Mapping Algorithms onto MPSoC

- Some Results from ARTISTDesign -

Lothar Thiele

Contents

- **Overview about DOL**
- Heterogeneous Modeling
- Interference Analysis for Multicore Systems
- Experimental Results

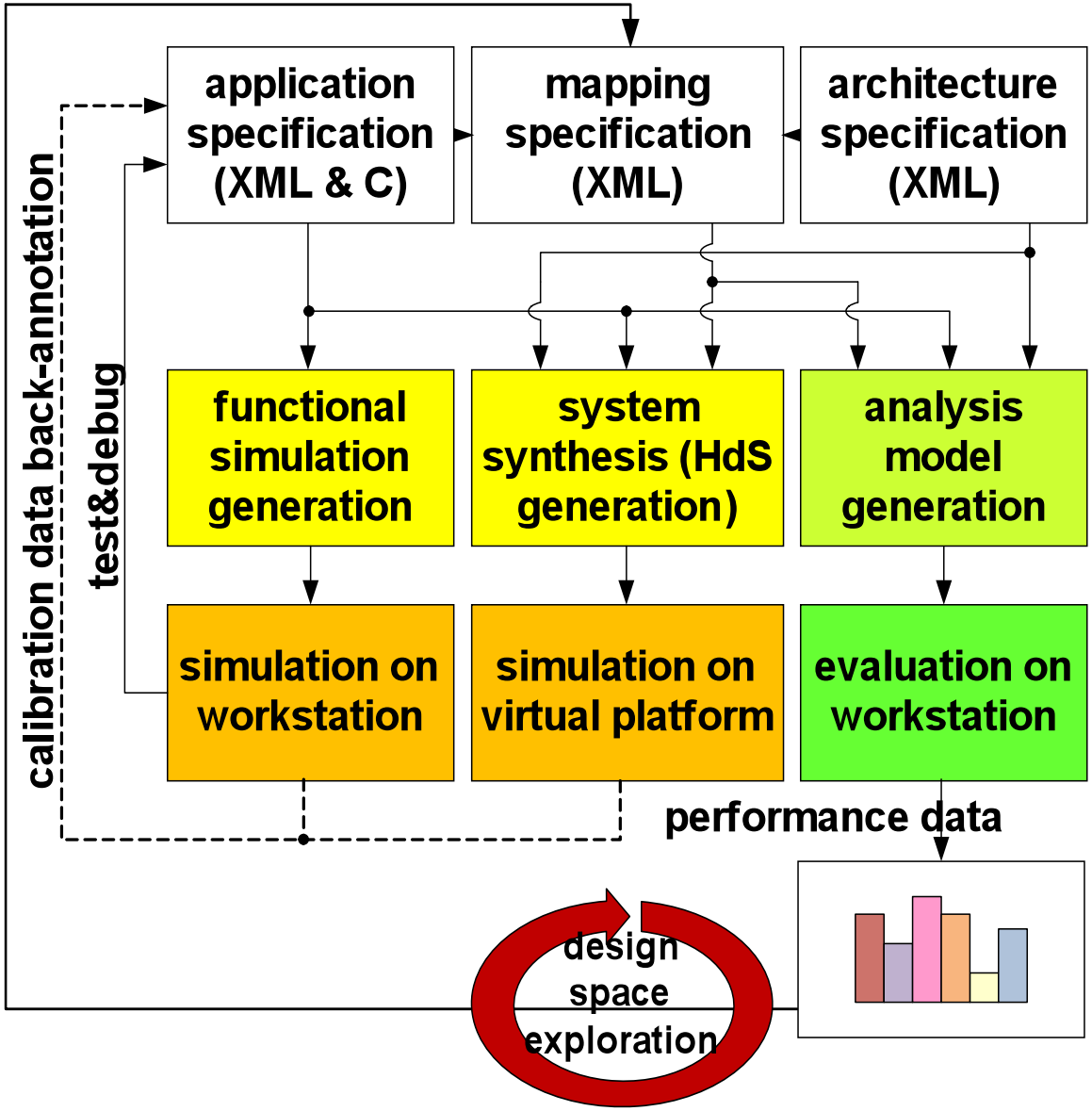
DOL Programming Environment

Goal

- Automatically generate an optimal mapping of a parallel application onto MPSoC

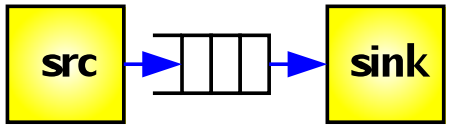
Challenges

- Scalable specification
- Automated synthesis
- Design space exploration
- System-level performance analysis



DOL Programming Model

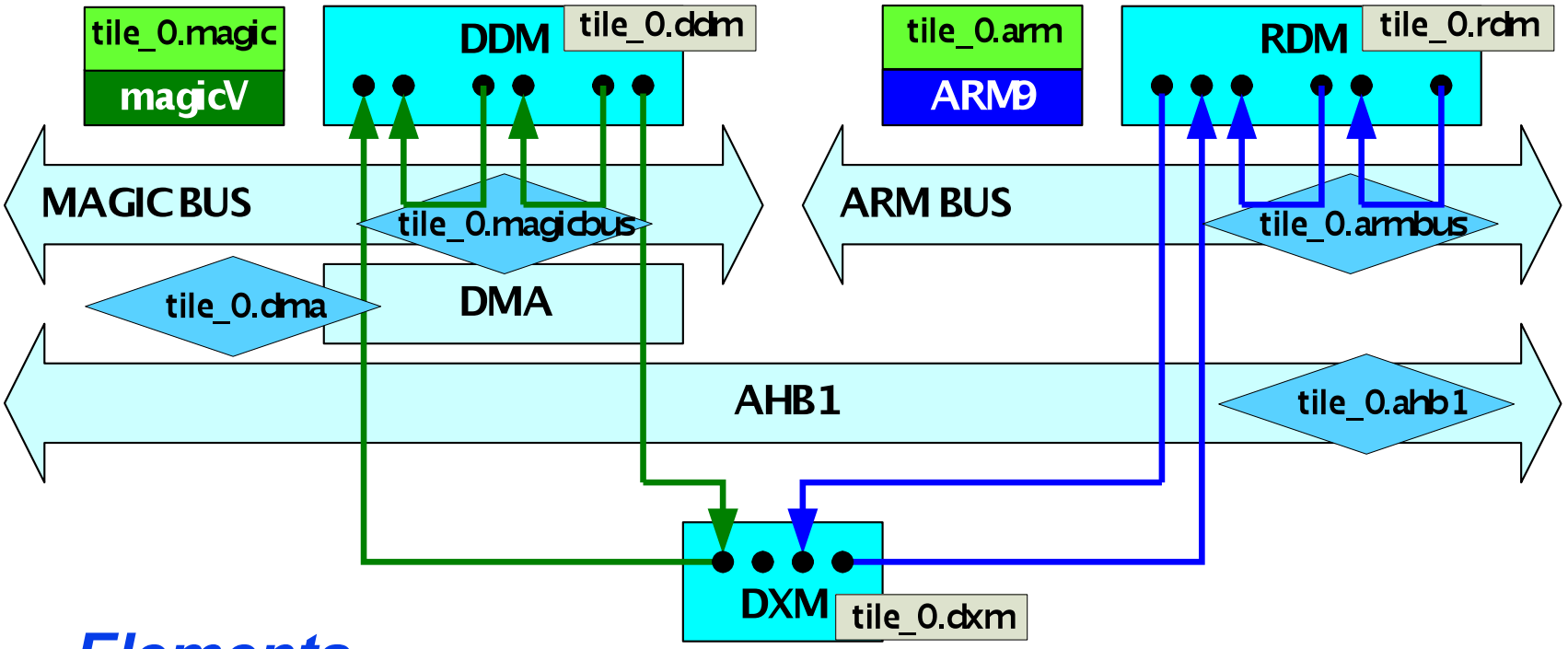
```
01: <process name="src">
02:   <port type="output" name="out"/>
03:   <source type="c" location="src.c"/>
04: </process>
05:
06: <process name="sink">
07:   <port type="input" name="in"/>
08:   <source type="c" location="sink.c"/>
09: </process>
10:
11: <connection name="con_src_fifo">
12:   <origin name="src"><port name="out"/></origin>
13:   <target name="sink"><port name="in"/></target>
14: </connection>
```




```
01: procedure INIT(Process p)
02:   //initialize process state
03: end procedure
04:
05: procedure FIRE(Process p)
06:   DOL_READ(input, size, buf)
07:   //processing
08:   DOL_WRITE(output, size, buf)
09: end procedure
```

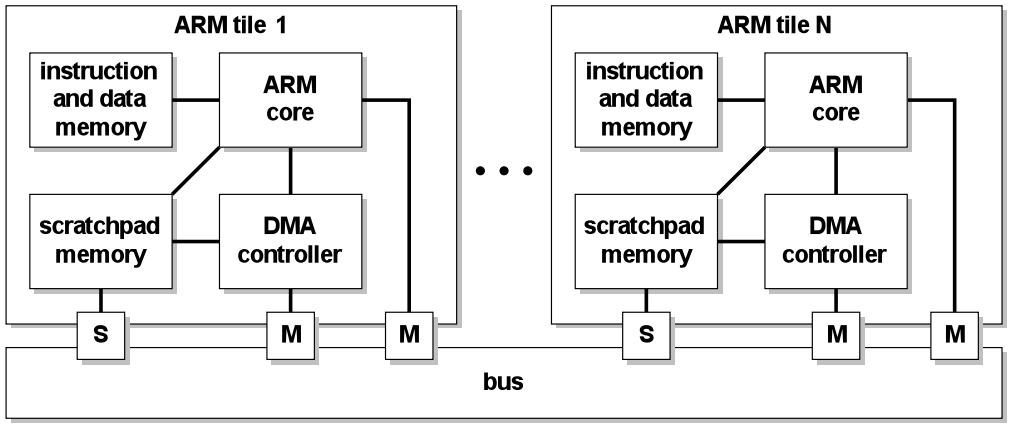
- Model of computation: Kahn process network
  - Coordination language: XML with performance annotations
  - Functionality of processes: C/C++ with specific programming DOL API
- Scalability: “iterators” for large, multi-tile descriptions

DOL Abstract Platform Modeling

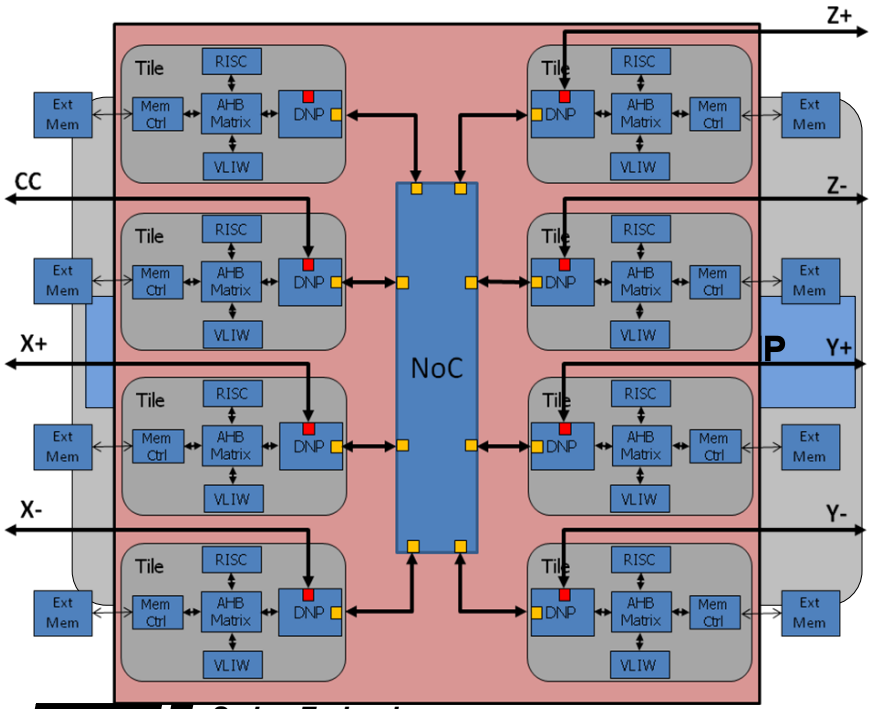


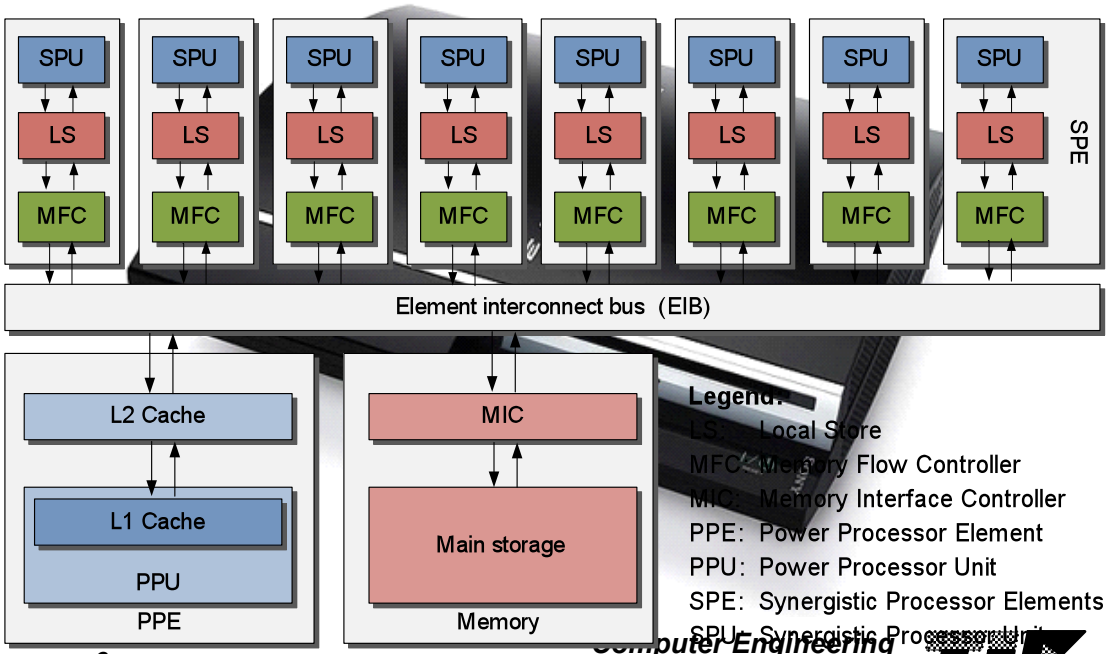
- ***Elements***
  - Structure: processors, peripherals, memories, buses, etc.
  - Interconnects: explicit read and write communication paths
  - Performance data: e.g. latency and bandwidth of HW communication channels
- ***Specification***: XML, including the “iterators” capability

