A Survey of Mixed-Criticality Systems Implementation Techniques

Authors:

V. Muttillo[1], L. Pomante[1], G. Valente[1]

[1] *Center of Excellence DEWS, University of L'Aquila, Italy*

EMC2 at Mixed-Criticality Cluster Workshop, Barcelona, 22 Nov

Summary

EMC²

1.

Introduction

“Brief Introduction to Mixed Criticality and Cyber Physical Systems”

Cyber-Physical Systems

- A cyber-physical system (CPS) is an integration of computation with physical processes whose behavior is defined by both cyber and physical parts of the system.

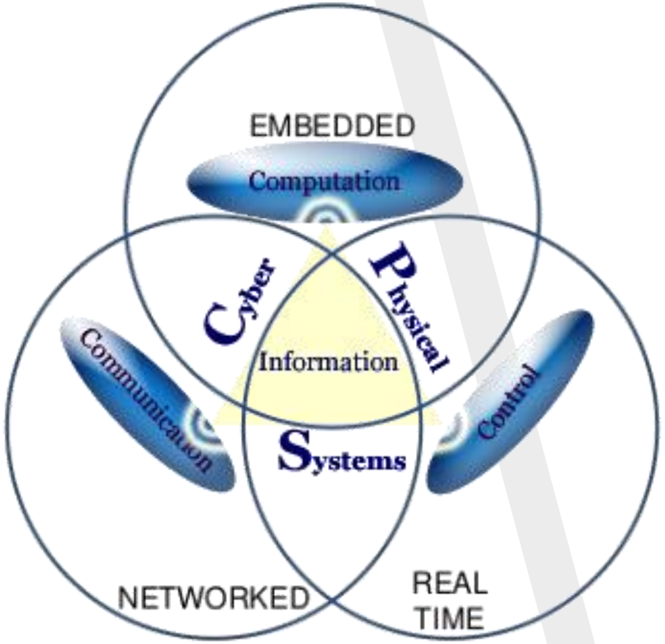
- Embedded computers and networks monitor and control the physical processes, usually with feedback loops where physical processes affect computations and vice versa.

Cyber-Physical Systems

- A cyber-physical system (CPS) is an integration of computation with physical processes whose behavior is defined by both cyber and physical parts of the system.

- Embedded computers and networks monitor and control the physical processes, usually with feedback loops where physical processes affect computations and vice versa.

- As an intellectual challenge, CPS is about the intersection, not the union, of the physical and the cyber*.



** Lee, E. A., Seshia, S. A.,: Introduction to Embedded Systems, a Cyber-Physical Systems approach, Second Edition, LeeSeshia.org, 2015*

Embedded Systems

- In contrast to a generic reprogrammable general purpose computer, an embedded system is composed of a set of tasks already known during the development. This make possible to identify a hardware/software combination specifically designed for such an application.

- Hardware can be reduced to a minimum in order to reduce area, consumption, processing times and manufacture cost, while considering F/NF requirements.

- Many embedded systems are also real-time systems, in which “the correctness of the system behavior depends not only on the logical results of the computations, but also on the time when these results are produced”

embedded/**real-time**

Mixed-Criticality Systems

- A mixed criticality system is "**an integrated suite of HW, OS, middleware services and application software that supports the concurrent execution of safety-critical, mission-critical, and non-critical software within a single, secure computing platform**", i.e. a system containing computer hardware and software that executes concurrently several applications of different criticality (such as safety-critical and non-safety critical).

- Different criticality applications are engineered to provide different levels of assurance, with high criticality applications being the most costly to design and verify.

- Mixed-Criticality systems are typically embedded in more complex systems such as an aircraft whose safety must be ensured.

Embedded/real-time/ safety-critical/**mixed-critical**

EMC²

2.

Safety-Related Standards

“Criticality is a designation of the level of assurance against failure needed for a system component”

Safety-Related Standards

- Most of the MCS-related researches published in the state-of-the-art cite the **safety-related standards** associated to each application domain (e.g. aeronautics, space, railway, automotive) to justify their methods and results. However, those standards are not, in most cases, freely available, and do not always clearly and explicitly specify the requirements for mixed-criticality

- New MC task model is in essence the result of combining the **standard hard real-time requirements** (studied by the real-time research community since the 70's) with the **notion of "criticality" of execution**. When transposed into the industrial world, the applications that better to such a MC model and its combined requirements are those in which a part of the core functionality is delivered by safety-critical components.

Safety-Related Standards

- **GENERAL** (IEC-61508) based on **SIL (Safety Integrity Level)**: Functional safety standards (of electrical, electronic, and programmable electronic)

  - **AUTOMOTIVE** (ISO26262) based on **ASIL (Automotive Safety Integrity Level)** (Road vehicles - Functional safety)
  - **NUCLEAR POWER** (IEC 60880-2)
  - **MEDICAL ELECTRIC** (IEC 60601-1)
  - **PROCESS INDUSTRIES** (IEC 61511)
  - **RAILWAY** (CENELEC EN 50126/128/129])
  - **MACHINERY** (IEC 62061)

- **AVIONIC** based on **DAL (Development Assurance Level)** related to ARP4761 and ARP4754
  - DO-178B (Software Considerations in Airborne Systems and Equipment Certification)
  - DO-178C (Software Considerations in Airborne Systems and Equipment Certification, replace DO-178B)
  - DO-254 (Airborne - Design), similar to DO-178B, but for hardware
  - DO-160F (Airborne - Test)

- **MEDICAL DEVICE**
  - FDA-21 CFR
  - IEC-62304

3. Mixed Criticality Systems Analysis

“The more confidence one needs in a task execution time bound (the less tolerant one is of missed deadlines), the larger and more conservative that bound tends to become in practice”

EMC²

MCS State-Of-The-Art Model

- Almost 200 papers treating of the scheduling of MCS have been referenced in Burns and Davis* paper, and tens of related papers are still published every year. Most of the works about MCS published by the real-time scheduling research community are based on a **model proposed by Vestal*** paper.

- This model assumes that the system has several modes of execution, say **modes {1, 2, … , L}**. The application system is a **set of real-time tasks**, where each task τ_i is characterized by a period T_i and a deadline D_i (as in the usual real-time task model), an assurance level li and a set of **worst-case computational estimates** $\{C_{i,1}, C_{i,2}, \dots , C_{i,l_i}\}$, under the assumption that $C_{i,1} \leq C_{i,2} \leq \dots \leq C_{i,l_i}$

- The different WCET estimates are meant to model estimations of the **WCET at different assurance levels**. The worst time observed during tests of **normal operational scenarios** might be used as $C_{i,1}$ whereas at **each higher assurance level** the subsequent estimates $\{C_{i,2} , \dots , C_{i,l_i} \}$ are assumed to be obtained by more conservative **WCET analysis techniques**.

