Tensilica Vision DSP Introduction

Imaging/Vision BU
7th Dec 2017

CONFIDENTIAL

CNN Algorithm Development Trends

Increasing Computational Requirements

(~16X in <4 years)

- AlexNet (2012)
- Inception (2015)
- ResNet (2015)

NETWORK	MACS/IMAGE
ALEXNET	724,406,816
INCEPTION V3	5,713,232,480
RESNET-101	7,570,194,432
RESNET-152	11,282,415,616

Network Architectures Changing Regularly

- AlexNet (bigger convolution); Inception V3 and ResNet (smaller convolution)
- Linear network vs. branch

New Applications and Markets

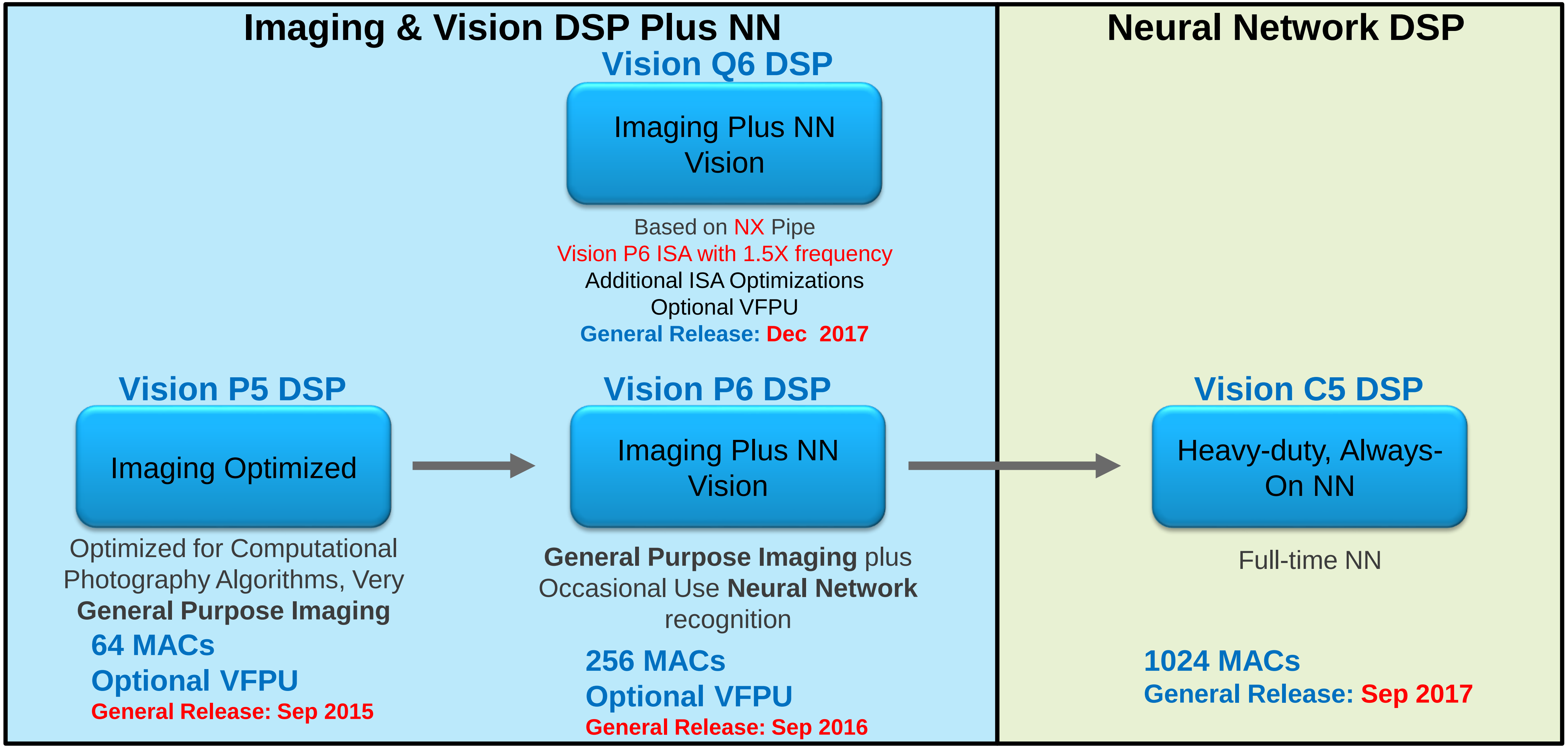
- Automotive, server, home (voice-activated digital assistants), mobile, surveillance

Lower Power

How do you pick an inference hardware platform today (2017) for a product shipping in 2019-2020+? How do you achieve low-power efficiency yet be flexible?

CONFIDENTIAL

Vision DSPs: Family of Imaging and NN DSPs



Imaging & Vision DSP Plus NN

Vision Q6 DSP

Imaging Plus NN Vision

Based on NX Pipe
Vision P6 ISA with 1.5X frequency
Additional ISA Optimizations
Optional VFPU
General Release: Dec 2017

Vision P5 DSP

Imaging Optimized

Optimized for Computational Photography Algorithms, Very **General Purpose Imaging**

64 MACs
Optional VFPU
General Release: Sep 2015

Vision P6 DSP

Imaging Plus NN Vision

General Purpose Imaging plus Occasional Use **Neural Network** recognition

256 MACs
Optional VFPU
General Release: Sep 2016

Neural Network DSP

Vision C5 DSP

Heavy-duty, Always-On NN

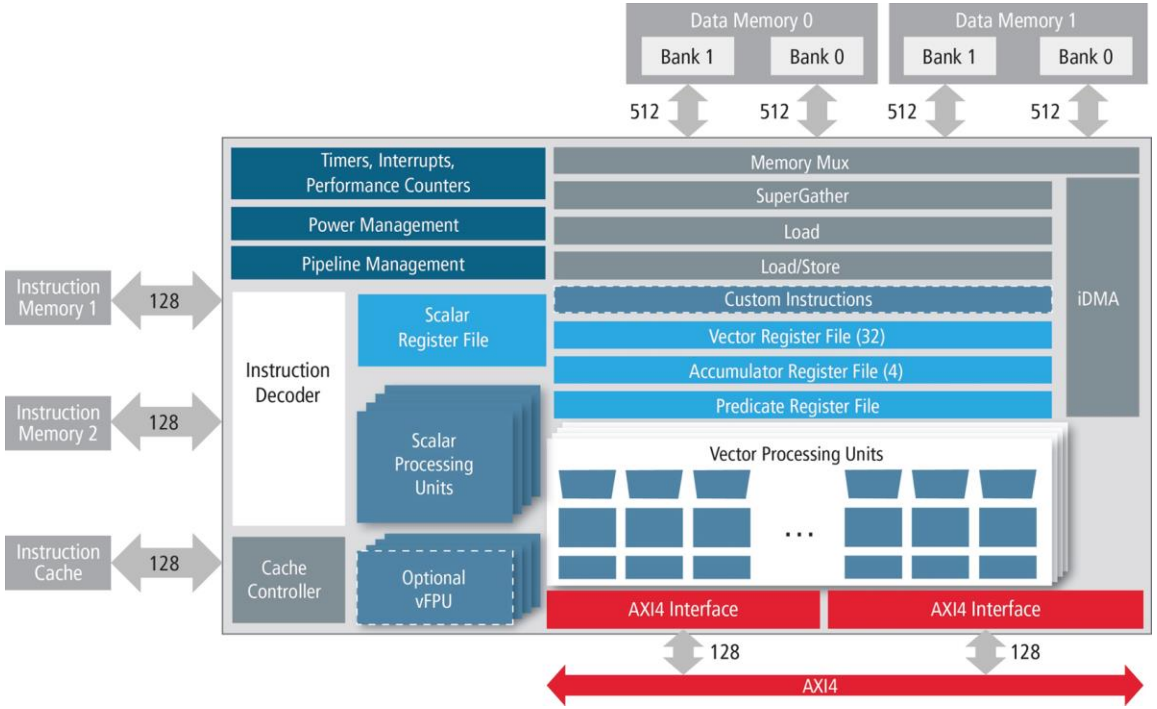
Full-time NN

1024 MACs
General Release: Sep 2017

Two product lines: Imaging/Vision & Neural network DSP

CONFIDENTIAL

Vision P5/P6 Architecture



VLIW & SIMD	5 slots 64way 8-bit 32way 16-bit 16way 32-bit
ALU Ops (MAX 4 out of 5 slots)	256 8-bit 128 16-bit 64 32-bit
MAC (1 of 5 slots)	**Vision P5: 64 (8x8)** **Vision P6: 256 (8x8)**
Memory Width	1024-bits 2 vector load/store units
# of Vector Registers	32
SuperGather	32 non-contiguous locations read/ written per instruction
Bus Interface	AXi4
iDMA	no alignment restrictions, local memory to local memory transfers, …
Target Frequency	800Mhz @28nm 1.1 GHz @16nm
Optional	Vector Floating Point, ECC

Vision P6 CNN Performance (Hand Code Runs)

Network	Author	Batch Size	FPS @ 1.1GHz (Vision P6)	DDR Latency (in cycles)
Inception V3 (299x299x3 Input ROI)	Multicoreware	1	33.4	With 0 DDR Latency
			32.7	With 100 DDR Latency
Alexnet (227x227x3 Input ROI)	Cadence	8	241	With 0 DDR Latency
			235	With 100 DDR Latency
Fast YOLO (Vehicle Detection)	Multicoreware	1	111	480p as input
GTSR (32x32 Input ROI)	Cadence	1	4598	16x8 MAC are used
Fast YOLO (People Detection)	BDTi	1	91	224x224 as input With 100 DDR Latency

Note:

- Inception V3, Alexnet & FastYolo are implemented with 8-bit data and 8-bit coefficient
- Reference memory configuration of 64KB 4-way I$ and 2x128 Data RAM has been used

CONFIDENTIAL

Vision P6 CNN Performance
(XNNC Runs)

