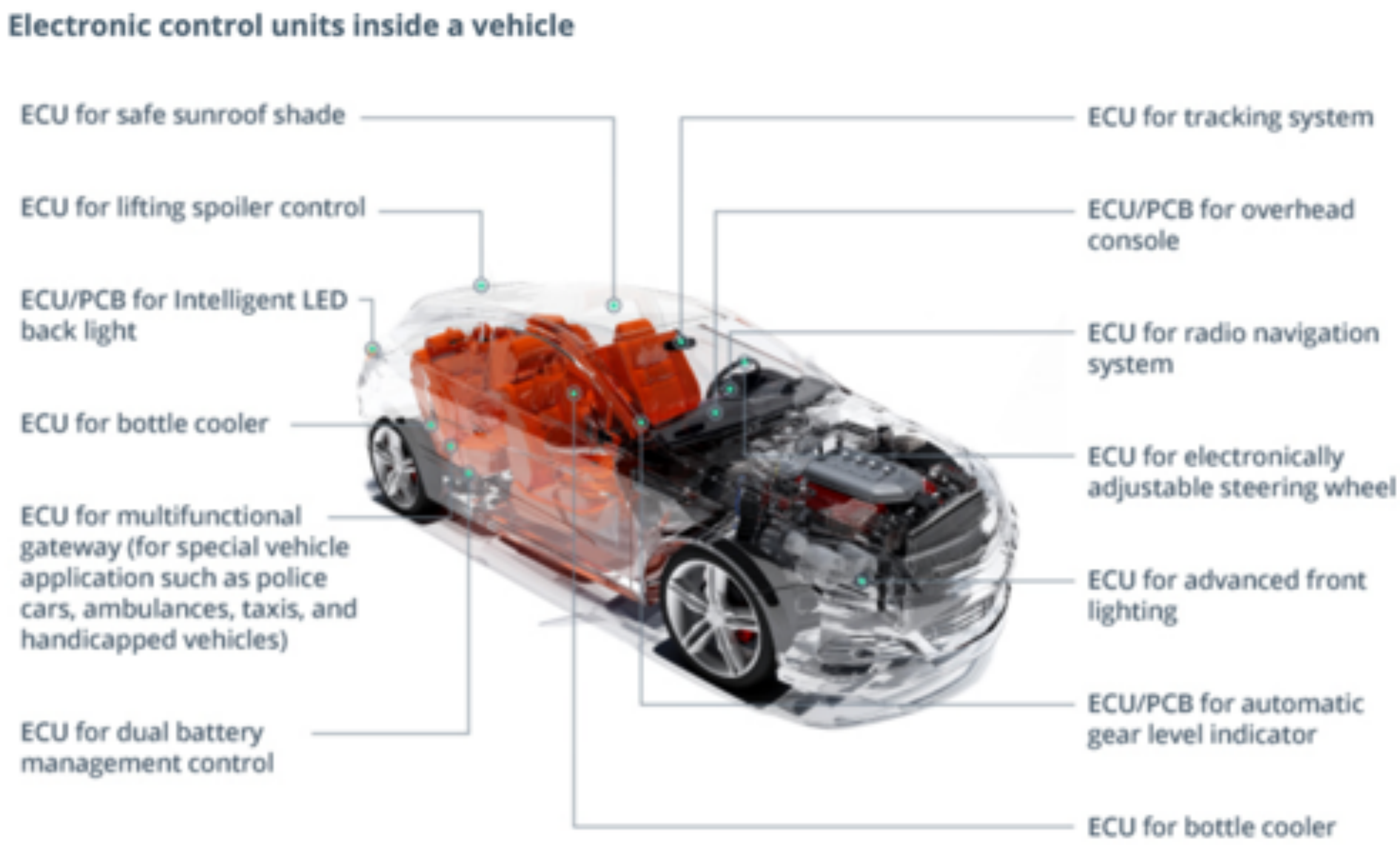
Introduction to Automotive ECU

Electronic Control Unit

Autoware Courses#4

JUNE 2020

www.kalrayinc.com



More than 80 ECU in a SAE Level 2* vehicle

In all domains

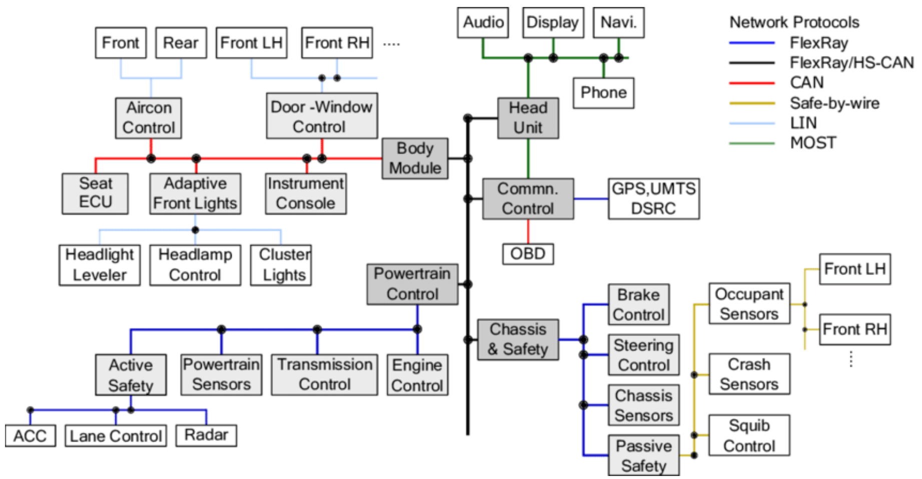
- Audio
- Video
- Navigation
- Detectors
- Controllers
- Motor Control
- Advanced Assistance System (ADAS)
- Autonomous Driving (AD)
- Connectivity
- Emergency call
- Comfort
- …

