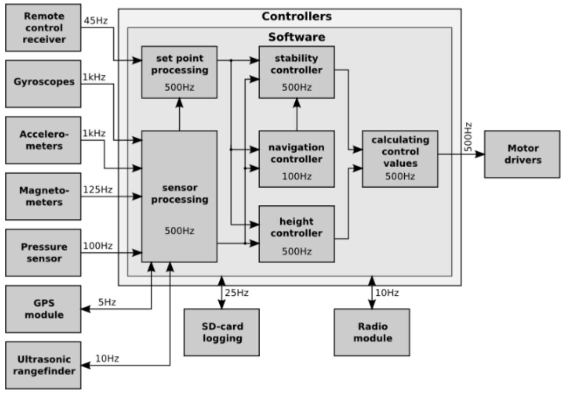
Architectures, Methods, and Tools for Mixed Criticality Applications on Multicores

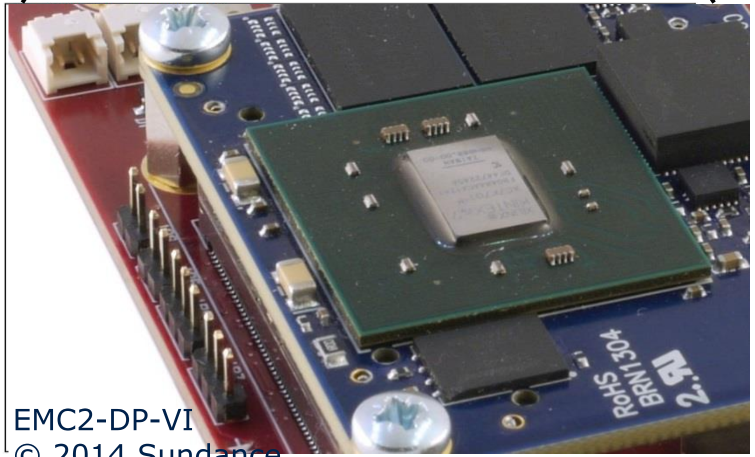
Frank Oppenheimer, OFFIS
Sascha Uhrig TU Dortmund, Germany

Overview

What is mixed-critical?

