

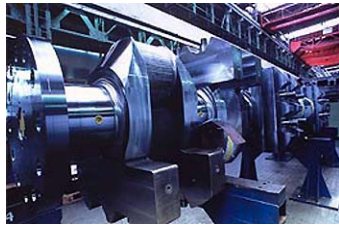
# Model-based Verification and Validation Embedded Systems

Kim G. Larsen

CISS – Aalborg University  
DENMARK



# ES are Pervasive



## Characteristica:

- Dedicated function
- Complex environment
- SW/HW/Mechanics
- Autonomous
- Ressource constrained
  - : Energy
  - : Bandwidth
  - : Memory
  - : ...
- **Timing constraints**



# ES are often Safety Critical



300 horse power  
100 processors

How to achieve ES that are:

- correct
- predicable
- dependable
- fault tolerant
- ressource minial
- cheap



## Model-Based Development



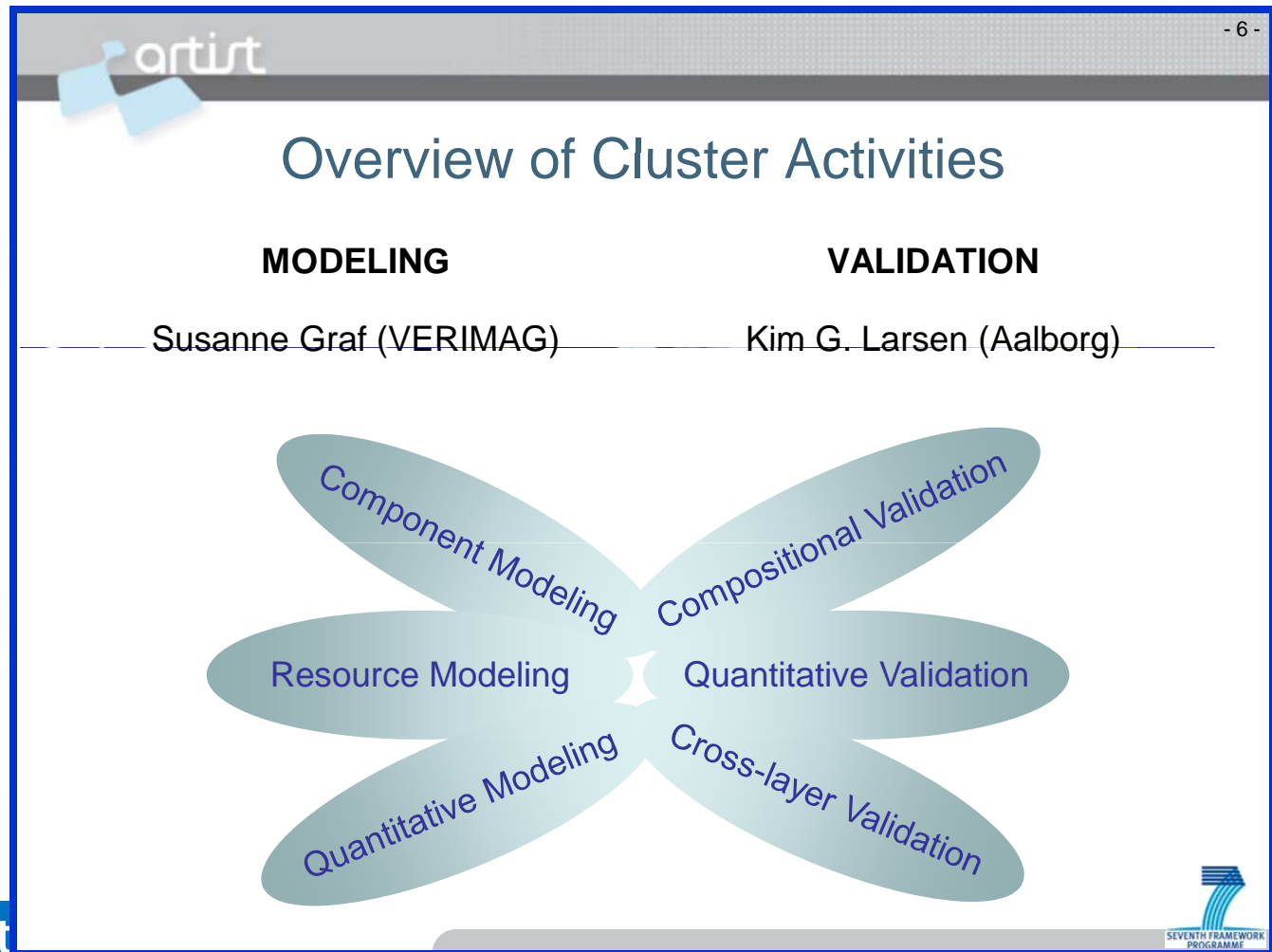
# Network of Excellence



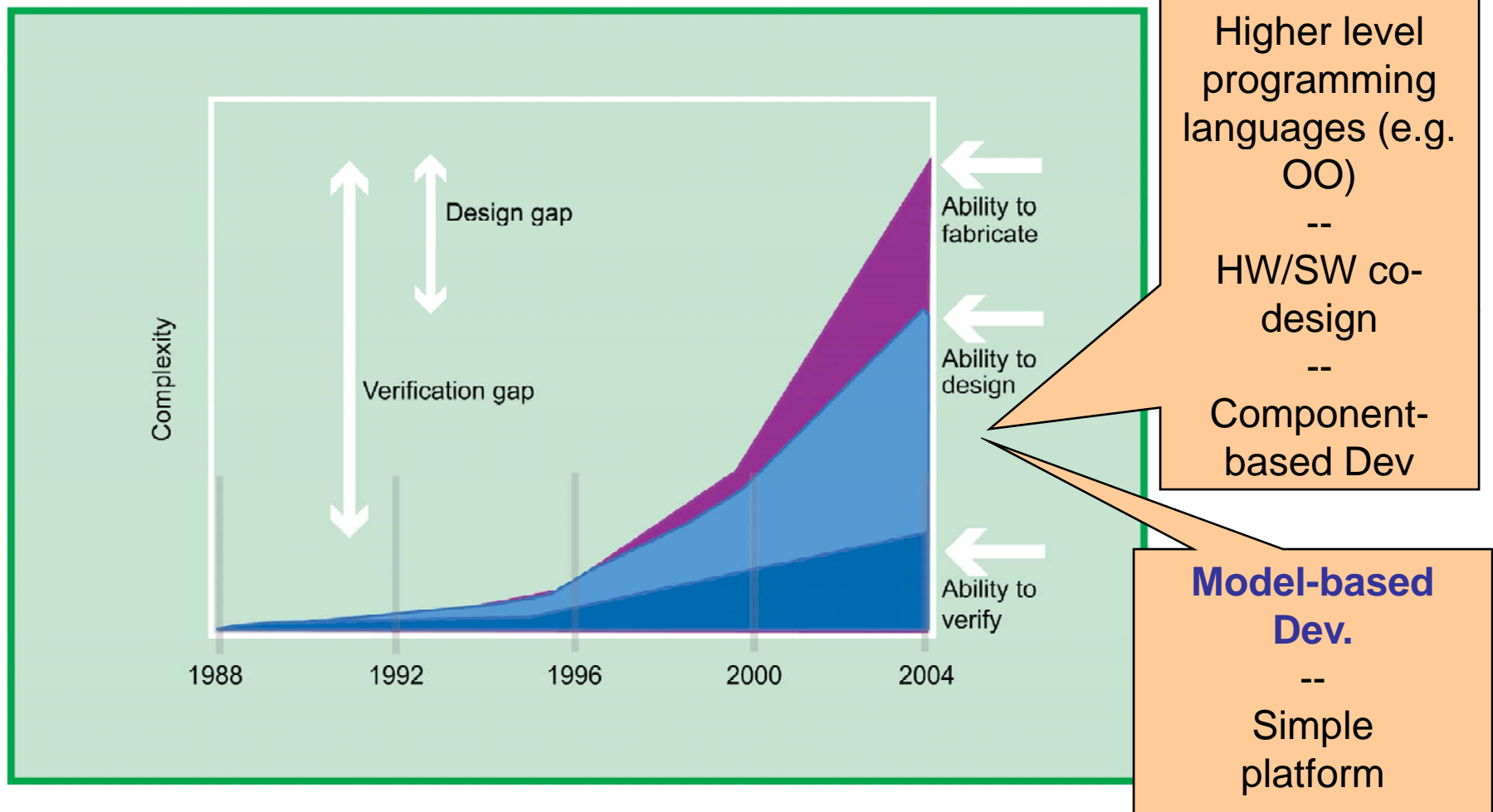
Coordinators  
Susanne Graf  
Kim G Larsen

31 Partners

Research Activities



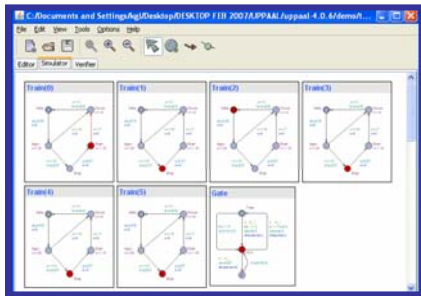
# Design & Verification Gap



Brian Bailey, Chief Technologist, Design Verification and Test Division, Mentor Graphics Corp.



# QUANTITATIVE Model Checking



System Description



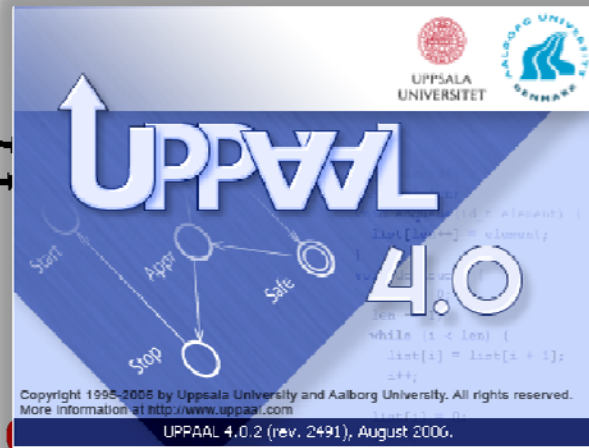
Time



Cost



Probability



No!

Debugging Information

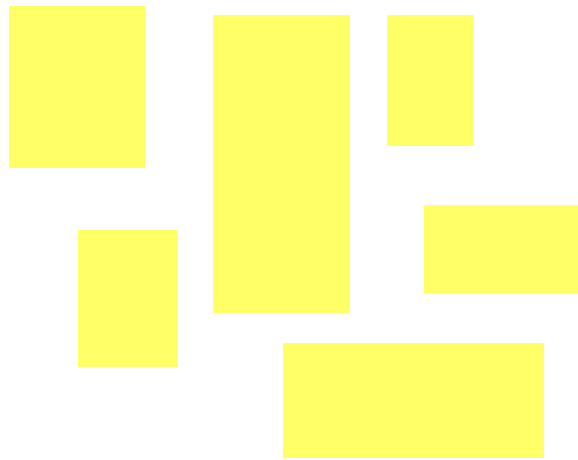
Requirement

Yes

- $A \square (\text{req} \Rightarrow A \diamond \text{grant})$
- $A \square (\text{req} \Rightarrow A \diamond_{t < 30s} \text{grant})$
- $A \square (\text{req} \Rightarrow A \diamond_{t < 30s, c < 5\$} \text{grant})$
- $A \square (\text{req} \Rightarrow A \diamond_{t < 30s, p > 0.90} \text{grant})$

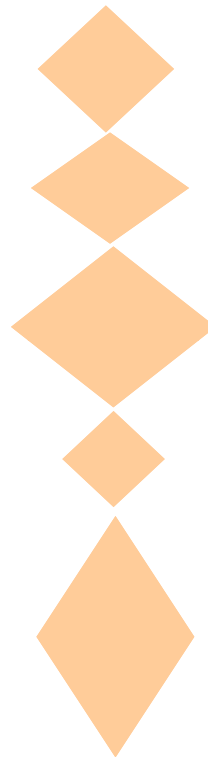


# Embedded Systems

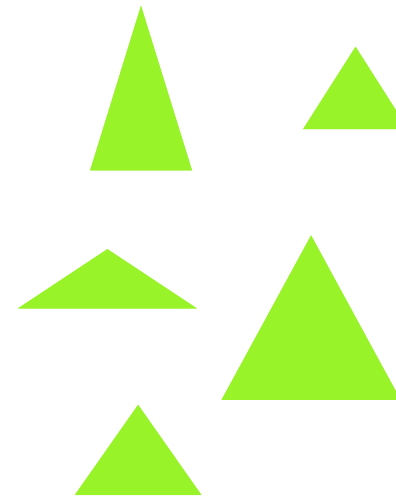


## Tasks:

Computation times  
Deadlines  
Dependencies  
Arrival patterns  
**uncertainties**



**Scheduling Principles (OS)**  
EDF, FPS, RMS, DVS, ..



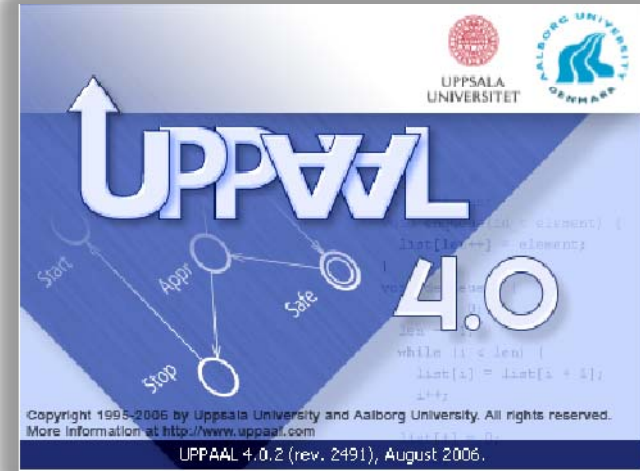
## Resources

Execution platform  
CPU, Memory  
Networks  
Drivers  
**uncertainties**



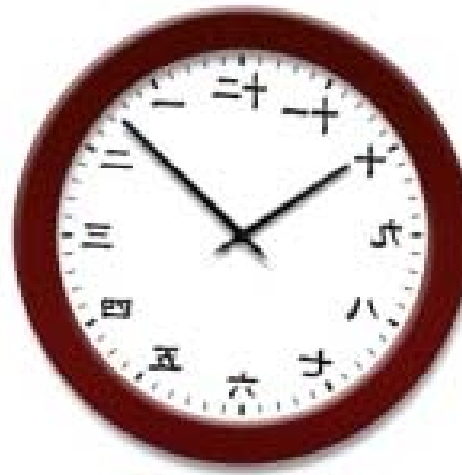
# Overview

- Scheduling
  - Timed Automata
- Optimal Scheduling
  - Priced Timed Automata
- Schedulability Analysis
  - Single Processor
  - Multi Processor
  - Herschel & Planck Satellite Control Software



