Multiprocessor schedulability analysis with AADL and Cheddar

Frank Singhoff, Stéphane Rubini, Hai Nam Tran, Sébastien Levieux

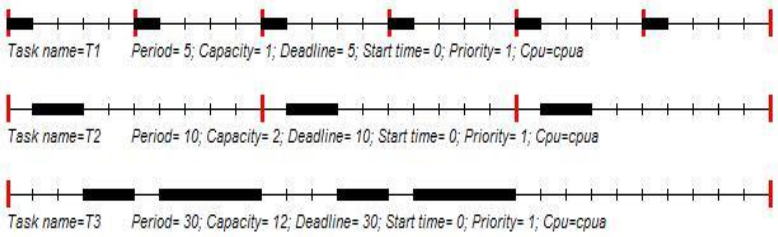
Workshop ADEPT 2024, Barcelona
June 2024

Lab-STICC, UMR CNRS 6285, University of Brest

Agenda

1. Introduction
2. AADL modeling of multiprocessor architecture
3. Schedulability analysis approaches
4. Case study
5. Conclusion

Introduction



$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{P_j} \right\rceil \cdot C_j$$

- **Real-time schedulability analysis**
  - Schedulability: do functions (tasks) meet their deadlines?
    - Feasibility tests, simulations, model-checking, …
  - Models of functions (or tasks)
    - E.g. periodic task model
  - Models of execution platform (e.g. computing units)
  - Interference: delay added to the execution time of a task caused by another entity

Introduction

- **Problem statement / ongoing work**
  - Efficient verification methods exists for uniprocessor platforms
  - 80% of stakeholders use multiprocessor platforms while verification of such architecture stays an active research track [Akesson et al. 2022]
  - We focus on early verification, i.e. certification purpose
  - How to model with AADL V2?
    - What do need to abstract or model with schedulability in mind?
    - What are missing in the current AADL standard?
  - What kind of schedulability analysis we can give to AADL users?
- **Expected contributions**
  - Modeling guidelines, AADL models and properties, schedulability in mind
  - Prototypes of schedulability methods (e.g. inside Cheddar/AADLInspector)

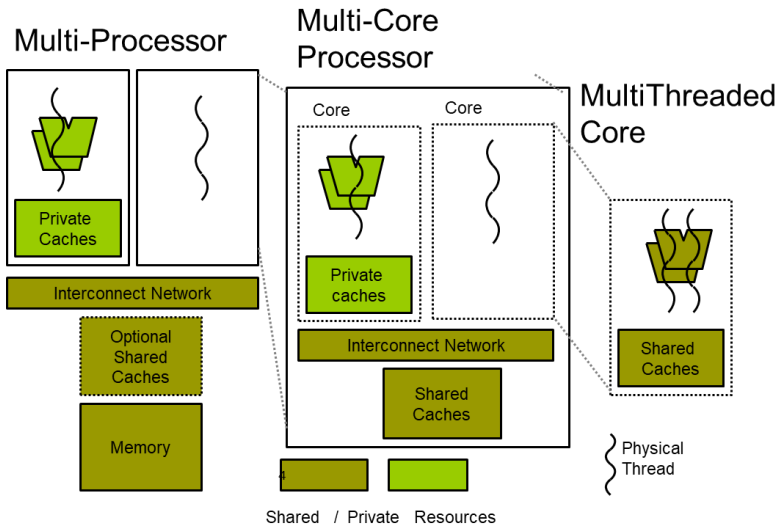
Akesson, B., Nasri, M., Nelissen, G., Altmeyer, S., & Davis, R. I. (2022). A comprehensive survey of industry practice in real-time systems. *Real-Time Systems*, *58*(3), 358-398.

Agenda

1. Introduction
2. AADL modeling of multiprocessor architecture
3. Schedulability analysis approach
4. Case study
5. Conclusion

