

Update on AADLInspector and Cheddar

P. Dissaux*, J. Legrand*, A. Schach*, S. Rubini+, J. Boukhobza+,
L. Lemarchand+, J.P. Diguët+, N. Tran+, M. Dridi+,
R. Bouaziz\$, F. Singhoff+

* Ellidiss Technologies
+ Lab-STICC UMR CNRS 6285/UBO
\$ ReDCAD Laboratory, University of Sfax



Summary

1. AADLInspector & Cheddar

2. AADLInspector 1.6 : new features

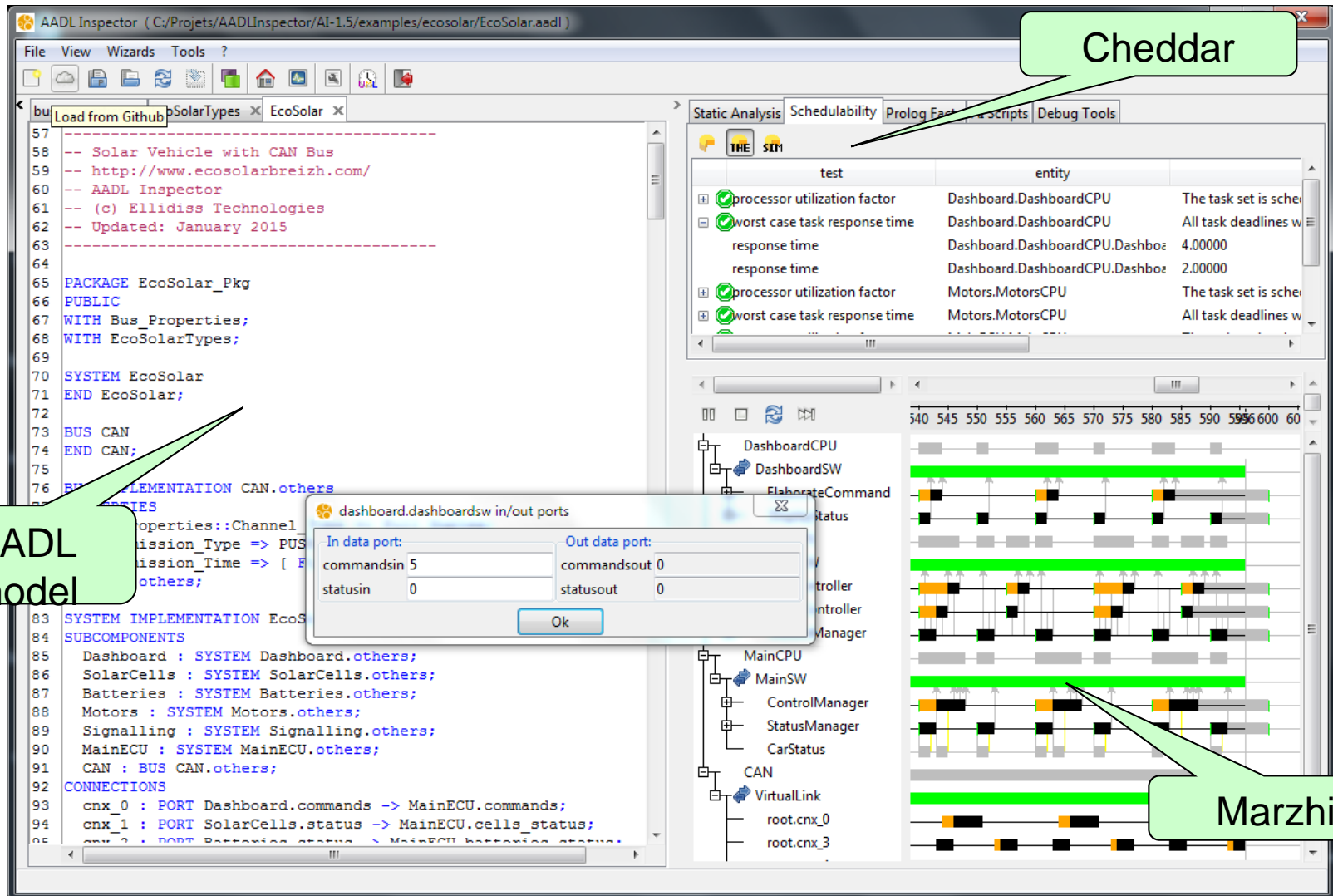
3. Multiprocessor analysis features

- 1. Features in AADLInspector**

- 2. Current research activities outcomes (in Cheddar but not yet in AADLInspector)**

AADLInspector

Model Processing Framework



The screenshot displays the AADLInspector application window. The left pane shows the AADL model code for 'EcoSolar'. The right pane is divided into two sections: 'Static Analysis' and 'Schedulability'. The 'Static Analysis' section shows a table of test results for 'processor utilization factor' and 'worst case task response time' for 'DashboardCPU' and 'MotorsCPU'. The 'Schedulability' section shows a timeline graph with task execution bars. A callout box labeled 'Cheddar' points to the 'Schedulability' tab. Another callout box labeled 'AADL model' points to the code editor. A third callout box labeled 'Marzhin' points to the timeline graph. A dialog box titled 'dashboard.dashboardsw in/out ports' is open in the center, showing input and output ports for 'commands' and 'status'.

