Scaling ADAS SoC Computation with Vision DSP-based Heterogenous Architecture
-用基于Vision DSP的异质架构扩展ADAS SoC的运算能力

Charles Qi, Sr. Design Engineering Architect
IPG System Engineering

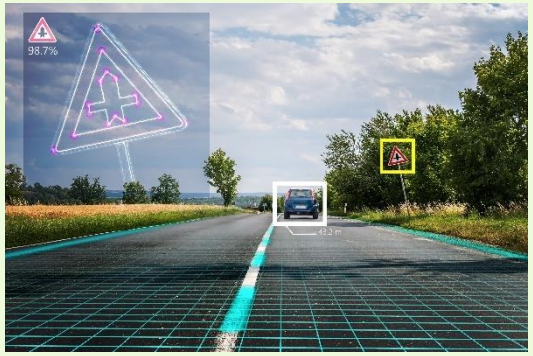
CDNLive APAC
August 2017

cadence®

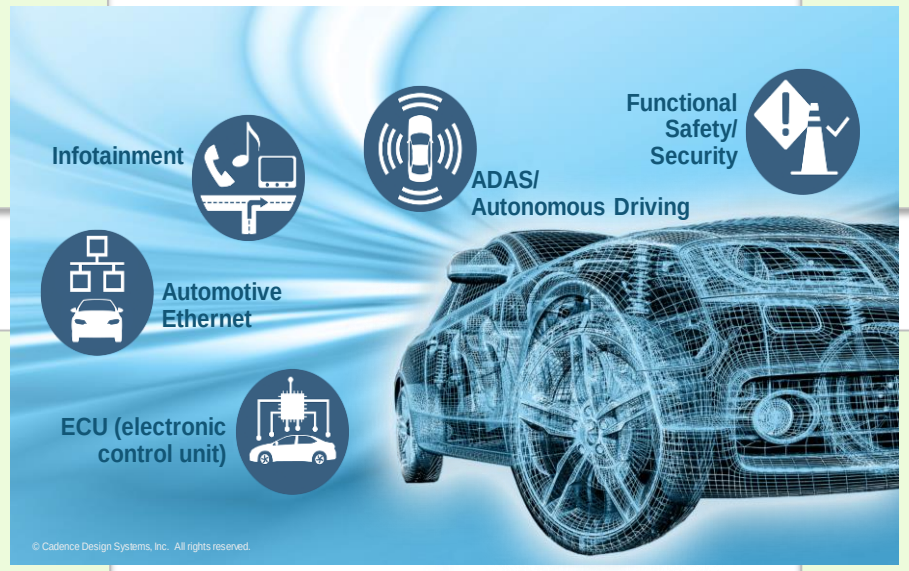
The Great Promise of Automotive

Many emerging technology opportunities for self-driving vehicles

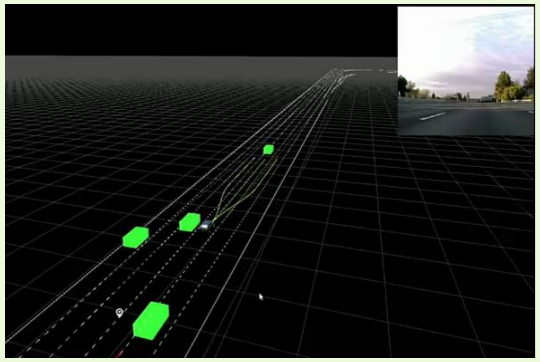
Environment Perception



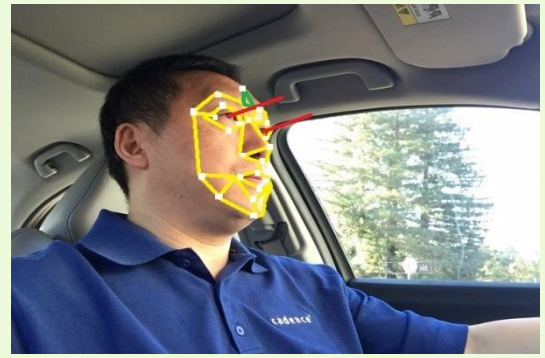
Car-2-X Communication



Map Localization & Path Planning

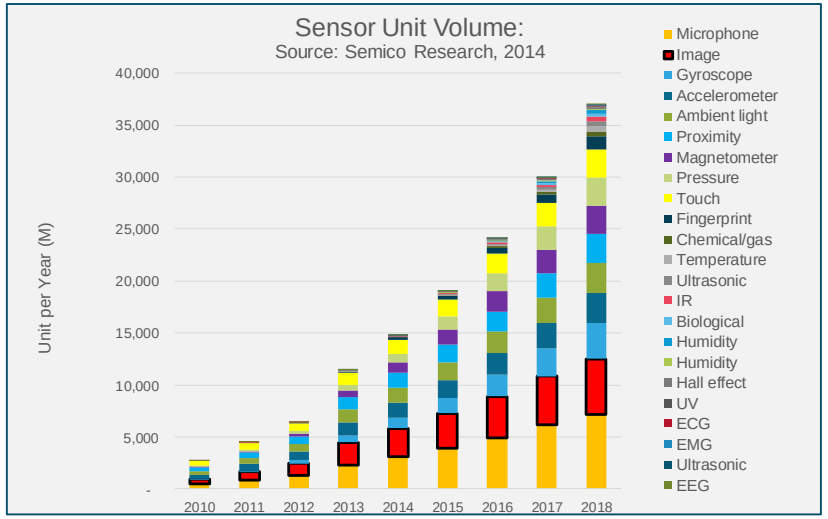


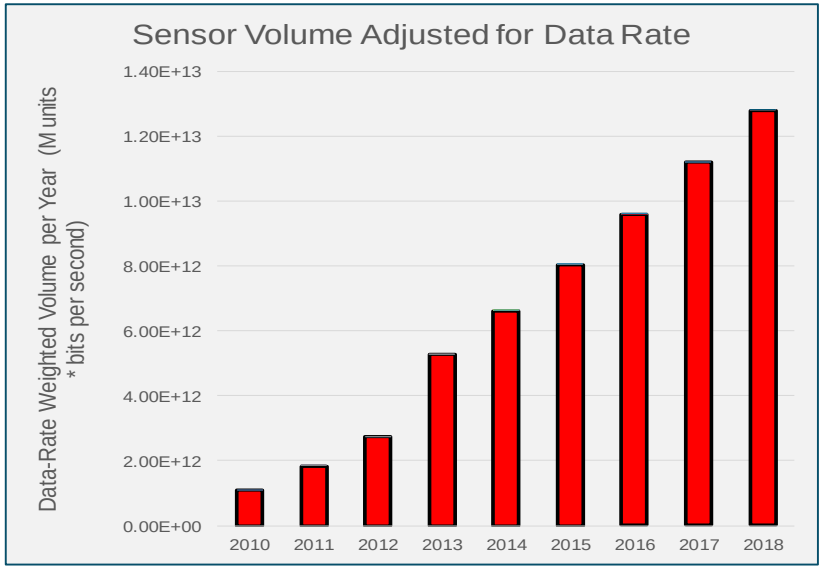
