KALRAY

Engineering a Manycore Processor for Accelerated Edge Computing

CPS & IOT 2021

Benoît Dupont de Dinechin, Kalray CTO

KALRAY IN A NUTSHELL

Kalray offers a new type of processor and solutions targeting the booming market of intelligent systems.

A Global Presence

- France (Grenoble, Sophia-Antipolis)
- USA (Los Altos, CA)
- Japan (Yokohama)
- Canada (Partner)
- China (Partner)
- South Korea (Partner)

Leader in Manycore Technology

3rd generation of MPPA® processor

~€85m R&D investment

30 Patent families

Industrial investors

NXP · RENAULT NISSAN MITSUBISHI · SAFRAN · MBDA

EURONEXT

- Public Company (ALKAL)
- Support from European Govts
- Working with 500 fortune companies

Outline

1. Edge Computing
2. Manycore Processors
3. MPPA Processors and IP
4. Accelerator Offloading
5. Applications & Outlook

Defining Edge Computing

Intel (https://www.intel.com/content/www/us/en/edge-computing)

What Is an Edge Device?

Edge computing solutions place Internet of Things (IoT) devices, gateways, and computing infrastructure as close as possible to the source of data

Types of Edge Devices

- Intelligent edge devices offer capabilities like onboard analytics or AI.
- Intelligent edge devices used in manufacturing may include vision-guided robots or industrial PCs
- Digital cockpit systems built into commercial vehicles can help support driver assistance

NVIDIA (https://blogs.nvidia.com/blog/2019/10/22/what-is-edge-computing/)

What Is Edge Computing?

Edge computing is the concept of capturing and processing data as close to the source of the data as possible via processors equipped with AI software

What Are the Benefits of Edge Computing?

- Reduced latency: bringing AI computing to where data is generated
- Improved security: the need to send sensitive data to the public cloud is decreased
- Greater range: edge computing processes data without internet access

Rise of Intelligent Systems

Cyber-physical systems

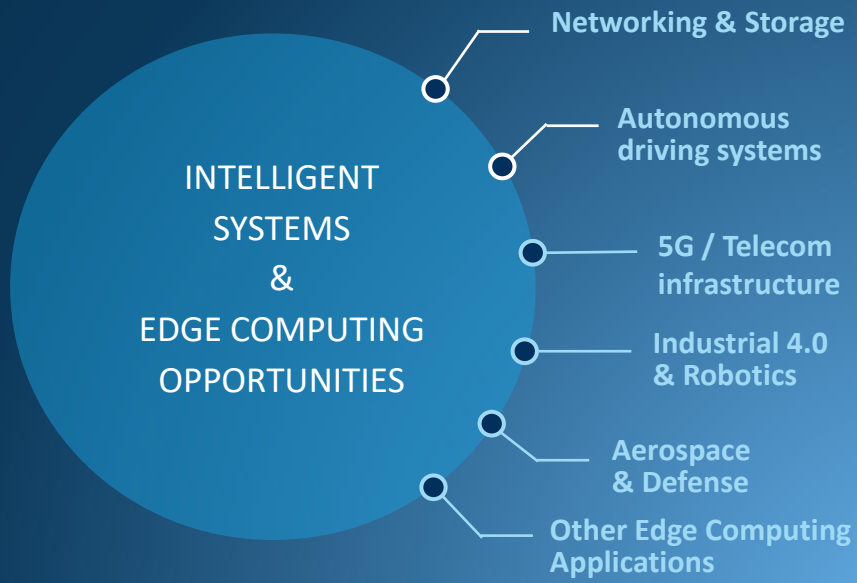
- Information processing and physical processes are tightly integrated
- Time constraints associated with information manipulation
- Functional safety and cyber-security
- Distributed systems (over Ethernet)

Artificial intelligence

- The science and engineering of creating intelligent machines (J. McCarthy, 1956)
- Mostly Machine Learning, in particular Deep Learning [Multiple processing layers to learn representations of data with multiple levels of abstraction -- Yann Le Cun et al., 2015]

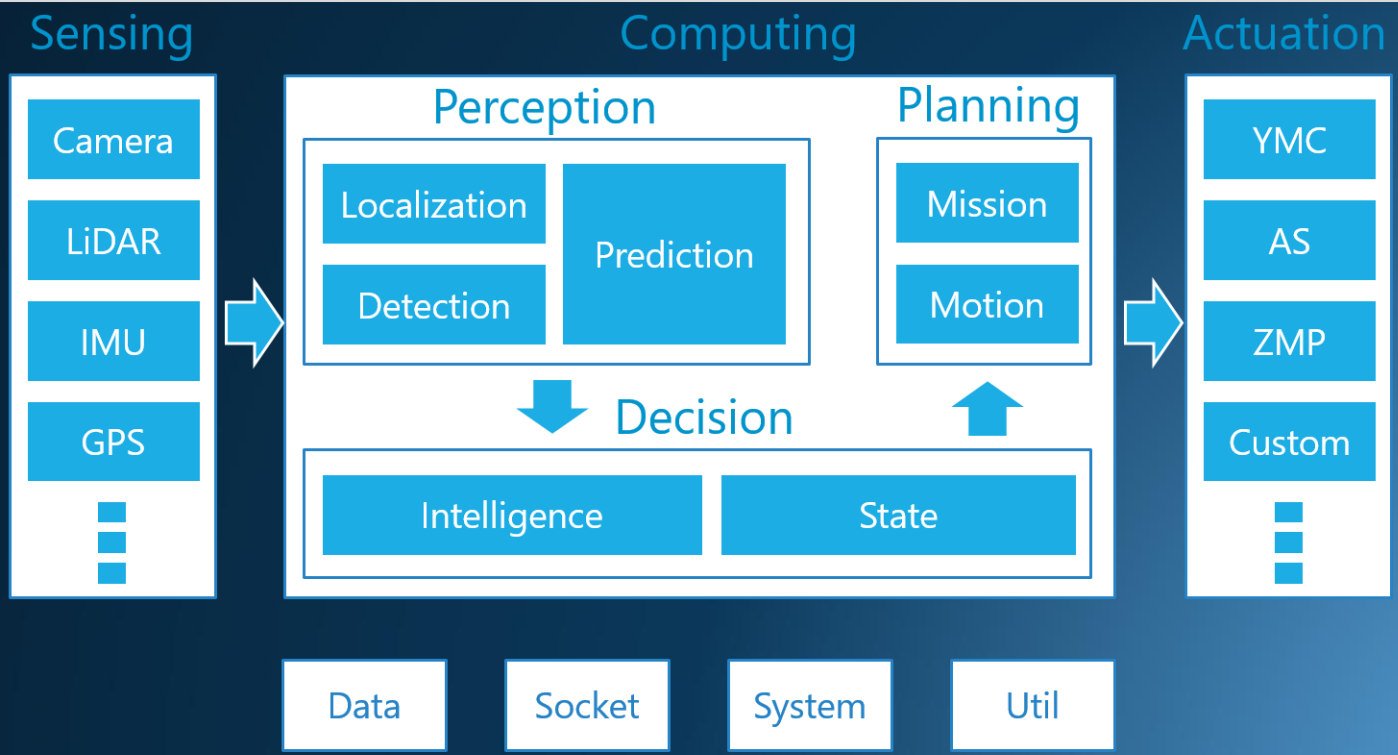
Intensive computing

- Image, signal, numerical, Galois fields, graphs



Autoware.Auto

[An open-source software stack based on ROS 2 for self-driving]



Autonomous Driving Systems Software Environment

POSIX PSE51 (AUTOSAR Adaptive)

- Pthreads, OpenMP, C++, std::thread, Boost::thread

Standard application libraries

- BLAS (BLIS), LaPACK, Eigen (machine learning and deep learning)
- PCL (Point Cloud Library), FLANN (Fast Library for Approximate Nearest Neighbors)
- OpenCV (legacy computer vision)

Standard programming frameworks

- CNN inference code generators for Berkeley Caffe, Google TensorFlow, Facebook ONNX, Khronos NNEF
- OpenCL (Khronos compute accelerator offloading standard)
- OpenVX (Khronos graph-based computing for computational imaging, extended for CNN inference)