Source: SlideShare – Automotive bus technologies

* *https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic*

KALRAY

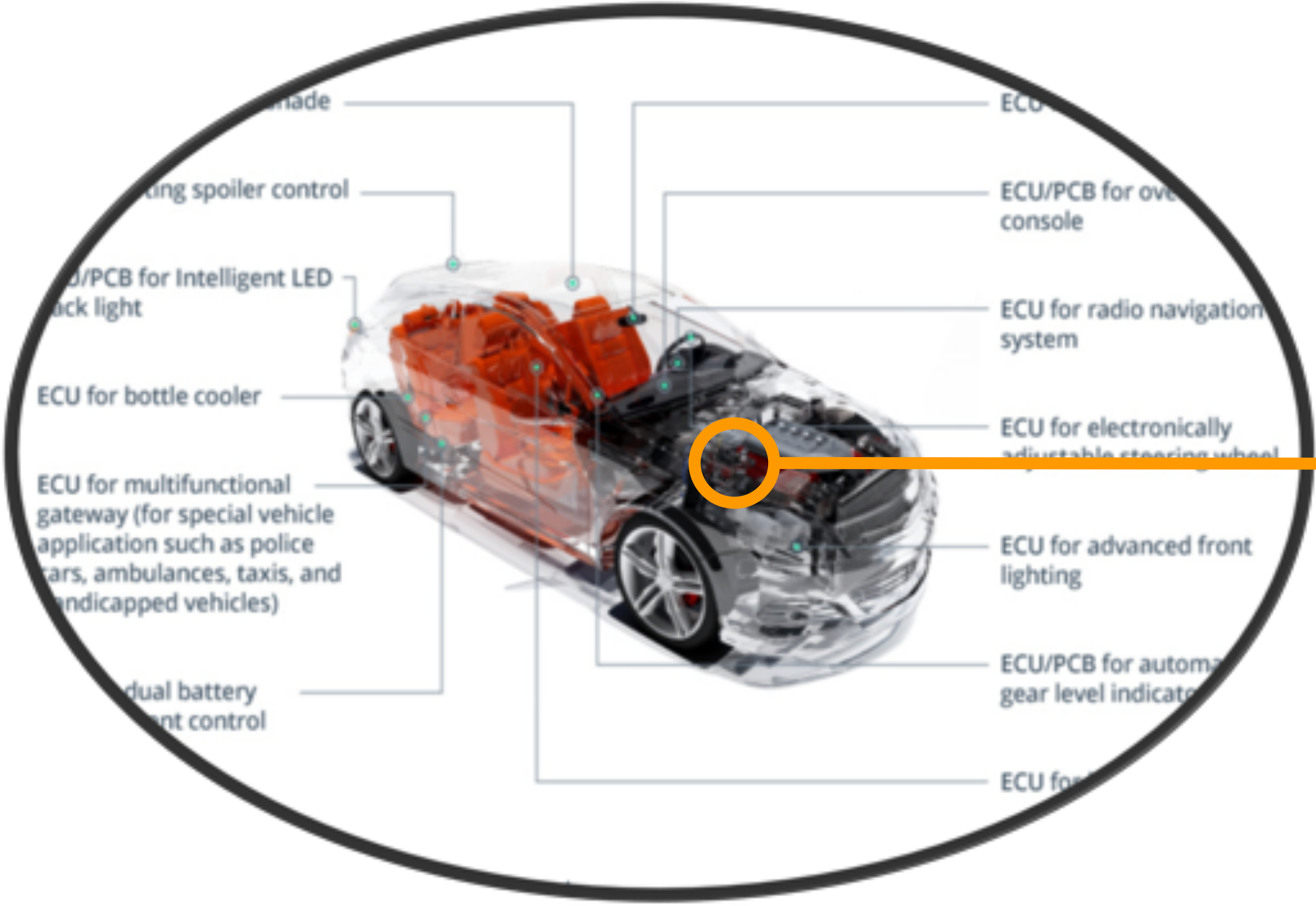
A REQUIRED EVOLUTION OF ARCHITECTURE



https://www.researchgate.net/figure/Typical-in-vehicle-network-architecture-in-a-modern-car_fig2_305499872

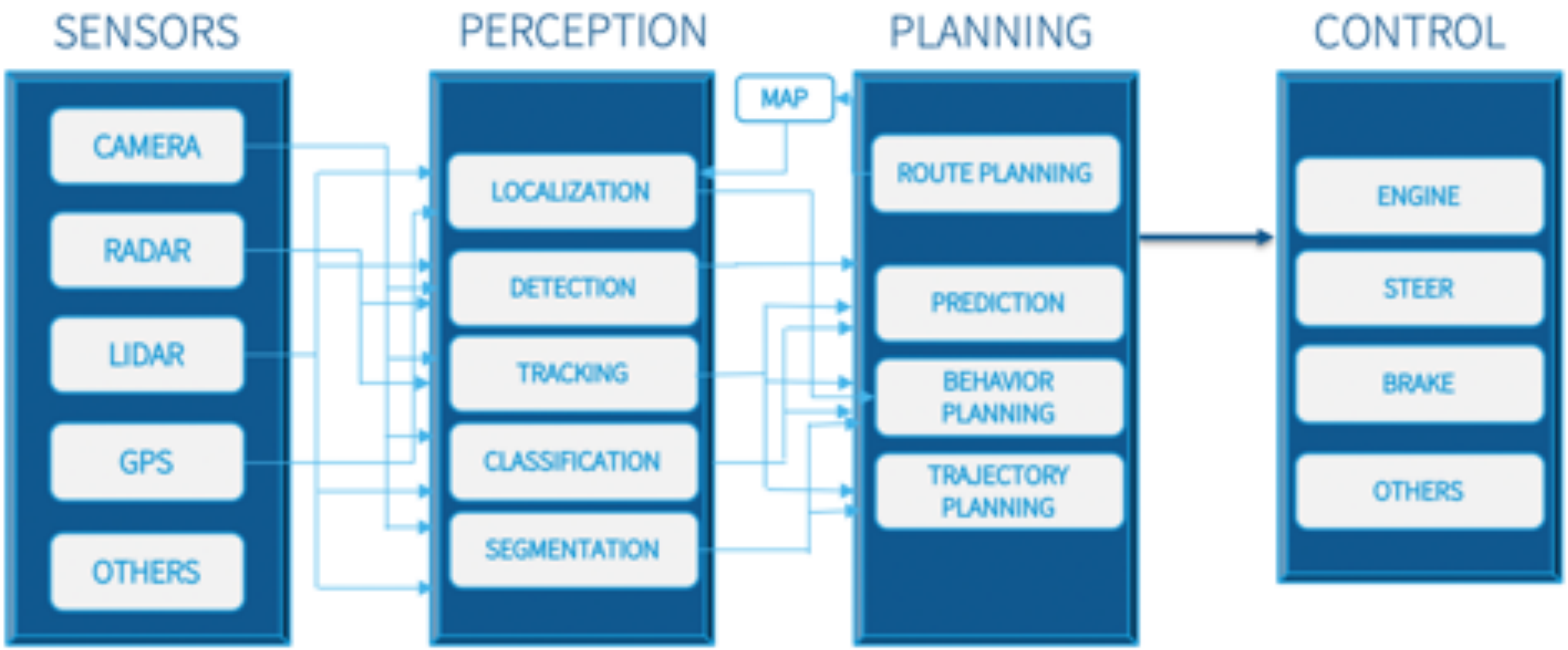
https://www.softwaretestingnews.co.uk/challenge-verifying-mission-critical-ethernet-network-performance-car/