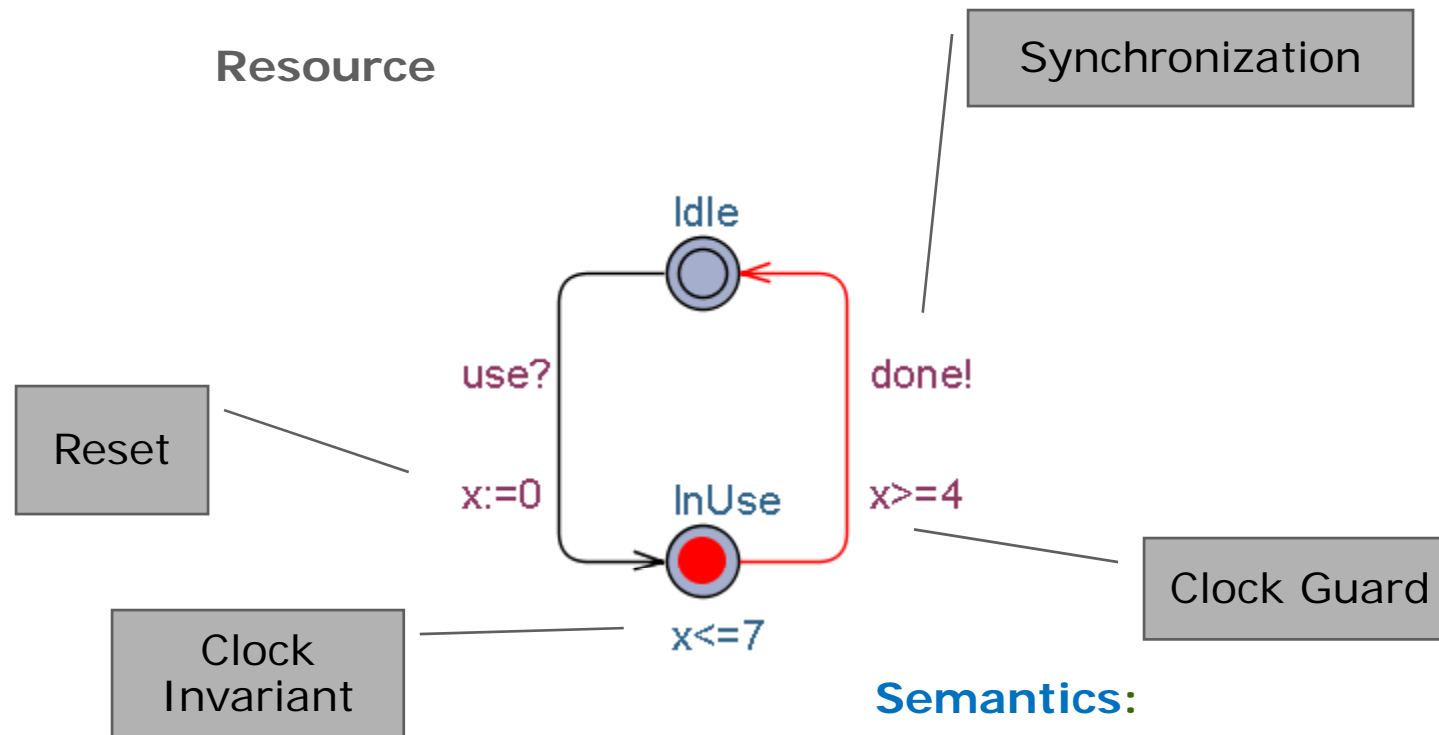
# Scheduling

## Timed Automata



# Timed Automata

[Alur & Dill'89]



## Semantics:

( Idle , x=0 )

d(2.5) → ( Idle , x=2.5 )

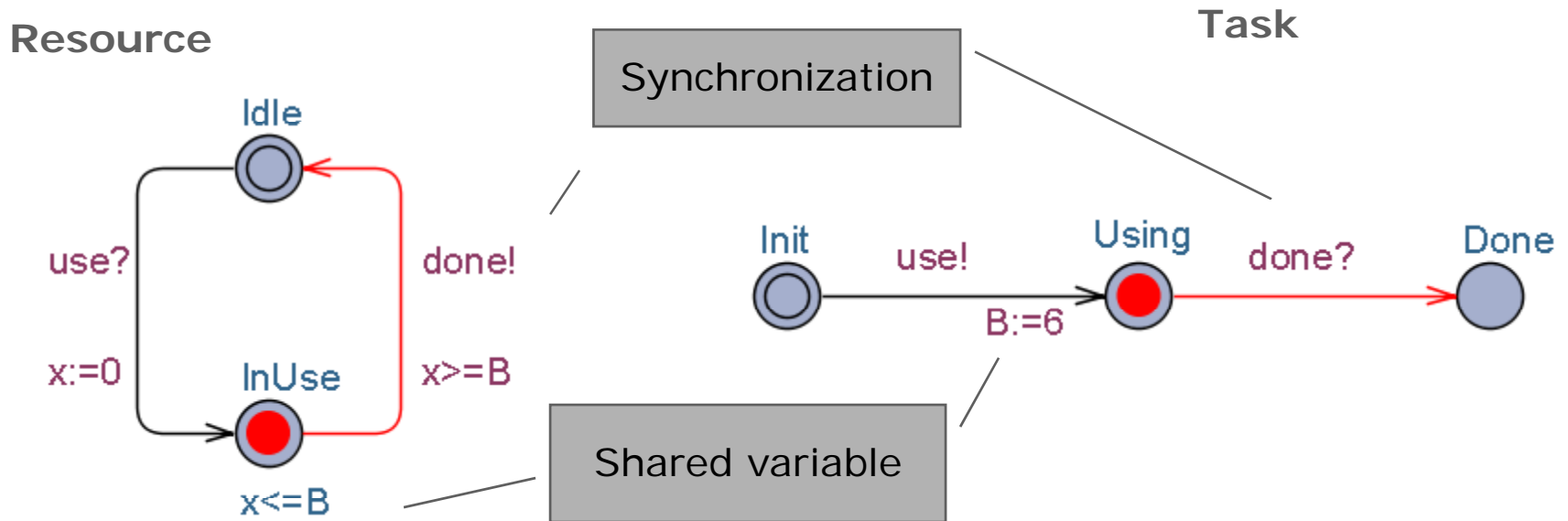
use? → ( InUse , x=0 )

d(5) → ( InUse , x=5 )

done! → ( Idle , x=5 )



# Composition

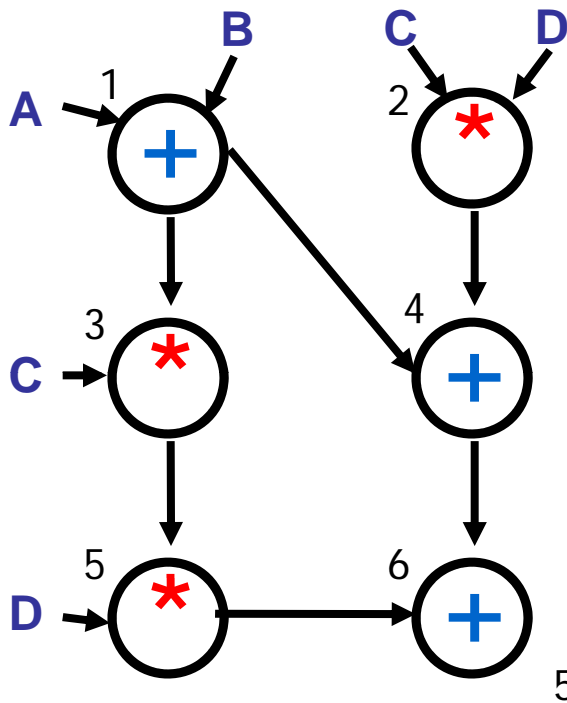


## Semantics:

$( \text{Idle} , \text{Init} , B=0 , x=0 )$   
 $d(3.1415) \rightarrow ( \text{Idle} , \text{Init} , B=0 , x=3.1415 )$   
 $\text{use} \rightarrow ( \text{InUse} , \text{Using} , B=6 , x=0 )$   
 $d(6) \rightarrow ( \text{InUse} , \text{Using} , B=6 , x=6 )$   
 $\text{done} \rightarrow ( \text{Idle} , \text{Done} , B=6 , x=6 )$



# Task Graph Scheduling - Example



Compute :  
 $(D * (C * (A + B))) + ((A + B) + (C * D))$   
 using 2 processors

P1 (fast)

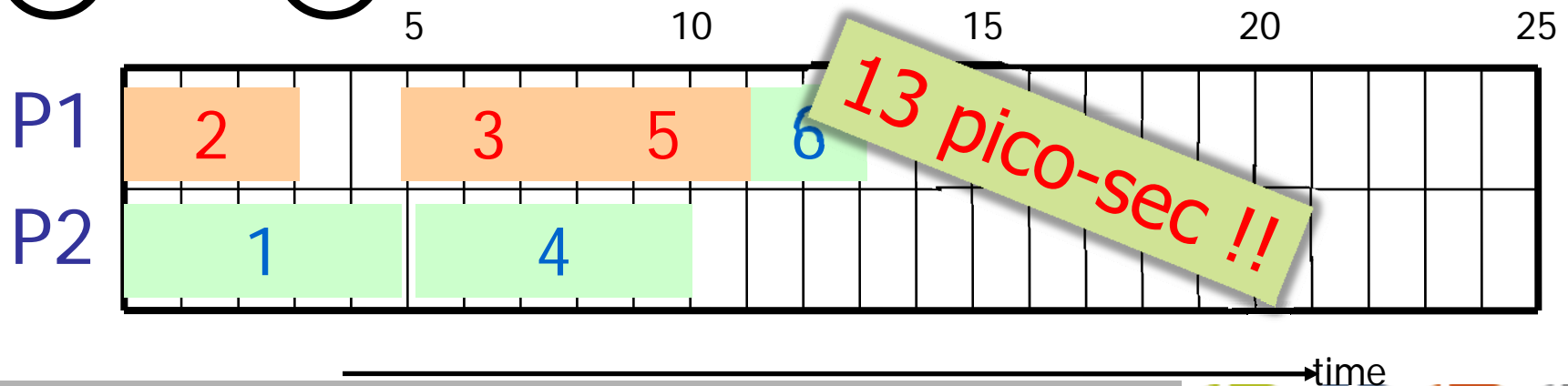
P2 (slow)



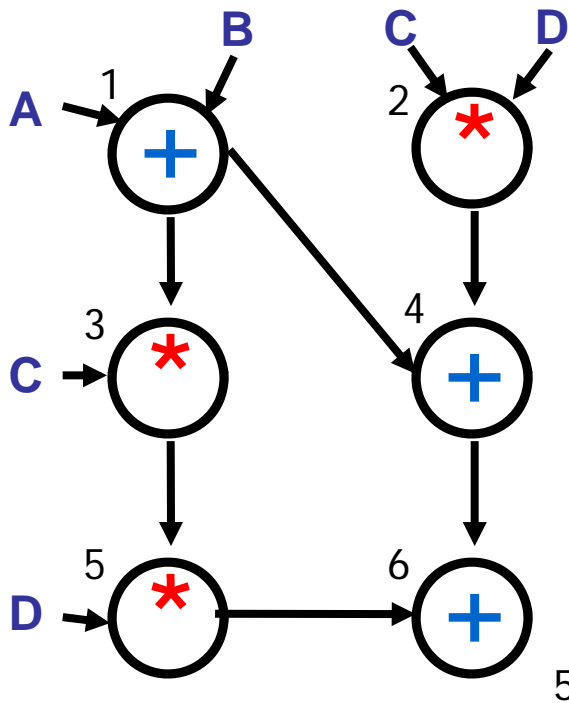
+	2ps
*	3ps



+	5ps
*	7ps



# Task Graph Scheduling - Example



**Compute :**  
 $(D * (C * (A + B)) + ((A + B) + (C * D)))$

using 2 processors

P1 (fast)

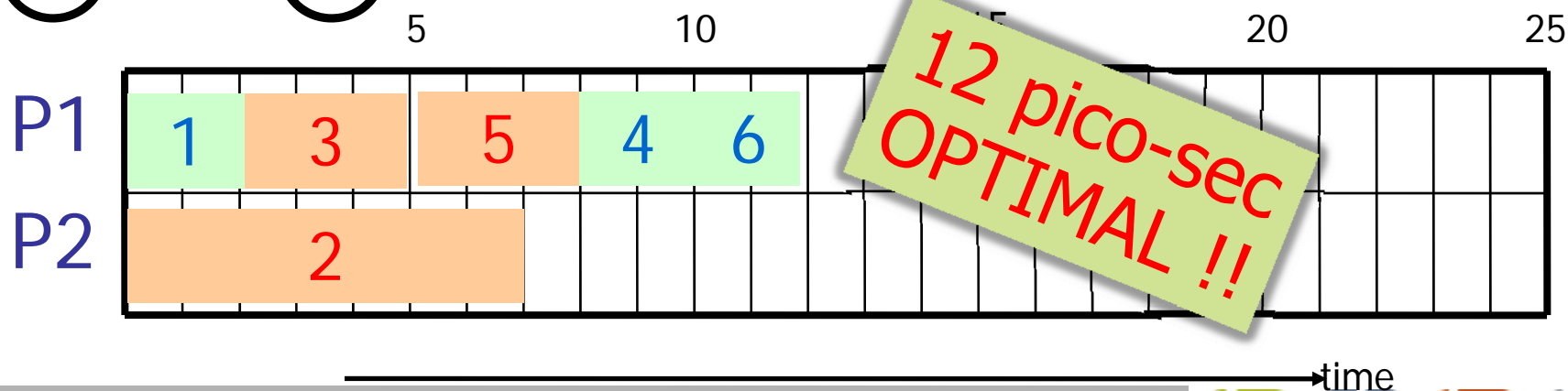
P2 (slow)



