Performance Analysis of Distributed Embedded Systems

Part 1: Modular Performance Analysis

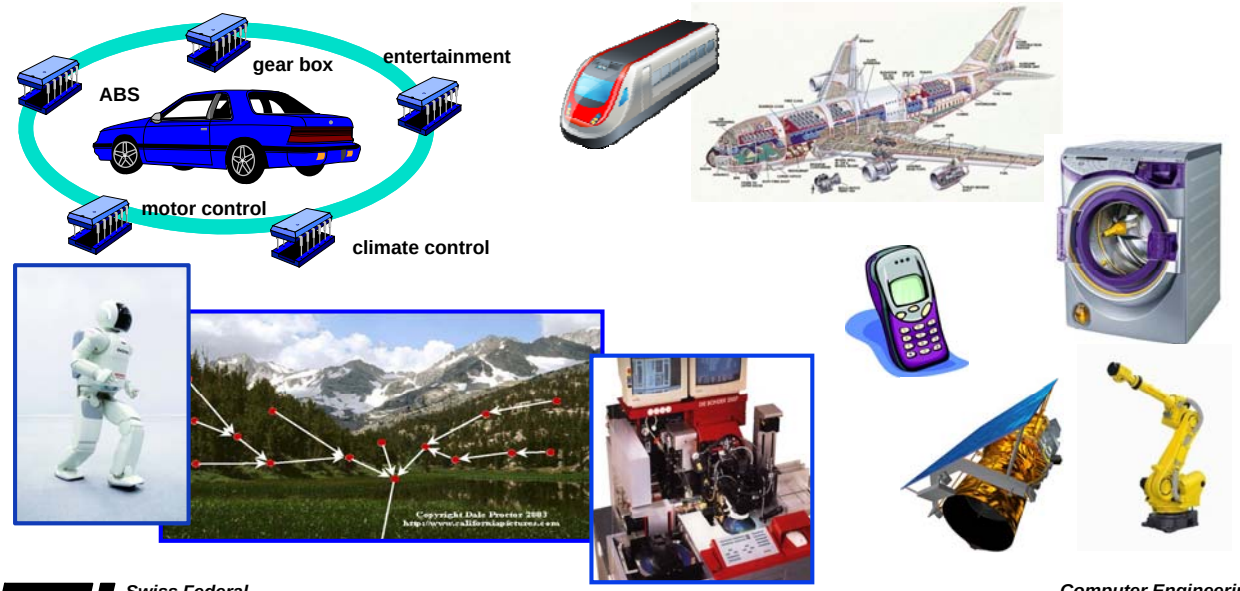
Contents

- ***Drivers***
- Compositional Analysis
  - Overview
  - Real-Time Calculus
- Examples
  - Shapers
  - Artificial Example
  - Shared Resources in Multicore Systems
- Extensions
- Comparison
- Challenges

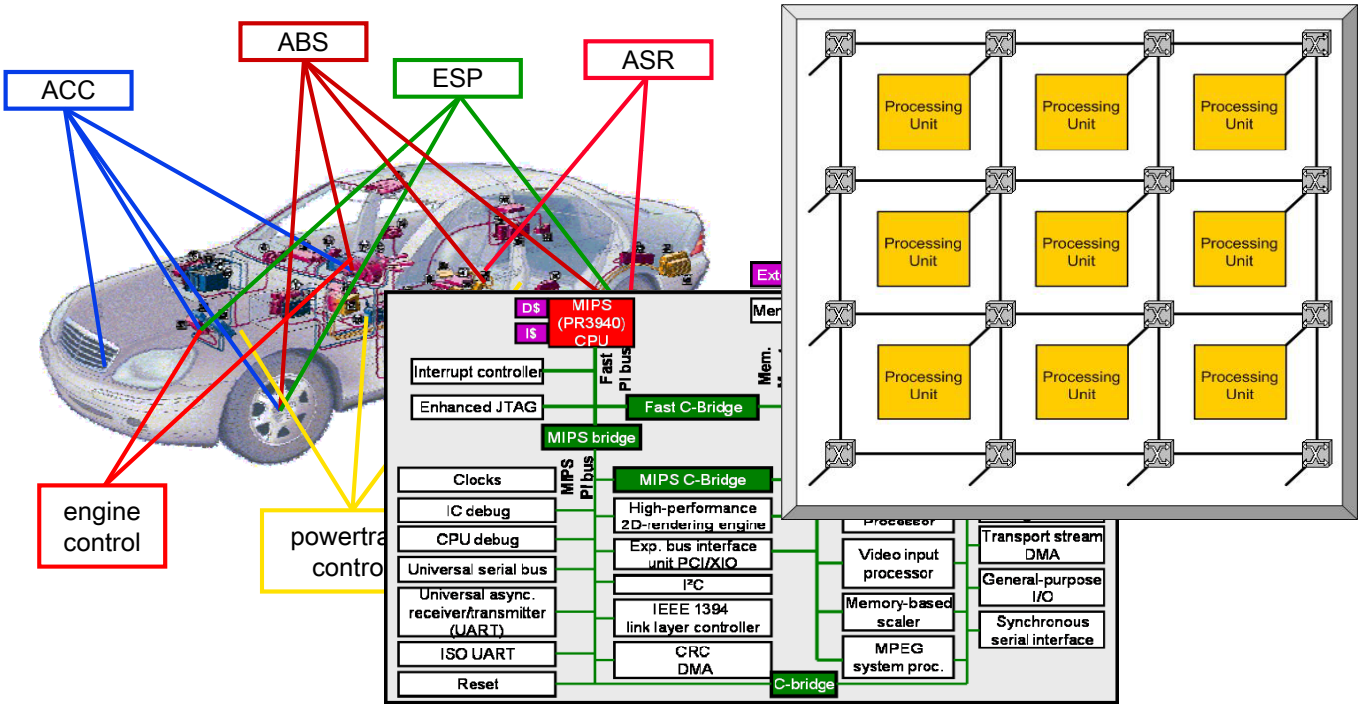
Drivers

Embedded Systems

Information processing system that is physically embedded within a larger system

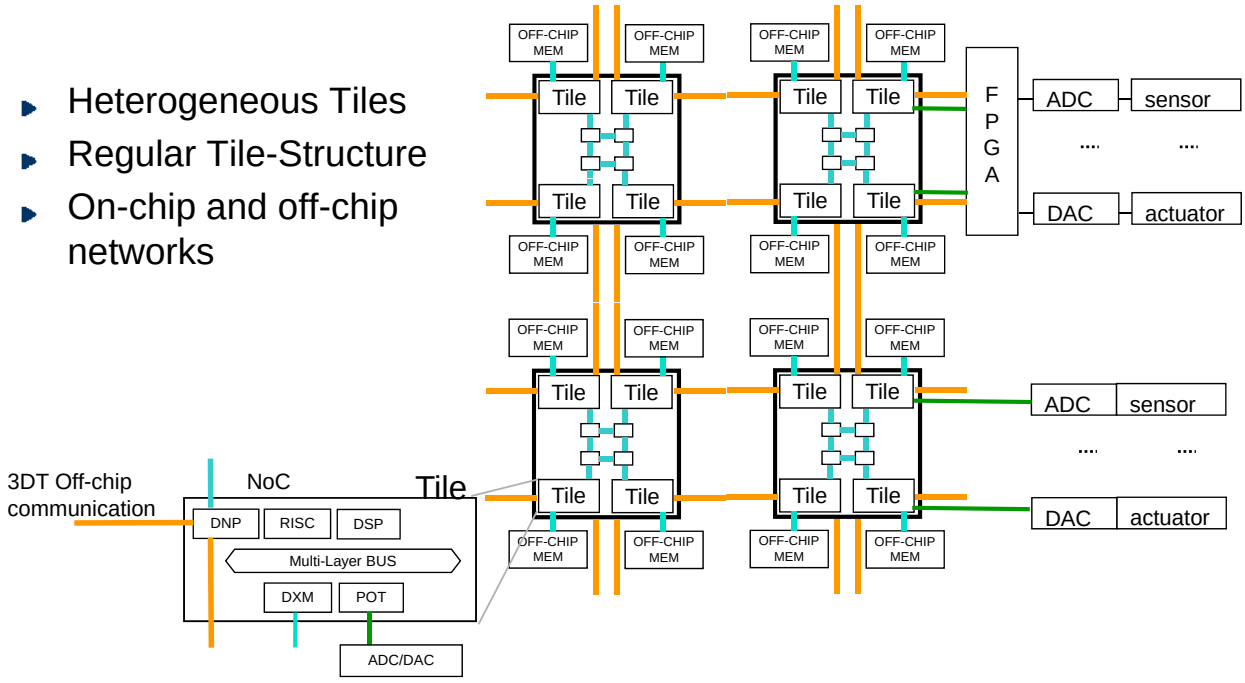


Target Platforms



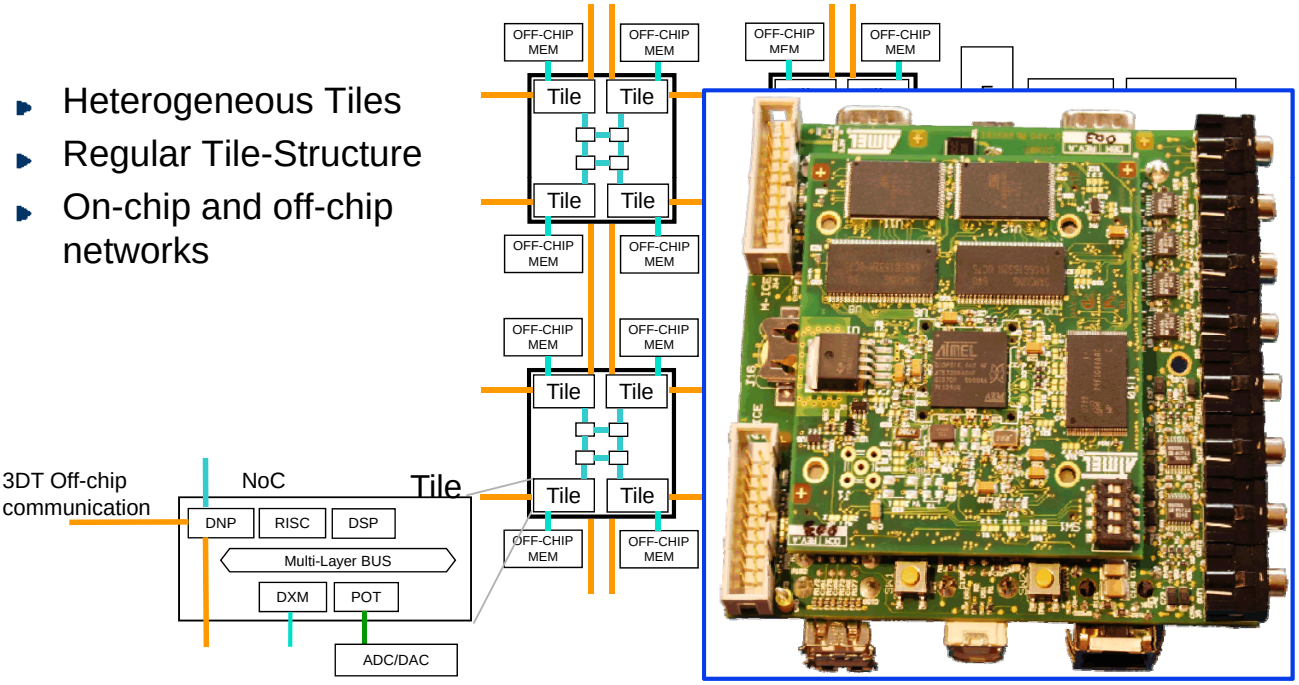
A Sample HW Architecture (EU-SHAPES)

- Heterogeneous Tiles
- Regular Tile-Structure
- On-chip and off-chip networks

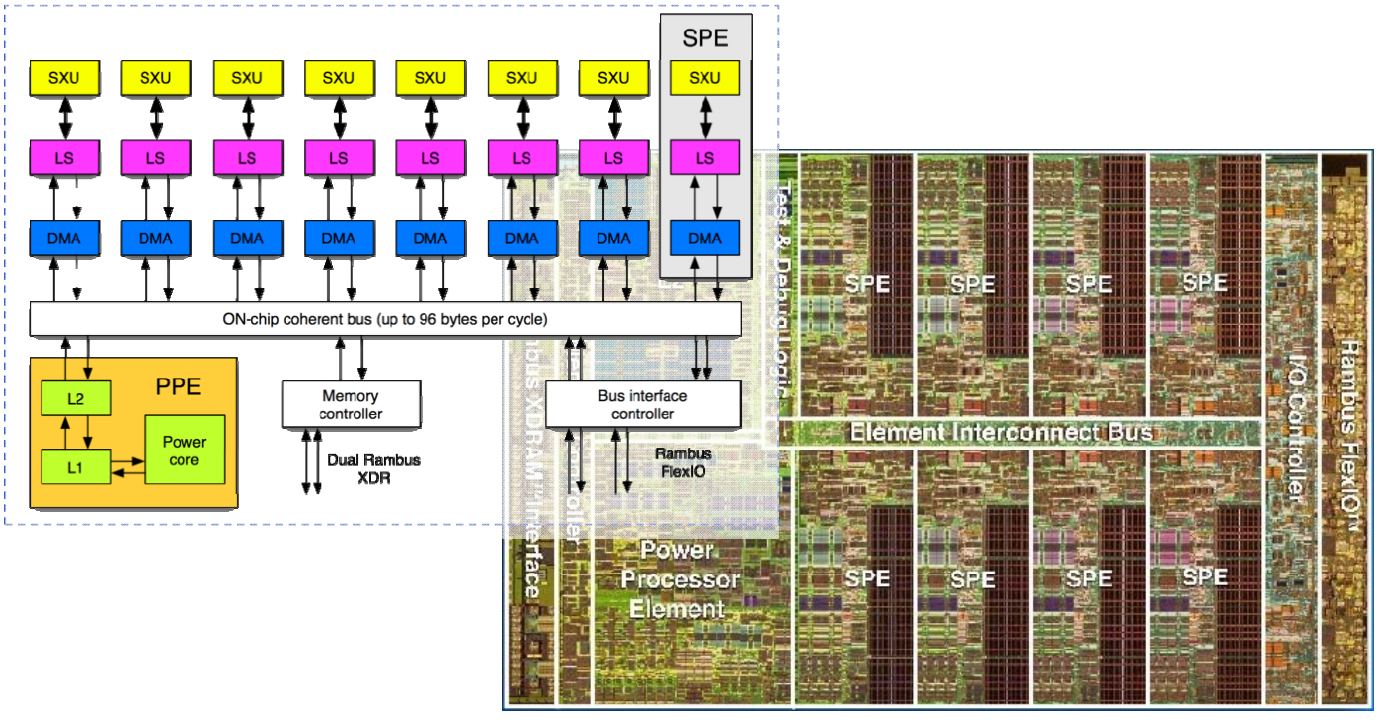


A Sample HW Architecture (EU-SHAPES)

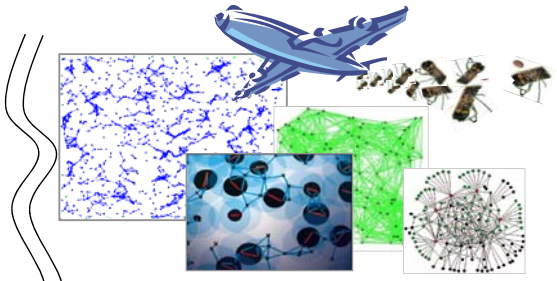
- Heterogeneous Tiles
- Regular Tile-Structure
- On-chip and off-chip networks



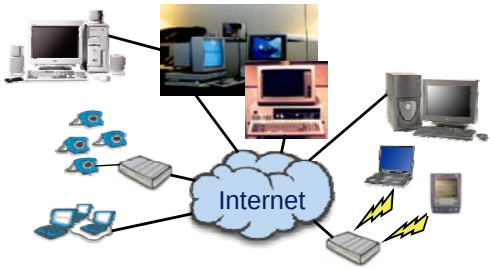
IBM Cell Processor



Big Picture



Centralized Systems → **Networked Systems** → **Large-scale Distributed Systems**



New Applications and System Paradigms

Contents

- Drivers
- ***Compositional Analysis***
  - Overview
  - Real-Time Calculus
- Examples
  - Shapers
  - Artificial Example
  - Shared Resources in Multicore Systems
- Extensions
- Comparison
- Challenges

Compositional Analysis

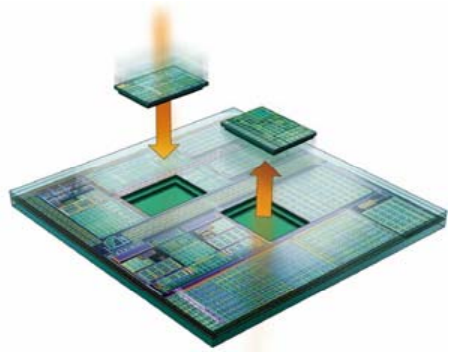
- Overview -

Analysis and Design

Embedded System =
Computation + Communication + Resource Interaction

Analysis:
Infer system properties from subsystem properties.

Design:
Build a system from subsystems while meeting requirements.



Challenge

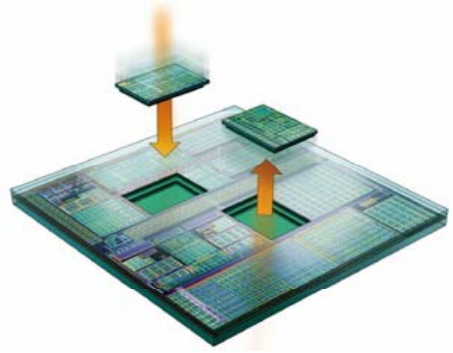
Make Analysis and Synthesis Compositional

Analysis:

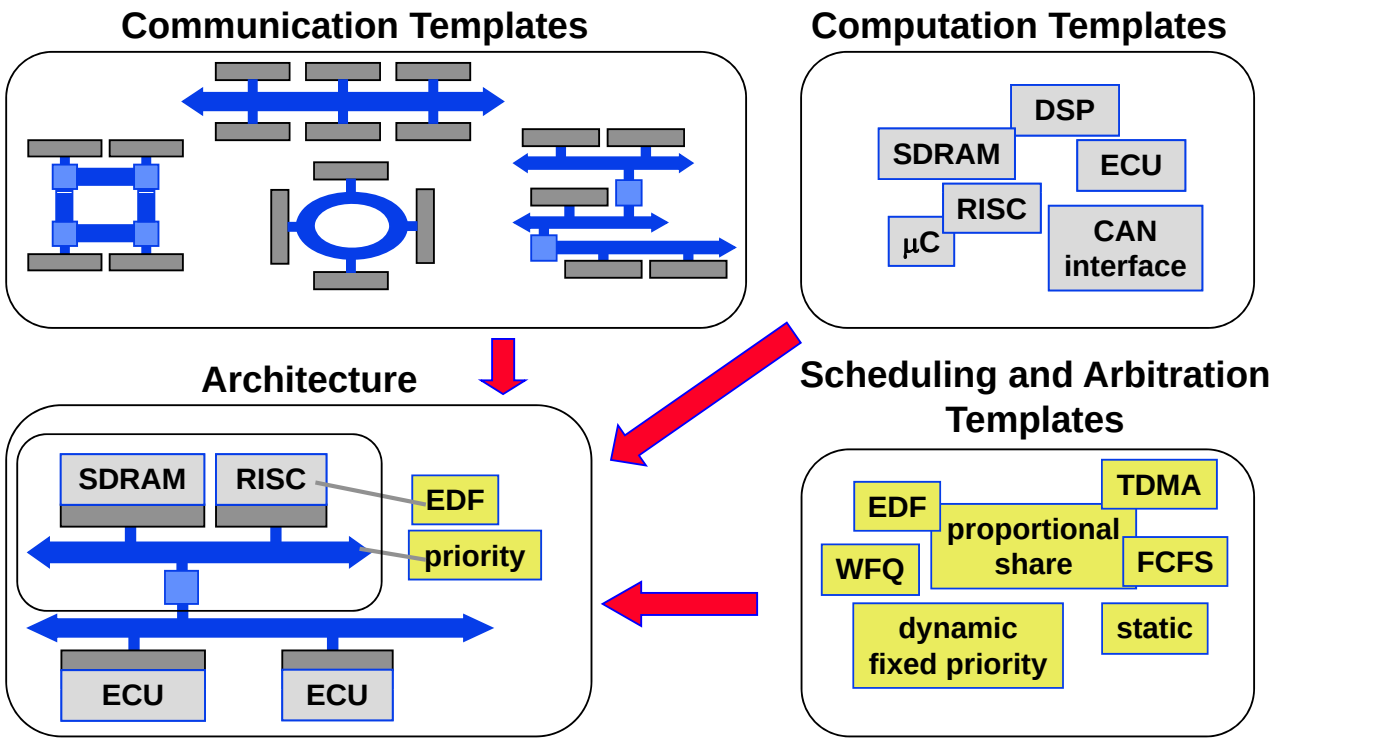
Infer system properties from subsystem properties.

Design:

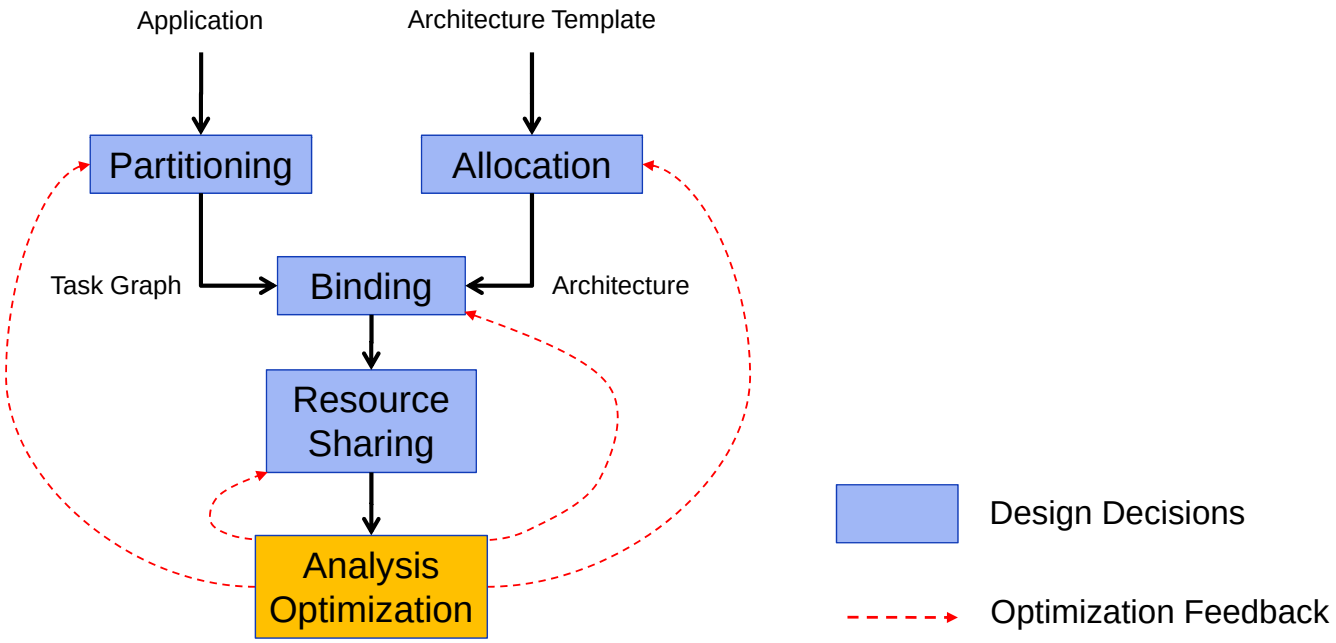
Build a system from subsystems while meeting requirements.



System Composition



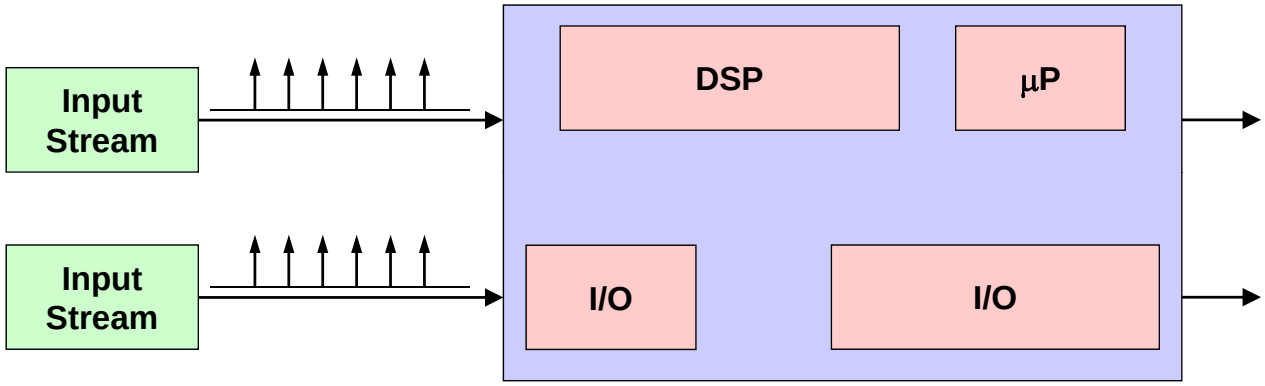
Design Exploration



Why Performance Analysis ?

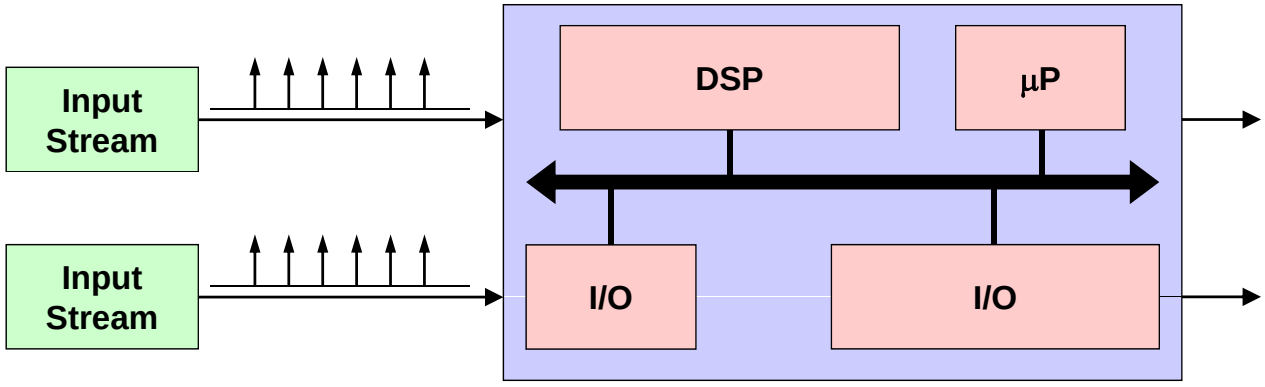
- Prerequisite for ***design space exploration (design decisions and optimization)***
  - part of the feedback cycle
  - get inside into design characteristics and bottlenecks
  - support early design decisions
- Design ***validation***
  - verify system properties
  - used at various design stages from early design until final implementation

Distributed Embedded System



Computational Resources ...

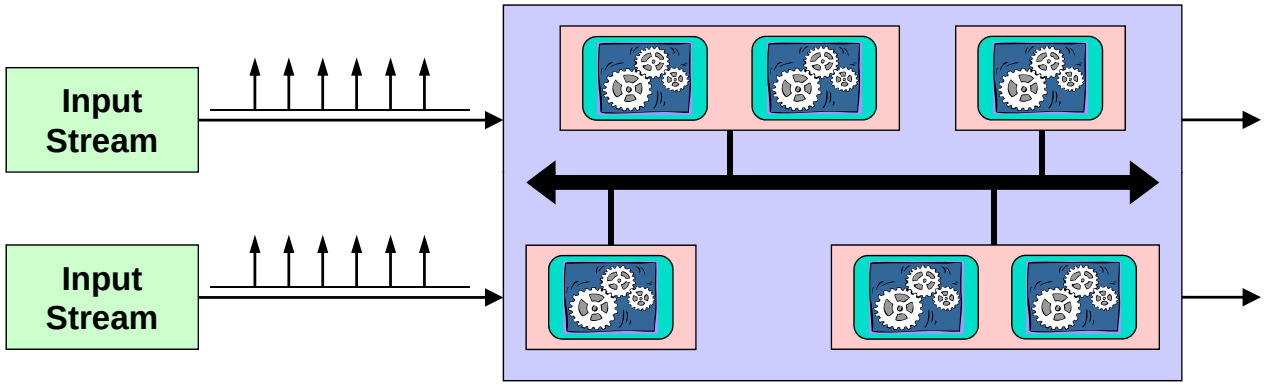
Distributed Embedded System



Computational Resources ...

... Communication Resources ...

Distributed Embedded System



Computational Resources ...

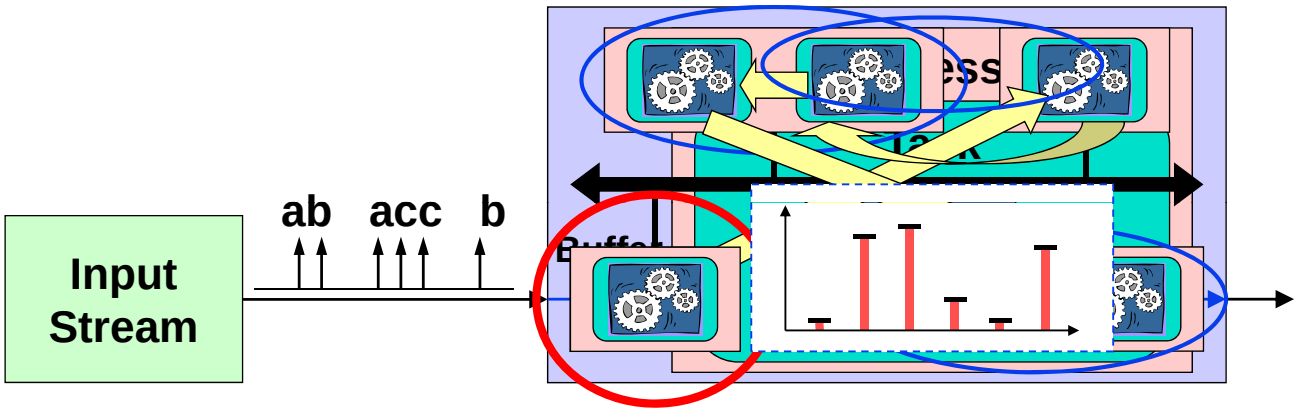
... Communication Resources ...

... Tasks

Why Is Evaluation Difficult ?

- ***Non-determinism:***
  - uncertain system environment, e.g. load scenarios
  - (non-deterministic) computations in processing nodes
- ***Interference:***
  - *sharing* exclusive resources (scheduling and arbitration)
  - interaction between *resource types*: exclusive (computation, communication) and shared (energy)
- ***Long-term dependencies***
  - *resource feedback*: internal data streams interact on exclusive resources which in turn change stream characteristics

Difficulties



Task Communication

Task Scheduling

Complex Input:

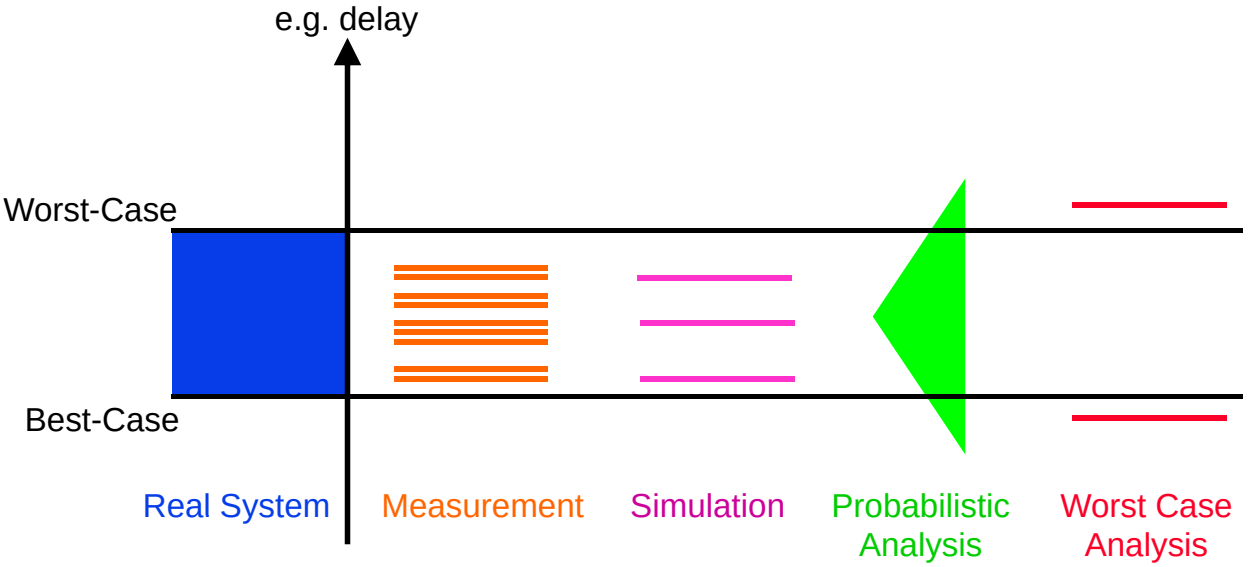
- Timing (jitter, bursts, ...)
- Different Event Types

Variable Resource Availability

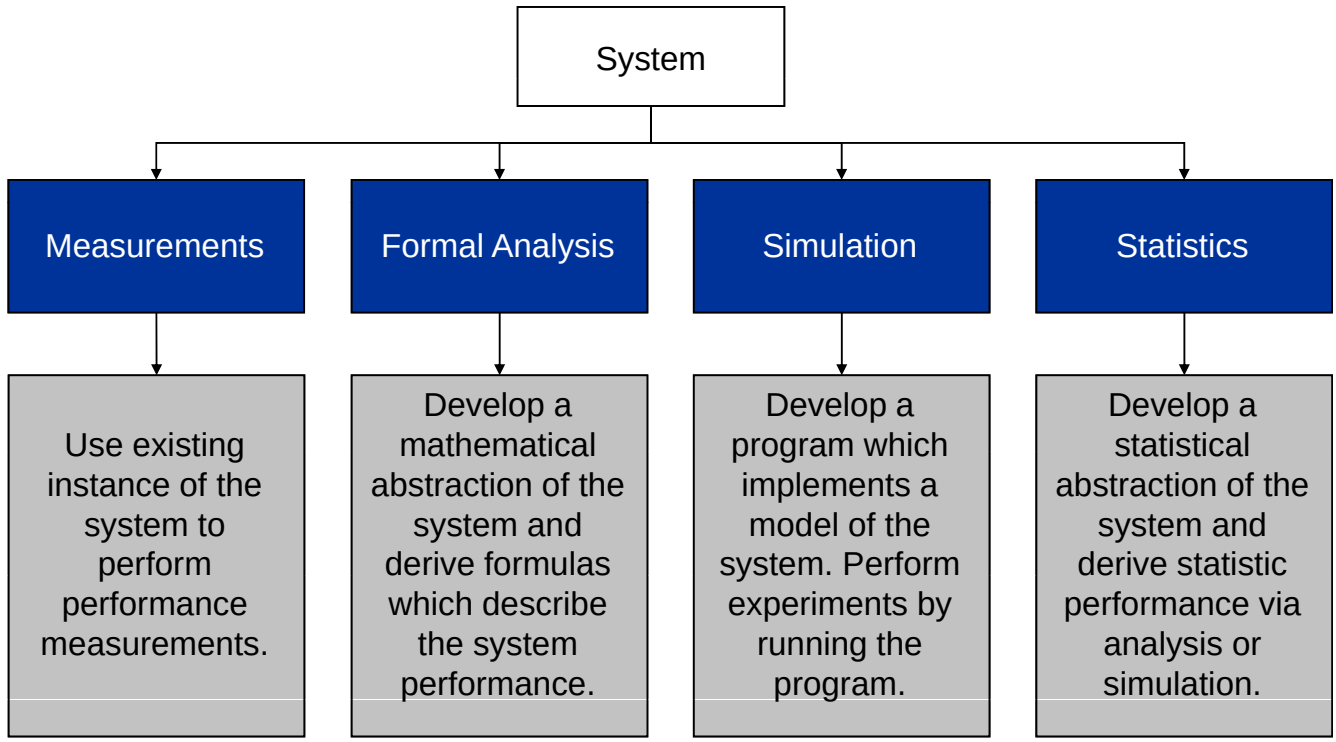
Variable Execution Demand

- Input (different event types)
- Internal State (Program, Cache, ...)

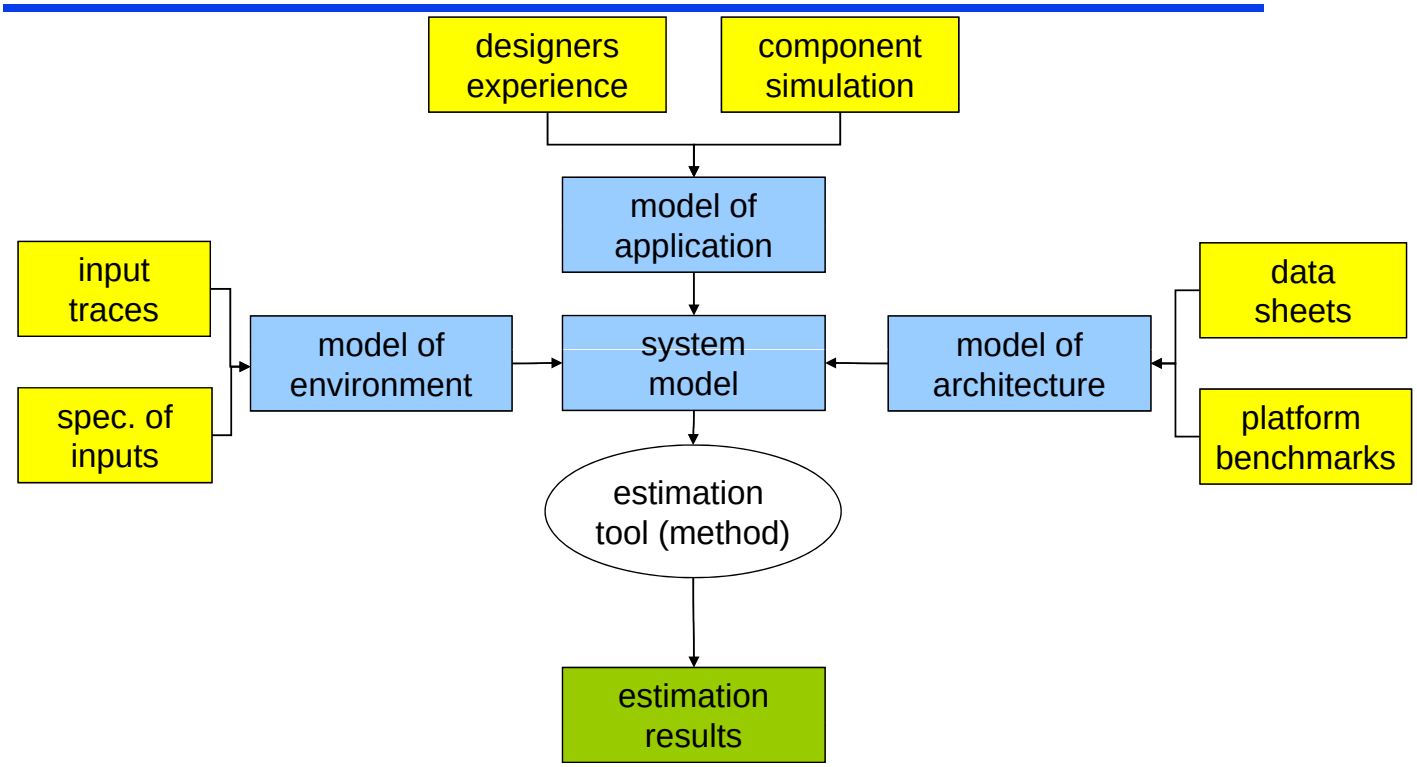
System-Level Evaluation Methods



Overview

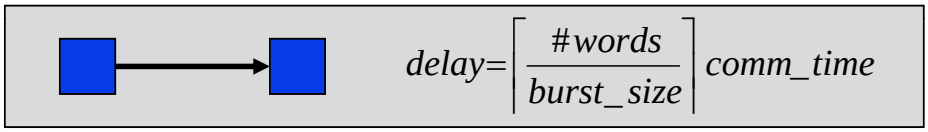


Performance Estimation Methods



1. Analytic Models

- **Static analytic (symbolic) models:**
  - Describe computing, communication, and memory resources by algebraic equations, e.g.



  - Describe system properties by parameters, e.g. data rate
  - Combine relations

- Fast and simple estimation
- Generally inaccurate modeling, e.g. resource sharing not modeled

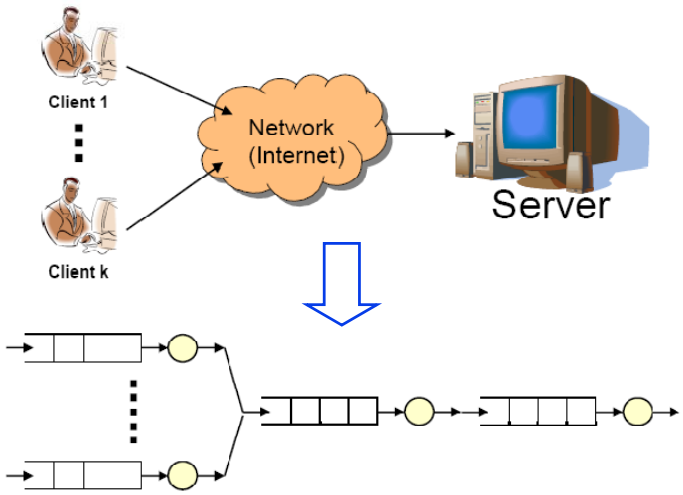
2. Dynamic Analytic Models

- Combination between
  - **Static models** possibly extended by non-determinism in run-time and event processing
  - **Dynamic models** for describing e.g. resource sharing mechanisms (scheduling and arbitration).

- Existing approaches
  - ***Classical real-time scheduling*** theory
  - ***Stochastic queuing theory*** (statistical bounds)
  - ***Non-deterministic queuing theory (***worst case/best case behavior)

Example - Queuing Systems

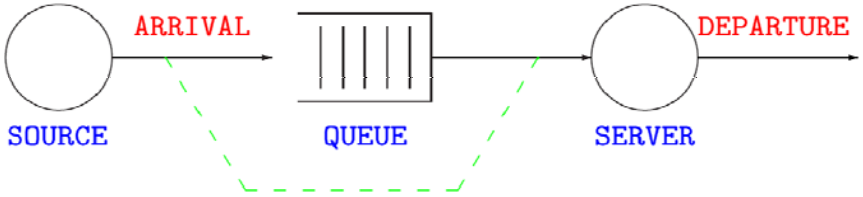
- ***Example:*** clients request some service from a server over a network.