Driver Monitoring

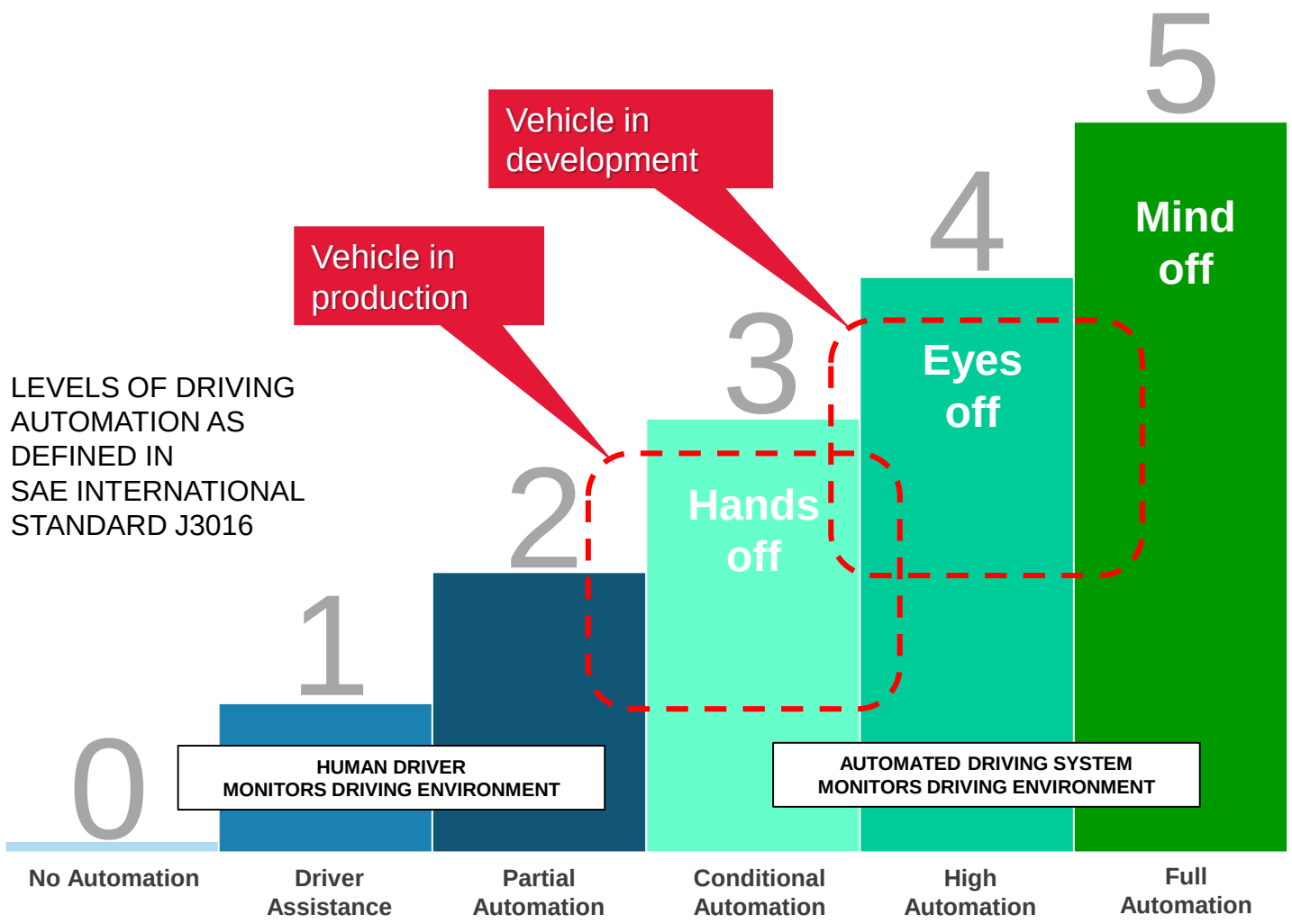


Key Driver to Technology: Increasing Autonomous Levels

Higher level: more sensors, higher resolution, higher computation

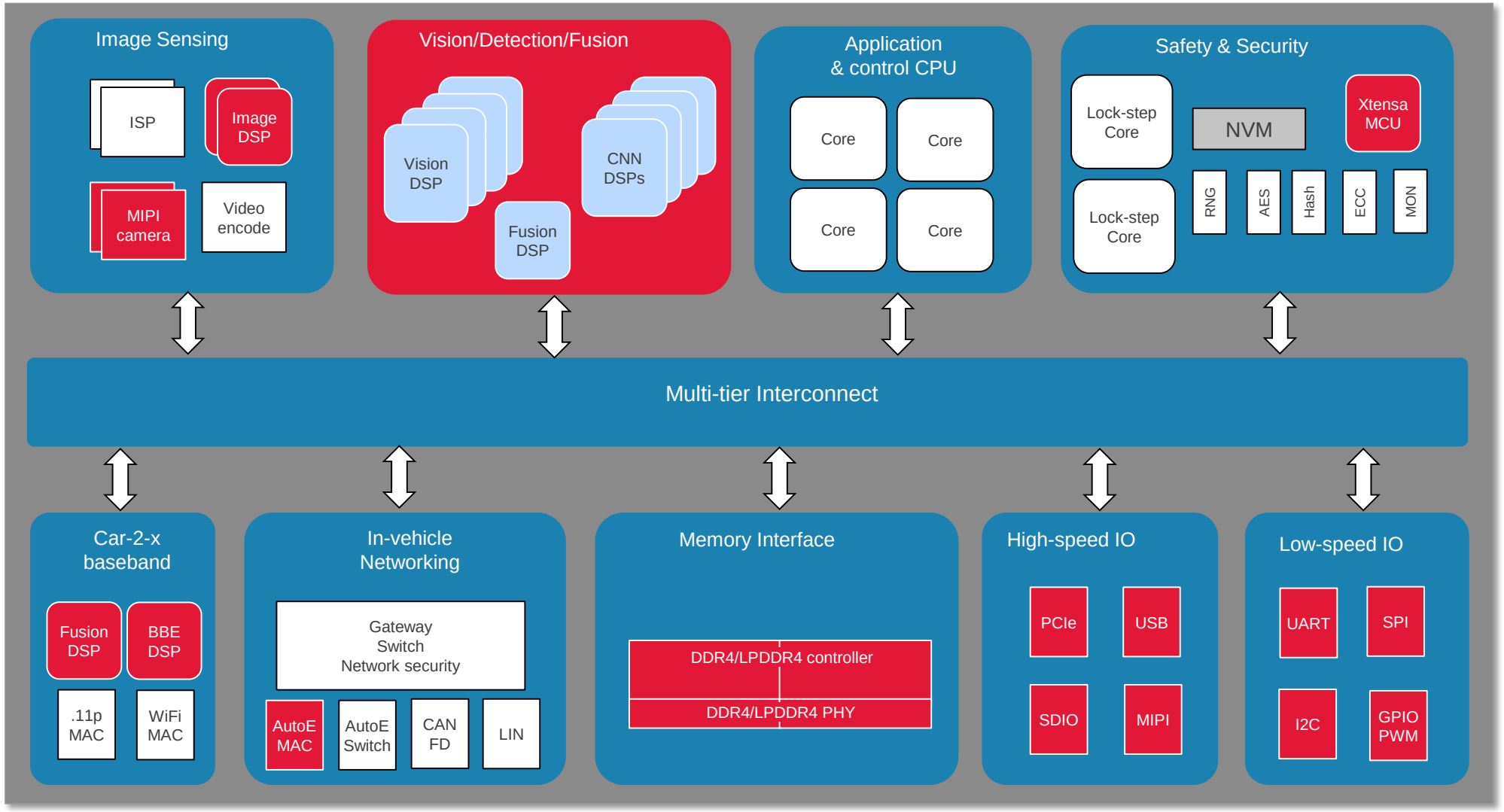






ADAS SoCs Become Super Complex Compute Platform

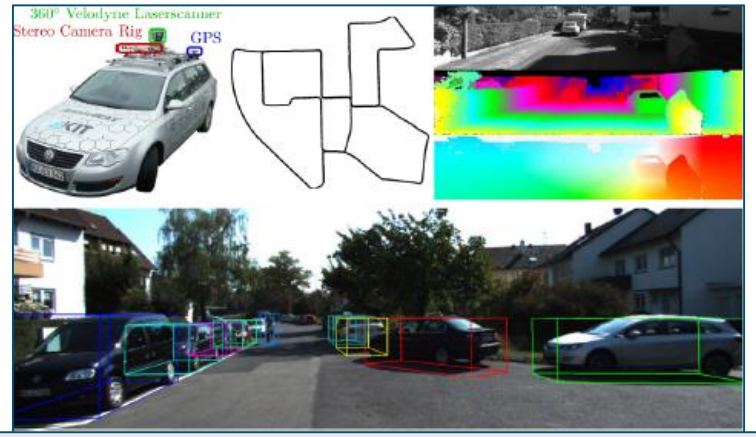
Multi subsystem, different signal processing functions, no one-size fits all



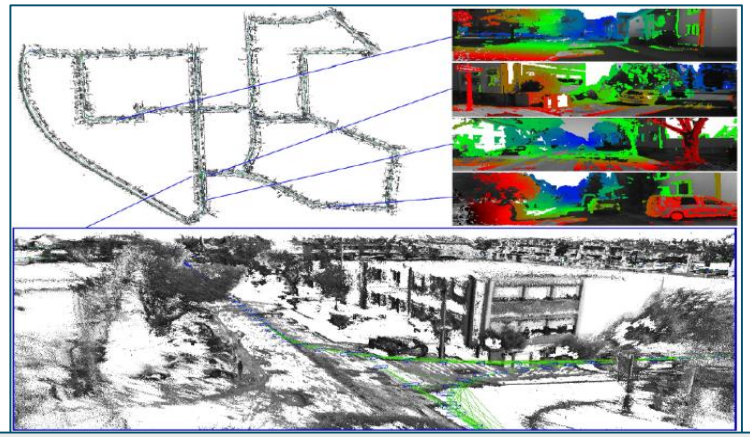
cadence

Fusion of Vision-Like Technology Dominates Self-Driving Systems

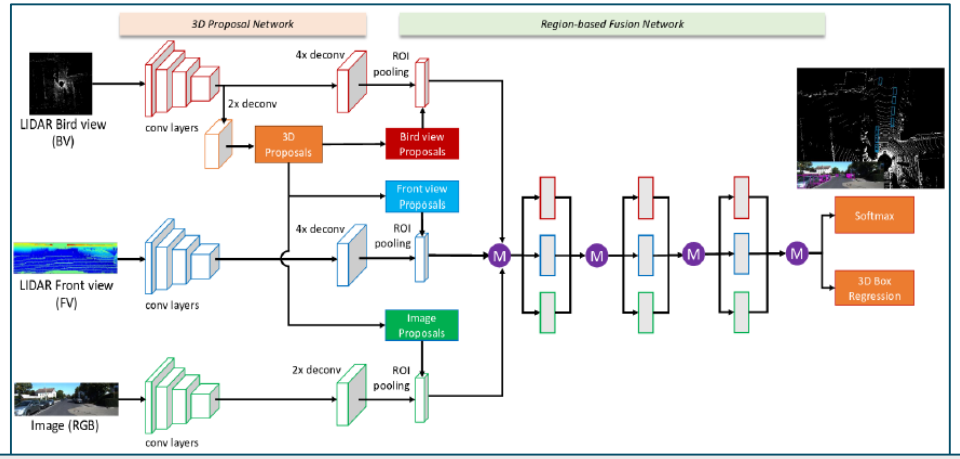
Improve accuracy in complex operating environment