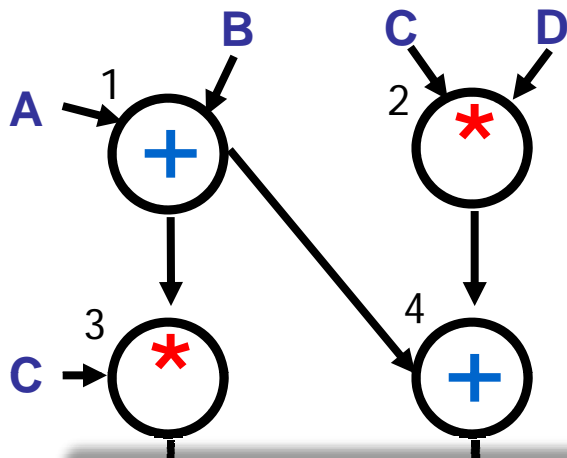
+	2ps
*	3ps



+	5ps
*	7ps

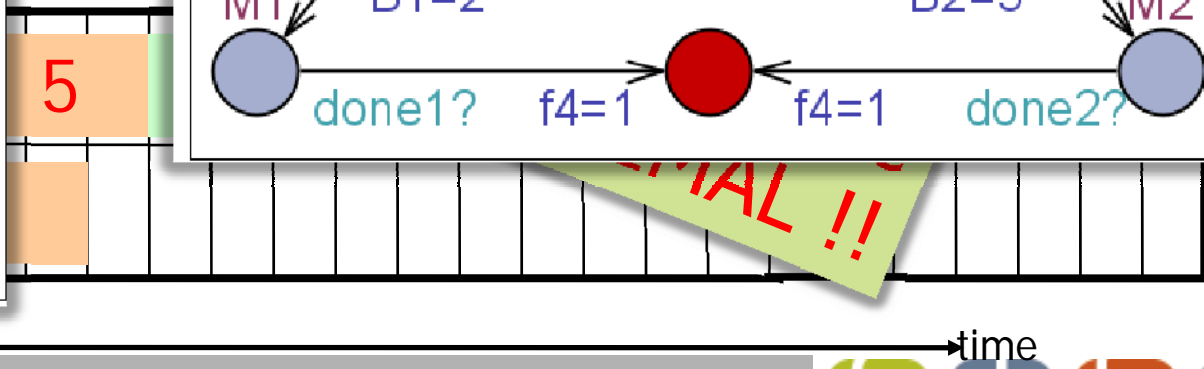
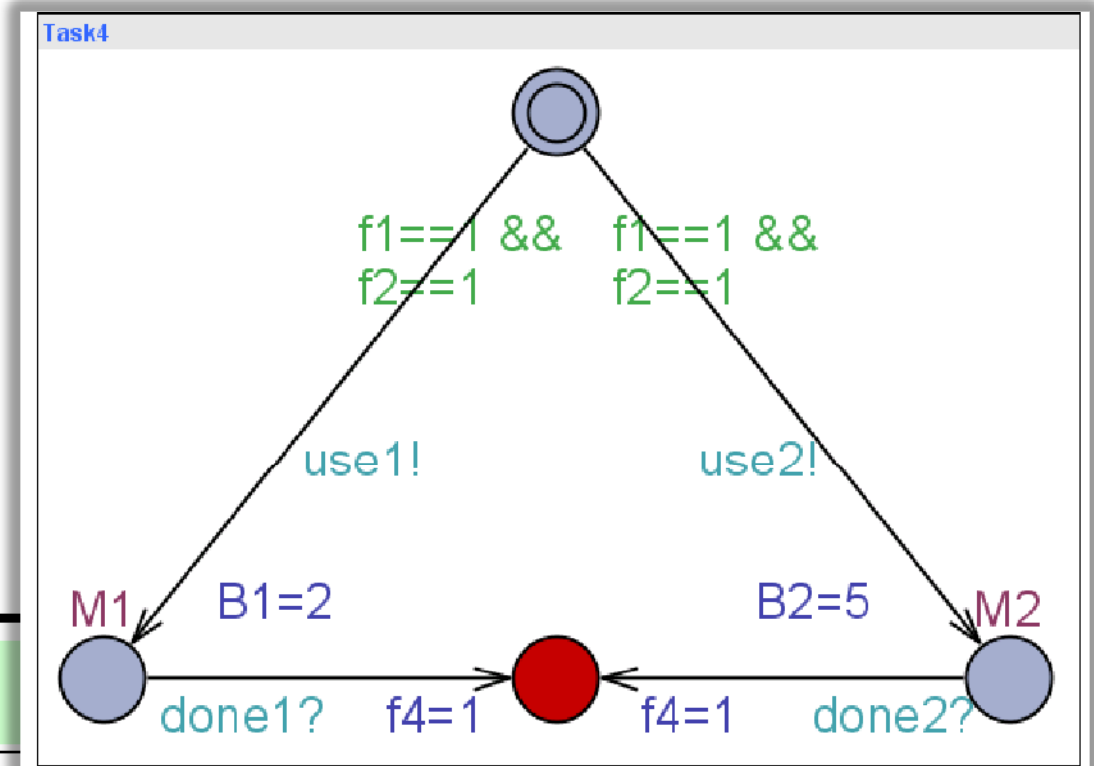
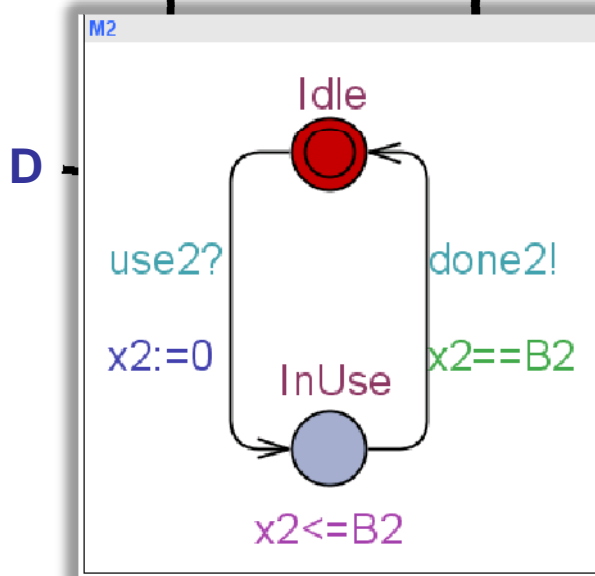


# Task Graph Scheduling - Example

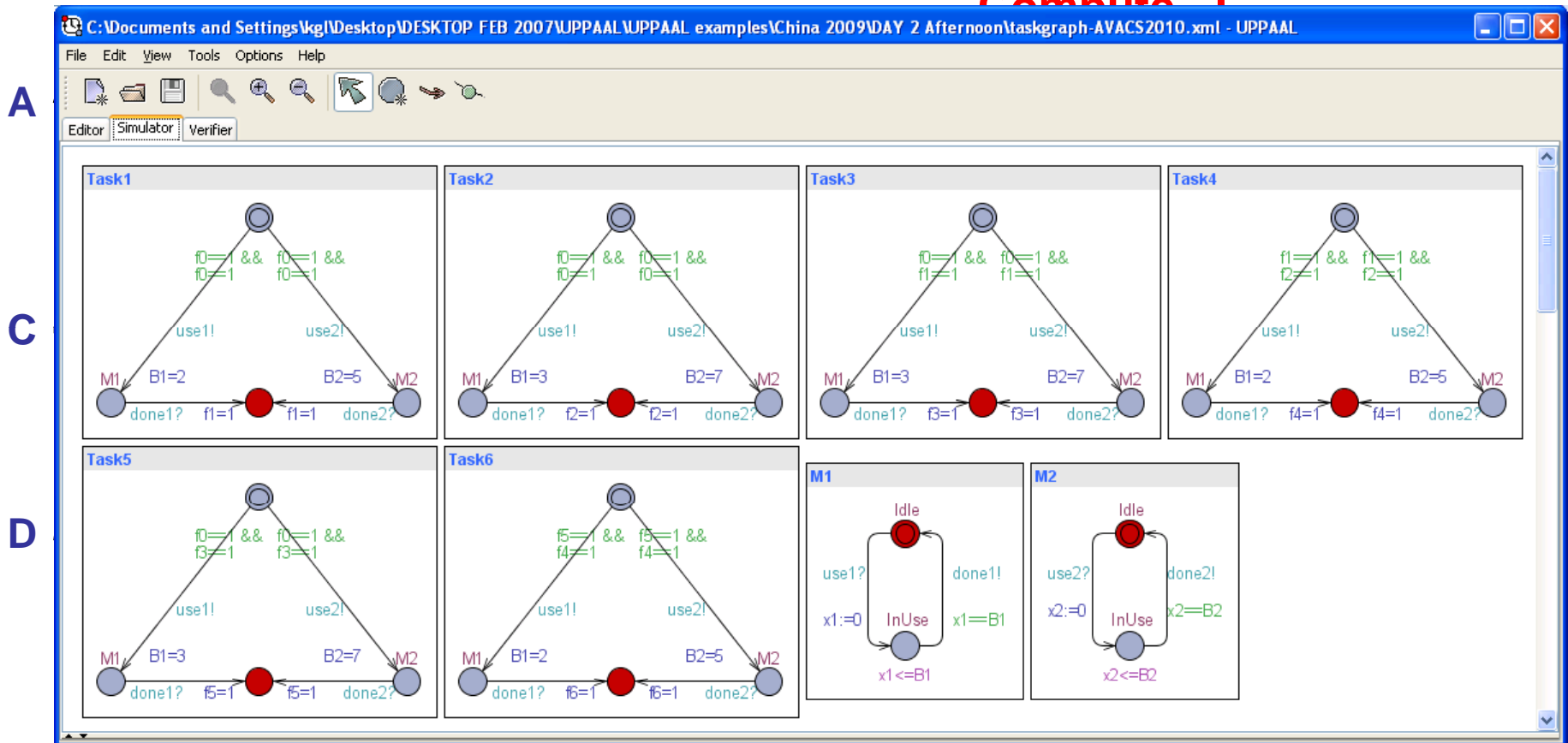


Compute :

$$(D * (C * (A + B)) + ((A + B) + (C * D)))$$



# Task Graph Scheduling - Example



P2

E<> (Task1.End and ... and Task6.End)

time



# Experimental Results

name	#tasks	#chains	# machines	optimal	TA
001	437	125	4	1178	1182
000	452	43	20	537	537
018	730	175	10	700	704
074	1007	66	12	891	894
021	1145	88	20	605	612
228	1187	293	8	1570	1574
071	1193	124	20	629	634
271	1348	127	12	1163	1164
237	1566	152	12	1340	1342
231	1664	101	16	t.o.	1137
235	1782	218	16	t.o.	1150
233	1980	207	19	1118	1121
294	2014	141	17	1257	1261
295	2168	965	18	1318	1322
292	2333	318	3	8009	8009
298	2399	303	10	2471	2473



Symbolic A\*  
Branch-&-Bound  
60 sec

Abdeddaïm, Kerbaa, Maler

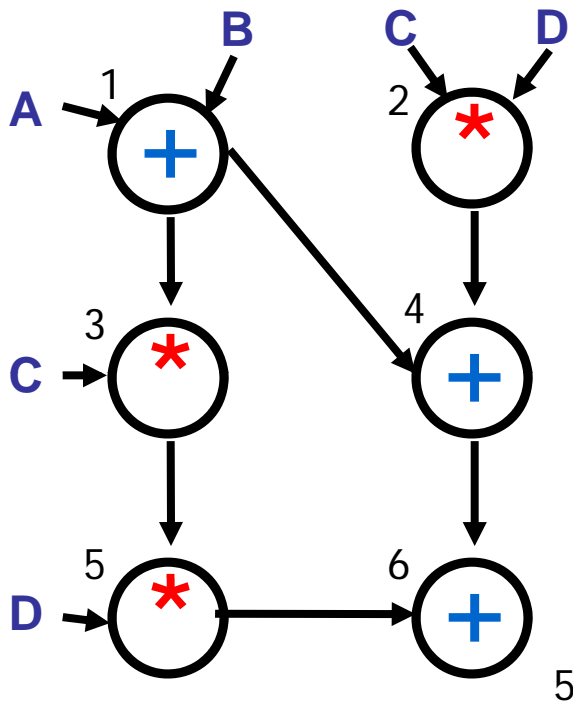


# Optimal Scheduling

## Priced Timed Automata




# Task Graph Scheduling – Revisited



**Compute :**  
 $(D * (C * (A + B)) + ((A + B) + (C * D)))$

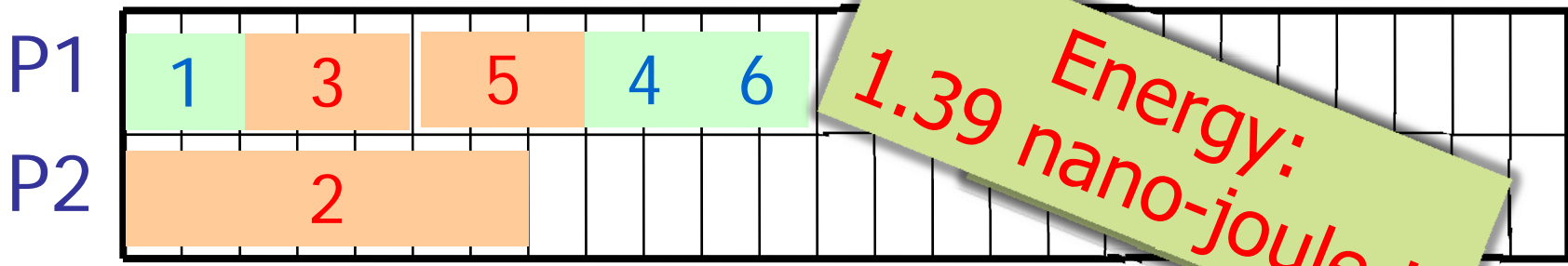
using 2 processors

**P1 (fast)**  **P2 (slow)**

+	2ps	+	5ps
*	3ps	*	7ps

Idle	10W	Idle	20W
In use	90W	In use	30W

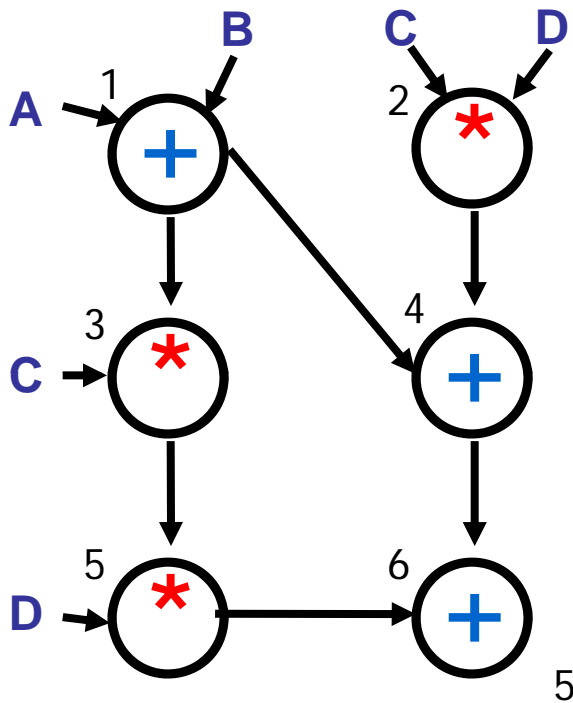
**ENERGY:**  
 10                      20



