TIOVX – TI’s OpenVX Implementation

Aish Dubey
Product Marketing, Automotive Processors

Embedded Vision Summit, 3 May 2017
TI SOC platform – heterogeneous cores

High level processing
- Object detection and tracking
- Classification
- Adaboost, SVM
- KNN, ANN

Mid level processing
- Optical Flow
- Stitching
- Integral Image
- Feature Extraction (HOG, SURF, SIFT, ORB,..)
- Disparity
- Detection of Corners,
- Detection of Edges

Low level processing
- Image Signal Processing
- Filtering
- Gradients
- Morphological Operations

Functional safety compliant architecture
Security
Deep learning acceleration

Scalable s/w and h/w across sensors

Hardware for fully connected and convolution layers
General purpose processing for sensor fusion
ARM, DSP

Specialized hardware units for key sensors
- Dense optical flow
- Stereo vision
- Distortion correction
- ISP
- Radar processing and acceleration

EVE & DSP Vector Coprocessor:
- High Bandwidth
- Pixel Operations
- SIMD Parallelism
- Energy Efficiency

TDA platform
- Legacy of ASIL D safety - safe island, safe IPs
- Legacy of security to implement automotive HSM
- Hardware for deep convolution network

Functional safety compliant architecture
Security
Deep learning acceleration
GOAL: Help users easily maximize performance on TI platforms while minimizing development cost.

Graph-based model
Defined at initialization time for optimal run time and latency

True Heterogeneous Compute
Abstracted access to heterogeneous cores

Optimized Libraries
Fully optimized OpenVX 1.1 kernels

DMA Integration
DMA interface for tiled access

Virtual Buffers
Intermediate buffers in internal memories

Software Abstraction
Application works across SW platforms from Linux to TI-RTOS

Open Standard
Conformant to OpenVX v1.1

Hardware Abstraction
Application works across TDA family of SoCs

Ease of use
PC based development environment

PyTIOVX tool for graph description generates OpenVX Application code

Result: Full entitlement on TI SoCs through performance portable OpenVX interface
TIOVX supports scalable TDA SOC family for ADAS (Open VX1.1)

SOCs
- TDA2x
- TDA2Eco
- TDA3x

OpenVX Cores

OS
- TI-RTOS

ADAS applications
- Front Cam
- Multi-Camera
- Fusion
- Rear Cam
- Radar
- DM/CMS/Emerg.
Distributed TIOVX graph for front camera analytics

TIOVX unique features
Distributed execution for better utilization
Camera/Display pipelining outside Open VX using TI SDK
Data IO acceleration and tiling support through DMA and virtual buffers

Abbreviations
- FUS: Sensor Fusion
- ACT: Actuation
- PD: Pedestrian Detect
- TSR: Traffic Sign Recognition
- VD: Vehicle Detect
- IPP: Image Pre-Processing
- OF: Optical Flow
- FPC: Feature Plane Compute
- PYR: Image Pyramid
- TLR: Traffic Light Recognition
- LD: Lane Detect
- SFM: Structure From Motion
- OC: Object Classification
- FCW: Forward Collision Warning

Open VX data objects

Texas Instruments
PyTIOVX - Automated OpenVX “C” Code Generation

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

```python
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, height, he
grad_y_img = Image(width, he
magnitude_img = Image(width, he
shift = Scalar(Type.INT32, 0, n

graph.add ( NodeSobel13x3 (in_image,
graph.add ( NodeMagnitude (grad_x,
graph.add ( NodePhase (grad_x, g
graph.add ( NodeConvertDepth (mag
graph.add ( NodeConvertDepth (gr
graph.add ( NodeConvertDepth (gr

context.add ( graph )

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

• TI's OpenVX 1.1 implementation supports true multi-core heterogeneous compute using ARM, DSP, EVE and HWAs on TDAx family of SoCs

• Distributed graph execution, DMA acceleration using BAM and pipelined camera/display with TI SDK – Vision helps achieve high performance, high core utilization, low latency and low CPU overheads

• Hardware abstractions allow OpenVX implementation to run on “Big ARM” CPUs with Linux as well as “MCU ARM” CPUs using TI-RTOS

• Ease of use via PyTIOVX tool, PC emulation mode help developers quickly ramp-up to use OpenVX on TI SoCs

TIOVX for OpenVX 1.1 will be available at www.ti.com/adas soon