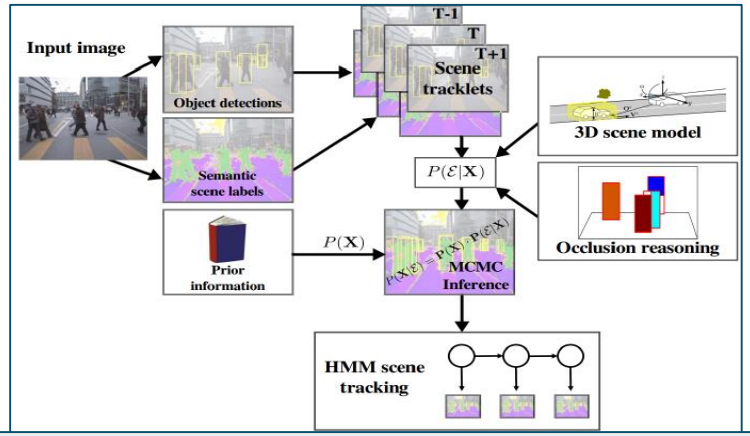
NN combined with Stereo vision disparity, optical flow for accurate object location regression (3D bonding box)



Stereo visual odometry combines stereo image, temporal multi-view image to compute real-time large scale SLAM



3D object detection combining LiDAR bird's eye view, LiDAR point cloud and Image in a deep fusion neural network

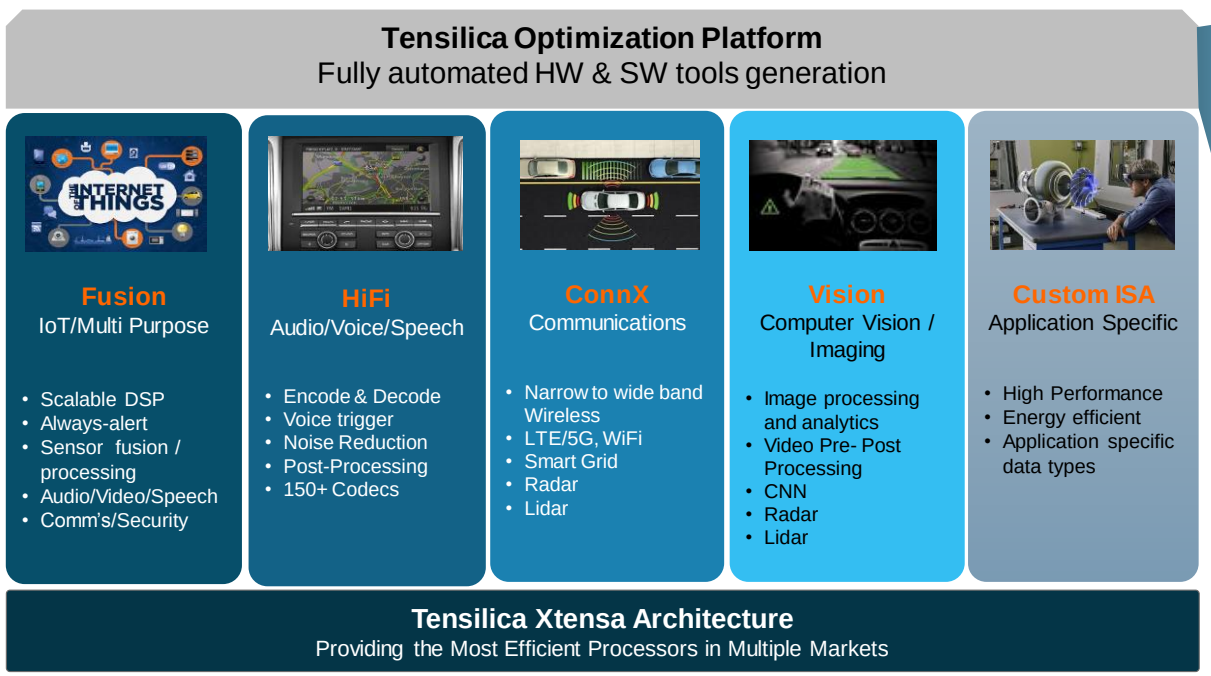


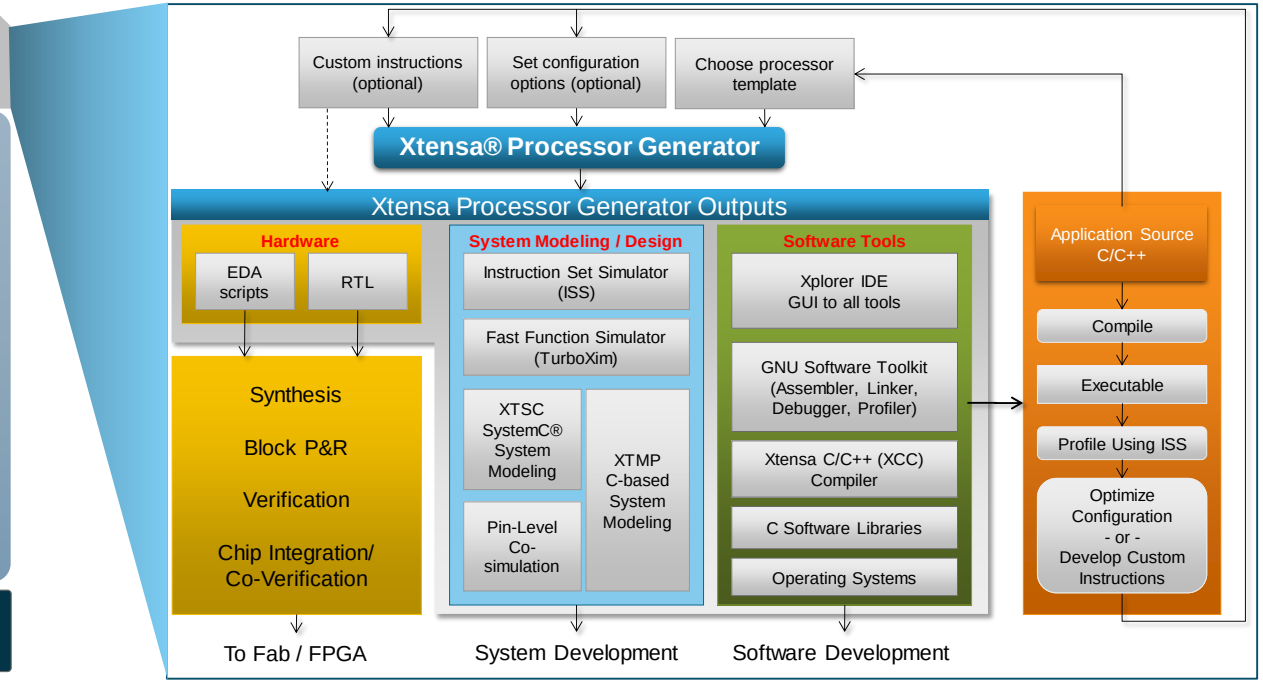
Scene understanding by combining NN objection detection, NN sematic segmentation and object model tracking

Source: courtesy of "Computer Vision for Autonomous Vehicles: Problems, Datasets and State-of-Art", J. Janai et al.

Viable Solution: Tensilica Scalable DSP Platform

DSP cores tailored for application specific needs





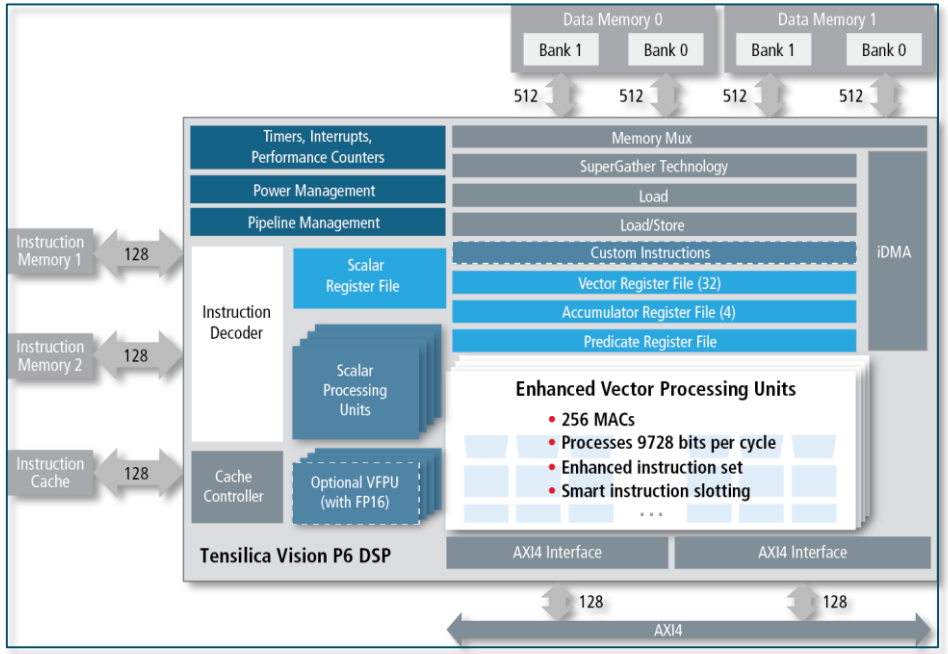
- **Scalable performance, consistent programming model**
- **Common baseline Xtensa architecture + application specific processing optimization**
- **Fully integrated HW/SW tool chain from user-friendly Xplorer IDE**

