TASTE Multi-core

ISAE / ONERA

Jérôme Hugues / Claire Pagetti

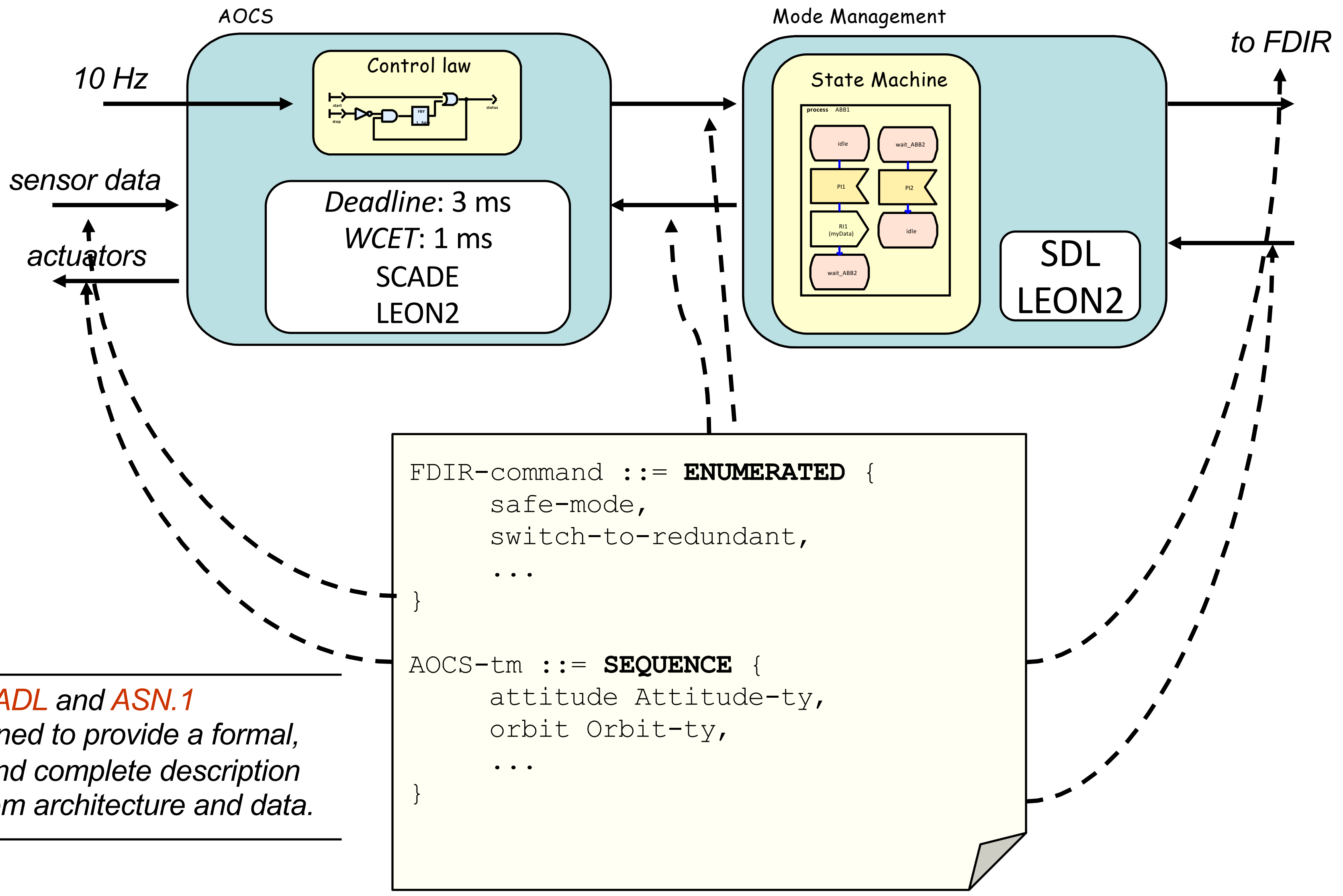
December 2016

General information

- **Duration of the project: 12 months**
- **Consortium**
  - Combined expertise in TASTE toolset, programming of multicore systems, including RTEMS and Xtratum
  - J. Hugues, ISAE: member of the TASTE project since its inception, expert on AADL, lead of the Ocarina project
  - C. Pagetti, ONERA: expertise in the design and implementation of safety-critical applications on multicore systems
  - E. Noulard, ONERA: expertise in low-level programming and RTOS, expertise in many/multicore systems
- **Global effort of 2.5 man months**

Outline

TASTE process in a nutshell

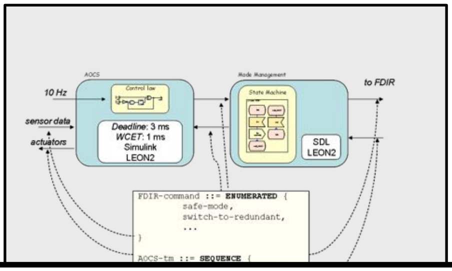


```
FDIR-command ::= ENUMERATED {
     safe-mode,
     switch-to-redundant,
     ...
}

AOCS-tm ::= SEQUENCE {
     attitude Attitude-ty,
     orbit Orbit-ty,
     ...
}
```

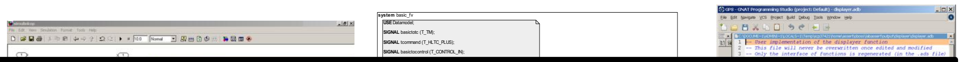
AADL and ASN.1
are combined to provide a formal, precise, and complete description of the system architecture and data.

TASTE process in a nutshell



```
FDIR-command ::= ENUMERATED {
        safe-mode,
        switch-to-redundant,
        ...
}
AOCS-tm ::= SEQUENCE {
```

① Generate "application skeletons" in Simulink, SDL, C, and Ada

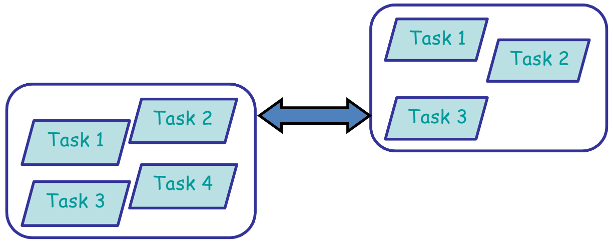


All these steps are **automated**, thanks

- Languages with good power of expression
  - AADL for architecture, ASN.1 for data typing,
  - SDL, Simulink, SCADE, C, Ada, etc. for behavior
- Tool to support this approach
  - TASTE toolchain (editors, code generators, orchestrator)