Stochastic Models - Queuing Systems

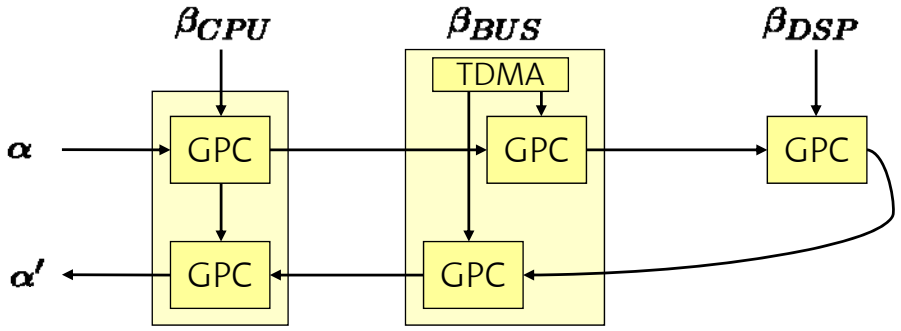
- A ***queuing system*** is described by
  - Arrival rate
  - Service mechanism
  - Queuing discipline
- ***Performance measures***
  - average delay in queue
  - time-average number of customers in queue.
  - proportion of time server is busy

The classical M/M/1 queuing system:
(M = Markovian (exp.) distribution)



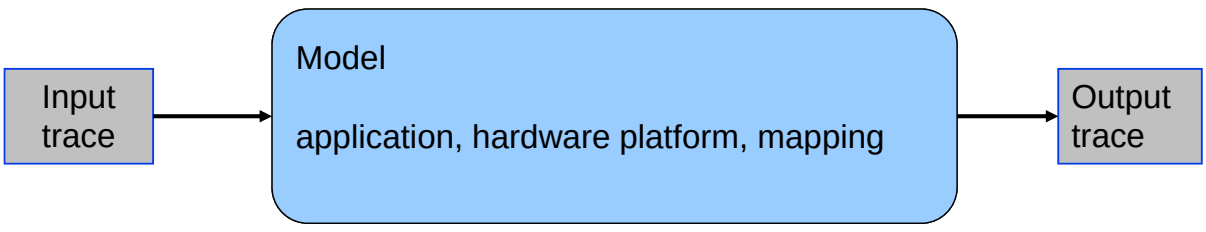
Nondeterministic Models - Queuing Systems

- A ***queuing system*** is described by
  - Arrival function (bounds on arrival times)
  - Service functions (bounds on server behavior)
  - Resource interaction
- ***Performance measures***
  - worst case delay in queue
  - worst-case number of customers in queue.
  - worst-case and best-case end-to-end delay in the system



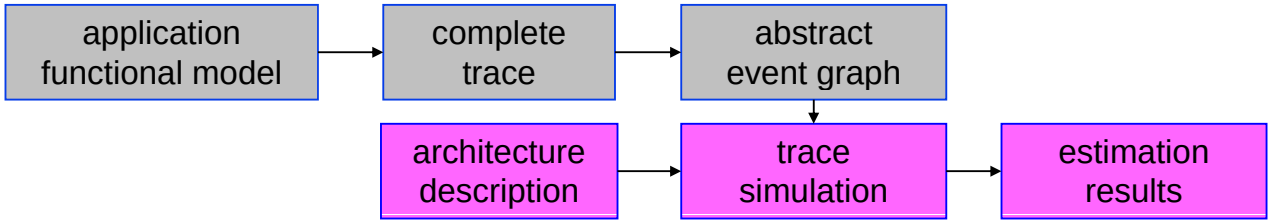
3. Simulation

- Consider the underlying hardware platform and the mapping of the application onto that architecture
- Combine functional simulation and performance data
- Evaluate average-case behavior, for one simulation scenario
- Complex set-up and extensive runtimes
- ... But accurate results and good debugging possibilities



Example: Trace-Based Simulation

- ***Abstract simulation at system-level without timing***
  - Faster than simulation, but still based on a single input trace
- ***Abstraction***
  - Application - represented by *abstract execution traces → graph of events: read, write, and execute*
  - Architecture - represented by *"virtual machines"* and *"virtual channels"* including non-functional properties (timing, power, energy)
- ***Steps***
  - Execution trace determined by functional application simulation
  - Extension of the event graph by non-functional properties



e.g. [Lahiri et al., 2001], [Pimentel et al., 2006]

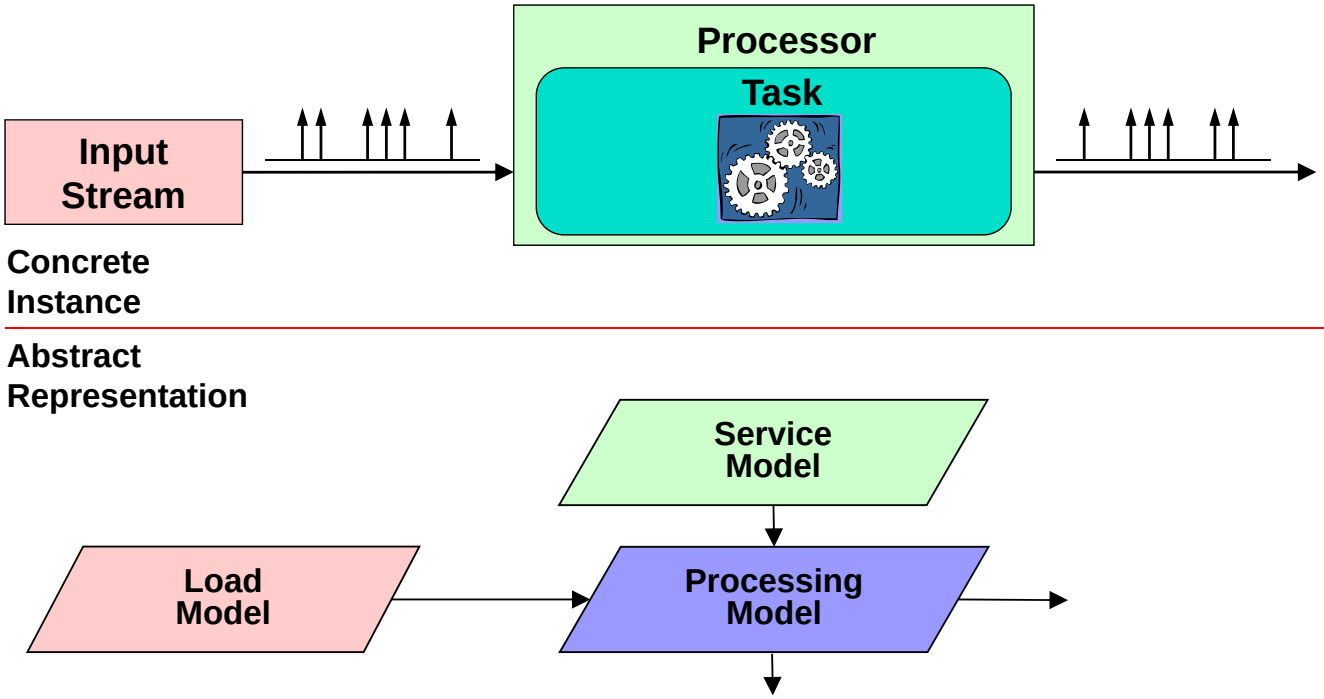
Compositional Analysis

- Real-Time Calculus -

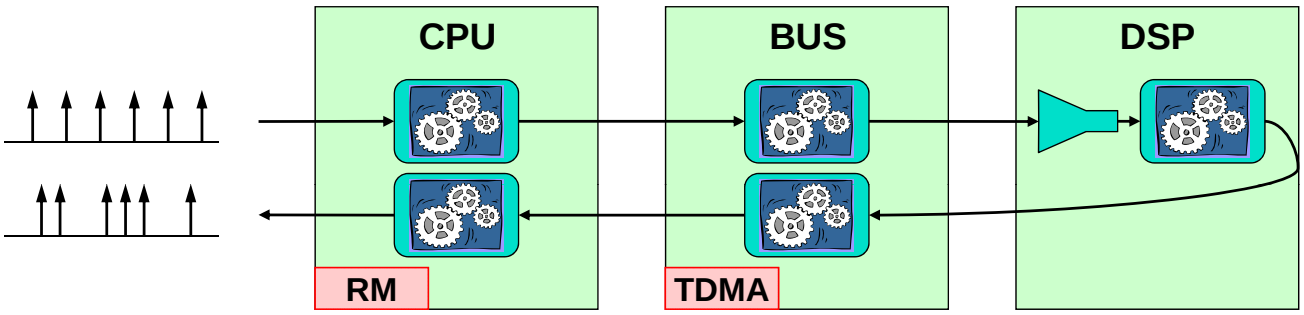
Network/Real-time Calculus Methods

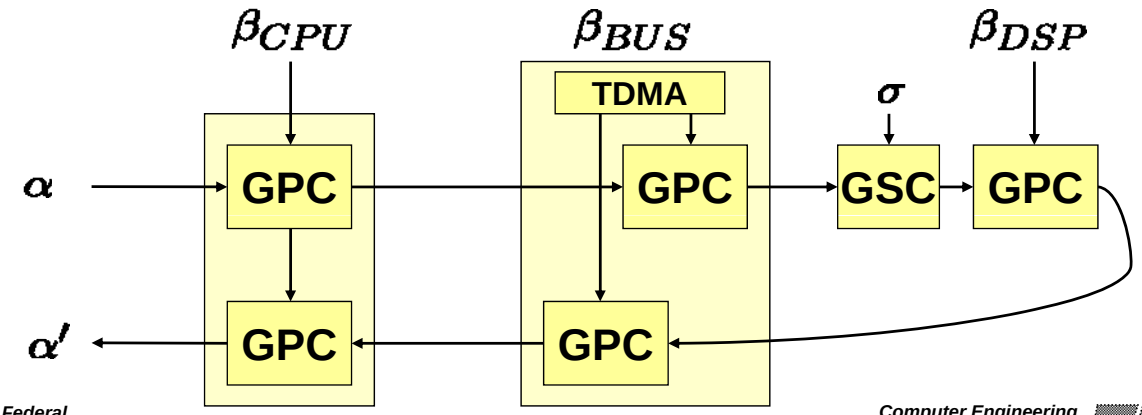
- ***Advantages***
  - More powerful abstraction than “classical” real-time analysis
  - Resources are first-class citizens of the method
  - *Allows composition in terms of (a) tasks, (b) streams, (c) resources, (d) sharing strategies.*
- ***Disadvantages***
  - Needs some effort to understand and implement
  - Extension to new arbitration schemes not always simple

Abstract Models for Performance Analysis

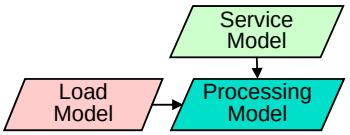


Modular System Composition



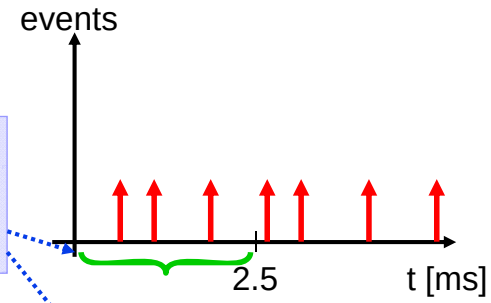


Load Model (Environment)



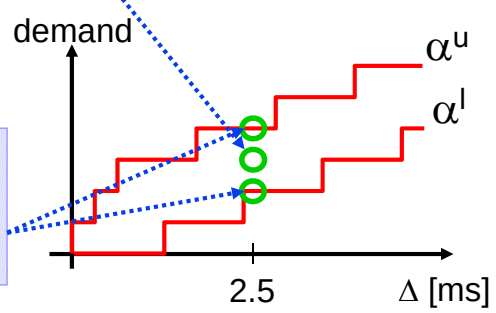
Event Stream

number of events in in t=[0 .. 2.5] ms



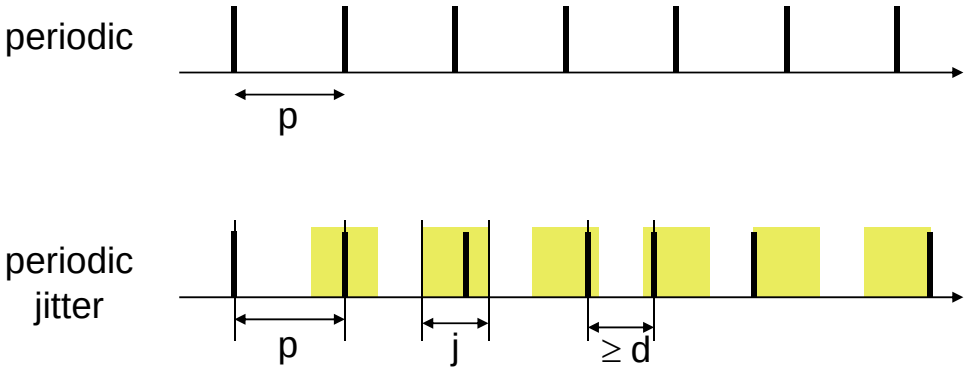
Arrival Curve α

maximum / minimum arriving demand in *any interval* of length 2.5 ms

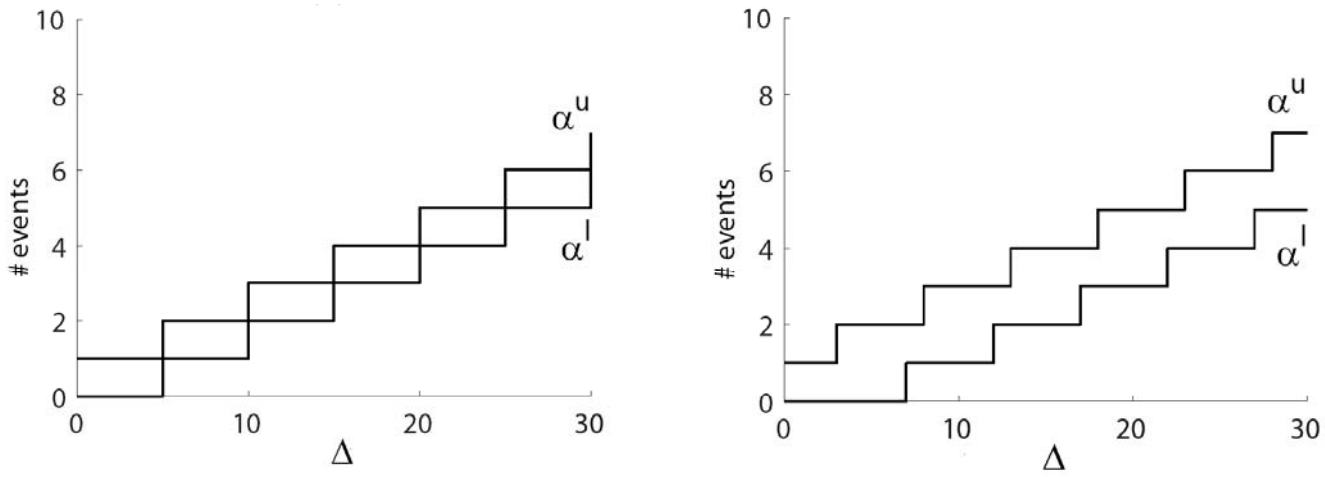


Example 1: Periodic with Jitter

- A ***common event pattern*** that is used in literature can be specified by the parameter triple (*p, j, d*), where *p* denotes the period, *j* the jitter, and *d* the minimum inter-arrival distance of events in the modeled stream.



Example 1: Periodic with Jitter

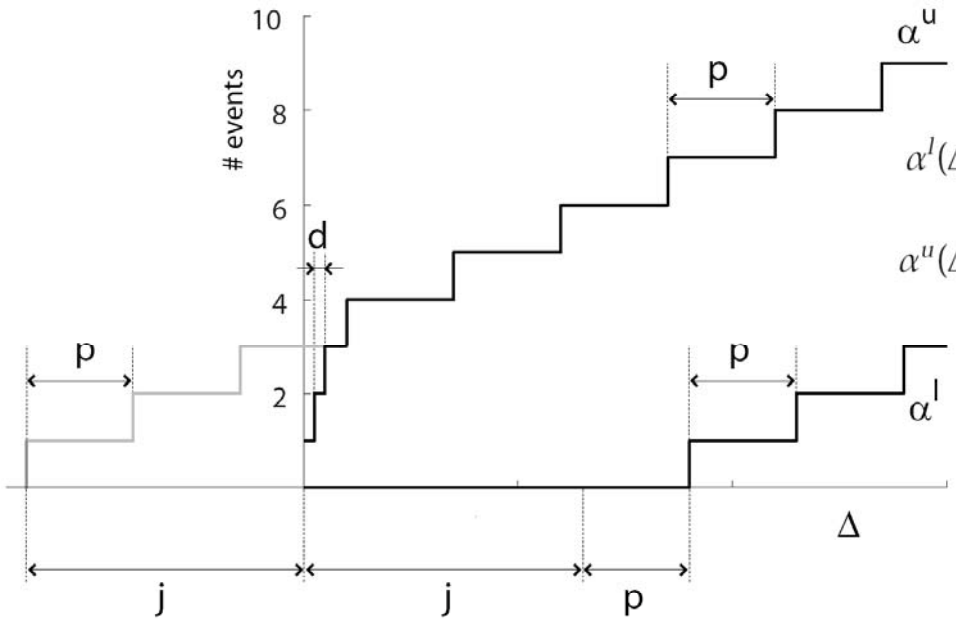


periodic

periodic with jitter

Example 1: Periodic with Jitter

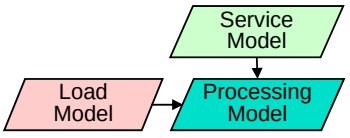
- ***Arrival curves***:

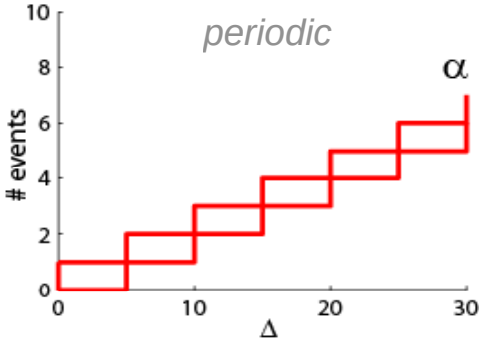


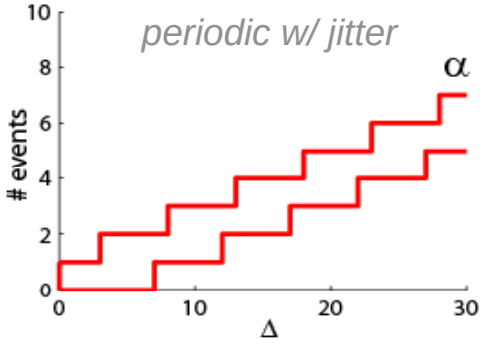
$$\alpha^l(\Delta) = \left\lfloor \frac{\Delta - j}{p} \right\rfloor$$

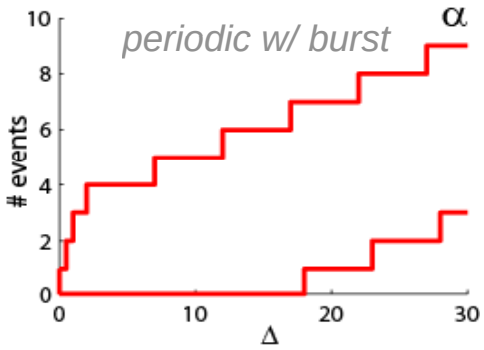
$$\alpha^u(\Delta) = \min\left\{ \left\lceil \frac{\Delta + j}{p} \right\rceil, \left\lceil \frac{\Delta}{d} \right\rceil \right\}$$

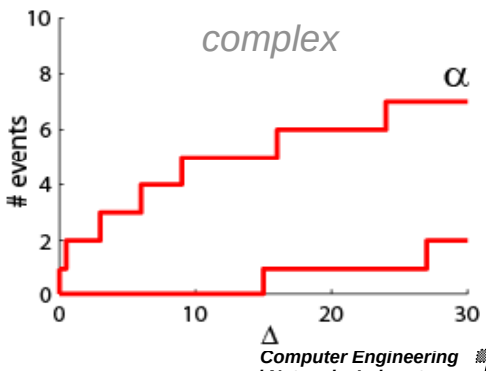
Load Model - Examples



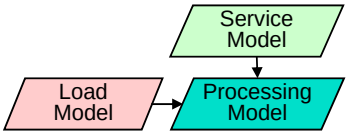






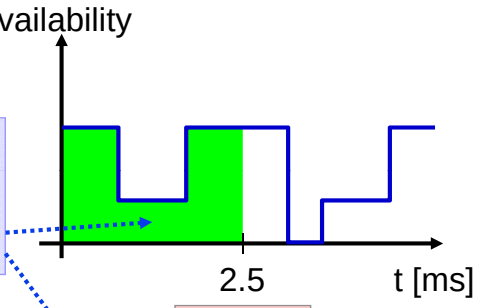


Service Model (Resources)



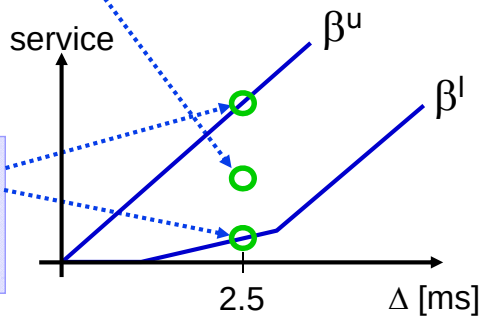
Resource Availability

available service in t=[0 .. 2.5] ms



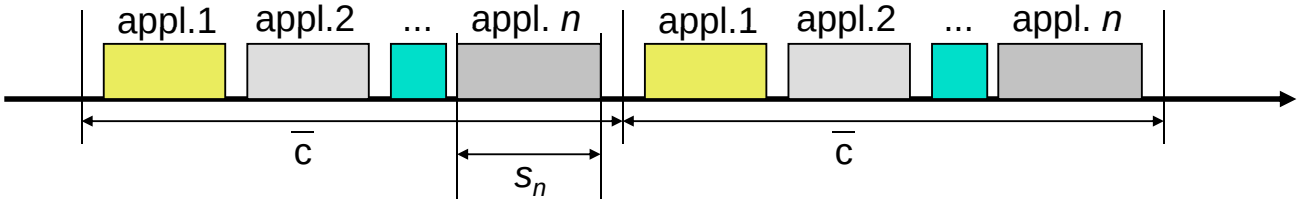
Service Curves [β^l, β^u]

maximum/minimum available service in *any interval* of length 2.5 ms



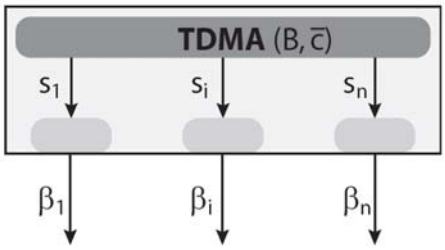
Example 2: TDMA Resource

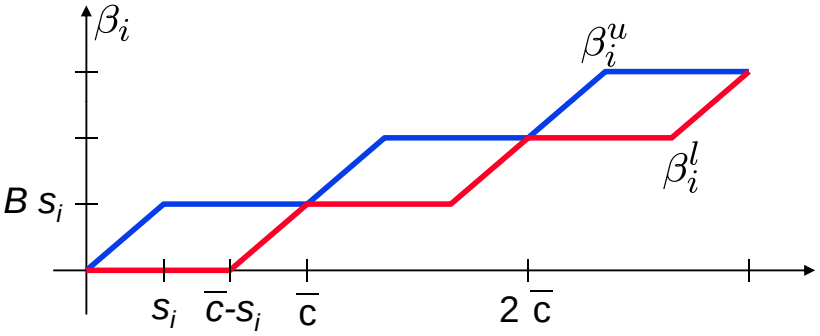
- Consider a real-time system consisting of *n* ***applications*** that are executed on a resource with bandwidth *B* that controls resource access using a ***TDMA policy***.
- Analogously, we could consider a distributed system with *n* ***communicating nodes***, that communicate via a shared bus with bandwidth *B*, with a bus arbitrator that implements a TDMA policy.
- ***TDMA policy***: In every TDMA cycle of length $\overline{c}$, one single resource slot of length s_i is assigned to application *i*.



Example 2: TDMA Resource

- ***Service curves*** available to the applications / node *i:*

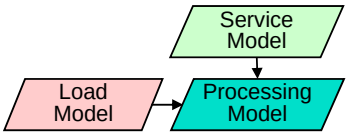


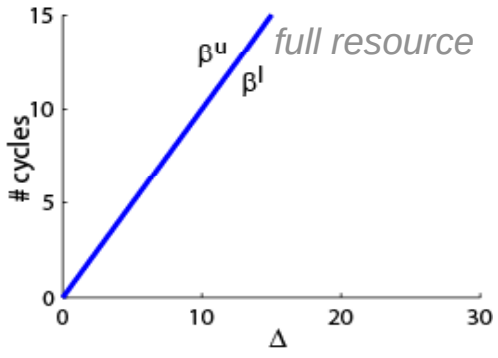


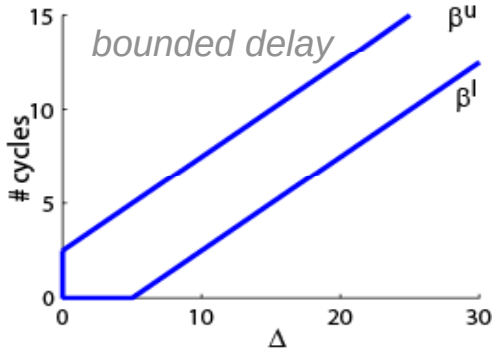
$$\beta_i^l(\Delta) = B \max\{\left\lfloor \frac{\Delta}{\bar{c}} \right\rfloor s_i, \Delta - \left\lceil \frac{\Delta}{\bar{c}} \right\rceil (\bar{c} - s_i)\}$$

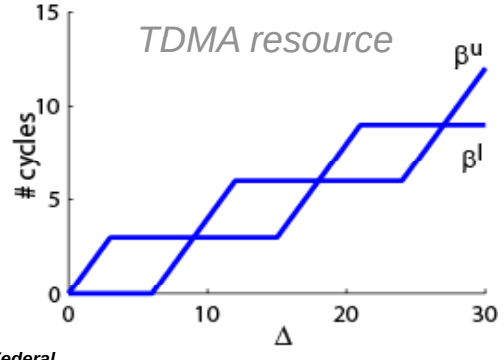
$$\beta_i^u(\Delta) = B \min\{\left\lceil \frac{\Delta}{\bar{c}} \right\rceil s_i, \Delta - \left\lfloor \frac{\Delta}{\bar{c}} \right\rfloor (\bar{c} - s_i)\}$$

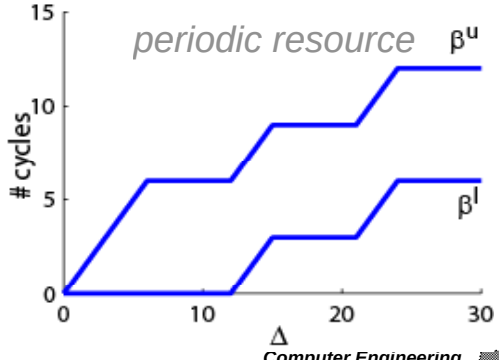
Service Model - Examples



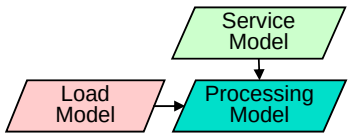






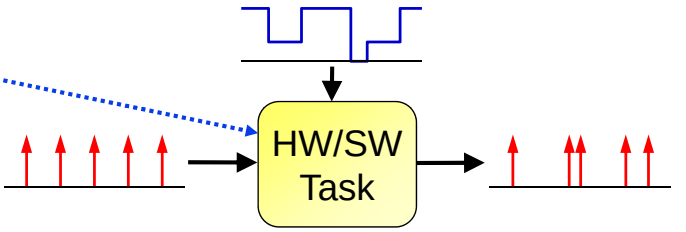


Processing Model (HW/SW)



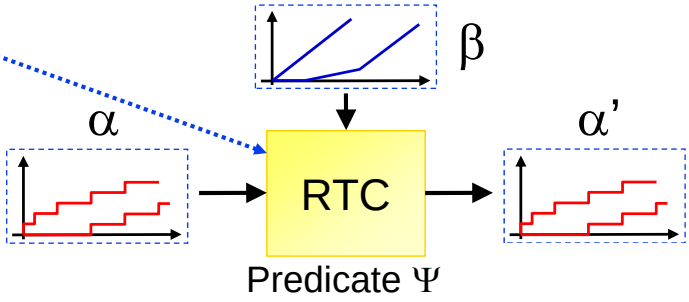
HW/SW Components

Processing semantics and functionality of HW/SW tasks



Abstract Components

$\alpha'(\Delta) = f(\alpha, \beta)$



Foundation

- Real-Time Calculus can be regarded as a ***worst-case/best-case variant of classical queuing theory***. It is a formal method for the analysis of distributed real-time embedded systems.
- ***Related Work***:
  - *Min-Plus Algebra*: F. Baccelli, G. Cohen, G. J. Olster, and J. P. Quadrat, Synchronization and Linearity --- An Algebra for Discrete Event Systems, Wiley, New York, 1992.
  - *Network Calculus*: J.-Y. Le Boudec and P. Thiran, Network Calculus - A Theory of Deterministic Queuing Systems for the Internet, Lecture Notes in Computer Science, vol. 2050, Springer Verlag, 2001.
  - *Adversarial Queuing Theory* [Andrews, Borodin, Kleinberg, Leighton, … 1996]

Comparison of Algebraic Structures

- ***Algebraic structure***
  - set of elements $\mathcal{S}$
  - one or more operators defined on elements of this set
- Algebraic structures ***with two operators*** $\boxplus$, $\boxdot$
  - plus-times: $(S, \boxplus, \boxdot) = (\mathbf{R}, +, \times)$
  - min-plus: $(S, \boxplus, \boxdot) = (\mathbf{R} \cup \{+\infty\}, \inf, +)$
- ***Infimum***:
  - The infimum of a subset of some set is the greatest element, not necessarily in the subset, that is less than or equal to all other elements of the subset.
  - $\inf\{[3,4]\} = 3, \quad \inf\{(3,4]\} = 3$
    $\min\{[3,4]\} = 3, \quad \min\{(3,4]\}$ not defined

Comparison of Algebraic Structures

- ***Properties*** of : $\boxdot$

  Closure of $\boxdot$: $a \boxdot b \in \mathcal{S}$
  Associativity of $\boxdot$: $a \boxdot (b \boxdot c) = (a \boxdot b) \boxdot c$
  Commutativity of $\boxdot$: $a \boxdot b = b \boxdot a$
  Existence of identity element for $\boxdot$: $\exists \nu : a \boxdot \nu = a$
  Existence of negative element for $\boxdot$: $\exists a^{-1} : a \boxdot a^{-1} = \nu$
  Identity element of $\boxplus$ absorbing for $\boxdot$: $a \boxdot \varepsilon = \varepsilon$
  Distributivity of $\boxdot$ w.r.t. $\boxplus$: $a \boxdot (b \boxplus c) = (a \boxdot b) \boxplus (a \boxdot c)$

- ***Example***:
  - plus-times: $a \times (b + c) = a \times b + a \times c$
  - min-plus: $a + \inf\{b, c\} = \inf\{a + b, a + c\}$

Comparison of Algebraic Structures

- ***Properties*** of $: \boxplus$

  Closure of $\boxplus$: $a \boxplus b \in \mathcal{S}$
  Associativity of $\boxplus$: : $a \boxplus (b \boxplus c) = (a \boxplus b) \boxplus c$
  Commutativity of $\boxplus$: $a \boxplus b = b \boxplus a$
  Existence of identity element for $\boxplus$: $\exists \varepsilon : a \boxplus \varepsilon = a$

- ***Differences*** $\boxplus$:
  - *plus-times*: Existence of a negative element for $\boxplus$:
    $$\exists (-a) : a \boxplus (-a) = \varepsilon$$
  - *min-plus*: Idempotency of $\boxplus$: $a \boxplus a = a$

Some Definitions and Relations

- $f \otimes g$ is called ***min-plus convolution***
  $$(f \otimes g)(t) = \inf_{0 \le u \le t} \{f(t-u) + g(u)\}$$
- $f \oslash g$ is called ***min-plus de-convolution***
  $$(f \oslash g)(t) = \sup_{u \ge 0} \{f(t+u) - g(u)\}$$
- For ***max-plus convolution and de-convolution***:
  $$(f \overline{\otimes} g)(t) = \sup_{0 \le u \le t} \{f(t-u) + g(u)\}$$
  $$(f \overline{\oslash} g)(t) = \inf_{u \ge 0} \{f(t+u) - g(u)\}$$
- Relation between ***convolution and deconvolution***
  $$f \le g \otimes h \Leftrightarrow f \oslash h \le g$$

Rules

- ***Rule 1 (Closure of $\otimes$)*** $(f \otimes g) \in \mathcal{F}$.
- ***Rule 2 (Associativity of $\otimes$)*** $(f \otimes g) \otimes h = f \otimes (g \otimes h)$.
- ***Rule 3 (The zero element for $\wedge$ is absorbing for $\otimes$)*** *The zero element for $\wedge$ belonging to $\mathcal{F}$ is the function ε, defined as $\varepsilon(t) = +\infty$ for all $t \geq 0$ and $\varepsilon(t) = 0$ for all $t < 0$. One has $f \otimes \varepsilon = \varepsilon$.*
- ***Rule 4 (Existence of a neutral element for $\otimes$)*** *The neutral element is δ_0, as $f \otimes \delta_0 = f$.*
- ***Rule 5 (Commutativity of $\otimes$)*** $f \otimes g = g \otimes f$.
- ***Rule 6 (Distributivity of $\otimes$ with respect to $\wedge$)*** $(f \wedge g) \otimes h = (f \otimes h) \wedge (g \otimes h)$.
- ***Rule 7 (Addition of a constant)*** *For any $K \in \mathbb{R}^+$, $(f + K) \otimes g = (f \otimes g) + K$.*
- ***Rule 10 (Isotonicity)*** *If $f \leq g$ and $f' \leq g'$ then $f \otimes f' \leq g \otimes g'$.*
- ***Rule 11 (Isotonicity of $\oslash$)*** *If $f \leq g$, then $f \oslash h \leq g \oslash h$ and $h \oslash f \geq h \oslash g$.*
- ***Rule 12 (Composition of $\oslash$)*** $(f \oslash g) \oslash h = f \oslash (g \otimes h)$.
- ***Rule 13 (Composition of $\oslash$ and $\otimes$)*** $(f \otimes g) \oslash g \leq f \otimes (g \oslash g)$.
- ***Rule 14 (Duality between $\oslash$ and $\otimes$)*** *$f \oslash g \leq h$ if and only if $f \leq g \otimes h$.*
- ***Rule 15 (Self-deconvolution)*** *$(f \oslash f)$ is a sub-additive function of $\mathcal{F}$ such that $(f \oslash f)(0) = 0$.*

Arrival and Service Curve

- The arrival and service curves provide bounds on event and resource functions as follows:

$$\alpha^l(t-s) \leq R(t) - R(s) \leq \alpha^u(t-s) \quad \forall s \leq t$$

$$\beta^l(t-s) \leq C(t) - C(s) \leq \beta^u(t-s) \quad \forall s \leq t$$

- We can determine valid variability curves from cumulative functions as follows:

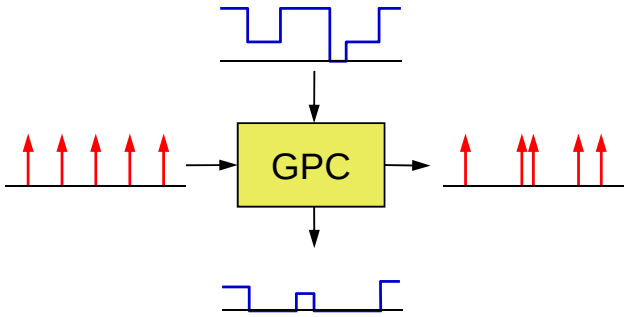
$$\alpha^u = R \oslash R; \quad \alpha^l = R \overline{\oslash} R; \quad \beta^u = C \oslash C; \quad \beta^l = C \overline{\oslash} C$$

- One proof:

$$\alpha^u = R \oslash R \Rightarrow \alpha^u(\Delta) = \sup_{u \geq 0} \{R(\Delta + u) - R(u)\} \Rightarrow$$

$$\alpha^u(\Delta) = \sup_{s \geq 0} \{R(\Delta + s) - R(s)\} \Rightarrow \alpha^u(t-s) \geq R(t) - R(s) \ \ \forall t \geq s$$

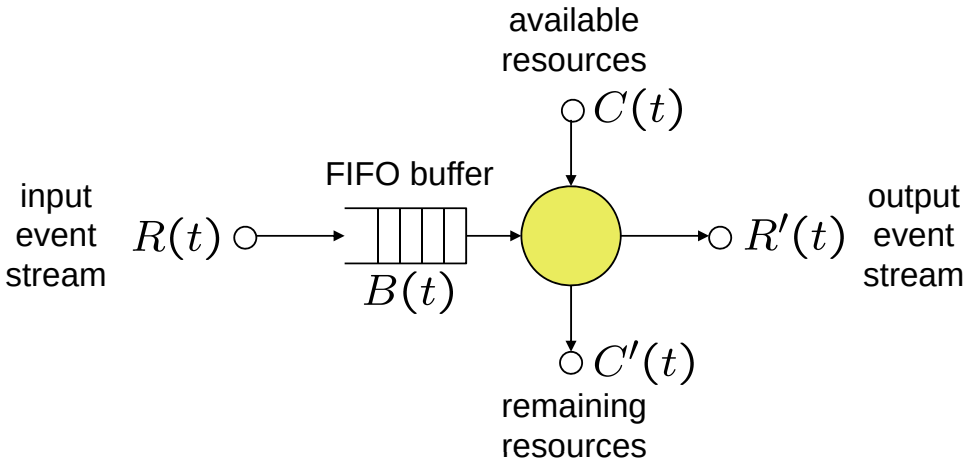
Greedy Processing Component



Behavioral Description

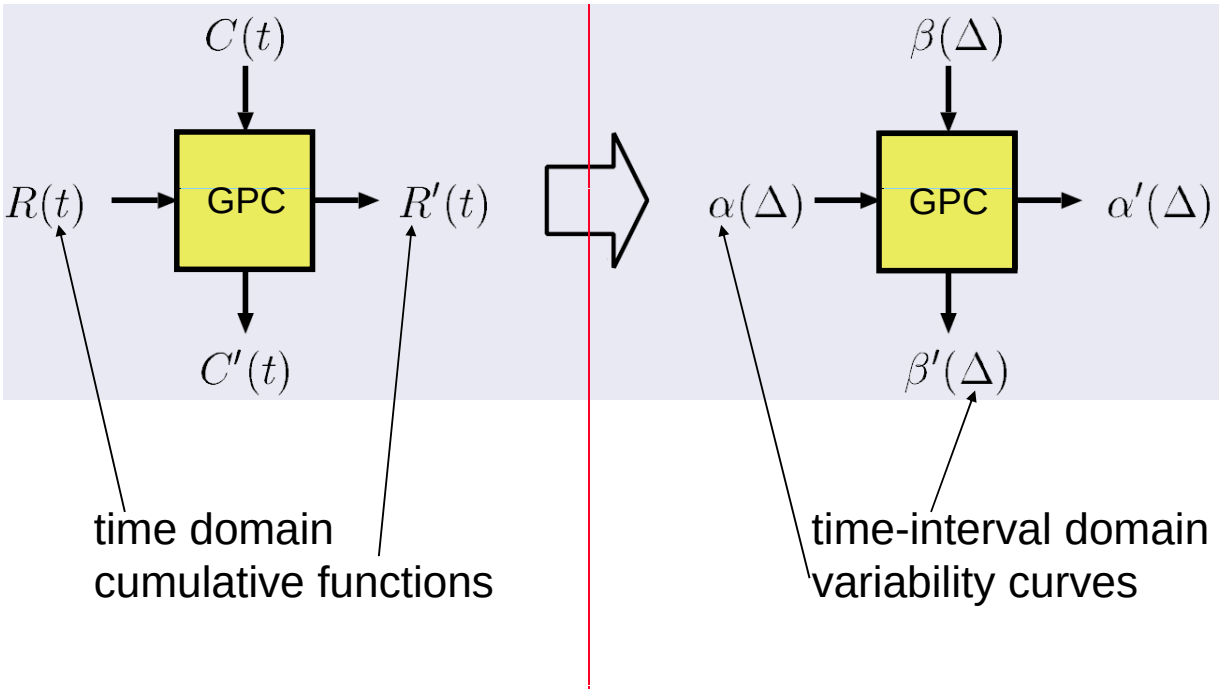
- Component is triggered by incoming events.
- A fully preemptable task is instantiated at every event arrival to process the incoming event.
- Active tasks are processed in a greedy fashion in FIFO order.
- Processing is restricted by the availability of resources.

Greedy Processing Component (GPC)



- ***Examples:***
  - computation (event – task instance, resource – computing resource [tasks/second])
  - communication (event – data packet, resource – bandwidth [packets/second])

Abstraction



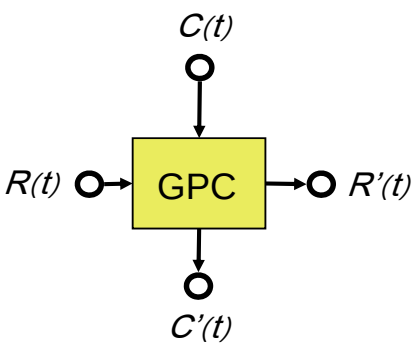
Greedy Processing Component (GPC)

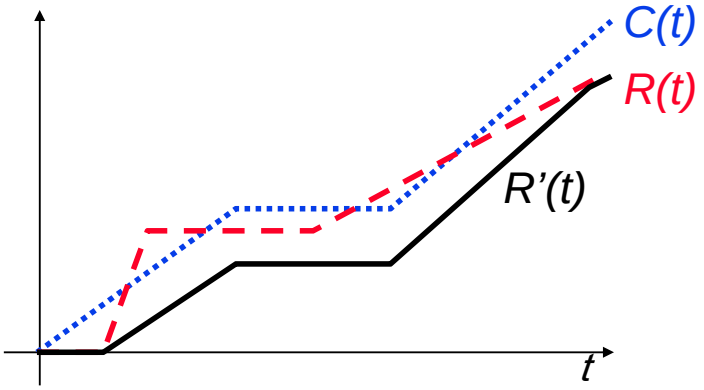
If the resource and event streams describe available and requested units of processing or communication, then

$$C(t) = C'(t) + R'(t)$$
$$B(t) = R(t) - R'(t)$$

Conservation Laws

$$R'(t) = \inf_{0 \le u \le t} \{R(u) + C(t) - C(u)\}$$

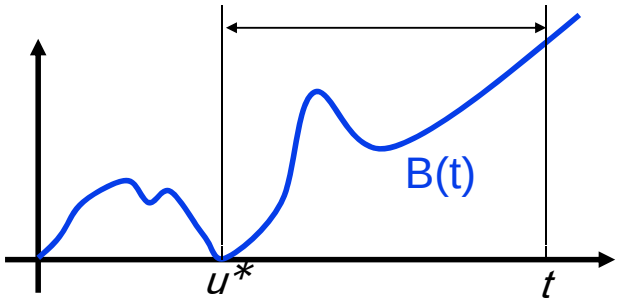




Greedy Processing

- For all times $u \leq t$ we have $R'(u) \leq R(u)$ (conservation law).
- We also have $R'(t) \leq R'(u)+C(t)-C(u)$ as the output can not be larger than the available resources.
- Combining both statements yields $R'(t) \leq R(u) + C(t) - C(u)$.
- Let us suppose that u^* is the last time before t with an empty buffer. We have $R(u^*) = R'(u^*)$ at u^* and also $R'(t) = R'(u^*) + C(t) - C(u^*)$ as all available resources are used to produce output. Therefore, $R'(t) = R(u^*) + C(t) - C(u^*)$.
- As a result, we obtain

$$R'(t) = \inf_{0 \leq u \leq t} \{R(u) + C(t) - C(u)\}$$



The Most Simple Relations

- The *output stream* of a component satisfies:

$$R'(t) \geq (R \otimes \beta^l)(t)$$

- The *output upper arrival curve* of a component satisfies:

$$\alpha^{u\prime} = (\alpha^u \oslash \beta^l)$$

- The *remaining lower service curve* of a component satisfies:

$$\beta^{l\prime}(\Delta) = \sup_{0 \leq \lambda \leq \Delta} (\beta^l(\lambda) - \alpha^u(\lambda))$$

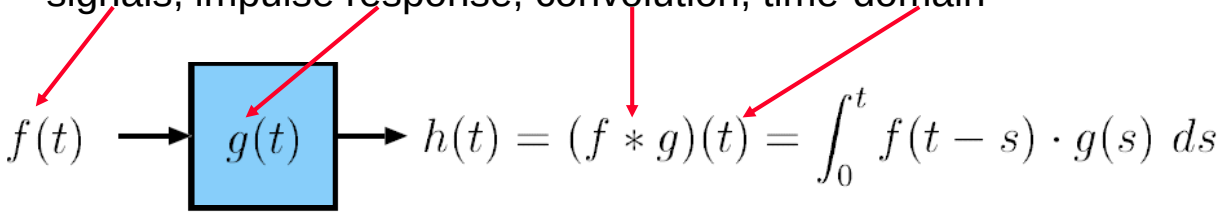
Two Sample Proofs

$$
\begin{aligned}
R'(t) &= \inf_{0 \le u \le t} \{R(u) + C(t) - C(u)\} \\
&\ge \inf_{0 \le u \le t} \{R(u) + \beta^l(t-u)\} \\
&= (R \otimes \beta^l)(t)
\end{aligned}
$$

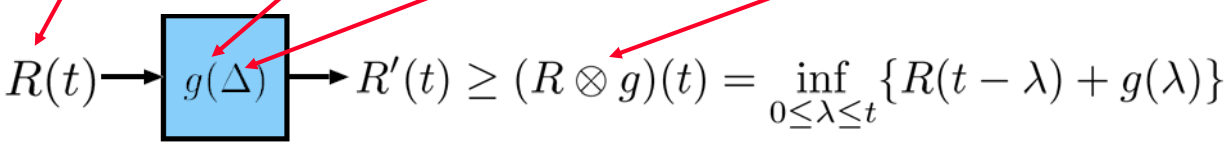
$$
\begin{aligned}
C'(t) - C'(s) &= \sup_{0 \le a \le t} \{C(a) - R(a)\} - \sup_{0 \le b \le s} \{C(b) - R(b)\} = \\
&= \inf_{0 \le b \le s} \{ \sup_{0 \le a \le t} \{(C(a) - C(b)) - (R(a) - R(b))\}\} \\
&= \inf_{0 \le b \le s} \{ \sup_{0 \le a-b \le t-b} \{(C(a) - C(b)) - (R(a) - R(b))\}\} \\
&\ge \inf_{0 \le b \le s} \{ \sup_{0 \le \lambda \le t-b} \{\beta^l(\lambda) - \alpha^u(\lambda)\}\} \ge \sup_{0 \le \lambda \le t-s} \{\beta^l(\lambda) - \alpha^u(\lambda)\}
\end{aligned}
$$

Comparison of System Theories

- ***Plus-times system theory***
  - signals, impulse response, convolution, time-domain



- ***Min-plus system theory***
  - streams, variability curves, time-interval domain, convolution

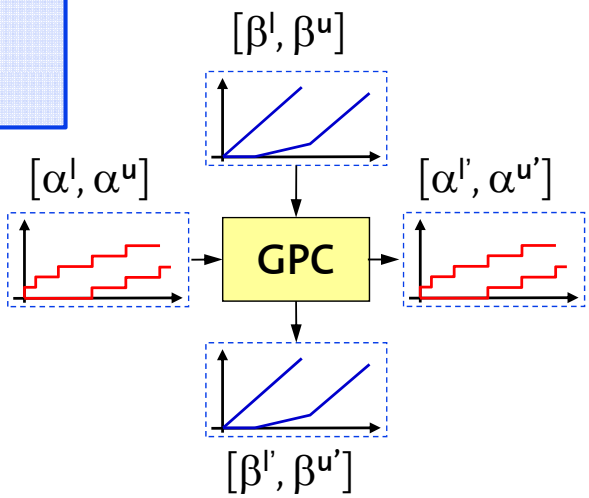


Tighter Bounds

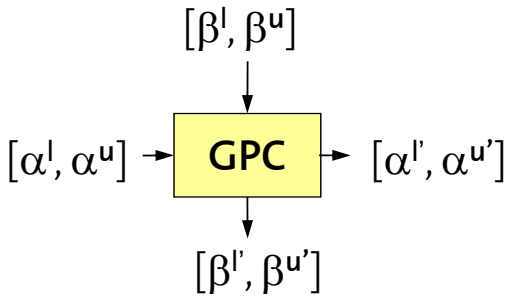
The greedy processing component transforms the variability curves as follows:

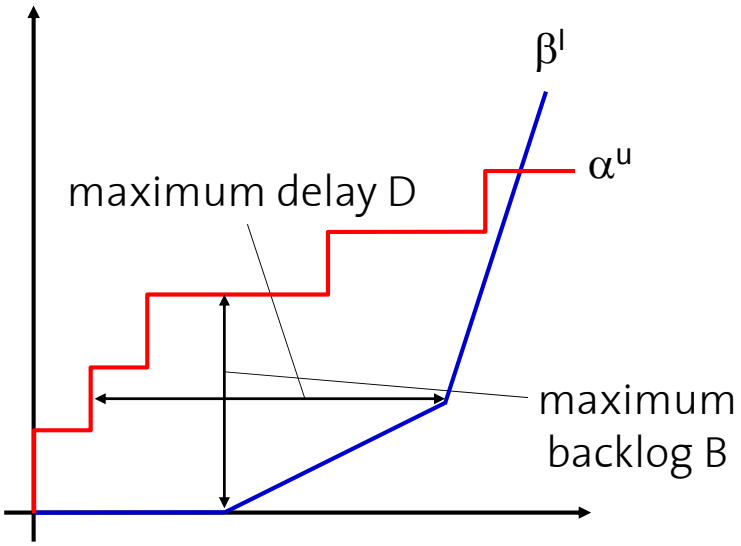
$$\alpha^{u\prime} = [(\alpha^u \otimes \beta^u) \oslash \beta^l] \wedge \beta^u$$
$$\alpha^{l\prime} = [(\alpha^l \oslash \beta^u) \otimes \beta^l] \wedge \beta^l$$
$$\beta^{u\prime} = (\beta^u - \alpha^l) \overline{\oslash} 0$$
$$\beta^{l\prime} = (\beta^l - \alpha^u) \overline{\otimes} 0$$

$$(f \otimes g)(t) = \inf_{0 \le u \le t} \{f(t-u) + g(u)\}$$
$$(f \oslash g)(t) = \sup_{u \ge 0} \{f(t+u) - g(u)\}$$
$$(f \overline{\otimes} g)(t) = \sup_{0 \le u \le t} \{f(t-u) + g(u)\}$$
$$(f \overline{\oslash} g)(t) = \inf_{u \ge 0} \{f(t+u) - g(u)\}$$



Delay and Backlog



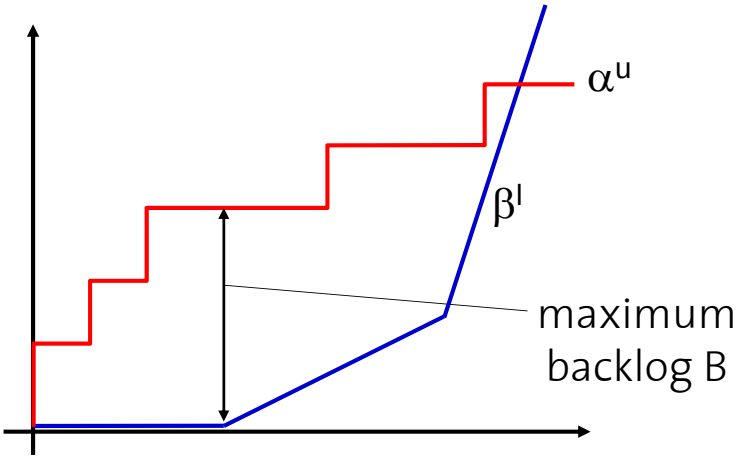


$$B = \sup_{t \ge 0} \{R(t) - R'(t)\} \le \sup_{\lambda \ge 0} \left\{\alpha^u(\lambda) - \beta^l(\lambda)\right\}$$

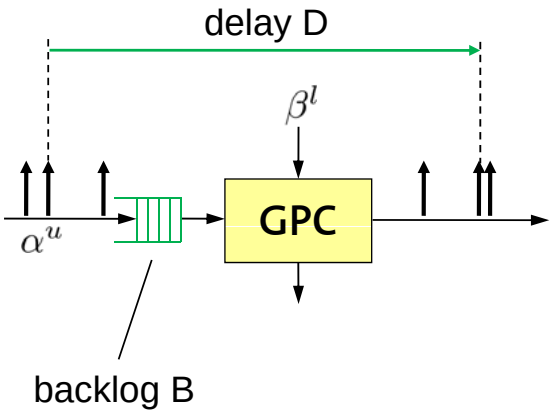
$$D = \sup_{t \ge 0} \{\inf \{\tau \ge 0 : R(t) \le R'(t+\tau))\}\}$$
$$= \sup_{\Delta \ge 0} \left\{\inf \left\{\tau \ge 0 : \alpha^u(\Delta) \le \beta^l(\Delta + \tau)\right\}\right\}$$

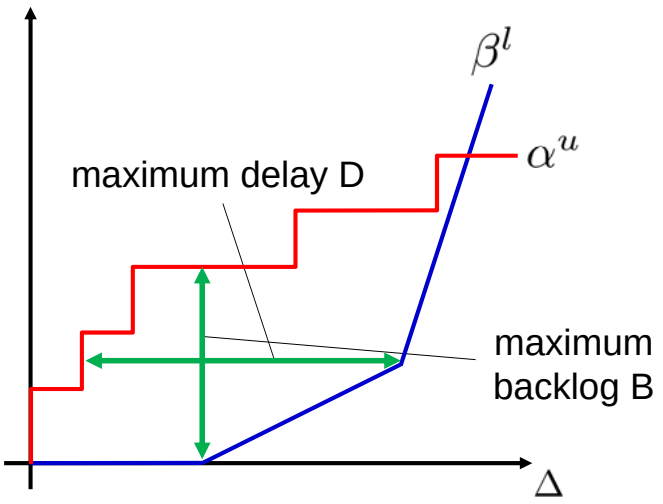
Proof of Backlog Bound

$$B(t) = R(t) - R'(t) = R(t) - \inf_{0 \le u \le t} \{R(u) + C(t) - C(u)\}$$
$$= \sup_{0 \le u \le t} \{(R(t) - R(u)) - (C(t) - C(u))\}$$
$$\le \sup_{0 \le u \le t} \{\alpha^u(t-u) - \beta^l(t-u)\}$$
$$\le \sup_{0 \le \lambda} \{\alpha^u(\lambda) - \beta^l(\lambda)\}$$

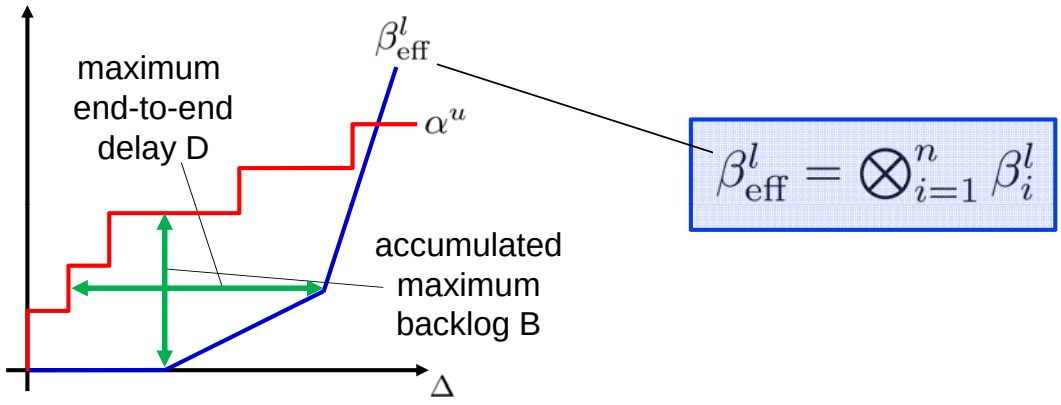


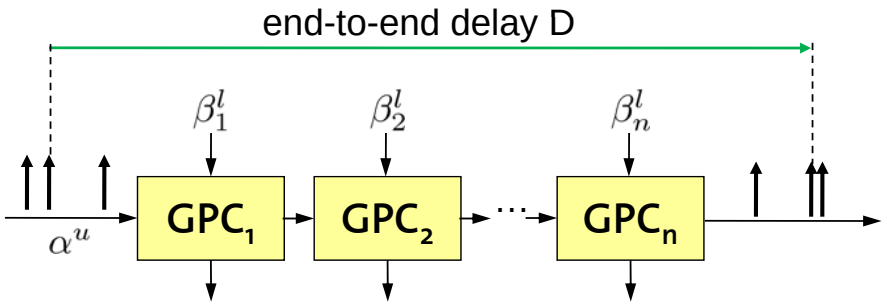
Delay and Backlog



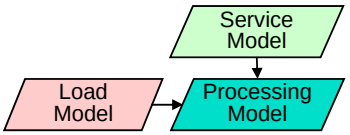


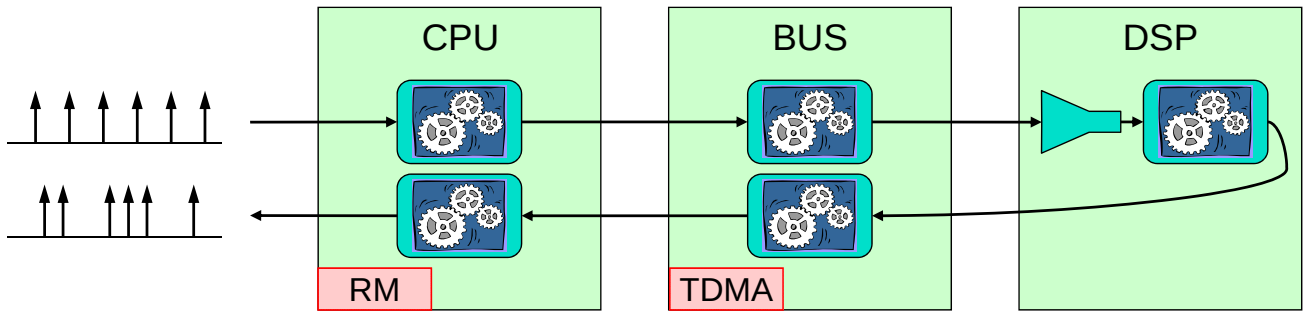
Celebrated Result on Delay and Backlog





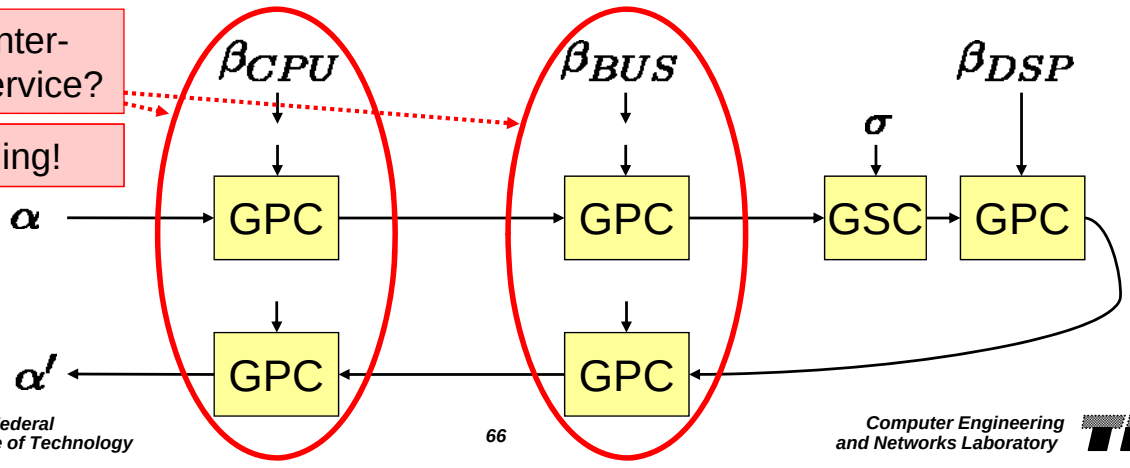
System Composition



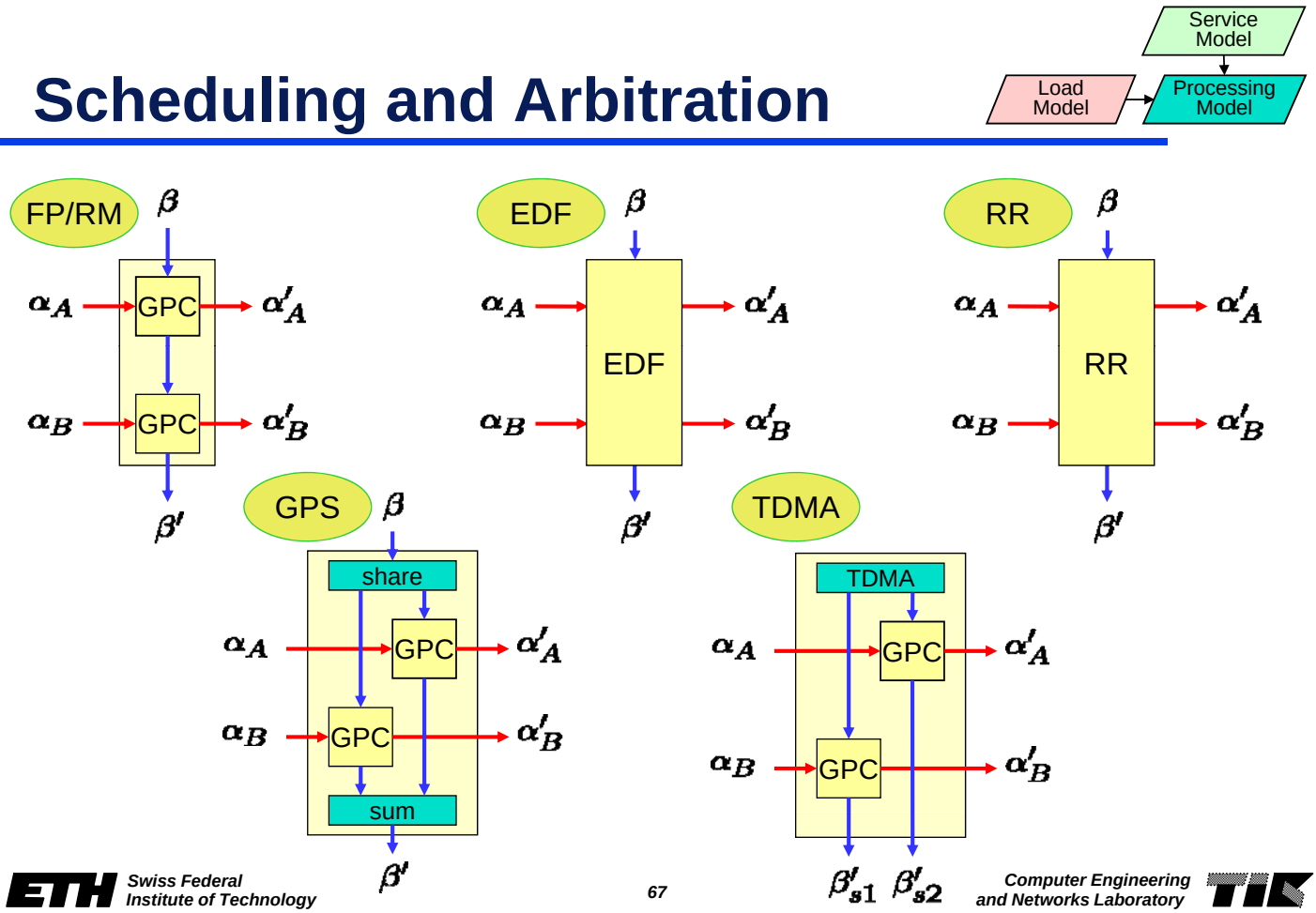


How to inter-connect service?

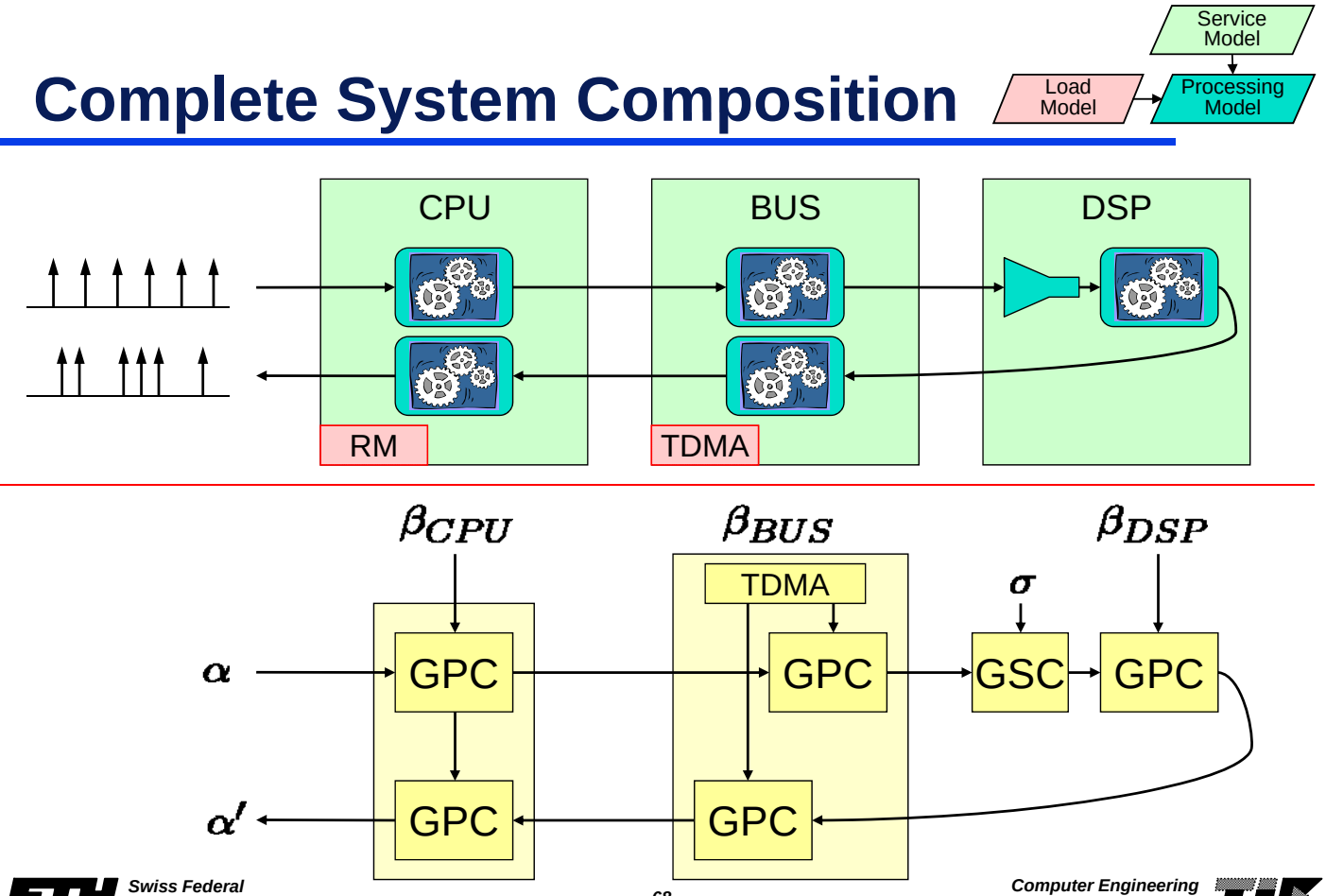
Scheduling!



Scheduling and Arbitration



Complete System Composition



Events and Workloads

(Event-Based Arrival Curves) *An event-based arrival curve* $\bar{\alpha}(\Delta) = [\bar{\alpha}^u(\Delta), \bar{\alpha}^l(\Delta)]$ *models an event stream, where* $\bar{\alpha}^u(\Delta)$ *and* $\bar{\alpha}^l(\Delta)$ *provide an upper and a lower bound on the number of events that arrive in any time interval* Δ*, respectively.*

(Event-Based Service Curves) *An event-based service curve* $\bar{\beta}(\Delta) = [\bar{\beta}^u(\Delta), \bar{\beta}^l(\Delta)]$ *models a resource, where* $\bar{\beta}^u(\Delta)$ *and* $\bar{\beta}^l(\Delta)$ *provide an upper and a lower bound on the number of events that can be processed in any time interval* Δ*, respectively.*

event

(Resource-Based Arrival Curves) *A resource-based arrival curve* $\alpha(\Delta) = [\alpha^u(\Delta), \alpha^l(\Delta)]$ *models an event stream, where* $\alpha^u(\Delta)$ *and* $\alpha^l(\Delta)$ *provide an upper and a lower bound on the resource demand imposed by the event stream in any time interval* Δ*, respectively.*

(Resource-Based Service Curves) *A resource-based service curve* $\beta(\Delta) = [\beta^u(\Delta), \beta^l(\Delta)]$ *models a resource, where* $\beta^u(\Delta)$ *and* $\beta^l(\Delta)$ *provide an upper and a lower bound on the available resource supply in any time interval* Δ*, respectively.*

workload

Events and Workloads

- Simple case of a constant workload d per event:

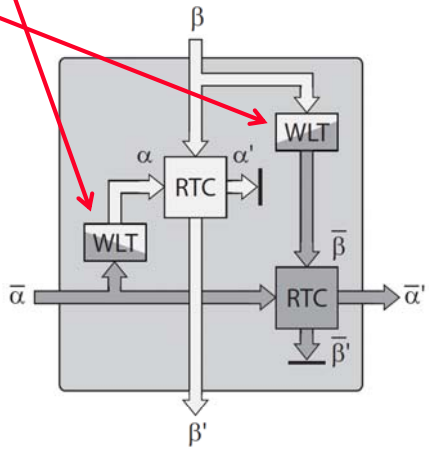
$$\alpha^u(\Delta) = \bar{\alpha}^u \cdot d \qquad \alpha^l(\Delta) = \bar{\alpha}^l \cdot d$$

$$\bar{\alpha}^u(\Delta) = \lceil \alpha^u / d \rceil \qquad \bar{\alpha}^l(\Delta) = \lfloor \alpha^l / d \rfloor$$

$$\beta^u(\Delta) = \bar{\beta}^u \cdot d \qquad \beta^l(\Delta) = \bar{\beta}^l \cdot d$$

$$\bar{\beta}^u(\Delta) = \lceil \beta^u / d \rceil \qquad \bar{\beta}^l(\Delta) = \lfloor \beta^l / d \rfloor$$

- Use in the simple GPC component:



Contents

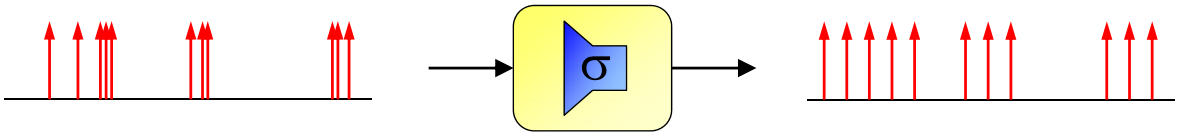
- Drivers
- Compositional Analysis
  - Overview
  - Real-Time Calculus
- ***Examples***
  - Shapers
  - Artificial Example
  - Shared Resources in Multicore Systems
- Extensions
- Comparison
- Challenges

Compositional Analysis Examples

- Shapers-

Greedy Traffic Shaper

- Access Shaper
  - delays access requests such that the resulting access pattern conforms to a given specification
- Greedy Access Shaper
  - no access request gets delayed any longer than necessary

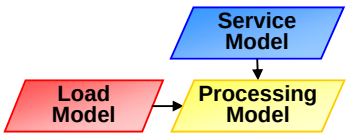


Why Access Shaping?

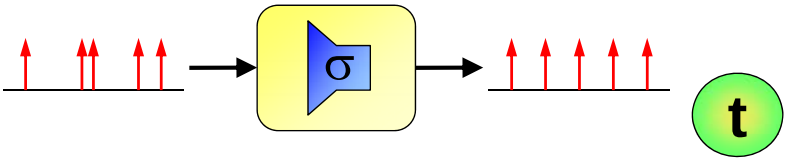
- Internal Re-Shaping
  - Reduces global buffer requirements
  - Reduces end-to-end delays
- External Input-Shaping
  - Ensures specification conformant system inputs

How to model and analyze greedy shapers?

Modeling of Greedy Shapers



Greedy Shaper

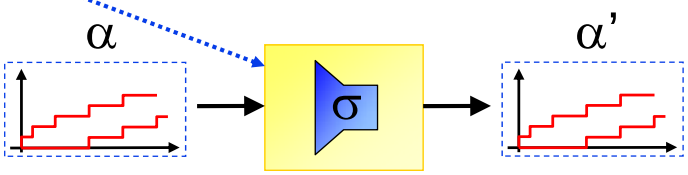


Abstract Greedy Shaper

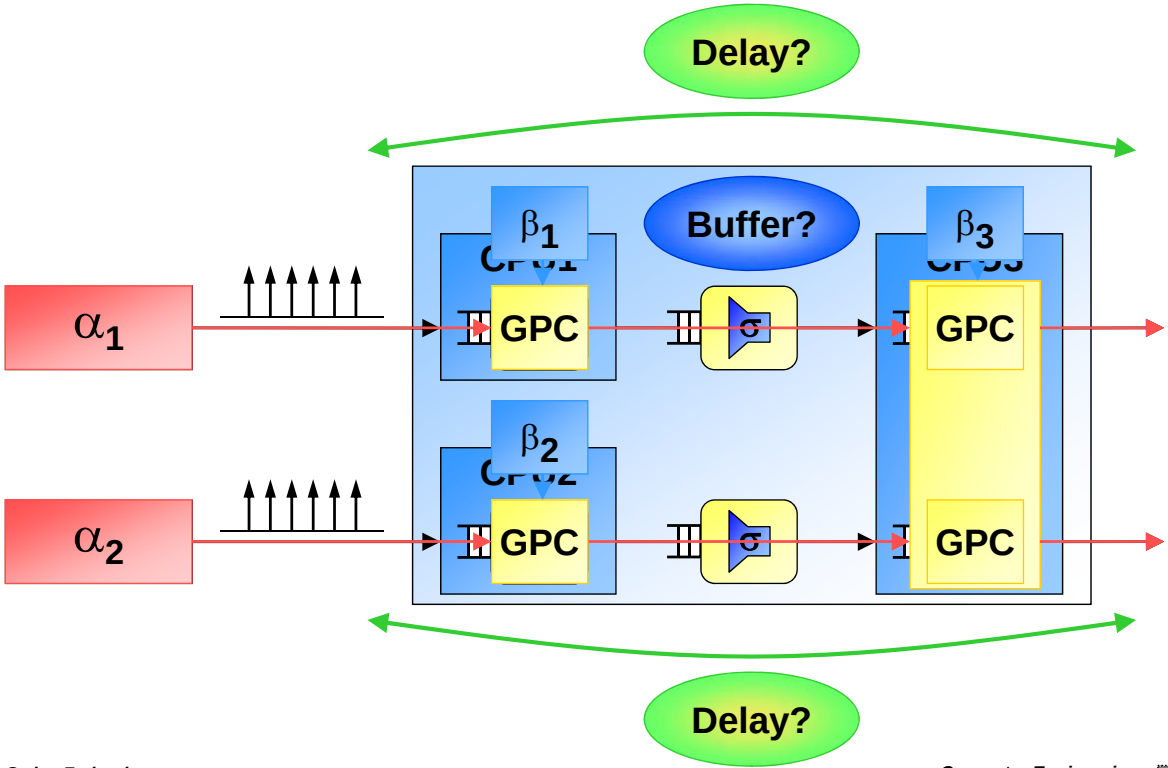
$$\alpha'(\Delta) = f(\alpha, \sigma)$$

$$\alpha'^{u} = \alpha^{u} \otimes \sigma$$
$$\alpha'^{l} = \alpha^{l} \otimes (\sigma \oslash \sigma)$$

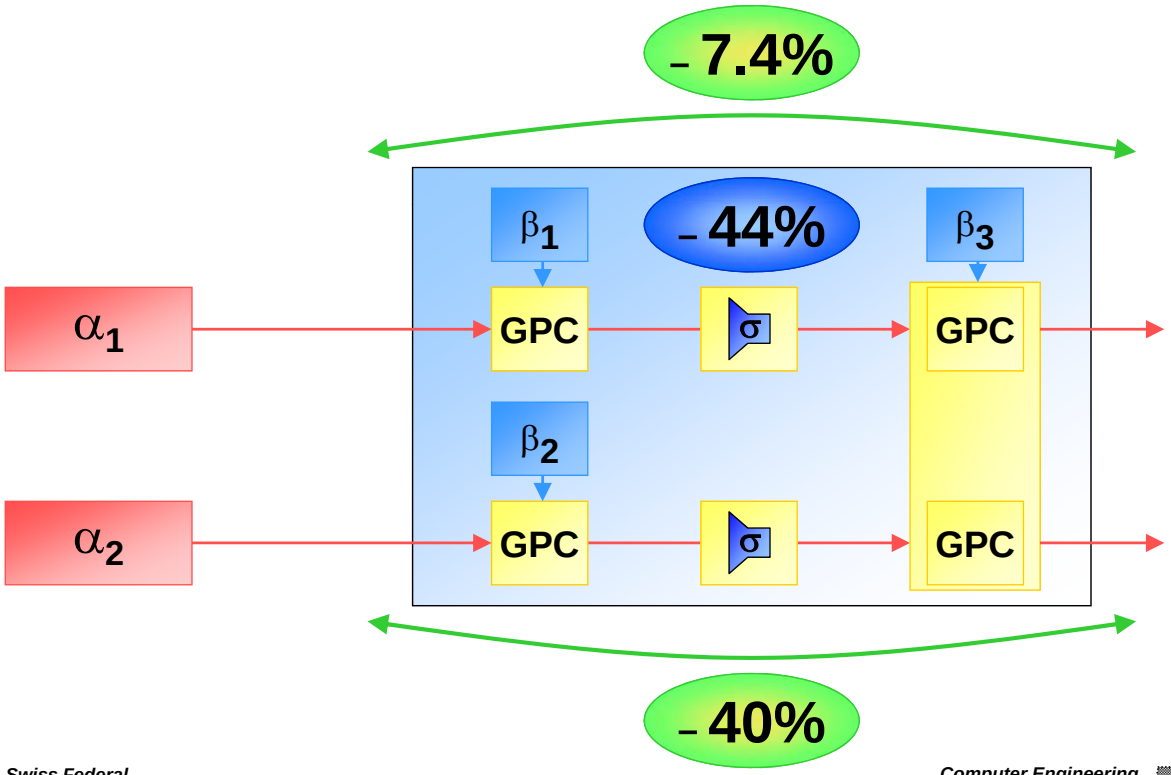
$$(f \otimes g)(\Delta) = \inf_{0 \le \lambda \le \Delta} \{f(\Delta - \lambda) + g(\lambda)\}$$
$$(f \oslash g)(\Delta) = \inf_{\lambda \ge 0} \{f(\Delta + \lambda) - g(\lambda)\}$$



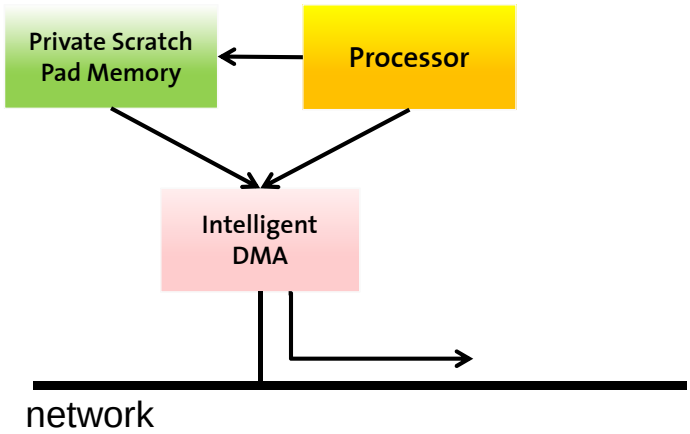
Internal Re-Shaping



Internal Re-Shaping



Model for resource sharing



Example - Communication

1. Processor writes data to private scratch pad memory and informs Intelligent DMA (iDMA) about where to send the data to
2. Processor continues execution while iDMA tries to send the data along to its destination

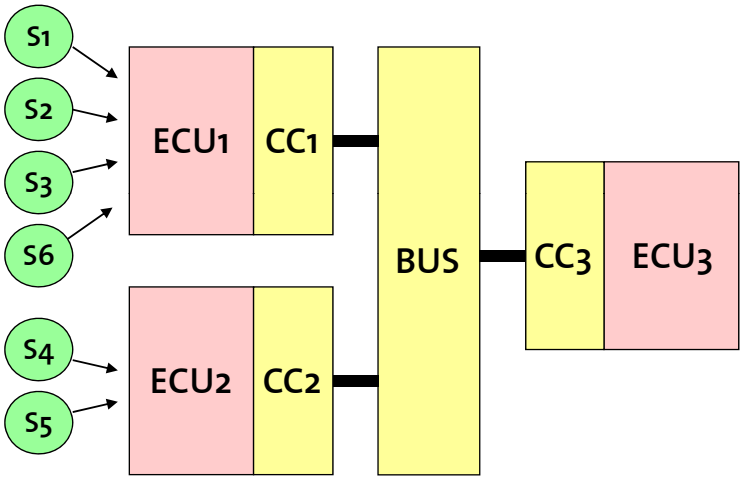
Intelligent DMA – What intelligence?

- Real Time systems need *guarantees* – in processing time, resource access (bus, memory)
- Communication a big challenge in providing such guarantees
- iDMA is a good opportunity to reclaim ground
- Intelligently guarantee a promised bandwidth to each processor by using the ideas of
  - (real-time) servers
  - isolation (remove interference between applications)
  - traffic shapers

Compositional Analysis Examples

- Artificial Example -

Case Study



6 Real-Time Input Streams
- with jitter
- with bursts
- deadline > period

3 ECU's with own CC's

13 Tasks & 7 Messages
- with different WCED

2 Scheduling Policies
- Earliest Deadline First (ECU's)
- Fixed Priority (ECU's & CC's)

Hierarchical Scheduling
- Static & Dynamic Polling Servers

Bus with TDMA
- 4 time slots with different lengths (#1,#3 for CC1, #2 for CC3, #4 for CC3)

Total Utilization:
- ECU1 59 %
- ECU2 87 %
- ECU3 67 %
- BUS 56 %

Specification Data

Stream	(p,j,d) [ms]	D [s]	Task Chain
S1	(1000, 2000, 25)	8.0	T1.1 → C1.1 → T1.2 → C1.2 → T1.3
S2	(400, 1500, 50)	1.8	T2.1 → C2.1 → T2.2
S3	(600, 0, -)	6.0	T3.1 → C3.1 → T3.2 → C3.2 → T3.3
S4	(20, 5, -)	0.5	T4.1 → C4.1 → T4.2
S5	(30, 0, -)	0.7	T4.1 → C4.1 → T4.2
S6	(1500, 4000, 100)	3.0	T6.1

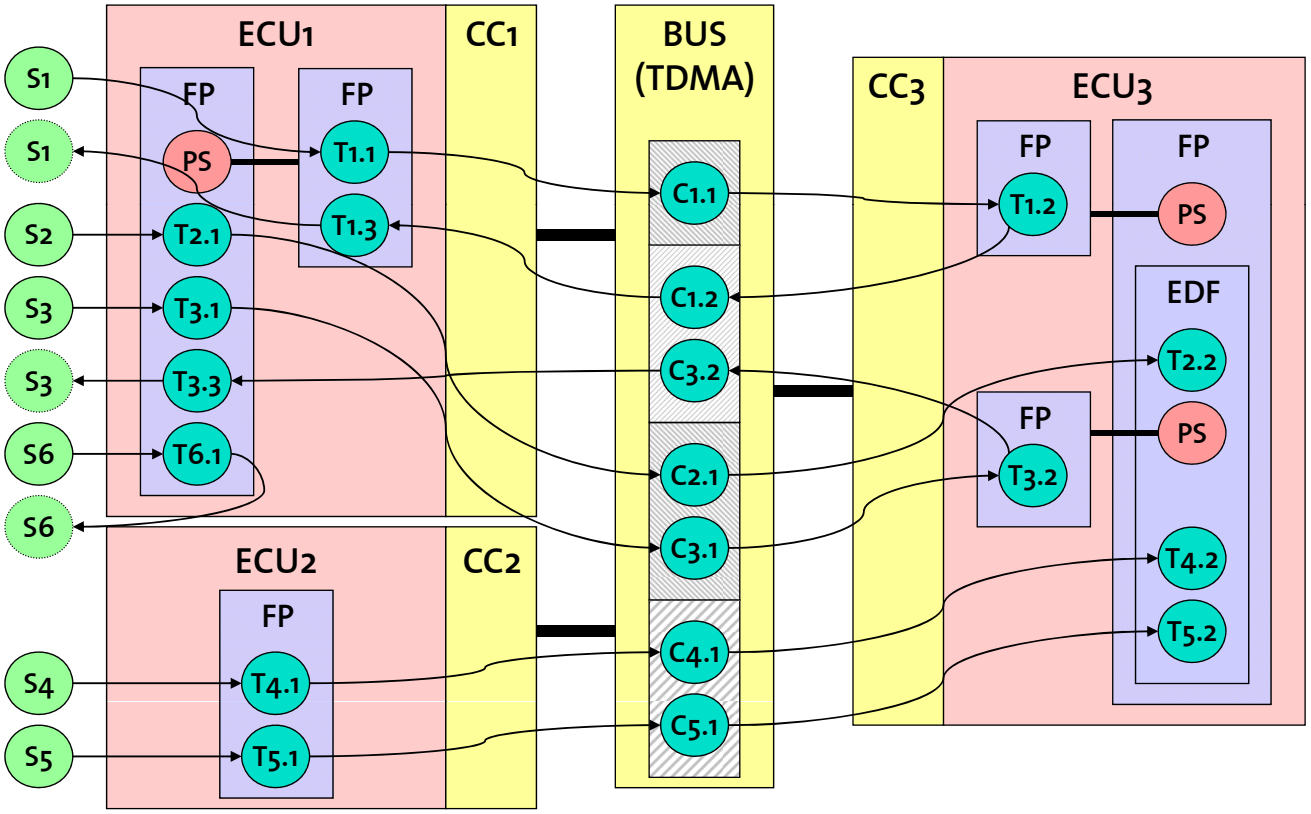
Task	e
T1.1	200
T1.2	300
T1.3	30
T2.1	75
T2.2	25
T3.1	60
T3.2	60
T3.3	40
T4.1	12
T4.2	2
T5.1	8
T5.2	3
T6.1	100

Message	e
C1.1	100
C1.2	80
C2.1	40
C3.1	25
C3.2	10
C4.1	3
C5.1	2

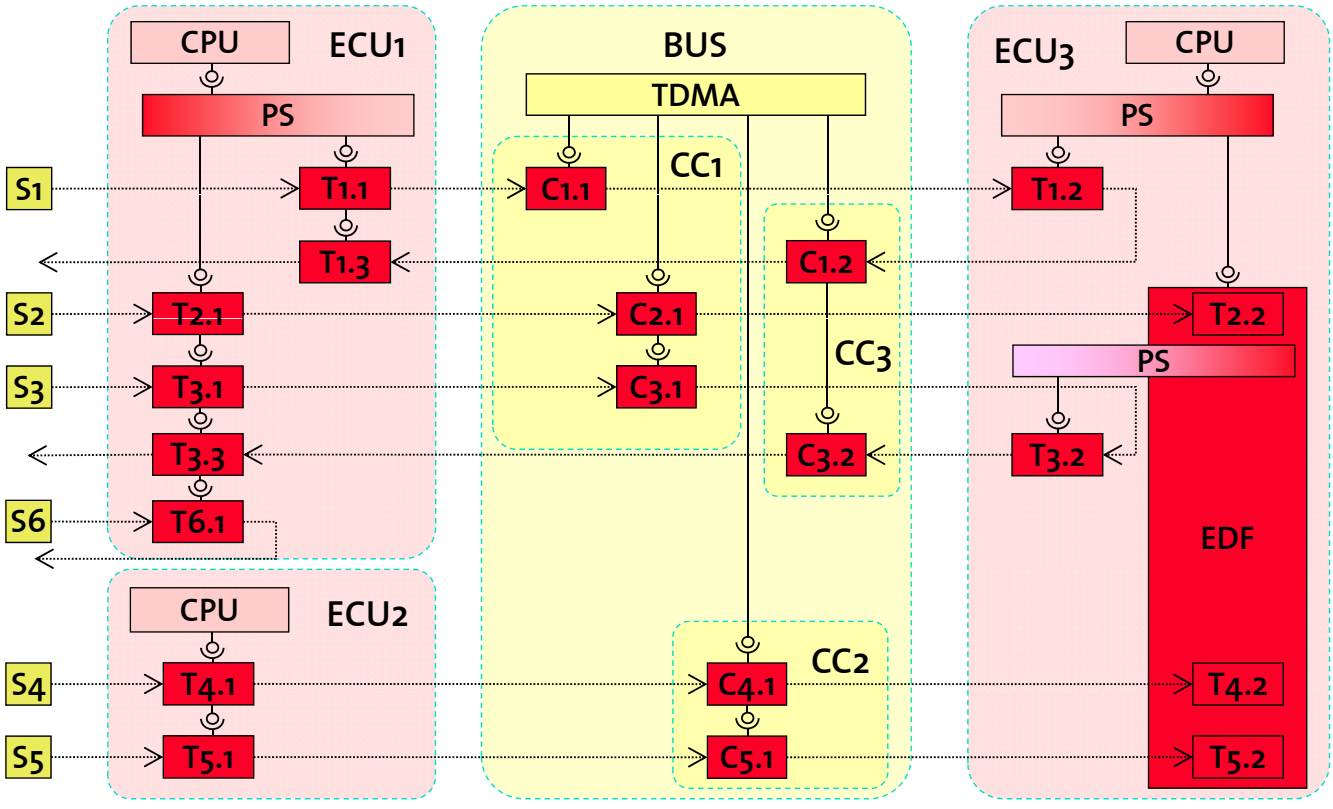
Perdiodic Server	p	e
SPS_{ECU1}	500	200
SPS_{ECU3}	500	250
DPS_{ECU3}	600	120

TDMA	t
Cycle	100
$Slot_{CC1a}$	20
$Slot_{CC1b}$	25
$Slot_{CC2}$	25
$Slot_{CC3}$	30

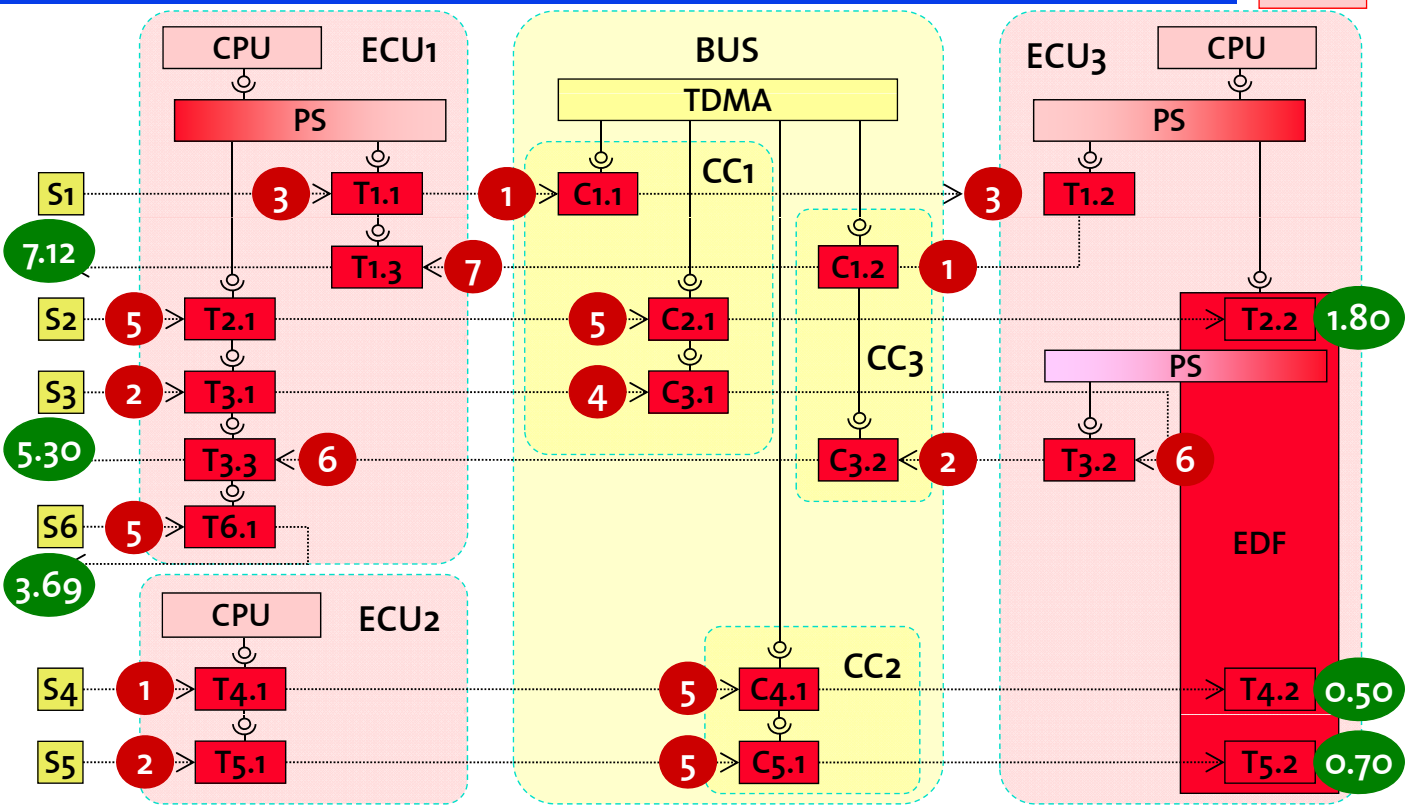
The Distributed Embedded System...