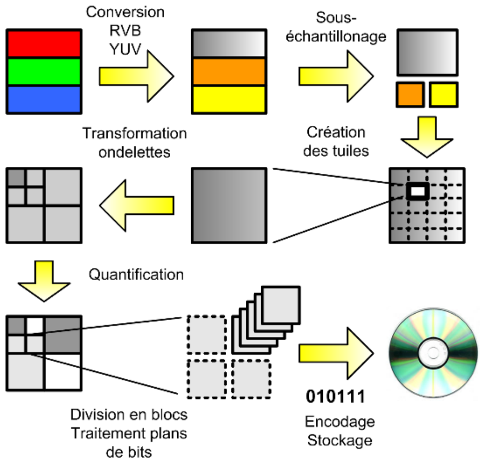
Safety





EMC2-DP-VI
© 2014 Sundance

Performance

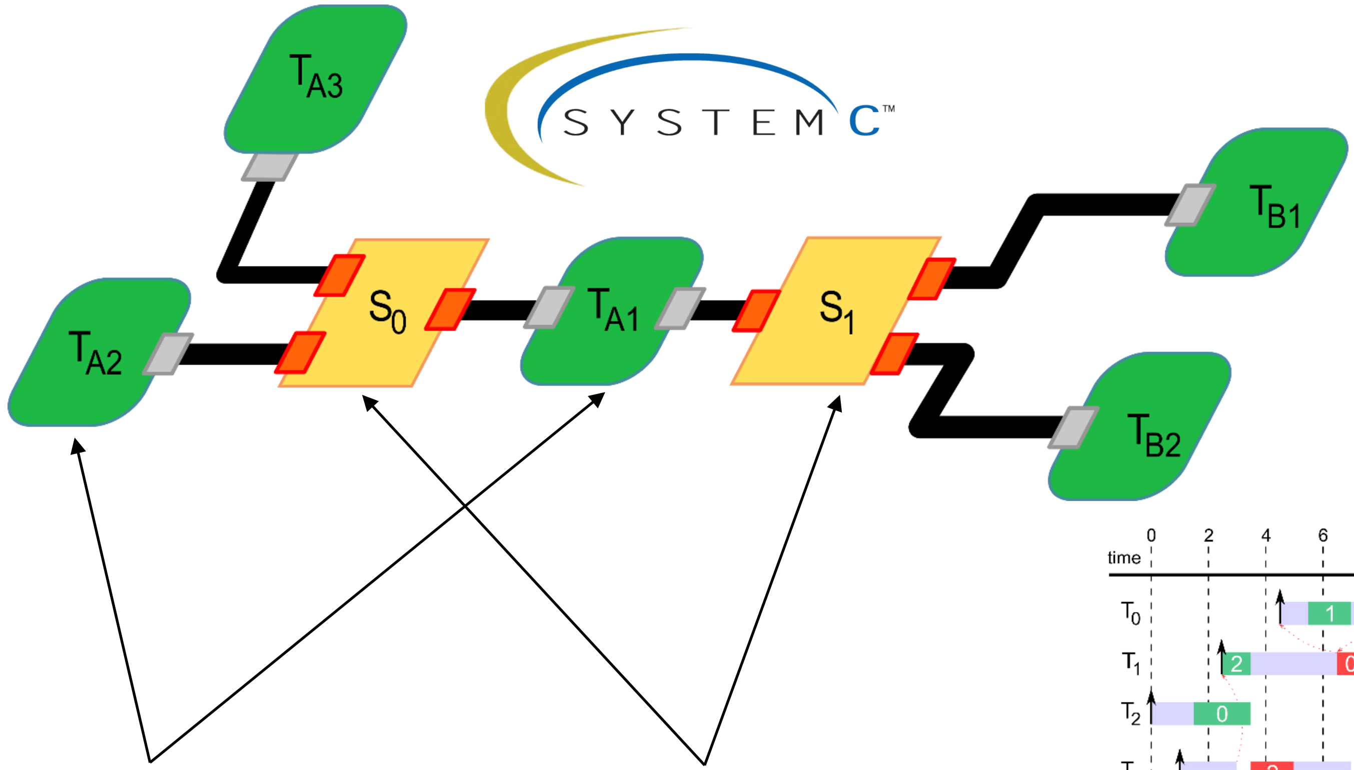


Modelling the application

- **Designer entry level with modelling primitives for:**
  - Tasks
  - Scheduling
  - Criticality
  - Communication / Synchronisation

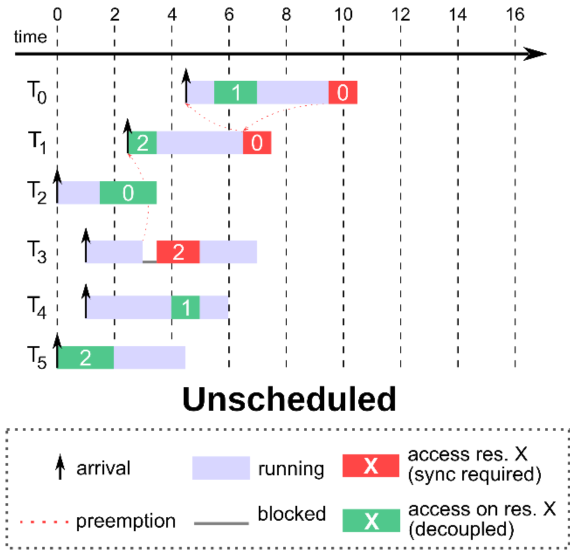
- **Executable Model to observe**
  - Functional behaviour
  - Segregation (functional)
  - Timing (requirements and execution times)

