Performance Analysis of Distributed Embedded Systems

- TUTORIAL AT ESWEEK 2007 -

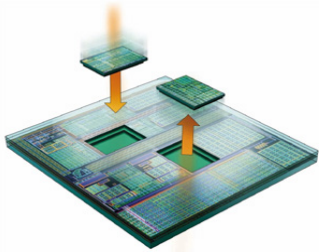
Contents

Analysis and Design

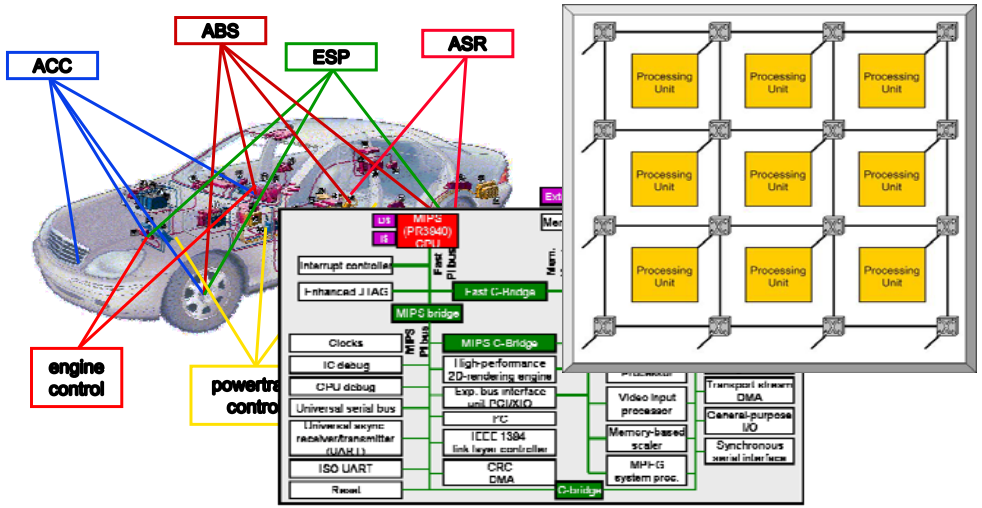
Embedded System =
Computation + Communication + Resource Interaction

Analysis:
Infer system properties from subsystem properties.

Design:
Build a system from subsystems while meeting requirements.

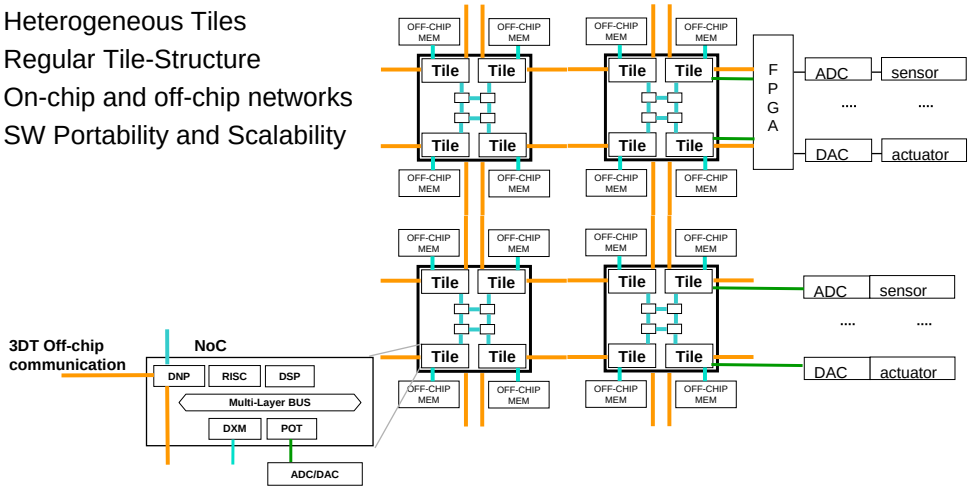


Target Platforms



Target Platforms (SHAPES)

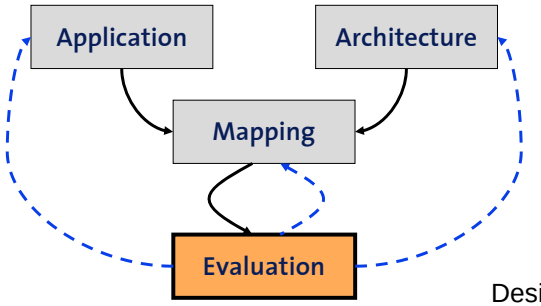
Heterogeneous Tiles
Regular Tile-Structure
On-chip and off-chip networks
SW Portability and Scalability



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

Design Exploration

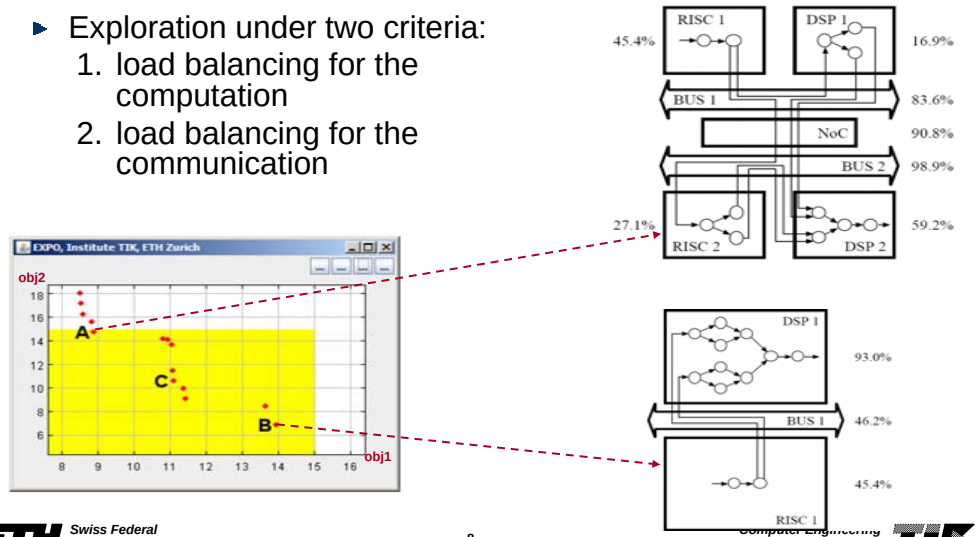


Design decisions (mapping):

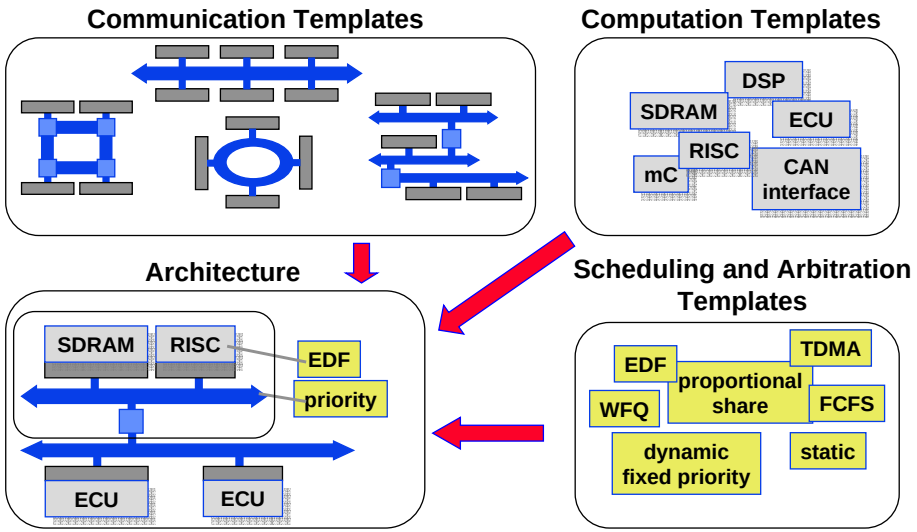
- system partitioning
- binding
- resource sharing

Example: Mapping Optimization

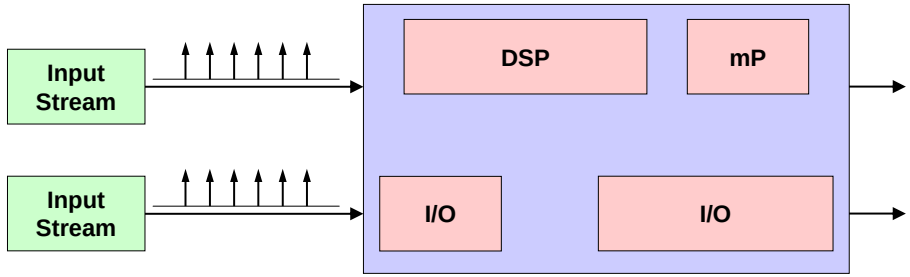
- Exploration under two criteria:
  1. load balancing for the computation
  2. load balancing for the communication



System Composition

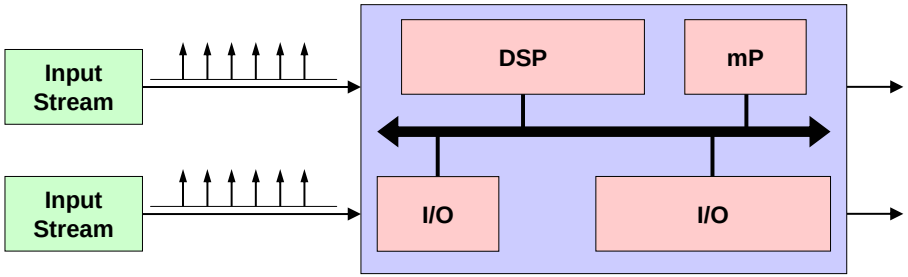


Distributed Embedded System



Computational Resources ...

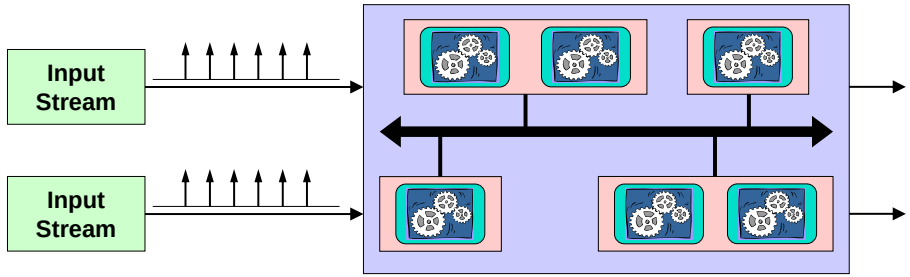
Distributed Embedded System



Computational Resources ...

... Communication Resources ...

Distributed Embedded System

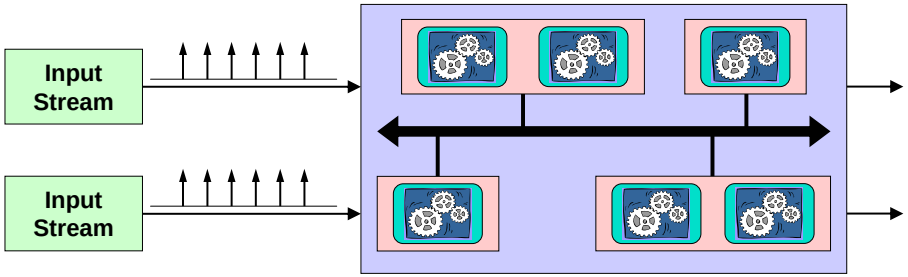


Computational Resources ...

... Communication Resources ...

... Tasks

System-Level Performance Analysis



Memory Requirements?

Timing Properties?

Bottleneck?

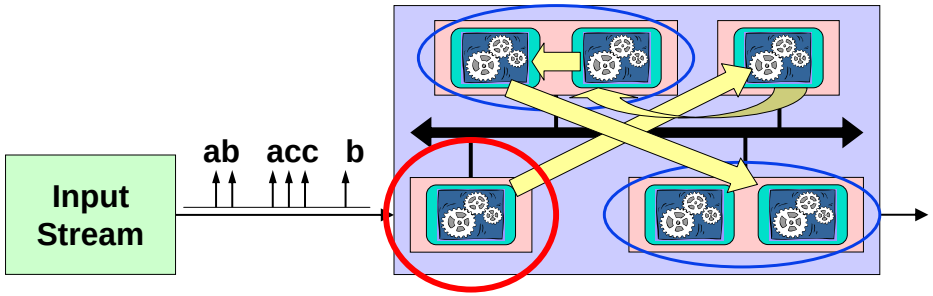
Processor Speeds?

Bus Utilization?

Why Is Evaluation Difficult ?