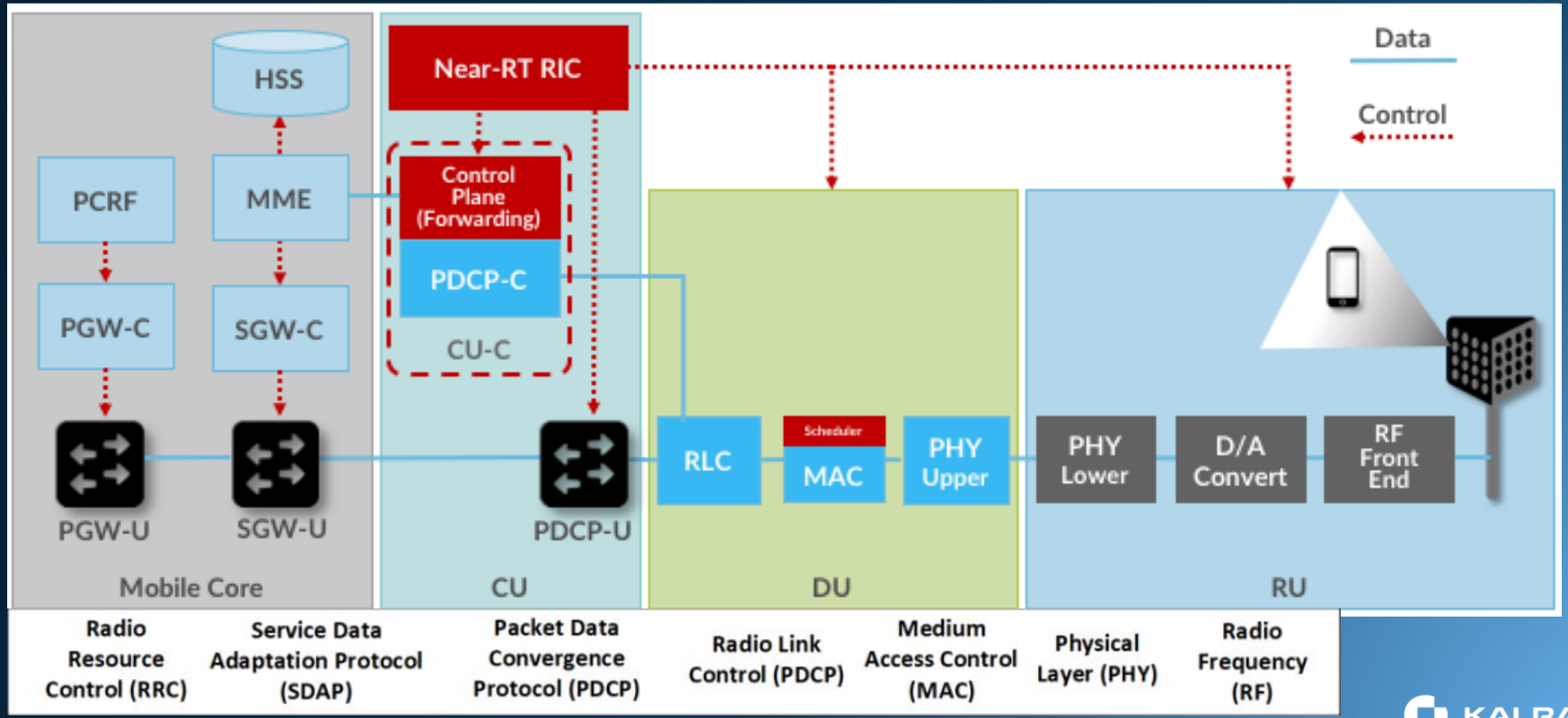
Standard communication middleware

- OMG Data Distribution Services (DDS), adopted since ROS 2.0 by the Open Source Robotics Foundation

KALRAY

Open Radio Access Network (O-RAN) for 5G

[5G Mobile Networks: A Systems Approach]



KALRAY

O-RAN Distributed Unit (DU) Acceleration

[O-RAN Cloud Platform Reference Designs]

DU system environment

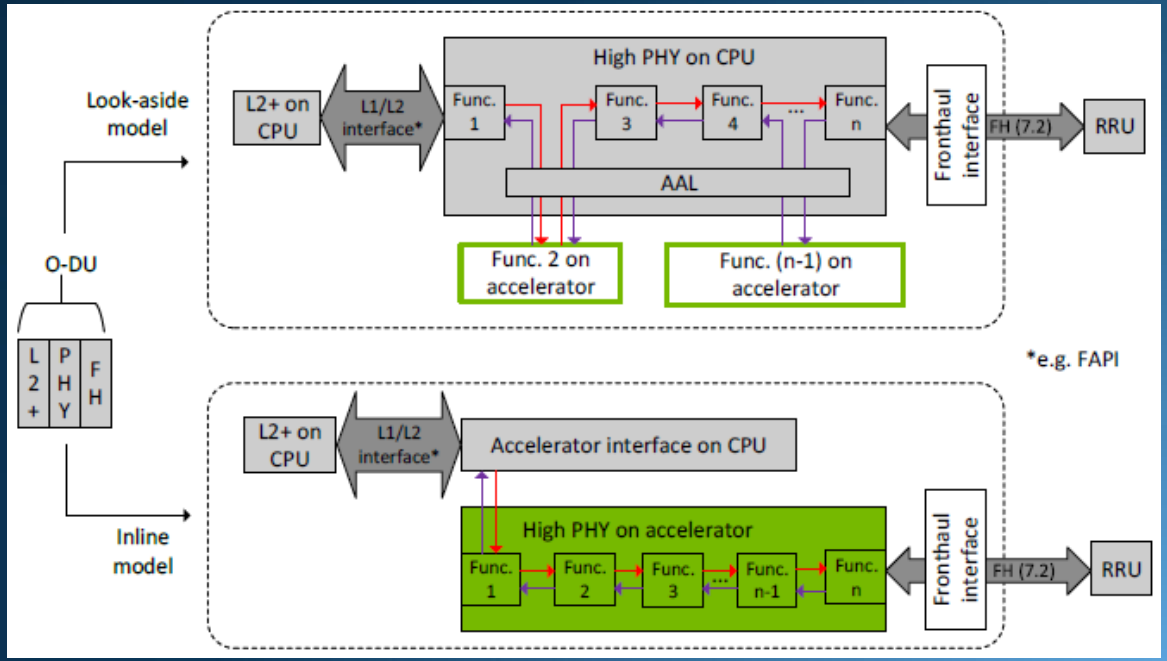
- Receives eCPRI frames from RU over Ethernet/PTP
- F1 interface between DU and CU over GTP-U/UDP/IP and SCPT/IP
- Acceleration Abstraction Layer

Look-Aside acceleration

- Typically FPGA or ASIC

Inline acceleration

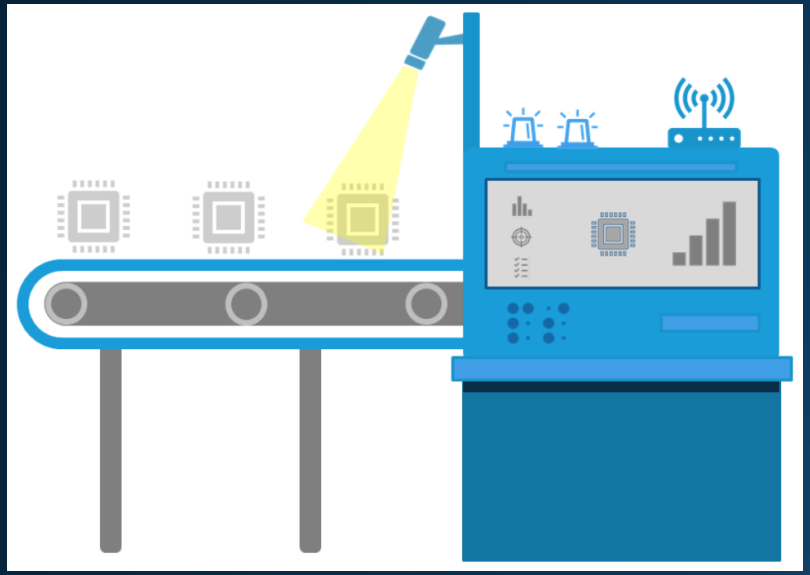
- Software-programmable accelerator processes from Upper-PHY to Lower-MAC (HARQ)



Machine Vision for Industry 4.0

Computer vision used in industrial IoT (IIoT)

- Defect detection on production line in automated factories
- Intelligent system embedded within a camera or using a frame grabber or centralized in an edge data center



Outline

1. Edge Computing
2. Manycore Processors
3. MPPA Processors and IP
4. Accelerator Offloading
5. Applications & Outlook

Homogeneous Multicore Processor

Multiple CPU cores integrated on a single circuit, sharing a cache-coherent memory hierarchy

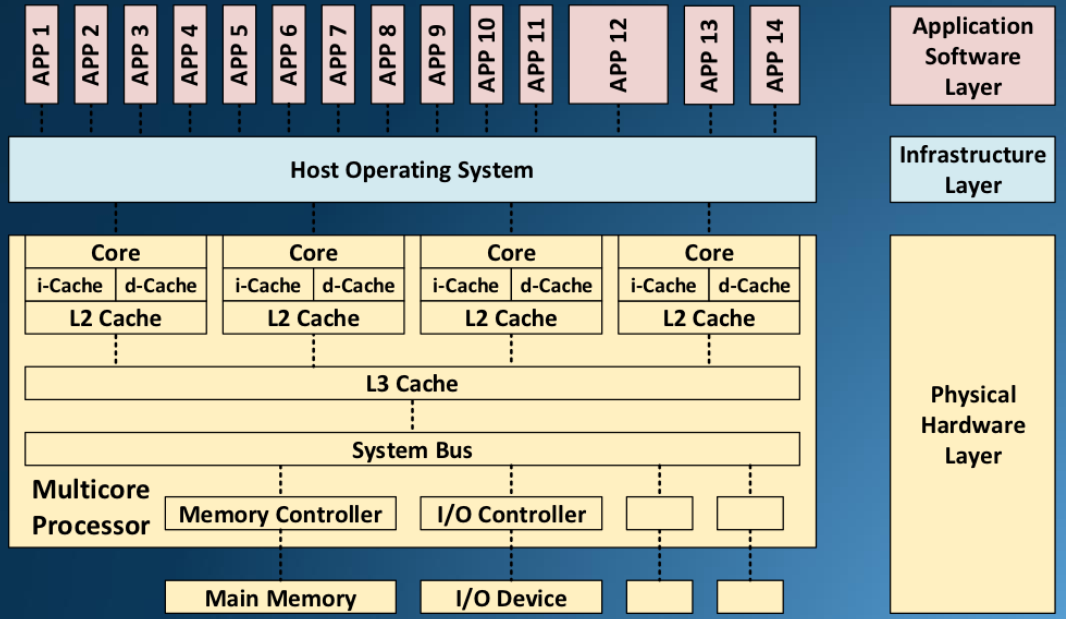
- Scalability by replicating CPU cores
- Programming models based on multi-threading