Basic Modelling elements

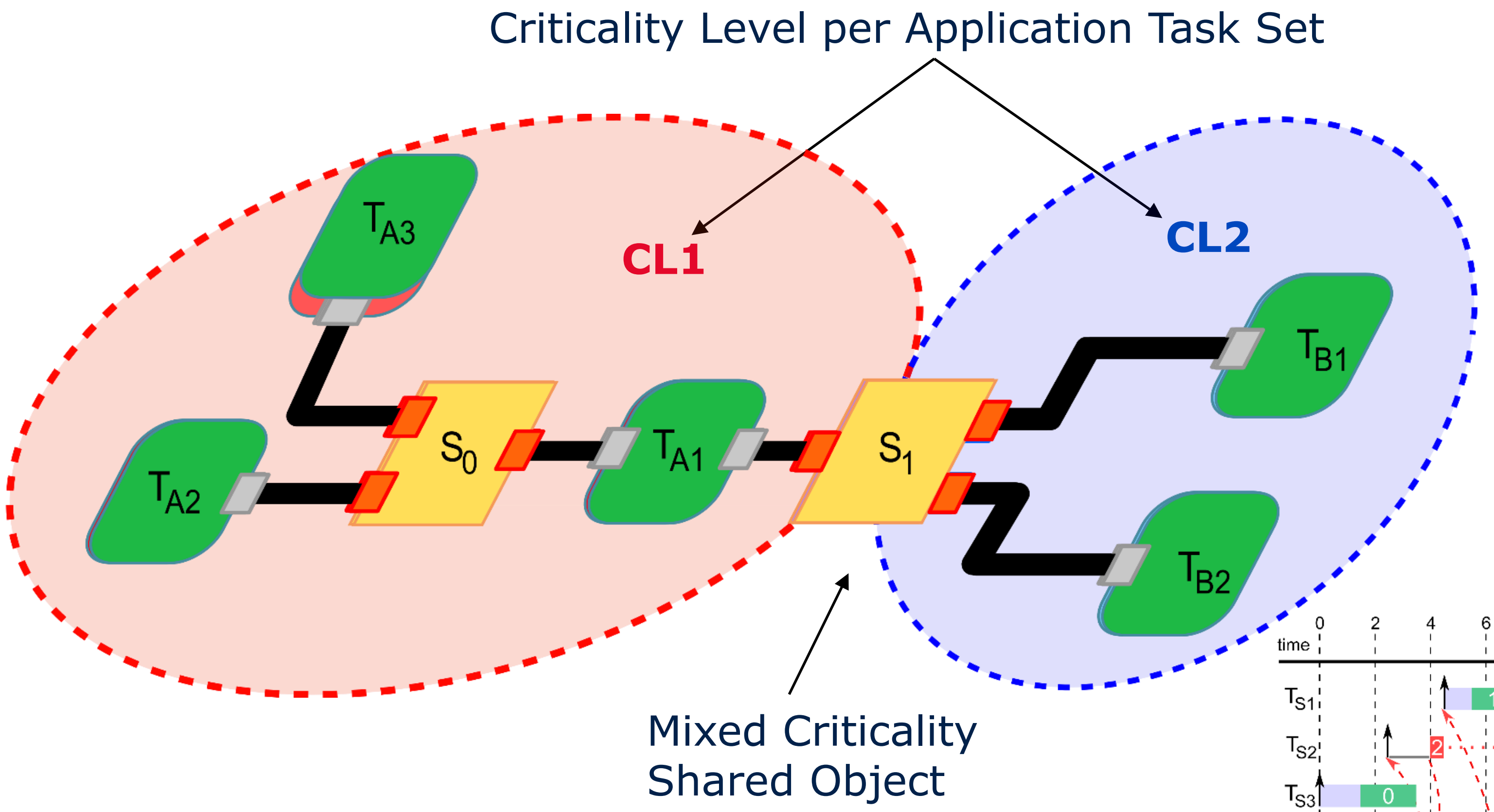


Tasks
- ☐ Behaviour
- ☐ Priority
- ☐ Deadline & Period
- ☐ Execution time
- ☐ Ports to Shared Objects

Shared Objects
- ☐ Access Policy
- ☐ Method interface
- ☐ Shared data structure
- ☐ Mutual exclusion

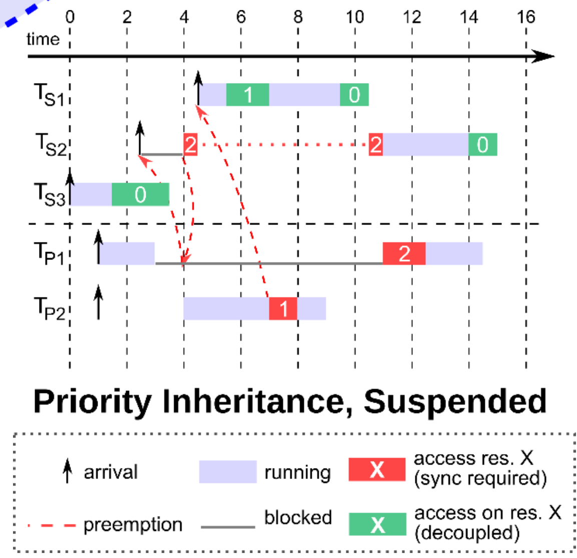


Adding Mixed Criticality


Criticality Level per Application Task Set
CL1
CL2
T_{A3}
T_{A2}
S_0
T_{A1}
S_1
T_{B1}
T_{B2}
Mixed Criticality Shared Object

