KALRAY

Deep Learning Inference on the MPPA®3 Manycore Processor

Benoît Dupont de Dinechin, CTO

EWC 2020

www.kalrayinc.com

Kalray at Glance

We offer a new type of processor targeting the booming market of intelligent systems

International Presence

France	Grenoble (HQ), Sophia-Antipolis,
USA	Los Altos, California
Japan	Yokohama

Breaktrough technology from **10** years of development

23 patent families

Financial and industrial investors

cea investissement · INOCAP Gestion · MBDA · Renault Nissan Mitsubishi · ACE · Pengpai · SAFRAN · bpifrance

EURONEXT

IPO in June 2018 **(ALKAL)**

KALRAY

Outline

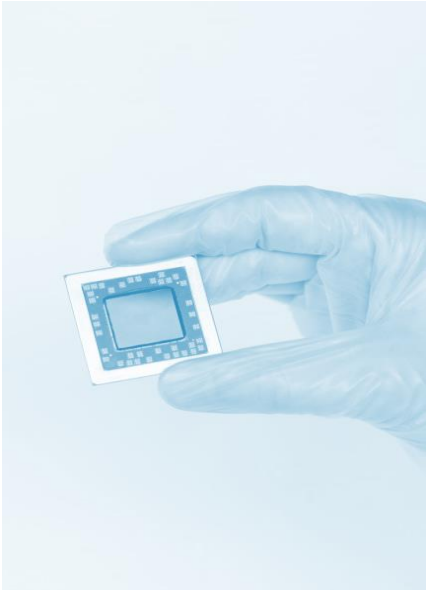
MPPA®3 Manycore Processor

Programming Environments

Deep Learning Inference

Applications & Outlook

Kalray's MPPA®



MPPA® (Massively Parallel Processor Array) Platform

Hardware

Manycore CPU architecture

Compute clusters of 16 high-performance CPU cores with local memory

DSP-like timing predictability

'Fully timing compositional' cores for accurate static timing analysis

Service guarantees of local memory system and network-on-chip

FPGA-like I/O capabilities

Software

CPU programming

Standard C/C++/OpenMP/OpenCL, OpenVX

Library code generators (MetaLibm, KaNN)

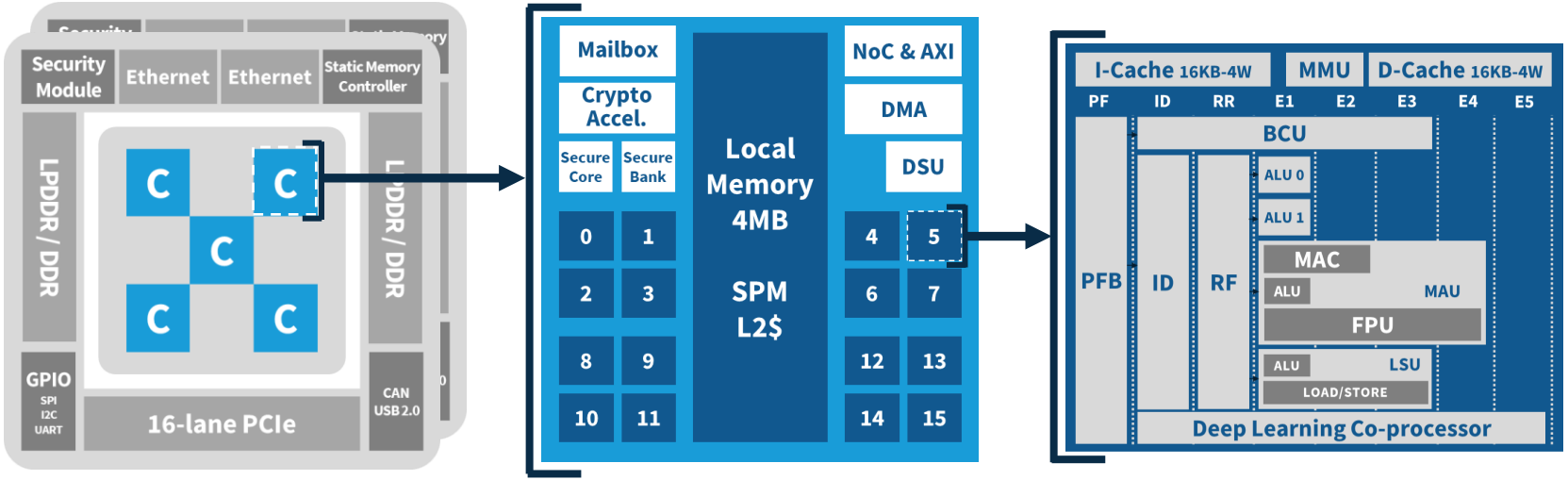
Model-based (SCADE Suite®, Simulink®)

MPPA® Processor Family and Roadmap

SAMPLES AVAILABILITY — 2019 — 2020 — 2021

	BOSTAN2	COOLIDGE1-80	COOLIDGE2 – 80 COOLIDGE2 – 160	DOLOMITES	
				DOLOMITES-HP	DOLOMITES-LC
PROCESS	28 nm	16 nm	16 nm	7nm	7nm
FIXED POINT OPERATIONS	1.3 TOPS	25 TOPS (8bit)	50 / 100 TOPS (8bit)	100 TOPS	10 TOPS
FLOATING POINT OPERATIONS	512 GFLOPS	3 TFLOPS (FP16)	3 / 6 TFLOPS (FP16&FP32)	10 TFLOPS	1 TFLOPS
CONSUMPTION	8 – 25W	5 – 20W	5 – 25W / 5 – 50W	5 – 20W	2-10W
FEATURES	• 288 Kalray VLIW Cores • 128 Crypto Copro • 2xDDR3 • 8x 1/10G GbE • 2xPCIe 8 lane Gen3	• 80 Kalray 64-bit cores • 80 Coprocessor for vision and learning • 2 x LP/DDR4 • 8x 1/10/25GbE • 16-lane PCIe Gen4	• 80/160 Kalray 64-bit cores • 80/160 Coprocessor for vision and learning • 2 x LP/DDR4 • 8x 1/10/25GbE • 16-lane PCIe Gen4	• High Performance • Funded by European Community	• Low cost optimized version
QUALIF/CERTIF	Industrial (-20/+85C)	• AEC-Q100 / QM	• ASIL B / ISO 26262	• ASIL B , ISO 26262	• ASIL D , ISO 26262
TARGET MARKET	• DATA CENTER • AUTO (proto)	• DATA CENTER • AUTOMOTIVE	• DATA CENTER • AUTOMOTIVE	• DATA CENTER • AUTOMOTIVE	• DATA CENTER • AUTOMOTIVE
	AVAILABLE	DEMO @CES2020	UNDER DEVELOPMENT	UNDER DEFINITION	UNDER DEFINITION

MPPA3-80 Manycore Processor (TSMC 16FFC, 1.2GHz)