**Energy: 1.39 nano-joule !!**




# Task Graph Scheduling – Revisited



**Compute :**  
 $(D * (C * (A + B)) + ((A + B) + (C * D)))$

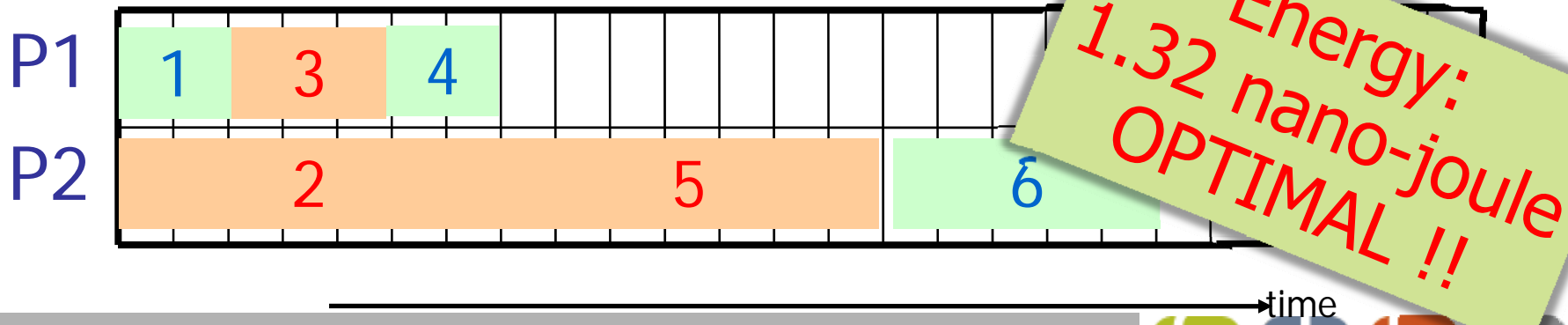
using 2 processors

**P1 (fast)**  **P2 (slow)**

+	2ps	+	5ps
*	3ps	*	7ps

Idle	10W	Idle	20W
In use	90W	In use	30W

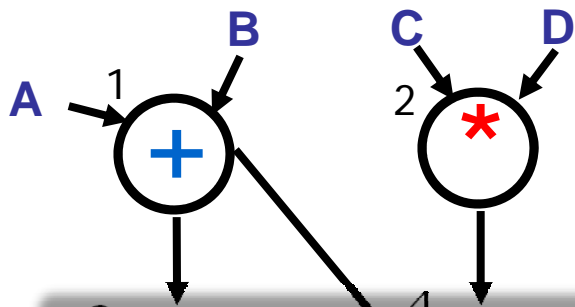
**ENERGY:**  
10



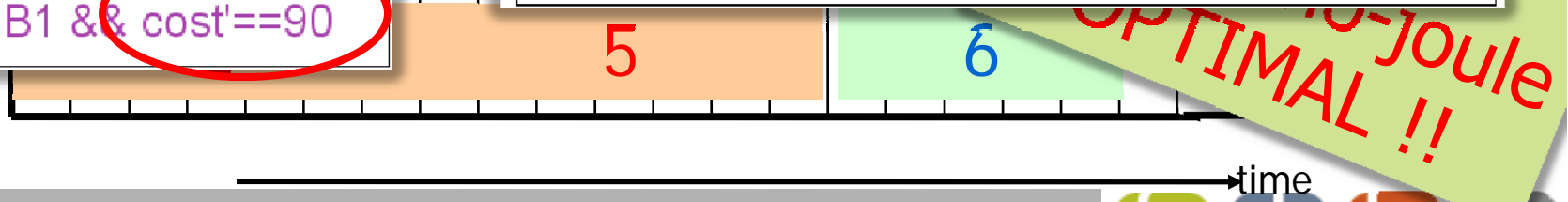
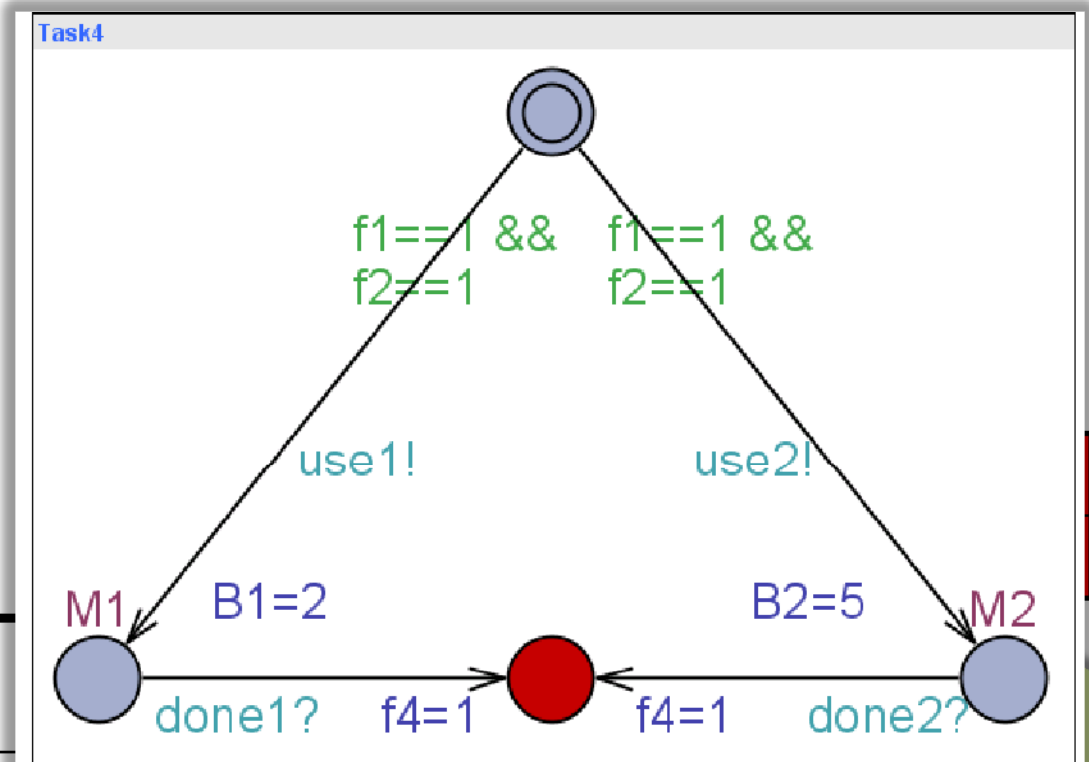
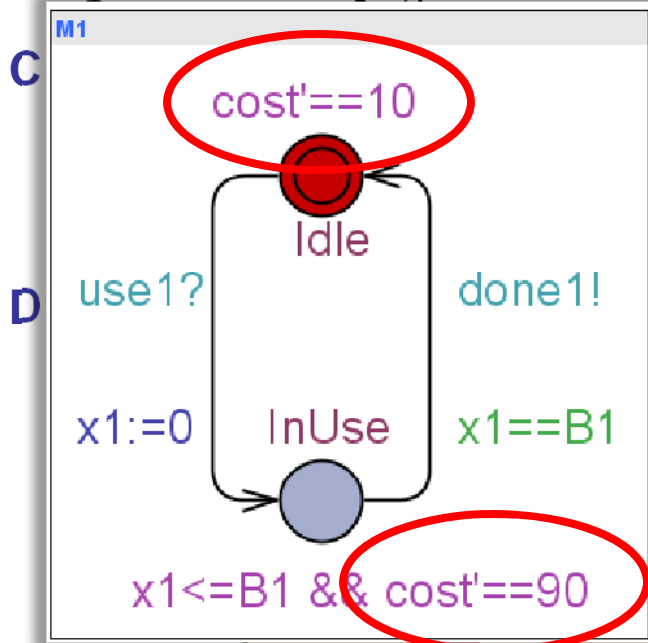
**Energy:**  
**1.32 nano-joule**  
**OPTIMAL !!**



# Task Graph Scheduling - Revisited

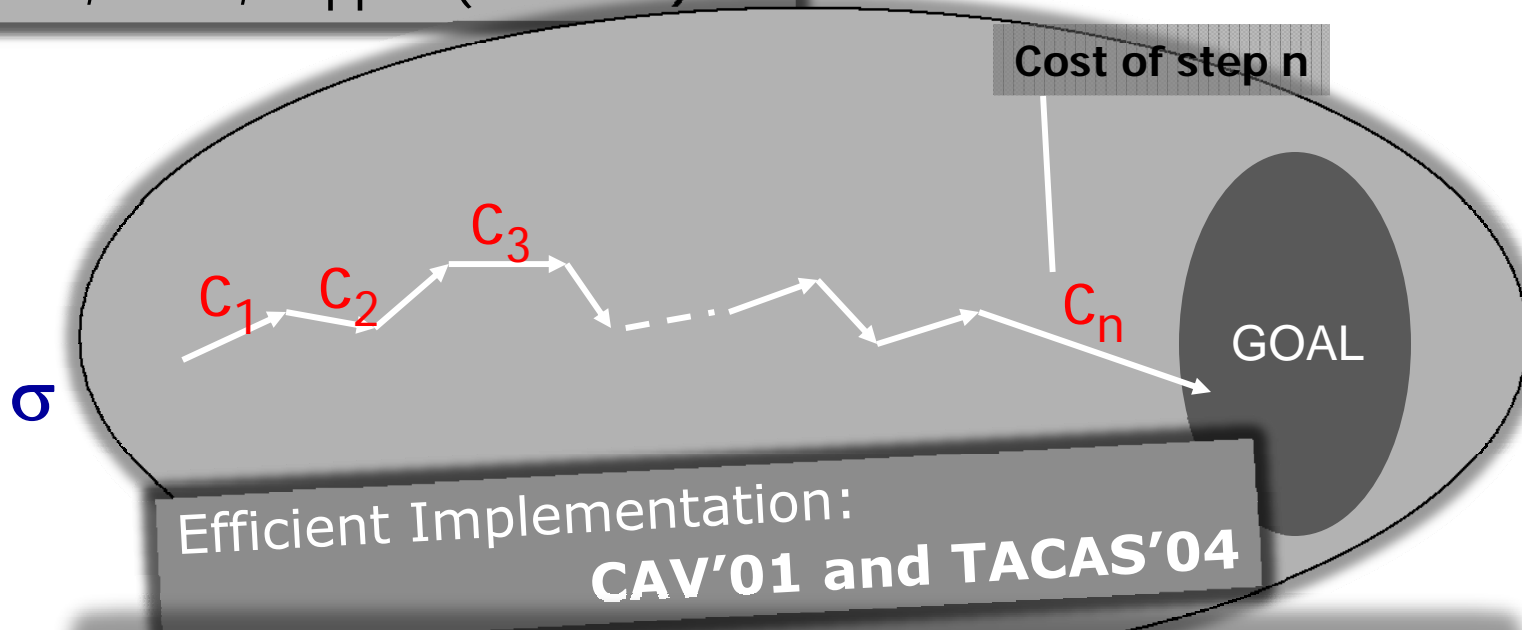


**Compute :**  
 $(D * (C * (A + B)) + ((A + B) + (C * D)))$



# Cost-Optimality Reachability

Behrmann, Fehnker, et al (HSCC'01)  
Alur, Torre, Pappas (HSCC'01)



**Competitive with and Complementary to MILP**

Optimal schedule  $\sigma^*$  :  $\text{val}(\sigma^*) = \inf_{\sigma} \text{val}(\sigma)$



# Multiple Objective Scheduling

The **Pareto Frontier** for Reachability in Multi Priced Timed Automata is computable  
[Larsen&Rasmussen: FoSSaCS05]

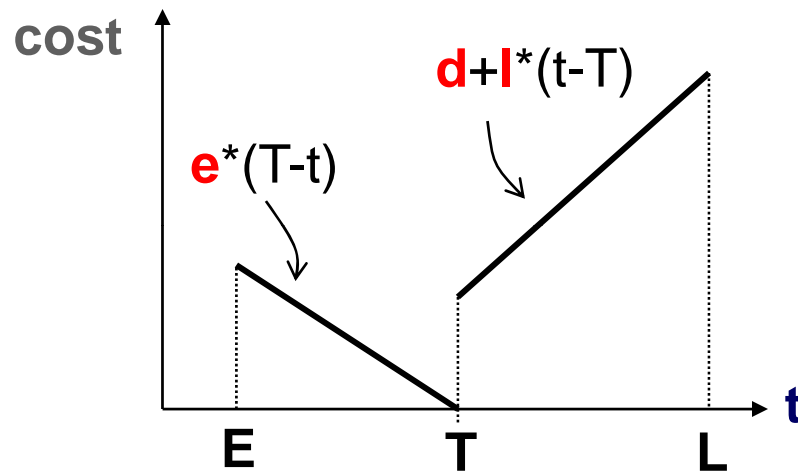
Pareto Frontier

cost<sub>1</sub>

cost<sub>2</sub>



# Aircraft Landing Problem



- E** earliest landing time
- T** target time
- L** latest time
- e** cost rate for being early
- l** cost rate for being late
- d** fixed cost for being late



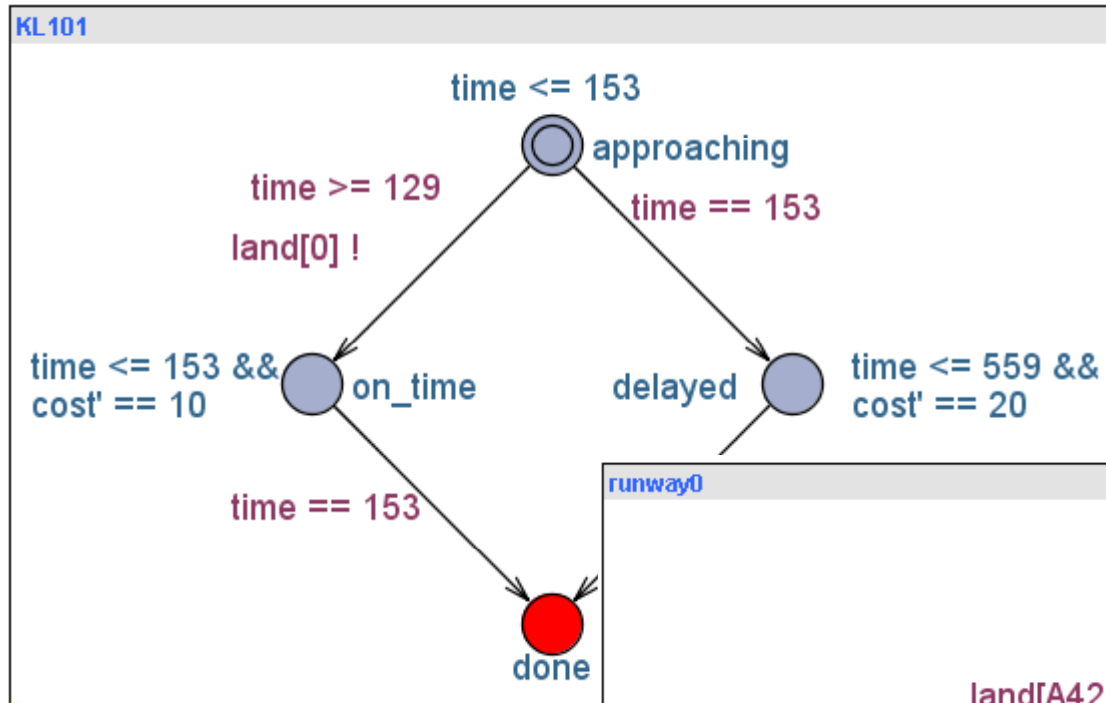
Planes have to keep separation distance to avoid turbulences caused by preceding planes



