Scheduling analysis of multiprocessor architectures with AADL & Cheddar

S. Rubini+, L. Lemarchand+, N. Tran+, M. Dridi+, V. Nicolas+, J. Boukhobza+, F. Singhoff (speaker)+, P. Dissaux*, J. Legrand*

* Ellidiss Technologies
+ Lab-STICC UMR CNRS 6285/UBO

Introduction

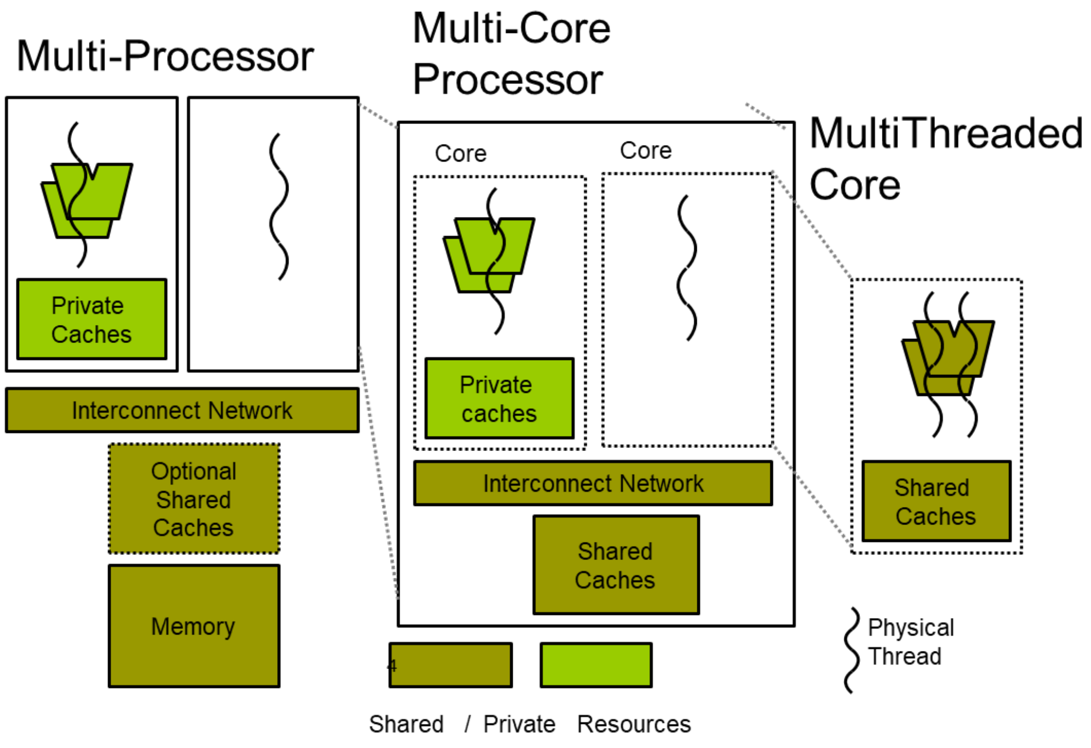
- **Schedulability:**
  - Simplified task models (e.g. Liu & Layland periodic task).
  - Verification tools (.e.g simulation over feasibility interval) to check if deadlines can be met.
- Schedulability methods for multi-core/many-core since 2012. Implemented into Cheddar. AADLInspector limited to uniprocessor right now.
- AADL V3 & prototyping activities around Cheddar OSATE plug-in since nov. 2018.
- How to do it with AADL V3? What is missing?

Agenda

1. Introduction
2. Multiprocessor scheduling analysis
3. Cheddar features and modeling requirements
4. With AADL

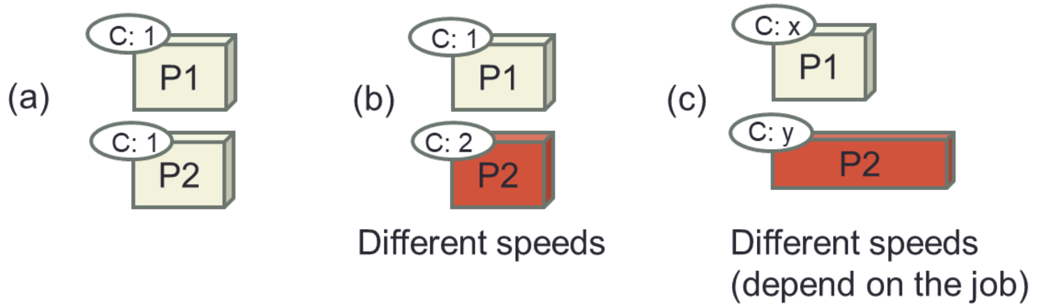
Various Multi-processing architecture designs