In the following, we focus in the Concurrency view level, leveraging AADL

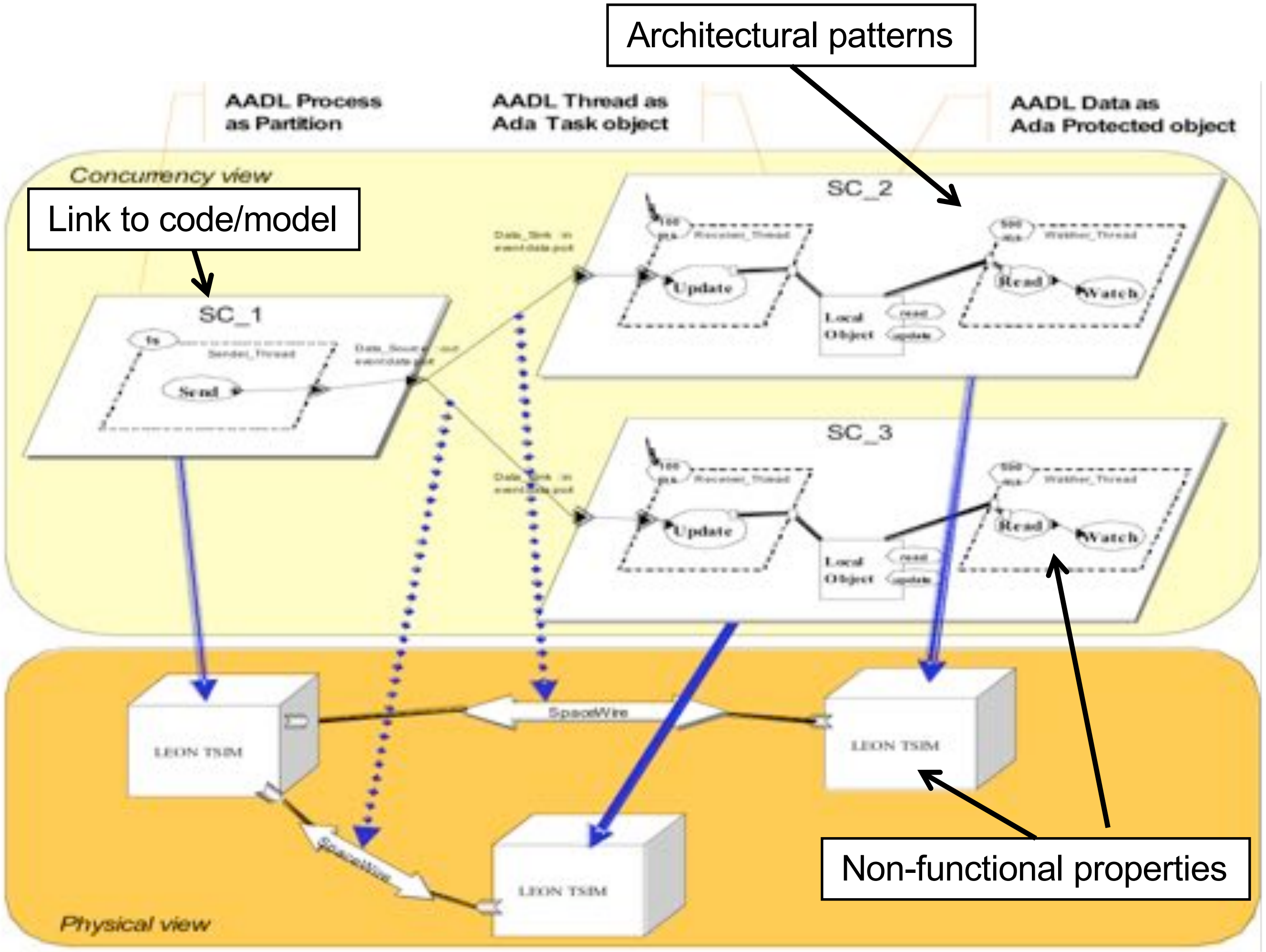
real-time architecture



to put everything together on a real-time operating system

Research on AADL @ ISAE

Architecture helps you focusing on the actual system



AADL covers many parts of the V cycle: model checking, scheduling, safety and reliability and code generation

Lead on the Ocarina toolset

Development of AADL:

4 books, tutorials, 30+ papers

Code generation :

Ada, C (POSIX, ARINC653, RTEMS)

TRL 6-7 with ESA (ECSS E-40)

SPARK, ACSL **TRL 2-3**

Scheduling: Cheddar, MAST

External metrics: stack usage (gnatstack), WCET (Bound-T)

TRL 4-5 with ESA

Architectural Constraints/Requirements checks

TRL 6, being standardized

Model checking: Petri Nets, LNT

TRL 2 (PhD contributions)

System engineering: SysML, Capella **TRL 2-3 (with IRT-SE)**

Ocarina: an AADL code generator
http://www.openaadl.org

> **Ocarina is a stand-alone tool for processing AADL models**
  - » Free Open Source Software (as in *Free* speech and *Free* beer)
  - » Command-line, or integrated third-party tools
    - OSATE (CMU/SEI), TASTE (ESA), AADL Inspector (Ellidiss)

> **Code generation facilities target PolyORB-HI runtimes**
  - » Ada HI integrity profiles, with Ada native and bare board runtimes
  - » C POSIX or RTEMS, for RTOS & Embedded
  - » C ARINC653 for avionics systems

> **Generated code quality tested in various contexts**
  - » For WCET exploration, support for device drivers, …

> **Written to meet most High-Integrity requirements**
  - » Follow Ravenscar model of computations, static configuration of all elements (memory, buffers, tasks, drivers, etc.)

> **Contributions from PhD students, partners (SEI, ESA, ENIS)**

Outline

Objectives

> **Main objective: provide an implementation strategy to support multicore systems** in TASTE, for both regular and Time/Space Paritionning OS
  - » Evaluate extenstions to core technologies (AADL) and editors
  - » Use cases as driving example

> **Inputs**
  - » TASTE: Support for mono-core demonstrated, for POSIX, RTEMS
  - » AADL: Support for multicore in discussion, TSP supported (ARINC653 annex)

> **Leverage public use cases**
  - » Adapt them for AADL,
    - code generation for RTEMS + POSIX
    - Generation of configuration files for XtratuM
  - » Provide manual implementation for XtratuM
  - » Discuss modeling patterns for TASTE graphical tools

Outline

1. **Introduction**

2. **Project inputs**
   - » **Executive layers: RTEMS & XtratuM**
   - » Use cases: ROSACE & GCU