AADL model

```

57
58 -- Solar Vehicle with CAN Bus
59 -- http://www.ecosolarbreizh.com/
60 -- AADL Inspector
61 -- (c) Ellidiss Technologies
62 -- Updated: January 2015
63
64
65 PACKAGE EcoSolar_Pkg
66 PUBLIC
67 WITH Bus_Properties;
68 WITH EcoSolarTypes;
69
70 SYSTEM EcoSolar
71 END EcoSolar;
72
73 BUS CAN
74 END CAN;
75
76 IMPLEMENTATION CAN.others
77
78 properties::Channel
79 mission_Type => PUS
80 mission_Time => [E
81 others;
82
83 SYSTEM IMPLEMENTATION EcoS
84 SUBCOMPONENTS
85 Dashboard : SYSTEM Dashboard.others;
86 SolarCells : SYSTEM SolarCells.others;
87 Batteries : SYSTEM Batteries.others;
88 Motors : SYSTEM Motors.others;
89 Signalling : SYSTEM Signalling.others;
90 MainECU : SYSTEM MainECU.others;
91 CAN : BUS CAN.others;
92 CONNECTIONS
93 cnx_0 : PORT Dashboard.commands -> MainECU.commands;
94 cnx_1 : PORT SolarCells.status -> MainECU.cells_status;
95 cnx_2 : PORT Batteries.status -> MainECU.batteries_status;
  
```

Cheddar

test	entity	
processor utilization factor	Dashboard.DashboardCPU	The task set is sche
worst case task response time	Dashboard.DashboardCPU	All task deadlines w
response time	Dashboard.DashboardCPU.Dashbo	4.00000
response time	Dashboard.DashboardCPU.Dashbo	2.00000
processor utilization factor	Motors.MotorsCPU	The task set is sche
worst case task response time	Motors.MotorsCPU	All task deadlines w

