TIOVX – TI’s OpenVX Implementation

28 Sept 2017

Agenda

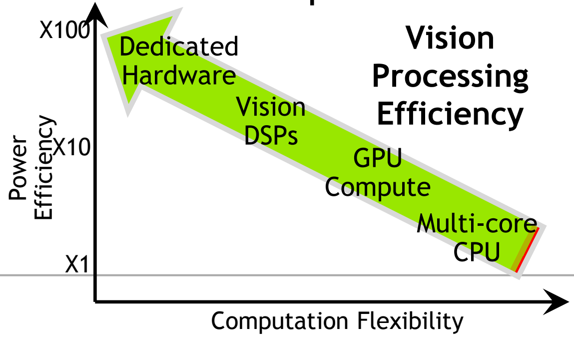
- Introduction to OpenVX
- OpenVX on TI SoCs – TIOVX
- Getting Started with TIOVX

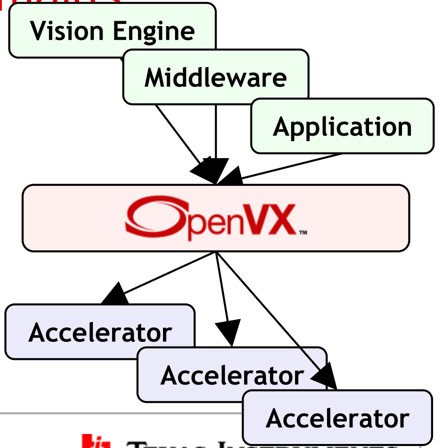
Agenda

- Introduction to OpenVX
- OpenVX on TI SoCs – TIOVX
- Getting Started with TIOVX

OpenVX – Low-Power Vision Acceleration

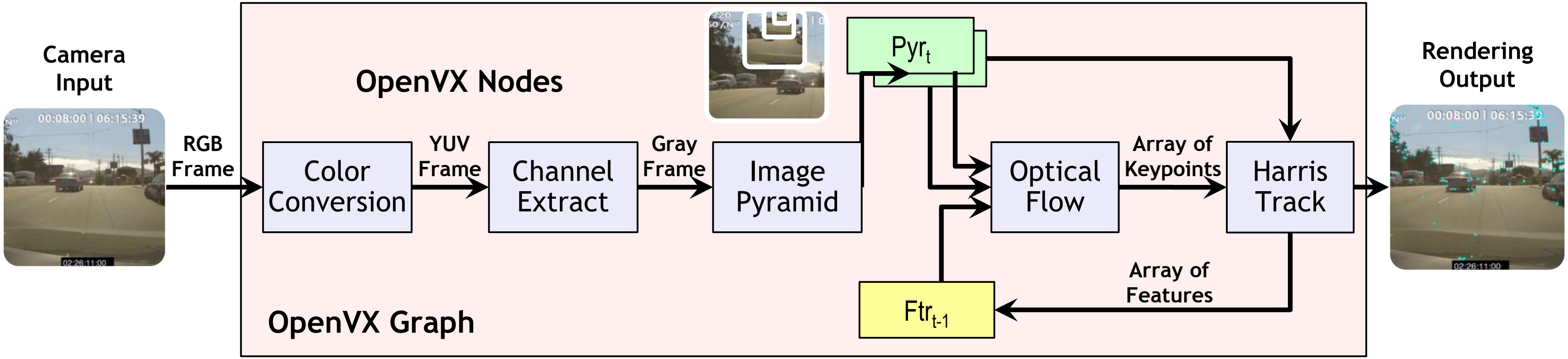
- Higher-level abstraction API
  - Targeted at real-time mobile and embedded platforms
- Performance portability across diverse architectures
  - Multi-core CPUs, GPUs, DSPs, ISPs, Dedicated hardware, …
- Extends portable vision acceleration to very low-power domains
  - Doesn't require high-power CPU/GPU Complex
  - Lower precision requirements than OpenCL





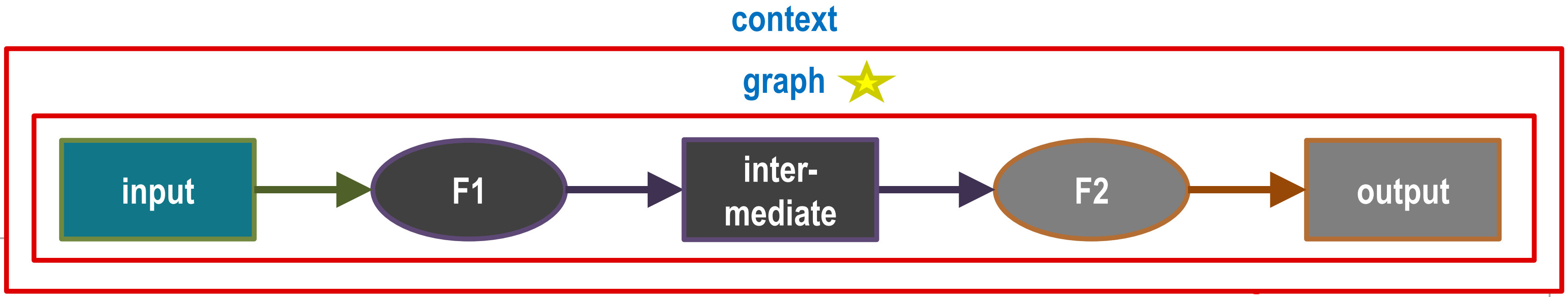
OpenVX Graphs

- OpenVX developers express a graph of image operations (‘Nodes’)
  - Nodes can be on any hardware or processor coded in any language
  - For example, on GPU, nodes may implemented in OpenCL
- Minimizes host interaction during frame-rate graph execution
  - Host processor can setup graph which can then execute almost autonomously