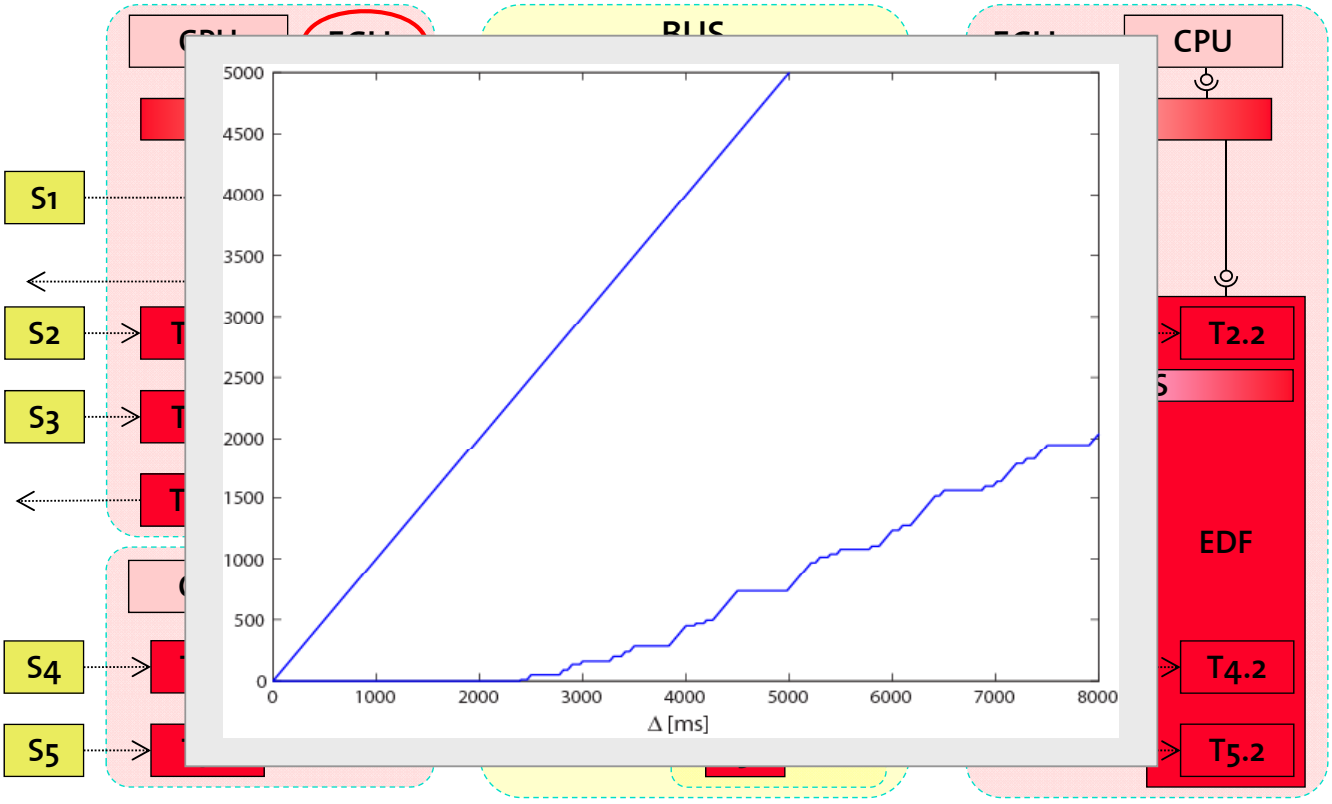
... and its MPA Model



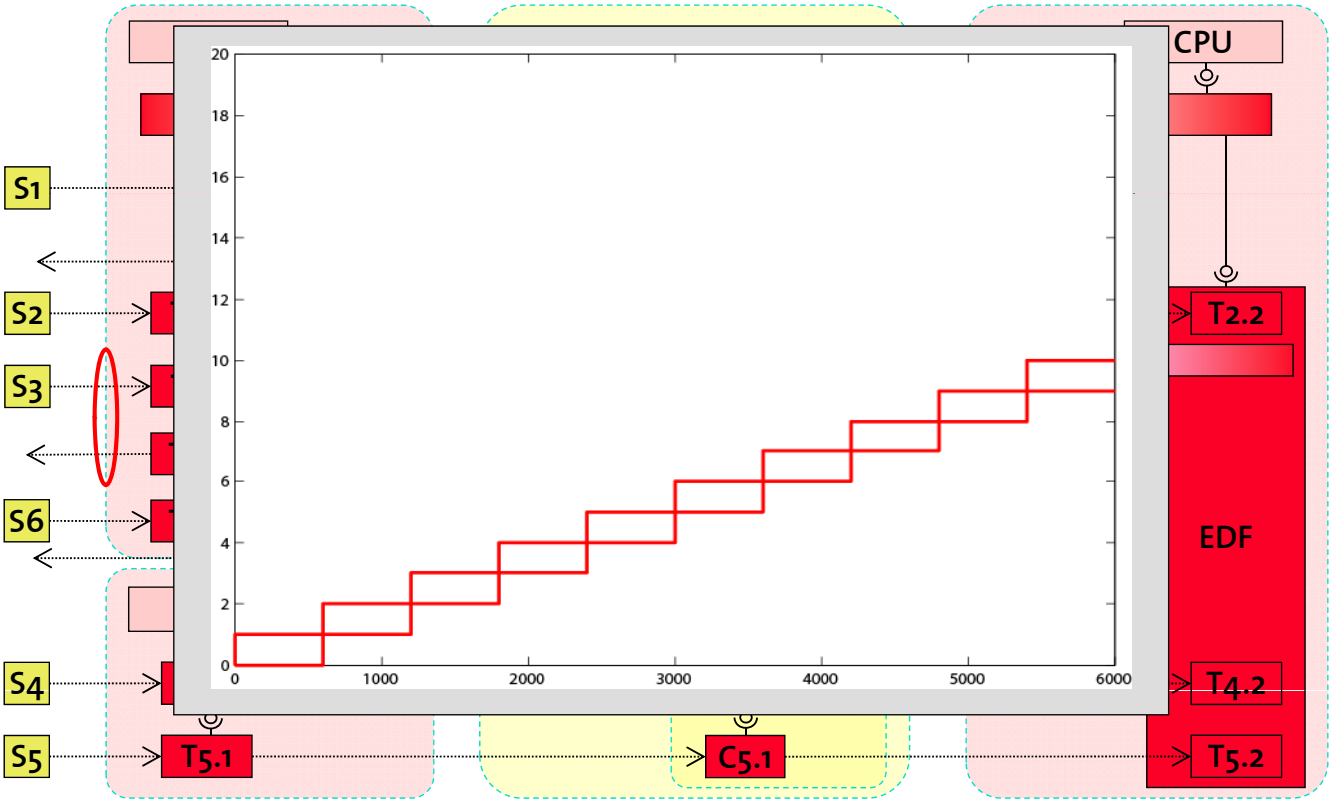
Buffer & Delay Guarantees



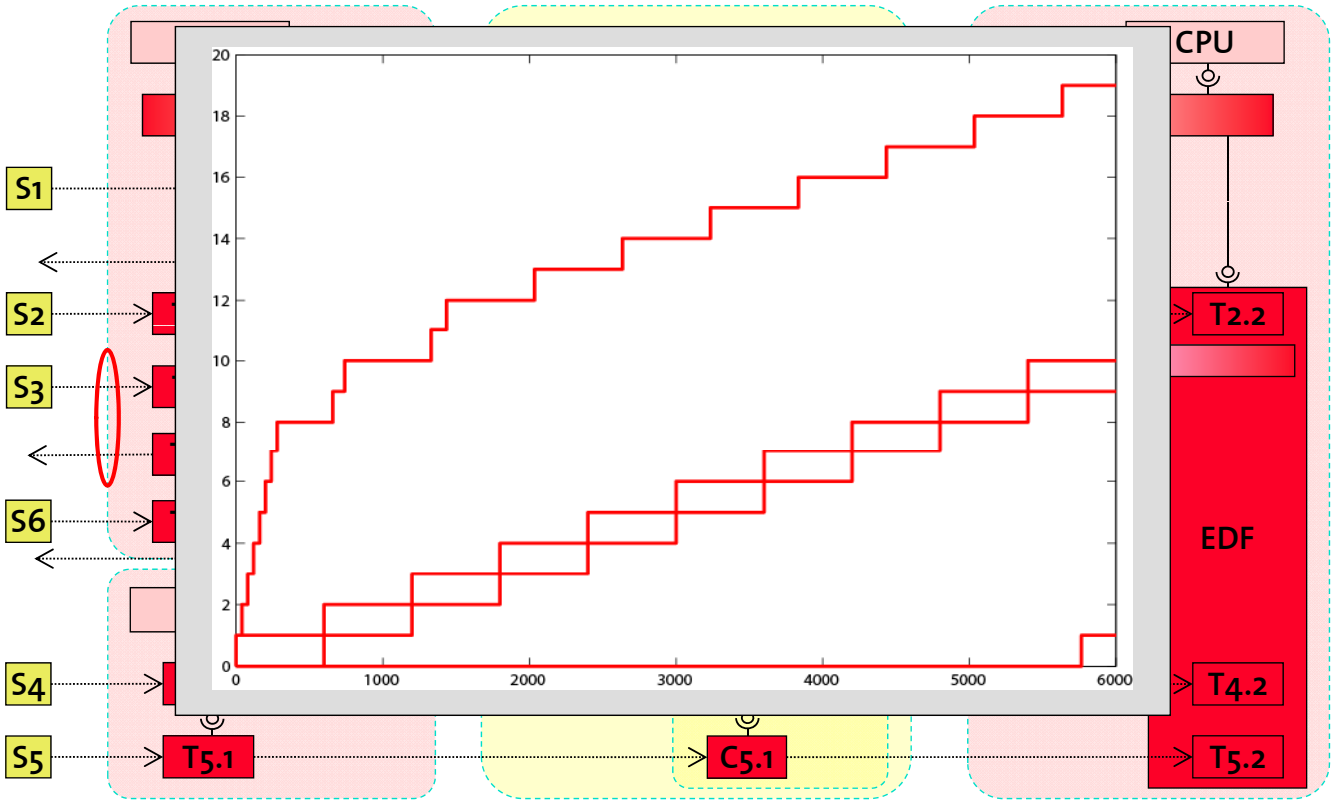
Available & Remaining Service of ECU1



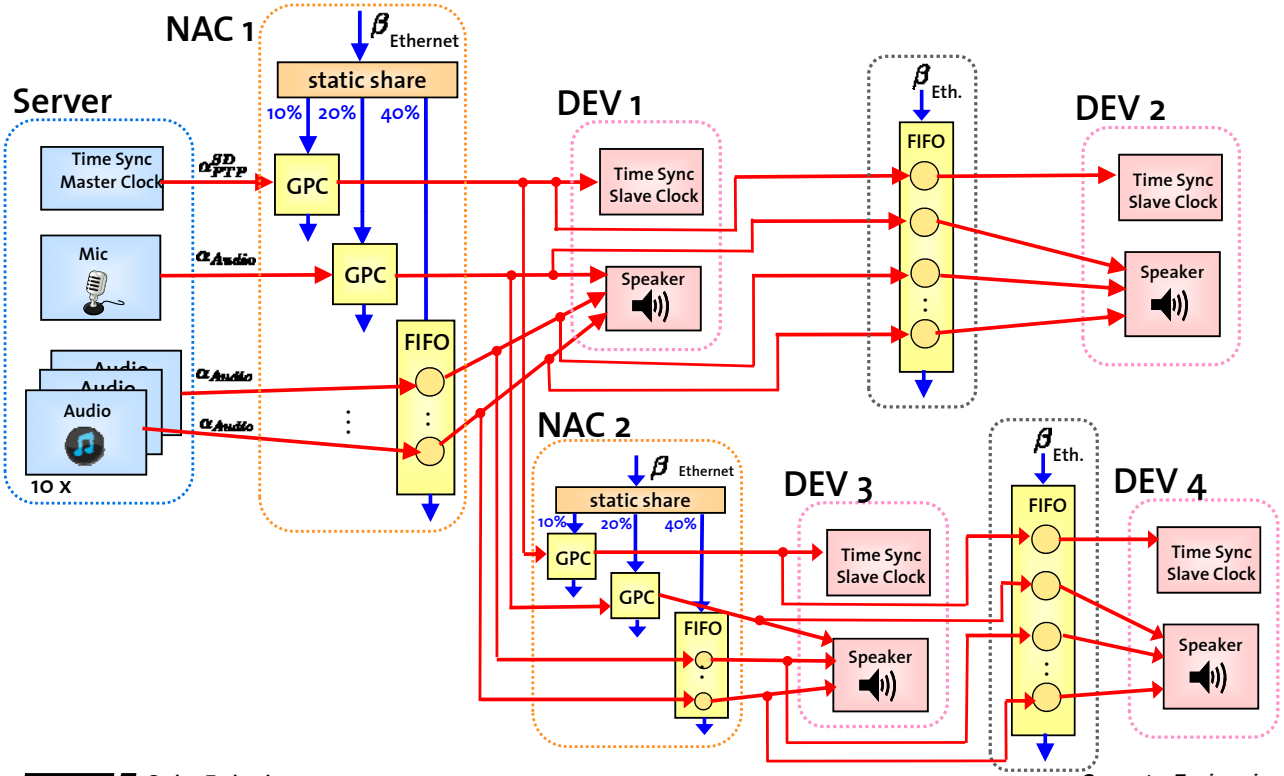
Input of Stream 3



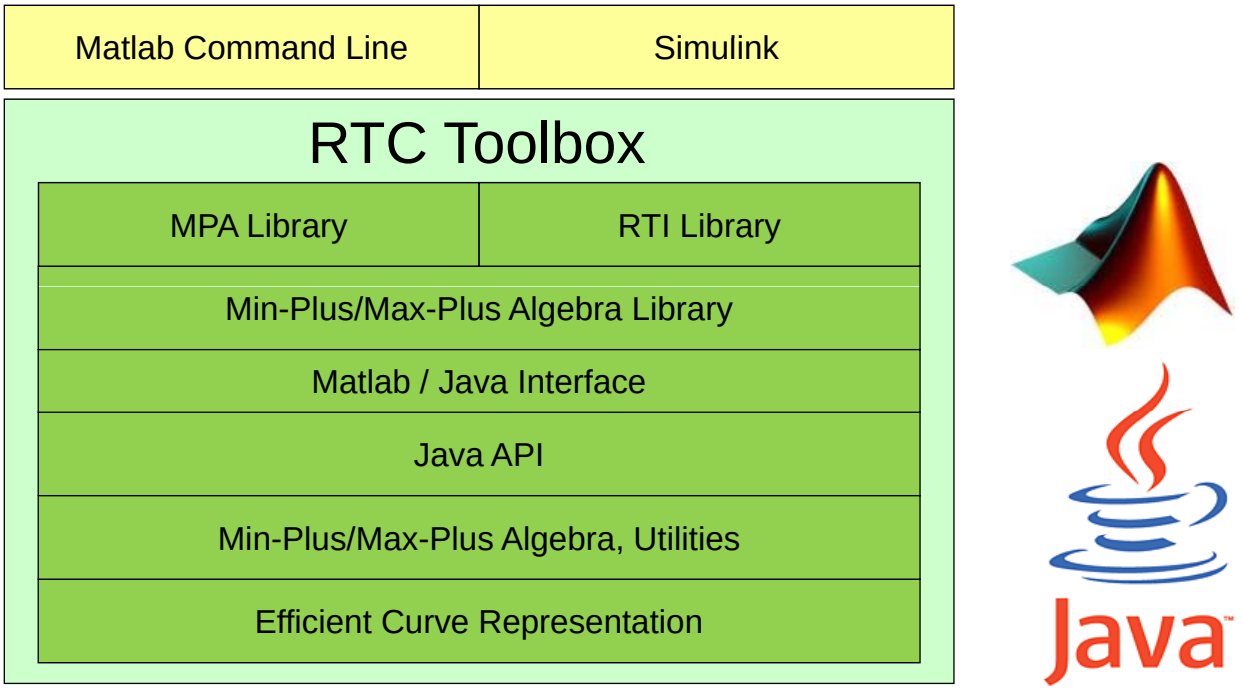
Output of Stream 3



Distributed Audio Communication



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



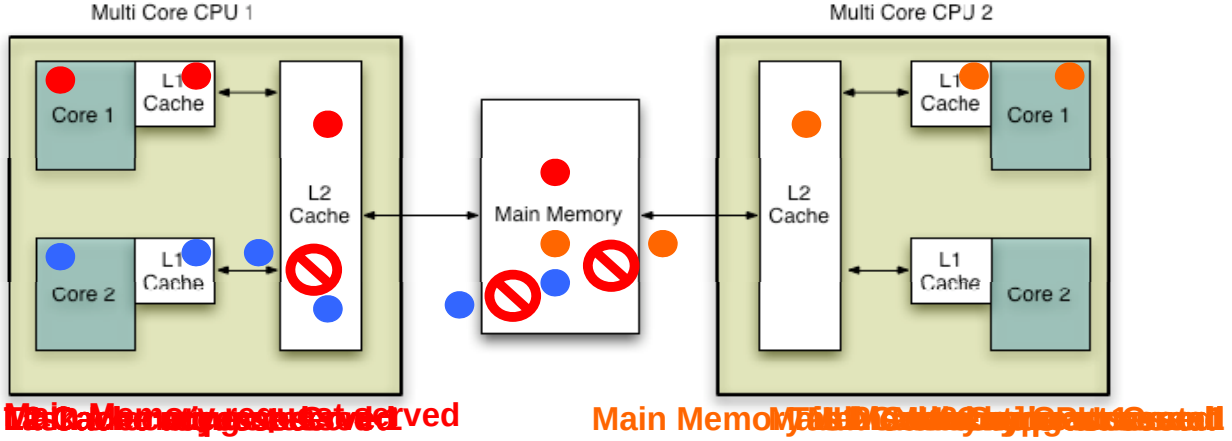
Compositional Analysis Examples

- Shared Resources in Multicore -

Interferences:
CPU1/Core2 blocked by **CPU1/Core1** on L2 Cache
CPU2/Core1 blocked by **CPU1/Core1** on Main Memory
CPU1/Core2 blocked by **CPU2/Core1** on Main Memory

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



Motivation (2)

Multi-Core Architecture with shared resource

- shared memory, communication peripherals, I/O peripherals

Stalling due to Interference

- Depends on structure of tasks on the cores
- Depends on blocking vs. non-blocking execution semantics
- Depends on arbitration policy on the shared resource
  - static access, for example TDMA
  - dynamic access, for example round robin, FCFS, priority driven

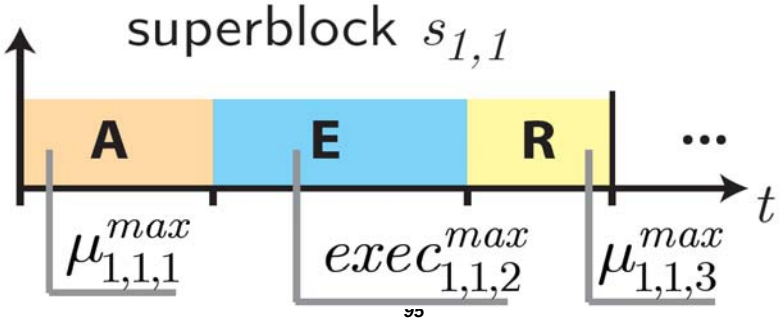
Related Work

- Schliecker et al. [CODES 2006, CODES 2008, DATE 2010]
  - Event models specify tasks interference in time windows
  - tasks active time increases by number of interferences
  - Iterative approach to compute WCET

- Rosen et al. [RTSS 2007]
  - static analysis delivers feasible execution traces
  - a given TDMA schedule the WCET is computed
  - efficient TDMA schedules are obtained using EA

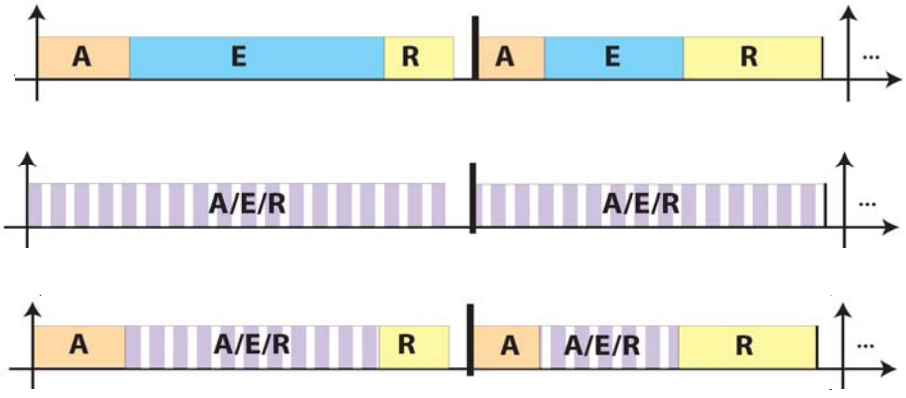
Task / Superblock Model (1)

- 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



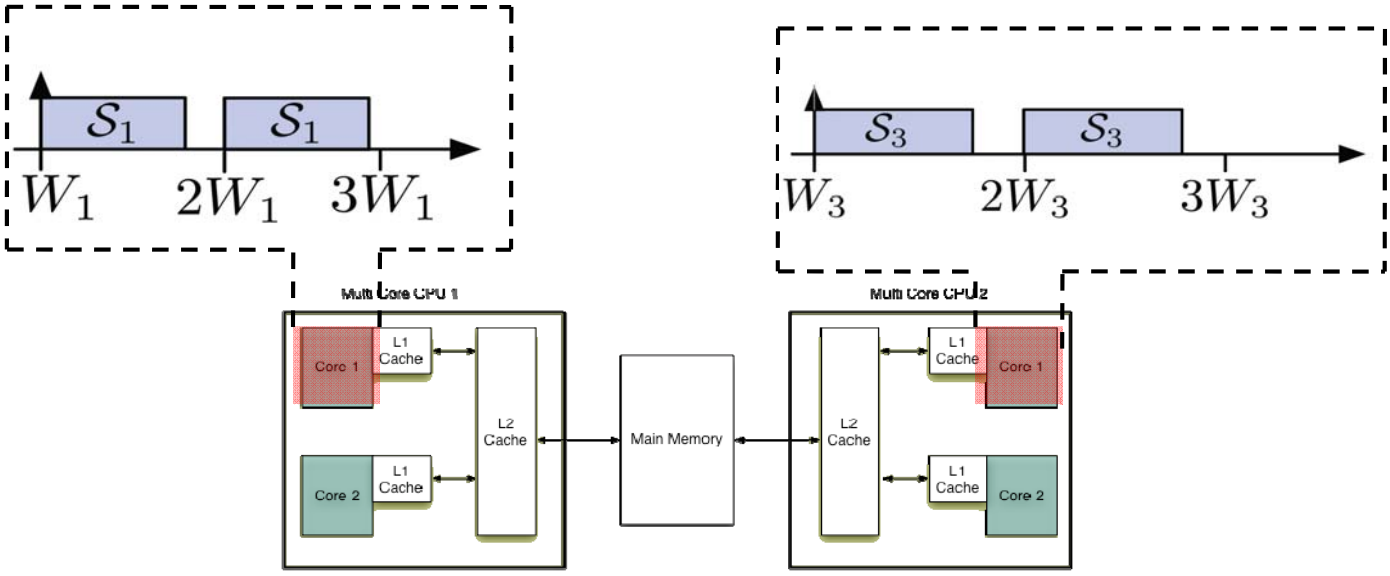
Task / Superblock Model (2)

- 3 Models to specify resource accesses:
  - Dedicated Model
  - General Model
  - Hybrid Model



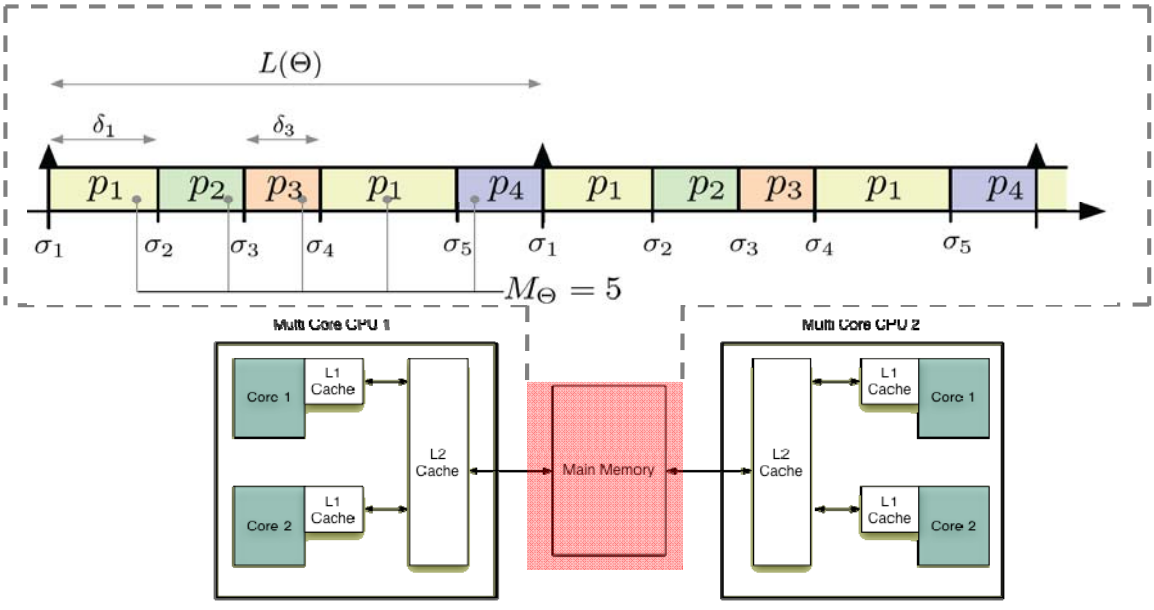
- 2 Models to execute superblocks:
  - Sequential
  - Time-triggered (superblocks, phases)

Static execution on the processing element



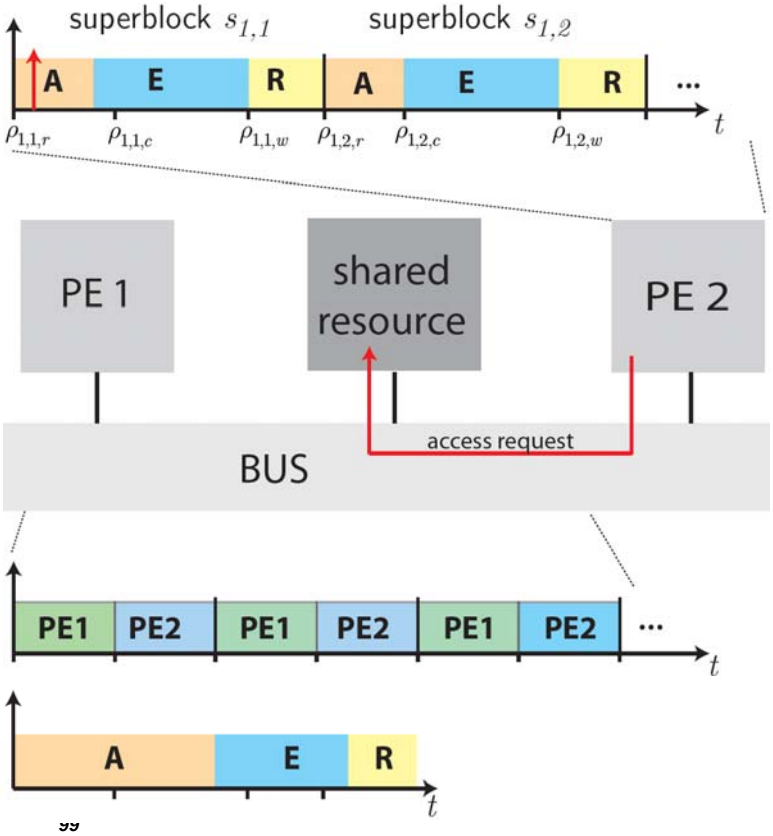
TDMA on the shared resource

Independence between tasks **single source of interference**



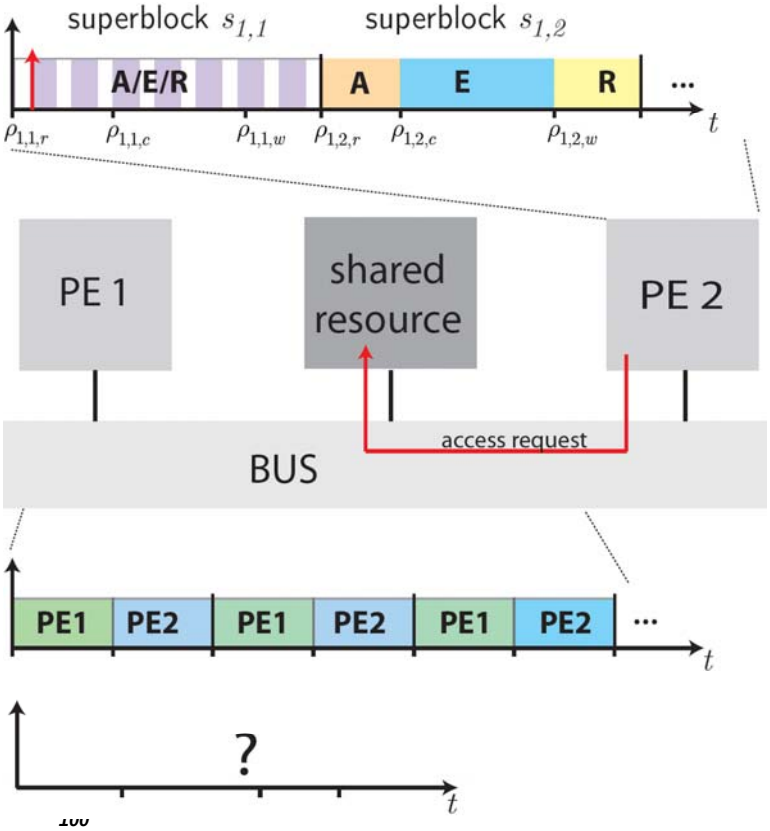
Static Arbitration (1)

- Analysis algorithm constructs worst-case trace
- Read/write request in acquisition/replication phase, access in active slot
- Execution phase is performed with no delay
- Example: assume PE2 requests access



Static Arbitration (2)

- Example: general superblock model
- Questions: where to place the access requests for worst case behavior?
- Algorithms exist that construct the worst case by maximizing stalling



Analysis for static arbitration - Summary

- analysis is complex
  - makes use of arrival and service curves (real-time calculus)
  - has been extended to dynamic resource sharing as well
- analysis handles dedicated and general phases
  - sequential and time-triggered execution
- analysis of mixed models possible by composition
  - superblocks can be specified using different models
- Time complexity
  - Dedicated phase: $O(M_\Theta)$
  - General phase: $O(M_\Theta \log(exec^{max}))$

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TIK
Computer Engineering and
Networks Laboratory

Resource Access Models (1)

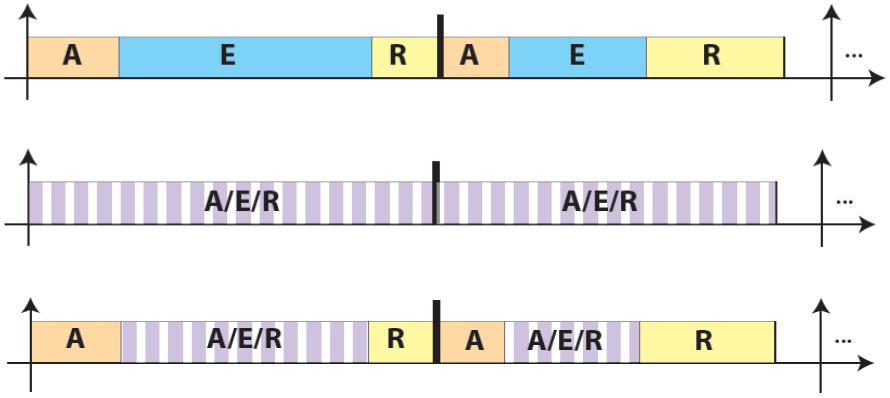
- What can we do with this kind of analysis?
  - Influence of different access models on schedulability
  - Influence of the execution model on predictability (equivalent WCRT)

- Intuition:
  - Separation of resource access and computation increases predictability
  - Everything time-triggered increases predictability

?

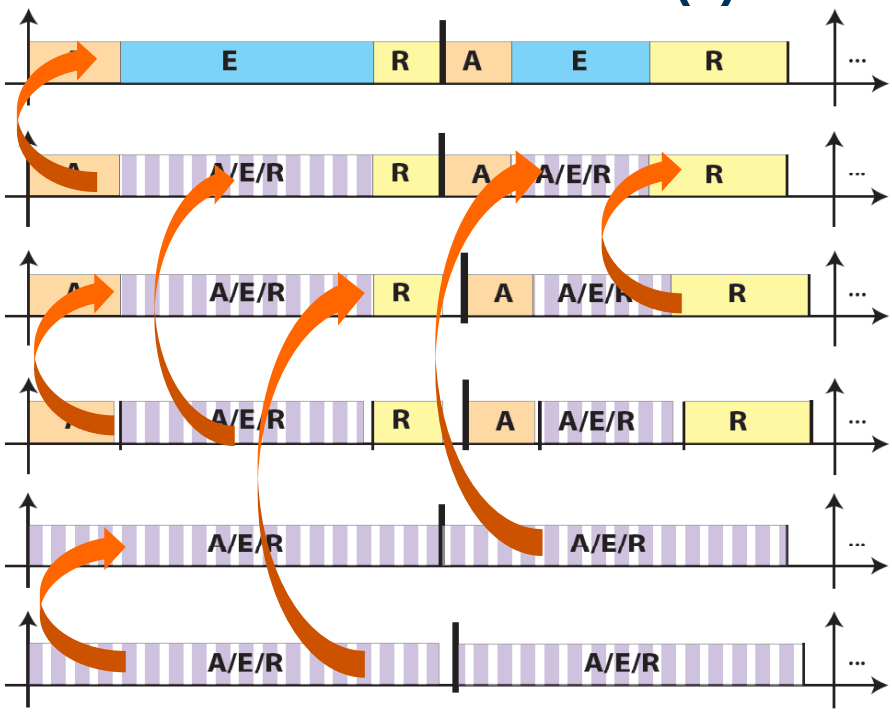
Resource Access Models (2) - Reminder

- 3 Models to specify resource accesses:
  - Dedicated Model
  - General Model
  - Hybrid Model



- 2 Models to execute superblocks:
  - Sequential
  - Time-triggered (superblocks, phases)

Resource Access Models (3)



DS dedicated sequential phases, sequential superblocks

HS hybrid sequential phases, sequential superblocks

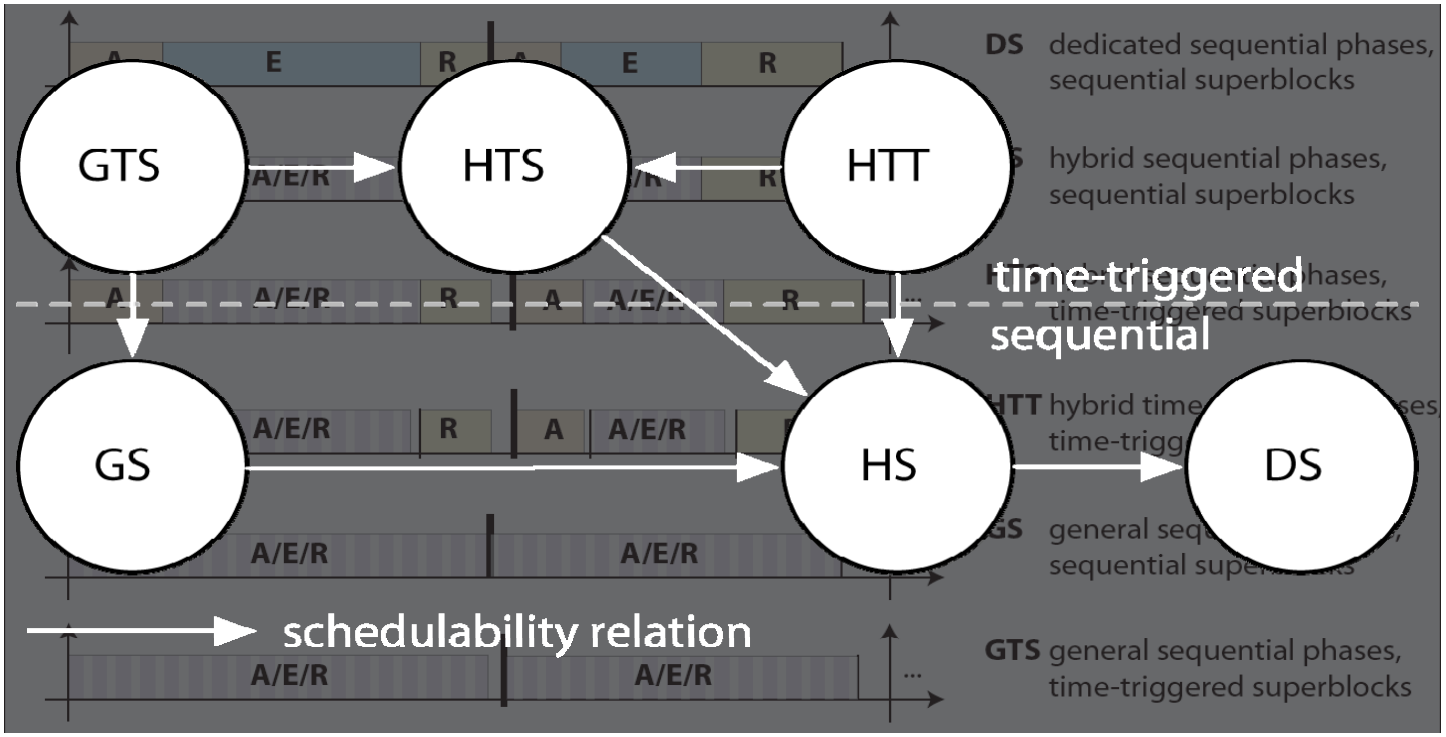
HTS hybrid sequential phases, time-triggered superblocks

HTT hybrid time-triggered phases time-triggered superblocks

GS general sequential phases, sequential superblocks

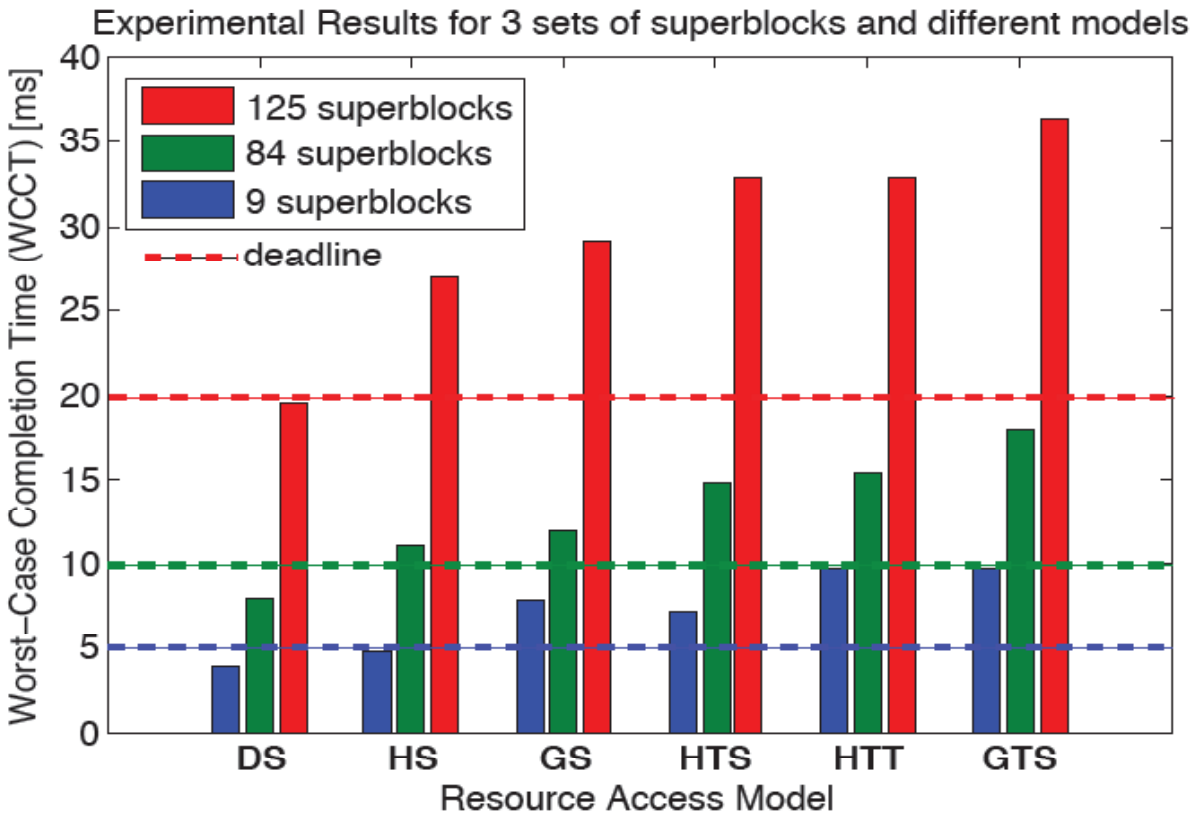
GTS general sequential phases, time-triggered superblocks

Schedulability between Models



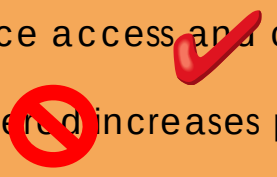
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TIK
Computer Engineering and
Networks Laboratory



Comparisons of Access Models

- Intuition:
  - Separation of resource access and computation increases predictability
  - Everything time-triggered increases predictability



?

- Excessive time-triggering may degrade performance
- No advantage in terms of predictability
- Model DS is model choice for resource sharing systems
  - Separate Memory Access and Computation

Conclusion

- Resource Sharing in Multi-Core Systems is an important issue in terms of
  - Analyzability
  - Predictability
  - Efficiency
- Static arbitration policies
  - Elimination of Interference
  - Tight bounds on WCCT can be derived
- Excessive time-triggering is counter productive

Even for simple models:
Resource Sharing is a hard Problem

Contents

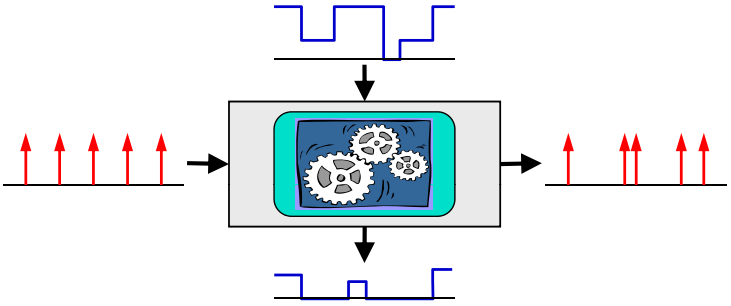
- Drivers
- Compositional Analysis
  - Overview
  - Real-Time Calculus
- Examples
  - Shapers
  - Artificial Example
  - Shared Resources in Multicore Systems
- ***Extensions***
- Comparison
- Challenges

Compositional Analysis Extensions

- State-based Modeling -

Extending the Framework

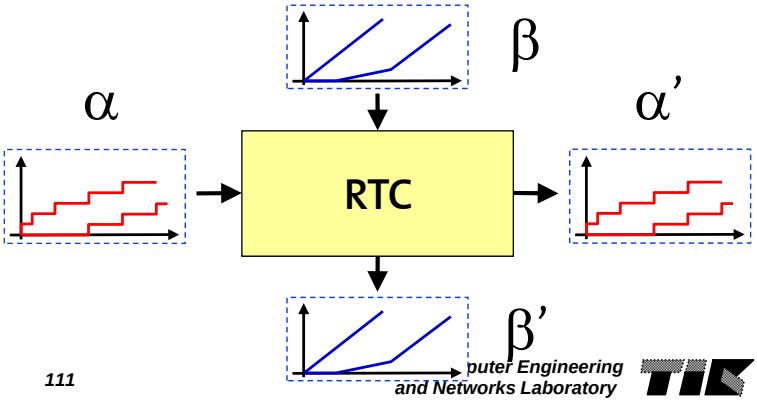
- New HW behavior
- New SW behavior
- New scheduling scheme
- …



- Find new relations:

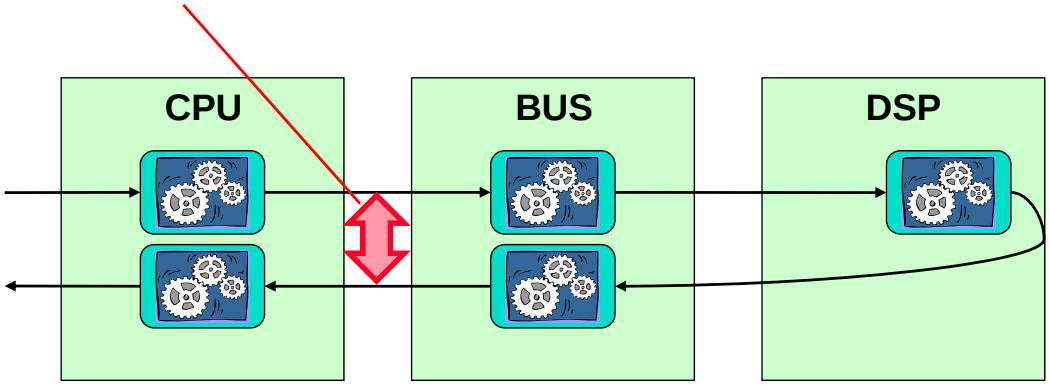
$$\alpha'(\Delta) = f_\alpha(\alpha, \beta)$$
$$\beta'(\Delta) = f_\beta(\alpha, \beta)$$

This is the hard part…!



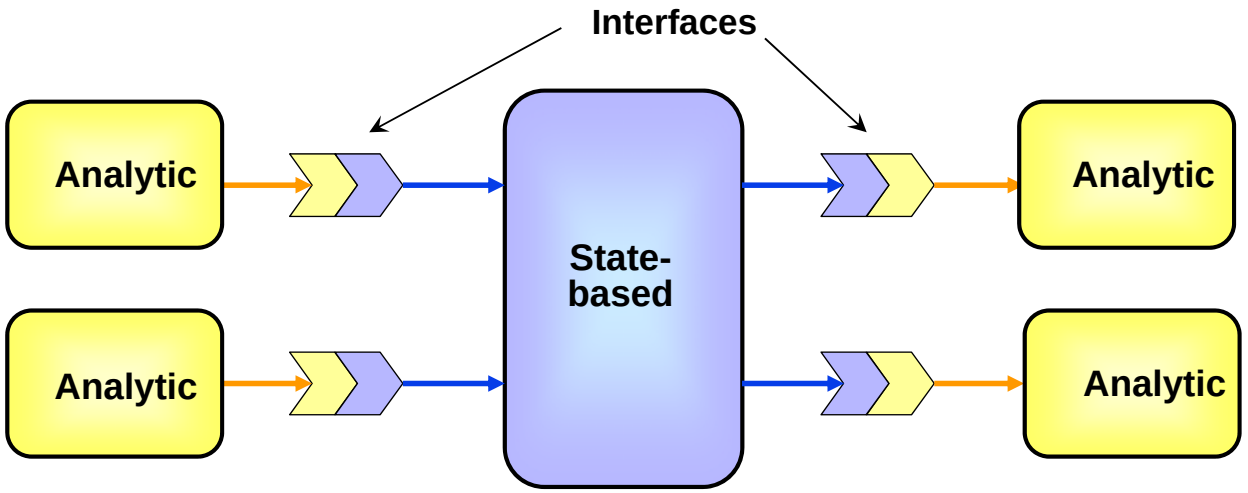
Compositional Methods

- … suffer from abstraction loss:
  - For example, we are not able to properly model timing correlations between streams.
  - Analysis results may be overly pessimistic
  - We need new models that are able to talk about timing **correlations** between event streams.