3. **TASTE multi-core**

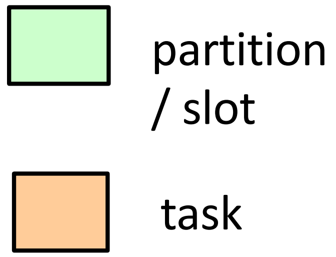
4. **Conclusion & perspectives**

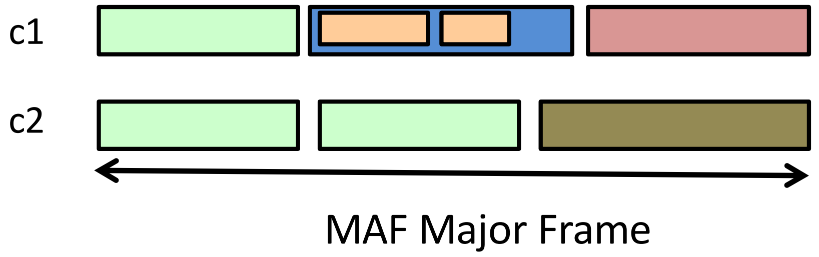
Executive layers: RTEMS & XtratuM

> **RTEMS supports SMP architecture as part of RTEMS4.12**
  - » No direct support for neither AMP (except through explicit multi-processing calls), nor TSP configurations
  - » Support for drivers extensive, through direct contributions form Gaisler who also implement the corresponding IP blocks
    - Same drivers adapted from GR-RASTA, GR712RC and GR740 systems
    - Support for: UART, Ethernet, SpaceWire, MIL1553 and CAN

XtratuM

- **TSP (time and space partitioning) & multi-core for LEON3MP & NGMP**
  - support for TSP mode by construction, in mono or multi-core;
  - drivers support is separated from kernel, on demand from FENTISS
- **Programming**
  - Application = one or several partitions
  - Partition = one or several slots, each with a start time and a length.
  - Slot = execution of several tasks
- **Configuration (or plan)**
  - MAF, length of repetition
  - a set of slots
  - static mapping





Outline

Two use cases

> **ROSACE (Research Open-Source Avionics and Control Engineering)**
- » C. Pagetti, D. Saussié, R. Gratia, E. Noulard and P. Siron. “The ROSACE Case Study : From Simulink Specification to Multi/Many-Core Execution”. In : 20th IEEE RTAS 2014
- » Repository URL: https://svn.onera.fr/schedmcore/branches/ROSACE_CaseStudy
- » Content
  - the SIMULINK specification (folder simulink)
  - a checker to verify that an implementation fulfills the high level properties (folder checker)
  - several implementations

> **GCU (Gestionnaire de Charge Utile / Payload Data Management System)**
- » Spacify consortium, « Etude de cas CNES : Modélisation Synoptic de la partie Commande / Contrôle du GCU (Gestionnaire de Charge Utile) », report 2010.
- » Content
  - Partial specification in Synoptic

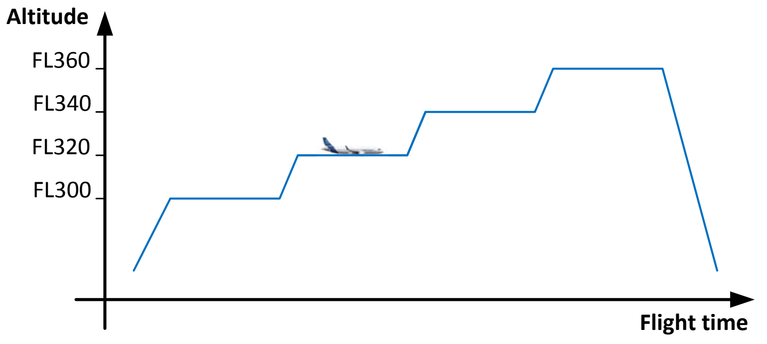
Implementation for TASTE multi-core

> **Implementation**
  - » Common functional C code for all multi-core implementations
  - » Manual computation of off-line mapping
  - » Specific C code encapsulation for each paradigm model

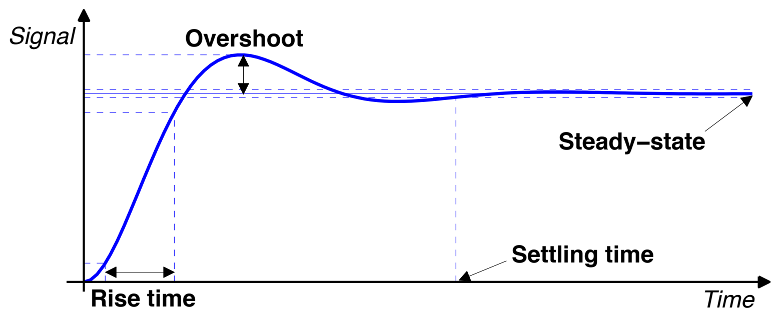
> **Execution settings**
  - » Targets: Zynq board, LEON3MP
  - » Test case

Avionic use case: Longitudinal Flight Controller

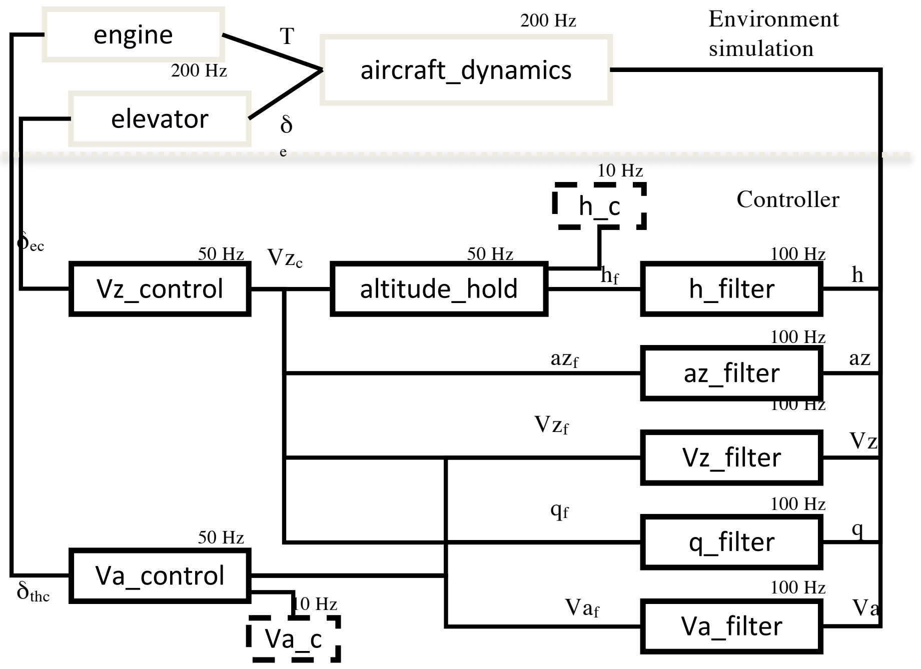
> **Longitudinal motion of a medium-range civil aircraft in *en-route* phase**
>
> » *Cruise:* maintains a constant altitude h and a constant airspeed Va
>
> » *Change of cruise level* subphases:
>
> commands a constant vertical speed Vz (rate of climb)
>
> - Ex: FL300 ⟶ FL320 ⟶ FL340 ⟶ FL360
> - FL300 = pressure altitude of 30000 ft