A distributed architecture, in a highly complex, secured, safe system

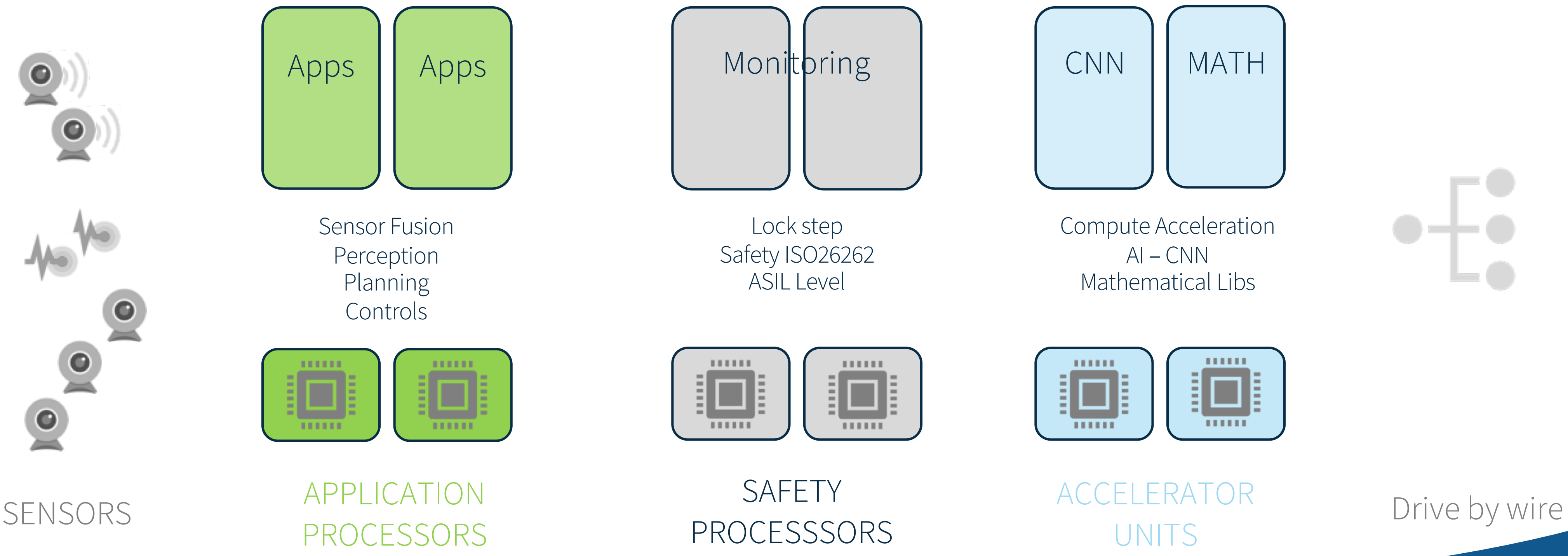


Focus of this lecture:
ADAS and AD ECU

ADAS/AD ECU FUNCTIONS

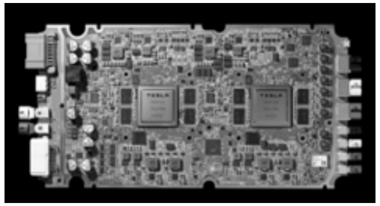


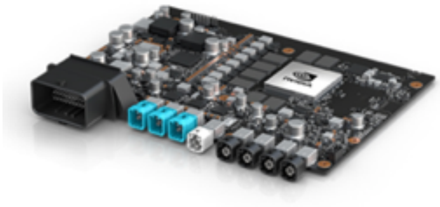
REQUIRED ECU COMPONENTS



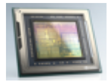
SPECTRUM OF BOARDS AND CHIPS FOR ECU

Production / Pre-production / Development Boards for Automotive ECU





Automotive Chips Categories



Heterogeneous Applications

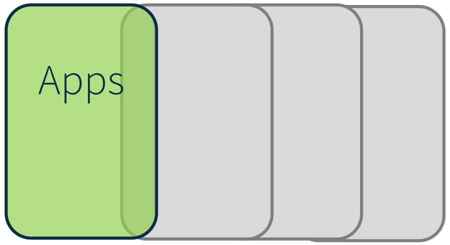
Dedicated Applications

Multi-Apps Accelerators

Safety dedicated

https://www.kalrayinc.com
https://www.nxp.com/design/development-boards/automotive-development-platforms/nxp-bluebox-autonomous-driving-development-platform:BLBX
https://www.nxp.com/products/processors-and-microcontrollers/arm-processors/s32-automotive-platform/s32g-processors-for-vehicle-networking:S32G274A
https://www.theverge.com/2019/4/22/18511594/tesla-new-self-driving-chip-is-here-and-this-is-your-best-look-yet
https://www.nvidia.com/en-us/self-driving-cars/