Scalability up to a few dozen cores

- Cache coherence is increasingly complex to implement and verify as the number of cores increases
- Performance and energy efficiency issues

Time-predictability issues

- No scratch-pad or local memory dedicated to a particular core



GPGPUs Manycore Processors [NVIDIA 2009]

Streaming Multiprocessors (SM)

- Each SM comprises 32 ‘streaming cores’ that share a local memory, caches and a global memory hierarchy
- Threads are scheduled and executed atomically by ‘warps’, which execute the same instruction or at any given time
- Hardware multithreading enables warp execution switching on each clock cycle, helping cover memory access latencies

Restricted programming models

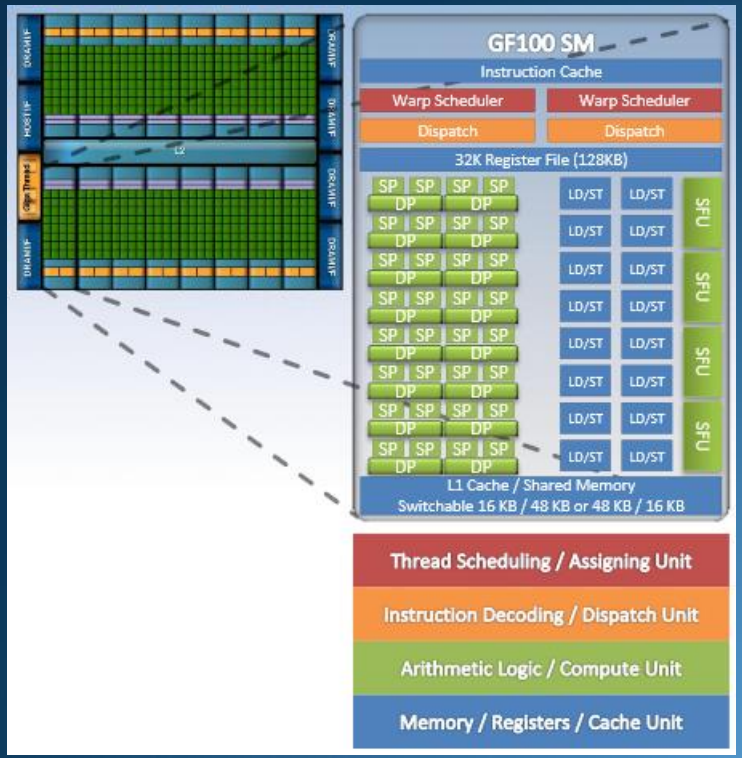
- CUDA, OpenCL

Performance effects of ‘thread divergence’

- Branch divergence when stream cores follow different execution paths
- Memory divergence when stream core memory accesses do not fall in the same cache blocks

Time-predictability issues

- Dynamic allocation of thread blocks to SMs
- Dynamic scheduling of warps inside a SM



CPU-Based Manycore Processors

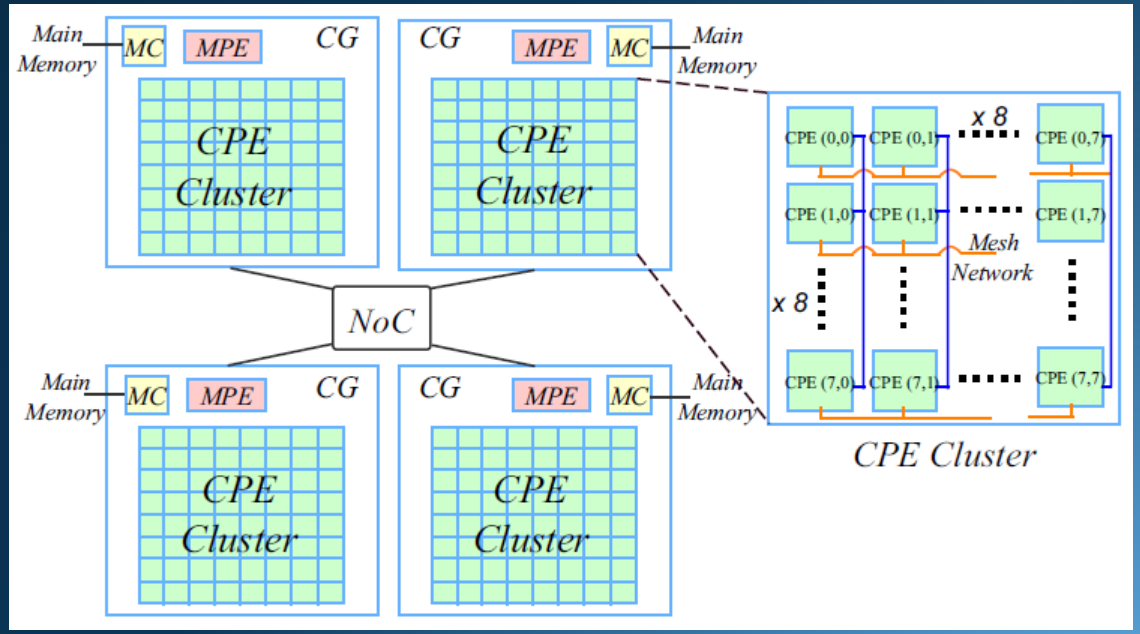
SW26010 Manycore processor

- Node of the Sunway TaihuLight supercomputer (#1 TOP 500 in 2016)
- 4 ‘core groups’ with MPE core, CPE core cluster, collective DMA engine
- 64KB scratch-pad memory per CPE core

Compute Unit as the unit of replication

- Group of cores
- Data move engine (DMA)
- Scratch-pad memory
- Local cache coherency
- Multicore programming

Network-on-chip (NoC) global interconnect



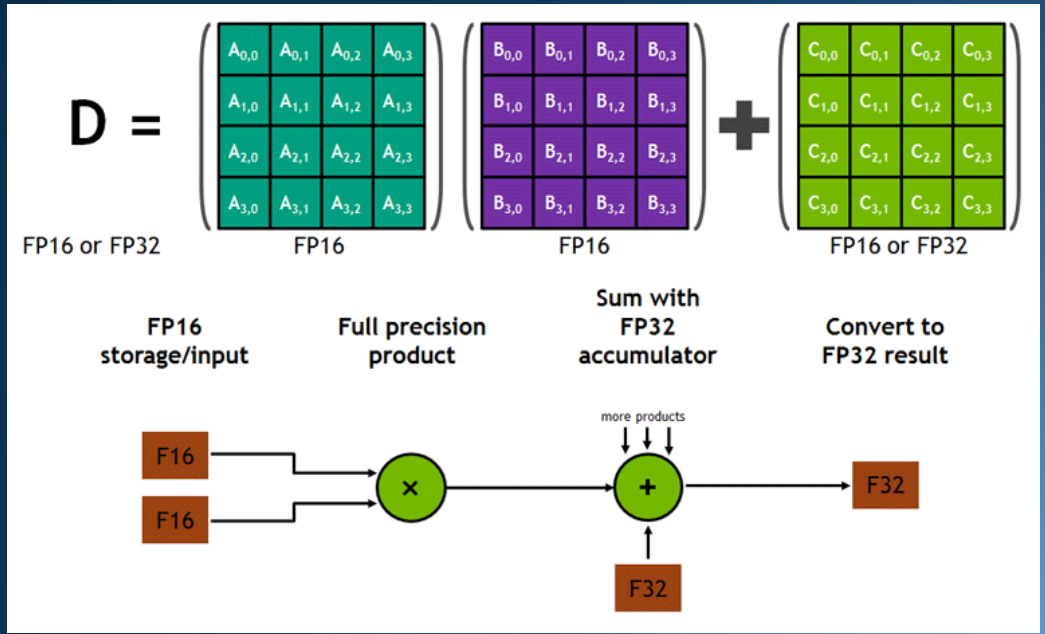
GPGPU Tensor Cores for Deep Learning [NVIDIA 2017]

NVidia Volta architecture

- 64x FP32 cores per SM
- 32x FP64 cores per SM
- 8x Tensor cores per SM

Tensor core operations

- Tensor Core perform D = A x B + C, where A, B, C and D are matrices
- A and B are FP16 4x4 matrices
- D and C can be either FP16 or FP32 4x4 matrices
- Higher performance is achieved when A and B dimensions are multiples of 8
- Maximum of 64 floating-point mixed-precision FMA operations per clock



Manycore Accelerator Compute Unit Options

	Design Choice	Advantages	Issues
Processing Engines	Single-core	Simpler memory hierarchy	Limited performances
	Multi-core	**OpenMP3 / Pthread multi-threading inside Compute Units**	**Multi-banked local memory**
	Core multi-threading	Overlap compute & transfers	Requires more registers and local memory capacity
Local Memory	**Sratch-pad memory**	**Energy-efficient, deterministic**	**Supported only by OpenCL and OpenVX; requires RDMA engine**
	Local cache coherence	**Required by OpenMP and PThread programming**	**Non time-predictable, must be disabled for hard real-time**
	Global cache coherence	Multi-core programming model across compute units	Not energy-efficient & not scalable [Proxy Architectures for Exascale]
	Global memory addressing	Scalable, applied by GPGPUs	Atomic operations are more difficult to implement
Global Memory	**Semi coherent with host cores**	**Enough for OpenCL support, OpenMP offloading possible**	**Support by system interconnect and cache coherency**
	Fully coherent with host cores	Simpler OpenMP offloading	Emerging standards CCIX and CXL; CXL requires PCIe Gen5
	Hardware prefetch engine	May improve performances	Less energy-efficiency than RDMA engines on prescribed addresses