- ***Non-determinism:***
  - uncertain system environment, e.g. input patterns, load scenarios
  - (non-deterministic) computations in processing nodes
- ***Interference:***
  - sharing computation and communication resources (scheduling and arbitration)
  - internal data streams interact on computing and communication resources which in turn change stream characteristics

Difficulties



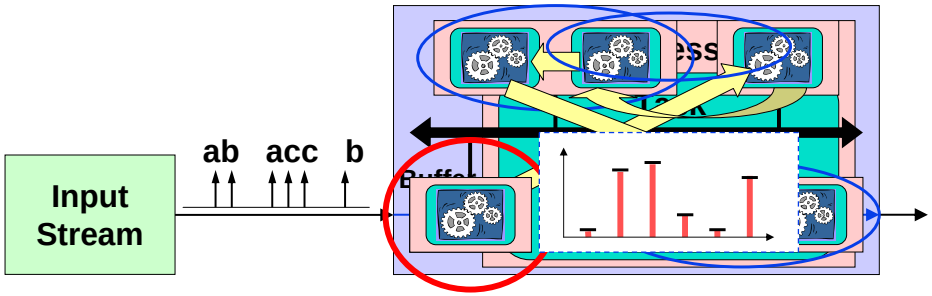
Task Communication

Task Scheduling

Complex Input:

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

Difficulties



Task Communication

Task Scheduling

Complex Input:

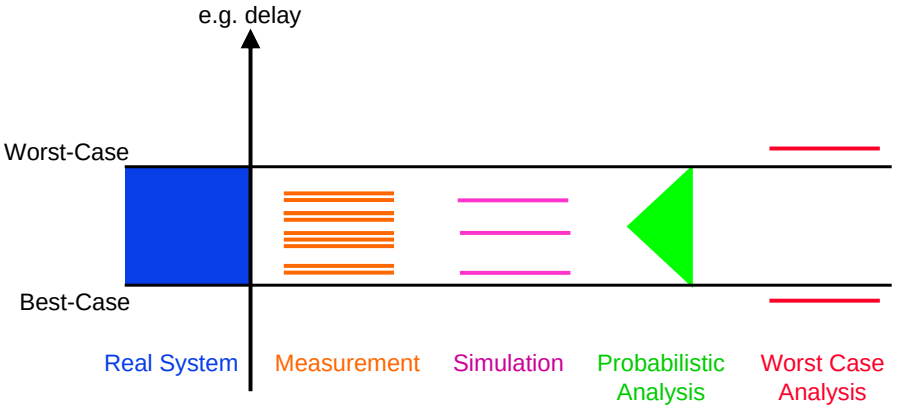
- Timing (jitter, bursts, ...)
- Different Event Types

Variable Resource Availability

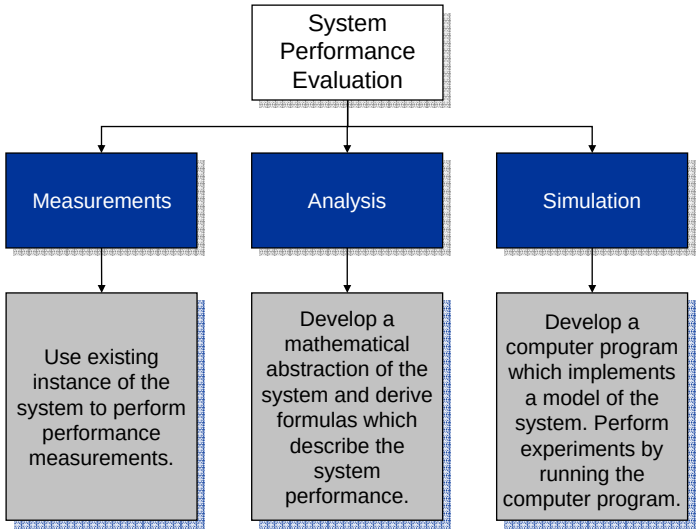
Variable Execution Demand

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

System-Level Evaluation Methods



System-Level Evaluation Methods



Static Analytic (Symbolic) Models

- ***Steps***:
  - Describe computing, communication and memory resources by algebraic equations, e.g.

$$delay = \left\lceil \frac{\#words}{burst\_size} \right\rceil comm\_time$$

  - Describe properties of inputs using parameters, e.g. input data rate
  - Combine relations

- Fast and simple estimation
- Generally inaccurate modeling of shared resources

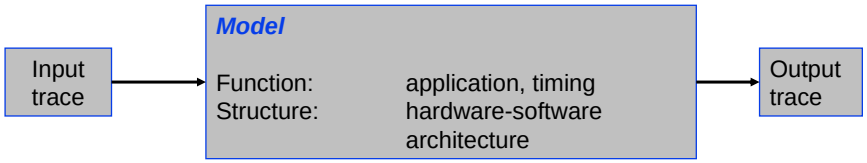
Dynamic Analytic Models

- ***Combination*** between
  - ***Static models***, possibly extended by their dynamic behavior, e.g. non-determinism in run-time and event processing
  - Dynamic models for describing ***shared resources*** (scheduling and arbitration)
  - Dynamic models for describing ***classes of inputs***

- ***Existing approaches***
  - Queuing theory (statistical models, average case)
  - Classical real-time scheduling theory
  - Real-time calculus (interval methods, worst/best case)

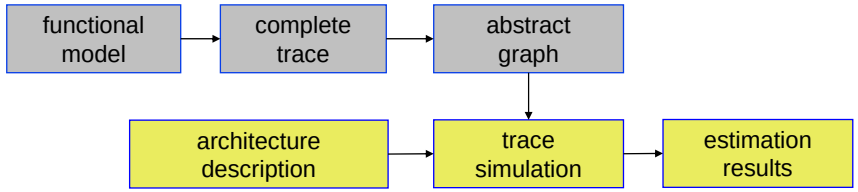
Simulation

- ***Target architecture co-simulation***
  - combines functional and performance validation
  - extensive runtimes but accurate results
- difficult interpretation of results
- complex set-up and debugging
- evaluating average-case behavior



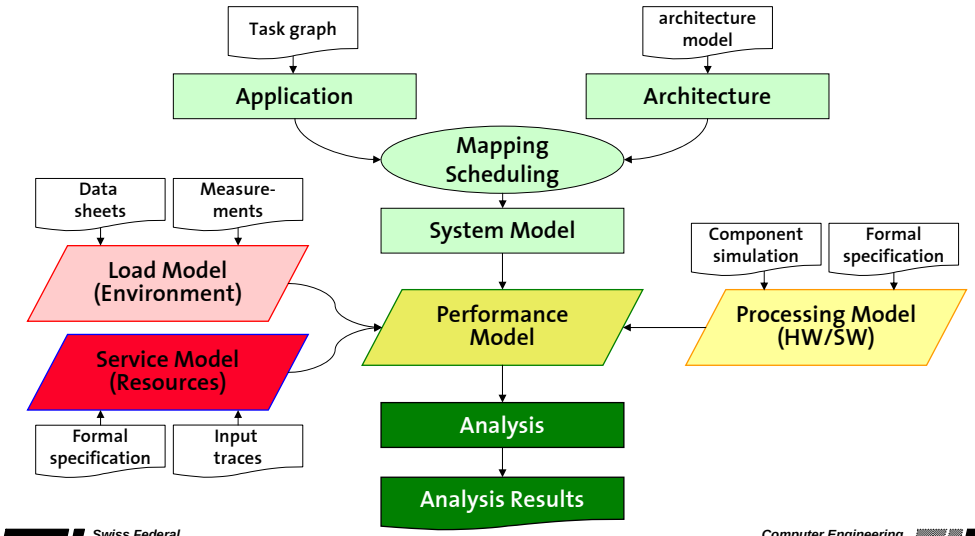
Trace-Based Simulation

- ***Steps***:
  - execution trace determined by co-simulation
  - abstract representation using communication graph
  - extension of graph by actual architecture
  - simulation of extended model
- Faster than simulation, but still based on single trace

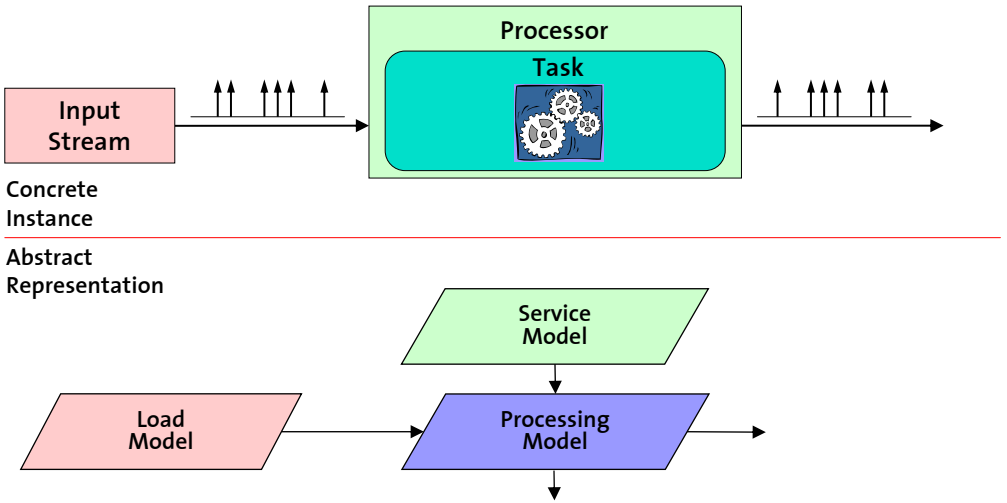


e.g. [Lahiri et al., 2001], [Pimentel et al., 2006]

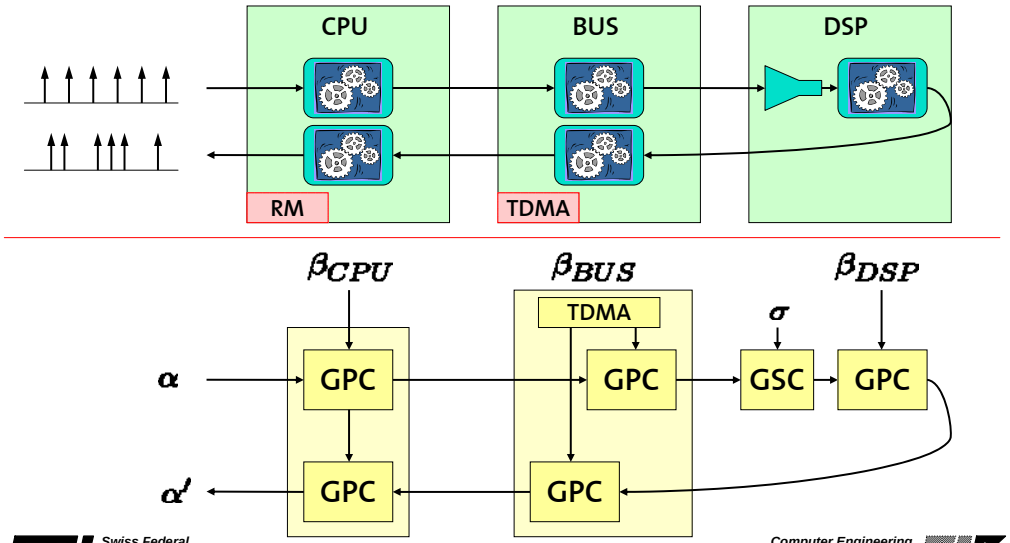
Modular Performance Analysis



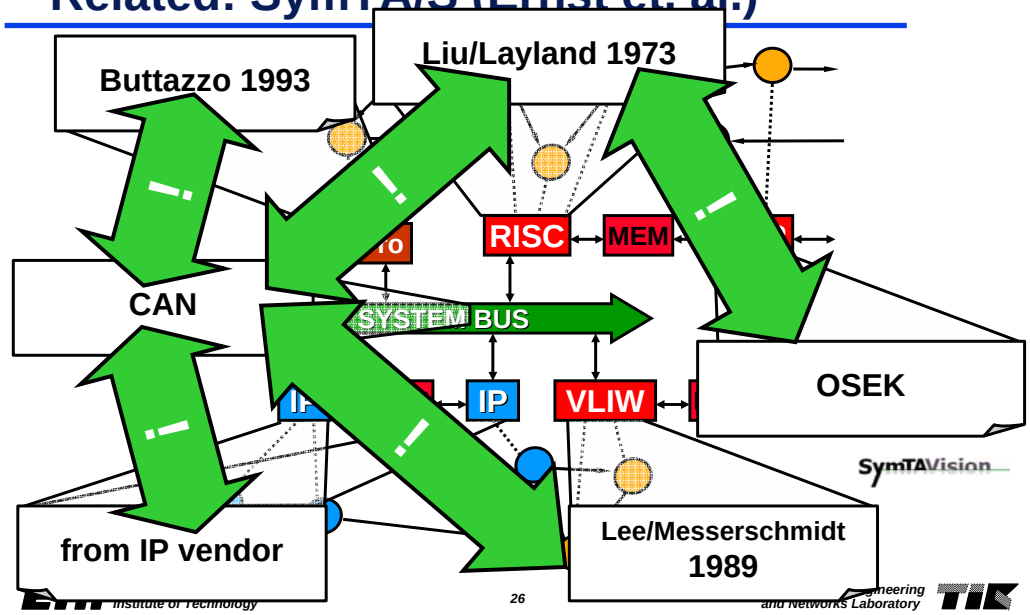
Abstract Models for Performance Analysis



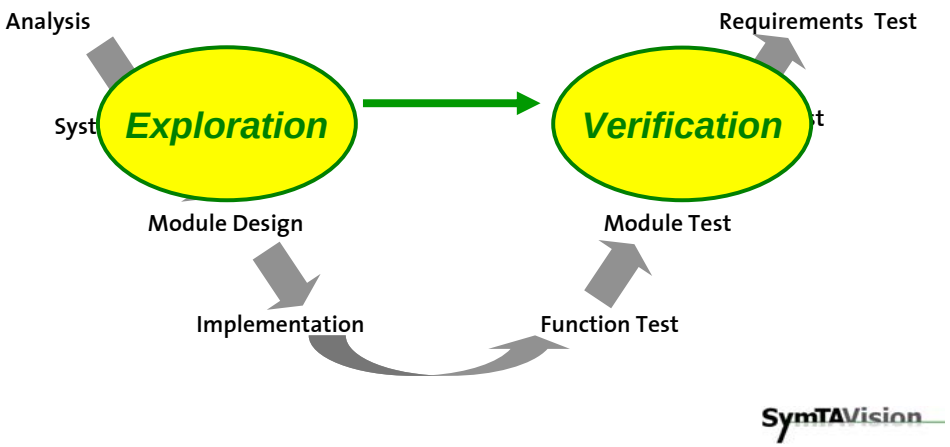
Modular System Composition



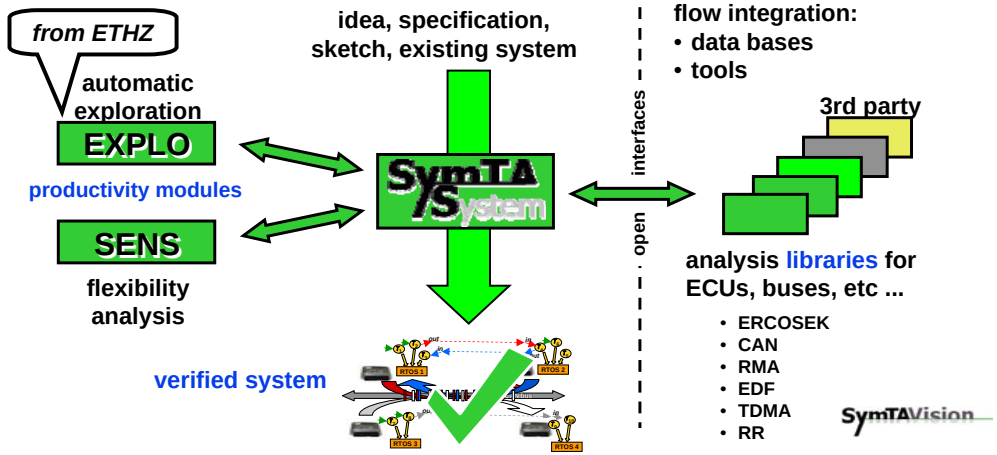
Related: SymTA/S (Ernst et. al.)