COOLIDGE PROCESSOR

5 compute clusters at 1200 MHz
2x 100Gbps Ethernet, 16x PCIe Gen4

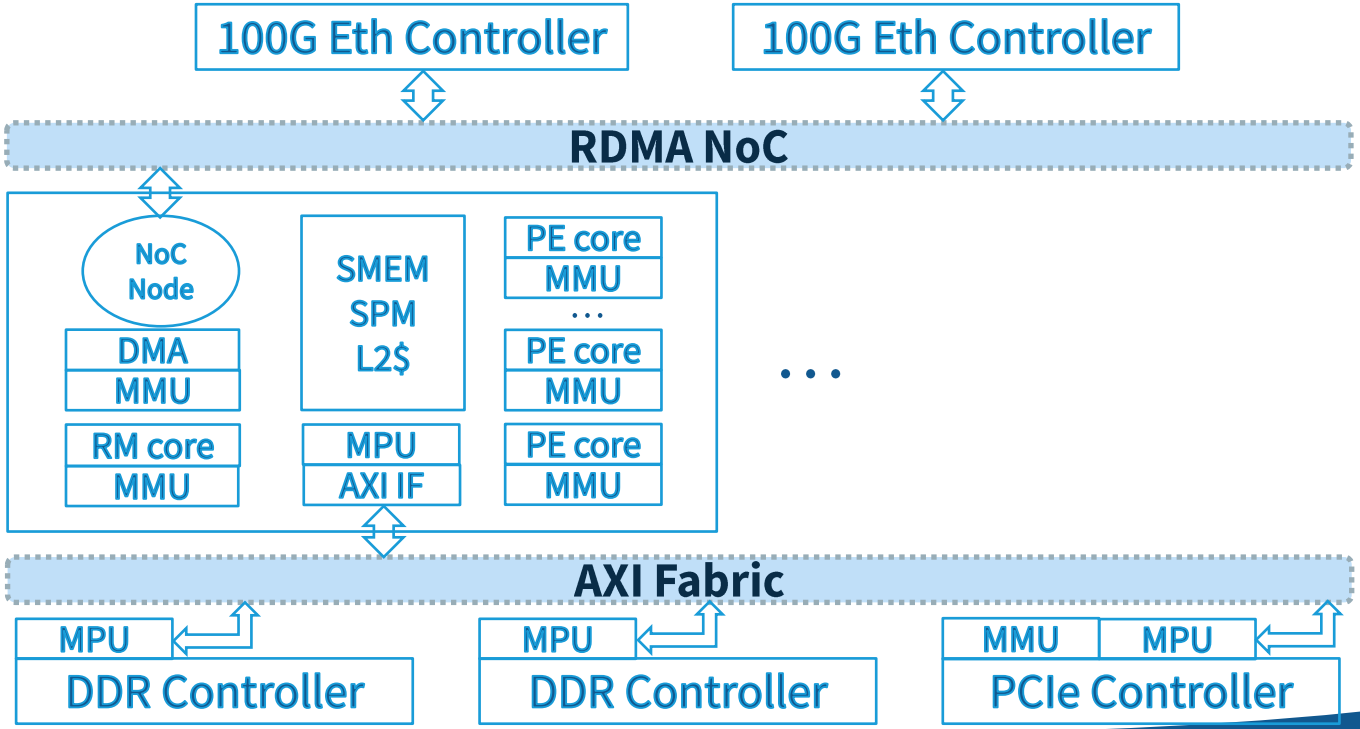
COMPUTE CLUSTER

16+1 cores, 4 MB local memory
NoC and AXI global interconnects

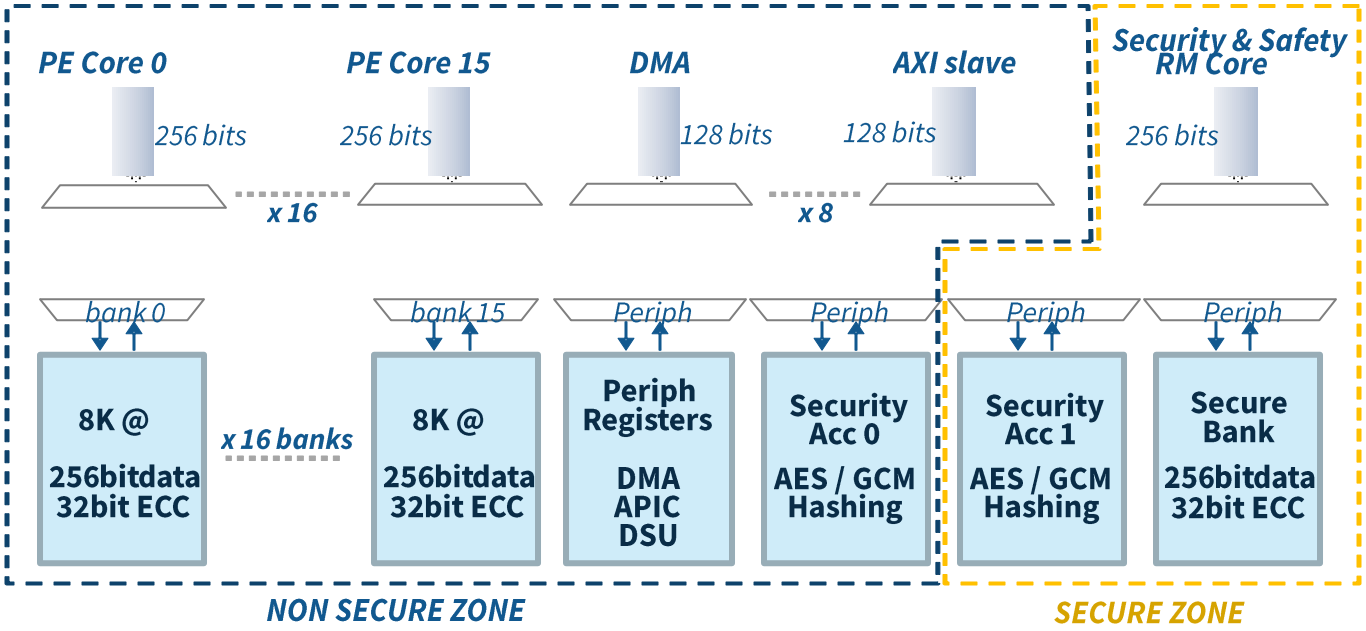
6-ISSUE VLIW CORE

64x 64-bit register file
128MAC/c tensor coprocessor

MPPA3 Global Interconnects



MPPA3 Compute Cluster



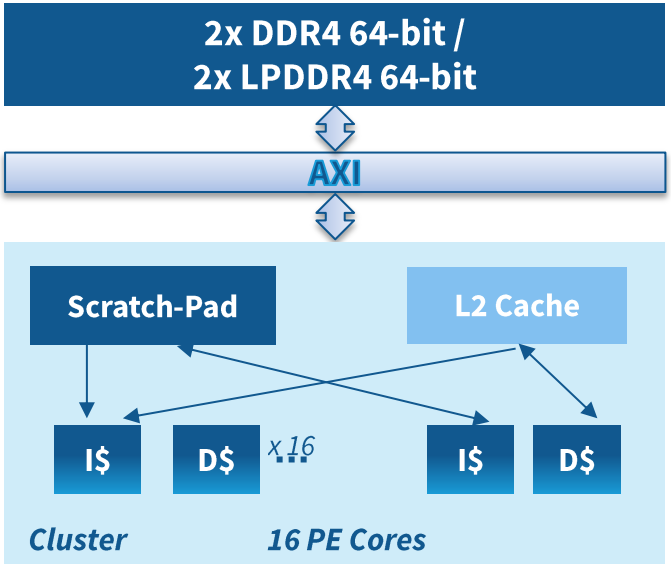
MPPA3 Memory Hierarchy

VLIW Core L1 Caches

- 16KB / 4-way LRU instruction cache per core
- 16KB / 4-way LRU data cache per core
- 64B cache line size
- Write-through, write no-allocate (write around)
- Coherency configurable across all L1 data caches

Cluster L2 Cache & Scratch-Pad Memory

- **Scratch-pad from 2MB to 4MB**
  - **16 independent banks, full crossbar**
  - **Interleaved or banked address mapping**
- L2 cache from 0MB to 2MB
  - 16-way Set Associative
  - 256B cache line size
  - Write-back, write allocate



L1 cache coherency	L2 cache coherency
enable /disable	enable /disable

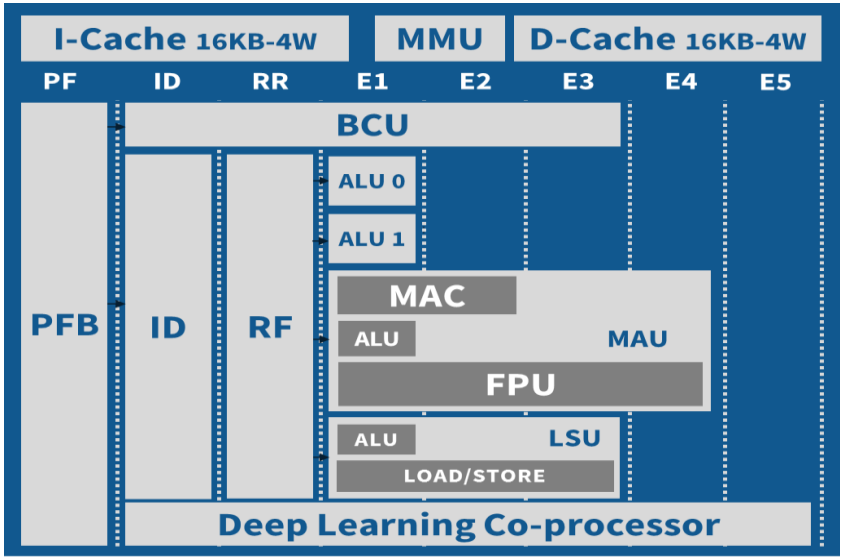
MPPA3 64-Bit VLIW Core

Vector-scalar ISA

- 64x 64-bit general-purpose registers
- Operands can be single registers, register pairs (128-bit) or register quadruples (256-bit)
- Immediate operands up to 64-bit, including F.P.
- 128-bit SIMD instructions by dual-issuing 64-bit on the two ALUS or by using the FPU datapath