Network	Batch Size	FPS @ 1.1GHz (with 100 cycles of DDR Latency)	Quantization Error (Top-1)
VGG-16	1	11	0.3%*
	8	12	
VGG-19	1	9	0.5%**
ResNet-50	1	36	0.9%*
ResNet-101	1	22	0.2%*
	8	23	
ResNet-152	1	14	0.7%**
Caffe Variant of Inception V3 (GoogleNet-bn)	1	66	0.3%*
	8	74	

*Tested over 50K Images
**Tested over 15K Images

- Between XNNC and Manual implementation we see ~20% difference, mostly coming from generic library and automatic code generation
- Reference memory configuration of 64KB 4-way I$ and 2x128 Data RAM has been used
- XNNC FPS are preliminary pending optimizations

Quantization: Top-1 Accuracy Loss

Floating Point → 8 Bit Fixed Point Weights
Less Than 1% Accuracy drop

Top-1 Accuracy in % for Networks manually Implemented on VP6

Network	Floating Point	16 bit Fixed Point	8 bit Fixed Point
AlexNet* (Cadence)	62.7%		61.6%
Fast YOLO on KITTI: mAP* (Multicoreware)	63.8%	63.8%	63.5%
Fast YOLO on MCW: mAP* (Multicoreware)	65.9%	65.8%	65.5%
Inception V3* (Multicoreware)	74.0%		73.3%

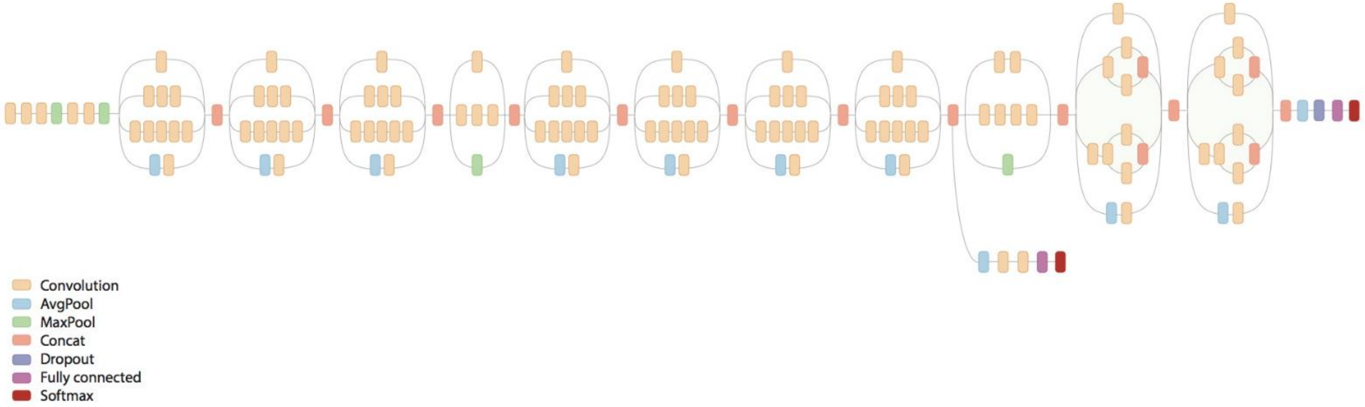
Top-1 Accuracy in % for Networks Implemented on VP6 using XNNC

Network	Floating Point	8 bit Fixed Point
VGG-16*	68.3%	68%
ResNet-101*	71.7%	71.5%
Inception V3* (caffe variant)	70.5%	70.2%
ResNet-152**	75.2%	74.5%
VGG-19**	67.8%	67.4%

*Tested over 50K ImageNet Dataset
**Tested over 15K ImageNet Dataset

Inception V3 Performance on Vision P6 (Batch size of 1)
(8bit Data and 8 bit Weights)

Inception V3 Details	
Input ROI	299x299x3
Number of Layers	110
Compute Requirement	5.78 GMAC
Bandwidth Requirement (8 bit Weights)	19.4 MB



Vision P6 FPS (@ 1.1GHz) With Tiling and DMA using Reference Configuration (256KB on chip memory)		
FPS	**Overall MAC Utilization**	**DDR Latency**
33.4	68.61%	0 cycles
32.7	67.16%	100 cycles

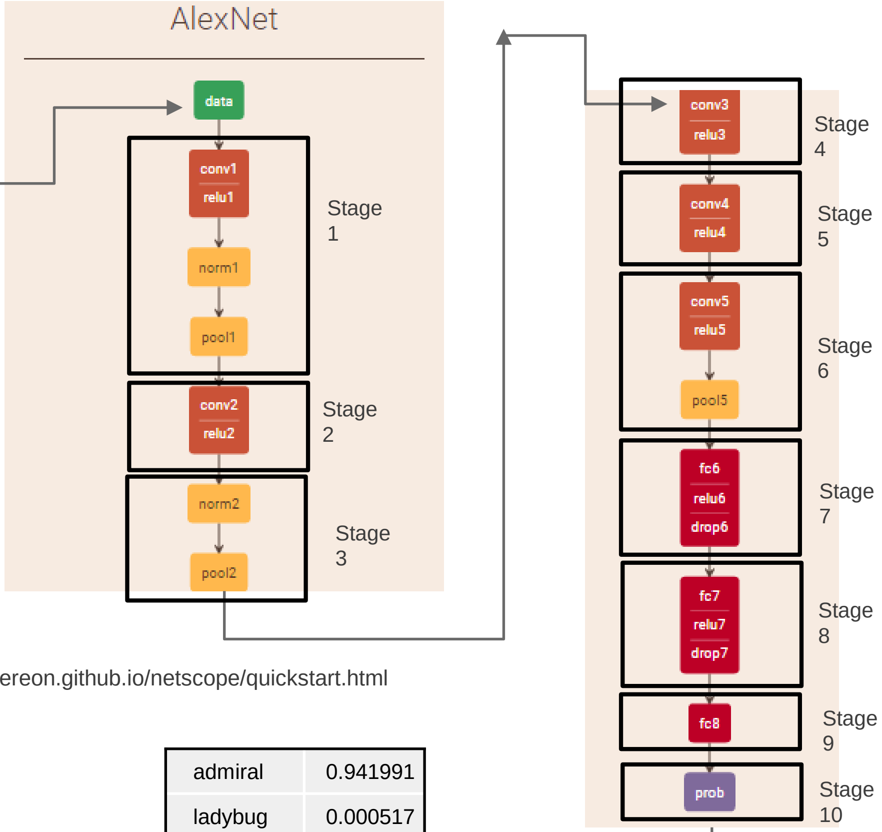
*Higher Batch sizes can provide better FPS

Vision P6 Accuracy (Loss <1%) Using 8bit Quantized Data & Weights		
Accuracy*	**Float**	**8bit Fixed Point**
Top-1 Accuracy	74.00%	73.29%
Top-5 Accuracy	91.62%	91.18%

*Accuracy tested over 50K images in ImageNet Val set

Vision P6 Running Alexnet Convolutional Neural Network

227x227 RGB image



Alexnet visualization from http://ethereon.github.io/netscope/quickstart.html

Vision P6 Detection Result

admiral	0.941991
ladybug	0.000517
monarch	0.000287
tench	0.000057
goldfish	0.000057

Alexnet:

Winner of the ImageNet (ILSVRC) 2012 Contest
Trained for 1000 different classes (images)
Most often quoted benchmark for CNN
Classifier CNN Example
5 Conv & 3 FC layers
Input image: 227x227 image patch (ROI)

Cadence Alexnet Implementation

Based on Caffe 32b floating point Alexnet model
Use 8 bit coefficients, 8 bit data computations
Pure C P6 implementation,
No library dependencies such as BLAS, NumPy, etc

Performance on Vision P6 @ 1.1GHz

- Including tiling and DMA with DDR latency of 0 is 241 fps
- Including tiling and DMA with DDR latency of 100 is 235 fps

Weighted dynamic power for core in mW/MHz

0.298 mW/MHz

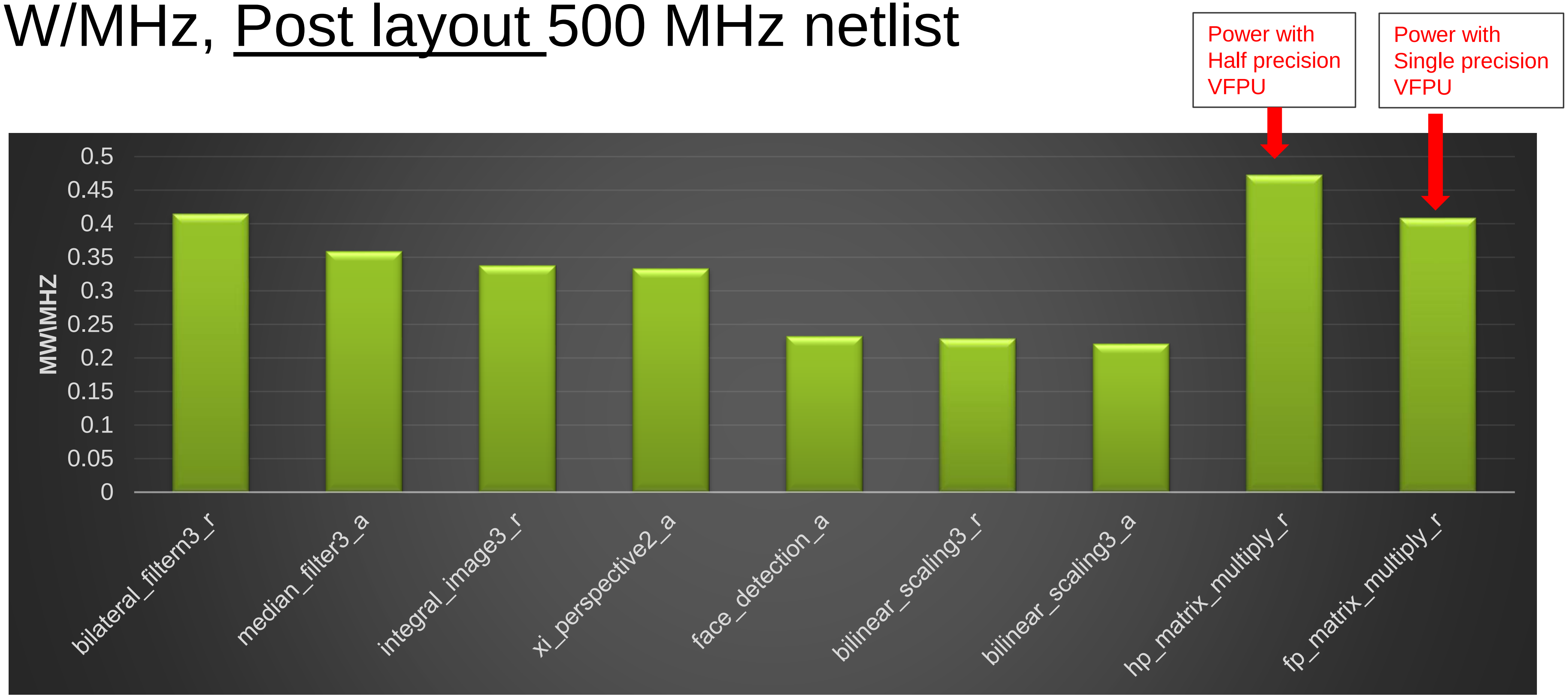
TSMC 16FF+ LL, RVT, 0.8V, 500MHz netlist