An OpenVX "Hello, World !!!" Program

```
vx_context context = vxCreateContext();
vx_graph graph = vxCreateGraph( context );

vx_image input = vxCreateImage( context, 640, 480, VX_DF_IMAGE_U8 );
vx_image output = vxCreateImage( context, 640, 480, VX_DF_IMAGE_U8 );
vx_image intermediate = vxCreateVirtualImage( graph, 640, 480, VX_DF_IMAGE_U8 );
vx_node F1 = vxF1Node( graph, input, intermediate );
vx_node F2 = vxF2Node( graph, intermediate, output );

vxVerifyGraph( graph );
vxProcessGraph( graph );
```



More Details on OpenVX Standard

- Khronos OpenVX website
  - https://www.khronos.org/openvx/
- OpenVX v1.1 specification and additional resources
  - https://www.khronos.org/registry/OpenVX/
  - https://www.khronos.org/openvx/resources
- Khronos OpenVX v1.1 Video Tutorials
  - https://youtu.be/JZZCNcfIqqs?list=PLYO7XTAX41FP01wTyWfwiNW3xq9IDRAnO

Agenda

- Introduction to OpenVX
- OpenVX on TI SoCs – TIOVX
- Getting Started with TIOVX

TIOVX - OpenVX Implementation on TI SoC

GOAL: Help customers easily maximize performance on TI platforms while minimizing development cost.

Performance

Graph-based model

All work defined at initialization time to maximize system utilization and minimize run-time latencies.

True Heterogeneous Compute

Easy/unified access to all ARM, DSP, Vision (EVE) Cores

Optimized Libraries

Fully optimized OpenVX 1.1 kernels on C6x DSP

DMA Integration

Simple interface to add block-based DMA to kernels (BAM)

Virtual Buffers

Intermediate buffers can be optimized out, or reside only in L2.

Open Standard

Conformant to **OpenVX v1.1**

Hardware Abstraction

Same application works across TDA2x/3x family of SoCs on range of SW environments from Linux to TI-RTOS

Ease of use

Full development, including DMA, DSP/EVE intrinsic emulation, can be done on PC using TI OpenVX, then simple recompile on platform.

PyTIOVX tool generates OpenVX Application code using compact graph description

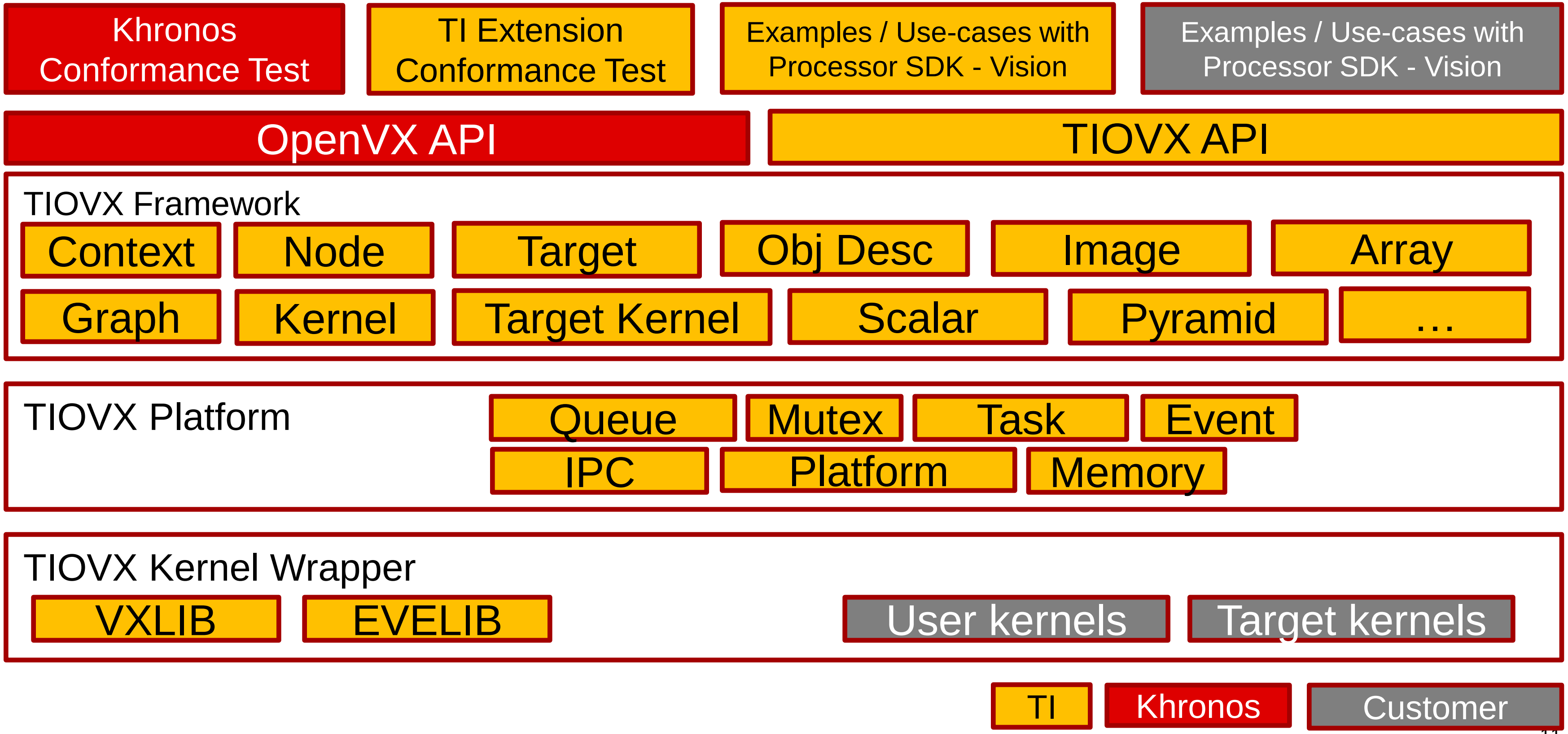
Development Cost

Result: Full entitlement on TI SoCs and remove barrier to entry for OpenVX developers