FPU capabilities

- 64-bit x 64-bit + 128-bit → 128-bit
- 128-bit op 128-bit → 128-bit
- FP16x4 SIMD 16 x 16 + 32 → 32
- FP32x2 FMA, FP32x4 FADD, FP32 FMUL Complex
- FP32 Matrix Multiply 2x2 Accumulate



K1C VLIW CORE PIPELINE

MPPA3 Tensor Coprocessor

Extend VLIW core ISA with extra issue lanes

- Separate 48x 256-bit wide vector register file
- Matrix-oriented arithmetic operations (CNN, CV …)

Full integration into core instruction pipeline

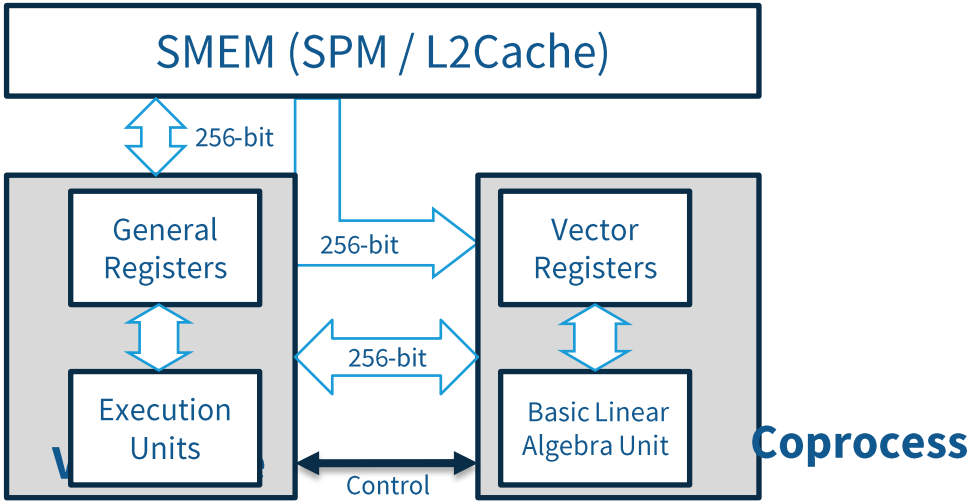
- Move instructions supporting matrix-transpose
- Proper dependency / cancel management

Leverage MPPA memory hierarchy

- SPM / L2$ directly accessible from coprocessor
- Memory load stream aligment operations

Arithmetic performances

- 128x INT8→INT32 MAC/cycle
- 64x INT16→INT64 MAC/cycle
- 16x FP16→FP32 FMA/cycle

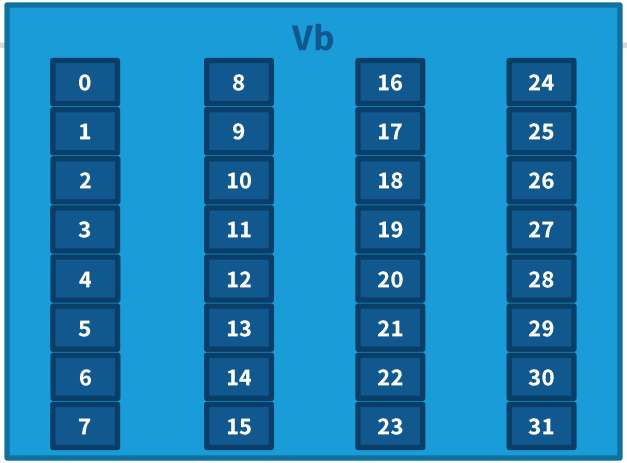


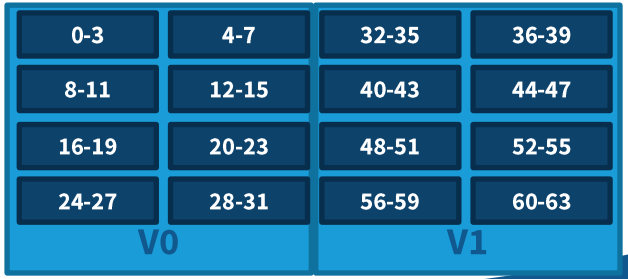
INT8.32 Matrix Multiply-Accumulate

Operand Va is a 4x8 submatrix of a row-major order matrix in memory (activations)

Operand Vb is a 8x4 submatrix of a column-major order matrix in memory (weights)