Scalable Heterogenous Architecture of Tensilica® Vision DSP

Vision P6 and C5: two latest cores customized for conventional vision and CNN

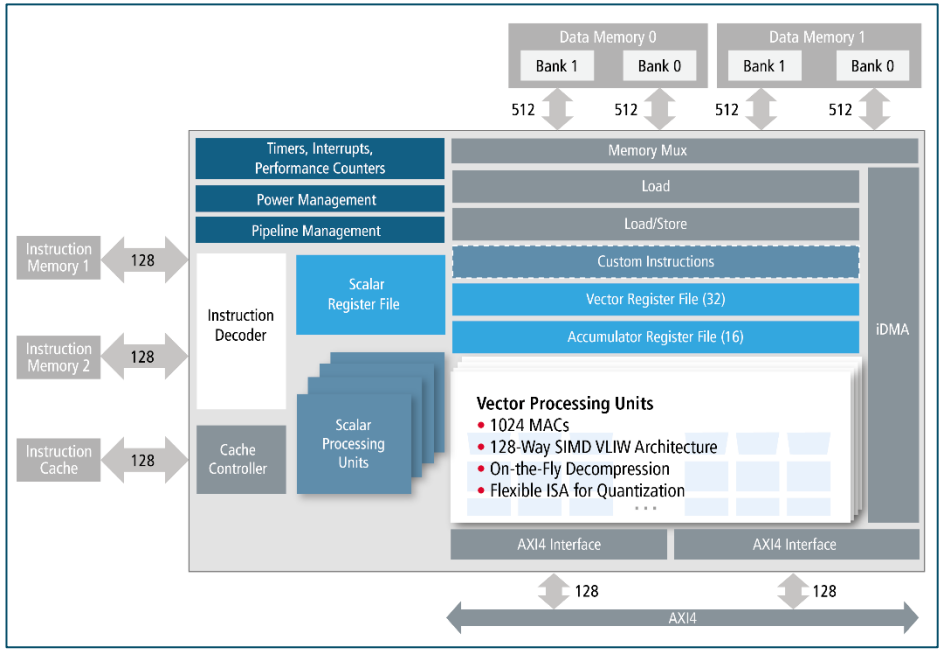
Tensilica Vision P6 DSP
– conventional vision

- **Diversified data processing pattern**
- **5 slot VLIW, 64 Way SIMD**
- **256 8-bit MAC, 4 accumulators**
- **Optional vectored FPU**
- **SuperGather scatter-gather**
- **TIE package for histogram**



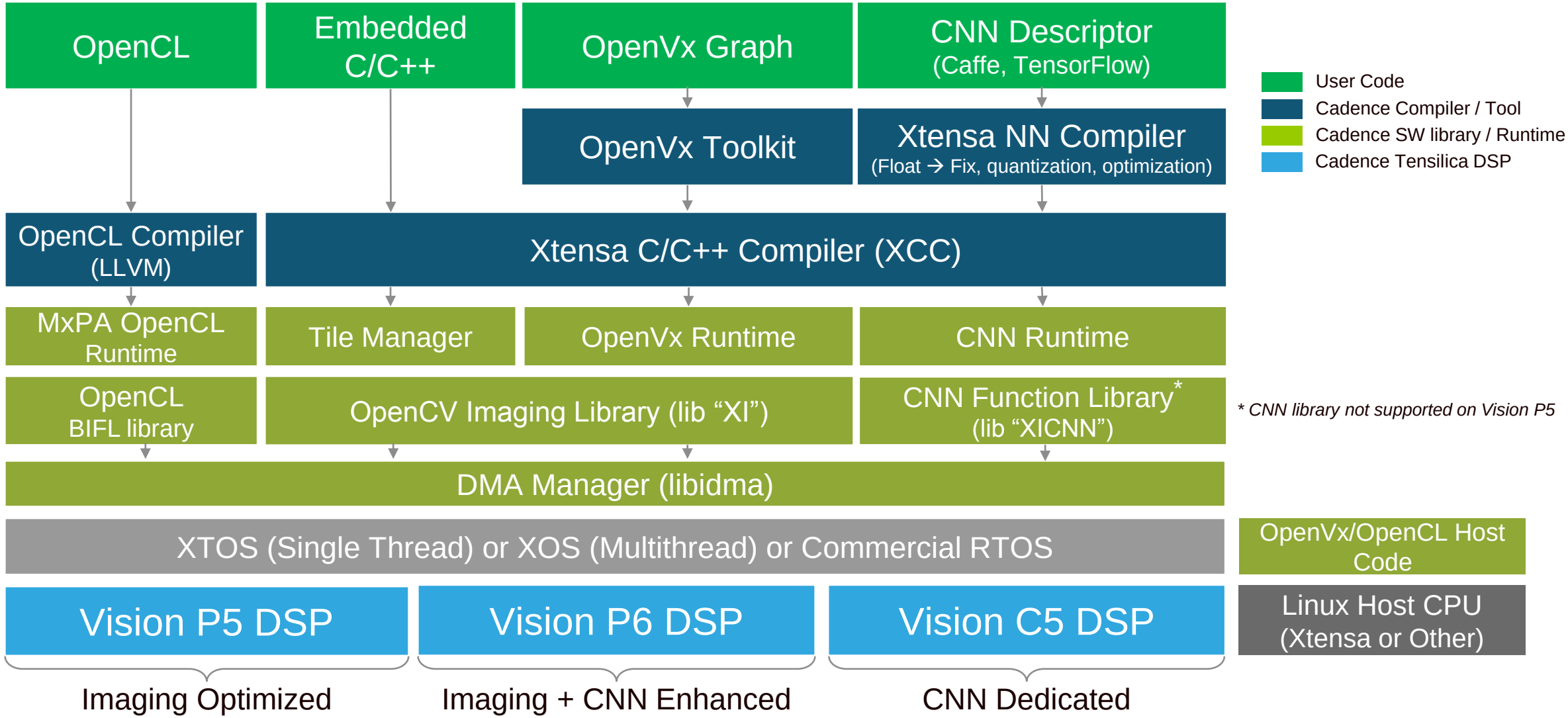
Tensilica Vision C5 DSP
– heavy duty CNN

- **MAC dominated data processing pattern**
- **4 slot VLIW, 128 Way SIMD**
- **1024 8-bit MAC, 16 accumulators**
- **Flexible data packing for convolution**
- **On-the-fly sparsity decompression**
- **Flexible ISA for quantization**



Consistent and Comprehensive Vision Software Solution

Full ecosystem of software frameworks and toolchain across common architecture



cadence

Key Vision DSP Optimization Principles for Vision Algorithms

Focus on speed-up parallelizable tasks and minimize sequential overhead

Increase computation efficiency

- ✓ ISA level parallelism based on SIMD
- ✓ Improve resource utilization with VLIW
- ✓ Specialized instructions for special tasks via TIE

Reduce memory access overhead

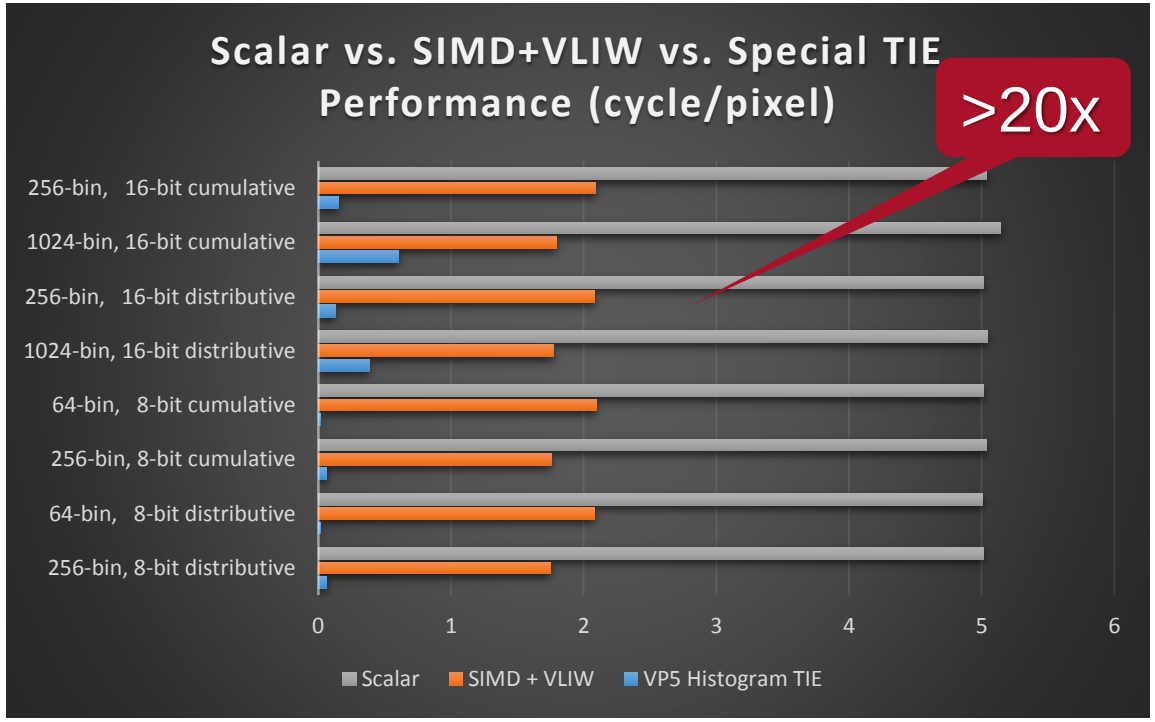
- ✓ Improve access efficiency with local RAM and tiling
- ✓ Hide access latency with DMA
- ✓ Minimize cycle count for non-contiguous access

