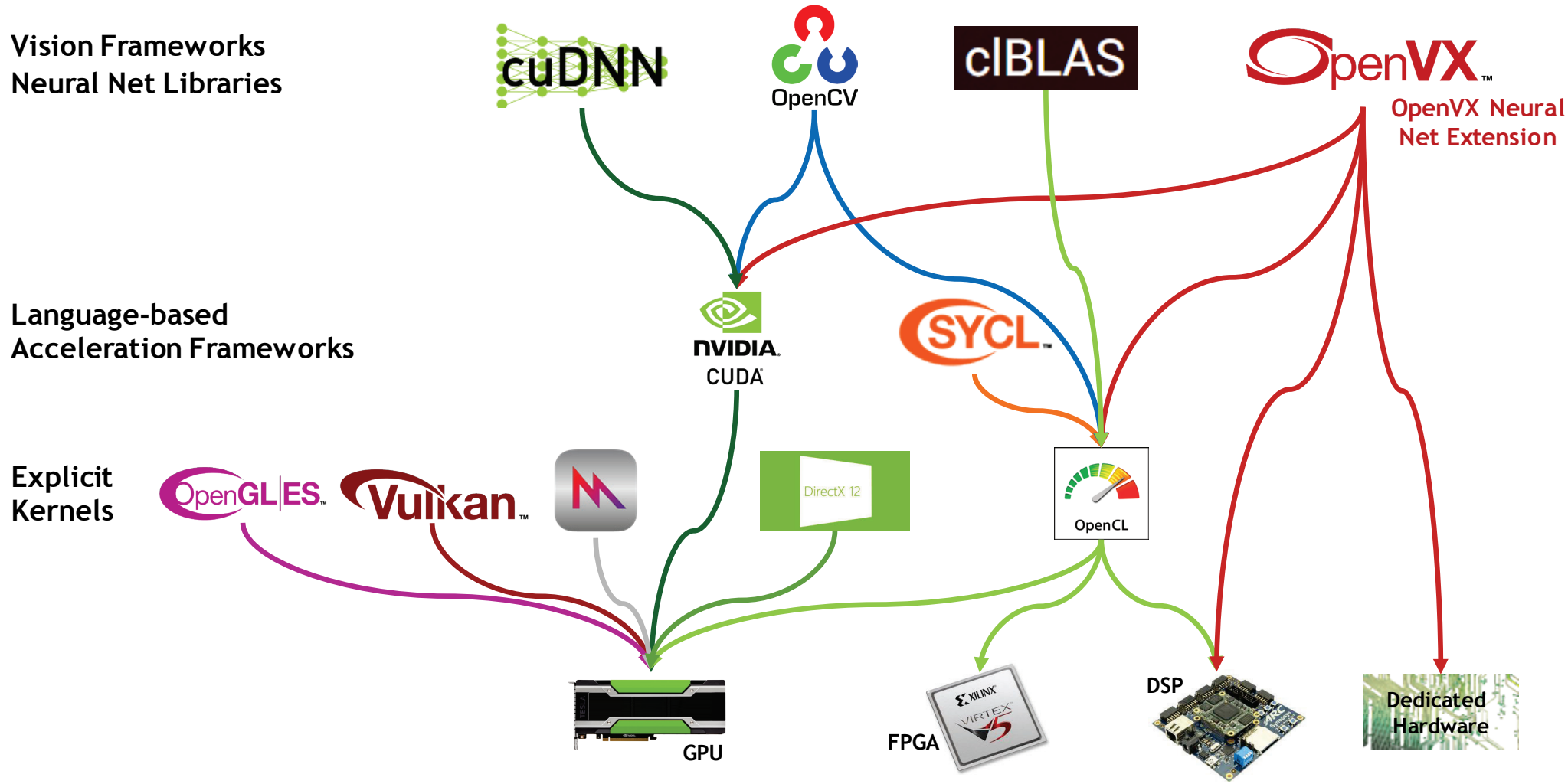
Neil Trevett
Vice President Mobile Ecosystem, NVIDIA | President, Khronos
ntrevett@nvidia.com | @neilt3d
September 2016

Khronos Open Standards



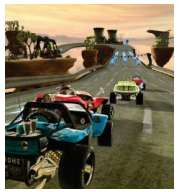
Khronos is an Industry Consortium of over 100 companies creating royalty-free, open standard APIs to enable software to access hardware acceleration for graphics, parallel compute and vision

Accelerated Vision API Landscape Overview



OpenGL ES

Fixed function
Pipeline



Vertex and
fragment shaders

32-bit integers and floats
NPOT, 3D/depth textures
Texture arrays
Multiple Render Targets

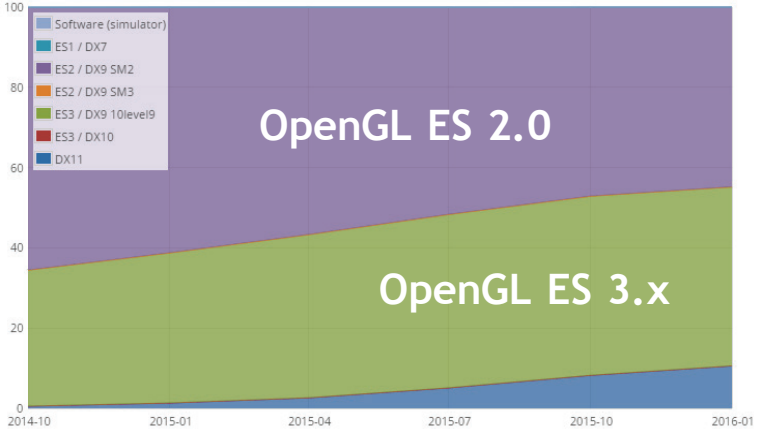
Compute Shaders

Tessellation and geometry shaders
ASTC Texture Compression
Floating point render targets
Debug and robustness for security

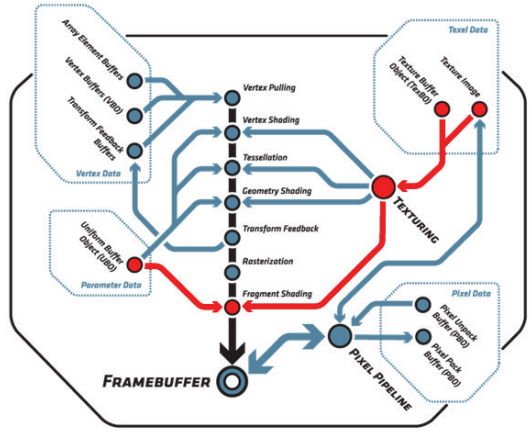
Epic's Rivalry demo using full Unreal Engine 4
https://www.youtube.com/watch?v=jRr-G95GdaM