Vision P6 Area: TSMC 16FF+ LL 9-Track
Post layout Core Cell, Memory Cell and final Floor Plan

Metric	1.1GHz	800 MHz	500MHz (VP6P16)	500MHz (VP6HP16)	500MHz	
Vision P6 reference with	64KB 4-way I$ 2 x 128 KB Data RAMs **No VFPU**	64KB 4-way I$ 2 x 128 KB Data RAMs **No VFPU**	64KB 4-way I$ 2 x 128 KB Data RAMs **No VFPU**	64KB 4-way I$ 2 x 128 KB Data RAMs **Half Precision VFPU (FP16)**	64KB 4-way I$ 2 x 128 KB Data RAMs **Single Precision VFPU**	
	ILVT, LVT and RVT Overdrive library (Nominal VDD=1.0V)	RVT, LVT, ILVT (VDD=0.8V)	RVT only – Vdd 0.8 V	RVT only – Vdd 0.8 V	RVT only – Vdd 0.8 V	
Memory compiler:	tcbn16ffplusllbwp16p90cpd, version 111a	ts1n16ffplllvt, version 110b	tcbn16ffplusllbwp16p90cpd, version 111a	tcbn16ffplusllbwp16p90cpd, version 111a	tcbn16ffplusllbwp16p90cpd, version 111a	
Standard Cell Area	**0.58 mm^2**	**0.576 mm^2**	**0.531 mm^2**	**0.615 mm^2**	**0.604 mm^2**	Post layout standard cell area of Vision P6 (including iDMA, Supergather)
Memory Area	0.366 mm^2	0.366 mm^2	0.366 mm^2	0.366 mm^2	0.366 mm^2	Area depends on amount of memory, memory compiler

Half Precision VFPU unit adds ~ 15.81% area
Single Precision VFPU unit adds ~ 13.75% area

Vision P6 Power: TSMC 16ff+ LL 9 Track
In mW/MHz, Post layout 500 MHz netlist



Data represent mw/MHz

CONFIDENTIAL

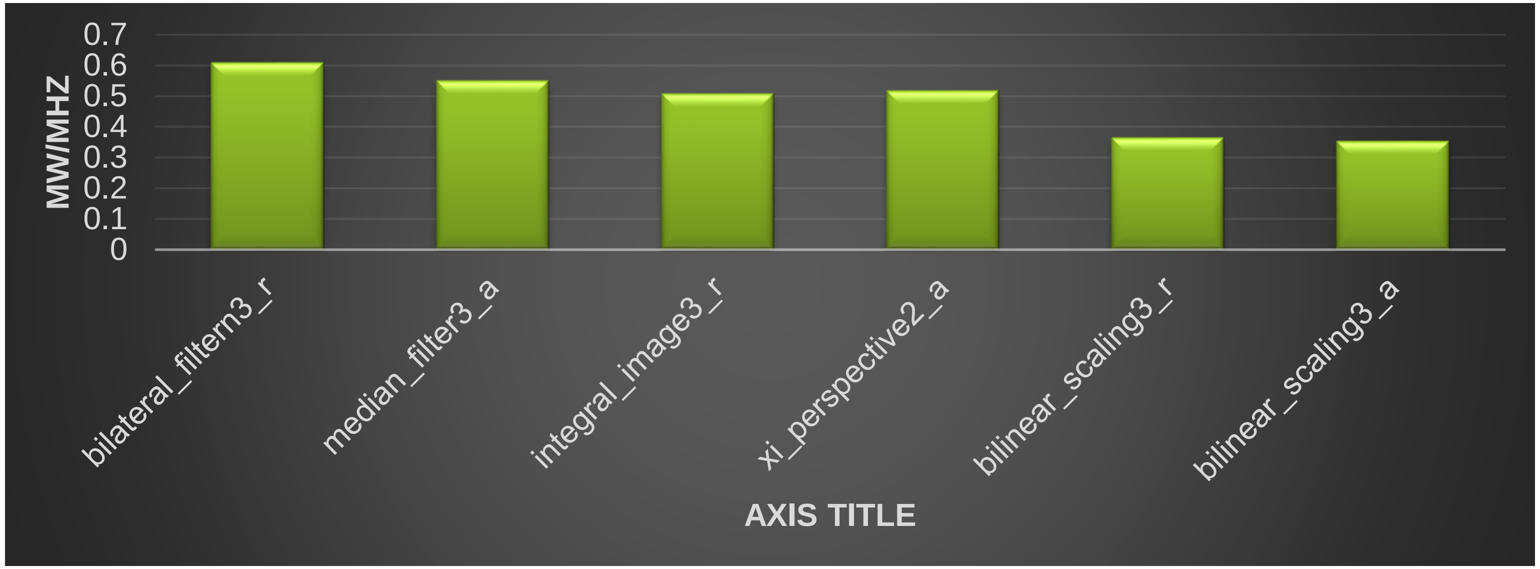
Vision P6 Area: TSMC 28HPC+ 9 Track
Post layout Core Cell, Memory Cell and final Floor Plan

Metric	500MHz	500MHz	500MHz (P6V_P28)	
Vision P6 reference with	64KB 4-way I$ 2 x 128 KB Data RAMs **No VFPU**	64KB 4-way I$ 2 x 128 KB Data RAMs **Half Precision VFPU**	64KB 4-way I$ 2 x 128 KB Data RAMs **Single Precision VFPU**	
	RVT only – Vdd 0.8 V	RVT only – Vdd 0.8 V	RVT only – Vdd 0.8 V	
Memory compiler:	tcbn28hpcplusbwp35p140, version 110c	tcbn28hpcplusbwp35p140, version 110c	tcbn28hpcplusbwp35p140, version 110c	
Post Layout Core Cell Area	**1.24 mm^2 ***	**1.436 mm^2 ***	**1.41 mm^2**	Post layout standard cell area of Vision P6 (including iDMA, Supergather)
Memory Area	0.6 mm^2	0.6 mm^2	0.6 mm^2	Area depends on amount of memory, memory compiler

*Numbers have been derived from VP6+SP VFPU based on mentioned scales

Half Precision VFPU unit adds ~ 15.81% area
Single Precision VFPU unit adds ~ 13.75% area

Vision P6 Power: TSMC 28HPC+ 9 Track
In mW/MHz, Post layout 500 MHz netlist



Data represent mw/MHz
Calculated on VP6 + SP VFPU

CONFIDENTIAL

Vision C5: Neural Network DSP

Tensilica® Vision C5 and Vision P6 DSPs:

Cadence Addressing All Market Segments

Processing Power

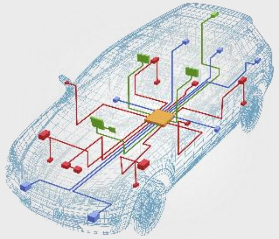
Up to 10TMAC/sec

1TMAC/sec

<200 GMAC/sec

Automotive (towards autonomous)