AADL modeling of multiprocessor architecture

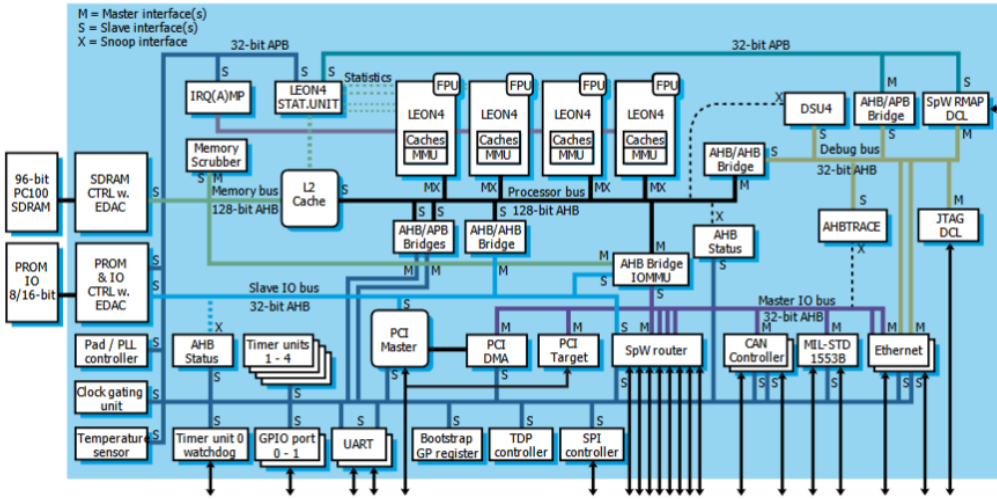
- **First research works on multiprocessor schedulability assumed simple models of computing units**
- **Not compliant with current SoC**



- **Various processing units**
  - Heterogeneous, different speeds
  - Cores, accelerators, GPU/NPU/TPU, physical threads
  - Core clusters, AMP/SMP
  - Multicore? Manycore?
- **Various scheduling parameters**
  - Partitioned, global scheduling
    - Threads mapping to processing units: off-line or on-line
    - Migration models
  - Scheduling policies and their parameters

AADL modeling of multiprocessor architecture

- SoC may include devices that are shared resources and lead to interference [Maiza 2019]
- Must be also accounted in the schedulability



- **Various memory units**: cache L1/L2, I or D, memory controller, DRAM, NVRAM, scratchpad
  - CPRD: Cache Related Preemption Delay
- **Various Interconnect units:** (memory) complex buses, crossbar, NoC
- **Specific devices or mechanisms:** DMA, Scrubber

Maiza, C., Rihani, H., Rivas, J. M., Goossens, J., Altmeyer, S., & Davis, R. I. (2019). A survey of timing verification techniques for multi-core real-time systems. *ACM Computing Surveys 52*(3), 1-38.

Proposed approach/Modeling guidelines

- AADL has been used to model multiprocessor architectures
  - Partitioned (no migration/global scheduling)
  - With interference due to thread or operating system
- But less to model
  - Interference due to hardware components
  - Global scheduling, thread migrations between processing units

- We want to model larger kinds of multiprocessor architecture
- Let focus on multicore platforms to illustrate

Proposed approach/Modeling guidelines

- We do not need to model hardware in the detail
- What do we need to model for schedulability analysis purposes:
  1. Any component that is shared and lead to interference: processing units, cache units, interconnect units
  2. Software entities that may suffer interference (thread)
  3. Properties for any behavior specifying component/resource sharing, e.g. scheduling policy, cache partitioning, bus protocol
  4. Properties for known interference values, e.g. CPRD (produced by measurement or analysis)

- Use AADL processor to model resource sharing/scheduling
- Use AADL system to model SoC and its internal units (processing, memory, interconnect)
- Use AADL thread/data to express interference inside units

Example: dual-core processing units with cache and memory units

```
package multicore_bus_units
public
       with aadlv3;
       with Cheddar_Multicore_Properties;
       with memory_units;

 processor uni_core
 properties
   Scheduling_Protocol=>(POSIX_1003_HIGHEST_PRIORITY_FIRST_PROTOCOL);
 end uni_core;

 system dual_core
 end dual_core;

 system implementation dual_core.impl
       subcomponents
                core1 : processor uni_core;
                core2 : processor uni_core;
                core1_L1_ICache : memory memory_units::Cache.impl;
                core2_L1_ICache : memory memory_units::Cache.impl;
     properties
                aadlv3::System_Soc_Type => SoC_Processing_Unit;
                Cheddar_Multicore_Properties::SoC_Interconnection_Type => Shared_Bus;
 end dual_core.impl;

end multicore_bus_units;
```