> **Test case**
>
> » **Change flight level to 11 000 feet**
>
> » **Expected results**
>
> −**P1 settling time**: ≤ 10s
>
> −**P2 overshoot:** ≤ 10%
>
> −**P3 rise time:** ≤ 6s



Longitudinal flight controller architecture



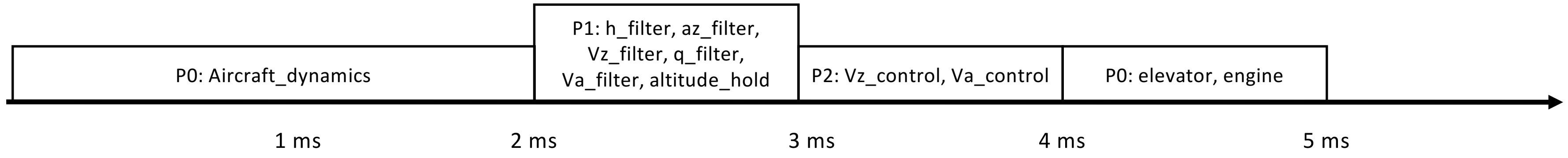
Flight condition:
h = 10000 m, Va = 230 m/s

Outputs	V_z	vertical speed
	V_a	true airspeed
	h	altitude
	a_z	vertical acceleration
	q	pitch rate
Filtered outputs	V_{z_f}	vertical speed
	V_{a_f}	true airspeed
	h_f	altitude
	a_{z_f}	vertical acceleration
	q_f	pitch rate
Reference inputs	h_c	altitude command
	V_{a_c}	airspeed command
Commanded inputs	V_{z_c}	vertical speed command
	δ_{e_c}	elevator deflection command
	δ_{th_c}	throttle command
Aircraft inputs	δ_{e_c}	elevator deflection
	T	engine thrust

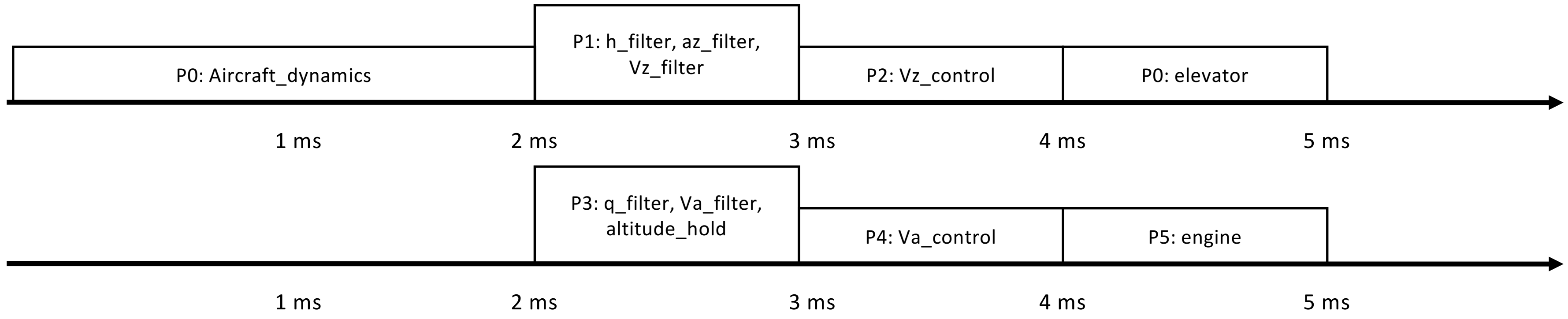
- 5 filters consolidate the measured outputs provided by the sensors
- 3 controllers track accurately: altitude (h_c), vertical speed (V_{zc}) and airspeed commands (V_{ac})
- rate choices
  1. for controllers:
     - closed-loop system with the continuous-time controller can tolerate a pure time delay of 1 s before destabilizing ➔ sampling period ≤ 1 Hz
     - performances ➔ sampling period ≤ 10 Hz
  2. for environment: 200 Hz to model a continuous-time phenomenon

Off-line mapping for ROSACE

Mono-core schedule


P0: Aircraft_dynamics
P1: h_filter, az_filter, Vz_filter, q_filter, Va_filter, altitude_hold
P2: Vz_control, Va_control
P0: elevator, engine
1 ms
2 ms
3 ms
4 ms
5 ms

Dual-core scheduling



Specifics in coding

- **RTEMS**
  - POSIX implementation

- **XtratuM**
  - Communication between partitions are done via XtratuM sampling ports
  - Code skeleton

```
void PartitionMain (void){
     ... init …  while (1) {
    aircraft_dynamics(delta_e, T, &res);
    send_buf(res.h, porth);
    …
    delta_th_c = rec_buf(portdthc);
    T = engine(delta_th_c);
     delta_e_c = rec_buf(portdec);
    delta_e = elevator(delta_e_c);
     XM_idle_self();}
}
```