Result is a 4x4 submatrix spanning two registers V0 and V1



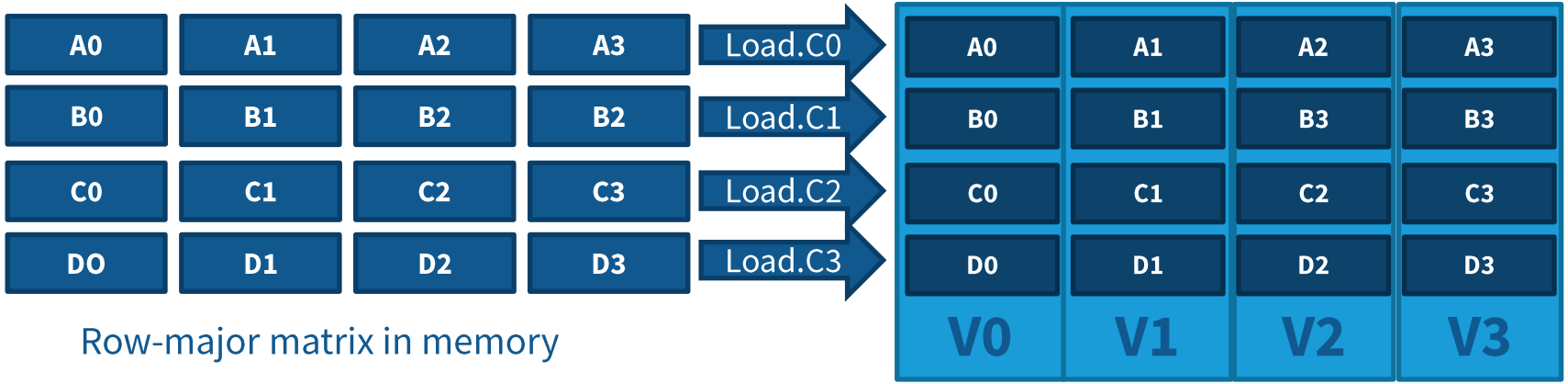


Numbers indicate byte index in 32-byte coprocessor registers

Load-Scatter Memory Operations

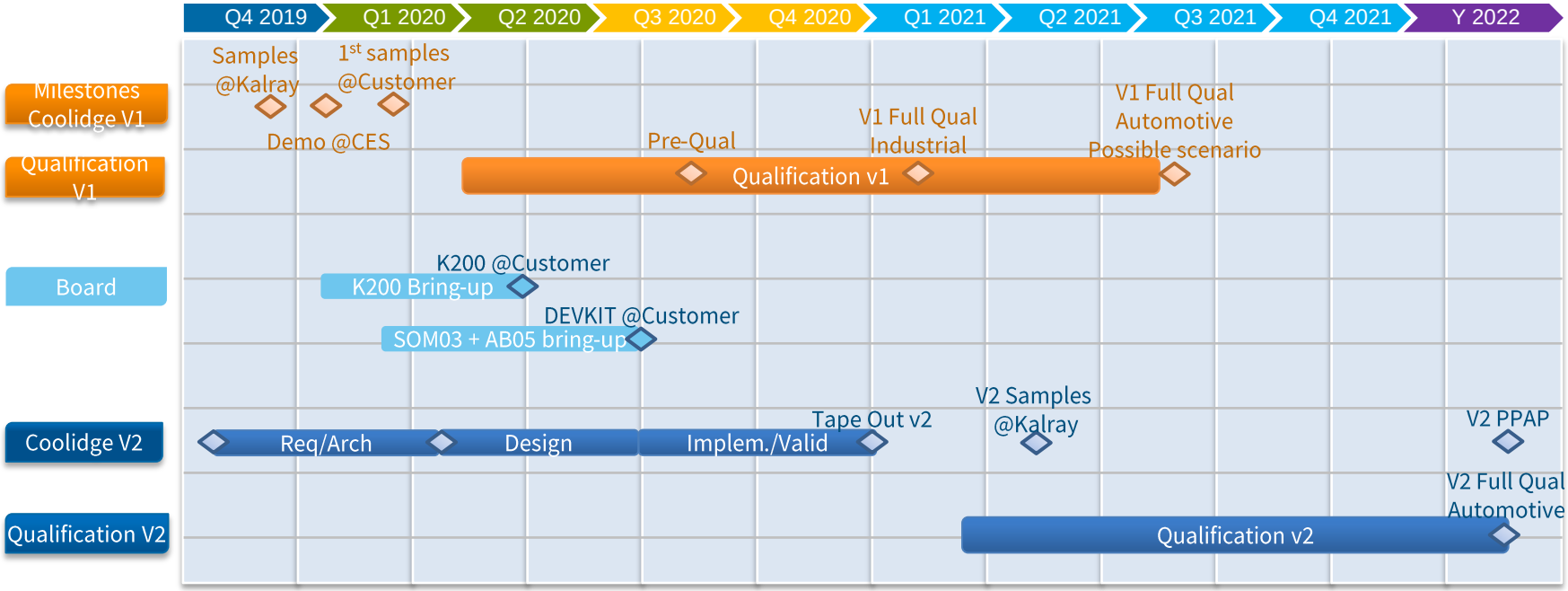
Support the key invariant that any coprocessor operand (1, 2 or 4 registers) is interpreted as a submatrix with four rows and a variable number of columns

Avoids the complexities of Morton memory indexing (Z-patterns for memory data layout)



Row-major matrix in memory

MPPA Coolidge – Go to Market Schedule



Outline

MPPA®3 Manycore Processor

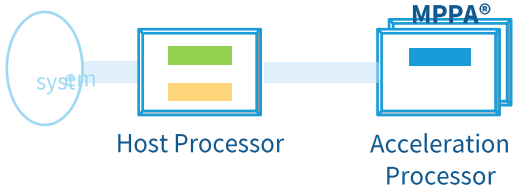
Programming Environments

Deep Learning Inference

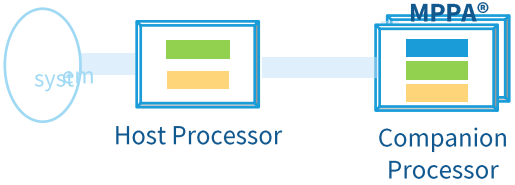
Applications & Outlook

MPPA® Platform Configurations

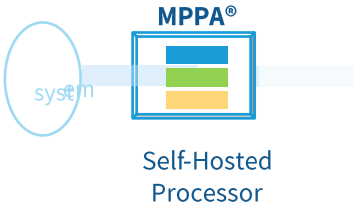
MPPA® Accelerator



MPPA® Companion



MPPA® Stand-alone



KALRAY

AccessCore™ SDK Programming Models

STANDARD PROGRAMMING ENVIRONMENTS

OPENCL 1.2 Programming

Standard accelerator programming model for offloading on MPPA®

- POSIX host CPU accelerated by MPPA device (OpenAMP interface)
- OpenCL 1.2 compatibility with POCL environment and LLVM for OpenCL-C
- OpenCL offloading modes:
  - Linearized Work Items on a PE (LWI)
  - Single Program Multiple Data (SPMD)
  - Native code called from kernels

C/C++ POSIX Threads Programming

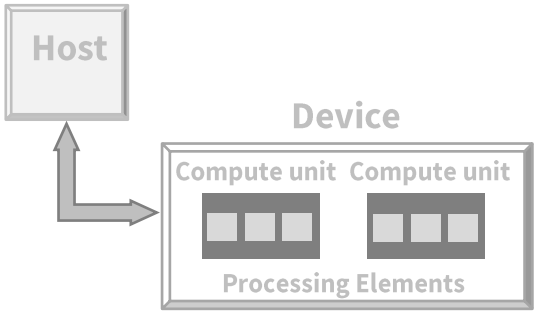
Standard multicore programming model with exposed MPPA® communications

- Linux and ClusterOS
- Standard C/C++ programming
  - GCC, GDB, Eclipse system trace
- POSIX threads interface
- GCC OpenMP support
- RDMA using the MPPA Asynchronous Communication library (mppa_async)

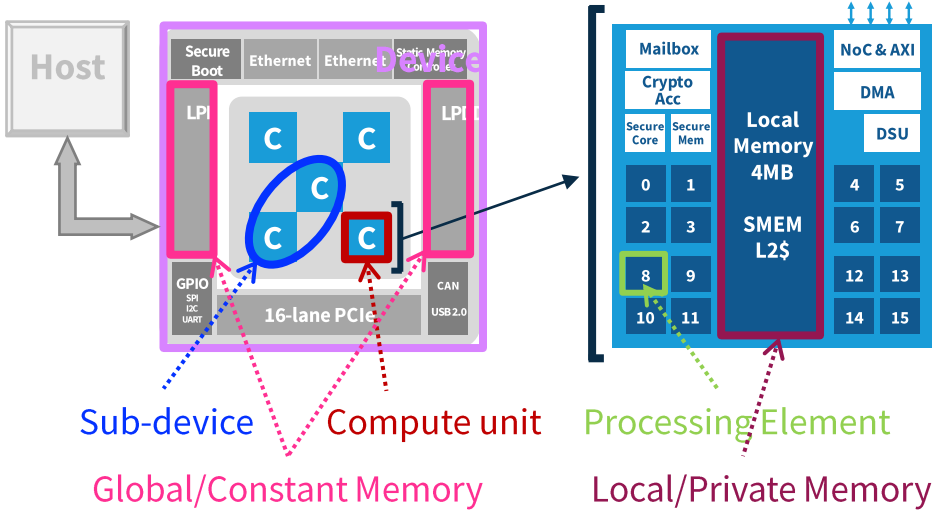
OpenCL Compute Platform Mapping

OpenCL Compute Platform Model

Topology: Host CPU connected to one or several Device(s)
Host: CPU which runs the application under a rich OS (Linux)
Device: Compute Unit(s) sharing a Global Memory
Hierarchy: Device => Sub-Device => Compute Unit(s) => Processing Elements



‘SPMD’ Mapping to MPPA® Architecture



OpenCL Native Function Extension

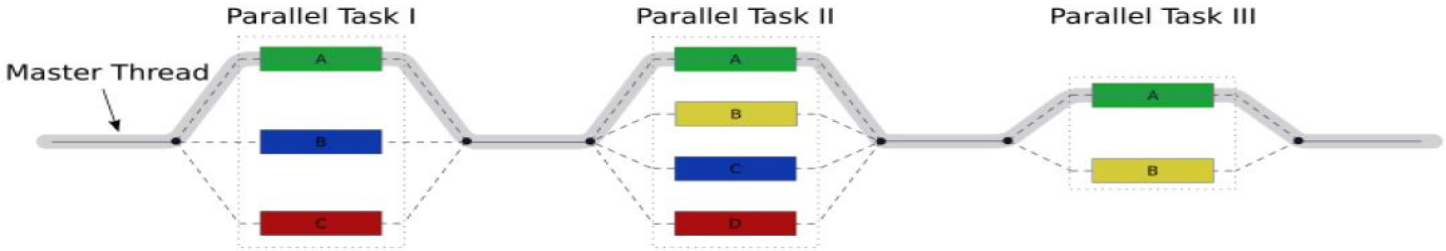
- Enable to call ASM, C/C++/OpenMP/POSIX (ClusterOS) code from OpenCL kernels
- Generalization of TI 'OpenMP Dispatch With OpenCL' for KeyStone-II platforms
- Used by Kalray KaNN deep learning compiler

```
void
my_vector_add(int *a, int *b, int *c, int n)
{
  #pragma omp parallel for
  for (int i = 0; i < n; ++i)
  {
    c[i] = a[i] + b[i];
  }
}
```

```
__attribute__((mppa_native))
void my_vector_add(__global int *a, __global int *b, __global int *c, int n);

__kernel void vector_add(__global int *a, __global int *b, __global int *c, int n) {
  my_vector_add(a, b, c, n);
}
```