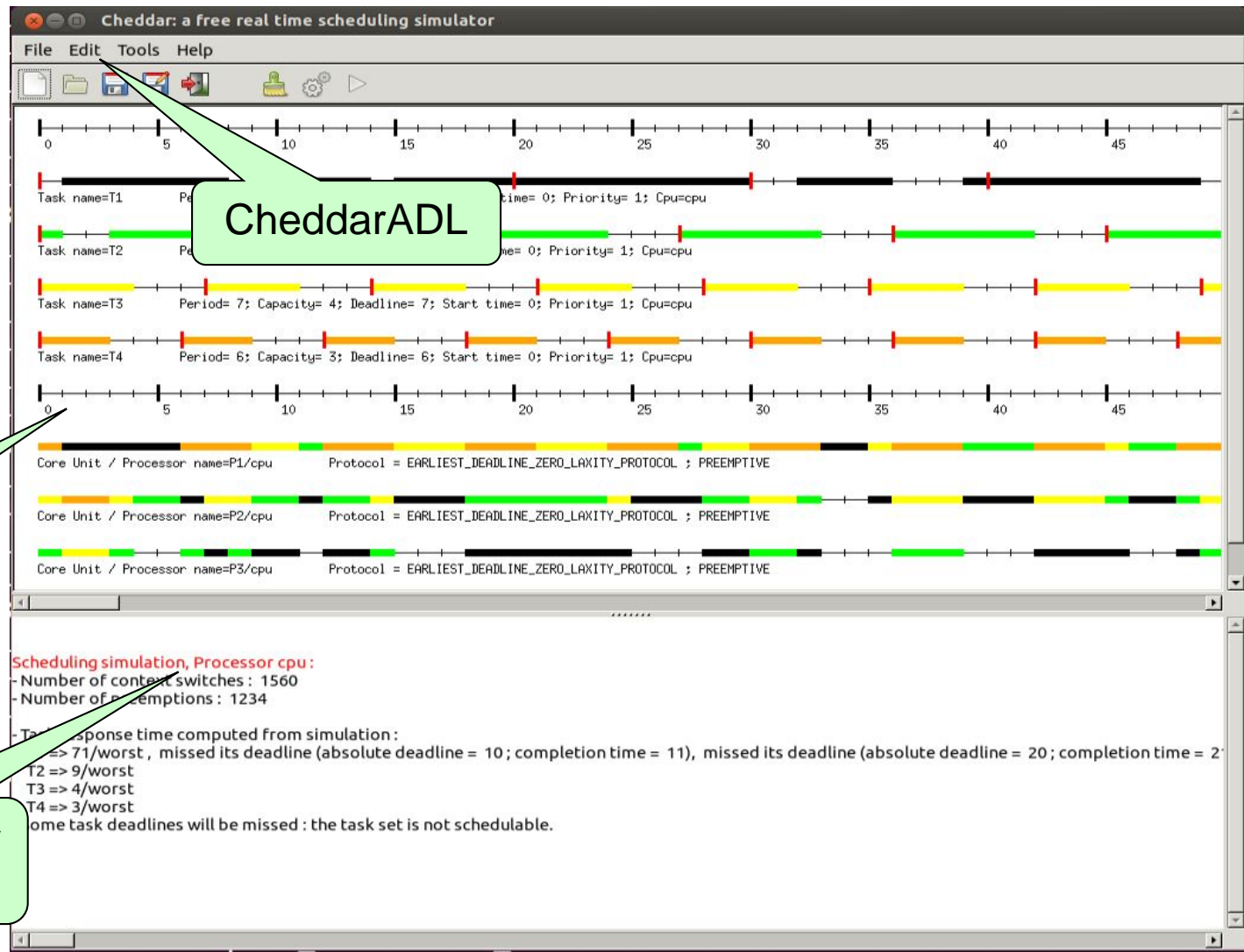
Marzhin

dashboard.dashboardsw in/out ports

In data port:	Out data port:
commandsin 5	commandout 0
statusin 0	statusout 0

Cheddar

Scheduling analysis framework

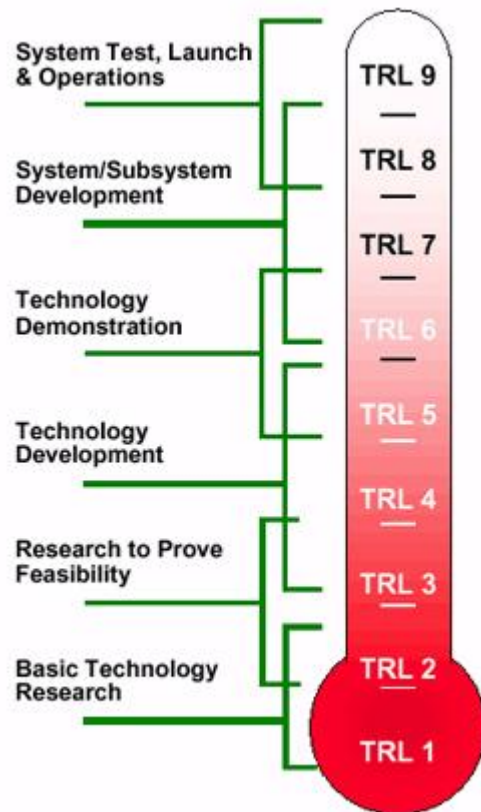


CheddarADL

Simulator

Schedulability
tests

From AADLInspector to Cheddar



AADLInspector

Industrialization
Tool packaging
Commercial support (Ellidiss)



R&D,
collaborative projects,
prototyping
(UBO + Ellidiss + others)

Cheddar



Research activities (Lab-STICC/UBO)

Summary

1. AADLInspector & Cheddar

2. AADLInspector 1.6 : new features

3. Multiprocessor analysis features

1. Features in AADLInspector

2. Current research activities outcomes (in Cheddar but not yet in AADLInspector)

AI 1.6 Features

- **Imports XML/XMI models:**
 - generic transformation process for ECore based models using LMP
 - existing prototypes for UML/MARTE, SysML, Capella, ...
- **AADL model processing:**
 - turnkey embedded tools:
 - Cheddar (scheduling analysis)
 - Marzhin (event based simulation)
 - Ocarina (AADL compliancy analysis, code generation)
 - customizable plugins using the LMP AADL toolbox:
 - AADL parser (aadlrev)
 - AADL processing libraries (instance model, legality rules, ...)
- **AADL projects manager:**
 - core 2.2 + annex sub-languages EMV1, EMV2, BA 2.0
 - interface with other AADL editors (Osate, Stood, ...) and github
 - hierarchical
- **Improved simulation interface**



AADL Inspector 1.6

The screenshot displays the AADL Inspector 1.6 interface. The **Project browser** on the left shows a tree of project files. The **AADL model** pane in the center shows the source code for the `sys_mars_pathfinder` system. The **Simulation I/O** section at the bottom left contains three analog gauges. The **Response Time** table in the top right provides timing analysis data. The **Simulation** view at the bottom right shows a Gantt chart of task execution.

Project browser

AADL model

```
44 package mars_pathfinder
45 public
46
47   with pathfinder_software;
48   with pathfinder_hardware;
49
50   -----system-----
51   -----
52   -----
53
54   system sys_mars_pathfinder
55   end sys_mars_pathfinder;
56
57   system implementation sys_mars_pat
58   subcomponents
59     rs_6000 : processor pathfinder_
60     prs_PSC : process pathfinder_sc
61
62   VME : bus pathfinder ha
```