- Processors, cores or physical threads may be seen as processing resources by the scheduler.
- Hardware resources shared or private
- Asymmetric/Symmetric Multiprocessing (AMP/SMP)



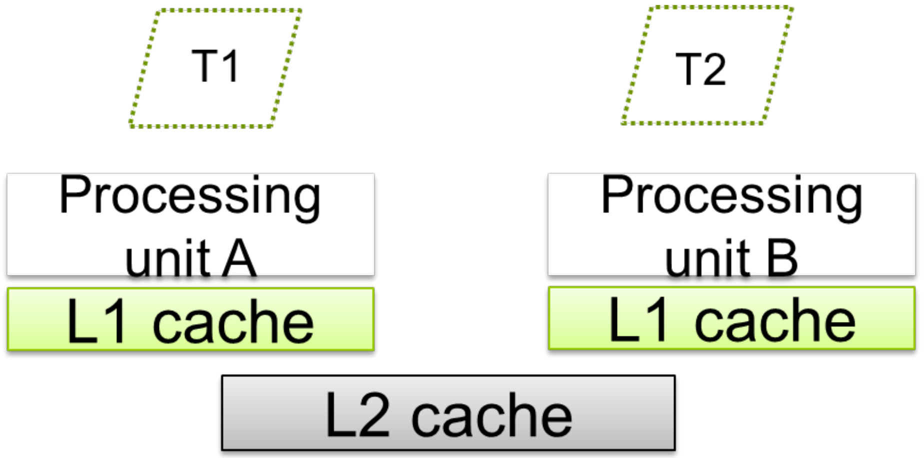
Various Multi-processing architecture designs

- Various kinds of « processing units », that have to be seen by scheduler
    - Identical (a) , uniform heterogeneous (b) , or unrelated heterogeneous (c) processing units



- Various memory units : cache, scratchpad, DDR3
- Various interconnect technologies between processing units and/or memory units : bus, Network on Chip

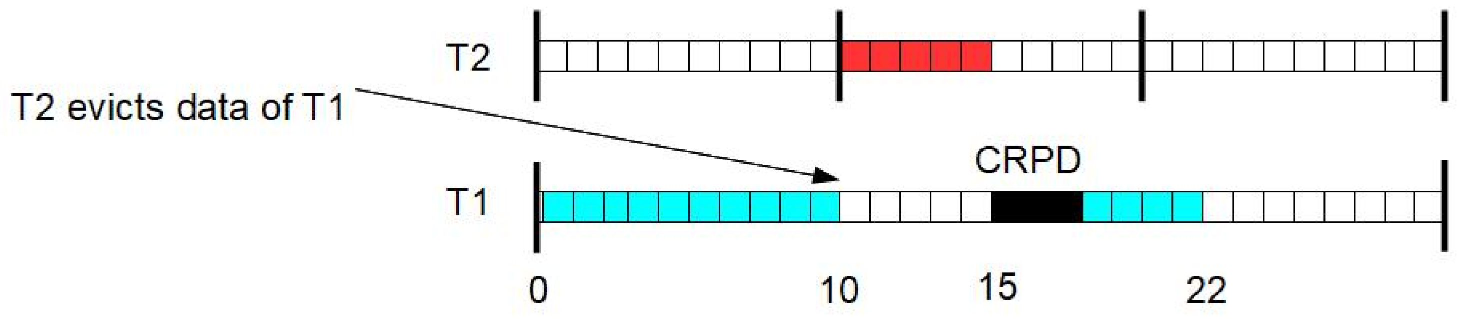
Shared resources and “hardware” interferences



- Schedulability = mastering interferences (software … and hardware)

- T1 and T2 are functionally independent but are not actually independent due to the hardware resources.
- Interferences due to hardware shared resources.