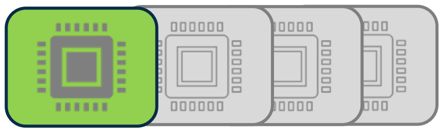
SUPPORTING SOFTWARE FOR ECU



Protocols and Communications — POSIX / DDS / ROS ...(ex:)

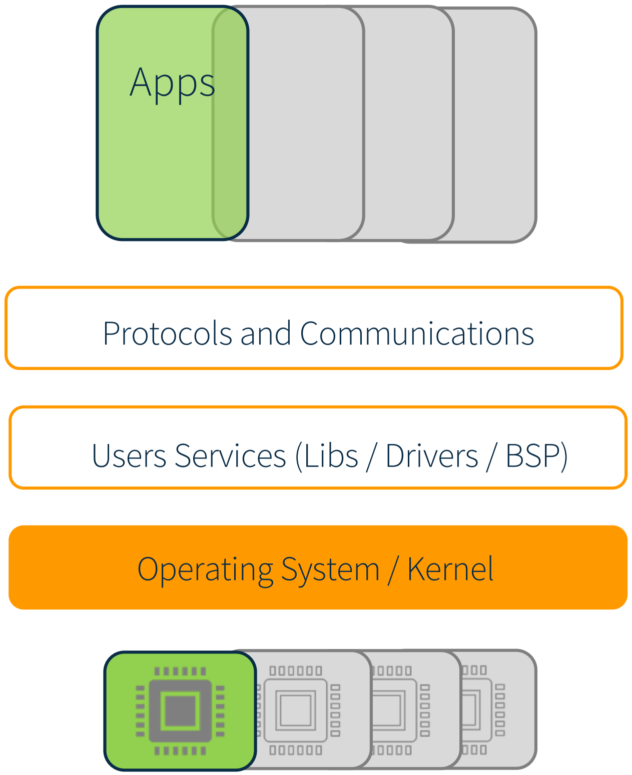
Users Services (Libs / Drivers / BSP) — Provide utilities and customization for ECU

Operating System / Kernel — Enable Hardware / Abstract multiple hardware



KALRAY

SUPPORTING SOFTWARE FOR ECU



POSIX / DDS / ROS …(ex:)

Provide utilities and customization for ECU

Enable Hardware / Abstract multiple hardware

KALRAY

Automotive Real-Time Operating Systems

Hardware Enabler for Developers

Autoware Courses#4

JUNE 2020

www.kalrayinc.com

SCOPE OF THE COURSE

In this lecture, we will identify

- What is an Operating System (OS)
- What makes an OS a Real Time Operating System: RTOS
- What are the key criteria to consider into a RTOS
- What are the main RTOS in the Automotive
- What *abstraction* levels are to be considered => POSIX / DDS / ROS / Virtualization

WHY A CHIP MAKER CARES ABOUT OS ?

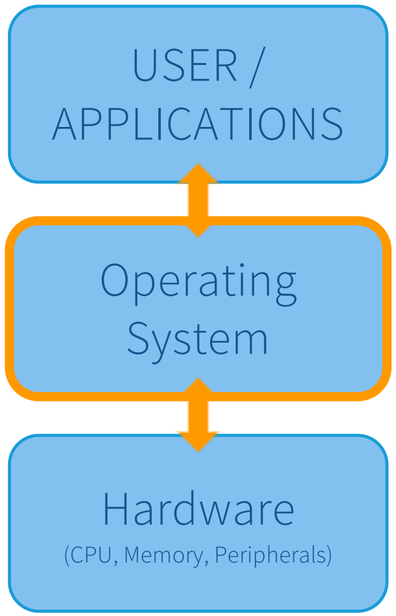
- The Operating System is the enabler of the Hardware
- This is the Software between the Hardware (Chip/ECU) and the User's Applications
- Adopting the right OS is key for providing the right Chip, depending on
  - Expected types of use cases (applications)
  - Expected types of programming (languages, system protocols)
  - Expected performance

INTRODUCTION

An Operating-what ?

OPERATION SYSTEMS

- For the user to take benefit of the hardware when developing, he needs an abstracted access to this hardware
  - Compute capabilities, Memory, peripherals
- Depending on the hardware, this abstraction can be straight forward (Micro-controller) or very complex (Security, heterogeneous CPU, memory hierarchy…)
  - From a single micro-processor 8-bit memory manager
  - To an Autonomous Driving Manycore 64 bits system considering multiple applications in parallel
- The Operating System's world is huge



KALRAY

FEATURES OVERVIEW

- An Operating System shall at least provide
  - Memory Management
  - I/O Management
  - Resources allocation
  - Error detection and handling
  - A kernel (or linked to a kernel) for
    - Task management: context switch scheduling, communication, synchronization
    - Interrupt management

Again, depending on your applications, your hardware, you need to tune your Operating System for your needs

KALRAY

Real-Time Operating System

"Not to confuse speed with haste"

