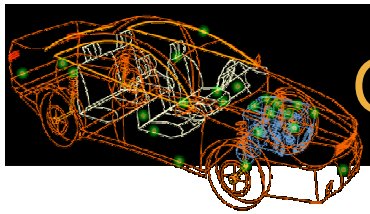
From analysis to optimization in the deployment of real-time distributed functions in modern automotive systems.

Marco Di Natale

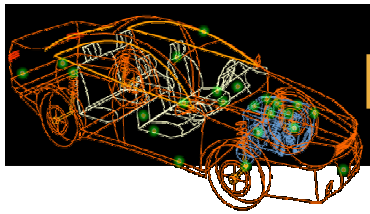
Research work performed with ***Abhijit Davare, Paolo Giusto, Claudio Pinello Alberto Sangiovanni-Vincentelli, Wei Zheng, Qi Zhu,***

Scuola Superiore S. Anna,
GM Research and Development
University of California at Berkeley,



Outline

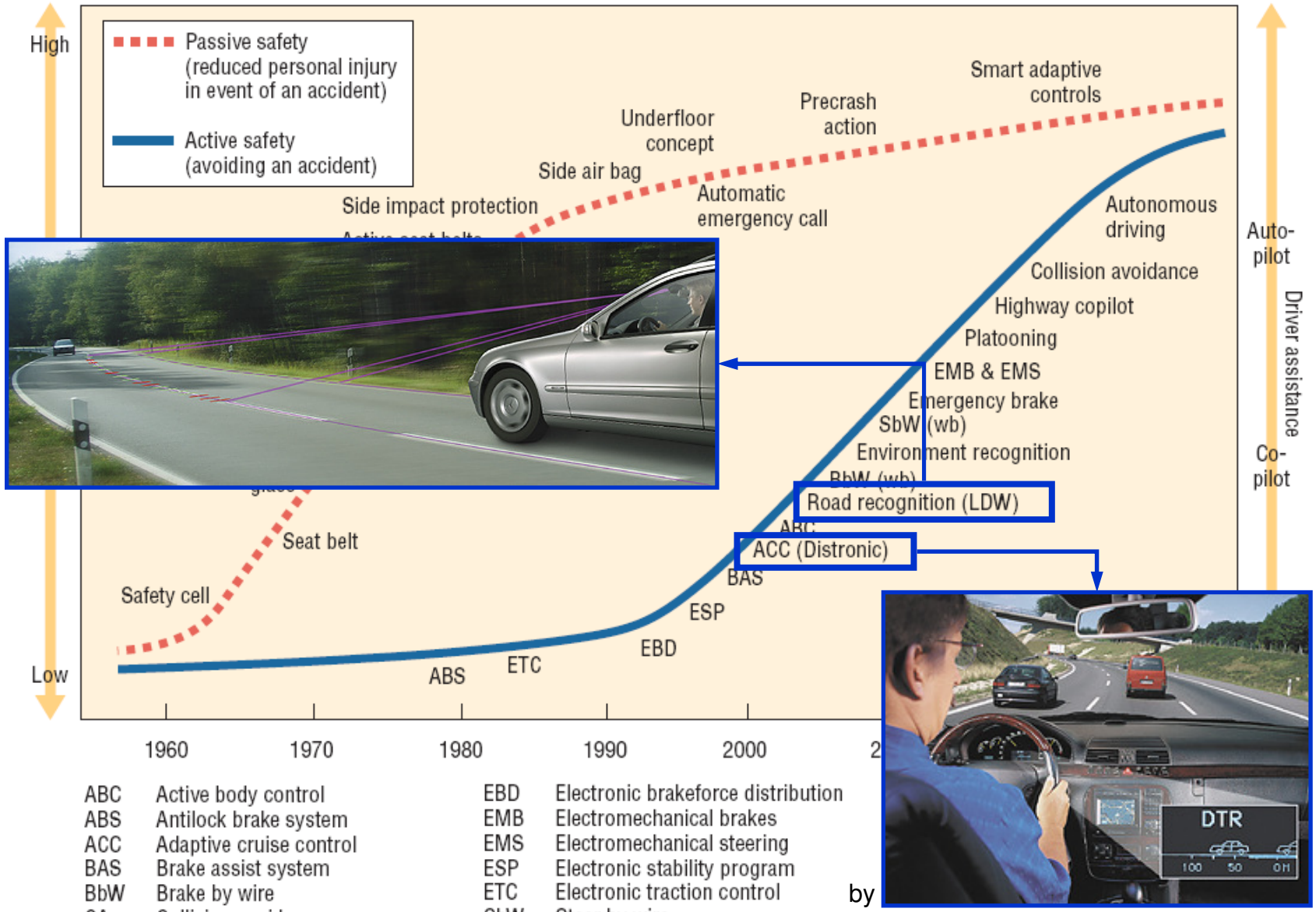
- Automotive architecture trends and challenges
- Architecture evaluation and selection: from analysis to synthesis
- AUTOSAR and *synthesis of task models*
- Activation models and end-to-end latencies
- Problem definition: feasibility region
- MILP Optimization
- Linearization of worst-case response time and exact analysis
- Metrics, Statistical and Stochastic analysis
- Case Studies
- Way to go …



Before we start ….

- **Maybe not your Synthesis**
- A lot of attention has been (rightfully) dedicated to the synthesis of SW implementation (code) starting from models
- This talk is not about the synthesis of code, but the synthesis of a task model (against real-time constraints) starting from a functional model (possibly SR/Simulink)
  - Next level up?
- The problem is quite complex and has several dimension
- Not suprisingly, several (semantics preservation) constraints apply
- *Working on an architecture without assumptions on time synchronization (TTA or LTTA, time-based scheduling or priority-based scheduling)*

Functions: Active and Passive Safety

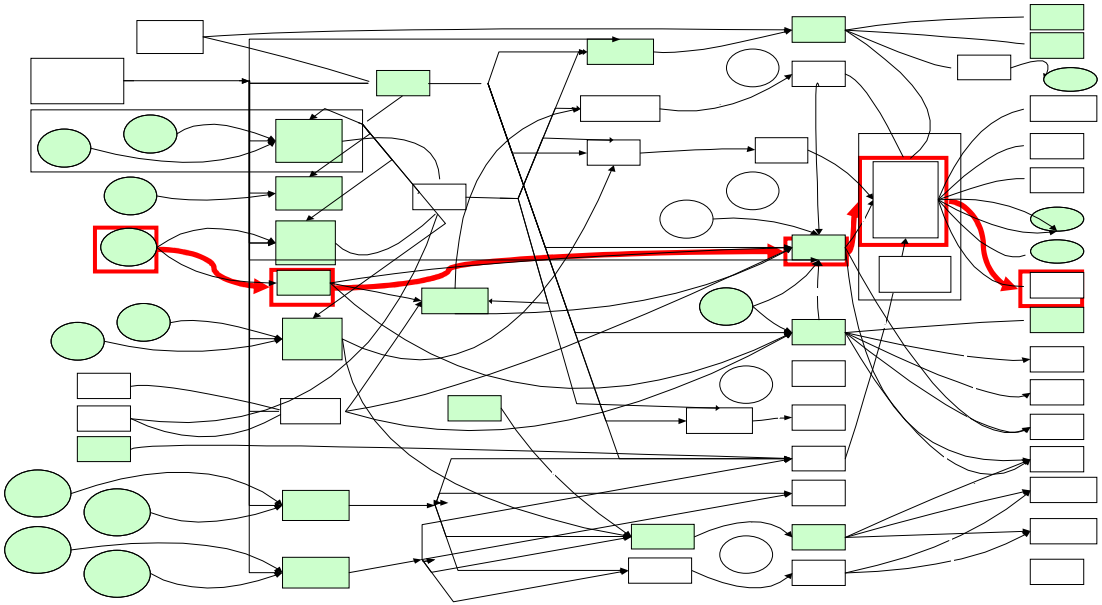


ABC	Active body control	EBD	Electronic brakeforce distribution
ABS	Antilock brake system	EMB	Electromechanical brakes
ACC	Adaptive cruise control	EMS	Electromechanical steering
BAS	Brake assist system	ESP	Electronic stability program
BbW	Brake by wire	ETC	Electronic traction control
CA	Collision avoidance	SbW	Steer by wire
DbW	Drive by wire	(wb)	with mechanical backup

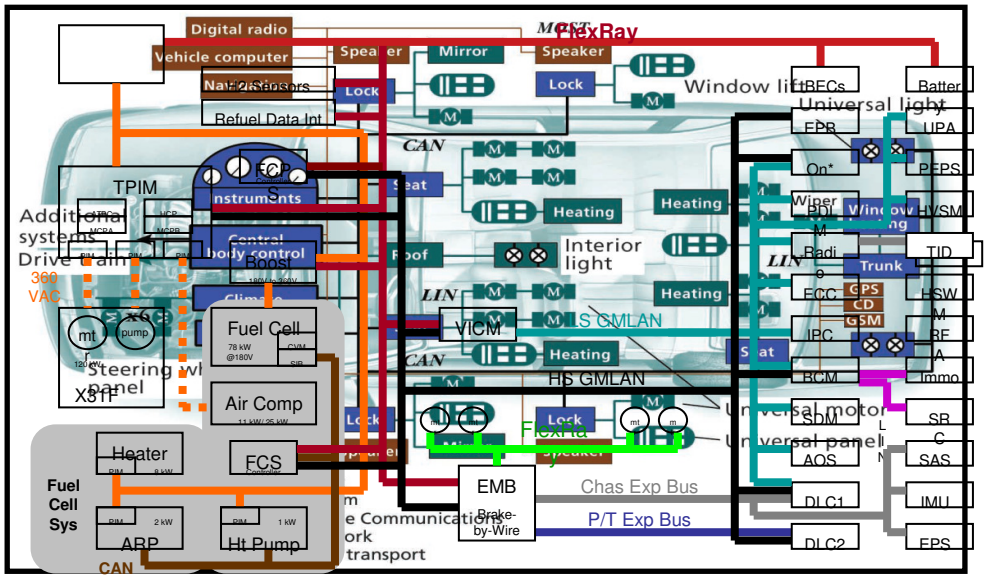
by

Distributed Systems of Integrated Functions

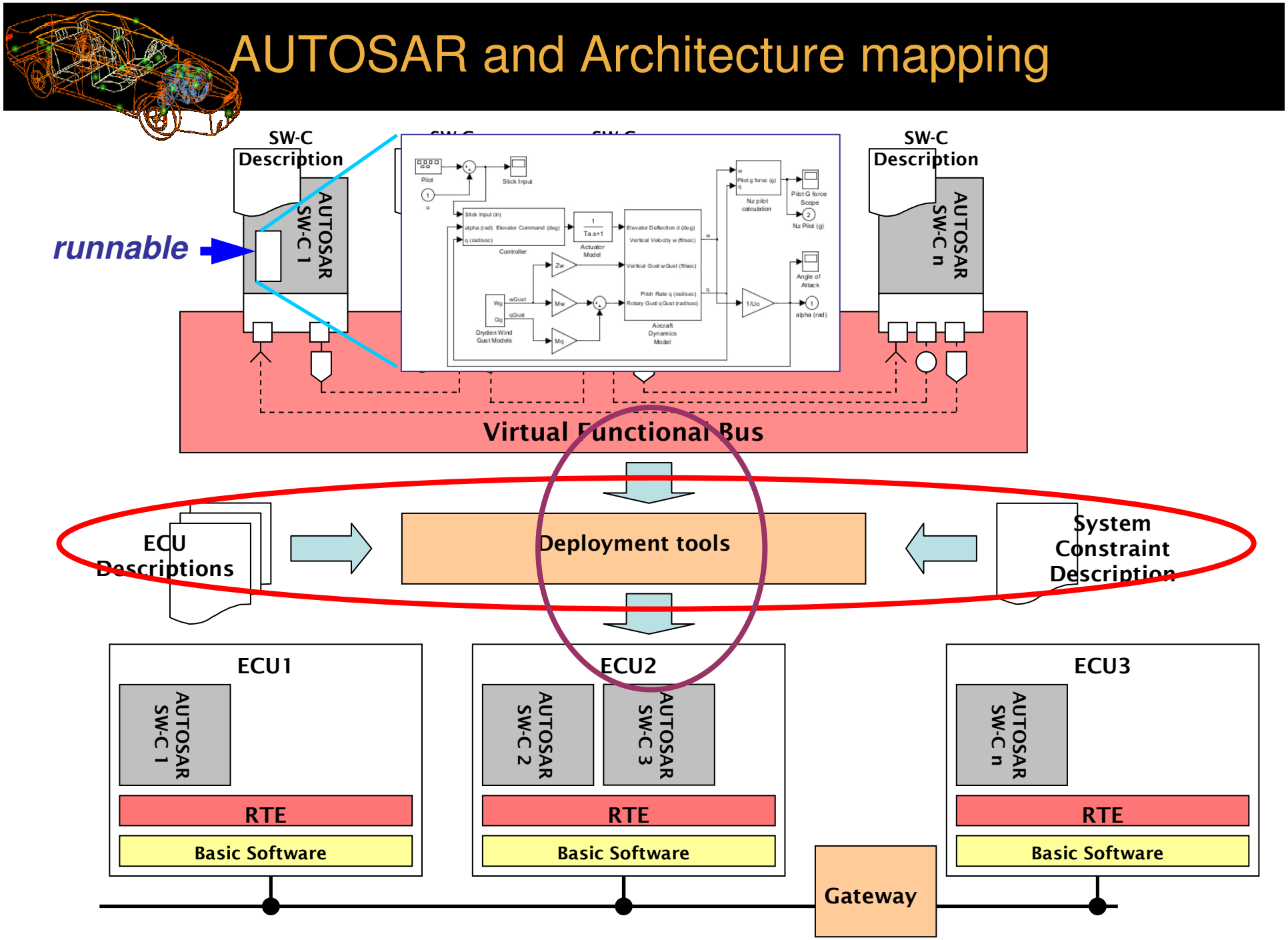
Complex functional graphs



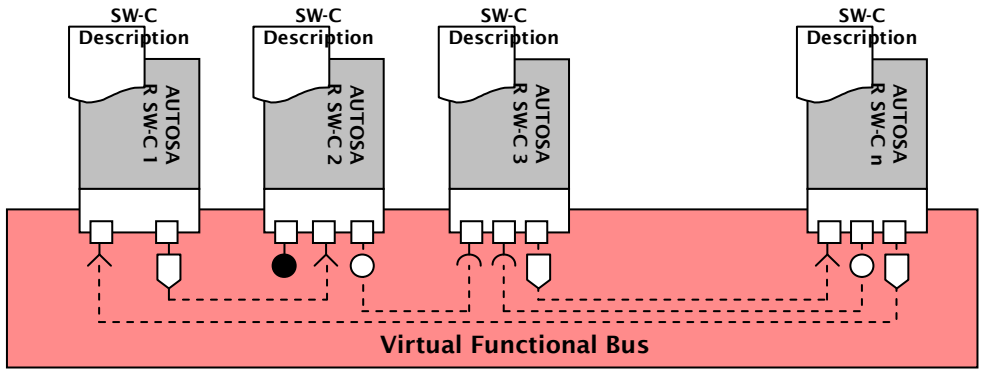
Mapped into complex distributed architectures

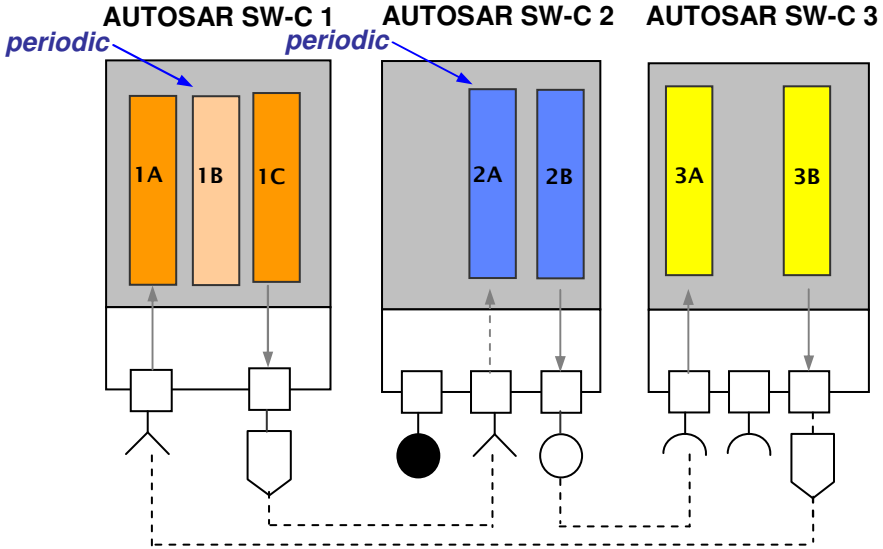


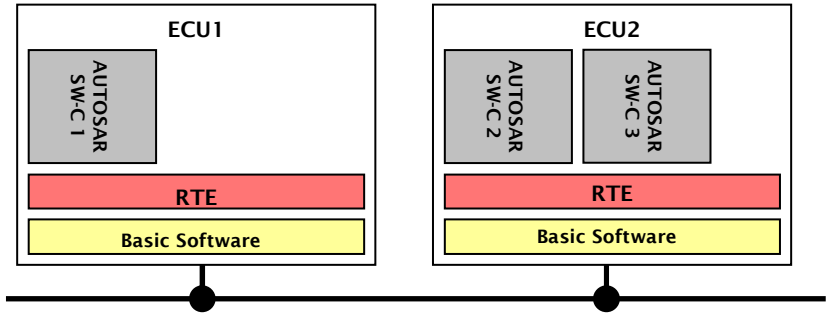
AUTOSAR and Architecture mapping

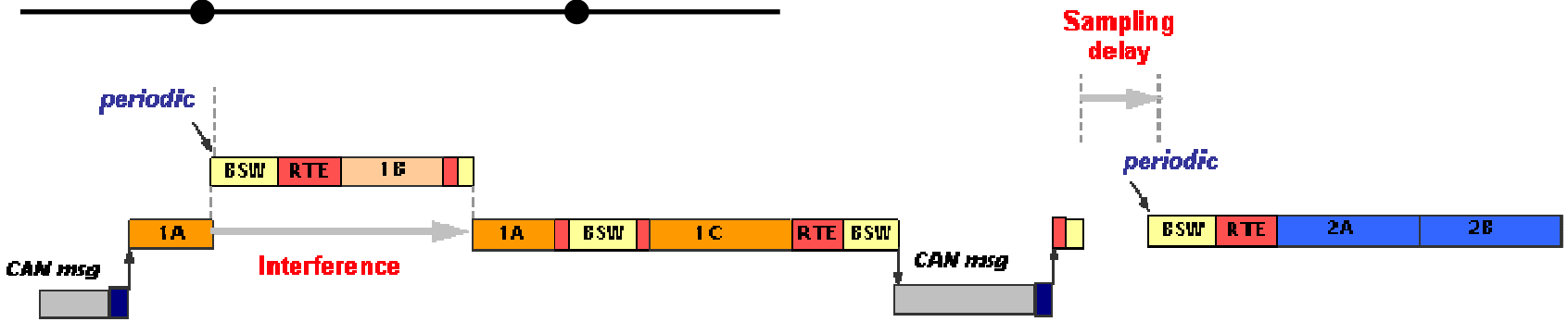


AUTOSAR 4.0 and the “Large scale” problem





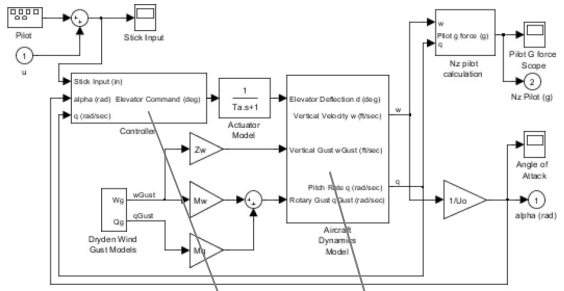




Models and deadlines

Designers are typically faced with this problem

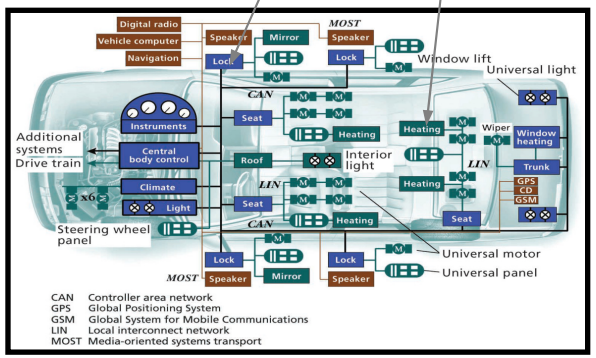
Given a model of the functions that I would like for my system, with their time constraints (T_i and D_i) …



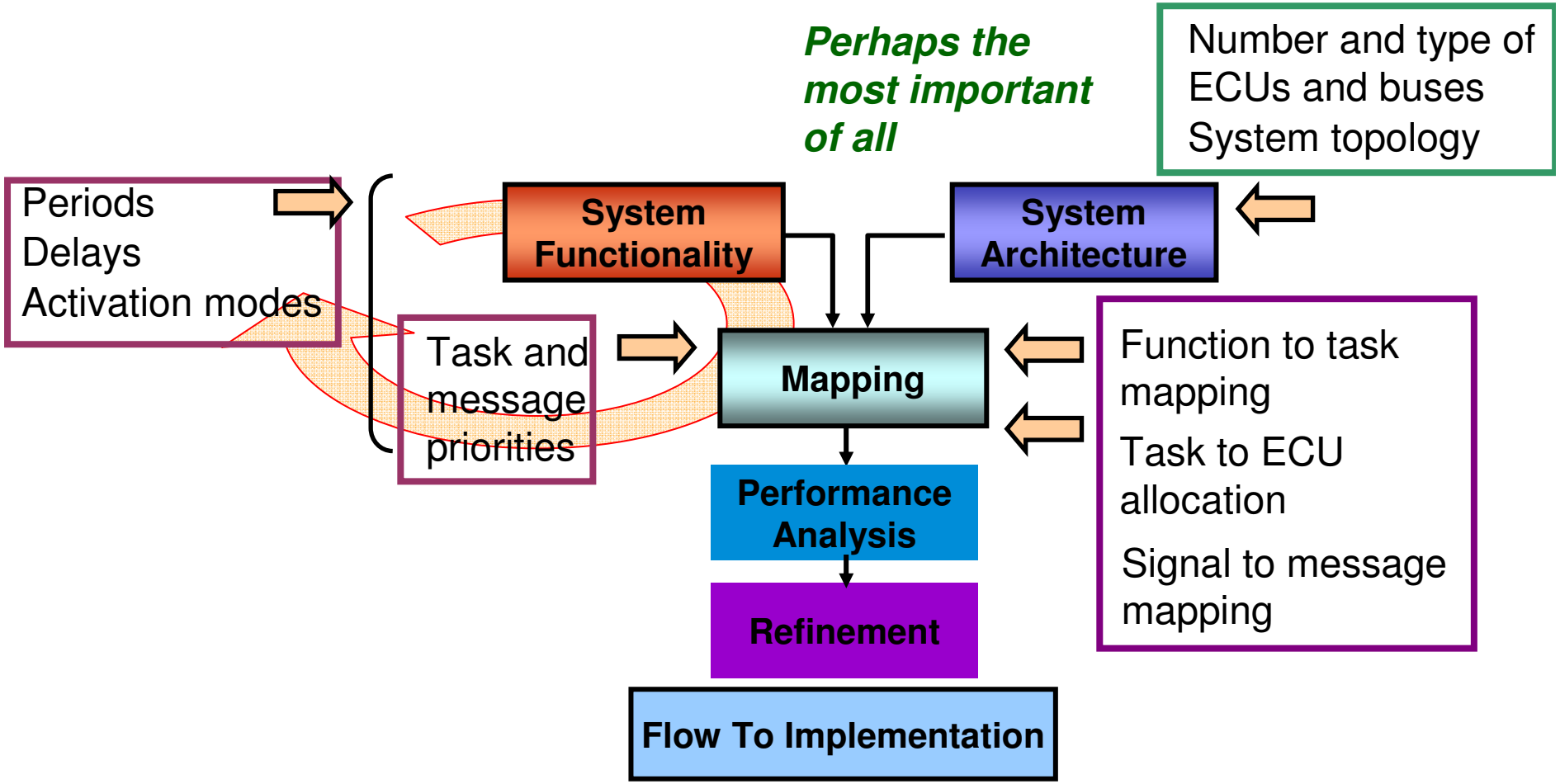
… and a SW architecture, with the task and message
functional
(extensibi
optimized

A design optimization (or synthesis) problem, rather than an analysis problem

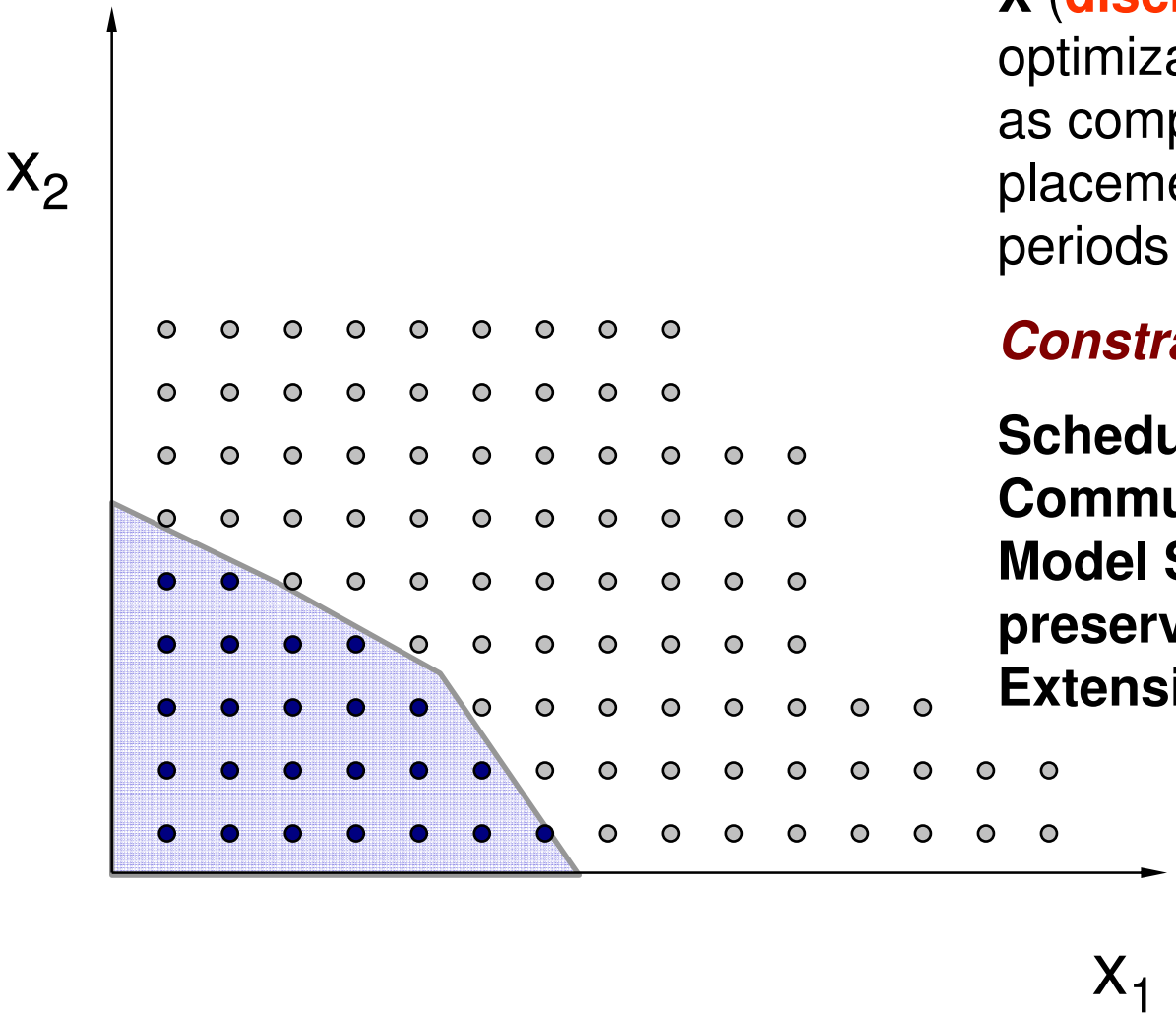
Define an execution architecture (possibly composing platform subsystems) …(C_i)



Opportunities for synthesis



Design Optimization: a simplistic view

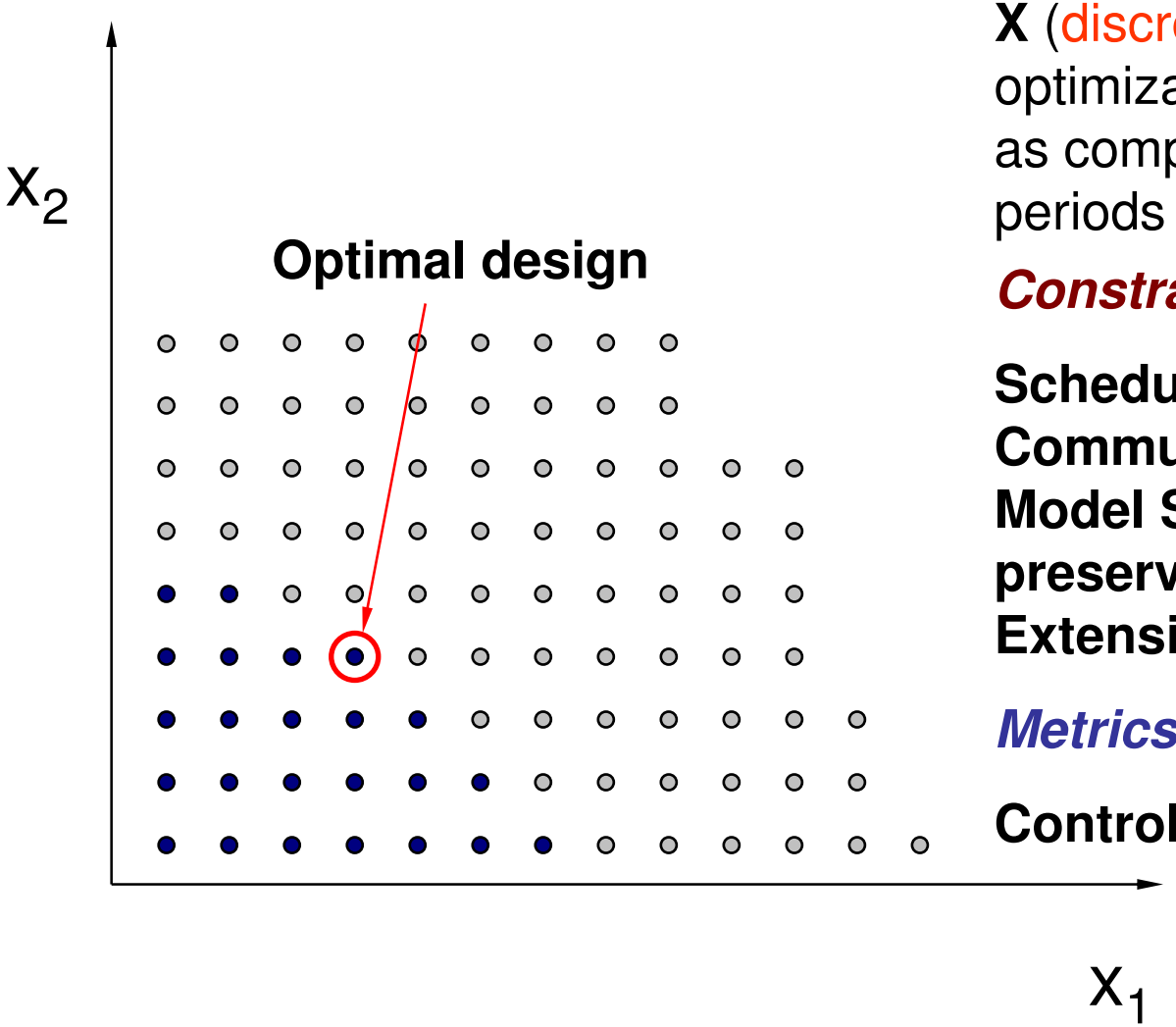


X (**discrete**) space of design optimization variables, such as computation times, placement, priorities, periods …

Constraints

Schedulability
Communication
Model Semantics preservation
Extensibility

Design Optimization: a simplistic view



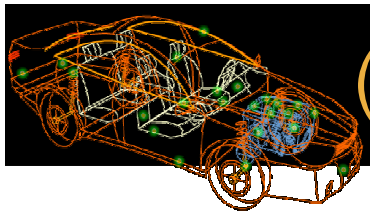
X (discrete) space of design optimization variables, such as computation times, periods …

Constraints

Schedulability
Communication
Model Semantics preservation
Extensibility

Metrics

Control related



(Example) Problem Formulation

Objective

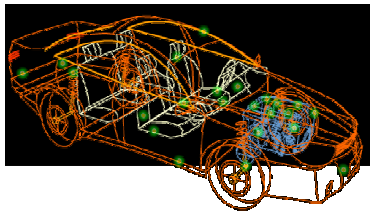
Minimization of (average case) end-to-end latencies ***?***

Subject to

- ***Constraints on end-to-end latencies***
- Constraints on messages size
- Constraints on utilization
- Constraints on message and task deadlines
- ***Semantics preservation constraints***

Design objectives (optimization variables)

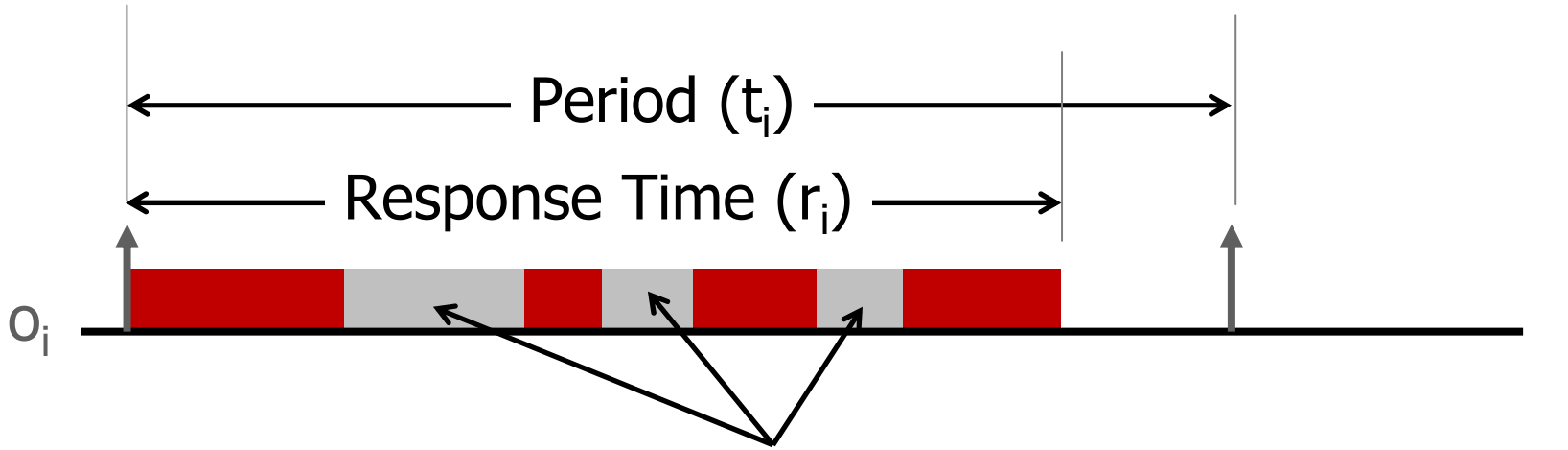
- Placement of tasks onto the CPUs
- Packing of signals to messages
- Assignment of priorities to tasks and messages
- Definition of activation modes/synchronization model
- Period optimization



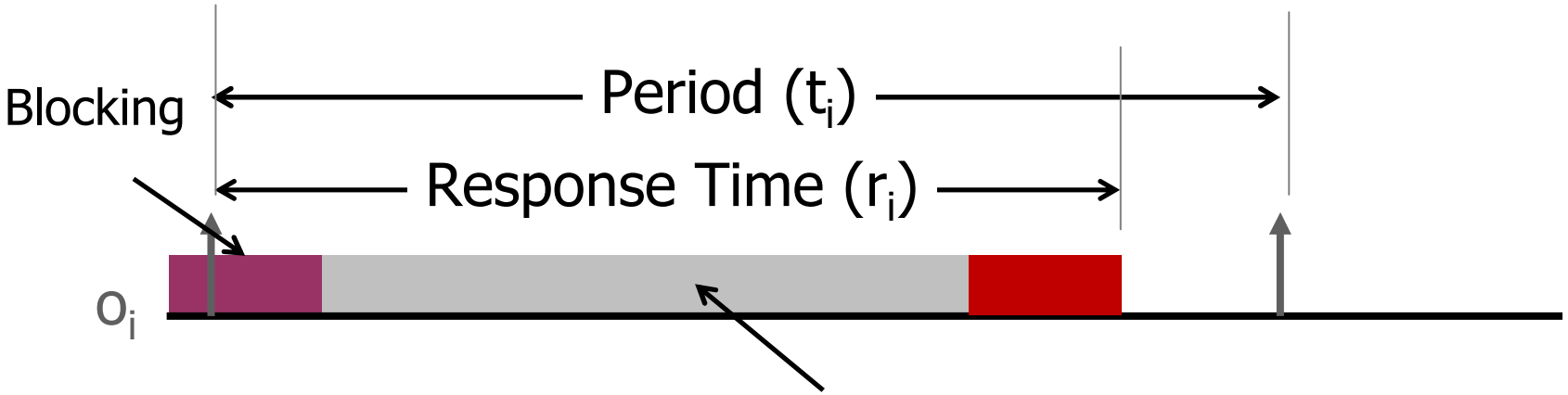
Approach: Mathematical Programming

- We would like to use Mathematical Programming (Convex Optimization, MILP, MIGP …)
  - Simplicity
    - Problem represented with:
      - Set of decision variables
      - Constraints
      - Objective function
    - “automatically” handles cross dependency among selection choices
  - Possible coding of multi-objective optimization
  - Standardized approach
    - Well established technique
    - Sound theory, methods
    - Availability of commercial solvers (in essence, search engines)
  - How good is your solution?
    - Provides safe estimate of optimal solution
    - Provides intermediate solutions of increasing quality
- Challenge:
  - Capture the problem and obtain efficient runtimes

Worst case analysis (Schedulability)



Interference I from higher priority tasks on the same resource



Interference I from higher priority msgs on the same resource

Worst Case Response Times

Not linear (or convex), not even for the single-processor case

Tasks: $$r_i = c_i + \boxed{\sum_{j \in hp(i)} \left\lceil \frac{r_i}{t_j} \right\rceil c_j} \qquad \forall o_i \in \mathcal{T}$$

Messages: $$r_i = c_i + b_i + \boxed{\sum_{j \in hp(i)} \left\lceil \frac{r_i - c_i}{t_j} \right\rceil c_j} \qquad \forall o_i \in \mathcal{M}$$

I_i

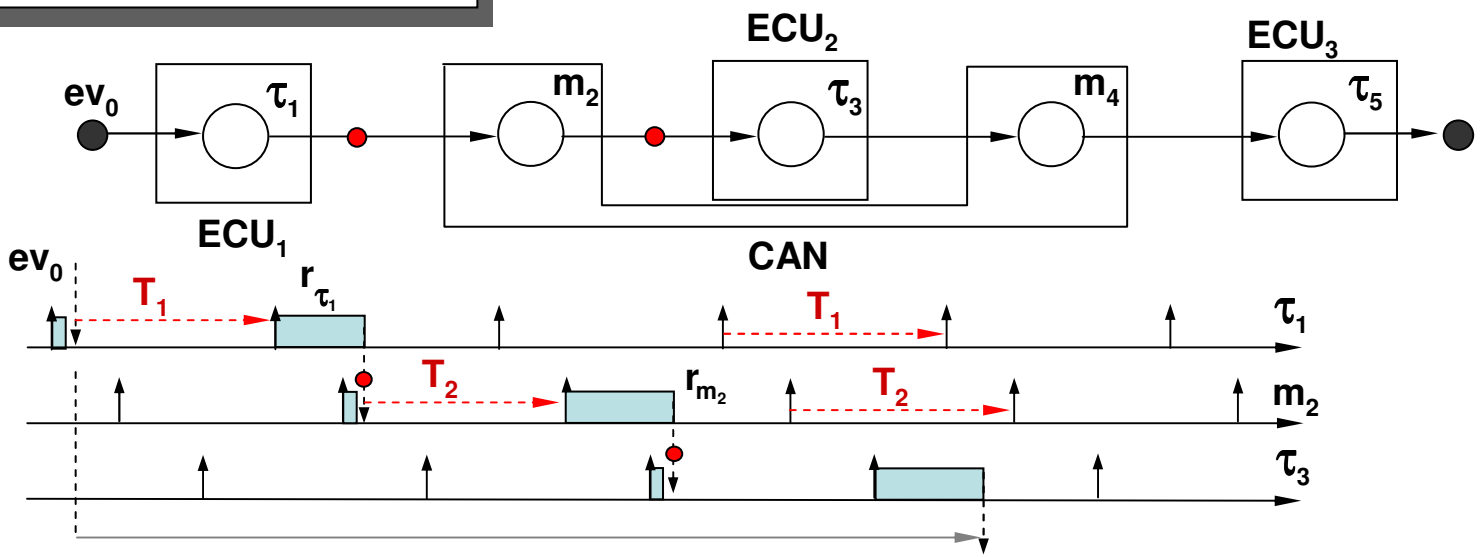
- Resource utilization
  - Fraction of time the resource (ECU or bus) spends processing its objects (tasks or messages)
- Utilization bounds less than x (with x<100%)
  - To allow for future extensibility

$$\left(\sum_{i:o_i \to R_j} \frac{c_i}{t_i} \right) \leq u_j \qquad \forall R_j \in \mathbf{R}$$

Periodic Activation Model

High latency, but allows decoupling the scheduling problem

End-to-end latency analysis ← Periodic asynchronous activation model



$$l_{(i,j)} = \sum_{k:o_k \in P(i,j)} (T_k + r_k)$$

where (approx.)

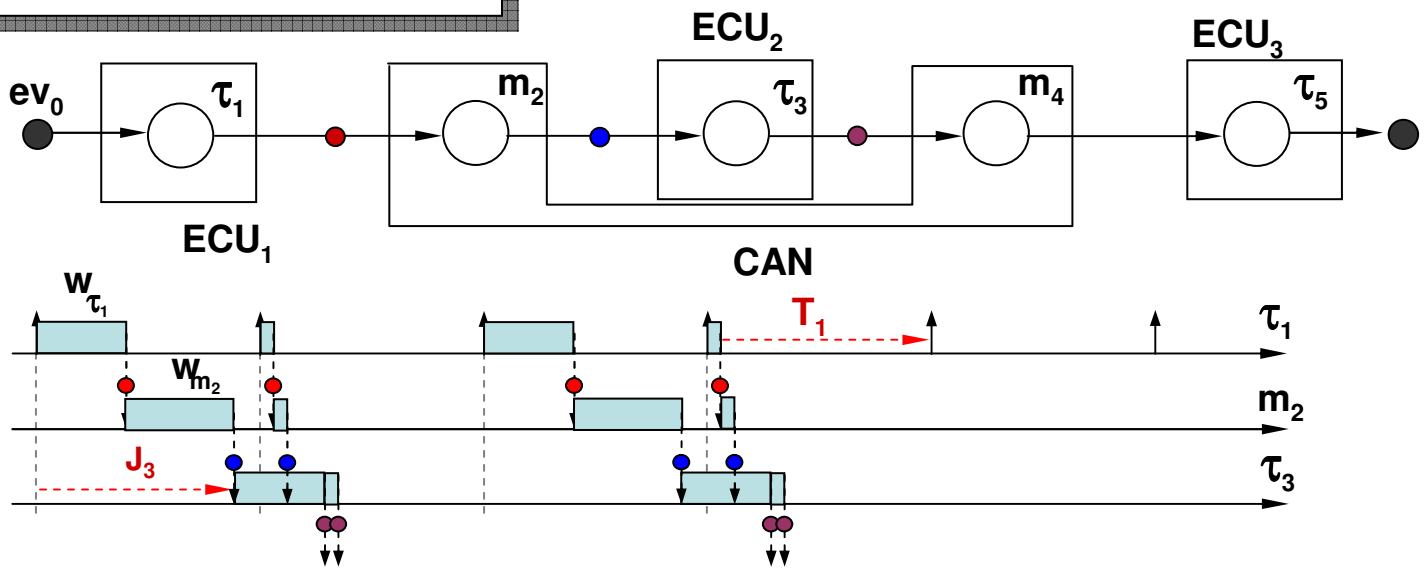
$$r_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{r_i}{T_j} \right\rceil C_j$$

Event-based Activation Model

Lower latency for high priority paths, jitter increases along the path

End-to-end latency analysis

Data-driven precedence constrained activation model



$$l_{(i,j)} = \sum_{k:o_k \in P(i,j)} w_k$$

where (approx.)

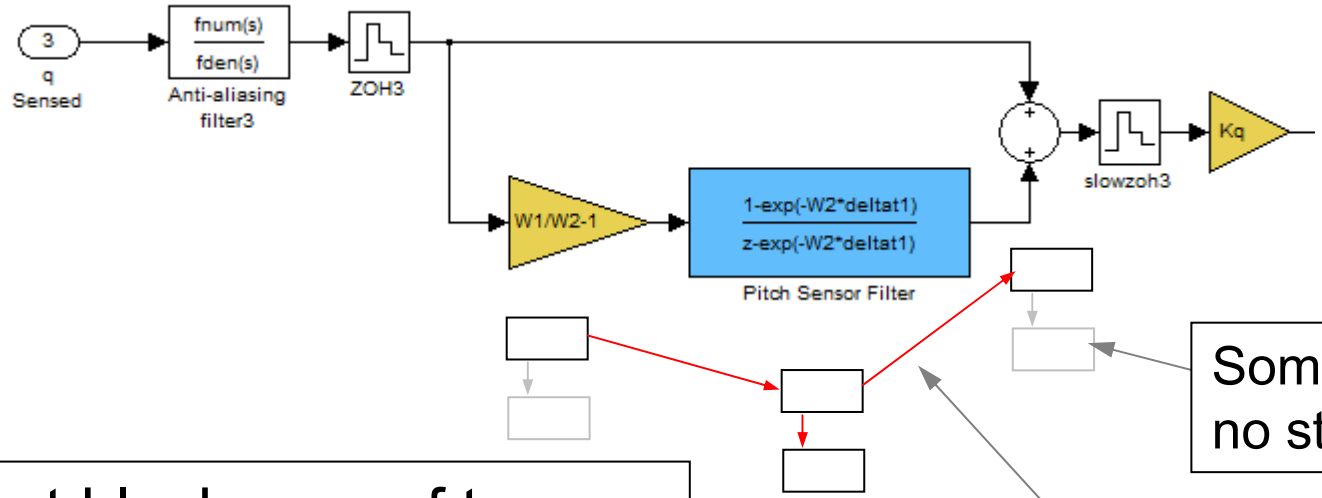
$$w_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i + J_j}{T_j} \right\rceil C_j$$

Task design optimization wrt deadlines

Challenges

- Linearize (or provide an efficient approximate formulation for the response time) – *partly solved*
- What are the optimization metrics?
  - Industry will give us many terms but no formal definition, and some definitions could be quite hard to achieve
    - Extensibility
    - Controls performance
    - Cost
- Design variables are very hard to tackle in a joint optimization process
  - Use stages ?
- Time constraints are hardly the only constraints
  - Semantics preservation constraints
  - Reliability constraints
  - Other resources (memory, power, cost …)

Simulink models (feedthrough)



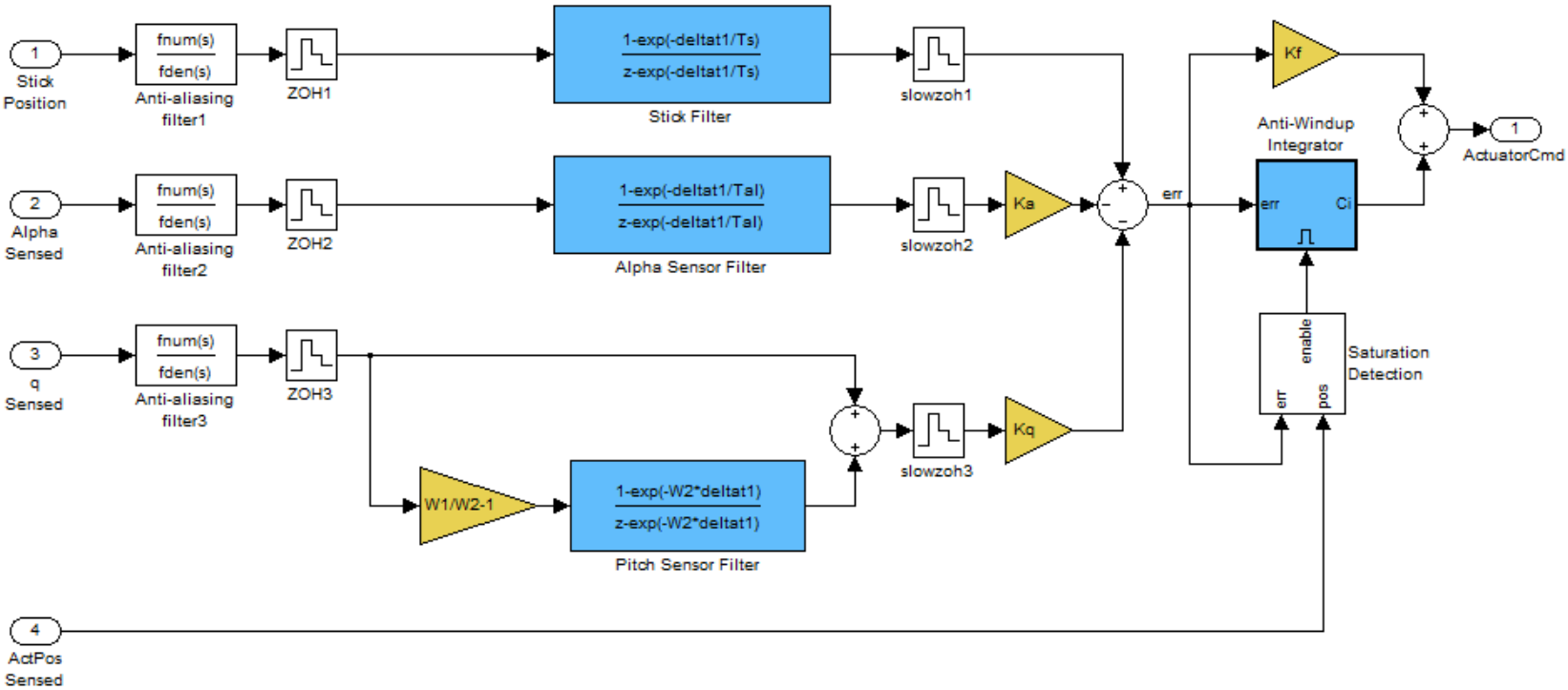
Some blocks have no state

Dependencies among outputs

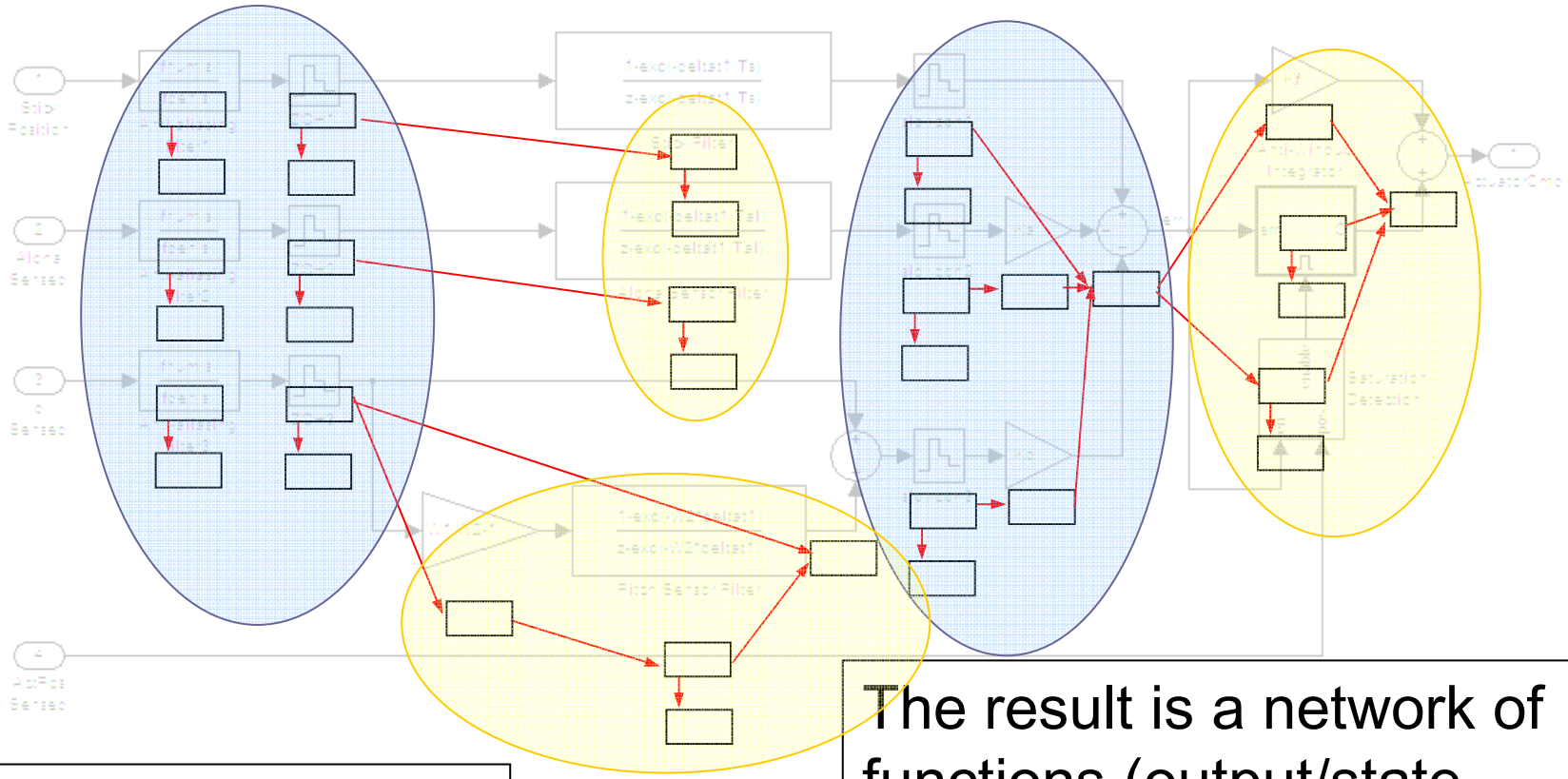
Most blocks are of type feedthrough (output depends on input)

This implies a precedence constraint in the computation of the block output functions

Simulink models



Simulink models

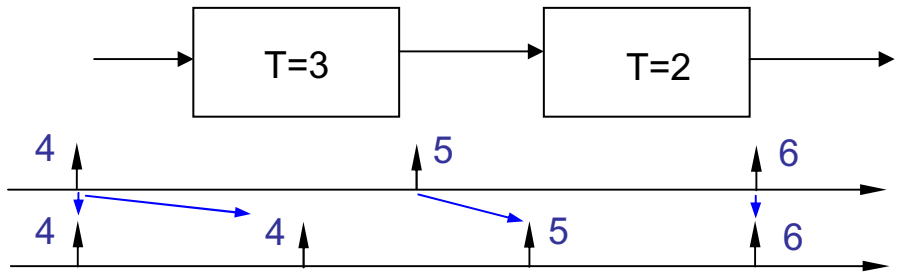


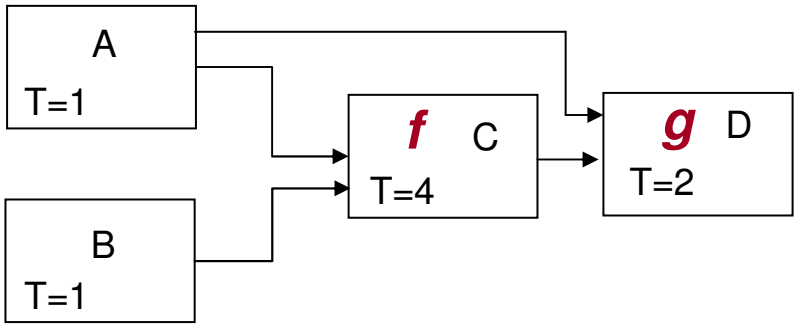
Each blockset is characterized by an execution rate

The result is a network of functions (output/state update) with a set of partial orders

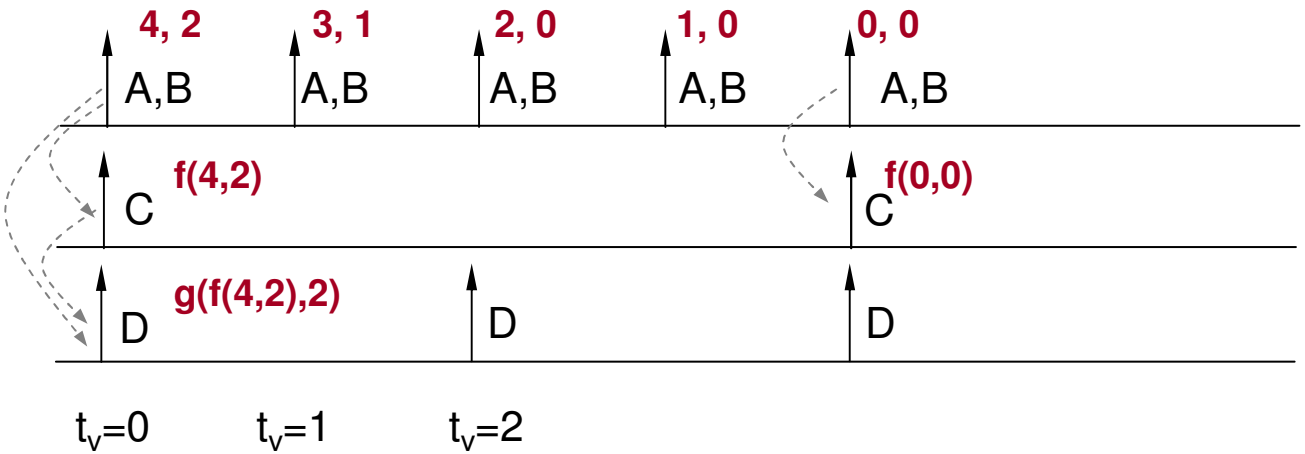
Simulation and Verification: Logical time execution

- Signals are persistent

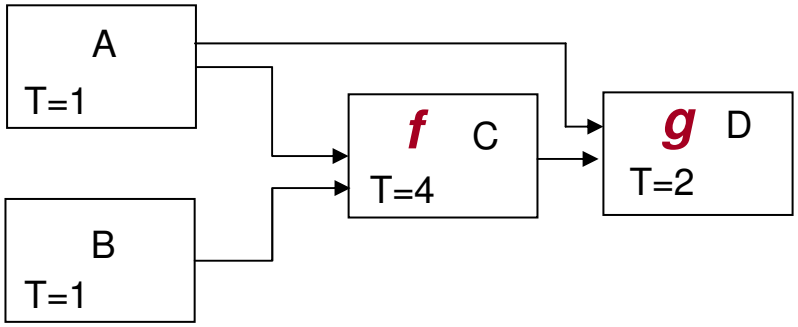




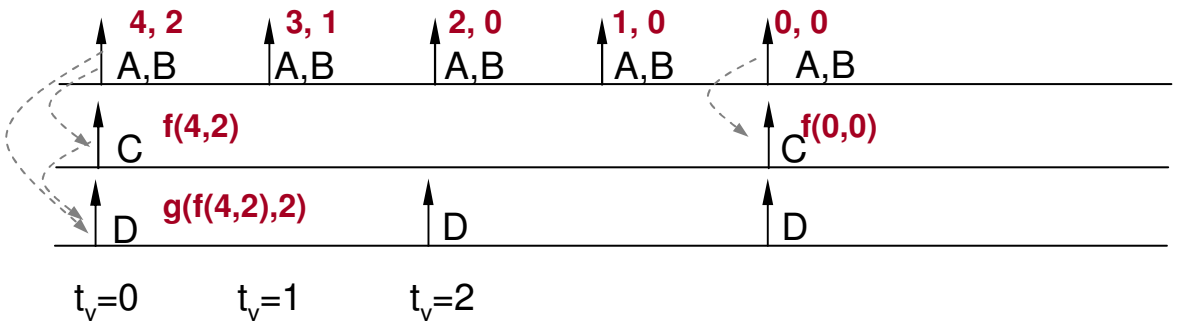
- The block semantics defines a partial order of execution

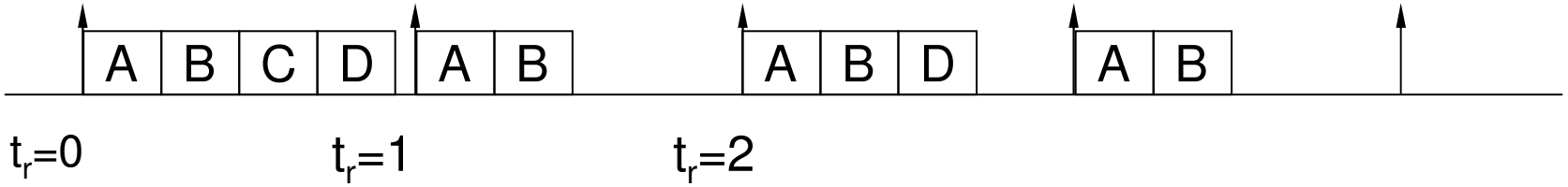


Runtime execution

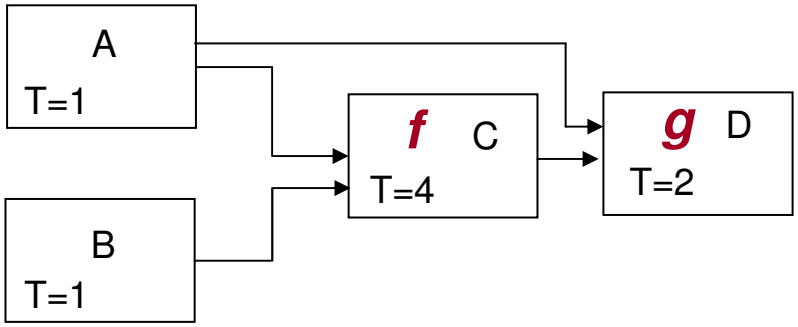


- Single task running at the base period

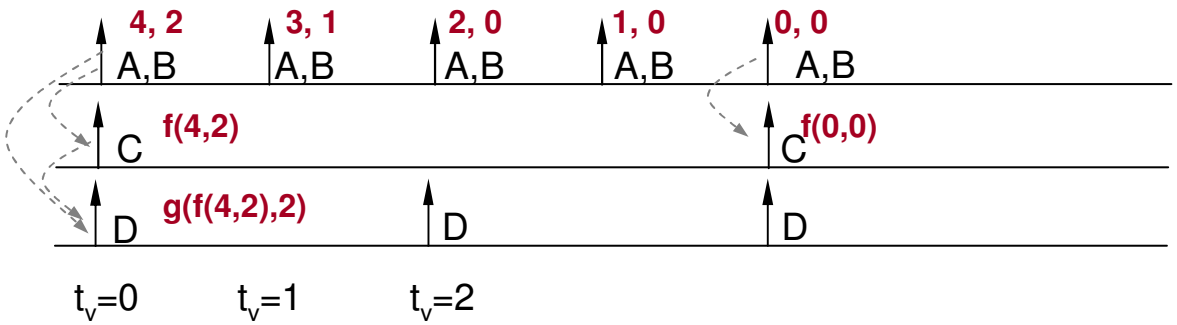


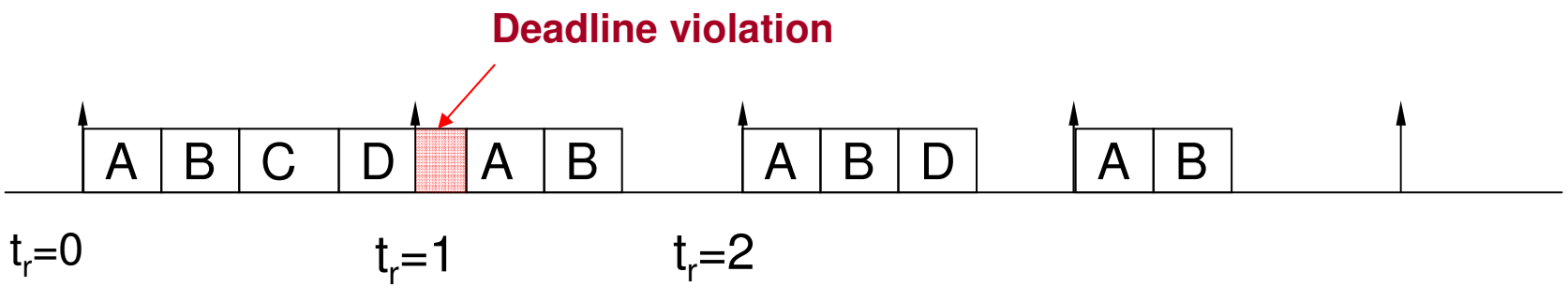


Runtime execution

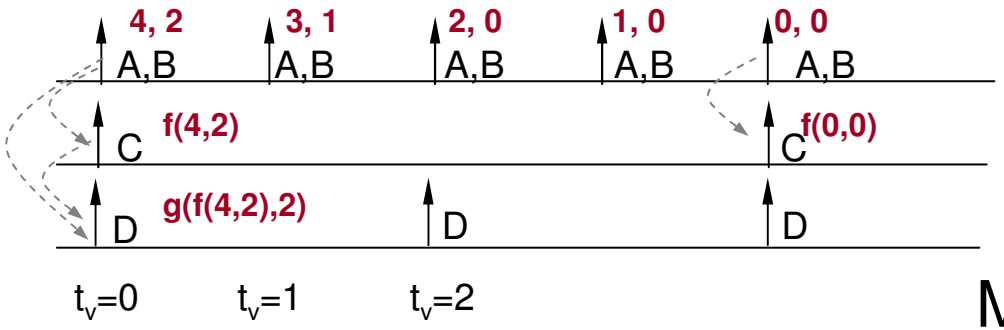


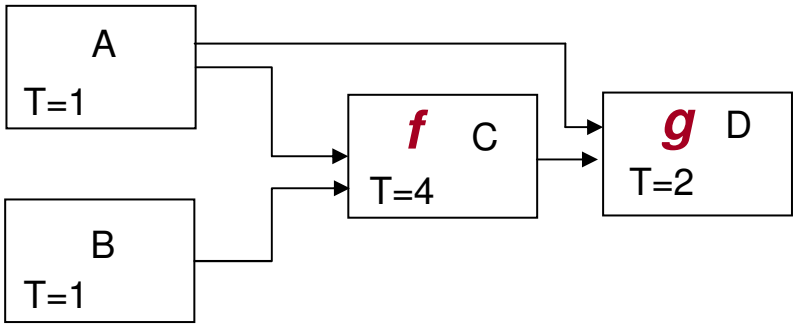
- Single task running at the base period



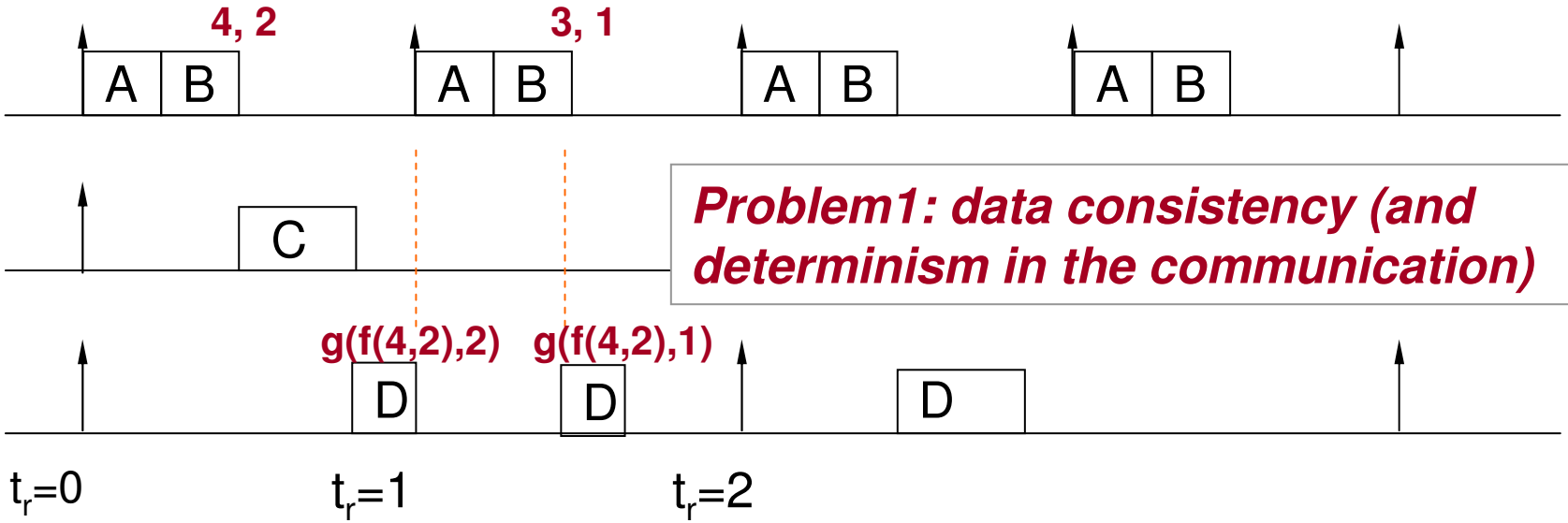


Runtime execution



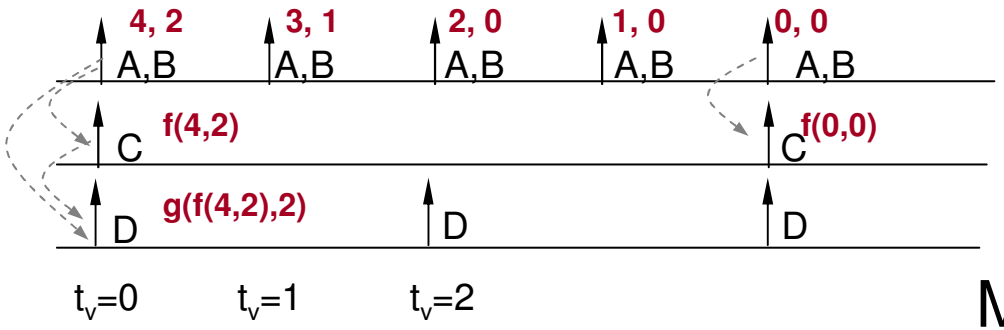


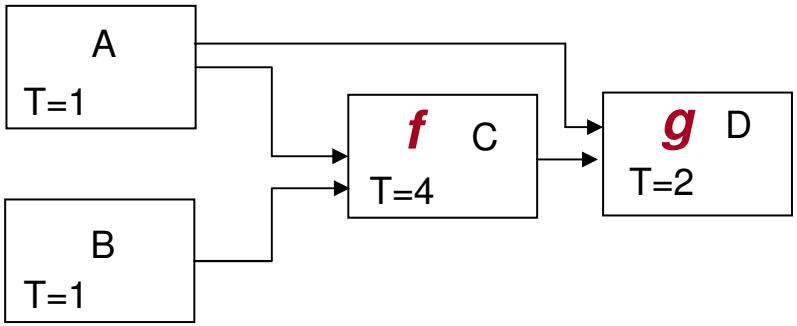
Multiple tasks, each executing all blocks at each period



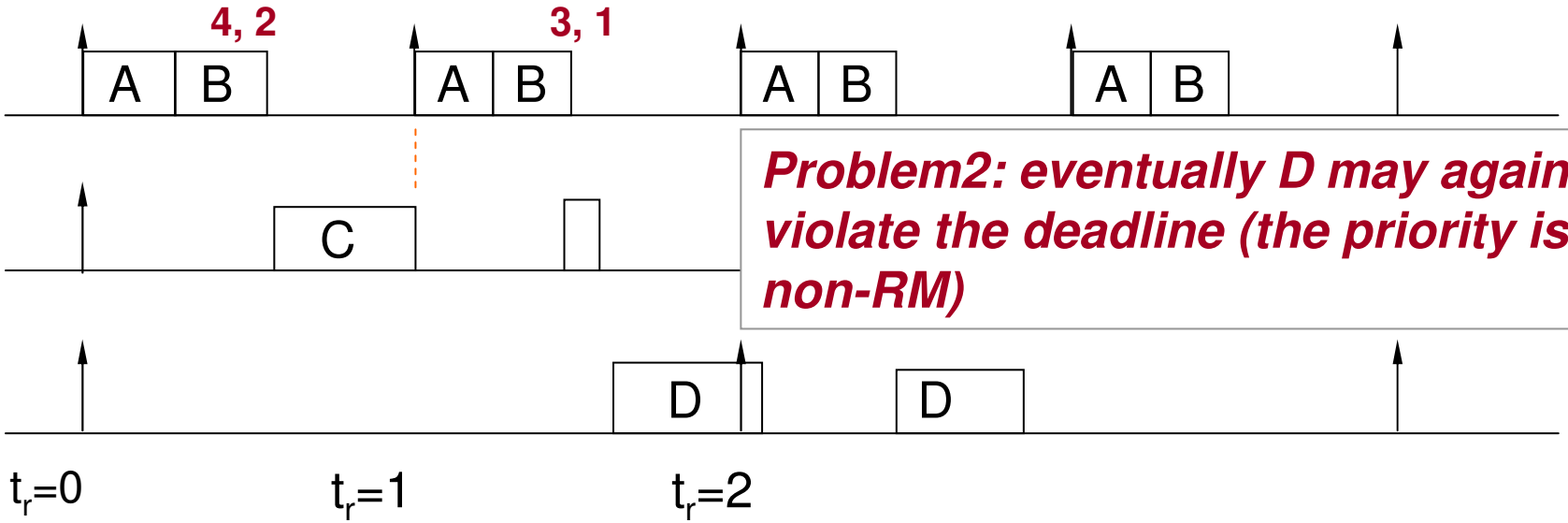
Problem1: data consistency (and determinism in the communication)

Runtime execution



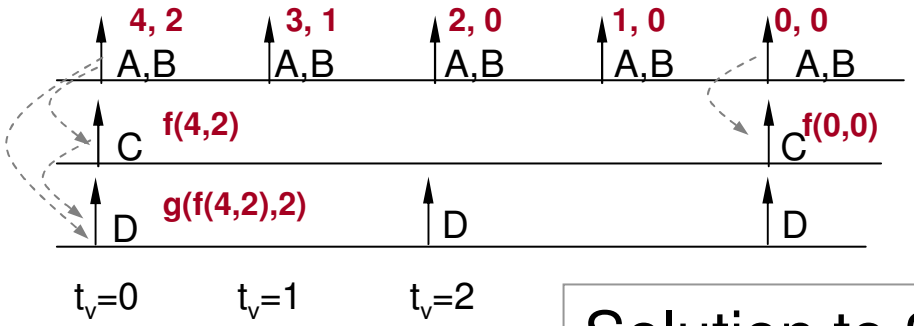


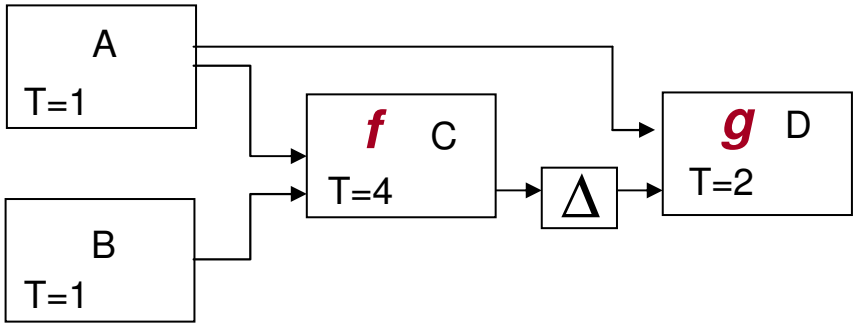
Multiple tasks, each executing all blocks at each period



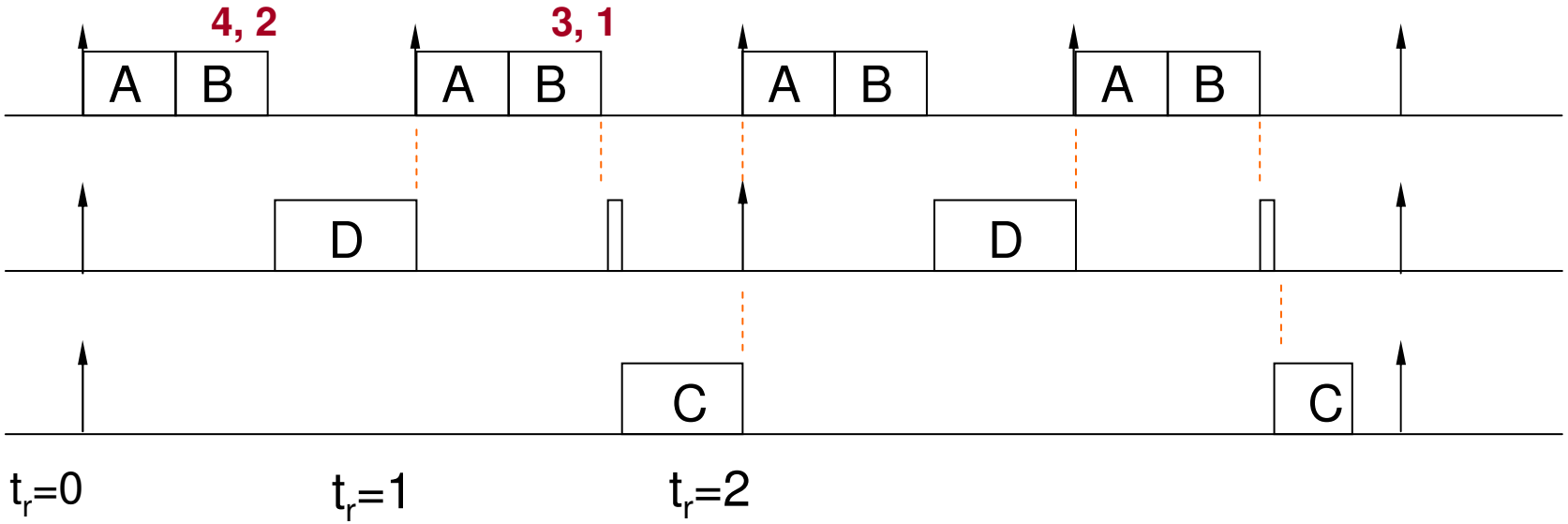
Problem2: eventually D may again violate the deadline (the priority is non-RM)

Runtime execution





Solution to 2: add a delay, now the execution order of C and D can be reversed (RM)



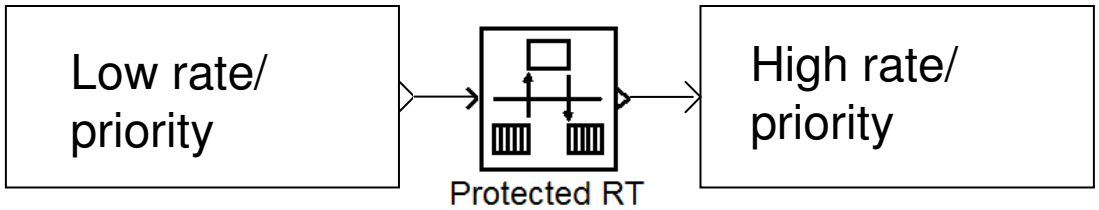
RT blocks: Low rate/priority to high rate/priority

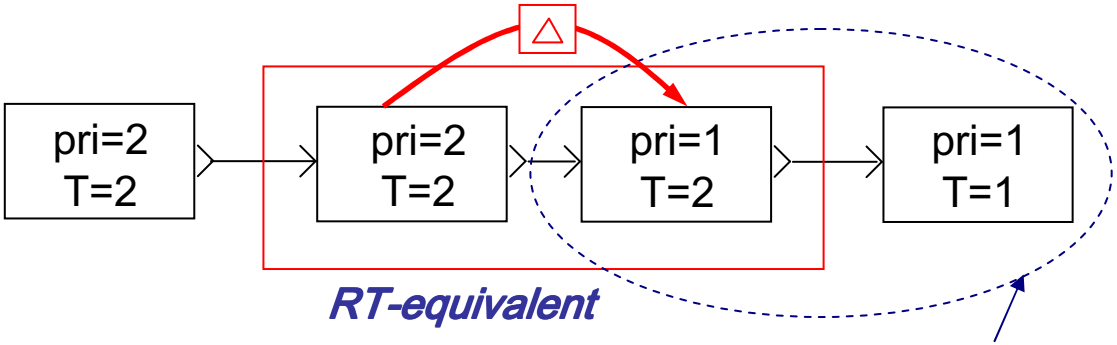
COST

space: 2 additional set of variables for each link

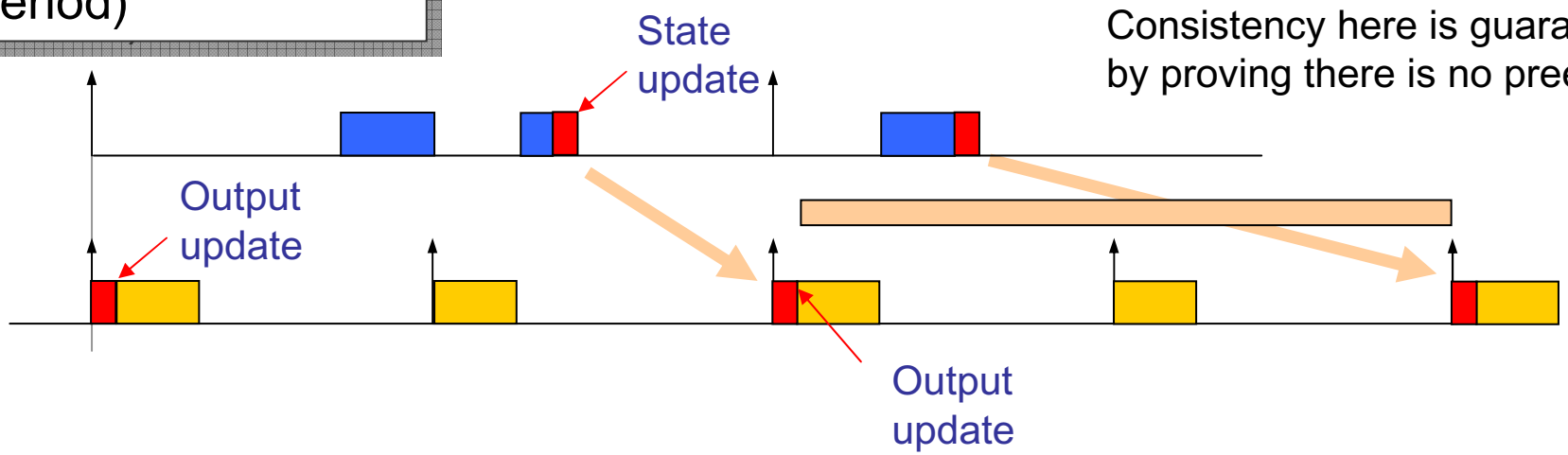
time: overhead of RT implement.

performance: 1-unit delay (low rate period)





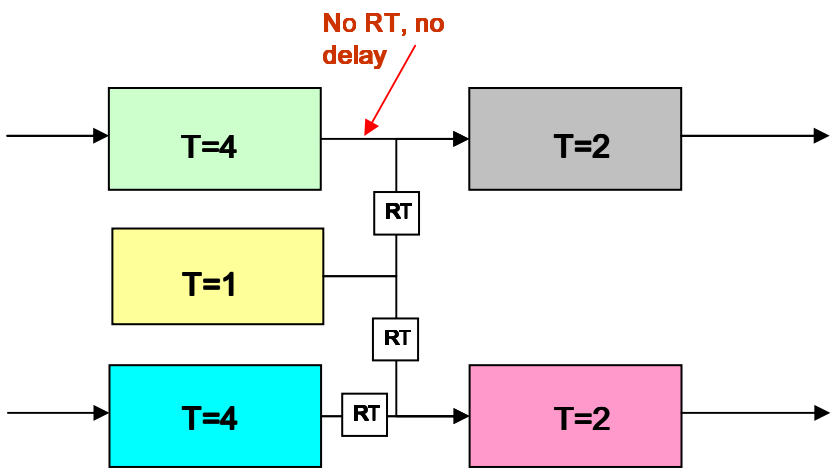
Consistency here is guaranteed by proving there is no preemption

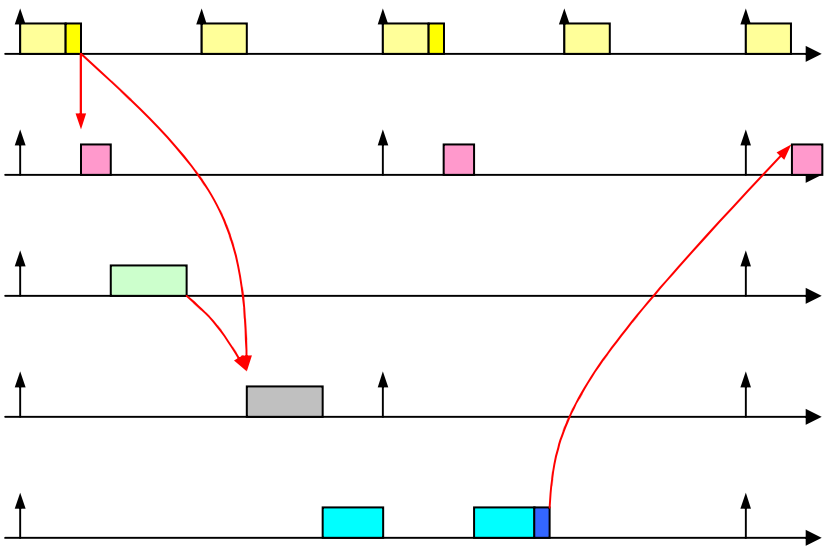


Design/Scheduling trade-offs

However ...

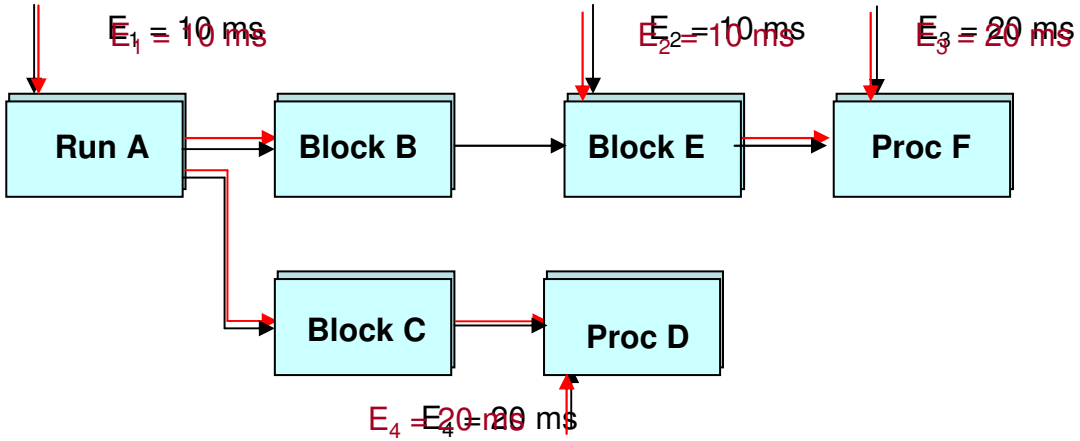
- Delay units have a cost (in terms of memory and performance)
- it is possible to selectively preserve the precedence order (giving higher priority to the slow block) at the expense of schedulability
  - Two tasks at the same rate, one high priority, the other low priority





Optimal Synthesis of Task and comm. Model

Q: what is the best runnable-to-task mapping?



Run A	Run E	Run B	Run F(*)	Task 1 - 10 ms	Proc A	Proc E	Proc B
Proc W				Task 2 - 40 ms	Proc W		
Proc C				Task 3 - 10 ms	Proc C		
Proc D				Task 4 - 20 ms	Proc D	Proc F	

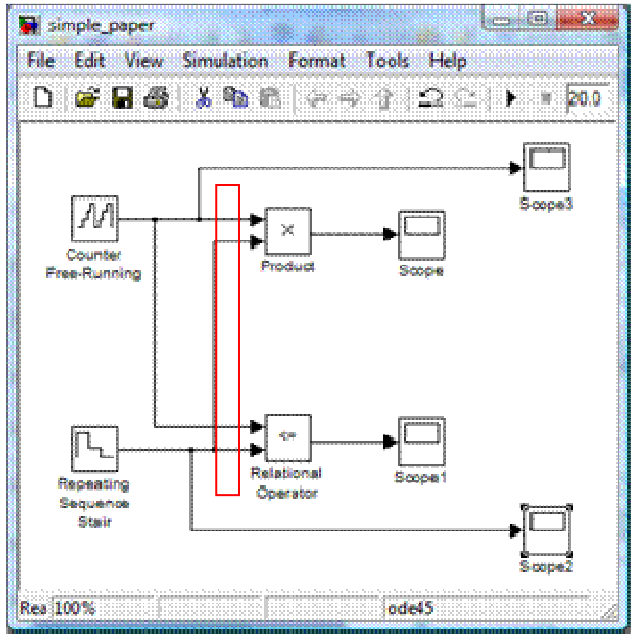
Pro: No need to protect communication between E and F.
Cons: Less scheduling flexibility, limited priority inversion

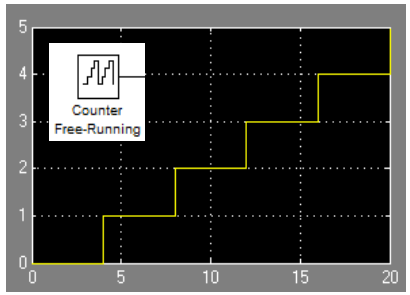
Distributed implementation of models

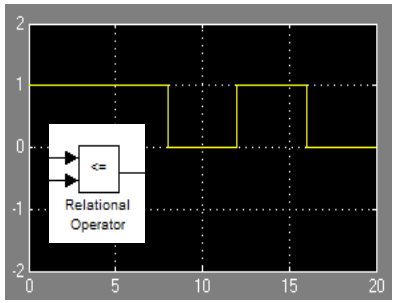
A very simple model with oversampling ….

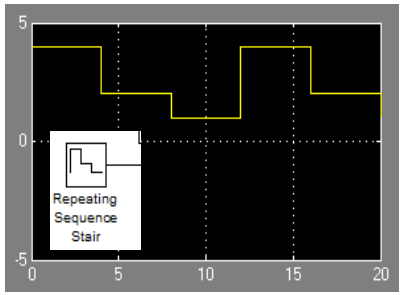
Imagine the data streams between source blocks and the multiplier/comparator are exchanged over a network.

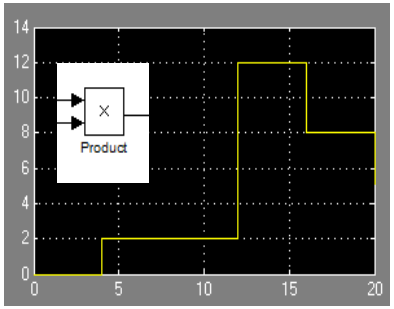
These are the results seen by the control engineer at design time





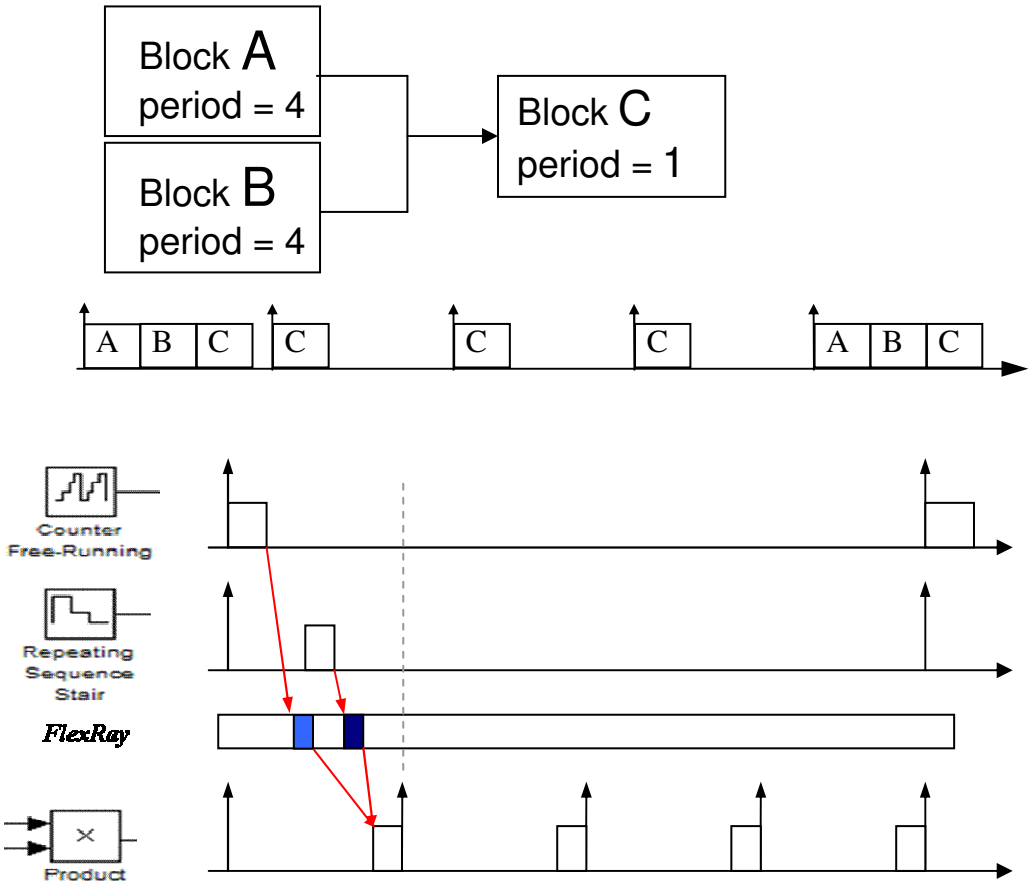


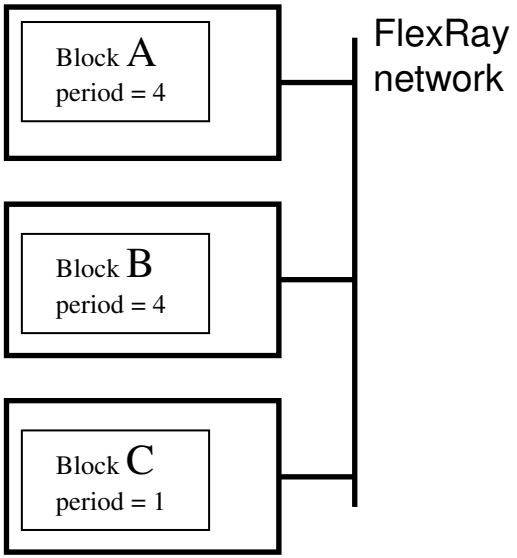




Delays from network

An example of the trade-offs between additional functional delays and scheduling feasibility

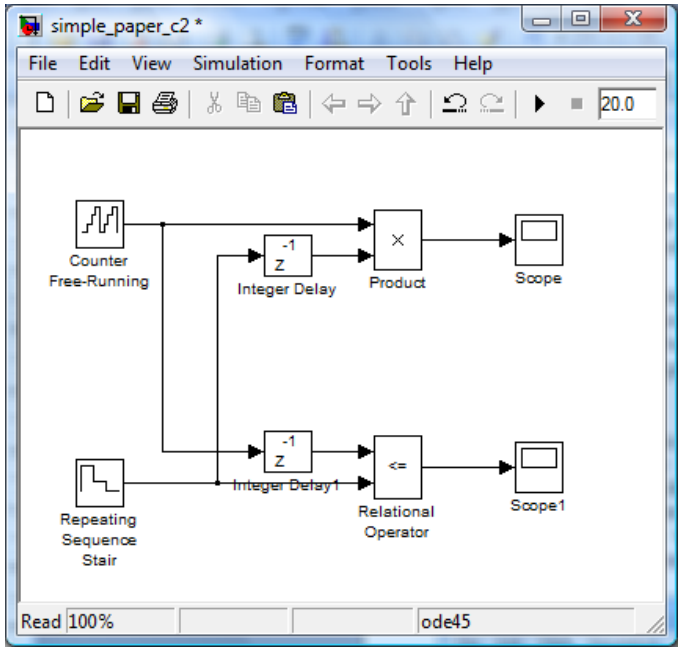


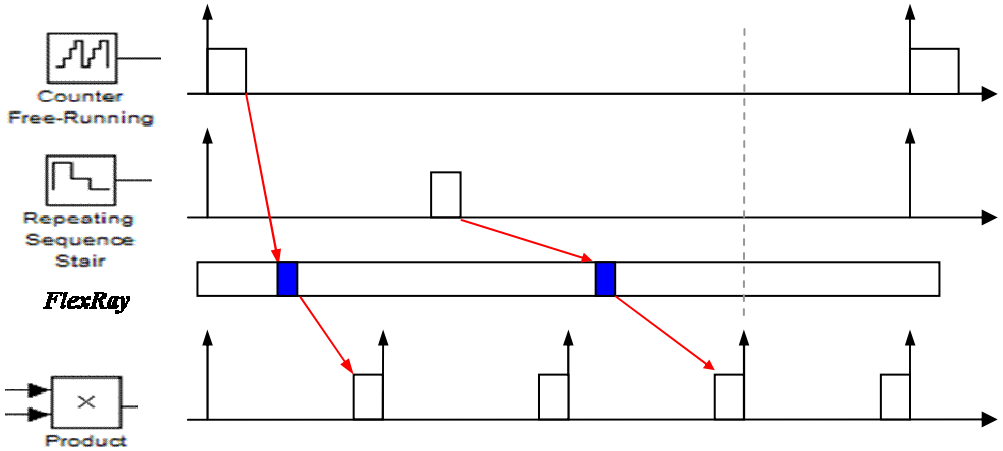


A semantics preserving implementation may be difficult to schedule because of a tight communication pattern

Delays from network

Designers may be tempted to ease the scheduling problem by choosing the instance of the receiving task/block





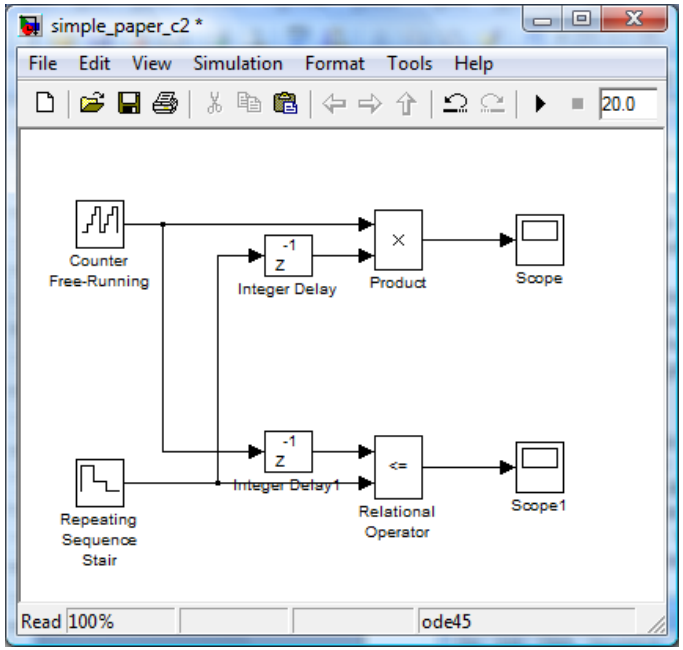
Delays from network

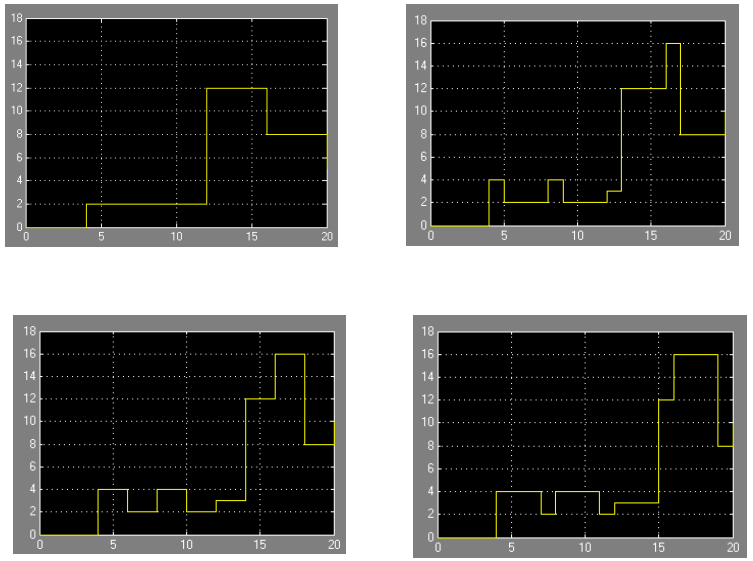
Unfortunately, by doing so, the behavior is different from the one simulated with 0-delay

Are the designers/developers fully aware of these issues ?

How can we help them ?

(Task and message design and scheduling are in the background)





Conclusions

- Schedulability theory and worst-case timing analysis …
  - From the run-time domain to the design domain (already happening)
  - From the analysis domain to optimization (synthesis) domain
  - Complemented by sensitivity analysis and uncertainty evaluation
  - Need efficient ways to lineariza response time computation
  - And possibly to partition end-to-end deadlines
  - Need theory to formally evaluate the performance cost of adding delay units
  - Or, in general, to relate control performance to control function periods and to end-to-end latencies
  - Need to understand how to tackle large problems with multiple design variables

Q&A

Thank you!