DOL Software Synthesis

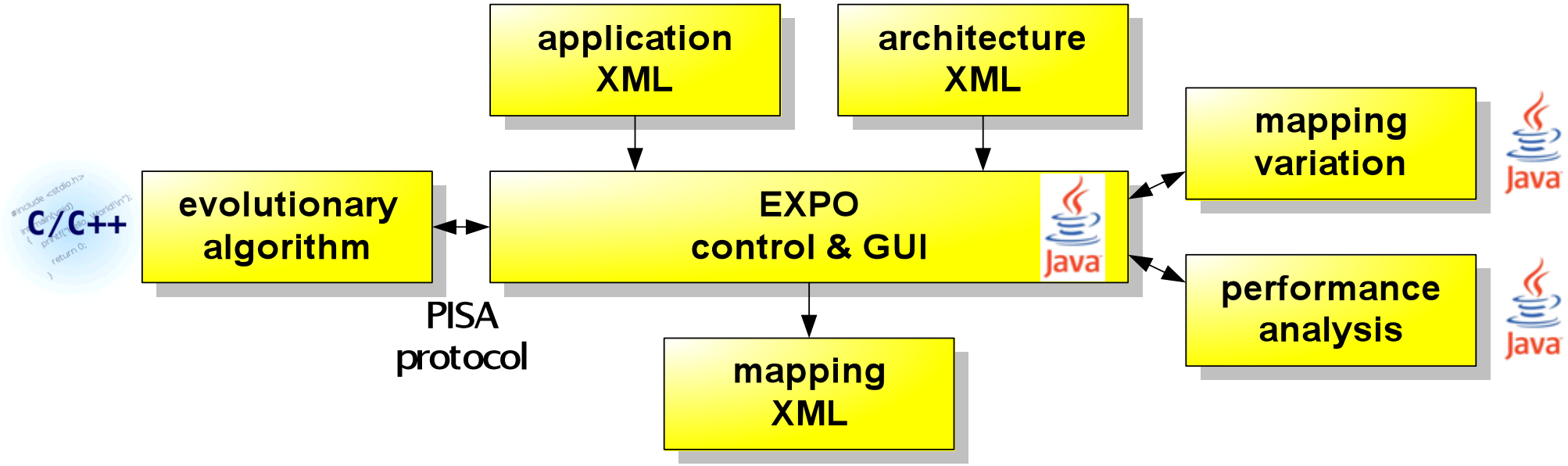


- **MPARM:** identical ARM7 tiles connected by AMBA bus
- **Atmel Diopsis 940:** tile:ARM9+DSP connected by an AMBA bus; several tiles connected via NoC
- **Cell BE:** PowerPC and 8 DSP-like SPEs connected via ring bus





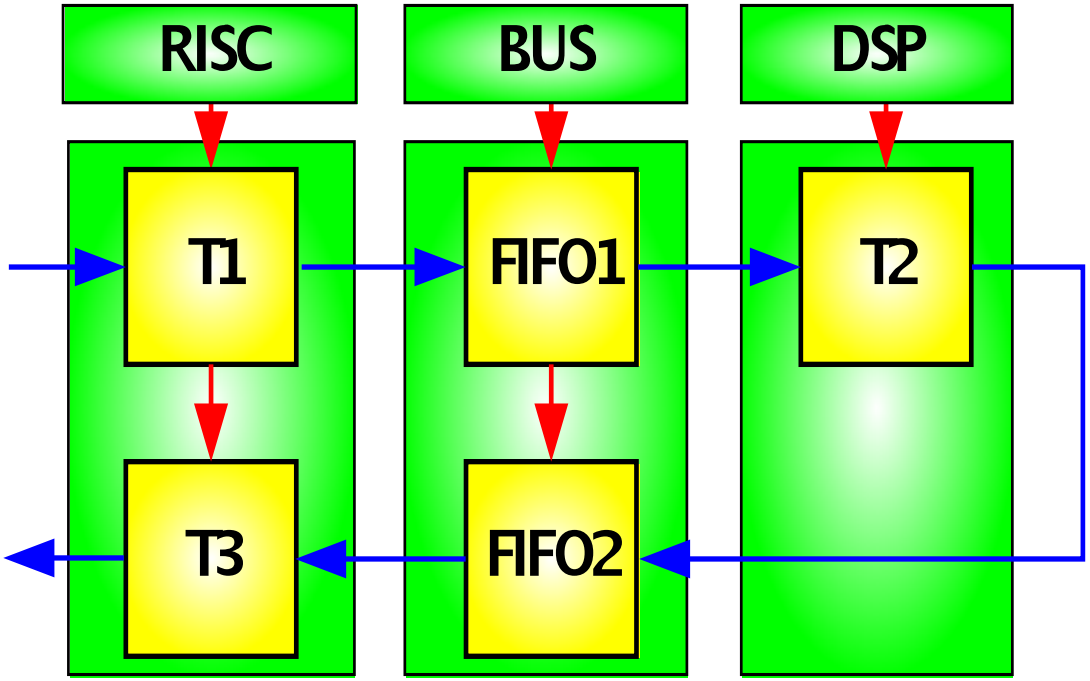
Mapping Optimization Framework



Modular framework: *optimization* and *performance analysis* plug-ins

- **Evolutionary algorithm:** SPEA2 (Strength Pareto Evolutionary Algorithm)
- **Performance analysis:** MPA (Modular Performance Analysis) analytic model http://www.mpa.ethz.ch
- **Control & GUI:** EXPO https://www.tik.ee.ethz.ch/expo
- **Interface:** PISA https://www.tik.ee.ethz.ch/pisa

Performance Analysis using MPA

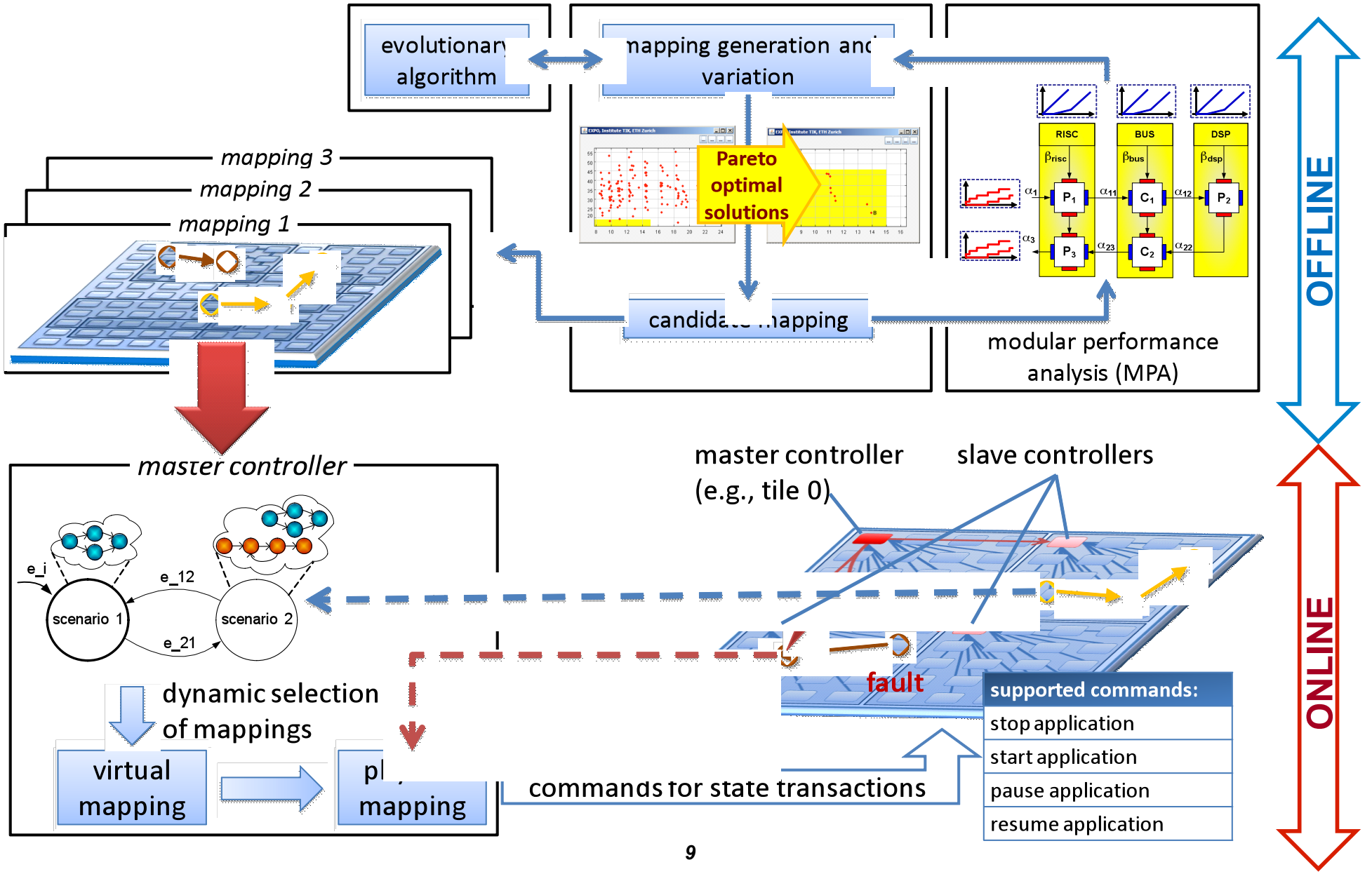


Design: **abstract event streams traversing a network of abstract processing units**

Output: **worst-case bounds on system properties**

- Modular Performance Analysis http://www.mpa.ethz.ch
- (Large) MPSoC extensions: complex activation schemes, timing correlations, blocking semantics, cyclic dependencies

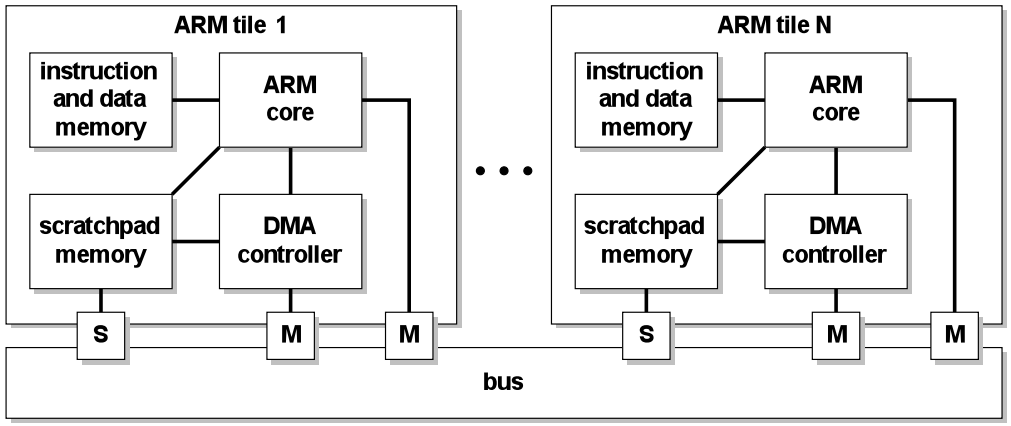
Extension: Distributed Applications



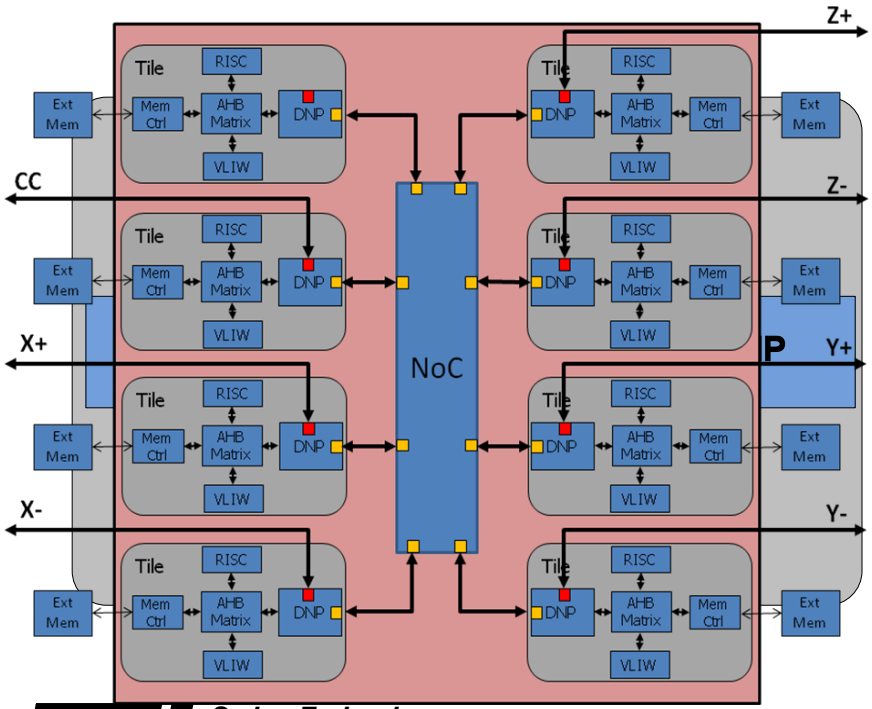
Contents

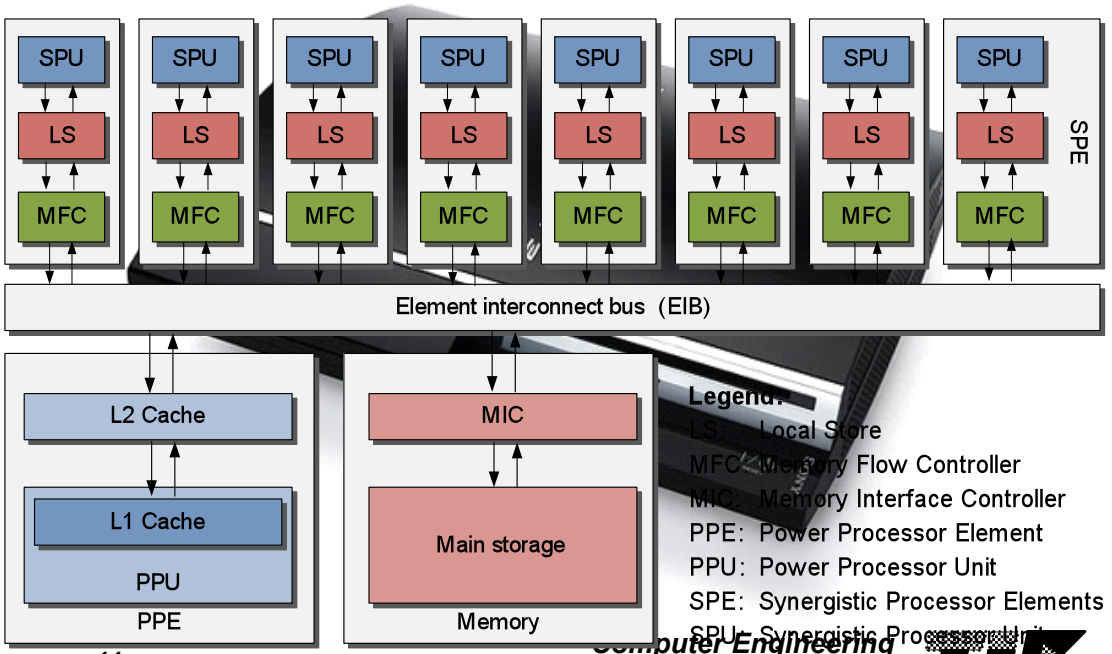
- Overview about DOL
- **Heterogeneous Modeling**
  - *Embedding into rigorous design flow (VERIMAG)*
  - *Integration of Symta/S (TU Braunschweig)*
  - *Heterogeneous Analysis (UPPAAL)*
  - *Hierarchical Modeling (TU Braunschweig)*
  - Comparison of Methods (U Cantabria, TU BS)
- Interference Analysis for Multicore Systems
- Experimental Results