Outline

1. Edge Computing
2. Manycore Processors
3. MPPA Architecture
4. Acceleration Software
5. Applications & Outlook

MPPA®3 Manycore Processor

5 Compute Units, 80 Accelerated VLIW Cores

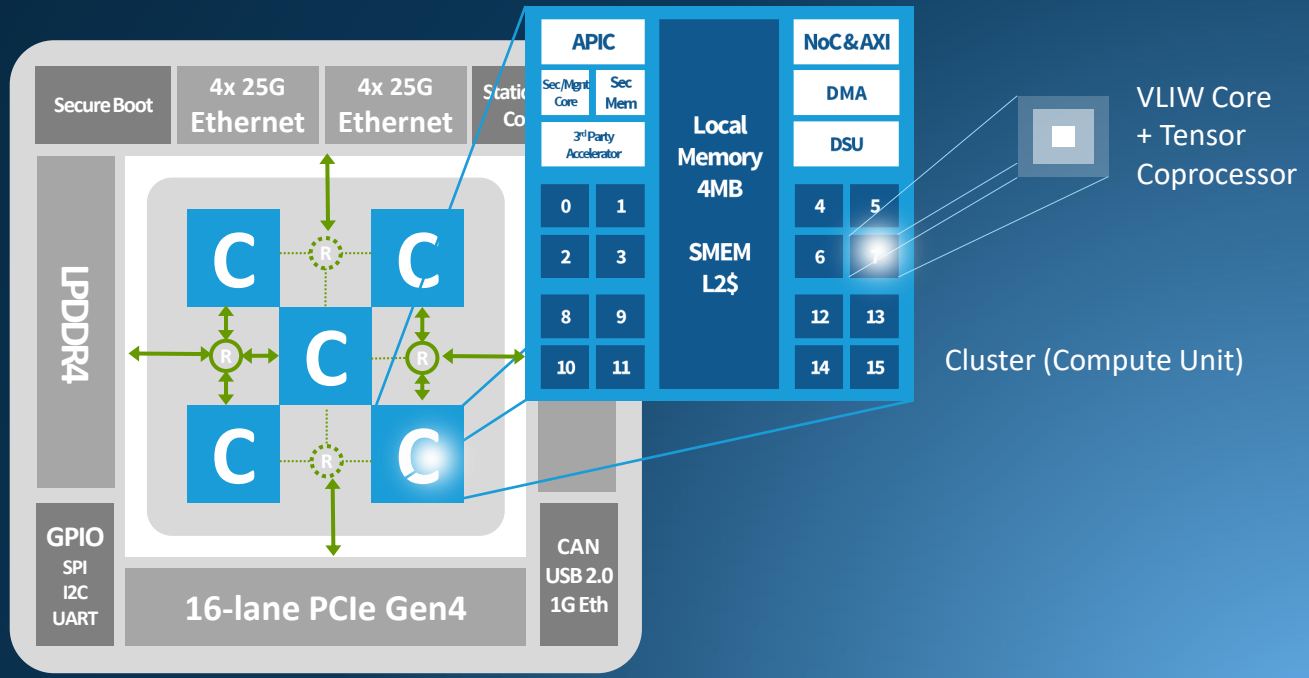
Peak Performances
200KMIPs, 25 DL TOPS at 1.2GHz

Power efficiency
40W Typical

High Speed I/F
200Gbs Ethernet,PCIe Gen4,

Functional Safety & Cyber-Security
Secure Islands, Secure Boot

Programming
Control Plane – Linux – 16 cores
Data Plane - 64 cores



Very Long Instruction Word (VLIW) Architectures

Energy-efficient, time-predictable instruction-level parallel execution

Classic VLIW architecture (J. A. Fisher)

- SELECT operation on Boolean value
- Conditional load/store/FPU operations
- Dismissible loads (non-trapping)
- [Multi-way conditional branches]

Key compiler techniques

- Trace scheduling (global instruction scheduling)
- Partial predication (S. Freudenberger algorithm)

Main examples

- Multiflow TRACE processors
- HP Labs Lx « Embedded Computing: a VLIW Approach »
- STMicroelectronics ST200 (media processor based on Lx)

EPIC VLIW architecture (B. R. Rau)

- Fully predicated ISA
- Speculative loads (control speculation)
- Advanced loads (data speculation)
- Rotating registers

Key compiler techniques

- Modulo scheduling (software pipelining)
- Full predication (R-K algorithm, J. Fang algorithm)

Main examples

- Cydrome Cydra-5
- HP-intel IA64
- TI C6x DSPs

MPPA®3 64-Bit VLIW Core

Vector-scalar ISA

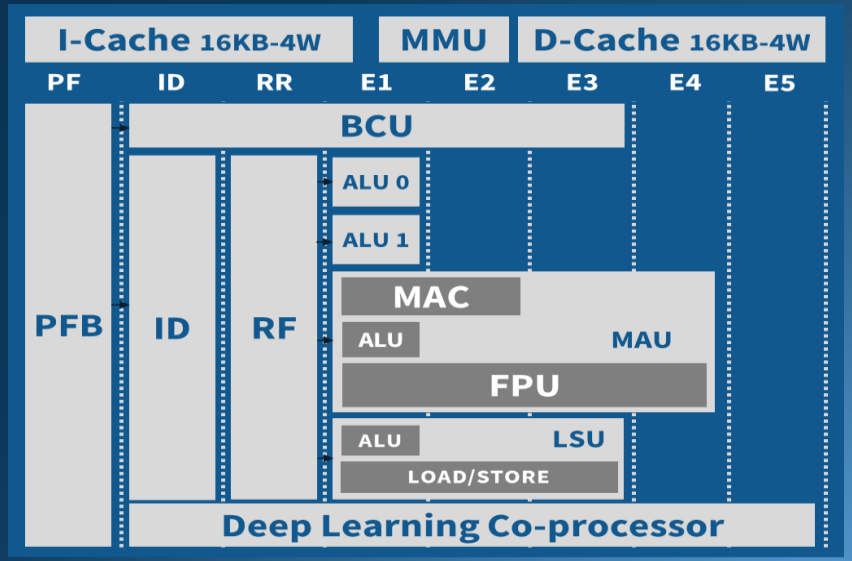
- 64x 64-bit general-purpose registers
- Operands can be single registers, register pairs (128-bit) or register quadruples (256-bit)
- 128-bit SIMD instructions by dual-issuing 64-bit on the two ALUS or by using the FPU datapath

DSP capabilities

- Counted or while hardware loops with early exits
- Non-temporal loads (L1 cache bypass)
- Non-trapping memory loads

CPU capabilities

- 4 privilege levels, MMU (runs Linux kernel)
- Advanced ISA virtualization support



VLIW CORE PIPELINE

MPPA®3 Tensor Coprocessor

Matrix-oriented arithmetic operations

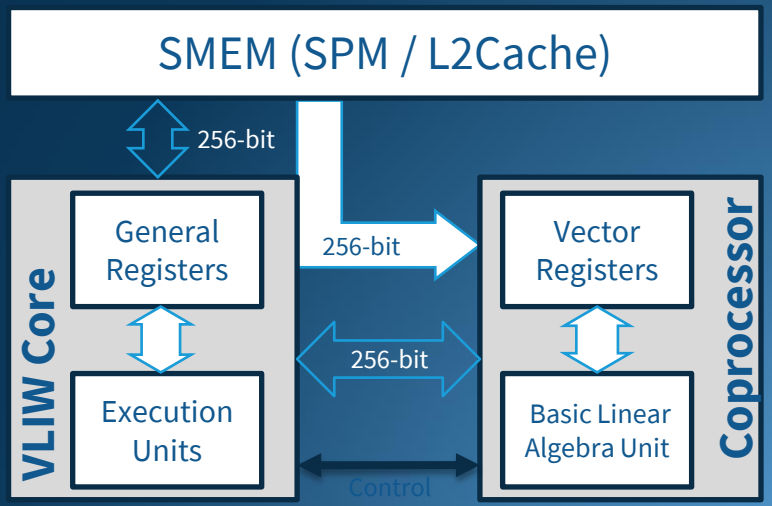
- Separate 256-bit wide vector register file
- Any coprocessor operand (1, 2 or 4 registers) is interpreted as a submatrix with four rows and a variable number of columns

Full integration into core instruction pipeline

- Extend VLIW core ISA with extra issue lanes
- Move instructions supporting matrix-transpose
- Register dependency / cancel management
- Memory directly accessible from coprocessor

Load-scatter memory operations

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



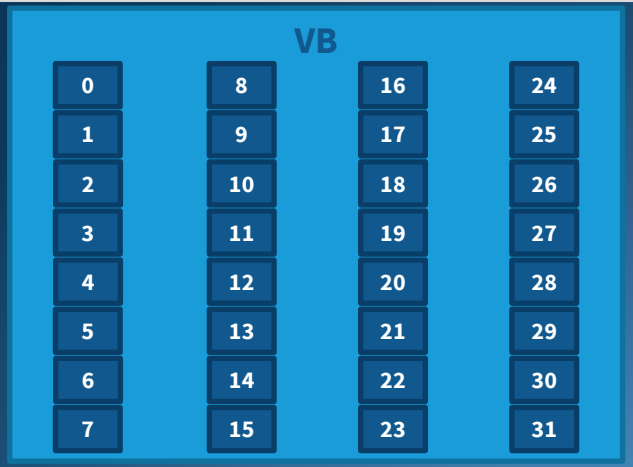
INT8.32 Matrix Multiply-Accumulate

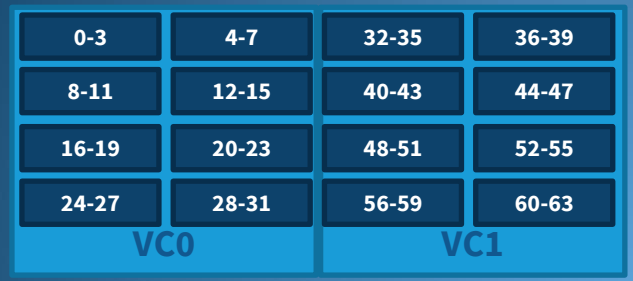
4x load-scatter operations load a 4x32 submatrix of a row-major order byte matrix A in memory (eg. activations) held in 4 consecutive registers, each holding a 4x8 byte submatrix