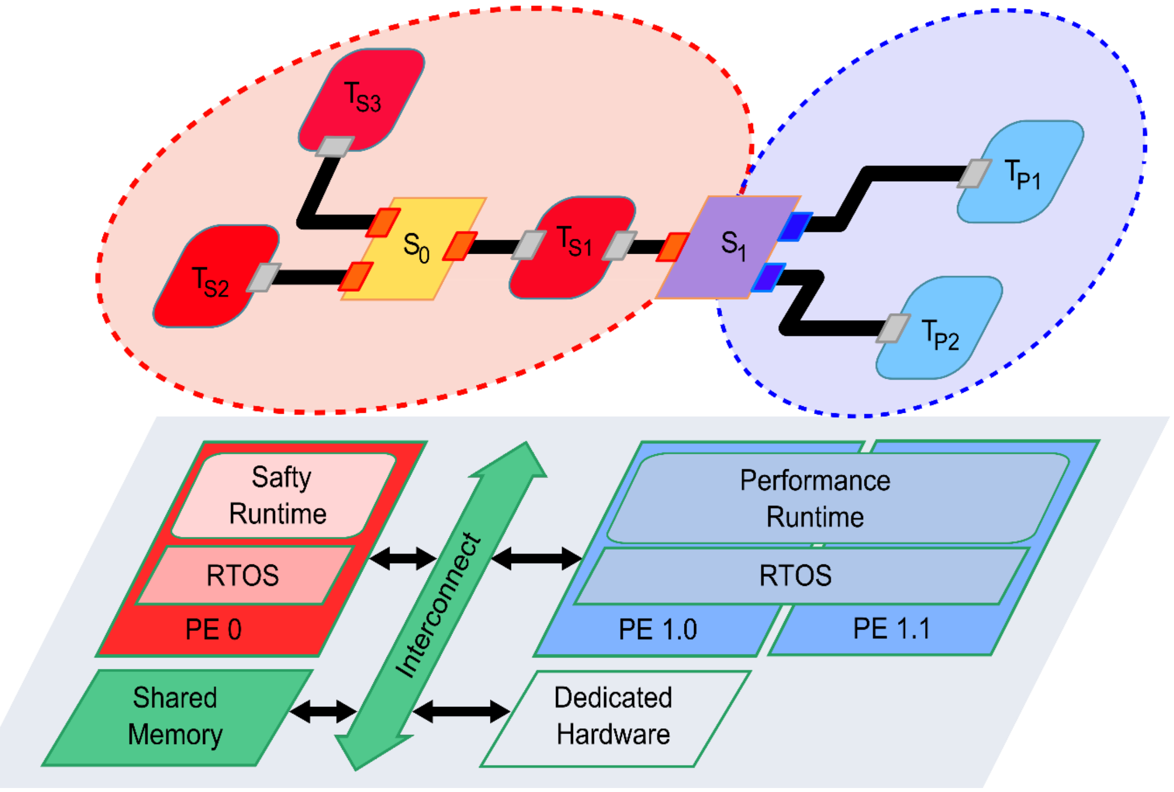
Functional segregation

- Scheduler per Task Set
- Task execution order (incl. Blocking)
- Access order



Priority Inheritance, Suspended

Shared resource contention effects

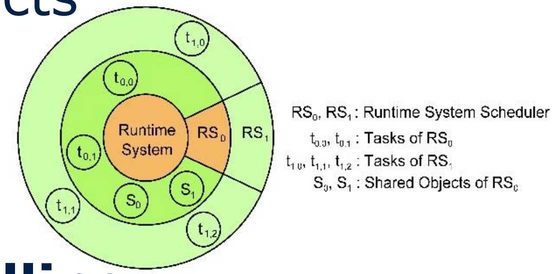


Shared resources like memory or busses require

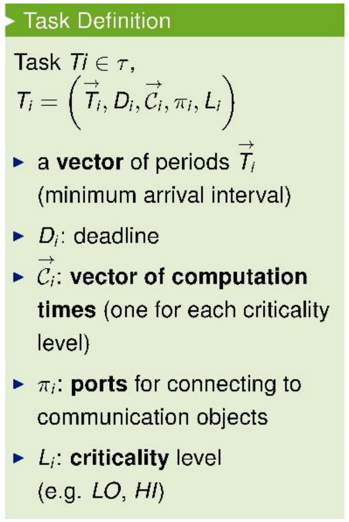
- Functional segregation
- Extra-functional segregation

EMC2 WP2/WP4 intended results

- **Executable Models of the Application**
  - Tasks
  - (Shared) Objects
  - Schedulers



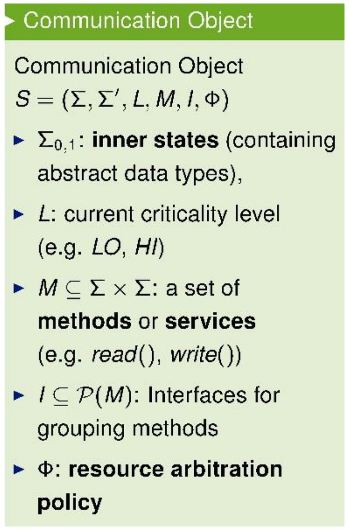
Runtime System
RS$_0$
RS$_1$
$t_{0,0}$
$t_{0,1}$
$t_{1,0}$
$t_{1,1}$
$t_{1,2}$
S_0
S_1
RS$_0$, RS$_1$: Runtime System Scheduler
$t_{0,0}$, $t_{0,1}$: Tasks of RS$_0$
$t_{1,0}$, $t_{1,1}$, $t_{1,2}$: Tasks of RS$_1$
S_0, S_1 : Shared Objects of RS$_0$



