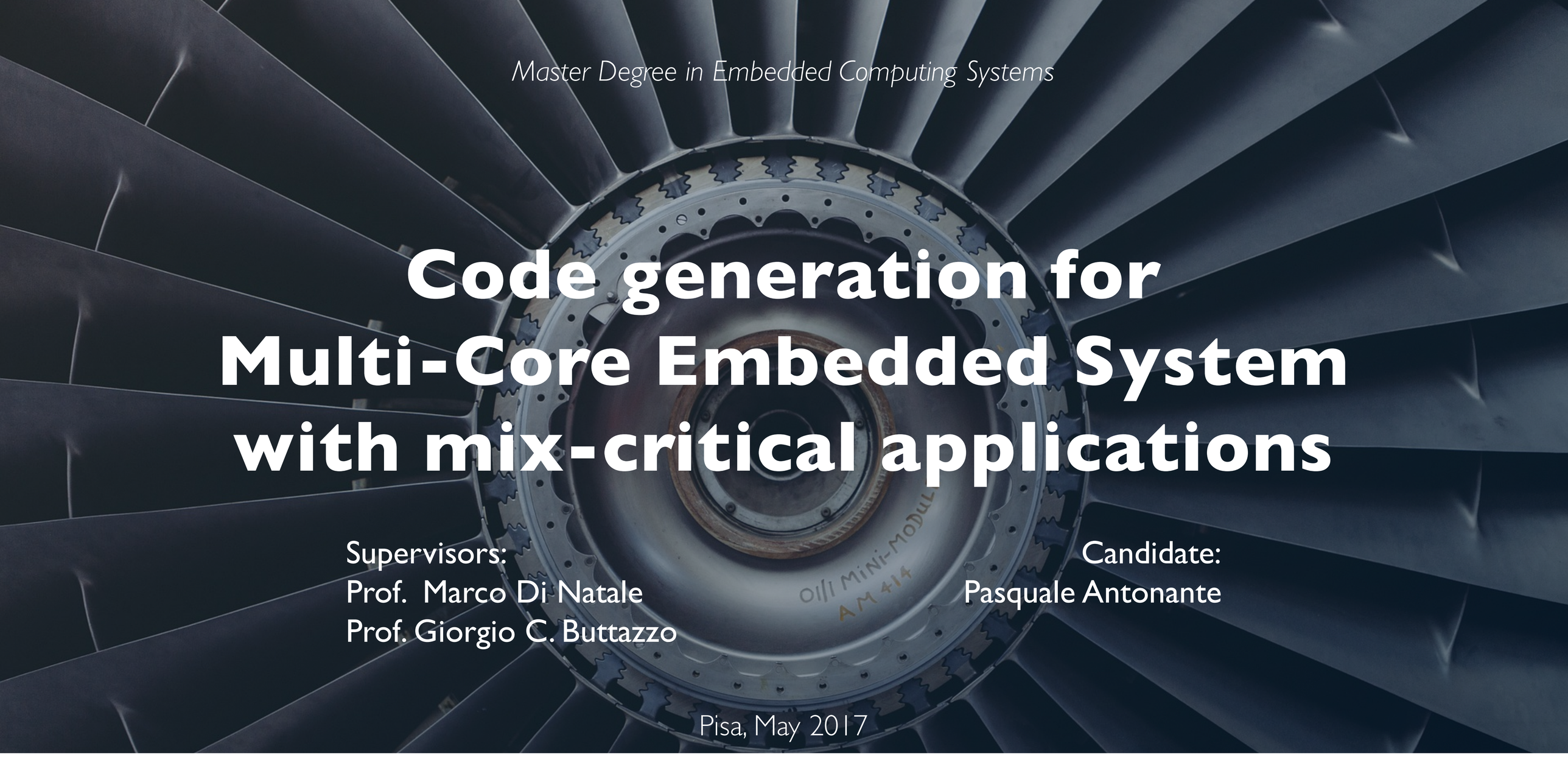


Master Degree in Embedded Computing Systems

Code generation for Multi-Core Embedded System with mix-critical applications

Supervisors:
Prof. Marco Di Natale
Prof. Giorgio C. Buttazzo

Candidate:
Pasquale Antonante

Pisa, May 2017

ISTITUTO DI TECNOLOGIE DELLA COMUNICAZIONE, DELL'INFORMAZIONE E DELLA PERCEZIONE
Scuola Superiore Sant'Anna

United Technologies Research Center

Objectives

Model-Based Design

Develop a Model-Based Design Framework

Mix-Criticality

Support the design and Implementation of mix-critical application

Multi-Core

Evaluate Multi-Core in Avionics use-cases

Virtualization

Evaluate the use of virtualization for mix-critical systems

Embedded Multi-core Systems for Mixed-Criticality Applications in Dynamic and Changeable Real-Time Environments – ARTEMIS EMC²

Partners
101 Partners of Embedded Industries

Countries
16 European Countries

Budget
~ 94 M€

Contributions

Framework for optimized **partitioning**, **allocation and scheduling** of tasks on **multi-core** systems.
Enable **Code Generate** from **Mix-Critical** models.