** Burns, A, Davis, R.I.: "Mixed Criticality Systems - A Review", University of York, 4 March 2016.*
*** S. Vestal, "Preemptive Scheduling of Multi-criticality Systems with Varying Degrees of Execution Time Assurance," Real-Time Systems Symposium (RTSS) 28th IEEE International on, Tucson, AZ, 2007, pp. 239-243.*

MCS Behavioral Model

- The system starts its execution in **mode 1** and **all tasks are scheduled** to execute on the core[s]. Then at runtime, if the system is running in **mode k** then each time the **execution budget $C_{i,k}$ of a task τ_i is overshot**, the **system switches to mode k+1**. It results from this transition from mode k to mode k+1 that **all the tasks of criticality not greater than k** (i.e., $l_i \geq k$) are suspended. Mechanisms have also been proposed to eventually re-activate the dropped tasks at some later points in time*.

- It must be noted that one of the simplifications of this model is the **Vestal's model** with **only two modes**, usually referred to as **LO** and **HI** modes (which stand for **Low- and High-criticality modes**). Multiple variations of that scheduling scheme exist (please refer to [3] for a comprehensive survey); some for single-core, others for multicore architectures. In the case of multicore, both global and partitioned scheduling techniques have been studied. Solutions for **fixed priority scheduling (RM), Earliest Deadline First (EDF)** and **time triggered scheduling** have been proposed. Note that some works also propose to **change the priorities or the periods of the tasks during a mode change** rather than simply stopping the less critical ones.

** F. Santy, G. Raravi, G. Nelissen, V. Nelis, P. Kumar, J. Goossens, and E. Tovar. Two protocols to reduce the criticality level of multiprocessor mixed-criticality systems. In RTNS 2013, RTNS '13, pages 183–192. ACM, 2013.*

4.

Mixed-Criticality Classification

"A major industrial challenge arises from the need to face cost efficient integration of different applications with different levels of safety and security on a single computing platform in an open context"

MCS Architectures

Separation technique:

- **Timing separation**: scheduling policy, temporal partitioning with HVP, NoC
- **Spatial separation**: one task per core, one task on HW ad hoc (DSP, FPGA), spatial partition with HVP, NoC, MMU, MPU etc.

➢HW:

- **Temporal isolation**: Scheduling HW
- **Spatial isolation**: separated Task on dedicated components (HW ad hoc, FPGA etc.)

➢Single core:

- **Temporal isolation**: Scheduling policy with SO o RTOS, Scheduling policy with HVP
- **Spatial isolation** : MMU, MPU, HVP Partitioning

➢Multi-core

- **Architecture**: shared memory systems, Uniform Memory Architecture, UMA (SMP), Not Uniform Memory Architecture, NUMA, distributed systems, NoC
- **Temporal isolation**: Scheduling policy with SO o RTOS, Scheduling policy with HVP
- **Spatial isolation**: MMU, MPU, HVP partitioning

➢Many-core

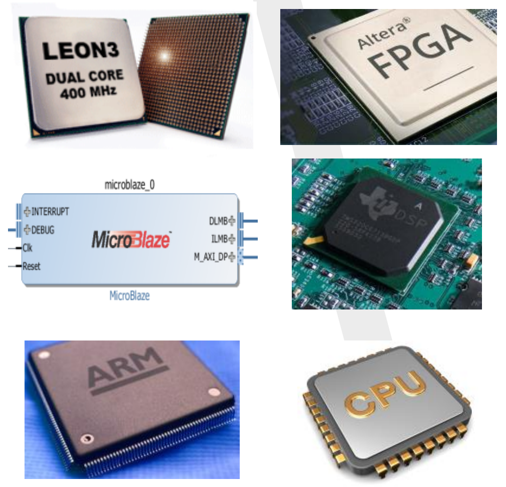
- **Work in progress**

MCS Technologies

Tecnologies:

- **Hardware:** HW ad hoc, FPGA, DSP, Processor
  - ➢ **Processor**: LEON3, ARM, MICROBLAZE etc.

Hardware



MCS Technologies

Tecnologies:

- **Hardware**: HW ad hoc, FPGA, DSP, Processor
  - ➢ **Processor**: LEON3, ARM, MICROBLAZE etc.
- **Software**: Bare-metal, OS, RTOS, HVP
  - ➢ **OS**: Linux etc.

Hardware

OS

EMC²

MCS Technologies

Tecnologies:

- **Hardware:** HW ad hoc, FPGA, DSP, Processor
  - ➢ **Processor**: LEON3, ARM, MICROBLAZE etc.
- **Software**: Bare-metal, OS, RTOS, HVP
  - ➢ **OS**: Linux etc.
  - ➢ **RTOS**: eCos, RTEMS, FreeRTOS, Threadx, VxWorks, EriKa etc.

Hardware

RTOS

THREADX
ecos®
VxWorks
RTEMS™
ERIKA ENTERPRISE
freeRTOS

OS

MCS Technologies

Tecnologies:

- **Hardware:** HW ad hoc, FPGA, DSP, Processor
  - ➢ **Processor**: LEON3, ARM, MICROBLAZE etc.
- **Software**: Bare-metal, OS, RTOS, HVP
  - ➢ **OS**: Linux etc.
  - ➢ **RTOS**: eCos, RTEMS, FreeRTOS, Threadx, VxWorks, EriKa etc.
  - ➢ **HVP**: PikeOS, Xtratum, Xen etc.