OpenGL ES

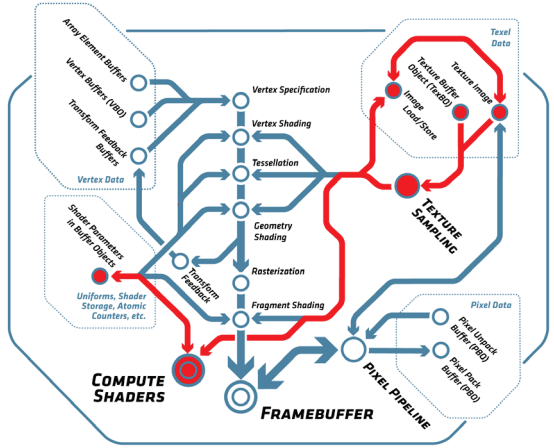
Close to 2 Billion OpenGL ES devices shipped in 2015



http://hwstats.unity3d.com/mobile/gpu.html



Vertex and Fragment Shaders



Compute Shaders

Three New Generation GPU APIs

Only Windows 10

Only Apple

'Half Way New Gen'
Retains Traditional Binding Model
Mixes OpenGL ES 3.1/OpenCL 1.2
C++11-based kernel language
Objective-C or Swift

Vulkan™
Cross Platform

Microsoft Windows xp

Windows 7

Windows 8

Windows 10

SteamOS

ubuntu

redhat

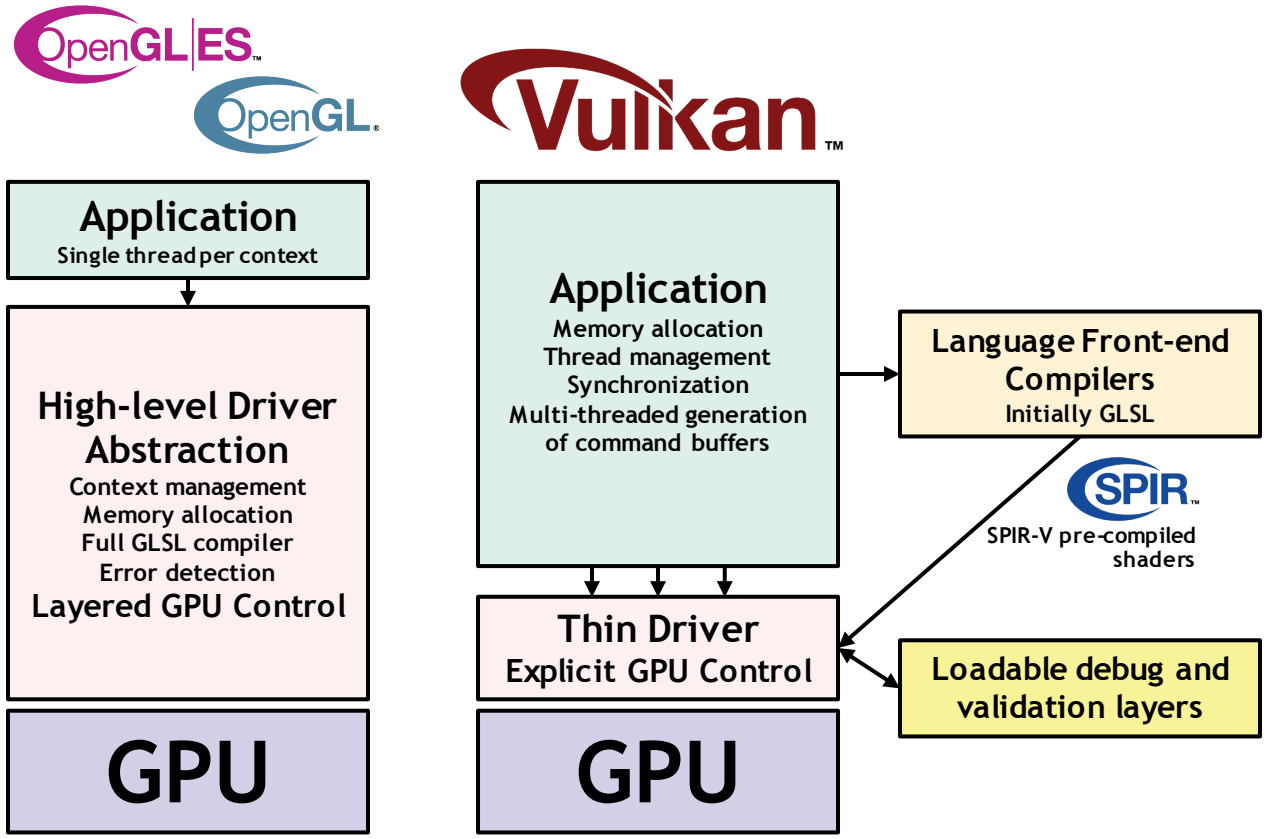
TIZEN™

N

Vulkan is extensible – just like OpenGL - and so is the new only generation API where hardware vendors can deliver innovations to market whenever they need

KHRONOS GROUP

Vulkan Explicit GPU Control



Vulkan 1.0 provides access to
OpenGL ES 3.1 / OpenGL 4.X-class GPU functionality
but with increased performance and flexibility

Vulkan Benefits

Resource management in app code:
Less driver hitches and surprises

Simpler drivers:
Improved efficiency/performance
Reduced CPU bottlenecks
Lower latency
Increased portability

Multi-threaded Command Buffers:
Command creation can be multi-threaded
Multiple CPU cores increase performance

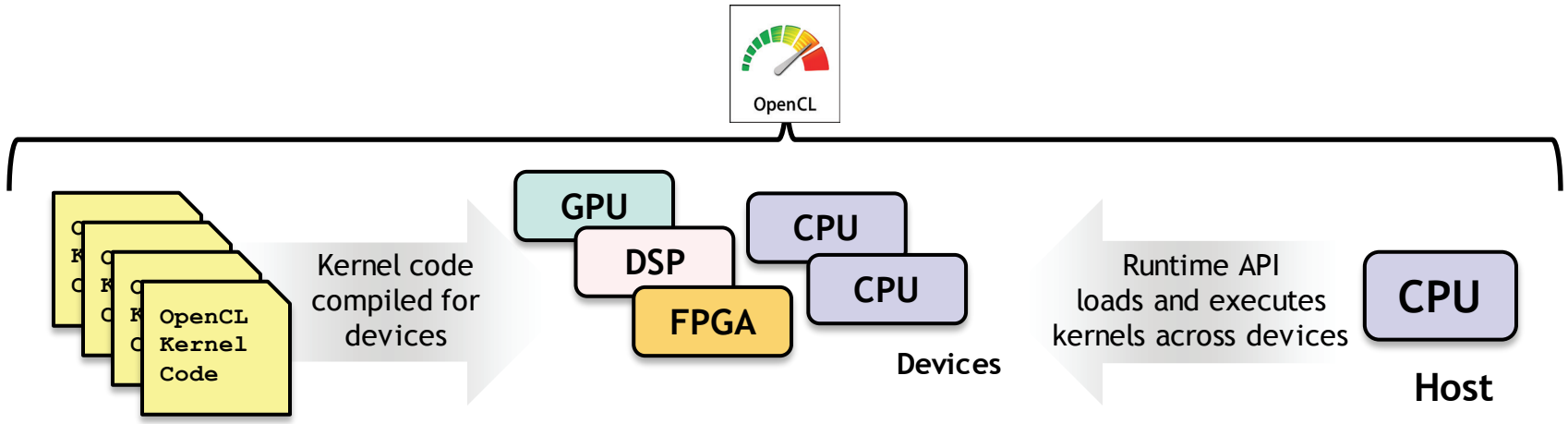
Graphics, compute and DMA queues:
Work dispatch flexibility

SPIR-V Pre-compiled Shaders:
No front-end compiler in driver
Future shading language flexibility

Loadable Layers
No error handling overhead in production code

OpenCL

- **Heterogeneous parallel programming of diverse compute resources**
  - One code tree can be executed on CPUs, GPUs, DSPs and FPGA
- **OpenCL = Two APIs and Two Kernel languages**
  - C Platform Layer API to query, select and initialize compute devices
  - C Runtime API to build and execute kernels across multiple devices
  - OpenCL C and OpenCL C++ kernel languages
- **New in OpenCL 2.2 - OpenCL C++ kernel language is a static subset of C++14**
  - Adaptable and elegant sharable code – great for building libraries
  - Templates enable meta-programming for highly adaptive software
  - Lambdas used to implement nested/dynamic parallelism



OpenCL 2.2 - Top to Bottom C++

Single Source C++ Programming
Full support for features in C++14-based Kernel Language

API and Language Specs
Brings C++14-based Kernel Language into core specification

Portable Kernel Intermediate Language
Support for C++14-based kernel language e.g. constructors/destructors

OpenCL C++ Kernel Language
SPIR-V 1.1 with C++ support
SYCL 2.2 for single source C++

SPIR-V in Core
Subgroups into core
Subgroup query operations
clCloneKernel
Low-latency device timer queries

Shared Virtual Memory
On-device dispatch
Generic Address Space
Enhanced Image Support
C11 Atomics
Pipes
Android ICD

Device partitioning
Separate compilation and linking
Enhanced image support
Built-in kernels / custom devices
Enhanced DX and OpenGL Interop

3-component vectors
Additional image formats
Multiple hosts and devices
Buffer region operations
Enhanced event-driven execution
Additional OpenCL C built-ins
Improved OpenGL data/event interop

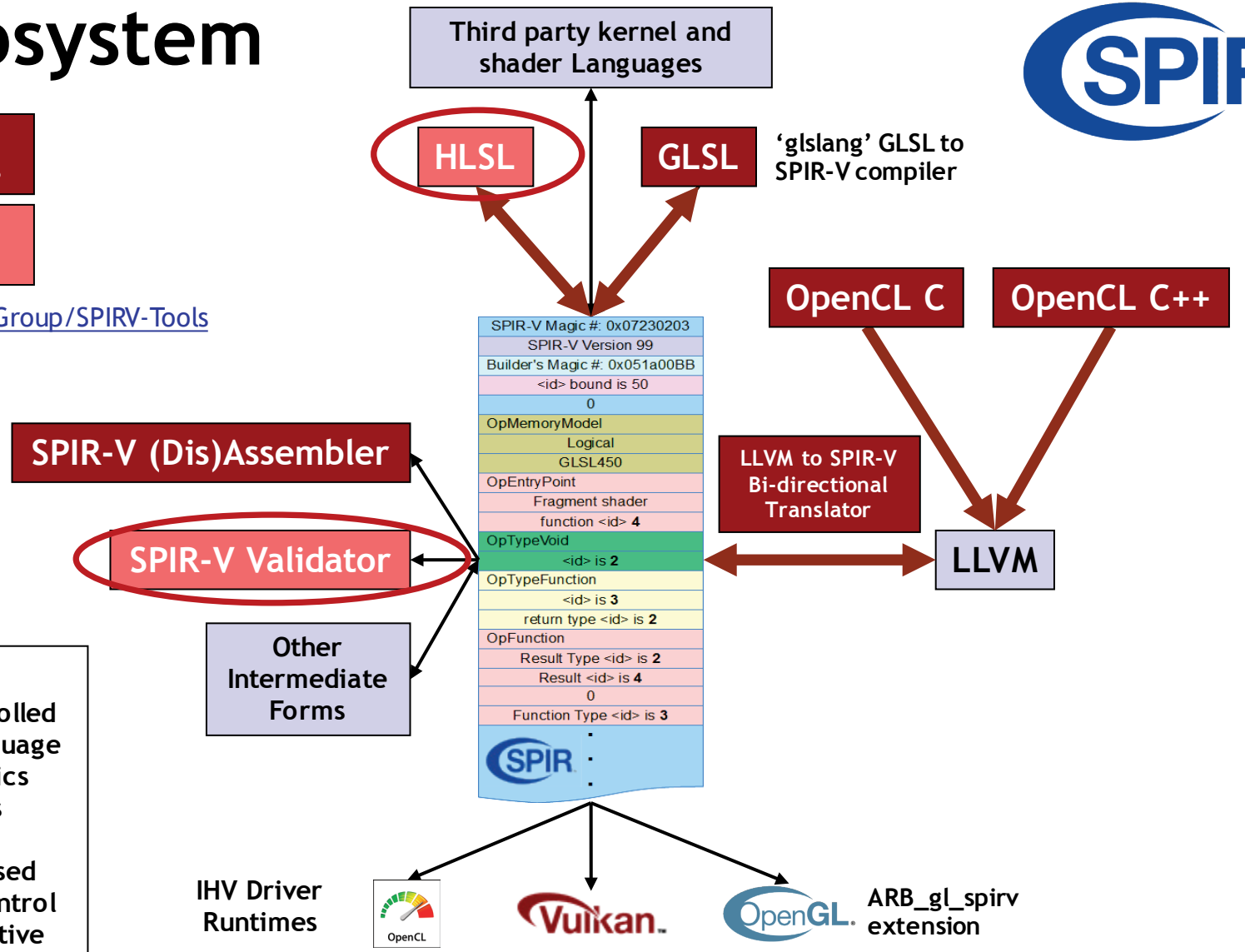
Dec08	18 months	Jun10	18 months	Nov11	24 months	Nov13	24 months	Nov15	7months	May16
OpenCL 1.0 Specification		OpenCL 1.1 Specification		OpenCL 1.2 Specification		OpenCL 2.0 Specification		OpenCL 2.1 Specification		OpenCL 2.2 PROVISIONAL