Example: dual-core processing units with cache and memory units

```
memory Cache
end Cache;

memory implementation Cache.impl
  properties
      Memory_Size => 1024Bytes;
      Cheddar_Multicore_Properties::Line_Size => 16Bytes;
      Cheddar_Multicore_Properties::Cache_Type => Instruction_Cache;
      Cheddar_Multicore_Properties::Cache_Level => 1;
      Cheddar_Multicore_Properties::Associativity => 1;
      Cheddar_Multicore_Properties::Cache_Size => 1024;
      Cheddar_Multicore_Properties::Block_Reload_Time => 1 us .. 2 us;
end Cache.impl;

system cache_example
end cache_example;

system implementation cache_example.impl
   subcomponents
     hard : system multicore_bus_units::dual_core.impl;
     soft : process Application.impl;
   properties
     actual_processor_binding => (reference(hard.core1)) applies to soft.ordo_bus;
     actual_processor_binding => (reference(hard.core1)) applies to soft.donnees;
     actual_processor_binding => (reference(hard.core2)) applies to soft.pilotage;
end cache_example.impl;
```

Agenda

1. Introduction
2. AADL modeling of multiprocessor architecture
3. Schedulability analysis approach
4. Case study
5. Conclusion

Schedulability analysis for multicore architectures

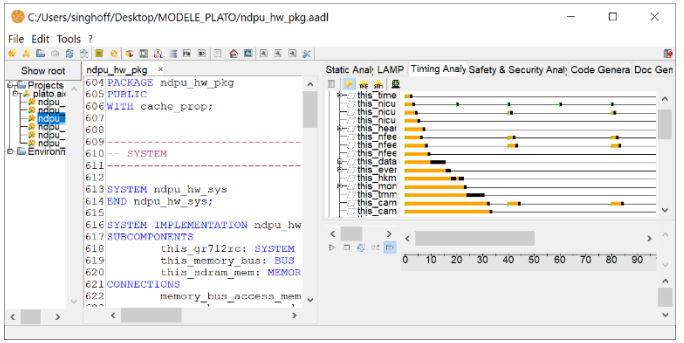
- Stay challenging after about 25 years of research
- Several approaches [Maiza at al. 2019], but no general solution
  1. Dedicated software/hardware architectures to limit/reduce interference
  2. WCRT/RTA oriented analysis: many works but focusing on one or few shared resources. May lead to proof, but how to combine them to support complex hardware?
  3. Scheduling simulation: extensible but may do not lead to proofs and may produce wrong results
     - Sustainability and feasibility intervals [Goosens et al. 1996]
  4. …
- Currently investigating with Cheddar
  - 3rd solution/scheduling simulation
  - Combined with interference measurements or analysis. Cache L1 model [Tran et al. 2017], DRAM model based on [Kim et al, 2016], Kalray memory model [Tran et al., 2019], NoC model [Dridi et al., 2021]

Agenda

1. Introduction
2. AADL modeling of multiprocessor architecture
3. Scheduling analysis approach
4. Case study
5. Conclusion

PLATO (PLAnetary Transits and Oscillations of stars) case study

- **ESA program that needs High Performance Computing. Launch planned in 2026.**
- **LESIA and CNES designed an AADL model for the PDR (Preliminary Design Review) purpose**



- **1,1k lines of AADL**
  - 34 threads, 560 property associations
- **Schedulability analysis**
  - LEON3 dual core with L1 caches
  - Interference values measured by the software execution platform (GERICOS)
  - No need to explicit AADL cache modeling
  - Scheduling interval: 54 seconds (feasibility interval?)
  - AADLInspector (Cheddar)

Plasson, P., Brusq, G., Singhoff, F., Tran, H. N., Rubini, S., & Dissaux, P. (2022). PLATO N-DPU on-board software: an ideal candidate for multicore scheduling analysis. In *11th ERTSS Congress ERTSS, Toulouse*.

Agenda

1. Introduction
2. AADL modeling of multiprocessor architecture
3. Schedulability analysis approach
4. Case study
5. Conclusion

Conclusion

- Multiprocessor schedulability stays difficult
- **To summarize/expected contributions**
  - Modeling guidelines for AADL
  - Schedulability based on scheduling simulation, interference values either by measures or analytically computed
  - Prototyping in AADLInspector, OSATE Cheddar Plugin
  - Example: PLATO ESA program
  - Low TRL: ongoing prototype …
- **Future works**
  - Extend the work to GR740 boards: collect interference measurements and adapt modeling/verification methods
  - PLATO: schedulabiltiy analysis Critical Design Review
  - Scheduling simulation with measurements in the loop

Issues raised by schedulability analysis based on scheduling simulation

- **Feasibility intervals**
  - Definition [Goossens et al., 2016]: a finite interval [a, b] such that if all the deadlines of jobs released in the interval are met, then the system is schedulable.