OpenMP for Multicore Programming



A parallel region starts redundant execution

Work sharing constructs assign different pieces of work to threads

Synchronization is explicit (here critical section) or implicit (barriers end constructs)

```
/* Create a team of threads and scope variables */
#pragma omp parallel shared(A,b,c,total) private(tid,i)
  {
  tid = omp_get_thread_num();
  /* Loop work-sharing construct - distribute rows of matrix */
  #pragma omp for private(j)
  for (i=0; i < SIZE; i++)
    {
    for (j=0; j < SIZE; j++)
      c[i] += (A[i][j] * b[i]);
      #pragma omp critical
      {
        total = total + c[i];
      }
    }   /* end of parallel i loop */
  } /* end of parallel construct */
```

GCC Support of Kalray VLIW Core

```
void
vector_add(long n, float a[], float b[], float c[restrict])
{
  for (long i = 0; i < n; i++) {
    c[i] = a[i] + b[i];
  }
}

/*

k1-elf-gcc -O2 -ftree-vectorize -S vector_add.c -fopt-info-vec-all

vector_add.c:4:3: note: ------>vectorizing statement: i_17 = i_20 + 1;
vector_add.c:4:3: note: ------>vectorizing statement: vectp_a.7_43 = vectp_a
vector_add.c:4:3: note: ------>vectorizing statement: vectp_b.10_46 = vectp_
vector_add.c:4:3: note: ------>vectorizing statement: vectp_c.14_50 = vectp_
vector_add.c:4:3: note: ------>vectorizing statement: if (n_12(D) > i_17)

loop at vector_add.c:5: if (ivtmp_53 < bnd.4_38)
vector_add.c:4:3: note: LOOP VECTORIZED

vector_add.c:2:1: note: vectorized 1 loops in function.

*/
```

automatically vectorized loop

```
        .align 8
        .global vector_add
        .type        vector_add, @function
vector_add:
        addd $r4 = $r0, -1
        cb.dlez $r0? .L1
        ;;
        srld $r5 = $r0, 3
        compd.leu $r4 = $r4, 6
        ;;
        cmoved.deqz $r5? $r5 = 1
        cb.dnez $r4? .L7
        ;;
        loopdo $r5, .L15
        ;;
.L4:
        lo.xs $r8r9r10r11 = $r4[$r1]
        ;;
        lo.xs $r32r33r34r35 = $r4[$r2]
        ;;
        faddwq $r8r9 = $r8r9, $r32r33
        ;;
        faddwq $r10r11 = $r10r11, $r34r35
        ;;
        so.xs $r4[$r3] = $r8r9r10r11
        addd $r4 = $r4, 1
        ;;
        # HW loop end
```

Outline

MPPA®3 Manycore Processor

Programming Environments

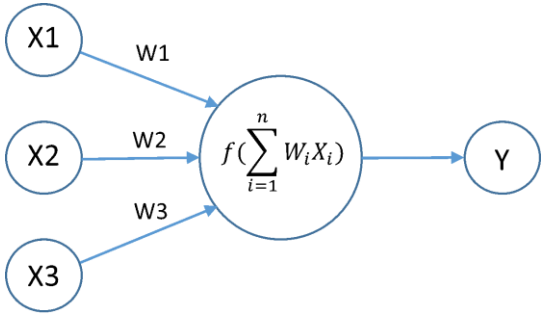
Deep Learning Inference

Applications & Outlook

Machine Learning (ML)

Give computers the ability to learn without being explicitly programmed (Arthur Samuel, 1959)

- **Rule Extraction** Goal is to identify statistical relationships in data
- **Clustering** Group similar data together, while increasing the gap between the groups
- **Classification & Regression** Map a set of new input data to a set of discrete or continuous valued output, respectively
- **Artificial Neural Networks (ANN)** General implementation model for nonlinear classifiers, trained by using back-propagation algorithms



Weighted sum of inputs

$$Y = f\big((W_1 \times X_1) + (W_2 \times X_2) + (W_3 \times X_3)\big)$$

$f(\,)$ is the "activation function"

$f(x) = \max(0, x)$, "Rectified Linear Unit"

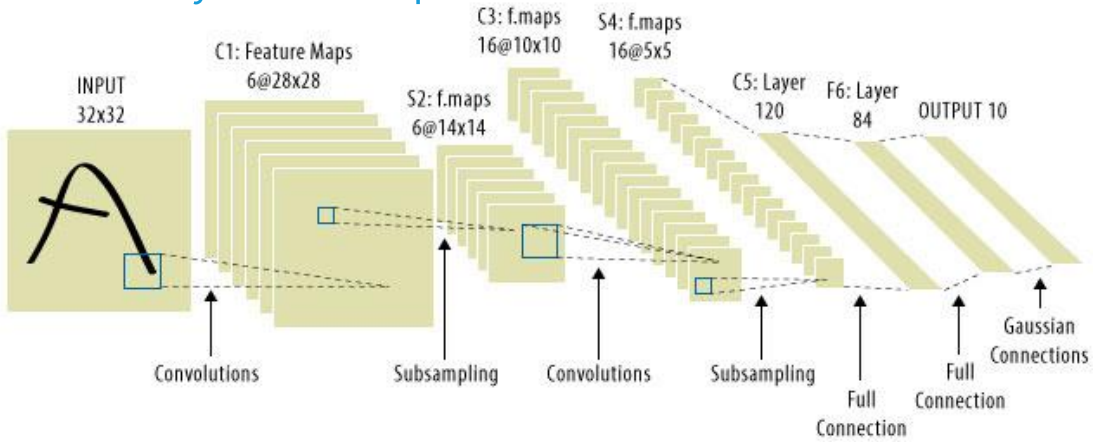
$f(x) = \tanh(x)$

…

Deep Learning (DL)

Computational models composed of multiple processing layers to learn representations of data with multiple levels of abstraction (Yann Le Cun et al., 2015)

- **Convolutional Neural Networks (CNN)** Networks where most filtering operations performed by feature maps are discrete convolutions



- **Recurrent Neural Networks (RNN)** Networks with feedback loops

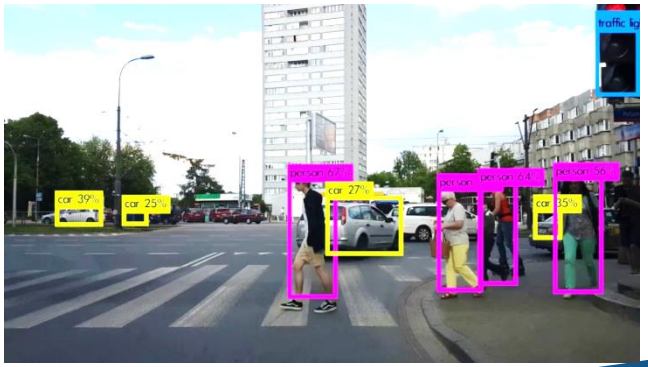
Machine Learning Steps

Training (datacenter)