FUNDAMENTALS

Embedded system requires predictable behavior

Non-deterministic

Soft Real-Time

No guarantee of time for task completion

Highly responsive to user's application

OS

Deterministic

Hard Real-Time

Guarantee of time for task completion

Highly responsive to external events

RTOS

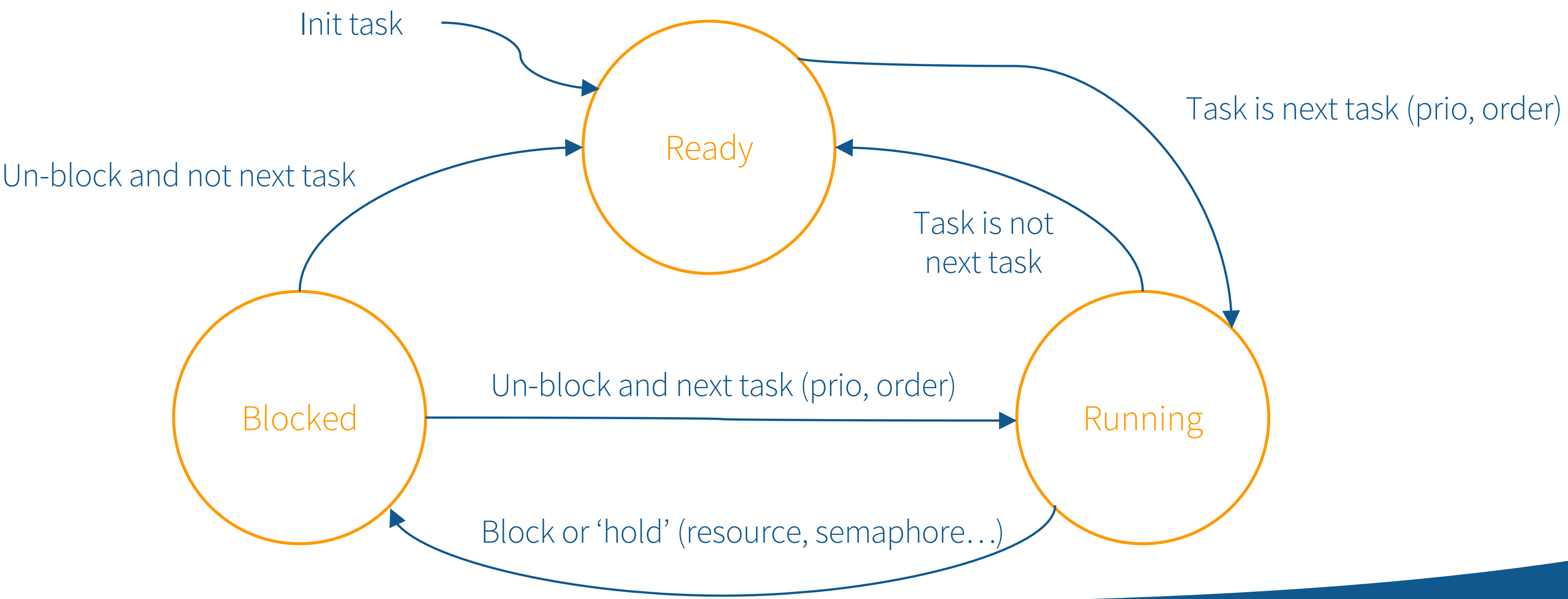
DETERMINISTIC

- This concept differentiates a real time programming from performance programming
- *Not to confuse speed with haste*
- The time interval between input event and output event must be predictable: the system always respond with a specified lapse of time
- Difficulties reside in running all system tasks, *each in a given time*: sharing available resources and available time of computation

To achieve determinism you need your OS to support several key features, which makes it a RTOS

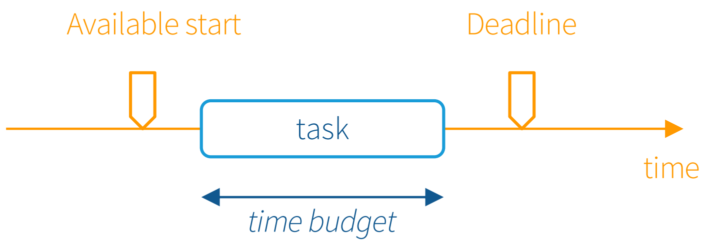
TASKS AND TASKS PRIORITIES

- In a RTOS, you will have many tasks to run, all time sensitive
- To enable hierarchy of tasks, you can rely on Task State and Priority



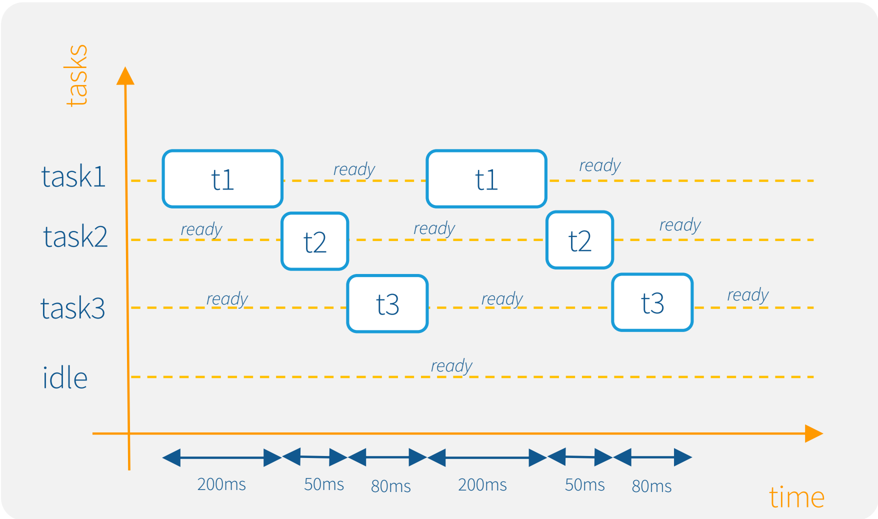
SCHEDULER

- The tasks to be executed within the RTOS must be carefully selected and “sequenced”
- “What is the appropriate next task to be loaded/run ?”



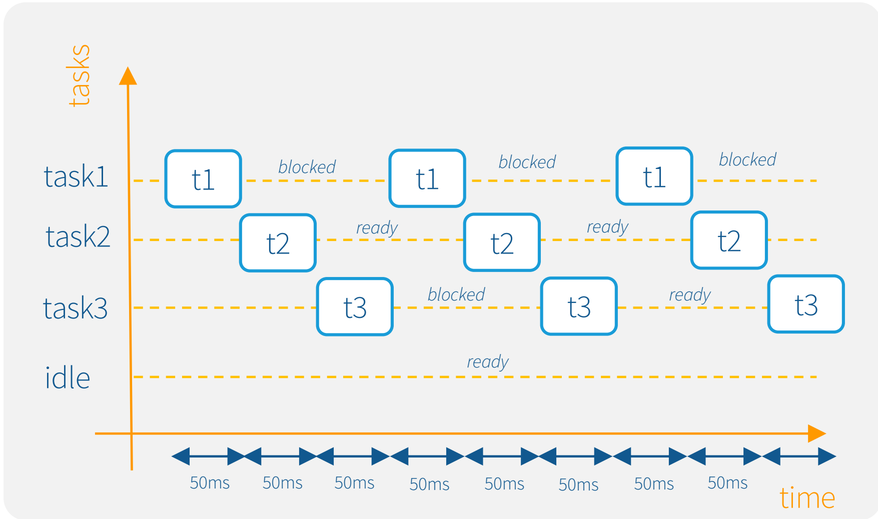
- Several scheduling algorithms, main ones are
  - Co-operative
  - Round-Robin
  - Pre-emptive
- Typical scheduling mechanism you will use is the pre-emptive algorithm