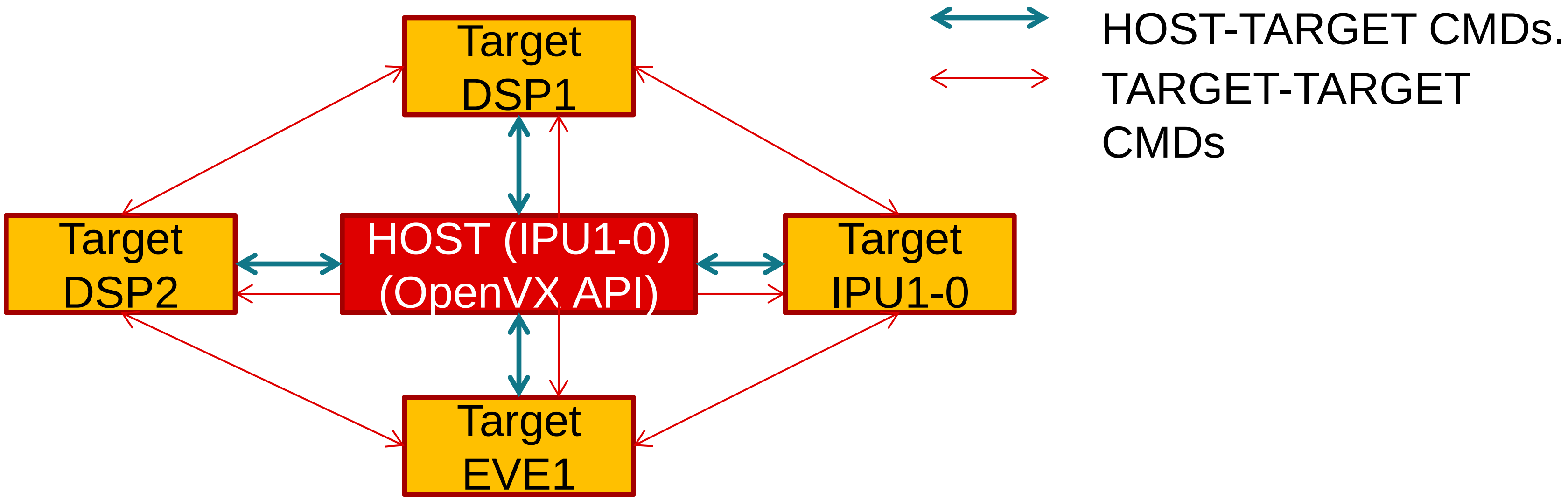
OpenVX 1.1 Supported Platforms

	TDA2x	TDA2Eco	TDA3x
OpenVX Target CPUs	ARM A15, C66x DSP, ARM M4, EVE	ARM A15, C66x DSP, ARM M4	ARM M4, C66x DSP, EVE
OS	TI-RTOS, Linux	TI-RTOS, Linux	TI-RTOS

TI OpenVX SW Stack

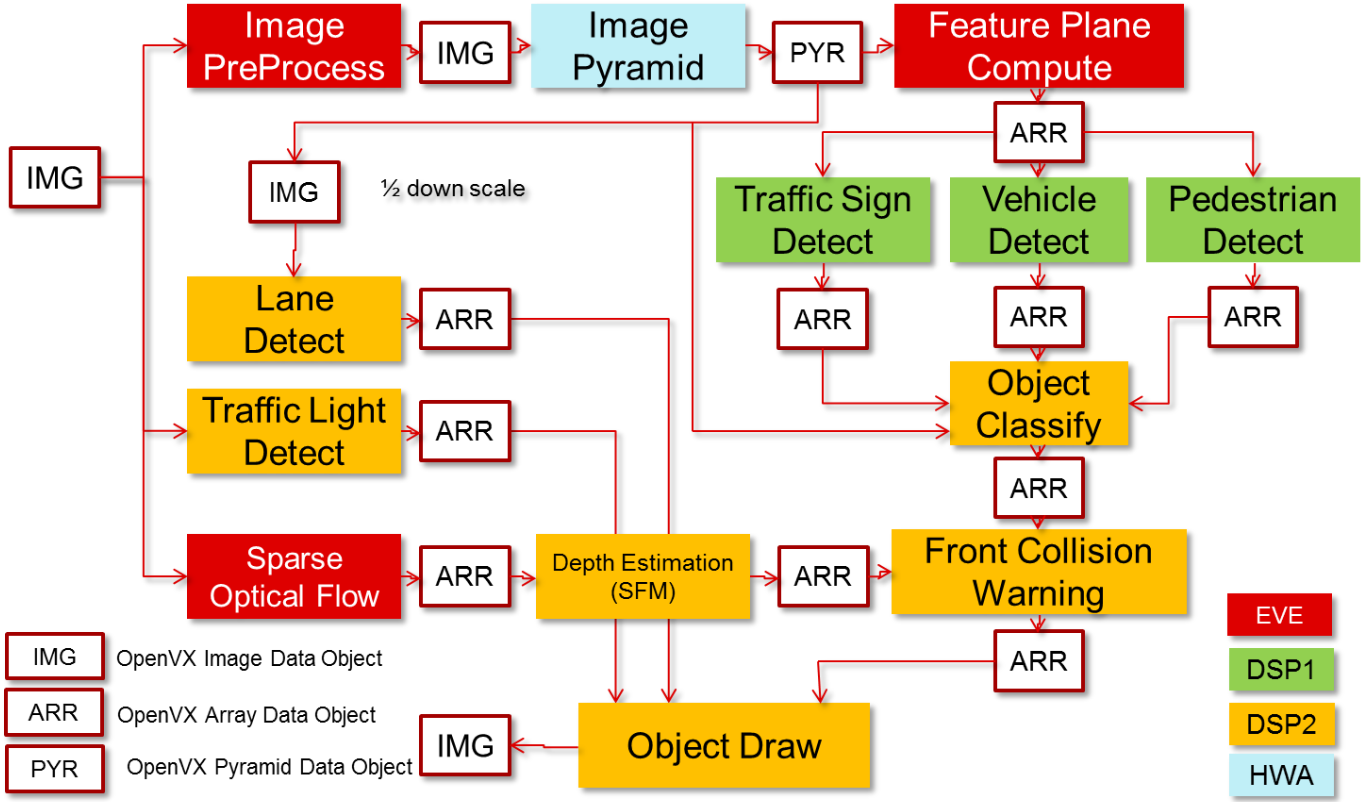


TI OpenVX on TDA3x

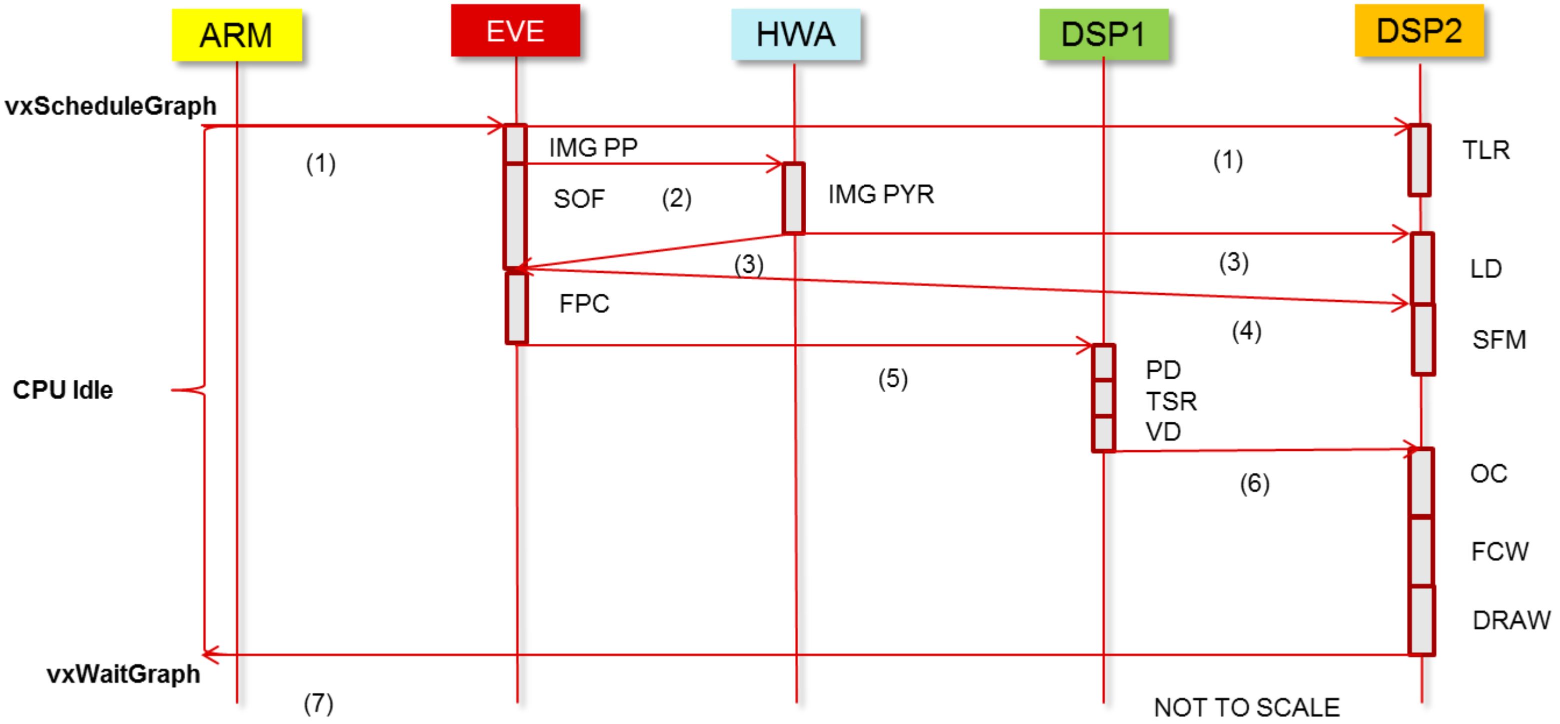


HOST	User Thread on HOST CPU, calls OpenVX APIs
Target	TI OpenVX “Target” executes vision kernels. Target can run on same CPU as HOST. Multiple Targets on a CPU possible. Target execute in parallel to each other