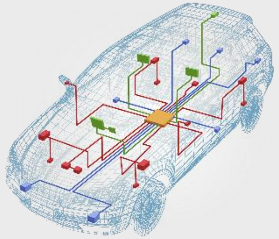
Multiple Vision C5 DSPs



Runs multiple NNs all the time

Surveillance / Automotive (semi-autonomous) Future Mobile

Vision C5 DSP



Runs a couple of NNs all the time

Today’s Mobile

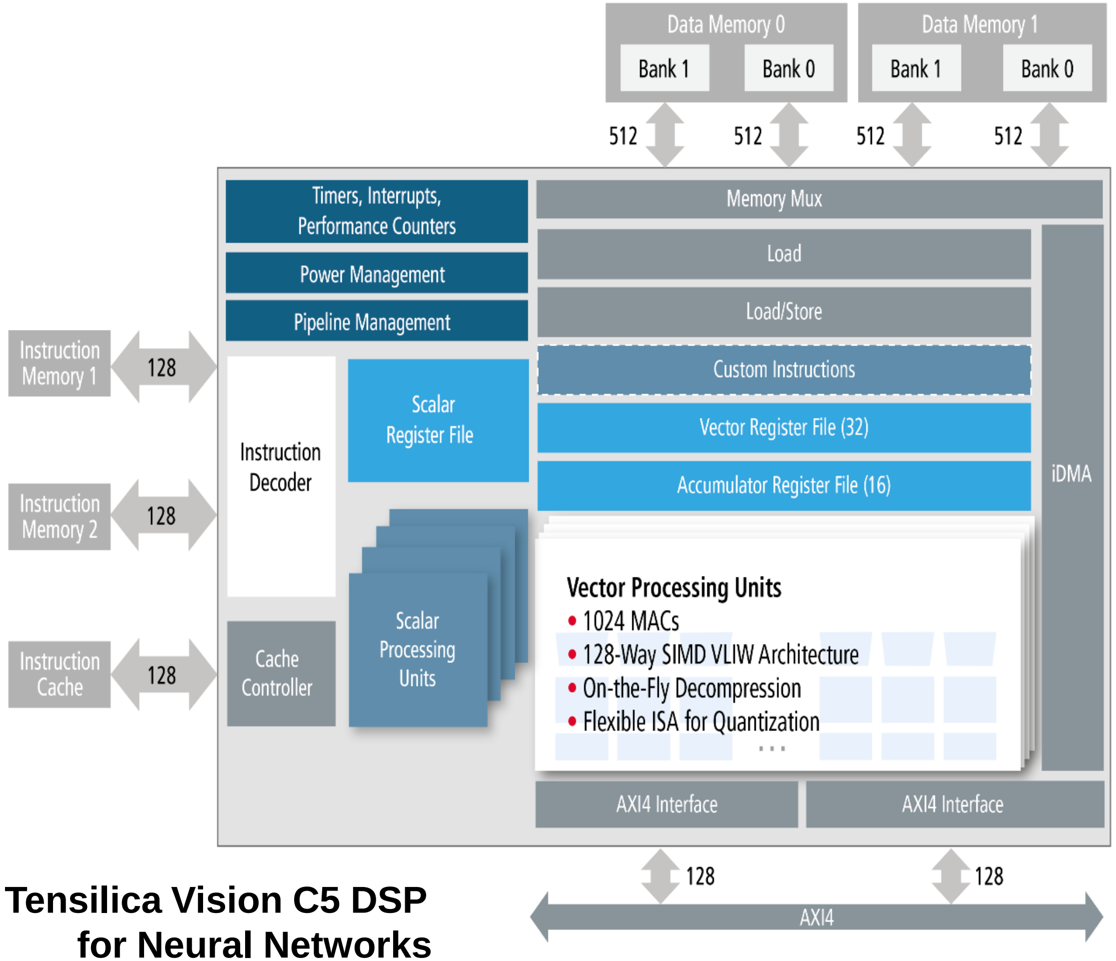
Vision P6 DSP

Runs a NN once in a while

Vision C5: DSP Architecture

- Fixed point DSP with 8-bit and 16-bit data type support
- 1024 8x8 MAC or 512 16x16 MAC throughput per cycle
  - Emphasis on high utilization of MACs across range of layer dimensions
- SIMD architecture for high performance vector computing
  - 512-bit vector register file that can work as 1024-bit register (pairing 2 512-bit registers)
    - 128-way SIMD for 8-bit data type, 64-way SIMD for 16-bit data type
  - 1536-bit wide accumulator register file that works as 3072-bit accumulator register (pairing 2 1536-bit accumulator)
- VLIW architecture to exploit instruction level parallelism
  - 88-bit wide VLIW instructions, supports 3 and 4 slot instruction formats
  - Ability to perform load/store, MAC/ALU/SELECT, PACK, decompress operations in parallel
- Dual load/store architecture, capable of two 512-bit loads or one load and one store in parallel
  - Including support for loading unaligned data from memory
  - Special addressing mode for efficient access of 3-D data

Vision C5: Memory Architecture



Tensilica Vision C5 DSP for Neural Networks

- TCM based local Data Memory
- Two Data Memory used in ping pong fashion
  - Hide system memory latencies
- Each Data RAM size expected between 128K to 256K depending on application
- Each Data RAM can be banked into 2 or 4 to achieve maximum data access throughput
- Integrated DMA engine to transfer data between local memory and system memory
- Instruction memory supports both caches and TCM
- Two 128-bit AXI interfaces
  - one dedicated to DMA
  - Another one for the processor load/store and instruction memory access to main memory

cadence

Vision C ISA Highlights: Load/Store

- Vector load/store of 64 8-bit elements; can also be viewed as 32 16-bit element load
- Vector loads of 32 8-bit elements, with each element sign/zero extended to 16-bit
- Vector stores of 32 16-bit elements, with each element saturated/truncated to 8-bits
- Vector stores of 32 8-bit elements (low or high half)
- Boolean register load/stores
- Unaligned vector loads of 64 8-bit elements
- Unaligned vector loads of 32 8-bit elements, with element sign/zero extended to 16-bit
- Variable vector load/stores of up to 64 8-bit elements
- Immediate and Index addressing mode, including post increment of address
    - Not all addressing modes supported for all types of loads/stores
- Special loads
    - Load with offset: 8-bit, 16-bit, aligned, unaligned. Example: IVP_LV2NX8_ORP
    - Unaligned loads for compressed data.
    - Interleaving loads and stores: Used for MOD vectorization, when data from 2 different locations are to be multiplied with the same set of coefficients. Uses alignment register to help interleave data from two separate loads into a vector register. Example: IVP_LVINTL16A2NX8_IP

Vision C ISA Highlights: ALU, Shifts and Moves

ALU Support

- 8b, 16b vector add, subtract
- 24b, 48b wide vector add, subtract
- Few variations of Reduction ADD
- Support for addition of upper and lower half of wide accumulator vectors
- Support for different variants of MIN, MAX, CMP (compare) for 8 and 16 bit
- Support for Bitwise AND, OR and XOR for 8 and 16 bit

Shift Operations

- 8b, 16b arithmetic and logic right shift
- 8b, 16b left shift

Move Operations

- Vector to vector register move
- Accumulator to accumulator move
- Vector to accumulator and vice versa
- AR to vector register move with replication

Vision C ISA Highlights: Select Operations

- Extract a scalar 8/16 bit value from vector register to AR
- Arbitrary selection of 8/16-bit elements from two input vector registers to the destination vector register
  - This is the typical SEL operation found in most of our DSPs
- Input can be two vector registers (output is one vector register) or two paired vector registers (output is one paired vector register)
- Special version of the above general operation, using an immediate operand to select from a table of most common select values
- Squeeze and Extract operations illustrated in the example related to avoiding “multiply by zero” data values

Vision C ISA Highlights: PACK Operations

- PACK operations are used to convert data from the (wider) “accumulator format” to the (narrower) “vector register” format
  - Right shift, saturate, optional rounding
- Vision C supports dual PACKs that pack 2 accumulators to 2 vectors; twice the throughput of VP6
- 256 24-bit elements from accumulator to 256 8-bit elements in vector register
  - Signed and unsigned saturation based on data type of result
- 128 48-bit elements from accumulator to 128 16-bit elements in vector register
- 128 24-bit elements from accumulator to 128 16-bit elements in vector register
- Reduction ADD + PACK: For output of “folded” multiplies
  - 2 way and 4 way reduction ADD before right shift and saturate
  - Result is either 8-bit or 16-bit

Tensilica® Vision C5 DSP vs NN Accelerator

Vision C5 DSP

A complete processor that stands on its own:
Accelerates all NN layers

Flexible and future-proof solution:

- Supports variable kernel sizes, depths, input dimensions
- Supports different compression/ decompression techniques
- Support for new layers can be added as they evolve

Main vision/imaging DSP free to run other applications while NN DSP runs NN

Simple **(single-processing)** programming model for NN

No need to move data between NN DSP and main vision/imaging DSP

NN Accelerator

Built to accelerate **only NN convolution functions**

HW accelerators are mostly designed based on current needs and hence provide a rigid and not future-proof solution

While running NN, **main vision/imaging DSP cannot run other applications**

Complicated **multi-processor** programming model

Need to move data between NN DSP and main vision/imaging DSP (wastes power)

Why Vision C5

Challenge #1: MAC architecture

Challenge: MAC Architecture

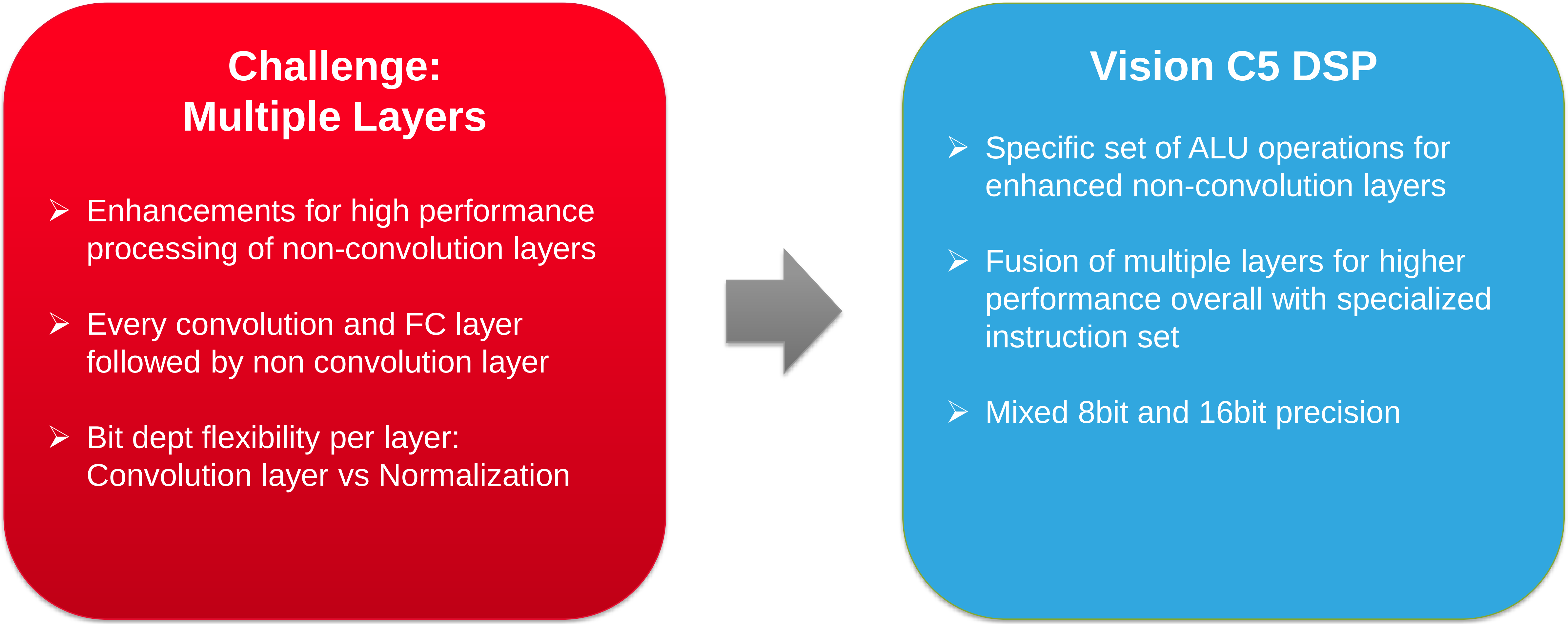
- ➢ Efficient implementation of Multiplier accumulator (MAC) architecture
- ➢ Achieve high MAC utilization
- ➢ MAC architecture has to work across different types of three dimensional convolution
- ➢ The MAC architecture has to work with different convolution sizes
  - ➢ 11x11, 7x7, 5x5,3x3, 1x1

