Mixed Critical systems issues in relation to spacecraft avionics

M. Verhoef (TEC-SWE), J. Lopez Trescastro (TEC-SWS), L. Fossati (TEC-EDD)

ADCSS Session 3 - 19-10-2016

Introduction to the session

- What is mixed criticality – what are mixed criticality systems
- Why are we interested in mixed criticality
- What are the key issues to solve
- How do we implement mixed criticality
- Short overview of ESA activities

- This session: 5 talks
  - Mixed criticality: overview and SothA in this R&D domain (BSC)
  - State of practice in Airbus / Astrium
  - State of practice in Thales Alenia Space (with application on Iridium)
  - State of practice in CNES
  - State of practice in other application domains (IKERLAN)

- Round-table: *Are we ready for tomorrow's platforms?*

What is mixed criticality?

- execute multiple software artifacts concurrently on a single computational unit
- those software artifacts being (potentially) at different criticality levels
- assuming each artifact is shown to be correct in isolation, what guarantees does that provide for the mixed criticality case?

Table D-1: Software criticality categories

Category	Definition
A	Software that if not executed, or if not correctly executed, or whose anomalous behaviour can cause or contribute to a system failure resulting in: → Catastrophic consequences
B	Software that if not executed, or if not correctly executed, or whose anomalous behaviour can cause or contribute to a system failure resulting in: → Critical consequences
C	Software that if not executed, or if not correctly executed, or whose anomalous behaviour can cause or contribute to a system failure resulting in: → Major consequences
D	Software that if not executed, or if not correctly executed, or whose anomalous behaviour can cause or contribute to a system failure resulting in: → Minor or Negligible consequences

ECSS-Q-ST-80C

Table 6-1: Severity of consequences

Severity	Level	Dependability (refer to ECSS-Q-ST-30) *Extract from ECSS-Q-ST-30*	Safety (ECSS-Q-ST-40)
Catastrophic	1	Failures propagation	Loss of life, life-threatening or permanently disabling injury or occupational illness; Loss of system; Loss of an interfacing manned flight system; Loss of launch site facilities; Severe detrimental environmental effects.
Critical	2	Loss of mission	Temporarily disabling but not life-threatening injury, or temporary occupational illness; Major damage to interfacing flight system; Major damage to ground facilities; Major damage to public or private property; Major detrimental environmental effects.
Major	3	Major mission degradation	---
Minor or Negligible	4	Minor mission degradation or any other effect	---
NOTE: When several categories can be applied to the system or system component, the highest severity takes priority			

ECSS-Q-ST-40C

Why are we interested in mixed criticality?

- Avionics trend: maximize performance and reduce cost, size, weight and power
  - Single core performance is increasing
  - Multi core architectures are gaining momentum (GR712, GR740)
  - Many core architectures are emerging

- IMA enabler for cost savings (less components, simplify integration and V&V)
  - exploit this computational power (optimal use of resources)
  - control the inherent system complexity (design predictability)
  - guarantee system correctness (with acceptable residual failure risk)

System correctness: functional correctness + timing / throughput predictability

Low-criticality applications must not affect high-criticality ones

Key challenge: how to share (hardware) resources to ensure timing predictability

- main sources of (timing) disturbances: FPU and cache behavior
- schedulability analysis on multi/many-core is far from trivial

How do wo address mixed criticality : TSP

Separation of concerns between functionally independent software components to contain and/or isolate faults and reduce the effort of the software integration, verification and validation process.

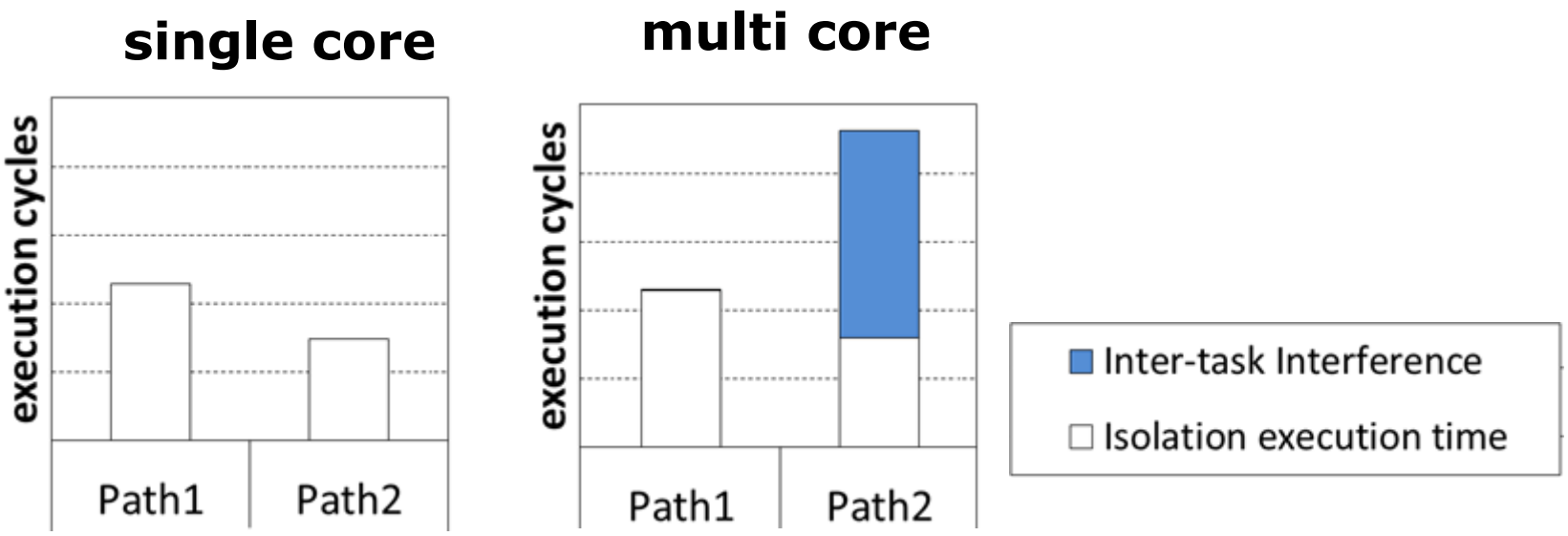
Time and Space Partitioning in Spacecraft Avionics, James Windsor, Kjeld Hjortnaes, Third IEEE International Conference on Space Mission Challenges for Information Technology

In practice:

- Tasks running on the various cores should not be able to access each other's memory locations/IOs – space partitioning
  - This can be addressed using MMUs and IOMMUs

- Tasks running on the various cores should not affect the execution time of each other – time partitioning
  - This can be *partly* solved: memory bus access protocols, cache partitioning

Multi core and hard real-time: challenging!

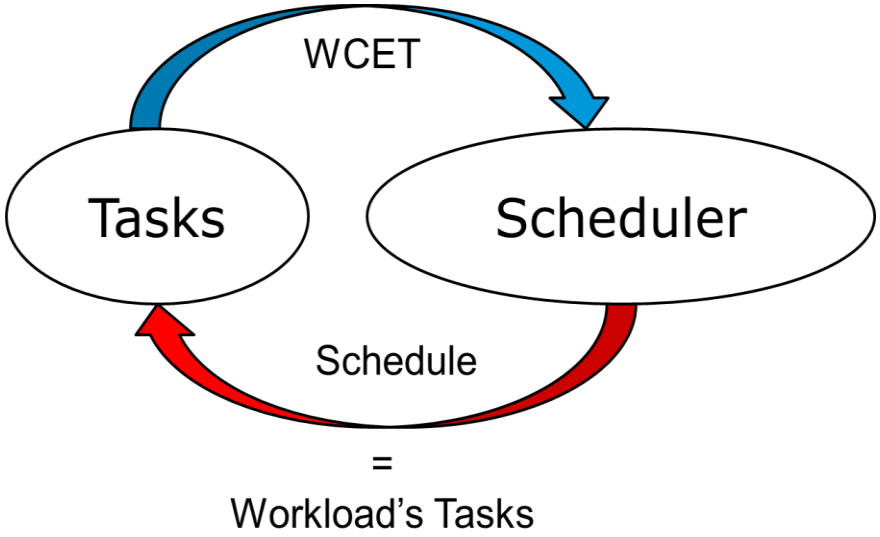


* Real data obtained from the NGMP-GR740 processor

- Execution time of a task in a multi-core depends on the co-running tasks
  - Tasks access hardware resources at the same time

- Harder to time analyze w.r.t. single-core chips
  - Complexity of analyzability explodes with number of tasks