Mono-camera Analytics Processing Graph (Example)

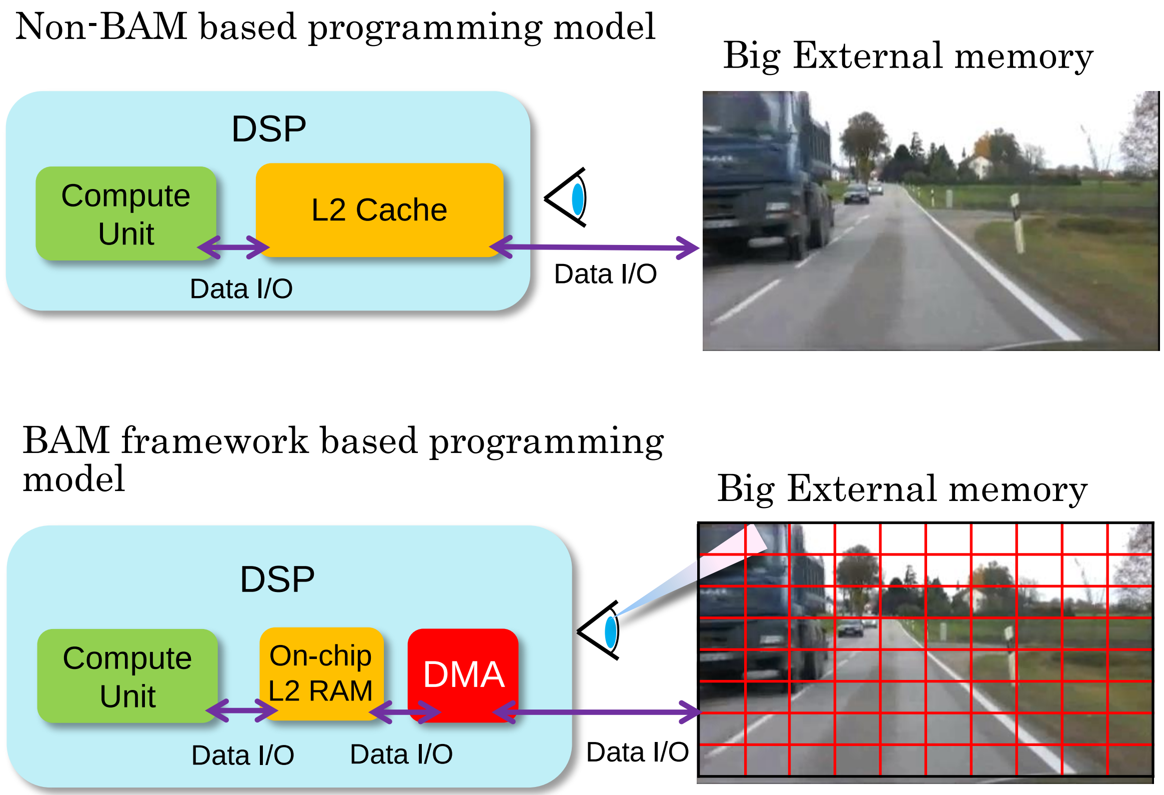


Distributed Graph Execution



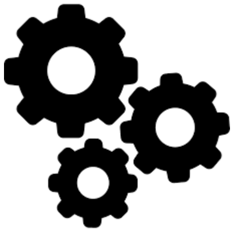
Distributed graph execution minimizes overheads at “HOST” ARM CPU and reduces system latency

Block Access Manager (BAM) Framework

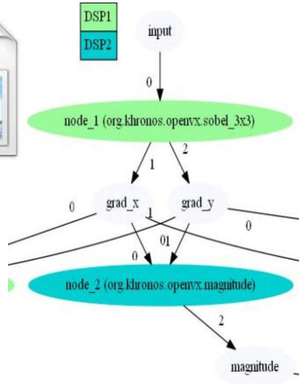


- Divides an input into smaller 2-D blocks and pipelines kernels using BAM
- BAM manages DMA, including abstracting the overlap reads required for filtering kernels.
- Reduces the input/output accesses made in the external memory
- "Virtual Image" in OpenVX is used to eliminate intermediate buffers.
- Most OpenVX v1.1 kernels optimized on DSP using BAM