Model-Based Design
Enable model-based design development

Isolation
Ensure Isolation among different criticality levels

Optimization
Automatic partitioning, allocation and scheduling optimized for multi-core

Code Generation
Automatic Code Generation from the model

Motivation

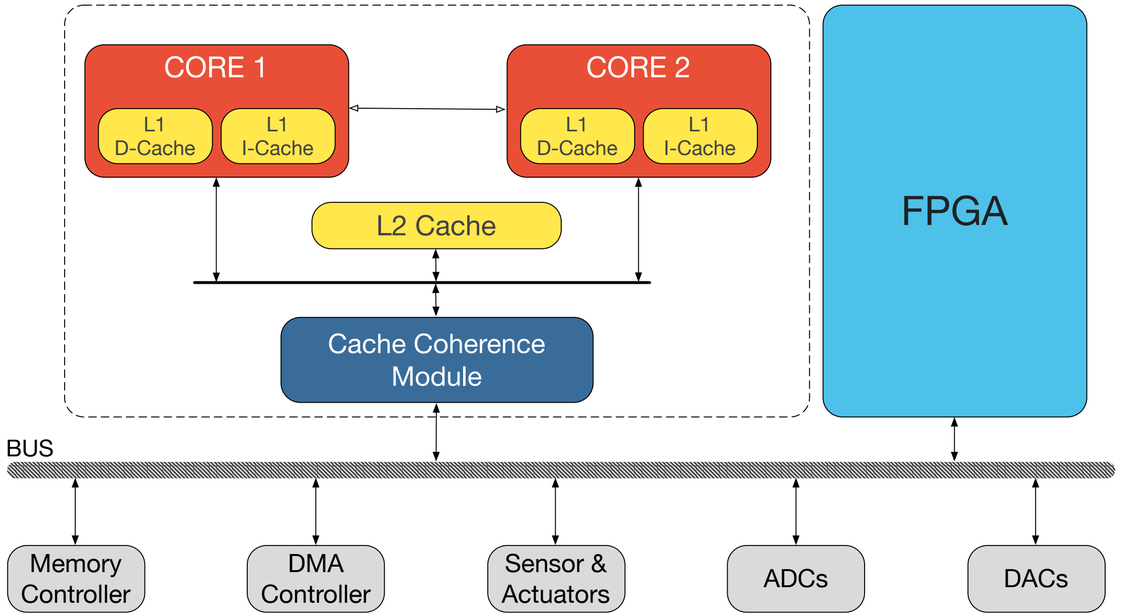
Multi-Core Embedded Systems

Increasing need of integrated functionality

Increasing need of processing power

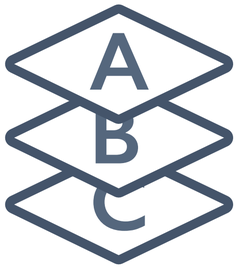
Multi-Core Commercial-off-the-Shelf

- **More processing power**
- **Reduced Size, Weight and Power (SWaP)**
- **Simplify the network topology**
- **Complexity of certification**



Unified Memory Model Architecture

Mix-Criticality



Certification authorities
define
Criticality Levels

In Avionics:

- **DO-178C:** considers the entire **software** life-cycle and provides a basis for avionic systems certification.
- **ARINC-653:** software specification for **safety-critical real-time operating systems**.

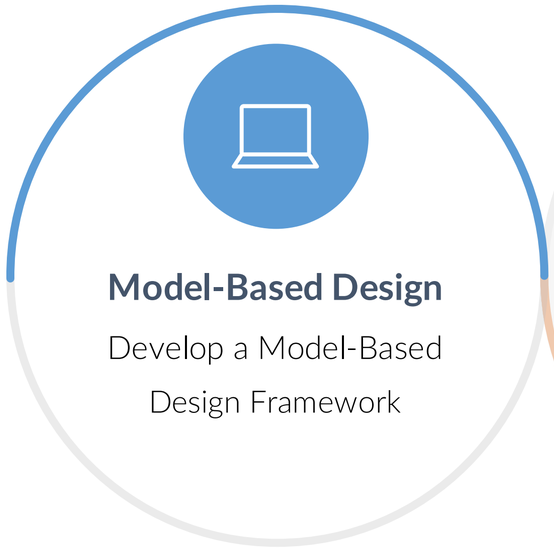
Robust Partitioning (ARINC-653)

"The objective of Robust Partitioning is to provide the same level of functional isolation as a federated implementation."

Level	Failure Condition	Failure Rate
A	Catastrophic	10^{-9}/h
B	Hazardous	10^{-7}/h
C	Major	10^{-5}/h
D	Minor	10^{-3}/h
E	No Effect	N/A