Shared resources and “hardware” interferences



- <span>**Cache related preemption delay (CRPD):**</span> **the additional time to refill the cache with memory blocks evicted by preemption.**

- **CRPD is a high preemption cost, up to 44% of the WCET of task** (Pellizzoni 2007).
- **Variability of CRPD makes schedulability difficult**

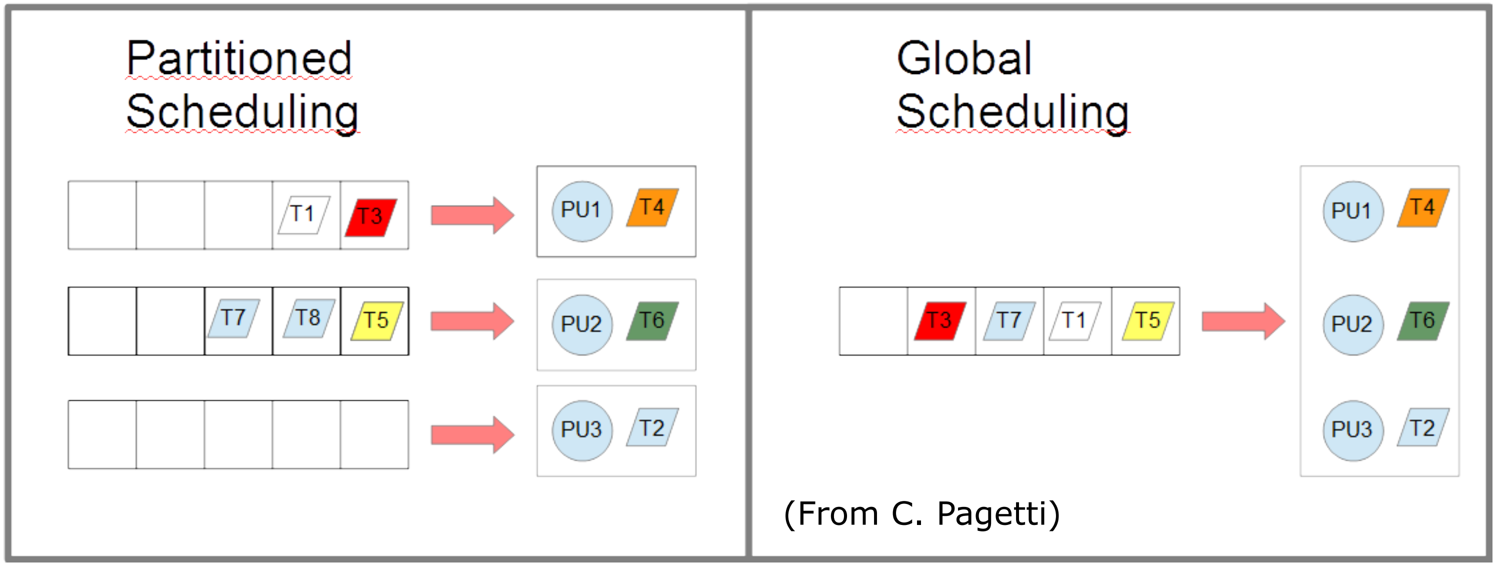
Multiprocessor schedulability analysis is difficult

- **Be humble:** difficult, not the main concern
- **Different/many schedulability results**
  - Ex1: sustainability and scheduling anomalies, i.e. schedulable task set on the worst case can be unschedulable in better scenarios.
- **Different/many scheduling algorithms**
  - Ex2: specific policies such as Proportional Fair (Anderson 2000), EDZL (Cho 2002), LLREF (Cho 2006), RUN (Regnier 2011)
- **Require an (abstracted) hardware model including shared resources** => interferences!