4x load-scatter operations load a 32x4 submatrix of a column-major order byte matrix B in memory (eg weights) held in 4 consecutive registers, each holding a 8x4 byte submatrix

Instruction accumulates the 4x8x4 product into a 4x4 submatrix of 32-bit elements into two consecutive registers VC0 and VC1

(numbers correspond to bytes and boxes to submatrix elements)





MPPA®3 Memory Hierarchy

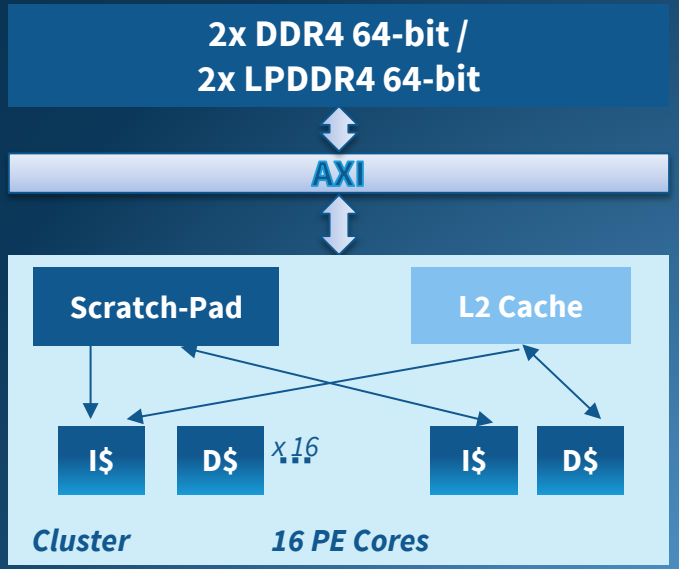
Memory model adapted to OpenCL and to multi-node ROS

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 memory 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

Network-on-Chip for Global Interconnects

NoC as generalization of busses

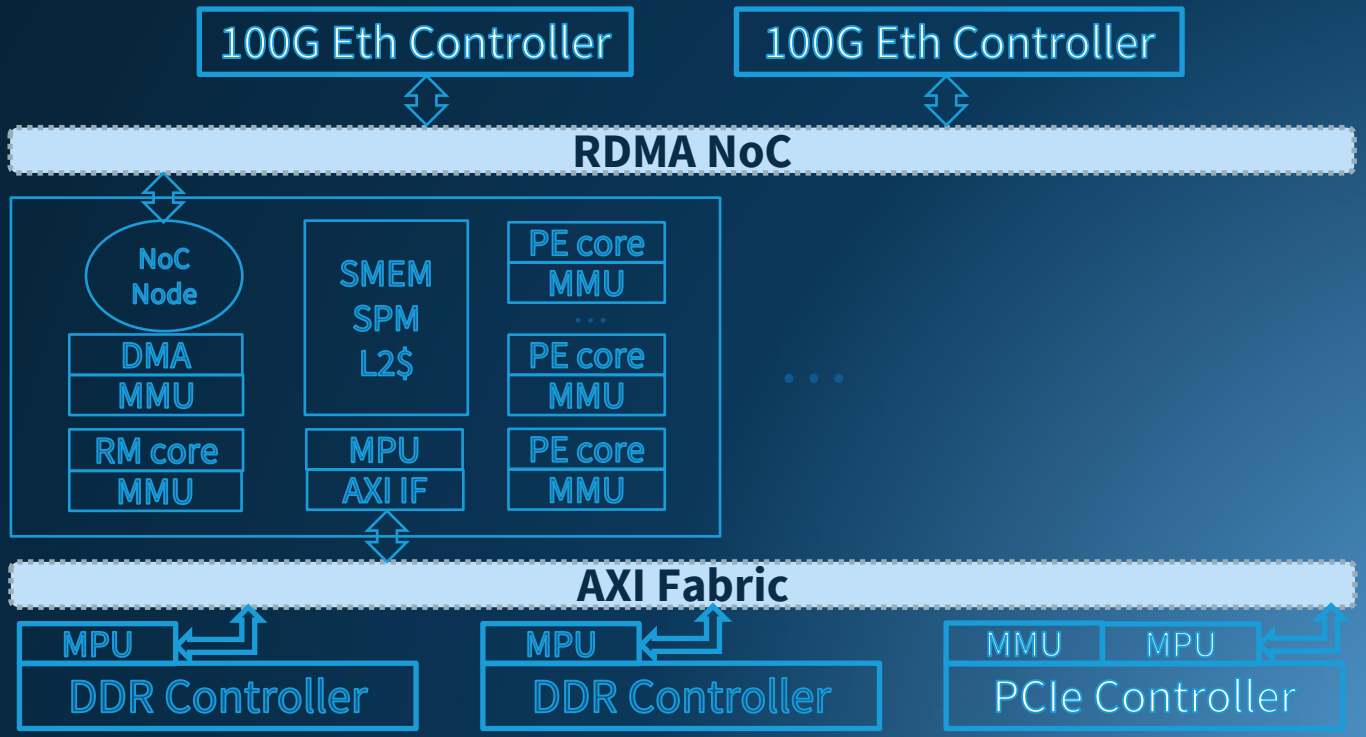
- Connectionless
- Address-based transactions
- Flit-level flow control
- Implicit routing
- Inside a coherence domain
- Reliable communication
- Coherency protocol messages
- Coordinate with DDR memory controller front-end (Ex. Arteris FlexMem Multi-Array Memory Scheduler)

NoC as integrated macro-network

- Connection-oriented
- Stream-based transactions
- [End-to-end flow control]
- Explicit routing
- Across address spaces (RDMA)
- [Packet loss or packet reordering]
- Traffic shaping for QoS (application of DNC)
- Terminate macro-network (Ethernet, InfiniBand)
- Support of multicasting

KALRAY

MPPA®3 Global Interconnects



Outline

1. Edge Computing
2. Manycore Processors
3. MPPA Architecture
4. Acceleration Software
5. Applications & Outlook

MPPA® High-Performance Programming Models

STANDARD PROGRAMMING ENVIRONMENTS

OPENCL 1.2 Programming

Standard accelerator programming model

- POSIX host CPU accelerated by MPPA device (OpenAMP interface)
- OpenCL 1.2 conformance based on POCL and LLVM for OpenCL-C

OpenCL offloading modes:

- Linearized Work Items on a PE (LWI)
- Single Program Multiple Data (SPMD)
- Native functions called from kernels

C/C++ POSIX Threads Programming

Standard multicore programming model

- MPPA Linux and ClusterOS
- Standard C/C++ programming
  - GCC, GDB, Eclipse system trace
- POSIX threads interface
- GCC and LLVM OpenMP support

Exposed MPPA® communications

- RDMA using the MPPA Asynchronous Communication library (mppa_async)

KaNN™, Kalray Neural Network Compiler

A comprehensive Neural Network inference offer

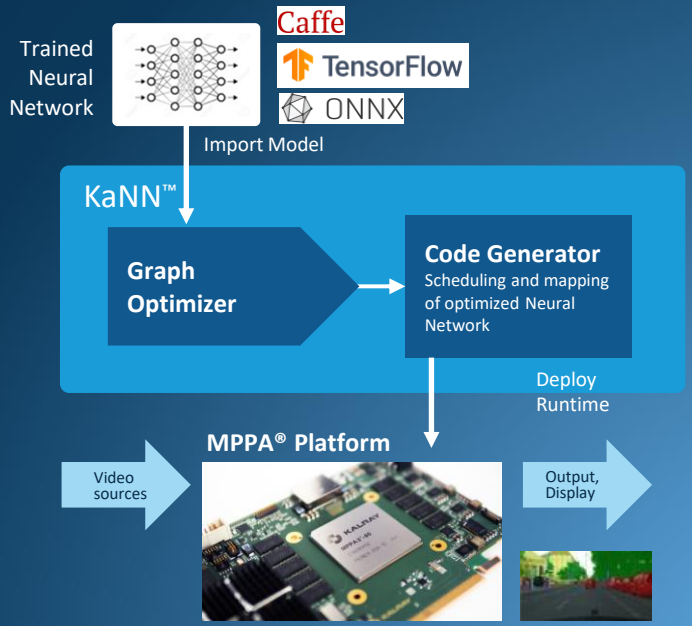
From trained models in standard CNN frameworks
to code generation, setup & concurrent CNN inferences

Deep Learning Inference Code Generator

- Optimization of neural networks for MPPA®
- Deployment of neural networks on MPPA®

Deep Learning Inference Runtime Support of:

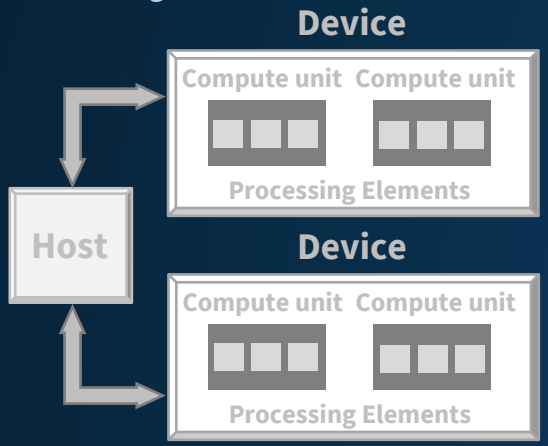
- Standard frameworks
- Major convolutional networks
- Custom networks



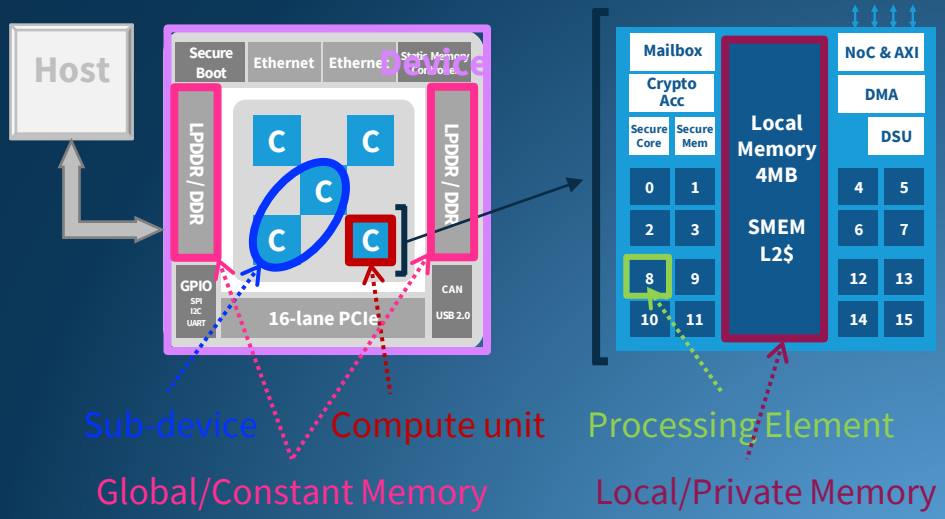
MPPA® OpenCL Compute Platform Mapping

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: Multi-Device => Device => Sub-Device => Compute Unit(s) => Processing Elements



OpenCL 'SPMD' Mapping to MPPA® Architecture



MPPA® OpenCL Native Function Extension

- Call standard C/C++/OpenMP/POSIX (ClusterOS) code from OpenCL kernels
- **Generalization of TI 'OpenMP Dispatch With OpenCL' for KeyStone-II platforms**
- Used by the Kalray KaNN deep learning inference compiler
- Used by BLAS and multi-cluster libraries

```
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);
}
```

MPPA Asynchronous One-Sided Operations API

generalization of OpenCL async_work_group_copy() callable from C/C++

Dense Transfers

- mppa_async_get
- mppa_async_put
- mppa_async_get_spaced
- mppa_async_put_spaced
- mppa_async_get_indexed
- mppa_async_put_indexed
- mppa_async_get_streamed
- mppa_async_put_streamed

Asynchronous Events

- mppa_async_event_wait
- mppa_async_event_test

Sparse Transfers

- mppa_async_sget_spaced
- mppa_async_sput_spaced
- mppa_async_sget_blocked2d
- mppa_async_sput_blocked2d
- mppa_async_sget_blocked3d
- mppa_async_sput_blocked3d

Remote queues

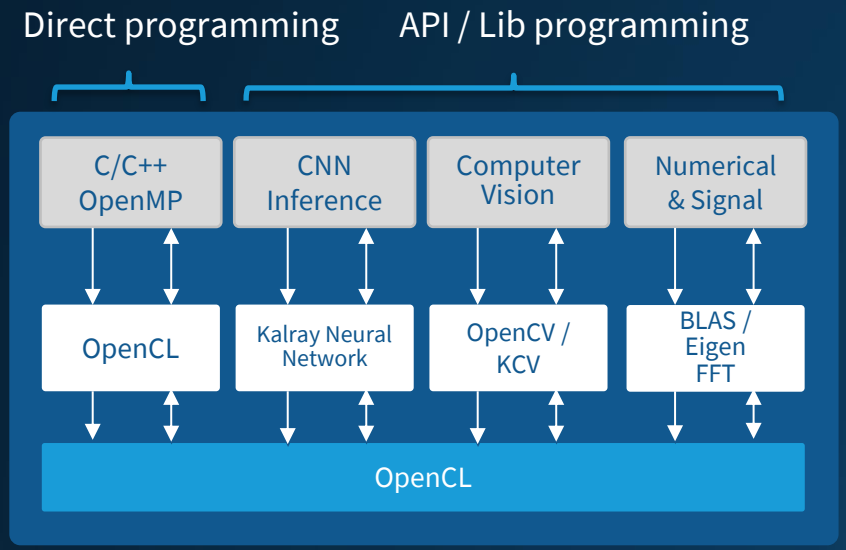
- mppa_async_enqueue
- mppa_async_dequeue
- mppa_async_dequeue_copy
- mppa_async_discard

Global Synchronization

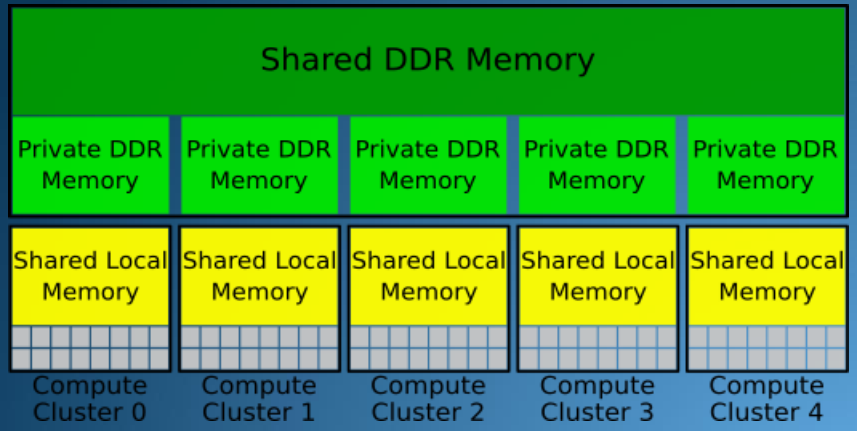
- mppa_async_quiet
- mppa_async_fence
- mppa_async_peek
- mppa_async_poke
- mppa_async_postadd
- mppa_async_fetchclear
- mppa_async_fetchadd
- mppa_async_evalcond

Kalray Acceleration Framework (KAF™)

A integrated way to program a manycore accelerator
based on OpenCL Sub-Devices and Native Functions extension



Memory Model for Native Functions



KALRAY

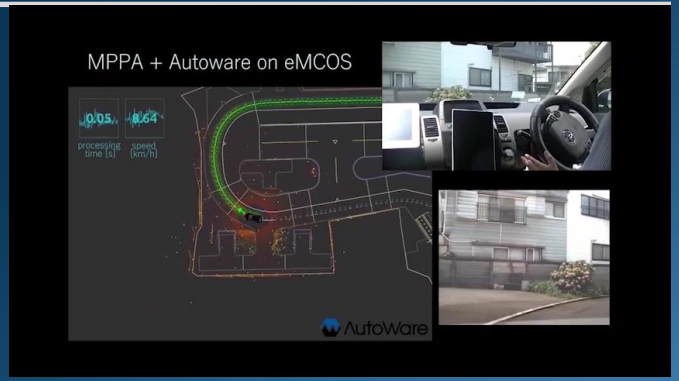
Outline

1. Edge Computing
2. Manycore Processors
3. MPPA Processors and IP
4. Accelerator Offloading
5. Applications & Outlook

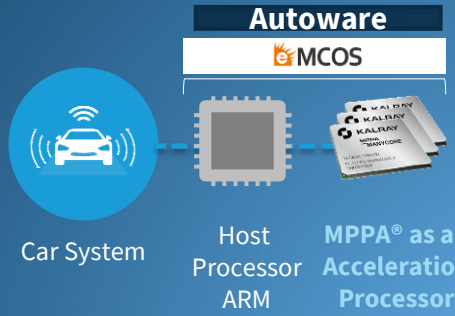
Autonomous Driving Applications

THE AUTOWARE FOUNDATION

- **Functions**
  - Automotive (Autonomous Driving / ADAS)
  - Object Tracking and Path Planning
- **Implementation**
  - Extensive use of **eMCOS POSIX** & ROS [1]
  - Autoware/ROS for control/vision
  - **MPPA® used as multi-accelerator** (vision and LiDAR)