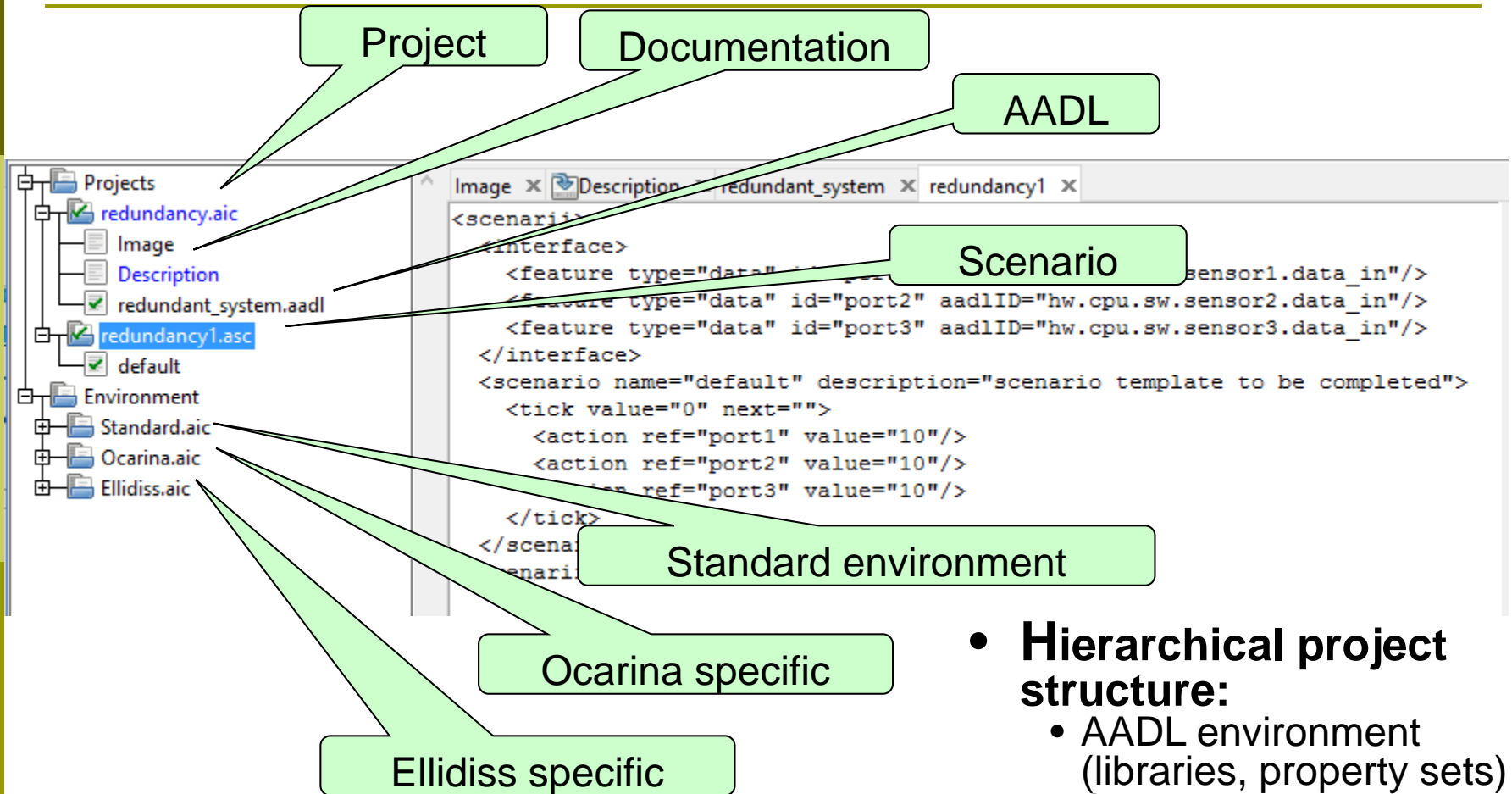
Response Time

	Deadline	Computed	Max Cheddar	Max Marzhin	Avg Cheddar	Avg Marzhin	Min
rs_6000		72.50 %		72.92 %			
prp_psc							
bus_scheduling	10	2.00000	2	2	2.00	2.00	
data_distribution	10	4.00000	8	8	4.10	4.11	
control_task	20	6.00000	10	10	6.20	6.21	
radio_task	20	8.00000	16	16	8.40	8.43	
camera_task	20	10.00000	18	18	10.40	10.44	
mesure_task	400	18.00000	18	18	18.00	18.00	
meteo_task	400	38.00000	26	26	26.00	26.00	

Simulation I/O

Simulation

AADL Projects manager



- **Hierarchical project structure:**
 - AADL environment (libraries, property sets)
 - Sharable sub-projects
 - Simulation scenari
 - Documentation sections

New presentation of the analysis results

Cheddar
Theoretical
Tests

Cheddar
Simulation

Marzhin
Simulation

	Deadline	Computed	Max Cheddar	Max Marzhin	Avg Cheddar	Avg Marzhin	Min Cheddar	Min Marzhin
<input checked="" type="checkbox"/> rs_6000		72.50 %		74.90 %				
<input checked="" type="checkbox"/> prs_psc								
<input checked="" type="checkbox"/> bus_scheduling	10	2.00000	2	2	2.00	2.00	2	2
<input checked="" type="checkbox"/> data_distributio	10	4.00000	8	8	4.10	4.16	4	4
<input checked="" type="checkbox"/> control_task	20	6.00000	10	10	6.20	6.32	6	6
<input checked="" type="checkbox"/> radio_task	20	8.00000	16	16	8.40	8.64	8	8
<input checked="" type="checkbox"/> camera_task	20	10.00000	18	18	10.40	10.64	10	10
<input checked="" type="checkbox"/> mesure_task	400	18.00000	18	18	18.00	18.00	18	18
<input checked="" type="checkbox"/> meteo_task	400	38.00000	26	26	26.00	26.00	26	26