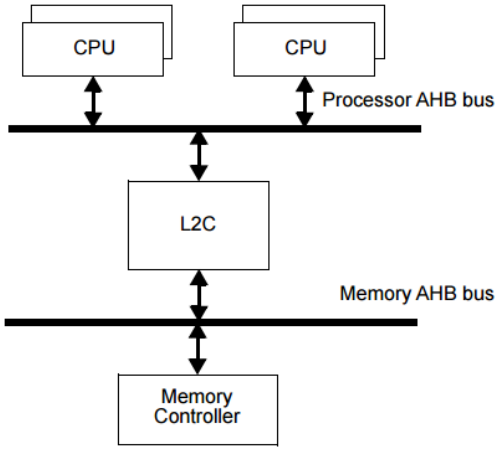
Time predictability and time partitioning : two sides of the same problem



- tasks in isolation (single core view) must be time predictable in order to be able to come-up with a per-core/partition schedule
  - already complicated in moderately advanced architectures, in particular with multi-level caches
- moving to multi-cores (with shared resources) also causes the various schedules (for each core/partition) to depend on each other
  - this makes schedulability analysis unmanageable in the traditional sense

Deterministic versus probabilistic WCET

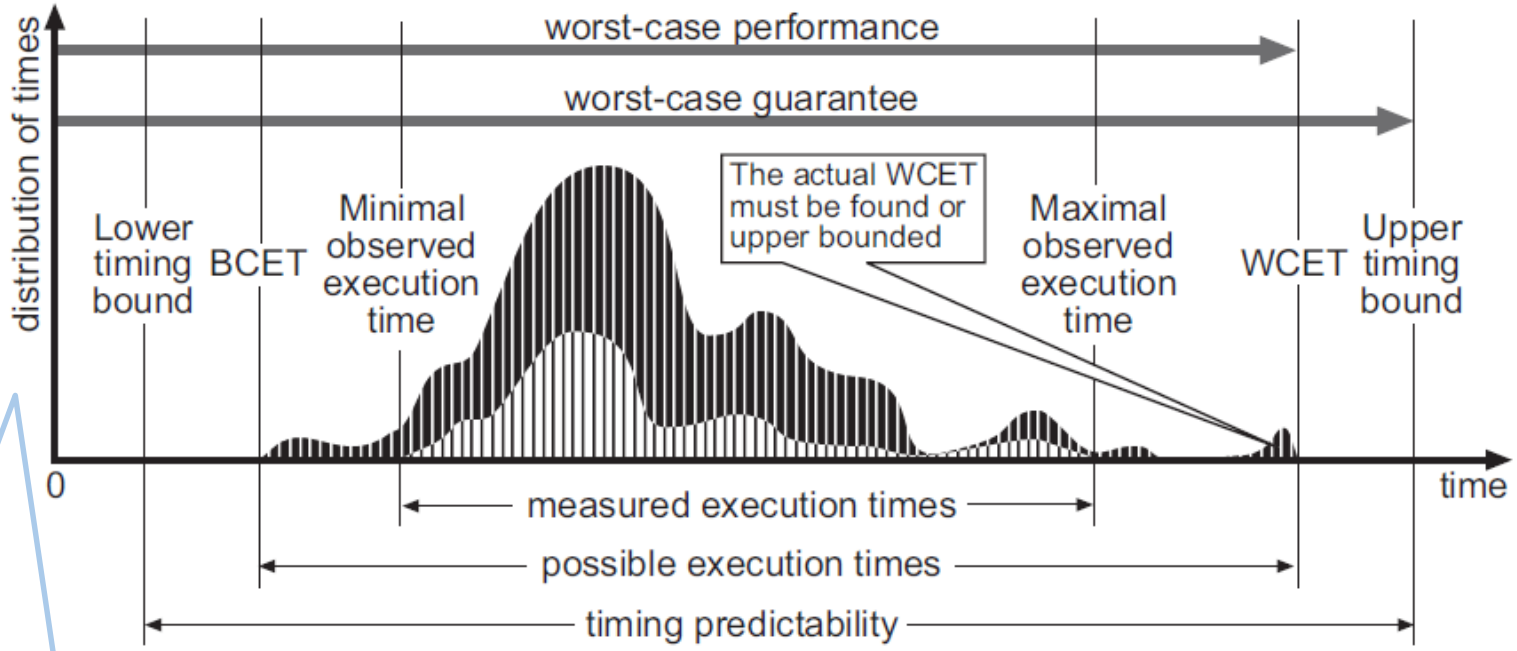
- A deterministic WCET can (of course) be computed:
  - For example on the NGMP, assume that every instruction fetch/data access to miss in the L1-cache, wait for the max arbitration time on the bus, miss in the L2-cache, access external memory: rough estimation of 1+12+20 = 33 cycles to retrieve an instruction from memory for a single core, 33*4=132 cycles considering the sharing of the processor bus → *two orders of magnitude slower than top performance*



