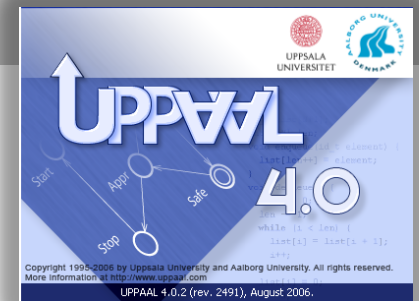


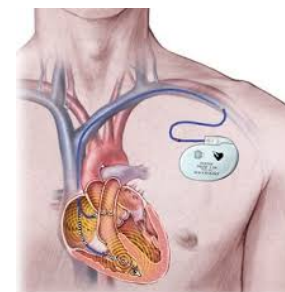
Real-Time Model Checking of Embedded Systems

Kim G. Larsen

CISS – Aalborg University
DENMARK



Embedded Systems



Embedded Systems

Networked ES

IoT
Cloud Computing
Big Data

Cyber-Physical Systems

ES are often Safety Critical



300 horse power
100 processors

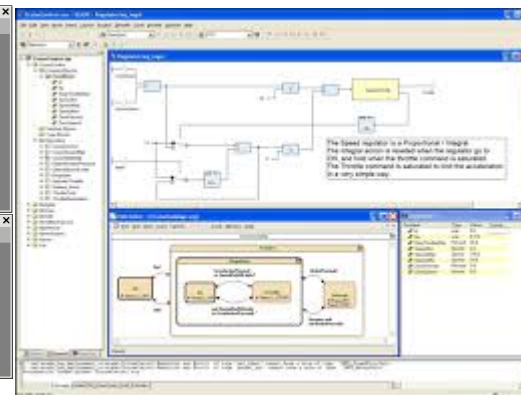
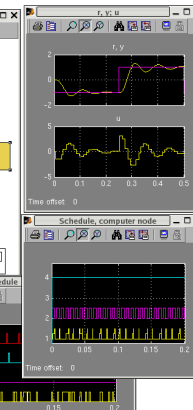
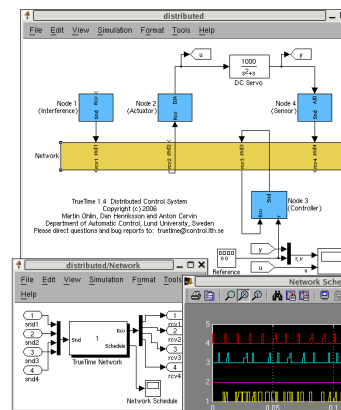
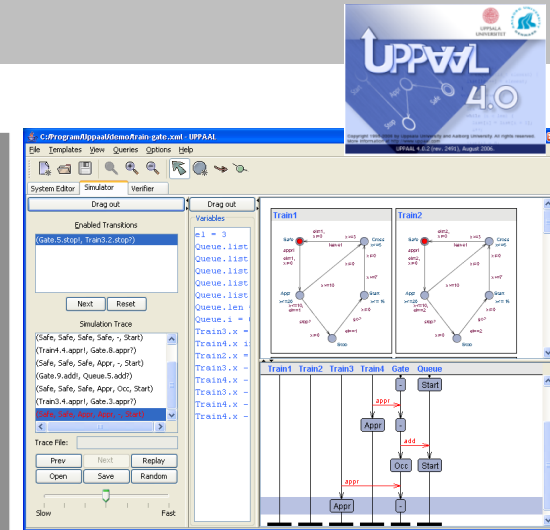
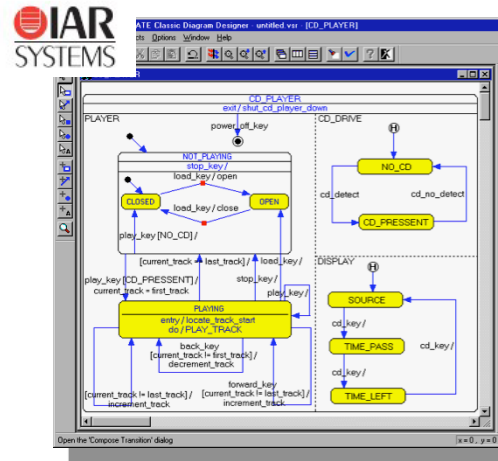
How to achieve ES that are:

- correct
- predicable
- fault tolerant
- timing constraints
- ressource minimal
- cheap
- ..



Model-Driven Development

- High-level designs
- Early design-space exploration
- Early error-detection
- Efficient code generation
- Automatization of testing.
- Verification & synthesis.
- Reduced time-to-market.
- Outsourcing
- Reuse and reconfiguration.



MBAT:

Model-Based Analysis & Test

MBAT will enable the product transportation product

MBAT Partners

- Large Company, technology user: Volvo, ENEA, IBM, Prover, MDH, KTH
- Large Tool Provider: Aalborg University
- SME, technology provider: EADS-IW, Ricardo, Alstom, Rockwell Collins, Airbus, Thales, ENS, CEA, GeenSoft, All4 Tec
- Researcher, technology provider: EADS-DE, Siemens, MBtech, BTC-ES, Pike Tec, Verified, Absint, OFFIS, FH IESE, TUM, AVL, Infineon Austria, AIT, TU Graz, VIF
- National Co-ordinator: Elvlor

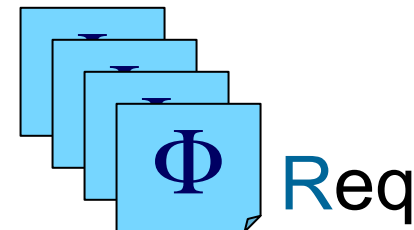
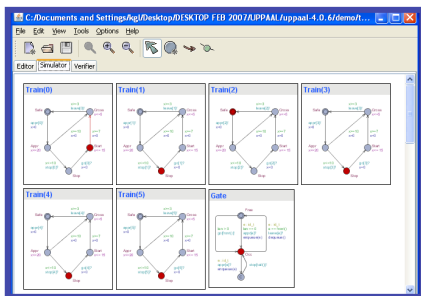
© MBAT Consortium 8 Sept. 2010

MBAT will provide Europe with a new leading-edge *Reference Technology Platform* for effective and cost-reducing Validation and Verification of Embedded Systems

© MBAT Consortium 20 8 Sept. 2010

Model-Driven Development

Model



Code

```
void HandleError(unsigned char ccArg)
{
    printf("Error code %c detected, exiting application.\n", ccArg);
    exit(ccArg);
}

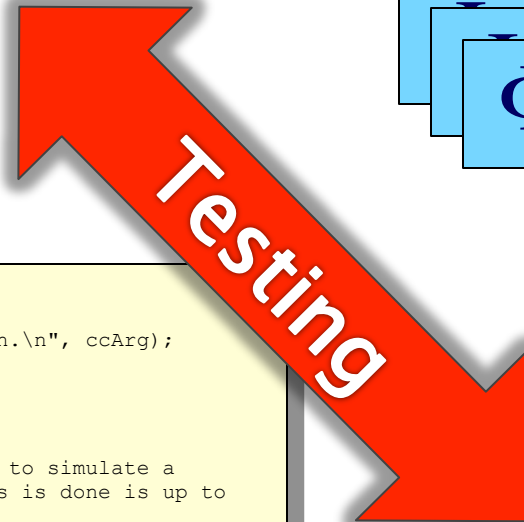
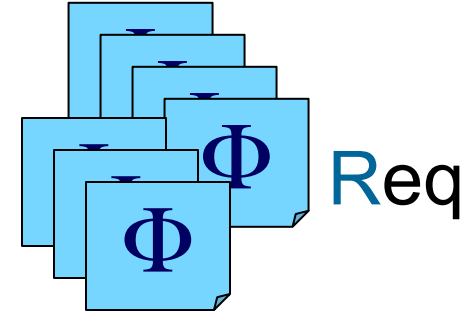
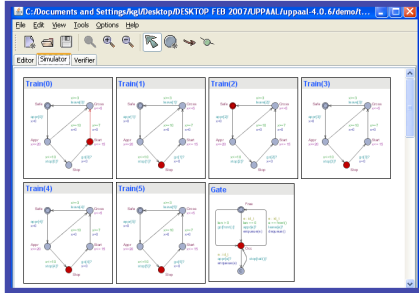
/* In d-241 we only use the OS_Wait call. It is used to simulate a
 * system. Its purpose is to generate events. How this is done is up to
 * you.
 */
void OS_Wait(void)
{
    /* Ignore the parameters; just retrieve events from the keyboard and
     * put them into the queue. When EVENT_UNDEFINED is read from the
     * keyboard, return to the calling process. */
    SEM_EVENT_TYPE event;
    int num;
```



Running System

Testing

Model



Code

```
void HandleError(unsigned char ccArg)
{
    printf("Error code %c detected, exiting application.\n", ccArg);
    exit(ccArg);
}

/* In d-241 we only use the OS_Wait call. It is used to simulate a
 * system. Its purpose is to generate events. How this is done is up to
 * you.
 */
```

Characteristics:

- System-based
- Very rich properties
- (Under) approximate
- Scalable

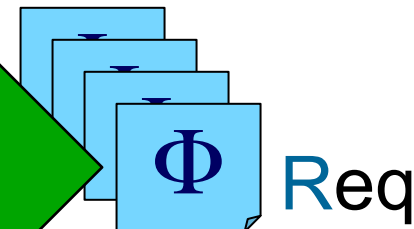
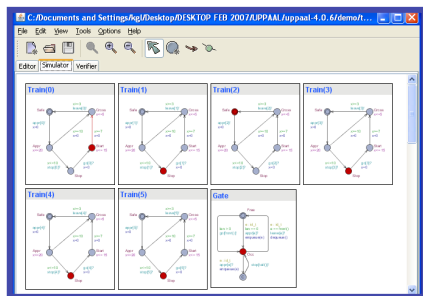
```
ts from the keyboard and
MINED is read from the
*/
```



Running System

Model Checking

Model



```
void HandleError(unsigned char ccArg)
{
    printf("Error code %c detected, exiting application.\n", ccArg);
    exit(ccArg);
}

/* In d-241 we only use the OS_Wait call. It is used to simulate a
... line is up to
```

Code

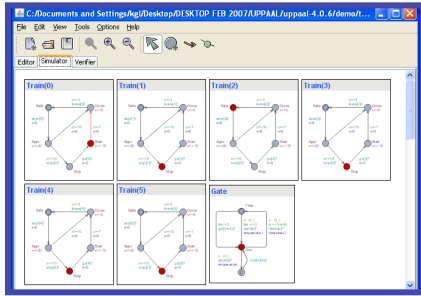
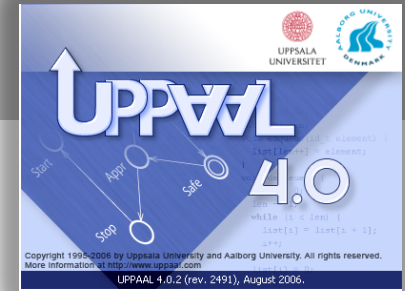
Characteristics:

- Automata-based
- Rich class of properties
- Exact Analysis
- State-space Explosion



Running System

QUANTITATIVE Model Checking



Time

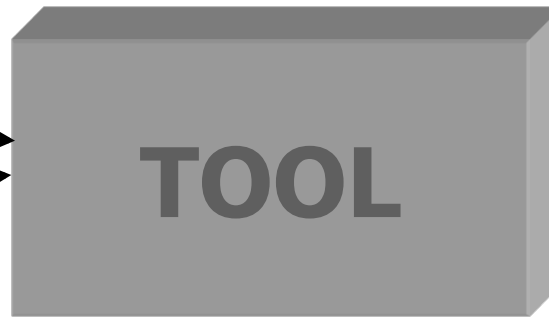


Cost



Probability

System Description



No!

Debugging Information

Yes

Prototypes
Executable Code
Test sequences

Requirement

$A[](\text{req} \Rightarrow A \langle \rangle \text{grant})$

$A[](\text{req} \Rightarrow A \langle \rangle_{t < 30\text{ms}} \text{grant})$

$A[](\text{req} \Rightarrow A \langle \rangle_{t < 30\text{ms}, c < 5\text{pJ}} \text{grant})$

$A[](\text{req} \Rightarrow A \langle \rangle_{t < 30\text{ms}, p > 0.90} \text{grant})$

UPPAAL Tool Suite

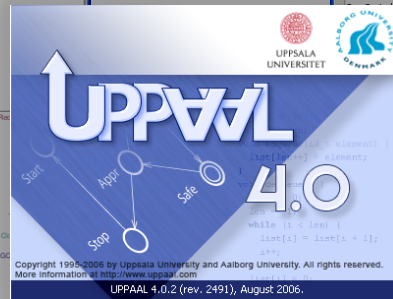
Editor

Discrete Control
Concurrency
Timing Constraints
Resources
Continuous Aspects
Stochasticity

Simulator

Enabled Transitions
ReqSpeed: GearControl->Engine

Simulation Trace
(Gear, GearN, Initial, Neutral, Closed)
Initiate, chkGearN, Initial, Neutral, Closed



Verifier

Overview

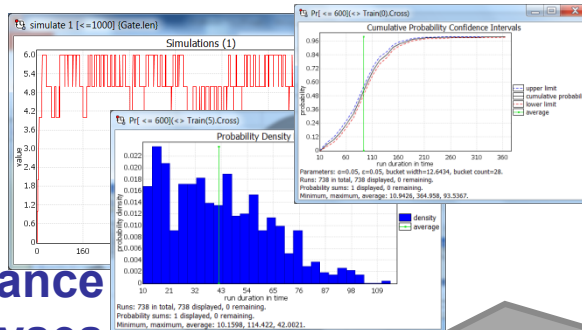
```

E<> GearControl.GearChanged
E<> ( Interface.Gear5 )
E<> ( Interface.GearR )
A[] ( GearControl.GearChanged and ( SysTimer<=100 ) )
A[] not ( GearBox.Neutral and ( Interface.Gear1 or I...
    
```

Query
E<> GearControl.GearChanged

Comment
P1. It is possible to change gear.

Status
Property is satisfied.
A[] (Clutch.Closed imply (GearControl.ReqTorqueC or GearControl.GearChanged or GearControl.Gear or GearC...
Property is satisfied.
A[] (GearBox.Idle imply (GearControl.ClutchClose or GearControl.CheckClutchClosed or GearControl.CClose
Property is satisfied.
A[] (GearBox.Neutral imply (GearControl.ReqSetGear or GearControl.ChkClutchClosed or GearControl.C...
Property is satisfied.
A[] (GearControl.ClutchClosed imply Clutch.Closed)



Making a TOOL

- Gerd Behrmann (DK)
- Alexandre David (F/DK)
- Marius Mikucionis (DK)
- Jakob I Rasmussen (DK)
- Emmanuel Fleury (F)
- Didier Lime (F)
- Franck Cassez (F)
- Peter Gjør Jensen (DK)
- Brian Nielsen, Arne Skou, Frits Vandrager, Pedro D'Argenio,..
- Wang Yi (S)
- Paul Pettersson (S)
- Johan Bengtson (S)
- Fredrik Larsson (S)
- Patricia Bouyer (F)
- Nicolas Markey (F)
- Jakob H Taankvist (DK)
- Axel Legay (B)
- Danny B Poulsen (DK)

Originators

Chief Eng

Abstractions / Cost

Synthesis

Probabilities

Users

Making the Tool **RIGHT**



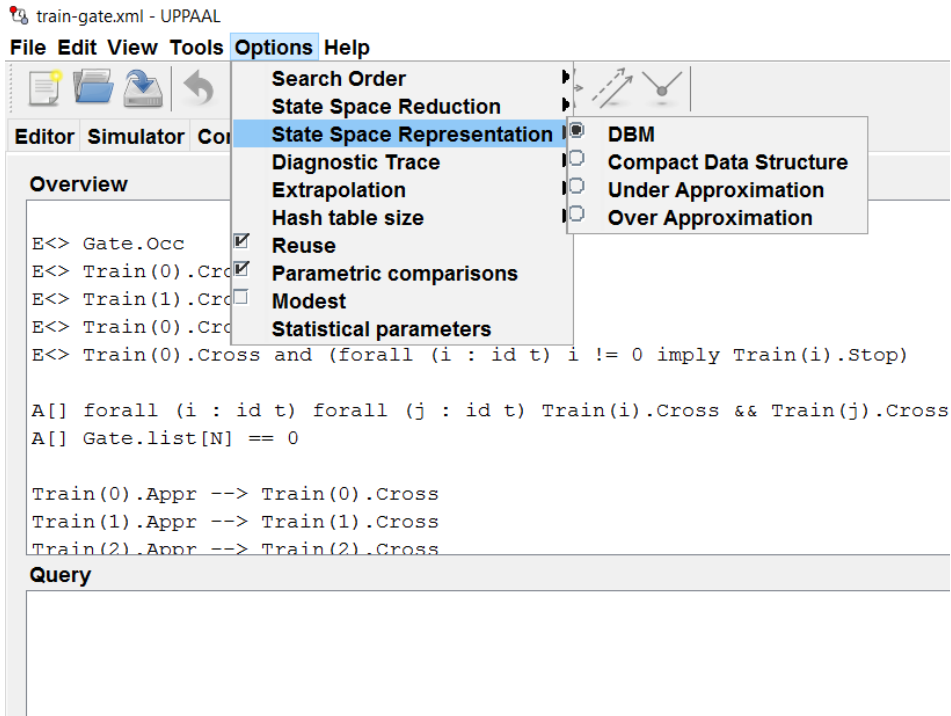
Making the **RIGHT** Tool



Making the Tool RIGHT



Verification Options



- Search Order
 - Depth First
 - Breadth First
 - Random Depth First

- State Space Reduction
 - None
 - Conservative
 - Aggressive
 - Extreme
- State Space Representation
 - DBM
 - Compact Form
 - Under Approximation
 - Over Approximation
- Diagnostic Trace
- Extrapolation
 - Automatic
 - None
 - Difference
 - Local
 - Lower/Upper

State Space Reduction

(to Store or not to Store)

117 states_{total}

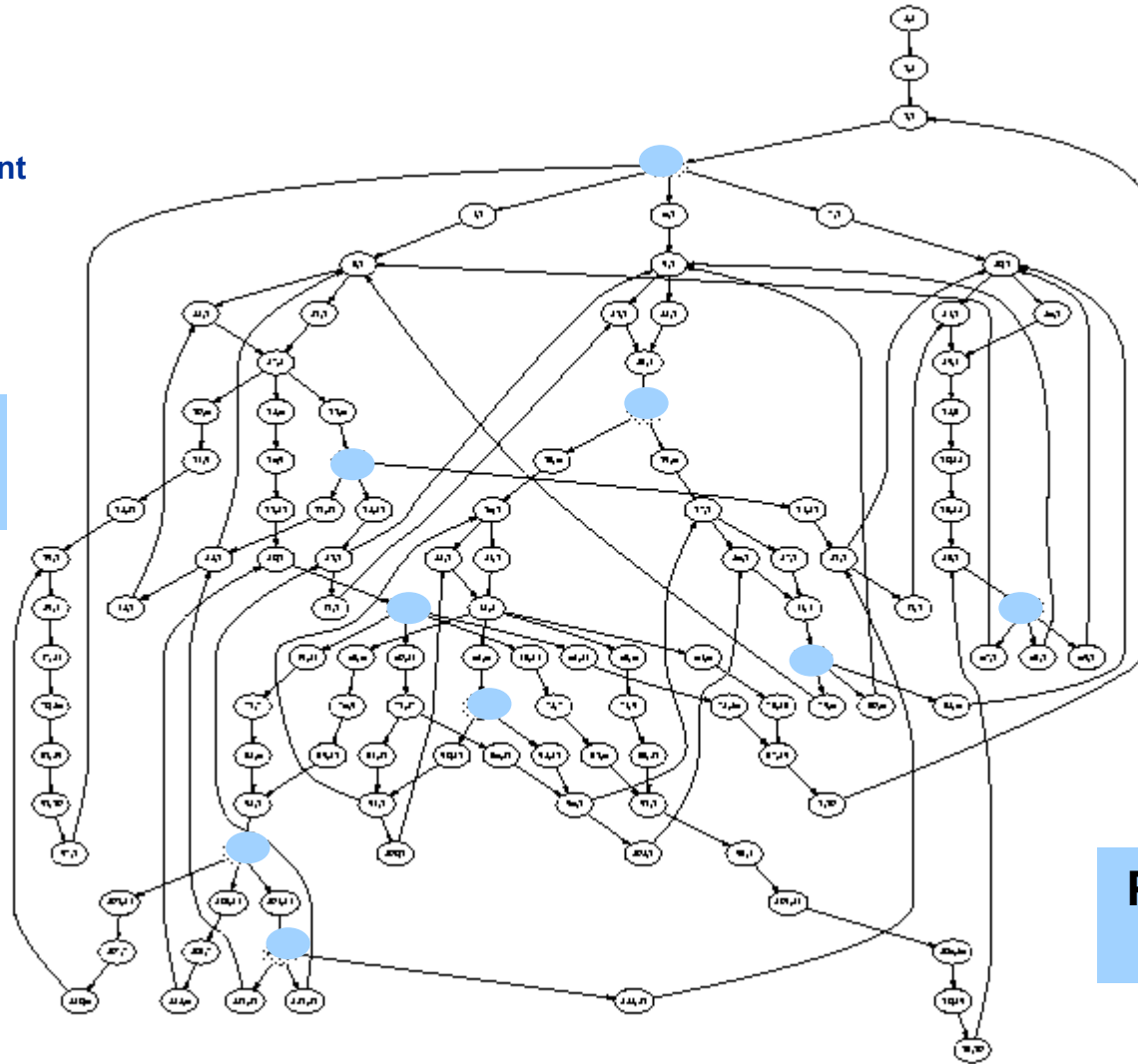
>

81 states_{entrypoint}

>

9 states

Time OH
less than 10%



Philips Audio
Protocol

[Behrmann, L, Pelanek, 2003]

The "secret" of UPPAAL

The screenshot displays the UPPAAL simulator interface for a file named "C:\Users\kg\Desktop\DESKTOP12\UPPAAL\UPPAAL examples\LCCC2013\SMC\TrainGateCPS14.xml". The interface includes a menu bar (File, Edit, View, Tools, Options, Help), a toolbar, and several tabs: Editor, Simulator, ConcreteSimulator, Verifier, and Yggdrasil.

The "Enabled Transitions" panel on the left shows "go[front()]: Gate → Train(5)". Below it are "Next" and "Reset" buttons. The "Simulation Trace" panel shows a sequence of events for Train(1): (Safe, Cross, Stop, Stop, Stop, Stop, Occ), followed by "leave[1]: Train(1) → Gate[1]", and then "(Safe, Safe, Stop, Stop, Stop, Stop, Free)". The "Trace File" section contains playback controls (Prev, Next, Replay, Open, Save, Random) and a speed slider from Slow to Fast.

The main window displays a state transition diagram with nodes representing states and transitions labeled with events like "go[1]", "leave[1]", and "Stop". A large black-bordered box in the center of the diagram contains the following list of constraints:

- Train(5).x ∈ [30,65]
- Train(4).x ∈ [23,60]
- Train(5).x ∈ [30,65]
- Train(0).x - time ≤ -50
- Train(0).x - Train(1).x ∈ [10,20]
- Train(0).x - Train(2).x ∈ [0,5]
- Train(3).x - Train(0).x ∈ [17,40]
- Train(4).x - Train(0).x ∈ [10,35]
- Train(2).x - Train(1).x ∈ [7,20]

Below the diagram, a vertical timeline shows the states of the Gate (Occ, Safe, Stop, Free) and the Train(5) (Occ). Red arrows indicate the timing of "leave[1]" and "go[front()]" events.

Zones & DBMs

THE "secret" UPPAAL

- DBM package
- Minimal Constraint Form

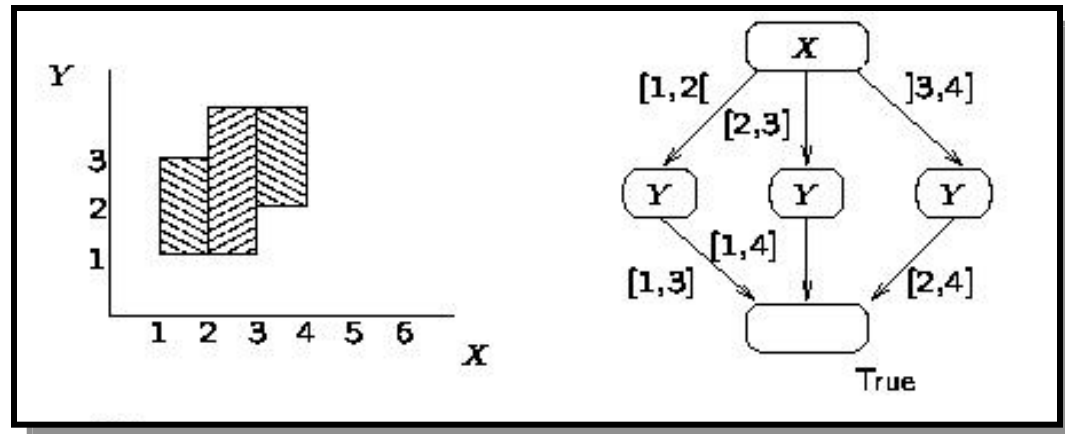
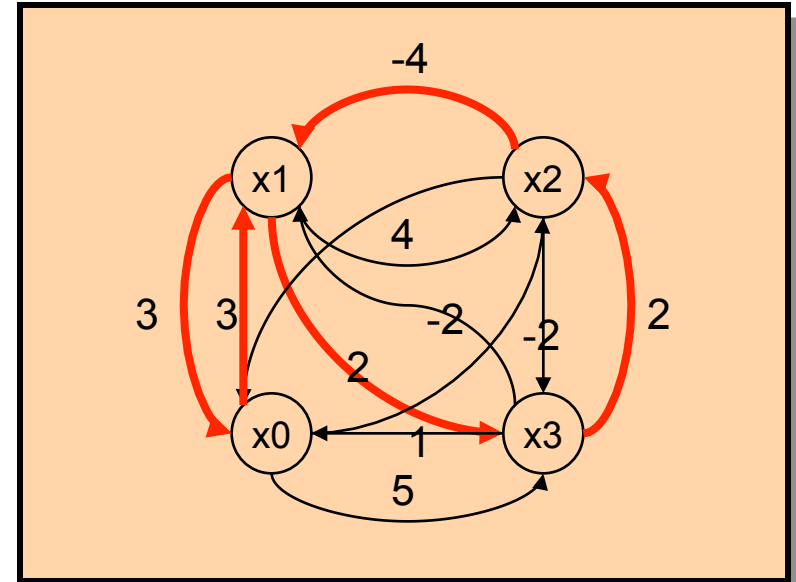
[RTSS97]

- Clock Difference Diagrams

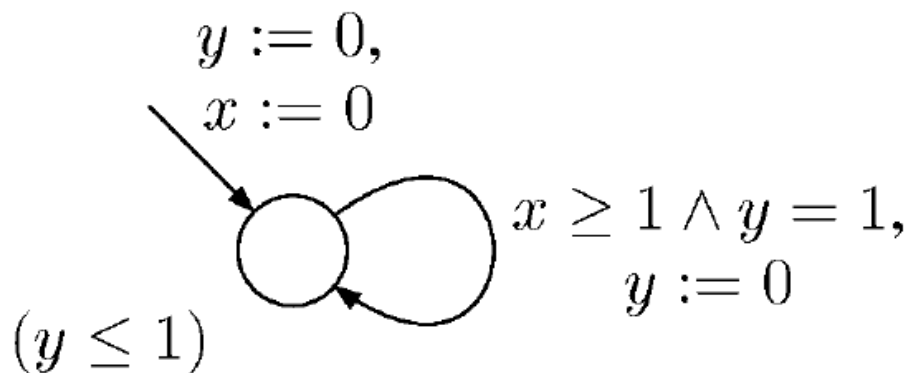
[CAV99]

- PW List

[SPIN03]

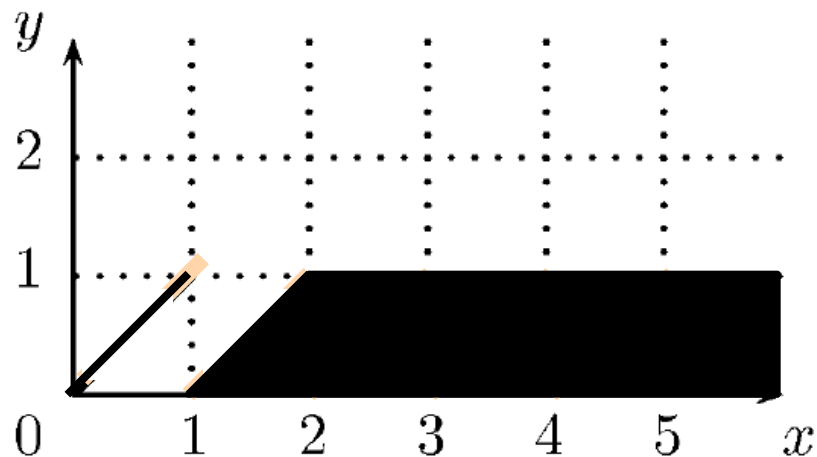


Forward Zone Exploration

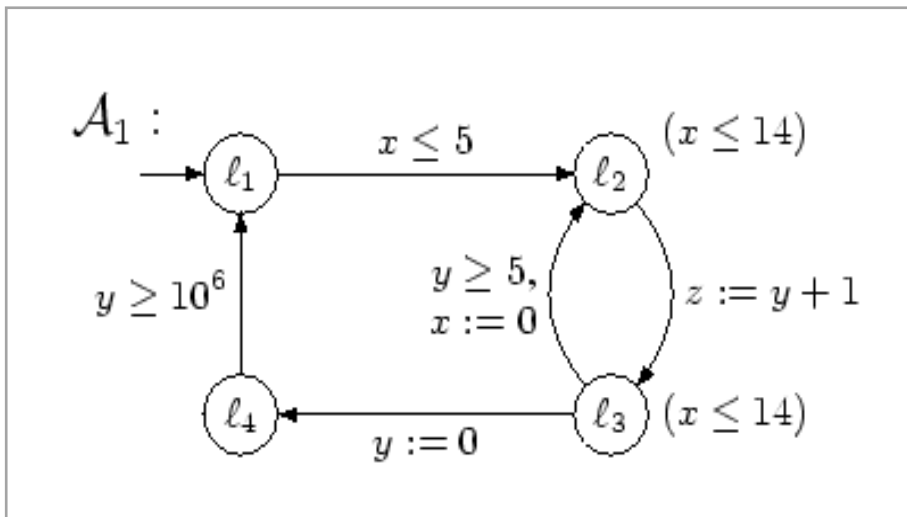
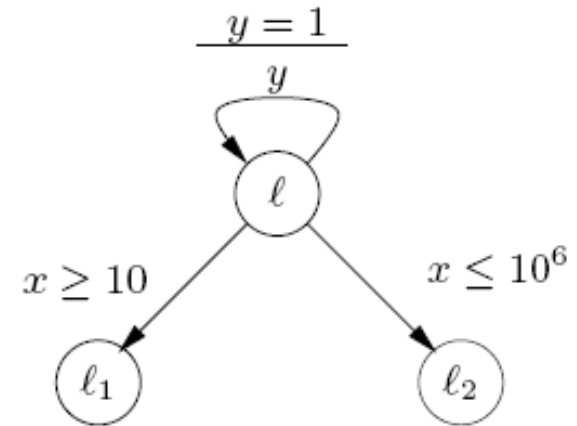
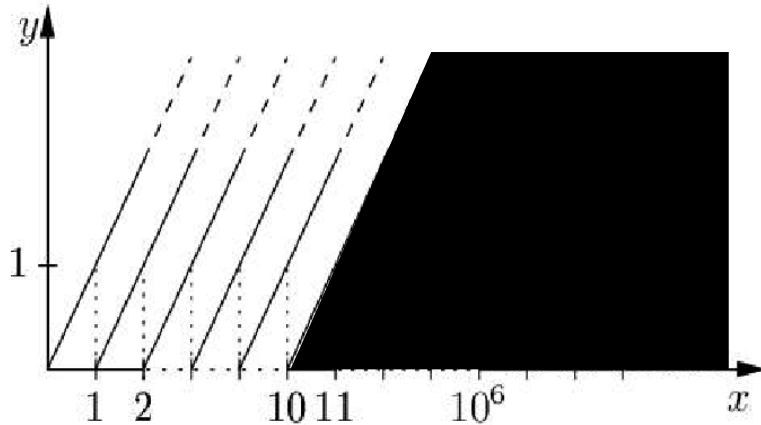


TERMINATION
not
garanteed

Need for
Finite
Abstractions



Zone Extrapolation



- Abstraction taking maximum constant into account necessary for termination
- Utilization of distinction between lower and upper bounds
- Utilization of location-dependency

[Behrmann, Bouyer, Fleury, L, Pelanek, 2003, 2004]

Zone Extrapolation

Model	Classical			Loc. dep. Max			Loc. dep. LU			Convex Hull		
	-n1			-n2			-n3			-A		
	Time	States	Mem	Time	States	Mem	Time	States	Mem	Time	States	Mem
f5	4.02	82,685	5	0.24	16,980	3	0.03	2,870	3	0.03	3,650	3
f6	597.04	1,489,230	49	6.67	158,220	7	0.11	11,484	3	0.10	14,658	3
f7				352.67	1,620,542	46	0.47	44,142	3	0.45	56,252	5
f8							2.11	164,528	6	2.08	208,744	12
f9							8.76	598,662	19	9.11	754,974	39
f10							37.26	2,136,980	68	39.13	2,676,150	143
f11							152.44	7,510,382	268			
c5	0.55	27,174	3	0.14	10,569	3	0.02	2,027	3	0.03	1,651	3
c6	19.39	287,109	11	3.63	87,977	5	0.10	6,296	3	0.06	4,986	3
c7				195.35	813,924	29	0.28	18,205	3	0.22	14,101	4
c8							0.98	50,058	5	0.66	38,060	7
c9							2.90	132,623	12	1.89	99,215	17
c10							8.42	341,452	29	5.48	251,758	49
c11							24.13	859,265	76	15.66	625,225	138
c12							68.20	2,122,286	202	43.10	1,525,536	394
bus	102.28	6,727,443	303	66.54	4,620,666	254	62.01	4,317,920	246	45.08	3,826,742	324
philips	0.16	12,823	3	0.09	6,763	3	0.09	6,599	3	0.07	5,992	3
sched	17.01	929,726	76	15.09	700,917	58	12.85	619,351	52	55.41	3,636,576	427

Making the RIGHT Tool



- Real-time protocols
- Real-time controllers
- Scheduling
- Schedulability analysis
- WCET analysis
- Testing
-



Bang & Olufsen IR-Link

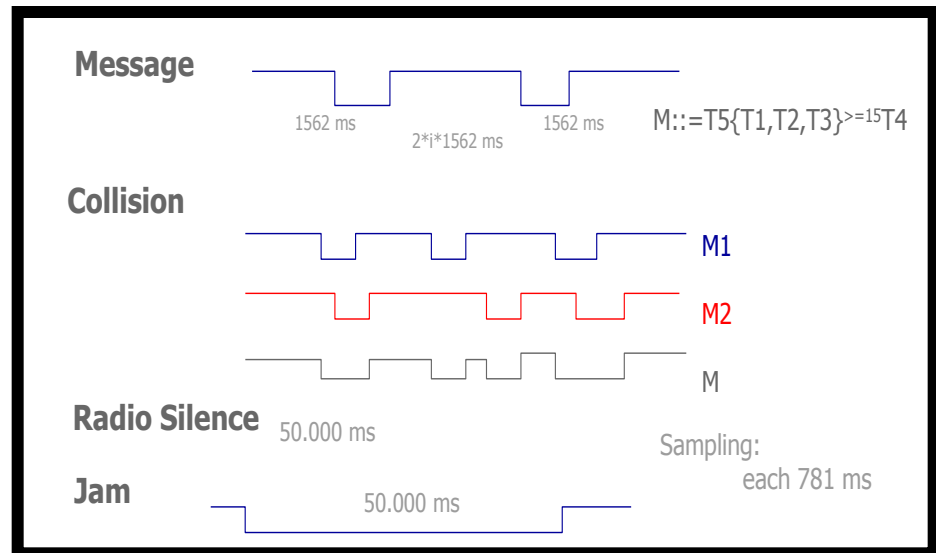
- Bug known to exist for 10 years
- Ill-described:
 - 2.800 lines of assembler code + 3 flowchart + 1 B&O eng.
- 3 months for modeling.
- UPPAAL detects error with 1.998 transition steps (shortest)
- Error trace was confirmed in B&O laboratory.
- Error corrected and verified in UPPAAL.

Arne Skou, Klaus Havelund



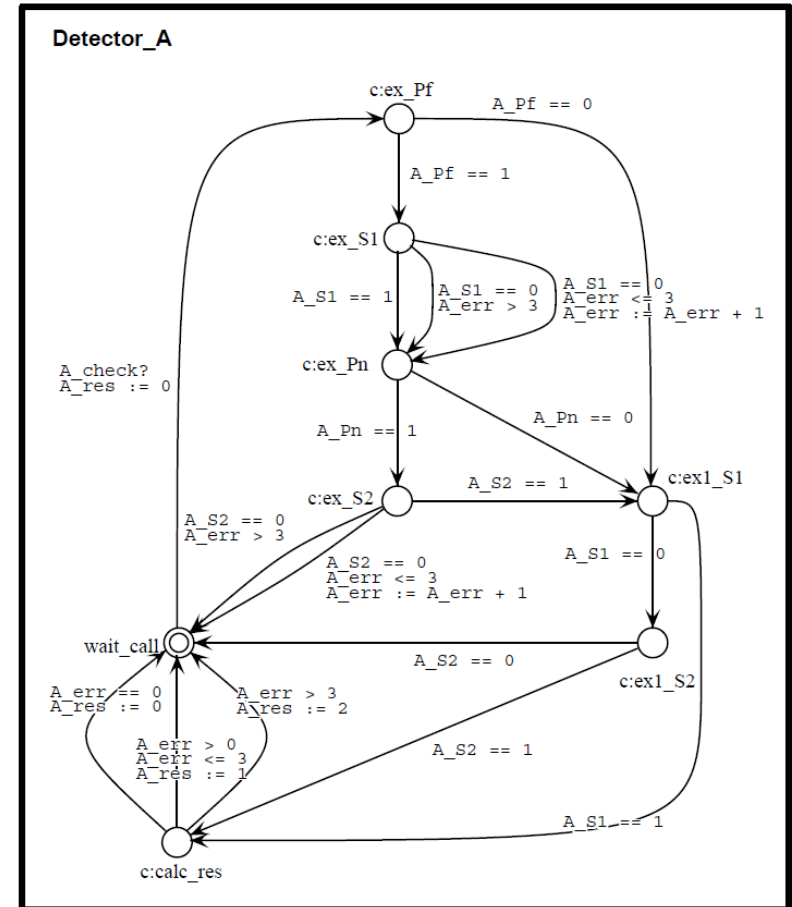
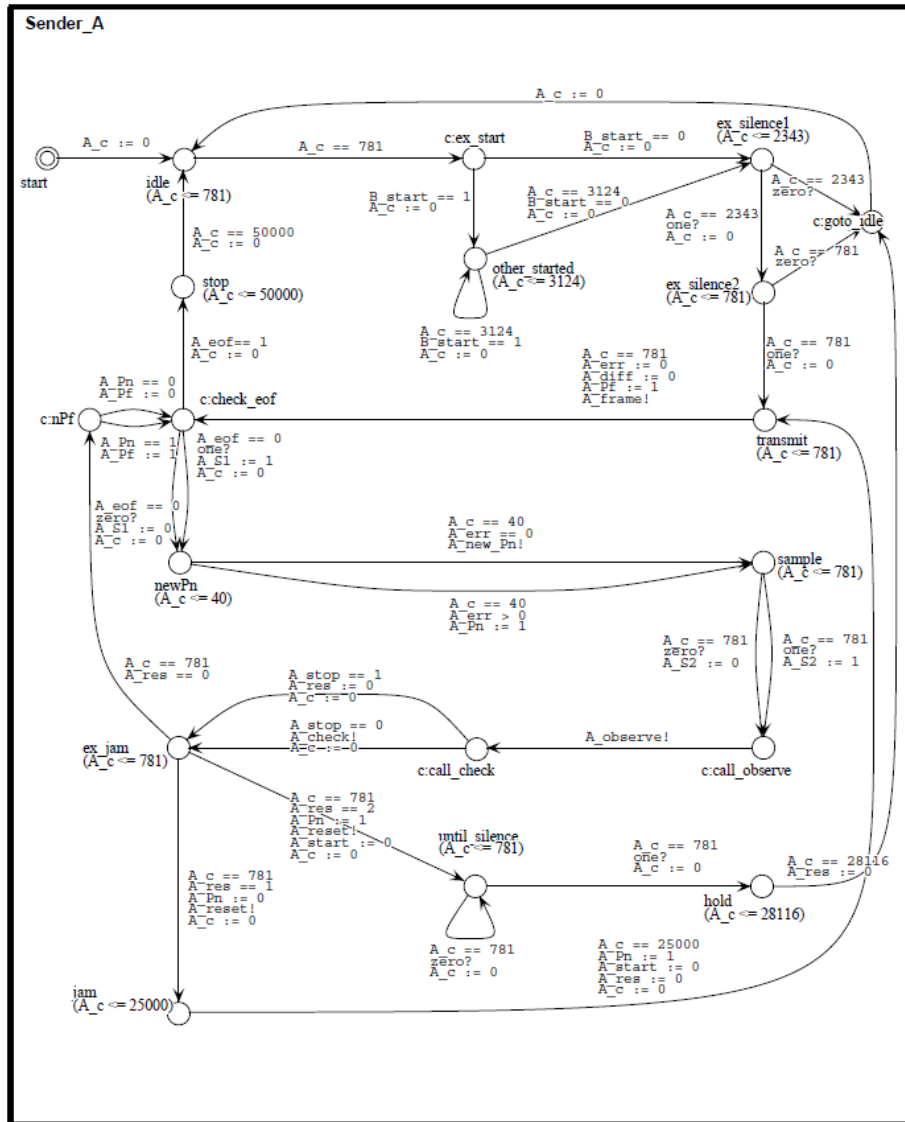
Bang & Olufsen IR-Link

- Bug known to exist for 10 years
- Ill-described:
 - 2.800 lines of assembler code + 3 flowchart + 1 B&O eng.
- 3 months for modeling.
- UPPAAL detects error with 1.998 transition steps (shortest)
- Error trace was confirmed in B&O laboratory.
- Error corrected and verified in UPPAAL.



1st RTSS'97 talk, Klaus Havelund

Bang & Olufsen IR-Link



Philips Bounded Retransmission Protocol



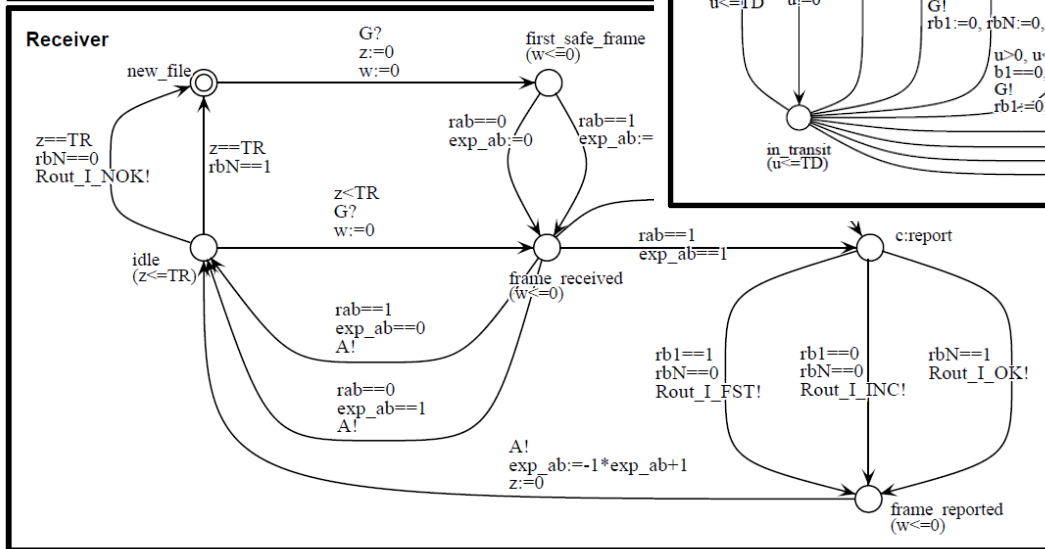
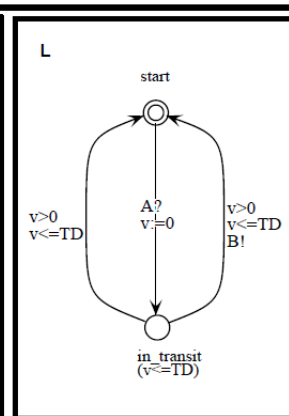
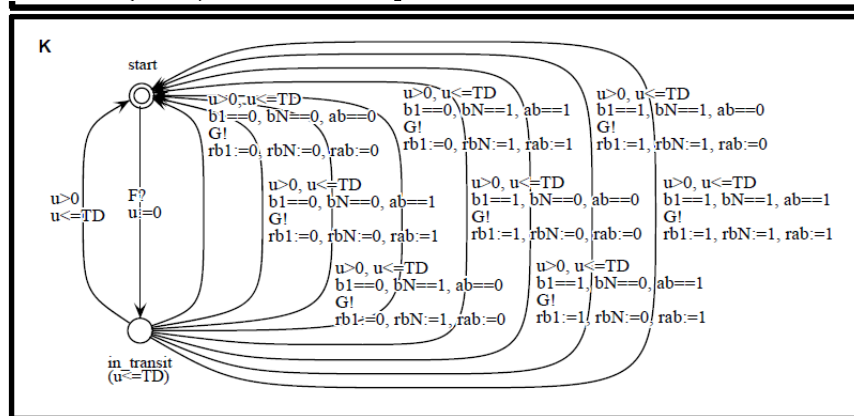
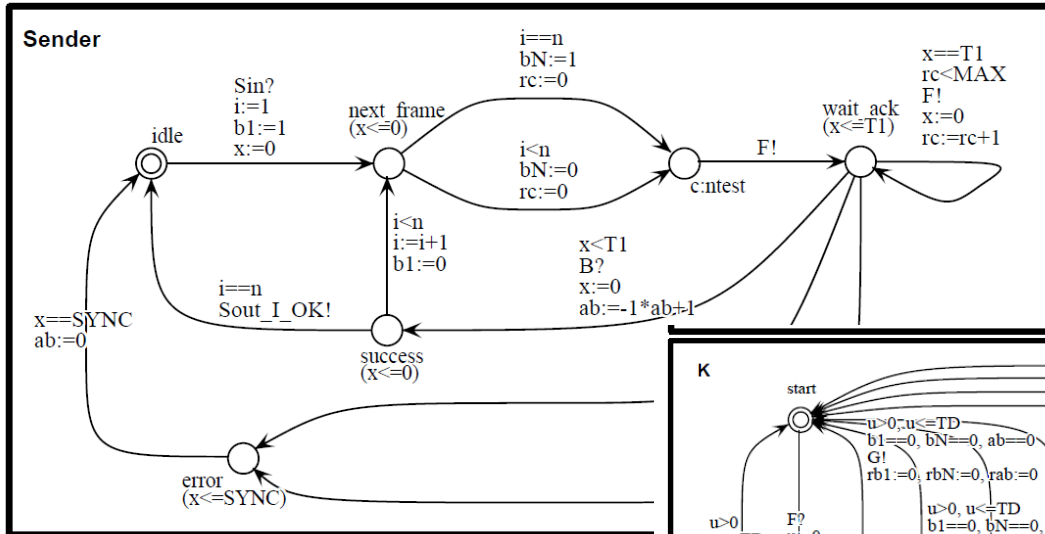
Pedro D'Argenio
Joost-Pieter Katoen
Theo Ruys
Jan Tretmans

Science):

```
> I should tell you that I am quite dissappointed with this new
> release of Uppaal ;). You take all the fun out of it!!. With this
> new releas I could verify everything in a couple of minutes,
> including a couple of properties that where impossible before!!!
> Moreover, I was playing with the simulator and I found a silly
> deadlock in the specification.
>
> ...
>
> I found this new Uppaal a quite huge leap from the previous version.
> As a user, I have had a really good first impression. I will
> compile a list of comments for tomorrow afternoon.
```

Regards,

Philips Bounded Retransmission Protocol

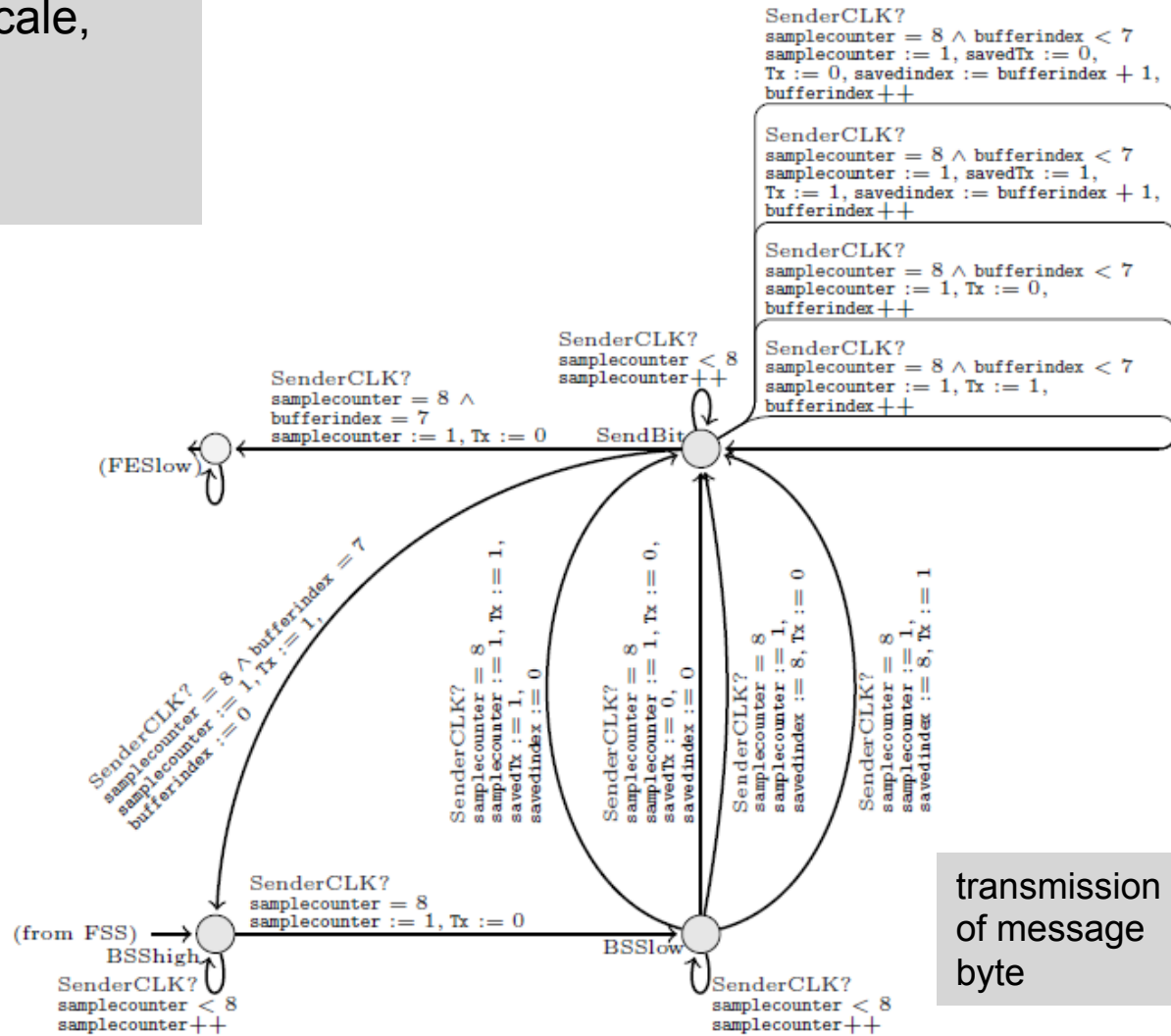


FlexRay



BMW, Bosch, Daimler, Freescale,
General Motors, NXP
Semiconductors, and
Volkswagen

Fault-tolerance
Timed hardware model
Parameterized error models
(glitches, jitter)
Voting & bit-clock alignment



transmission
of message
byte

BMW, Bosch, Daimler, Freescale,
General Motors, NXP
Semiconductors, and
Volkswagen

(a) Standard parameter values.

Parameter	Value	Corresponds to
CYCLE	10000	$\frac{1}{80 \text{ MHz}} = 12.5 \text{ ns}$
DEVIATION	30	$\pm 0.15 \%$
SETUP	368	460 ps
HOLD	1160	1450 ps
PMIN	12	15 ps
PMAX	1160	1450 ps
ERRDIST	4	1 out of 5

(b) Changed parameter values.

Changed parameter	Tolerable glitches
$\text{PMAX} - \text{PMIN} \leq 6086$	1 out of 4
$\text{PMAX} - \text{PMIN} \leq 6086$	at most 2
$\text{PMAX} - \text{PMIN} \leq 9616$	at most 1
$\text{DEVIATION} \leq 92$	1 out of 4
$\text{DEVIATION} \leq 92$	at most 2
$\text{DEVIATION} \leq 218$	at most 1
$\text{DEVIATION} \leq 348$	none
Voting window size = 3	1 out of 3
Voting window size = 5	1 out of 4
Voting window size = 7	1 out of 5
Voting window size = 9	1 out of 6

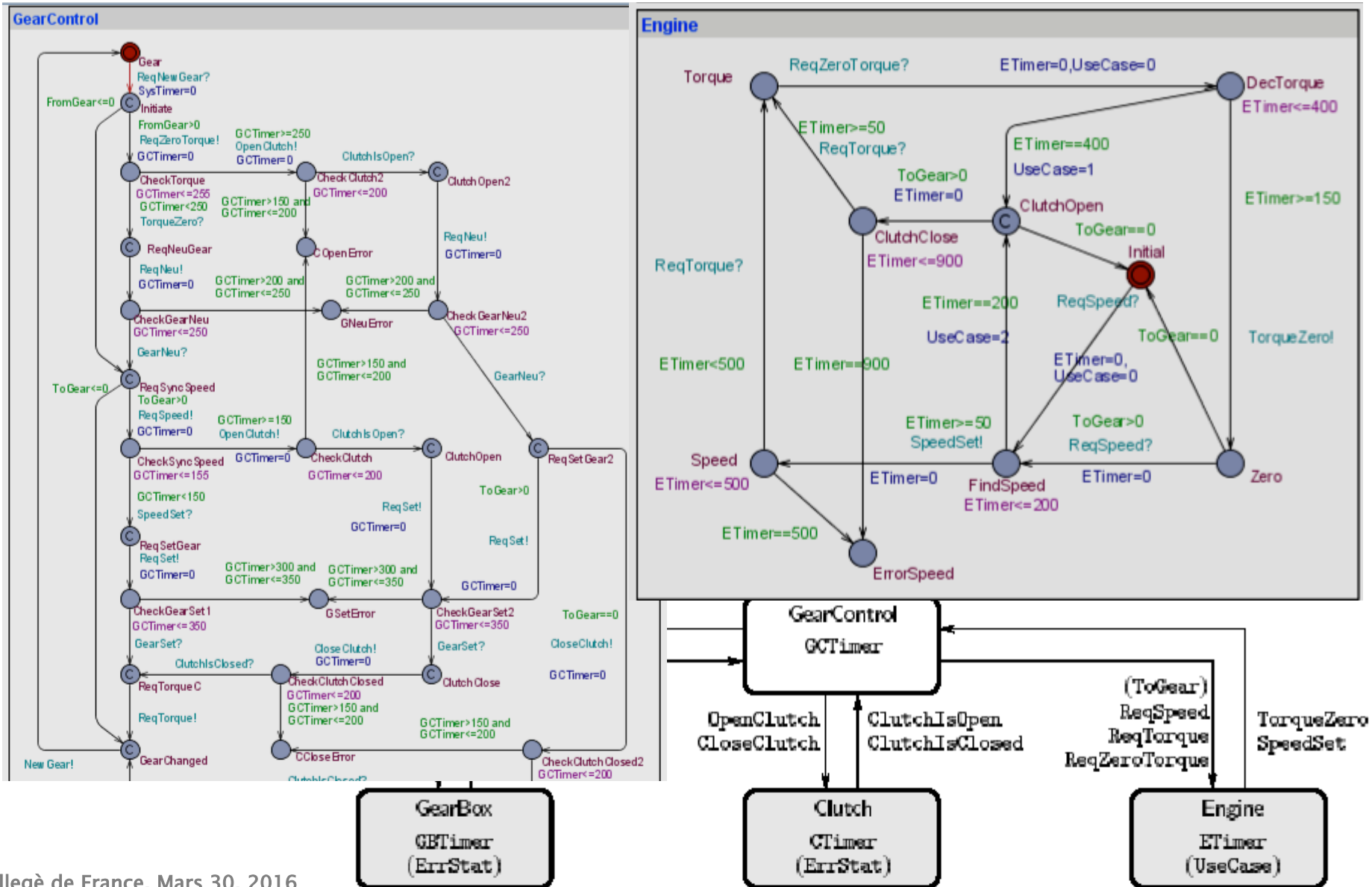
(Wireless) Protocols in UPPAAL

- Bang & Olufsen IR Link
- Philips Audio Protocol
- Collision-Avoidance Protocol
- Bounded Retransmission Protocol
- TDMA Protocol
- Multimedia Streams
- ATM ABR Protocol
- Lamport's Leader Election Protocol
- ABB Fieldbus Protocol
- IEEE 1394 Firewire Root Contention
- Bluetooth Protocol
- Distributed Agreement Protocol
- FlexRay
- CHES MAC Protocol
- Proprietary WSN, Other Big Danish Company

MECEL AB

Gear Controller

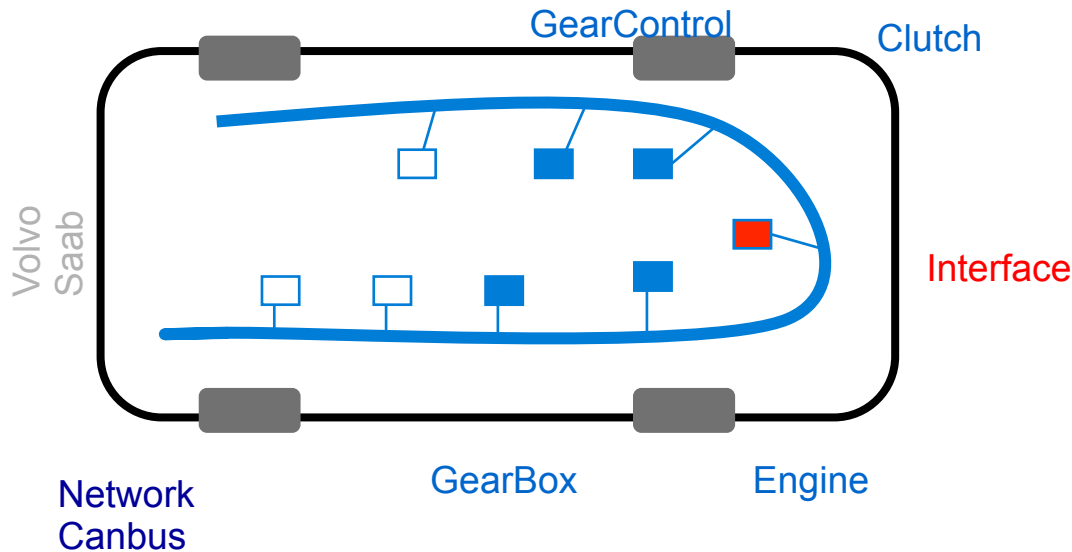
Lindahl, Pettersson, Yi 1998



MECEL AB

Gear Controller

Lindahl, Pettersson, Yi 1998



Paul Pettersson

$$\text{GearControl@Initiate} \rightsquigarrow_{\leq 1500} ((\text{ErrStat} = 0) \Rightarrow \text{GearControl@GearChanged}) \quad (1)$$

$$\text{GearControl@Initiate} \rightsquigarrow_{\leq 1000}$$

$$((\text{ErrStat} = 0 \wedge \text{UseCase} = 0) \Rightarrow \text{GearControl@GearChanged}) \quad (2)$$

$$\text{Clutch@ErrorClose} \rightsquigarrow_{\leq 200} \text{GearControl@CCloseError} \quad (3)$$

$$\text{Clutch@ErrorOpen} \rightsquigarrow_{\leq 200} \text{GearControl@COpenError} \quad (4)$$

$$\text{GearBox@ErrorIdle} \rightsquigarrow_{\leq 350} \text{GearControl@GSetError} \quad (5)$$

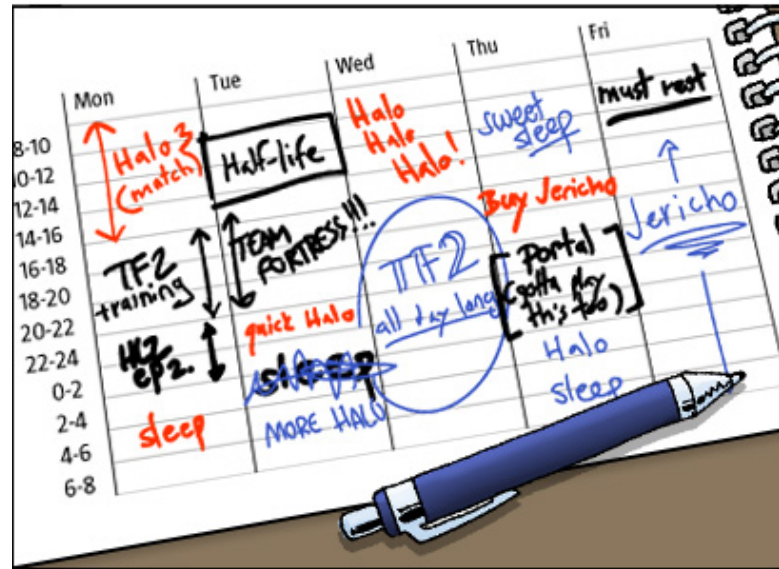
$$\text{GearBox@ErrorNeu} \rightsquigarrow_{\leq 200} \text{GearControl@GNeuError} \quad (6)$$

$$\text{Inv} (\text{GearControl@CCloseError} \Rightarrow \text{Clutch@ErrorClose}) \quad (7)$$

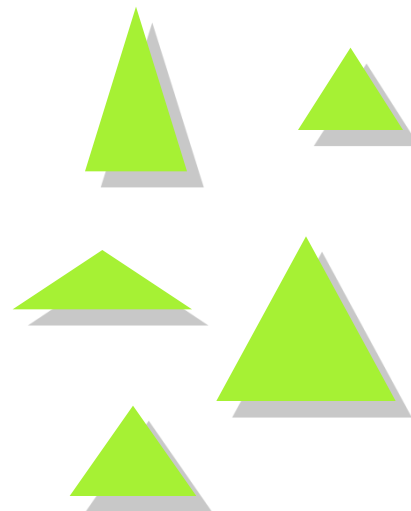
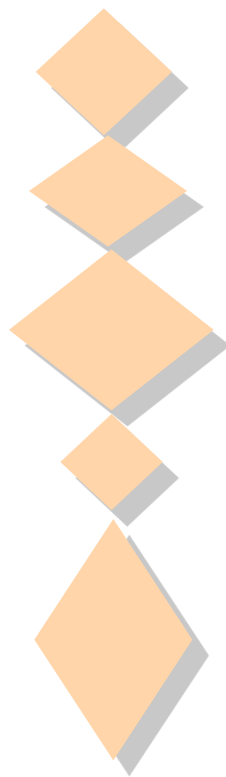
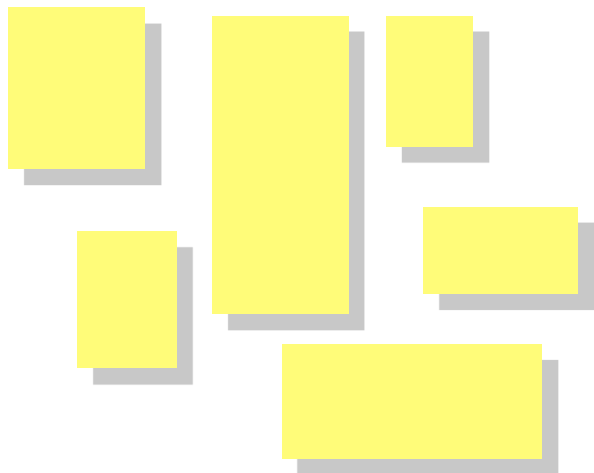
$$\text{Inv} (\text{GearControl@COpenError} \Rightarrow \text{Clutch@ErrorOpen}) \quad (8)$$

$$\text{Inv} (\text{GearControl@GSetError} \Rightarrow \text{GearBox@ErrorIdle}) \quad (9)$$

Scheduling



Embedded Systems



Tasks:

Computation times
Deadlines
Dependencies
Arrival patterns
uncertainties

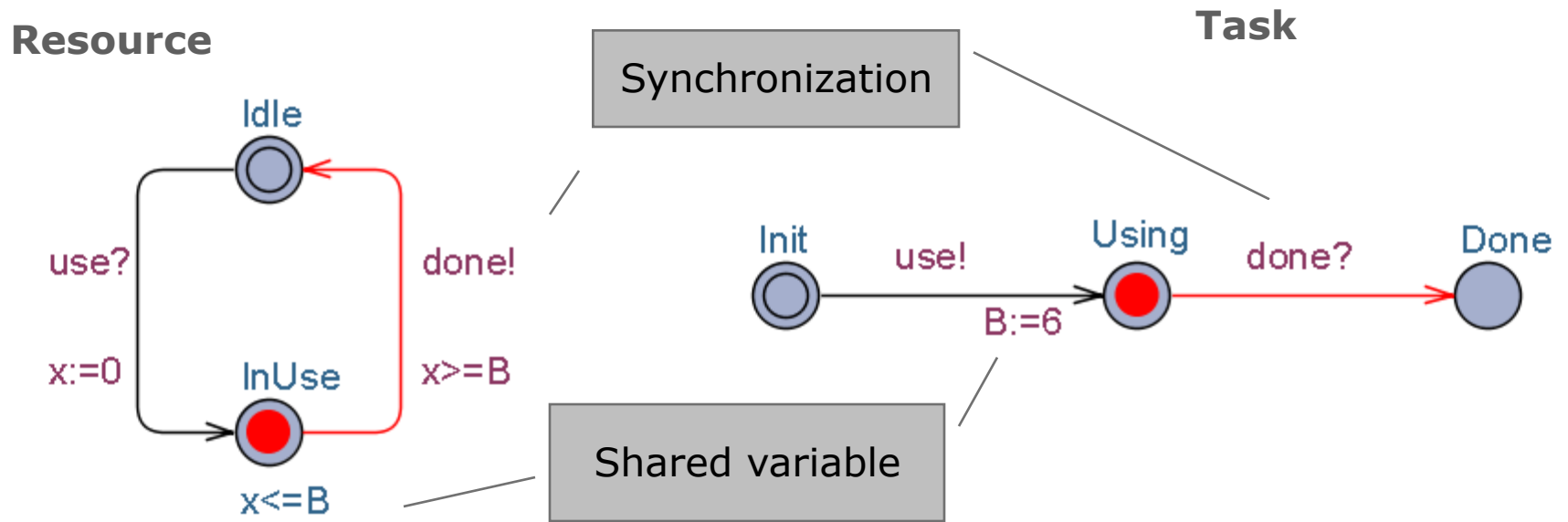
Scheduling Principles (OS)

EDF, FPS, RMS, DVS, ..

Resources

Execution platform
PE, Memory
Networks
Drivers
uncertainties

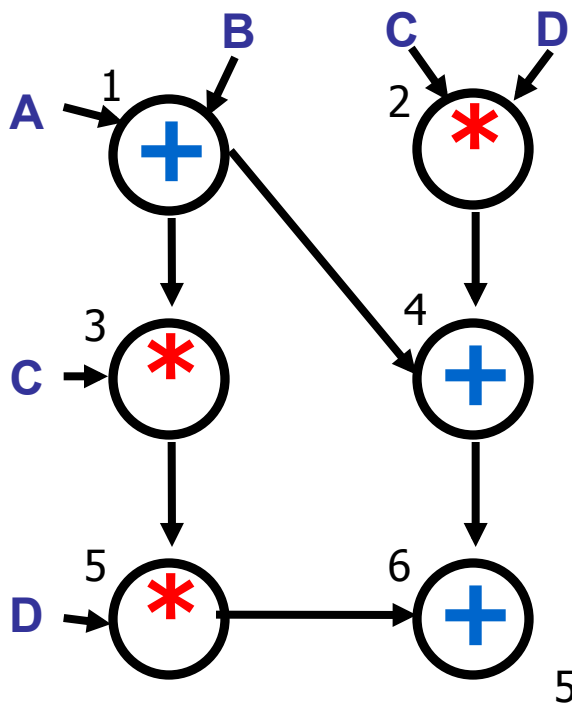
Ressource & Task



Semantics:

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

Optimal Scheduling – TIME



Compute :

$$(D * (C * (A + B)) + ((A + B) + (C * D)))$$
 4

using 2 processors

P1 (fast)

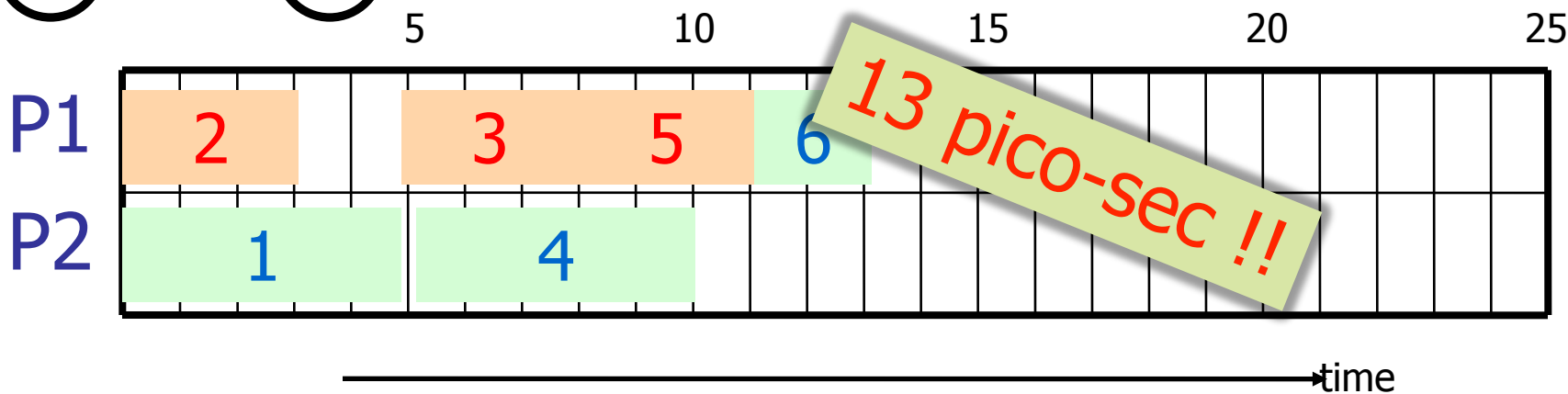
P2 (slow)



+	2ps
*	3ps



+	5ps
*	7ps



Optimal Scheduling – TIME

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

using 2 processors

P1 (fast)

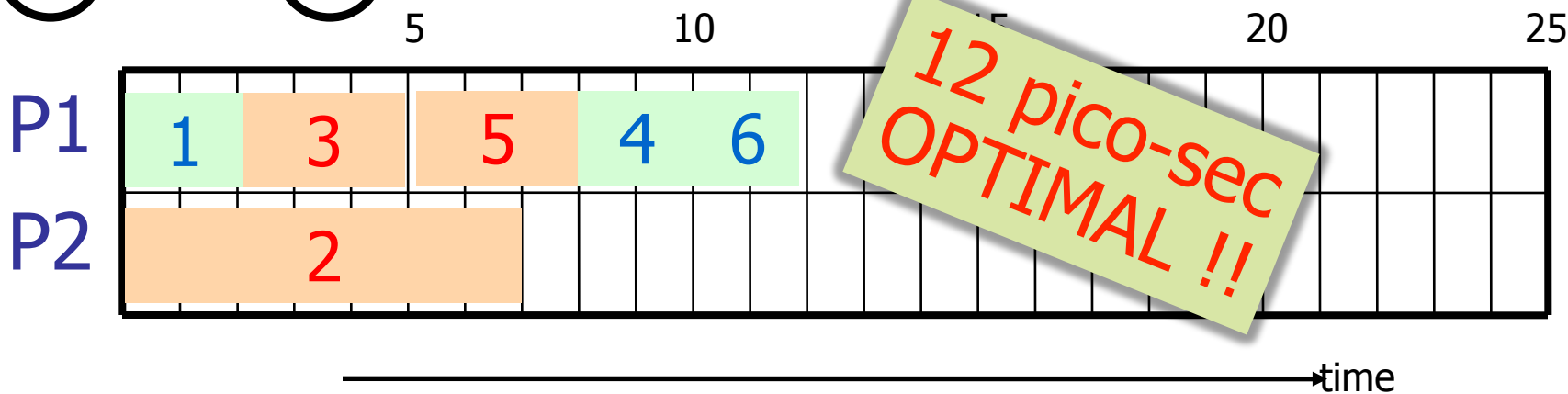
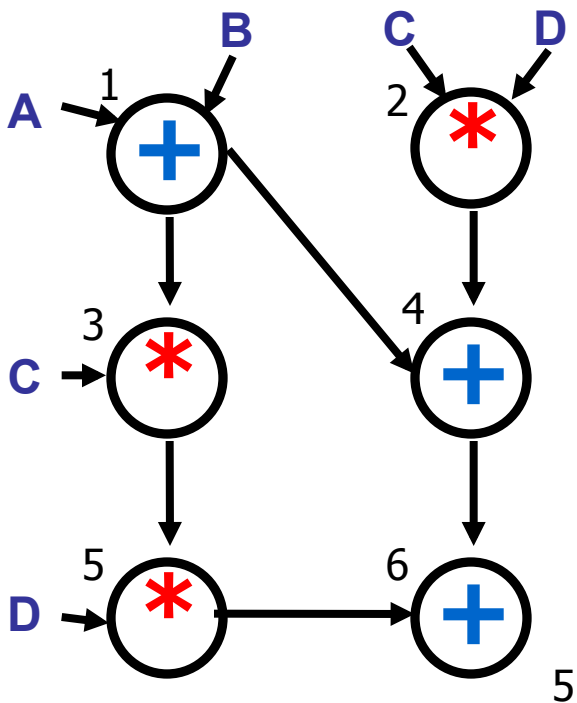
P2 (slow)



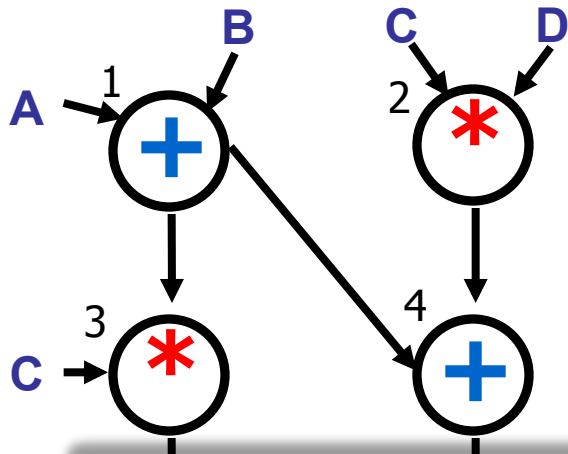
+	2ps
*	3ps



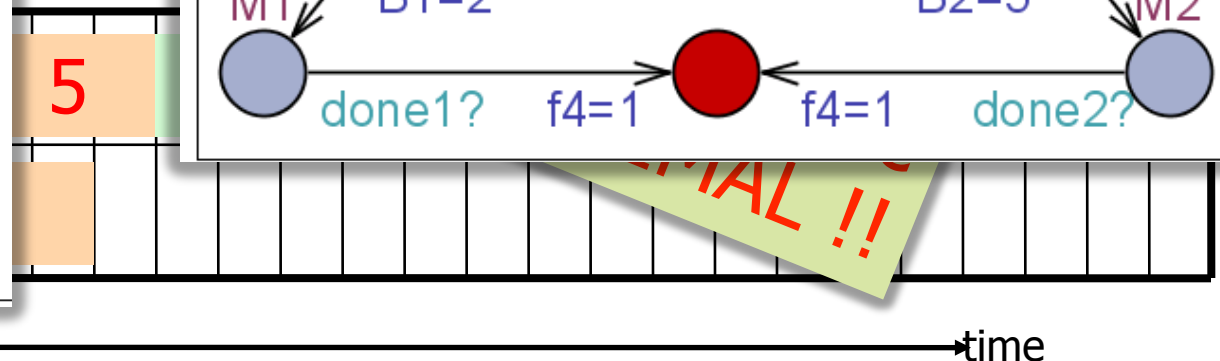
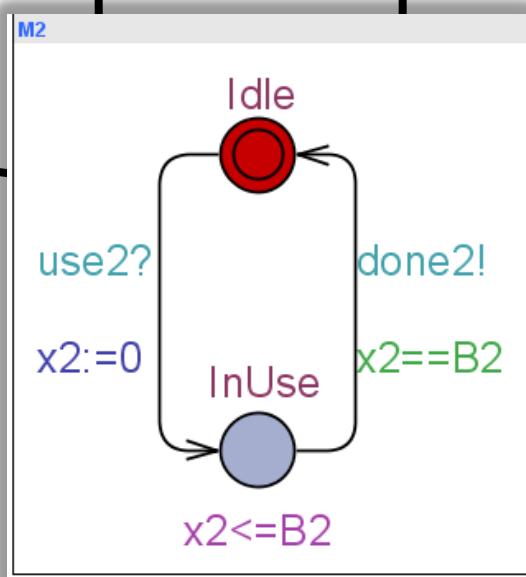
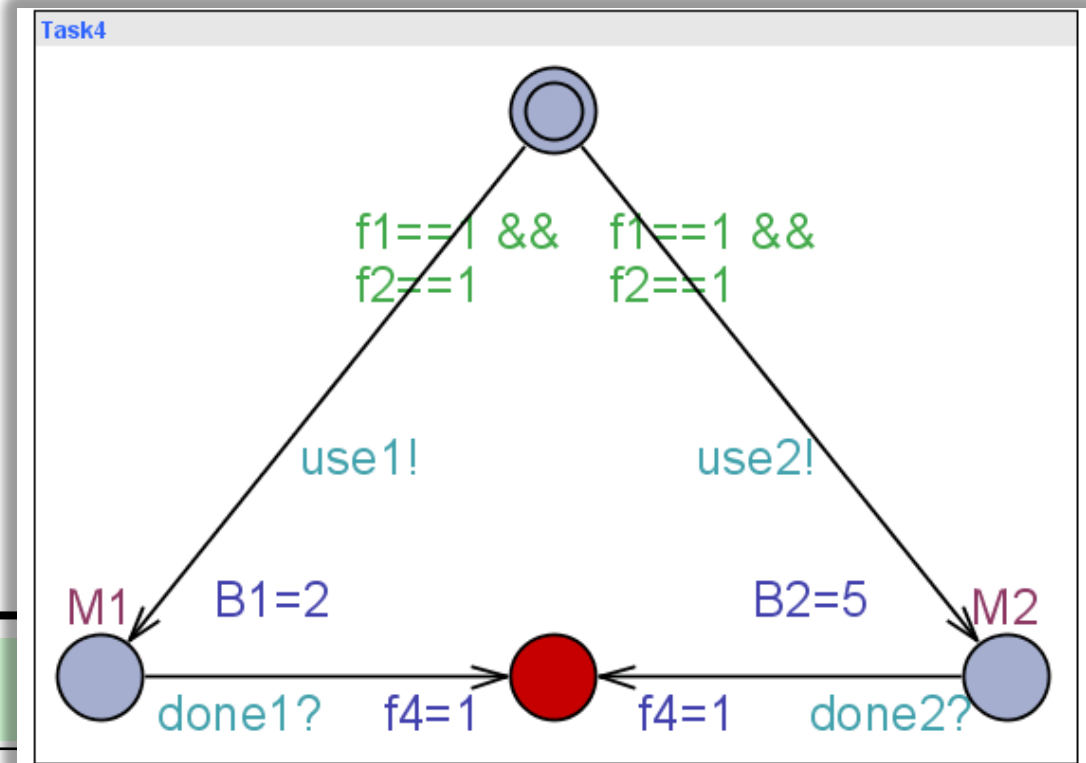
+	5ps
*	7ps



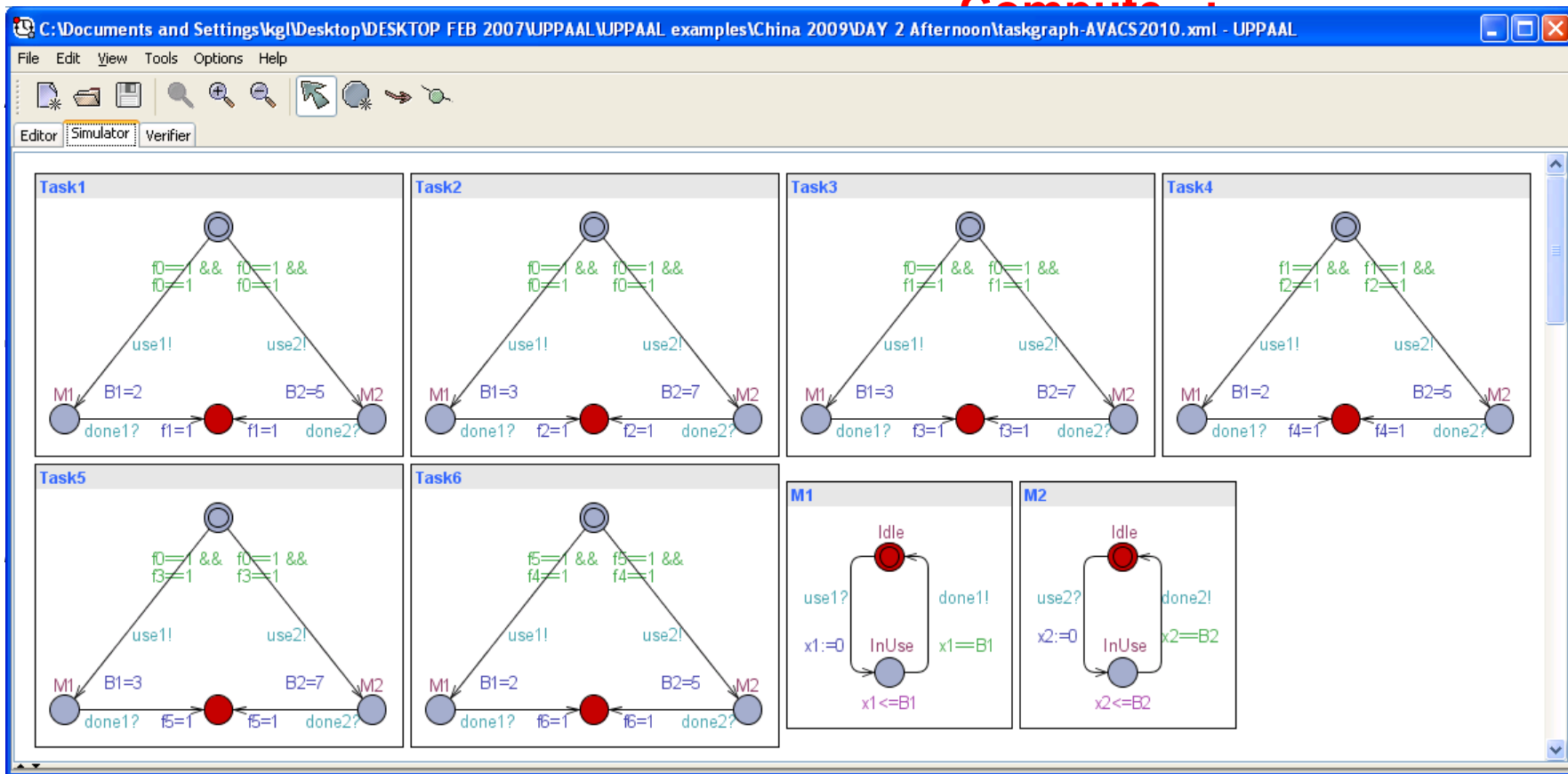
Optimal Scheduling – TIME



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



Optimal Scheduling – TIME



P2

$E \leftrightarrow$ (Task1.End and ... and Task6.End)

time

Experimental Results

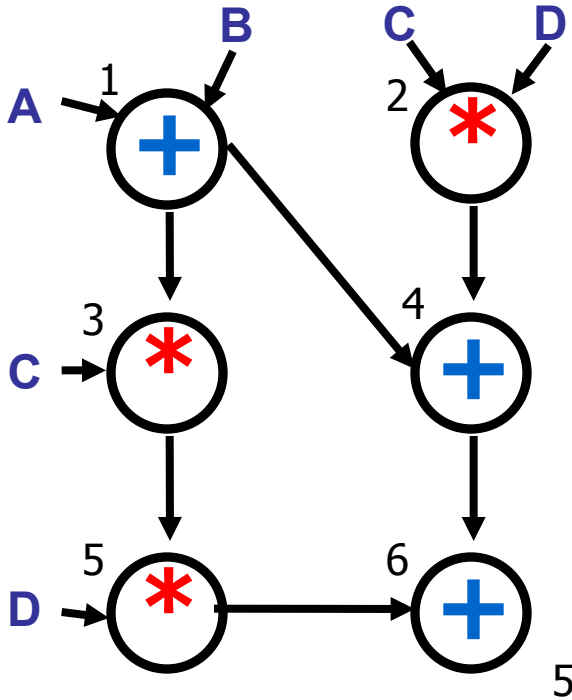
name	#tasks	#chains	# machines	optimal	TA
001	437	125	4	1178	1182
000	452	43	20	537	537
018	730	175	10	700	704
074	1007	66	12	891	894
021	1145	88	20	605	612
228	1187	293	8	1570	1574
071	1193	124	20	629	634
271	1348	127	12	1163	1164
237	1566	152	12	1340	1342
231	1664	101	16	t.o.	1137
235	1782	218	16	t.o.	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

Task Graph Scheduling – Revisited



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

using 2 processors

P1 (fast)

P2 (slow)



+	2ps
*	3ps

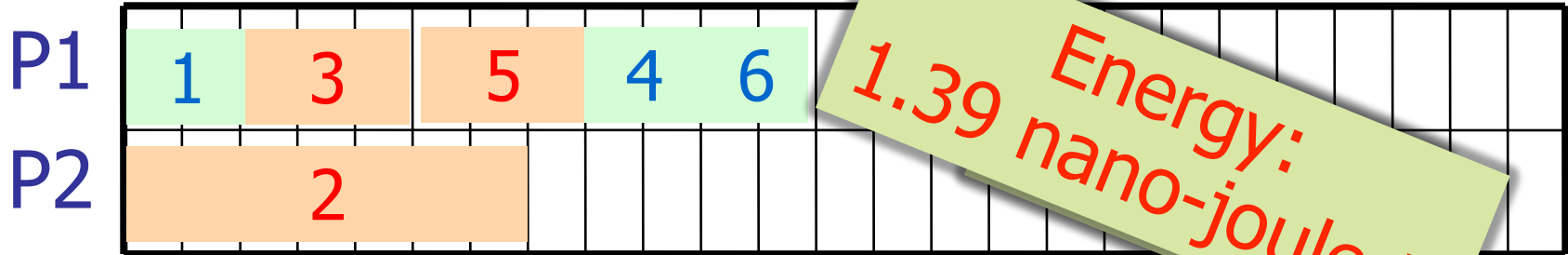
+	5ps
*	7ps

Idle	10W
In use	90W

Idle	20W
In use	30W

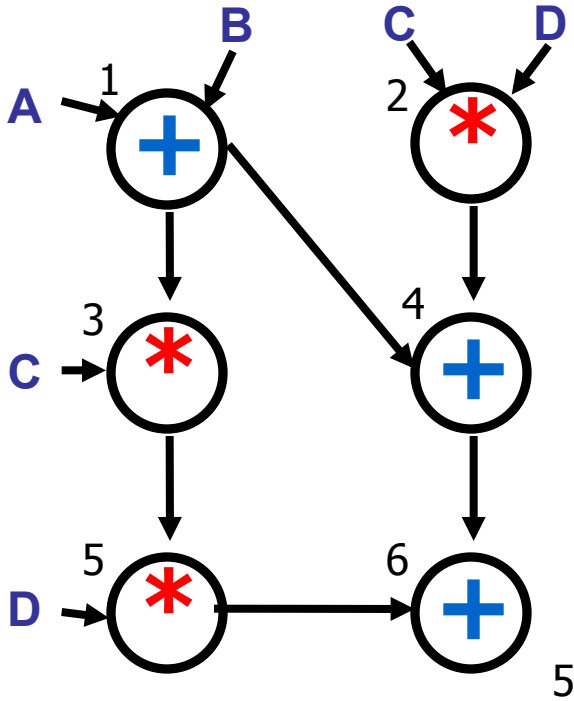
ENERGY:
10

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)



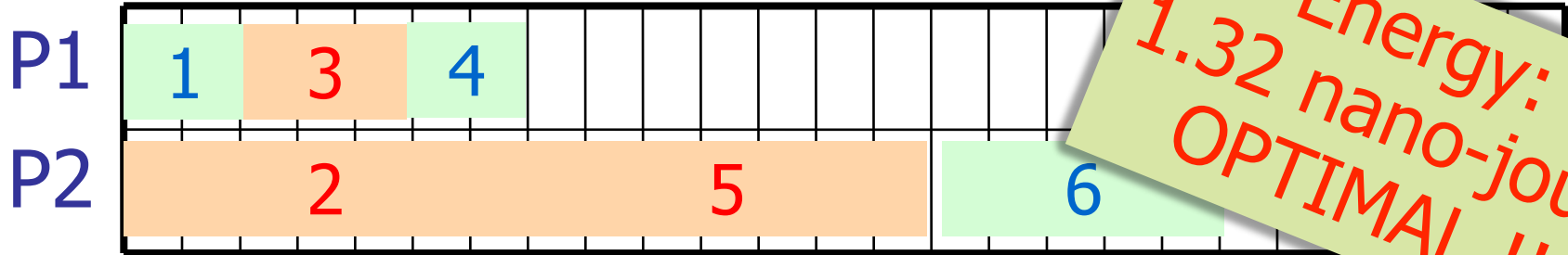
+	2ps
*	3ps

+	5ps
*	7ps

Idle	10W
In use	90W

Idle	20W
In use	30W

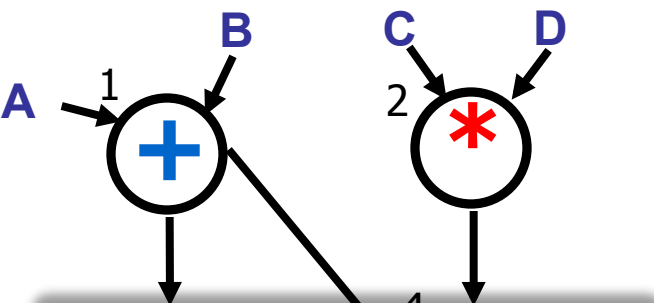
ENERGY:
10



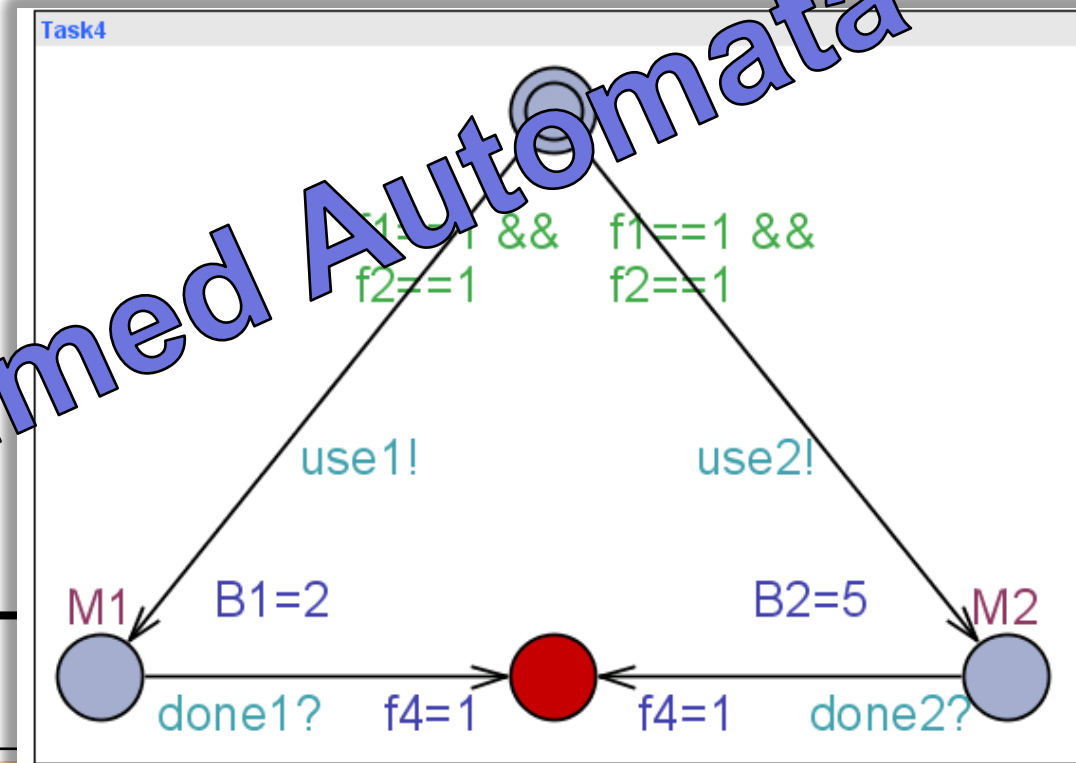
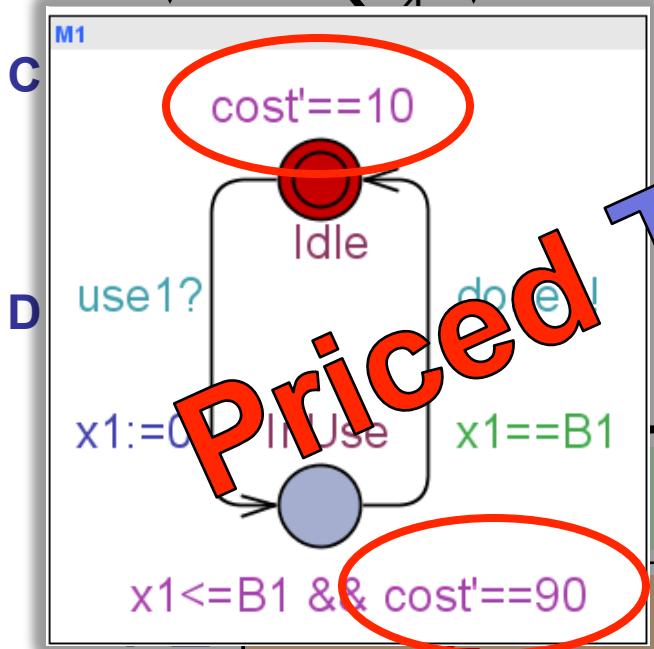
Energy:
1.32 nano-joule
OPTIMAL !!

time

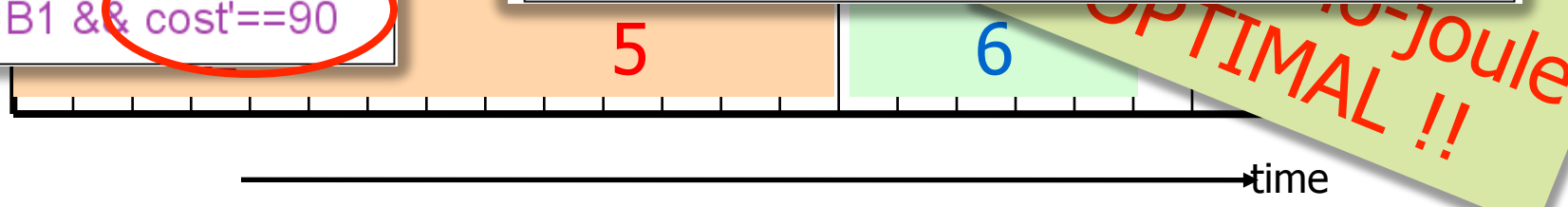
Task Graph Scheduling – Revisited



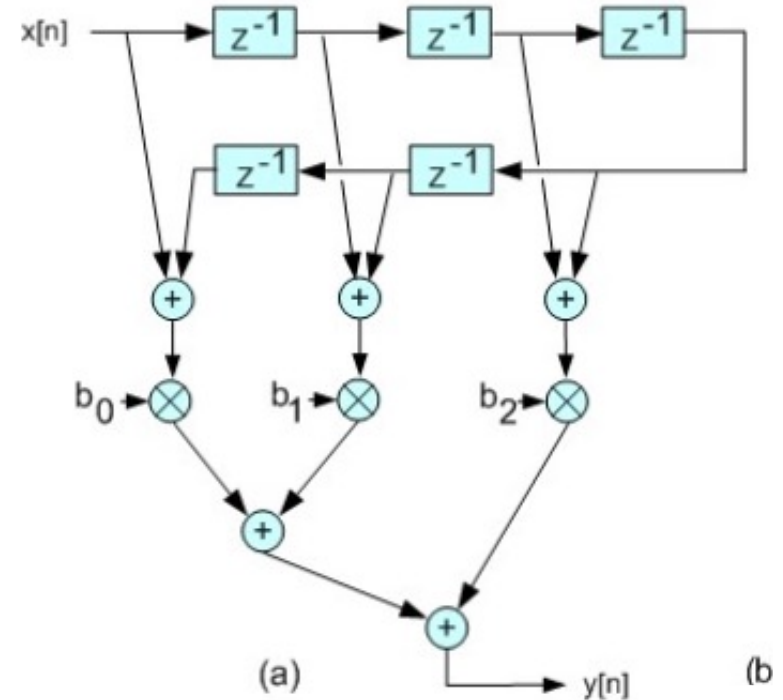
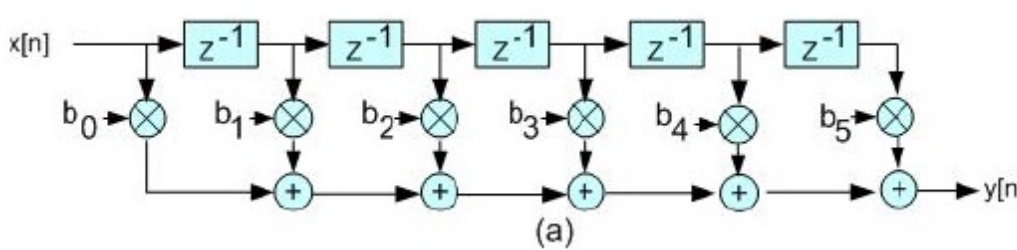
Compute :
 $(D * (C * (A + B)) + ((A + B) + (C * D)))$



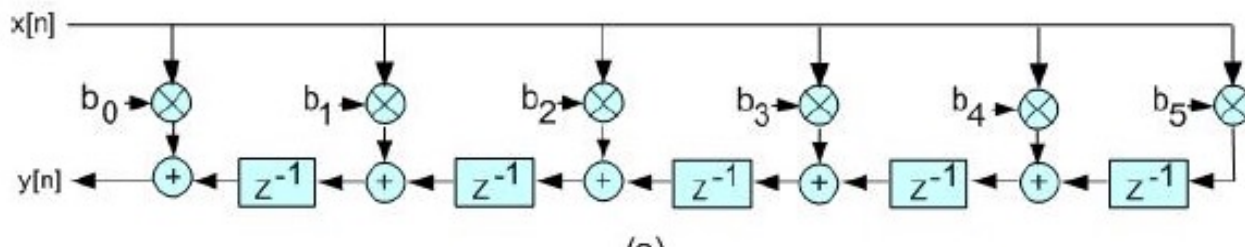
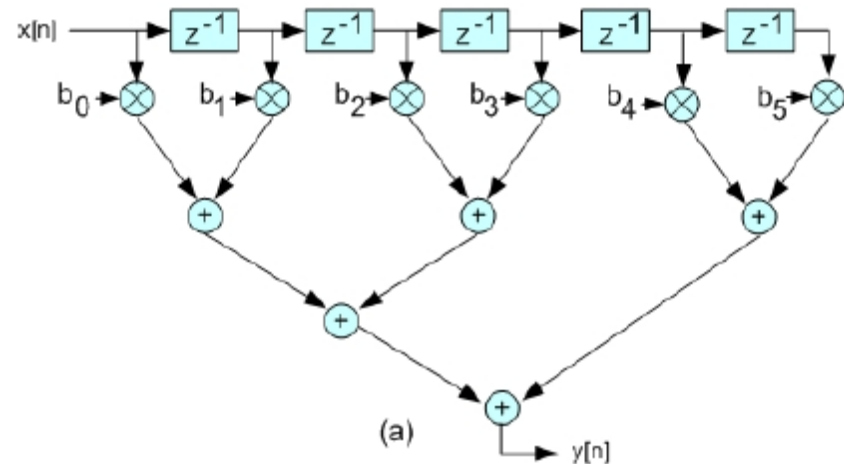
Priced Timed Automata



Energy Aware Scheduling of Finite Impulse Response filters



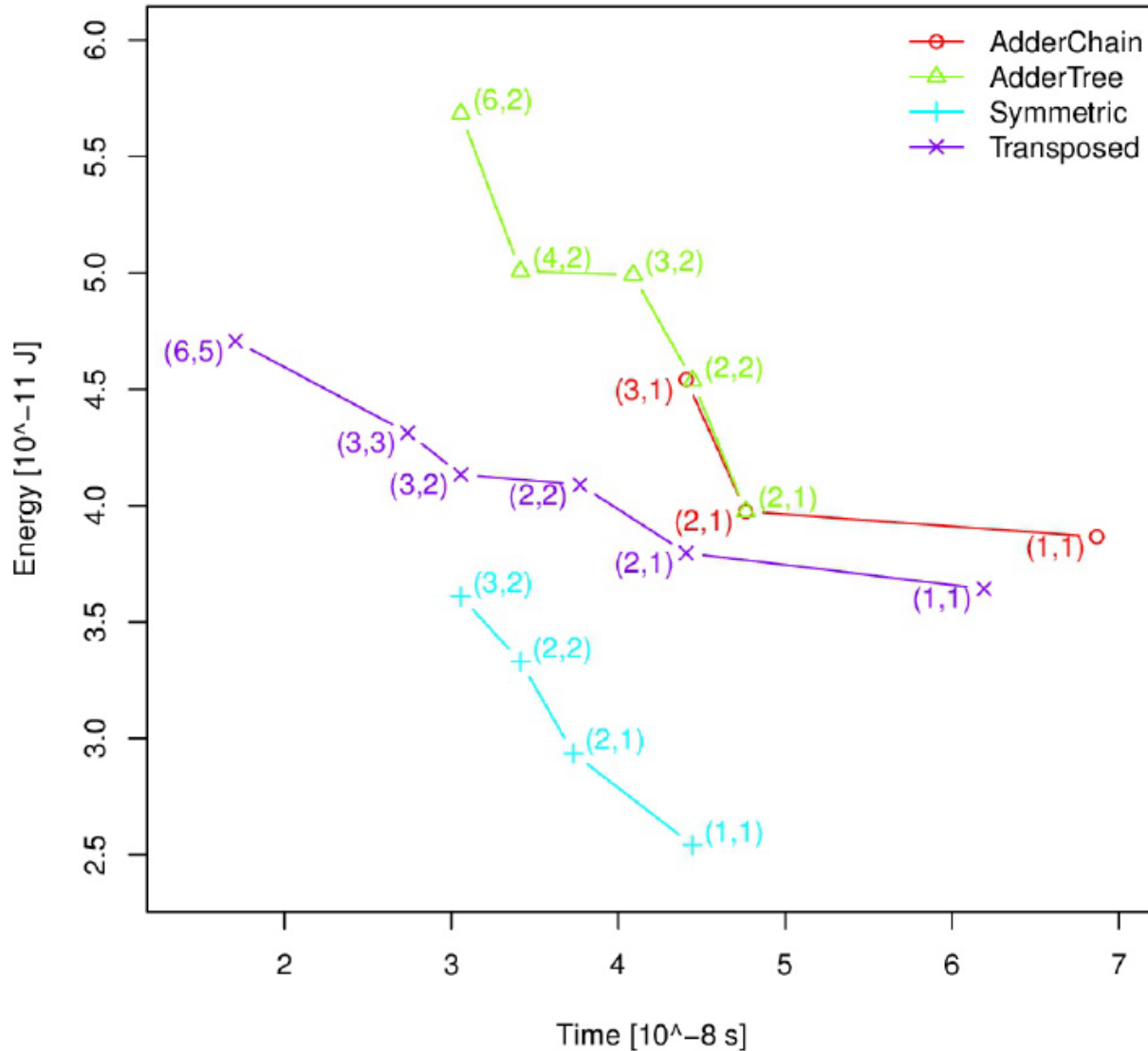
(b)



[Koch, Wognsen 15]



Energy Aware Scheduling of Finite Impulse Response filters



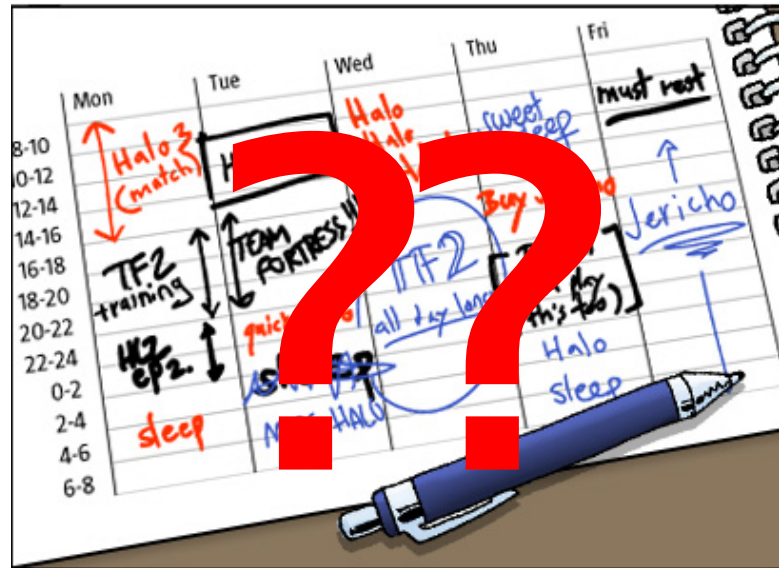
(# mult, # add)

Altera Cyclone IV
FPGA

[Koch, Wognsen 15]



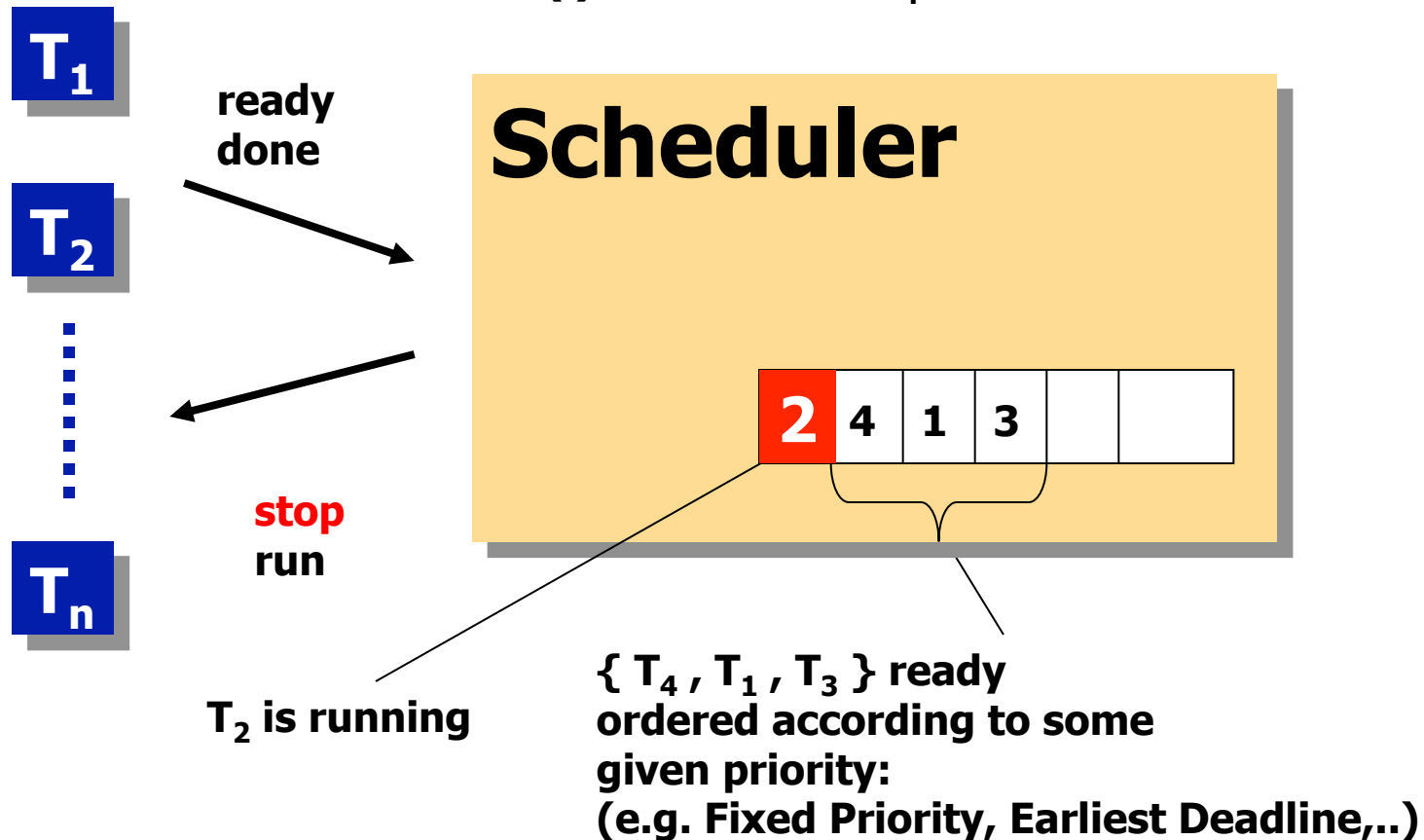
Schedulability Analysis



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

Classical WCRT Analysis



- “Classical” scheduling analysis technique
- For all tasks i : $WCRT_i \leq Deadline_i$

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

Blocking times for priority inheritance protocol (BSW):

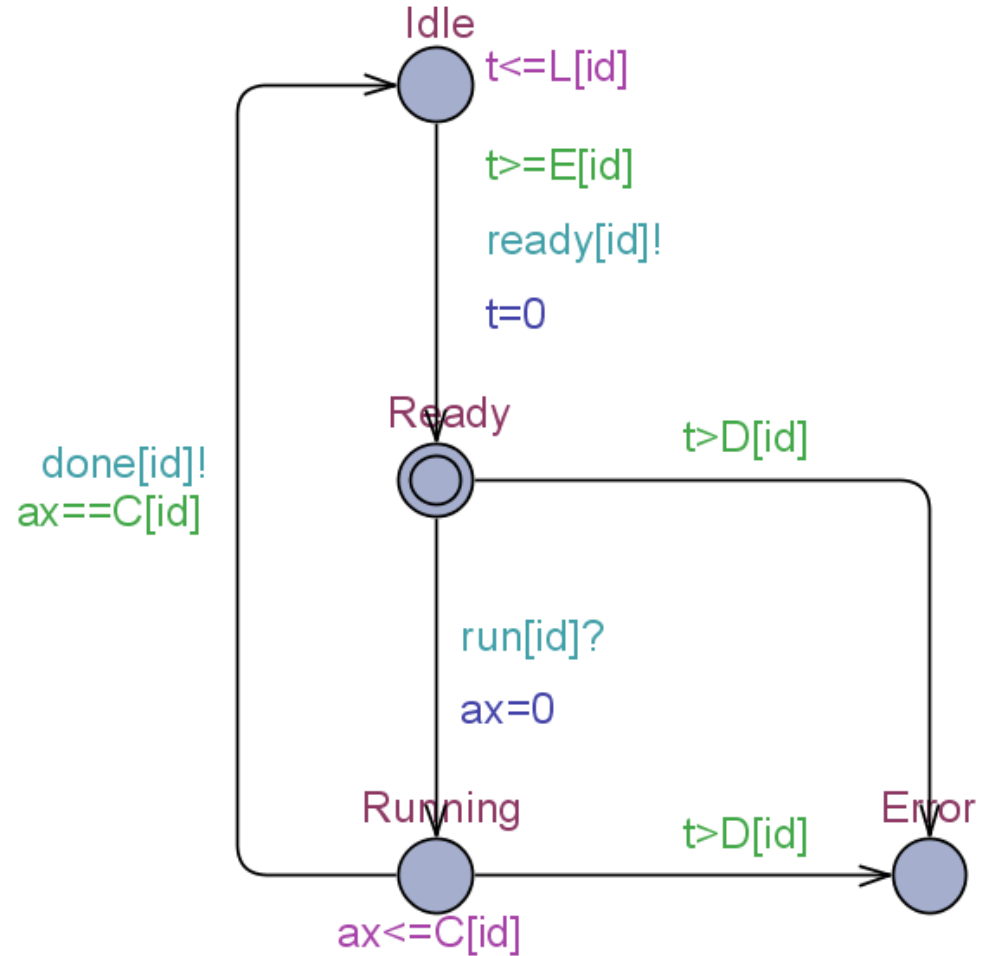
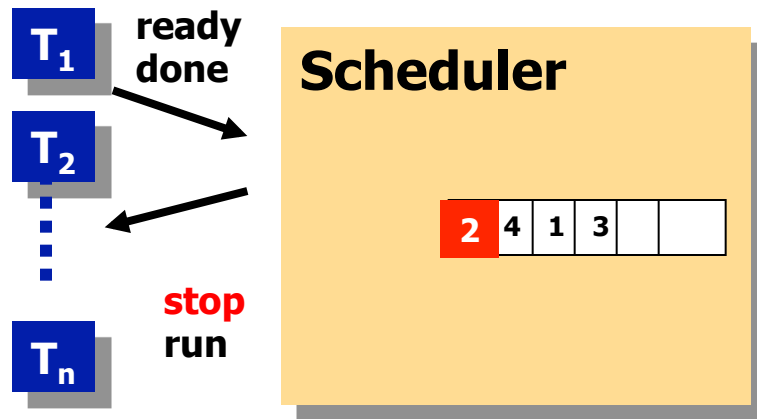
-
- $Blocking(i) = \sum_{r=1}^R usage(r, i) WCET_{CriticalSection}(r)$
-

Blocking times for priority ceiling protocol (ASW):

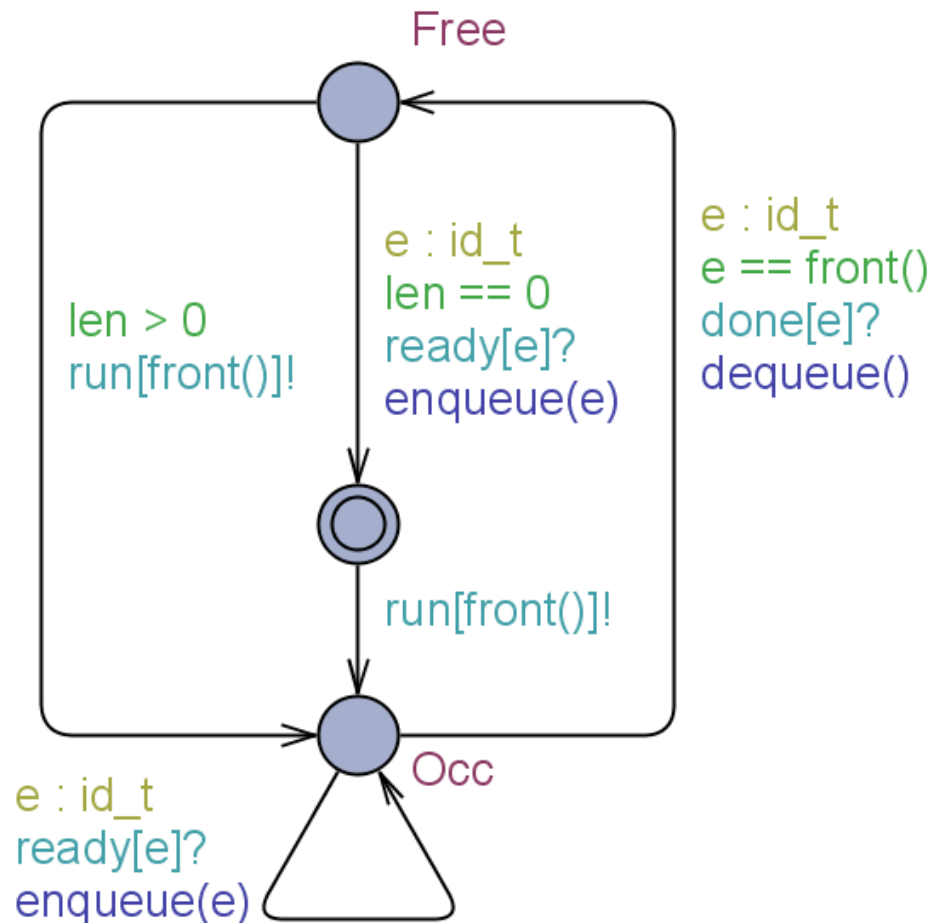
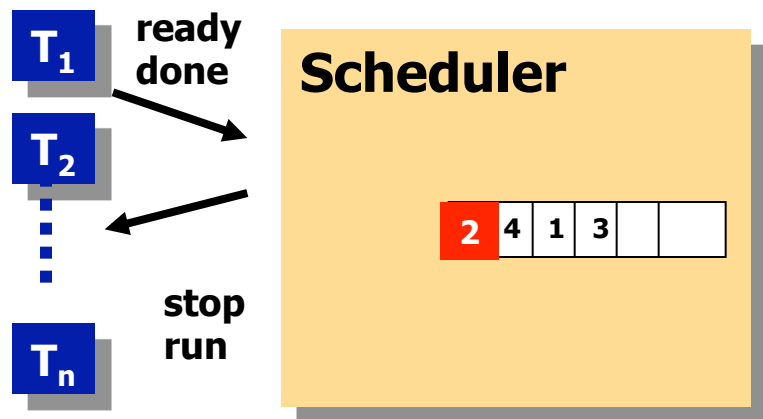
$$Blocking(i) = \max_{r=1}^R usage(r, i) WCET_{CriticalSection}(r)$$

- ✓ Simple
- conservative
- Limited settings
- Single CPU

Modeling Task

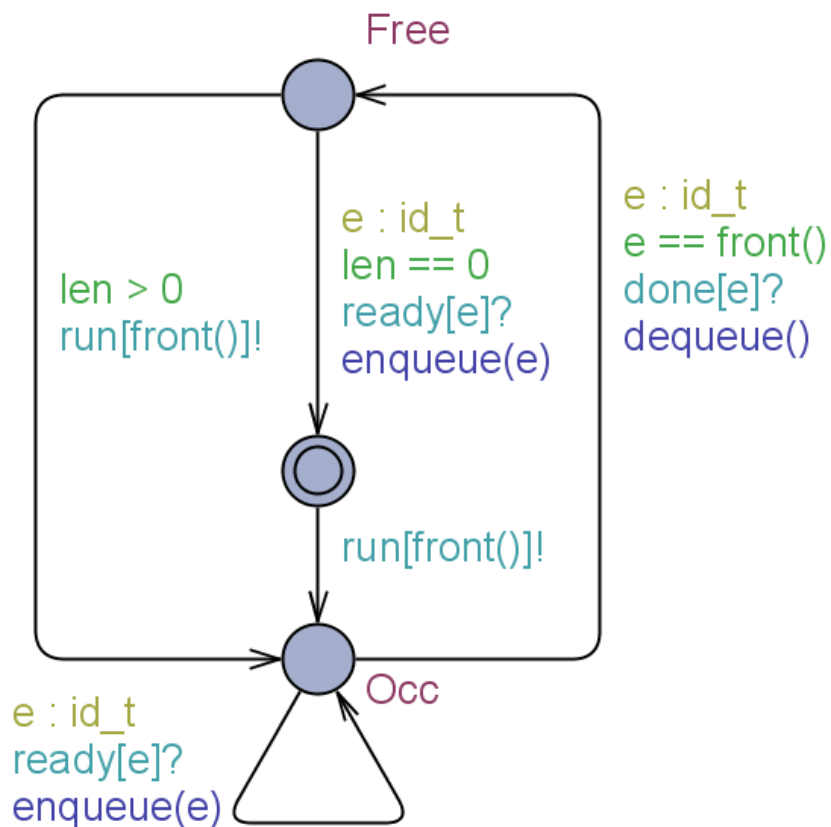


Modeling Scheduler



Modeling Queue

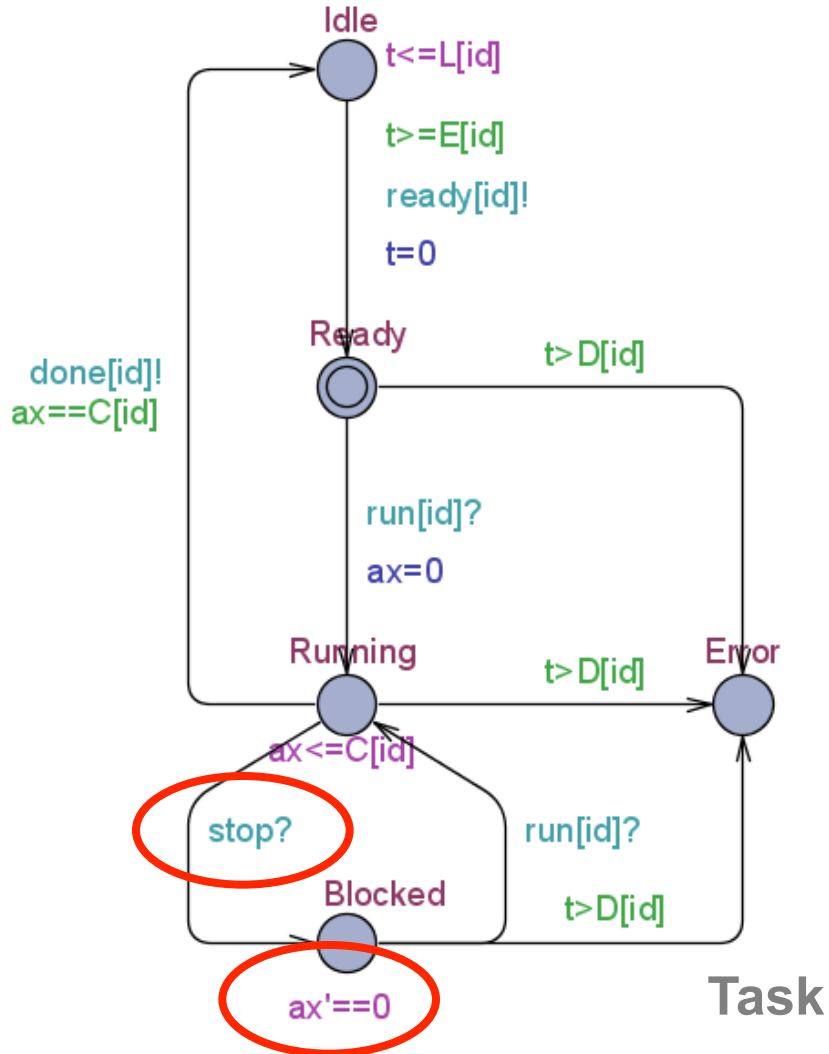
In UPPAAL 4.0
User Defined Function



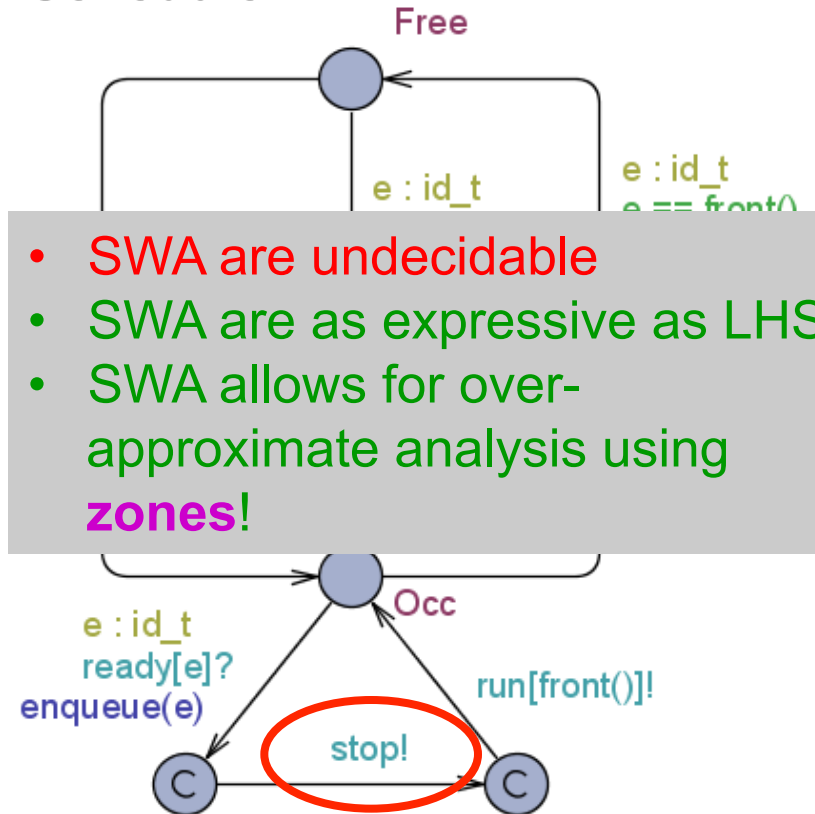
```
// Put an element at the end of the queue
void enqueue(id_t element)
{
  int tmp=0;
  list[len++] = element;
  if (len>0)
  {
    int i=len-1;
    while (i>1 && P[list[i]]>P[list[i-1]])
    {
      tmp = list[i-1];
      list[i-1] = list[i];
      list[i] = tmp;
      i--;
    }
  }
}

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

Preemption – Stopwatches!

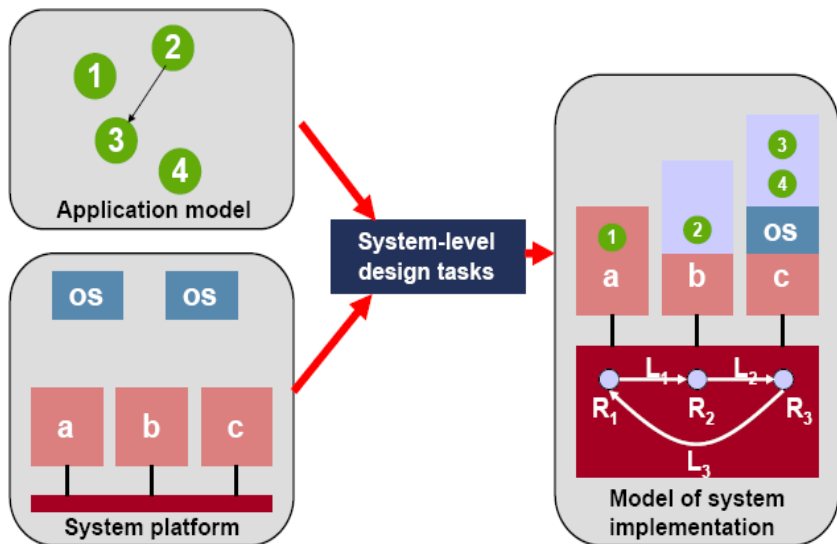


Scheduler

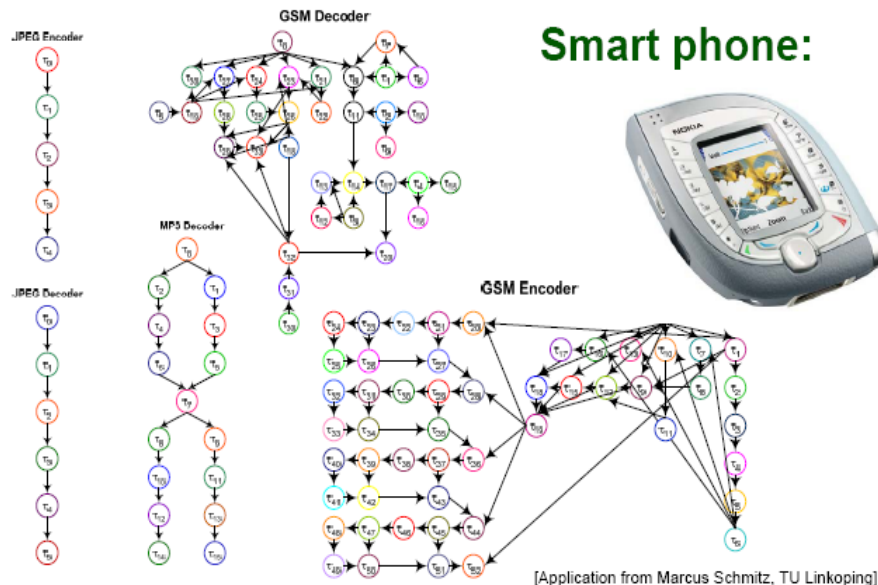


- SWA are undecidable
- SWA are as expressive as LHS
- SWA allows for over-approximate analysis using zones!

Handling real applications?

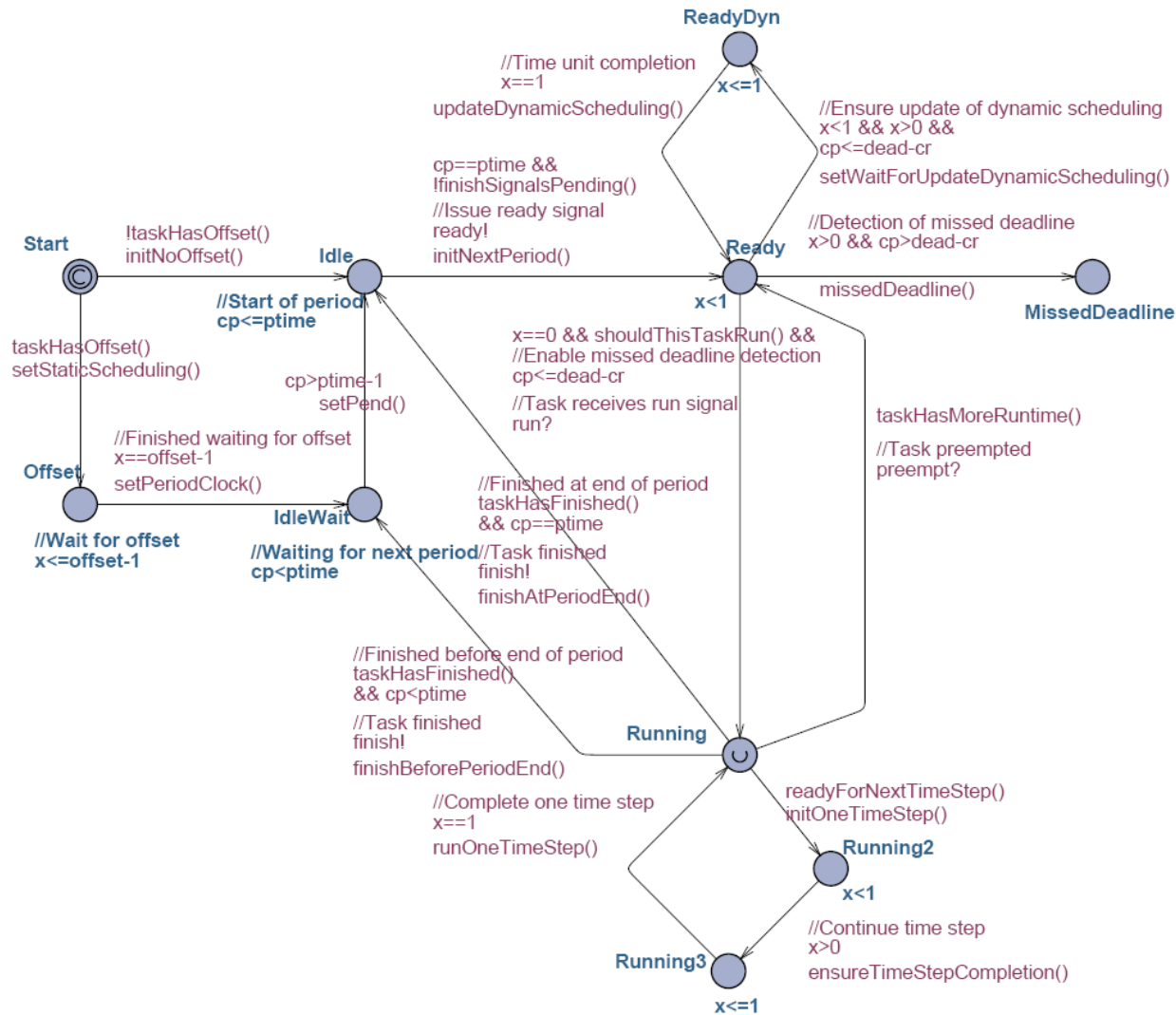


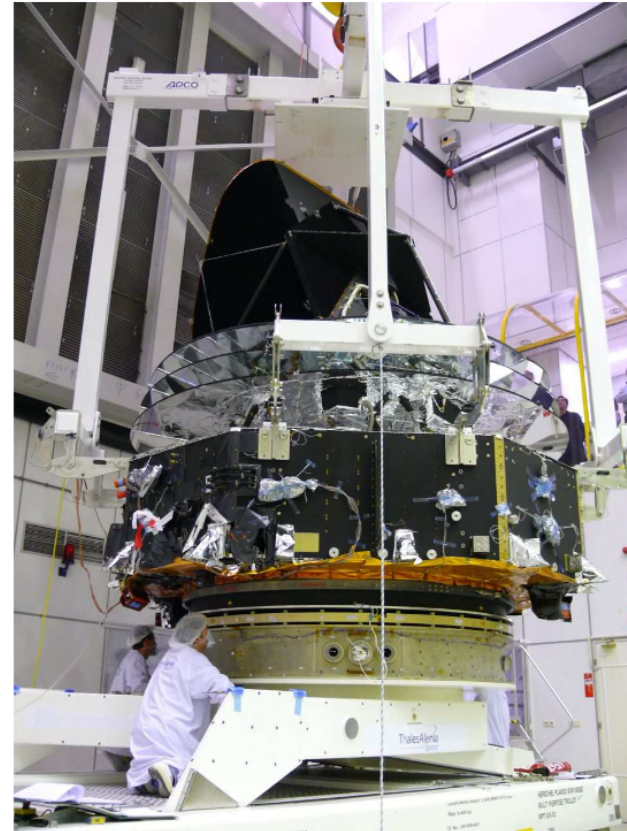
Jan Madsen, Michael R Hansen, Eske Brekling



- Tasks: 114
- Deadlines: [0.02: 0.5] sec
- Execution: [52 : 266.687] cycles
- Platform: 6 processors, 25 MHz, 1 bus
- *Verified by UPPAAL in 1 hours!*

Timed Automata for a task





Attitude and Orbit Control Software
TERMA A/S Steen Ulrik Palm, Jan Storbank Pedersen, Poul Hougaard

- **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

UPPAAL 4.1 Framework ISoLA 2010

The screenshot displays the UPPAAL 4.1 Framework interface. On the left, there are simulation controls including a transition chooser, delay settings (13.5), and speeder controls. A central list shows task queues from 0 to 33. The main area contains three state transition diagrams:

- Scheduler:** A state machine with states like 'initialize!', 'Running', 'Preempt', and 'Schedule'. Transitions are labeled with actions such as 'schedule[task]', 'release[CPU_R]', and 'enqueue?'.
- Bkgnd_P:** A state machine with states like 'starting', 'Idle', 'Ready', and 'Error'. It includes variables like 'x' and 'job[33]'.
- secondF_2:** A state machine with states like 'Idle', 'Blocked', 'WaitForCPU', and 'WaitForOther'. It features complex transitions involving 'release[CPU_R]', 'enqueue!', and 'lockCell'.

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

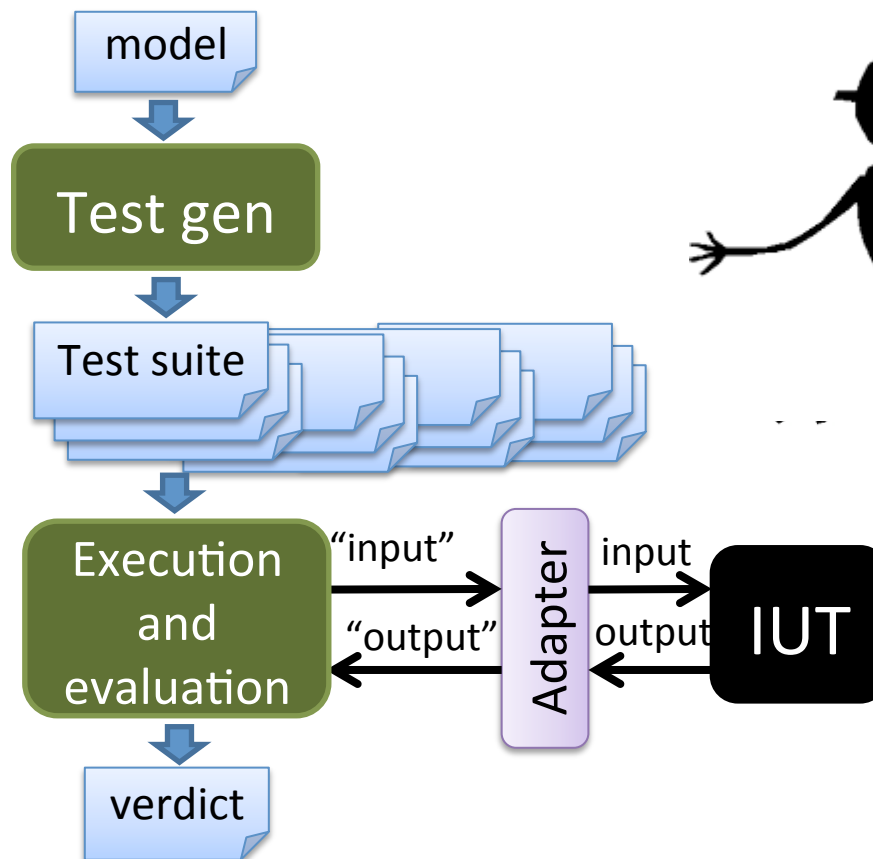
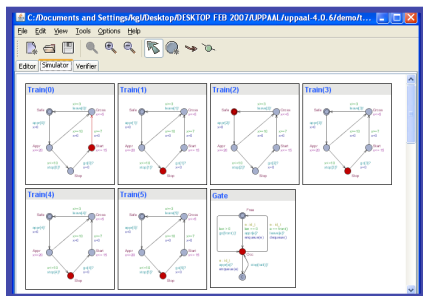
CONCLUSION

- Schedulability framework made available in UPPAAL
- Provides more exact analysis than classical methods
 - Depending on WCET information the task set is schedulable or not.
- Performance:
 - 1–2 minutes: $BCET = WCET$ or $BCET/WCET < 0.5$
 - 1 hour: $0.5 < BCET/WCET < 0.8$

Testing



Model-Based Testing



Yggdrasil (offline)

MBAT Daimler Case (2014)

The screenshot displays the Yggdrasil software interface. The main window shows a simulation trace with the following text: (wait, -, idle, -, -, -, off, off, off, off). Below the trace, there are controls for navigating through the trace, including buttons for Prev, Next, Replay, Open, Save, and Random. A speed slider is also visible, ranging from Slow to Fast.

On the right side of the interface, several state transition diagrams are displayed for different components:

- StatusCar**: A large, complex state transition diagram with many nodes and edges.
- FlashingCmd**: A smaller state transition diagram with a few nodes.
- FlashingCmdDisplayLeft**: A state transition diagram with a few nodes.
- FlashingCmdDisplayRight**: A state transition diagram with a few nodes.
- LampDisplayRight**: A state transition diagram with a few nodes.

A green callout box with a black border is overlaid on the diagrams, containing the text: **GOAL:** Cover all aspects (locations, transitions, ...)

Yggdrasil (offline)

MBAT Daimler Case (2014)

The screenshot shows the Yggdrasil software interface. The title bar indicates the file path: C:\Users\kgi\Desktop\DESKTOP12\UPPAAL\UPPAAL examples\FM Forum2014\TModelRequirement0829.xml - UPPAAL. The menu bar includes File, Edit, View, Tools, Options, and Help. The toolbar contains icons for file operations and simulation. The main window is divided into several sections:

- Options:** Includes checkboxes for "Query file" (checked), "Depth search" (checked), and "Single step" (checked). A text field contains the value "200". There are "Generate" and "Output" buttons.
- Traces:** A list of trace coverage results for various components, including "Query", "Depth", and "Single" sections. The total coverage is highlighted in blue as "Total Coverage: 84/84".
- Trace statistics:** A list of statistics for various components, including "Locations" (48/48 = 100%), "Environment.wait", "MessageHandling.L0", and various "StatusCar.L" components.
- Outout folder:** A text field containing the path "C:\Users\kgi\Desktop\DESKTOP12\UPPAAL\uppaal-4.1.20-beta2\testcases" and a "Browse" button.

Test Code & Output

The screenshot displays a software development environment with a state machine diagram for a lamp. The main window shows a state machine with two states: 'off' and 'on'. The transitions are as follows:

- From 'off' to 'on': `lights_onoff[light]==1` and `TIM_ON?`
- From 'on' to 'off': `lights_onoff[light]==1` and `TIM_OFF?`

An 'Edit Edge' dialog box is open, showing the 'Test Code' tab with the following code:

```
turnLightOff();
```

The background window shows a project structure with 'Lamp' selected, and a toolbar with various editing tools. The main window title is 'C:\Users\kg\Desktop\DESKTOP12\UPPAAL\UPPAAL examples\FM Forum2014\TIMModelRequirement0829.xml - ...'.

Test Code & Output

The image shows a software development environment with a code editor, a dialog box, and a code block.

The code editor displays a project structure with a file named "Lamp". The code in the editor is:

```
DrumScaleOneWeighing_IsInTopPos = 2;  
DrumScaleOneWeighing_StartSTM( &me->itsDrumScaleOneWeighing );  
HouseSettings_SetRamChar( &me->itsHouseSettings,  
HS_CURRENT_ACTIVE_SILO, (UCHAR)0);  
me->itsDrumScaleWeighingStable.IsReady_Return = 1;  
me->itsDrumScaleRollDrum.IsReady_Return = 1;  
Test_Validate( &me->itsTest, "DrumScaleOneWeighing_running",  
(UCHAR)IS_IN( &me->itsDrumScaleOneWeighing,  
DrumScaleOneWeighing_running ) );  
Test_Validate( &me->itsTest,  
"DrumScaleOneWeighing_StartOneWeighingState", (UCHAR)IS_IN(  
&me->itsDrumScaleOneWeighing,  
DrumScaleOneWeighing_StartOneWeighingState ) );  
PrintCurrentState(me);  
DrumScaleOneWeighing_StartOneWeighing(&me->  
itsDrumScaleOneWeighing, (FLOAT32)12.0, (void *) (void * const,  
FLOAT32))OneWeighingFinishedCb, (UCHAR *) (void * const))  
OkToRollDrumCb, me);  
Test_Validate( &me->itsTest,  
"DrumScaleOneWeighing_WeighingDrumEmpty", (UCHAR)IS_IN( &me->  
itsDrumScaleOneWeighing,  
DrumScaleOneWeighing_WeighingDrumEmpty ) );  
PrintCurrentState(me);  
Test_Comment( &me->itsTest, "DrumScaleRollDrum_IsInTopPos is %  
ld", (UCHAR)DrumScaleRollDrum_IsInTopPos(&me->  
itsDrumScaleRollDrum) );  
me->itsDrumScaleWeighingStable.StableWeighingFinishedCb(me->  
itsDrumScaleWeighingStable.owner, 0.6F, 9.62F );
```

The dialog box, titled "Edit Edge", has a "Test Code" tab and contains the following code:

```
turnLightOff();
```

The code block on the right contains the following code:

```
DrumScaleOneWeighing_IsInTopPos = 2;  
DrumScaleOneWeighing_StartSTM( &me->itsDrumScaleOneWeighing );  
HouseSettings_SetRamChar( &me->itsHouseSettings,  
HS_CURRENT_ACTIVE_SILO, (UCHAR)0);  
me->itsDrumScaleWeighingStable.IsReady_Return = 1;  
me->itsDrumScaleRollDrum.IsReady_Return = 1;  
Test_Validate( &me->itsTest, "DrumScaleOneWeighing_running",  
(UCHAR)IS_IN( &me->itsDrumScaleOneWeighing,  
DrumScaleOneWeighing_running ) );  
Test_Validate( &me->itsTest,  
"DrumScaleOneWeighing_StartOneWeighingState", (UCHAR)IS_IN(  
&me->itsDrumScaleOneWeighing,  
DrumScaleOneWeighing_StartOneWeighingState ) );  
PrintCurrentState(me);  
DrumScaleOneWeighing_StartOneWeighing(&me->  
itsDrumScaleOneWeighing, (FLOAT32)12.0, (void *) (void * const,  
FLOAT32))OneWeighingFinishedCb, (UCHAR *) (void * const))  
OkToRollDrumCb, me);  
Test_Validate( &me->itsTest,  
"DrumScaleOneWeighing_WeighingDrumEmpty", (UCHAR)IS_IN( &me->  
itsDrumScaleOneWeighing,  
DrumScaleOneWeighing_WeighingDrumEmpty ) );  
PrintCurrentState(me);  
Test_Comment( &me->itsTest, "DrumScaleRollDrum_IsInTopPos is %  
ld", (UCHAR)DrumScaleRollDrum_IsInTopPos(&me->  
itsDrumScaleRollDrum) );  
me->itsDrumScaleWeighingStable.StableWeighingFinishedCb(me->  
itsDrumScaleWeighingStable.owner, 0.6F, 9.62F );
```

The code block also includes a diagram of a light switch. The switch is currently in the "off" position, indicated by a yellow arrow pointing to the "off" label. The switch is labeled "light" and "TIM".

Yggdrasil Industrial Use

- Novo Nordisk
 - Reduction in time for testing a module 30 days
30 days → 2 days
- Skov A/S
- TK Validate
 - Ambitious business plan

- Evaluation at
 - Daimler
 - Infineon Austria
 - EADS
 - Bombardier
 - Cov. Inc 40%
 - Reduced test time
20% (80% for unit test)



DAIMLER



BOMBARDIER

Conclusion

- UPPAAL has been applied extensively to
 - verification, debugging, optimization and testing of industrial real-time systems.
- UPPAAL is used world-wide in academia
- Extensions:
 - UPPAAL **CORA**
(priced TA → optimization)
 - UPPAAL **TIGA**
(timed games → synthesis)
 - UPPAAL **SMC**
(stochastic TA → performance evaluation)

References (tool & theory)

- Gerd Behrmann, Alexandre David, Kim Guldstrand Larsen, Paul Pettersson, Wang Yi:
Developing UPPAAL over 15 years.
Softw., Pract. Exper. 41(2): 133–142 (2011)
- Gerd Behrmann, Patricia Bouyer, Emmanuel Fleury, Kim Guldstrand Larsen:
Static Guard Analysis in Timed Automata Verification.
TACAS 2003: 254–277
- Gerd Behrmann, Patricia Bouyer, Kim Guldstrand Larsen, Radek Pelánek:
Lower and Upper Bounds in Zone Based Abstractions of Timed Automata.
TACAS 2004: 312–326
- Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim Guldstrand Larsen, Paul Pettersson, Judi Romijn, Frits W. Vaandrager:
Minimum–Cost Reachability for Priced Timed Automata.
HSCC 2001: 147–161
- Patricia Bouyer, Uli Fahrenberg, Kim G. Larsen, Nicolas Markey:
Quantitative analysis of real–time systems using priced timed automata.
Commun. ACM 54(9): 78–87 (2011)
- Franck Cassez, Kim Guldstrand Larsen:
The Impressive Power of Stopwatches.
CONCUR 2000: 138–152

References (applications)

- Klaus Havelund, Arne Skou, Kim Guldstrand Larsen, K. Lund:
Formal modeling and analysis of an audio/video protocol: an industrial case study using UPPAAL.
RTSS 1997: 2–13
- Pedro R. D'Argenio, Joost–Pieter Katoen, Theo C. Ruys, Jan Tretmans:
The Bounded Retransmission Protocol Must Be on Time!
TACAS 1997
- Magnus Lindahl, Paul Pettersson, Wang Yi:
Formal design and analysis of a gear controller.
Software Tools for Technology Transfer 3(3) (2001)
- Aske Wiid Brekling, Michael R. Hansen, Jan Madsen:
Models and formal verification of multiprocessor system–on–chips.
J. Log. Algebr. Program. 77(1–2): 1–19 (2008)
- Marius Mikucionis, Kim Guldstrand Larsen, Jacob Illum Rasmussen, Brian Nielsen, Arne Skou, Steen Ulrik Palm, Jan Storbak Pedersen, Poul Hougaard:
Schedulability Analysis Using Uppaal: Herschel–Planck Case Study.
ISoLA (2) 2010: 175–19
- Michael Gerke, Rüdiger Ehlers, Bernd Finkbeiner, Hans–Jörg Peter:
Model Checking the FlexRay Physical Layer Protocol.
FMICS 2010: 132–147

And many many more !!!

www.uppaal.{org,com}

UPPAAL - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://www.uppaal.org/

UPPAAL

RELATED TOOLS: TIMES | CORA | TRON | TIGA | SMC | COVER | PORT | PRO

UPPAAL

Home

Home | About | Documentation | Download | Examples | Web Help | Bugs

UPPAAL is an integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed automata, extended with data types (bounded integers, arrays, etc.).




The tool is developed in collaboration between the [Department of Information Technology](#) at Uppsala University, Sweden and the [Department of Computer Science](#) at Aalborg University in Denmark.

Download

News: The current official release is UPPAAL 4.0.13 (Sep 27, 2010). Compared to version 3, the 4.0 release is the result of over 2.5 years of additional development, and many new features and improvements are introduced (see also this [release note](#) and the web help section [new features](#)). To support models created in previous versions of UPPAAL, version 4.0 can convert most old models directly from the GUI (alternatively it can be run in 3.4 compatibility mode by defining the environment variable `UPPAAL_OLD_SYNTAX`, see also item 2 of the [FAQ](#)).

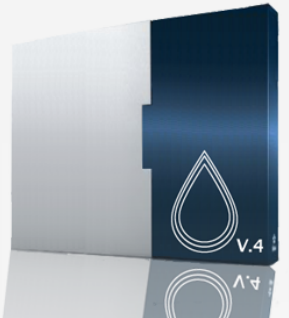
Since Feb 26 2008, we also distribute a development snapshot of the forthcoming UPPAAL 4.2. The current development snapshot version is 4.1.4 released Jul 11, 2011.

Figure 1: UPPAAL on screen.



DESIGN VERIFICATION FOR EMBEDDED SYSTEMS

Our world-leading and internationally acclaimed model-checking tool UPPAAL is now available for commercial use!



HOME PRODUCT SOLUTIONS PARTNERS SUPPORT WEB HELP CONTACT US ABOUT UP4ALL

DOWNLOAD & BUY UPPAAL SOFTWARE

NON PROFIT USER?

NEWS & EVENTS
Feb. 15, 2011 - UP4ALL in OEM agreement with Elivor OU.

SUCCESSFUL USECASES
See how UPPAAL is used to verify industrial systems.

P.O. Box 337, SE-75105 Uppsala, Sweden

+46 21 151741

sales@uppaal.com

login

Copyright © 2009 Up4All International AB. All rights reserved