Agenda

1. Introduction
2. Multiprocessor scheduling analysis
3. Cheddar features and modeling requirements
4. With AADL

Partitioned vs global/shared management, ex processing units

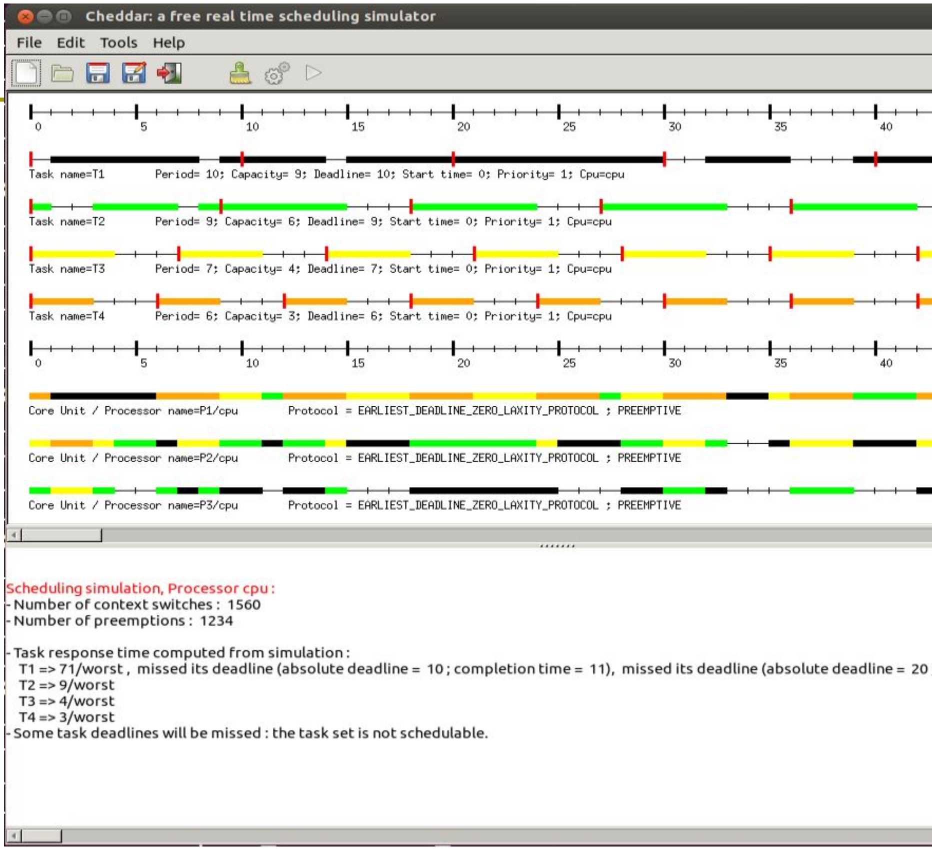


- **Partitioned scheduling:** first assign off-line each task on a processing unit ; each processing unit schedules its own task set.

- **Global scheduling:** choose the next task to run on any available processing unit (or preempt if all busy).

Multiprocessor scheduling analysis features in Ellidiss/Lab-STICC tools

- ❑ **AADLInspector**
  - ❑ Partitioned scheduling only and uniprocessor execution plaform (i.e. no shared resource)
  - ❑ Classical policies (fixed priority, EDF, including ARINC 653, …)
  - ❑ Partitioning policies: Best fit, First Fit, Next Fit, GT, SF