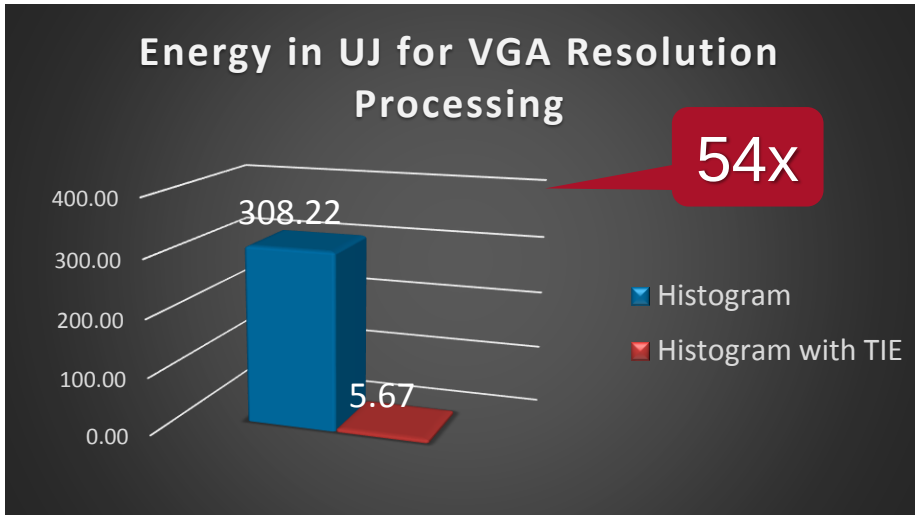
Example of Special TIE Optimization Package

Achieve significant performance while reducing power

```
//specialized TIE created for histogram processing
 for (int y = 0; y < height; ++y) {
   IVP_COUNTEQZ4NX8(vdst, sr, vec0, vec1); vhist0 = IVP_ADDNX16(vhist0, vdst); //  0 .. 31
   IVP_COUNTEQ4NX8(vdst, sr, vec0, vec1); vhist1 = IVP_ADDNX16(vhist1, vdst); //  32 .. 63
   IVP_COUNTEQ4NX8(vdst, sr, vec0, vec1); vhist2 = IVP_ADDNX16(vhist2, vdst); //  64 .. 95
   IVP_COUNTEQ4NX8(vdst, sr, vec0, vec1); vhist3 = IVP_ADDNX16(vhist3, vdst); //  96 .. 127
   IVP_COUNTEQ4NX8(vdst, sr, vec0, vec1); vhist4 = IVP_ADDNX16(vhist4, vdst); // 128 .. 159
   IVP_COUNTEQ4NX8(vdst, sr, vec0, vec1); vhist5 = IVP_ADDNX16(vhist5, vdst); // 160 .. 191
   IVP_COUNTEQ4NX8(vdst, sr, vec0, vec1); vhist6 = IVP_ADDNX16(vhist6, vdst); // 192 .. 223
   IVP_COUNTEQ4NX8(vdst, sr, vec0, vec1); vhist7 = IVP_ADDNX16(vhist7, vdst); // 224 .. 255
}
```

Specialized TIE for histogram





	Energy for VGA Res (uJ)*
Histogram	308.22
Histogram with TIE	5.67

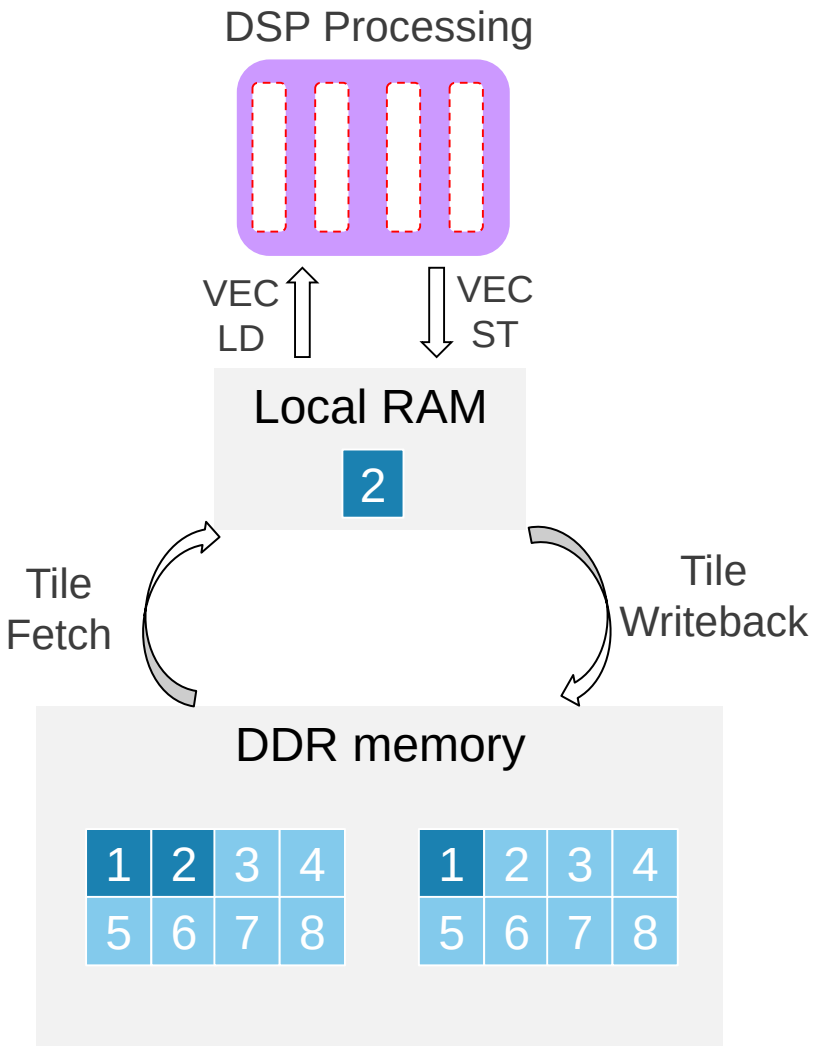
* TSCM 28nm HPM 9 track, post layout energy with 600 MHz netlist

cadence

Optimize Memory Access with Tiling

Minimize frequent access to DDR

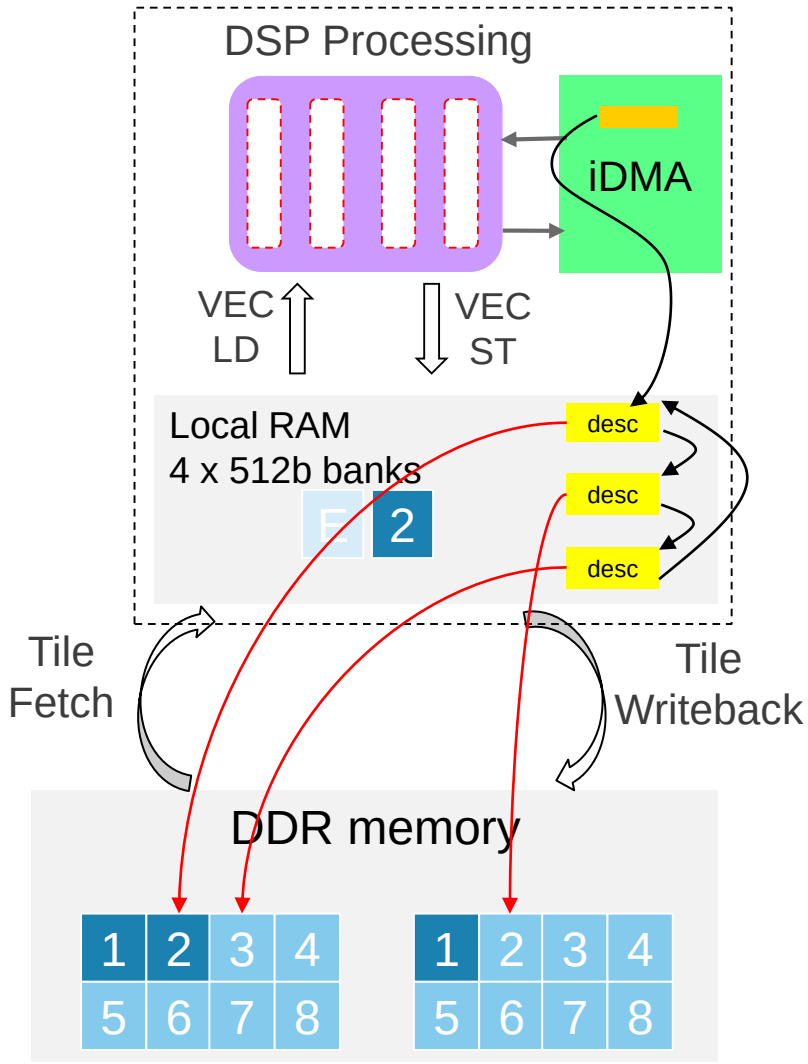
- Large disparity of latency/power between local RAM and DDR
- Poor locality in image data makes data caching ineffective
- Predictability allows programmed directed pre-fetch
- High efficiency access to local RAM
- Apply maximum processing steps to tile in local RAM
- Tile loop outside of processing steps, intermediate result not written back to DDR