DOL Software Synthesis



- **MPARM:** identical ARM7 tiles connected by AMBA bus
- **Atmel Diopsis 940:** tile:ARM9+DSP connected by an AMBA bus; several tiles connected via NoC
- **Cell BE:** PowerPC and 8 DSP-like SPEs connected via ring bus



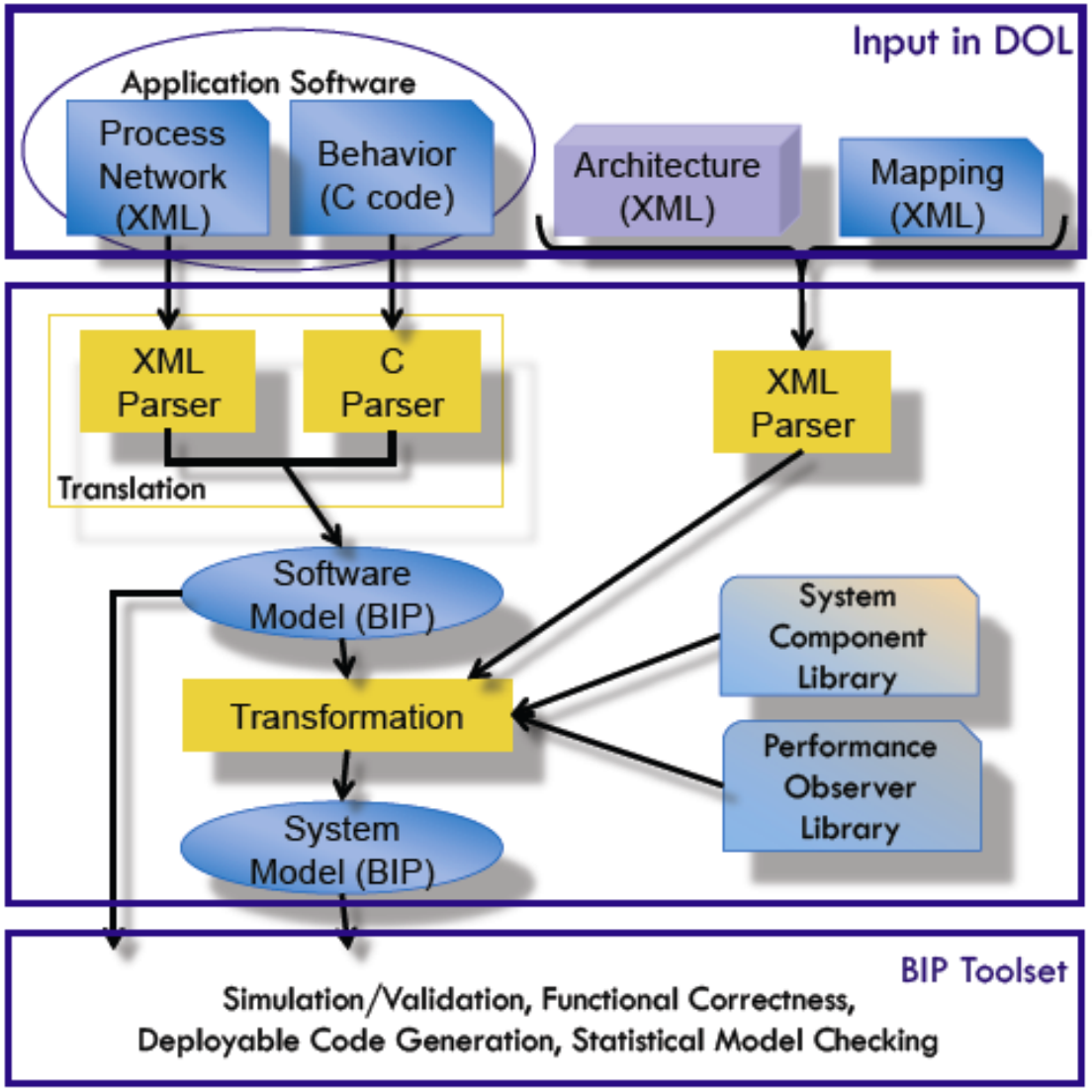


Rigorous Design Flow (VERIMAG)

The ***frontend*** transforms networks of processes written in DOL into BIP models

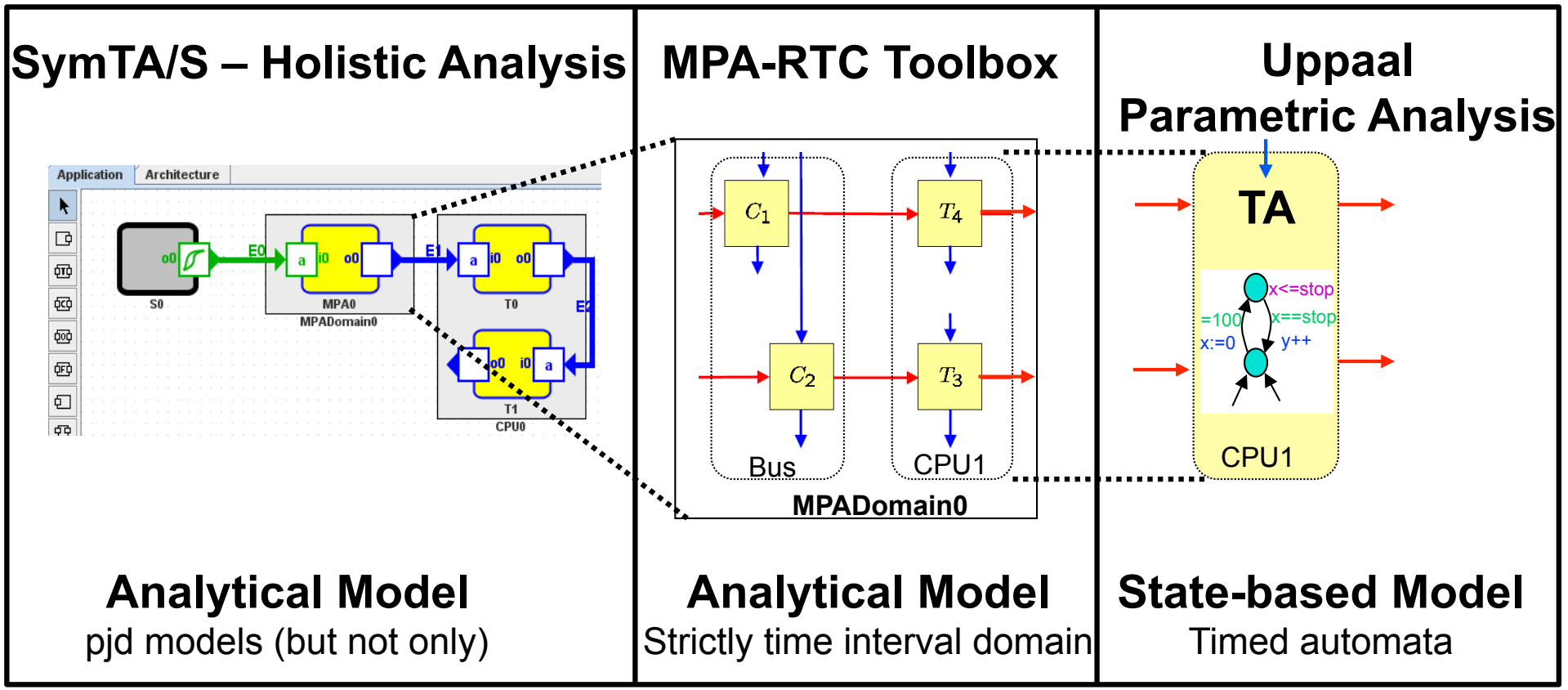
The ***backend*** applies a series of source-to-source correct-by-construction transformations over the BIP software model.

The generated system model includes can be validated and analyzed



Global Picture

Mixing time-interval-based and time-based abstractions

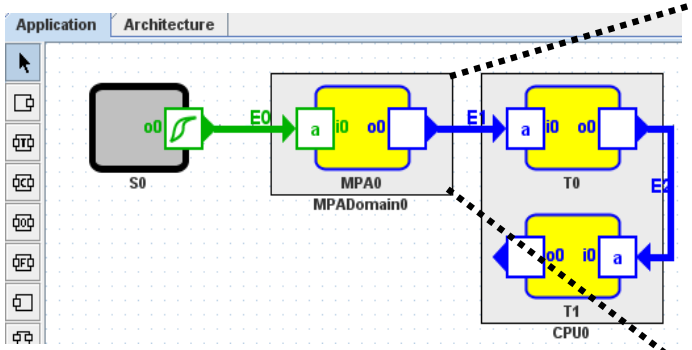
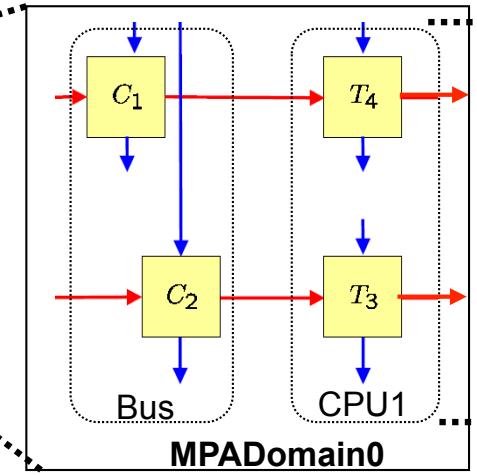
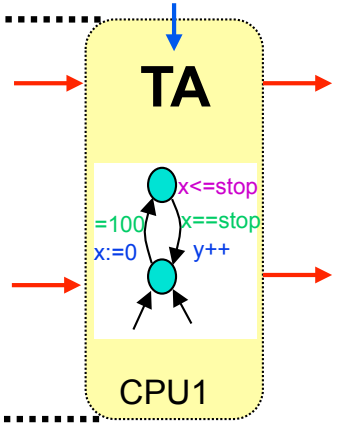


Global Picture

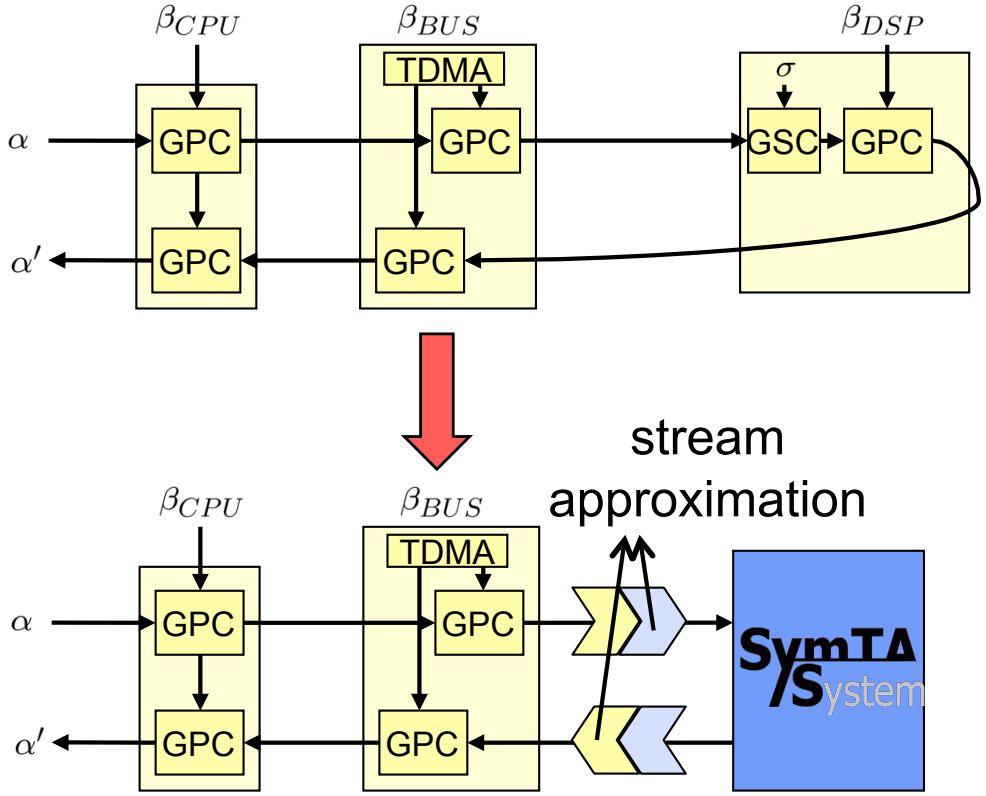
TU BS ETHZ

UPPSALA EPFL Trento

Mixing time-interval-based and time-based abstractions

SymTA/S – Holistic Analysis	MPA-RTC Toolbox	Uppaal Parametric Analysis
		
Analytical Model pjd models (but not only)	**Analytical Model** Strictly time interval domain	**State-based Model** Timed automata