Related: SymTA/S (Ernst et. al.)



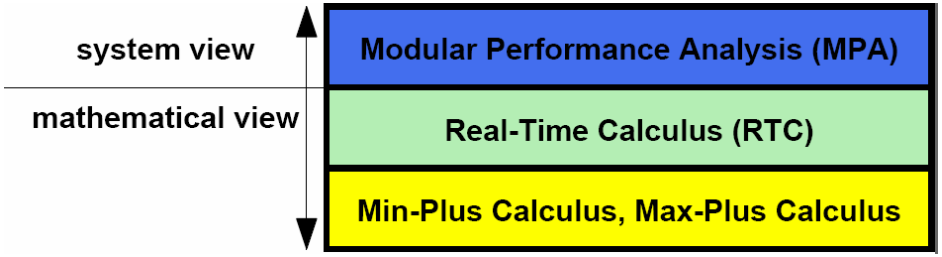
SymTA/S Tool Suite



- ***commercially available*** at SymtaVision GmbH

Contents

Overview



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.

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: $\{S, \oplus, \boxdot\} = \{\mathbf{R} \cup +\infty, \inf, +\}$
- ***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

- ***Common properties*** $\boxdot$:

  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

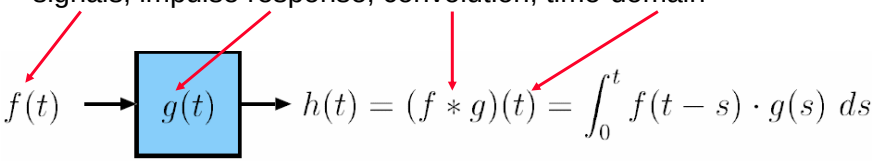
- ***Common properties*** $\boxplus$:

  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 $\boxplus$: $\exists \varepsilon : a \boxplus \varepsilon = a$

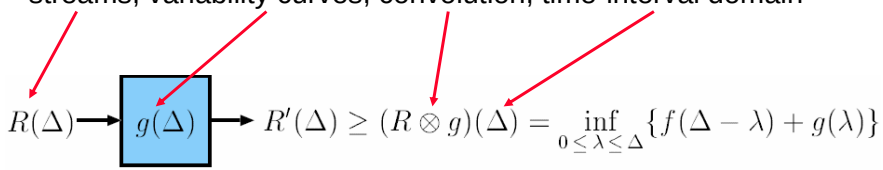
- ***Differences*** $\boxplus$:
  - *plus-times*: Existence of a negative element for $\boxplus$:
    $\exists (-a) : a \boxplus (-a) = \varepsilon$
  - *min-plus*: Idempotency of $\boxplus$: $a \boxplus a = a$

Comparison of System Theories

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

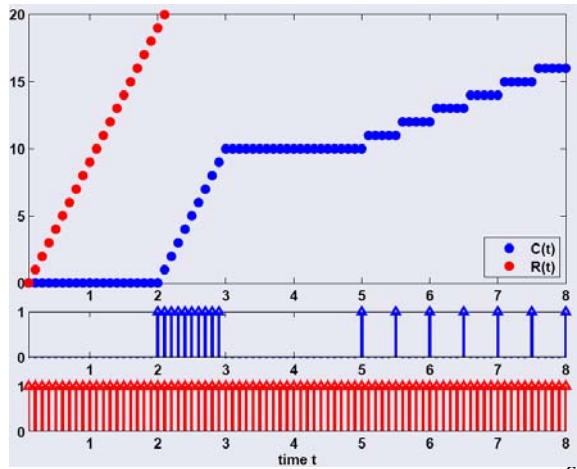


- ***Min-plus system theory***
  - streams, variability curves, convolution, time-interval domain

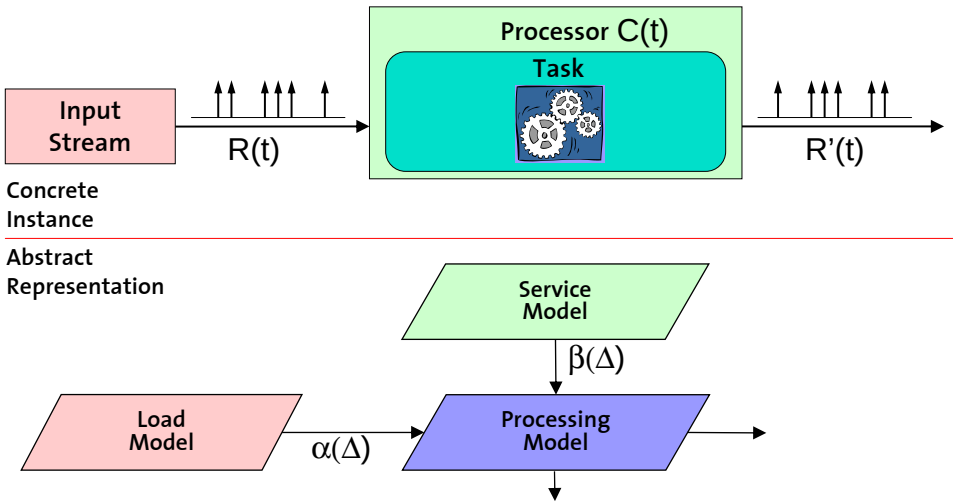


From Streams to Cumulative Functions

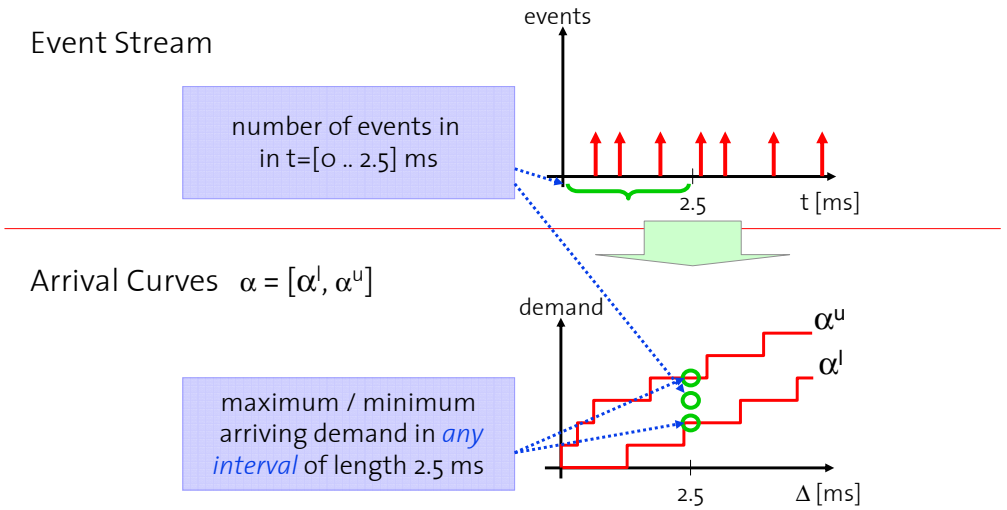
- *Data streams*: R(t) = number of events in [0, t)
- *Resource stream*: C(t) = available resource in [0, t)



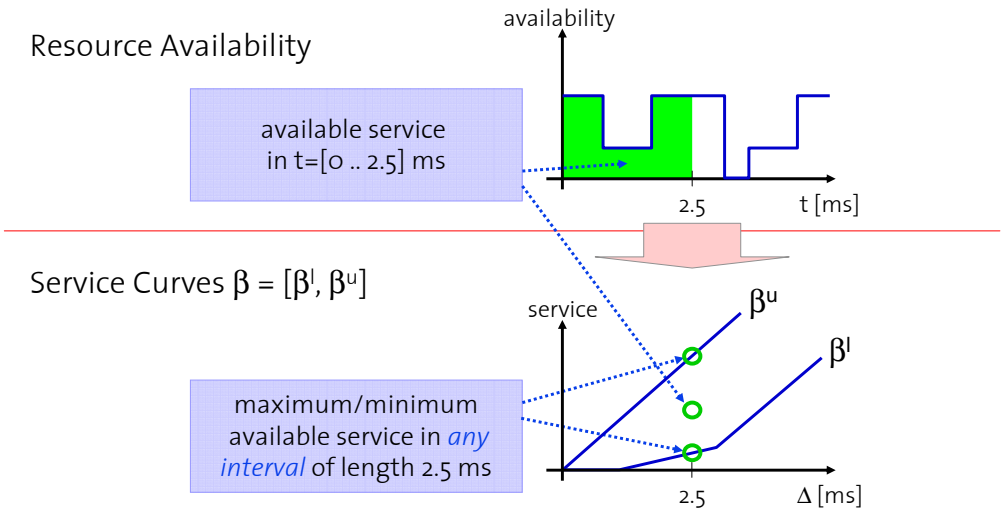
Abstract Models for Performance Analysis



From Event Streams to Arrival Curves

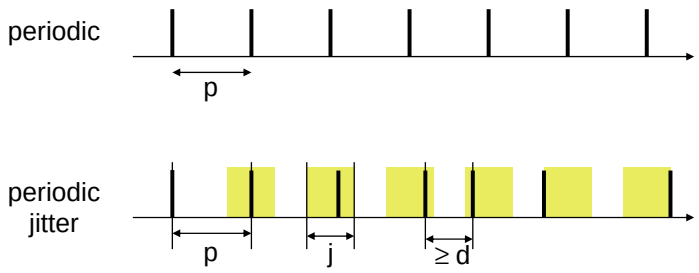


From Resources to Service Curves

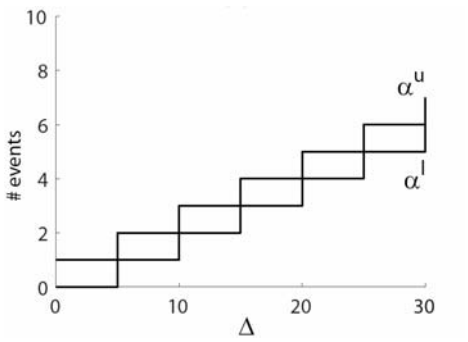


Example 1: Periodic with Jitter

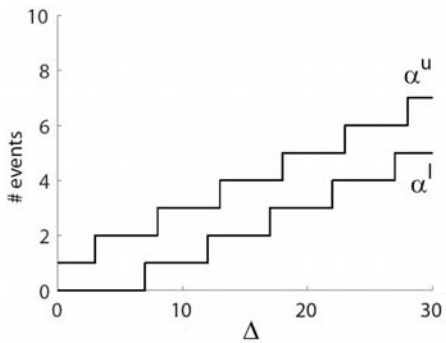
- A ***common event pattern*** that is used in literature can be specified by the parameter triple (*p, j, d*), where *p* denotes the period, *j* the jitter, and *d* the minimum inter-arrival distance of events in the modeled stream.



Example 1: Periodic with Jitter



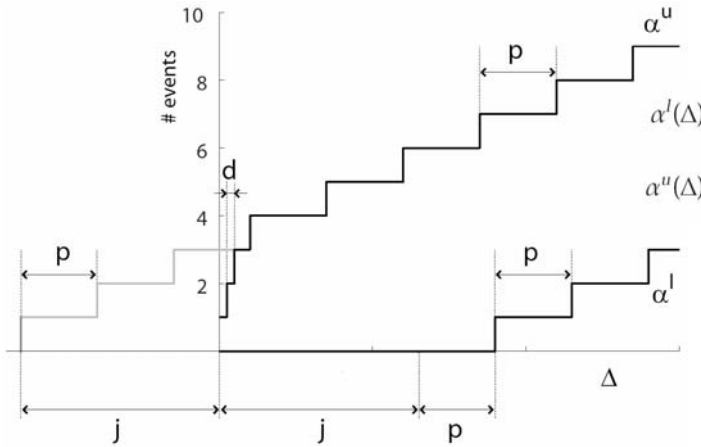
periodic



periodic with jitter

Example 1: Periodic with Jitter

- ***Arrival curves***:

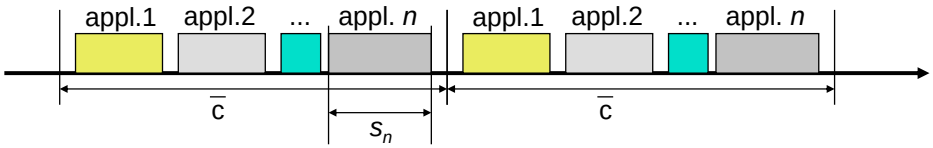


$$\alpha^l(\Delta) = \left\lfloor \frac{\Delta - j}{p} \right\rfloor$$

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

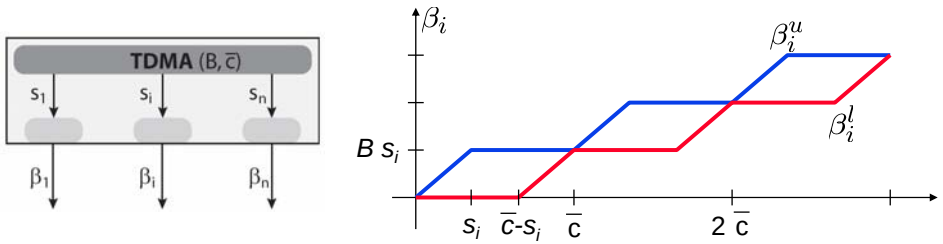
Example 2: TDMA Resource

- 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 $\bar{c}$, one single resource slot of length s_i is assigned to application *i*.