- To have deterministic WCET, huge margins have to be taken, which would make the use of multi-core very unattractive → *this is not a practical solution*

- New approaches must be found: probabilistic timing analysis

Probabilistic WCET



- Qualitative Diagram (source Proartis/Proxima FP7 projects)
- Exec time variations are due to input data, inter-task and inter-core interferences, IO, etc.

- We can shift the curve left/right (varying the average performance) …
- We can sharpen the bell (reducing the probability of very long execution times) …
- … **but there will always be a tail!**

Reduction of interferences

- The right tail (wcet) cannot be eliminated, only its probability can be reduced
- But can we reduce it to acceptable levels? What are acceptable levels?

- Tricks up or sleeve to reduce these interferences
  - Main processor bus: Round-Robin arbitration, 128-bit width, limited burst length
  - L2 cache: Way partitioning per master, possibility of connecting the IOMMU (so the IO masters) to the memory bus
  - Line locking, preventing eviction

- *Under investigation (smoothing out outliers by improving randomness)*:
  - SPLIT response for L2-cache misses, and non-blocking pipeline in L2
  - Multilayer bus and multi-port L2 cache
  - Mechanisms to limit the shared resource usage
  - Mechanisms to influence the shared resource usage

Ways to implement mixed criticality

Symmetric Multi Processing:

One single OS is managing all the cores

Pros: performance

Cons: all applications have to use the same OS, complex OS to qualify

(open source) example: RTEMS-SMP

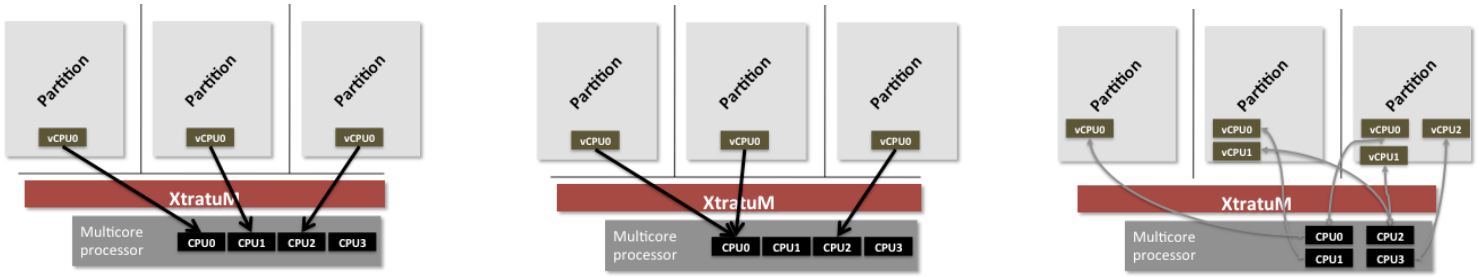
Asymmetric Multi Processing:

Different OS on different cores

Pros: OS diversity

Cons: all the OS needs to be qualified, space separation difficult to enforce

(open source) example: Xtratum



Overview of ESA R&D activities

Hardware Oriented (around 2.0M€ allocated so far)

- NGMP Phases 1 and 2, commercial prototypes → GR740 Quad-core LEON4FT
- NPI/Ph.D student on Architectural solutions for the timing predictability of next-generation multi-core processors

Software Oriented (around 1.6M€ allocated so far + FP7 projects)

- Porting of XTRATUM Hypervisor on the NGMP
- Creation of an SMP version of RTEMS (2 parallel contracts) + follow-up activity
- Porting of AIR Hypervisor on the NGMP
- Various benchmarking activities
- European FP7 projects: Merasa, Proartis, Proxima, Multipartes
- Promixa4Space
- IMA4Space, IMA Tools
- Schedulability Analysis of Multi-Core architectures (2 parallel projects)
- Parallel programming models for space (ITI)
- In planning: IMA and RTEMS-SMP qualification