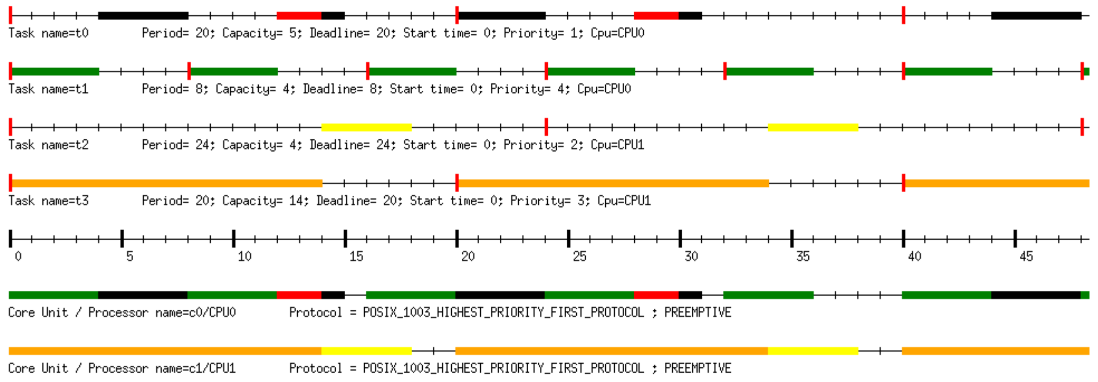
```
Task name=T1        Period= 10; Capacity= 9; Deadline= 10; Start time= 0; Priority= 1; Cpu=cpu
Task name=T2        Period= 9; Capacity= 6; Deadline= 9; Start time= 0; Priority= 1; Cpu=cpu
Task name=T3        Period= 7; Capacity= 4; Deadline= 7; Start time= 0; Priority= 1; Cpu=cpu
Task name=T4        Period= 6; Capacity= 3; Deadline= 6; Start time= 0; Priority= 1; Cpu=cpu
Core Unit / Processor name=P1/cpu        Protocol = EARLIEST_DEADLINE_ZERO_LAXITY_PROTOCOL ; PREEMPTIVE
Core Unit / Processor name=P2/cpu        Protocol = EARLIEST_DEADLINE_ZERO_LAXITY_PROTOCOL ; PREEMPTIVE
Core Unit / Processor name=P3/cpu        Protocol = EARLIEST_DEADLINE_ZERO_LAXITY_PROTOCOL ; PREEMPTIVE
```

```
Scheduling simulation, Processor cpu :
- Number of context switches :  1560
- Number of preemptions :  1234

- Task response time computed from simulation :
  T1 => 71/worst ,  missed its deadline (absolute deadline =  10 ; completion time =  11),  missed its deadline (absolute deadline =  20
  T2 => 9/worst
  T3 => 4/worst
  T4 => 3/worst
- Some task deadlines will be missed : the task set is not schedulable.
```

- ❑ **Cheddar 3.x only (not in AI and OSATE yet)**
  - ❑ Global scheduling: any uniprocessor policies + specific policies such as EDZL, LLREF, Pfair, DAG/list scheduling
  - ❑ Partitioning heuristics based on PAES (Pareto Archived Evolution Strategy)
  - ❑ Hardware shared resources support: cache, NoC, memory bus

Cache-Aware Schedulability analysis



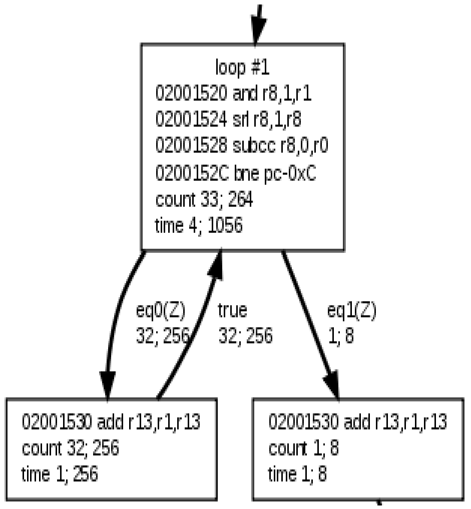
```
Task name=t0        Period= 20; Capacity= 5; Deadline= 20; Start time= 0; Priority= 1; Cpu=CPU0
Task name=t1        Period= 8; Capacity= 4; Deadline= 8; Start time= 0; Priority= 4; Cpu=CPU0
Task name=t2        Period= 24; Capacity= 4; Deadline= 24; Start time= 0; Priority= 2; Cpu=CPU1
Task name=t3        Period= 20; Capacity= 14; Deadline= 20; Start time= 0; Priority= 3; Cpu=CPU1
Core Unit / Processor name=c0/CPU0        Protocol = POSIX_1003_HIGHEST_PRIORITY_FIRST_PROTOCOL ; PREEMPTIVE
Core Unit / Processor name=c1/CPU1        Protocol = POSIX_1003_HIGHEST_PRIORITY_FIRST_PROTOCOL ; PREEMPTIVE
```