Example 2: TDMA Resource

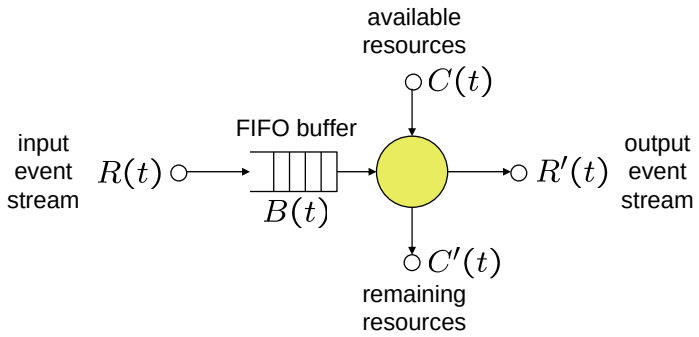
- ***Service curves*** available to the applications / node *i:*



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

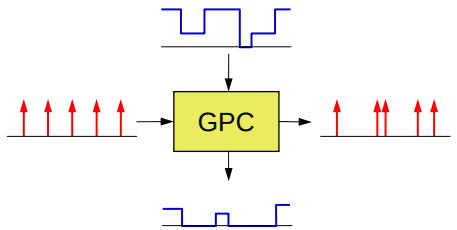
$$\beta_i^u(\Delta) = B \min\left\{ \left\lceil \frac{\Delta}{\bar{c}} \right\rceil s_i, \Delta - \left\lfloor \frac{\Delta}{\bar{c}} \right\rfloor (\bar{c} - s_i) \right\}$$

Greedy Processing Component (GPC)



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

Greedy Processing Component



Behavioral Description

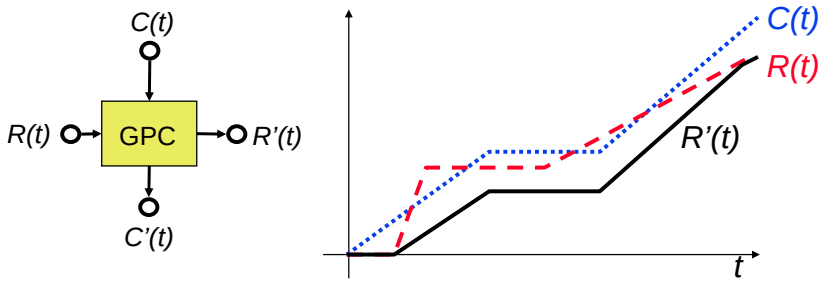
- Component is triggered by incoming events.
- A fully preemptable task is instantiated at every event arrival to process the incoming event.
- Active tasks are processed in a greedy fashion in FIFO order.
- Processing is restricted by the availability of resources.

Greedy Processing Component (GPC)

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

$$\left.\begin{aligned} C(t) &= C'(t) + R'(t) \\ B(t) &= R(t) - R'(t) \end{aligned}\right\} \text{Conservation Laws}$$

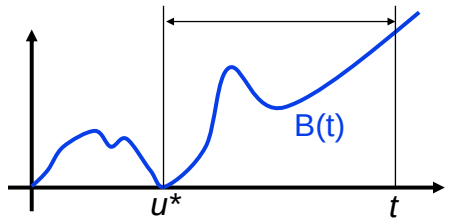
$$R'(t) = \inf_{0 \le u \le t} \{R(u) + C(t) - C(u)\}$$



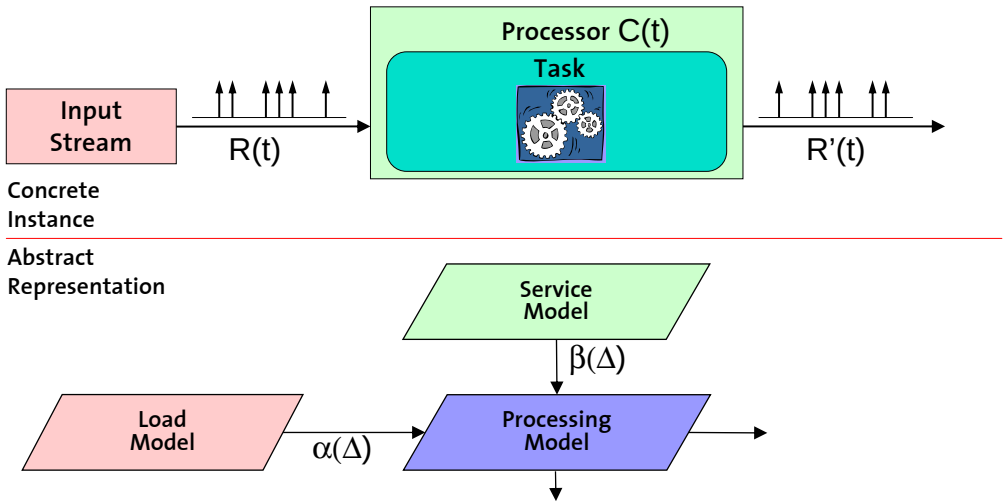
Greedy Processing

- For all times $u \le t$ we have $R'(u) \le R(u)$ (conservation law).
- We also have $R'(t) \le R'(u)+C(t)-C(u)$ as the output can not be larger than the available resources.
- Combining both statements yields $R'(t) \le R(u) + C(t) - C(u)$.
- 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

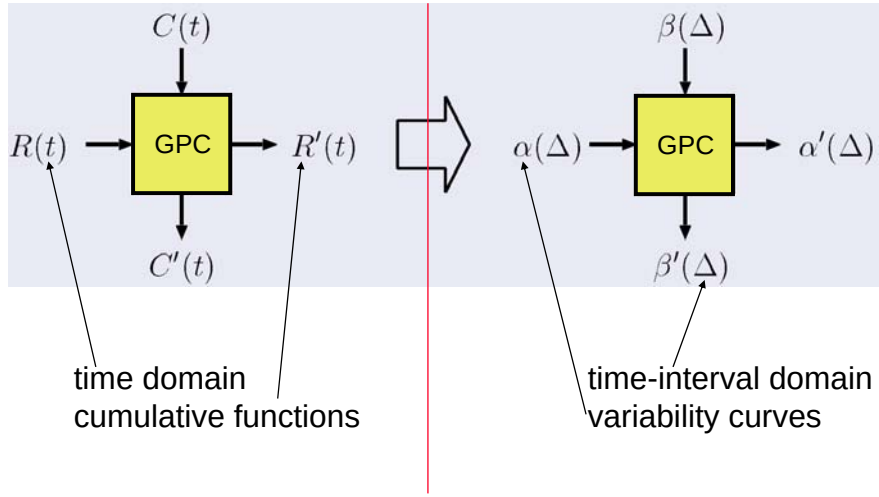
$$R'(t) = \inf_{0 \le u \le t} \{R(u) + C(t) - C(u)\}$$



Abstract Models for Performance Analysis



Abstraction



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***

$$f \le g \otimes h \Leftrightarrow f \oslash h \le g$$

Arrival and Service Curve

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

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

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

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

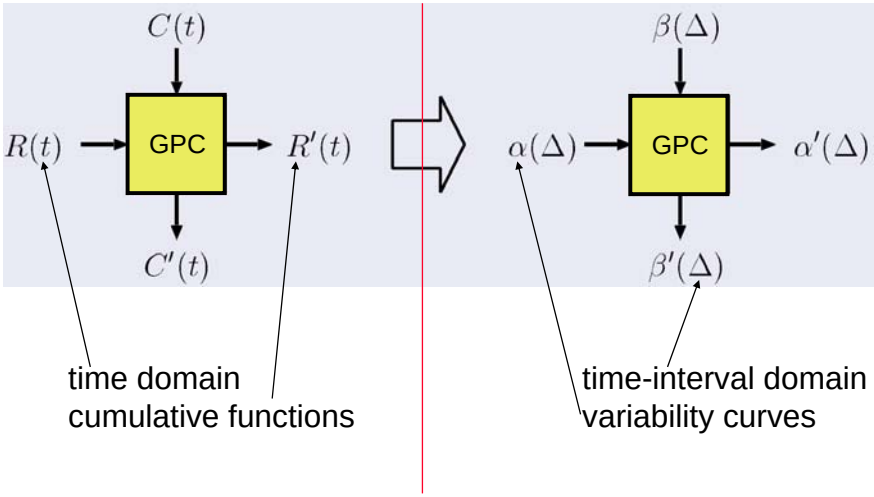
$$\alpha^u = R \oslash R; \quad \alpha^l = R \overline{\oslash} R; \quad \beta^u = C \oslash C; \quad \beta^l = C \overline{\oslash} C$$

- One proof:

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

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

Abstraction



The Most Simple Relations

- The *output stream* of a component satisfies:

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

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

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

- 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

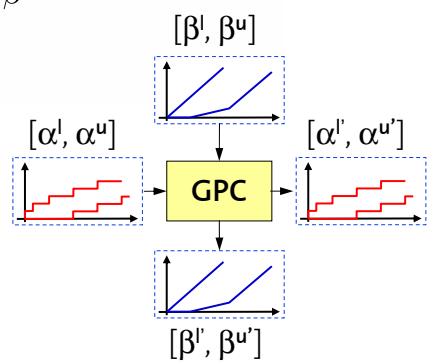
$$\begin{aligned} R'(t) &= \inf_{0 \leq u \leq t} \{R(u) + C(t) - C(u)\} \\ &\geq \inf_{0 \leq u \leq t} \{R(u) + \beta^l(t-u)\} \\ &= (R \underline{\otimes} \beta^l)(t) \end{aligned}$$

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

Tighter Bounds

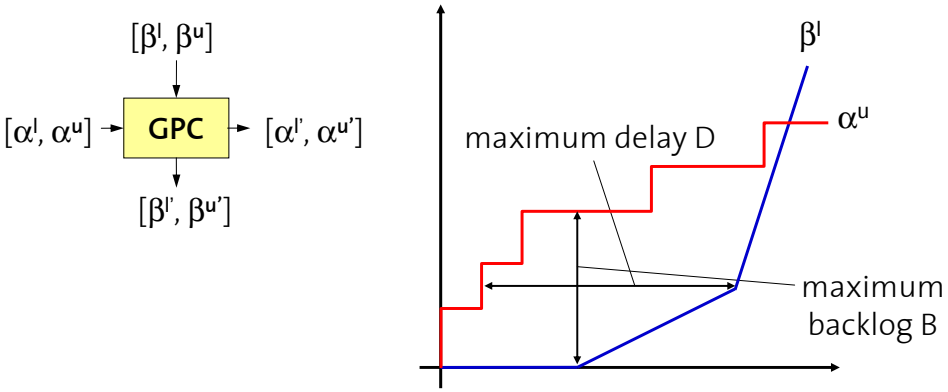
The greedy processing component transforms the variability curves as follows:

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



Without proof ... .

Delay and Backlog

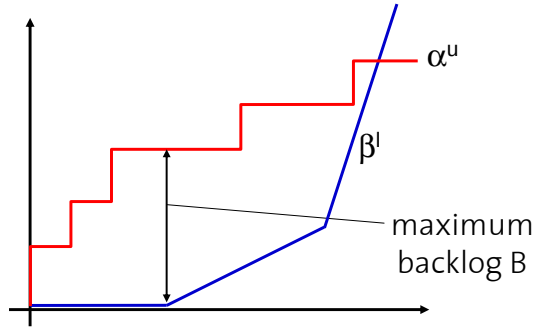


$$B = \sup_{t\geq 0}\{R(t) - R'(t)\} \leq \sup_{\lambda\geq 0}\left\{\alpha^u(\lambda) - \beta^l(\lambda)\right\}$$

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

Proof of Backlog Bound

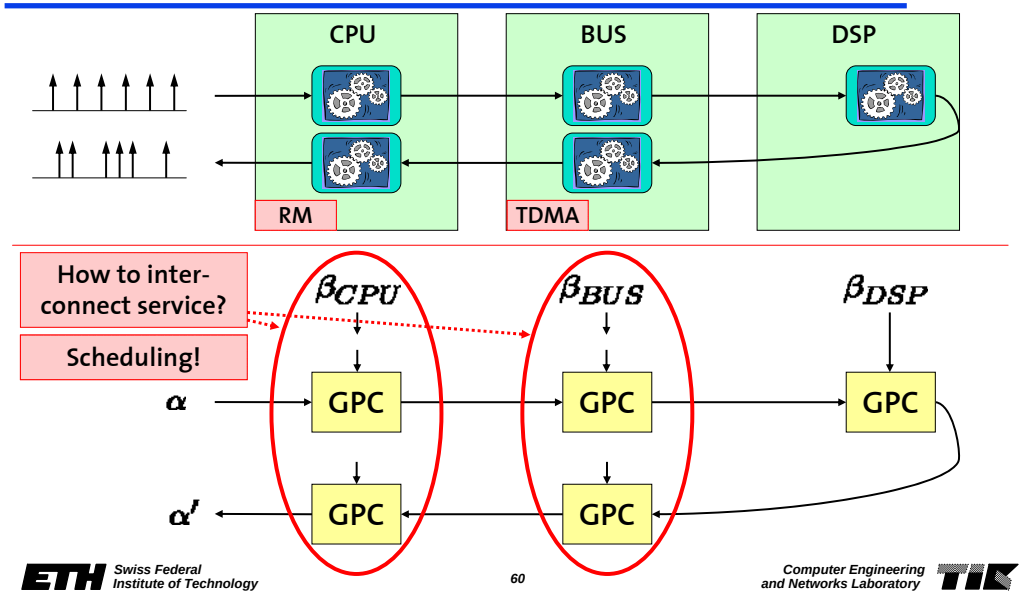
$$B(t) = R(t) - R'(t) = R(t) - \inf_{0\leq u\leq t}\{R(u) + C(t) - C(u)\}$$
$$= \sup_{0\leq u\leq t}\{(R(t) - R(u)) - (C(t) - C(u))\}$$
$$\leq \sup_{0\leq u\leq t}\{\alpha^u(t-u) - \beta^l(t-u)\}$$
$$\leq \sup_{0\leq \lambda}\{\alpha^u(\lambda) - \beta^l(\lambda)\}$$