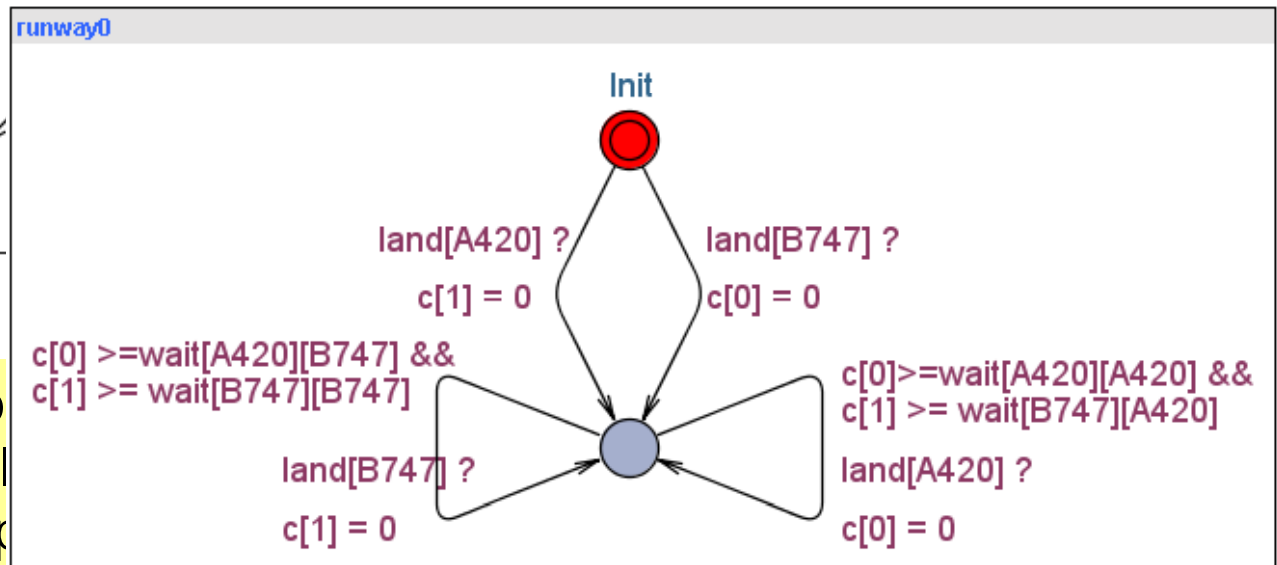
# Modeling ALP with PTA

129: Earliest landing time  
 153: Target landing time  
 559: Latest landing time  
 10: Cost rate for early  
 20: Cost rate for late

Runway handles 2 types of planes



Planes have to keep sep distance to avoid turbul caused by preceding p



# Aircraft Landing

## Comparison with MILP

[Beadsly'2000]

	problem instance	1	2	3	4	5	6	7
	number of planes	10	15	20	20	20	30	44
	number of types	2	2	2	2	2	4	2
1	optimal value	700	1480	820	2520	3100	24442	1550
	explored states	481	2149	920	5693	15069	122	662
	cputime (secs)	4.19	25.30	11.05	87.67	220.22	0.60	4.27
2	optimal value	90	210	60	640	650	554	0
	explored states	1218	1797	669	28821	47993	9035	92
	cputime (secs)	17.87	39.92	11.02	755.84	1085.08	123.72	1.06
3	optimal value	0	0	0	130	170	0	
	explored states	24	46	84	207715	189602	62	N/A
	cputime (secs)	0.36	0.70	1.71	14786.19	12461.47	0.68	
4	optimal value				0	0		
	explored states	N/A	N/A	N/A	65	64	N/A	N/A
	cputime (secs)				1.97	1.53		



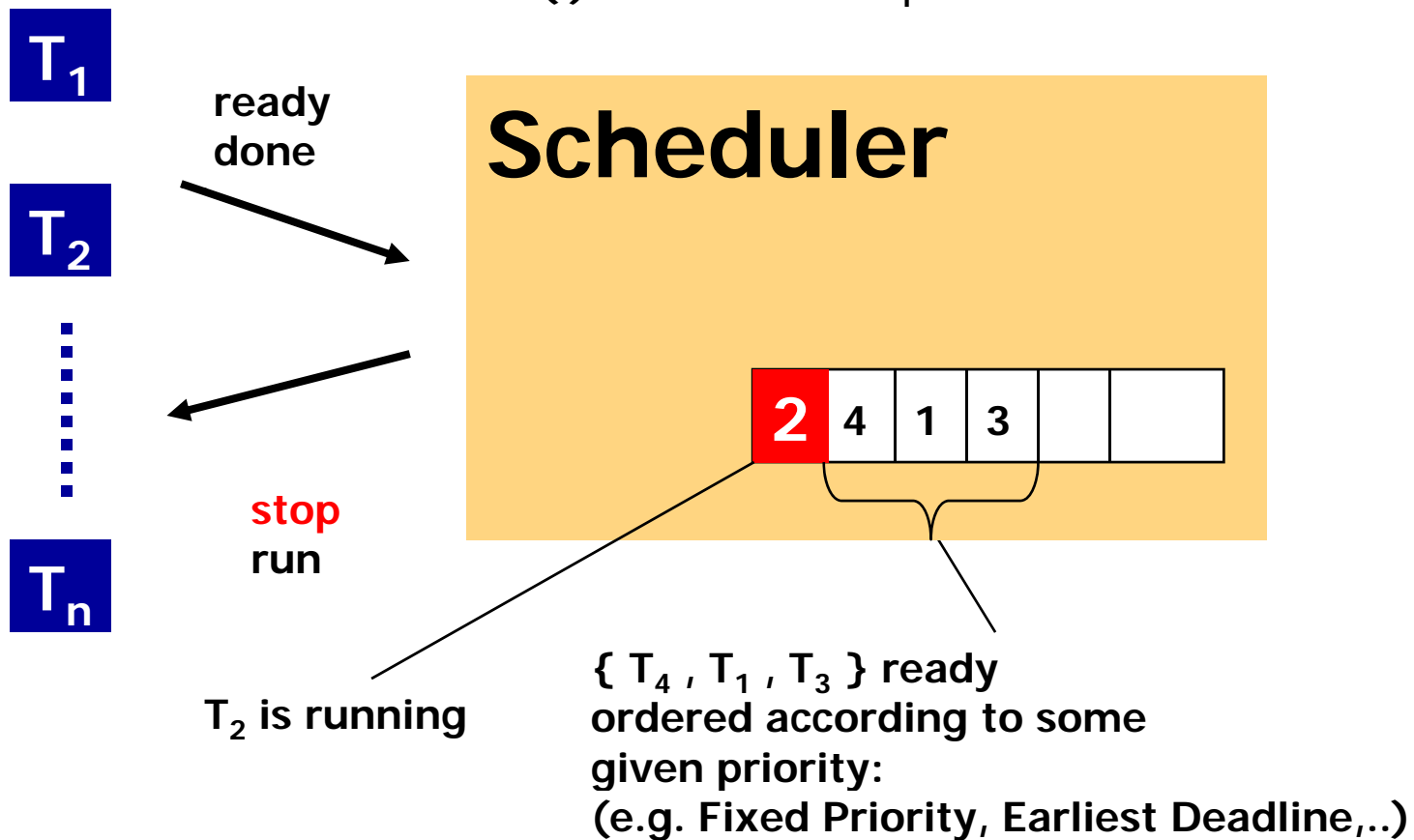
# Schedulability Analysis



# Task Scheduling

*utilization of CPU*

$P(i), [E(i), L(i)], ..$  : period or  
earliest/latest arrival or .. for  $T_i$   
 $C(i)$ : execution time for  $T_i$   
 $D(i)$ : deadline for  $T_i$



# Classical Scheduling Theory

## Utilisation-Based Analysis

- A simple **sufficient but not necessary** schedulability test exists

$$U \equiv \sum_{i=1}^N \frac{C_i}{T_i} \leq N (2^{1/N} - 1)$$

$$U \leq 0.69 \text{ as } N \rightarrow \infty$$

Where C is WCET and T is period

41

## Response Time Equation

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

Where  $hp(i)$  is the set of tasks with priority higher than task  $i$

Solve by forming a recurrence relationship:

$$w_i^{n+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n}{T_j} \right\rceil C_j$$

The set of values  $w_i^0, w_i^1, w_i^2, \dots, w_i^n, \dots$  is monotonically non decreasing  
When  $w_i^n = w_i^{n+1}$  the solution to the equation has been found,  $w_i^0$  must not be greater than  $R_i$  (e.g. 0 or  $C_i$ )

42

## Classical WCRT Analysis



- "Classical" scheduling analysis technique
- For all tasks  $i$ :  $WCRT_i \leq \text{Deadline}_i$

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

Blocking times for priority inheritance protocol (BSW):

- $$\text{Blocking}(i) = \sum_{r=1}^R \text{usage}(r, i) WCET_{\text{CriticalSection}(r)}$$

Blocking times for priority ceiling protocol (ASW):

- $$\text{Blocking}(i) = \max_{r=1}^R \text{usage}(r, i) WCET_{\text{CriticalSection}(r)}$$

Quasimodo Workshop, Eindhoven, Nov 6, 2009

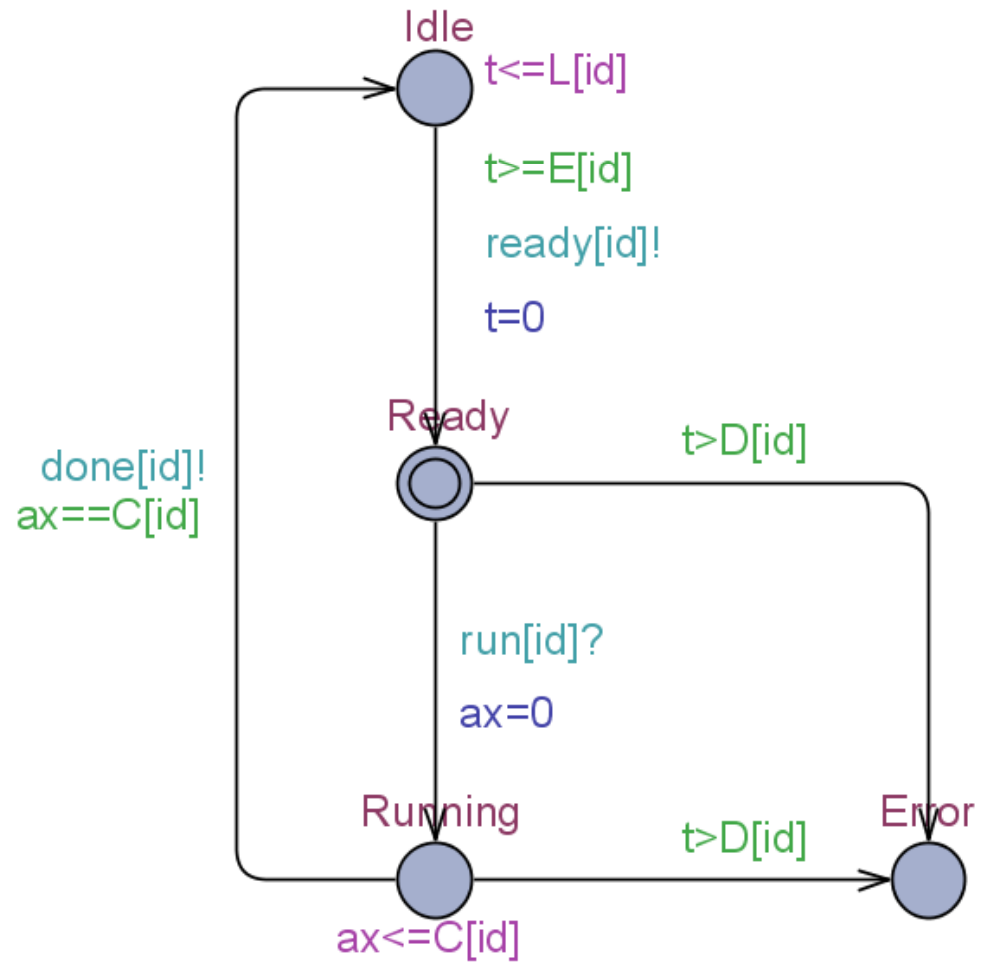
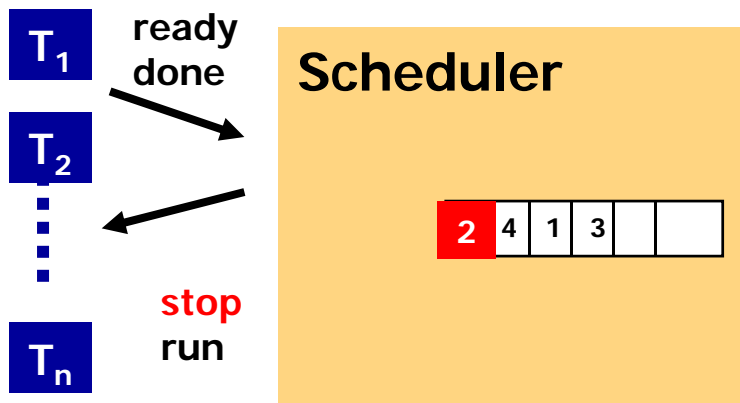
Page 21