- Extend Audsley's priority assignment algorithm (Audsley, 1995) to take into account CRPD
- **Various means to compute CRPD:** UCB, ECB, UCB-Union, UCB-union-Multiset, ECB-Union-Multiset, combined Multiset,
- **First step to scheduling simulation with cache**
  - Very limited: L1 uniprocessor instruction caches only
  - But sustainable CPRD model (prooved): `load block time ≤ task execution time`
  - And known feasibility interval (prooved): `[0, LCM(Pi)]`

Cache-Aware schedulability

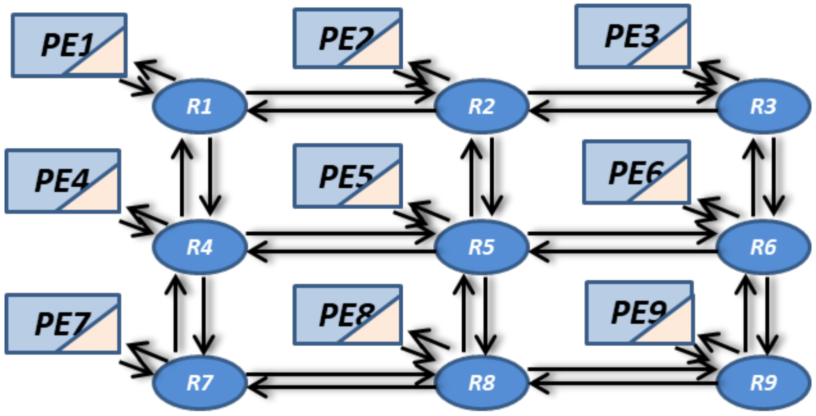
Scheduling analysis for systems with cache

Cache Configuration	**Control Flow Graph**	**Worst-Case Execution Time**
Memory Layout	**Cache Access Profile**	**Scheduling Policy**


```
loop #1
02001520 and r8,1,r1
02001524 srl r8,1,r8
02001528 subcc r8,0,r0
0200152C bne pc-0xC
count 33; 264
time 4; 1056
```
```
eq0(Z)
32; 256
```
```
true
32; 256
```
```
eq1(Z)
1; 8
```
```
02001530 add r13,r1,r13
count 32; 256
time 1; 256
```
```
02001530 add r13,r1,r13
count 1; 8
time 1; 8
```

- ❑ **Require to model**: cache unit, cache access profile and memory layout, task *Control flow graph* **(**CFG)

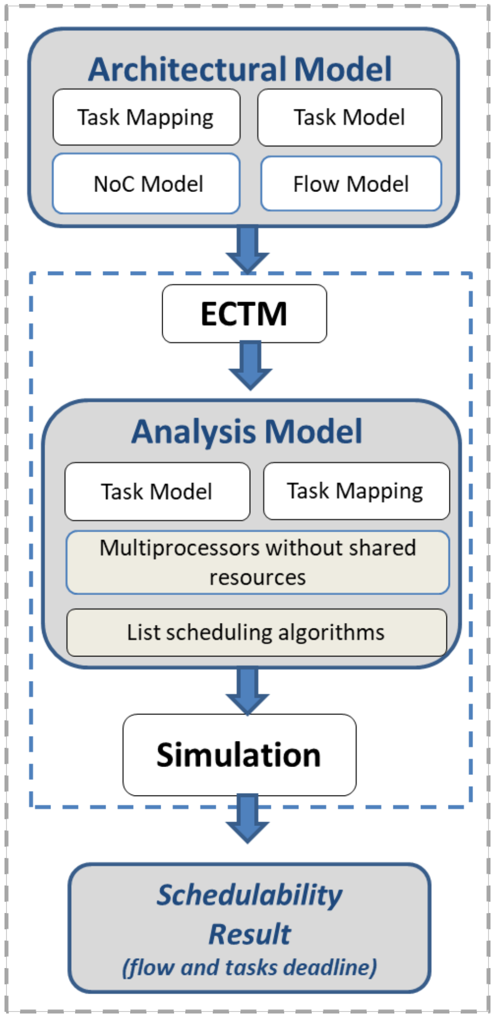
Networks-on-Chip Aware Schedulability Analysis



- ❑ Processing units interconnection
- ❑ 3x3 Networks-on-Chip
- ❑ Communication infrastructure based on links and routers that interconnect processing or memory unit
- ❑ Links can be shared between different messages
- ❑ Store & Forward or Wormhole switching mode

Networks-on-Chip Aware Schedulability Analysis

- ❑ **ECTM :** converts flow of messages scheduling to periodic tasks scheduling.
- ❑ **Analysis model**
  - ❑ Each flow of message is transformed to a set of dependent periodic tasks
  - ❑ DAG Periodic task set + identical multiprocessor
  - ❑ Simulation over feasibility interval with HLFET (list scheduling)
- ❑ **Require to model:** shared link, routing protocol, processing and memory units, task mapping



Agenda

1. Introduction
2. Multiprocessor scheduling analysis
3. Cheddar features and modeling requirements
4. With AADL

What to model for schedulability

- **Don't need a detailed hardware model:** express interferences only

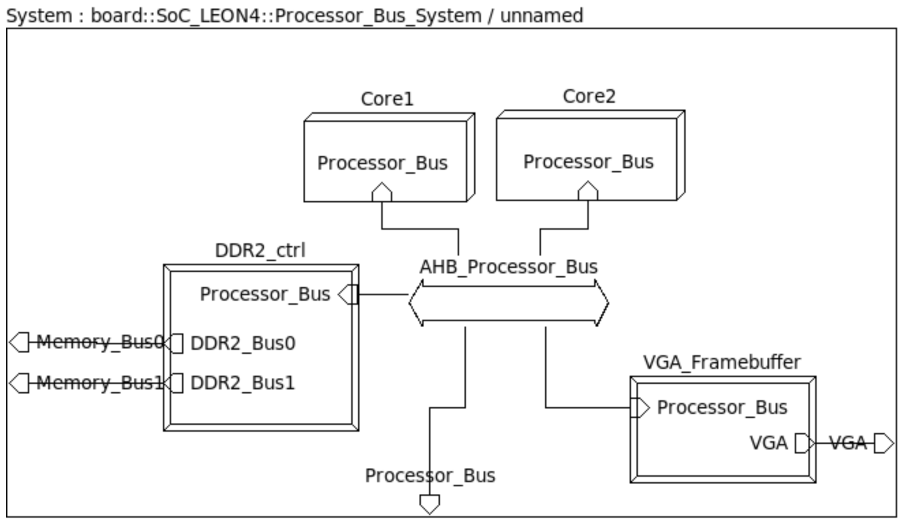
- **We need:**
  1. **Define entities/components** (resources and users) + properties: cache, memory layout, thread, …
  2. **Express component relationships leading to interferences,** e.g. set of threads using set of shared caches
     - Shared or private (partitioned)?
  3. **Sharing protocol**: e.g. Scheduling_Protocol

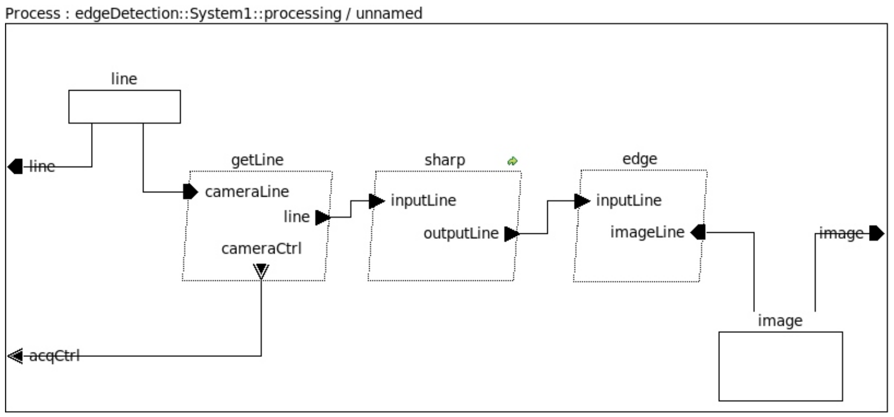
Example with processing units

- **SoC is an AADL "system".**
- **Processor/memory instances inside a "system" component:** e.g. model any processing unit, in the same or in different chips (i.e. core)
- Bindings between soft and SoC system