SCHEDULING MECHANISM



Cooperative

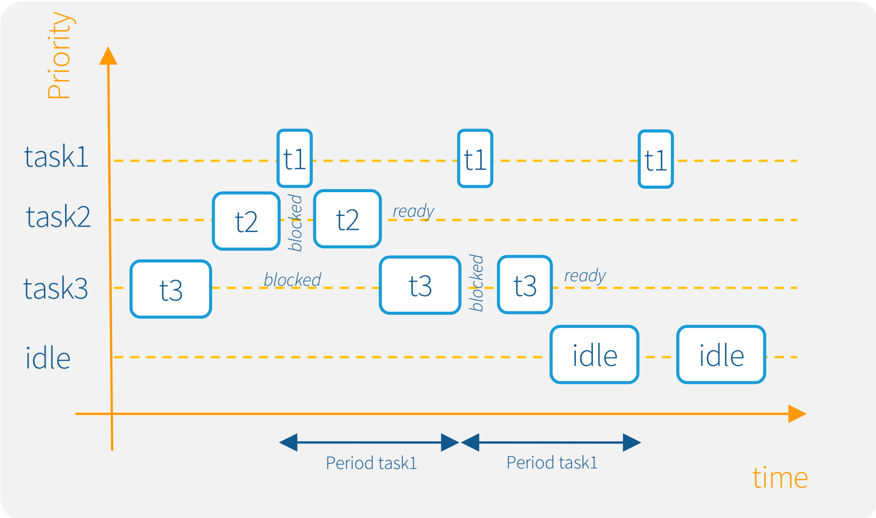
No preemption
Each task must be designed to executed to completion
Each task completes its workload



Round-robin

Each task is *preempted* after that *quantum* has elapsed
idle task could never runs
Every task is preempted *prior its completion* (most likely)

SCHEDULING MECHANISM



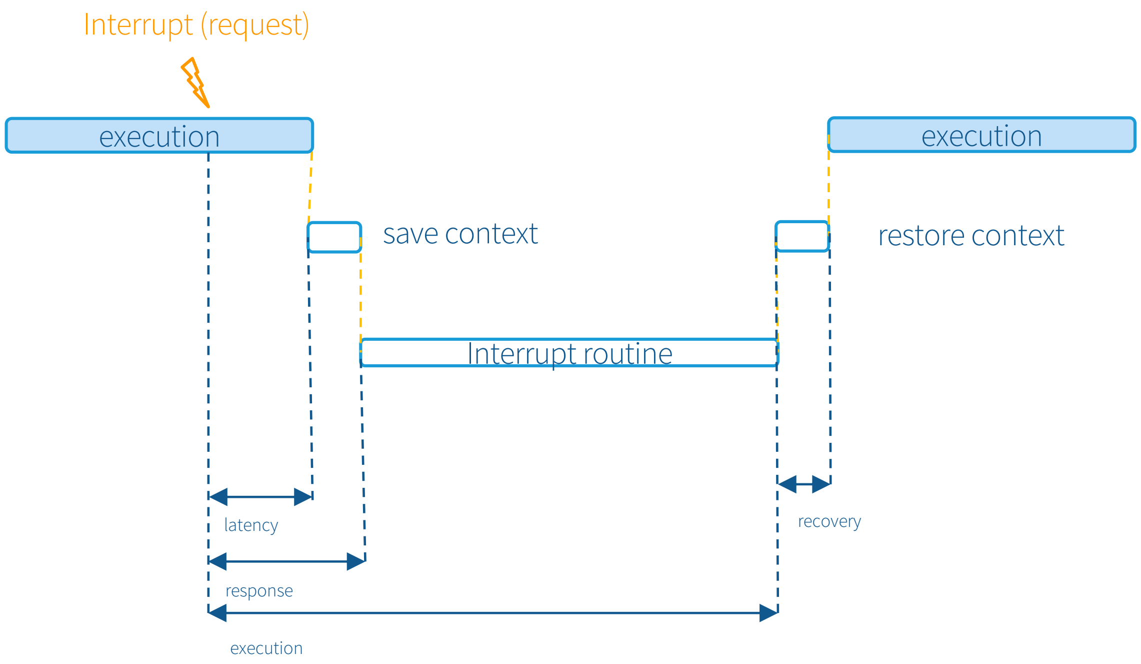
Pre-emptive

Task priority based
Ensure period of execution per tasks priority
Strong deterministic behavior depending on system design

INTERRUPTS

- An interrupt breaks the sequence of of operations
  - External Interrupts: generated by a peripherals
  - Internal Interrupts (or signals): special instruction in a program, or exception
- When an interrupt occurs
  - Suspend execution of the task
  - Save the context
  - Set the PC to start address of interrupt handler routine
  - Process the interrupt handler
  - Restore the context

INTERRUPTS : TIMING



Interrupt Latency

- Time between interrupt generation and the start of the handler execution

Interrupt Response

- Time between interrupt reception and the start of handler execution

Interrupt Recovery

- Time between handler completion and the context restored

Interrupt Execution Time

- Time between interrupt generation and the end of the handler execution

MEMORY MANAGEMENT

- Obviously, proper memory management is critical for Real-time systems
- A simple example is about memory allocation
  - Access to RAM or disk will generally generates pagefaults (avoiding run out of memory)
  - During a page fault, computation are hold while loading missing pages
  - This is unpredictable operation
- Another example is about dynamic memory allocation
  - Which can cause poor real-time performance (again pagefaults)
  - But also memory fragmentation, resulting in unpredictable amount of time to allocate memory for instance

KALRAY

MEMORY MANAGEMENT

- Obviously, proper memory management is critical for Real-time systems
- A simple example is about memory allocation
  - Access to RAM or disk will generally generates pagefaults (avoiding run out of memory)
  - During a page fault, computation are hold while loading missing pages
  - This is unpredictable operation
- Another example is about dynamic memory allocation
  - Which can cause poor real-time performance (again pagefaults)
  - But also memory fragmentation, resulting in unpredictable amount of time to allocate memory for instance