Integration by Stream Approximation



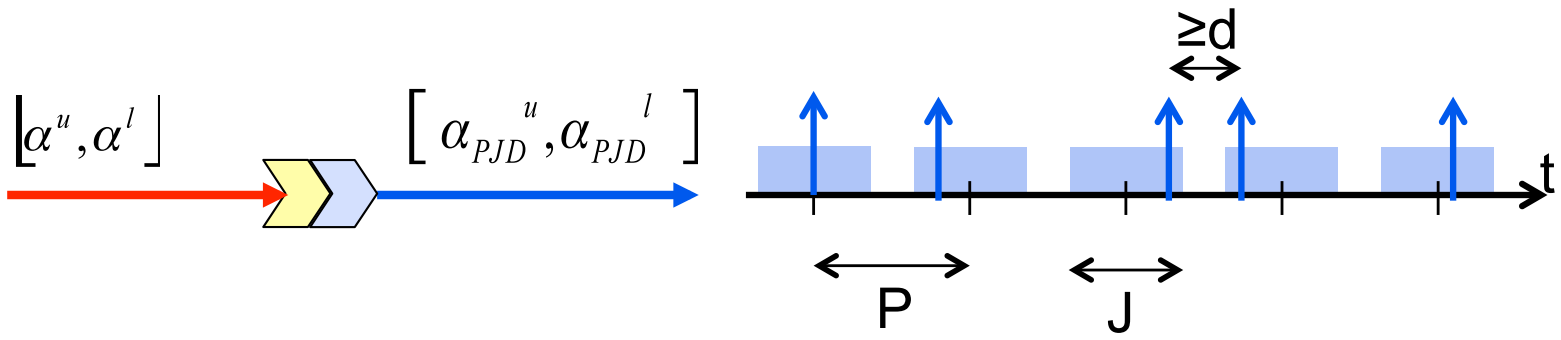
Replace RTC component by

- classical single-resource real-time analysis
- holistic scheduling analysis (MAST, ***Gonzales***, Tindell, Yen, ***Eles***, …)
- Symta/S compositional analysis (***Ernst***)

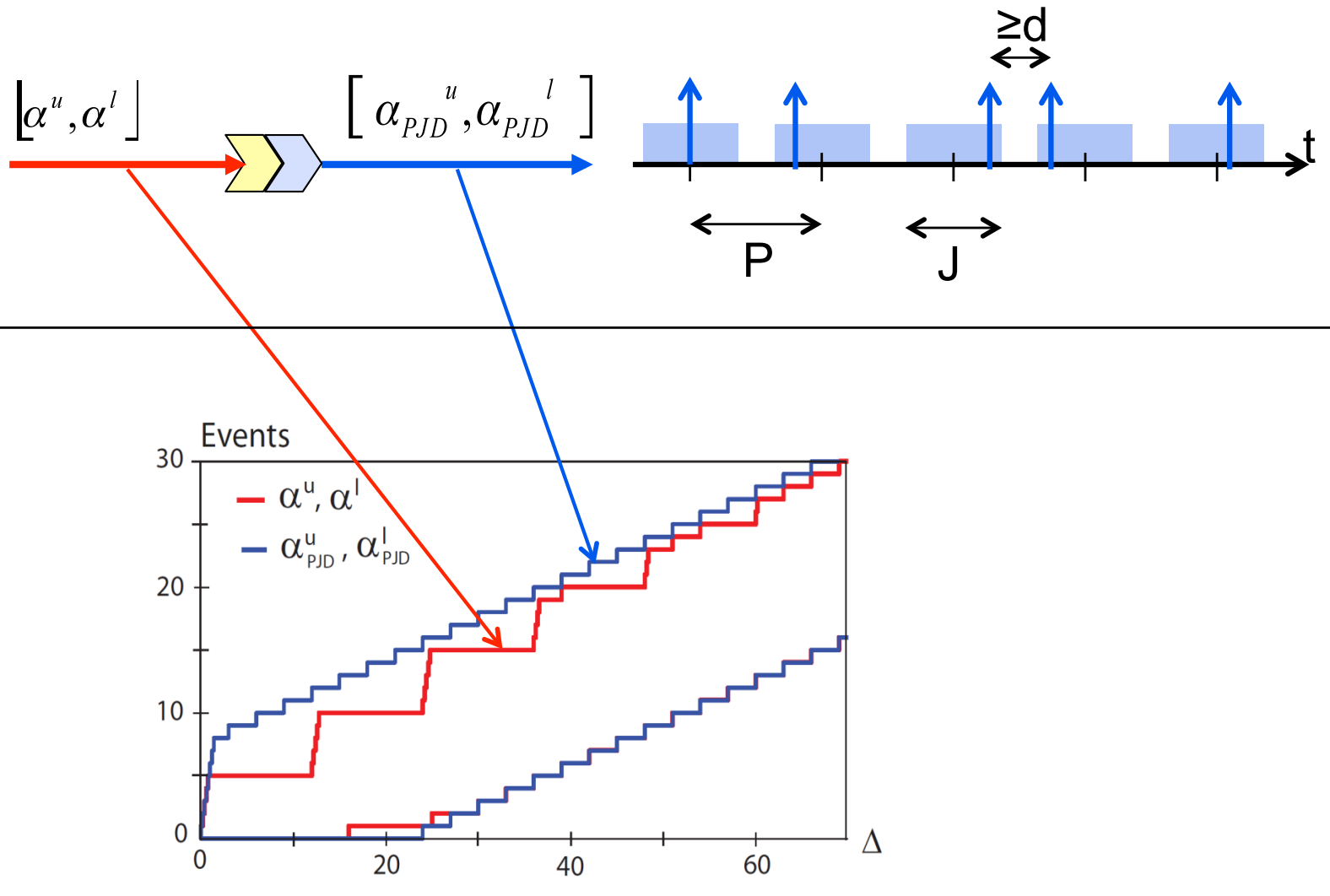
Benefits

- incorporate advanced anaysis methods; adapt analysis method to component characteristics
- reduce analysis complexity

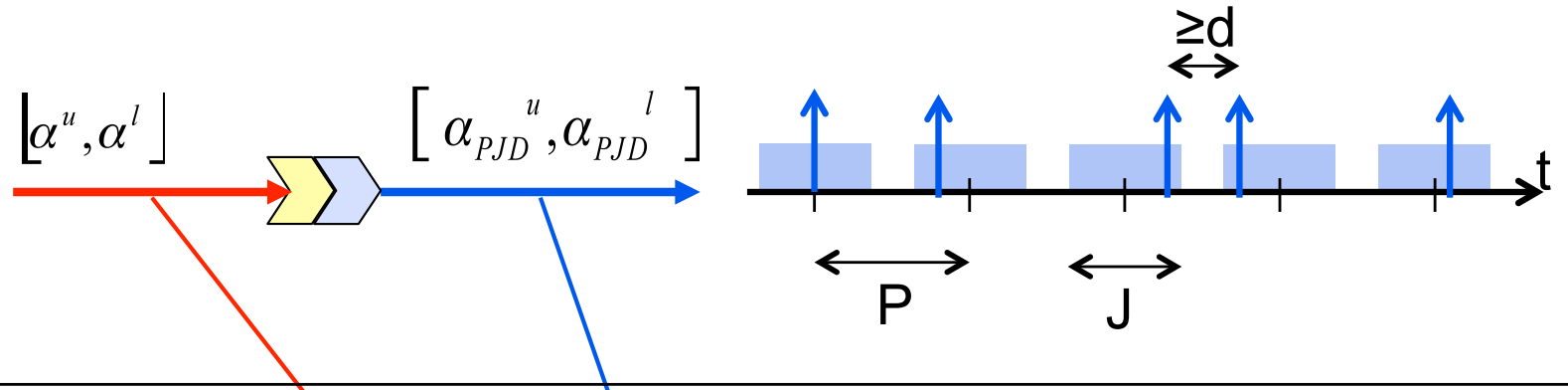
Stream Approximation - PJD

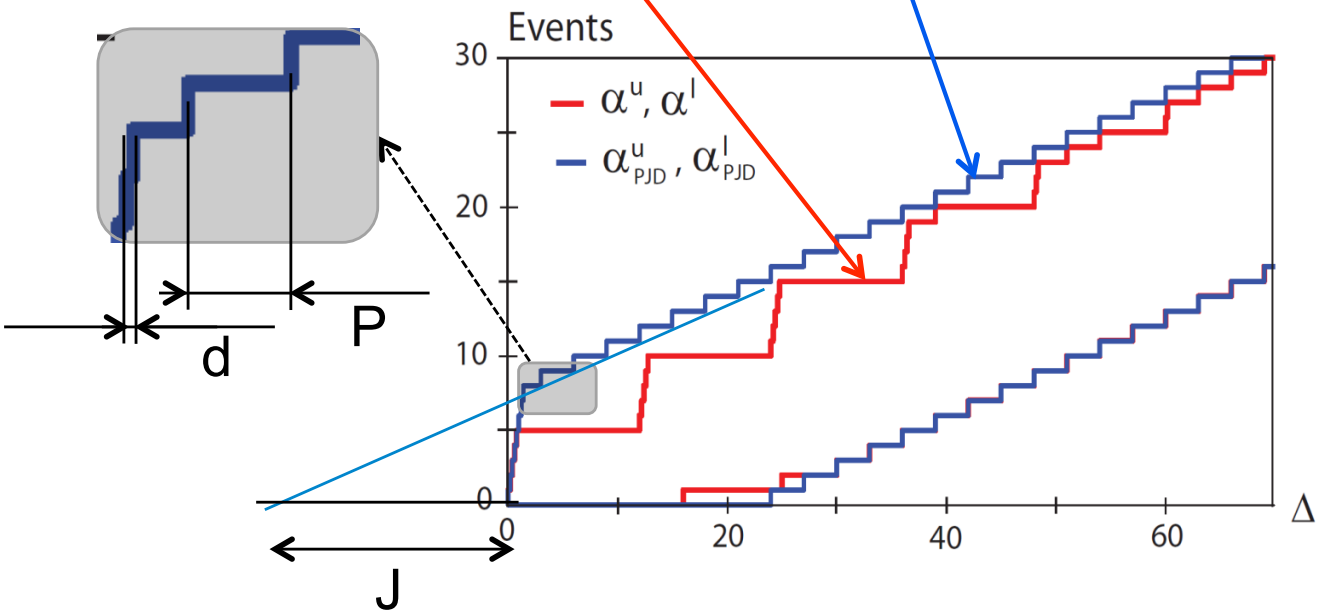


Stream Approximation - PJD

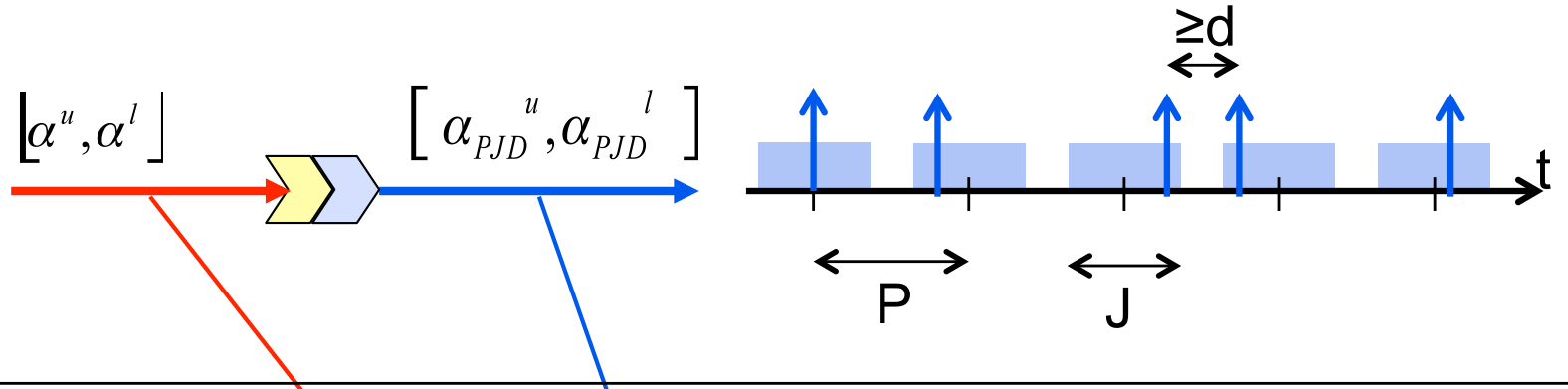


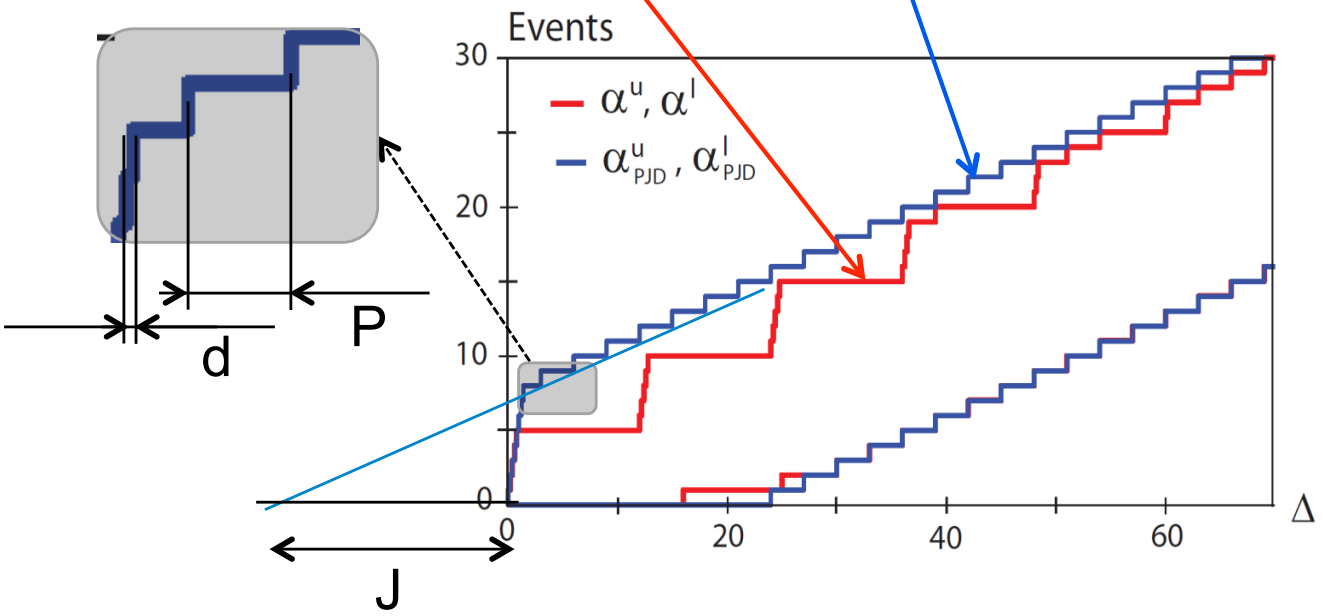
Stream Approximation - PJD





Stream Approximation - PJD





Safe approximation

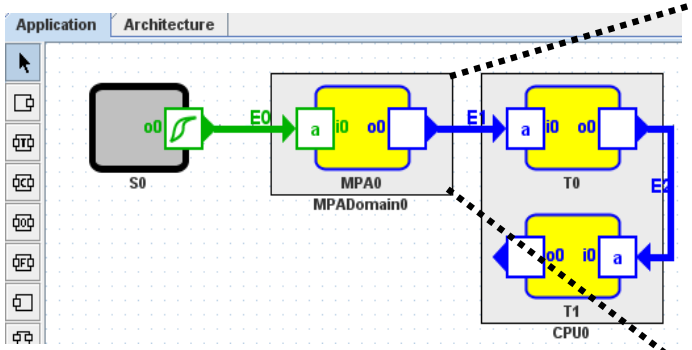
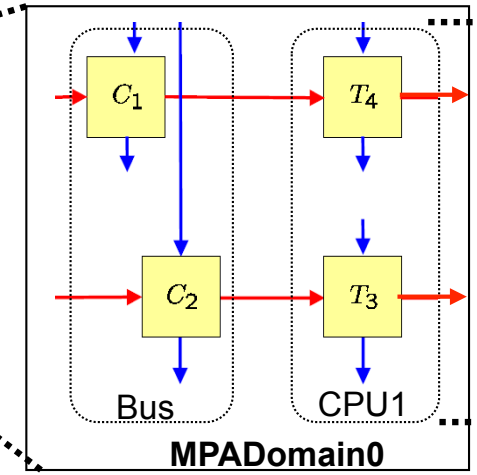
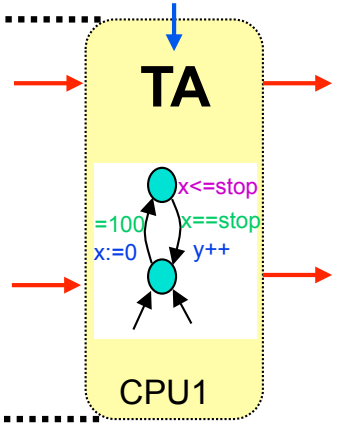
$$[\alpha_{PJD}^{\ u}, \alpha_{PJD}^{\ l}] \succeq [\alpha^u, \alpha^l]$$