Vision C5 DSP

- ➢ Specialized dual quad MAC architecture to get almost 100% MAC utilization for inner loops
- ➢ Optimizations for very small convolution dimensions
- ➢ Enhanced instruction set for multiple vectorization schemes (depth vs width) for best performance in different cases
- ➢ Data compression to avoid multiplication by zero

Why Vision C5

Challenge #2: Non-Convolution Layers



Challenge: Multiple Layers

- Enhancements for high performance processing of non-convolution layers
- Every convolution and FC layer followed by non convolution layer
- Bit dept flexibility per layer: Convolution layer vs Normalization

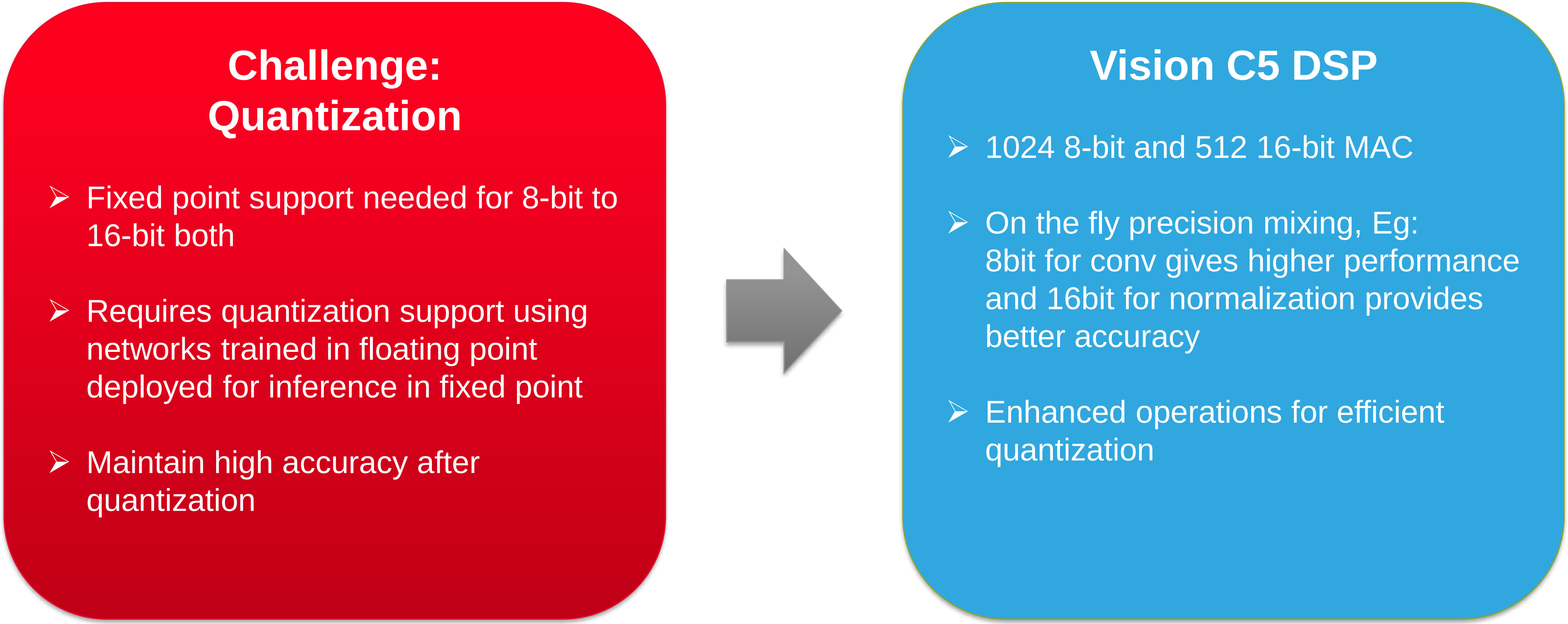
Vision C5 DSP

- Specific set of ALU operations for enhanced non-convolution layers
- Fusion of multiple layers for higher performance overall with specialized instruction set
- Mixed 8bit and 16bit precision

CONFIDENTIAL

Why Vision C5

Challenge #3: Quantization



Challenge: Quantization

- Fixed point support needed for 8-bit to 16-bit both
- Requires quantization support using networks trained in floating point deployed for inference in fixed point
- Maintain high accuracy after quantization

Vision C5 DSP

- 1024 8-bit and 512 16-bit MAC
- On the fly precision mixing, Eg: 8bit for conv gives higher performance and 16bit for normalization provides better accuracy
- Enhanced operations for efficient quantization

Why Vision C5

Challenge #4: Memory Bandwidth

Challenge:
Memory Usage

- Memory bandwidth is limited and often shared
- Access to memories must be efficient and necessary
- Reduce data manipulation impact on processor cycles to keep processing efficiency high

Vision C5 DSP

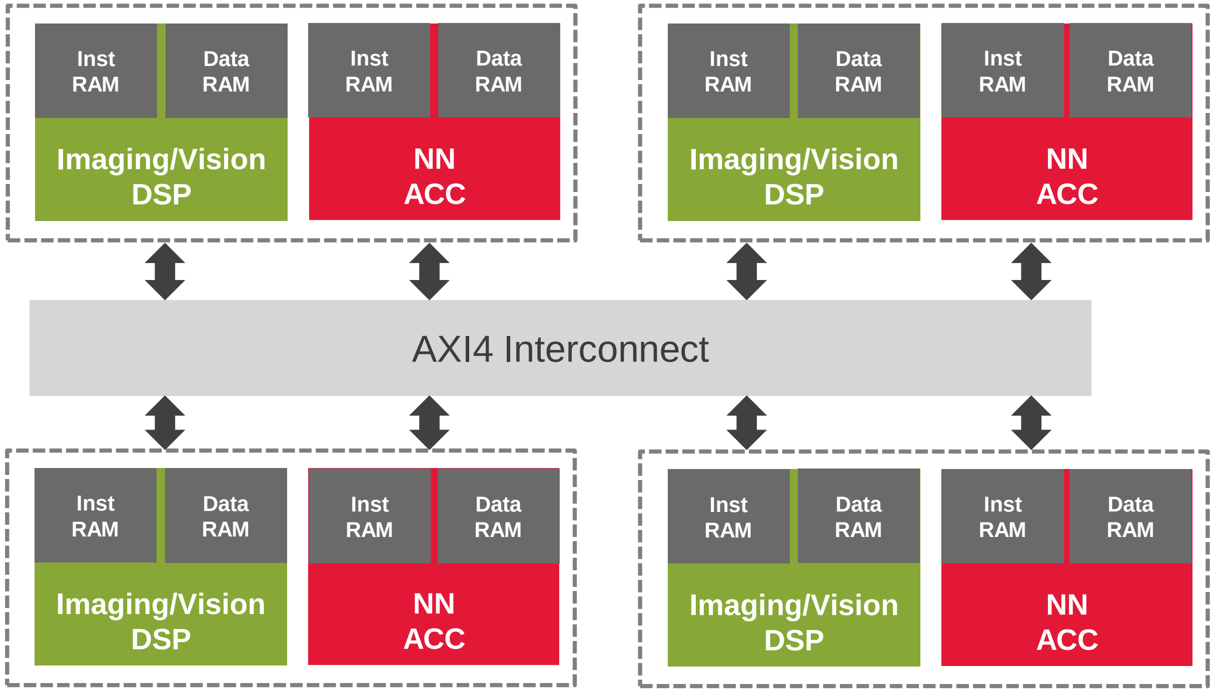
- Integrated DMA:
  No need for controller, can manage its own data movement
- Ping-pong memory buffering hides DMA latency
- Rich and wide set of select instructions for data manipulation
- On the fly weight decompression for bandwidth reduction

Tensilica® Vision C5 DSP vs NN Accelerator

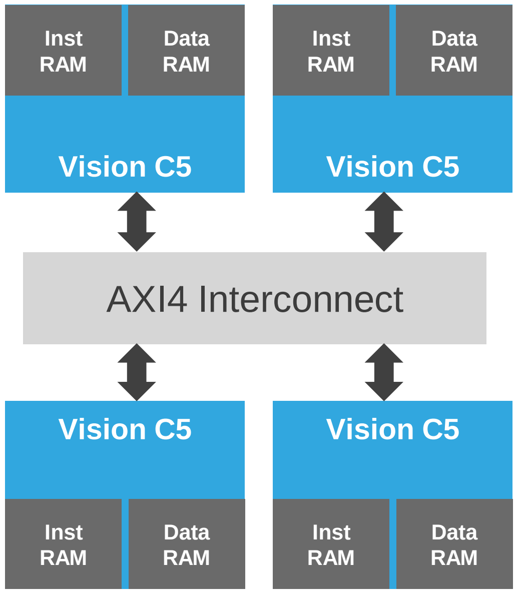
	Neural Network	Accelerator	Neural Network DSP
MULTIPLE	Convolution Layer	Runs on NN accelerator Move Data	Entire NN runs on Vision C5 DSP No extraneous data movement Does not occupy the imaging/vision DSP
	Normalization/ Max Pooling	Runs on companion imaging/vision DSP Move Data	
MULTIPLE	Fully Connected Layer	Runs on NN accelerator Move Data	
	Normalization/ Max Pooling	Runs on companion imaging/vision DSP Move Data	

