THALES

ECRTS 2016
5-8 July
Toulouse, France

Overview of Potential Software solutions making multi-core processors predictable for Avionics real-time applications

Marc Gatti, Thales Avionics
Sylvain Girbal, Xavier Jean, Daniel Gracia Pérez,
Jimmy le Rhun, Thales Research&Technology

28th Euromicro Conference on Real-Time Systems (ECRTS16)

SUMMARY

Context

Multicore Introduction

Problem Statement

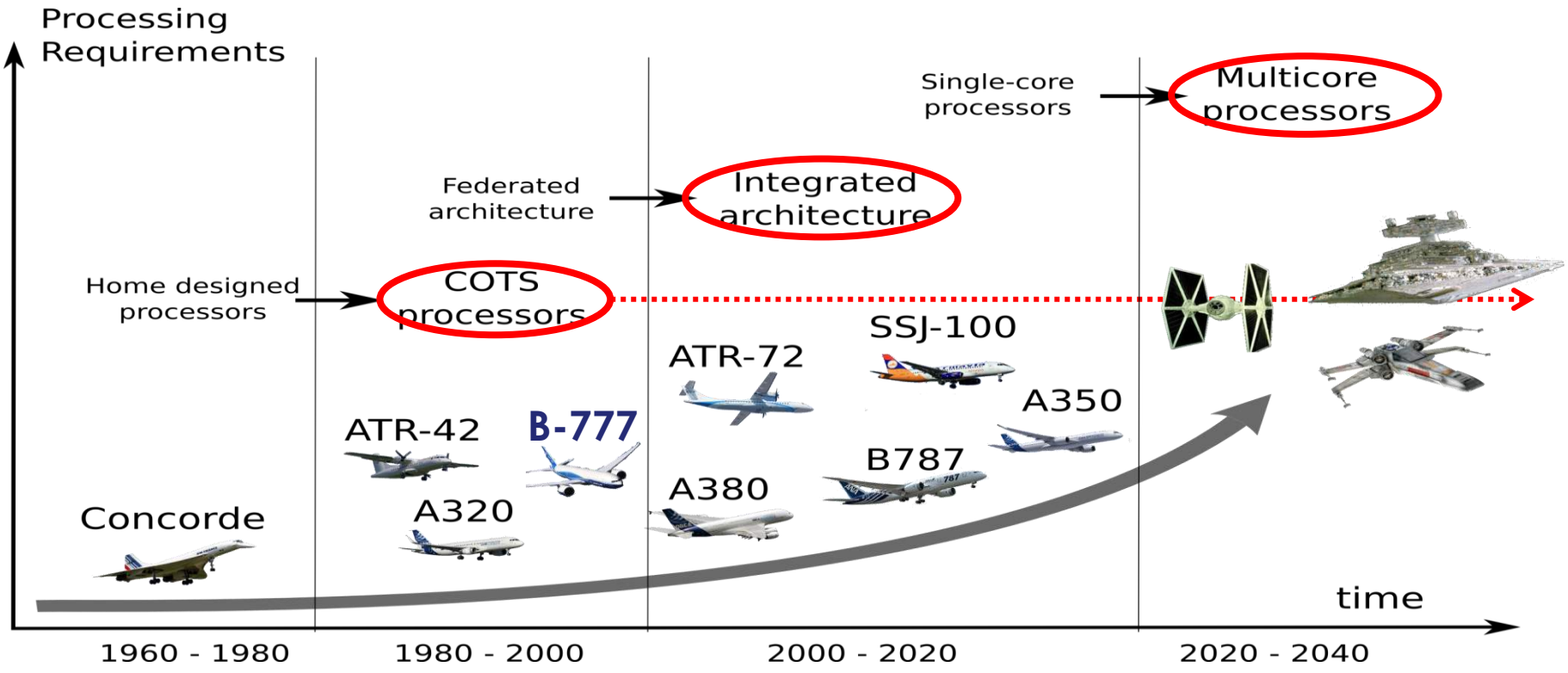
Current Studies for IMA

Overview of Potential SW sol.

Conclusion / future works

SUMMARY

DIGITAL AVIONIC SYSTEMS EVOLUTION



AVIONICS STAKES & PARADIGM

Improving the SWaP (Size, Weight and Power) of the IMA embedded platform

- Reduce the Size, the Weight and the environmental Footprint (Power Consumption)

While Increasing

- Availability, Safety, Reliability
- Security
- And the performances per a significant factor compare to the current generation

While continuing to Master Certification Issue

AVIONICS & SAFETY CRITICAL SYSTEMS CERTIFICATION

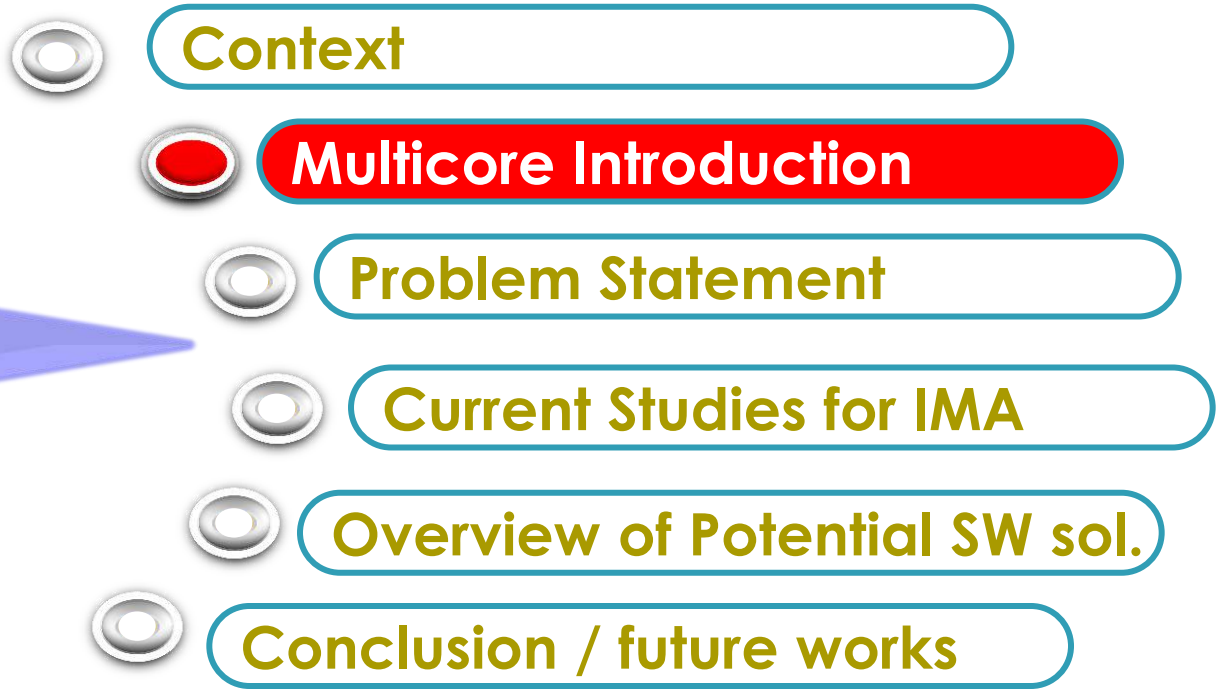
- Aircraft functions are hosted on Aircraft Embedded Systems.
- Aircraft Embedded Systems have to be certified following certification requirements of Federal Aviation Administration (FAA), European Aviation Safety Agency (EASA) and other agencies as required.
- Increasing Integration and Complexity of the Aircraft Embedded Systems requires procedures and guidance to ensure the proper operation and safety.
- Guidelines have to be followed for development of systems that implement aircraft level functions through out the lifecycle of the systems.
- Multicore platforms despite of advantages introduce significant certification challenges.