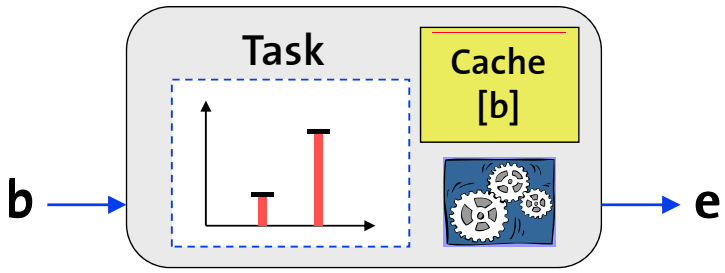
Compositional Methods

- … suffer from abstraction loss:
  - For example, we are not able to properly state-based behavior of components.



Refined Processing Component Model

Formal Specification
Program Analysis
Data Sheets
…

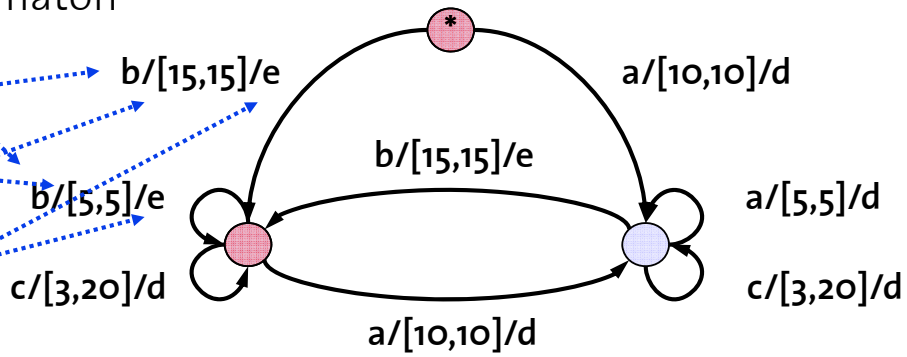


Functional Unit Automaton

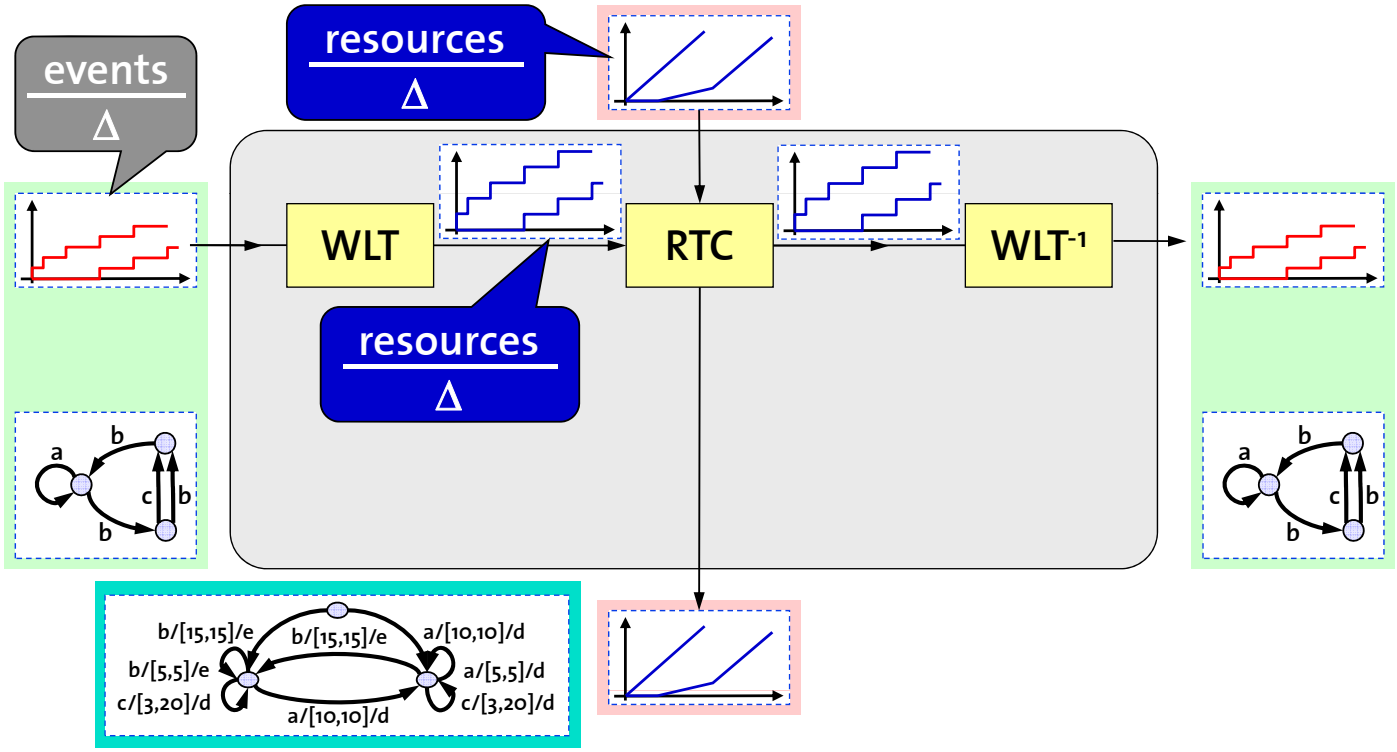
triggering event

min/max resource demand

produced event



Processing Component

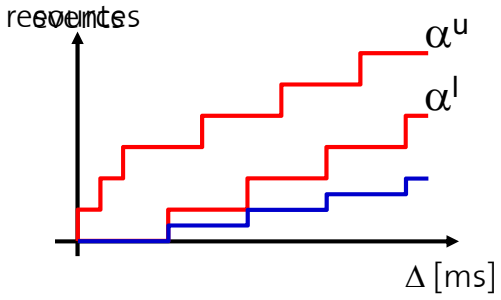


Classical Workload Characterization

Worst Case Execution Demand
&
Best Case Execution Demand

WCED = 2 [resources/event]
BCED = 0.5 [resrouces/event]

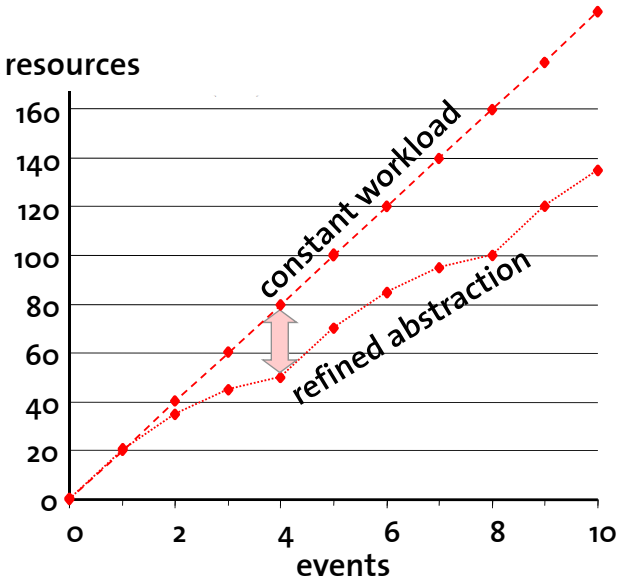
events	BCED	WCED
1	0.5	2
2	1	4
3	1.5	6
4	2	8



Improvement: Workload Curves

(Workload Curves) *Let $W(u)$ denote the total resource demand created on a component by u consecutive events of an incoming event stream. For every event sequence on the incoming event stream, the lower workload curve γ^l and the upper workload curve γ^u satisfy the relation:*

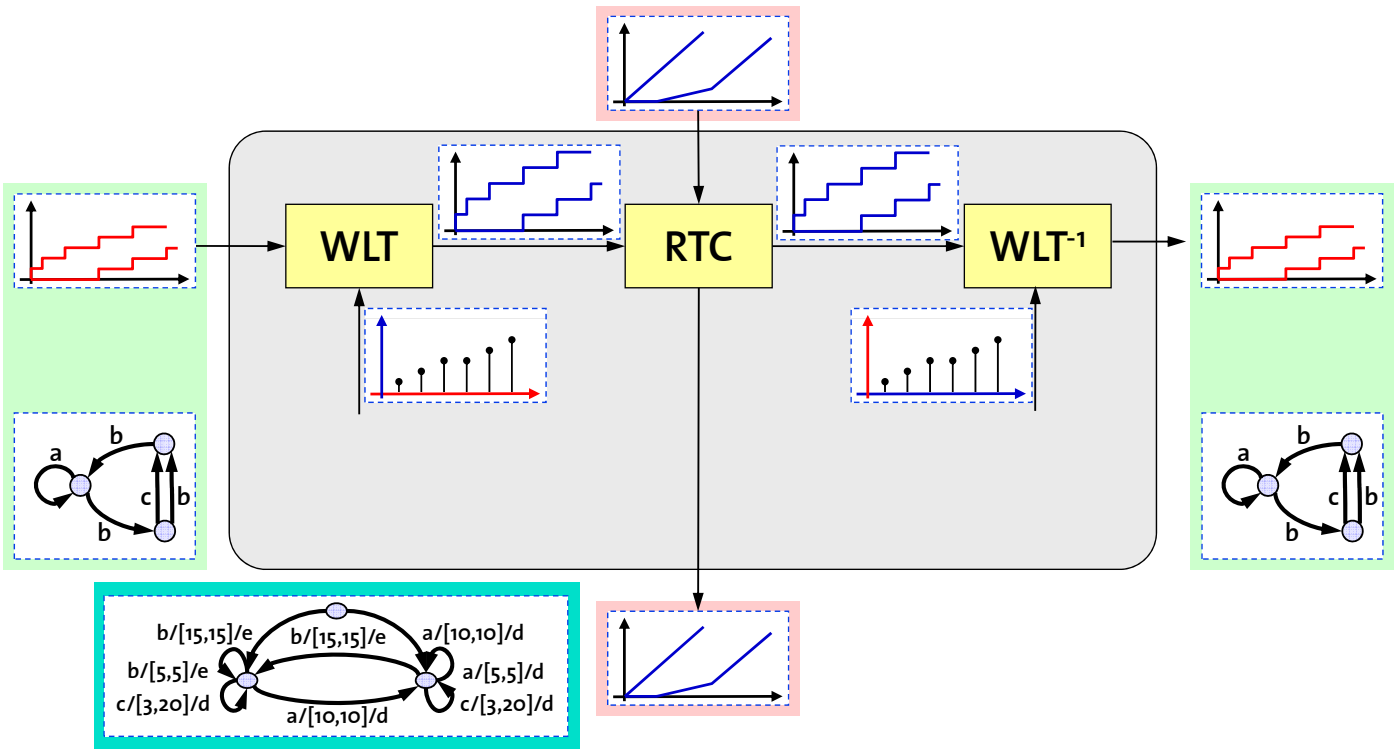
$$\gamma^l(v-u) \leq W(v) - W(u) \leq \gamma^u(v-u)$$



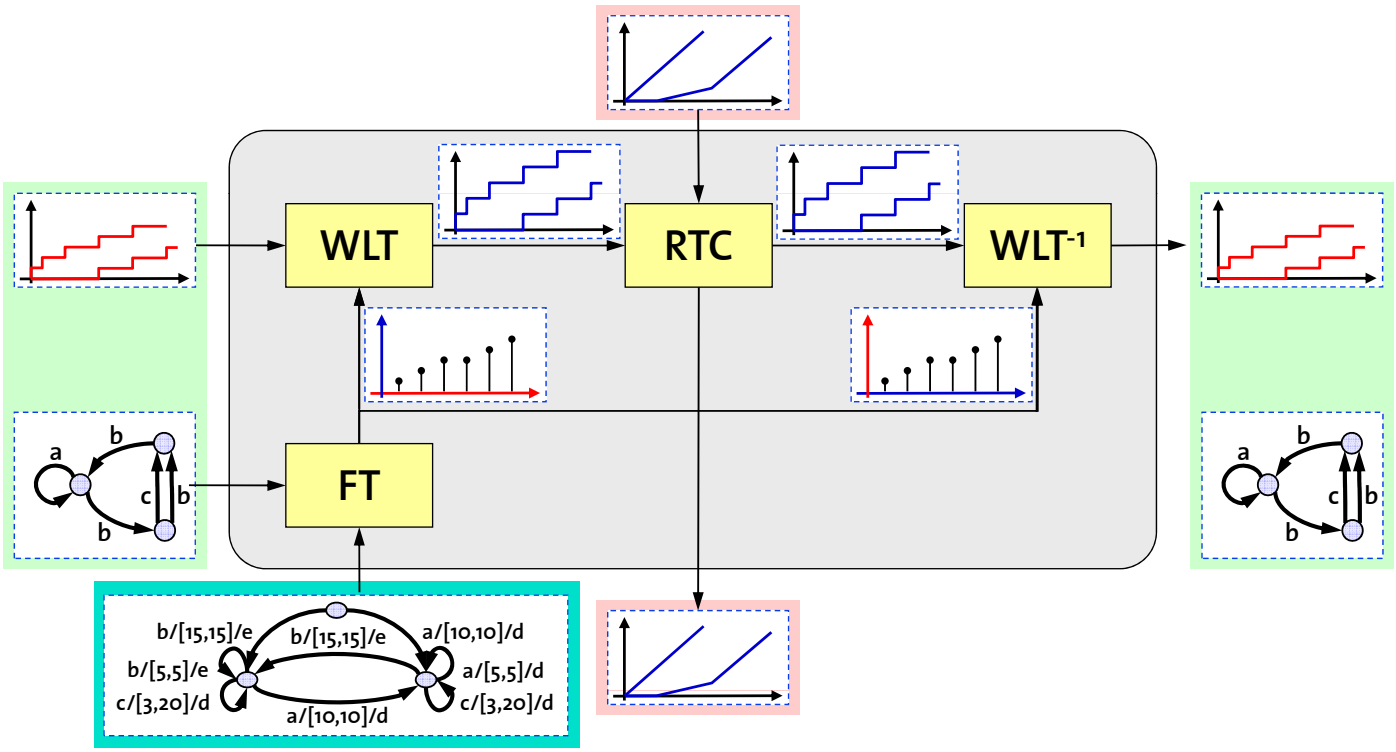
$$\alpha^u(\Delta) = \gamma^u(\overline{\alpha}^u(\Delta)) \qquad \alpha^l(\Delta) = \gamma^l(\overline{\alpha}^l(\Delta))$$
$$\overline{\alpha}^u(\Delta) = \gamma^{l-1}(\alpha^u(\Delta)) \qquad \overline{\alpha}^l(\Delta) = \gamma^{u-1}(\alpha^l(\Delta))$$
$$\beta^u(\Delta) = \gamma^u(\overline{\beta}^u(\Delta)) \qquad \beta^l(\Delta) = \gamma^l(\overline{\beta}^l(\Delta))$$
$$\overline{\beta}^u(\Delta) = \gamma^{l-1}(\beta^u(\Delta)) \qquad \overline{\beta}^l(\Delta) = \gamma^{u-1}(\beta^l(\Delta))$$

transformation of arrival and service curves

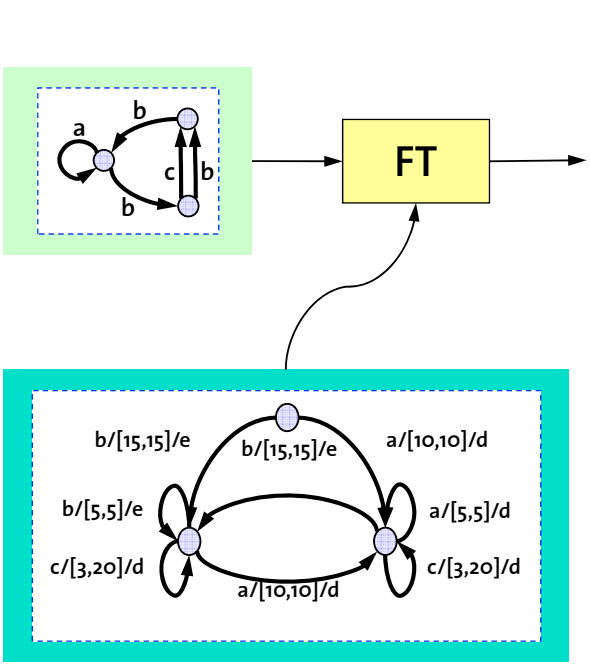
Processing Component

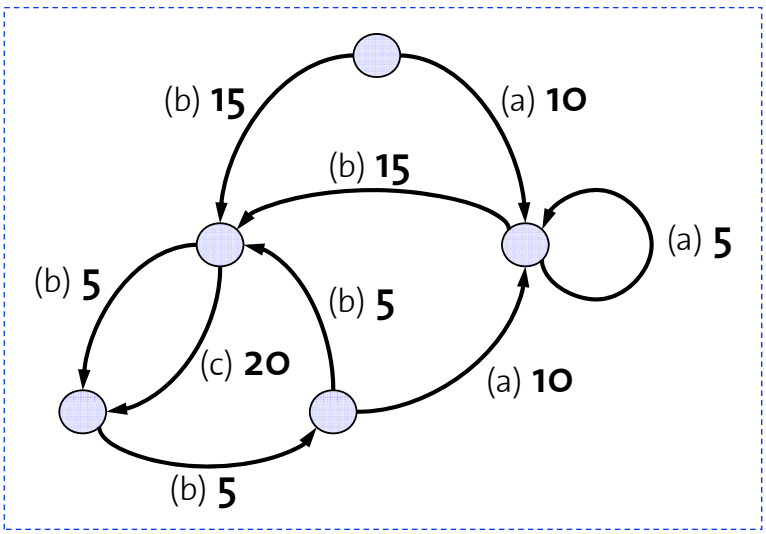


Processing Component



WLT with Abstracted Functionality

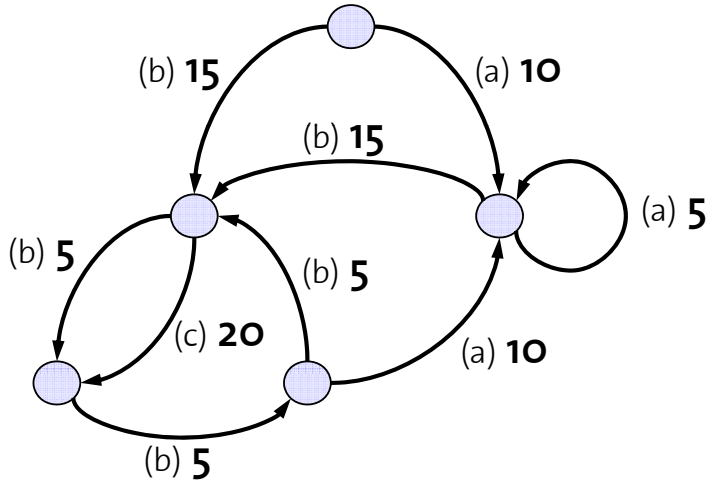




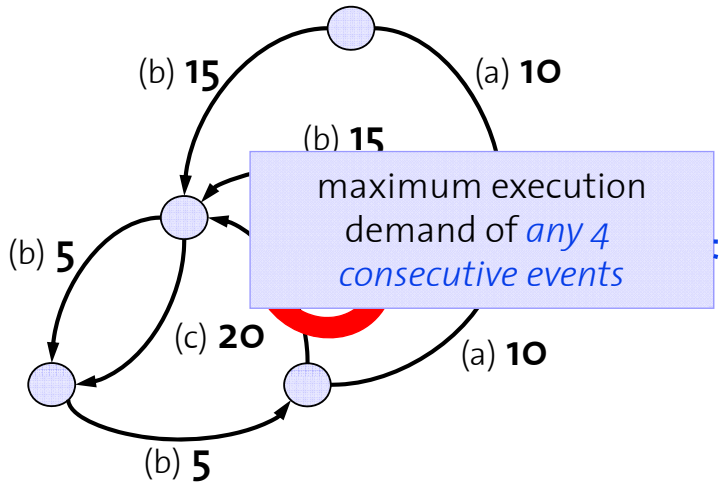
Edges: triggering events

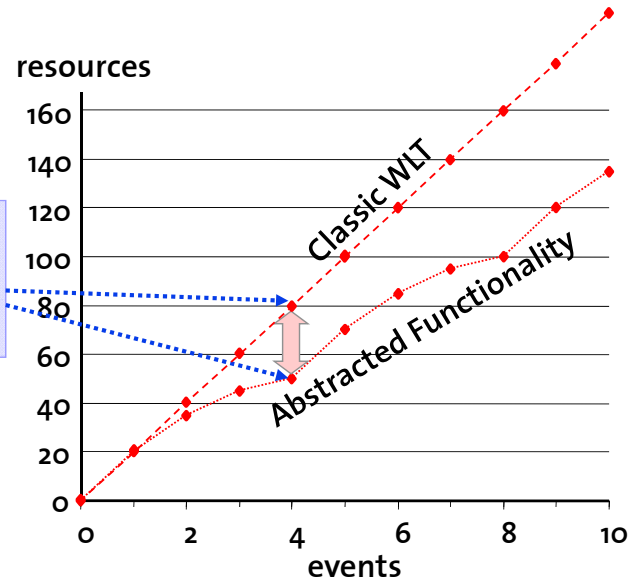
Weights: WCED

WLT with Abstracted Functionality



WLT with Abstracted Functionality

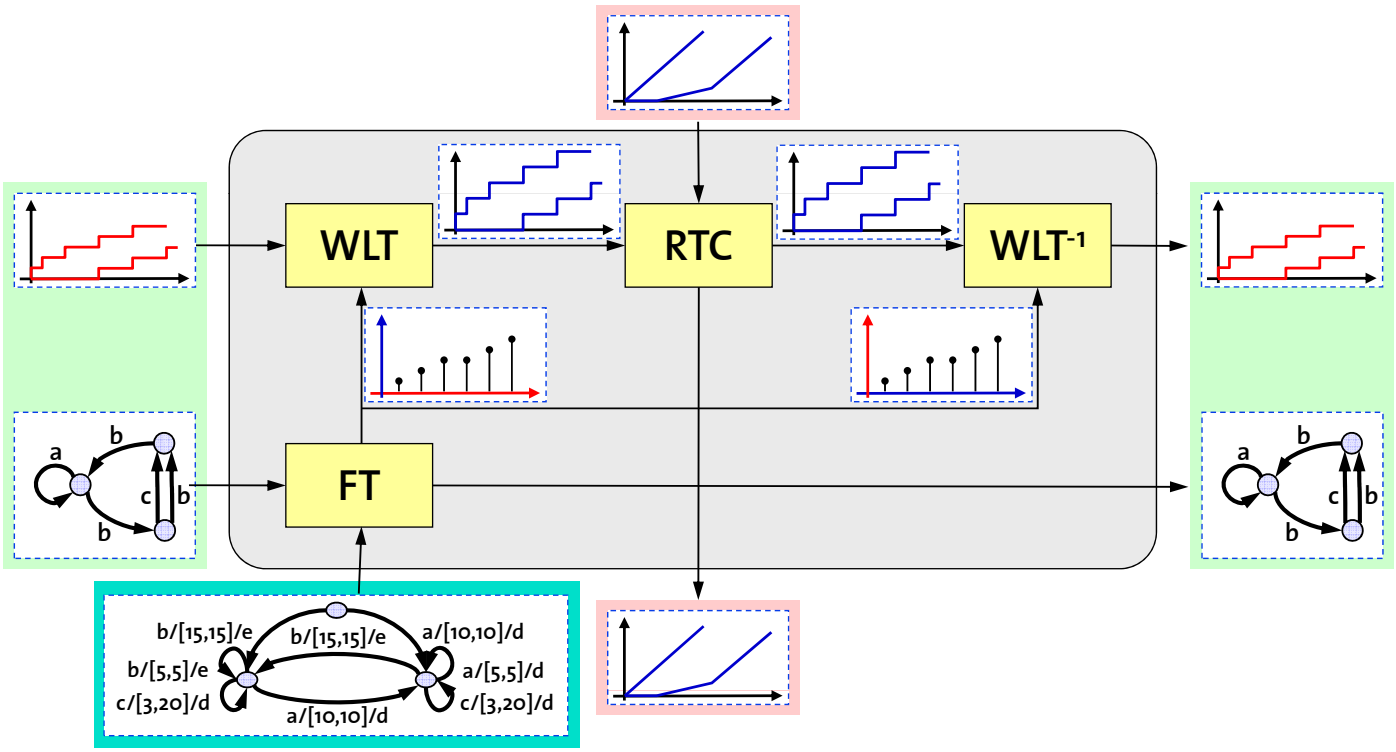




Execution demand of *n* consecutive events:

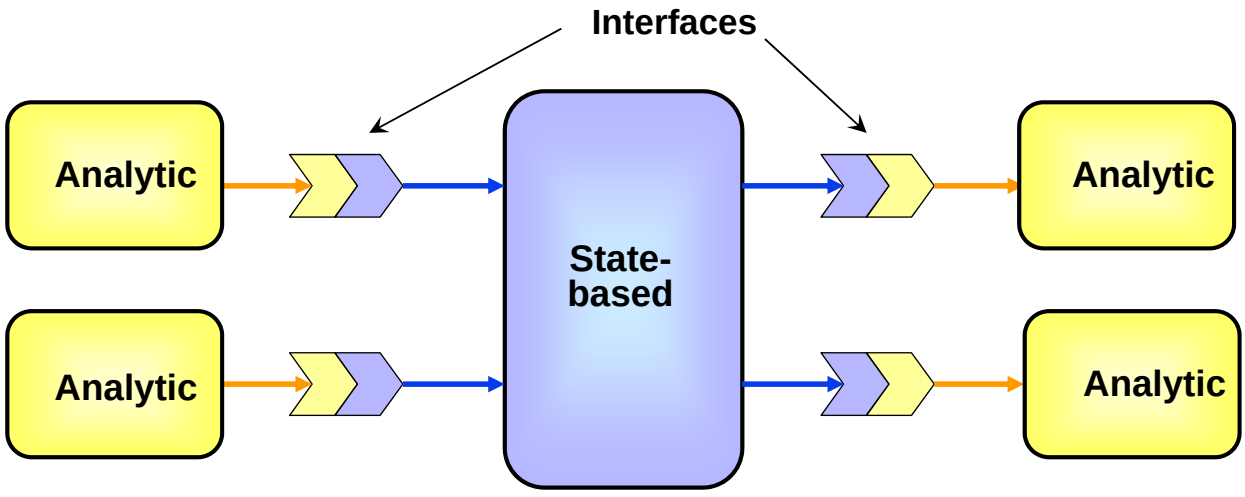
WCED(*n*) = max-weight path of length *n*

System Module



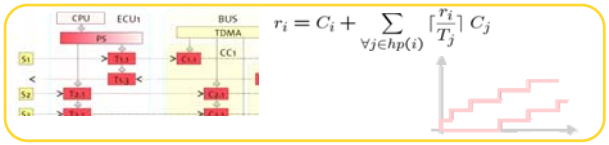
Compositional Methods

- … suffer from abstraction loss:
  - For example, we are not able to properly state-based behavior of components.



Comparsion of Different Abstractions

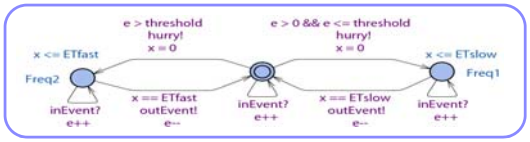
Analytic Real-Time Analysis



Solution of closed form expressions

Examples: RTC, SymTA/S, MAST, ...

State-based Real-Time Analysis



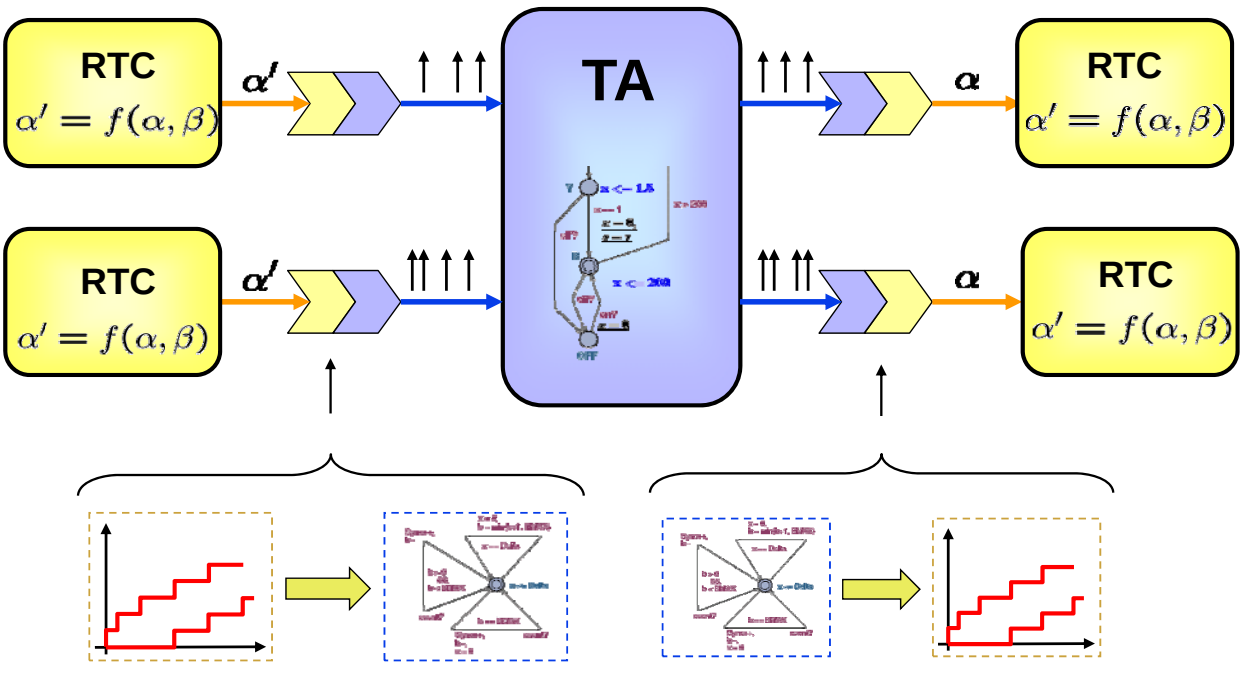
Model checking of properties

Examples: Timed Automata (TA), FSM, ...

+ Good scalability	− Poor scalability
+ Fast analysis	− Slow verification
− Limited to few specific measures (e.g. delays, buffer sizes)	+ Verification of functional and non-functional properties
− Systems restricted to specific models	+ Modeling power
− Overly conservative results	+ Exact results

State space explosion

Interfacing



Related work

- **Event Count Automata**

  L. T. X. Phan, S. Chakraborty, P. S. Thiagarajan, and L. Thiele. ***Composing functional and state-based performance models for analyzing heterogeneous real-time systems***. In Proc. of the 28th IEEE Real-Time Systems Symposium (RTSS 2007), pages 343–352. IEEE Computer Society, 2007.

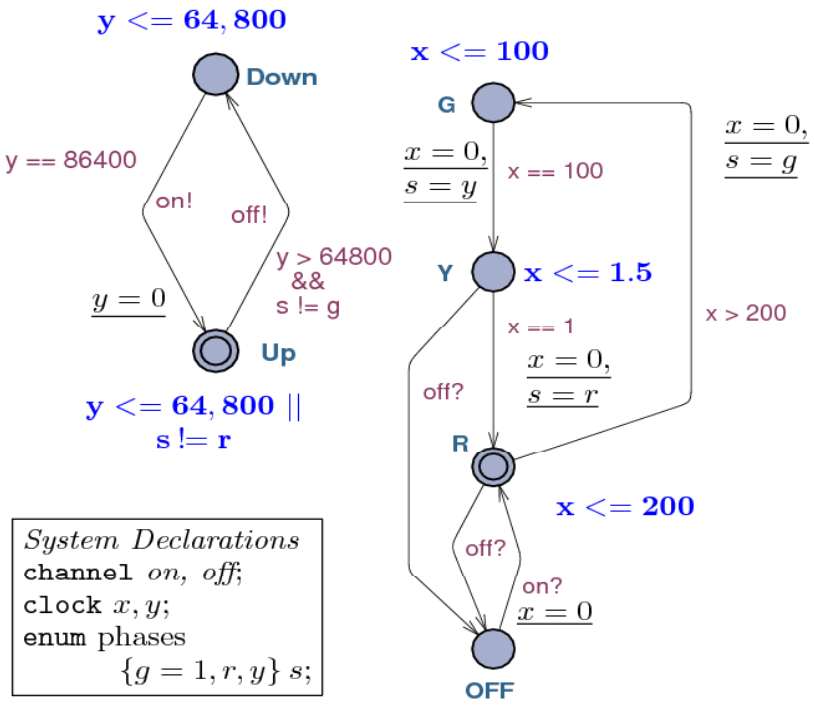
- **CATS Tool**

  P. Krcal, L. Mokrushin, and W. Yi. ***A tool for compositional analysis of timed systems by abstraction*** (extended abstract). In Proc. of NWPT07, the 19th Nordic Workshop on Programming Theory, October 2007.

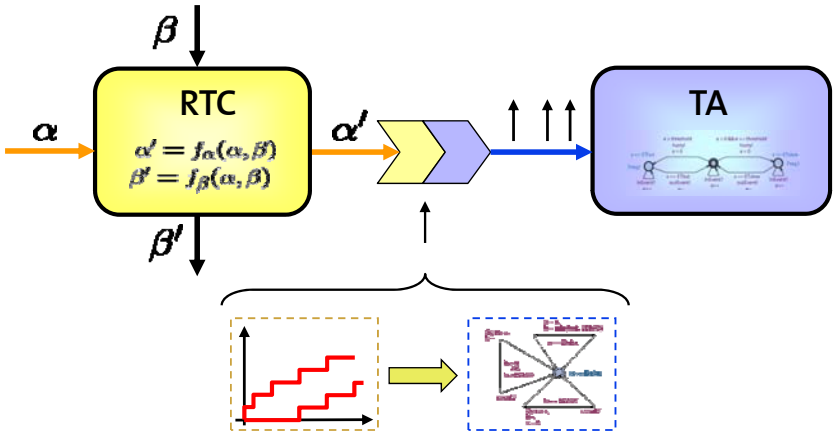
- **Efficient Model-Checking for Real-Time Task Networks**

  H. Dierks, A. Metzner, and I. Stierand. ***Efficient Model-Checking for Real-Time Task Networks***. In Int. Conf. on Embedded Software and Systems 2009. Accepted for publication.

Timed Automata (TA)

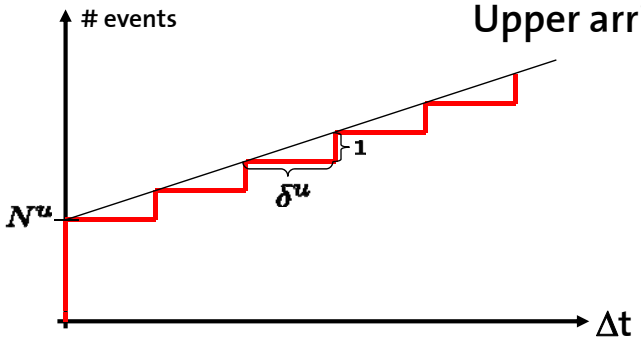


Interface RTC → TA



How to represent arrival curves as TA?

Linear arrival curves

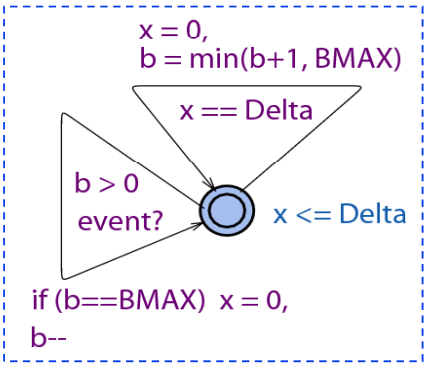


$$\alpha^u(\Delta) = N^u + \left\lfloor \frac{\Delta}{\delta^u} \right\rfloor$$

Max fill level: N^u

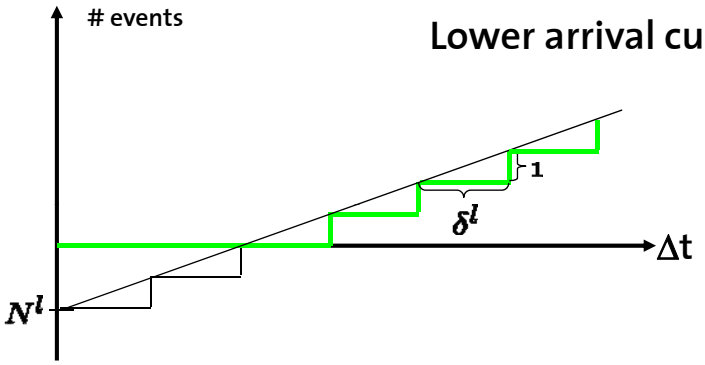
Fill rate: $1/\delta^u$

Event emission allowed if fill level > 0



Automaton for linear upper arrival curve

Linear arrival curves

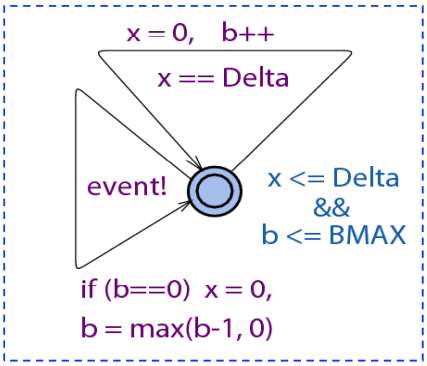


$$\alpha^l(\Delta) = \max\left\{0, N^l + \left\lfloor \frac{\Delta}{\delta^l} \right\rfloor\right\}$$

Max fill level: $|N^l|$

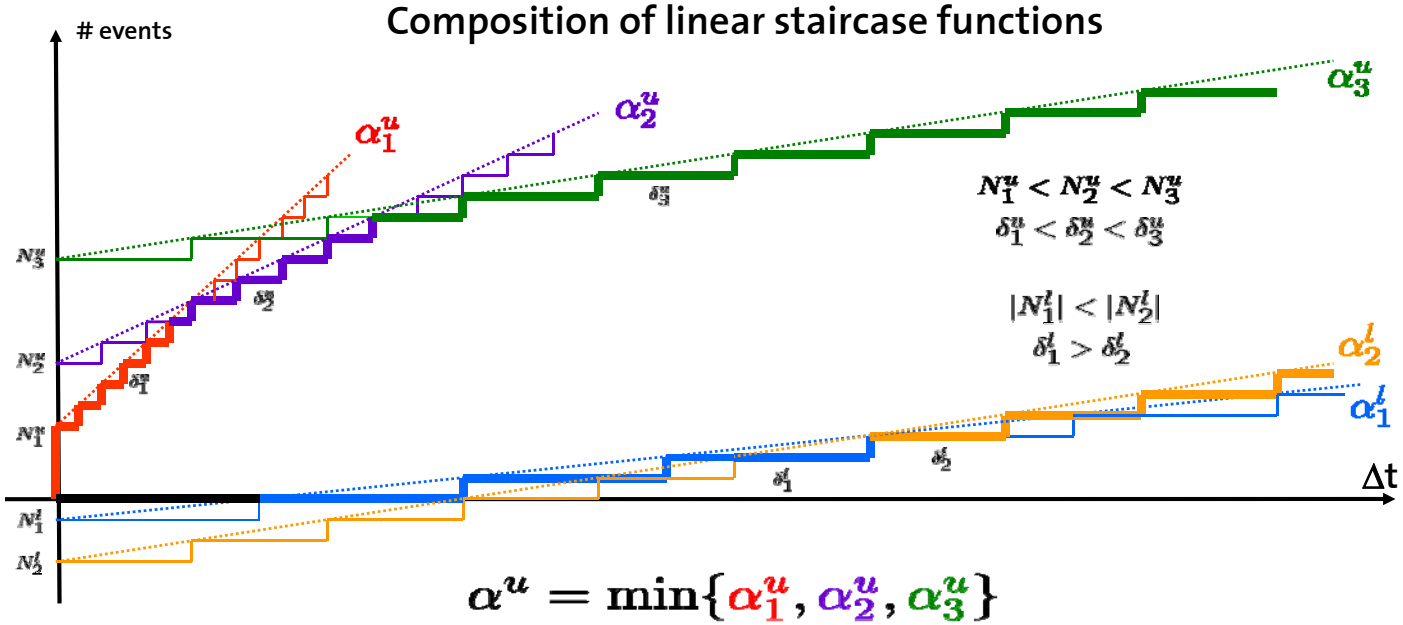
Fill rate: $1/\delta^l$

Event emission enforced if maximum fill level reached



Automaton for linear lower arrival curve

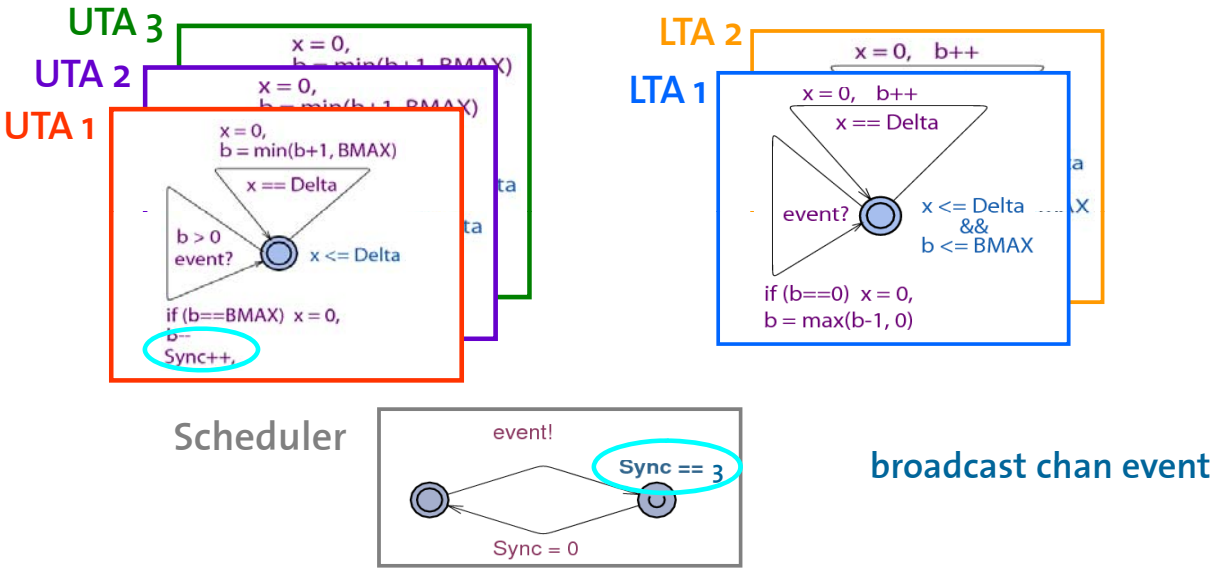
Convex and concave patterns



$$\alpha^u = \min\{\alpha_1^u, \alpha_2^u, \alpha_3^u\}$$

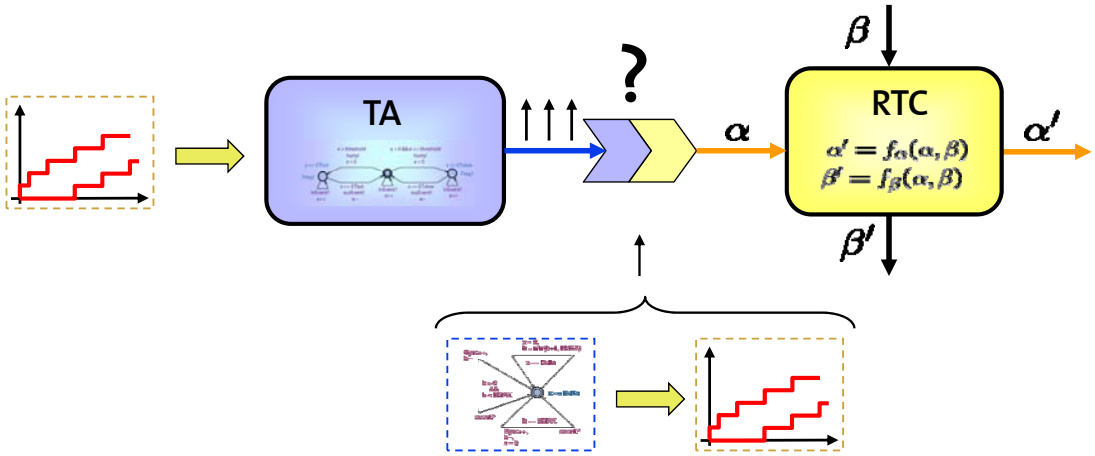
$$\alpha^l = \max\{0, \alpha_1^l, \alpha_2^l\}$$

Convex and concave patterns



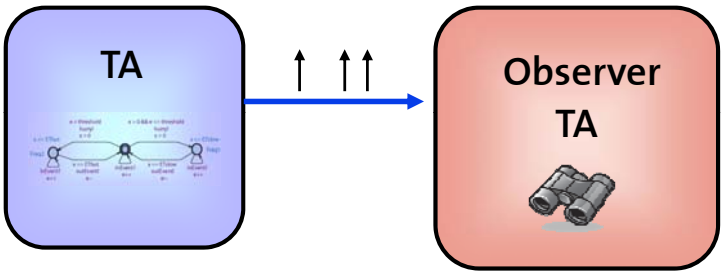
- Event generation only if all UTA permit it (AND composition)
- Single LTA can enforce event generation (OR composition)

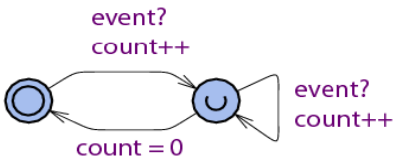
Interface TA → RTC



How to derive output arrival curves from a TA sub-system model?

Interface TA → RTC





Verify

```
A[] (count<=estimate)
```

Key parameters of curve (e.g. max burst) are determined by appropriate observer TA and binary search

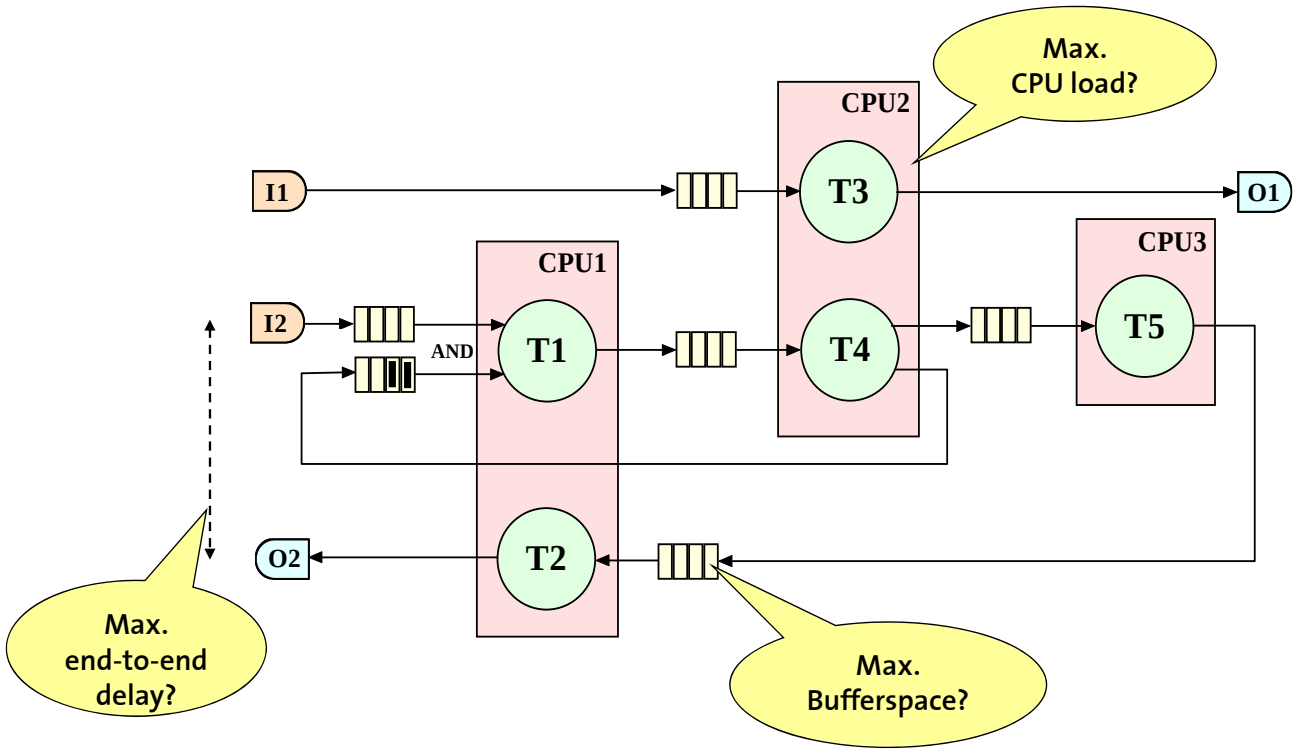
Contents

- Drivers
- Compositional Analysis
  - Overview
  - Real-Time Calculus
- Examples
  - Shapers
  - Artificial Example
  - Shared Resources in Multicore Systems
- Extensions
- ***Comparison***
- Challenges

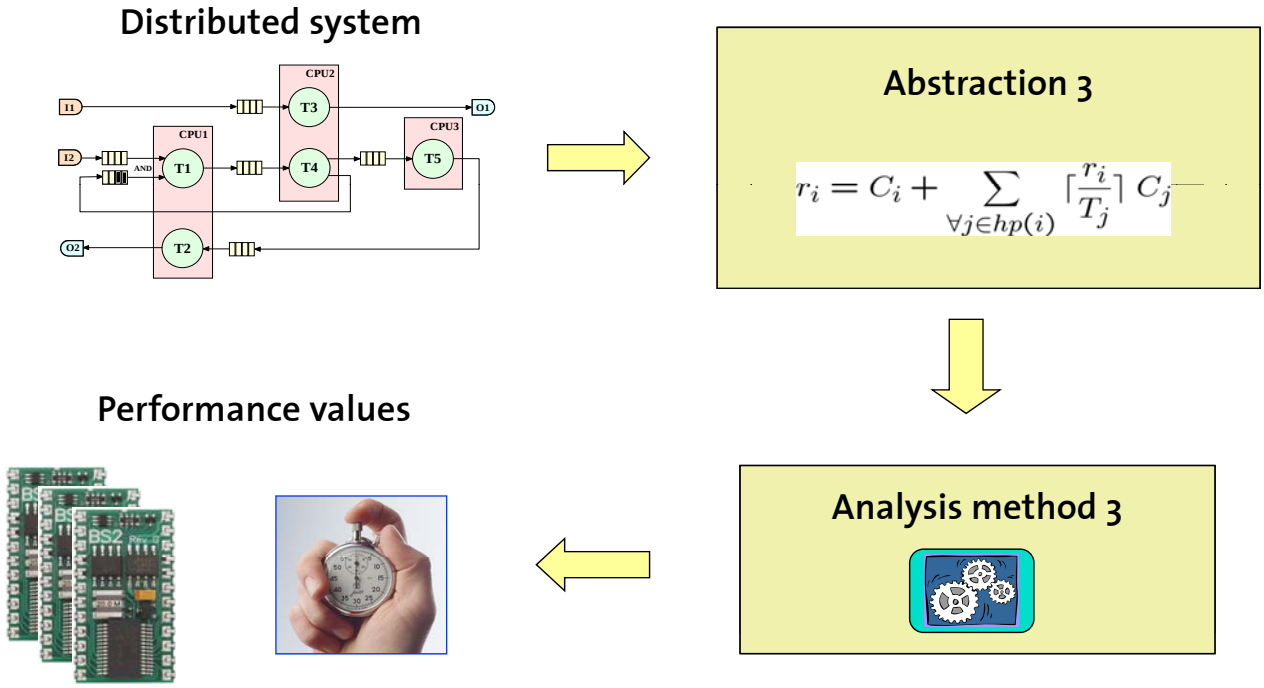
Compositional Analysis

- Comparsion-

System Level Performance Analysis



Formal analysis methods

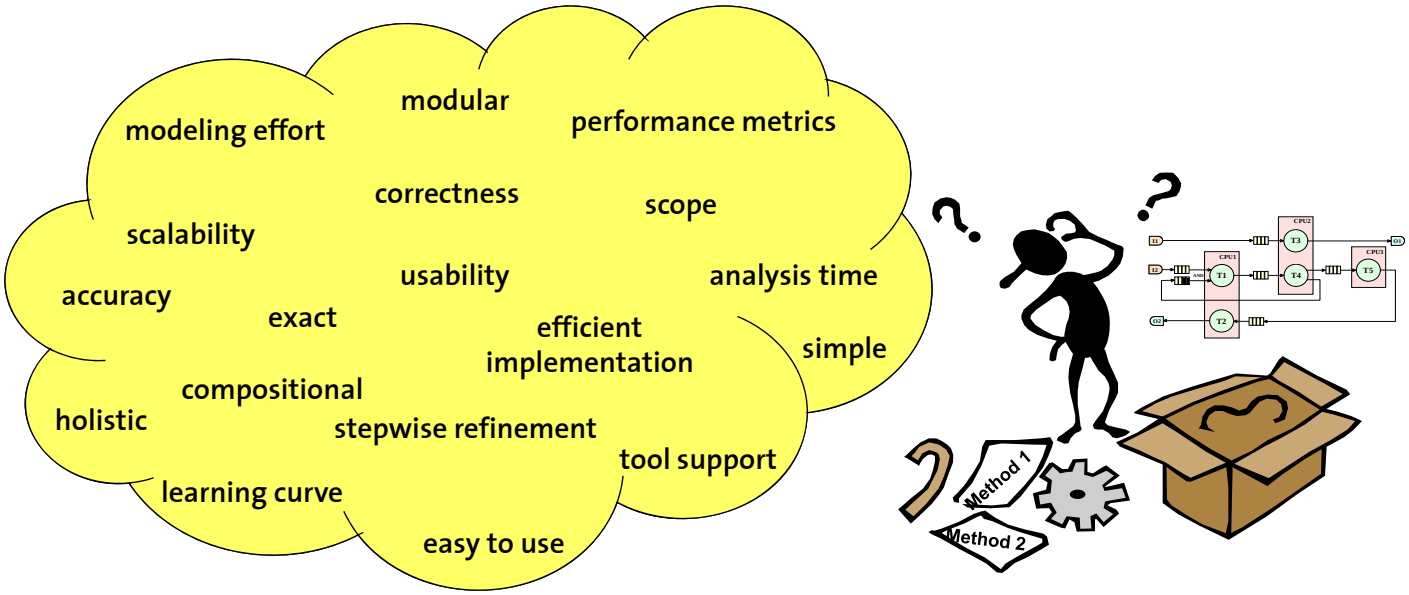


Motivating Questions

- What is the influence of the different models on the analysis accuracy ?
- Does abstraction matter ?
- Which abstraction is best suited for a given system ?

Evaluation and comparison of abstractions is needed !

How Can We Compare?



Intention

Compare ***models and methods*** that analyze the timing properties of distributed systems:

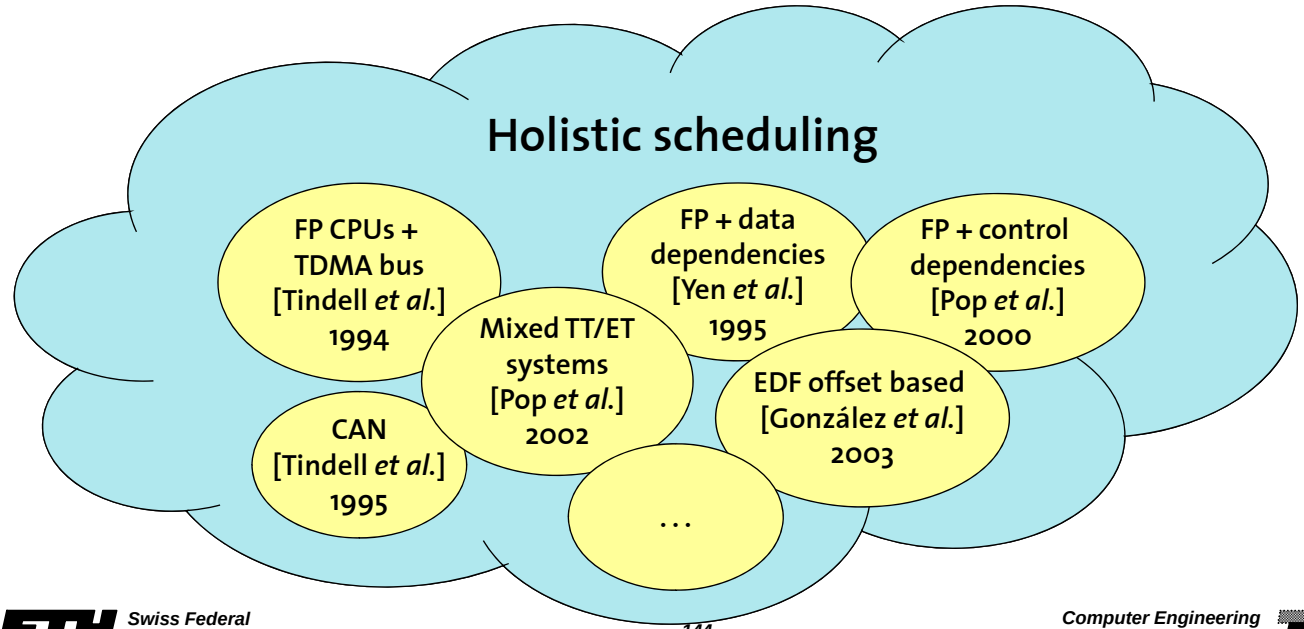
- SymTA/S [Richter *et al.*]
- MPA-RTC [Thiele *et al.*]
- MAST [González Harbour *et al.*]
- Timed automata based analysis [Yi *et al.*]
- …

Contributions

- We define a set of ***benchmark systems*** aimed at the evaluation of performance analysis techniques
- We apply different analysis methods to the benchmark systems and compare the results obtained in terms of ***accuracy*** and ***analysis times***
- We point out several analysis difficulties and investigate the causes for deviating results

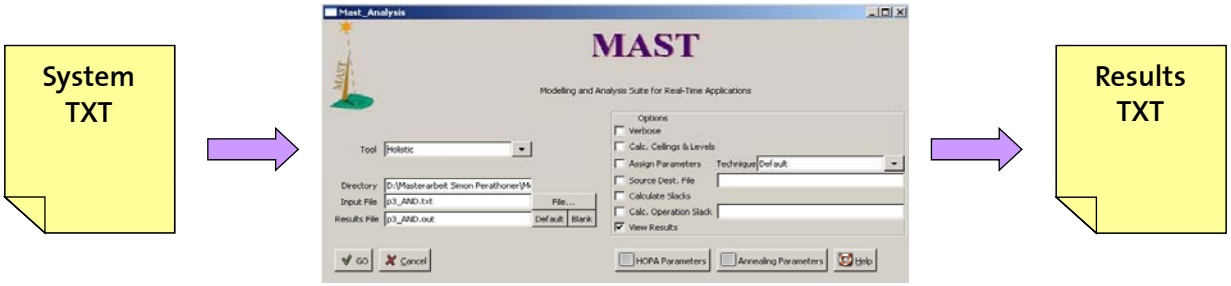
Abstraction 1 - Holistic Scheduling

***Basic concept*: extend concepts of classical scheduling theory to distributed systems**



Holistic Scheduling – MAST tool

MAST - The Modeling and Analysis Suite for Real-Time Applications [González Harbour *et al.*]



Abstraction 2 – The SymTA/S Approach

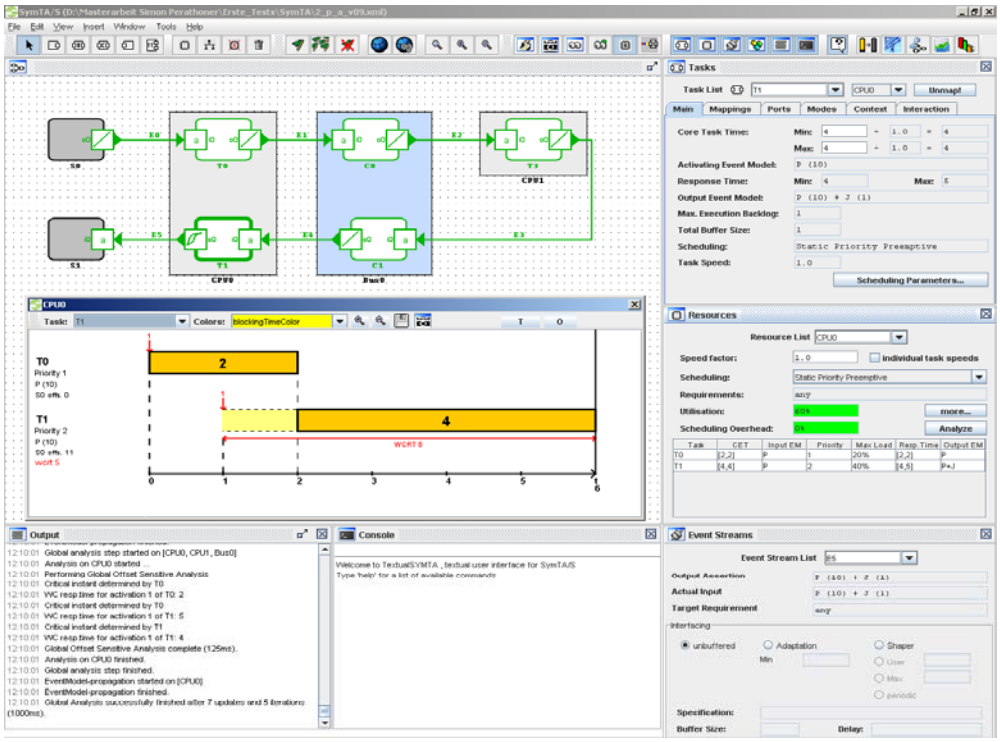
Basic concept: **Application of classical scheduling techniques at resource level and propagation of results to next component**

Problem: **The local analysis techniques require the input event streams to fit given standard event models**

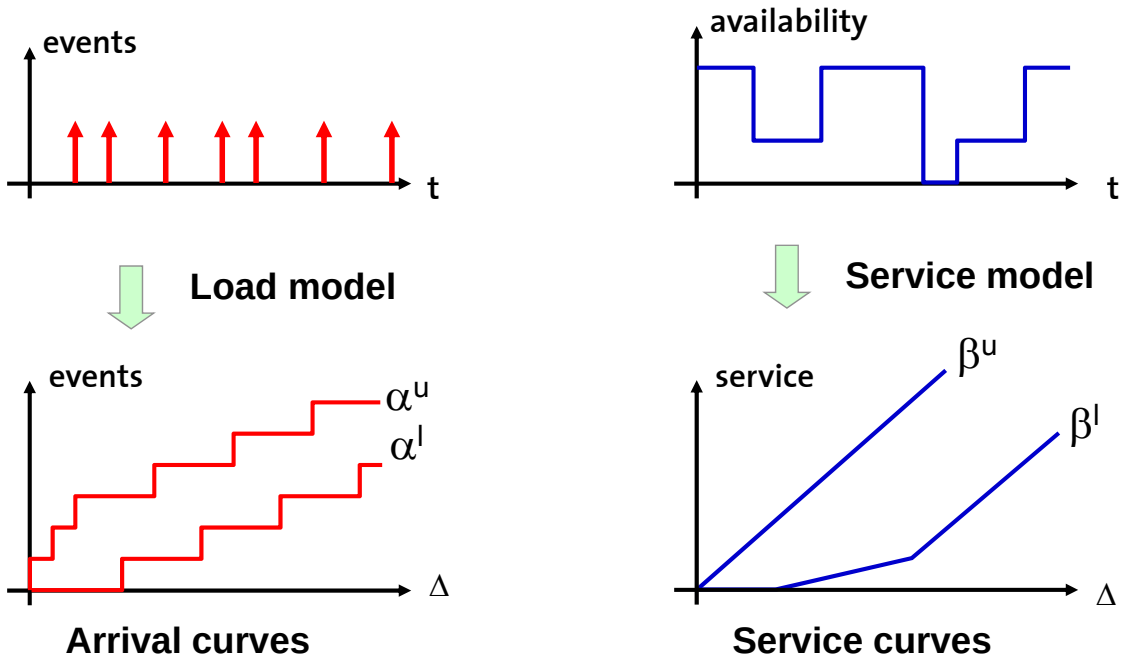
Solution: **Use appropriate interfaces: EMIFs & EAFs**

SymTA/S – Tool

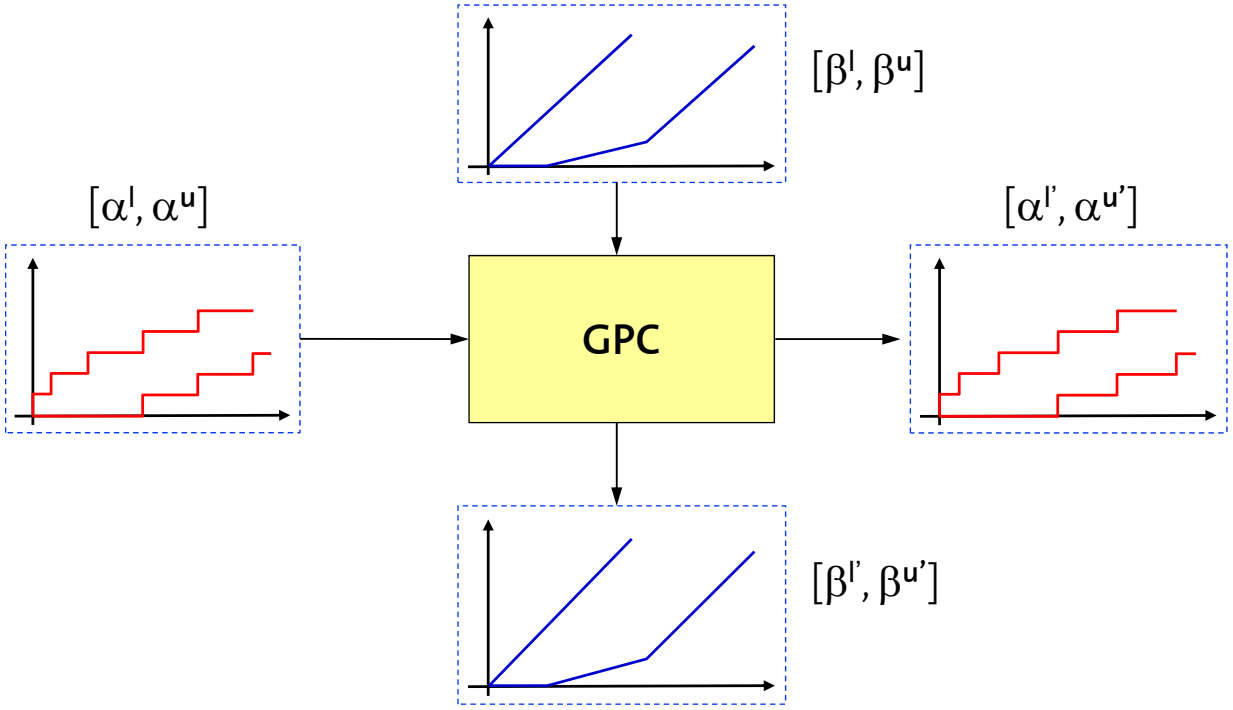
SYMTA VISION



Abstraction 3 – MPA-RTC



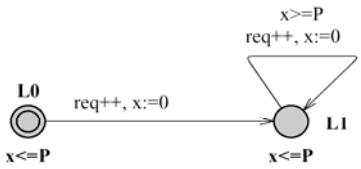
Abstraction 3 – MPA-RTC



Abstraction 4 – Timed Automata

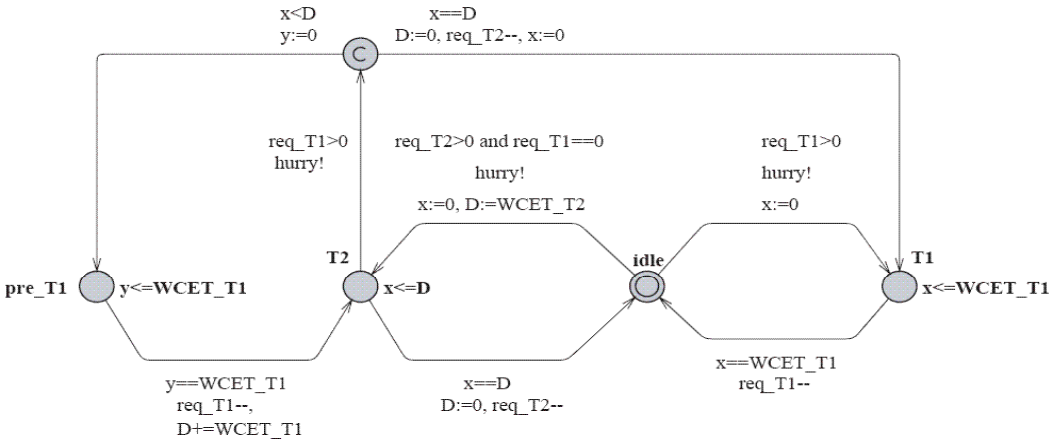
Verification of performance properties by model checking (UPPAAL)

Exact performance values

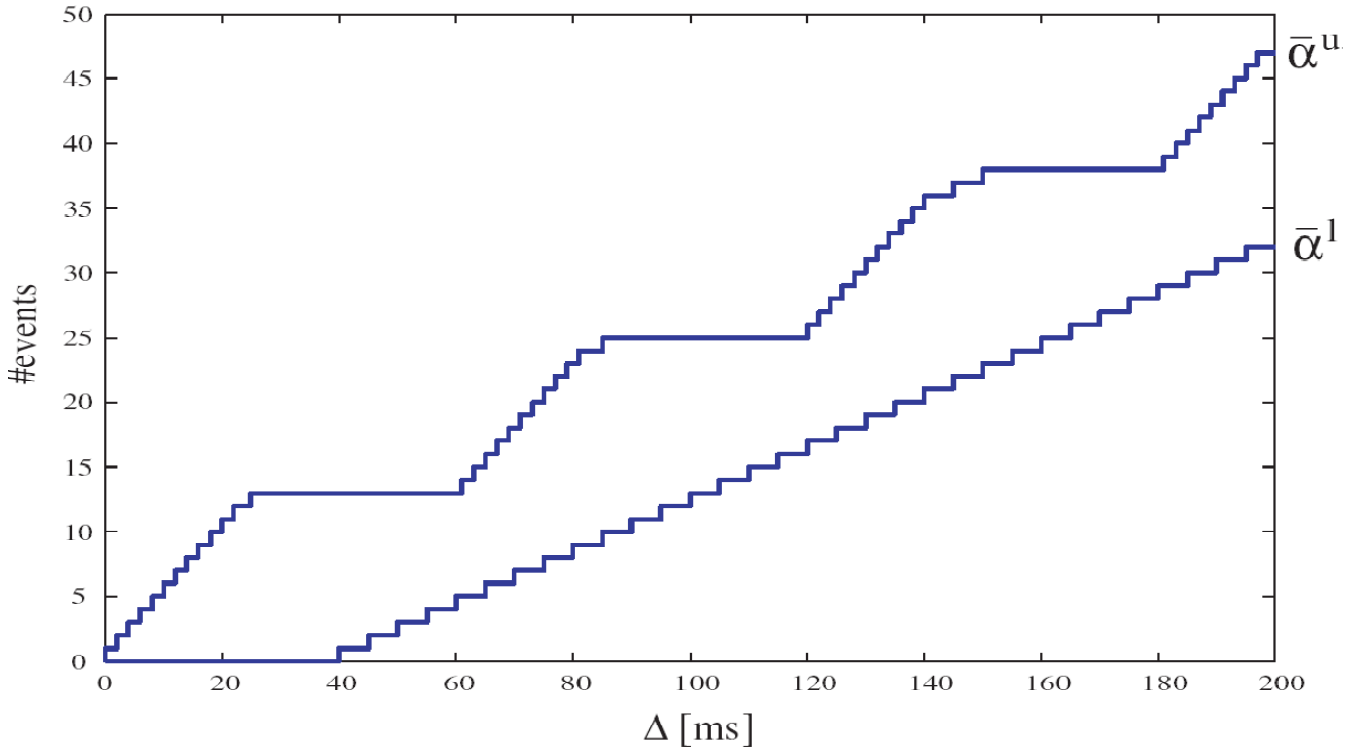


periodic stream

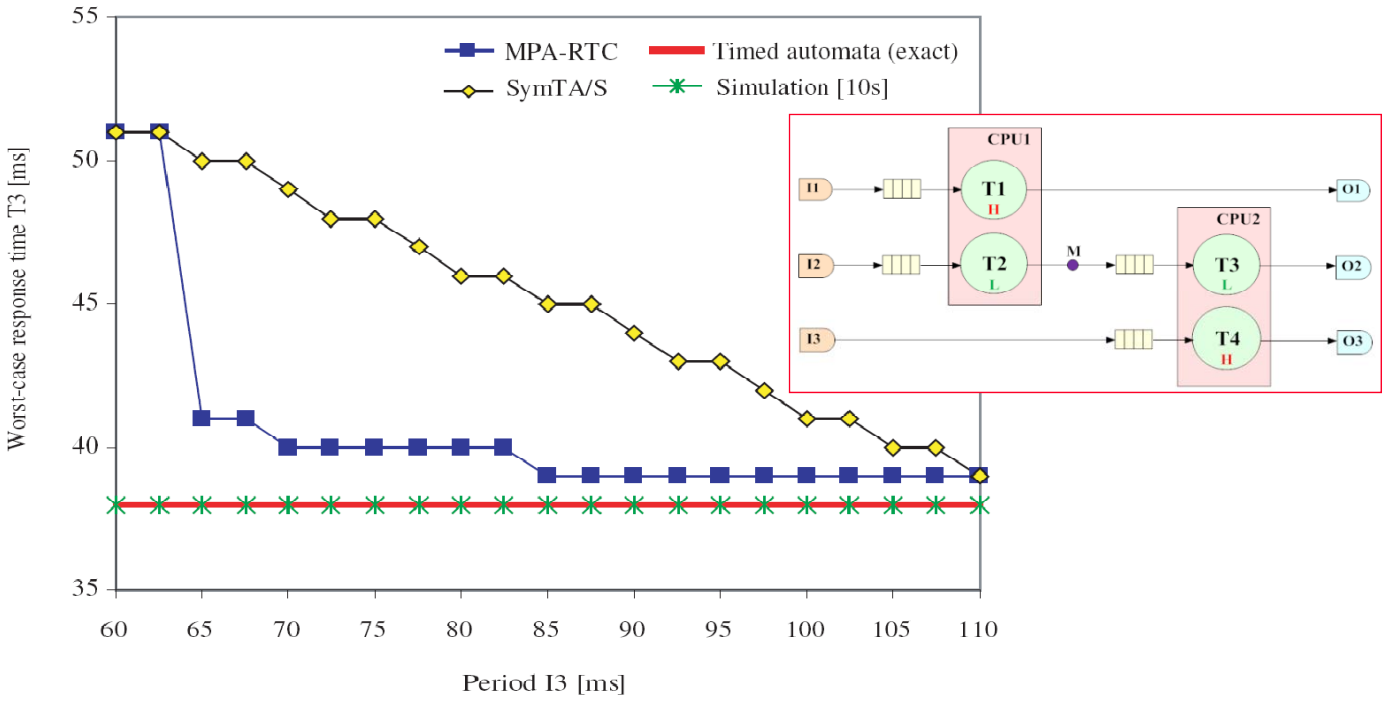
fixed priority scheduling



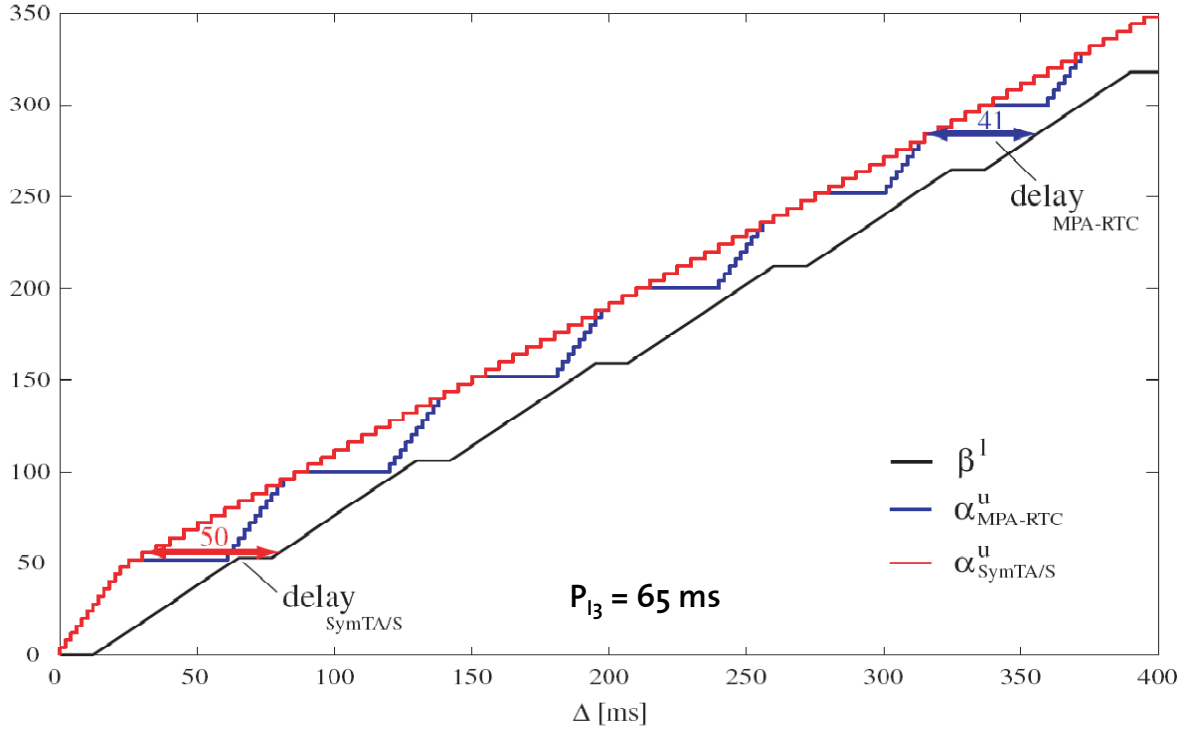
Benchmark 1 – Complex Activation



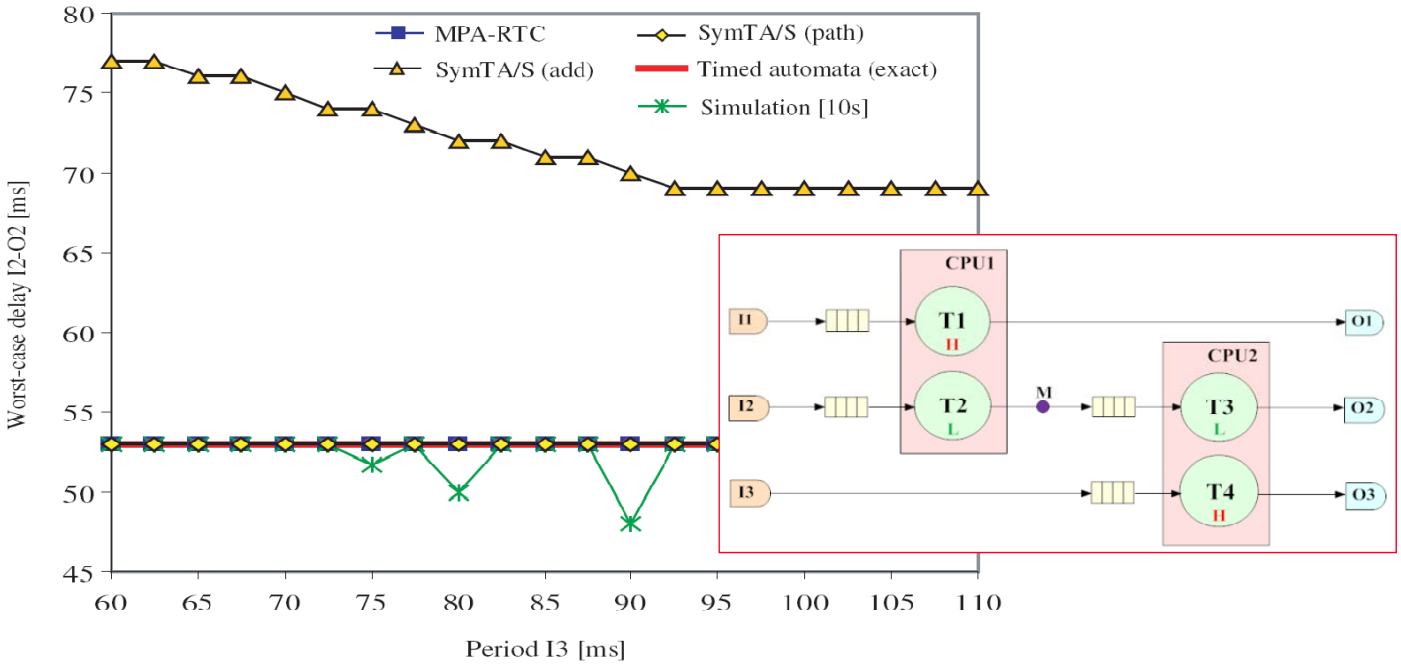
Benchmark 1 – Analysis Results



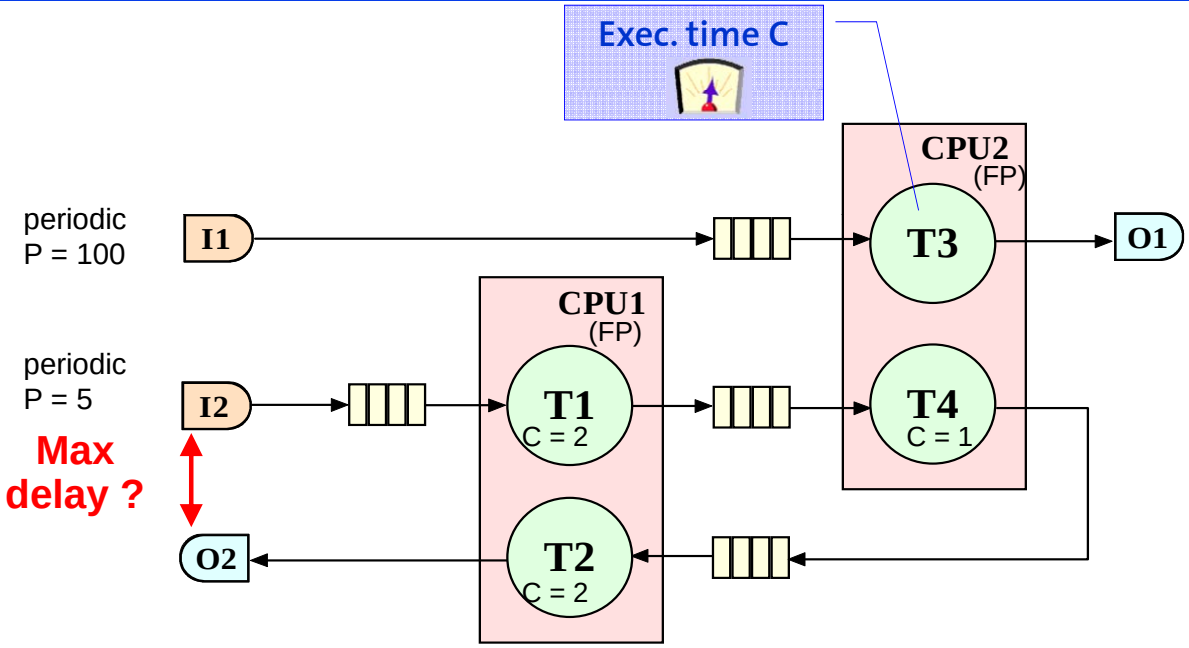
Benchmark 1 – Result Interpretation



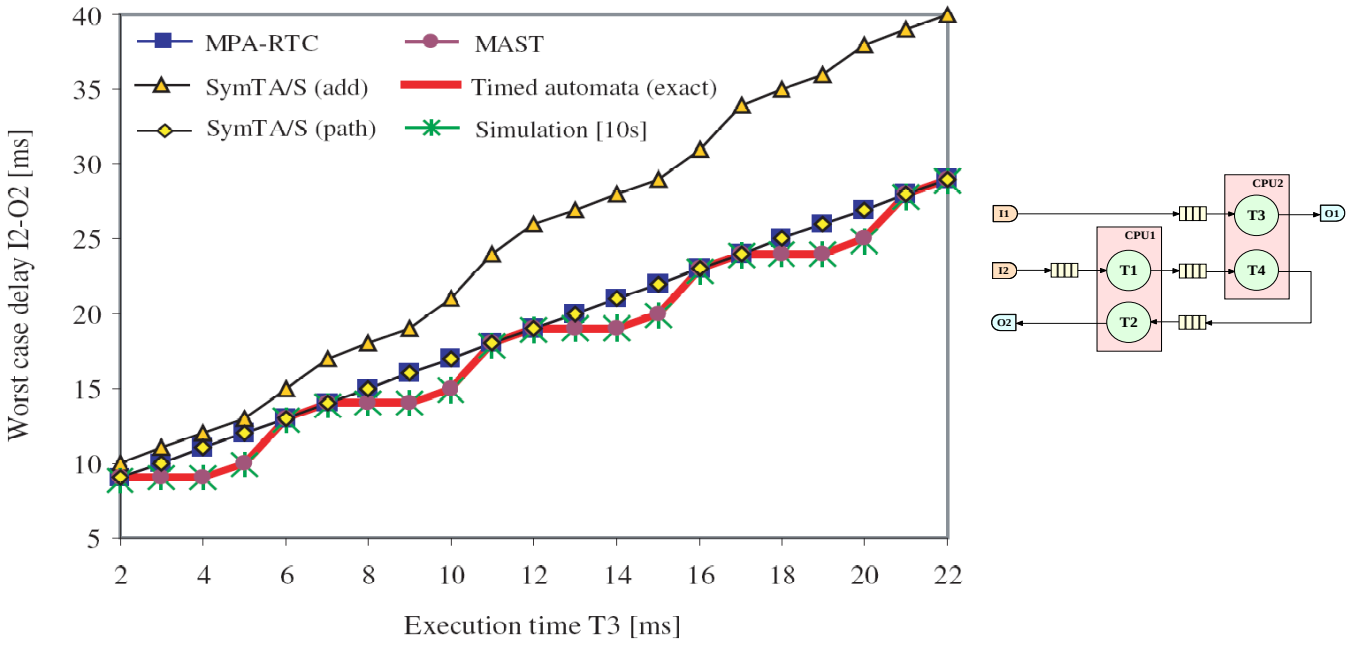
Benchmark 1 – Worst Case Delay I2-O2



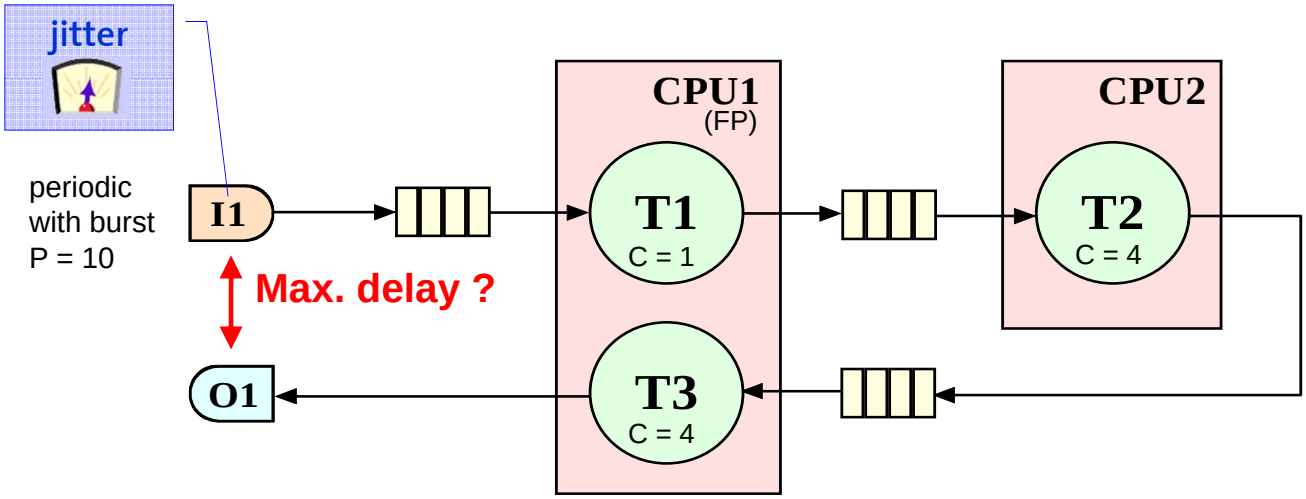
Benchmark 2 – Variable Feedback



Benchmark 2 – Analysis Results

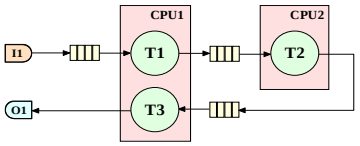


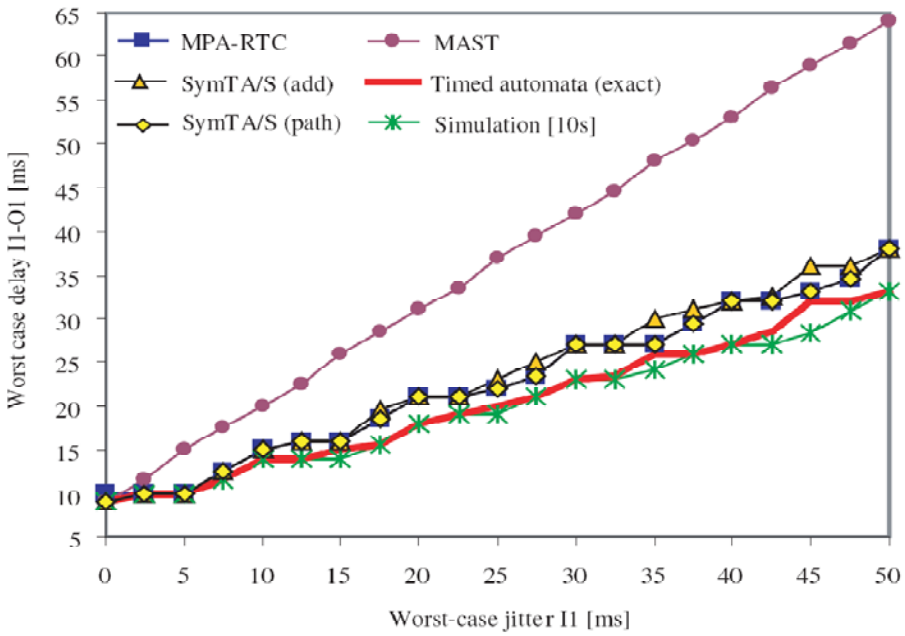
Benchmark 3 – Cyclic Dependencies



Benchmark 3 – Analysis Results

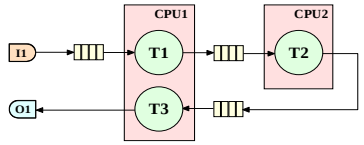
Scenario 1: priority T1 = high
priority T3 = low

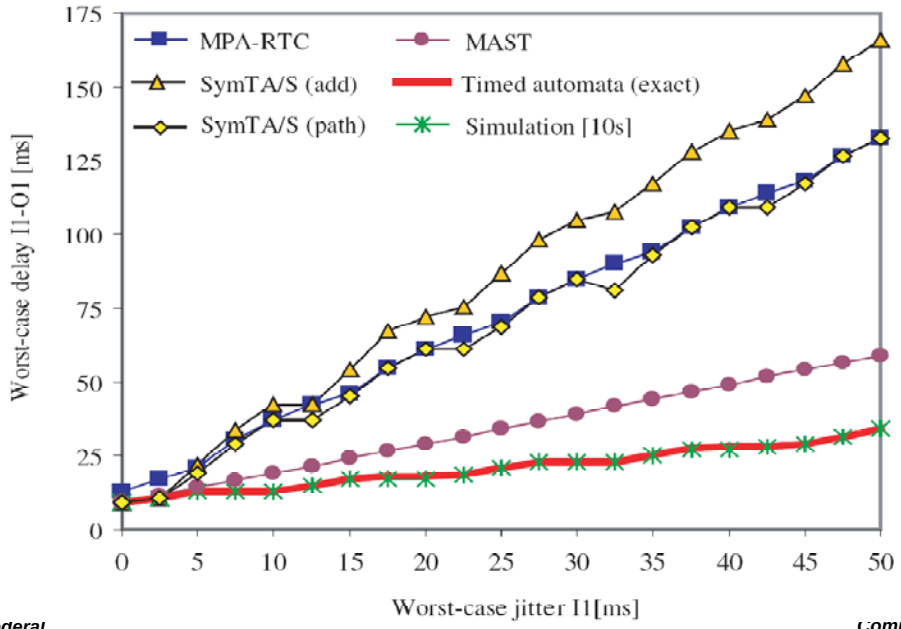




Benchmark 3 – Analysis Results

Scenario 2: priority T1 = low
priority T3 = high





Analysis Times [s]

		B1	B2	B3 (sc.1)	B3 (sc.2)	B4
MPA-RTC	min	0.60	0.03	0.01	0.04	0.03
	med	1.06	0.04	0.01	0.15	0.05
	max	1.97	0.08	0.04	0.30	0.20
SymTA/S	min	0.05	0.03	0.03	0.03	0.06
	med	0.09	0.05	0.06	0.34	0.09
	max	1.50	0.23	0.09	0.80	0.31
MAST	min	-	< 0.5	< 0.5	< 0.5	< 0.5
	med	-	< 0.5	< 0.5	< 0.5	< 0.5
	max	-	< 0.5	< 0.5	< 0.5	< 0.5
Timed aut.	min	18.0	< 0.5	< 0.5	< 0.5	< 0.5
	med	34.5	< 0.5	1.0	< 0.5	< 0.5
	max	60.5	< 0.5	52.0	5.5	< 0.5
Simulation	min	1.0	< 0.5	0.5	0.5	< 0.5
	med	1.0	< 0.5	0.5	0.5	< 0.5
	max	1.0	< 0.5	0.5	0.5	< 0.5

Conclusions

- The ***analysis accuracy*** and the analysis time ***depend highly on the specific system characteristics***.
- The analysis results of the different approaches are ***remarkable different*** even for apparently simple systems.
- The choice of an appropriate analysis ***abstraction matters***.
- The problem to provide accurate performance predictions for general systems is still ***far from solved***.