New features to interact during simulation

```
<scenarii>
<interface>
  <feature type="data" id="port1" aadlID="hw.cpu.sw.sensor1.data_in"/>
  <feature type="data" id="port2" aadlID="hw.cpu.sw.sensor2.data_in"/>
  <feature type="data" id="port3" aadlID="hw.cpu.sw.sensor3.data_in"/>
</interface>
<scenario name="default" description="scenario template to be completed">
  <tick value="0" next="">
    <action ref="port1" value="10"/>
    <action ref="port2" value="10"/>
    <action ref="port3" value="10"/>
  </tick>
</scenario>
</scenarii>
```

Scenarii

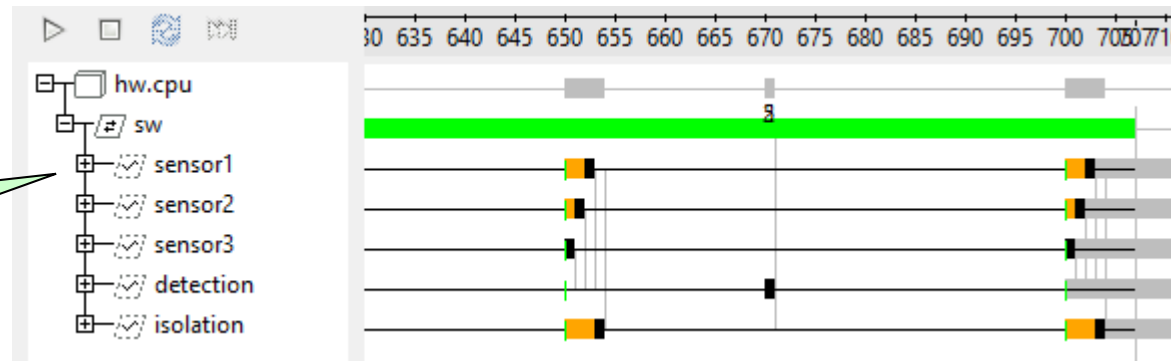
Process I/O

root.hw.cpu.sw in/out ports

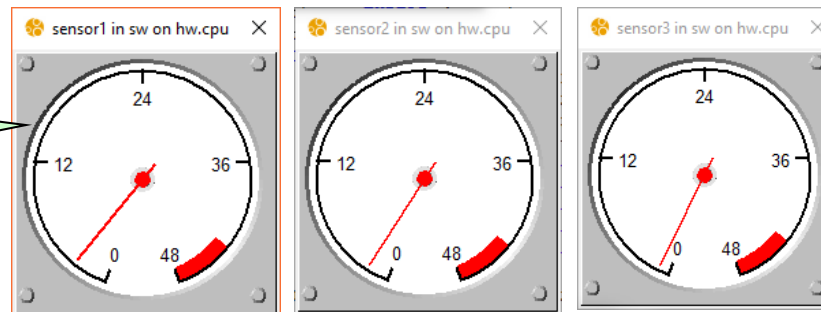
In data port:	Out data port:
data_in 5	data_out 5
	status 2