Table 1.1 Failure rate per DO-178C criticality level

Objectives

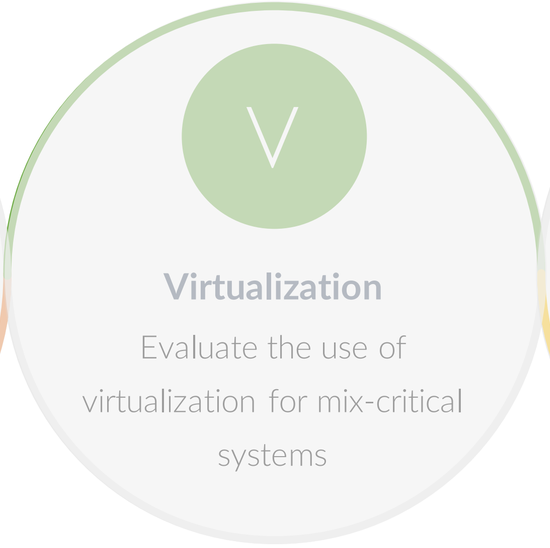


Model-Based Design

Develop a Model-Based Design Framework

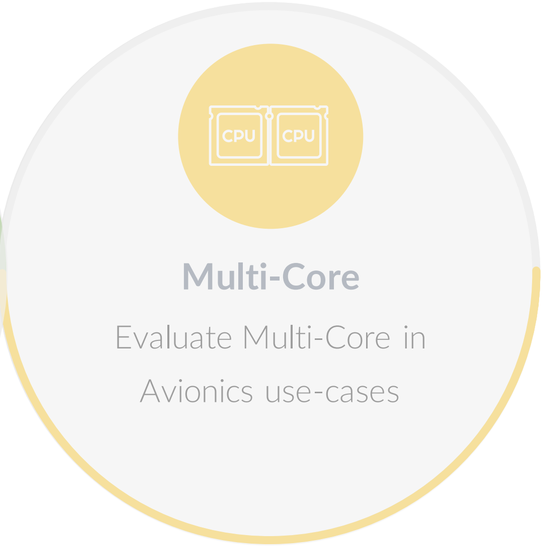
Mix-Criticality

Support the design and Implementation of mix-critical application



Virtualization

Evaluate the use of virtualization for mix-critical systems



Multi-Core

Evaluate Multi-Core in Avionics use-cases

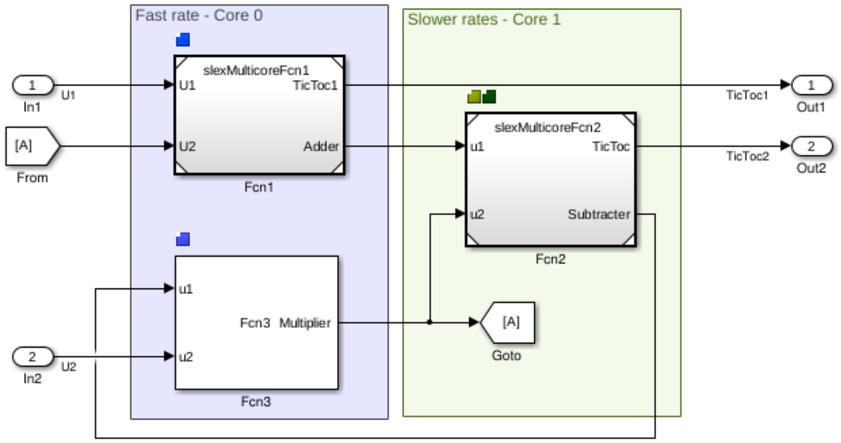
Model Based Design - Simulink

Model-Based Design Tool:

MATLAB® & SIMULINK®

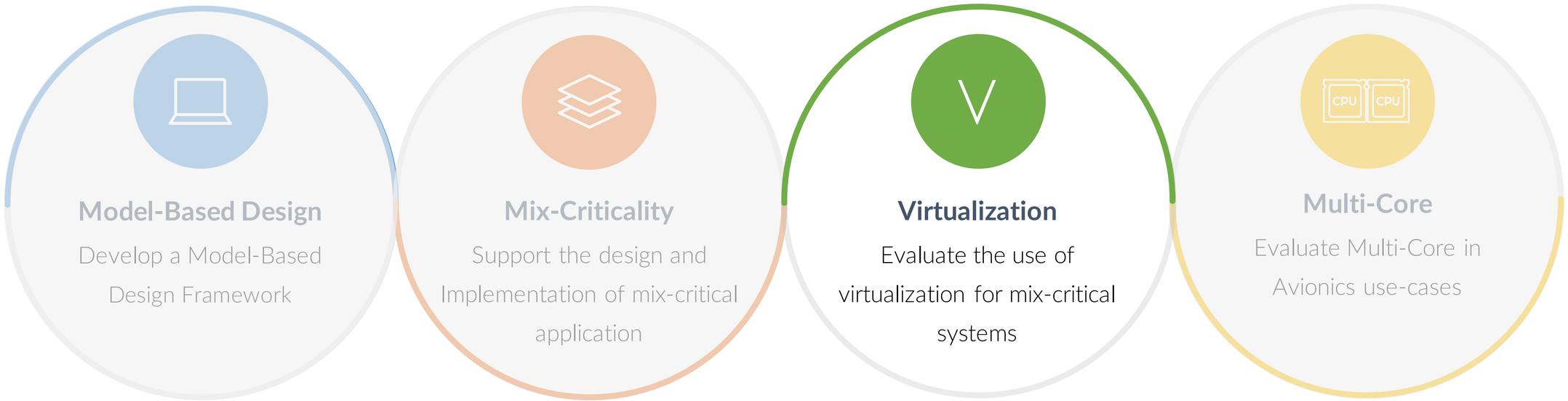
- ✓ Mature Modelling and Simulation tool
- ✓ Support Model Verification
- ✓ Mature C/C++ Code Generator
- × Does not support mixed-criticality

Concurrent Workflow:

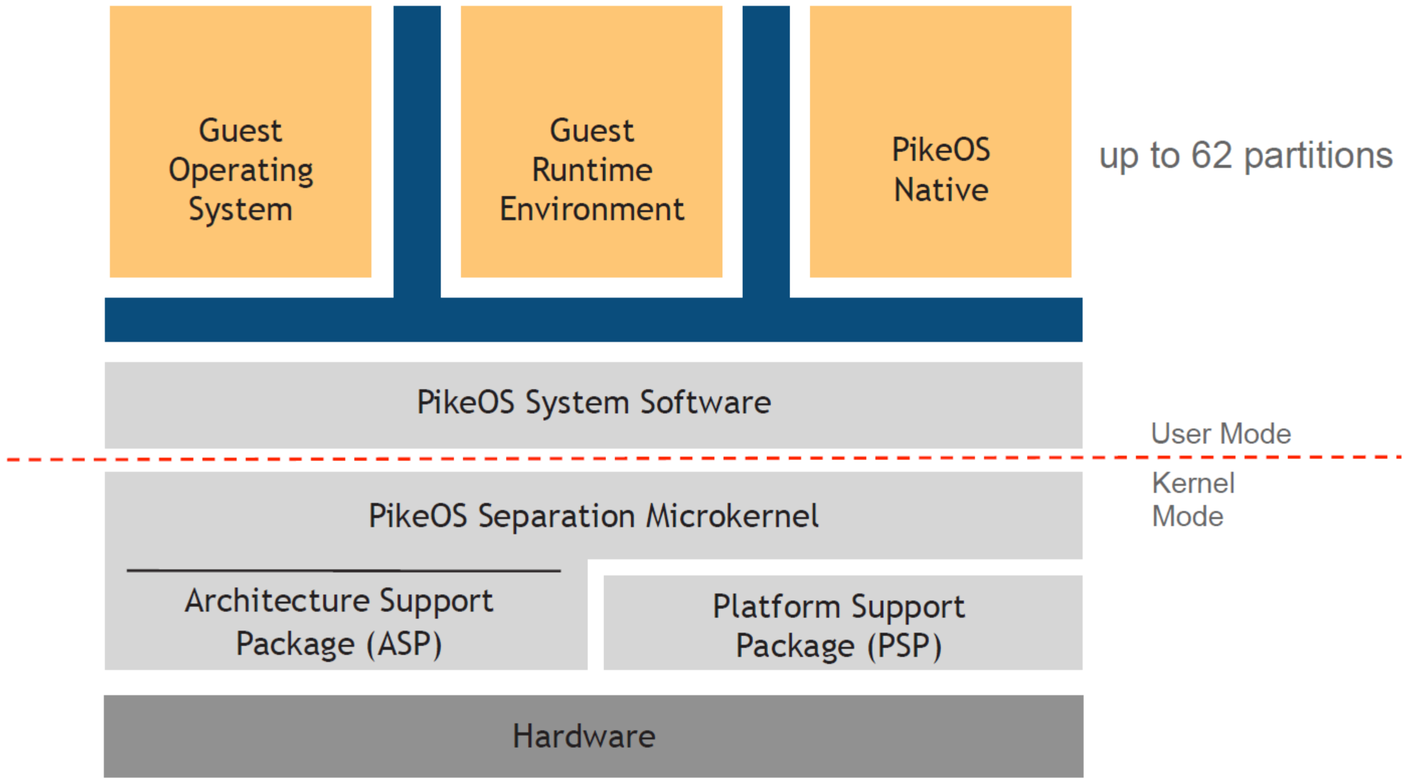


- ✓ Support Multi-core allocation
- × Implicit/Explicit Partitioning for tasks only
- × Complex and not yet mature

Objectives



Hipervisor - PikeOS



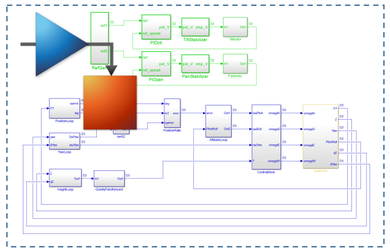
PikeOS System Architecture

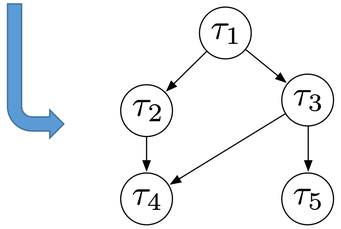
PikeOS at glance:

- Micro-Kernel based RTOS
- Hypervisor
- Separation Kernel
- Trusted, Certified
  - DO-178B/C (avionics)
  - ARINC-653 (avionics)
  - IEC 61508 (industrial)
  - EN 50128 (railways)
  - ISO 26262 (automotive)
  - IEC 62304 (medical)
- Multi-core support

Allocation and Scheduling Steps

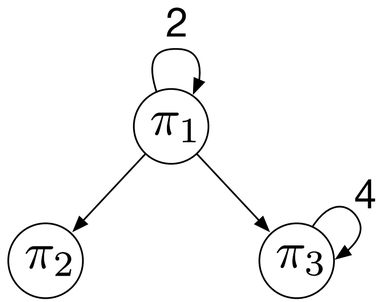
Functional Model





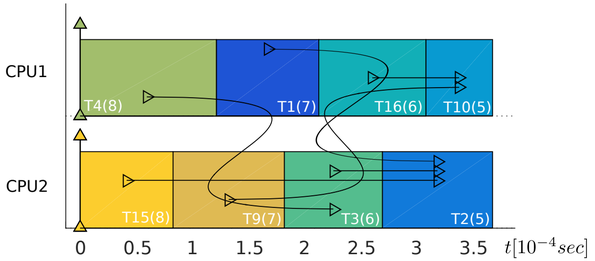
1. The **task-set** is extracted from the model as a **DAG** and **enriched with the designer information**

Partitioning



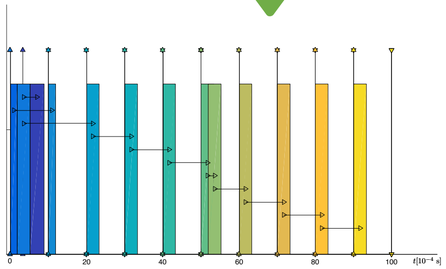
2. **Partition** are automatically generated using **Task periods**, **Resource Usage** and **Criticality**

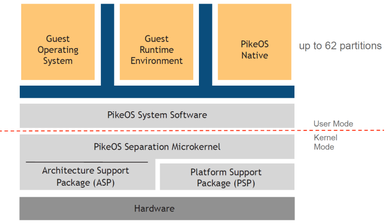
Intra-Partition Scheduling



3. A **MILP** opt. solution **allocates** and **schedules tasks** in every partition. Than **priorities** are **assigned**.