AVIONICS AND SAFETY CRITICAL SYSTEMS CERTIFICATION

Applicable guidelines for complex avionics systems development

- ARP4754 : Certifications Considerations for Highly- Integrated or Complex Aircraft Systems
- DO 254: Design assurance guidelines for airborne electronic hardware
- DO 178B/C : Software considerations in airborne systems and equipment certification
- DO 297 : Integrated Modular Avionics (IMA) Development, Guidance and Certification Considerations
- ARINC 653 : Avionics Application Software Standard Interface Part 1
- ARINC 651 : Design guidance for Integrated Modular Avionics
- EASA : EASA CM - SWCEH - 001 Issue No.: 01
- EASA : Generic CRI (Certification Review Item)

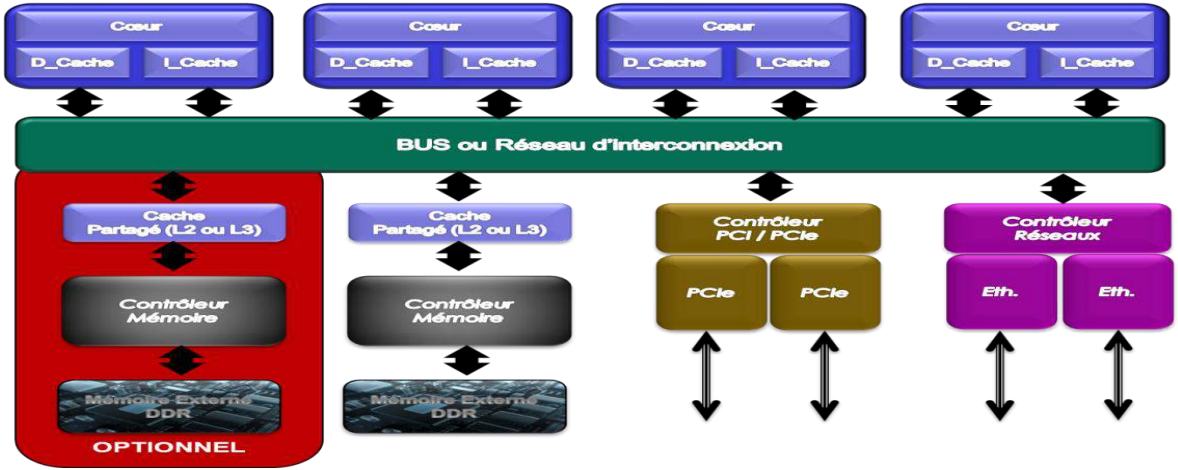
SUMMARY



MULTI-CORE: INTRODUCTION

What's a multicore processor?

- Multicore processor is characterized by N (N ≥ 2) processing cores + a set of shared interconnected resources (Memories, PCIe, Ethernet, Cache, Registers, etc.)



Two main types of processors

- First one where interconnect between cores is based on an arbitrated bus
- Second one where interconnect between cores is based on a network like

Multicore management in certified embedded platform can be summarized to Interferences management

THALES

WHY DO WE NEED MULTI CORE ?

Power Wall

- Thanks to Moore law : Higher performance ➔ operating frequency was increased.
- Dynamic power consumption is directly proportional to (capacitance × voltage2 × switching frequency).
- Above GHz, half of the power consumption is link to maintain static state and generates heat that requires advanced cooling decreases the reliability and shortens the longevity
- Multi-core approach: Reduce power consumption of the CPU while maintaining or increasing performance ➔ replace frequency increasing by core number increasing

Frequency Wall

- Increasing frequency also leads to power wall.

Memory Wall

- Increase in on-core speed is not matched by the speed of off-core / off-chip memory and IO subsystems.
- Memory density is not growing on par with processing
- Adding Fast caches increases both silicon size and power consumption

SUMMARY

- Context
- Multicore Introduction
- **Problem Statement**
- Current Studies for IMA
- Overview of Potential SW sol.
- Conclusion / future works

INTEGRATED MODULAR AVIONICS CONCEPT

Set of Hardware and Software components

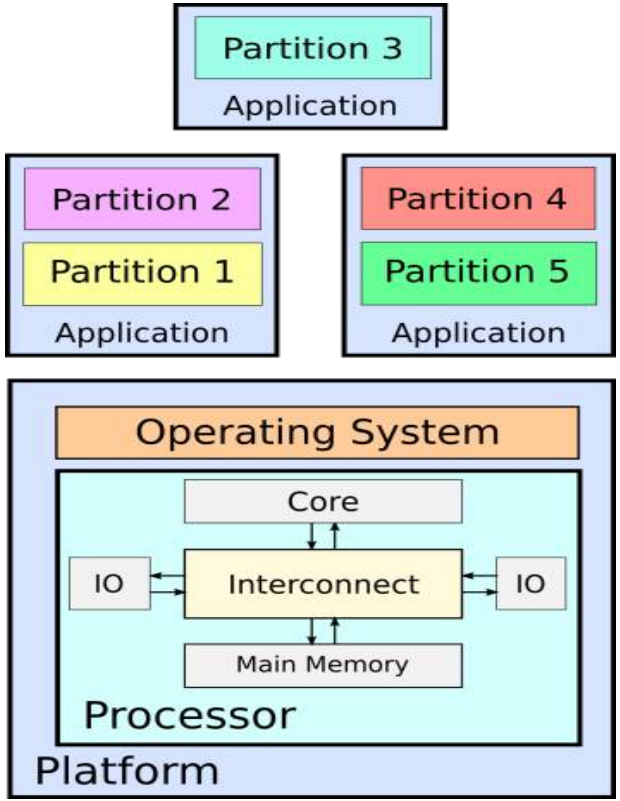
- Modular development
- Incremental certification DO-297

Dependability constraints

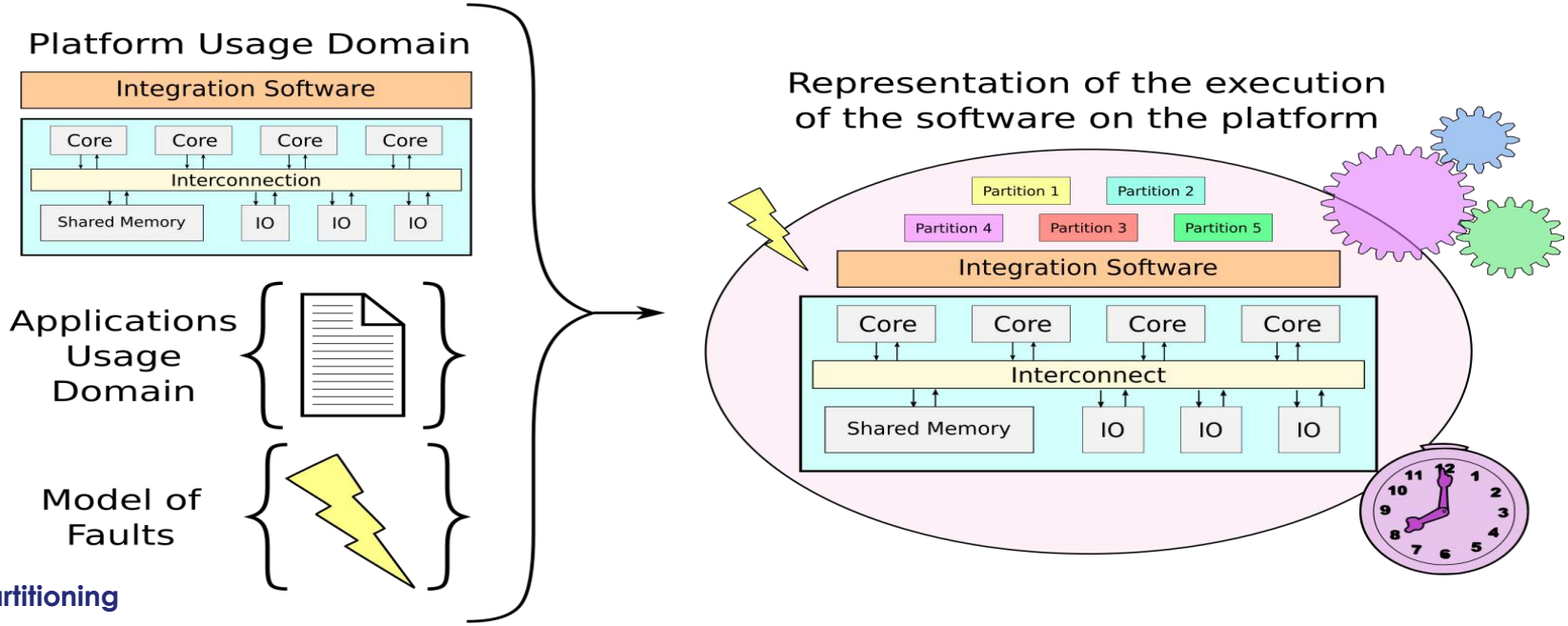
- Worst Case Execution Time computability
  - Safety of WCET computation
- Failure Isolation : Robust Partitioning
  - Modularity of WCET computation