Ok Send all

Event bases
simulation



Threads
Response
Time



Multiprocessor with AADLInspector & Cheddar

1. AADLInspector & Cheddar

2. AADLInspector 1.6 : new features

3. Multiprocessor analysis features

1. Features in AADLInspector

2. Current research activities (in Cheddar but not yet in AADLInspector)

Multiprocessor with AADLInspector & Cheddar

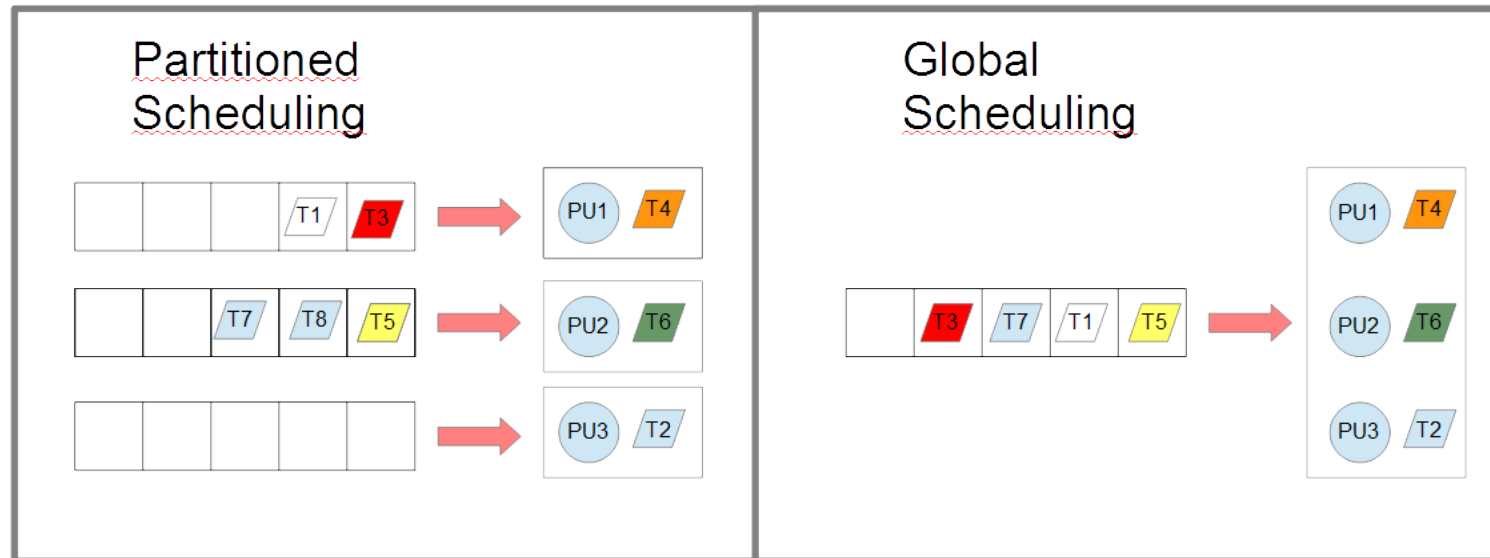
☐ **SMART project and later:**

- ☐ Define typical distributed/multiprocessor architectures AADLInspector should support
- ☐ How to model such distributed/multiprocessor architectures with AADL
- ☐ Choose or design scheduling analysis features for those patterns
- ☐ Prototype in Cheddar, to be made available in AADLInspector

☐ **Focus on:**

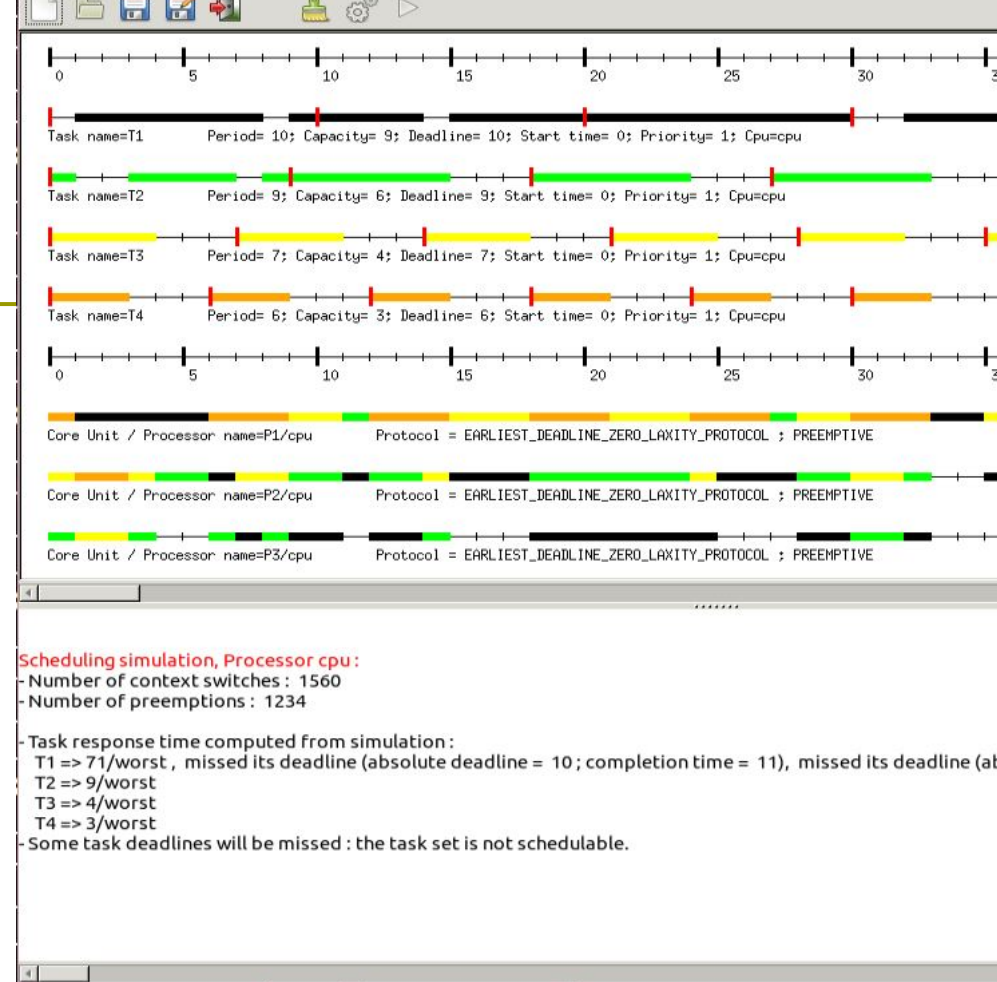
- ☐ Classical multiprocessor scheduling algorithms : partitioned vs global scheduling algorithms
- ☐ Shared resources between processing units, e.g. cache, memory bus, NoC

Multiprocessor with AADLInspector & Cheddar



- ❑ **Partitioned scheduling** : first assign offline each task on a processing unit ; each processing unit schedules its own task set.
 - ❑ No migration. Both online and off-line.
- ❑ **Global scheduling**: choose the next task to run on any available processing unit (or preempt if all busy).
 - ❑ With migration. Fully on-line.