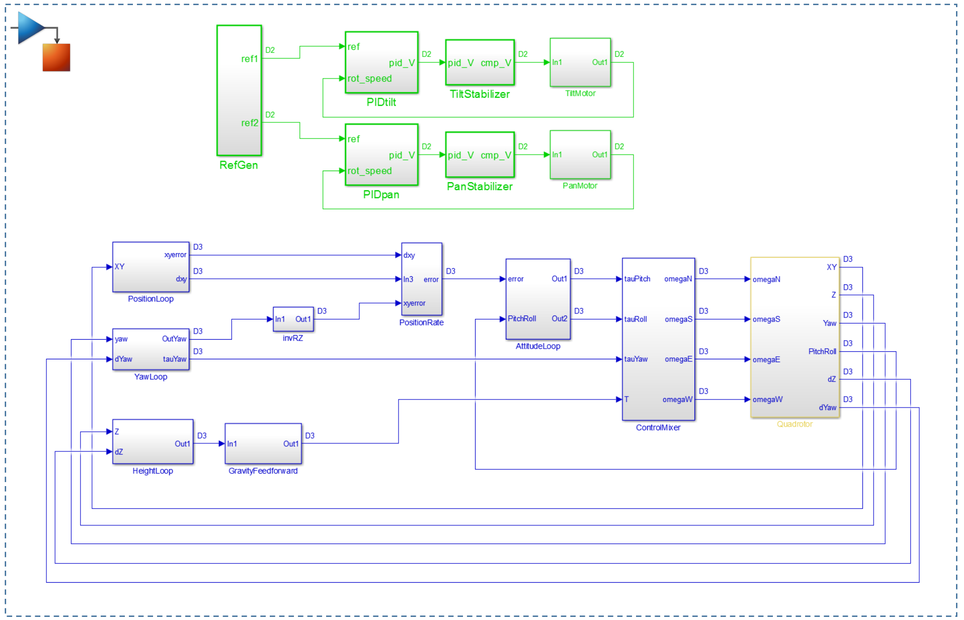
Inter-Partition Scheduling



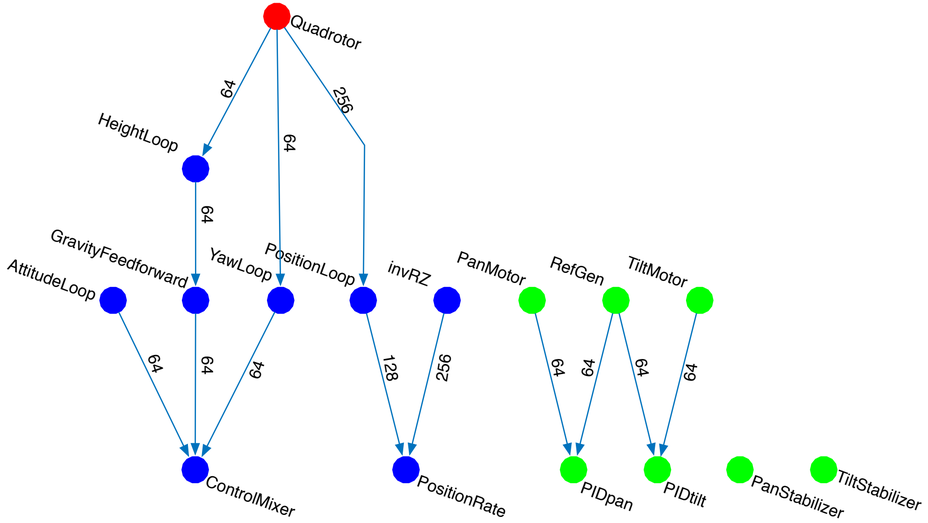


4. **Partitions** are then **scheduled** as if they were on a **Single-Core** (Bratley Algorithm)

Functional Model Extraction



The Simulink API via Matlab



Once the model simulation is satisfactory, **the designer defines tasks using Simulink Subsystems**, no spare blocks are allowed among subsystems.