Platform efficiency

- Raw Performances for each Partition
- Number of Hosted Partitions



INTEGRATED MODULAR AVIONICS: MANDATORY REQUIREMENTS



Robust partitioning

Platform determinism

Platform limitations for WCET scenario definition

Why ensuring robust partitioning is so difficult on multicore platforms ?

THALES

MULTICORE PROCESSORS INTEGRATION IN IMA SYSTEMS

Conflicts Management

Spatial Management:

- How to manage accesses to be sure that one core can't access to a space reserved for another core.

Temporal Management:

- How to manage accesses done by one core to all shared resources (Memories, I/O, etc.) to be sure that accesses can be limited in time whatever activities of other core are (normal or abnormal).
- Upper bound will be used for WCET computation

Memory Accesses Management

- Spatial Management is done by MMU and IOMMU (when existing)
- Temporal Management is more complex linked to interconnect (transaction management), Memory Controller and Memory (transaction realization).

Operating System

Architecture Choice regarding Industry needs

- Computer Number Reduction with low impact on legacy application ➔ AMP
- Application Performance Improvement ➔ SMP

MULTICORE PROCESSORS INTEGRATION IN IMA SYSTEMS

ARINC 653 Partitions Deployment

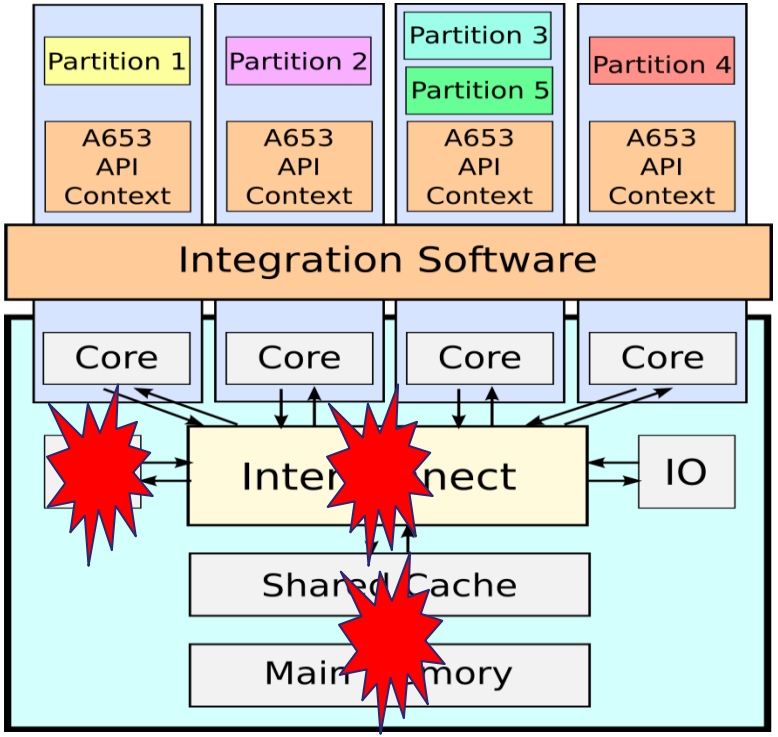
- Asymmetrical Multi-Processing
- Backward compatibility on legacy
- No global constraint on schedule

Hardware Resources Allocation

- Private versus Shared resources
- Interleaving of concurrent transactions in the interconnect

Inter-Core Conflicts Occurrences

- Resources sharing policy driven by the hardware



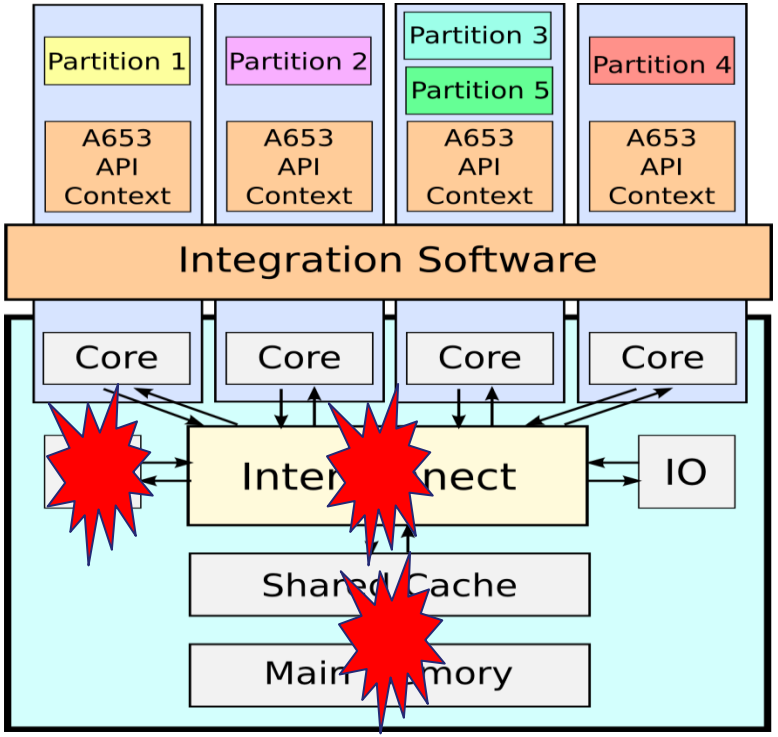
PROBLEM STATEMENT

How to compute a WCET ?

- Simulate the core's worst case behavior executing the application
- Consider any access to a shared resource as taking its Worst Case Access Time (WCAT)

Problem : How to determine WCAT to shared resources ?