Multi-Core Solution with Tensilica® Vision C5 DSP vs NN Accelerator

Multi-Core with Vision DSP + NN Accelerator



Multi-Core with Vision C5 DSP



- Vision C5 DSP scales elegantly
- NN accelerator approach requires imaging/vision DSP with each core; increased area and power

Vision C5 Area: TSMC 16FFC 9-Track
Post layout Core Cell, Memory Cell and final Floor Plan

Metric	500MHz	Notes
Vision C5 reference with	64KB 4-way I$ 2 x 128 KB Data RAMs	
	RVT only – Vdd 0.8 V	
Memory compiler:	ts1n16ffcllsvta64x20m4sw version 100a	
Standard Cell Core Area	**0.895 mm^2**	Post layout standard cell core area of Vision C5 not including utilization ratio (including iDMA)
Memory Area	0.334 mm^2	Area depends on amount of memory, memory compiler

Vision C5 8b MAC/mm2 = 2.8x Vision P6 8b MAC/mm2
Vision C5 16b MAC/mm2 = 5.6x Vision P6 16b MAC/mm2

Vision C5 Area: TSMC 28HPC+ 9-Track
Post layout Core Cell, Memory Cell and final Floor Plan

Metric	500MHz	750MHz	
Vision C5 reference with	64KB 4-way I$ 2 x 128 KB Data RAMs	64KB 4-way I$ 2 x 128 KB Data RAMs	
	RVT only – Vdd 0.9 V	RVT only – Vdd 0.9 V	
Memory compiler:	Tcbn28hpcplusbwp35p140 version 110c	Tcbn28hpcplusbwp35p140 version 110c	
Standard Cell Area	**1.99 mm^2**	**2.679 mm^2**	Post layout standard cell area of Vision C5 (including iDMA)
Memory Area	0.571 mm^2	0.571 mm^2	Area depends on amount of memory, memory compiler

Vision C5 8b MAC/mm2 = 2.8x Vision P6 8b MAC/mm2
Vision C5 16b MAC/mm2 = 5.6x Vision P6 16b MAC/mm2

CONFIDENTIAL

AlexNet Performance Estimate on Vision C5
(8bit Data and 8 bit Weights)

AlexNet Details	
Input ROI	227x227x3
Number of Layers	14

Vision P6 FPS/GHz		Vision C5 FPS/GHz	
FPS	**DDR Latency**	**FPS**	**DDR Latency**
214	100 cycles	700	100 cycles

VC5 AlexNet Weighted Power (mW/MHz)	
TSMC 16FFC, RVT, 0.8V	0.65
TSMC 28HPC+, RVT, 0.9V	1.05

- **Performance**: Vision C5 = 3.27x Vision P6
- **Efficiency (Perf/mm2)**: Vision C5 = 2.3x Vision P6
- **Efficiency (Perf/mW)**: Vision C5 = 1.5x Vision P6
- FPS numbers are from manual optimized implementation
- Batching is used in layers that are potentially memory limited
- Performance results are preliminary pending further optimizations

ResNet50 Performance Estimate on Vision C5
(8bit Data and 8 bit Weights)

ResNet50 Details	
Input ROI	224x224x3
Number of Layers	223

Vision P6 FPS/GHz		Vision C5 FPS/GHz	
FPS	**DDR Latency**	**FPS**	**DDR Latency**
33	100 cycles	129	100 cycles

VC5 ResNet50 Weighted Power (mW/MHz)	
TSMC 16FFC, RVT, 0.8V	0.82
TSMC 28HPC+, RVT, 0.9V	1.31

- **Performance**: Vision C5 = 3.7x Vision P6
- **Efficiency (Perf/mm2)**: Vision C5 = 2.6x Vision P6
- Vision P6 is measured from XNNC implementation and Vision C5 is estimated from hand written kernels extrapolated to all layers
- Performance results are preliminary pending further optimizations
- Reference memory configuration of 64KB 4-way I$ and 2x128 Data RAM has been used
- Batching is used in any layer that is potentially memory limited

InceptionV3 Performance Estimate on Vision P6 and Vision C5 (8bit Data and 8 bit Weights)

Inception V3 Details	
Input ROI	299x299x3
Number of Layers	110

Vision P6 FPS/GHz		Vision C5 FPS/GHz	
FPS	**DDR Latency**	**FPS**	**DDR Latency**
30	100 cycles	103	100 cycles

- **Performance**: Vision C5 = **3.4x** Vision P6
- **Efficiency (Perf/mm2)**: Vision C5 = **2.4x** Vision P6
- Vision P6 is measured from XNNC implementation and Vision C5 is estimated from hand written kernels extrapolated to all layers
- Reference memory configuration of 64KB 4-way I$ and 2x128 Data RAM has been used
- C5 results are preliminary pending further optimizations

Vision C5 Benchmark (Early Estimates)

Layer	Dimensions			Precision	Vector Approach	MAC Utilization (in %)	
	Input (W x H x D)	Number of Kernels	Kernel (W x H)			Inner loop (static schedule)	Kernel (with mem model)
Custom Convolution	960x5x32	16	5x5	8b x 8b	MOW	100	86
Custom Convolution	480x5x32	16	5x5	16b x 16b	MOW	100	86
Custom Convolution	1020x5x32	16	5x5	8b x 8b	MOW	100	93
Custom Convolution	160x6x64	64	5x5	8b x 8b	MOW	100	71
Custom Convolution	160x6x64	64	5x5	8b x 8b	MOD	89	74
Res2a_branc2b	56x16x64	64	3x3	8b x 8b	MOW	100	75
Res2a_branc2b	56x16x64	64	3x3	8b x 8b	MOD	89	75
Resnet 1x1	56x56x256	8	1x1	8b x 8b	MOD	94	75
Resnet 1x1	28x28x256	8	1x1	16b x 16b	MOD	89	73
FC (32 Batch)	2x1x256	128	2x1x256	8b x 8b	MOD	94	66

- Tile sizes are based on 2 x 128K Data RAM
- Performance is for tiles in local memory
- DMA is assumed to be in background and not included in above cycles

Vision C5 DSP vs Nvidia TX1

AlexNet

Up to 6X* faster

Inception V3

Up to 9X** faster

ResNet50

Up to 4.5x*** faster

Note:
https://arxiv.org/pdf/1605.07678v2.pdf
Both cores running at 690MHz on 16nm
* AlexNet data with 8 batch
** Inception V3 data with single batch
*** ResNet50 with batching

Vision P6 vs Vision C5

	Vision P6	Vision C5
Focus	Imaging and NN	NN
MAC (8x8)	256	1024
MAC (16x16)	64	512
Single and Half Precision VFPU (optional)	32 way FP16 16 way FP32	Not Required for Inference
Accumulators (to support higher MAC capability)	4 x 1536b	8 x 3072b
MAX SIMD Width	64 way SIMD	128 way SIMD
Special Features	Scatter Gather (needed by Imaging Applications)	• On the Fly Decompression Support • Special addressing modes • Richer set of convolution multipliers (signed and unsigned) • Extensive data rearrangement and selection

Cadence Tensilica Vision C5 DSP Summary

For all neural network inference applications

DSP optimized to process all neural network inference layers

- Instruction set and DSP architecture designed specifically for NN
- Not just a “convolution accelerator”

Scalable from 1 TeraMAC/sec computational capacity in less than 1mm^2

- Easily scales to multi-TMAC/sec systems, no limit to the number of Vision C5 DSPs
- Designed for high-availability (always-on) neural network computational needs