PyTIOVX - Automated OpenVX “C” Code Generation



PyTIOVX

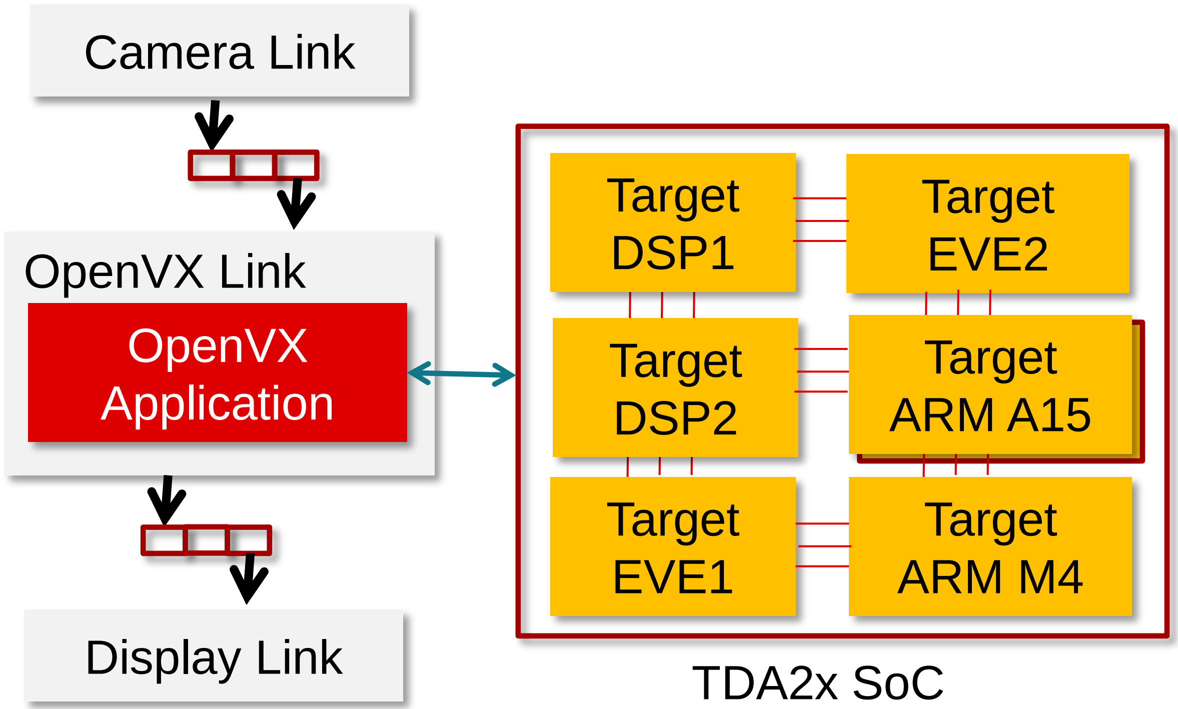



```
'''
from tiovx import *

context = Context("vx_tutorial_
graph = Graph()

width = 640
height = 480

in_image = Image(width, height,
grad_x = Image(width, height, D
grad_y = Image(width, height, D
magnitude = Image(width, height
phase = Image(width, height, Df
grad_x_img = Image(width, heigh
grad_y_img = Image(width, heigh
magnitude_img = Image(width, he
shift = Scalar(Type.INT32, 0, n

graph.add ( NodeSobel3x3(in_ima
graph.add ( NodeMagnitude(grad_
graph.add ( NodePhase(grad_x, g
graph.add ( NodeConvertDepth(ma
graph.add ( NodeConvertDepth(gr
graph.add ( NodeConvertDepth(gr

context.add ( graph )

ExportImage(context).export()
ExportCode(context).export()
```


```c
if (status == VX_SUCCESS)
{
    usecase->input = vxCreateImage(context, 640, 480, VX_DF_IMAGE_U8);
    if (usecase->input == NULL)
    {
        status = VX_ERROR_NO_RESOURCES;
    }
    vxSetReferenceName( (vx_reference)usecase->input, "input");
}
if (status == VX_SUCCESS)
{
    usecase->grad_x = vxCreateImage(context, 640, 480, VX_DF_IMAGE_S16)
    if (usecase->grad_x == NULL)
```

- Generated C code can run on SoC without modifications
- Visualize graph connections
- Trap and fix common mistakes before executing on target SoC

Pipelined Graph Execution with Processor SDK - Vision



- OpenVX used for compute
- Processor SDK – Vision used for Camera, Display, system level control
- Pipelined execution of OpenVX with camera and display improves system utilization

Summary

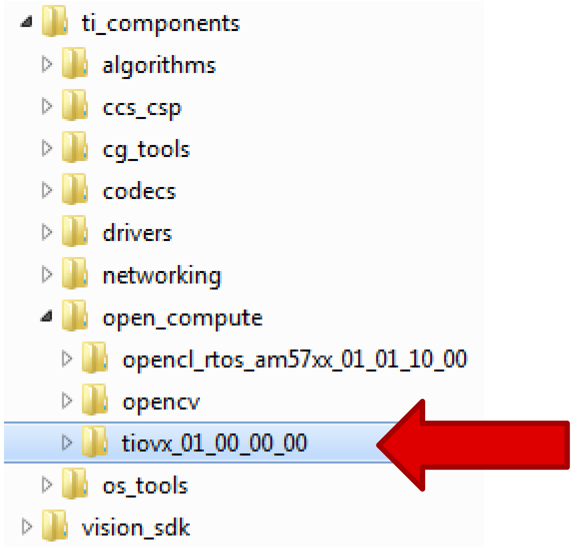
- TI OpenVX supports true multi-core heterogeneous compute on TDA2x/3x SoCs
- TI OpenVX implementation differentiates via
  - Distributed graph execution
  - DMA acceleration using BAM
  - Pipelined graph execution and streaming IO nodes (camera/display)
  - Ease of use via code generation (PyTIOVX) tool, PC emulation mode
  - Ability to run on “Big ARM” CPUs with HLOS as well as “MCU ARM” CPUs using RTOS

Agenda

- Introduction to OpenVX
- OpenVX on TI SoCs – TIOVX
- Getting Started with TIOVX

TIOVX within Processor SDK - Vision

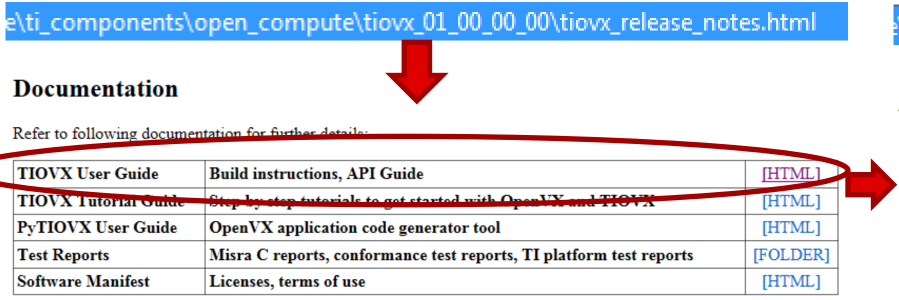
- TIOVX is present in Processor SDK – Vision at the location shown below