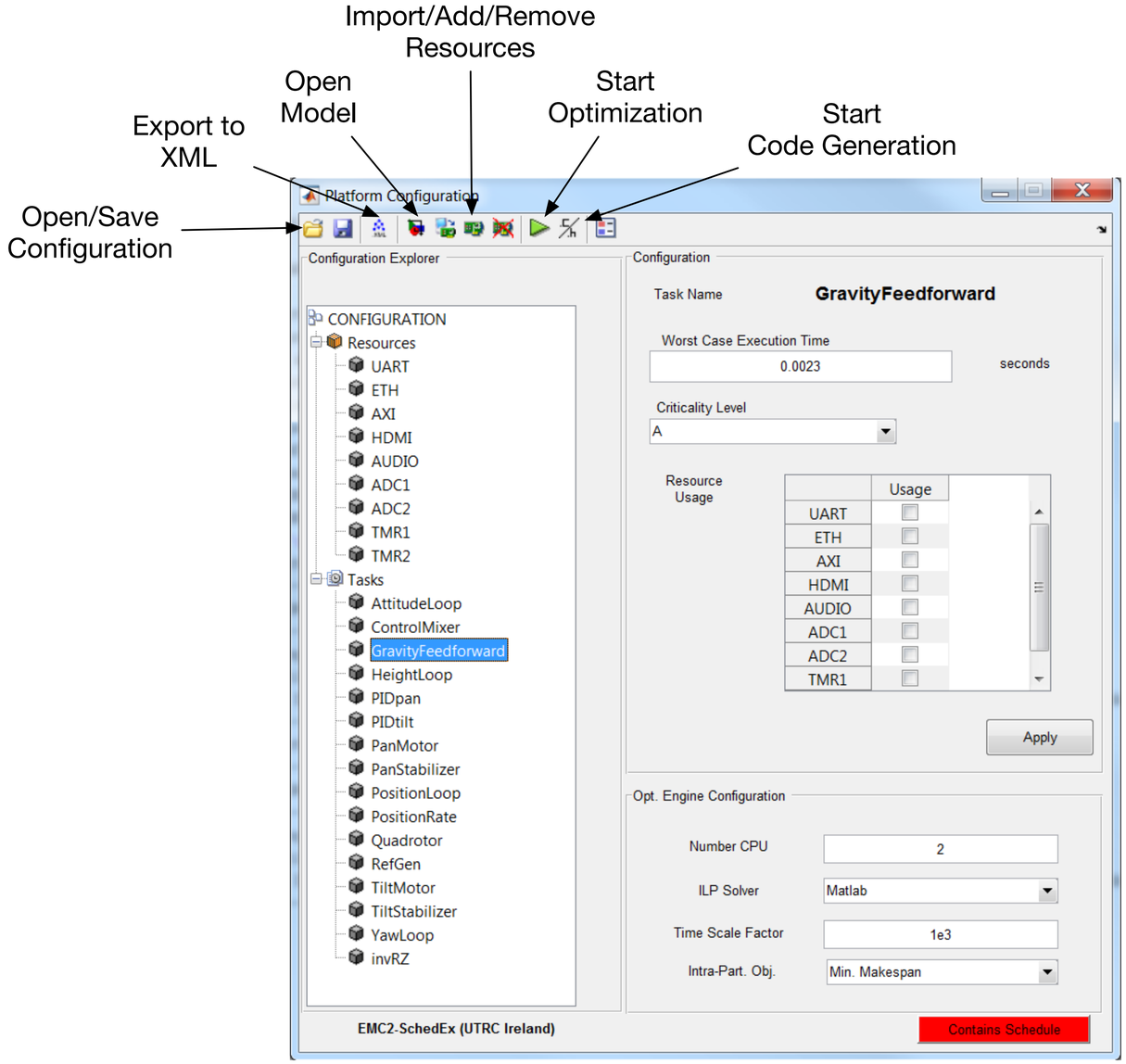
Using Simulink API, a **Direct Acyclic Graph (DAG) is extracted** a **functional representation** of the model

Process Configuration

The system designer specifies:

- WCET for each task
- Criticality level of each task
- Resources used by each task
- Number of available Cores

> Once ready, the **Scheduling Optimization** can be started



Scheduling and Allocation for Mix-Criticality

Traditional model:

$$\tau_i = \{T_i, D_i, k_i, \vec{C}_i, \rho_i\}$$

T_i	Period
D_i	Deadline
k_i	Criticality Level
$\vec{C}_i$	WCET for each criticality level
ρ_i	Priority

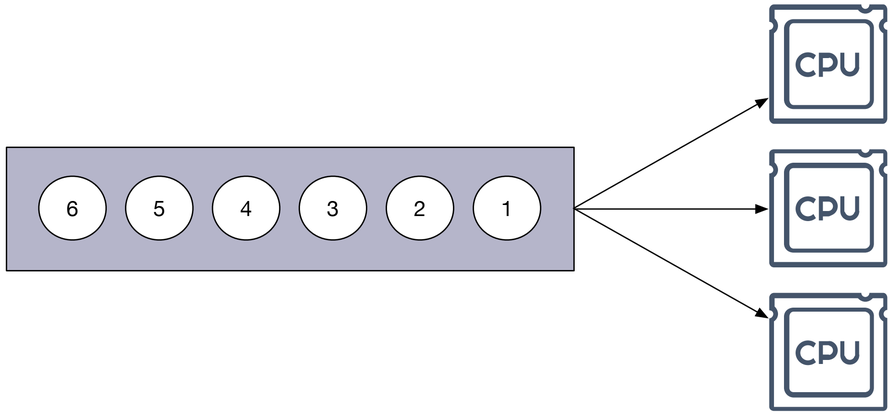
Typical scheduler policy:

For two levels of criticality:

1. The system remains in **low-critical** state until some of the high criticality task trespasses its C_{LO} value.
2. Then, the system goes into **high-criticality** mode
3. Only high criticality tasks are **guaranteed** to execute
4. Upon selected conditions, the system goes back in **low-criticality** mode.

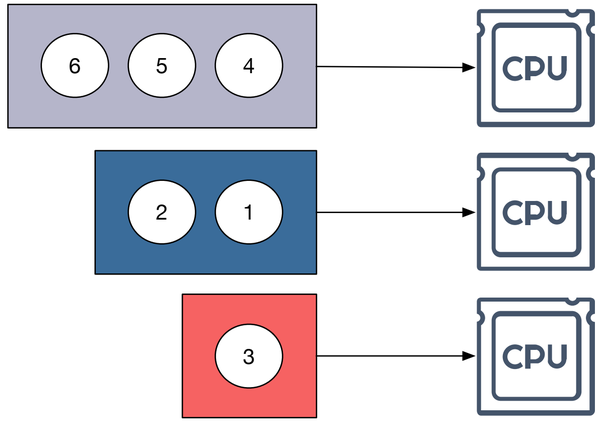
Global vs Partitioned Scheduling

Global:



- ✓ Better CPU usage
- ✓ Lower Average Response time
- ✓ Easy to implement
- × Poor separation

Partitioned:



- ✓ Better isolation
- ✓ More determinism
- × Cannot exploit unused capacity
- × NP-Hard

Problem Formulation

Tasks:

$$\tau_i = \{t_i, c_i, k_i, \mathbb{R}, \pi_i, s_i, \mu_i, \rho_i\}$$

t_i	Period
c_i	WCET
k_i	Criticality Level
$\mathbb{R}$	Resources
π_i	Partition
s_i	Starting Time
μ_i	Affinity Mask
ρ_i	Priority