Global Picture

TU BS ETHZ

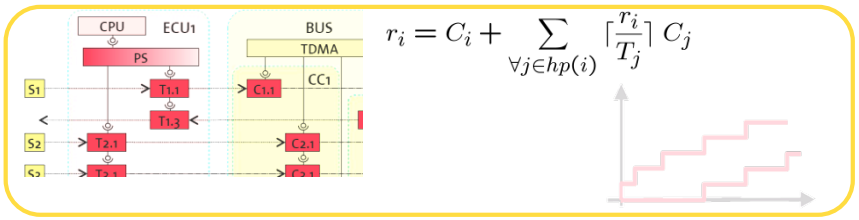
UPPSALA EPFL Trento

Mixing time-interval-based and time-based abstractions

SymTA/S – Holistic Analysis	MPA-RTC Toolbox	Uppaal Parametric Analysis
		
Analytical Model pjd models (but not only)	**Analytical Model** Strictly time interval domain	**State-based Model** Timed automata

Analytic vs. State-Based Abstraction

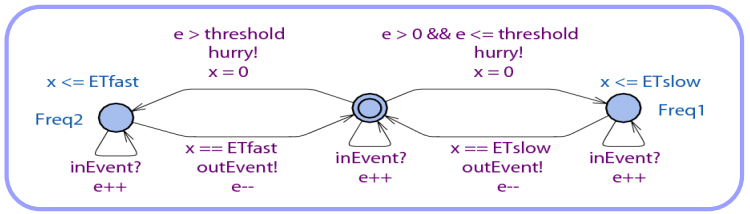
Analytic Real-Time Analysis



Solution of closed form expressions

Examples: RTC, SymTA/S, MAST, …

State-based Real-Time Analysis



Model checking of properties

Examples: Timed Automata (TA), FSM, …

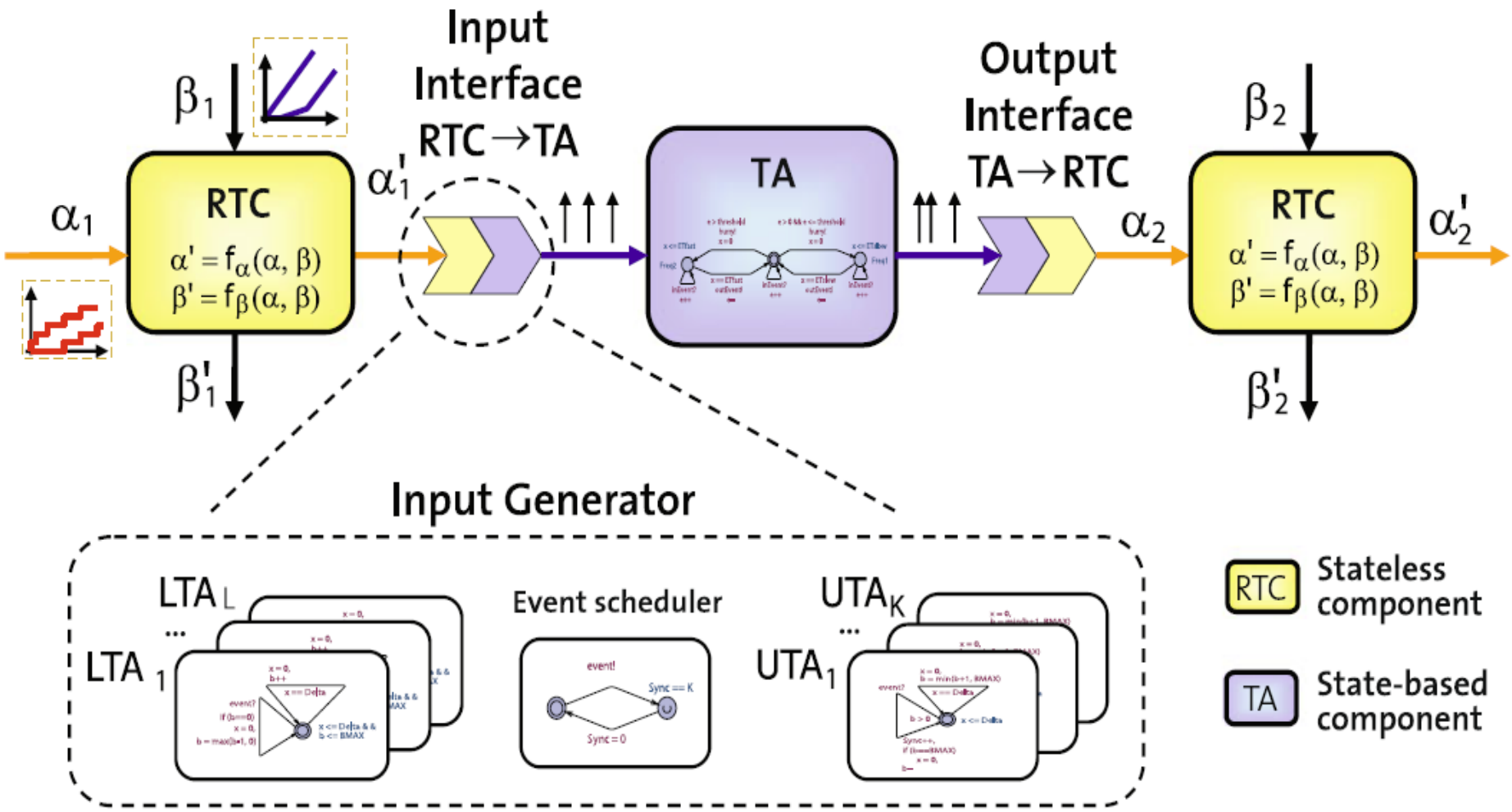
+ Good scalability
+ Fast analysis
– Limited to few specific measures (e.g. delays, buffer sizes)
– Systems restricted to specific models
– Overly conservative results

– Poor scalability
– Slow verification

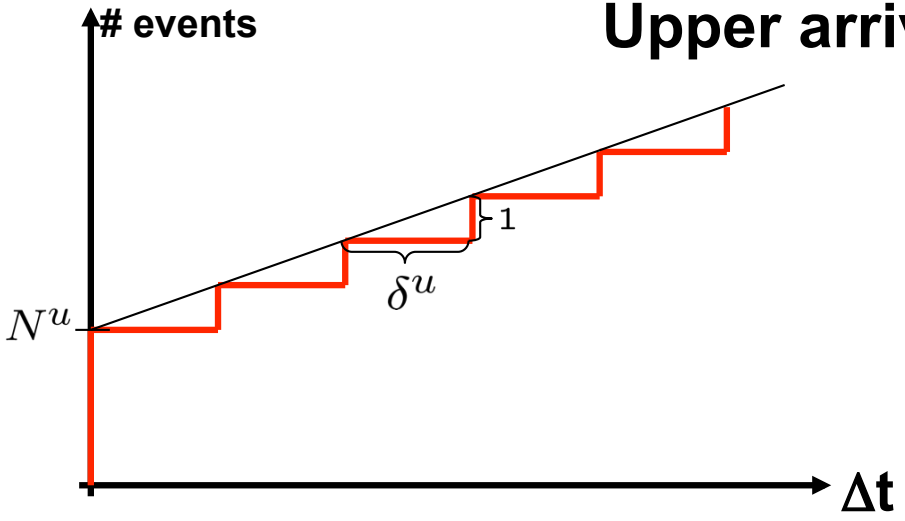
} State space explosion

+ Verification of functional and non-functional properties
+ Modeling power
+ Exact results

Composition Principle



Linear arrival curves

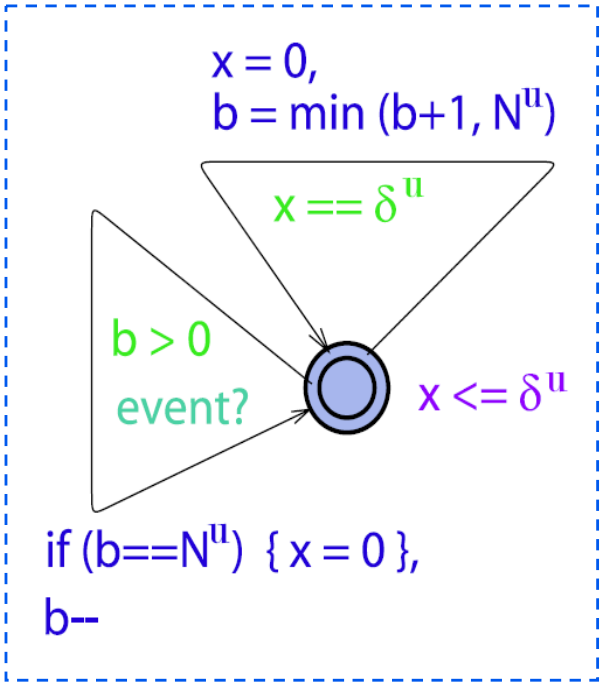


$$\alpha^u(\Delta) = N^u + \left\lfloor \frac{\Delta}{\delta^u} \right\rfloor$$

Max fill level: N^u

Fill rate: $1/\delta^u$

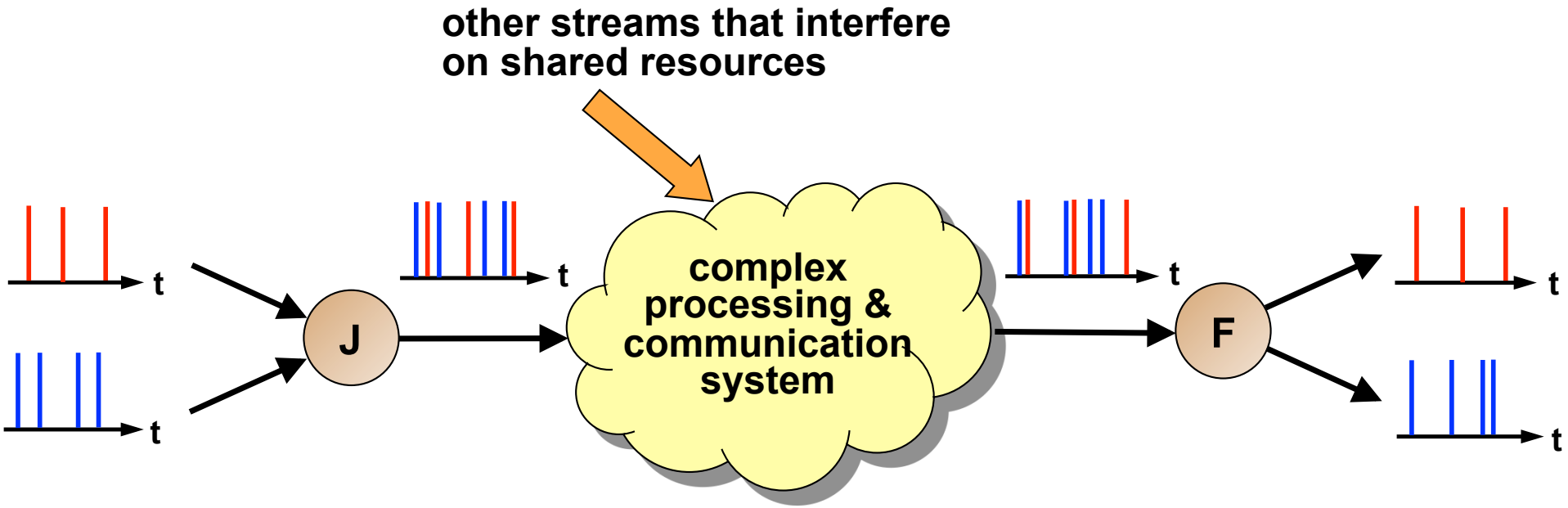
Event emission allowed if fill level > 0