Flexible and programmable

- To meet evolving neural network requirements

Targeted at surveillance, automotive, drone and mobile/wearable markets

- Optimized for vision, radar/lidar and fused-sensor applications

Development Tools

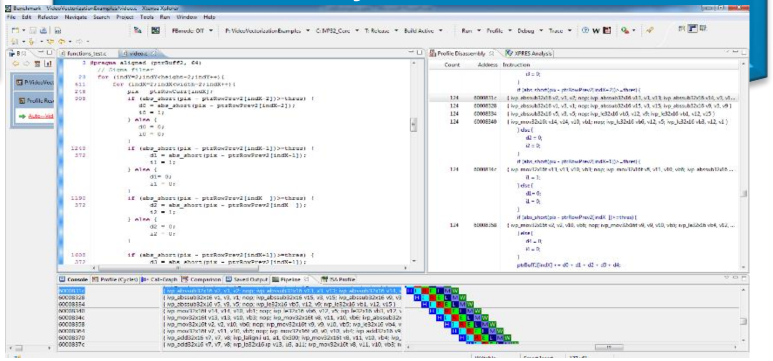
Vision Family Development Tools

Developed on rich Xtensa heritage

- >17 year history of Xtensa architecture
- Billions of core in production

Seamless development environment

- Eclipsed based IDE GUI
- Xtensa C/C++ (XCC) Compiler with auto-vectorization
- GNU Software Toolkit (Assembler, Linker, Debugger, Profiler)
- Vision Library



```
   3 #pragma aligned (ptrBuff2, 64)
       // Sigma filter
  23   for (indY=2;indY<height-2;indY++){
 611     for (indX=2;indX<width-2;indX++){
 218       pix   ptrRowCurr[indX];
 908       if (abs_short(pix - ptrRowPrev2[indX-2])>=thres) {
             d0 = abs_short(pix - ptrRowPrev2[indX-2]);
             i0 = 1;
           } else {
             d0 = 0;
             i0 = 0;
           }
1240       if (abs_short(pix - ptrRowPrev2[indX-1])>=thres) {
 372           d1 = abs_short(pix - ptrRowPrev2[indX-1]);
               i1 = 1;
           } else {
               d1= 0;
               i1 = 0;
           }
1190       if (abs_short(pix - ptrRowPrev2[indX  ])>=thres) {
 372           d2 = abs_short(pix - ptrRowPrev2[indX  ]);
               i2 = 1;
           } else {
               d2 = 0;
               i2 = 0;
           }
1608       if (abs_short(pix - ptrRowPrev2[indX+1])>=thres) {
 372           d3 = abs_short(pix - ptrRowPrev2[indX+1]);
```

```
Count  Address   Instruction
                   i3 = 0;
                 }
                 if (abs_short(pix - ptrRowPrev2[indX-2])>=thres) {
124    6000831c  { ivp_abssub32x16 v2, v1, v2; nop; ivp_abssub32x16 v13, v3, v13; ivp_abssub32x16 v14, v3, v1...
124    60008328  { ivp_abssub32x16 v1, v3, v1; nop; ivp_abssub32x16 v15, v3, v15; ivp_abssub32x16 v9, v3, v9 }
124    60008334  { ivp_abssub32x16 v5, v3, v5; nop; ivp_le32x16 vb3, v12, v9; ivp_le32x16 vb1, v12, v15 }
124    60008340  { ivp_mov32x16t v14, v14, v10, vb1; nop; ivp_le32x16 vb6, v12, v5; ivp_le32x16 vb3, v12, v1 }
                 } else {
                   d2 = 0;
                   i2 = 0;
                 }
                 if (abs_short(pix - ptrRowPrev2[indX-1])>=thres) {
124    6000834c  { ivp_mov32x16t v13, v13, v10, vb3; nop; ivp_mov32x16t v8, v11, v10, vb6; ivp_abssub32x16 ...
                   i1 = 1;
                 } else {
                   d1 = 0;
                   i1 = 0;
                 }
                 if (abs_short(pix - ptrRowPrev2[indX ])>=thres) {
124    60008358  { ivp_mov32x16t v2, v2, v10, vb6; nop; ivp_mov32x16t v9, v9, v10, vb5; ivp_le32x16 vb4, v12, ...
                 } else {
                   d4 = 0;
                   i4 = 0;
                 }
                 ptrBuff2[indX] += d0 + d1 + d2 + d3 + d4;
```

```
6000831c  { ivp_abssub32x16 v2, v1, v2; nop; ivp_abssub32x16 v13, v3, v13; ivp_abssub32x16 v14,
60008328  { ivp_abssub32x16 v1, v3, v1; nop; ivp_abssub32x16 v15, v3, v15; ivp_abssub32x16 v9, v3
60008334  { ivp_abssub32x16 v5, v3, v5; nop; ivp_le32x16 vb3, v12, v9; ivp_le32x16 vb1, v12, v15 }
60008340  { ivp_mov32x16t v14, v14, v10, vb1; nop; ivp_le32x16 vb6, v12, v5; ivp_le32x16 vb1, v12, v
6000834c  { ivp_mov32x16t v13, v13, v10, vb3; nop; ivp_mov32x16t v8, v11, v10, vb6; ivp_abssub32x
60008358  { ivp_mov32x16t v2, v2, v10, vb6; nop; ivp_mov32x16t v9, v9, v10, vb5; ivp_le32x16 vb4, v
60008364  { ivp_mov32x16t v7, v11, v10, vb5; nop; ivp_mov32x16t v0, v0, v10, vb4; ivp_add32x16 v9
60008370  { ivp_add32x16 v7, v7, v8; ivp_lalign.i u1, a1, 0x300; ivp_mov32x16t v8, v11, v10, vb4; ivp_
6000837c  { ivp_add32x16 v7, v7, v8; ivp_le32x16.ip v13, u8, a11; ivp_mov32x16t v8, v11, v10, vb3; iv
```

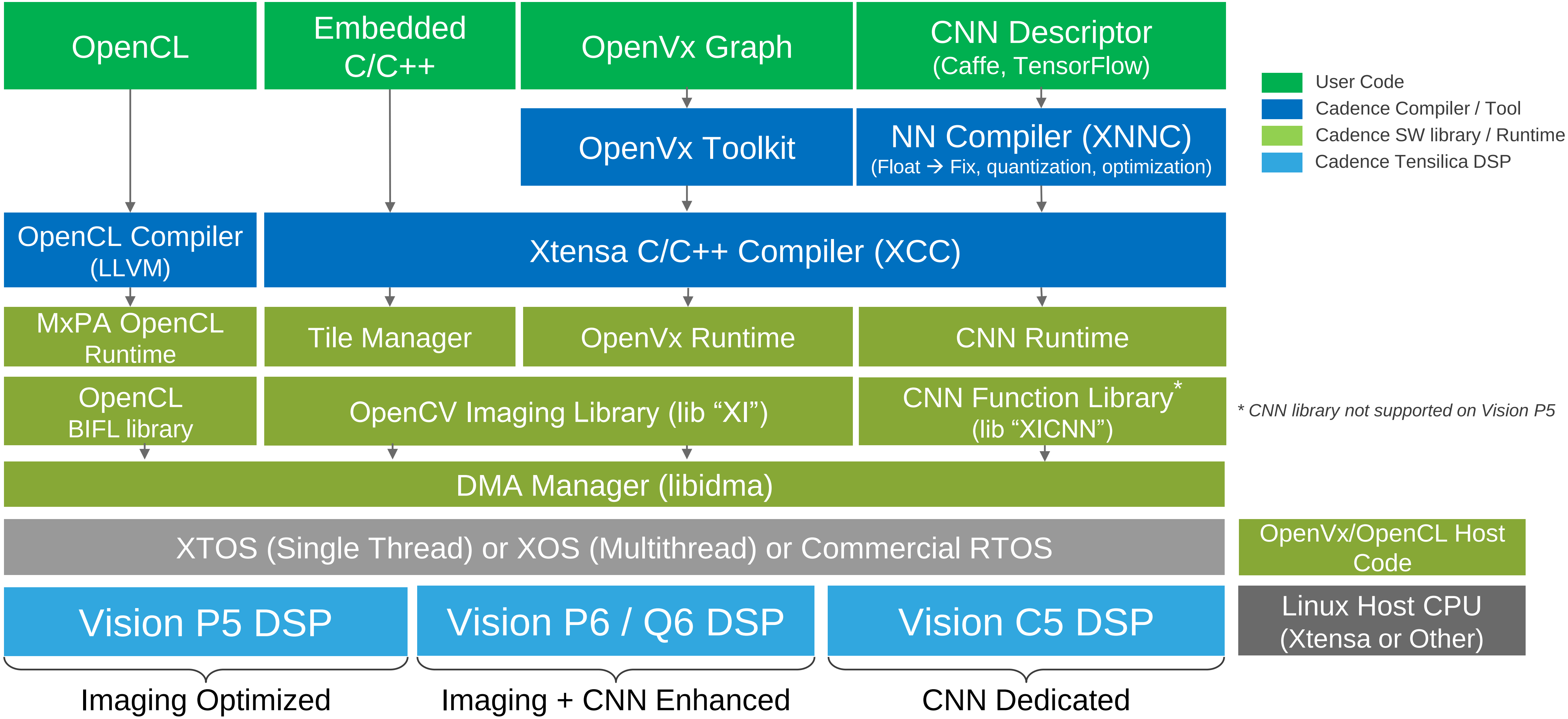
Complete package for integration

- RTL, EDA script
- Cycle accurate Instruction Set model
- Fast Function Simulator (TurboXIM)
- XTSC System C system modeling

CONFIDENTIAL

Tensilica: Comprehensive Vision Software & Hardware Solutions

Full Ecosystem of Software Frameworks and Compilers for all Vision and Imaging Programming Styles



OpenCL

Embedded C/C++

OpenVx Graph

CNN Descriptor (Caffe, TensorFlow)

OpenVx Toolkit

NN Compiler (XNNC) (Float → Fix, quantization, optimization)

OpenCL Compiler (LLVM)

Xtensa C/C++ Compiler (XCC)

MxPA OpenCL Runtime

Tile Manager

OpenVx Runtime

CNN Runtime

OpenCL BIFL library

OpenCV Imaging Library (lib “XI”)

CNN Function Library* (lib “XICNN”)

DMA Manager (libidma)

XTOS (Single Thread) or XOS (Multithread) or Commercial RTOS

OpenVx/OpenCL Host Code

Vision P5 DSP

Vision P6 / Q6 DSP

Vision C5 DSP

Linux Host CPU (Xtensa or Other)