Hiding Memory Access Latency via 2-D DMA

Offload tile fetch and writeback

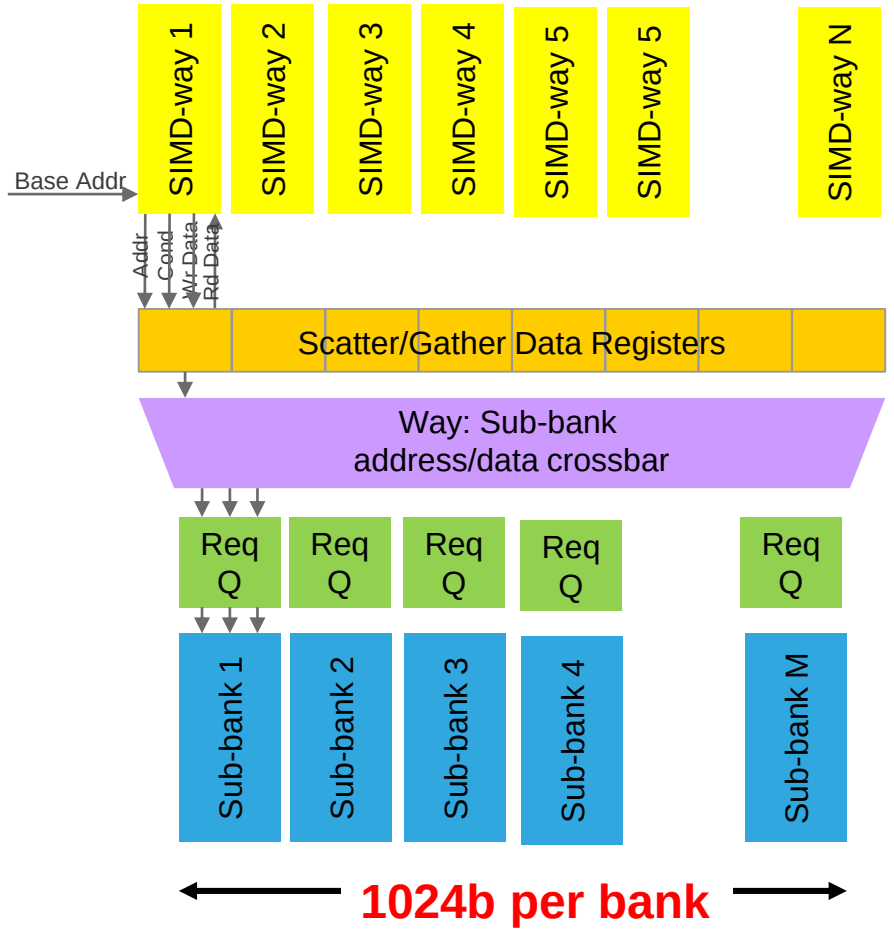
- Tile fetch and writeback by software is inefficient, blocking data processing
- Build-in DMA performs data moving in parallel
  - DMA access to local memory at up to 512b per cycle
  - Dedicated NoC master port for DMA
  - 48 outstanding requests in pipe to memory
- 2D descriptor ring supports complex memory access patterns
  - DMA is aware of the row-dominant image storage format by skipping row pitch at the end of each row
  - 3D/4D tile stream can be composed using multiple descriptors
  - Flexible data alignment
- DMA register-mapped inside processor to minimize initiation/termination overhead
- Ping-pong tile buffers in local RAM removes dependency



Scatter-Gather Efficient Load/Store for Non-Contiguous Data

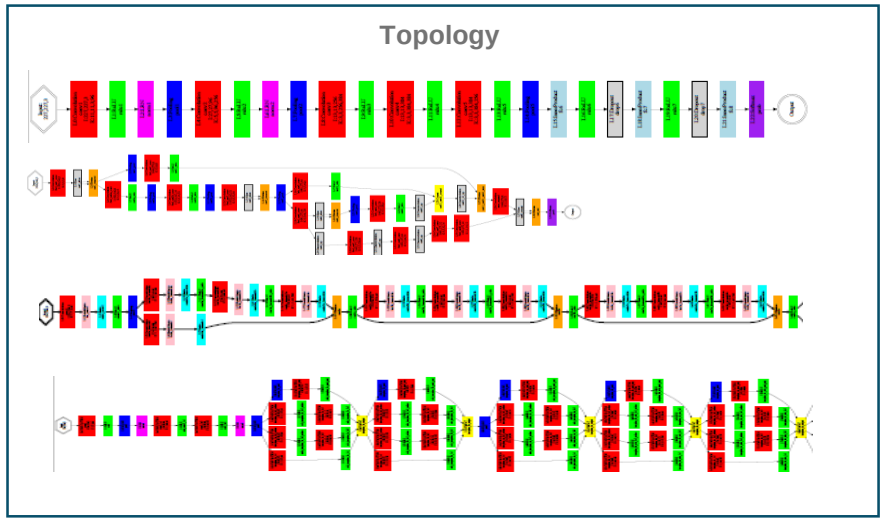
"SuperGather" sustains vectorized parallel processing

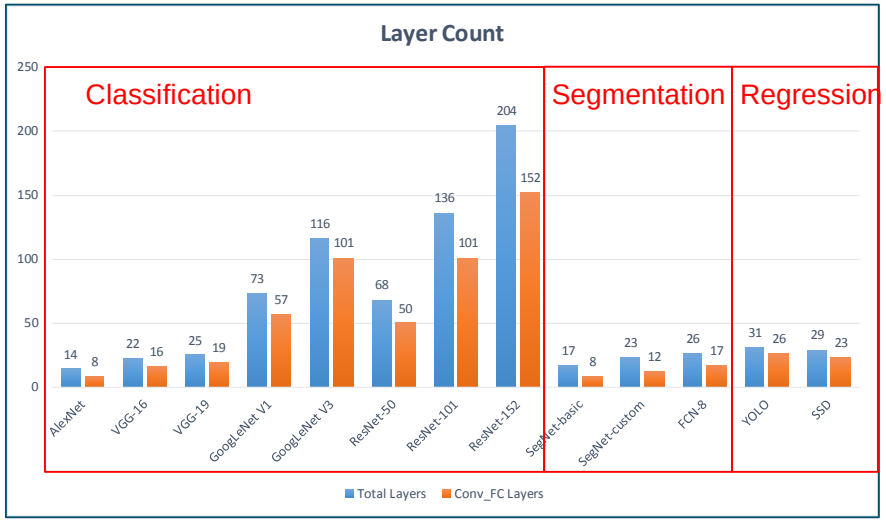
- Reads and writes many non-contiguous address in parallel across SIMD ways
- Independent request queues for sub-banks within 1024b wide data RAM bank
- Overlap between subsequent requests cuts average queue time
- Split-transaction read: 1-32 cycle access is non-blocking until data needed by dependent ops
- Configurable sub-bank width tunes throughput
- Dramatically improves non-uniform access algorithms: image warping, edge tracing, non-rectilinear patch access

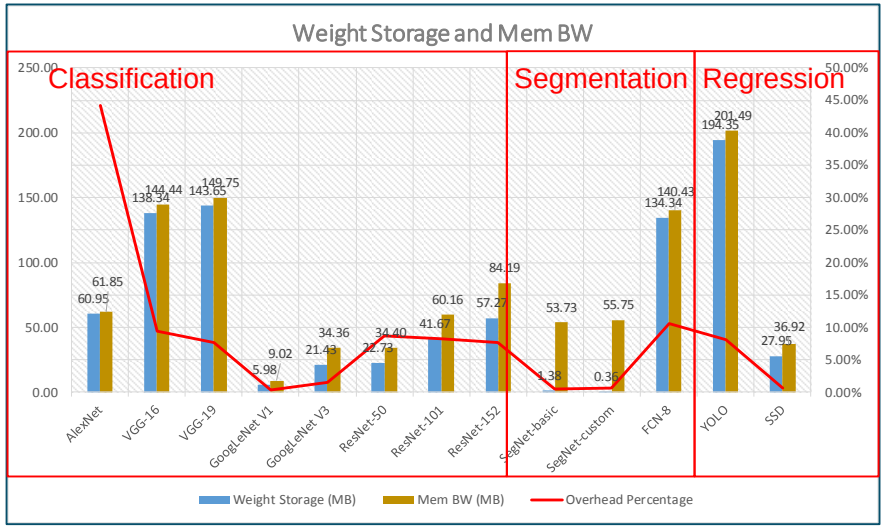


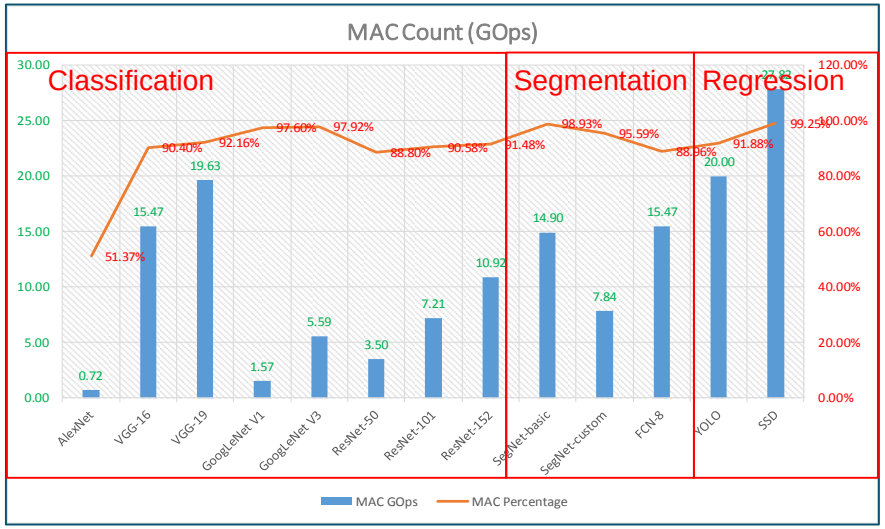
Diversity of Neural Networks in Automotive ADAS