Task Definition

Task $Ti \in \tau$,
$T_i = \left(\vec{T_i}, D_i, \vec{C_i}, \pi_i, L_i\right)$

- a **vector** of periods $\vec{T_i}$ (minimum arrival interval)
- D_i: deadline
- $\vec{C_i}$: **vector of computation times** (one for each criticality level)
- π_i: **ports** for connecting to communication objects
- L_i: **criticality** level (e.g. *LO*, *HI*)

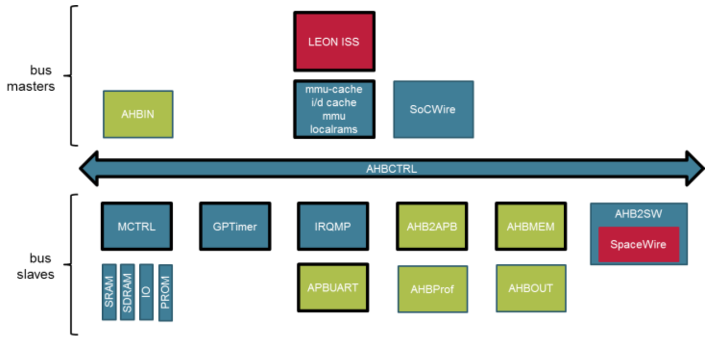


Communication Object

Communication Object
$S = (\Sigma, \Sigma', L, M, I, \Phi)$

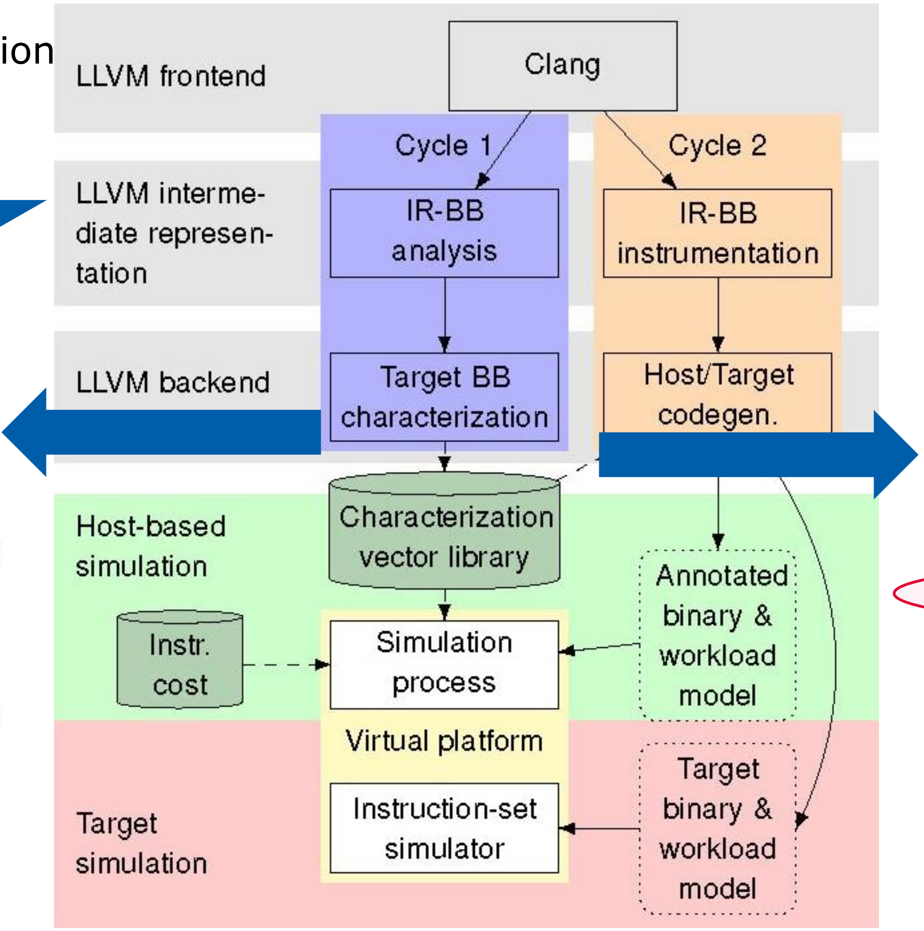
- $\Sigma_{0,1}$: **inner states** (containing abstract data types),
- L: current criticality level (e.g. *LO*, *HI*)
- $M \subseteq \Sigma \times \Sigma$: a set of **methods** or **services** (e.g. *read*(), *write*())
- $I \subseteq \mathcal{P}(M)$: Interfaces for grouping methods
- Φ: **resource arbitration policy**