TIOVX package

TIOVX Getting Started

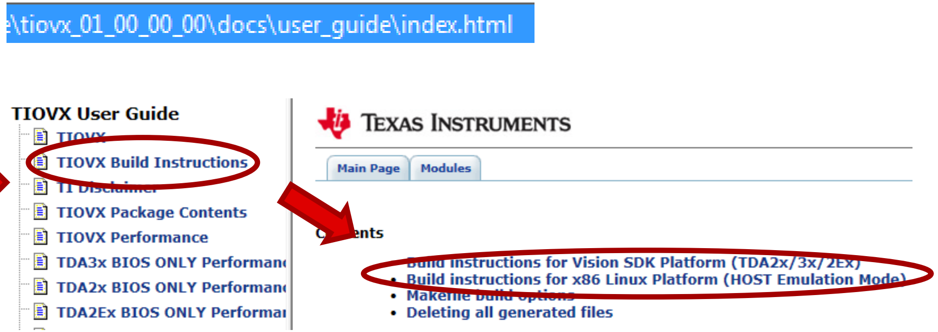
- TIOVX sample application can be run on Linux x86 PC as well as TI TDA2x/3x SoC/EVM
- Follow steps in user guide to run sample applications on Linux x86 PC or SoC/EVM


e\ti_components\open_compute\tiovx_01_00_00_00\tiovx_release_notes.html

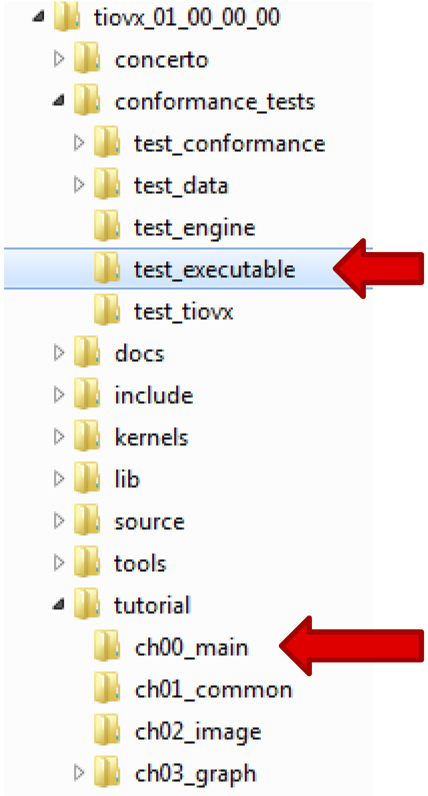
Documentation

Refer to following documentation for further details:

TIOVX User Guide	Build instructions, API Guide	[HTML]
TIOVX Tutorial Guide	Step-by-step tutorials to get started with OpenVX and TIOVX	[HTML]
PyTIOVX User Guide	OpenVX application code generator tool	[HTML]
Test Reports	Misra C reports, conformance test reports, TI platform test reports	[FOLDER]
Software Manifest	Licenses, terms of use	[HTML]



TIOVX Sample Applications



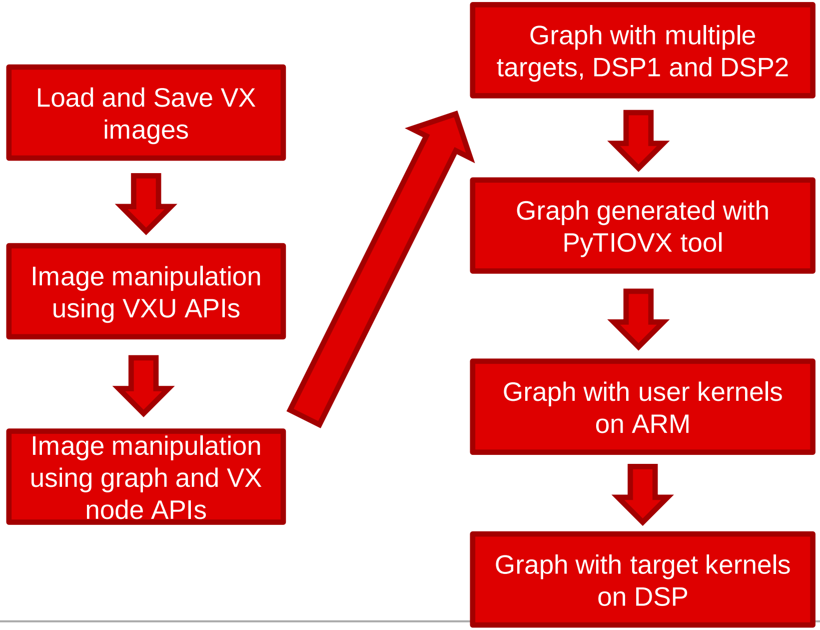
- main() for **Khronos conformance test suite**, including TI extension test suite on Linux x86 PC
- Helps confirm installation is fine and TI implementation meets OpenVX conformance

- main() for **TI OpenVX Step-by-step Tutorials** on Linux x86 PC
- **Recommended starting point to learn TI OpenVX**

TIOVX Tutorials

- Step by step examples to understand OpenVX, followed TI extensions to OpenVX including developing kernels on TI C6xx DSP