You must avoid pagefaults ☺

You must manage memory allocation carefully

RTOS MEMORY MANAGEMENT

A RTOS will take care of memory management

- Providing you with dedicated mechanism
- Allowing static memory allocation as needed
- Dedicated API
- Monitoring and debug capabilities

TIPS AND TRICKS ON LINUX

You can lock memory (prefault stack)

```
if (mlockall(MCL_CURRENT|MCL_FUTURE) == -1) {
  perror("mlockall failed");
  exit(-2);
}
unsigned char dummy[MAX_SAFE_STACK];

memset(dummy, 0, MAX_SAFE_STACK);
```

And allocate dynamic memory pool

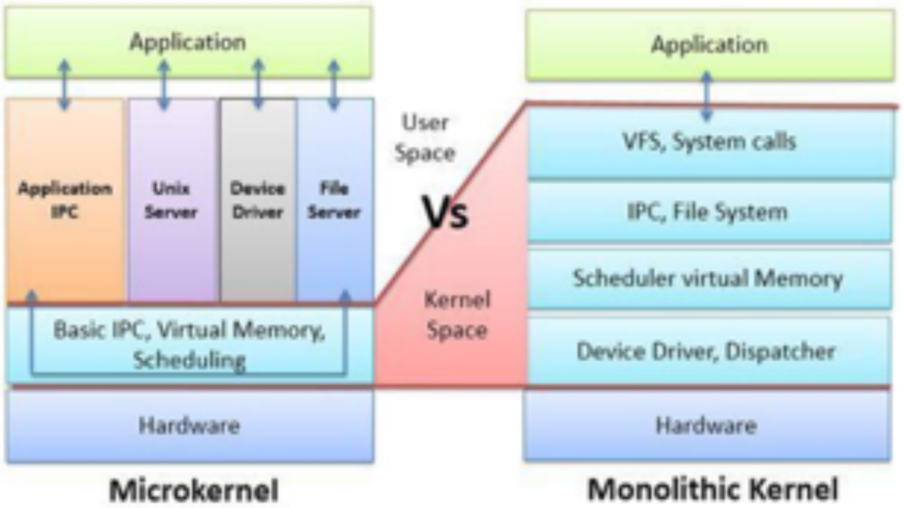
- Mostly providing real-time safe allocation
- Must accurately predict bounded memory size for the process!
- Using STL containers is therefore dangerous (unbounded sizes)
- In practice, only works for processes with small memory footprint

```
if (mlockall(MCL_CURRENT | MCL_FUTURE))
  perror("mlockall failed:");

/* Turn off malloc trimming.*/
mallopt(M_TRIM_THRESHOLD, -1);

/* Turn off mmap usage. */
mallopt(M_MMAP_MAX, 0);

page_size = sysconf(_SC_PAGESIZE);
buffer = malloc(SOMESIZE);

for (i=0; i < SOMESIZE; i+=page_size) {
  buffer[i] = 0;
}
free(buffer);
```

Thanks you ! https://design.ros2.org/articles/realtime_background.html

TYPES OF KERNELS

- You can consider two main types of kernel: Microkernel and Monolithic Kernel



Microkernel
User services and kernel services are not in the same address space

Monolithic Kernel
User services and kernel services are in the same address space

Picture from: https://techdifferences.com/difference-between-microkernel-and-monolithic-kernel.html

COMPARISON AND EXAMPLES

	Microkernel	Monolithic kernel	
Size	smaller	larger	
Execution	slower	faster	*Not to confuse speed with haste*
Stability	Unaffected by user services crash	Affected by user services crash	
Extendibility	Facilitated	More complex	
Debug	Facilated	More complex	*Arguable*
Examples	QNX, Integrity, eMCOS…	Linux, BSD…	

KALRAY

SPATIAL AND TEMPORAL ISOLATION

As ECU and even CPU are becoming more and more complex by being heterogeneous

- Management of such system requires sometimes more than one RTOS
  - To manage the diversity of hardware (CPU)
  - To manage the diversity of Operating Systems
  - To manage the sharing of common resources (devices)
  - To manage the application’s environments (including safety and security considerations)

Main concepts are

- Spatial Isolation
  - Mechanisms to partition access to resources: memory partitioning
- Temporal Isolation
  - Mechanisms to slice time execution on resources (CPU)

MECHANISMS

Hypervisor Type 2

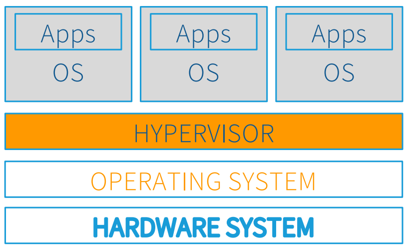
On top of your RTOS, you have an abstraction for applications can manage multiple OSes

Ex: Vmware

Mainly use in Servers/Desktop

Flexibility of OS/ applications distribution and maintenance

Costly to deploy



Hypervisor Type 1

A bare metal hypervisor, managing both spatial and temporal isolation

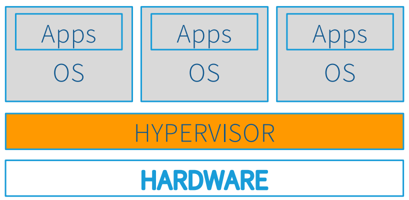
Ex: Mentor Grphic Hypervisor, sMCOS hypervisor (as OS extension), QNX hypervisor

Mainly use in Embedded systems

Usually dedicated implementation per CPU type

Complex to configure and to maintain

Complex to certify (security, safety)



Hardware Spatial Isolation

By hardware design, isolation of compute (CPU) and resources (device) are managed by MMU and MPU

Ex: Manycore MPPA

Easily configurable and maintainable

Enable safety and security

Tools for Temporal Isolation

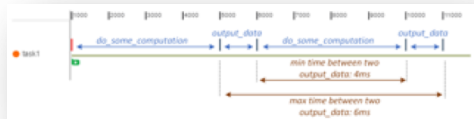
Tools for designing SW temporal behavior during development and runtime during execution

Ex: Asterios

Monitor temporal execution

Enabler of determinism

Facilitate design (MBD)



http://www.krono-safe.com/demystifying-the-psyc-language/

You can combine mechanisms !