- No constraint on embedded partitions
- No guarantees on a minimal bandwidth granted to each core
- In practice we observe pathological situations



Hardware management of shared resources not safe enough

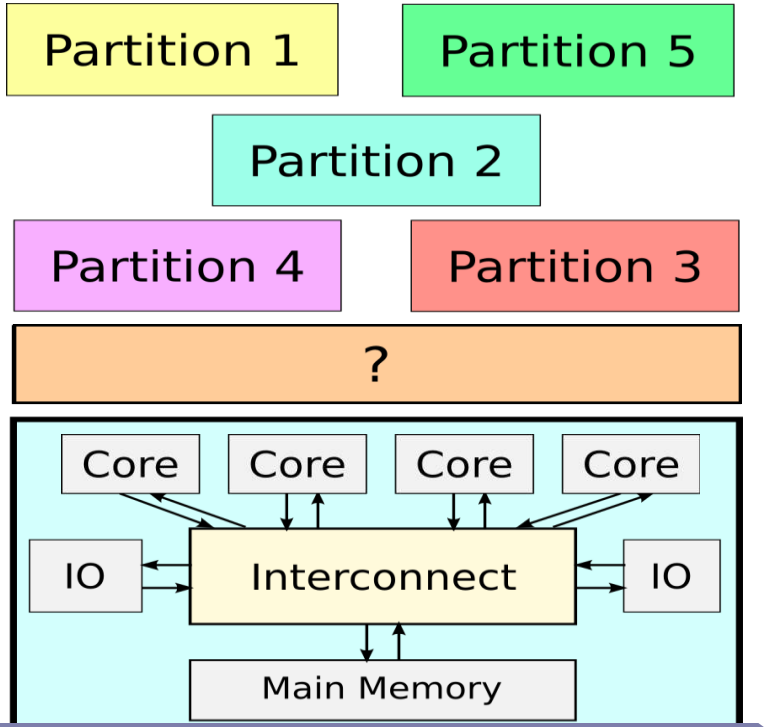
SUMMARY

- Context
- Multicore Introduction
- Problem Statement
- **Current Studies for IMA**
- Overview of Potential SW sol.
- Conclusion / future works

MULTICORE FOR IMA, "GOOD PROPERTIES"

How could Avionics Platforms take benefit of multicore processors ?

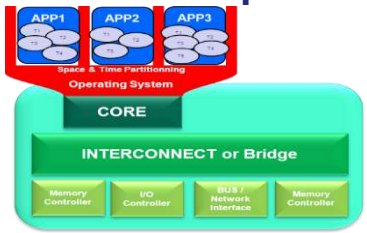
- Allow all cores to be used whatever the level of criticality
- Minimize porting and re-certification efforts of legacy applications
- Compatibility with ARINC 653 and ARINC 664 guidelines for APEX and Network partitioning
- Incremental certification

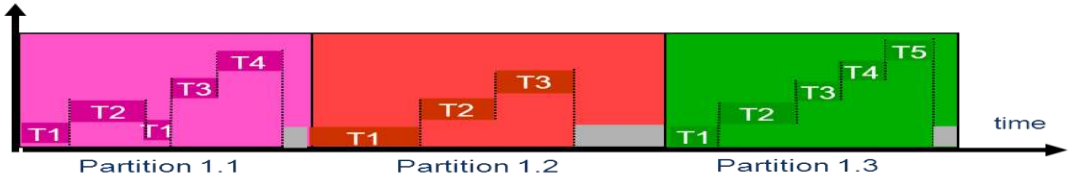


Digital avionic systems confidence have never regressed during technological steps

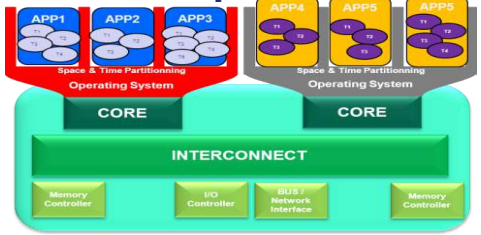
TIME PARTITIONING

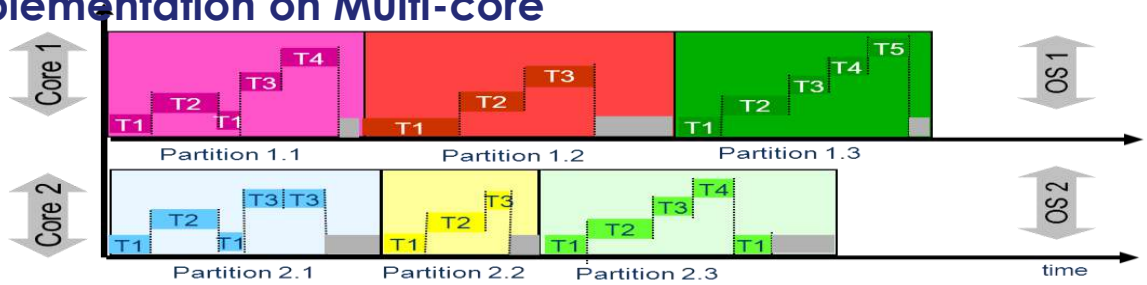
Current Implementation on Mono-core



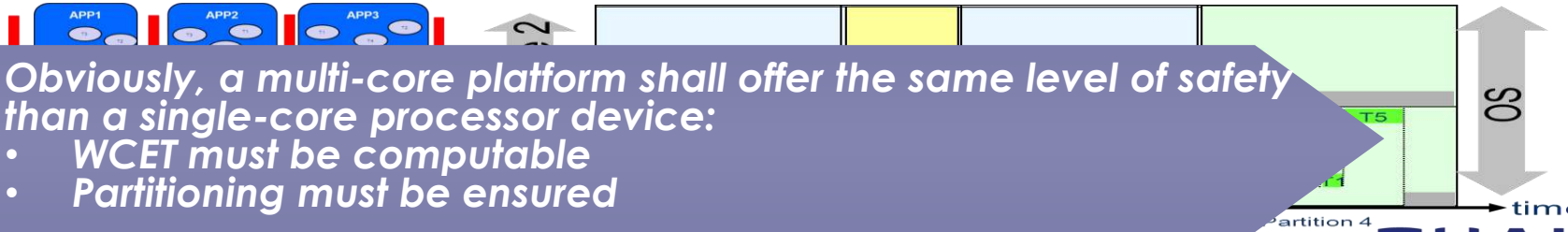


An example of AMP implementation on Multi-core





An example of SMP implementation on Multi-core



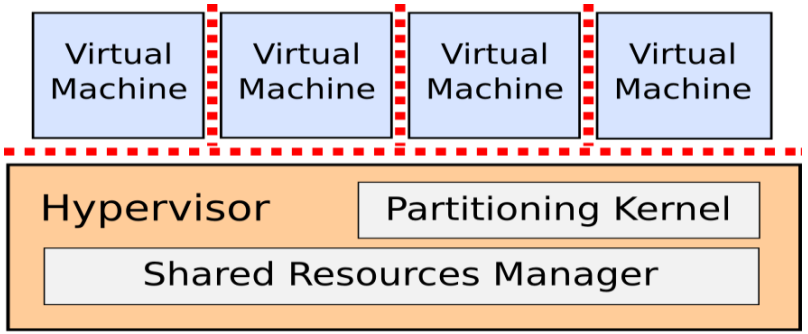
Obviously, a multi-core platform shall offer the same level of safety than a single-core processor device:

- ***WCET must be computable***
- ***Partitioning must be ensured***

APPROACH FOR SHARED RESOURCES MANAGEMENT

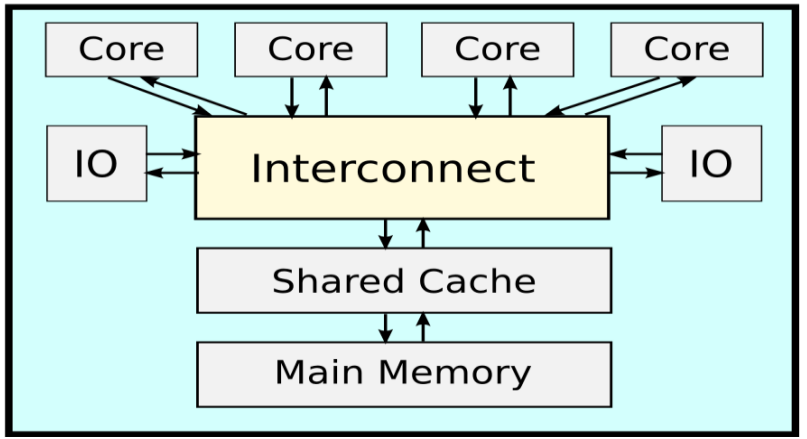
Software approach implemented in a hypervisor

- One virtual machine per core hosting an Operating System
- Shared resources management policy hidden to guest software



Resource sharing policy setup and configuration

- Interconnect bandwidth quota allocated to each core
- On the fly control of accesses to cope with the allocated quota
- Detection of pathological situations



What is the impact on application's performances ?