✓ Simple to perform

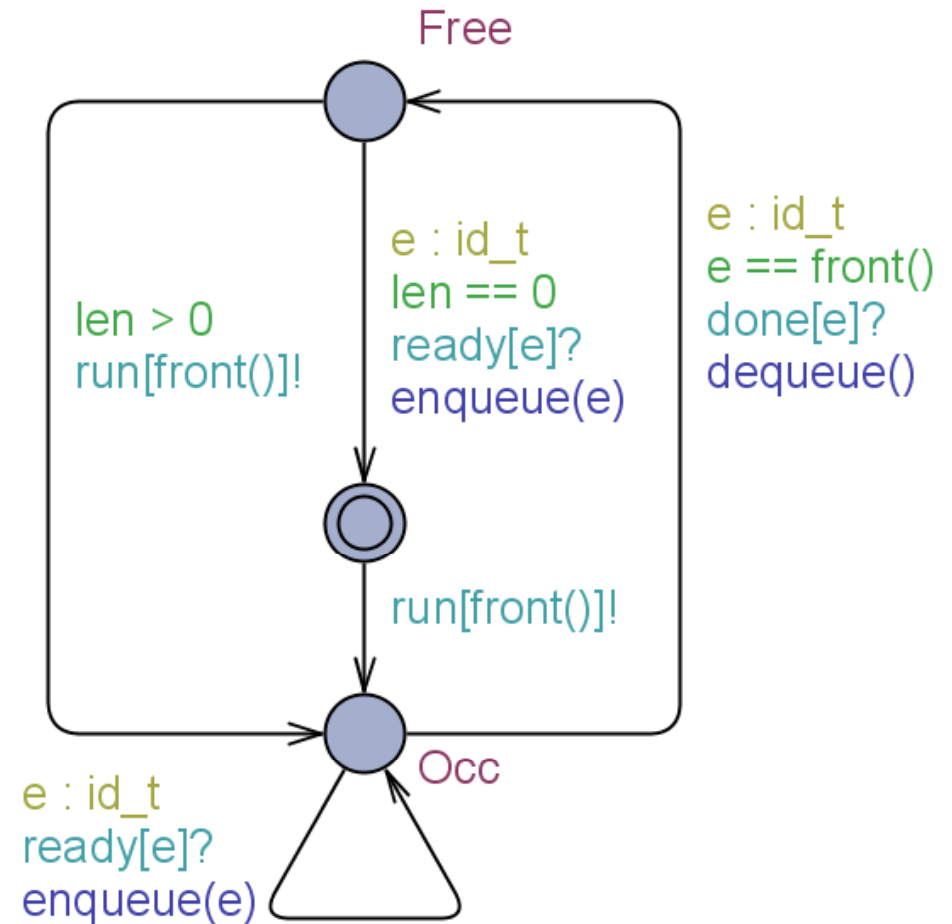
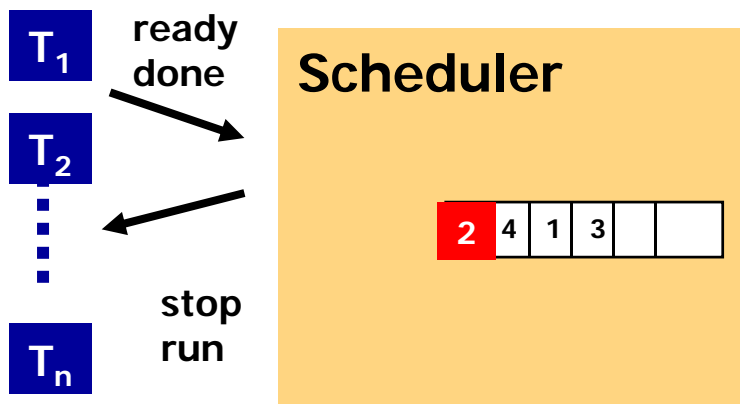
- Overly conservative
- Limited settings
- Single-processor



# Modeling Task

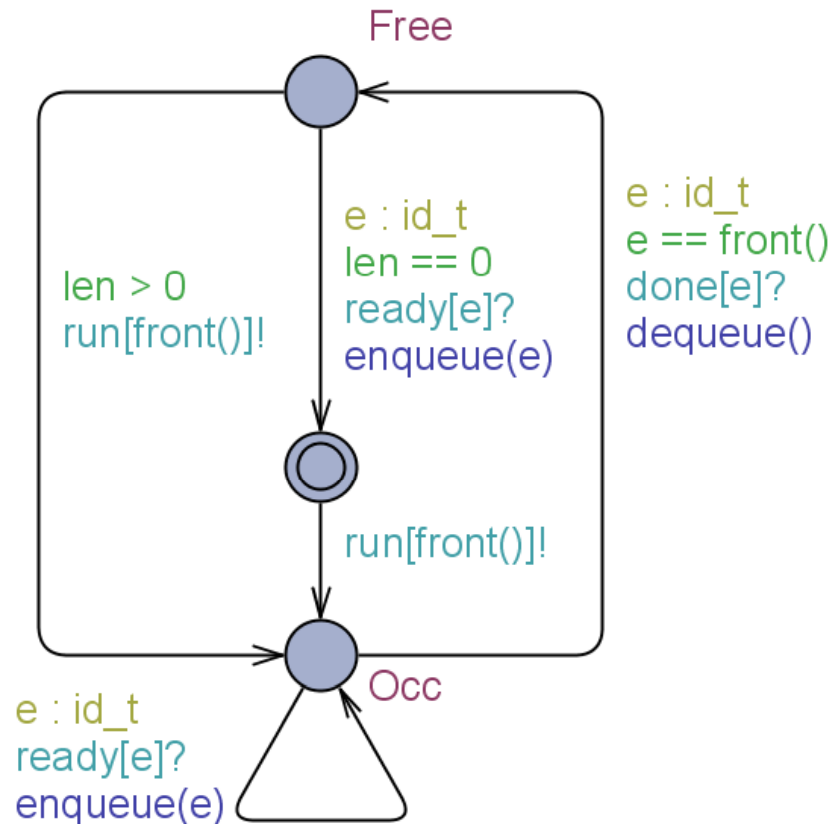


# Modeling Scheduler



# Modeling Queue

In UPPAAL 4.0  
User Defined Function

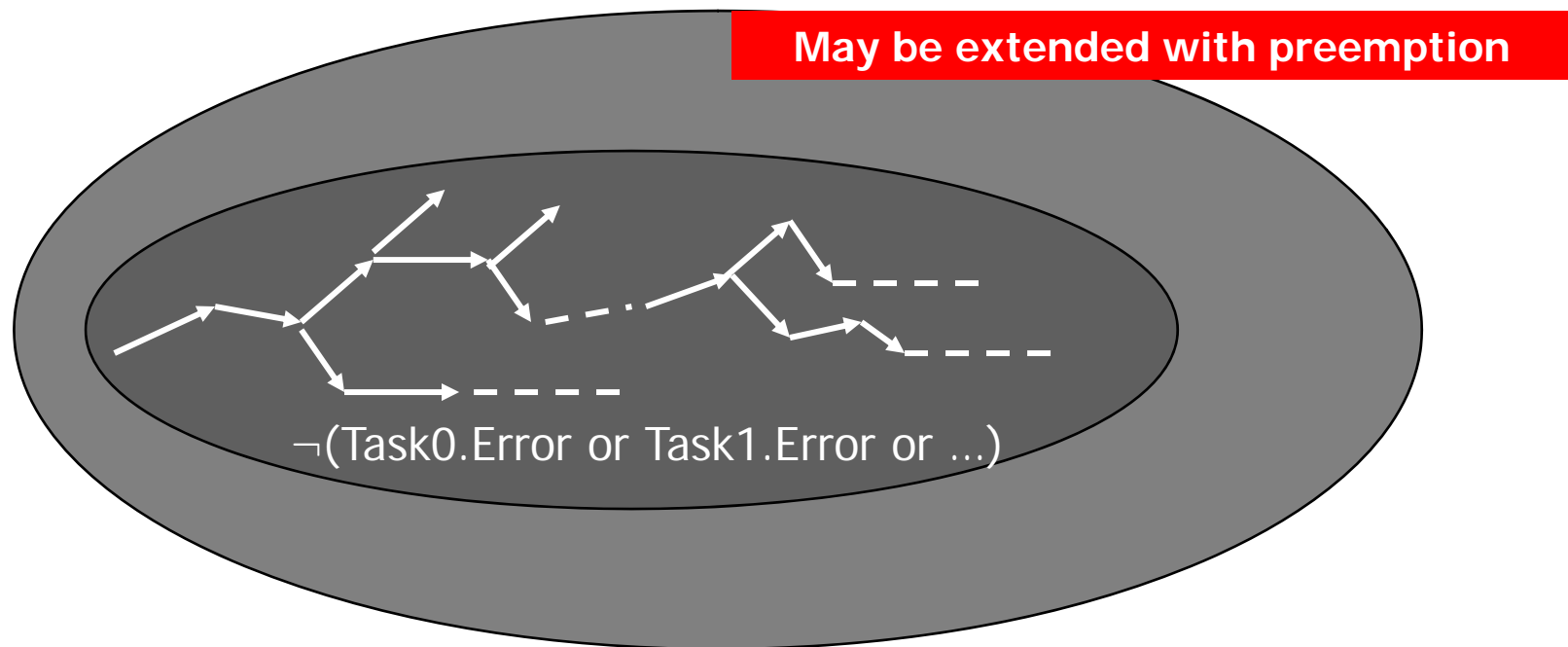


```
// Put an element at the end of the queue
void enqueue(id_t element)
{
  int tmp=0;
  list[len++] = element;
  if (len>0)
  {
    int i=len-1;
    while (i>1 && P[list[i]]>P[list[i-1]])
    {
      tmp = list[i-1];
      list[i-1] = list[i];
      list[i] = tmp;
      i--;
    }
  }
}

// Remove the front element of the queue
void dequeue()
{
  .....
}
```

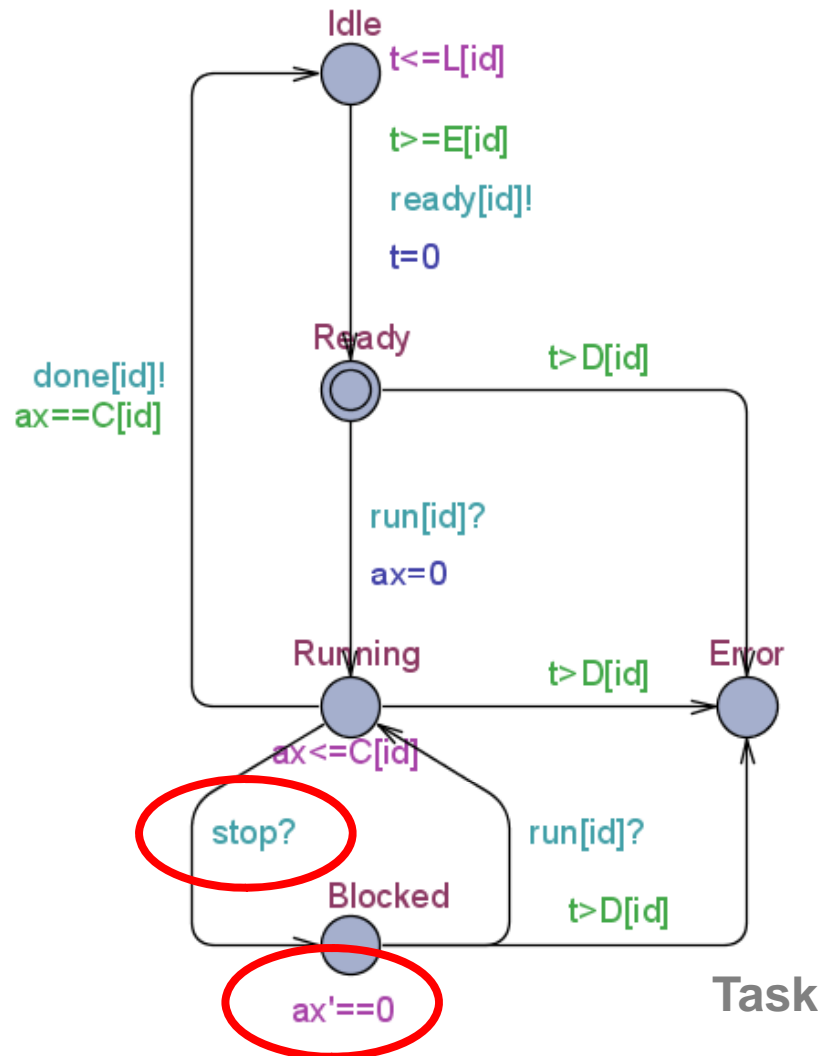


# Schedulability = Safety Property

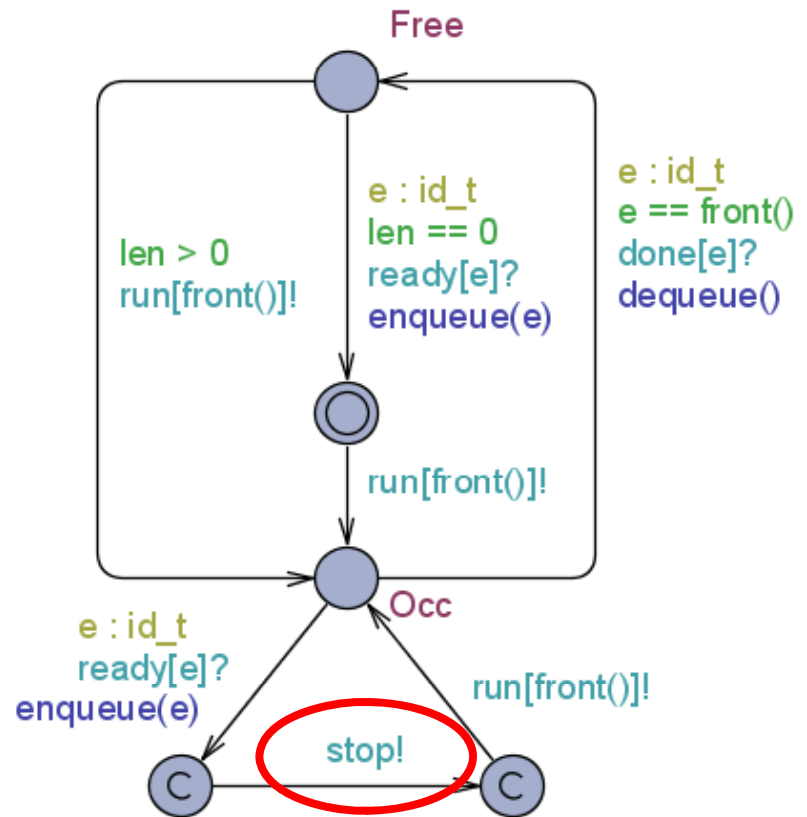


$A \Box \neg(\text{Task0.Error or Task1.Error or ...})$

# Preemption – Stopwatches!



## Scheduler



Defeating undecidability 😊



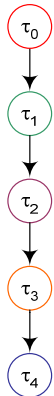
# Handling Realistic Applications?