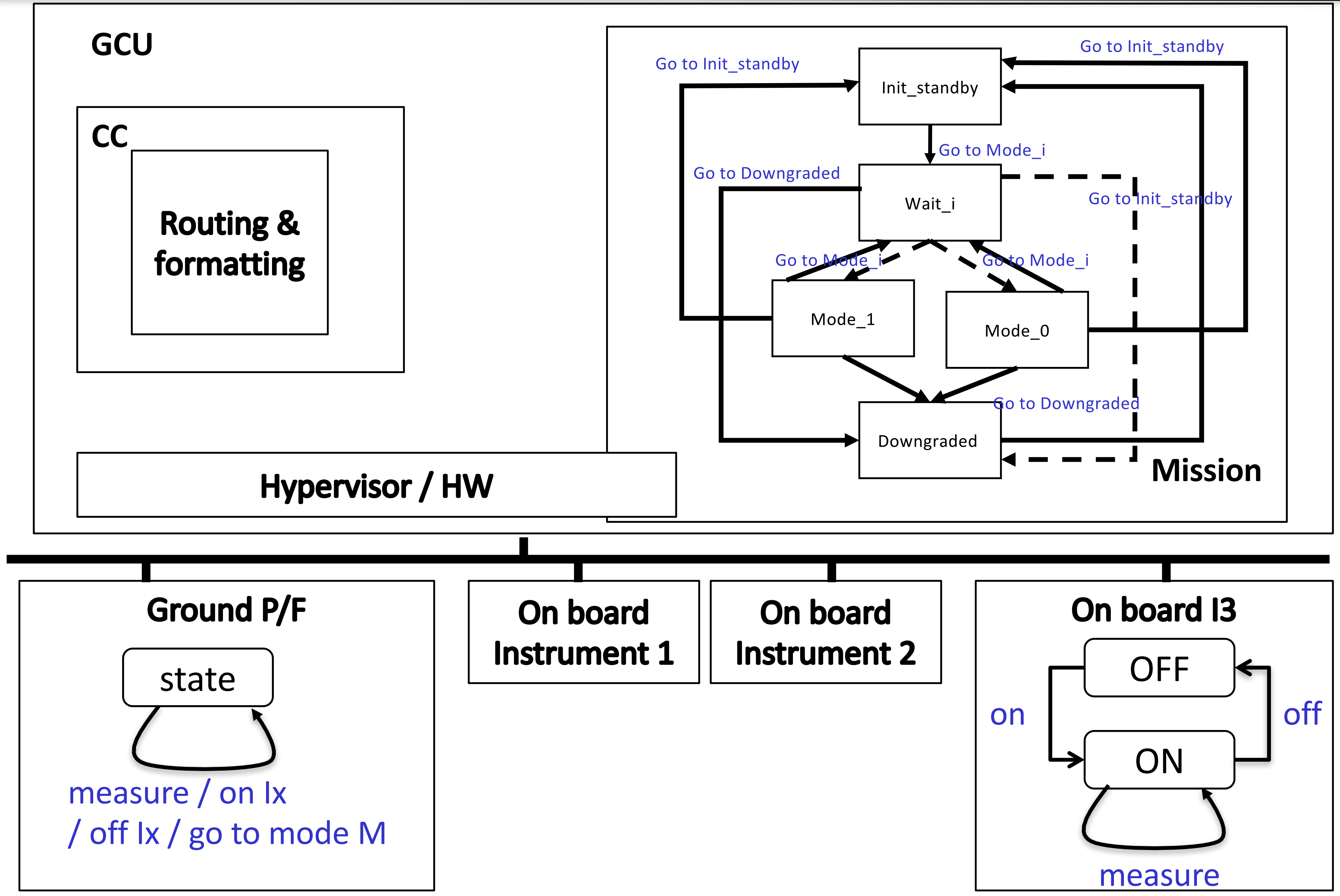
- **log passed the checker**

Space use case: Payload Data Management System

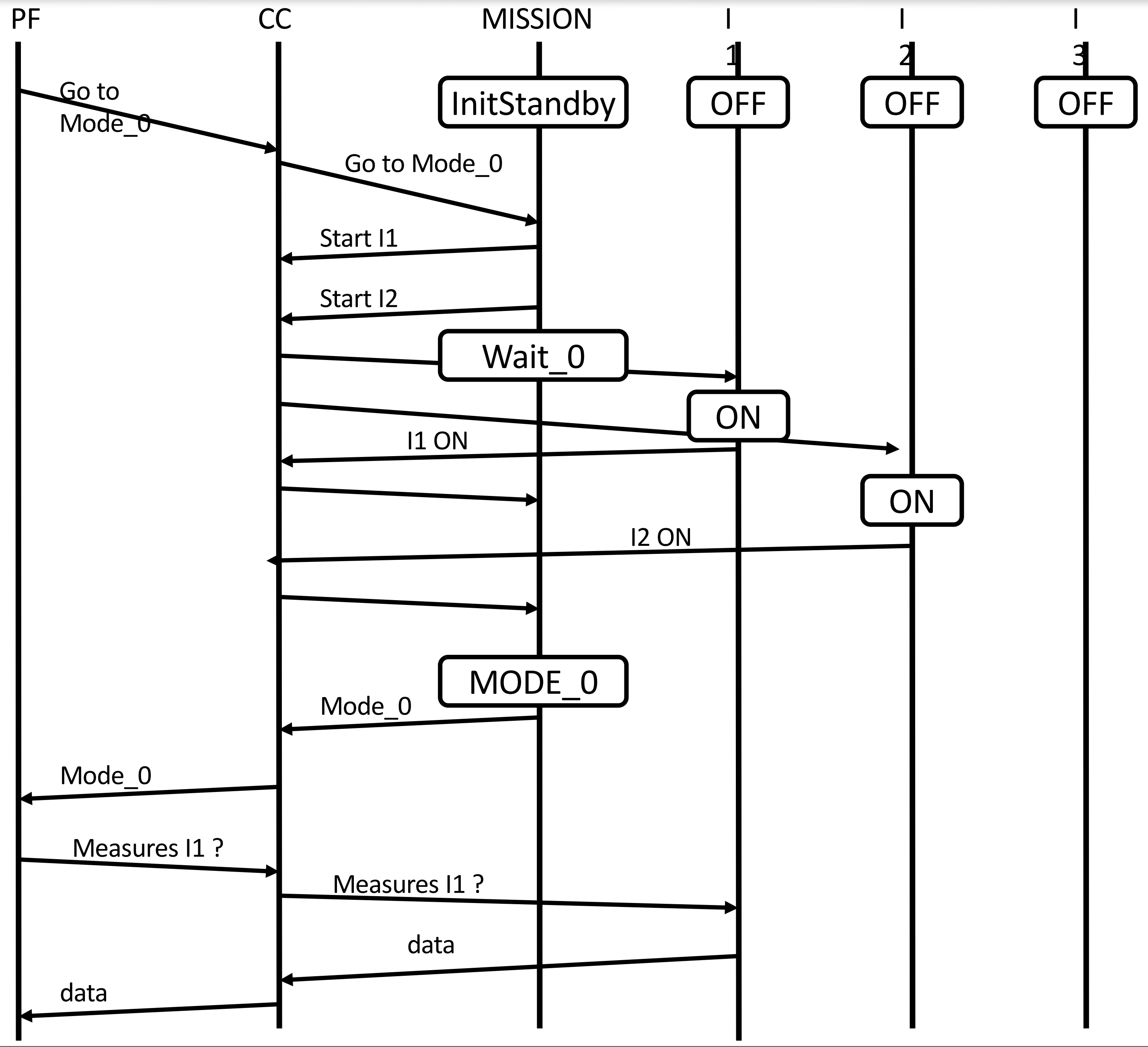
> apply commands from the ground to move in a given mode and to confirm to the ground that requests have been correctly applied