EXAMPLE OF COMPLEX PERFORMANT SYSTEM ON CHIP

The « **FREE–FROM-INTERFERENCE**» & « **DETERMINISTIC** » & « **HIGH PERFORMANCE** » MPPA® architecture

Architecture

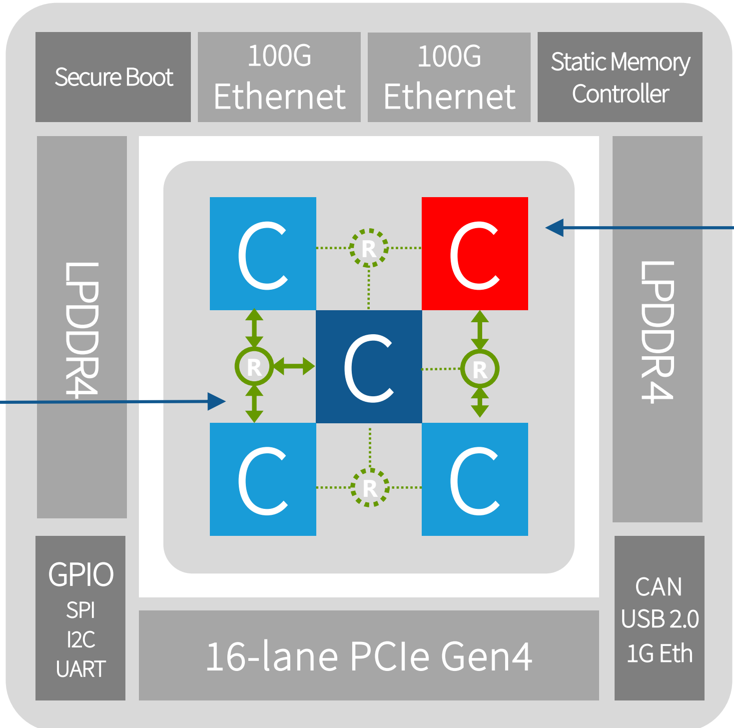
80 CPU cores with 80 Co-Processor

5 safety/security cores

On-chip Communication Fabric and Memory Hierarchy

Ensuring Spatial Isolation

ISO26262 ASIL B



Compute Cluster

16 CPU 64-bit cores

16 Co-processor

Safety/Security 64-bit core

Core/Co-processor Association

Multiple arithmetic

From 8-bit to 64-bit

Integer to Floating Point

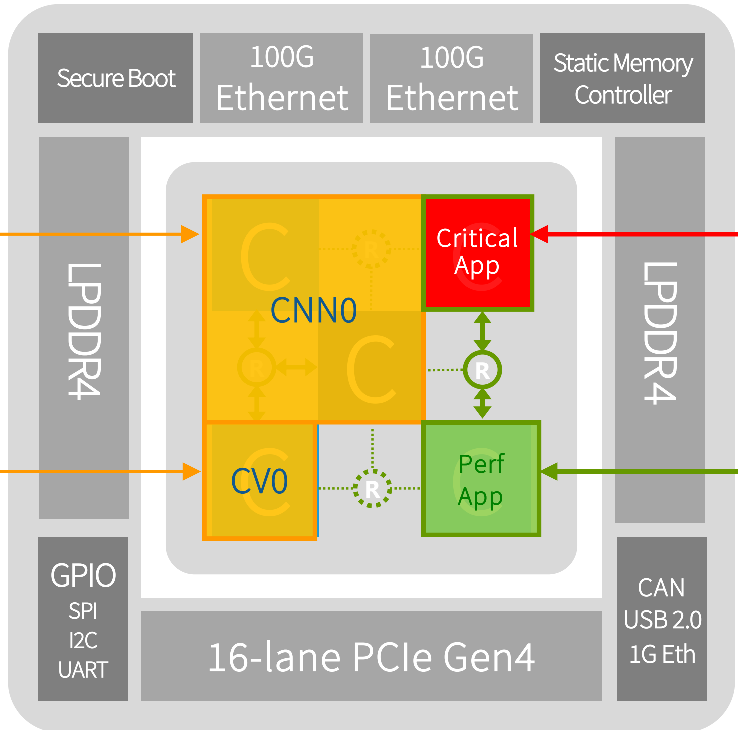
C Cluster

EXAMPLE OF COMPLEX PERFORMANT SYSTEM ON CHIP

The « **FREE–FROM-INTERFERENCE**» & « **DETERMINISTIC** » & « **HIGH PERFORMANCE** » MPPA® architecture

High Performance Compute Accelerations
CNN

High Performance Compute Accelerations
Computer Vision



Hard Real Time OS eMCOS
Temporal Isolation with Asterios

Rich OS or RTOS
Linux, eMCOS, Integrity…

C Cluster

KALRAY

MAIN RTOS IN AUTOMOTIVE (ADAS/AD)

Company	RTOS
QNX-Blackberry	QNX
Wind River	VxWorks
GreenHills	Integrity
Mentor Graphic	Nucleus
eSOL	eMCOS
Krono-Safe	Asterios
SysGo	PikeOS

YOUR FOCUS

- So prior moving to protocols layers, on top of the RTOS such as DDS, ROS
- So prior moving to Applications, such as Autoware AD apps

1. Define your use cases
2. Define your system topology
3. Define your determinism strategy
4. Select your hardware
5. Select your RTOS with dependencies on latency, libraries…
6. Set priority task carefully
7. Select and Setup the scheduler
8. Be careful about your interrupt handler
9. Enjoy !

THANK YOU FOR YOUR ATTENTION

and stay safe

THANK YOU

KALRAY S.A. - GRENOBLE - FRANCE
180, avenue de l'Europe
38 330 Montbonnot - France
Tel: +33 (0)4 76 18 90 71
email: info@kalrayinc.com

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

KALRAY