Jan Madsen / DTU

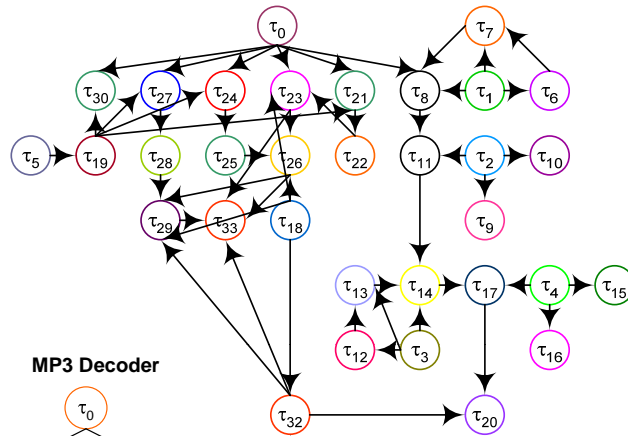
## Smart phone:



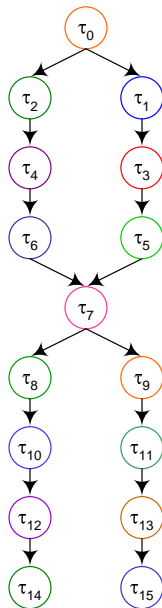
JPEG Encoder



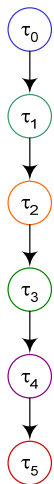
GSM Decoder



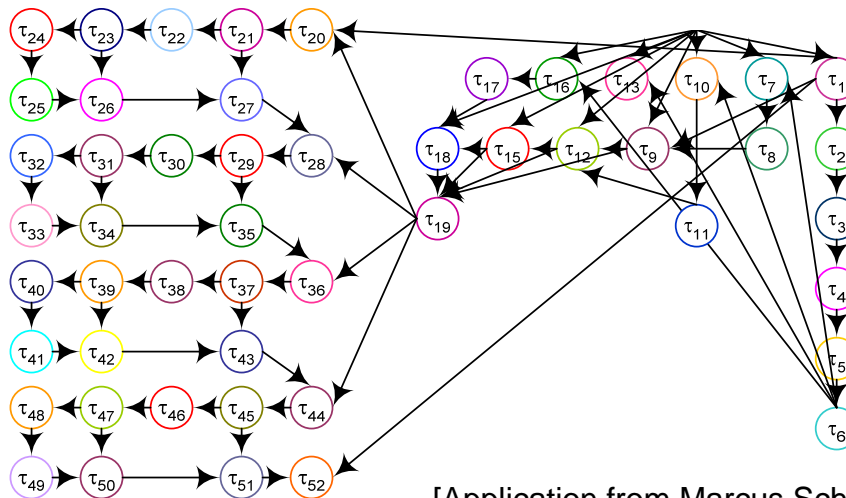
MP3 Decoder



JPEG Decoder



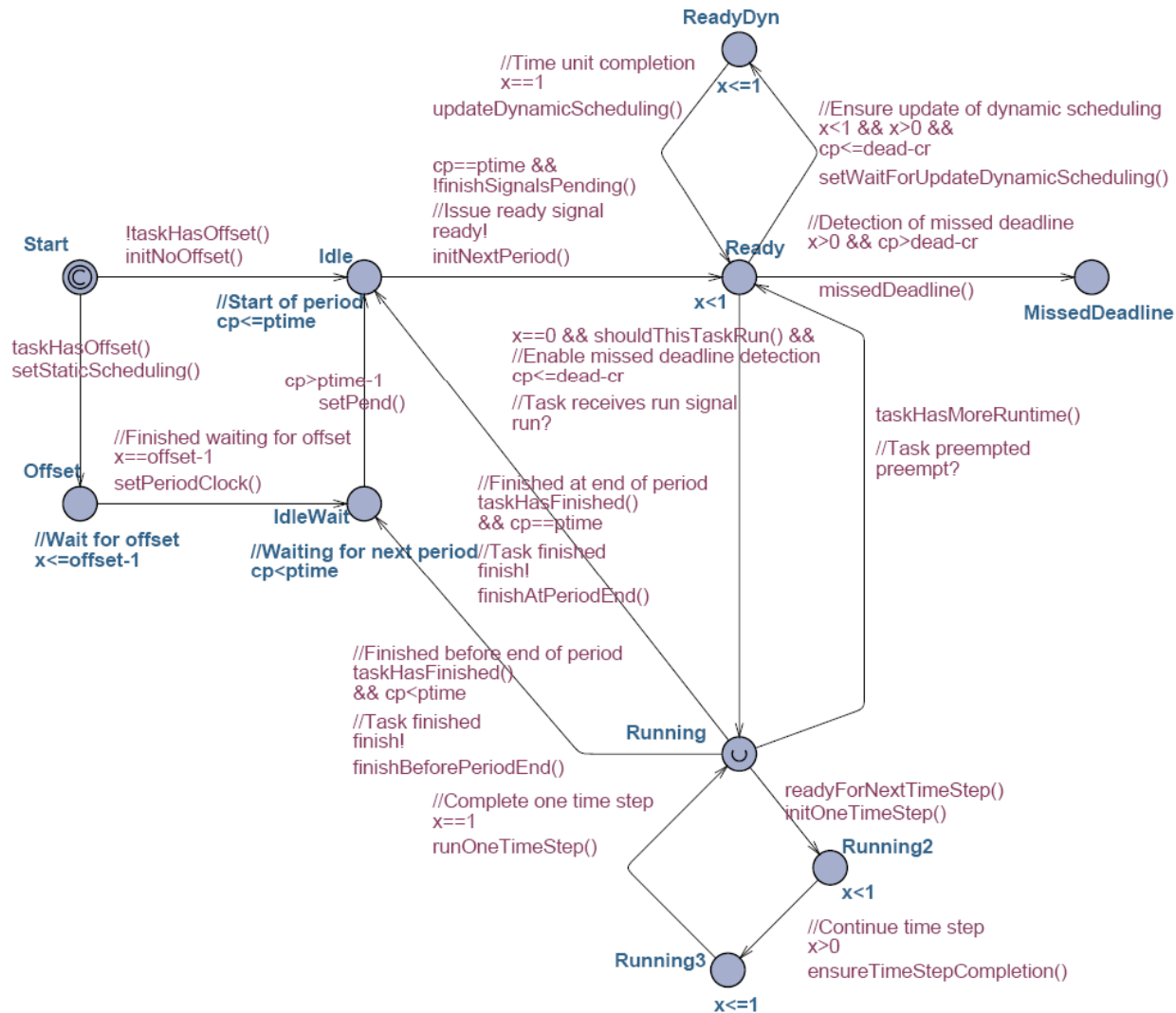
GSM Encoder



[Application from Marcus Schmitz, TU Linköping]

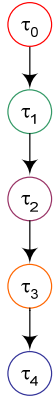


# Timed Automata for a task

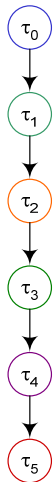


# Smart phone

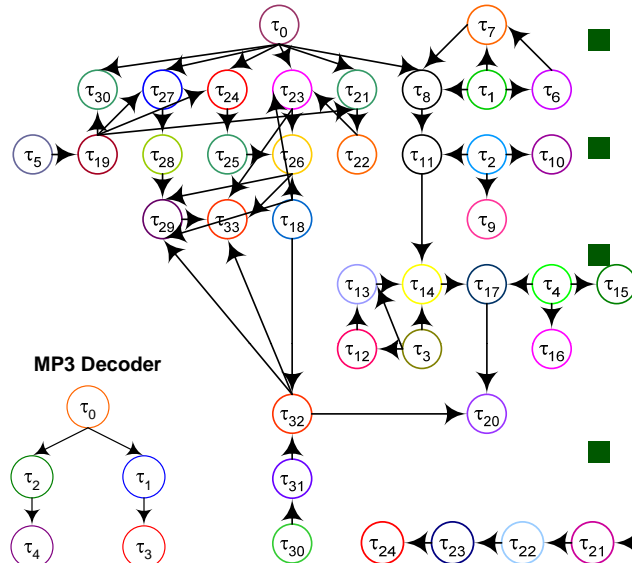
JPEG Encoder



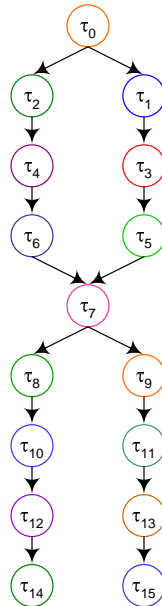
JPEG Decoder



GSM Decoder

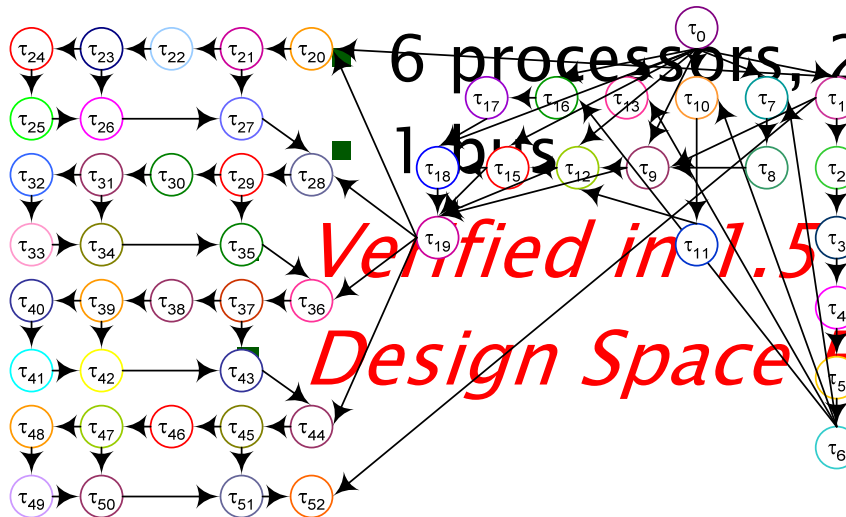


MP3 Decoder



- Tasks: 114
- Deadlines: [0.02: 0.5] sec
- Execution: [52 : 266.687] cycles
- Platform:

GSM Encoder



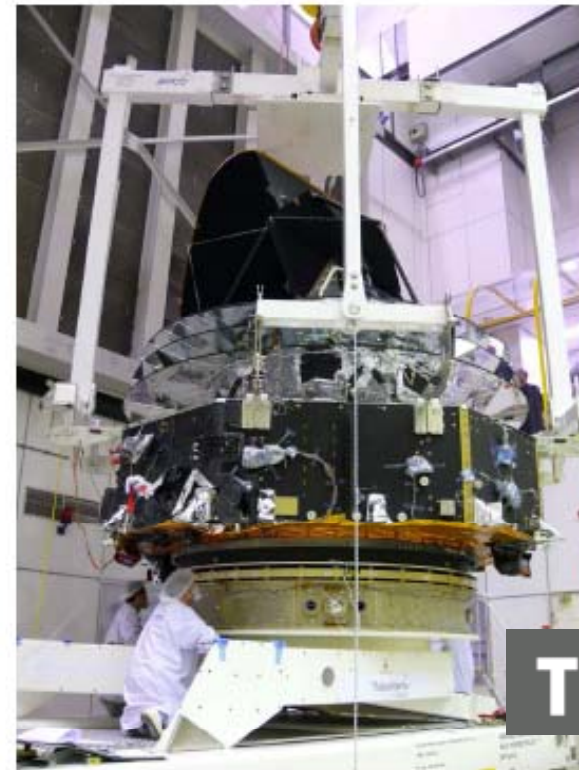
6 processors, 25 MHz

1 bus

*Verified in 1.5 hours!*  
*Design Space Exploration!*



# Herschel & Planck ESA Mission



Subsystem: software for **Attitude and Orbit Control System** – ensures that satellite points to the right direction and is in the correct orbit.



# Satellite Architecture

TERMA<sup>®</sup>

- **Application software (ASW)**
  - built and tested by Terma:
  - does attitude and orbit control, tele-commanding, fault detection isolation and recovery.
- **Basic software (BSW)**
  - low level communication and scheduling periodic events.
- **Real-time operating system (RTEMS)**
  - Priority Ceiling for ASW,
  - Priority Inheritance for BSW
- **Hardware**
  - single processor, a few buses, sensors and actuators

Application Software (ASW)

Basic Software (BSW)

Hardware

## Requirements:

Software tasks should be schedulable.

CPU utilization should not exceed 50% load



# Classical Scheduling

TERMA<sup>®</sup>

System	Herschel		Planck	
Mode	Nominal	+Events	Nominal	+Events
Max Utilization	58.7%	62.4	66.1%	70.8

- *Not* schedulable in one configuration (without harmful effects)
- Worst case utilization too high (> 50%)



- One template for CPU scheduler:
  - maintains a queue of ready tasks
  - schedules tasks with highest priority in the queue
  - reschedule if higher priority task arrives to the queue
- One template per **each** ASW task.
- Two templates for BSW tasks: 1 plain, 1 using resource.
- One template for “idle” task to count CPU **utilization**.
- **WCET** is modeled by stopwatches and lower bound.
- **WCRT** is modeled by stopwatches.
- **Deadline** is enforced as guard on WCRT.
- System is **schedulable** if no deadline violated.
- CPU **load** is  $\frac{\text{used time}}{\text{total time}}$ .