- **Sustainability analysis**
  - Definition [Goossens et al., 1996]: a given scheduling policy and/or a schedulability test is sustainable if any system that is schedulable under its worst-case specification remains so when its behavior is better than the worst-case

Goossens, J., Grolleau, E., & Cucu-Grosjean, L. (2016). Periodicity of real-time schedules for dependent periodic tasks on identical multiprocessor platforms. *Real-time systems*, *52*, 808-832.

AADL guidelines

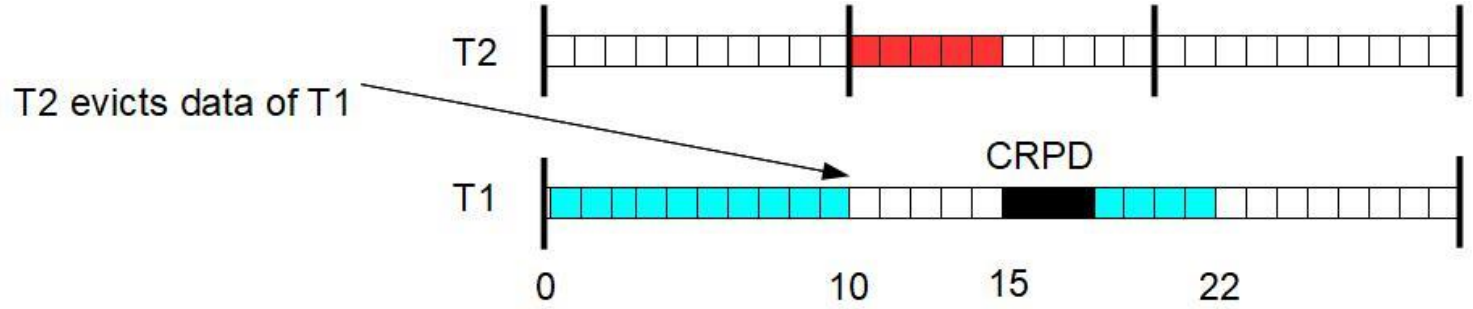
- Use AADL processor to model resource sharing/scheduling
- Use AADL system to model SoC, or processing, memory, interconnect units
- New properties: `Cheddar_Multicore_Properties`[1]
  - Specify bevahior of SoC components and properties related to interference
  - Processing units: migration type, core type, ISA, speed/MIPS, …
  - Memory units, …
  - Interconnect units, …

```
Supported_Soc_Type : type enumeration (
        SoC_Processing_Unit,
        SoC_Memory_Unit,
        Soc_Interconnection_Unit);
System_Soc_Type : Supported_SoC_Type applies to (system);
```

[1]http://beru.univ-brest.fr/svn/CHEDDAR/trunk/project_examples/aadl/Cheddar_Multicore_Properties.aadl

Example of hardware interference

- With multiprocessor, interference does not come only from tasks or system services, but also by hardware resources



- Cache Related Preemption Delay (CRPD)
  - CRPD, additional time to refill the cache with memory blocks evicted by preemption.
  - CRPD: up to 44% of the WCET [Pellizzoni et al. 2007]
- CRPD depends on preemption time, then on scheduling, then on execution time, and increase execution time => cyclic dependency ☹

Pellizzoni, R., Caccamo, M.. Toward the predictable integration of real-time cots based systems. In 28th International Real-Time Systems Symposium (RTSS), pages 73–82. IEEE, 2007.

Cheddar schedulability analysis based on scheduling simulation

- Scheduling simulation built with 3 features

1. Multiprocessor scheduling policies (about 15 policies)
   - Global classical policies (EDF, RM, LLF, …)
   - Specific multiprocessor policies (eg. EDZL, Pfair, RUN, …)
   - Hierarchical (e.g. ARINC 653)
   - Mixed criticality policies (e.g. AMC)
2. Feasibility intervals (almost [Goosens et al. 2016] results)
3. Interference analysis
   - Measurement (e.g. PLATO)
   - Analytical: