Silicon Acceleration APIs
Embedded Technology 2016, Yokohama

Neil Trevett
Vice President Developer Ecosystem, NVIDIA | President, Khronos
ntrevett@nvidia.com | @neilt3d
November 2016
Khronos and Open Standards

Khronos is an Industry Consortium of over 100 companies
We create royalty-free, open standard APIs for hardware acceleration of
Graphics, Parallel Compute, Neural Networks and Vision
Accelerated API Landscape

Vision Frameworks
Neural Net Libraries

High-level Language-based Acceleration Frameworks

3D Graphics
Explicit Kernels

OpenVX Neural Net Extension

OpenCL
OpenCV
cuDNN
clBLAS
SYCL
NVIDIA CUDA
OpenGL ES
OpenGL SC
Vulkan
GPU
FPGA
DSP
Dedicated Hardware
OpenCL - Low-level Parallel Programing

- Low level programming of heterogeneous parallel compute resources
  - One code tree can be executed on CPUs, GPUs, DSPs and FPGA

- OpenCL 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

- New in OpenCL 2.2 - OpenCL C++ kernel language - 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
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.
Embedded Vision Acceleration
Embedded Technology, Yokohama

Shorin Kyo, Huawei
OpenVX - Low Power Vision Acceleration

- Targeted at vision acceleration in real-time, mobile and embedded platforms
  - Precisely defined API for production deployment
- 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
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...

Camera Input

OpenVX Nodes

- Color Conversion
- Channel Extract
- Image Pyramid
- Optical Flow
- Harris Track

Array of Keypoints

Array of Features

OpenVX Graph

Feature Extraction Example Graph

© Copyright Khronos Group 2016 - Page 9
OpenVX Efficiency through Graphs..

**Graph Scheduling**
Split the graph execution across the whole system:
- CPU / GPU / dedicated HW

**Memory Management**
Reuse pre-allocated memory for multiple intermediate data

**Kernel Merge**
Replace a sub-graph with a single faster node

**Data Tiling**
Execute a sub-graph at tile granularity instead of image granularity

- Faster execution or lower power consumption
- Less allocation overhead, more memory for other applications
- Better memory locality, less kernel launch overhead
- Better use of data cache and local memory
Example Relative Performance

<table>
<thead>
<tr>
<th>Category</th>
<th>NVIDIA Implementation Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometric</td>
<td>Geometric mean of &gt;2200 primitives, grouped into each category, running at different image sizes and parameter settings</td>
</tr>
</tbody>
</table>

Relative Performance

- Arithmetic: 1.1
- Analysis: 2.9
- Filter: 8.7
- Geometric: 1.5
- Overall: 2.5

OpenCV (GPU accelerated)
OpenVX (GPU accelerated)
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.

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 and drive to lower power implementations.

**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 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
  - Expands node functionality AND enhances graph framework
- OpenVX is EXTENSIBLE
  - Implementers can add their own nodes at any time to meet customer and market needs
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
NNEF - Neural Network Exchange Format

NNEF encapsulates neural network structure, data formats, commonly used operations (such as convolution, pooling, normalization, etc.) and formal network semantics.

NNEF 1.0 is currently being defined. OpenVX will import NNEF files.
Open GL 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=jRrG95GdaM

Closed to 2 Billion OpenGL ES devices shipped in 2015

ES 1.0 ES 1.1 ES 2.0 ES 3.0 ES 3.1 ES 3.2 AEP

Driver Update  Silicon Update  Silicon Update  Driver Update  Silicon Update  Driver Update

ES 1.0 ES 1.1 ES 2.0 ES 3.0 ES 3.1 ES 3.2 AEP

Driver Update  Silicon Update  Silicon Update  Driver Update  Silicon Update  Driver Update

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

© Copyright Khronos Group 2016 - Page 16
The Key Principle of Vulkan: Explicit Control

- Application tells the driver what it is going to do
  - *In enough detail* that driver doesn’t have to guess
  - *When* the driver needs to know it

- In return, driver promises to do
  - *What* the application asks for
  - *When* it asks for it
  - *Very quickly*

- *No driver magic - no surprises*
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 Multi-threading Efficiency

1. Multiple threads can construct Command Buffers in parallel. Application is responsible for thread management and synch.

2. Command Buffers placed in Command Queue by separate submission thread.

Applications can create graphics, compute and DMA command buffers with a general queue model that can be extended to more heterogeneous processing in the future.
Khronos Safety Critical APIs

DO-178B/C
ISO 26262

Experience and Guidelines
New Generation APIs for safety certifiable vision, graphics and compute

OpenGL SC 1.0 - 2005
Fixed function graphics subset

OpenGL SC 2.0 - April 2016
Shader programmable pipeline subset

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

Please Consider Joining Khronos!