- **Platform Modelling**
  - Timing analysis tools
  - Automated Back-Annotation of platform specific Behaviour
  - Predictable Cache coherency
  - Platform Models for SoCRocket,LEON ZYNQ, AURIX



Automated Back-Annotation of Platform specific Behaviour

Characterisation vector



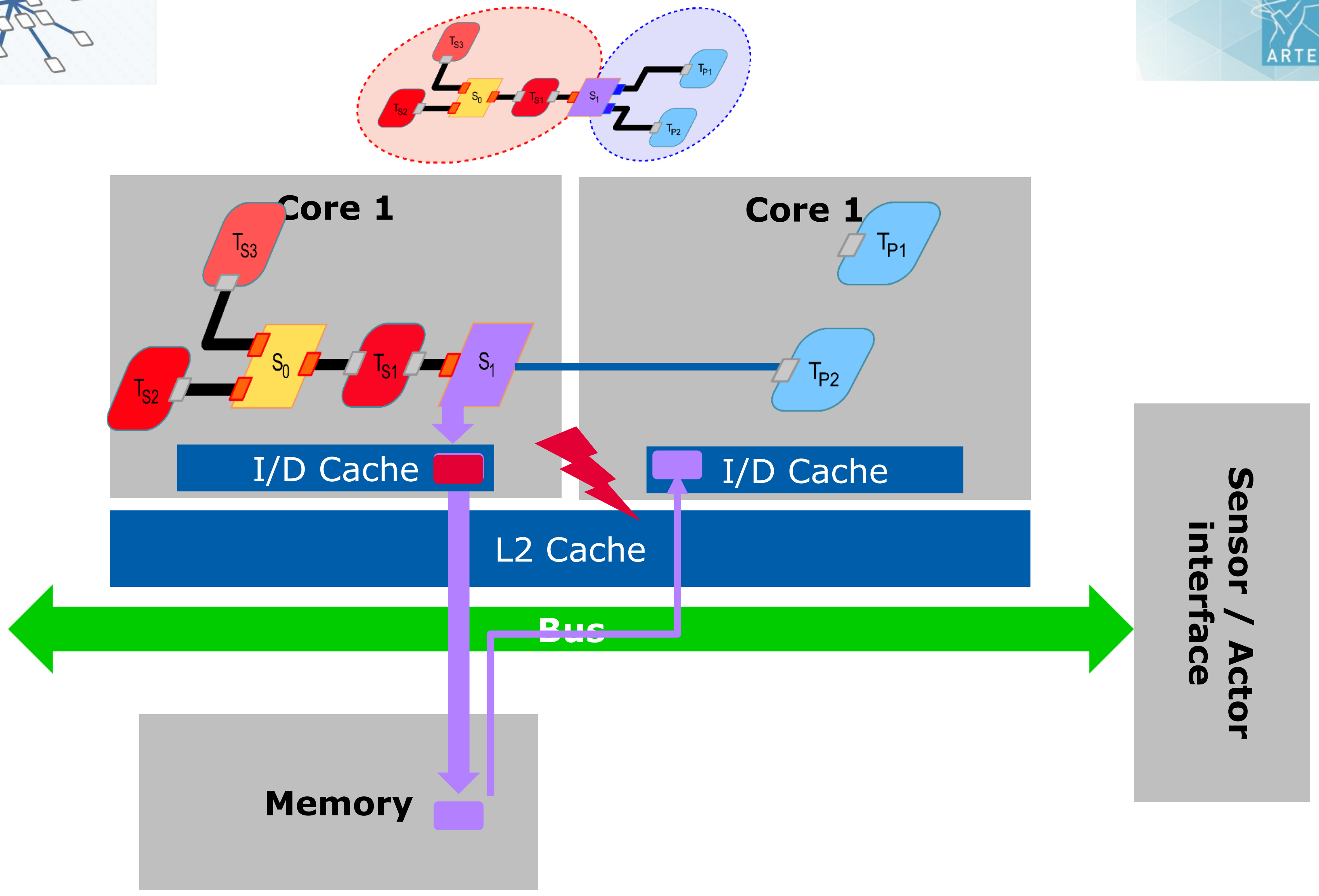
ARM Target Code

	V_0
`ldr r0, [pc + 36]`	$(1, R)$
`ldr r0, [r0]`	$(1, R)$
`ldr r1, [r0 + 4]`	$(1, R)$
`ldr r2, [r0 + 8]`	$(1, R)$
`add r1, r1, r2`	$(1, N)$
`str r1, [r0]`	$(1, W)$
`ldr r1, [r0 + 16]`	$(1, R)$
`ldr r2, [r0 + 20]`	$(1, R)$
`add r1, r1, r2`	$(1, N)$
`str r1, [r0 + 12]`	$(1, W)$
`bx lr`	$(1, N)$