Automaton for linear upper arrival curve

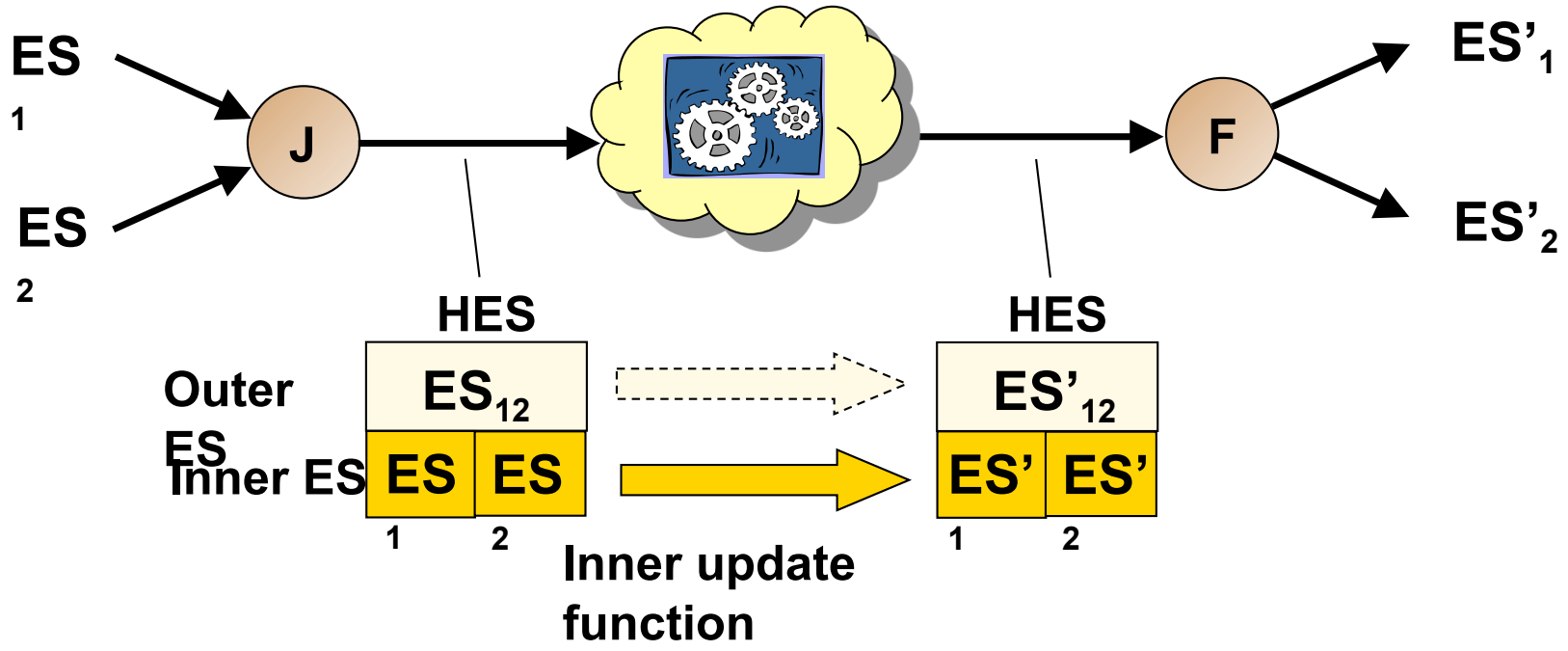
Structured Event Streams (ETHZ – TU BS)

Relevant design pattern:



Hierarchical Event Model

J. Rox & R. Ernst, *Modeling Event Stream Hierarchies with Hierarchical Event Models*

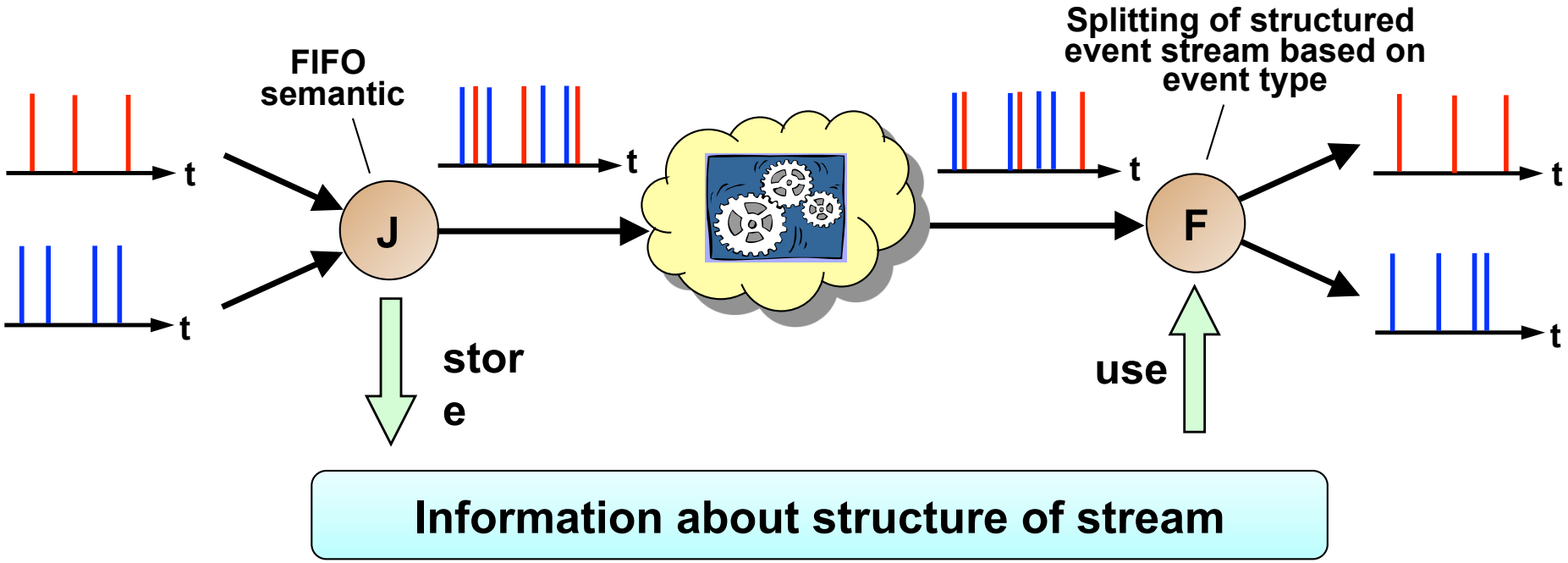


Limitations: - Method is not transparent to existing component models

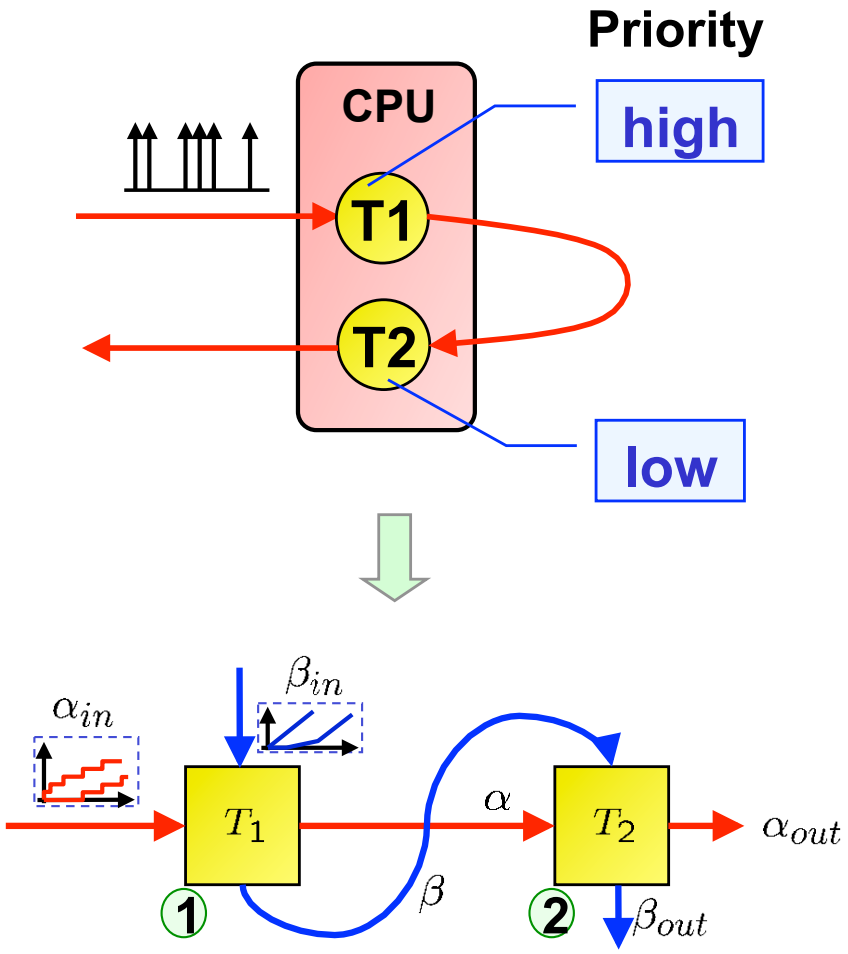
- Deep processing of HES required by component models

Structured Event Streams

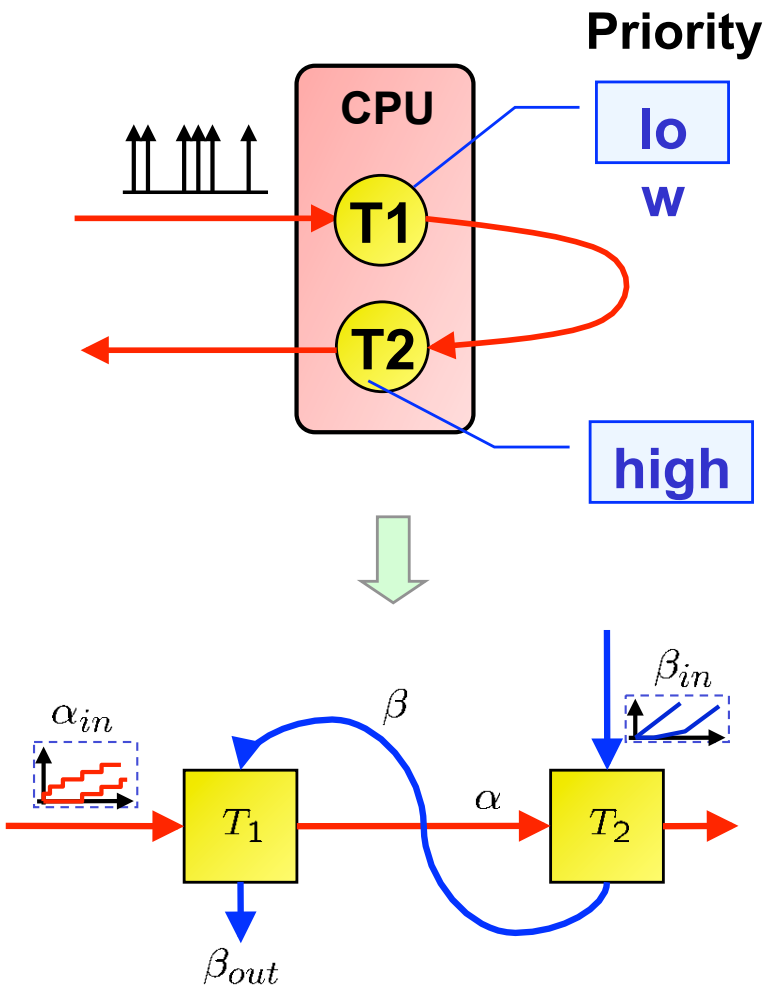
Assumption: The processing components preserve the order of the events in a stream



Cyclic Resource Dependencies *(ETHZ – UPPSALA)*



Acyclic dependency



Cyclic dependency

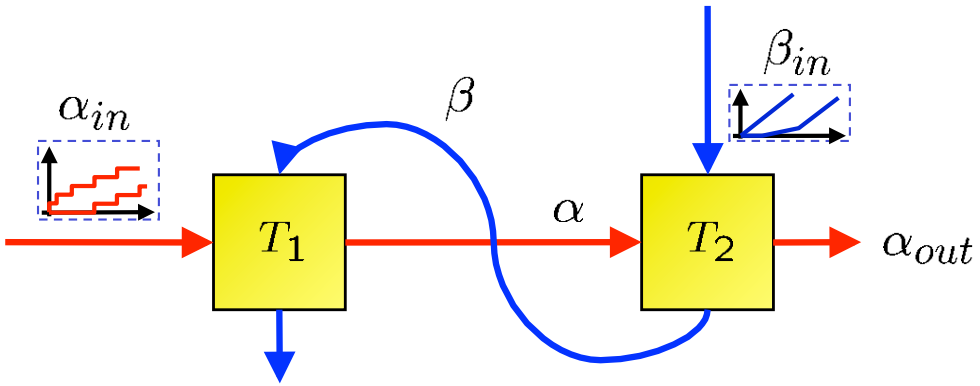
Cyclic Resource Dependencies

$$\Sigma := (\alpha_{in}, \beta_{in}, \alpha, \beta)$$

$$\Psi : \Sigma \rightarrow \Sigma'$$

(reflects MPA transfer functions)

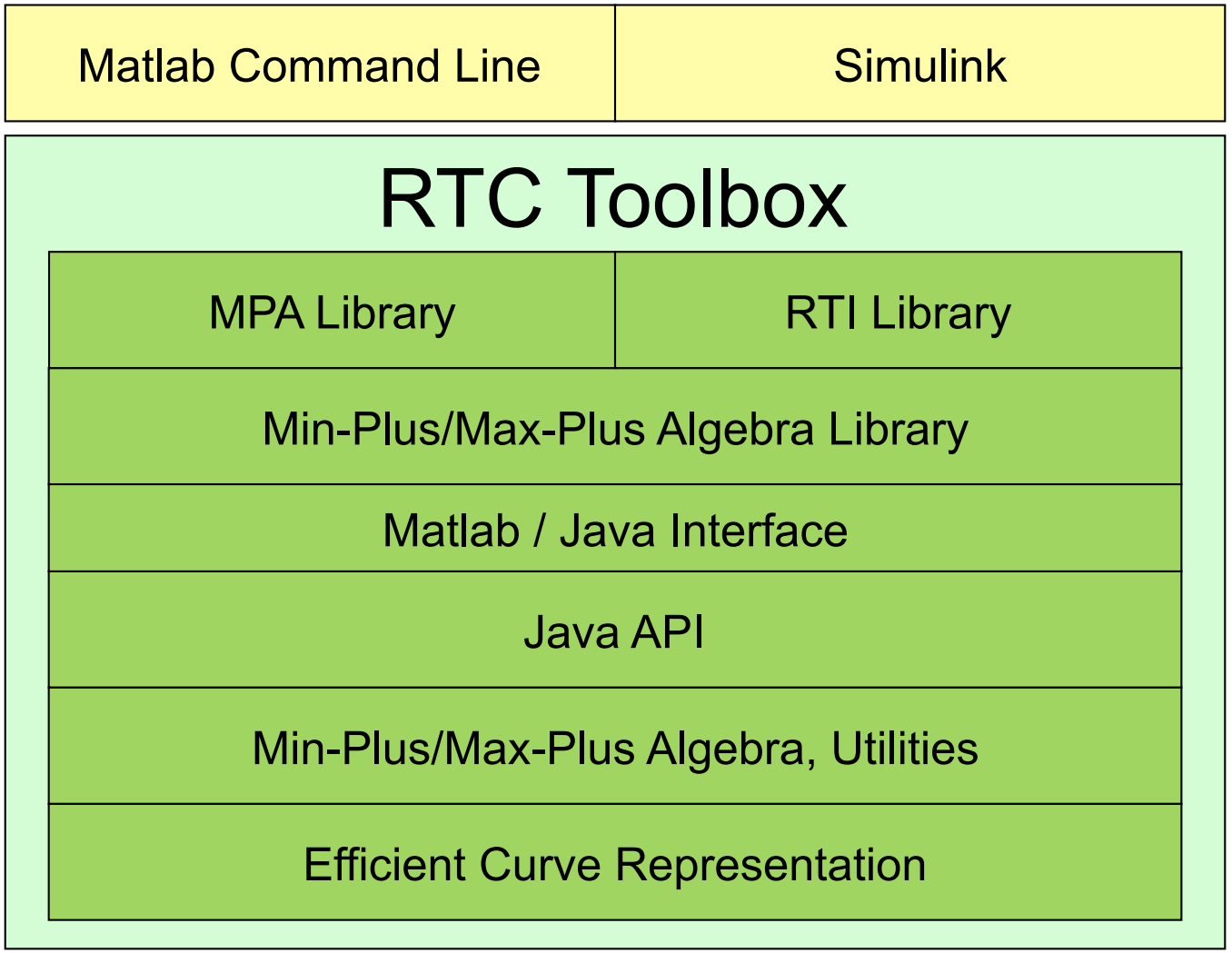
$$\Sigma^0, \Psi(\Sigma^0), \Psi^2(\Sigma^0), \ldots \xrightarrow{?} \Sigma^*$$