Contents

- Drivers
- Compositional Analysis
  - Overview
  - Real-Time Calculus
- Examples
  - Shapers
  - Artificial Example
  - Shared Resources in Multicore Systems
- Extensions
- Comparison
- ***Challenges***

Challenges

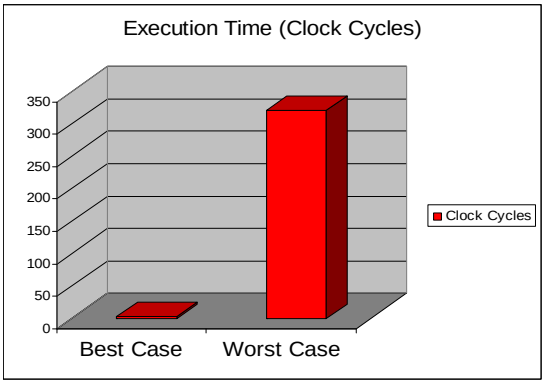
WCET

x = a + b; →

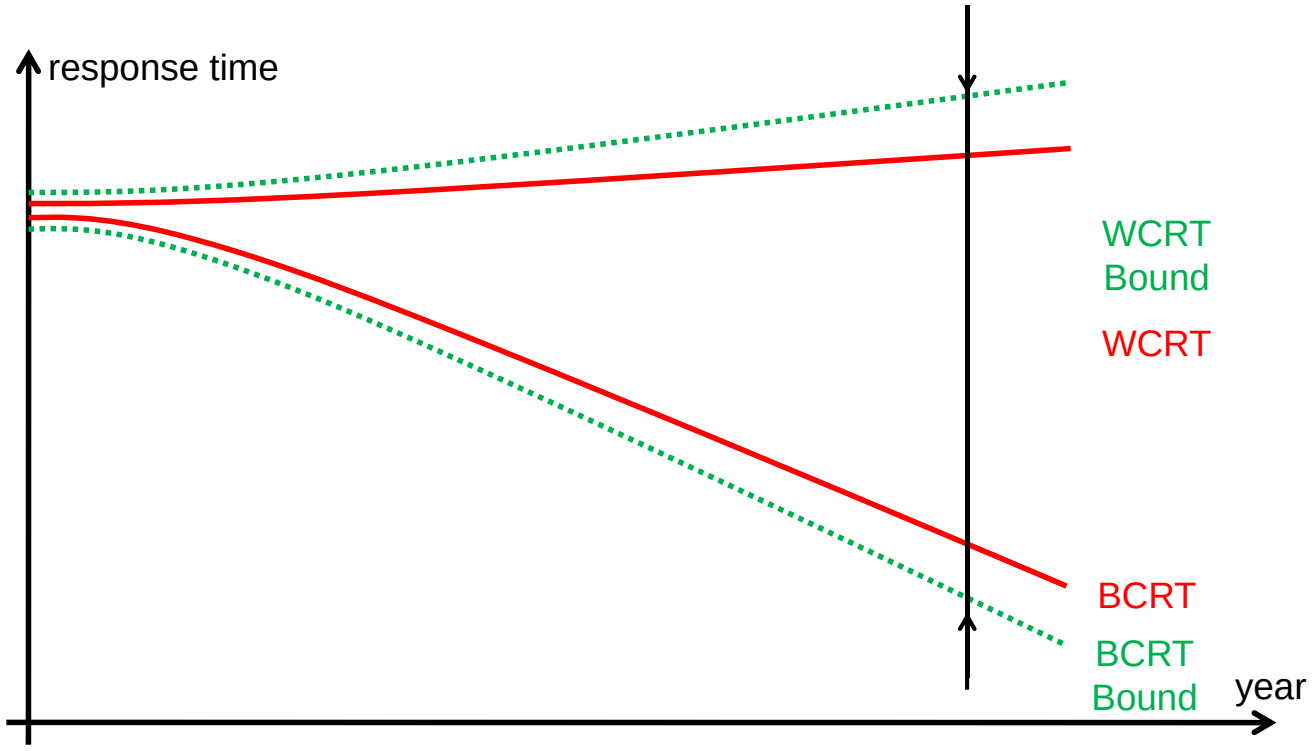
```
LOAD    r2,  a
LOAD    r1, _b
ADD     r3,r2,r1
```

↓

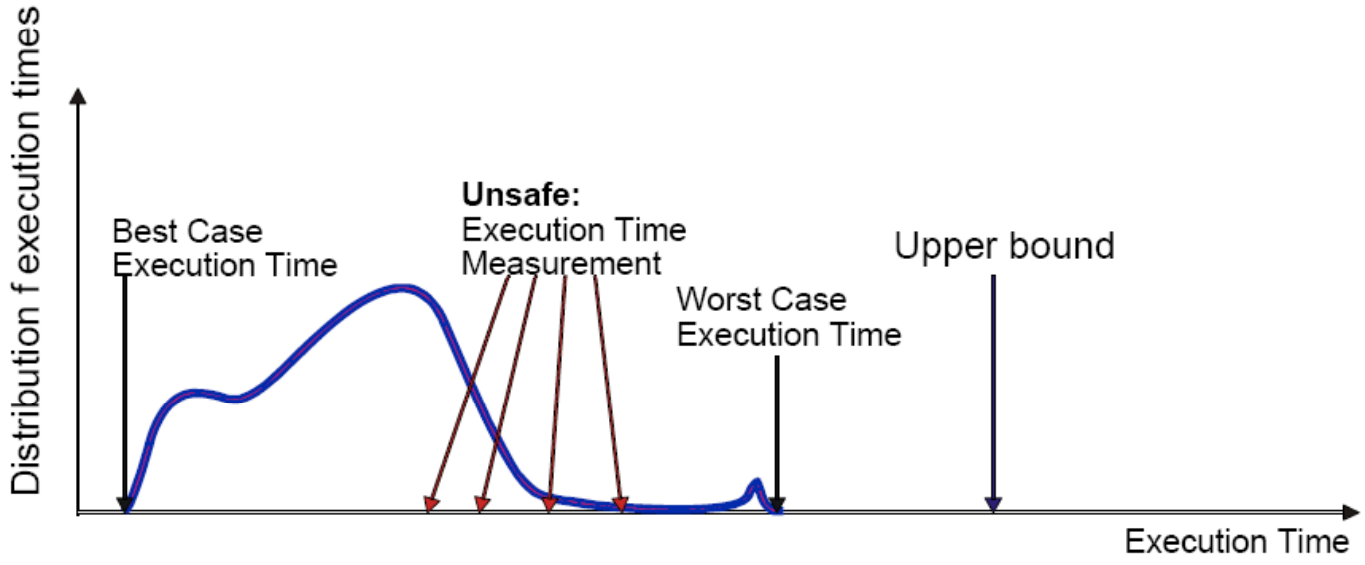
PPC 755



(Timing) Predictability

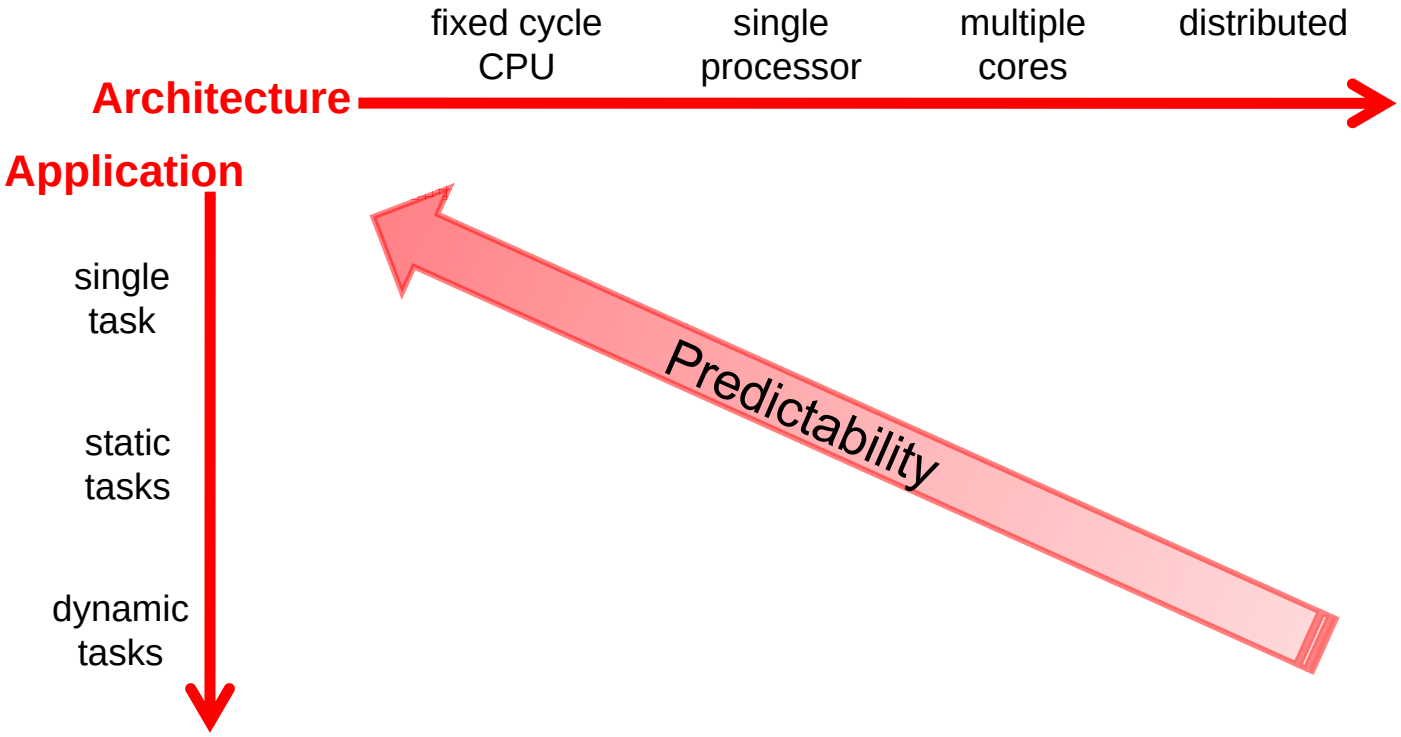


WCET

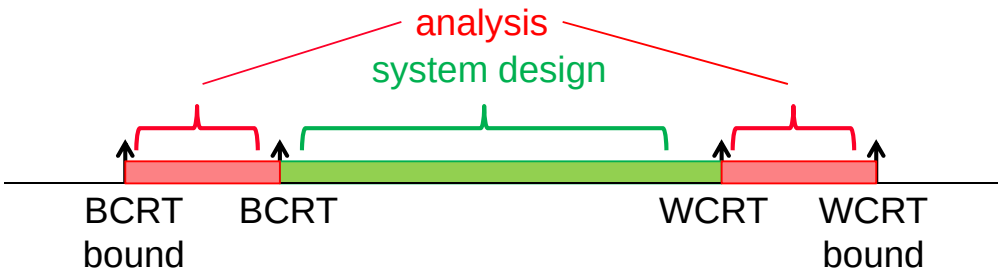


© Reinhard Wilhelm

Application and Architecture

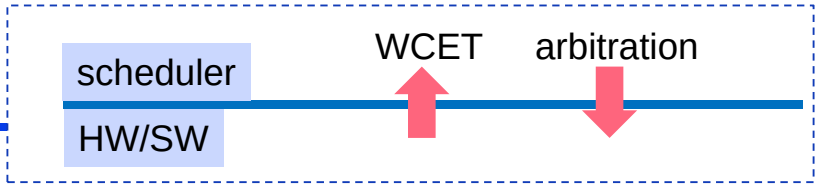


Classification of Predictability Loss



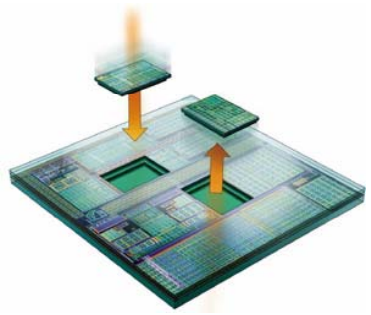
- *Analysis Loss:*
  - Construct system that can be easily analyzed
  - Use appropriate abstractions (models and methods)
- *System Design Loss:*
  - Decrease interference, long-range dependencies
  - Increase robustness of components
  - Use appropriate interfaces

Interfaces



- A task is (classically) characterized by its WCET.
  - May be useful in case of simple processors, but we have long-range state-dependent uni-processor behavior (pipelines, caches, speculation).
  - In case of multi-processors, we have additional interferences on the communication system which heavily influences WCET. We also may have intra-task parallelism.
  - WCET can no longer be considered as a useful interface between these abstraction layers.
- What about the other interfaces ?
  - Is the classical ISA (using instructions that abstract away time) still appropriate?

Acknowledgement



- ***Co-workers:***
  Jan Beutel, Jian-Jia Chen, Iuliana Bacivarov, Kai Lampka, Clemens Moser, Wolfgang Haid, Ernesto Wandeler, Simon Perathoner, Nikolay Stoimenov, Kai Huang, …

- ***Funding:***
  EU-SHAPES, EU-PREDATOR, EU-COMBEST, EU-ARTISTDESIGN, EU-EURETILE, EU-PRO3D, IBM, Siemens, NCCR-MICS, KTI

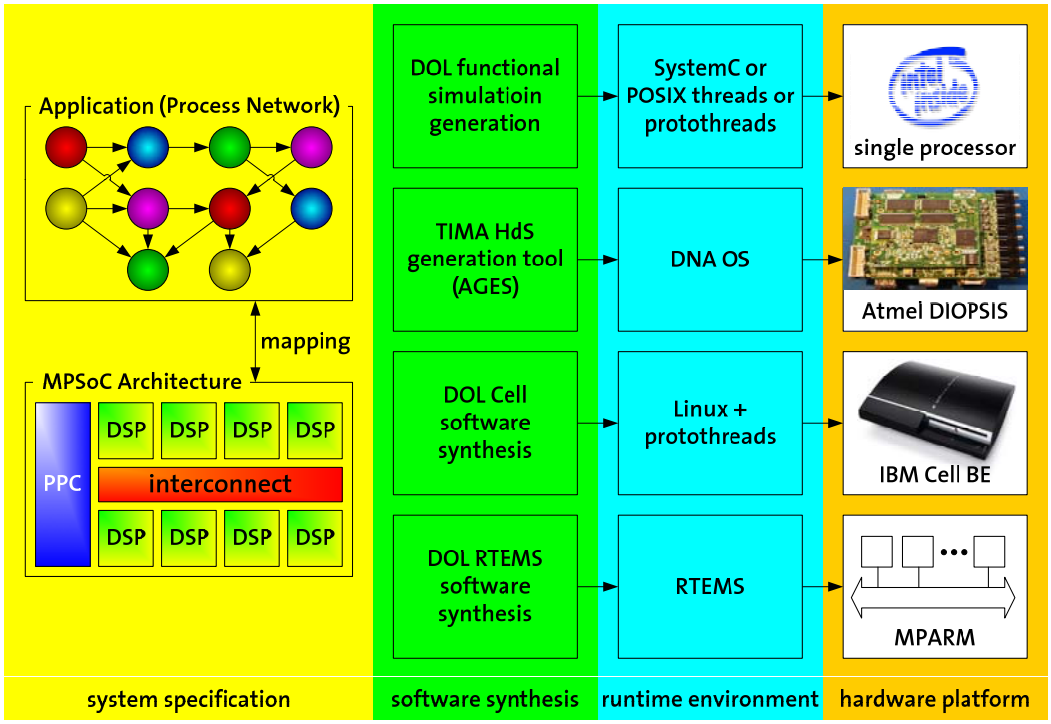
Performance Analysis of Distributed Embedded Systems

Part 2: MPSoC Software Design

© Lothar Thiele
Iuliana Bacivarov, Wolfgang Haid, Kai Huang
http://www.tik.ee.ethz.ch/~shapes/

Modular Performance Analysis for MPSoC Design

Versatile MPSoC Software Design Flow



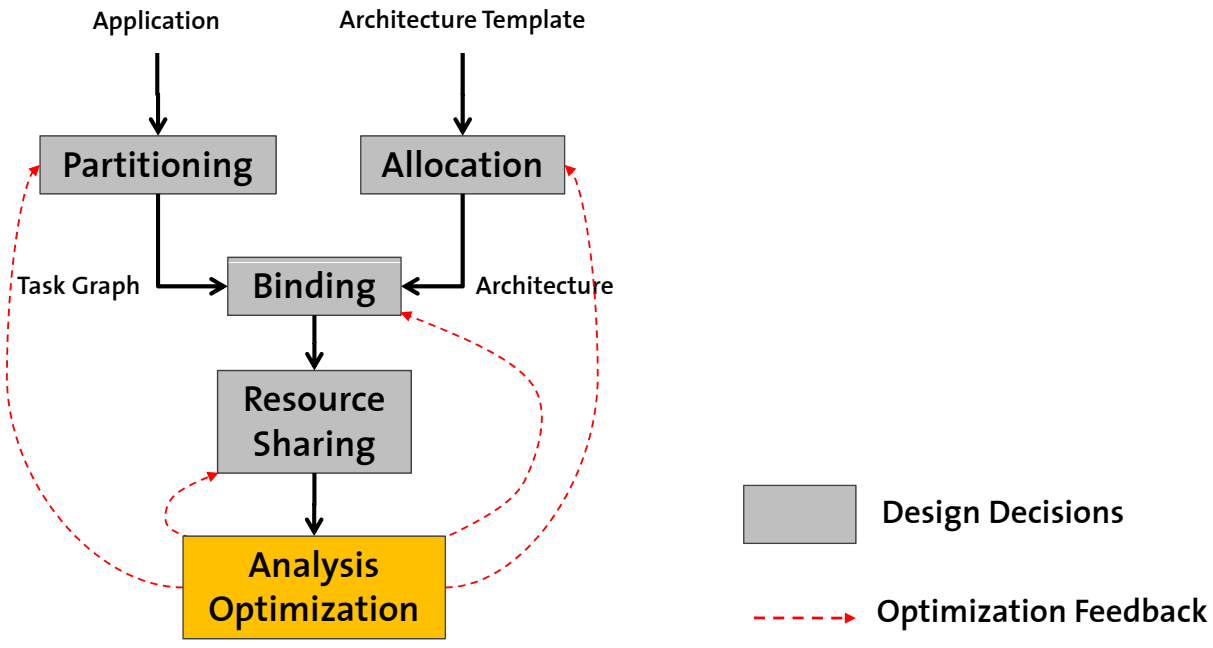
Some Challenges in MPSoC Programming

- Design Process
- Programming Model
- Optimization
- Scalability
- Calibration

Some Challenges in MPSoC Programming

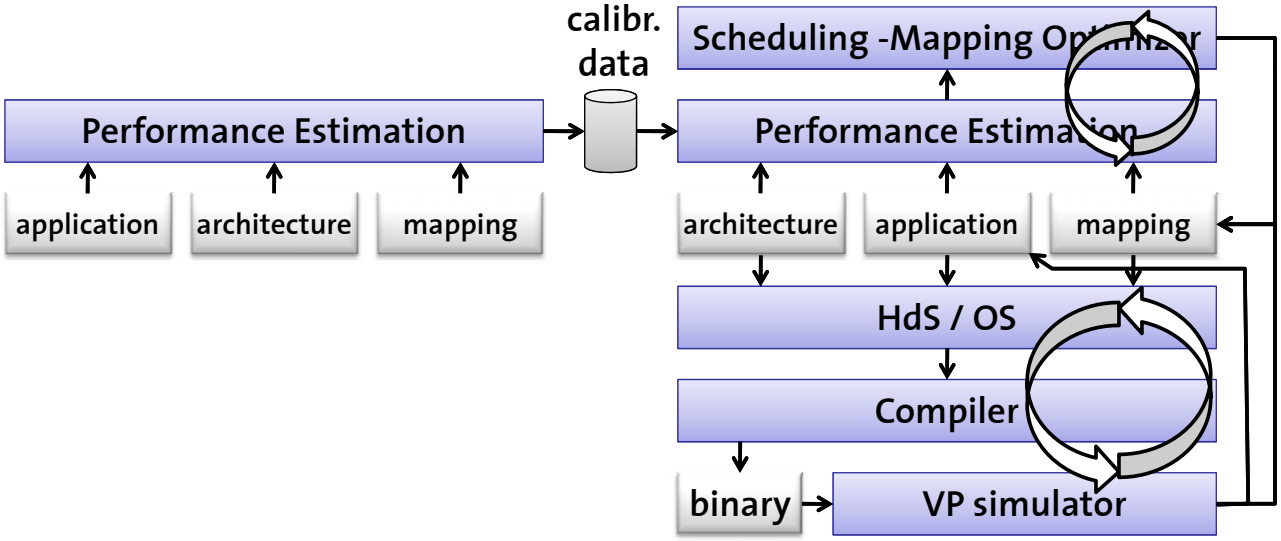
- **Design Process**
- Programming Model
- Optimization
- Scalability
- Calibration
- **DOL (Distributed Operation Layer)**

Design Exploration



DOL Design Flow

Correctness | Scalability | Optimality | Accuracy



Some Challenges in MPSoC Programming

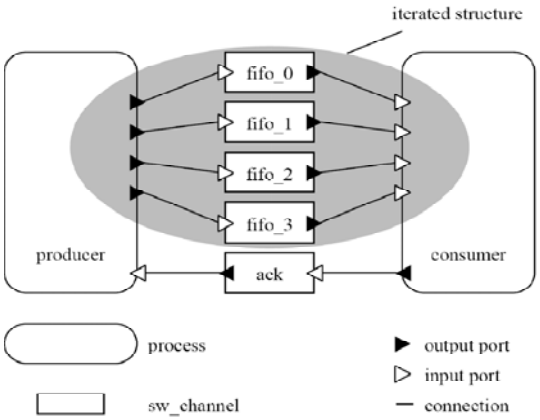
- Design Process
- **Programming Model**
- Optimization
- Calibration
- Scalability

- DOL (**D**istributed **O**peration **L**ayer)
- **Process networks and explicit communication**

Application Specification

Structure

- **Process Network**
  - Processes
  - SW channels (FIFO behavior)
- **Iterators**
  - Scalability for processes, SW channels, entire structures



Functional specification

- **Language:** C/C++
- **API:** DOL primitives

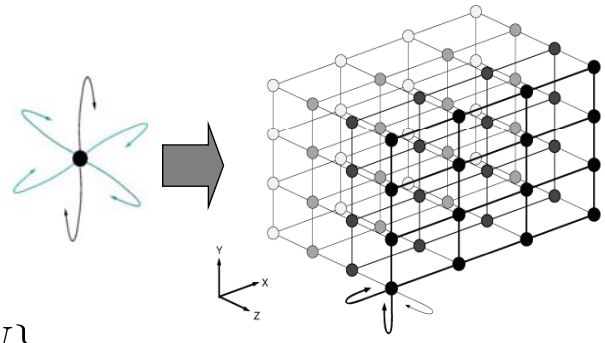
Algorithm 1 Process Model

```
1: procedure INIT(DOLProcess p)                ▷ initialization
2:     initialize local data structures
3: end procedure

4: procedure FIRE(DOLProcess p)                ▷ execution
5:     DOL_read(INPUT, size, buf)              ▷ blocking read
6:     manipulate
7:     DOL_write(OUTPUT, size, buf)            ▷ blocking write
8: end procedure
```


Scalability at Specification Level

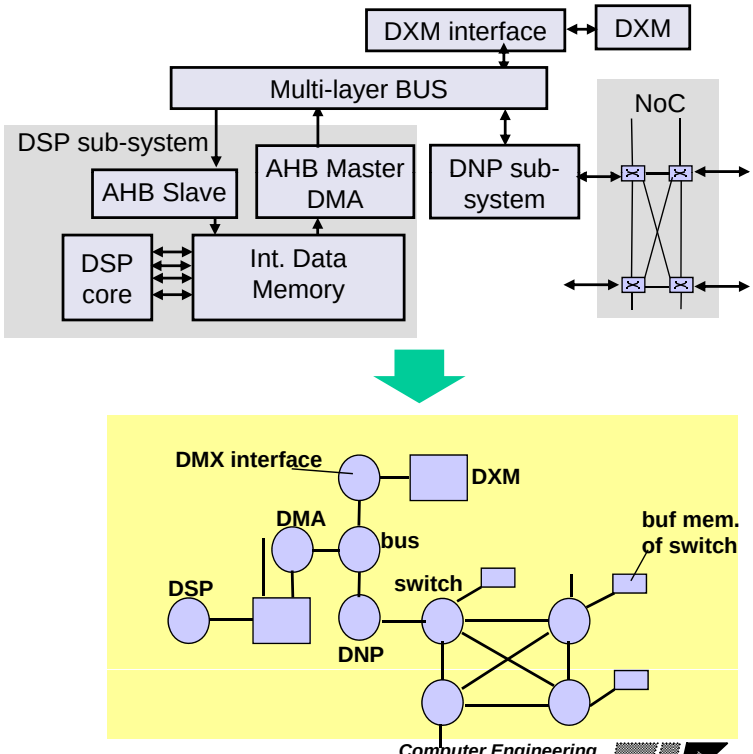
- Separation of instruction/thread level parallelism (inside processes) and process-level parallelism.
- Use of iterators in
  - architecture specification
  - application specification
  - mapping specification



$$\{(i,j) : 1 \leq i \leq N \wedge i \leq j \leq N\}$$

Target Platform Abstraction (1)

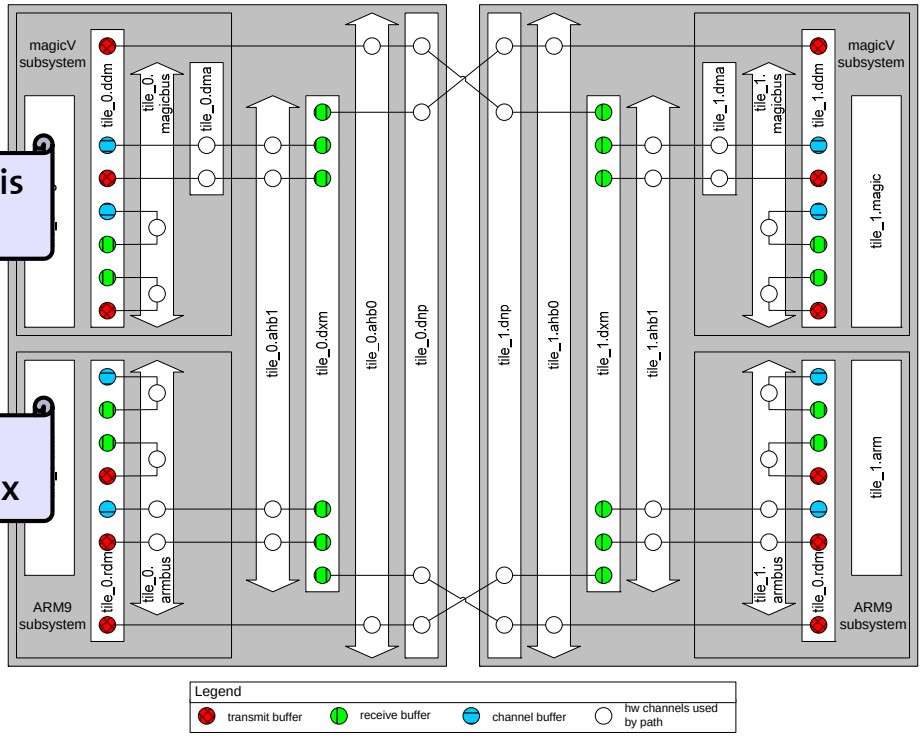
- *Topology modeled by a graph*
  - two node types:
    - execution and comm. resources
    - storage resources
- *Execution resources*
  - RISCs, DSPs, ...
- *Communication resources*
  - buses, switches, links, I/Os
- *Storage resources*
  - RAMs, HW FIFOs, ...



Target Platform Abstraction (2)

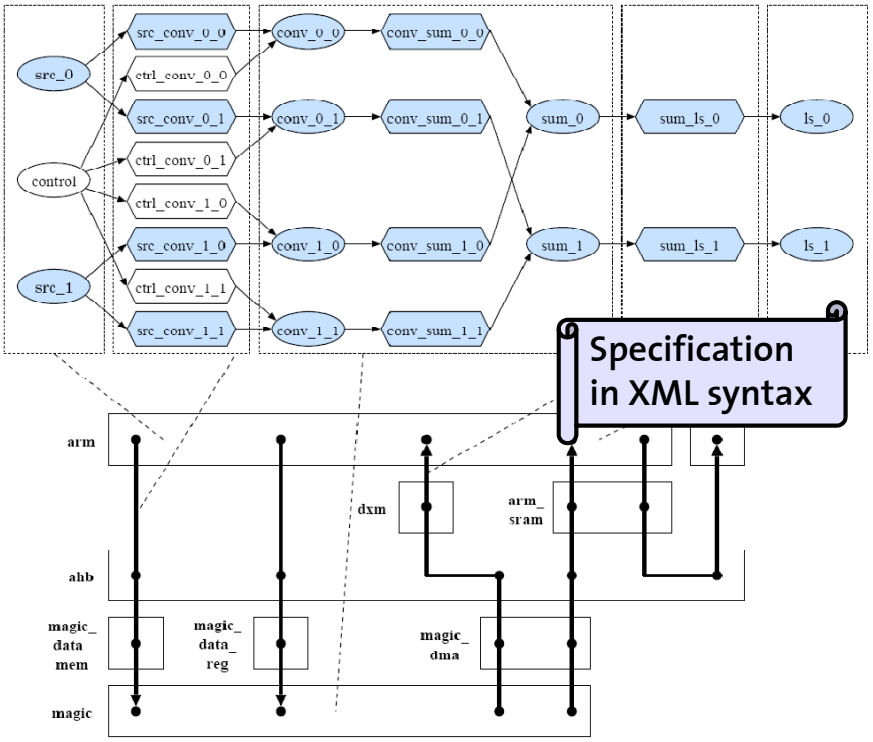
ATMEL Diopsis Platform

Specification in XML syntax



Mapping Specification

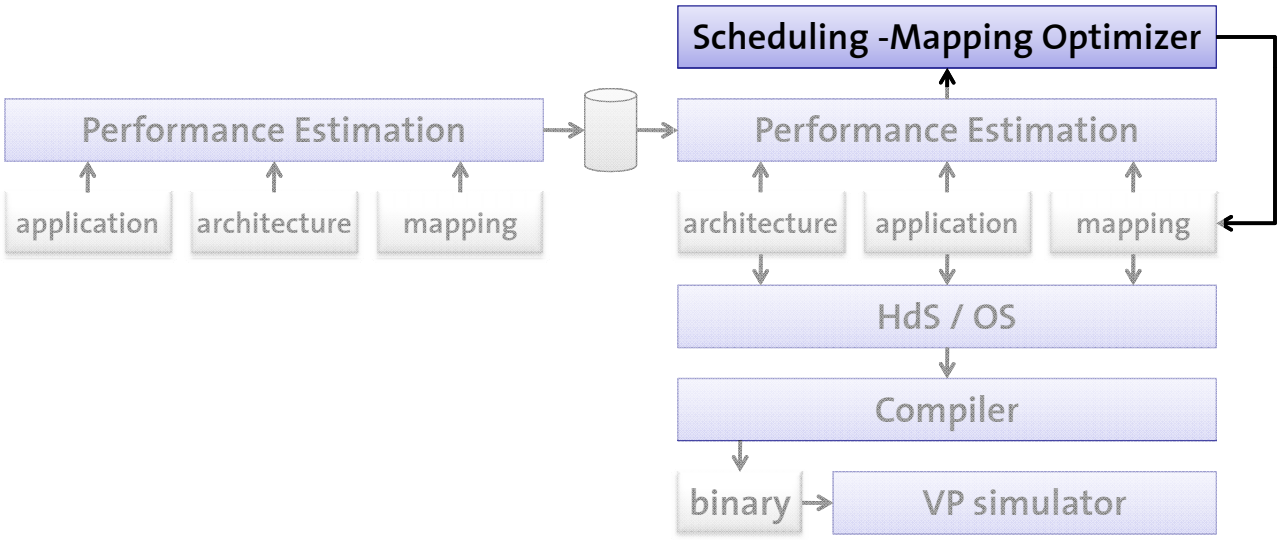
- **Binding**
  - Processes to execution resources
  - SW channels to read/write paths
- **Scheduling**
  - Processors
  - Communication
- **Constraints**
  - For Hardware-dependent Software (HdS) generation



Some Challenges in MPSoC Programming

- Design Process
- Programming Model
- **Optimization**
- Scalability
- Calibration
- DOL (**D**istributed **O**peration **L**ayer)
- Process networks and explicit communication
- **Hybrid black-box methods**

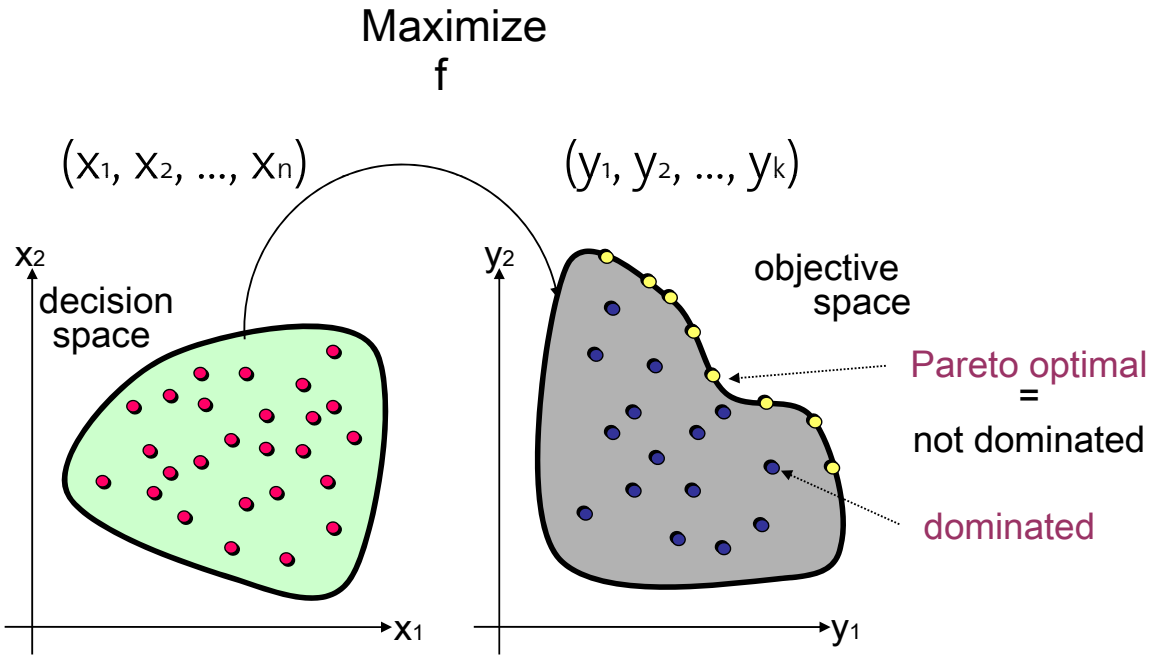
DOL Design Flow



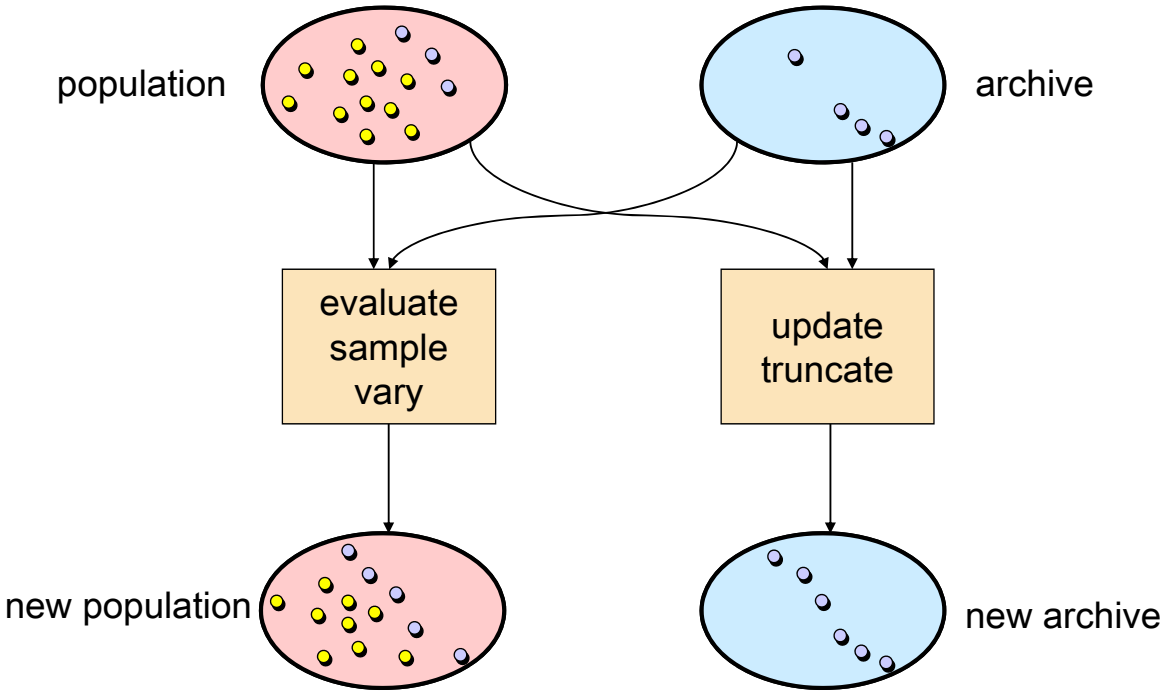
Optimization Criteria and Method

- ***Correctness:***
  - avoid memory/buffer overflow / underflow
  - respect mapping constraints
- ***Performance:***
  - end-to-end deadlines, throughput
  - jitter and burstiness
  - small sensitivity / large robustness
- ***Optimization Method:***
  - Population-based ***multi-objective optimization.***
  - Constraint handling embedded into optimizer
  - Exploration based on bottleneck and robustness information

Multiobjective Optimization

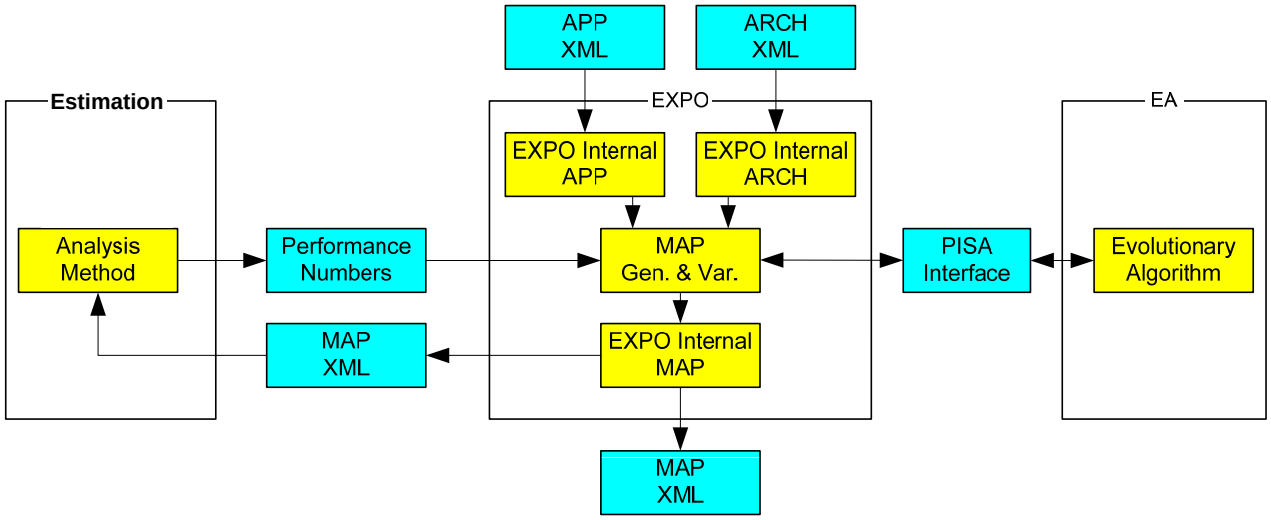


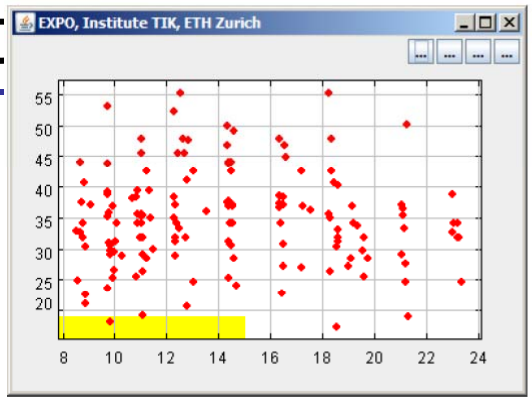
A Generic Multiobjective EA

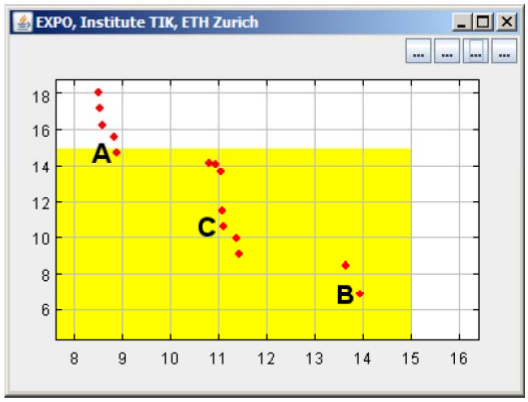


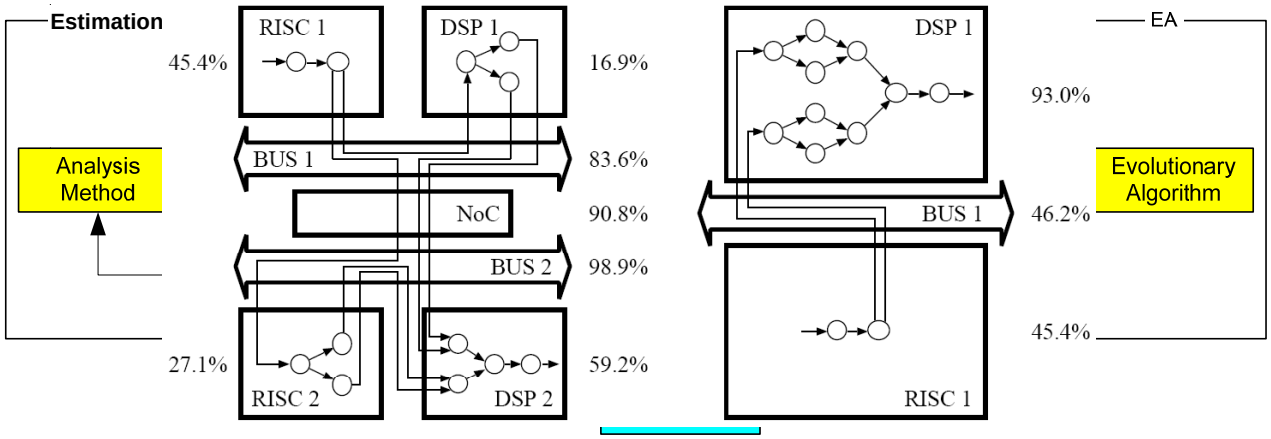
Design Space Exploration Framework

- **PISA&EXPO**
  - multi-objective optimization using evolutionary algorithms
  - [PISA] https://www.tik.ee.ethz.ch/pisa;
  - [EXPO] https://www.tik.ee.ethz.ch/expo