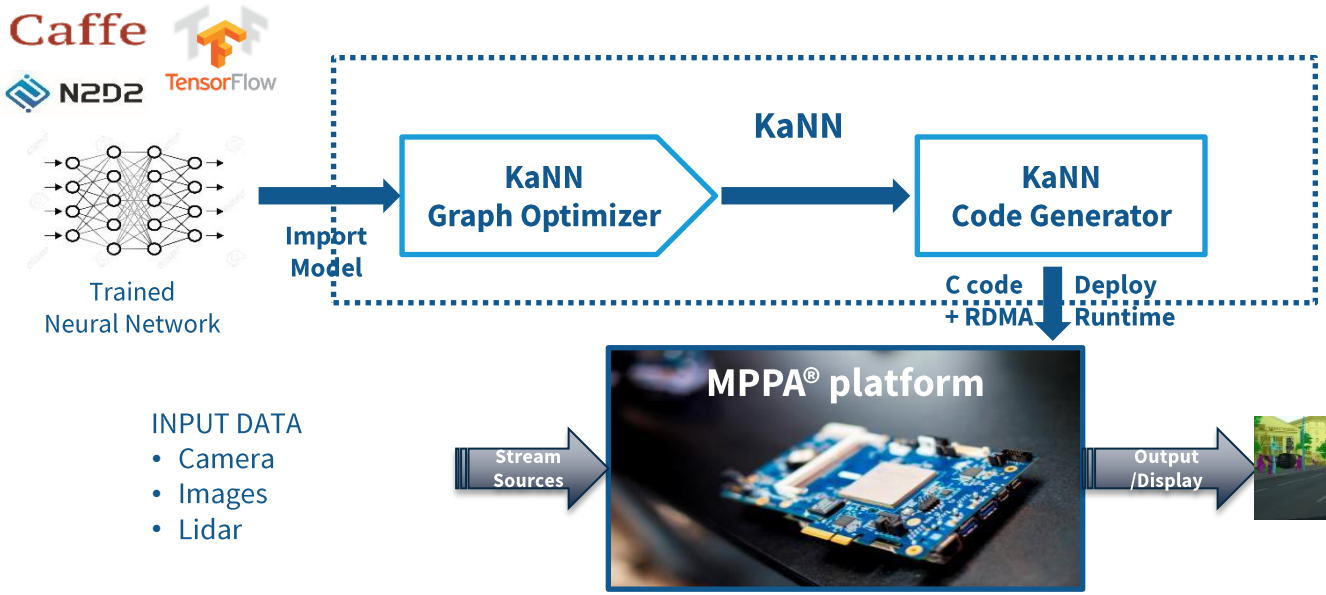
- Learning part or Machine Learning
  - Supervised (classification & regression)
  - Unsupervised (clustering)
  - Reinforcement (decision-making)
- Off-line processing of large data sets
- Floating-point 32-bit arithmetic

Inference (deployed system)

- Classification / Segmentation / Detection
- On-line / Real-time data stream processing
- Floating-point 16.32-bit arithmetic (FP16, BF16)
- Integer 8.32 arithmetic for CNN (quantization)



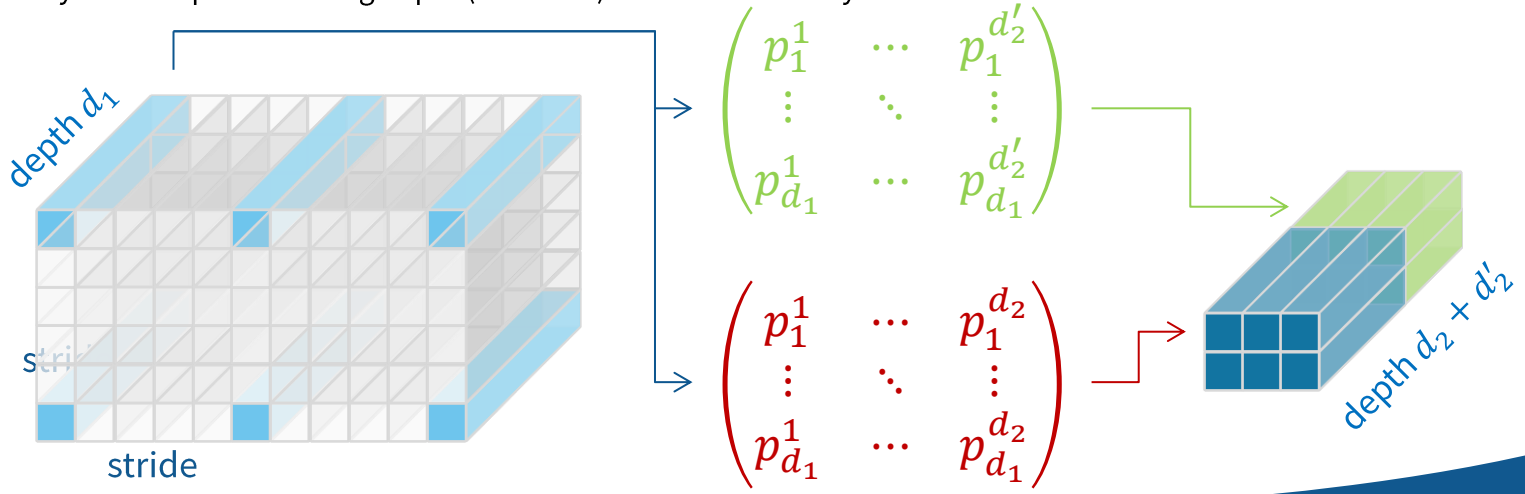
KaNN (Kalray Neural Network) Inference Compiler



CNN Inference on a MPPA Processor (1)

Execute one DNN operator at a time in topological sort order of the computation graph

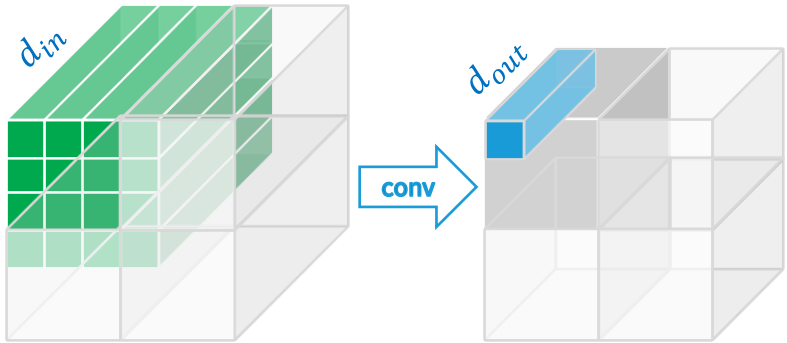
- Computation graph is the neural network after operator fusion and data layout optimizations
- Each operator execution is distributed onto a set of compute clusters identified as an OpenCL sub-device
- Activation layout is sequential along depth (channels) for dense memory accesses



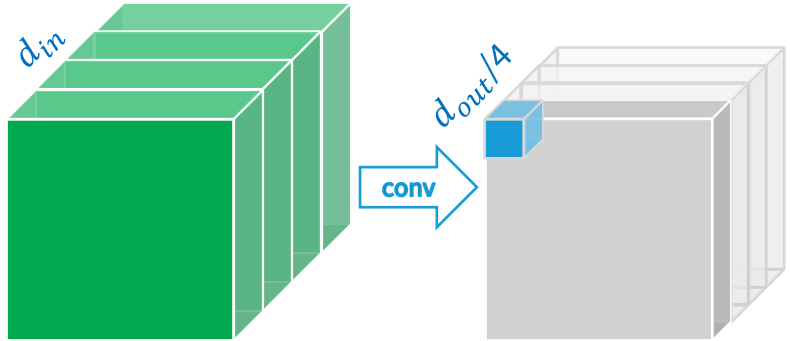
CNN Inference on a MPPA Processor (2)

Distribute activations across cluster SPMs, splitting along spatial and/or depth dimensions

- Spatial dimension splitting requires that the full set of parameters be loaded from external memory
- Channel dimension splitting requires access to the whole input activations and a subset of the parameters
- RDMA NoC multicasting of parameters from external memory in case of spatial dimension splitting



$[3][3][d_{in}][d_{out}]$

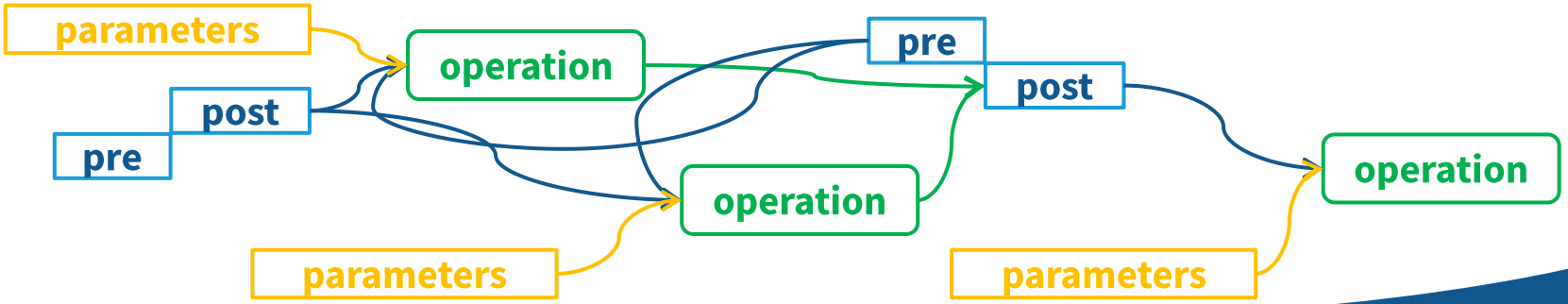


$[3][3][d_{in}][d_{out}/4]$

CNN Inference on a MPPA Processor (3)