Questions:

- **Can there be several fixpoints ?**
- **Is any fixpoint of Ψ correct with respect to the system traces ?**
- **Can the optimal fixpoint be computed with the above sequence ?**
- **How to choose the initial approximation Σ^0 ?**

RTC Toolbox (www.mpa.ethz.ch/rtctoolbox)

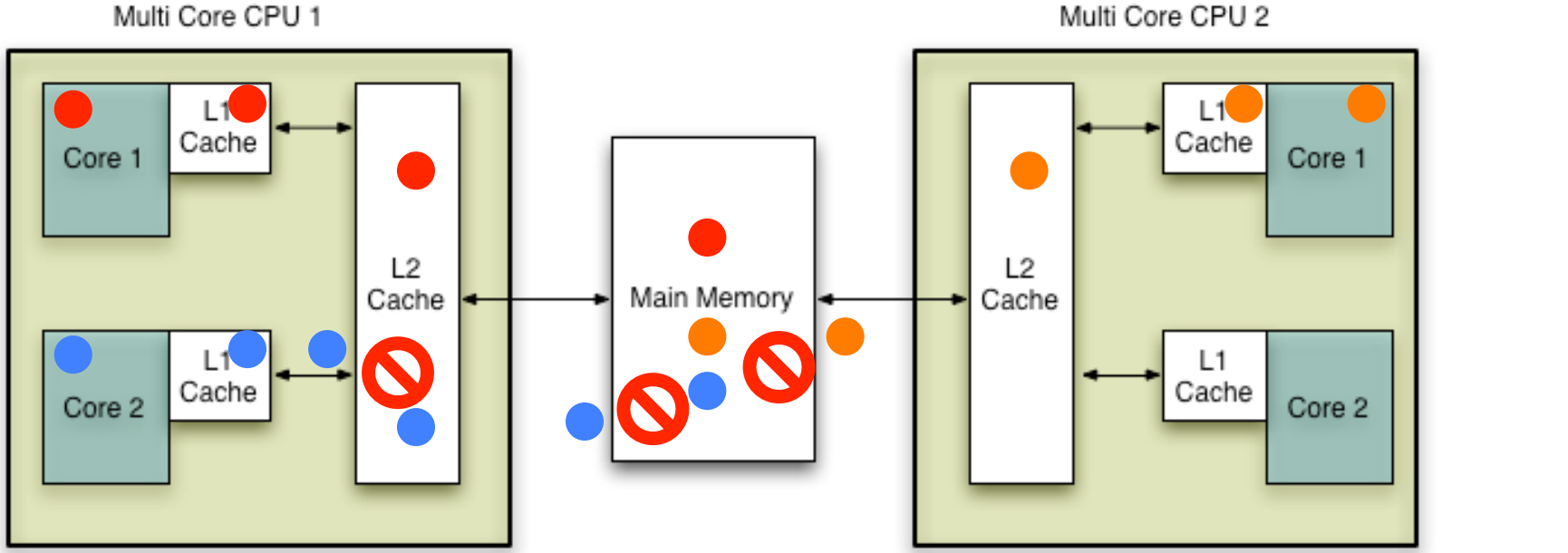


Contents

- Overview about DOL
- Heterogeneous Modeling
- **Interference Analysis for Multicore Systems**
  - *Joint work with Marco Caccamo (Int. Cooperation)*
- Experimental Results

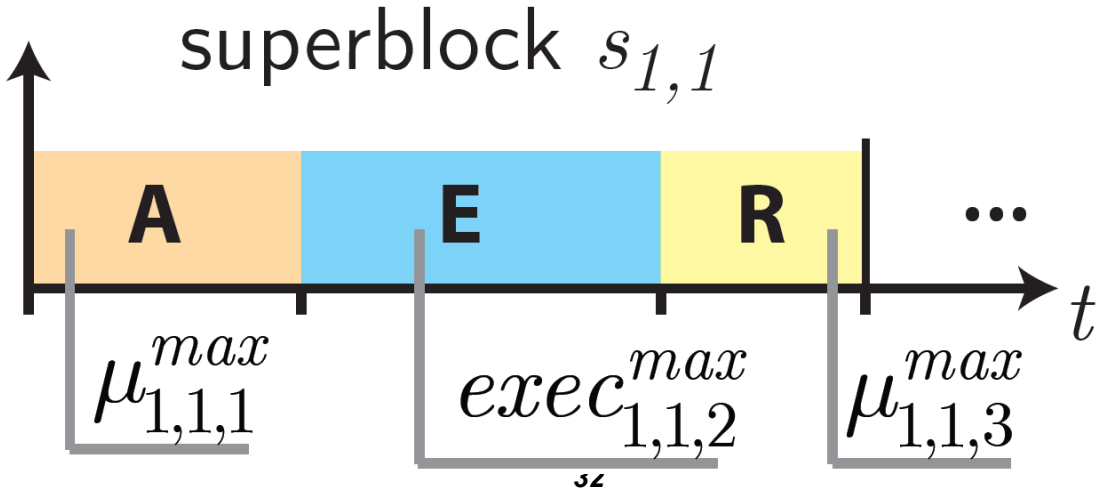
Motivation

- COTS Systems use shared resources (Memory, Bus)
- Multiple entities competing for shared resources
  - waiting for other entities to release the resource
  - accessing the resources

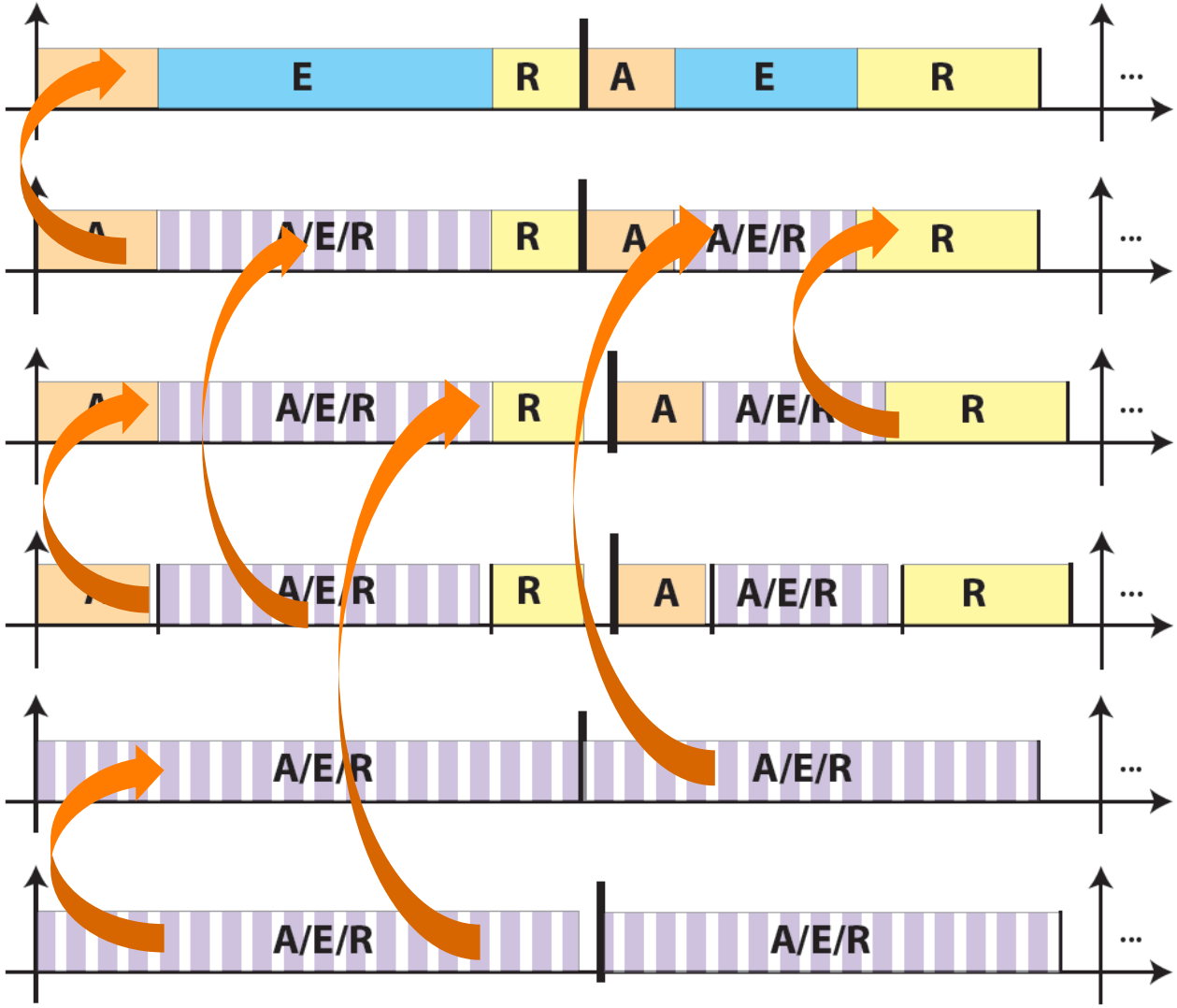


Task / Superblock Model

- Tasks are structured as sequences of superblocks
  - fixed order of execution
  - upper bounds on execution and communication demands
- Dedicated phases for resource access and computation
  - phases have different amount of access requests
  - structure increases predictability (in terms of WCRT)
  - model motivated by industrial applications in the automotive industry



Resource Access Models



DS dedicated sequential phases, sequential superblocks

HS hybrid sequential phases, sequential superblocks

HTS hybrid sequential phases, time-triggered superblocks

HTT hybrid time-triggered phases time-triggered superblocks

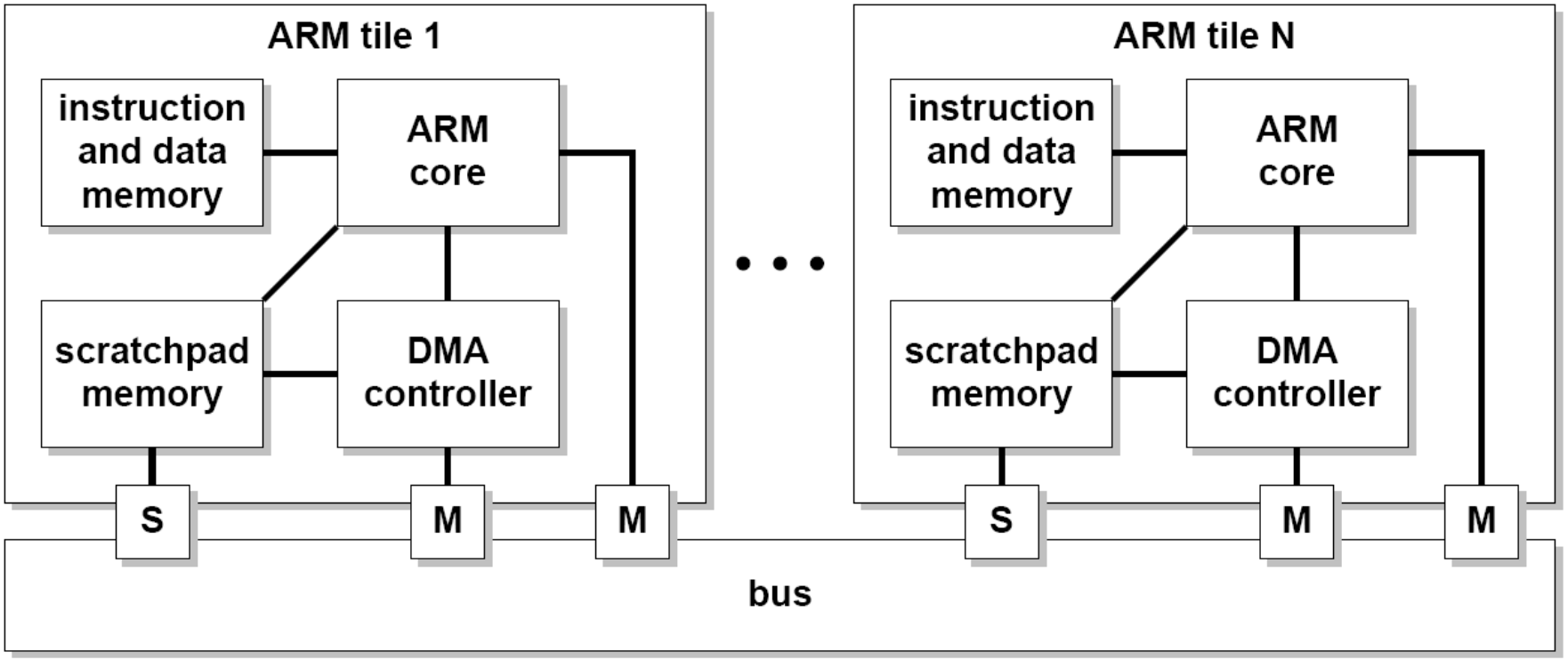
GS general sequential phases, sequential superblocks

GTS general sequential phases, time-triggered superblocks

Contents

- Overview about DOL
- Heterogeneous Modeling
- Interference Analysis for Multicore Systems
- **Experimental Results**
  - *MPARM Simulator Luca Benini (U Bologna)*