> event triggered architecture

GCU architecture

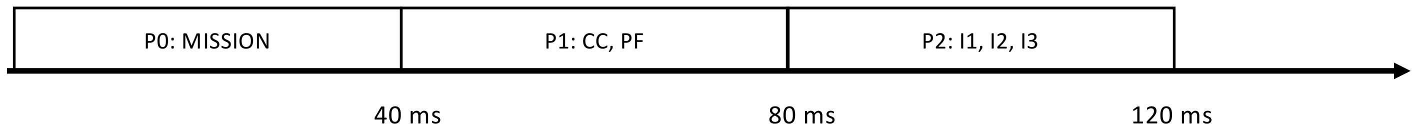


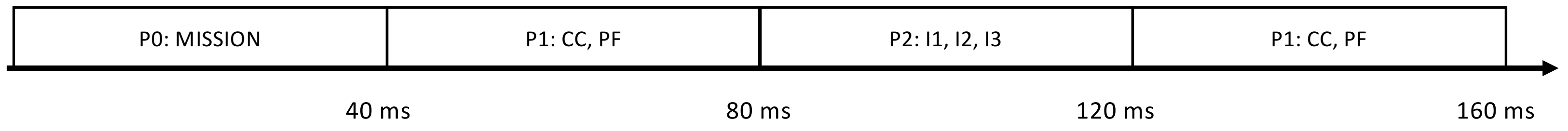
Example of GCU test case



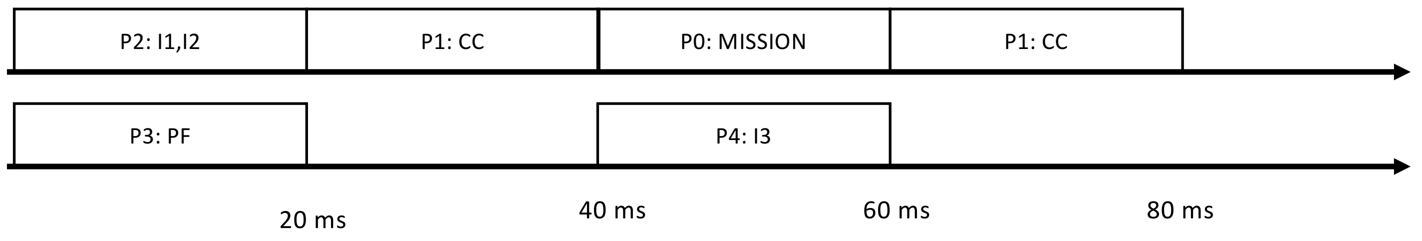
Off-line mapping for GCU

> **Two mono-core schedules**





> **One dual-core schedule**

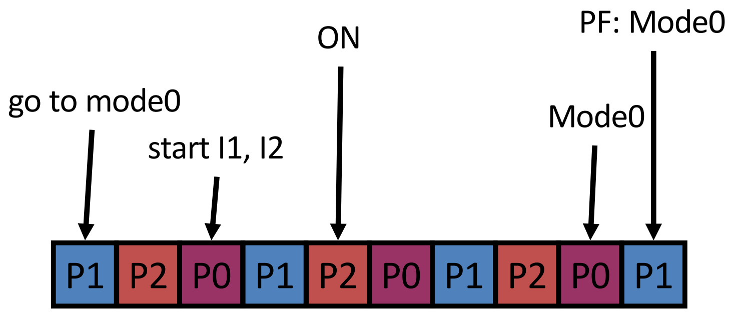


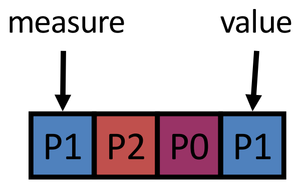
Specifics in coding

- **RTEMS**
  - POSIX implementation
- **XtratuM**
  - Communication between partitions are done via XtratuM sampling ports
  - Code skeleton (similar ROSACE)
- **log compliant with expected behaviours**