SPIR-V Ecosystem

Khronos has open sourced these tools and translators

Khronos plans to open source these tools soon

https://github.com/KhronosGroup/SPIRV-Tools



SPIR-V

- Khronos defined and controlled cross-API intermediate language
- Native support for graphics and parallel constructs
- 32-bit Word Stream
- Extensible and easily parsed
- Retains data object and control flow information for effective code generation and translation

SYCL for OpenCL

- **Single-source heterogeneous programming using STANDARD C++**
  - Use C++ templates and lambda functions for host & device code
- **Aligns the hardware acceleration of OpenCL with direction of the C++ standard**
  - C++14 with open source C++17 Parallel STL hosted by Khronos

Developer Choice

The development of the two specifications are aligned so code can be easily shared between the two approaches

C++ Kernel Language

Low Level Control

'GPGPU'-style separation of device-side kernel source code and host code

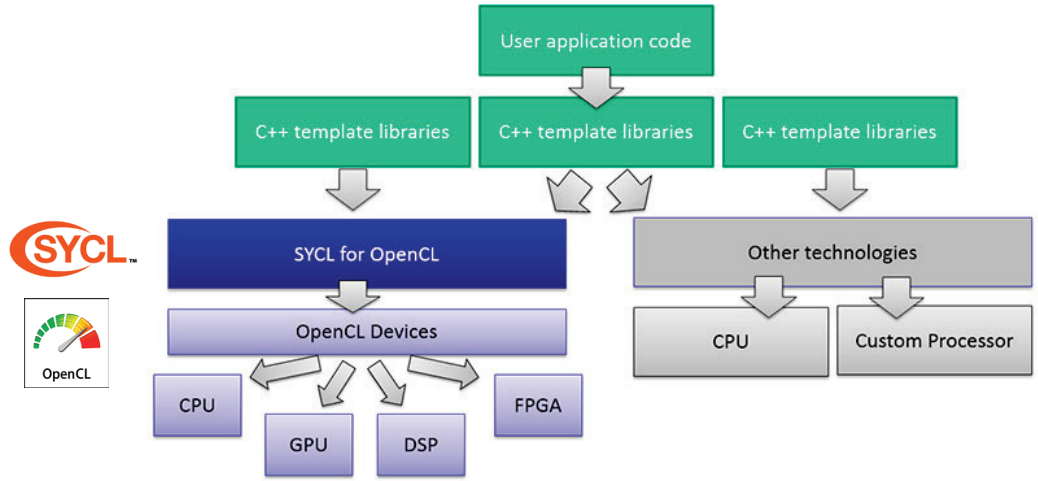
OpenCL

Single-source C++

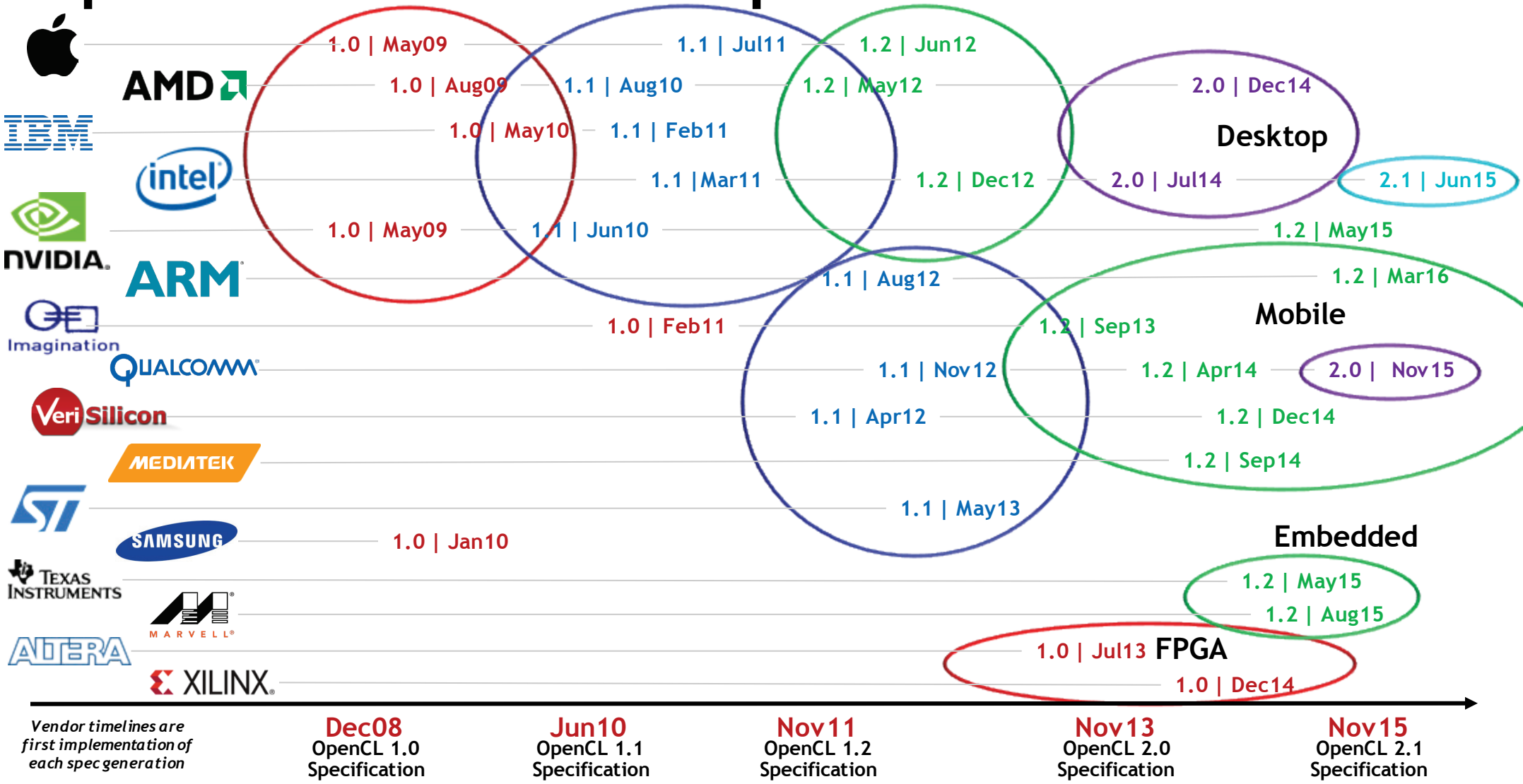
Programmer Familiarity

Approach also taken by C++ AMP and OpenMP

SYCL™

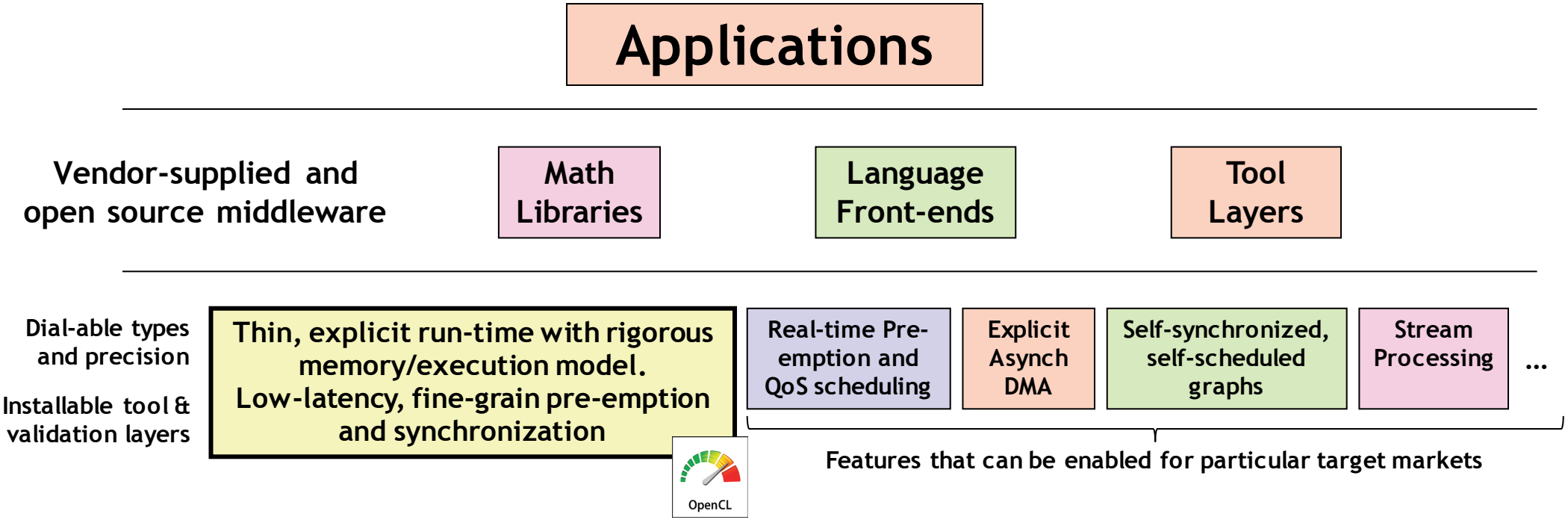


OpenCL Conformant Implementations



OpenCL Roadmap Discussions…

- Thin, powerful, explicit run-time for control and predictability
- Feature sets and dial-able precision for target market agility
- Installable tools and three layer ecosystem for flexibility



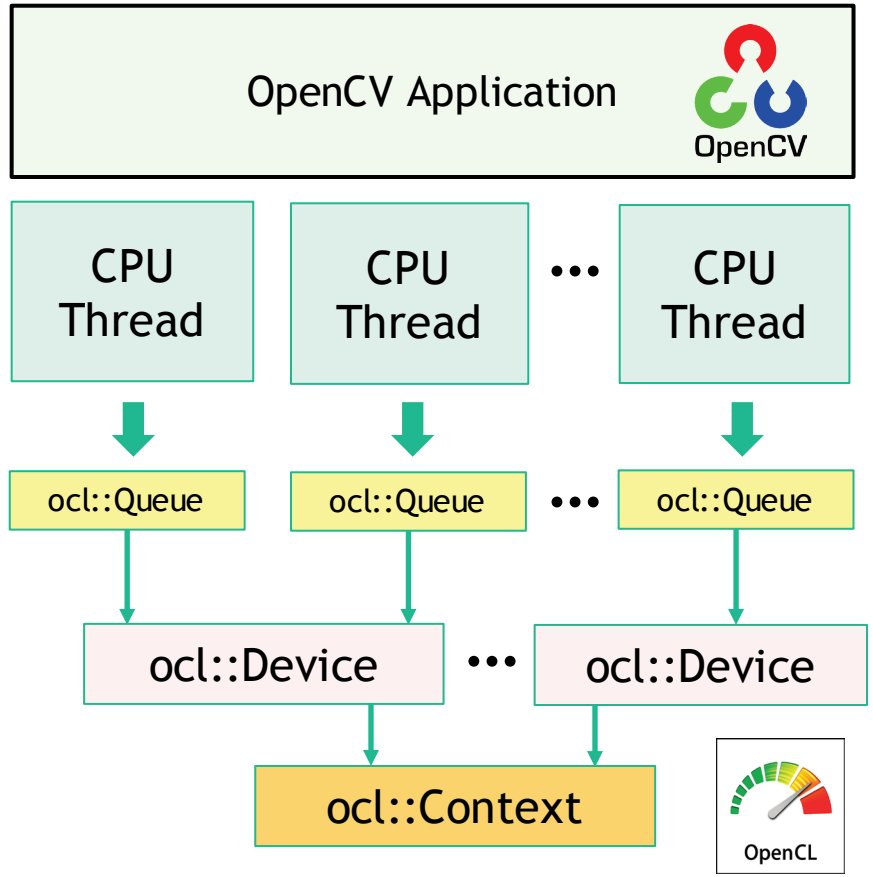
OpenCV

- **Extensive and widely used open source vision library - written in optimized C/C++**
  - Free-use BSD license
- **C++, C, Python and Java interfaces**
  - Windows, Linux, Mac OS, iOS and Android
- **Increasingly taking advantage of heterogeneous processing using OpenCL**
  - OpenCV 3.X Transparent API; single API entry for each function/algorithm
  - Dynamically loads OpenCL runtime if available; otherwise falls back to CPU code
  - Runtime Dispatching; no recompilation!

OpenCV is active open source - not an API specification
A strength and a weakness!
Production deployment often needs tightly defined callable API

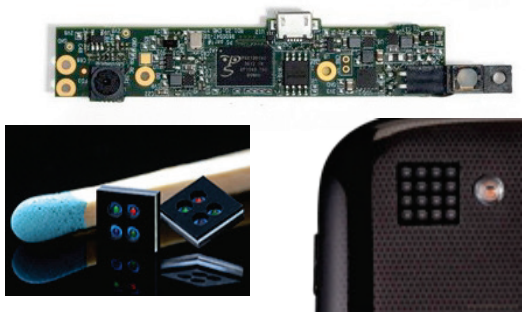
OpenCV Transparent API for OpenCL Kernel Offload

- One OpenCL queue per CPU thread
- CPU threads can share a device
- OpenCL kernels are executed asynchronously

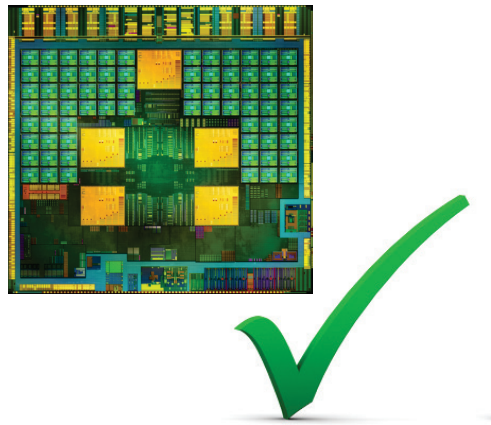


Vision Pipeline Challenges and Opportunities

Growing Camera Diversity



Diverse Vision Processors



Sensor Proliferation

Flexible sensor and camera control to GENERATE an image stream

OpenKCam™

Use efficient acceleration to PROCESS the image stream

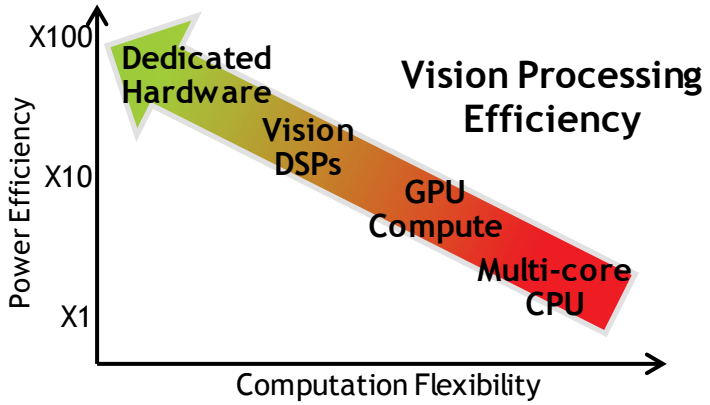
OpenVX™

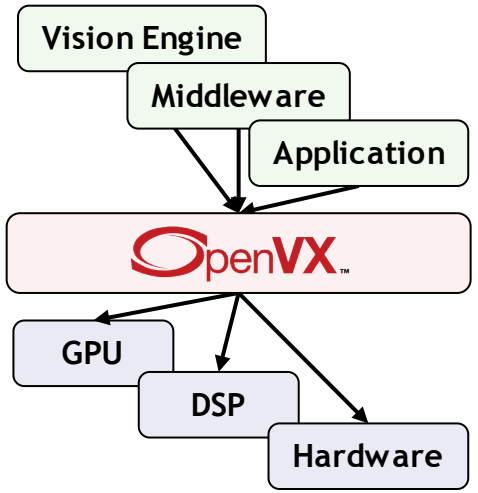
Combine vision output with other sensor data on device

StreamInput™

OpenVX – Low Power Vision Acceleration

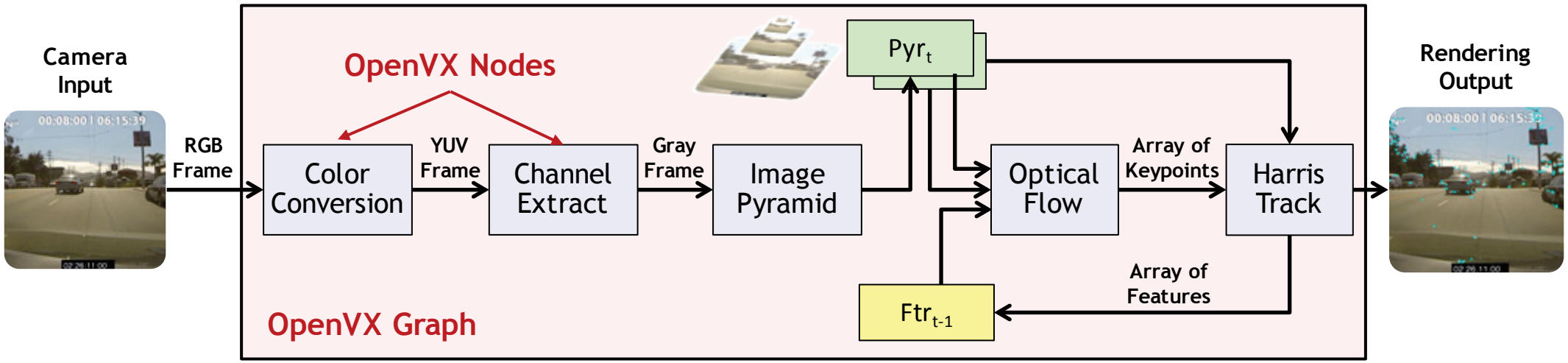
- **Precisely defined API for production deployment of vision acceleration**
  - Targeted at real-time mobile and embedded platforms
- **Higher abstraction than OpenCL for performance portability across diverse architectures**
  - Multi-core CPUs, GPUs, DSPs and DSP arrays, ISPs, Dedicated hardware…
- **Extends portable vision acceleration to very low power domains**
  - Doesn't require high-power CPU/GPU Complex or OpenCL precision
  - Low-power host can setup and manage frame-rate graph





OpenVX Graphs

- **OpenVX developers express a graph of image operations ('Nodes')**
  - Nodes can be on any hardware or processor coded in any language
- **Graphs can execute almost autonomously**
  - Possible to Minimize host interaction during frame-rate graph execution
- **Graphs are the key to run-time optimization opportunities…**



Feature Extraction Example Graph

OpenVX Efficiency through Graphs..

Graph Scheduling

Split the graph execution across the whole system: CPU / GPU / dedicated HW

Faster execution or lower power consumption

Memory Management

Reuse pre-allocated memory for multiple intermediate data

Less allocation overhead, more memory for other applications

Kernel Merge

Replace a sub-graph with a single faster node

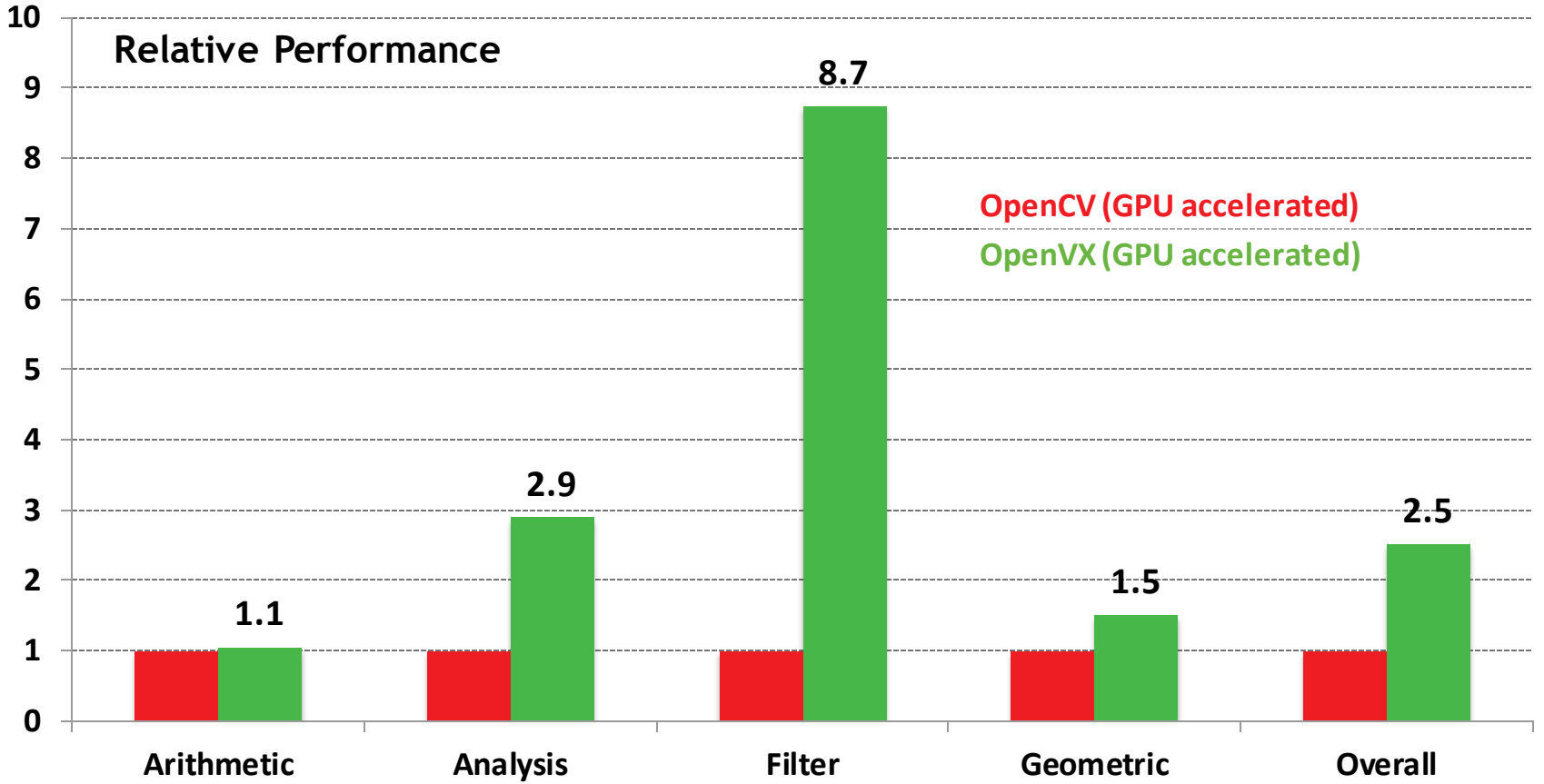
Better memory locality, less kernel launch overhead

Data Tiling

Execute a sub-graph at tile granularity instead of image granularity

Better use of data cache and local memory

Example Relative Performance



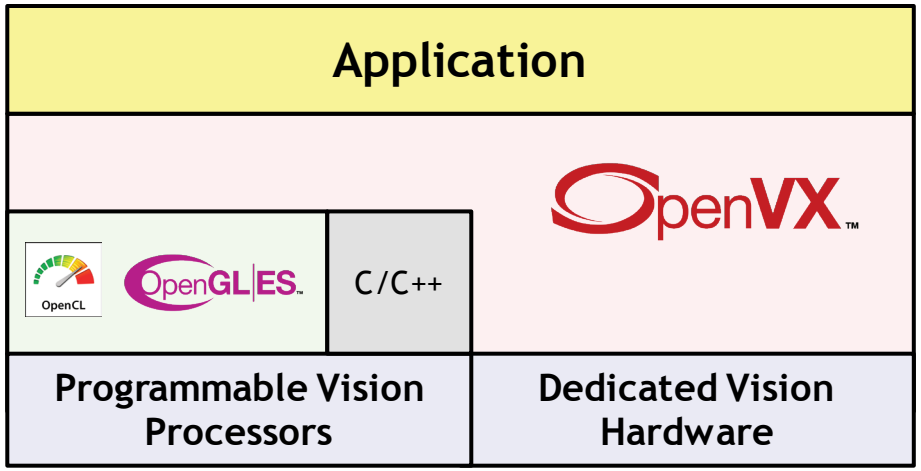
NVIDIA implementation experience.

Geometric mean of >2200 primitives, grouped into each categories, running at different image sizes and parameter settings

Layered Vision Processing Ecosystem

Implementers may use OpenCL or Compute Shaders to *implement* OpenVX nodes on programmable processors

And then developers can use OpenVX to enable a developer to easily *connect* those nodes into a graph



AMD OpenVX

- Open source, highly optimized for x86 CPU and OpenCL for GPU
- "Graph Optimizer" looks at entire processing pipeline and removes/replaces/merges functions to improve performance and bandwidth
- Scripting for rapid prototyping, without re-compiling, at production performance levels

http://gpuopen.com/compute-product/amd-openvx/

OpenVX enables the graph to be *extended* to include hardware architectures that don't support programmable APIs

The OpenVX graph enables implementers to *optimize* execution across diverse hardware architectures an drive to lower power implementations

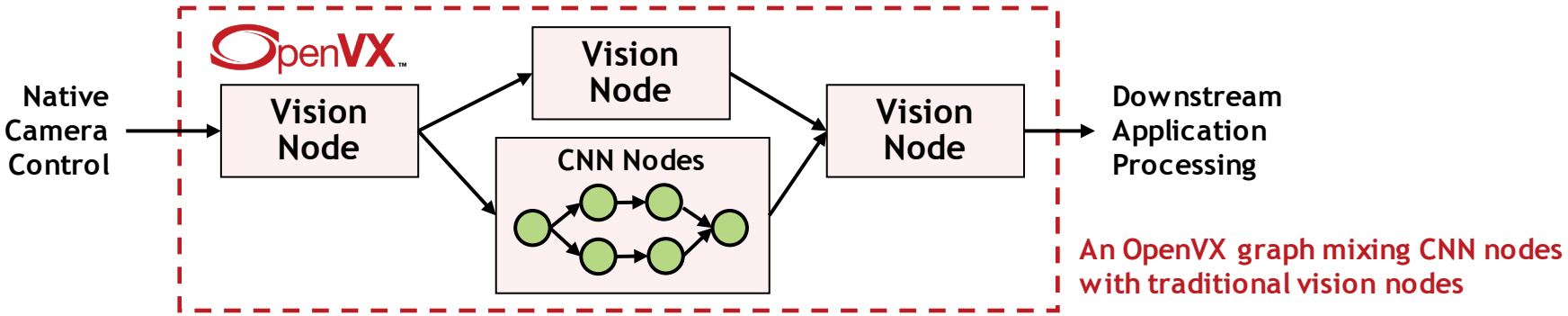
OpenVX 1.0 Shipping, OpenVX 1.1 Released!

- **Multiple OpenVX 1.0 Implementations shipping – spec in October 2014**
  - Open source sample implementation and conformance tests available
- **OpenVX 1.1 Specification released 2nd May 2016 at Embedded Vision Summit**
  - Expands node functionality AND enhances graph framework
- **Roadmap Discussions**
  - More nodes, programmable nodes (OpenCL or SPIR-V)
  - Feature Sets to enable market-targeting without fragmentation
- **OpenVX is EXTENSIBLE**
  - Implementers can add their own nodes at any time to meet customer and market needs

= shipping implementations

OpenVX Neural Net Extension

- **Convolution Neural Network topologies can be represented as OpenVX graphs**
  - Layers are represented as OpenVX nodes
  - Layers connected by multi-dimensional tensors objects
  - Layer types include convolution, activation, pooling, fully-connected, soft-max
  - CNN nodes can be mixed with traditional vision nodes
- **Import/Export Extension**
  - Efficient handling of network Weights/Biases or complete networks
- **The specification is provisional**
  - Welcome feedback from the deep learning community



Safety Critical APIs

OpenGL SC 1.0 - 2005
Fixed function graphics subset

OpenGL SC 2.0 - April 2016
Shader programmable pipeline subset

Experience and Guidelines

New Generation APIs for safety certifiable vision, graphics and compute
e.g. ISO 26262 and DO-178B/C

OpenGL ES 1.0 - 2003
Fixed function graphics

OpenGL ES 2.0 - 2007
Shader programmable pipeline

Khronos Launches Safety Critical Advisory Panel - and Invites Industry Experts to Participate

Experienced practitioners in the field of safety critical system design are invited to apply for Advisory Panel membership simply by sending an email to khronos_scap_apply@khronos.org. Please include your contact information, a short history of your experience along with why you feel you could help us set the future direction of safety critical APIs.

Read the press release

https://www.khronos.org/news/press/advisory-panel-to-create-design-guidelines-for-safety-critical-apis

OpenVX
Safety Critical vision processing

OpenCL
Safety Critical heterogeneous compute

Vulkan
Small driver size
Advanced functionality
Graphics and compute

Thanks – and Please Get Involved!

- **A diverse set of vision APIs in the industry**
  - Developer choice is good – but need to choose wisely!
- **Many APIs originally created to program GPUs**
  - But embedded vision processing needs are increasingly driving API roadmaps
- **Industry will tend to consolidate around leading APIs**
  - Working toward a multi-layer API ecosystem
  - Powerful foundational hardware APIs enabling rich middleware APIs and libraries
- **Any company or organization is welcome to join Khronos for a voice and a vote in any of its standards**
  - www.khronos.org

KHRONOS™ GROUP