- **For a global scheduling:**
  1. Define threads (consumers) and processing units (resources).
  2. Express which subset of processing units run which threads
  3. Specification of the sharing protocol (e.g. Scheduling_Protocol with global EDF for all processor components)

Partitioned Scheduling

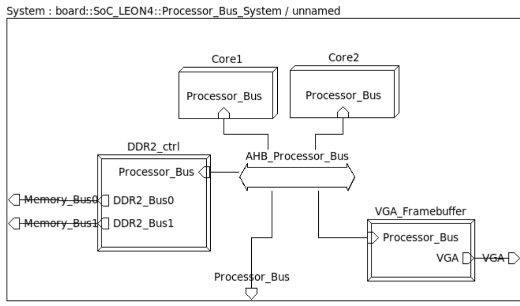




```
SYSTEM IMPLEMENTATION product.impl
SUBCOMPONENTS
  hard : SYSTEM soc_leon4::soc.asic_leon4;
  bank0 : MEMORY ram.ddr2;
  bank2 : MEMORY ram.ddr2;
  soft : PROCESS edgeDetection.impl;
PROPERTIES
  actual_processor_binding => (REFERENCE(hard.Proc_System.Core1)) APPLIES TO soft.getLine;
  actual_processor_binding => (REFERENCE(hard.Proc_System.Core2)) APPLIES TO soft.sharp;
  actual_processor_binding => (REFERENCE(hard.Proc_System.Core2)) APPLIES TO soft.edge;
  Scheduling_Protocol => (Rate_Monotonic_Protocol) applies to hard.core1;
  Scheduling_Protocol => (Rate_Monotonic_Protocol) applies to hard.core2;
END product.impl;
```

Global Scheduling

```
Actual_Processor_Binding => (REFERENCE(hard.Proc_System.Core1),
                             REFERENCE(hard.Proc_System.Core2))
                                     APPLIES TO soft.getLine;
Actual_Processor_Binding => (REFERENCE(hard.Proc_System.Core1),
                             REFERENCE(hard.Proc_System.Core2))
                                     APPLIES TO soft.sharp;
Actual_Processor_Binding => (REFERENCE(hard.Proc_System.Core1),
                             REFERENCE(hard.Proc_System.Core2))
                                     APPLIES TO soft.edge;
Scheduling_Protocol => Rate_Monotonic_Protocol applies to hard.Core1;
Scheduling_Protocol => Rate_Monotonic_Protocol applies to hard.Core2;
```



The binding may be applied at the high container level

```
Actual_Processor_Binding => (REFERENCE(hard.Proc_System.Core1),
                             REFERENCE(hard.Proc_System.Core2)) applies to soft;
```

Other shared resources: cache units

- **Similar requirements for most of shared resources**: cache, bus, NoC
- **Memory component instances:** models any cache units, i.e. L1, L2, protocols, data, instruction, …
- **For a shared cache:**
  1. Define cache units (resources) and consumers (processing units or threads)
  2. Express cache units used by processing units or threads
  3. A specification of the sharing protocol (e.g. LRU, line allocation)
- **For a partitioned cache**: only ONE processing unit/thread accessing a private part of the cache

SoC modeling for schedulability

- SoC designs exercised with AADL since 2013 … but no summary

- **Implemented_As?** Does not allow to express all binding/resource allocation & interferences (e.g. memory/part of cache to a thread).
- Bindings abilities ? e.g. virtual memory/processor
- **processor or virtual processor?**
  - Virtual = logical point of view? Virtual processor and ARINC 653 hierarchical scheduling?
- **Some properties at "system" level (**e.g. Scheduling_Protocol):
  - Global scheduling, the protocol **must** be the same for all the scheduled processors/Virtual processors?

- **CFG:** BA? Or separate model + properties?
- Cache task access profile: cache & memory layout

Conclusion

- **Multiprocessors:** several shared resources, not only the processing units
- **Interferences between software and hardware units** are needed for schedulability analysis

- **AADLInspector:** partitioned only and without shared hardware resources.
- **Cheddar/OSATE:** prototyping work for multiprocessor, experiment V3
- **Ongoing analysis:** memory interferences analysis