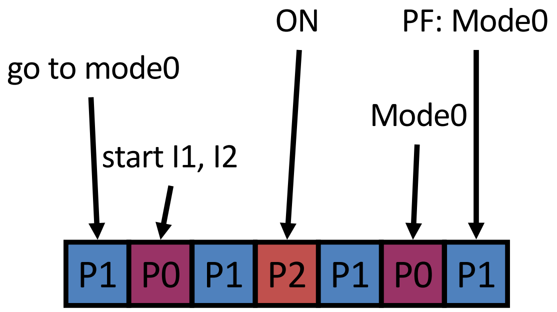
Response times for the schedules

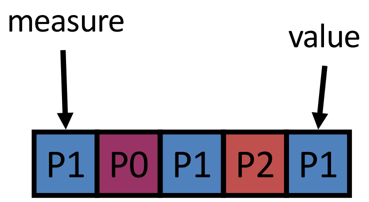
> **Mono-core schedule 1**





> **Mono-core schedule 2**

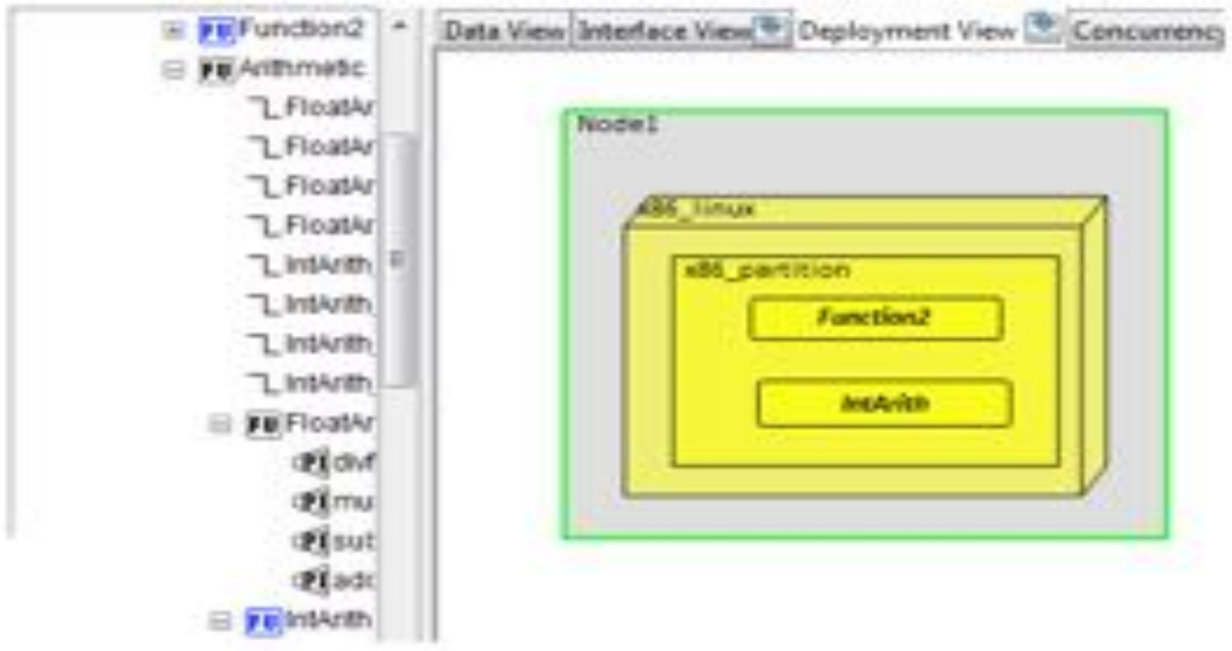




Outline

About the TASTE-DV

- **The TASTE Deployment View gathers**
  - The definition of the hardware platform
  - Binding of functions to this platform
- **Concepts: node (OS), processor, partition (memory space)**
  - Platform read from library of AADL models
  - Shared patterns between TASTE-*V and Ocarina



Definition of multicore processors: AADL

> **Ongoing work as part of AADLv3 work**

```
virtual processor a_core end a_core;
virtual processor implementation a_core.impl end a_core.impl;
processor implementation POSIX_CPU.Cores4
subcomponents
        Cpu0 : virtual processor a_core.impl {Processor_Properties::Core_Id => 0;};
        Cpu1 : virtual processor a_core.impl {Processor_Properties::Core_Id => 1;};
-- ...
end POSIX_CPU.Cores4;
```

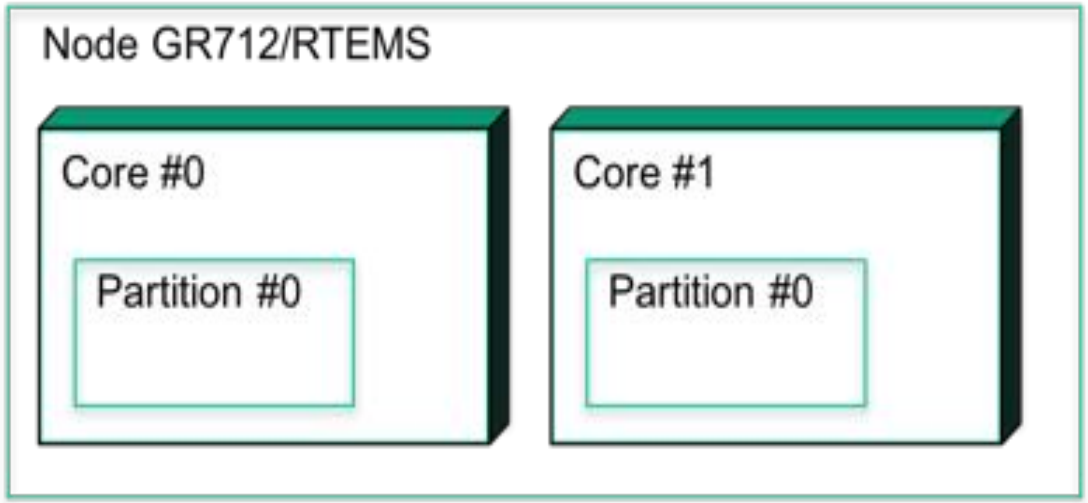
> **Add one property to specify id of the core**

```
-- Core #1 binding
Actual_Processor_Binding => (reference (Hardware.Cpu1))
   applies to Software.Aircraft_Dynamics;
```

- » TASTE-CV/AADL: support done for POSIX, RTEMS
- » TASTE-DV: needs additional support to capture bindings

Definition of multicore processors: TASTE-*V

- **Extensions of existing TASTE-DV concepts**
  - One node, multiple cores inside, one partition per core
- **Could be implemented by**
  - Updating parser of TASTE-DV to support AADL pattern
  - Extending library of AADL models with reference design
    - Natural candidates: GR712RC, NGMP, ARM A9, etc.



Definition of TSP: memory

> **Process can be bound to specific memory location**
  - » Supported as part of regular AADLv2 language

```
memory implementation myram.sdram extends myram.stram
subcomponents
  segment1 : memory segment.i {Base_Address => 16#40100000#;
                               Byte_Count => 524_288;};
  segment2 : memory segment.i {Base_Address => 16#40180000#;
                               Byte_Count => 524_288;};
end myram.sdram;
```

> **Impact on TASTE**
  - » Additional tabular editor to capture parameters for a node
  - » Additional legality rules to check configuration is sound
    - • Already done use REAL annex language in UML&AADL'10
  - » Supported as part of XtratuM configuration backend

Definition of TSP: partitions – AADL

- **Follow ARINC653 annex document**

```
processor implementation leon3.xtratum_partitions extends leon3.xtratum
subcomponents
   P0 : virtual processor xtratum_partition.generic
      { Deployment::Execution_Platform => LEON3_XM3;
        ARINC653::Partition_Identifier => 0;
        ARINC653::Partition_Name       => "P0";    };
-- ...
```

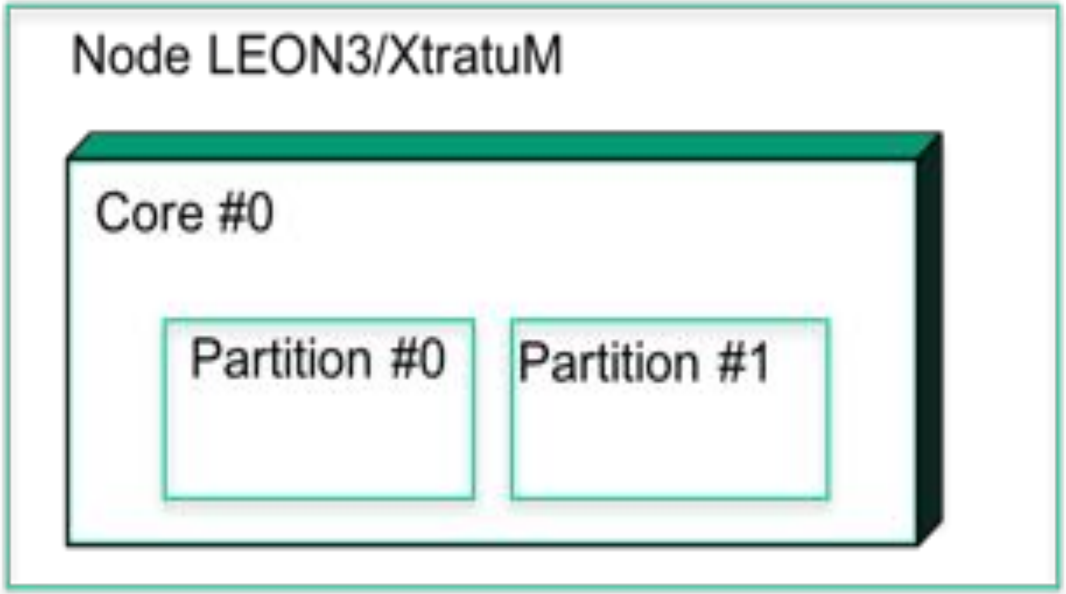
- **Same impact as memory partition on TASTE toolchain**

```
properties
   ARINC653::Module_Schedule =>   ( [Partition => reference (P0);
                                     Duration  => 2 ms;
                                     Periodic_Processing_Start => true;],
-- ...
```