Combination of RTOS-POSIX with Multi-Accelerator

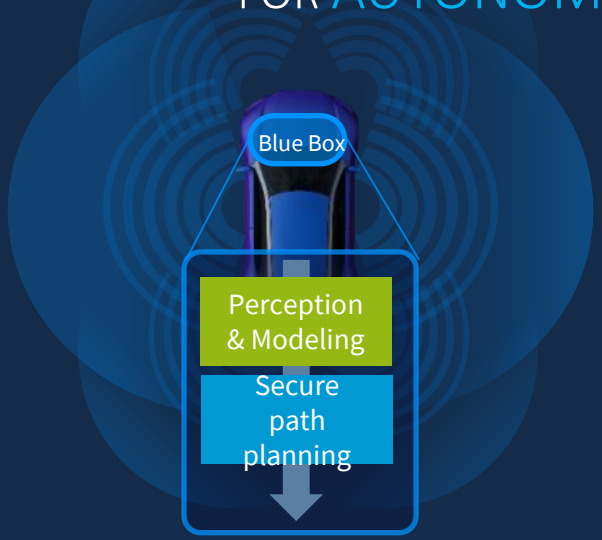


[1] ROS = Robot Operating System

KALRAY

NXP – Kalray Partnership

CENTRAL COMPUTING SOLUTION FOR AUTONOMOUS VEHICLES



KALRAY
Perception & Modeling

NXP
Secure path planning

- For Level 3 vehicles
- Roadmap up to Level 5

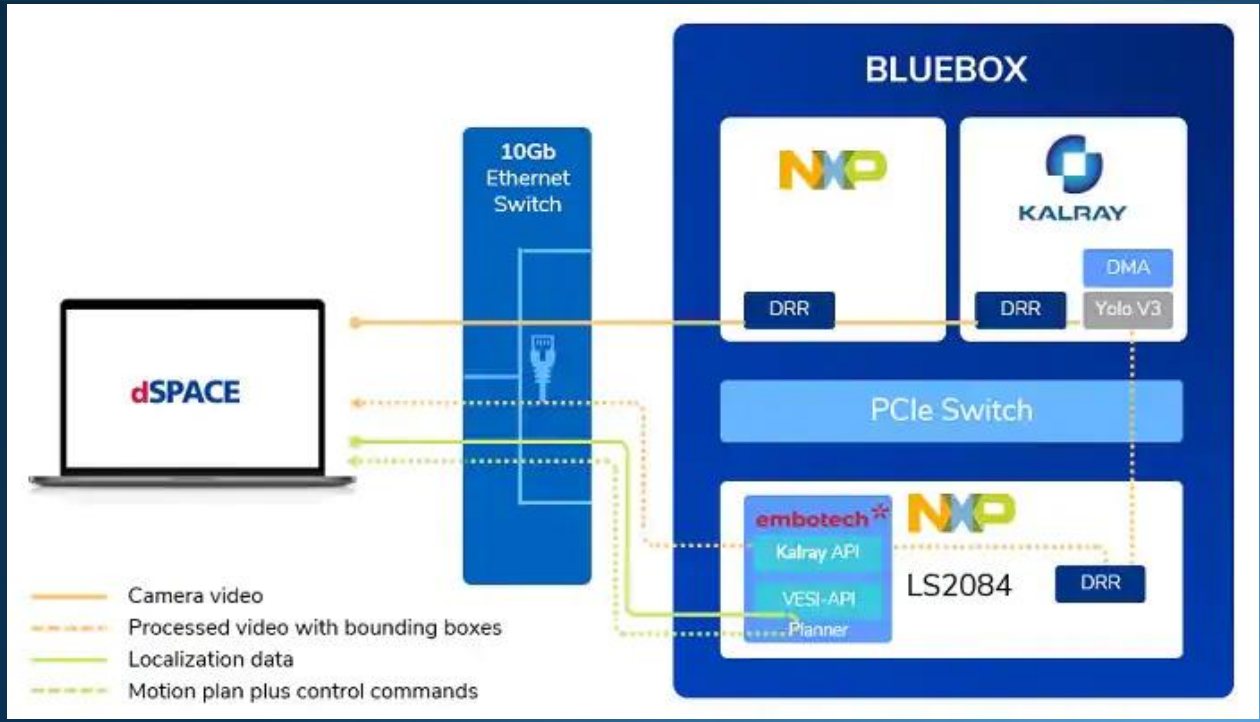
NXP #1 in automotive semiconductors (11% market share[1])
#2 in automotive processors (28% market share[2])

1 Gartner 2017 market share
2 NXP 2017 market share

KALRAY

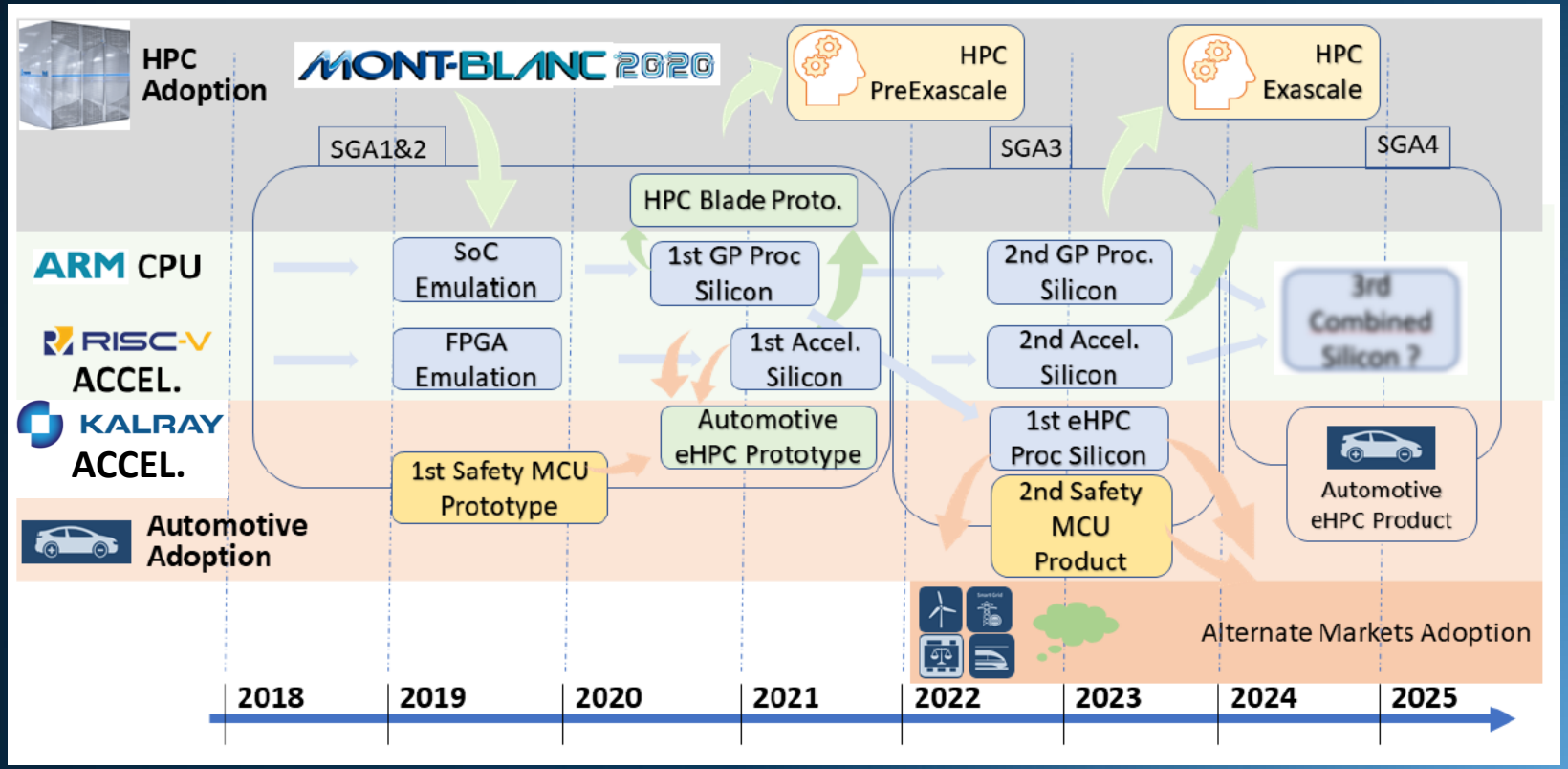
CES 2020 NXP Demonstration

- NXP BlueBox 2nd generation Autonomous Driving Development platform with production ready automotive silicon
- Kalray 3rd Generation MPPA Accelerator and AI Software for Perception
- 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.