Net topology, layer count, feature map and kernel sizes differ significantly







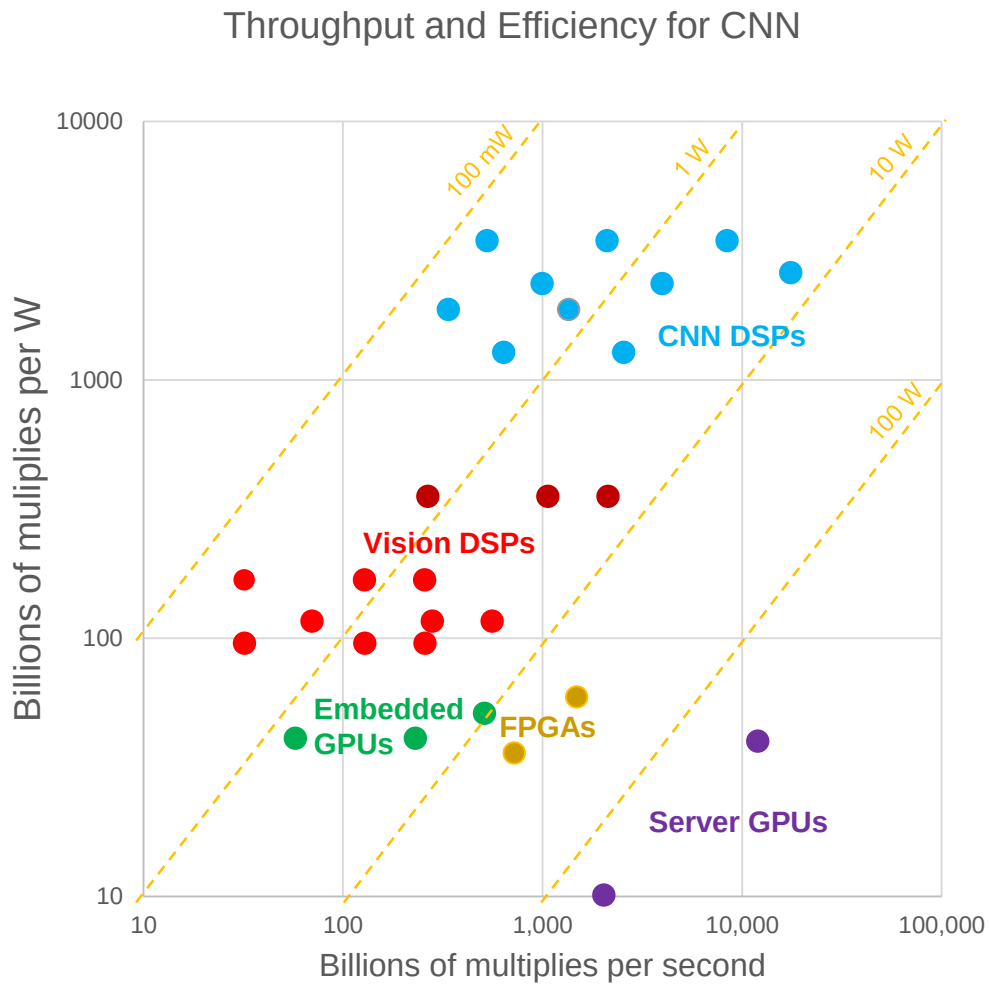


cadence

Architecture Choices of Accelerating Neural Network

Vision C5 NN DSP achieve better Perf/Watt for embedded NN

- **Performance** — 1TMAC Performance @1GHz
- **Flexibility** — Full DSP programming capability for all NN layers
- **Power Efficiency** — Custom TIE to achieve high HW utilization
- **Safety** — ISO26262 ready design methodology



cadence

Comparison Against NN Hardware Accelerators

Vision C5 Optimize the Trade-off of Performance vs. Flexibility

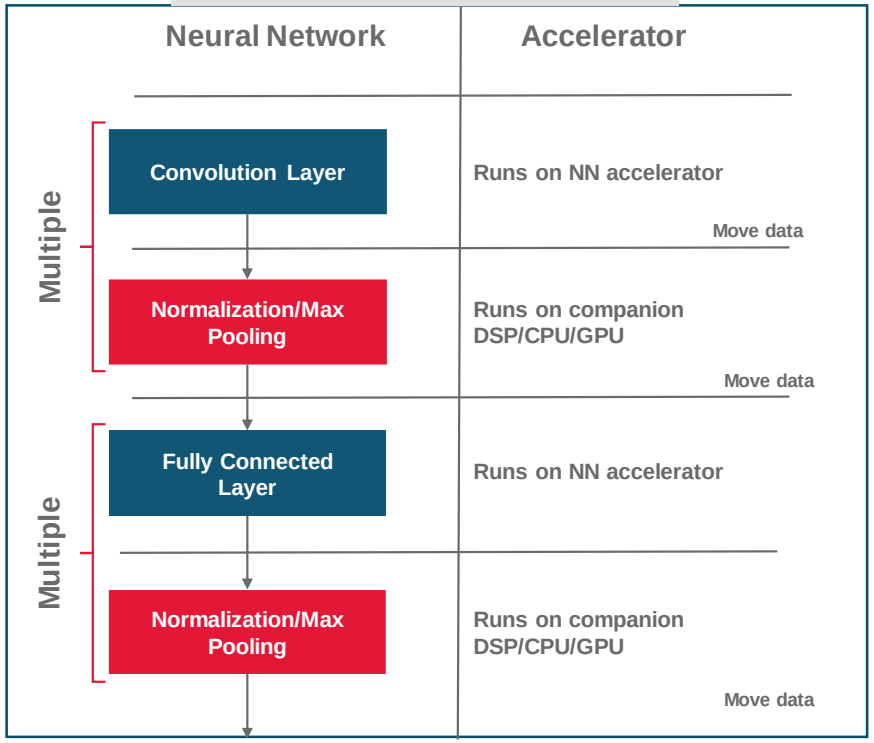
Hardware Accelerator

- Typically relies on an host CPU for task scheduling
- Typically only offload conv/FC layers
- High overhead in data exchange with CPU
- May or may not have built-in DMA

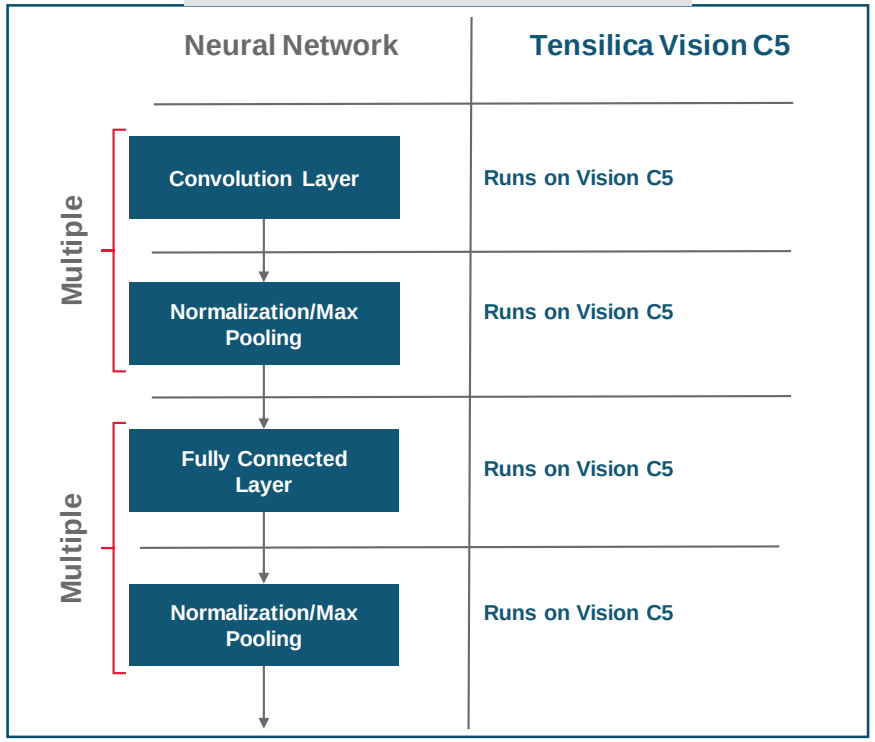
Vision C5 DSP

- Self-contained to support all NN layers
- Seamless speed-up parallelizable software code
- Flexible for various data/kernel/topology
- Easier to scale to multi-core

Accelerator Operating Model



Vision C5 Operating Model



cadence

Vision C5 DSP ISA Designed for Neural Network Processing

Uniquely designed ISA to strive for processing efficiency

- 1024 8x8, 512 16x16 multiple-accumulate operations per cycle
- Vector-by-vector, vector-by-scalar operations with multiple accumulators
- Flexible multi-row data selection and SIMD folding to maximize HW utilization
- Special addressing for load/store 3D data without nested loop
- Acceleration for pooling, nonlinear activation layers
- On-the-fly Coefficient Compression/Decompression

