Kalray in a Nutshell

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

**Industrial investors**
- **Leader in Manycore Technology**
- **3rd generation of MPPA® processor**
- **€85m R&D investment**
- **30 Patent families**

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

**3rd generation of MPPA® processor**
- France (Grenoble, Sophia-Antipolis)
- USA (Los Altos, CA)
- Japan (Yokohama)
- Canada (Partner)
- China (Partner)
- South Korea (Partner)

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

1. MPPA®3 Manycore Processor
2. Standard Programming Environments
3. Model-Based Development Environments
4. Consolidation of Application Functions
Intelligent Systems, a Disruptive Challenge Requiring a New Generation of Processors

- **1970**: CENTRAL COMPUTERS
  - Remote processing

- **1990**: SMARTPHONES
  - Connectivity & mobility

- **2020**: INTELLIGENT SYSTEMS
  - Cyber-physical systems with machine learning

**Moore's law**
- Up to a dozen cores
- “More than Moore’s law”

**Manycore, the next frontier**
- Unlimited number of independent cores

**Multicore**
- Up to a dozen cores
- “More than Moore’s law”

**Single core**
- “Moore’s law”

- 1980s-2000s
- 2000s-2010s
- 2020s & beyond
Multicore and Manycore Processors

Homogeneous Multicore Processor

Multiple CPU cores sharing a cache-coherent memory hierarchy
- Scalability by replicating CPU cores
- Standard programming models

Energy efficiency issues
- Global cache coherence scaling

Time-predictability issues
- No scratch-pad or local memories

GPGPU Manycore Processor

Multiple Streaming Multiprocessors
- Restricted programming models

Performance issues of ‘thread divergence’
- Branch divergence slow down the execution
- Memory divergence: non-coalesced accesses

Time-predictability issues
- Dynamic allocation of thread blocks
- Dynamic scheduling of warps

CPU-Based Manycore Processor

Multiple “Compute Units” connected by a network-on-chip (NoC)
- Scalability by replicating Compute Units
- Standard multicore programming inside a Compute Unit

Compute Unit
- Group of cores + DMA
- Scratch-pad memory (SPM)
- Local cache coherency
MPPA®3 Manycore Processor
5 Compute Units, 80 Accelerated VLIW Cores

- **Peak Performances**: 200KMiPs, 25 DL TOPS at 1.2GHz
- **Power efficiency**: 25W 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

![Diagram of MPPA®3 Manycore Processor](image-url)
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 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
MPPA®3 Global Interconnects

100G Eth Controller

RDMA NoC

NoC Node
DMA
MMU
RM core
MMU
SMEM
SPM
L2$

PE core
MMU
...

PE core
MMU
PE core
MMU

AXI IF

AXI Fabric

MPU

DDR Controller

MPU

DDR Controller

MMU

MPU

PCle Controller

©2020 Kalray SA. All rights reserved.
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)
- 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
MPPA®3 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
- Register dependency / cancel management

Leverage MPPA the memory hierarchy
- SMEM directly accessible from coprocessor
- Memory load stream alignent operations

Arithmetic performances (MPPA3-v1)
- 128x INT8→INT32 MAC/cycle
- 64x INT16→INT64 MAC/cycle
- 16x FP16→FP32 FMA/cycle
MPPA®3 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 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
Outline

1. MPPA®3 Manycore Processor
2. Standard Programming Environments
3. Model-Based Development Environments
4. Consolidation of Application Functions
MPPA® Accelerator and Stand-Alone Modes

ACCELERATOR MODE
Host running Linux

STAND-ALONE MODE
Cluster 0 used as Host
MPPA® High-Performance Programming Models

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

‘SPMD’ Mapping to MPPA® Architecture

<table>
<thead>
<tr>
<th>Mailbox</th>
<th>Crypto Acc</th>
<th>Secure Core</th>
<th>Secure Mem</th>
<th>DMA</th>
<th>NoC &amp; AXI</th>
<th>DSU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>14</td>
<td>15</td>
<td></td>
</tr>
</tbody>
</table>

Local Memory
4MB

SMEML2$

Processing Element

Local/Private Memory

Global/Constant Memory

Sub-device

Compute unit

Processing Elements
Kalray Acceleration Framework (KAF™)

A integrated way to program manycore architecture based on OpenCL Sub-Devices and Native Functions
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

```c
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);
}
```
KaNN™, Kalray Neural Network Inference Compiler