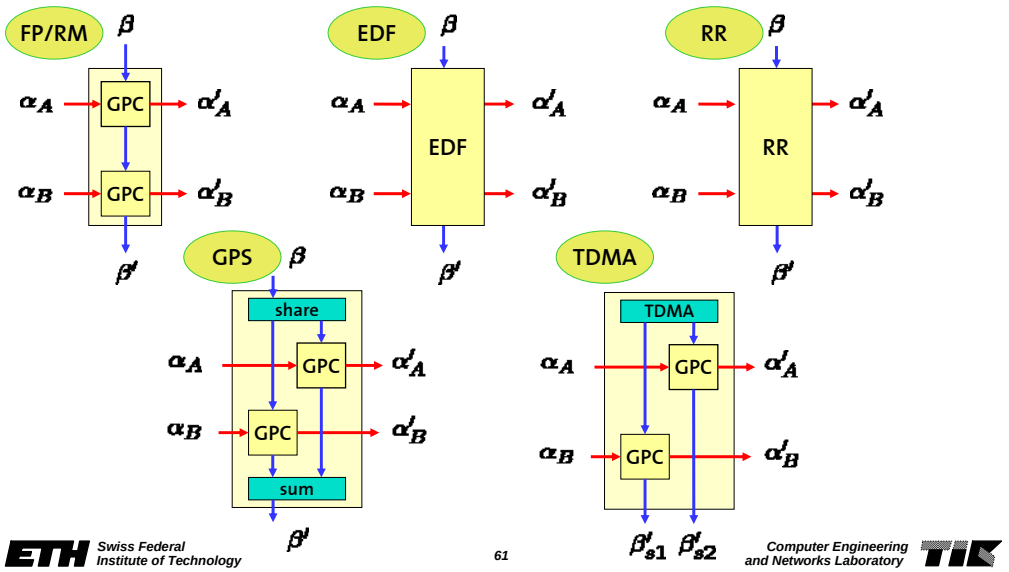
Contents

- Overview
- Real-Time Calculus
- ***Modular Performance Analysis***
- Comparison
- Examples

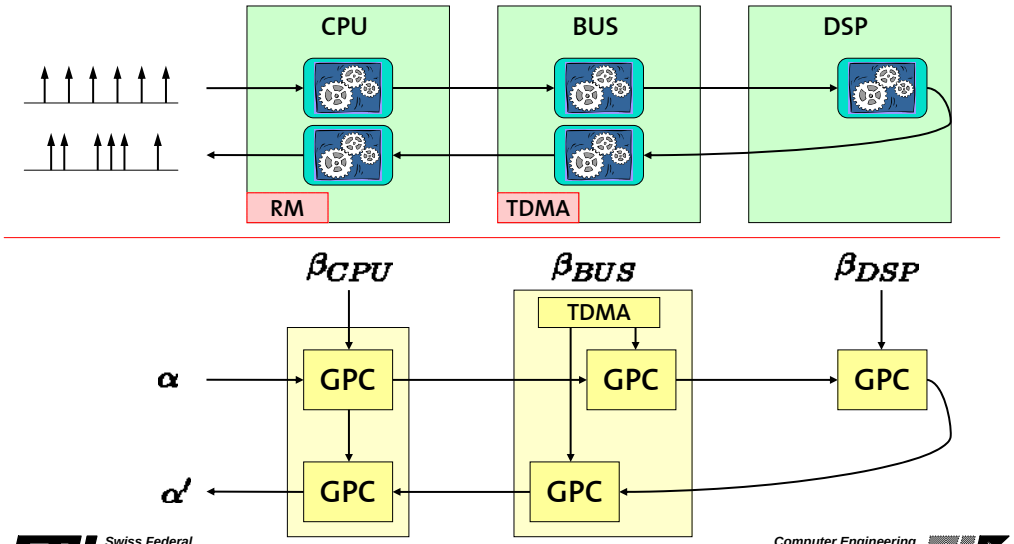
System Composition



Scheduling and Arbitration

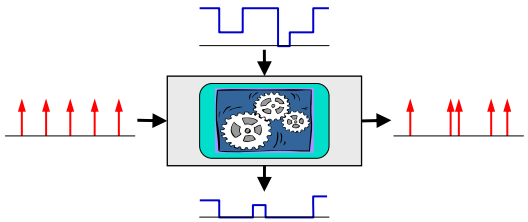


Complete System Composition



Extending the Framework

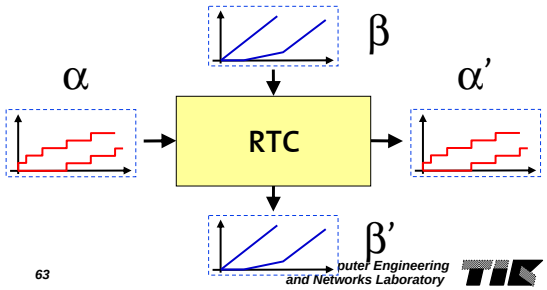
- New HW behavior
- New SW behavior
- New scheduling scheme
- …



- Find new relations:

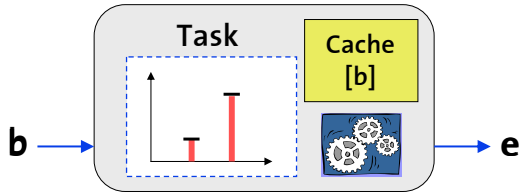
$$\alpha'(\Delta) = f_\alpha(\alpha,\beta)$$
$$\beta'(\Delta) = f_\beta(\alpha,\beta)$$

This is the hard part…!

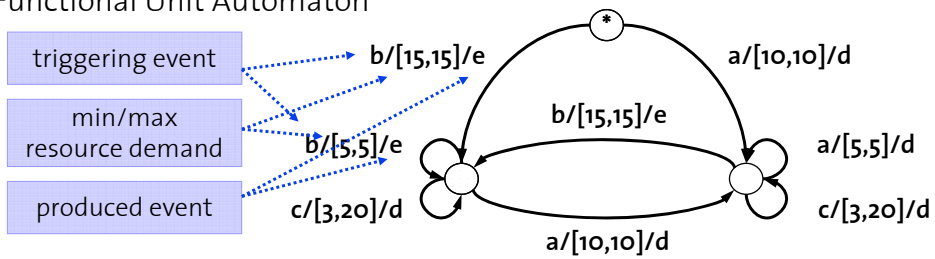


Refined Processing Component Model

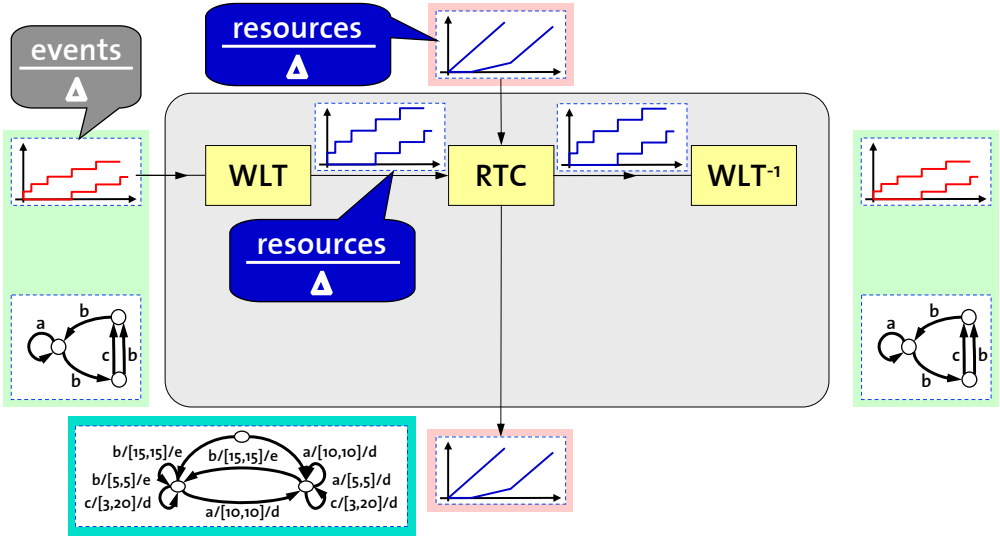
Formal Specification
Program Analysis
Data Sheets
…



Functional Unit Automaton



Processing Component

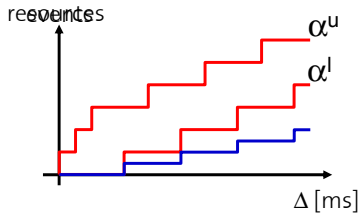


Classical Workload Transformation

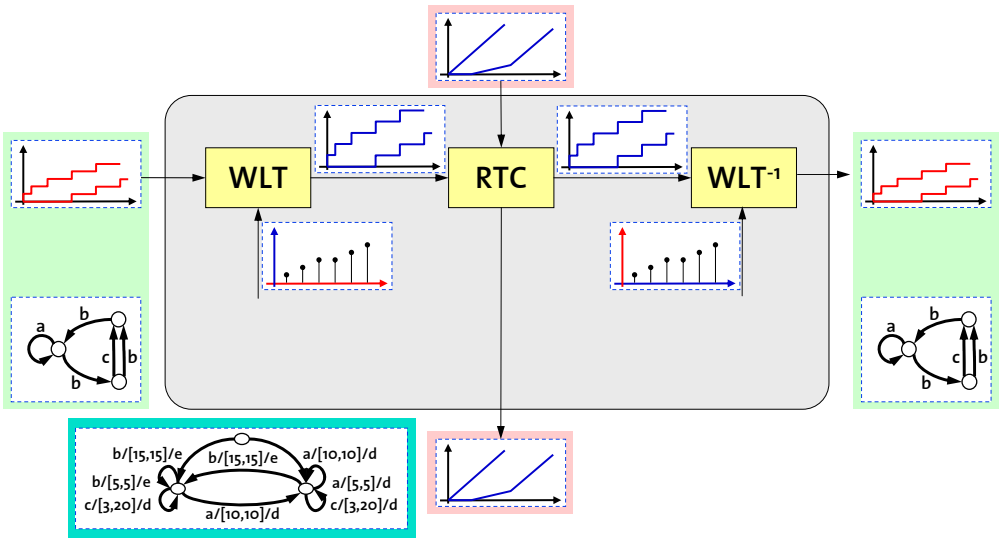
Worst Case Execution Demand
&
Best Case Execution Demand

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

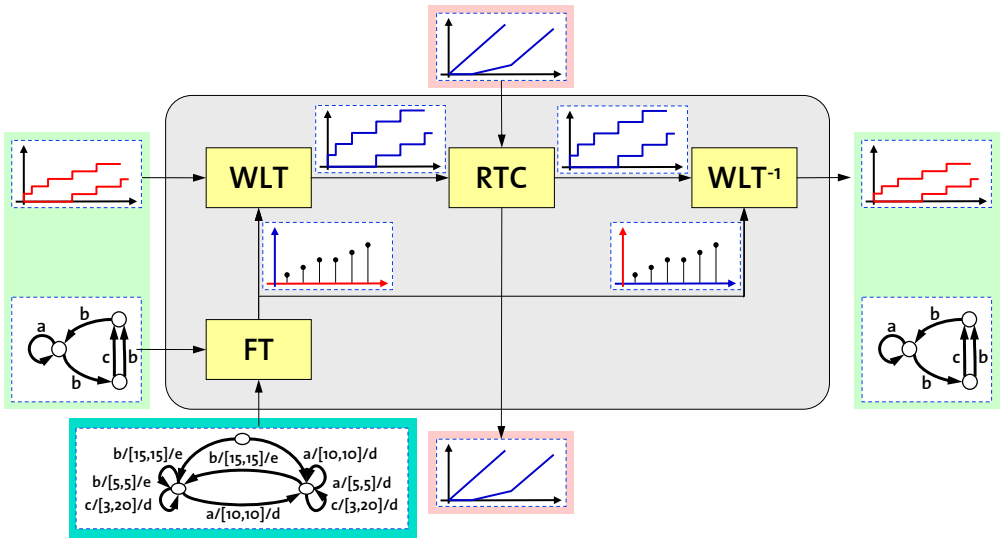
events	BCED	WCED
1	0.5	2
2	1	4
3	1.5	6
4	2	8



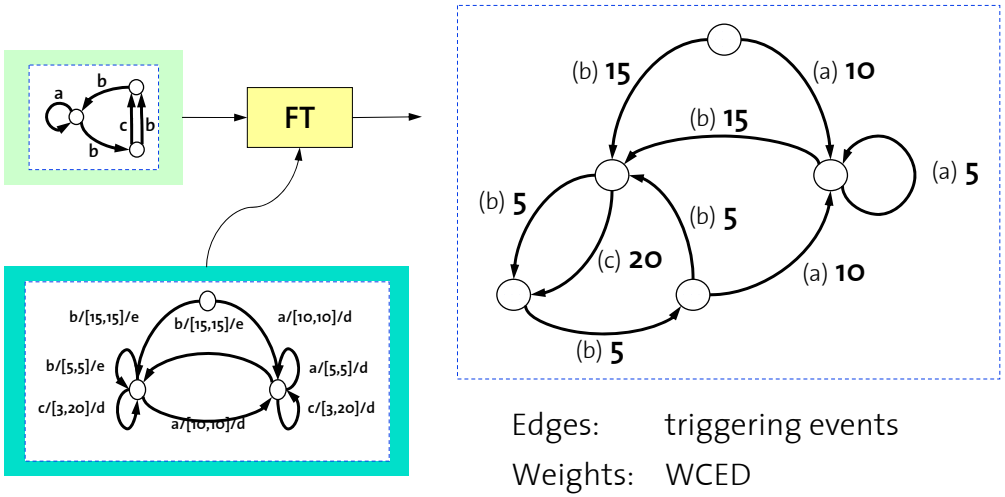
Processing Component



Processing Component



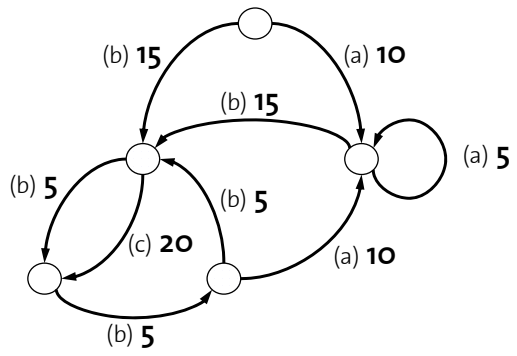
WLT with Abstracted Functionality



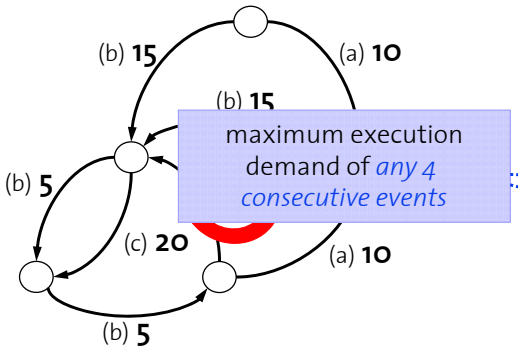
Edges: triggering events

Weights: WCED

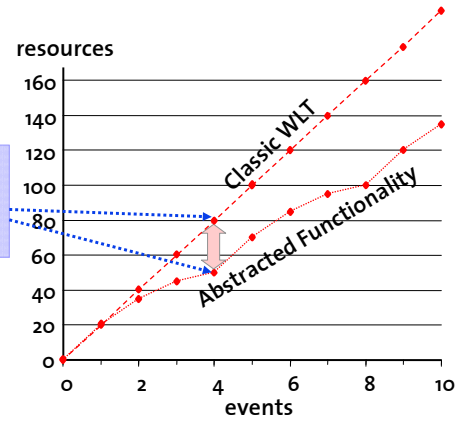
WLT with Abstracted Functionality



WLT with Abstracted Functionality



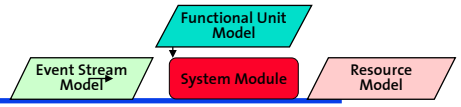
maximum execution demand of *any 4 consecutive events*

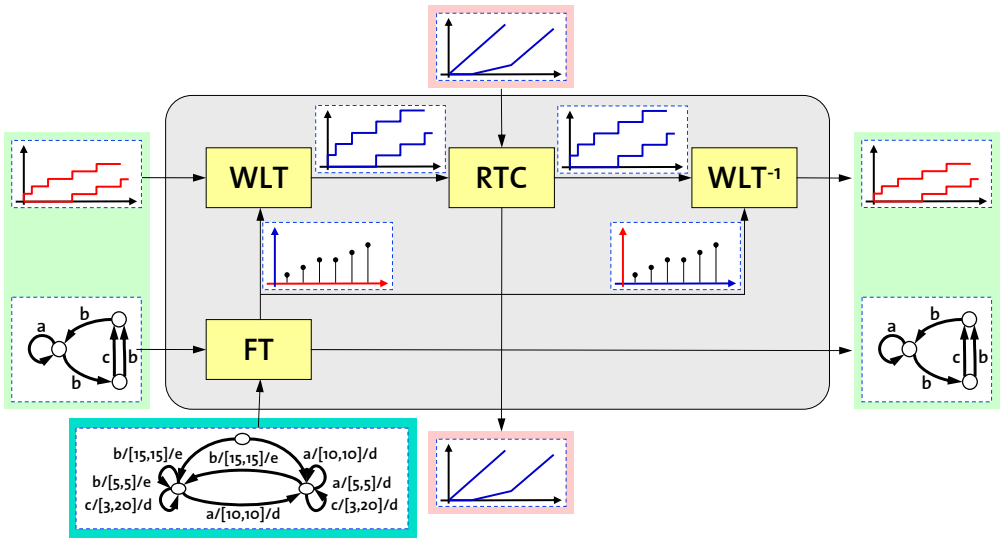


Execution demand of n consecutive events:

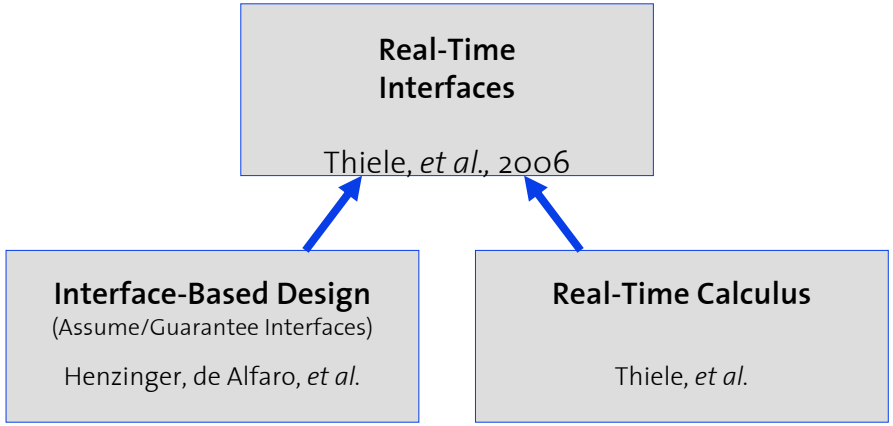
WCED(n) = max-weight path of length n

System Module

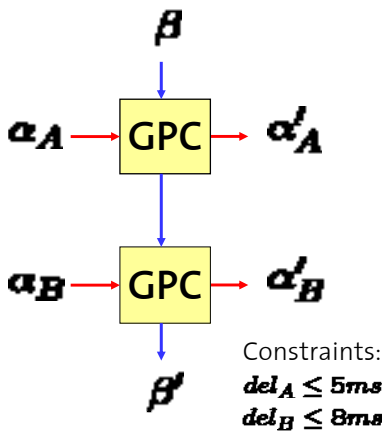




Real-Time Interfaces



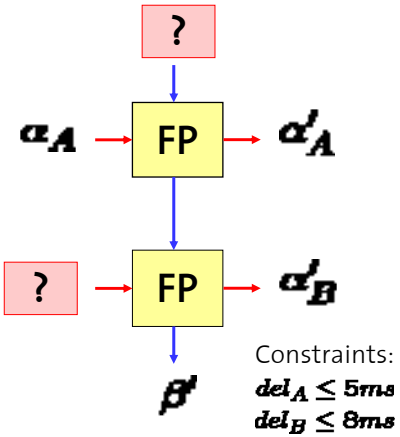
Component-Based Design



Schedulable?

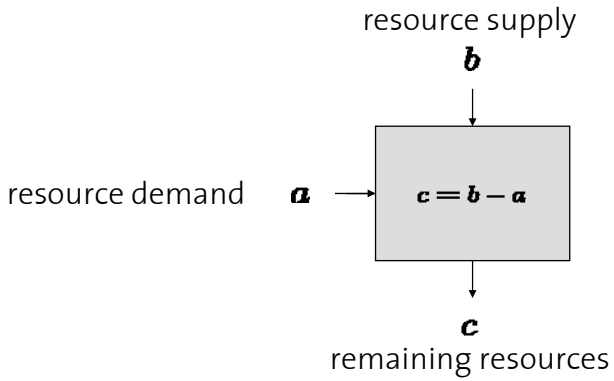
- ***Analysis***
  - Given: all components, their interconnections structure and all inputs from environment
  - Question: do the components work together properly?

Interface-Based Design

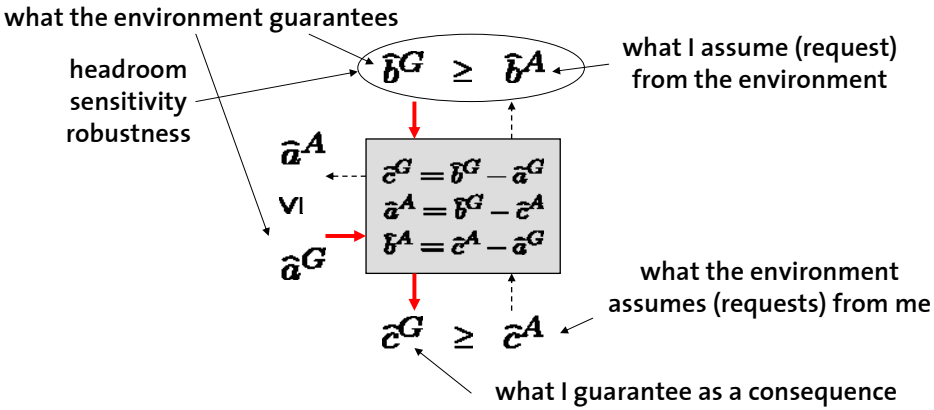


- ***Design and Composition***
  - Given: some components, some inputs and some requirements
  - Questions:
    - What are the system assumptions towards the environment (inputs and/or requirements)?
    - What are the corresponding assumptions for the components (so that they can adapt)?

From an Abstract Component ...



... to its Adaptive Interface



Correct by successful composition

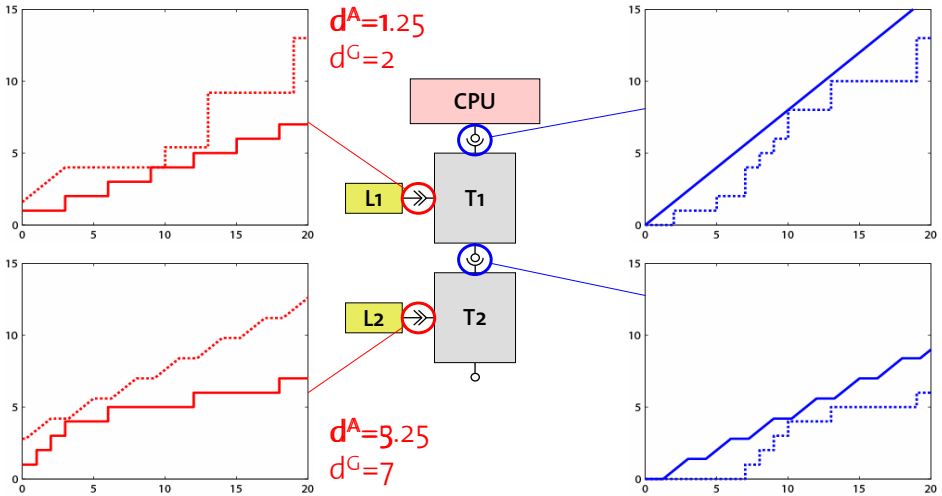
$$(\hat{a}^G \leq \hat{a}^A) \wedge (\hat{b}^G \geq \hat{b}^A) \Rightarrow (\Leftrightarrow)\ \hat{c}^A \leq \hat{c}^G$$

Applications

- ***Interface-Based Design***
  - Check of Real-Time Requirements at Composition-Time
  - Incremental Design
  - Independent Implementability
  - …
- ***Answering design questions***, e.g. resource dimensioning
- ***On-Line***
  - Load Adaption
  - Service Adaption
  - Admission Tests

On-Line Service Adaption

1. Check
2. Admit
3. Update



Contents

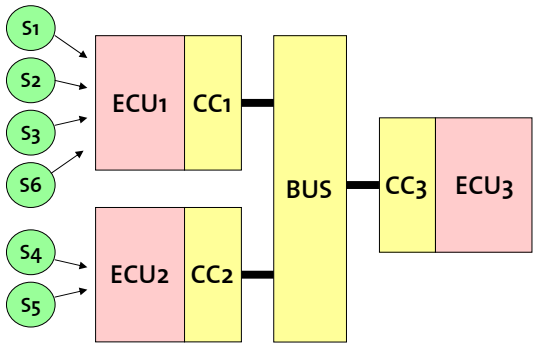
- Overview
- Real-Time Calculus
- Modular Performance Analysis
- ***Comparison (see EMSOFT 07)***
- Examples

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

Case Study



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

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)