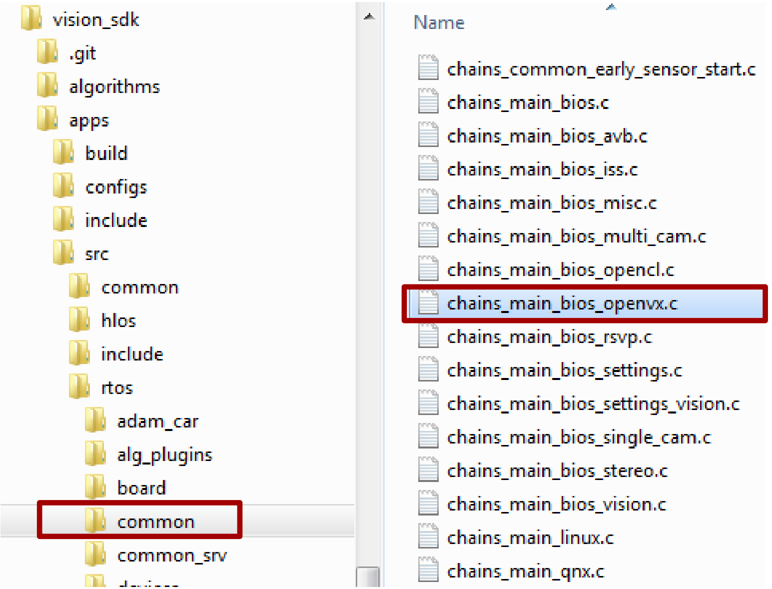
e\tiovx_01_00_00_00\docs\tutorial_guide\index.html



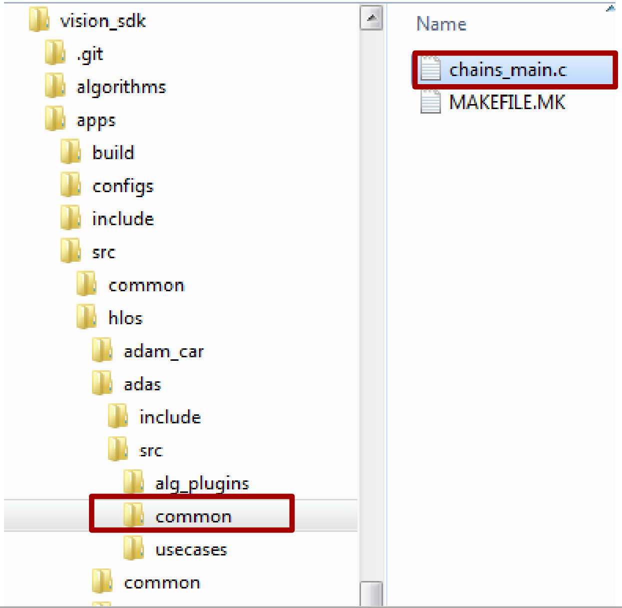
TIOVX on TI SoC/EVM

- TI OpenVX sample application entry point to run on TI SoC/EVM can be found within Processor SDK – Vision

RTOS OpenVX applications

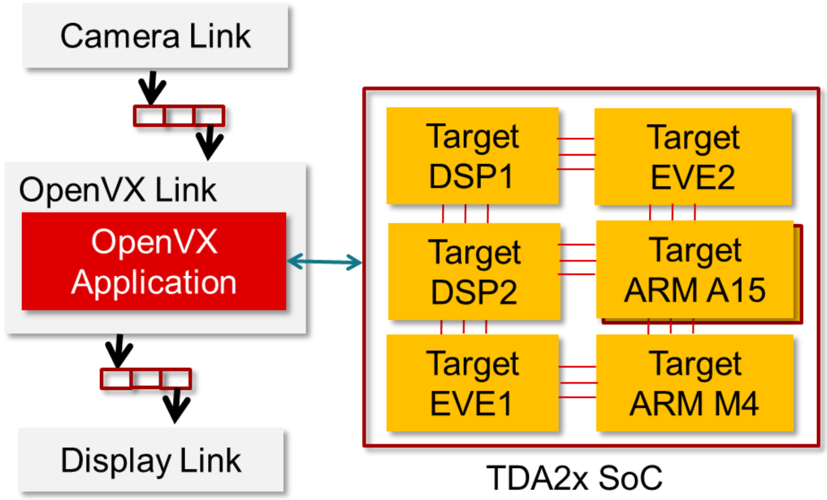


Embedded Linux OpenVX applications



TIOVX on TI SoC/EVM with Capture and Display

- A TI OpenVX sample application shows interaction of OpenVX with links framework for capture and display.



OpenVX use-case with capture and display “links”

\vision_sdk\apps\src\rtos\usecases\vip_single_cam_openvx

OpenVX “link” used in the use-case

\vision_sdk\apps\src\rtos\alg_plugins\openvx

Additional TIOVX Resources

- Release notes **– READ this first**
  - \tiovx_xx_xx_xx_xx\tiovx_release_notes.html
- User guide, tutorial guide, PyTIOVX guide
  - \tiovx_xx_xx_xx_xx\docs\user_guide\index.html
  - \tiovx_xx_xx_xx_xx\docs\tutorial_guide\index.html
  - \tiovx_xx_xx_xx_xx\docs\pytiovx_guide\index.html
- Processor SDK – Vision resources
  - \vision_sdk\docs\Index.htm
- Web resources
  - http://www.ti.com/processors/automotive-processors/tdax-adas-socs/overview.html
  - http://www.ti.com/tool/processor-sdk-vision
  - https://e2e.ti.com/support/arm/automotive_processors/f/1021

Thank You !!!