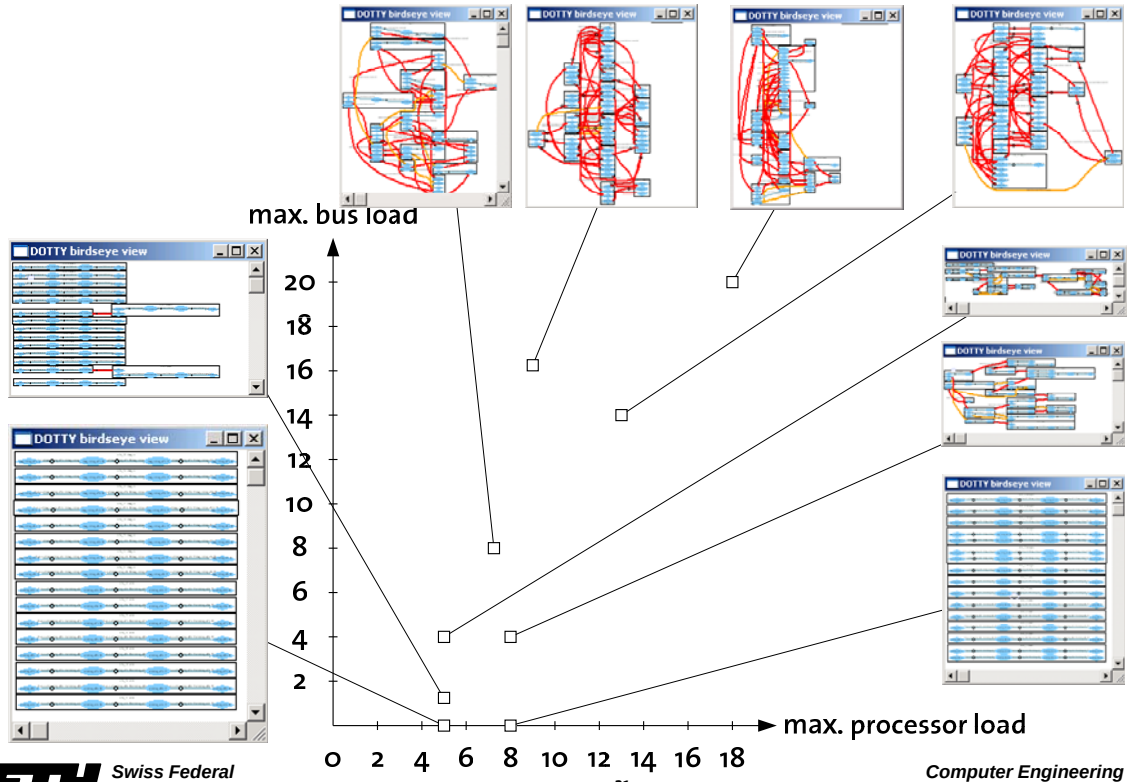








Design Space Exploration



Design Space Exploration

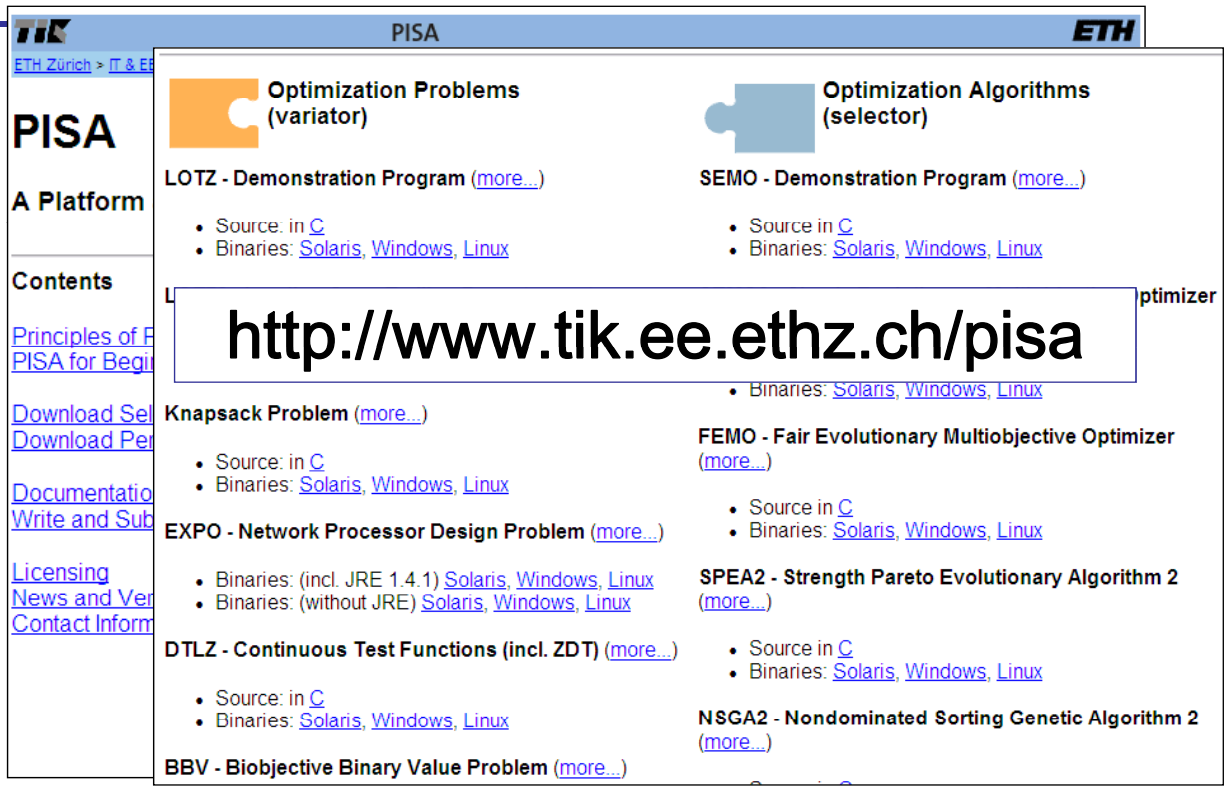
- **Example for ATMEL Multitile Platform**:
  - 64 processes, 16 processors, optimal mapping known
  - 32 processes execute efficiently on ARM, 32 efficiently on mAgic
  - different interconnection structures between processes
  - $16^{64} \approx 1.15 \cdot 10^{77}$ possible mappings (including symmetric ones)
  - Evaluation of 10.000 mappings

	naive	evolutionary algorithm	optimum
64 indep. tasks	7/0	4/0	4/0
16 4-stage pipelines	9/10	4/0	4/0
64-stage pipeline	9/14	4/4	4/2

max. processor load

max. interconnect load

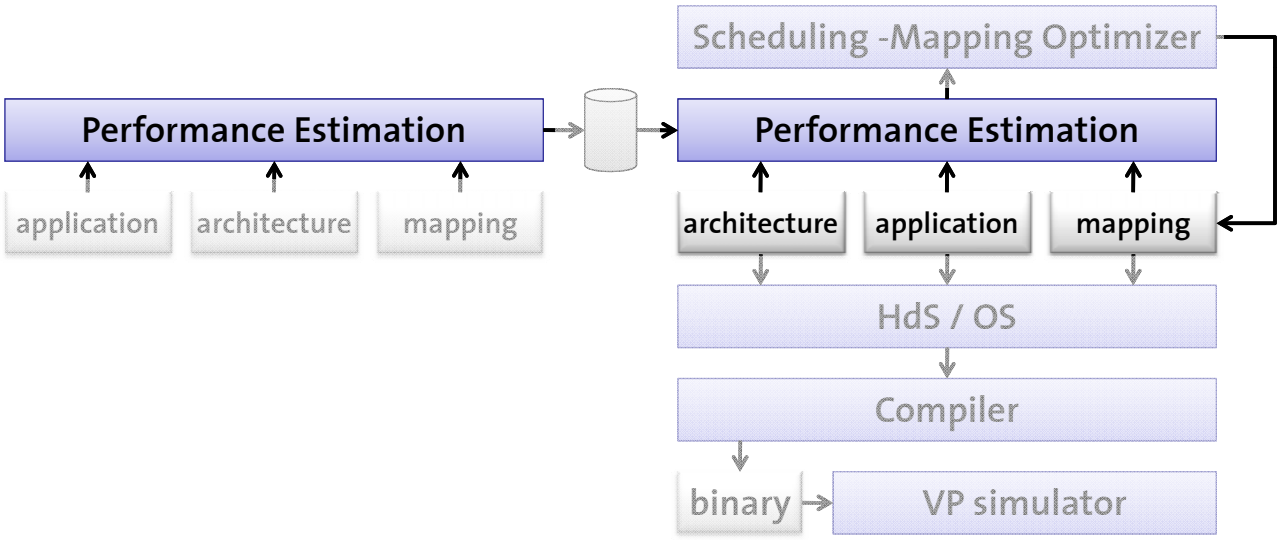
PISA Website



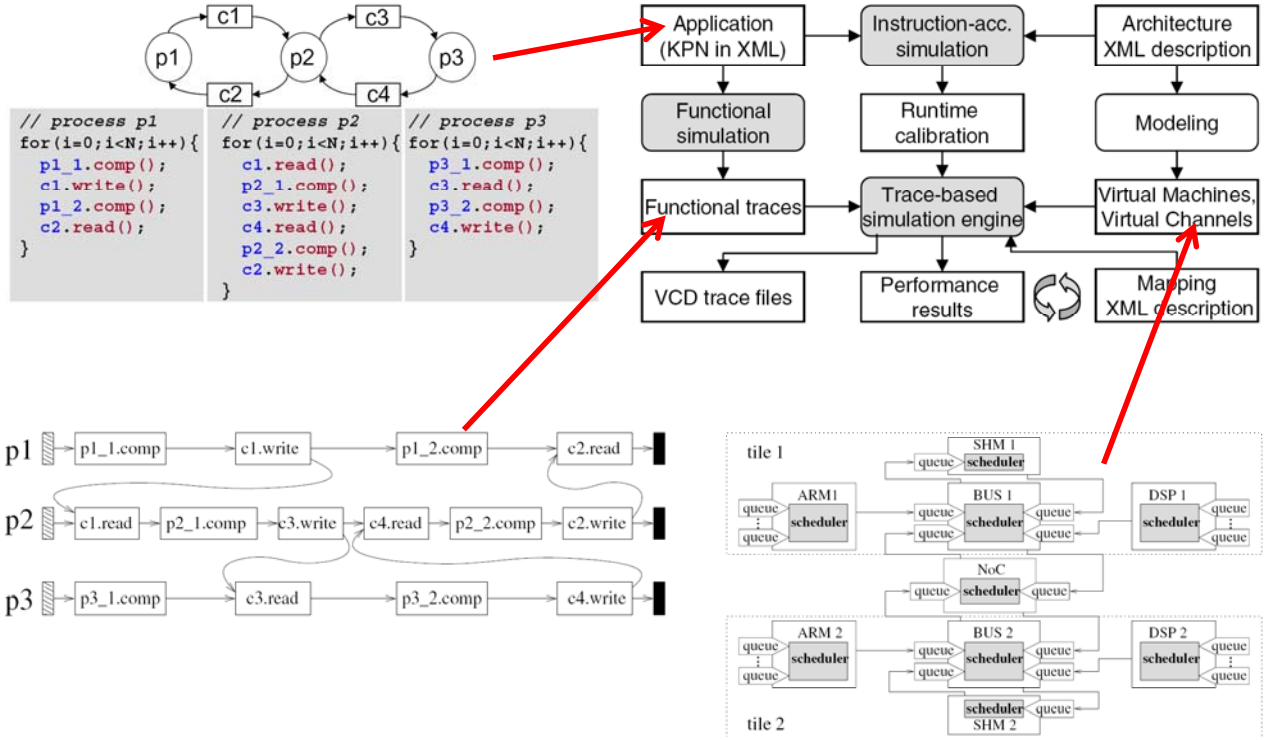
Some Challenges in MPSoC Programming

- Design Process
- Programming Model
- Optimization
- **Scalability**
- Calibration
- DOL (**D**istributed **O**peration **L**ayer)
- Process networks and explicit communication
- Hybrid black-box methods
- **Multi-level performance estimation**

DOL Design Flow



Trace-based Performance Analysis



Example Trace-based Analysis

- ATMEL Multitile (1 ... 8 tiles) with MPEG2 decoder

EST. TIME AND SIM. TIME FOR THE THREE MAPPING CASES, WHEN PROCESSING A 15S VIDEO.

	Est. time (hh:mm:ss)			Sim. time (hh:mm:ss)		
	TSim	VPA	Error	TSim	VPA	Speedup
1	1:36:51	1:38:26	-3%	0:03:30	30:34:00	611
2	1:54:13	1:52:20	+2%	0:03:30	31:30:00	630
3	0:17:17	0:16:46	+3%	0:03:28	74:19:00	1466

different mappings (one tile)

Virtual Platform Analyzer

Trace-based Simulation

Scalability

SIMULATION TIME FOR 1, 2, 4, 8 TILES.

Simulation time (hh:mm:ss)			
1 tile	2 tiles	4 tiles	8 tiles
0:05:08	0:05:29	0:05:43	0:06:4

ESTIMATED EXECUTION TIME FOR VARYING CHANNEL SIZES FOR CASE 1.

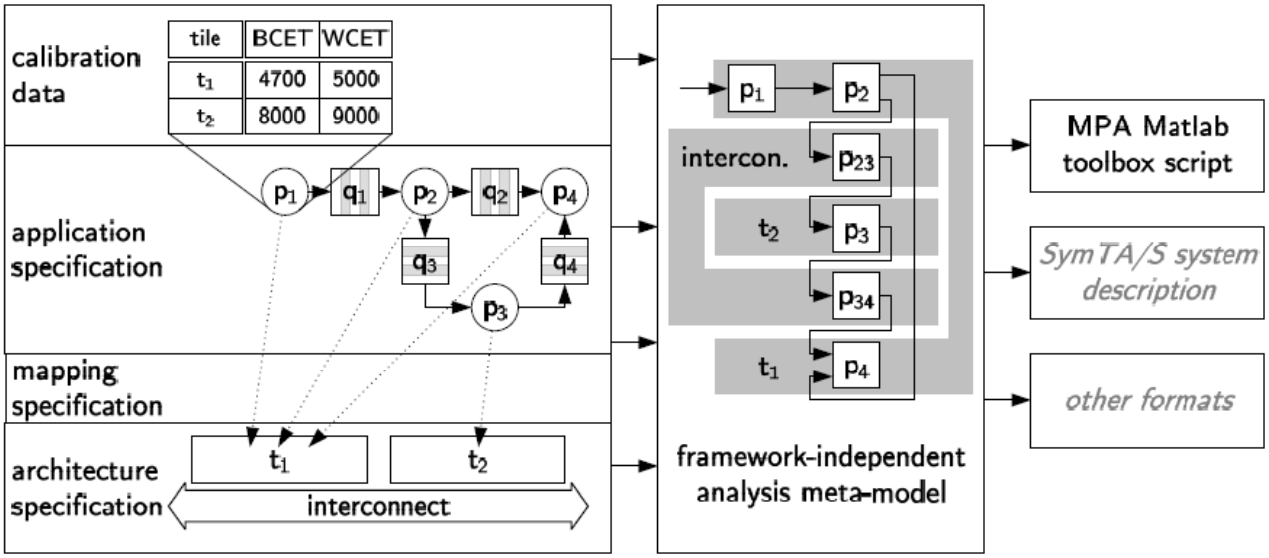
	Est. runtime (hh:mm:ss)			
Chan. size	600 bytes	1 Kbyes	8 Kbyes	16 Kbyes
VPA	1:56:38	1:44:25	1:38:52	1:38:26
TSim	1:53:03	1:45:35	1:37:24	1:36:51
Error	-3%	+1%	-2%	-2%

Accuracy

Advanced Analytic Methods

- *Classification*
  - Combine Binding and Resource Allocation:
    - Multiprocessor scheduling: Extension of uni-processor scheduling theory to multiple processors.
  - Holistic Analysis
    - [Tindell et al.]: Based on response time analysis and fixed point calculations
  - Component-Based Analysis
    - Symta/S [Ernst et. al.]: Concatenation of classical results from uni-processor real-time analysis
    - *Network calculus [Cruz et. al.]: Generalized modeling of streams and resources based on arrival and service curves*

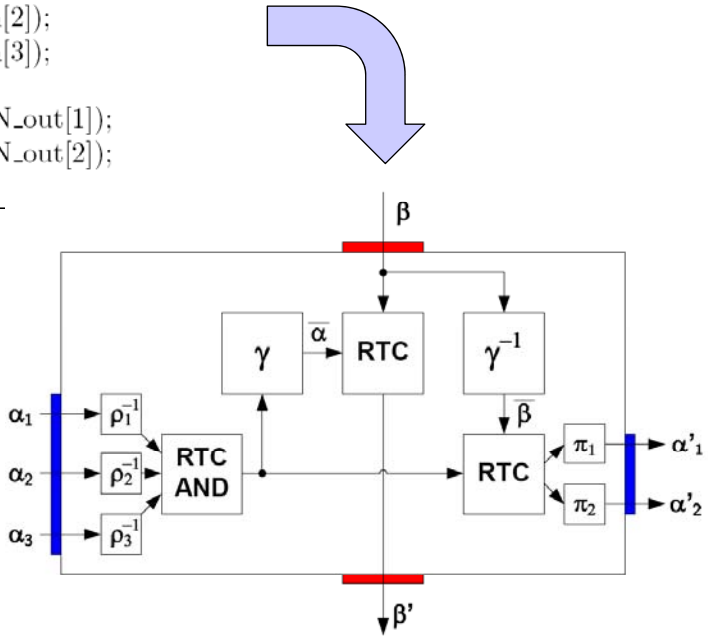
Integration



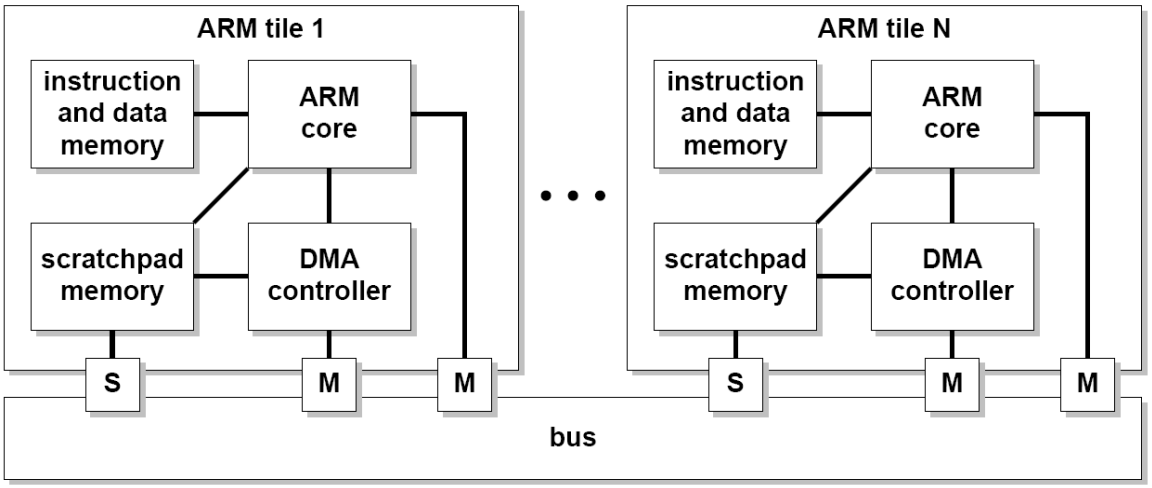
Integration / Application Modeling

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
```



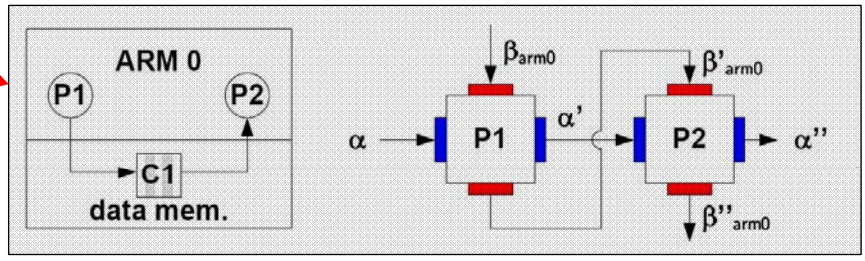
Integration / Architecture Template

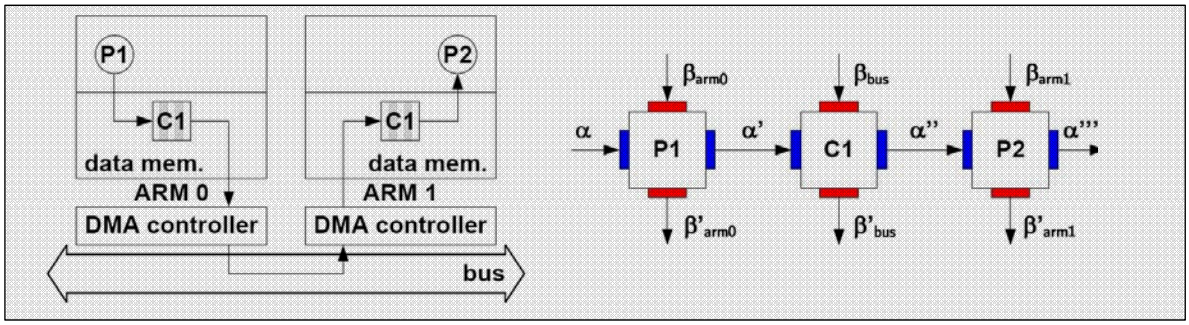


Integration / Communication Modeling

intra-processor communication

inter-processor communication

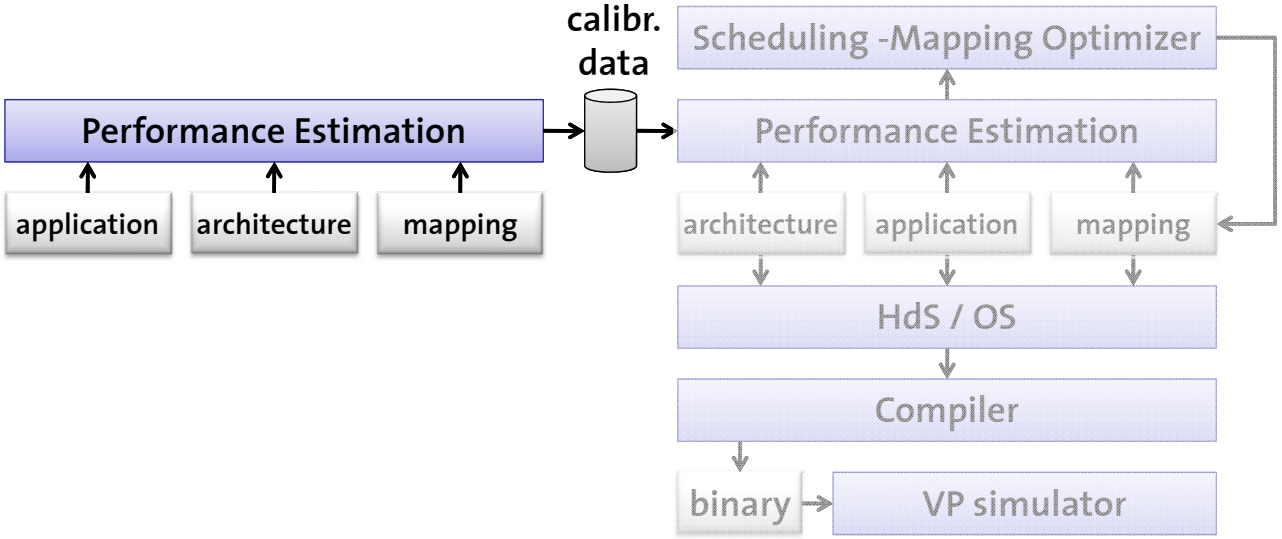




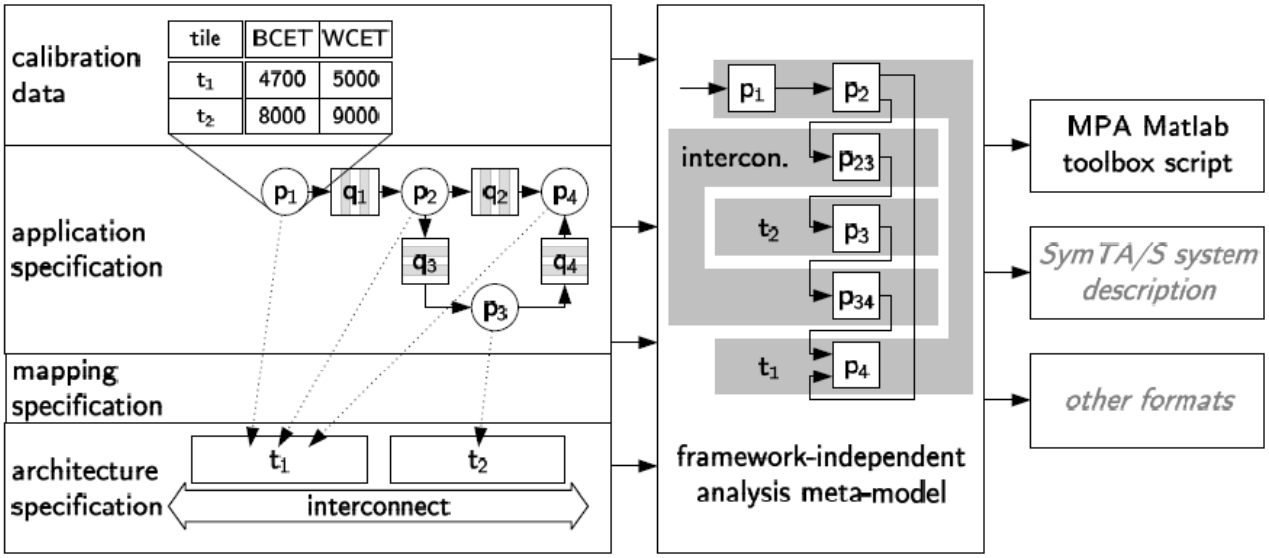
Some Challenges in MPSoC Programming

- Design Process
- Programming Model
- Optimization
- **Calibration**
- Scalability
- DOL (Distributed Operation Layer)
- Process networks and explicit communication
- Hybrid black-box methods
- **Reference points**

DOL Design Flow



Integration

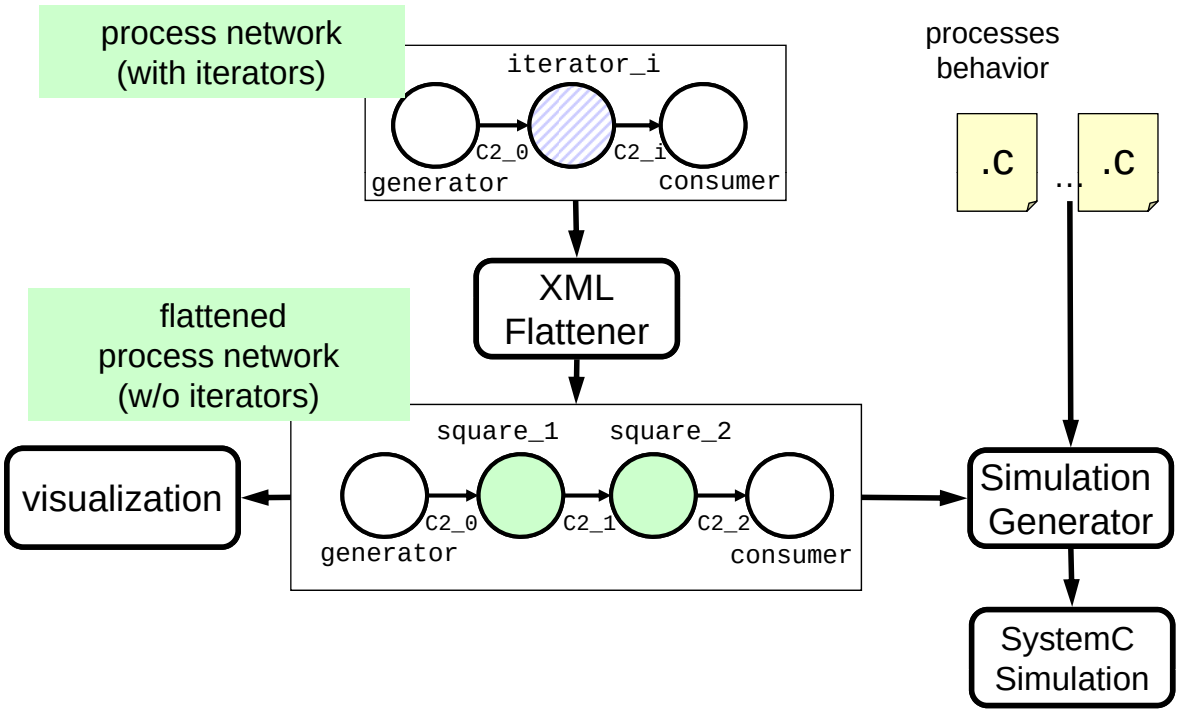


SHAPES Multitile Calibration

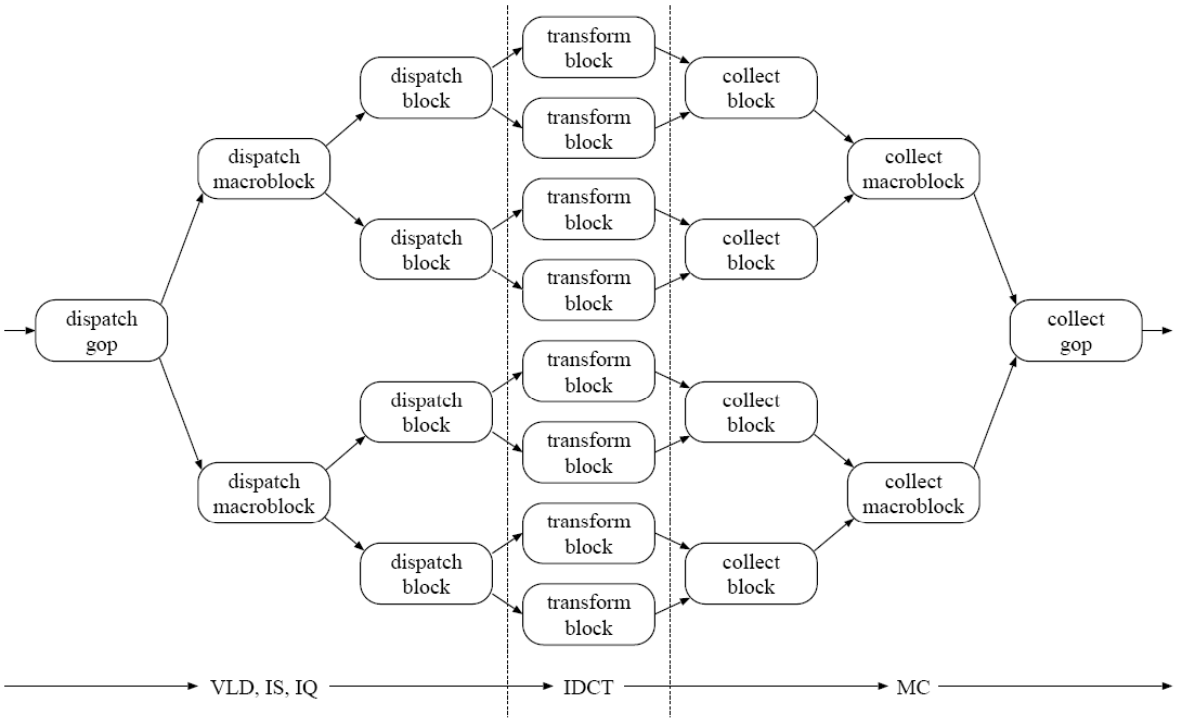
MODEL PARAMETERS REQUIRED FOR MPA.

Entity	Parameter	Unit	Source
process p	best-/worst-case execution time BCET(p), WCET(p)	cycles/act.	low-level sim.
queue q	minimal/maximal token size $N_{\min}(q)$, $N_{\max}(q)$	bytes/access	functional sim.
	write rate, read rate $w(q)$, $r(q)$	1	functional sim.
processor	clock frequency	cycles/s	HW data-sheet
	best-/worst-case CPU utilization of run-time environment	cycles/s	low-level sim.
	best-/worst-case context switch time	cycles/s	low-level sim.
interconnect	throughput	bytes/s	HW data-sheet
environment	system input (arrival curve)	bytes/s	system spec.

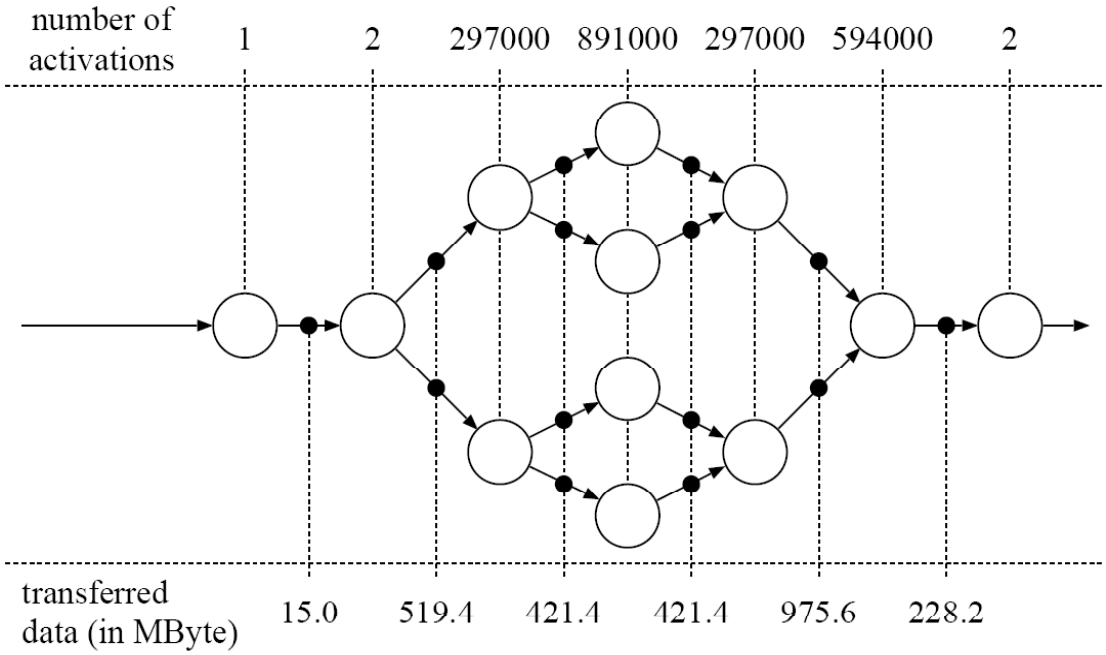
Functional Simulation



Example 1: MJPEG Process Network

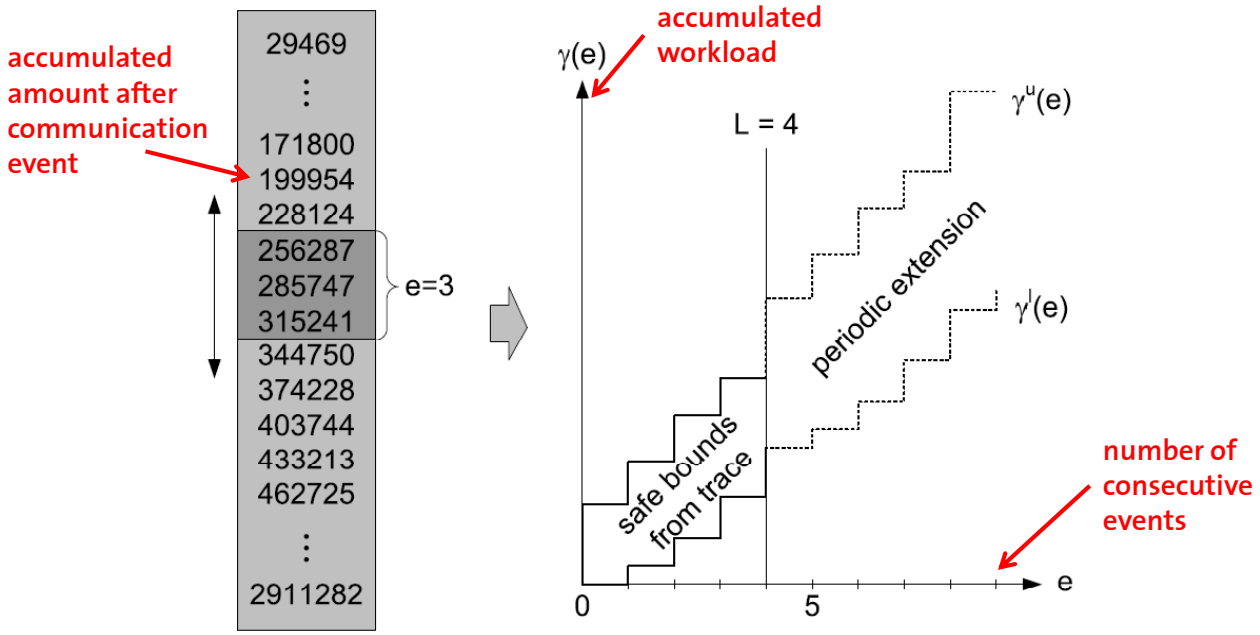


Example 1: MJPEG Functional Simulation

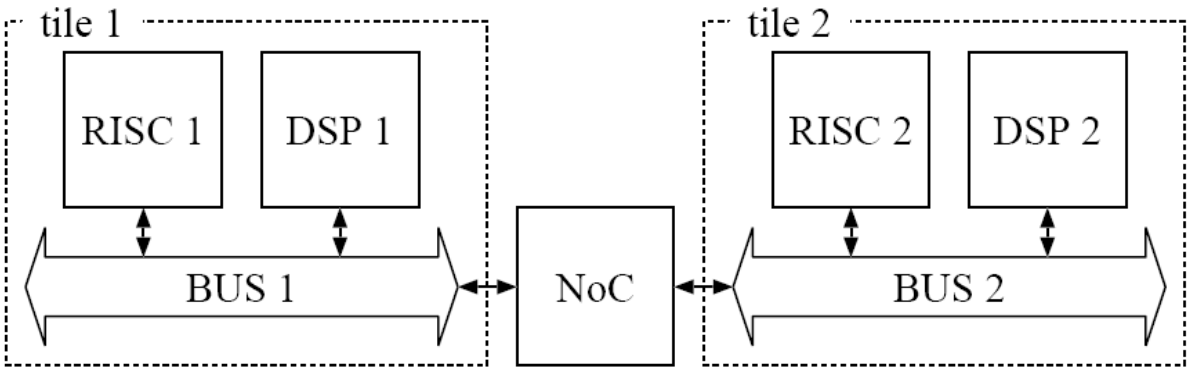


Example 1: Workload Extraction

functional simulation workload bounds



Example 1: Platform



Example 1: Reference Points

- Platform benchmarks
  - communication bandwidth of network components
- Individual task simulations

Process	Runtime on RISC	Runtime on DSP
dispatch gop	0.13	0.20
dispatch macroblock	6.68	8.52
dispatch block	0.06	0.04
transform block	2.00	1.25
collect block	0.05	0.04
collect macroblock	12.33	8.51
collect gop	0.18	0.30

Example 1: Calibration Times

BOUNDS OBTAINED BY MPA COMPARED TO (AVERAGE) QUANTITIES OBSERVED IN SIMULATION.

Quantity	MPA	Simulation
total buffer requirements	< 5272 Bytes	4760 Bytes
delay (source → loudspeaker)	< 1.19 ms	0.83 ms
RISC utilization	31.4 % – 37.5 %	34.1 %
DSP utilization	50.2 % – 56.6 %	52.1 %

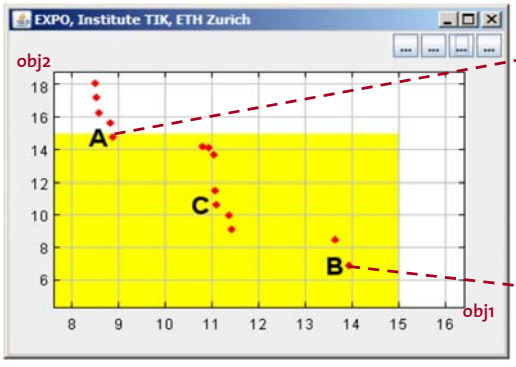
DURATION OF ANALYSIS MODEL GENERATION AND CALIBRATION, MEASURED ON A 2 GHZ AMD ATHLON XP 2800+ MACHINE.

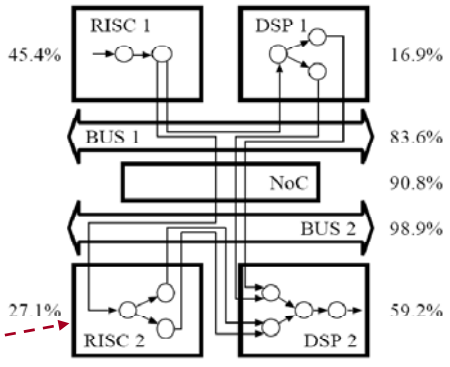
Step		Duration
model calibration (one-time effort)	functional simulation generation	35 s
	functional simulation	3 s
	synthesis (generation of binary)	176 s
	simulation on virtual platform	1300 s
	log-file analysis and back-annotation	50 s
model generation		0.5 s
performance analysis based on generated model		2 s

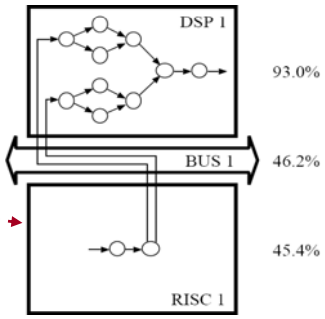
To be done for every exploration cycle

Example 1: Mapping Optimization

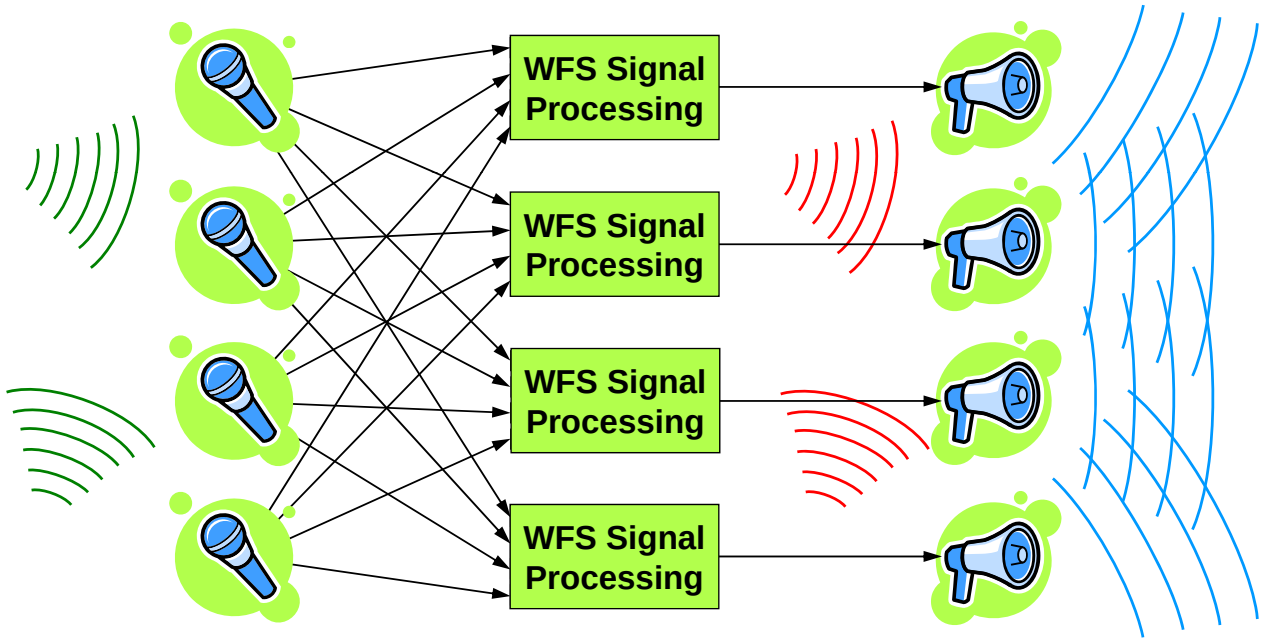
- Exploration under two criteria:
  1. load balancing for the computation
  2. load balancing for the communication



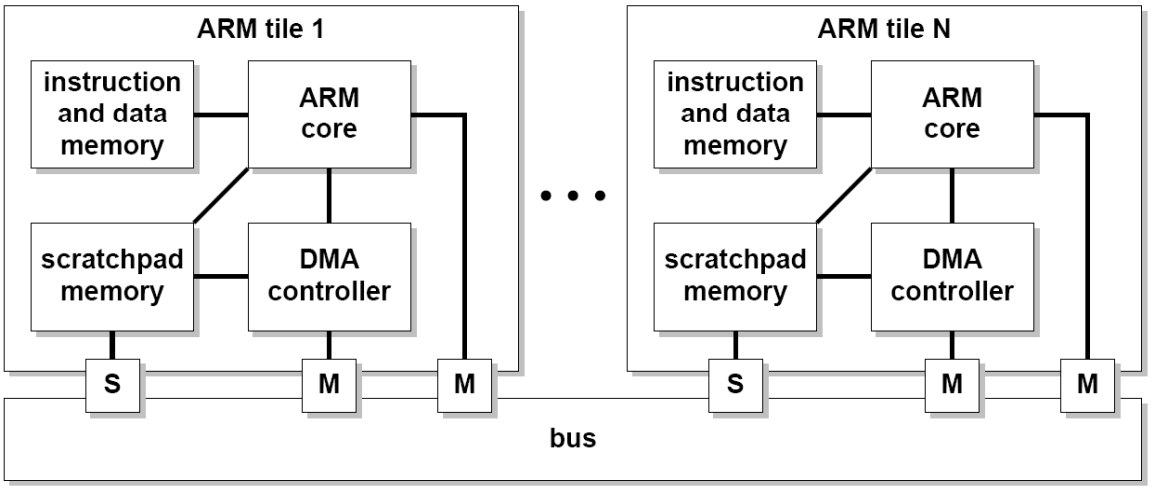




Example 2: Wave Field Synthesis



Example 2: Platform



Example 2: 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 2: 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)