Vision C5 DSP Performance vs Commercially Available GPUs

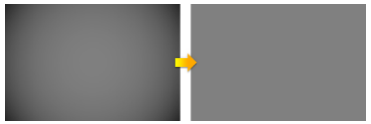
AlexNet Performance up to 6x* faster

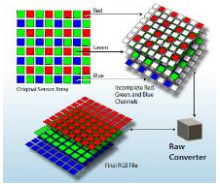
Inception V3 Performance up to 9x** faster

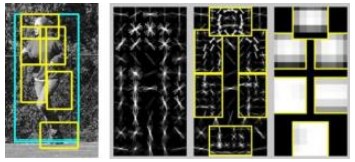
Note:
Both cores running at 690MHz on 16nm FinFET
* AlexNet data with 8 batch
** Inception V3 data with single batch

Vision DSP Reference Usage Model in ADAS SoC

Support and augment various processing functions in image perception pipeline







Sensor

HW

Lens Shading
Defect Correction
AF

HW

Demosaic
RGB
Denoise
Tone map

HW

Color
Conversion
Dewarping
Scaling

VP5/6

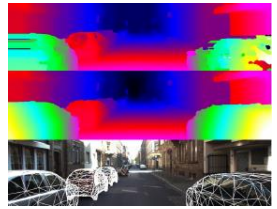
2D/3D Noise
Reduction
Image Stabilization
HDR/WDR
Super resolution

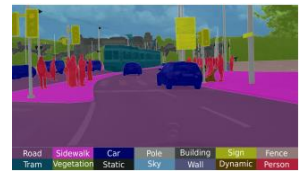
VP6

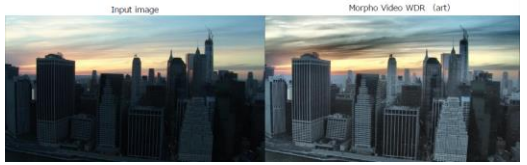
Object detection
Stereo disparity
Optical flow

VC5

Hi-resolution object
detection
Classification
Sematic segmentation



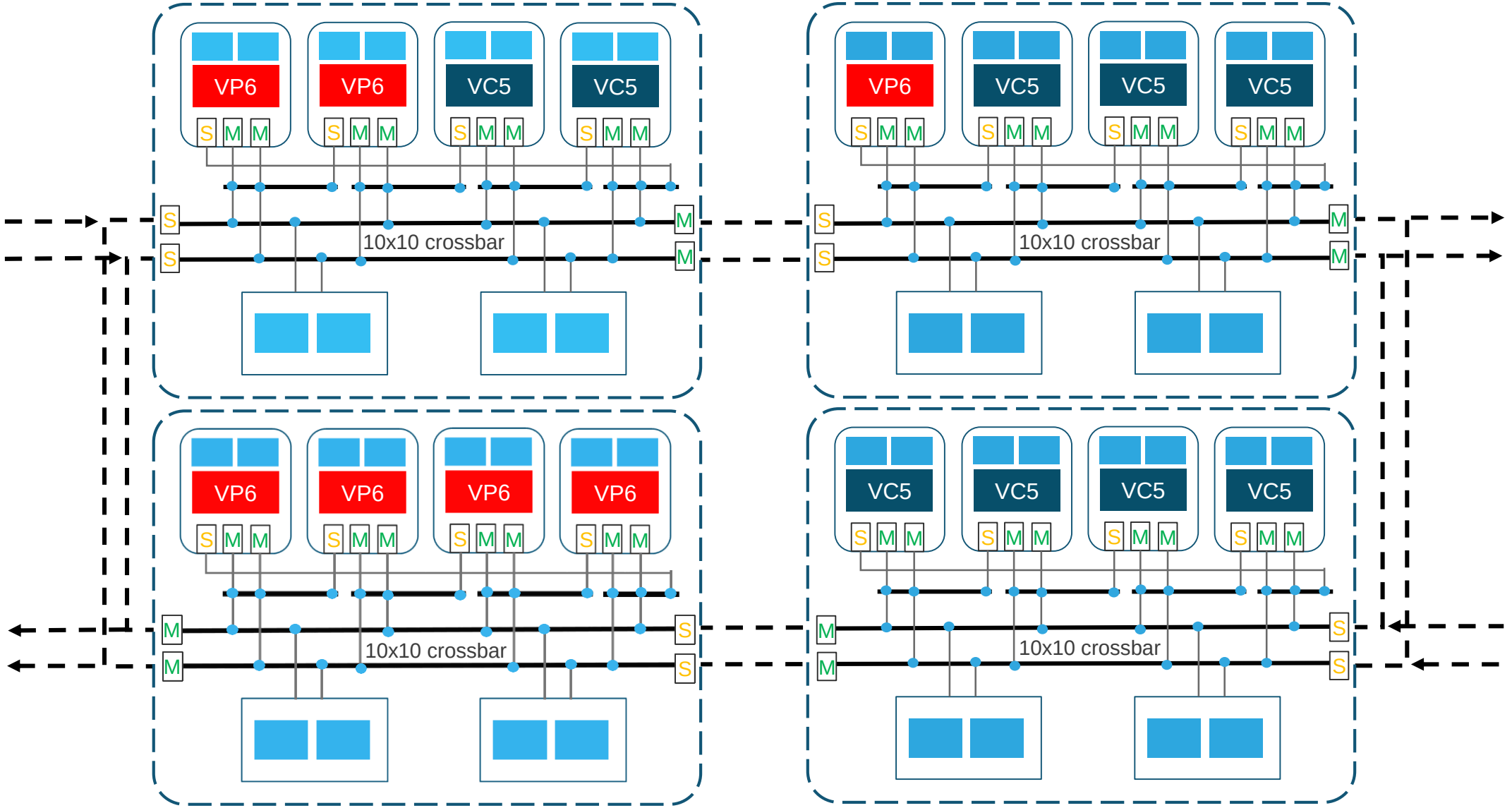




cadence

Scale Vision Sub-System Heterogeous Multi-core

Flexibility to customize MP cluster under the same programming model



cadence

Multi-Core NN Load Partition Example

Split load across layer/batch/kernel

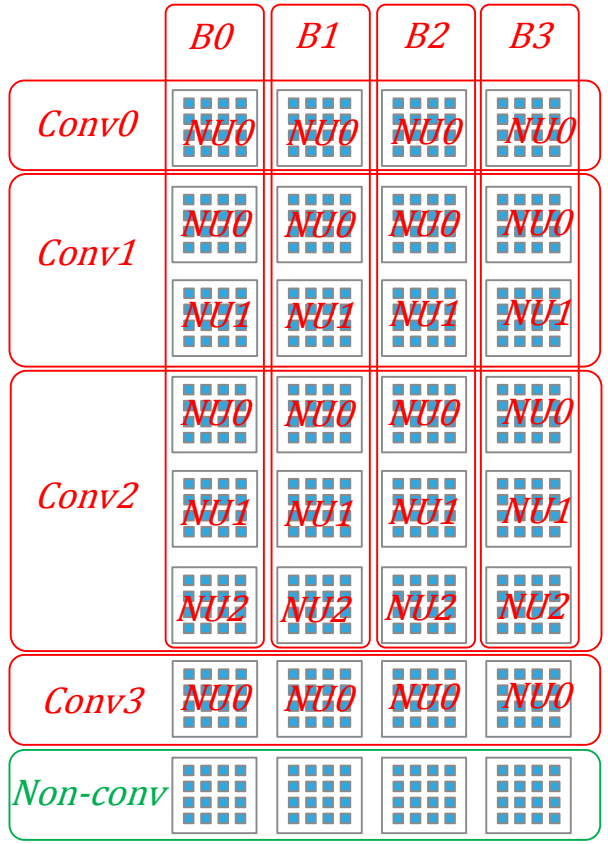
Operation:

$$F_0(x,y,n) = \sum_{z=0}^{Z-1} \sum_{r=-R}^{R} \sum_{r=-R}^{R} W(r,r,z) * FI(x+r, y+r, z) \;\; \forall x \in X, y \in Y, n \in N$$

Implementation as nested loops:

```
layers          for (l=0; l<L; l++)
                  if(layers[l] == CONV) {
Ifmap batch         for (b=0; b<B; b++)
kernels               for (nu=0; nu<NU; nu++)
                        for (nv=0; nv<NV; nv++)
Ifmap height              for (y=0; y<Y; y+=S)
Ifmap width                 for (x=0; x<X; x+=S)
Ifmap channels                for (z=0; z<Z; z++)
Kernel width                    for (ky=0; ky<KY; ky++)
Kernel height                     for (kx=0; kx<KX; kx++)
                                    F0(x,y,n) += Wl_b,n(kx,ky,z) * FI(x+kx, y+ky, z)
                  } #end if
```

SIMD vectorization in NV or X



- L and B loops are distributed across MP cores
- N is split into two loops, NU, NV and N = NU*NV
  - NU is distributed across cores
  - NV is handled by vectored SIMD

Cores are designated to certain layers and batches. Most efficient if layers can be load-balanced, no overlap in weight coefficients across cores.

Summary

Tensilica® DSP solution brings optimal performance and scalability

Meeting Demand for Future ADAS Computing

- Family of products addressing diversified ADAS processing needs
- Mature libraries, toolchain and middleware framework
- Consistent programming model based on common architecture

Optimizing Vision and Neural Network Processing

- Customized instruction to achieve computation efficiency
- High utilization of hardware resource without sacrificing flexibility
- Efficient memory addressing and DMA schemes

Delivering heterogenous Multi-core Scalability

- Self-contained core with standard HW/SW interfaces
- Cluster of cores can be individually tailored
- Flexible load distribution model for NN layers