Outline

Definition of TSP: partitions – TASTE-DV

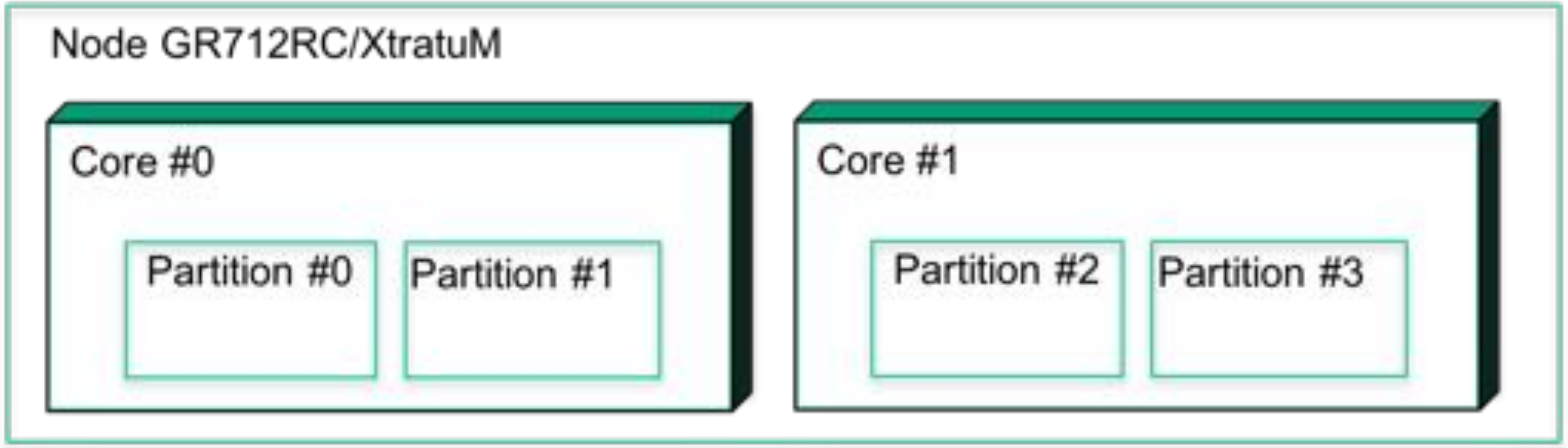
- **Extensions of existing TASTE-DV concepts**
  - One node, one core, no partition (default), capability to add partitions
- **Could be implemented by**
  - Updating parser of TASTE-DV to support AADL pattern
  - Tabular editor to configure TSP parameters, attached to the node (definition of the RTOS abstraction)
- **GUI: need a graphical icon for TSP configuration, to distinguish from regular non-TSP**



Combining SMP and TSP

> **Done as a combination of the previous patterns**
  - » AADL: use virtual processors for either core or partitions
  - » TASTE-DV: combine previous approaches

> **Same recommendations as previously**
  - » # of cores is static, one editor per core to time config., one editor per node for memory



Other topics

> **For now, focused mostly on modeling TSP/multicore**

> **Initial study illustrated needs to constraint designs**

  » Exactly one core per thread

  » Exactly one core per partition

  » Verification of TSP configuration

  » Must add a specific design checker in TASTE. For the moment, it is part of the vertical transformation, too late !

> **Also, need to perform optimizations of designs**

  » E.g. place threads to ensure schedulability

  » Sequence of partitions to optimize latency

> **Call for a specific TASTE-Configurator tool**

Ocarina components for Multicore

- **Updated property sets**
  - ARINC653 (new release, per AS5506/1A) for TSP
  - Property for specifying cores for multi-core
- **Updated PolyORB-HI/C runtime**
  - Support for multicore for RTEMS and RT-POSIX
- **Updated backends**
  - New properties
  - PolyORB-HI/C: consider SMP
  - XtratuM configuration: updates for SMP and support of 4.2.1
- **Integrated to GitHub and Gitlab (ESA) master branch**

Outline

Implementation work

> **For ROSACE and GCU, delivery of**

> **XtratuM manual implementation, tested on ARM9 2 cores (ONERA)**

> **AADL/TASTE-CV implementation done for the following configurations**

- » RT-POSIX case, 1-core and 4-core: code generation and execution
- » RTEMS (mono-core + SMP): code generation and execution
- » XtratuM (mono-core + SMP): configuration generation, compilation of XML generated by XtratuM tools
  - Note: support of a RTEMS BSP for XtratuM in PolyORB-HI/C could not be tested, status unknown
- » Log reports provided as reference output from XtratuM runs

> **Addition of communication: target altitude configured from an external source: delayed due to lack of SpaceWire in RTEMS/SMP**

- » Mitigation solution is to implement these outside of this study, in the scope of PERASPERA once RTEMS 4.11 stabilizes

Outline

1. Introduction
2. Project inputs
3. TASTE multi-core
4. Conclusion & perspectives

Conclusion

> **Study defined two case studies to evaluate support for SMP and TSP in TASTE**

> **Case studies helped to**

- » Derive requirements for TASTE-DV updates
  - • Ellidiss confident these could be implemented using the existing technology
- » Consolidate support of SMP and TSP in TASTE-CV/AADL
  - • Front-end, property sets and backends prepared
    - – SMP for RTEMS and POSIX;TSP for Xtratum
    - – Code generation update for RTEMS and POSIX
    - – Generation of configuration for XtratuM

> **All contributions will be integrated to TASTE VM**