Task-set DAG:

$$G = (\Gamma, E, C, T, K)$$

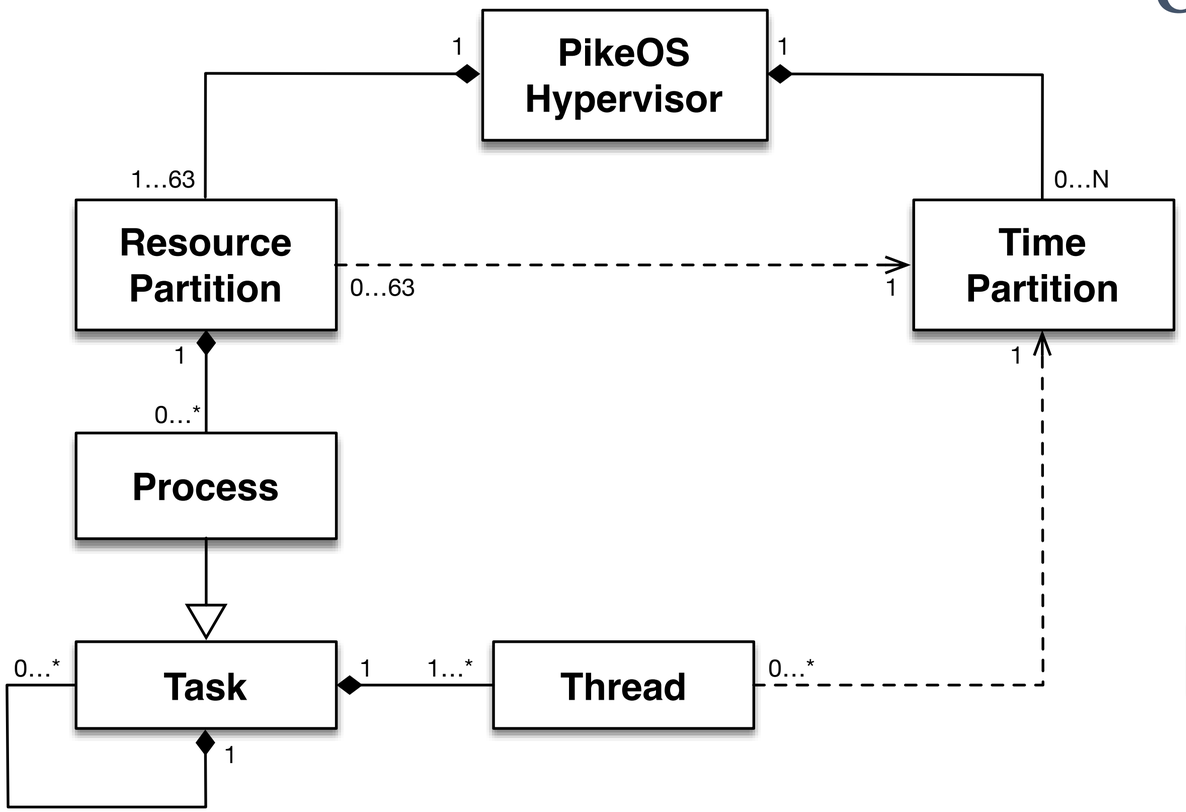
Γ	Task-Set
E	Edges
C	WCET
T	Periods
K	Criticalities

Partitions DAG:

$$\mathbb{P} = (\Pi, H, L, R)$$

Π	Partitions
H	Edges
L	Duration
R	Periodicity

Execution Entities

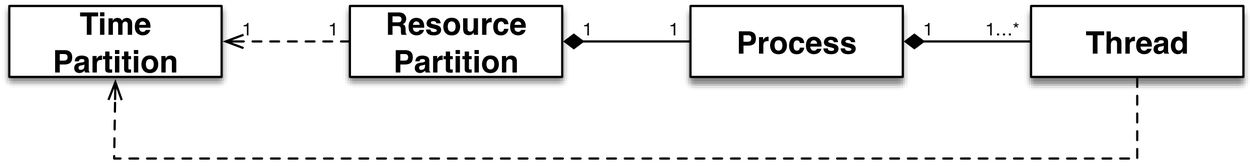


PikeOS Execution Entities

Once a **feasible schedule** is obtained, the **Code Generation** can be started from the GUI

With respect to the PikeOS terminology:

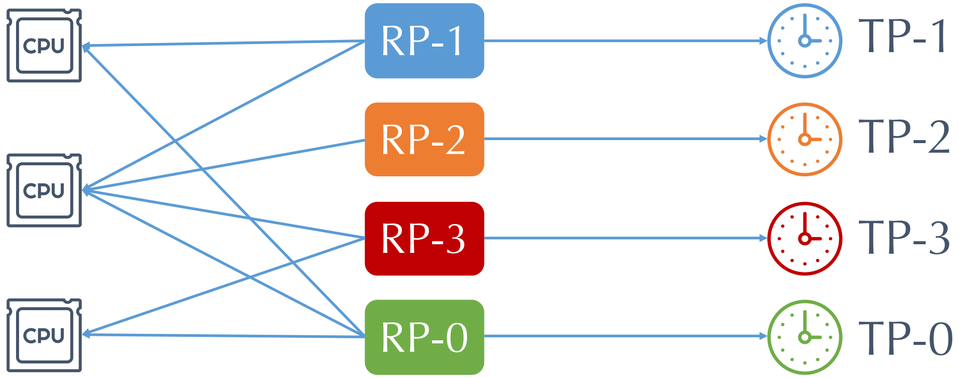
- Each **Subsystem** is a **Thread**
- Each **Partition** is a **Process** inside a **Resource Partition**
- Each **Time Partition** contains a single **Resource Partition**

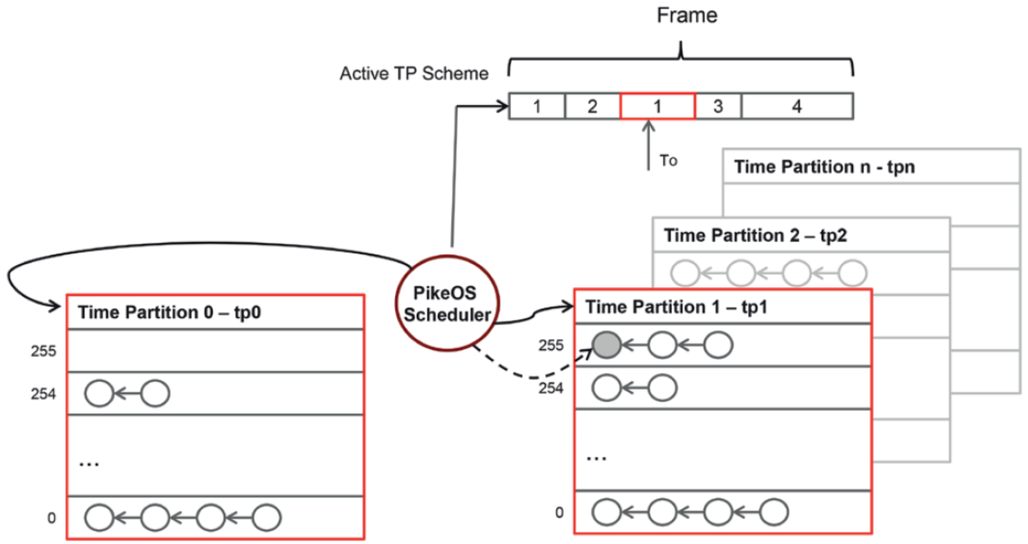


PikeOS Execution Entities

Code Generation

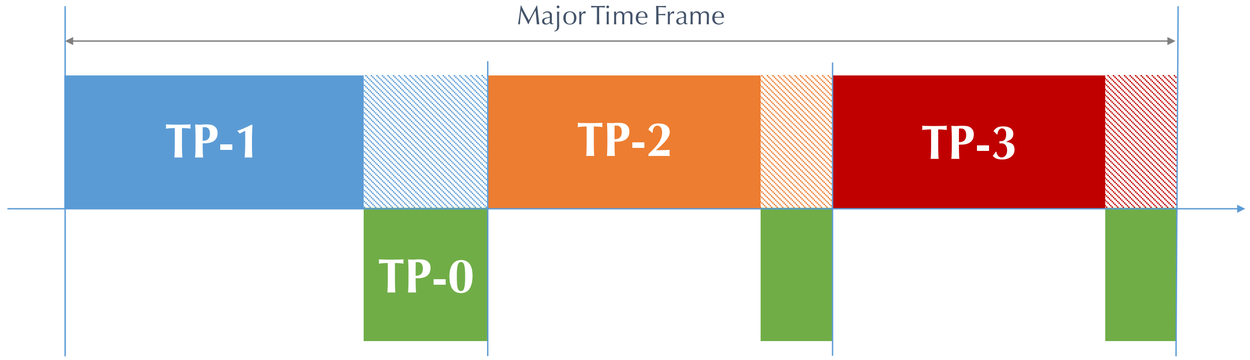
PikeOS Real-Time Scheduling





Active Partition

Background Partition



Code Generation Process

Comm. Adaptation	Subsystem Gen.	Native Processes Gen.	OS Conf. Snippets
1. Intra-Partition comm. are adapted to the SW configuration. **In/Out port that are at the edge of two different partitions** must be **replaced** by the operations on a **Sampling port**.	2. **Each subsystem is generated** by Simulink. A **description (XML) file** is also **generated** for each subsystem, it describes the code interface.	3. Each XML description file is parsed and a **Native PikeOS Process** is **generated** for each Partition.	4. PikeOS requires a static configuration. Several **XML snippets** are generated **for the Integration Project**.
Achieved by a **Custom Blockset.**	The process is driven by a **Custom System Target File**.	Achieved by a **custom TLC file.**	Achieved by a **custom TLC file.**

Conclusions

Contribution:
Model-Based Design **framework**, supported by **Simulink**, for **mix-criticality application** in **multi-core** embedded system

Future Works - Scheduling

- Improvements on the partitioning algorithm
- Add automatic Worst-Case Execution Time Analysis
- Formal verification to prove that hazardous states are never reached

Future Works – Code Generation

- Add support to non-periodic tasks
- Support Hardware-in-the-loop

References

[1] Vestal, Steve. "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance." *Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International*. IEEE, 2007.

[2] Ernst, Rolf, and Marco Di Natale. "Mixed Criticality Systems—A History of Misconceptions?." *IEEE Design & Test* 33.5 (2016): 65-74.

[3] Global scheduling in multiprocessor real-time systems. Alessandra Melani, Retis.

[4] Artemis EMC2 European Project. http://www.artemis-emc2.eu/.

[5] Pikeos safe real-time scheduling. adaptive time-partitioning scheduler for en 50128 certi ed multi-core platforms. SYSGO Whitepaper.

[7] Marc Fumey Xavier Jean, Marc Gatti Guy Berthon. The use of multicore processors in airborne systems (mulcors), 2012. https://www.easa.europa.eu/ document-library/research-projects/easa20116.