Multiprocessor analysis features



❑ AADLInspector 1.6 :

- ❑ Partitioned scheduling only
- ❑ Available partitioning policies : Best fit, First Fit, Next Fit, GT, SF

❑ Cheddar 3.x only (not in AI yet):

- ❑ Global scheduling : any uniprocessor policies + specific policies such as EDZL, LLREF, Pfair, ... **(finished now)**
- ❑ Shared resources on multiprocessor architectures : Cache, NoC, memory **(on going)**
- ❑ Partitioning optimization approaches based on PAES **(on going)**

Cache/CRPD-Aware Priority Assignment Algorithm

- ❑ In fixed priority preemptive scheduling context, tasks can preempt and evict data of other tasks in the cache.
- ❑ Cache related preemption delay (**CRPD**): additional time to refill the cache with the cache blocks evicted by the preemption.
- ❑ **Problem statement:**
 - ❑ CRPD is high, non-negligible preemption cost. It can present up to 44% of the WCET of a task (Pellizzoni et al., 2007)
 - ❑ No fixed priority assignment algorithm takes CRPD into account.

Cache/CRPD-Aware Priority Assignment Algorithm

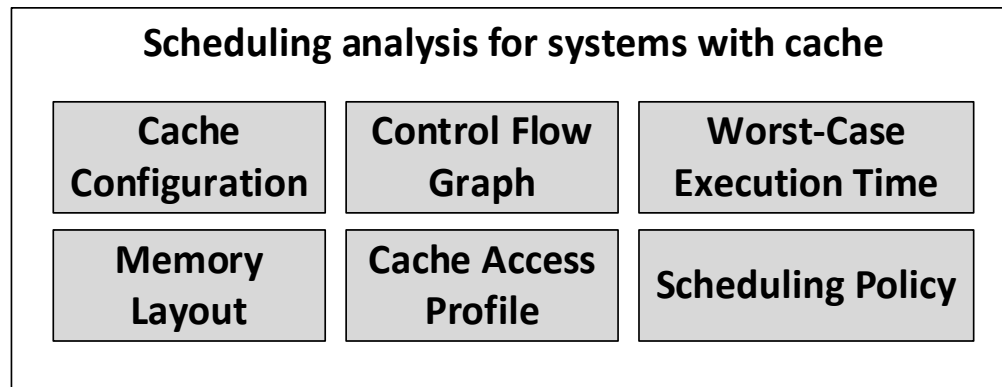
- ❑ Extend Audsley's priority assignment algorithm (Audsley, 1995) to take into account CRPD.
- ❑ CRPD-aware priority assignment algorithms (**CPA**) that assign priority to tasks and verify their schedulability.
- ❑ 5 algorithms with different levels of schedulability efficiency (1) and complexity (2,3).
- ❑ Implemented into Cheddar

	CPA-PT-Simplified	CPA-PT	CPA-Tree	Exhaustive Search
(1)	0.65	0.72	0.80	0.87
(2)	Low	Medium	High	
(3)	100 tasks	30 tasks	10 tasks	

Cache-Aware Scheduling Simulator

❑ Problem Statement:

- ❑ Various parameters need to be taken into account in scheduling analysis of systems with cache.
- ❑ Lack of tool addressing all parameters in the state-of-the-art work.
- ❑ Theoretical issues (feasibility interval, sustainability)



❑ Approach:

- ❑ Extend Cheddar component modeling related cache entities.
- ❑ Extend Cheddar scheduling simulator.

Networks-on-Chip Aware Scheduling Analysis

☐ **Context :**

- ☐ Networks-on-Chip (NoC)
- ☐ Communication infrastructure based on links and routers that interconnect PU or memory unit, providing packet-based data transfer.

☐ **Problem statement :**

- ☐ Relationships between thread models and communication models
- ☐ Various AADL thread communication design patterns
- ☐ Various NoC designs
- ☐ Today : AADL data port & 4x4 Wormhole XY NoC

- ☐ How to model both thread and communication models in order to enforce schedulability?

Networks-on-Chip Aware Scheduling Analysis

❑ **Dual Task and Flow Model (DTFM) :**

- ❑ Computes the flow model from a task model, task mapping and precedence constraints
- ❑ Identify/compute delays induced by the NoC architecture and perform scheduling analysis

❑ **DTFM Implemented into Cheddar**

- ❑ **DTFM is evaluated with a multiscale toolset**
composed of a tick accurate real-time scheduling tool (Cheddar) and a cycle accurate SystemC NoC simulator (SHOC).