THALES

SUMMARY

- Context
- Multicore Introduction
- Problem Statement
- Current Studies for IMA
- **Overview of Potential SW sol.**
- Conclusion / future works

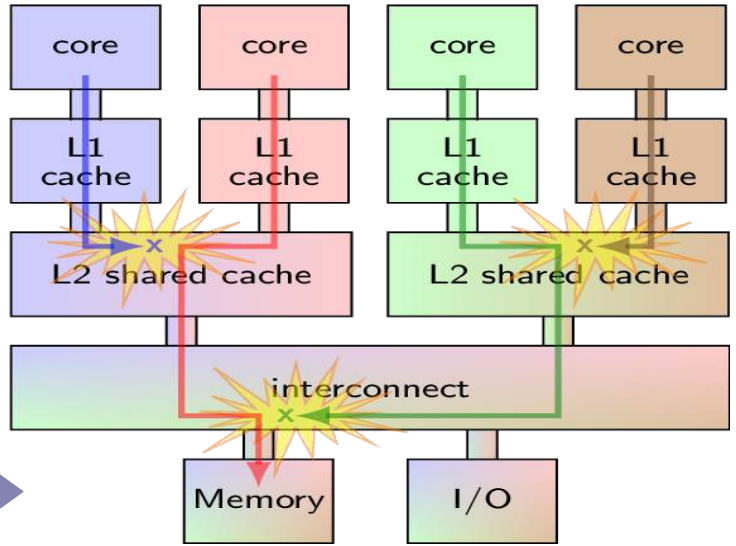
CONTEXT & THE ISSUE

Using Multi-Core COTS in avionic equipment

- Facing an exponential increase of performance requirements in avionics
- Multi-Core COTS ➔ best computing performance compromise for a reasonable size, weight, power & cost.
- Lack of predictability ➔ significant impact on WCET ➔ over-provisioning

The problem: inter-core interferences

- Multi-core are characterized by shared hardware resources
- Concurrent accesses to these resources are involving arbitration mechanism at hardware level
- Hardware contention is introducing unpredictable interferences appearing as extra time delays



How to solve the predictability issue and ensure determinism with Multi-Core COTS?

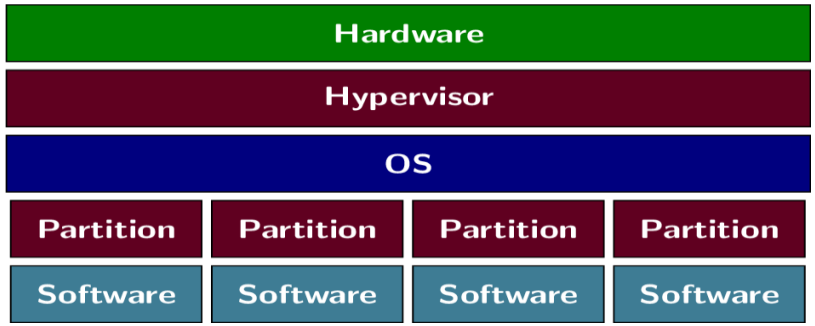
DETERMINISTIC PLATFORM SOFTWARE FOR MULTI-CORE COTS

To have a deterministic usage of an unpredictable hardware to

- Ensure spatial isolation and timing isolation properties
- Either eliminate, control or react to interferences to mitigate their impact on determinism

Deterministic Platform Software

- Between the workload application and the hardware
- Provides determinism by either
  - Controlling the access to the hardware
  - Reactively regulating the resource usage



What are the criterions to evaluate a Deterministic Platform Software in an industrial context?

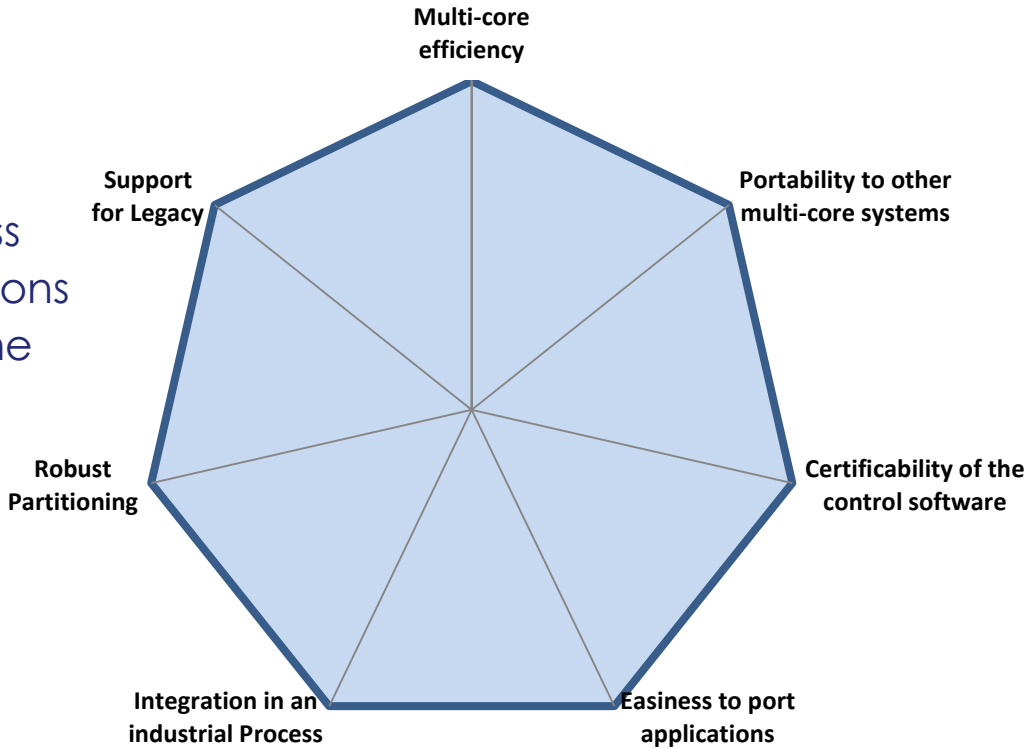
STAKES

Key Properties

- Support for **Legacy** Applications
- Performance Efficiency
- Robust **Partitioning**
- **Integration** into an Industrial Process
- **Easiness to adapt** Existing Applications
- **Complexity** and **Certificability** of the platform software

Assessment Level

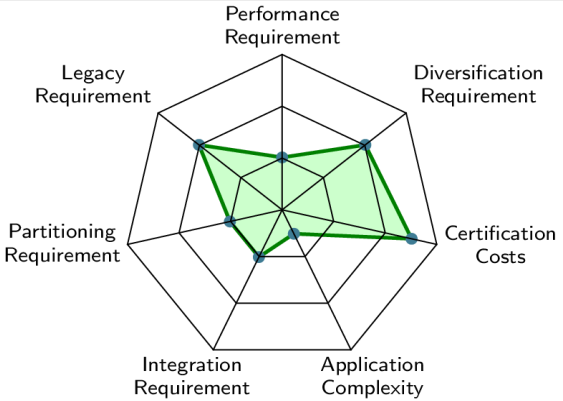
- 0: Not fulfilled
- 1: Fulfilled for a significant cost
- 2: Fulfilled for a small cost
- 3: Fulfilled at no cost



AVIONIC CASE STUDIES (1/2)

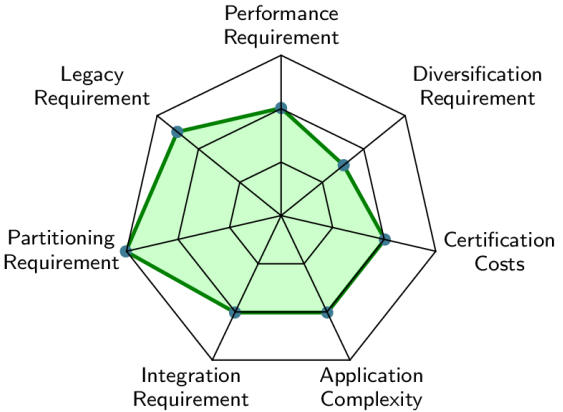
Full Authority Digital Engine Control (FADEC)

- DAL-A
- Controls aircraft engine performance
- Provides optimum efficiency for given flight conditions
- Control-command
- No manual override: FADEC failure ➔ engine failure



Integrated Modular Avionics (IMA)

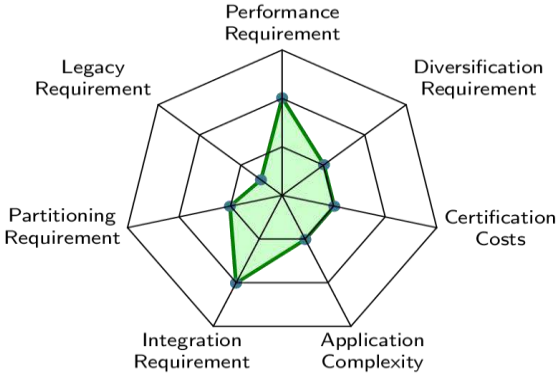
- DAL-A ➔ DAL-D
- Runs several mission computing software on the same hardware to reduce weight, space, energy and costs
- **Modularity** enables concurrent conception and certification & reduce maintenance and upgrade costs
- Runs a **wide variety** of different software
- Strongly relies on **robust partitioning**



AVIONIC CASE STUDIES (2/2)

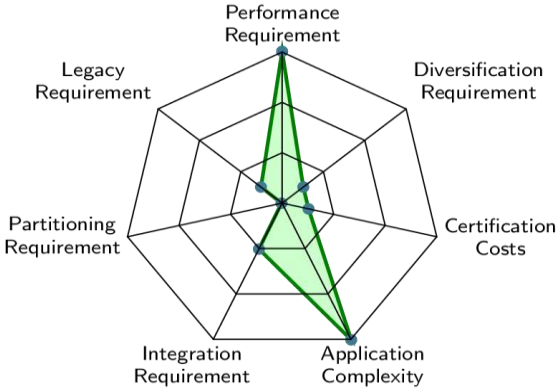
Data Server

- DAL-C ➔ DAL-D
- Manages the communication with the satellites, the crew communication and the maintenance interface
- Performance characterized in term of **I/O throughput** rather than computation power



In-Flight Entertainment (IFE)

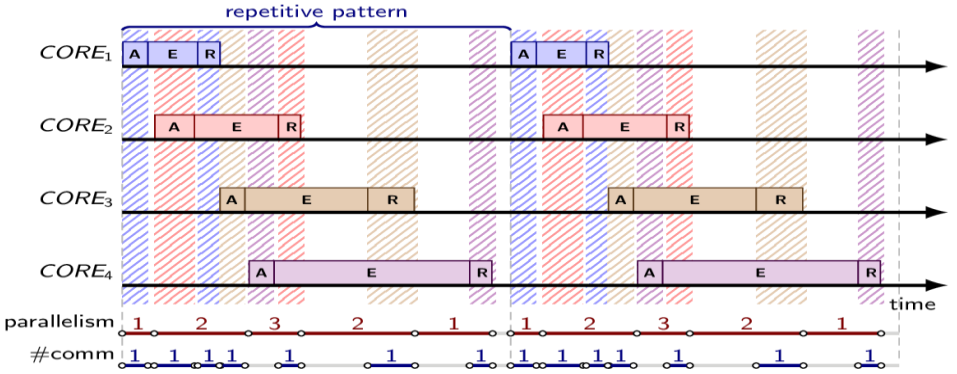
- DAL-E
- Runs the passengers' entertainment systems composed of multimedia applications
- No real safety requirements
- High **processing** and **communication** requirements
- New security related issues (Wi-Fi)



CONTROL-ORIENTED DETERMINISTIC PLATFORM SOFTWARE: DETERMINISTIC EXECUTION MODEL (APPLICATION AWARE)

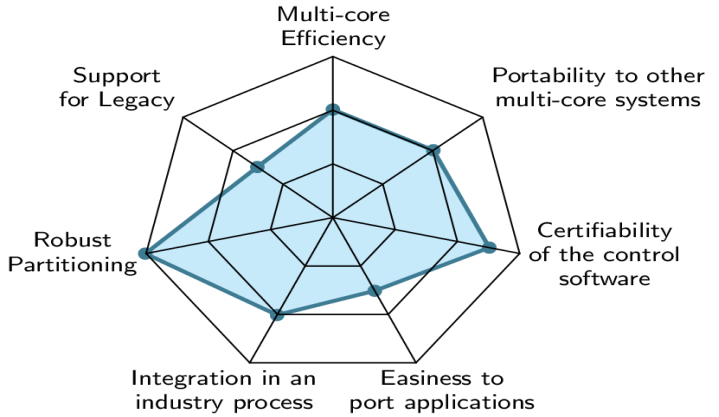
Principles

- Applies a strict execution model to the software
- Decoupling tasks into **execution** and **communication** phases
- Determine a **static schedule** with execution phases running in parallel and sequential communication phases



Evaluation against the properties

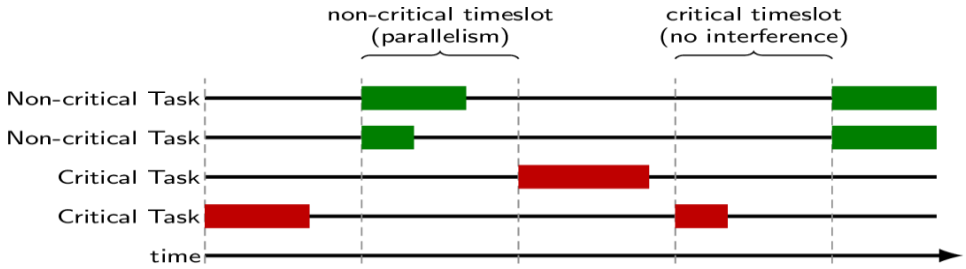
- Ensures time partitioning
- Restrict parallelism to execution sections
- Requires to alter the source code
- Static scheduling ➔ the platform software remains very simple



CONTROL-ORIENTED DETERMINISTIC PLATFORM SOFTWARE: DETERMINISTIC ADAPTIVE SCHEDULING (APPLICATION UNAWARE)

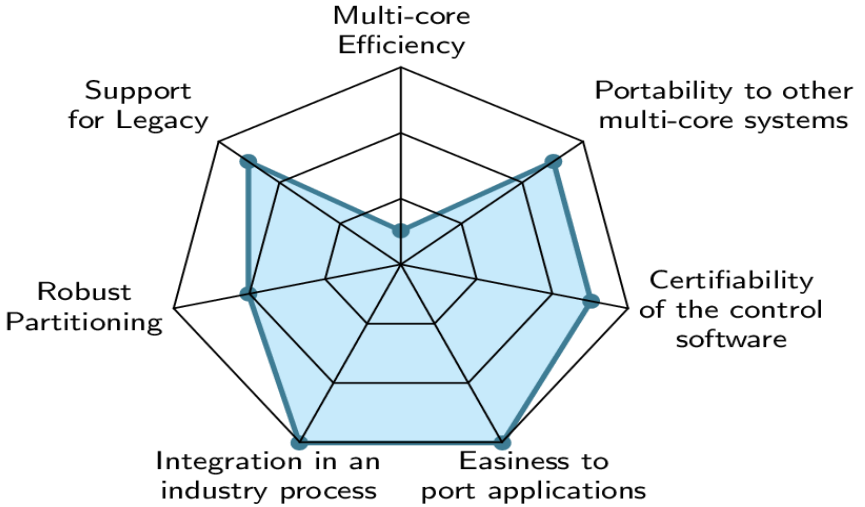
Principles

- For the usage of multi-core in a mixed critical context
- System-wide **time partitioning** with critical and non-critical time slots.
- Critical applications are running standalone ➔ **no interferences**
- Non-critical applications can run in parallel



Evaluation against the properties

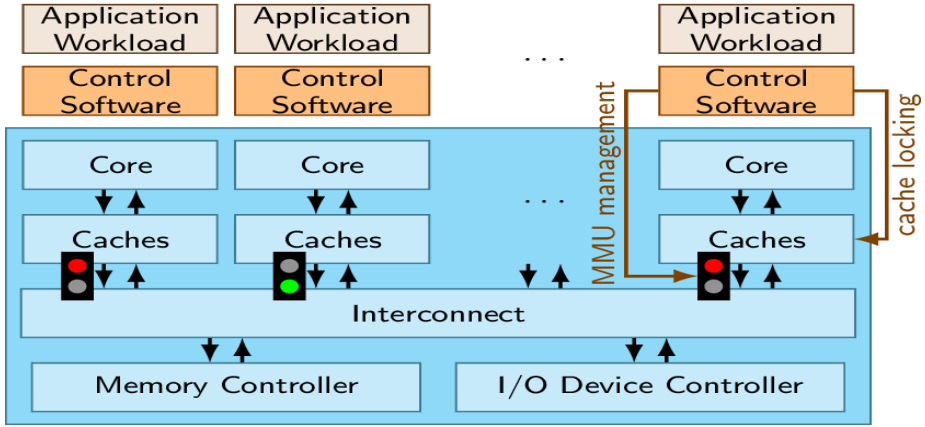
- Only exploit multi-core efficiency for non-critical tasks
- Can rely on existing processes to certify critical applications
- No modification to legacy software
- Already used in an industry process



CONTROL-ORIENTED DETERMINISTIC PLATFORM SOFTWARE: MARTHY (APPLICATION UNAWARE)

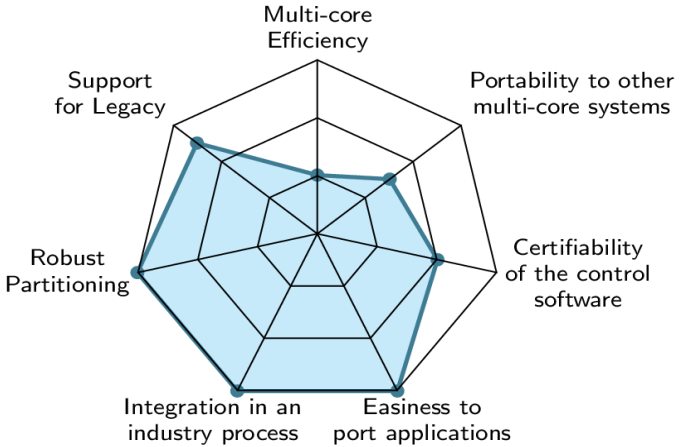
Principles

- During a timeslot, only allows one of the running tasks to access the shared hardware resources
- Relies on **cache locking** and **MMU reprogramming** to prevent tasks to access resources beyond their private caches
- **Stall** cores trying to access shared resources outside of their **TDMA slot**



Evaluation against the properties

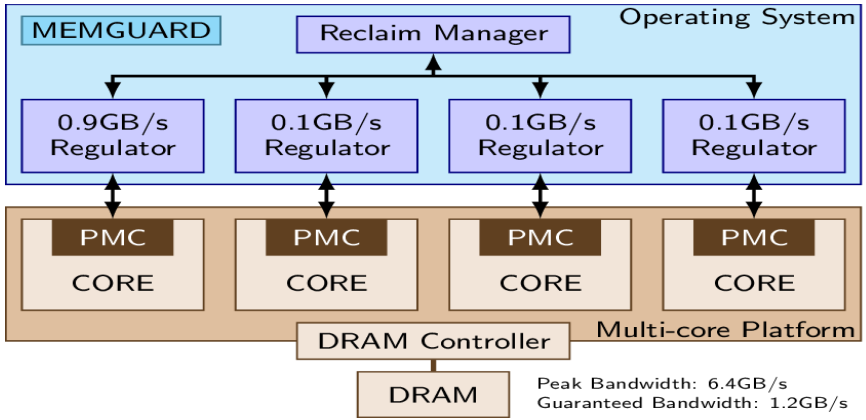
- Full support for time partitioning through TDMA
- Legacy applications can run unmodified
- Involves a complex and architecture dependent platform software
- Efficiency depends on cache locality



REGULATION-ORIENTED DETERMINISTIC PLATFORM SOFTWARE: MEMGUARD

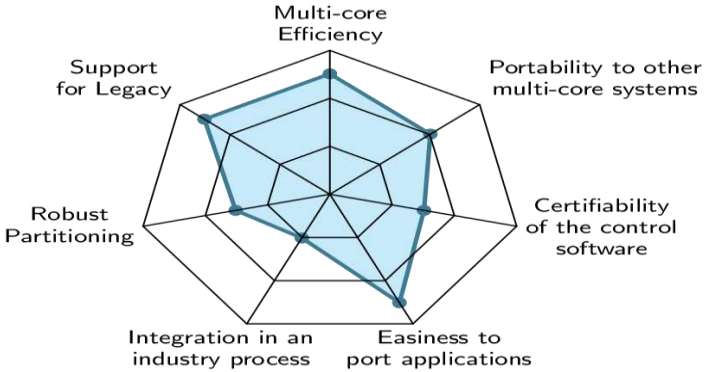
Principles

- **Allocates** a maximum **bandwidth** usage per timeslots
- Relies on **performance counters** to count #accesses per slot
- **Limits interferences** by keeping the total bandwidth below a guaranteed available bandwidth



Evaluation against the properties

- Support for legacy applications, potentially exploiting multi-core parallelism
- Requires to evaluate the required bandwidth for each timeslot of each application, and to determine an guaranteed bandwidth value
- Robust partitioning is not really ensured, only keeping interference within an acceptable range



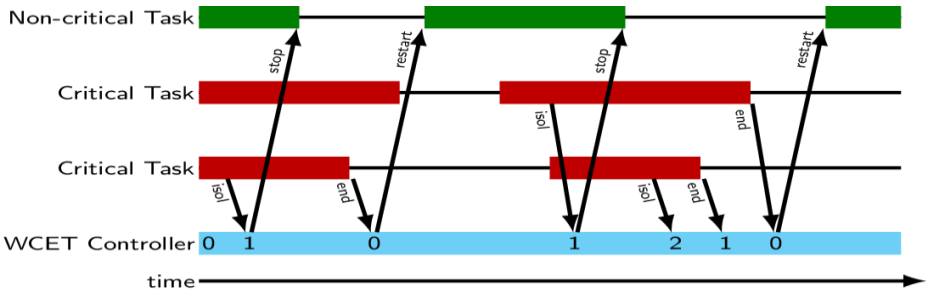
Memguard: Memory Bandwidth Reservation System for Efficient Performance Isolation in Multicore Platforms

H. Yun, G. Yao, R. Pellizzoni, M. Caccamo, L. Sha

REGULATION-ORIENTED DETERMINISTIC PLATFORM SOFTWARE: DISTRIBUTED RUNTIME WCET CONTROLLER

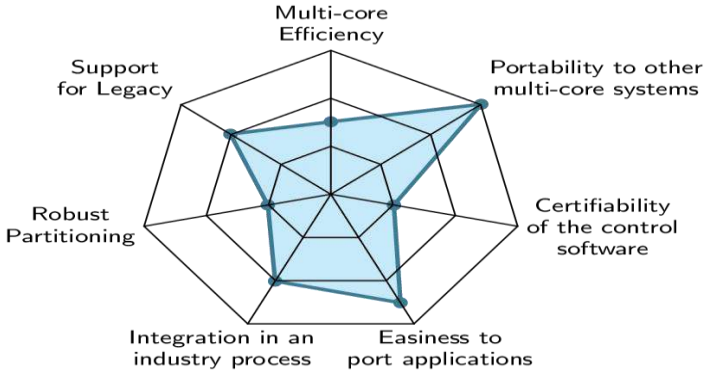
Principles

- Regularly checks if the critical tasks can **tolerate** the **interferences** due to other co-running tasks
- Critical tasks can request the controller to temporarily **suspends** non-critical tasks to run in **isolation**



Evaluation against the properties

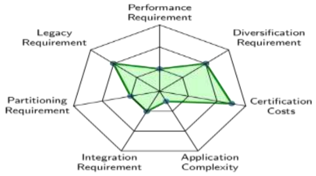
- Robust partitioning is not strictly ensured, again focusing on controlling the impact of interferences
- Enforce to modify the critical applications to insert isolation sections and checkpoints
- Being reactive, it is easily applicable to a large set of architectures



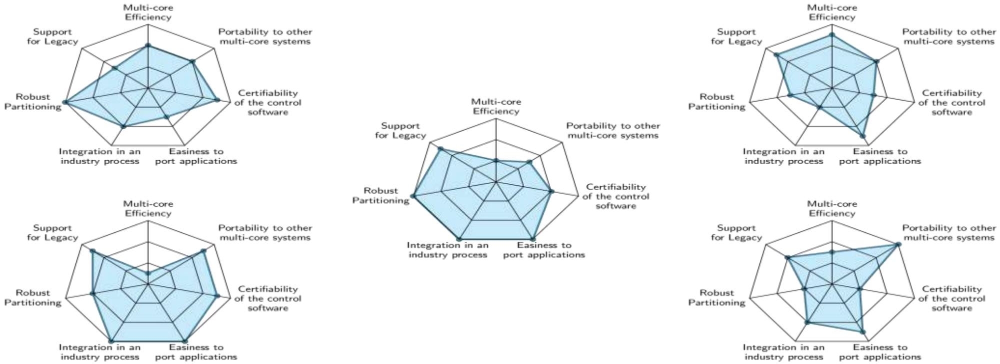
[**Distributed Run-Time WCET Controller for Concurrent Critical Tasks in Mixed-Critical Systems**
A. Kritikakou, C. Pagetti, C. Rochange, M. Roy, M. Faugère, S. Girbal, D. Gracia Pérez

EVALUATING DETERMINISTIC PLATFORM SOFTWARE AGAINST AVIONIC CASE STUDIES

Evaluation Principles

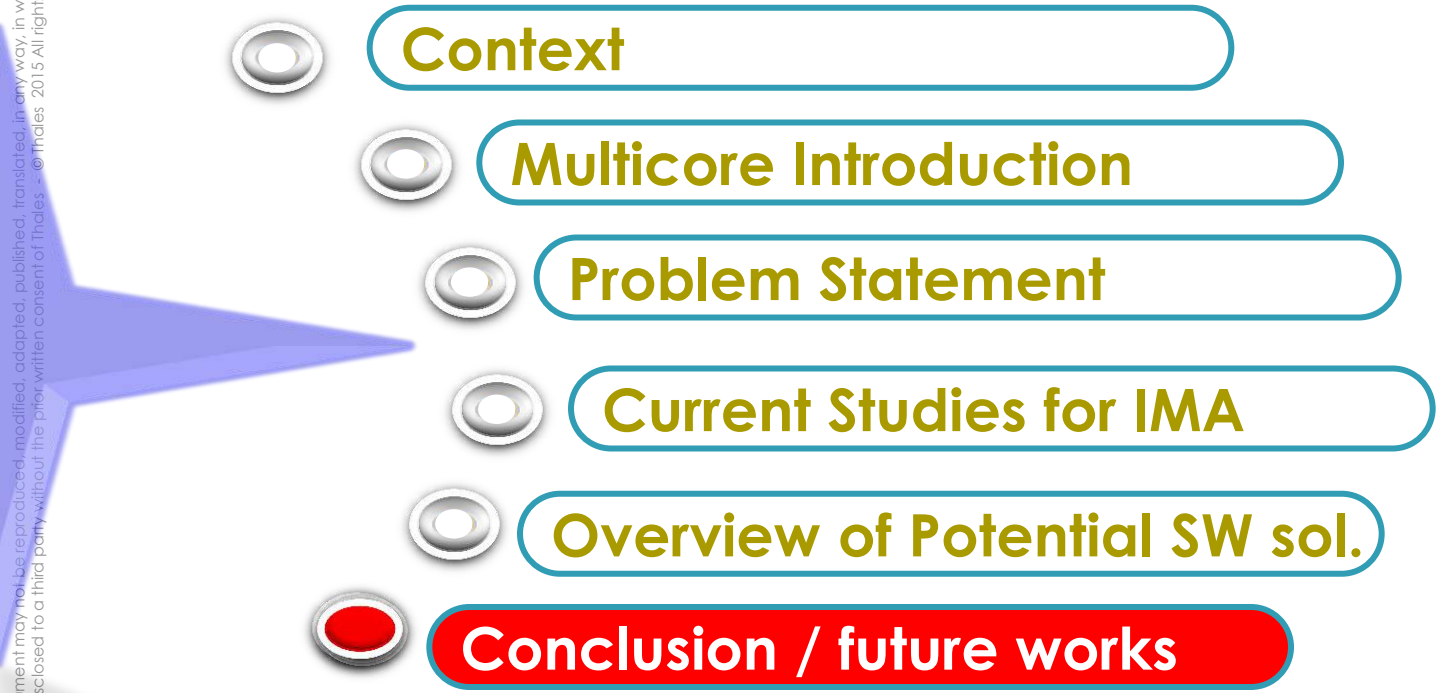


VS



	Good	Average	Bad
Deterministic Execution Model	FADEC IMA	DS	IFE
Deterministic Adaptive Scheduling	FADEC IFE	IMA	DS
Marthy	IMA	FADEC DS	IFE
Memguard	IFE	DS	FADEC IMA
Runtime WCET Controller	DS IFE	-	FADEC IMA

SUMMARY



CONCLUSION

Survey of Deterministic Platform Software

- Identifying key properties for avionic applications
- Performing a high-level evaluation of property assessments for 5 DPS solutions

Evaluation versus 4 avionic Case Studies

- Covering various safety levels
- Including different level of requirements

Conclusion of the Evaluation

- All proposed solutions are reaching a different compromise on the identified key properties
- All the proposed properties both perfectly fit one of the case study while not being compliant with another
- All these solutions also have a different level of maturity

QUESTIONS ?