HVP

PikeOS

XtratuM

Xen

RTOS

THREADX

eCos®

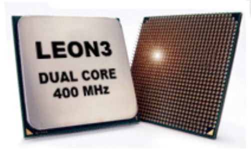
VxWorks

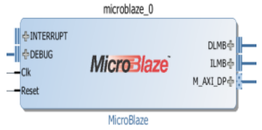
RTEMS™

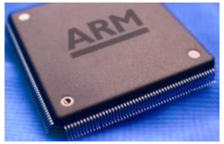
ERIKA ENTERPRISE

freeRTOS

Hardware

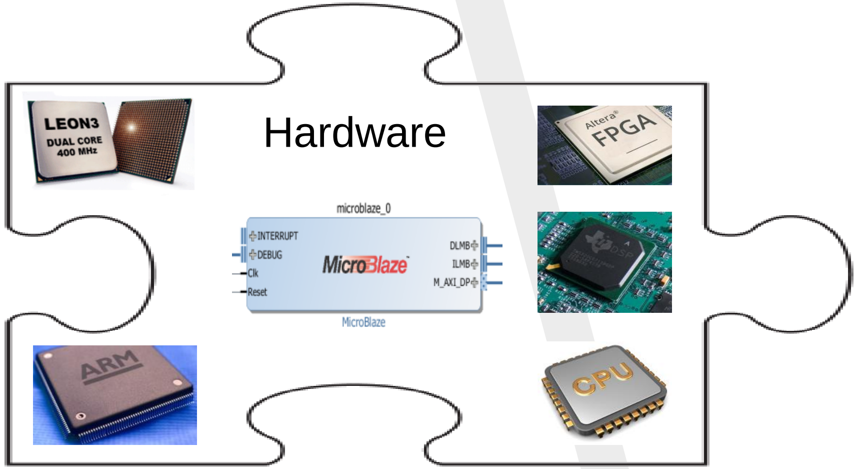






OS

MCS Implementations



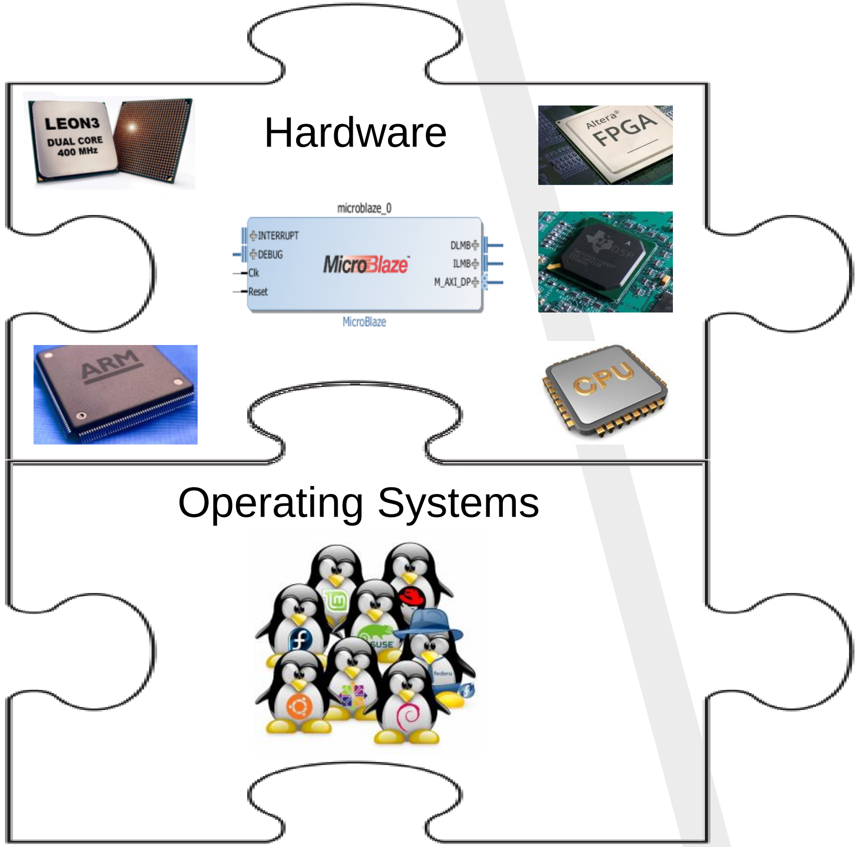
Scheduling:

- **O-Level**

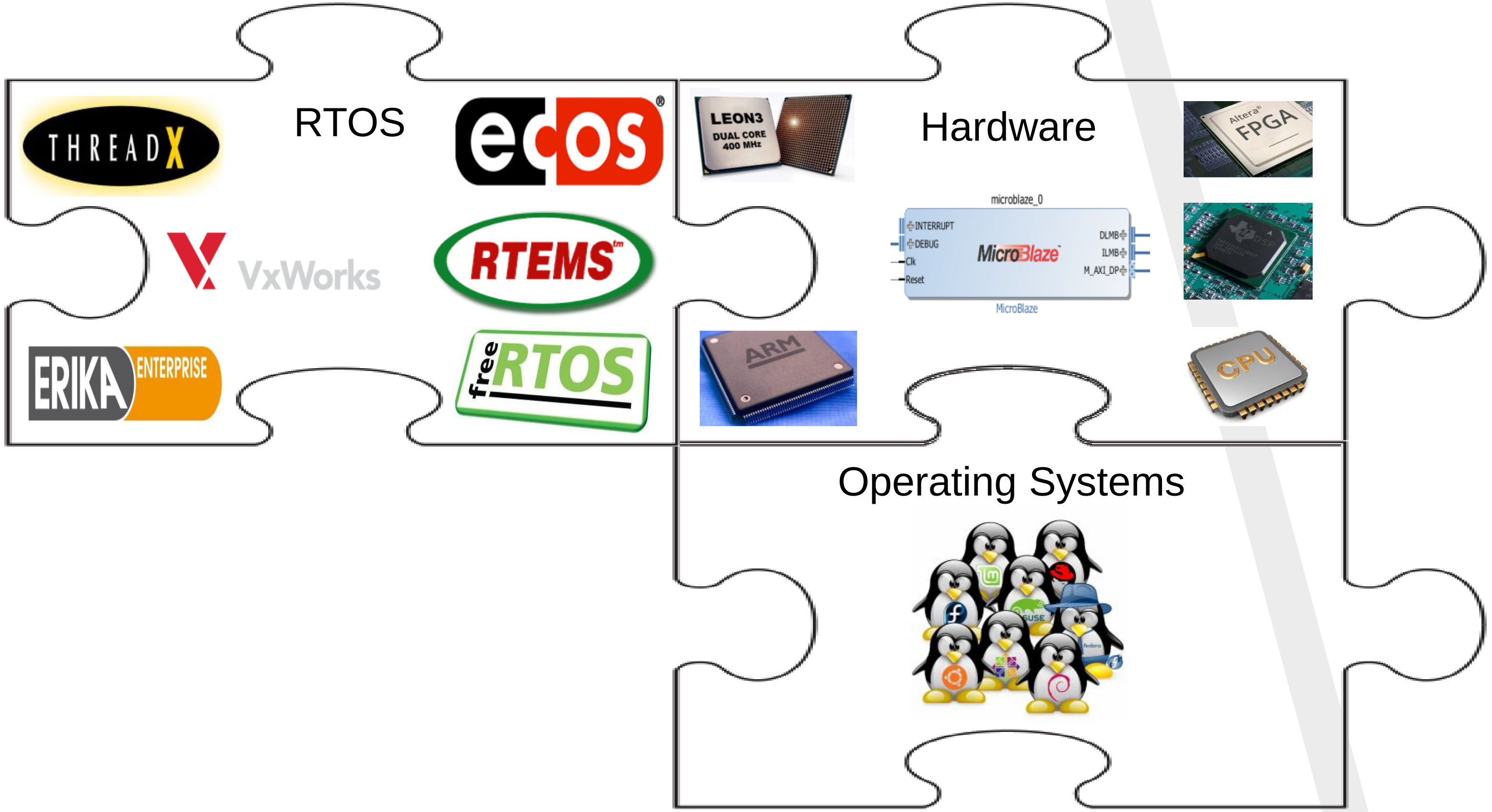
MCS Implementations

Scheduling:

- 0-Level
- 1-Level



MCS Implementations



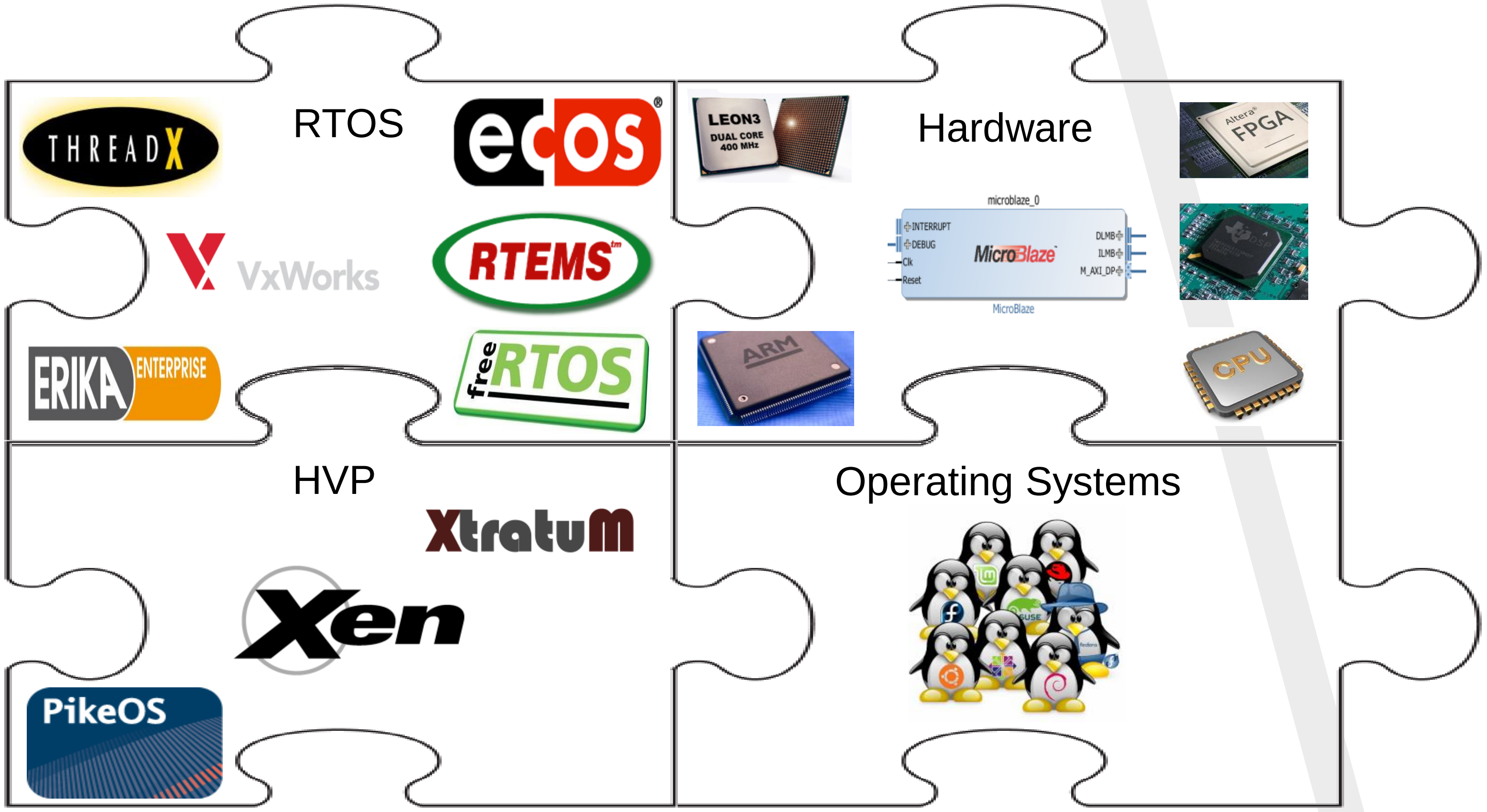
Scheduling:

- 0-Level
- 1-Level

MCS Implementations

Scheduling:

- 0-Level
- 1-Level
- 2-Level



MCS Classification

Separation Technique	HW	Single core	Multi-core
Spatial	0-level scheduling [10]	0-level scheduling [11][16]	0-level scheduling [15]
		1-level scheduling [2][5][10][13][16]	1-level scheduling [4][9][15][16]
		2-level scheduling [6][11]	2-level scheduling [3][4][6][7][8][9][14]
Temporal	0-level scheduling [10]	0-level scheduling [11][16]	0-level scheduling [15][16]
		1-level scheduling [1][2][10][13][16]	1-level scheduling [4][9][12][15][16]
		2-level scheduling [6][11][16]	2-level scheduling [1][4][6][7][8][9][14]

Reference

[1] S. Vestal, “Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance,” in Proc. Int’l Real-Time Systems Symp. (RTSS), 2007

[2] A. Gerstinger, H. Kantz, C. Scherrer: ”TAS Control Platform: A Platform for Safety-Critical Railway Applications”, ERCIM NEWS 75, Oct. 2008

[3] D. Muench, O. Isfort, K. Mueller, M. Paulitsch, A. Herkersdorf: "Hardware-Based I/O Virtualization for Mixed Criticality Real-Time Systems Using PCIe SR-IOV," 2013 IEEE 16th International Conference on Computational Science and Engineering, Sydney, NSW, 2013, pp. 706-713

[4] M. Paulitsch, O. M. Duarte, H. Karray, K. Mueller, D. Muench, J. Nowotsch: "Mixed-Criticality Embedded Systems -- A Balance Ensuring Partitioning and Performance" Digital System Design (DSD), 2015 Euromicro Conference on, Funchal, 2015, pp. 453-461

[5] M. G. Hill, T. W. Lake: "Non-interference analysis for mixed criticality code in avionics systems," Automated Software Engineering, 2000. Proceedings ASE 2000. The Fifteenth IEEE International Conference on, Grenoble, France, 2000, pp. 257-260.

[6] J. E. Kim, M. K. Yoon, S. Im, R. Bradford, L. Sha: “Optimized Scheduling of Multi-IMA Partitions with Exclusive Region for Synchronized Real-Time Multi-Core System” in Proceedings of the 16th ACM/IEEE Design, Automation, and Test in Europe (DATE 2013), Mar. 2013.

[7] J. E. Kim, M. K. Yoon, R. Bradford, L. Sha, "Integrated Modular Avionics (IMA) Partition Scheduling with Conflict-Free I/O for Multicore Avionics Systems," to appear in Proceedings of the 38th IEEE Computer Software and Application Conference (COMPSAC 2014), Jul. 2014.

[8] F. Federici, V. Muttillo, L. Pomante, G. Valente, D. Andreetti, D. Pascucci: “Implementing mixed-critical applications on next generation multicore aerospace platforms”, CPS Week 2016, EMC^2 Summit, Vienna, Austria

[9] B. Huber, C. El Salloum, and R. Obermaisser. A resource management framework for mixed-criticality embedded systems. In 34th IEEE IECON, pages 2425–2431, 2008

Reference

[10] R. Pellizzoni, P. Meredith, M. Y. Nam, M. Sun, M. Caccamo, L. Sha,: “Handling Mixed-criticality in SoC-based Real-time Embedded Systems”, Proceedings of the Seventh ACM International Conference on Embedded Software, 2009

[11] M. Zimmer, D. Broman, C. Shaver and E. A. Lee, "FlexPRET: A processor platform for mixed-criticality systems" 2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS), Berlin, 2014, pp. 101-110.

[12] M. Mollison, J. Erickson, J. Anderson, S. Baruah, J. Scoredos: “Mixed-criticality real-time scheduling for multicore systems,” in Proc. of the 10th IEEE International Conference on Computer and Information Technology (CIT), 2010, pp. 1864–1871.

[13] K. Goossens, A. Azevedo, K. Chandrasekar, M. D. Gomony, S. Goossens, M. Koedam, Y. Li, D. Mirzoyan, A. Molnos, A. B. Nejad, A. Nelson, S. Sinha: Virtual execution platforms for mixed-time-criticality systems: the CompSOC architecture and design flow. SIGBED Rev. 10, 3 (October 2013), 23-34.

[14] G. Heiser, B. Leslie: “The OKL4 microvisor: convergence point of microkernels and hypervisors”, In: Proceedings of the first ACM asia-pacific workshop on Workshop on systems (APSys '10). ACM, New York, NY, USA, 19-24

[15] M. Schoeberl, S. Abbaspour, B. Akesson, N. Audsley, R. Capasso, J. Garside, K. Goossens, S. Goossens, S. Hansen, R. Heckmann, S. Hepp, B. Huber, A. Jordan, E. Kasapaki, J. Knoop, Y. Li, D. Prokesch, W. Puffitsch, P. Puschner, A. Rocha, C. Silva, J. Sparsø, A. Tocchi: “T-CREST: Time-predictable multi-core architecture for embedded systems, Journal of Systems Architecture”, Volume 61, Issue 9, October 2015, Pages 449-471

[16] W, Weber, A. Hoess, J. van Deventer, F. Oppenheimer, R. Ernst, A. Kostrzewa, P. Dorè, T. Goubier, H. Isakovic, N. Druml, and others: “The EMC2 Project on Embedded Microcontrollers Technical Progress after Two Years”. Digital System Design (DSD), Euromicro Conference on. Pp. 524-531

5.

EMC² Examples

"Multi-core and many-core computing platforms have to significantly improve system (and application) integration, efficiency and performance"

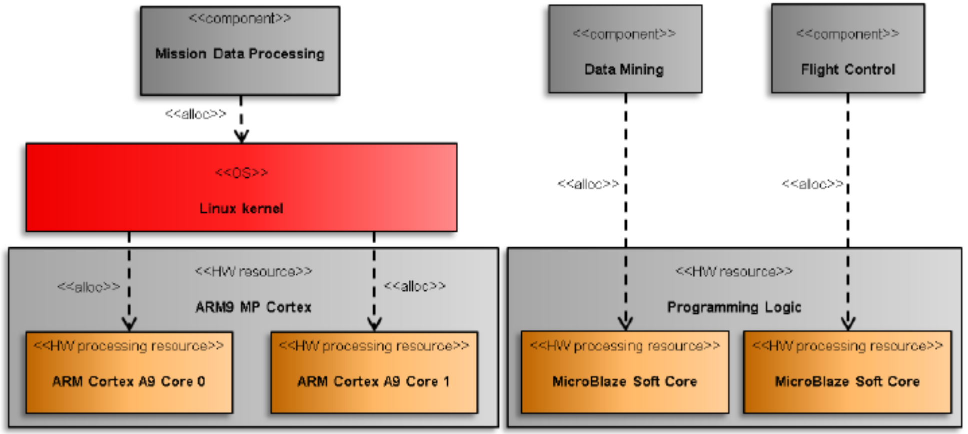
EMC² Examples

Separation Technique	HW	Single core	Multi-core
Spatial	0-level scheduling [10]	0-level scheduling [11]**[16]**	0-level scheduling [15] **[16]**
		1-level scheduling [2][5][10][13]**[16]**	1-level scheduling [4][9][15] **[16]**
		2-level scheduling [6][11]	2-level scheduling [3][4][6][7]**[8]**[9][14]
Temporal	0-level scheduling [10]	0-level scheduling [11]**[16]**	0-level scheduling [15][16]
		1-level scheduling [1][2][10][13]**[16]**	1-level scheduling [4][9][12][15][16]
		2-level scheduling [6][11]	2-level scheduling [1][4][6][7]**[8]**[9][14]

Multi-core Implementation

EMC² WP2 - *4-Copter Demonstrator* [16]

- Flight and Position control
  - Execution on Soft-Cores in FPGA
  - Bare metal, no OS support
  - Interfaces for I2C, PPM and GPIO used
- Object tracking
  - Execution on Dual ARM-Core
  - Needs Linux as OS
  - Multimedia Libraries
  - Needs interfaces USB und Network



Xilinx Zynq 7020:

- ARM dual-core Cortex-A9 (866MHz)
- Artix-7 FPGA (85k Logik Zellen)

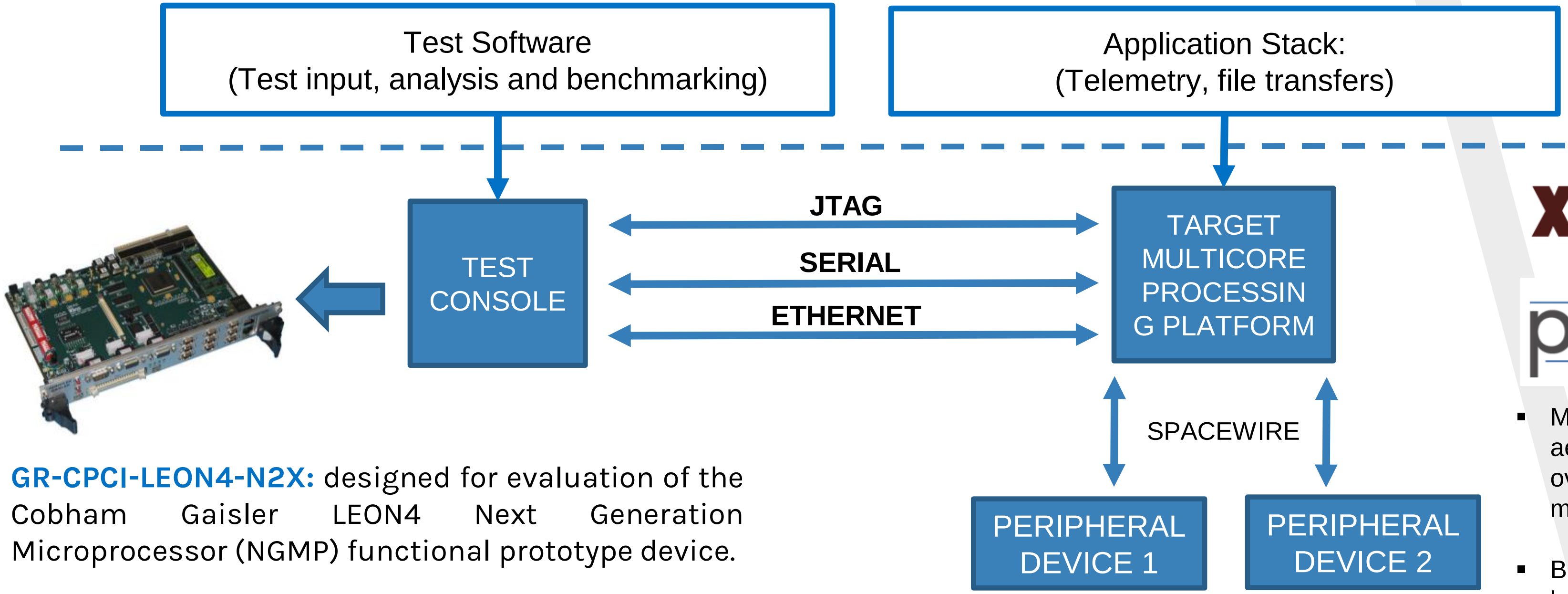
Safety critical tasks: All tasks which are needed for a stable and safety flight of the multi-rotor system, e.g. the flight and navigation controllers. An error, like missing a deadline, will cause a crash-landing!

Mission critical tasks: All tasks which are not needed for a safe flight, but may also have defined deadlines, e.g. tasks which are belonging to the payload processing, like video processing.

Uncritical tasks: All tasks which are not needed either for a safe flight or a correct execution of the mission task, e.g. control of the debug LEDs or transmission of telemetry data.

Multi-core Implementation

Univaq EMC² UC - Satellite Demo Platform (Hardware and Software) [8]



GR-CPCI-LEON4-N2X: designed for evaluation of the Cobham Gaisler LEON4 Next Generation Microprocessor (NGMP) functional prototype device.

Processor: Quad-Core 32-bit LEON4 SPARC V8 processor with MMU, IOMMU

- Migrate a typical aerospace application over a modern multicore platform
- Benchmarking hypervisors
- Compare different virtualization solutions

F. Federici, V. Muttillo, L. Pomante, G. Valente, D. Andreetti, D. Pascucci,: "Implementing mixed-critical applications on next generation multicore aerospace platforms", CPS Week 2016, EMC² Summit, Vienna, Austria

6.

Conclusion and Future Works

“Embedded systems are the key innovation driver to improve mechatronic products with cheaper and even new functionalities. They support today’s information society as inter-system communication enabler. Consequently, boundaries of application domains are alleviated and ad-hoc connections and interoperability play an increasing role”

Conclusions and Future Work

- This talk presents the MC/CPS domain, respect to implementation technologies and techniques

- During the work more than 200 papers related to the Mixed-Criticality Systems Implementation were analyzed (in this talk a subset of these papers has been presented)

- The work divide the whole set of the possible MC solution in different classes related to the architecture and to the specific technologies

- A survey related to this work will be submitted to journal and international conference in order to help designer in their design flow

- EMC^2 result will be used to improve this survey and to refine the implementation classes arising the whole ecosystem and works related to the MC scientific world

THANKS!

Any questions?

Backup Papers

MCS Classification

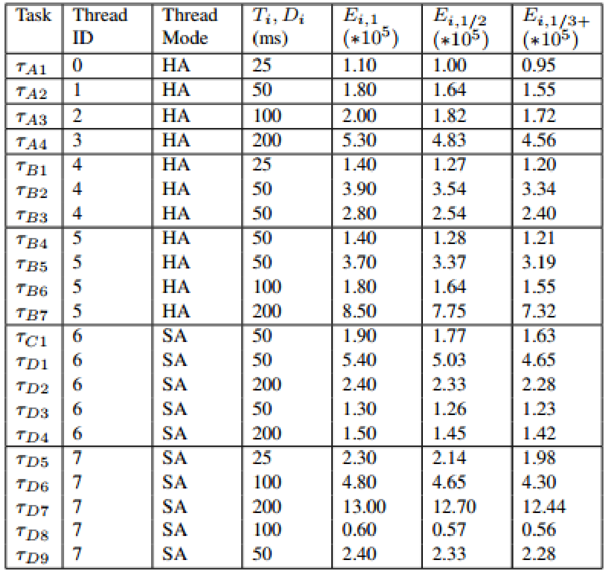
Separation Technique	HW	Single core	Multi-core
Spatial	0-level scheduling [10]	0-level scheduling [11][16]	0-level scheduling [15][16]
		1-level scheduling [2][5][10][13][16]	1-level scheduling [4][9][15][16]
		2-level scheduling [6][11]	2-level scheduling [3][4][6][7][8][9][14]
Temporal	0-level scheduling [10]	0-level scheduling [11][16]	0-level scheduling [15][16]
		1-level scheduling [1][2][10][13][16]	1-level scheduling [4][9][12][15][16]
		2-level scheduling [6][11]	2-level scheduling [1][4][6][7][8][9][14]

Hardware Implementation

FlexPRET: Processor Platform for Mixed-Criticality Systems [11]

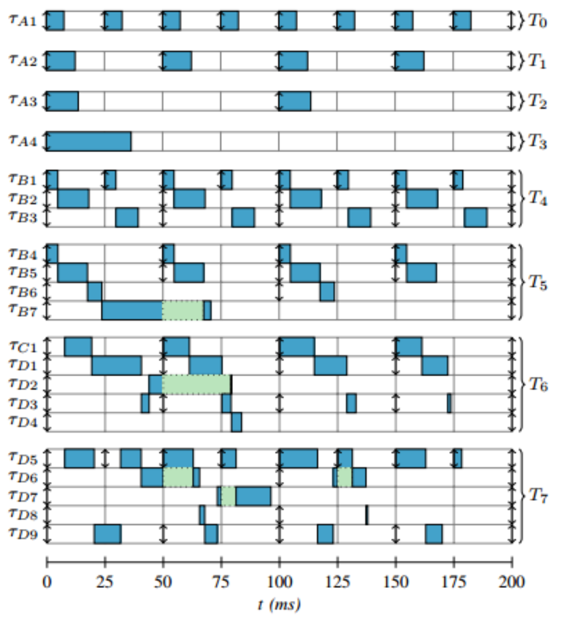
FlexPRET is a 32-bit, 5-stage, fine-grained multithreaded processor with software-controlled, flexible thread scheduling. It uses a classical RISC 5-stage pipeline: instruction fetch (F), decode (D), execute (E), memory access (M), and writeback (W). Predict not-taken branching and software-controlled local memories are used for fine-grained predictability. It also implements the RISC-V ISA [20], an ISA designed to support computer architecture research, that we extended to include timing instructions.

FlexPRET is implemented in Chisel [26], a hardware construction language that generates both Verilog code and a cycle-accurate C++-based simulator.



Task	Thread ID	Thread Mode	T_i, D_i (ms)	$E_{i,1}$ $(*10^5)$	$E_{i,1/2}$ $(*10^5)$	$E_{i,1/3+}$ $(*10^5)$
τ_{A1}	0	HA	25	1.10	1.00	0.95
τ_{A2}	1	HA	50	1.80	1.64	1.55
τ_{A3}	2	HA	100	2.00	1.82	1.72
τ_{A4}	3	HA	200	5.30	4.83	4.56
τ_{B1}	4	HA	25	1.40	1.27	1.20
τ_{B2}	4	HA	50	3.90	3.54	3.34
τ_{B3}	4	HA	50	2.80	2.54	2.40
τ_{B4}	5	HA	50	1.40	1.28	1.21
τ_{B5}	5	HA	50	3.70	3.37	3.19
τ_{B6}	5	HA	100	1.80	1.64	1.55
τ_{B7}	5	HA	200	8.50	7.75	7.32
τ_{C1}	6	SA	50	1.90	1.77	1.63
τ_{D1}	6	SA	50	5.40	5.03	4.65
τ_{D2}	6	SA	200	2.40	2.33	2.28
τ_{D3}	6	SA	50	1.30	1.26	1.23
τ_{D4}	6	SA	200	1.50	1.45	1.42
τ_{D5}	7	SA	25	2.30	2.14	1.98
τ_{D6}	7	SA	100	4.80	4.65	4.30
τ_{D7}	7	SA	200	13.00	12.70	12.44
τ_{D8}	7	SA	100	0.60	0.57	0.56
τ_{D9}	7	SA	50	2.40	2.33	2.28

A mixed-criticality avionics case study

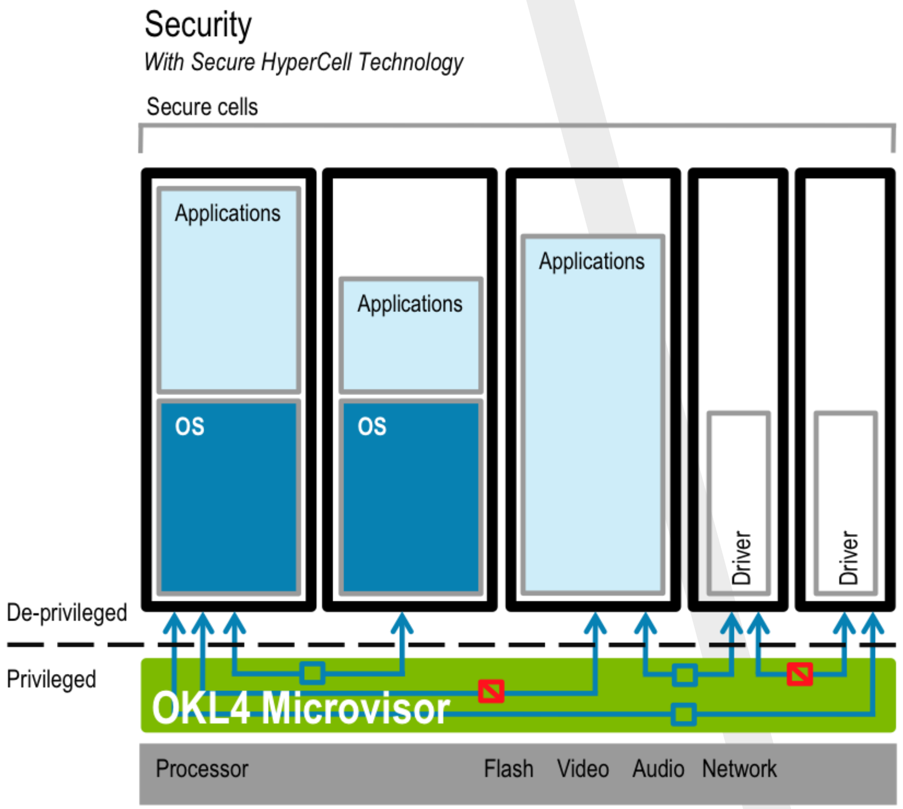


FlexPRET-8T (8 physical number of threads available) executing a mixed-criticality avionics case study

Single-core Implementation

OKL4 Microvisor [14]

- The **OKL4** Microvisor is an advanced secure type-1 hypervisor developed by General Dynamics C4 Systems and supports all ARM processors with MMU hardware
- Supporting virtualization with the lowest possible overhead, the microvisor's abstractions are designed with:
- the microvisor's execution abstraction is that of a virtual machine with one or more virtual CPUs (vCPUs), on which the guest OS can schedule activities;
- the memory abstraction is that of a virtual MMU (vMMU), which the guest OS uses to map virtual to (guest) physical memory;
- the I/O abstraction consists of memory-mapped virtual device registers and virtual interrupts (vIRQs);
- communication is abstracted as vIRQs (for synchronisation) and channels. The latter are bi-directional FIFOs with a fixed (configurable per channel) buffer allocated in user space (run also TCP/IP on a channel).



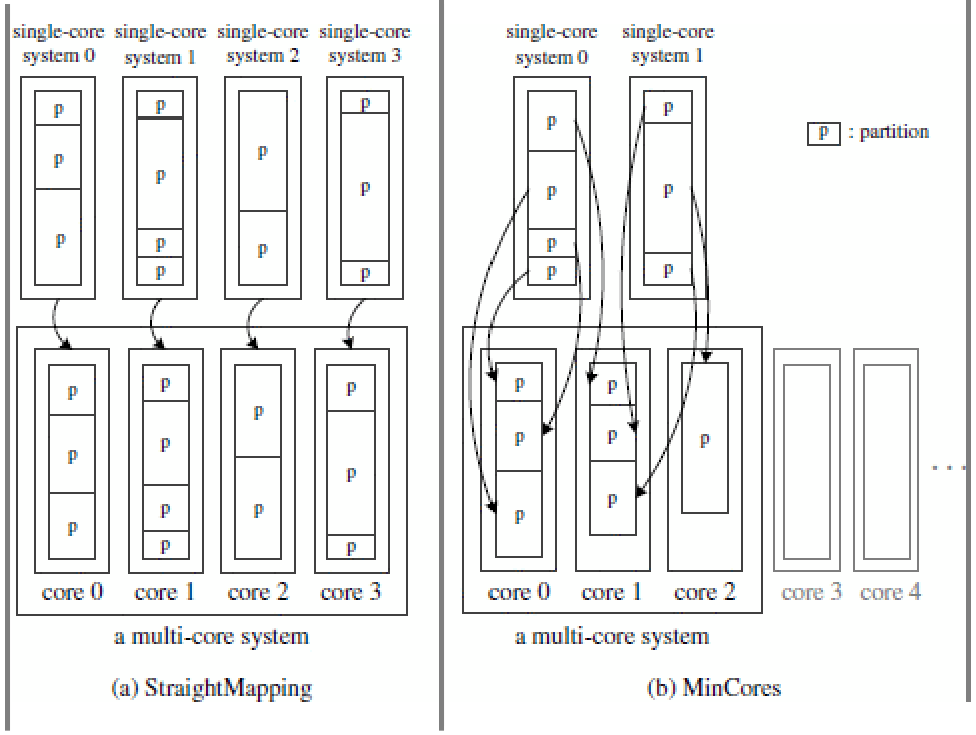
Multi-core Implementation

Multi-IMA Partitioning [6]

Given a set of:

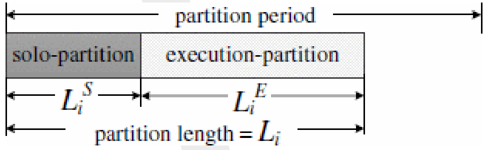
- Single-core IMA systems
- Partitions
- Multi-core system

All partitions from a single core must be scheduled on the same core of the multi-core system (they can be rescheduled within the same core)



Multi-IMA partition scheduling optimization where a partition consists of two logical regions:

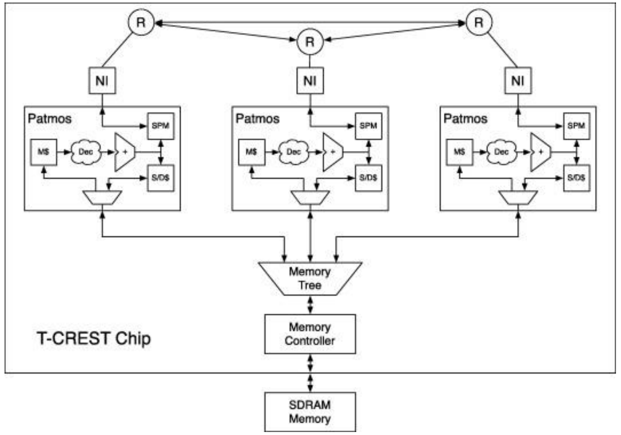
- solo- partition (in avionics systems performing I/O transactions)
- execution-partition

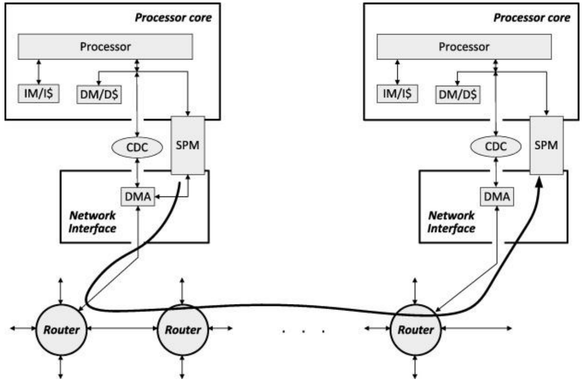


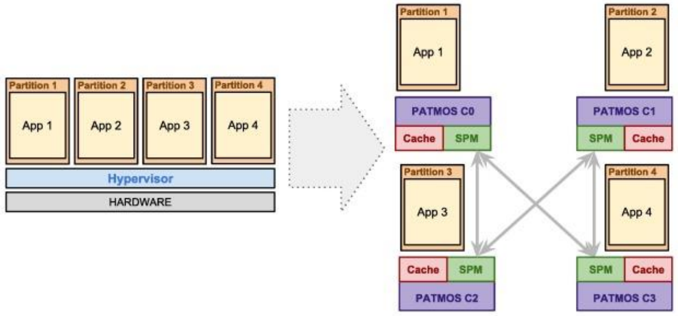
Partition can be scheduled on a core if it doesn't interfere with the solo-partitions of other partitions and doesn't overlap with other partitions assigned to the same core. Each partition is strictly periodic and non-preemptive. This supports **temporal** and **spatial** isolation among partitions.

Many-core Implementation

T-CREST Multi-core Architecture [15]





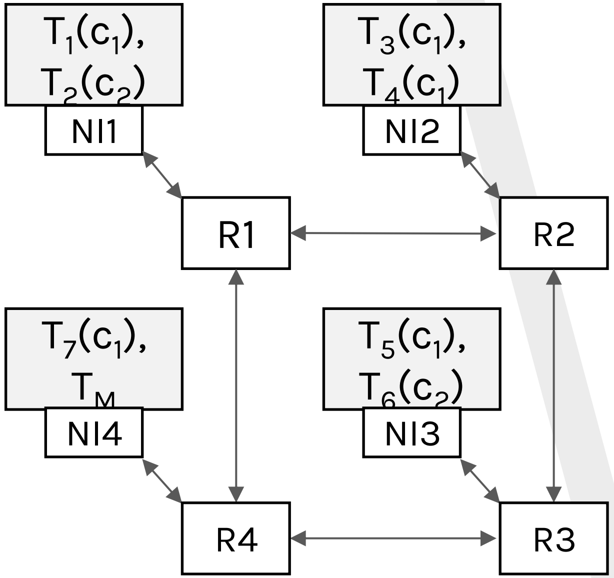


- The ***T-CREST*** platform consisting of Patmos processor nodes that are connected via an on-chip network for message passing communication and a memory tree to a memory controller for shared memory access
- Data transfer in the **T-CREST** core-to-core message passing NoC.
- Mapping of **ARINC 653** partitions to cores onto the **T-CREST** platform.

Many-core Implementation

University of L'Aquila CRAFTERS Case Study

- **Hardware mechanisms to support isolation in a Network-on-Chip**
  - Isolation of different application classes on NoC architectures
  - Hardware mechanisms supporting isolation to be introduced into existing network interfaces
  - Support for the execution of multiple applications with different criticality levels
  - Strategy: message exchange supervision



L. Pomante, C. Tieri, F. Federici, M. Colizza, M. Faccio, R. Cardinali, B. Iorio. "HW Mechanisms to Support Isolation in Mixed-Criticality NoC". Euromicro Conference on Digital System Design - WIP Session, Verona, August 2014.