Understand early industry requirements!

Influence how standards evolve!

Ship products that conform to international standards!

Access draft specs to build products faster!

Khronos is proven to RAPIDLY generate hardware API standards that create significant market opportunities

Any company or organization is welcome to join Khronos for a voice and a vote in any of its standards

www.khronos.org
Computer Vision solution with OpenVX over “VIP”

Gaku Ogura
Simon Jones

18 November 2016
VeriSilicon - Who We Are

- Founded in 2001
- Over 650 employees
- 70% dedicated to R&D
VeriSilicon – What we can do?

- Idea to Spec
- Spec to RTL
- RTL to Netlist
- Netlist to GDS
- IP Development & Licensing
- System Architecture
- Silicon Design
- Shipping
- Test
- Package
- Manufacture
Movement of Intelligent Devices

▲ Multi-Media
  ► Graphics, Video, Audio, Voice

Movement for next decade

▲ Many applications – need flexibility
▲ 1000s of algorithms – new one every day
▲ Compute intensive – need hardware acceleration
▲ Algorithms keep changing – need programmability

• Natural User Interface
  - VISION
  - Natural Language Processing
  - Sensors
Deep Learning: Neural Networks Return with Vengeance

- With big data and high density compute, deep neural networks can be trained to solve “impossible” computer vision problems.
The Basics of Deep Neural Networks (DNN)

- **Training**: $10^{16}$-$10^{22}$ MACs/dataset
  
  Offline (mostly)
  
  Analogous to compiling

- **Inference**: $10^6$-$10^{11}$ MACs/image
  
  On-the-fly

  Analogous to running s/w

  Embedded systems use this
Challenges of Real-Time Inference

Classification

Detection

Segmentation

- Majority of deep neural net computation is Multiply-Accumulate (MAC) for convolutions and inner products
- Memory bandwidth is a bigger challenge
  - Convolution in deep neural nets works with not just 2D images, but high-dimensional arrays (i.e. “tensor”)
  - Example: AlexNet (classification) has 60M parameters (240MB FP32), DDR BW with brute force implementation: 35GB/s
Processor Architectures

Programmability

Energy

CPU

GPU
- VeriSilicon
- ARM
- Imagination

DSP
- VeriSilicon
- CEVA
- Videantis
- Cadence
- Synopsys

Custom RTL

OpenCL
OpenVX
OpenCV

VIP
VeriSilicon
Efficient Processor Architecture to Enable DNN for Embedded Vision

- Programmable engine to cope with new CV algorithms
- Future new NN layers can be implemented here
- Scalable architecture to support different PPA (Performance, Power, Area) requirements

High utilization MACs for DNN Scalable architecture to support different PPA (Performance, Power, Area) requirements

Handles other DNN functions (normalization, pooling,...) Handles pruning, compression, batching... to increase MAC utilization and decrease DDR BW

Custom RTL to accelerate mature CV algorithms for 24/7 low power operation

I/F to extend HW acceleration for application specific purpose

For data synchronization between threads/blocks and minimize DDR BW
Use Case Study: Faster RCNN

• One of state-of-the-art object detection CNNs
  - Network configuration
    - 300 region proposals
    - 60M parameters, 30G MACs/image

• Collaborative computing in VIP
  - NN Engine: number crunching
    - Convolution layers, full connected layers
  - Tensor Processing Fabric: data rearrangement
    - LRN, max pooling, ROI pooling
  - Programmable Engine: precision compute
    - RPN, NMS, softmax

• Real-time performance (30fps HD) under 0.5W @ 28nm HPM
Scalable Performance from VIP8000

<table>
<thead>
<tr>
<th>VIP8000 Series</th>
<th>VIP Nano</th>
<th>VIP8000UL</th>
<th>VIP8000L</th>
<th>VIP8000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Execution cores</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>Clock Frequency SVT @WC125C (MHz)</td>
<td>800</td>
<td>800</td>
<td>800</td>
<td>800</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HD Performance@800MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perspective Warping</td>
</tr>
<tr>
<td>Optical Flow LK</td>
</tr>
<tr>
<td>Pedestrian Detection</td>
</tr>
<tr>
<td>Convolutional Neural Network (AlexNet)</td>
</tr>
</tbody>
</table>

- Scalable performance with SAME application code on different processor variants
Comprehensive Multi-Media/Pixel Subsystems

- CPU
- VPUCore (Video)
- GC Core (GPU)
- VIP Core (Vision/Image)
- ZSP (Audio/Voice)
- ISP

VeriSilicon Universal Compression

- AXI Bus
- IO Controller
- DMA Engine
- DC (Display)
- Memory Controller
- Wireless Connectivity
- Mixed Signal IPs

Local Bus