From standard Machine Learning frameworks to code generation, setup and multiple CNN execution

Deep Learning Inference Code Generator and Runtime
- Optimization of neural networks for MPPA®
- Deployment of neural networks on MPPA®
- Execution on the specified number of clusters

Support of:
- Major frameworks
- Major networks
- Custom networks
- FP16.32 & FP32 arithmetic
- INT8.32 integer quantization
Outline

1. MPPA®3 Manycore Processor
2. Standard Programming Environments
3. Model-Based Development Environments
4. Consolidation of Application Functions
ROSACE Case Study for MBD on Multicore

• Simplified controller for the longitudinal motion of a medium-range civil aircraft in en-route phase: cruise and change of cruise level sub-phases

• Application has 3 harmonic periods: F, 2F, 4F
SCADE Suite Multi-Core Code Generation Flow

- Partitioning Information: Separate from model for re-targeting
- Model to C Mapping
- Scheduling ECU/Core Allocation
- Target integration: Application code generation Allocation to workers
- CompCert C compiler

Process:
1. Partitioning Information
2. Model to C Mapping
3. Scheduling ECU/Core Allocation
4. Target integration: Application code generation Allocation to workers
5. CompCert C compiler
6. Application C code
7. Scade 6 application

Errors
SCADE Suite MCG Code Generation

• Exploit the MPPA cluster configuration for ‘high-integrity’ execution
  • Cluster local memory address mapping assigns one bank per core

• Precisely compute the task WCETs (Worst-Case Execution Times)
  • Static analysis or measurement for the WCET of tasks in isolation
  • Refine the WCET with interferences [Rihani RTNS'16][Schuh DATE’20]
  • 2-phased Predictable Execution Model better than 3-phased [Schuh RTSS’20]
Time-Triggered Multicore Scheduling [Schuh DATE’20]

- Given a task mapping and release dates, schedule by forward time sweep
- Release a task when its dependencies are satisfied and after its release date
- Adjust interferences considering to the subset of currently executing tasks
eSOL eMBP Multi-Core Code Generation Flow

- Block structure extraction
- Performance estimation
- Core mapping
- Parallelized code generation
- Visualization

User
Modeling
Simulink Model
Embedded Coder
C code

SHIM xml
Reference to performance calculation

Hardware B
Hardware A

Target B
Target A

Parallelized C code
Use your compiler to build

Visualization
Outline

1. MPPA®3 Manycore Processor
2. Standard Programming Environments
3. Model-Based Development Environments
4. Consolidation of Application Functions
Mapping Functions to Compute Units

Running Multiple applications and OS Concurrently

Secure Communication

High Performance Compute Acceleration
Machine Learning

Secure Boot

4x 25G Ethernet

4x 25G Ethernet

Static Memory Controller

LPDDR4

LPDDR4

16-lane PCIe Gen4

GPIO

SPI

I2C

UART

CAN

USB 2.0

1G Eth

Hard real-time application

Rich OS

Linux

SCADE
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 (Yolo v3 416x416 at 20FPS, NVIDIA Xavier is at 18FPS)
- 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.
Autonomous Driving Use Case

• 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
MPPA® for Machine Vision Applications

Acquisition Logic

Pre-processing:
- Image signal processing,
- Rules-based algorithms

Deep learning
- Inference (application dependent neural network)

Post processing:
- Video analytics,
- Communication

Motion Control

Machine Vision
- Lens and Sensors

AI Acceleration
- Rules-based Algorithms

Computer Vision

Video analytics

Motion Control
MPPA FOR EDGE APPLICATIONS
Compute-Intensive, Time-Critical, Safety & Security

Edge Computing

- Autonomous Driving
- Smart Cities
- Industry 4.0 Robotics
- 5G/Telecom
- Healthcare
- Drones/Avionics
- Video Surveillance
# KALRAY PRODUCT OFFER

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Evaluation &amp; Development</th>
<th>Prototyping</th>
<th>Production</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Customer wants to benchmark, evaluate, learn/train &amp; Develop</strong></td>
<td>Customer wants to test in its environment including vehicle prototypes: customize, adapt, fine tune, qualify</td>
<td>Go to production</td>
<td>Fully qualified HW and SW</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hardware</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MPPA®-DEV Kalray development Platform</td>
<td>Kalray PCI Card</td>
<td>Reference Design</td>
<td>Chip Acceleration Module (with 3rd party)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software &amp; Tools</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>

[Image of Hardware and Software]
Thank You