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

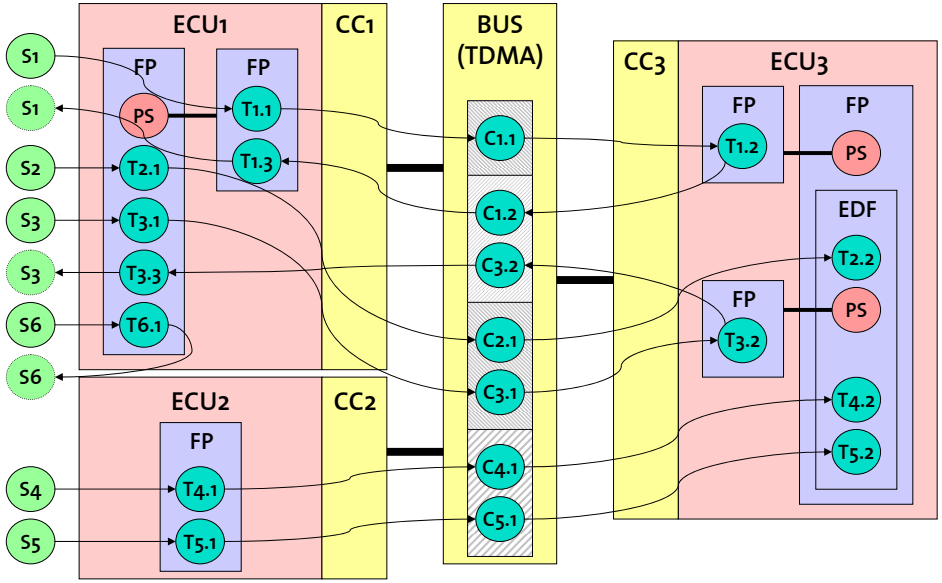
Task	e
T1.1	200
T1.2	300
T1.3	30
T2.1	75
T2.2	25
T3.1	60
T3.2	60
T3.3	40
T4.1	12
T4.2	2
T5.1	8
T5.2	3
T6.1	100

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

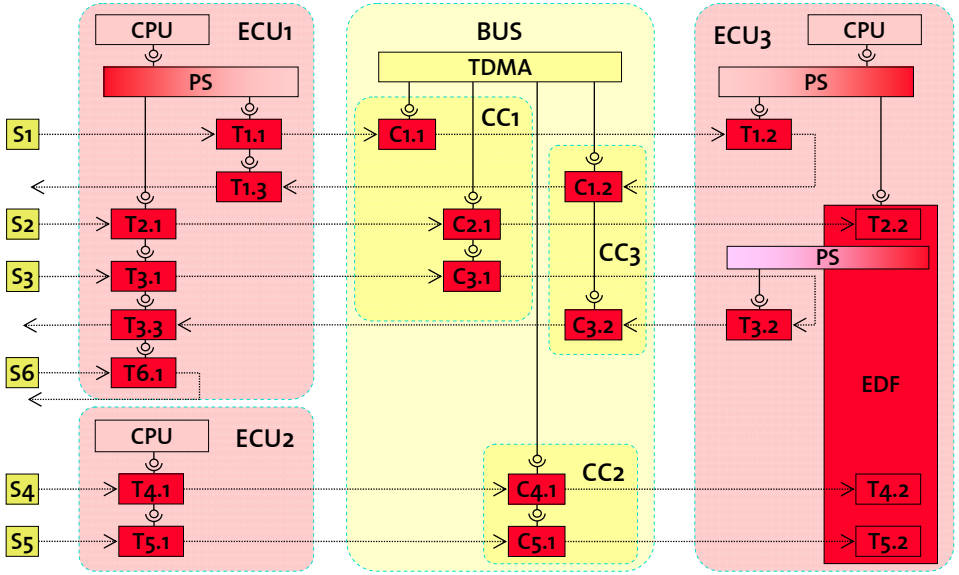
Perdiodic Server	p	e
SPS_{ECU1}	500	200
SPS_{ECU3}	500	250
DPS_{ECU3}	600	120

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

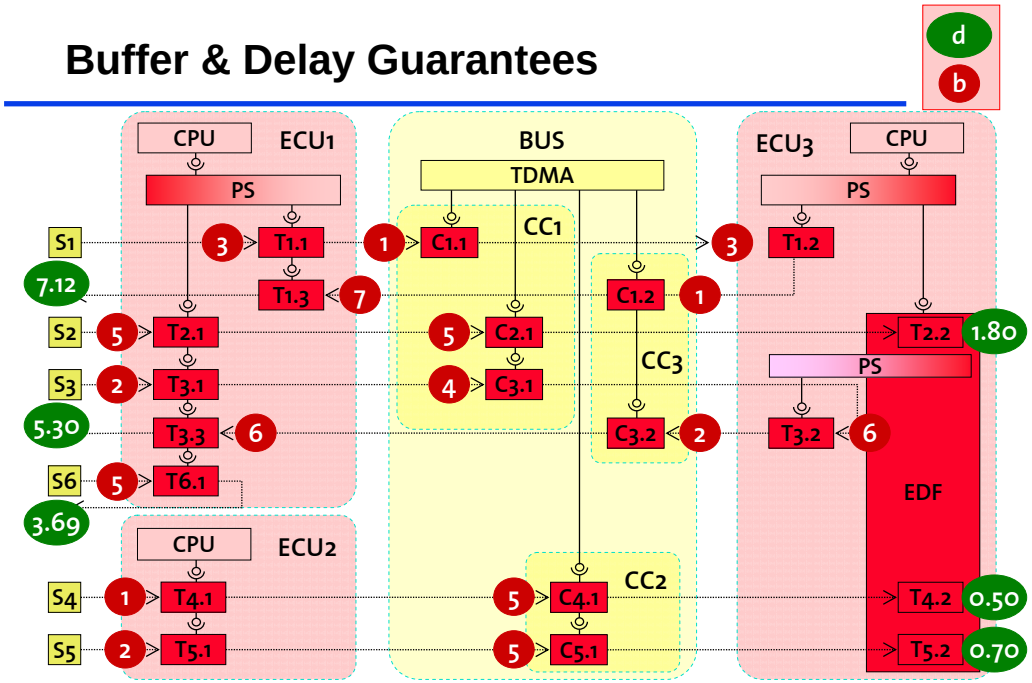
The Distributed Embedded System...



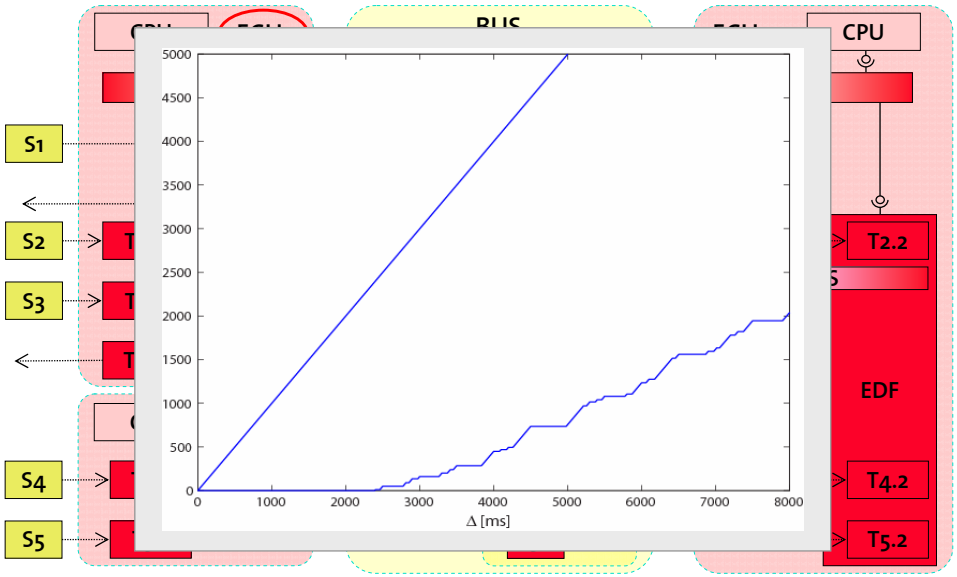
... and its MPA Model



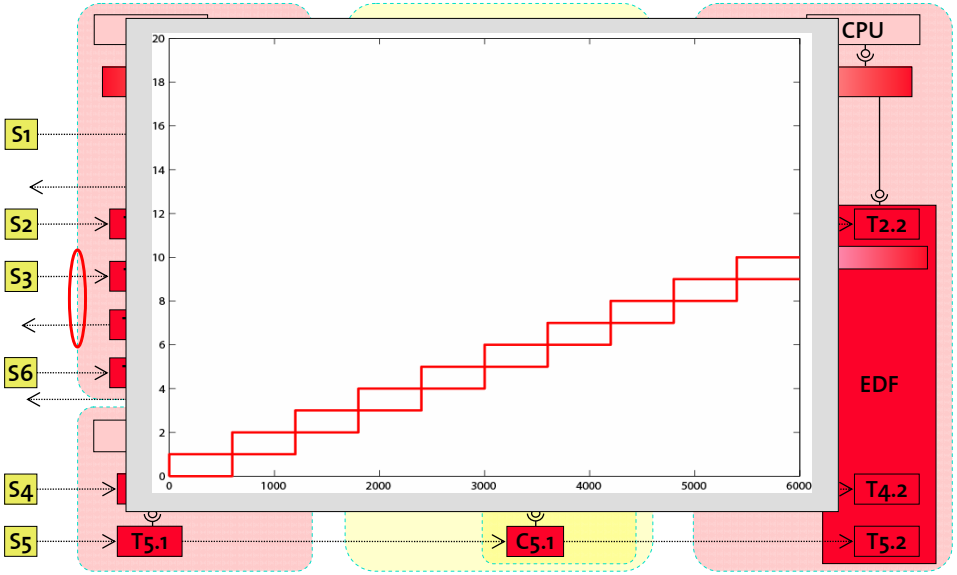
Buffer & Delay Guarantees



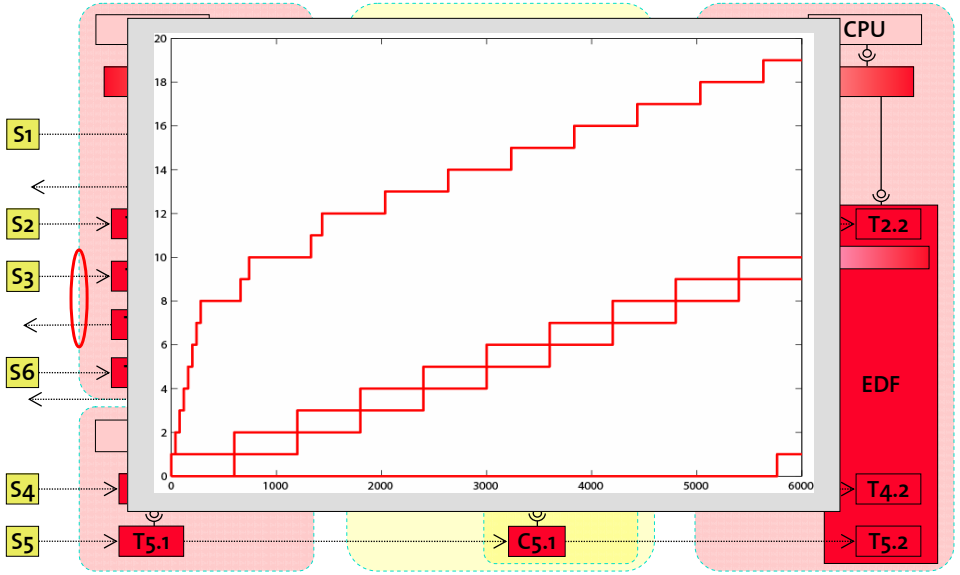
Available & Remaining Service of ECU1



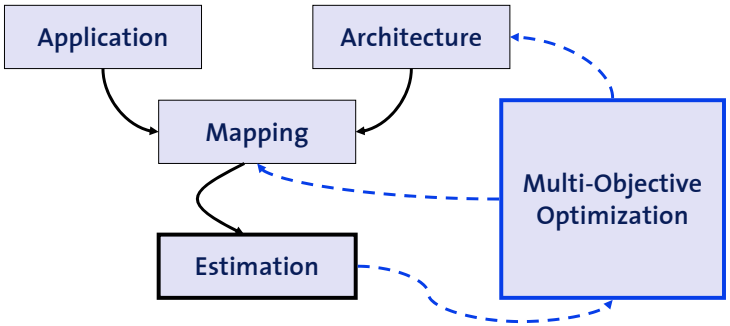
Input of Stream 3



Output of Stream 3

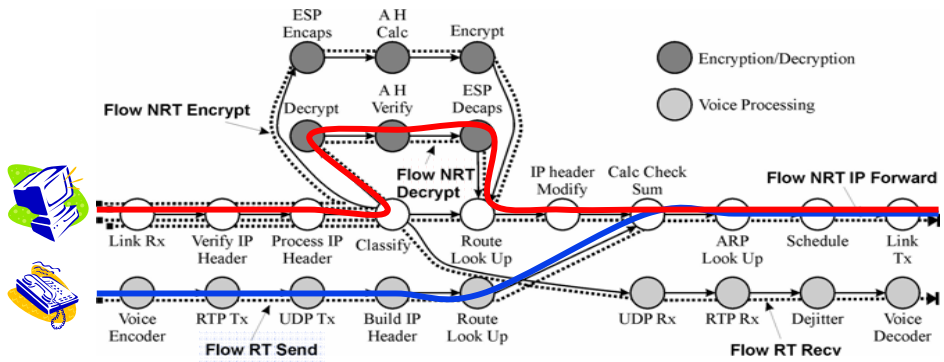


Automated Design Space Exploration

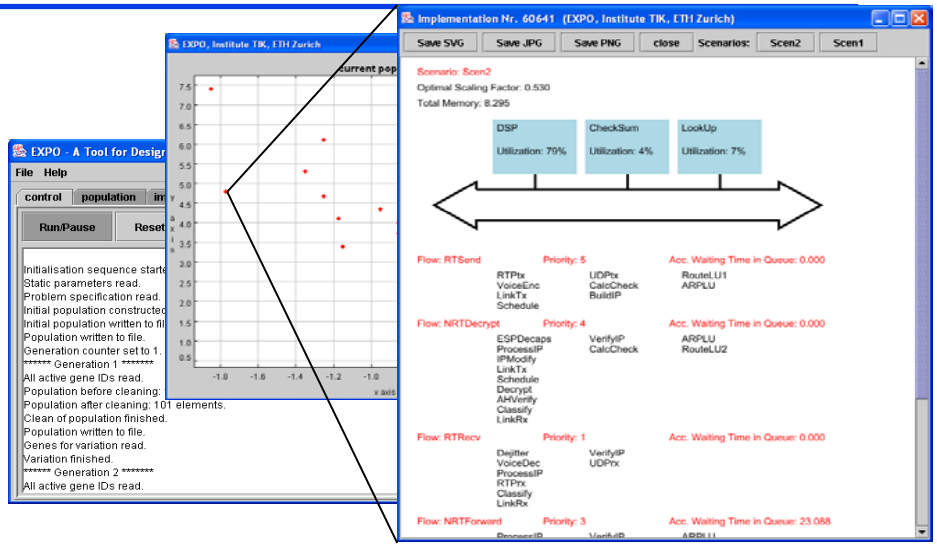


We use evolutionary algorithms for multi-objective optimization!