# Modeling in UPPAAL

TERMA<sup>®</sup>

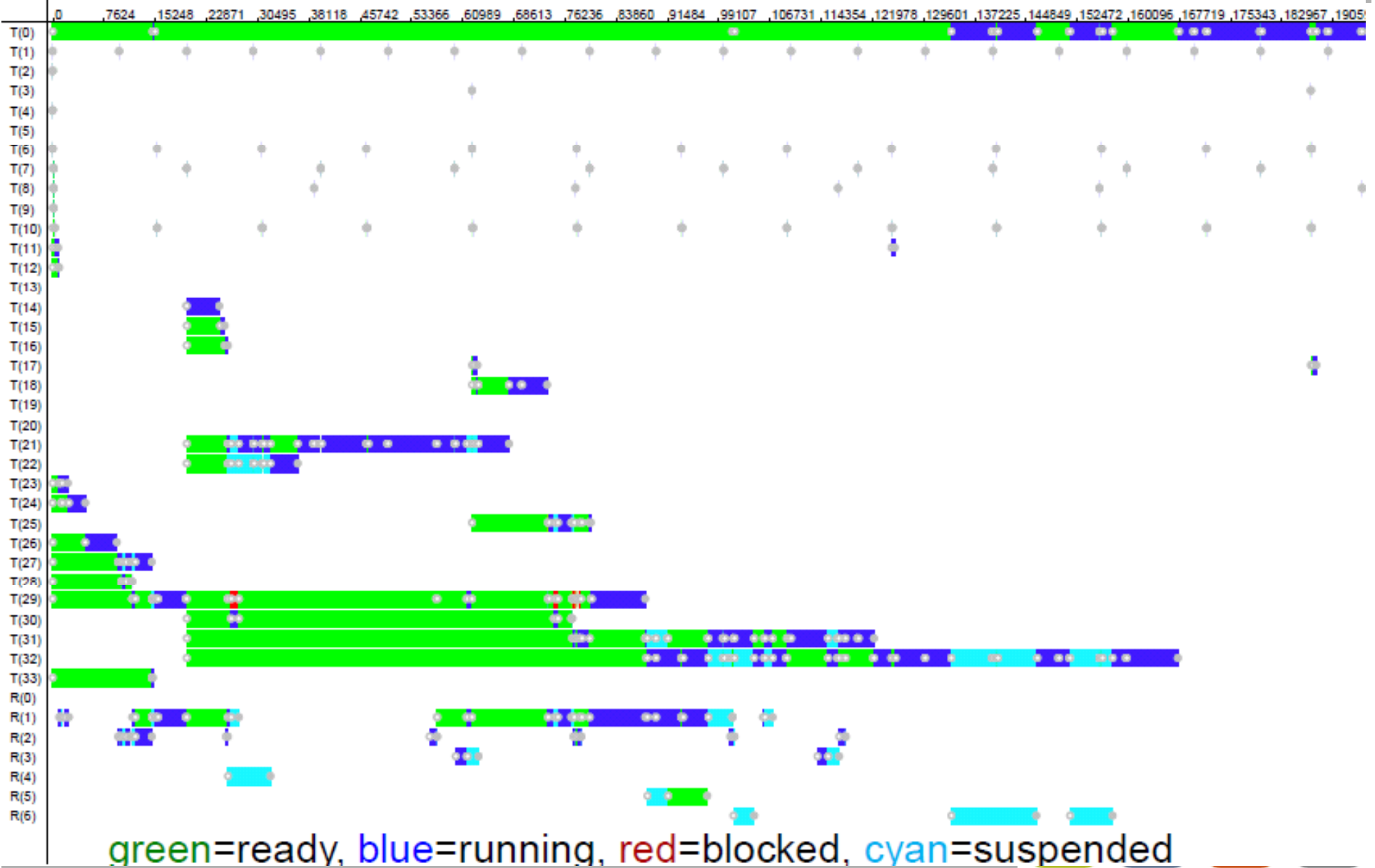
UPPAAL 4.0 Framework, 2009  
Submission ISoLA 2010

The screenshot displays the UPPAAL 4.0 software interface. On the left, there is a 'Transition chooser' with a list of transitions (0.0 to 7.0) and a 'Delay' field set to 13.5. Below it are 'Trace controls' (First, Prev, Play, Next, Last) and a 'Speeder' slider. The 'Simulation Trace' shows a sequence of events: 'initialize: Scheduler --> Bkgnd\_P, NominalE...', 'enqueue: RTEMS\_RTC --> Scheduler', 'schedule: Idle, Idle, Idle, Idle, Idle, Idle, I...', 'preempt[ctask]: Scheduler --> IdleTask', and 'Preempt, Idle, Idle, Idle, Idle, Idle, I...'. The main area contains several state transition diagrams: 'Scheduler' (with states like initialize!, main(), Running, enqueue?, preempt[ctask], cprio[task], cprio[task], and taskqueue[]), 'Bkgnd\_P' (with states like starting, Idle, Ready, and Error), 'secondF\_2' (with states like Idle, Blocked, WaitForCPU, WaitForOther, and Handle Pending TCWthBoth), and 'secondF\_1' (with states like Idle, Blocked, Reschedule, WaitForCPU, DetermineUnit HealthWithSgm\_R, DetermineUnit Health, and DetermineState). The interface also includes a 'Drag out' panel on the left and a 'Drag out' panel at the top.



# Gantt Chart 1. cycle

TERMA<sup>®</sup>



# Blocking & WCRT

TERMA<sup>®</sup>

ID	Task	Specification			Blocking times			WCRT		
		Period	WCET	Deadline	Terma	UPPAAL	Diff	Terma	UPPAAL	Diff
1	RTEMS_RTC	10.000	0.013	1.000	0.035	0	0.035	0.050	0.013	0.037
2	AswSync_SyncPulseIsr	250.000	0.070	1.000	0.035	0	0.035	0.120	0.083	0.037
3	Hk_SamplerIsr	125.000	0.070	1.000	0.035	0	0.035	0.120	0.070	0.050
4	SwCyc_CycStartIsr	250.000	0.200	1.000	0.035	0	0.035	0.320	0.103	0.217
5	SwCyc_CycEndIsr	250.000	0.100	1.000	0.035	0	0.035	0.220	0.113	0.107
6	Rt1553_Isr	15.625	0.070	1.000	0.035	0	0.035	0.290	0.173	0.117
7	Bc1553_Isr	20.000	0.070	1.000	0.035	0	0.035	0.360	0.243	0.117
8	Spw_Isr	39.000	0.070	2.000	0.035	0	0.035	0.430	0.313	0.117
9	Obdh_Isr	250.000	0.070	2.000	0.035	0	0.035	0.500	0.383	0.117
10	RtSdb_P_1	15.625	0.150	15.625	3.650	0	3.650	4.330	0.533	3.797
11	RtSdb_P_2	125.000	0.400	15.625	3.650	0	3.650	4.870	0.933	3.937
12	RtSdb_P_3	250.000	0.170	15.625	3.650	0	3.650	5.110	1.103	4.007
14	FdirEvents	250.000	5.000	230.220	0.720	0	0.720	7.180	5.153	2.027
15	NominalEvents_1	250.000	0.720	230.220	0.720	0	0.720	7.900	5.873	2.027
16	MainCycle	250.000	0.400	230.220	0.720	0	0.720	8.370	6.273	2.097
17	HkSampler_P_2	125.000	0.500	62.500	3.650	0	3.650	11.960	5.380	6.580
18	HkSampler_P_1	250.000	6.000	62.500	3.650	0	3.650	18.460	11.615	6.845
19	Acb_P	250.000	6.000	50.000	3.650	0	3.650	24.680	6.473	18.207
20	IoCyc_P	250.000	3.000	50.000	3.650	0	3.650	27.820	9.473	18.347
21	PrimaryF	250.000	34.050	59.600	5.770	0.966	4.804	65.470	54.115	11.355
22	RCSControlF	250.000	4.070	239.600	12.120	0	12.120	76.040	53.994	22.046
23	Obt_P	1000.000	1.100	100.000	9.630	0	9.630	74.720	2.503	72.217
24	Hk_P	250.000	2.750	250.000	1.035	0	1.035	6.800	4.953	1.847
25	StsMon_P	250.000	3.300	125.000	16.070	0.822	15.248	85.050	17.863	67.187
26	TmGen_P	250.000	4.860	250.000	4.260	0	4.260	77.650	9.813	67.837
27	Sgm_P	250.000	4.020	250.000	1.040	0	1.040	18.680	14.796	3.884
28	TcRouter_P	250.000	0.500	250.000	1.035	0	1.035	19.310	11.896	7.414
29	Cmd_P	250.000	14.000	250.000	26.110	1.262	24.848	114.920	94.346	20.574
30	NominalEvents_2	250.000	1.780	230.220	12.480	0	12.480	102.760	65.177	37.583
31	SecondaryF_1	250.000	20.960	189.600	27.650	0	27.650	141.550	110.666	30.884
32	SecondaryF_2	250.000	39.690	230.220	48.450	0	48.450	204.050	154.556	49.494
33	Bkgnd_P	250.000	0.200	250.000	0.000	0	0.000	154.090	15.046	139.044



Marius Micusionis



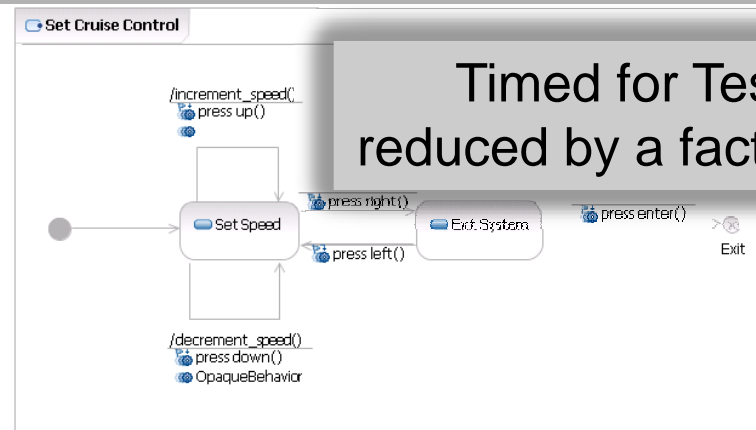
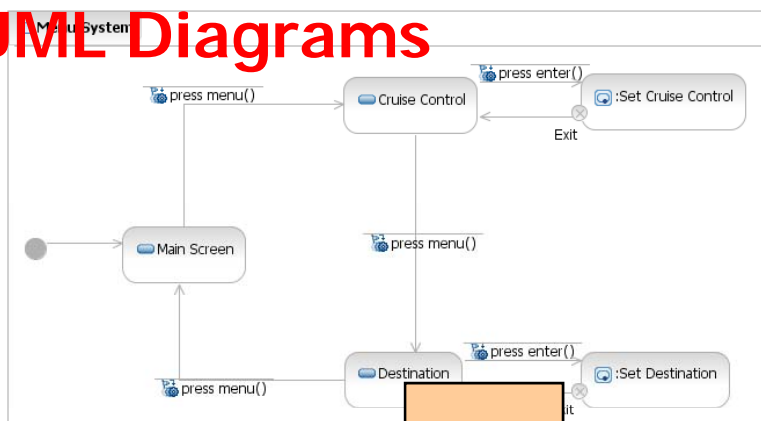
# Schedulability using UPPAAL

- TIMES tool [TACAS 2002]
- Multitasking applications under OSEK [RTS 2008]
- CREOL Modular schedulability [FSEN09]
- SARTS Java byte code on FPGA [JTRES08]
- Schedulability Analysis Using UPPAAL 4.1  
[Model-Based Design for ES, CRC Press 2010]
- ARTS: MPSoC Schedulability  
[Model-Based Design for ES, CRC Press 2010]



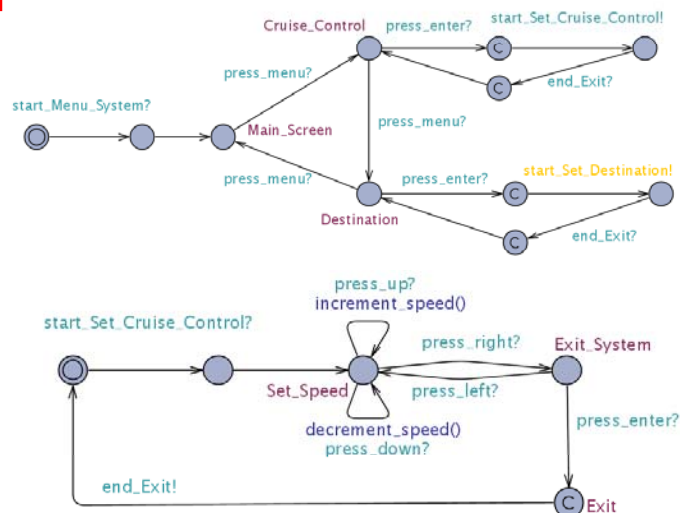
# Model Based Testing

## UML Diagrams



Timed for Testing  
reduced by a factor of 10.

## Uppaal Models



## Test Scripts

```

//Begin Test
assert_view_state('Menu::Main Screen');
user_press(MENU);
assert_view_state('Menu::Cruise Control');
user_press(ENTER);
assert_view_state('CC::Adjust speed');
user_press(DOWN);
user_press(RIGHT);
assert_view_state('CC::Exit Selected');
user_press(ENTER);
assert_view_state('Menu::Cruise Control');
user_press(MENU);
assert_view_state('Menu::Destination');
user_press(MENU);
assert_view_state('Menu::Main Screen');
user_press(MENU);
assert_view_state('Menu::Cruise Control');
user_press(ENTER);
assert_view_state('CC::Adjust speed');
user_press(UP);
assert_variable_value('CC::speed',20);
user_press(DOWN);
assert_variable_value('CC::speed',10);
user_press(UP);
assert_variable_value('CC::speed',20);
user_press(RIGHT);
assert_view_state('CC::Exit Selected');
user_press(LEFT);
assert_view_state('CC::Adjust speed');
//End Test
    
```

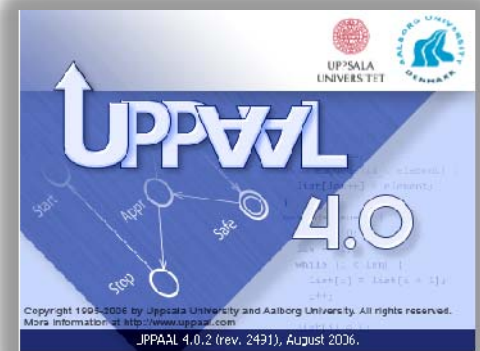


# Conclusion

- TA mature verification technology
- TA mature for schedulability analysis for multiprocessor and -core systems
- Worst Case Execution Time

- Testing **TRON**
- MARTE UML → UPPAAL
- POOSL → UPPAAL
- UPPAAL → Simulink (CIF)

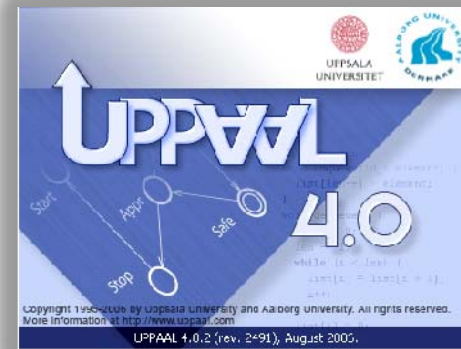
- **Ambition:**  
impact on industrial practice



[www.uppaal.com](http://www.uppaal.com)



Thank you for the attention!



[www.uppaal.com](http://www.uppaal.com)