Code generation is driven by the (optimized) computation graph

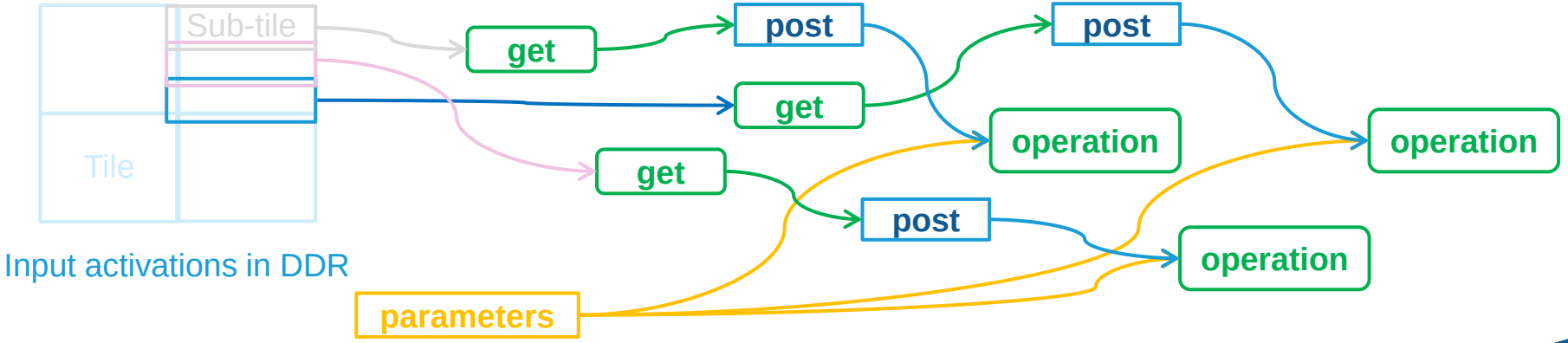
- Build a local memory buffer allocation and task execution schedule in each cluster
- Overlap parameter transfers from external memory with computations on local memory
- Allocation and scheduling are performed on the computation graph
  - activation lifetime corresponds to pre and post tasks,
  - layer compute operations corresponds to a malleable task
  - pre tasks load biases from external memory into the local memory buffer



CNN Inference on a MPPA Processor (4)

For layers where activations do not fit on-chip, stream sub-tiles from DDR memory

- All clusters remote write their tile of output activations to DDR memory, then enter a synchronization barrier
- After clusters leave the barrier, they pipeline the remote read from DDR / operate / put to DDR of sub-tiles
- Larger sub-tiles factor more control overhead but reduce the amount of pipelining



Outline

MPPA®3 Manycore Processor

Programming Environments

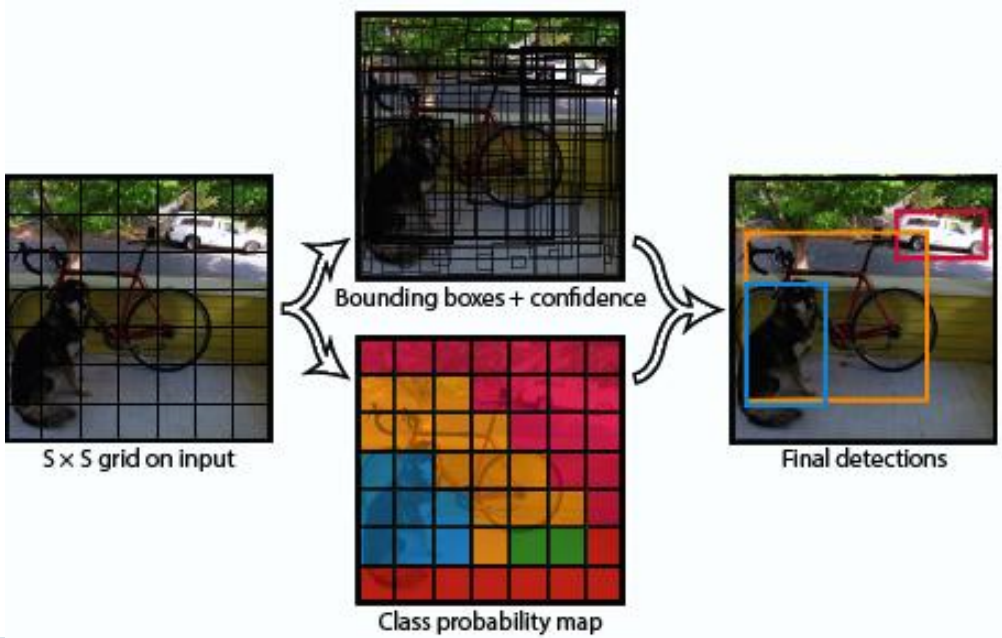
Deep Learning Inference

Applications & Outlook

YOLO v1-v3 « You Only Look Once » (Redmon 2016-2018)

Single-step (unlike « R-CNN » family)

- A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes
- Yolo v3 inference on 416x416 images requires 65.9 10^9 operations when using DarkNet-53 as backbone network
- 35 FPS on a NVIDIA Pascal Titan X GPU (16nm CMOS), 18 FPS on NVIDIA Jetson Xavier



MPPA® Deep Learning Inference
Object detection CNN

FPS – Batch 1	Precision	Faster-RCNN (VGG16)	SSD (MobileNet)	Tiny-YOLO v3	YOLO v3 (416x416)
MPPA Coolidge80 v1 @1.2 Ghz (2)	INT8 INT16 FP16	**150** **75** **18**	**600** **300** **75**	**2000** **1000** **250**	**140** **70** **20**
MPPA Coolidge80 v2 @1.2 Ghz (2)	INT8 INT16 FP16	**300** **150** **18**	**1200** **600** **75**	**4000** **2000** **250**	**280** **140** **20**
MPPA Bostan @500 Mhz (1)	INT8 INT16 FP16/FP32	- - **3**	- - **13**	- - **44**	- - **6.5**
NVIDIA Tegra TX2 (1)	FP32	6.8			3.7
NVIDIA Xavier developer.nvidia.com	FP16				18

(1) Measurements of computing on MPPA®
(2) Conservative performance estimation

KALRAY

Batch-1 Inference FPS / Watt[(1)]

			GoogleNet	ResNet50			
Provider	**Product**	**Overall Power**	**Fps/watt**	**Fps/watt**	**Thermal**	**PCIe Interface**	**DDR Capacity (GB)**
NVIDIA Jetson (Inference/Device)	Xavier Xavier (2020)	27 W 29 W	38 79	21 48	Passive Passive	Gen4 Gen4	8/16 8/16
NVIDIA (Training)	T4 V100	70 W 107 W	25 15	17 9	Passive .*	Gen3 *	8/16 *
Xilinx (Inference)	Alveo 200 Alveo 250	89 W 112 W	35 37	* *	Pass./Active Pass./Active	Gen3+Gen4 Gen3+Gen4	* *
Kalray (Inference)	MPPA3 v1 80 MPPA3 v2 160	40 W 90 W	70 112	43 68	Passive Passive	Gen4 Gen4	4/8/16/32 4/8/16/32

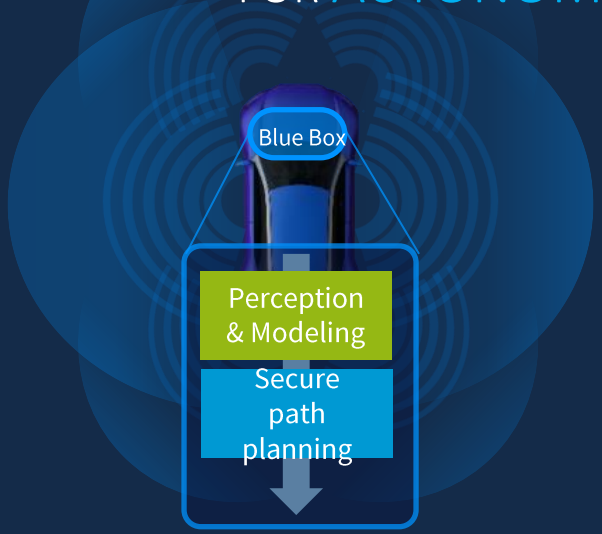
(1) Kalray estimates based on internal data as well as data disclosed publicly