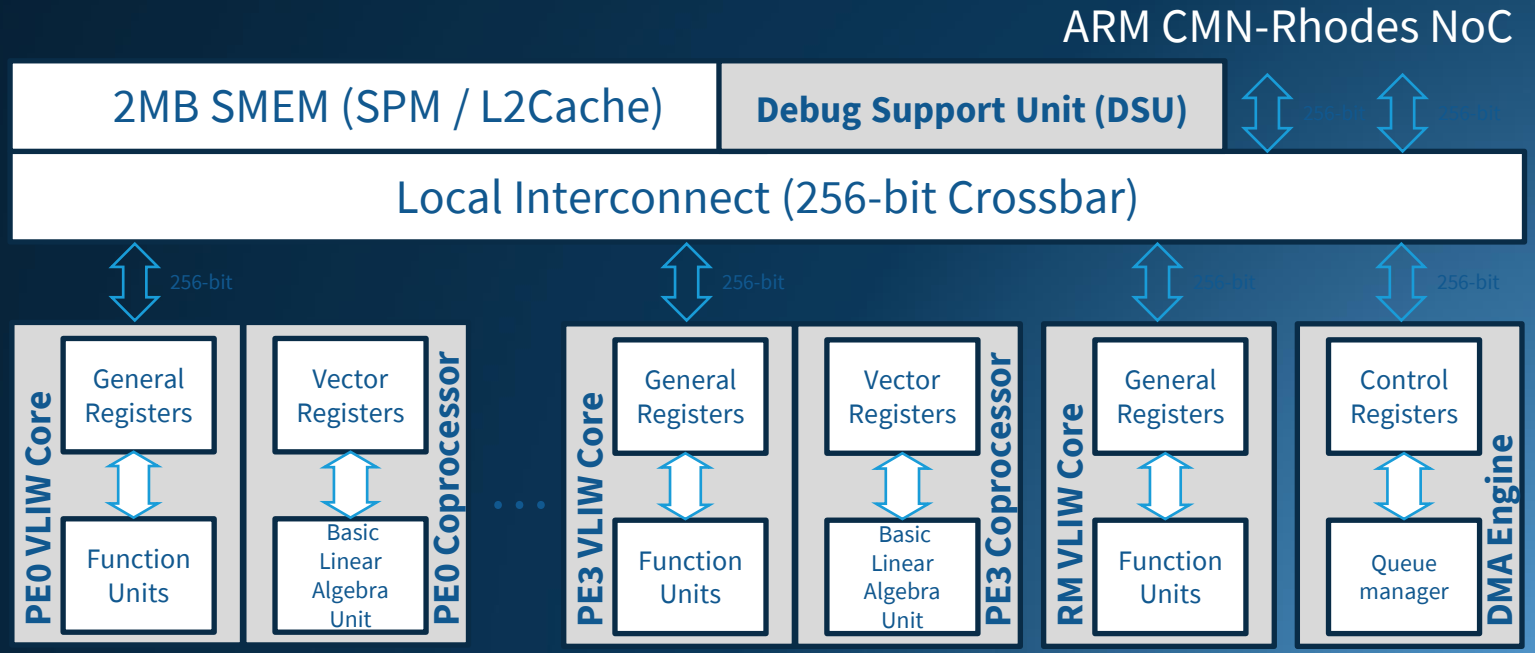
Page 36

Mont-Blanc 2020 and EPI Projects



Page 37

MPPA Acceleratot Tile Delivered to EPI Project (TSMC 6nm)



KVX Accelerator Tile for the EPI SGA-2

RISC-V Cores

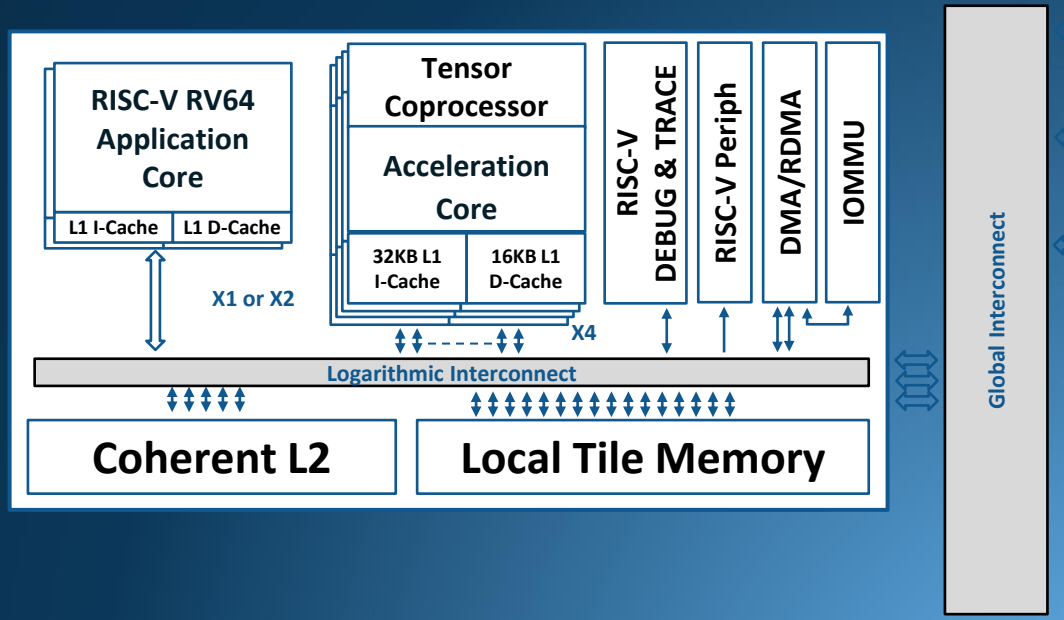
- A general-purpose 64-bit application core is used for running RDMA, MPI and storage software stacks

KVX Cores + coprocessors

- 4 Kalray VLIW cores, each with a dedicated tensor coprocessor, provide the HPC/Edge floating-point performance 128 DP FLOP/cycle

Local multi-banked memory

- Supports the required local load/store bandwidth (32 bytes per KVX core)
- Data move by DMA/RDMA engine



OpenMP for Accelerator Offloading

First map data to the accelerator, then distribute work to the accelerator threads

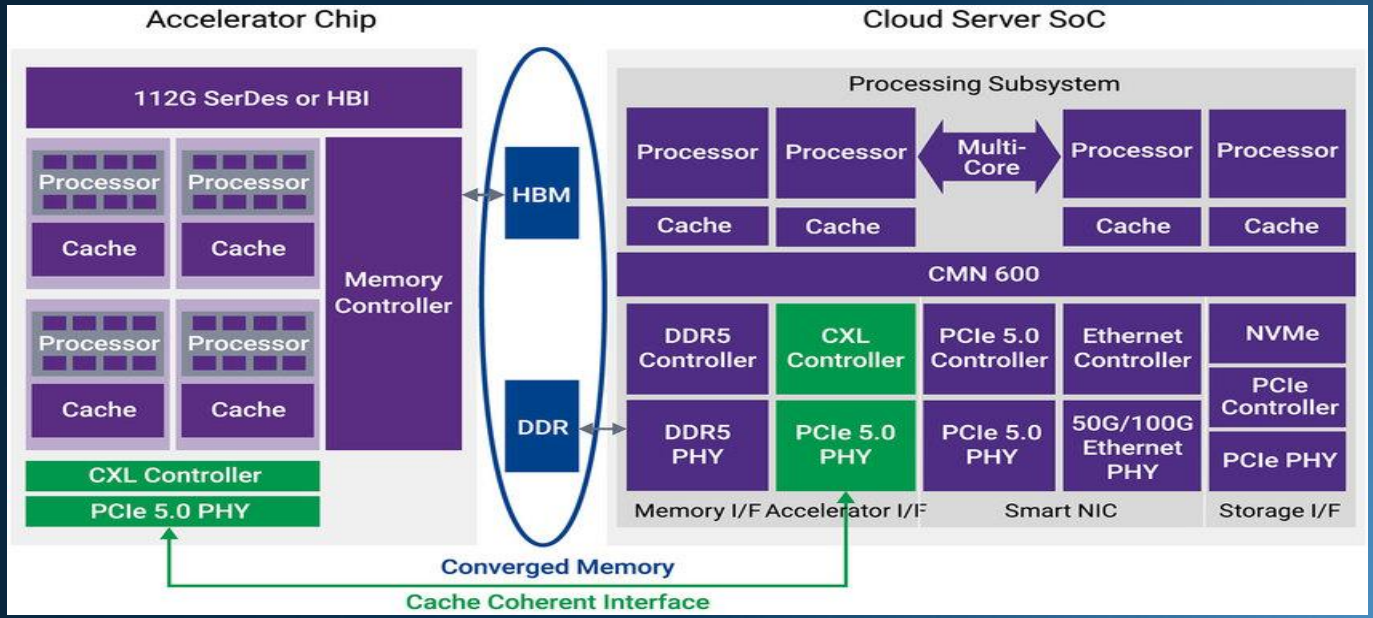
```
while ( error > tol && iter < iter_max )
{
  error = 0.0;
  #pragma omp target map(alloc:Anew[:n+2][:m+2]) map(tofrom:A[:n+2][:m+2])
  {
    #pragma omp target teams distribute parallel for reduction(max:error)
    for( int j = 1; j < n-1; j++) {
      for( int i = 1; i < m-1; i++ ) {
        Anew[j][i] = 0.25 * ( A[j][i+1] + A[j][i-1]
                            + A[j-1][i] + A[j+1][i]);
        error = fmax( error, fabs(Anew[j][i] - A[j][i]));
      }
    }
    #pragma omp target teams distribute parallel for collapse(2)
    for( int j = 1; j < n-1; j++) {
      for( int i = 1; i < m-1; i++ ) {
        A[j][i] = Anew[j][i];
      }
    }
  }
  if(iter++ % 100 == 0) printf("%5d, %0.6f\n", iter, error);
}
```

Accelerator Interconnect Beyond PCIe

Rise of CCIX and CXL near-memory interconnect standards

Avoiding DDR round-trips between GPP and accelerator saves latency and energy

- Accelerator memory is part of the system memory (coherent cached)
- Example of system architecture from Synopys for CXL



ACKNOWLEDGEMENTS

This work was performed in the scopes of the ES3CAP and CPS research projects under the Bpifrance Invest for the Future Program (Programme d'Investissements d'Avenir — PIA), and the European Union's Horizon 2020 Research and Innovation programme, European Processor Initiative and ECSEL OCEAN12 project.

Thank You

KALRAY S.A.
Corporate Headquarters
180, avenue de l'Europe
38 330 Montbonnot, France
Phone: +33 (0)4 76 18 90 71
contact@kalrayinc.com

CERTIFIED ORGANIZATION ISO 9001
CERTIFICATION IN PROGRESS ISO 26262

KALRAY INC.
America Regional Headquarters
4962 El Camino Real
Los Altos, CA - USA
Phone: +1 (650) 469 3729
contact@kalrayinc.com

KALRAY JAPAN - KK
Represented by MACNICA Inc. Strategic Innovation Group
Macnica Building, No.1, 1-6-3 Shin-Yokohama
Kouhoku-ku, Yokohama 222-8561, Japan
Phone: +81-45-470-9870

KALRAY S.A.
Sophia-Antipolis
1047 allée Pierre Ziller
Business Pôle – Bâtiment B, Entrée A
06560 Sophia-Antipolis, France
Phone: + 33(0) 4 76 18 09 18

www.kalrayinc.com