Integration / Architecture Template

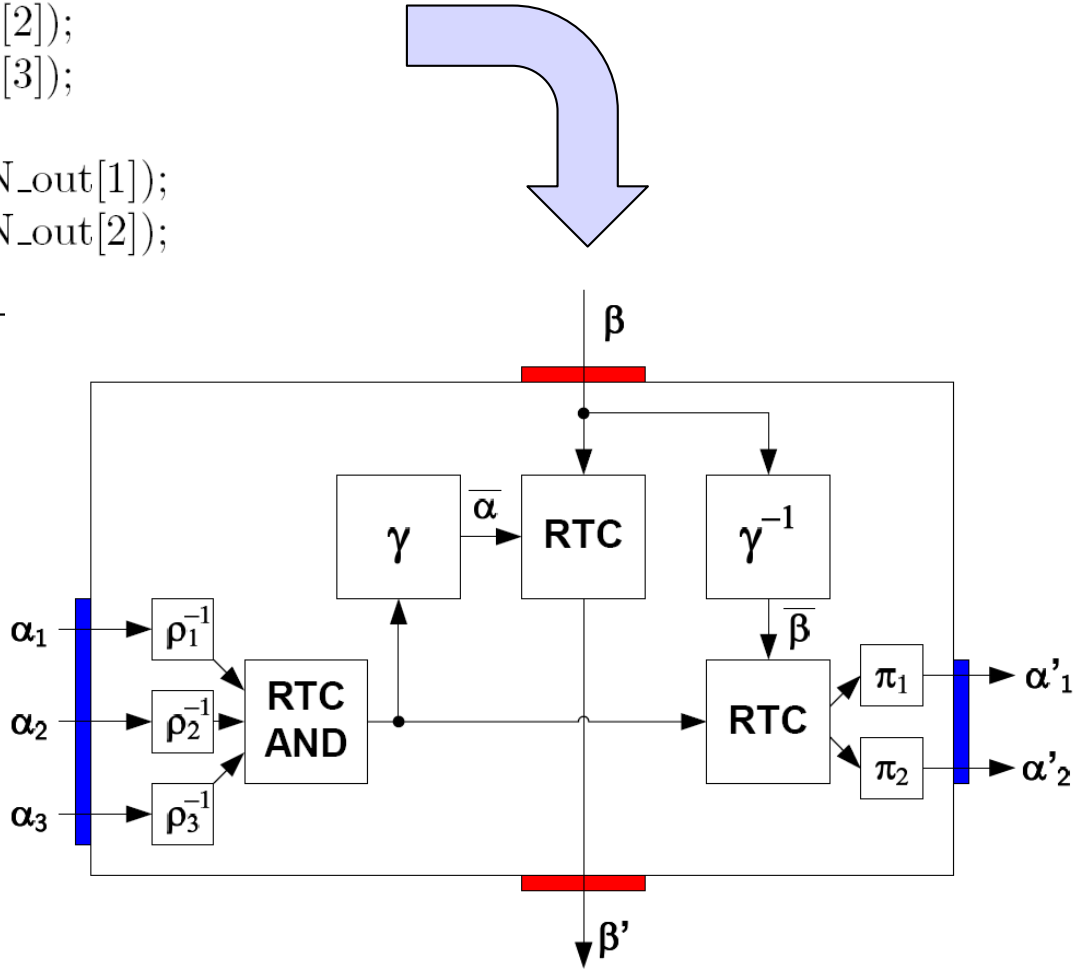


(ETHZ – U BOLOGNA)

Integration / Application Modeling

Algorithm 1 Example of a process with multiple inputs and outputs.

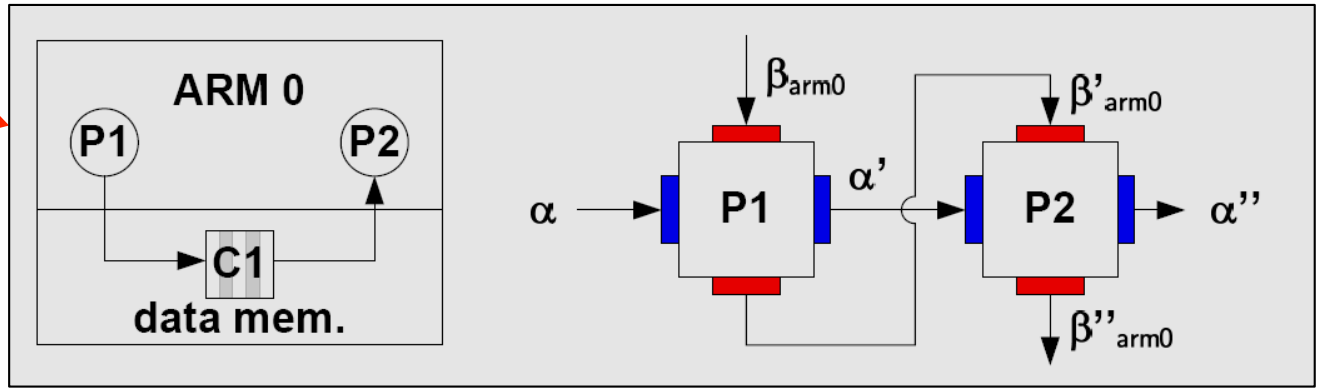
```
1: function FIRE(DOLProcess *p)
2:     DOL_read(input[1], buffer_in[1], N_in[1]);
3:     DOL_read(input[2], buffer_in[2], N_in[2]);
4:     DOL_read(input[3], buffer_in[3], N_in[3]);
5:     execute;
6:     DOL_write(output[1], buffer_out[1], N_out[1]);
7:     DOL_write(output[2], buffer_out[2], N_out[2]);
8: end function
```

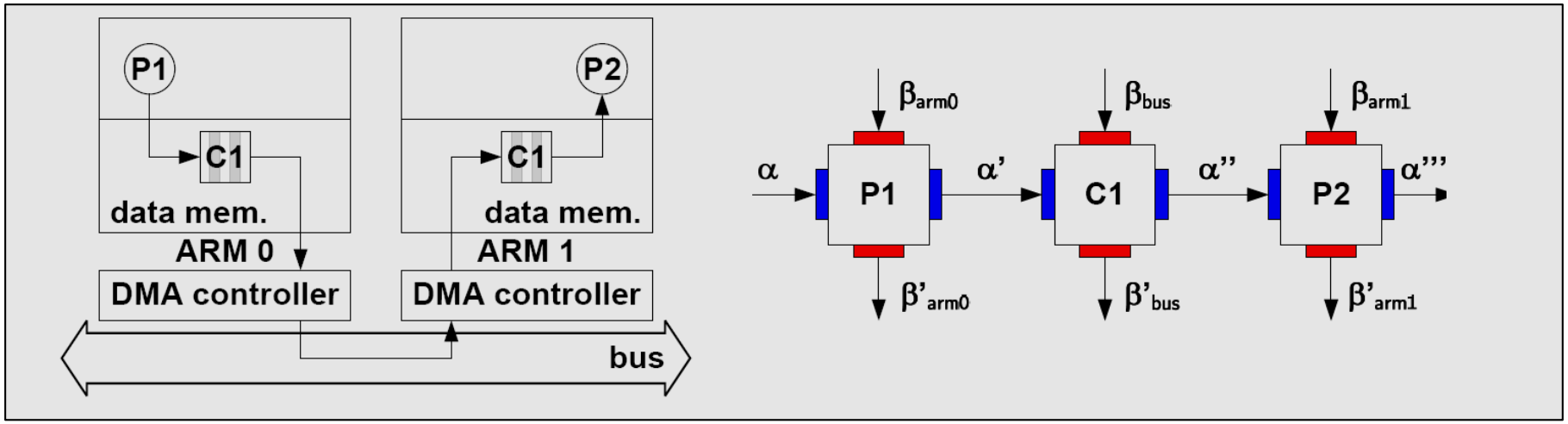


Integration / Communication Modeling

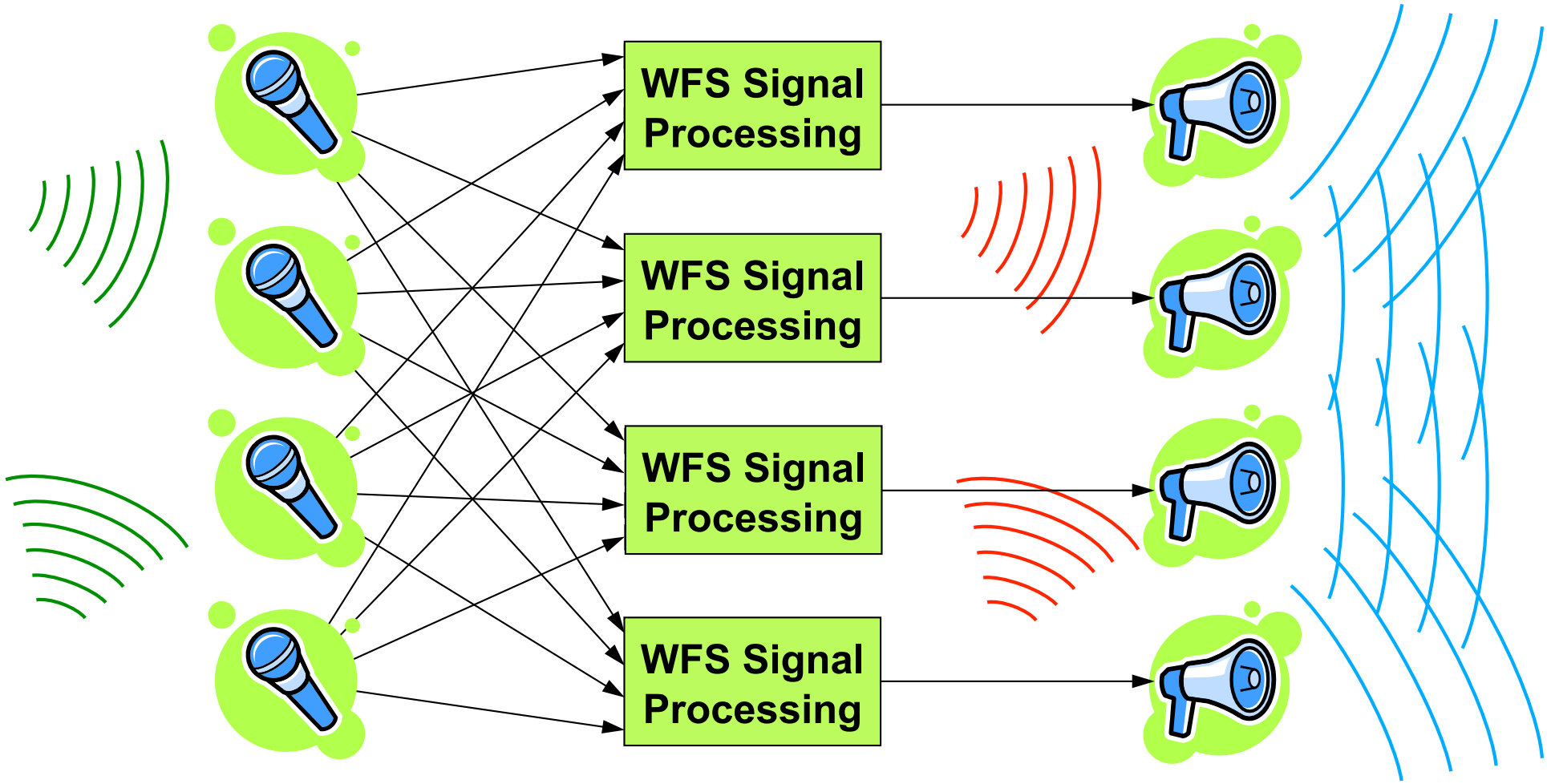
intra-processor communication

inter-processor communication





Example: Wave Field Synthesis



Example: Compilation Times

step		duration		
		P–C	MJPEG	WFS
model calibration (one-time effort)	functional simulation generation	22 s	42 s	35 s
	functional simulation	0.2 s	3.6 s	2.4 s
	synthesis (generation of binary)	2 s	4 s	3 s
	simulation on MPARM	23 s	13550 s	740 s
	log-file analysis and back-annotation	1 s	12 s	3 s
model generation		1 s	1 s	1 s
performance analysis based on generated model		0.2 s	2.5 s	1.4 s

Example: Accuracy

observed estimated

process	proc.	pr.	delay	backlog	pr.	delay	backlog
p–c.p1	1	1	209 ($\leq$ 223)	5 ($\leq$ 6)	2	357 ($\leq$ 401)	6 ($\leq$ 8)
p–c.p3	1	2	329 ($\leq$ 371)	7 ($\leq$ 9)	1	37 ($\leq$ 43)	1 ($\leq$ 2)
p–c.p2	2	1	29 ($\leq$ 38)	1 ($\leq$ 2)	1	30 ($\leq$ 35)	1 ($\leq$ 2)
mjpeg.ss	1	1	203 ($\leq$ 240)	4 ($\leq$ 6)	2	321 ($\leq$ 441)	3 ($\leq$5)
mjpeg.ms	1	2	694 ($\leq$ 781)	1 ($\leq$ 3)	1	133 ($\leq$ 190)	1 ($\leq$ 1)
mjpeg.sf	2	1	2591 ($\leq$ 3014)	5 ($\leq$ 6)	2	3226 ($\leq$ 4315)	6 ($\leq$ 6)
mjpeg.mf	2	2	1881 ($\leq$ 2143)	2 ($\leq$ 4)	1	307 ($\leq$ 340)	1 ($\leq$ 2)
mjpeg.zii	3	1	6164 ($\leq$ 6762)	4 ($\leq$ 6)	1	5971 ($\leq$ 6663)	4 ($\leq$ 6)
wfs.ctrl	1	1	202 ($\leq$ 235)	3 ($\leq$ 5)	3	405 ($\leq$ 795)	5 ($\leq$ 7)
wfs.src	1	2	292 ($\leq$ 387)	4 ($\leq$ 5)	2	228 ($\leq$ 357)	3 ($\leq$ 5)
wfs.ls	1	3	4931 ($\leq$ 5402)	8 ($\leq$ 12)	1	4996 ($\leq$ 5512)	9 ($\leq$ 14)
wfs.comp1	2	1	1606 ($\leq$ 1919)	12 ($\leq$ 15)	2	6157 ($\leq$ 7720)	26 ($\leq$ 30)
wfs.comp2	2	2	5960 ($\leq$ 6838)	25 ($\leq$ 26)	1	1940 ($\leq$ 2156)	15 ($\leq$ 20)


artist
European Network of Excellence
on Embedded Systems Design