* Missing information or information not confirmed

KALRAY

NXP Partnership: Collaboration with a Global Leader

CENTRAL COMPUTING SOLUTION FOR AUTONOMOUS VEHICLES



KALRAY
Perception & Modeling

NXP
Secure path planning

- For Level 3 vehicles
- Roadmap up to Level 5

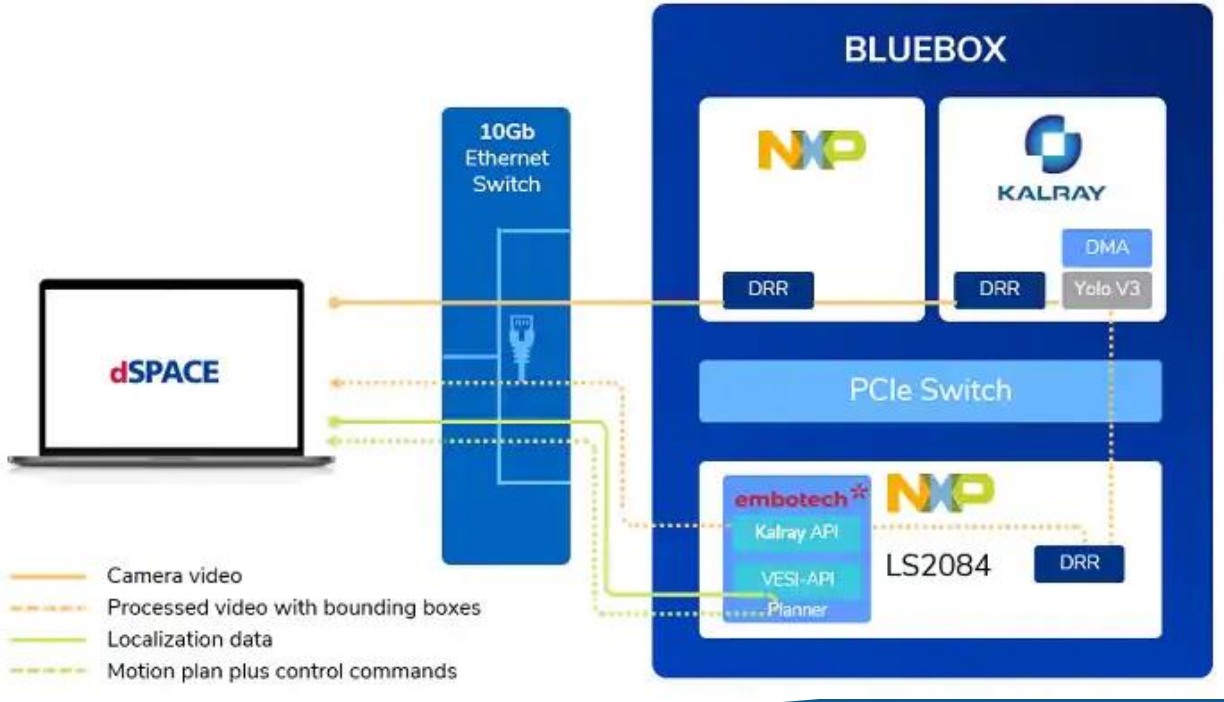
NXP #1 in automotive semiconductors (11% market share[1])
#2 in automotive processors (28% market share[2])

1 Gartner 2017 market share
2 NXP 2017 market share

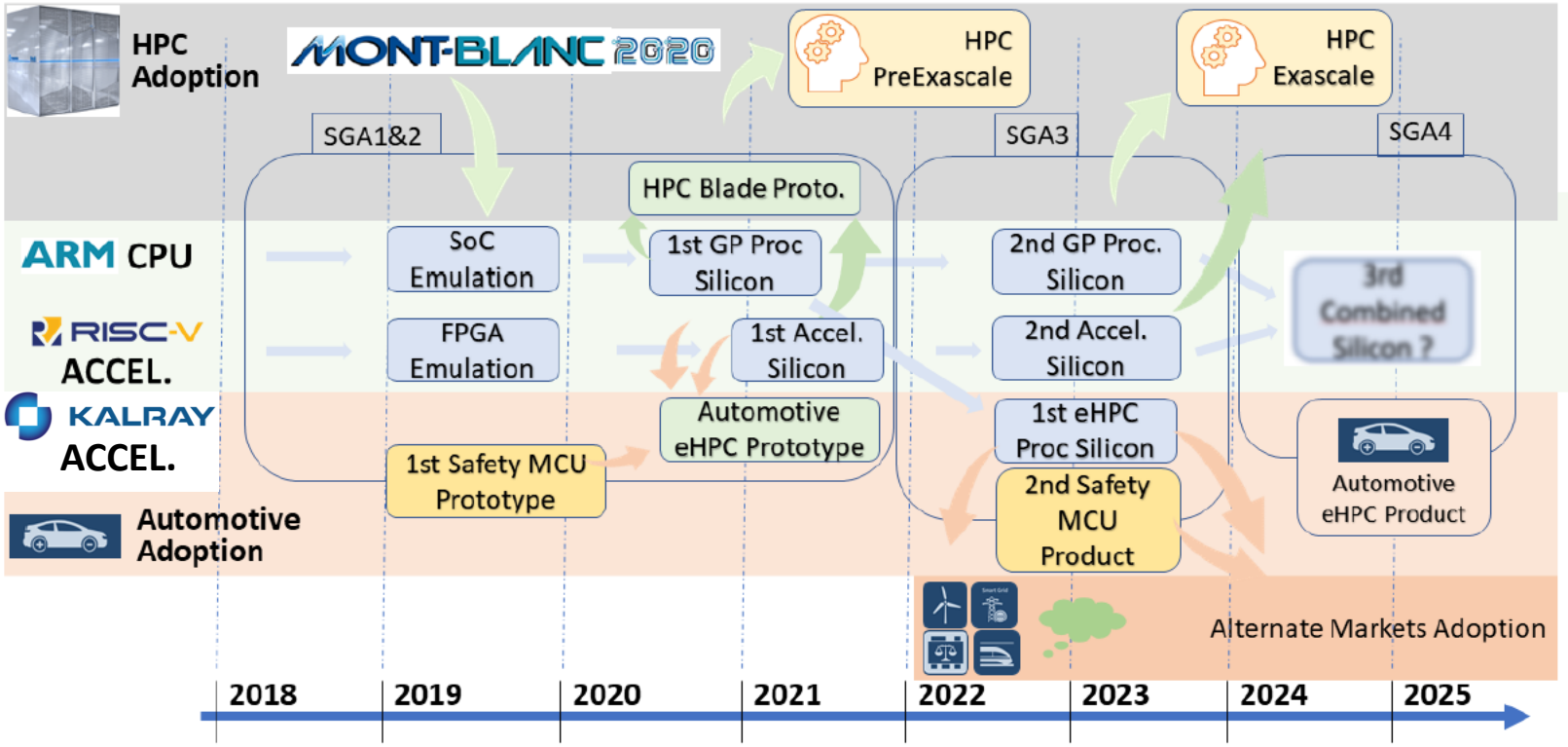
KALRAY

CES 2020 NXP Demonstration

- NXP BlueBox 2nd generation Autonomous Driving Development platform with production ready automotive silicon
- Kalray Coolidge 3rd Generation MPPA Perception Accelerator and AI Software
- Embotech Forces Pro and ProCruiser Real-time optimal control software and Highway planner solution
- dSPACE ASM Traffic Real time simulation environment with traffic, sensor simulation, full VD and BEV powertrain.



Mont-Blanc 2020 and EPI Projects



See Us 3A-303

KALRAY S.A. - GRENOBLE - FRANCE
180 avenue de l'Europe,
38 330 Montbonnot - France
Tel: +33 (0)4 76 18 09 18
email: info@kalray.eu

KALRAY INC. - LOS ALTOS - USA
4962 El Camino Real
Los Altos, CA - USA
Tel: +1 (650) 469 3729
email: info@kalrayinc.com

KALRAY