Partitioning methods, multi-objective optimization

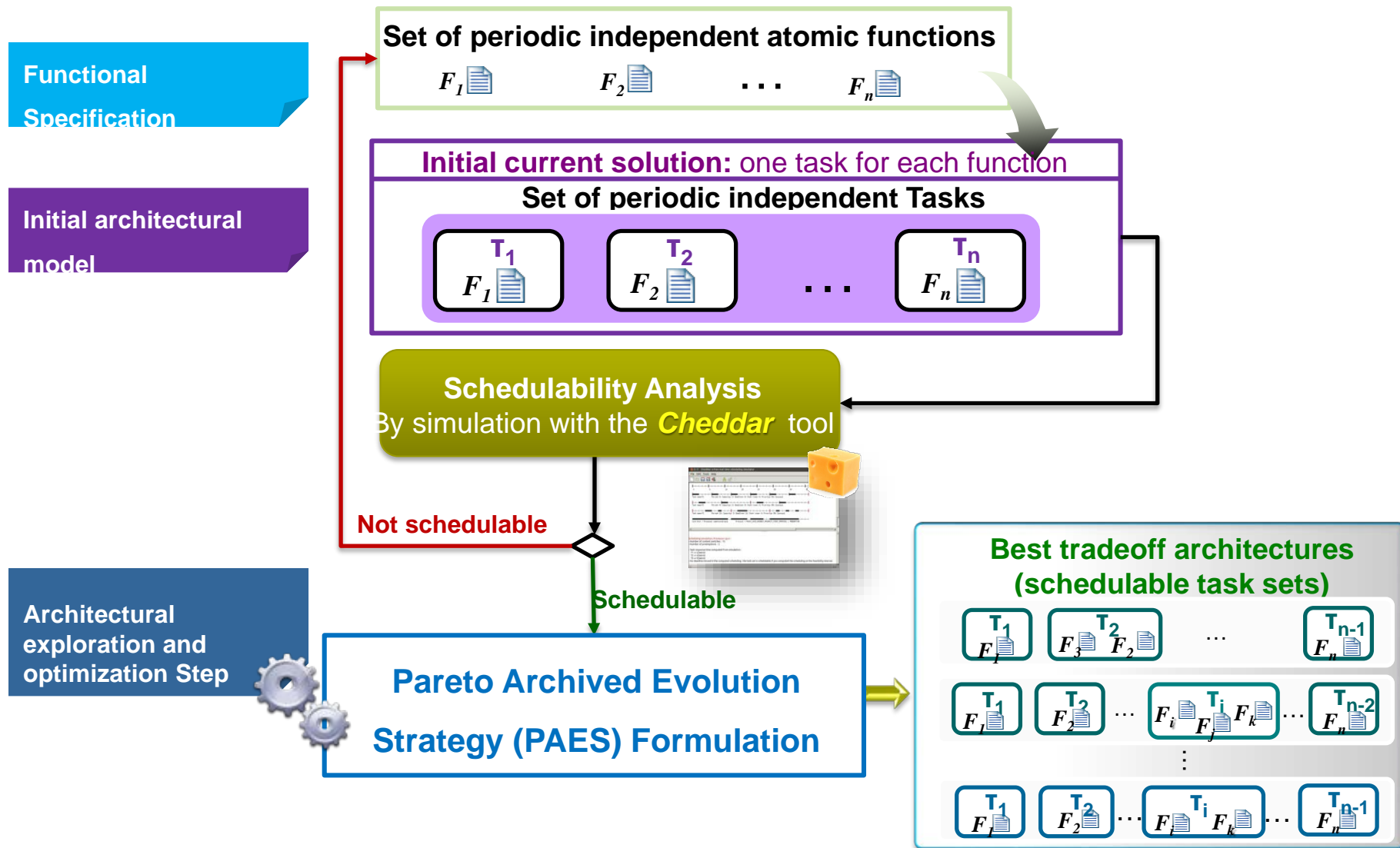
■ Problem Statement:

- Mapping functions to software architectures (i.e. RTOS tasks)
- Conflicting objective functions, e.g. number of preemptions vs laxity
 - Tradeoffs between large number of candidate software architectures

■ Contributions:

- Formulation based on PAES (Pareto Archived Evolution Strategy) to explore possible functions to tasks assignments
- Implementation into Cheddar, both sequential and parallel implementation
- Uniprocessor only right now

Partitioning methods, multi-objective optimization



Conclusion

☐ **New features in AADLInspector & Cheddar: about multiprocessors analysis:**

1. Typical multiprocessor architectures (SMART project)
2. Classical multiprocessor scheduling algorithms: partitioning vs global scheduling algorithms
3. New analysis features when (hardware) shared resources between computing units

☐ **AADL models handled during those activities**

- ☐ Bindings and Implemented_As ??
- ☐ Need a white paper

☐ **Questions ?**