Imaging Optimized

Imaging + CNN Enhanced

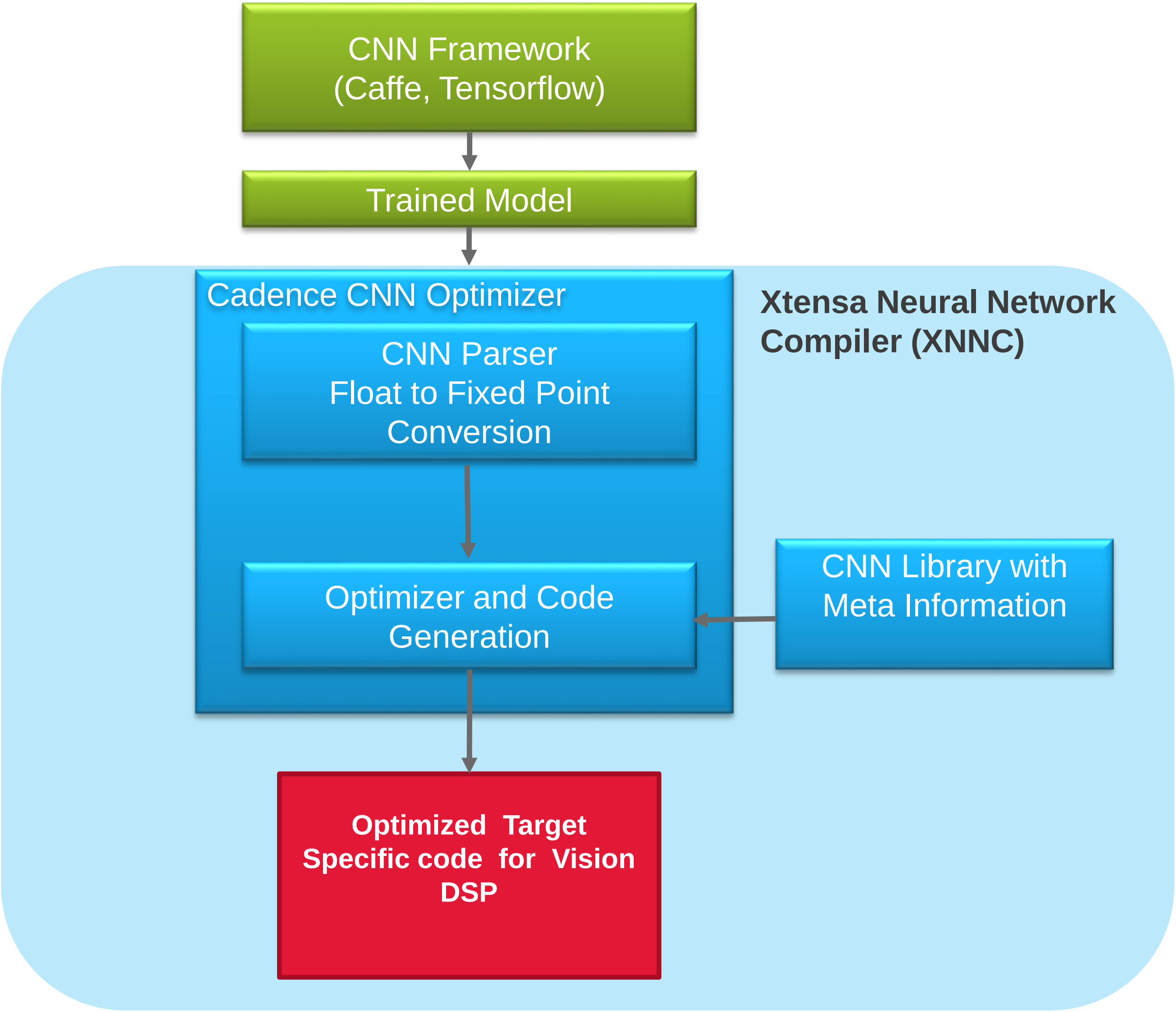
CNN Dedicated

🟩 User Code
🟦 Cadence Compiler / Tool
🟩 Cadence SW library / Runtime
🟦 Cadence Tensilica DSP

** CNN library not supported on Vision P5*

CONFIDENTIAL

Xtensa Neural Network Compiler (XNNC)
(Starting From Vision P6)


CNN Framework (Caffe, Tensorflow)
Trained Model
Cadence CNN Optimizer
CNN Parser Float to Fixed Point Conversion
Optimizer and Code Generation
Xtensa Neural Network Compiler (XNNC)
CNN Library with Meta Information
Optimized Target Specific code for Vision DSP

- Connects to existing industry CNN frameworks by using their Trained Model descriptions
- and **auto-generates Trained Model optimized code for Cadence CNN DSPs**

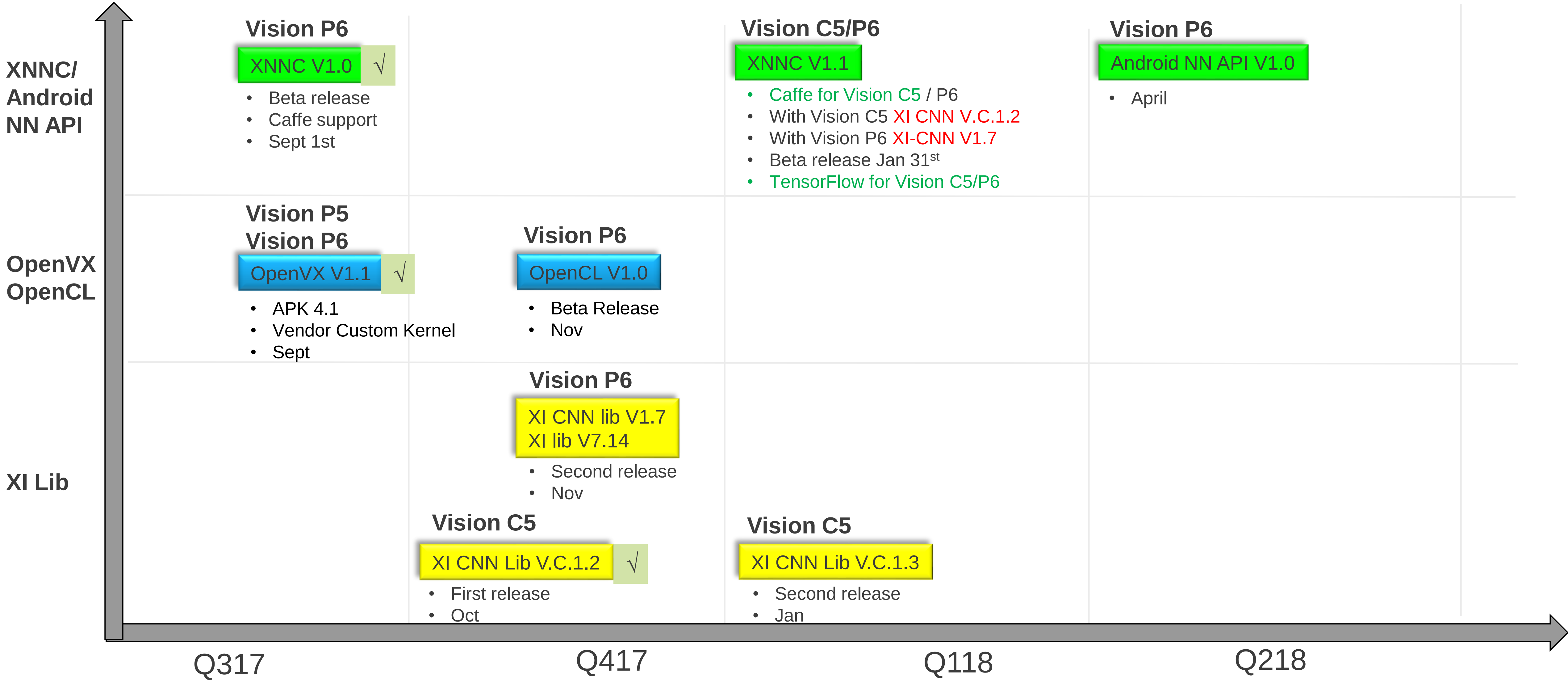
- Three Major components to XNNC
- CNN Parser: Float to Fixed Point conversion
- CNN Code Generation and Optimization
- CNN Library for Vision DSP

- First CNN Framework support: Caffe, followed by Tensorflow
- For both Vision P6 and Vision C5

All dates and plans preliminary and subject to change; no commitment to any product development schedule is implied

Vision DSP SW Roadmap

Preliminary
Oct 2017



XNNC/ Android NN API

Vision P6

XNNC V1.0 √

- Beta release
- Caffe support
- Sept 1st

Vision C5/P6

XNNC V1.1

- Caffe for Vision C5 / P6
- With Vision C5 XI CNN V.C.1.2
- With Vision P6 XI-CNN V1.7
- Beta release Jan 31st
- TensorFlow for Vision C5/P6

Vision P6

Android NN API V1.0

- April

OpenVX OpenCL

Vision P5
Vision P6

OpenVX V1.1 √

- APK 4.1
- Vendor Custom Kernel
- Sept

Vision P6

OpenCL V1.0

- Beta Release
- Nov

XI Lib

Vision P6

XI CNN lib V1.7
XI lib V7.14

- Second release
- Nov

Vision C5

XI CNN Lib V.C.1.2 √

- First release
- Oct

Vision C5

XI CNN Lib V.C.1.3

- Second release
- Jan

CONFIDENTIAL

Applications Across Market and Cadence Ecosystem

	Mobile	Auto	Security	Gaming	Wearable	Partners (Sources)	
Face Detect	🟩	🟩	🟩	🟩	🟩	Irida Labs, Cadence	Developed by Cadence Irida Labs
People Detection (HOG)		🟩	🟩			Uurmi System (Now Mathworks), MultiCoreware, Cadence (demo)	Developed by Cadence
DoG Differences of Gaussian	🟩	🟩	🟩		🟩	BDTi, Cadence (XI Lib)	Parameterized and Optimized Version
CNN	🟩	🟩	🟩		🟩	Multicoreware, BDTi	CNN prediction done by Multicoreware and BDTi (Yolo)
Lane Departure Warning		🟩				Cadence (Demo)	Available
Fog removal		🟩				Uurmi System (Now Mathworks)	Available on Vision P5 (Uurmi)
Traffic Sign Detection	🟩	🟩	🟩			Cadence (Demo)	Developed by Cadence (CNN)
Stereo Sensors	🟩	🟩	🟩	🟩	🟩	Uurmi(Now Mathworks), Multicoreware	Available on Vision P5 (Uurmi)
Super Resolution	🟩					Alamalence, Uurmi Systems(Now Mathworks)	Available on Vision P5
Video Stabilizer	🟩		🟩		🟩	Irida Labs, Morpho, Uurmi Systems(Now Mathworks)	Morpho
Video WDR	🟩	🟩	🟩			Morpho	Morpho
Low-Light	🟩	🟩	🟩	🟩	🟩	Irida labs, Morpho	Available (Irida labs)
3D Noise filtering	🟩	🟩	🟩	🟩	🟩	Morpho, Irida labs	Available (Irida labs)
Face & Voice Authentication	🟩	🟩	🟩	🟩	🟩	Sensory	Available
360 Defish		🟩	🟩			Multicoreware	Available

New Engagements

- Vangogh Imaging
  - For SLAM Algorithm
- Arcsoft
  - Face detection and face beautification
- Corephotonics
  - Stereo Image fusion
- ReadSense
  - Face recognition/People recognition
- Thundersoft
  - Computer Vision and Android

Summary

Imaging DSP and Neural Network DSP

- Market needs both Imaging DSP and Neural Network DSP
- Imaging processing requirement continues to increase: higher resolution and dual sensor
- Neural Network are evolving – computational capacity continue to increase

Vision C5 DSP for Neural Network

- Complete, standalone DSP that runs all layers of NN : Not an accelerator
- General purpose and programmable
- 1 TMAC/Sec computational capacity: 4X MAC capacity compare to Vision P6

Vision P6 DSP

- Up to 4X neural network performance compare to Vision P5
- 4X MAC capacity
- Easy migration of GPU floating-point code, with optional 16-bit floating-point unit
- Up to 4X performance improvement on well-known imaging/vision benchmarks

Relative numbers presented are in comparison with Tensilica Vision P5 DSP cadence®