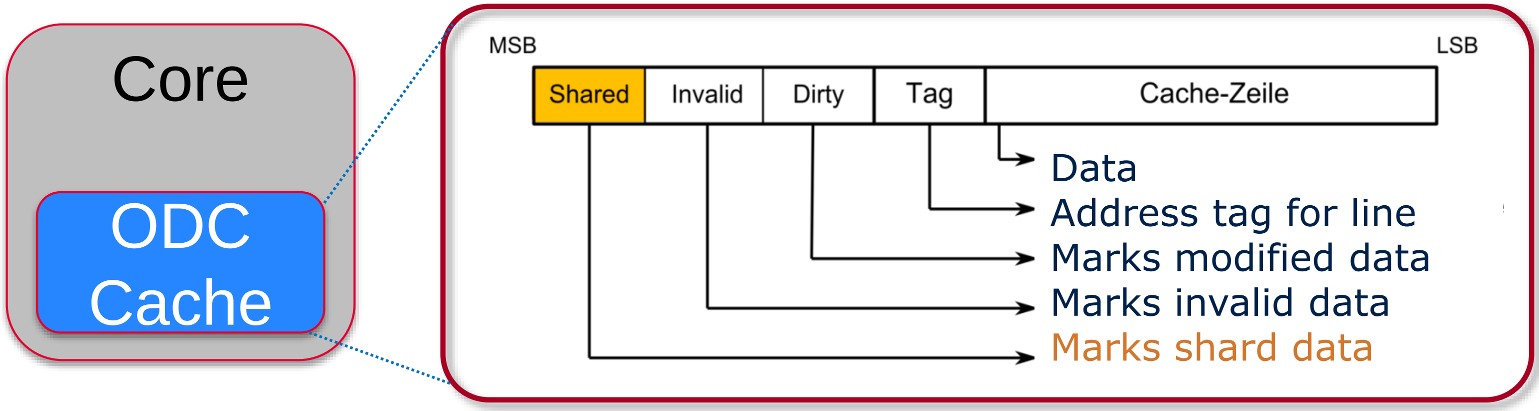
Native code and model update

```
mov     $0x0,%eax # BB id 0
mov     0x4,%ecx
# [...]
add     0x14,%ecx
mov     %ecx,0xc
movl    $0x0,(%esp)
mov     %eax,-0x4(%ebp)
call    3a # call __bb_sync(V0)
add     $0x8,%esp
pop     %ebp
ret
```

Functional interference via platform

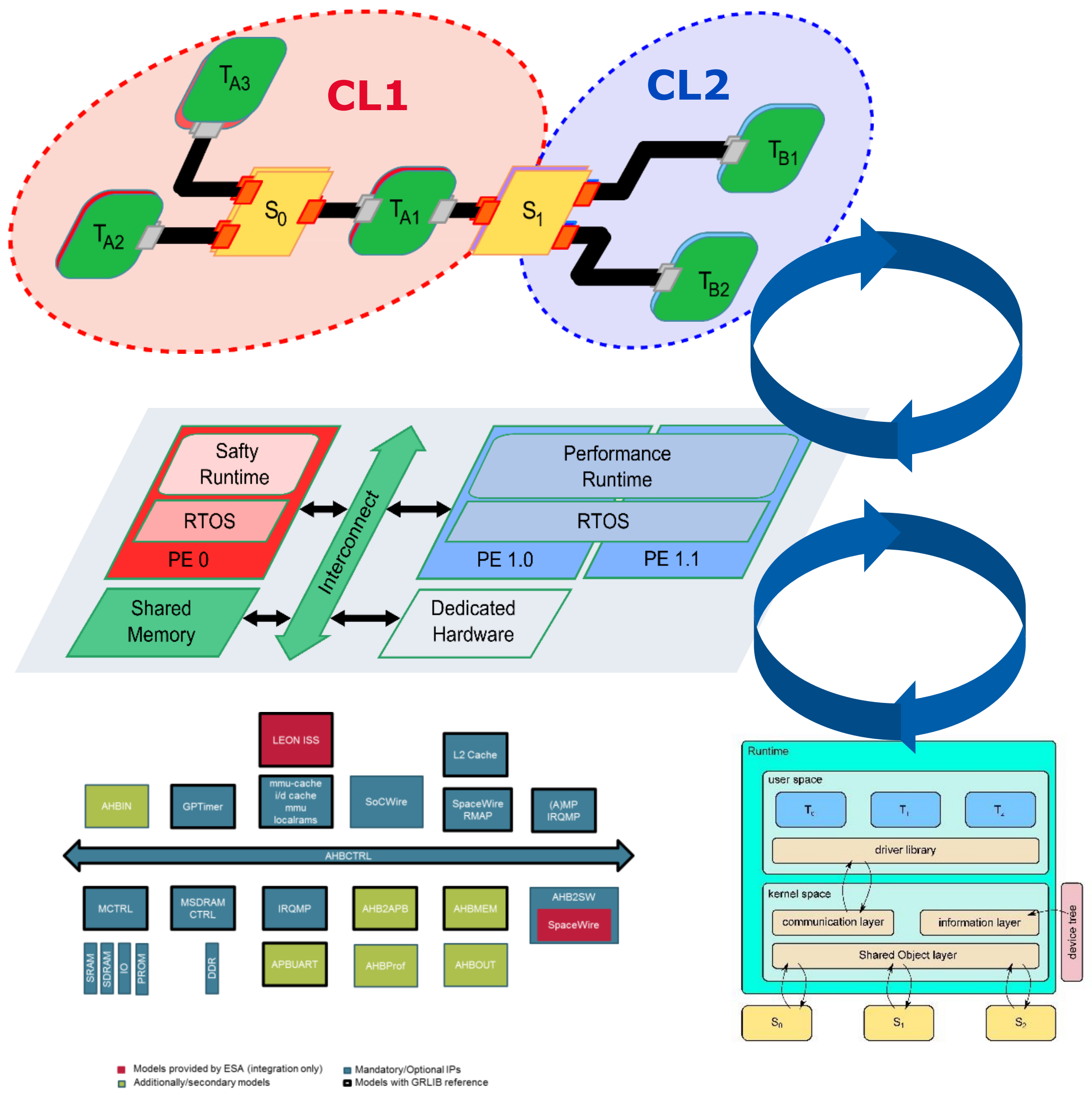


On-Demand Coherent Cache (ODC²)



Hardware	Software
• Additional Bit per Cache-Line • Extra logic for address snooping • Write strategy for shared data.	• Switch between private and shared mode • Shared data leads to un-cached write access

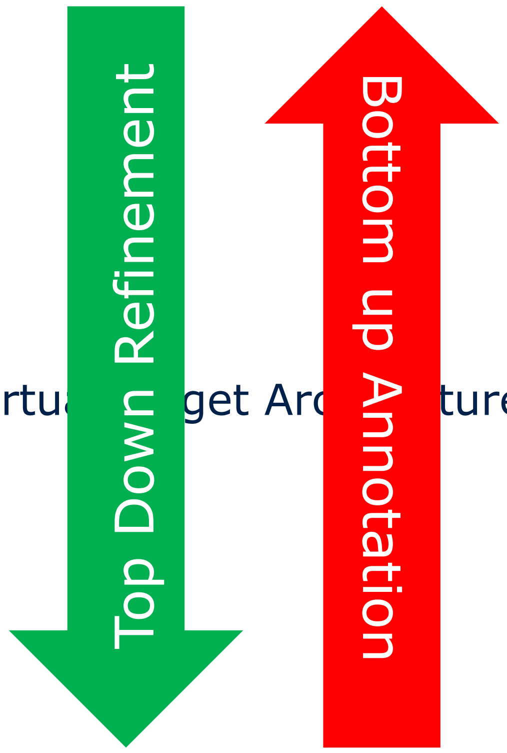
Application to Platform flow



Application Layer

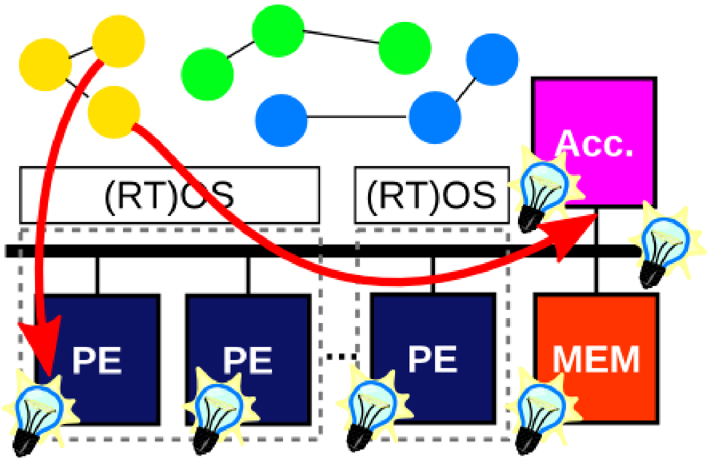
Virtual Target Architecture

Platform simulation level



Summary & Outlook

- Early **tool set for modelling** of application and technology platform
- **Simulation and analysis** on different levels
- **Technology support** (Cores, Caches, runtime support)
- **Platforms**: ZYNQ (ARM), LEON, NoC, AURIX, SoCRocket, …



- WP2 internal Mixed-Critical Quadrocopter use-case – as early executable test vehicle

- Several industrial use-cases in EMC2 for further evaluation