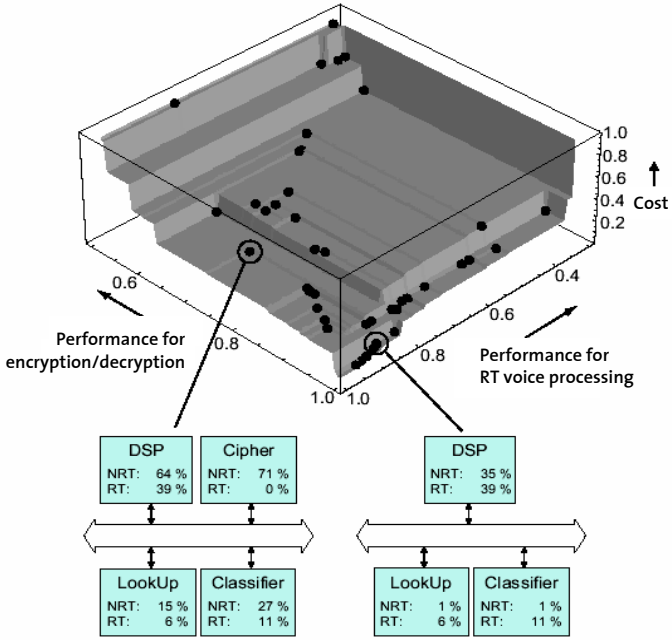
Network Processor Task Model



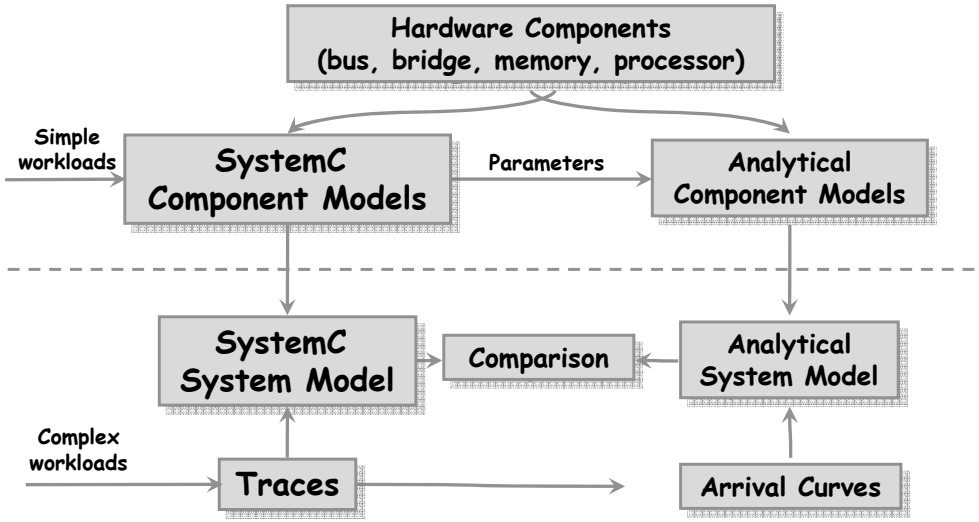
EXPO



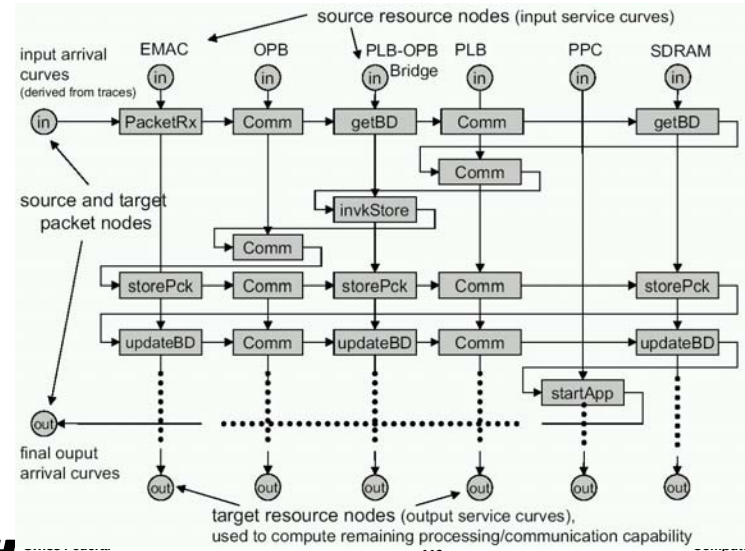
Results



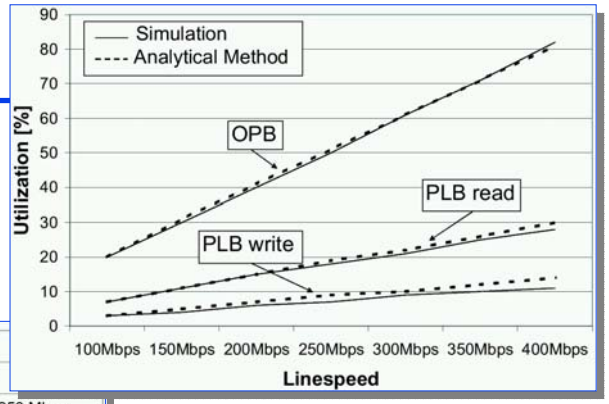
Validation Strategy (IBM)

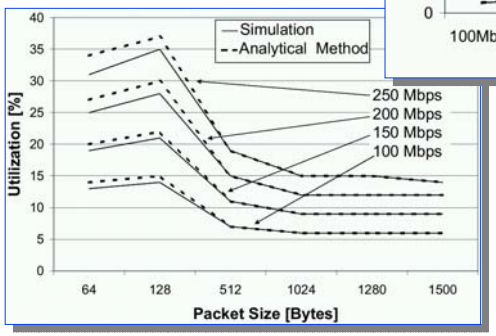


Analytical System Model



Comparison

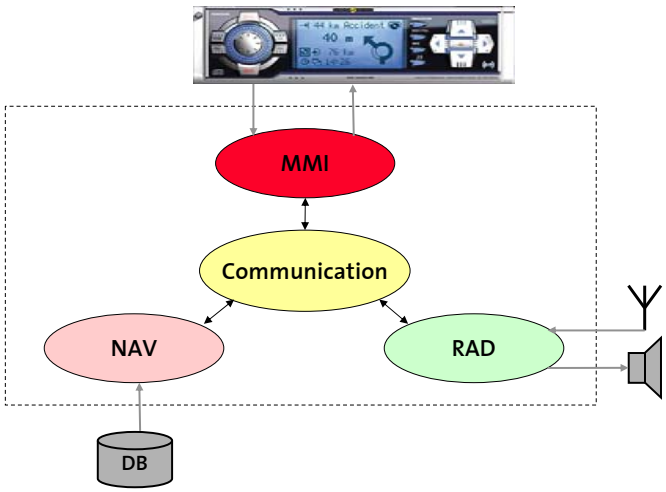




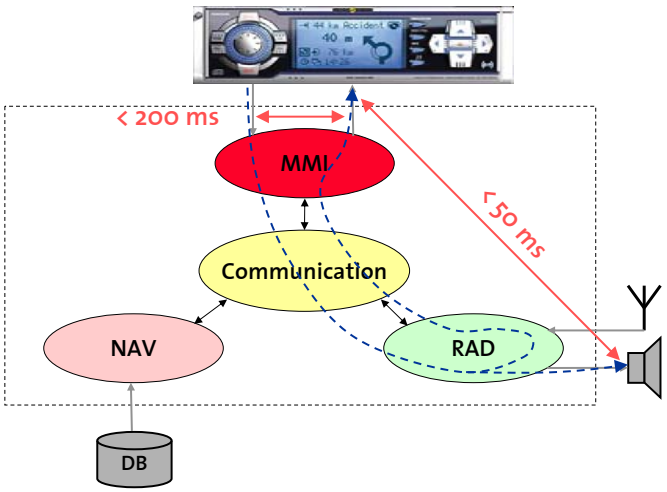
Application Scenario: In-Car Navigation System

- Car radio with navigation system
- User interface needs to be responsive
- Traffic messages (TMC) must be processed in a timely way
- Several applications may execute concurrently

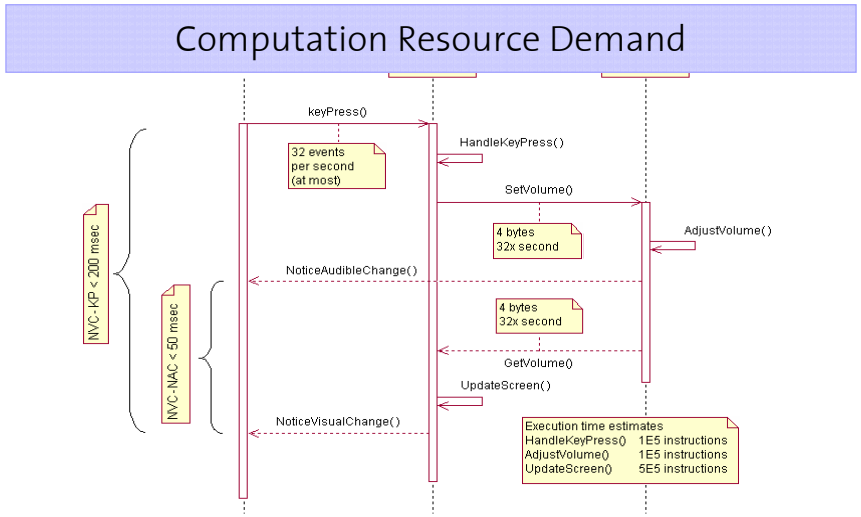
System Overview



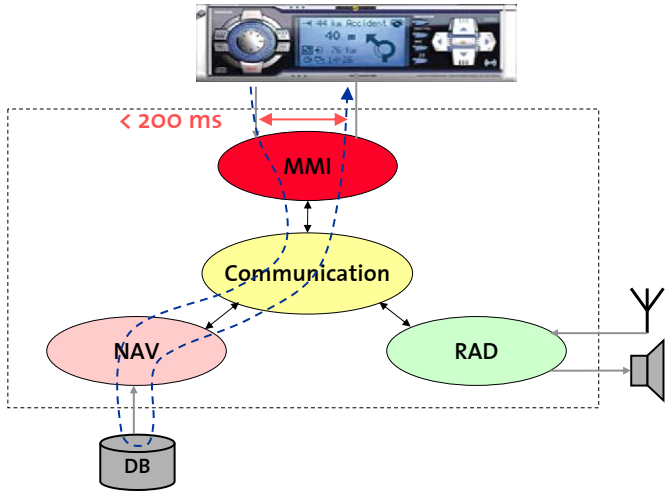
Application 1: Change Audio Volume



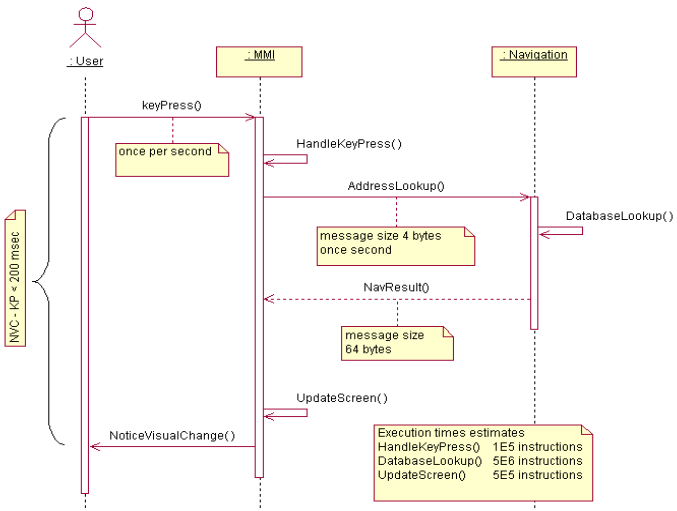
Application 1: Change Audio Volume



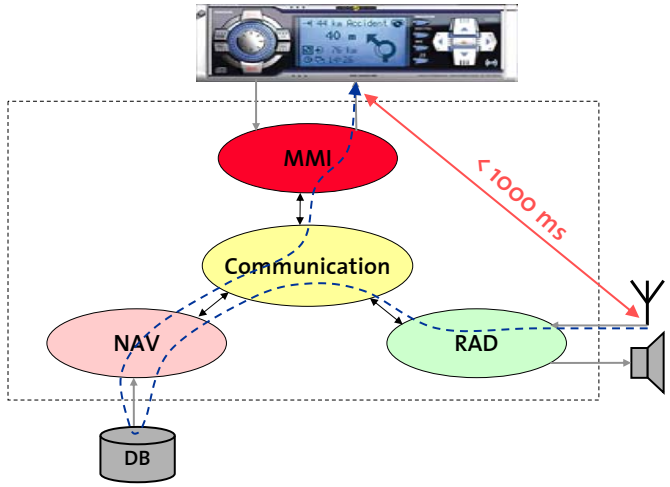
Application 2: Lookup Destination Address



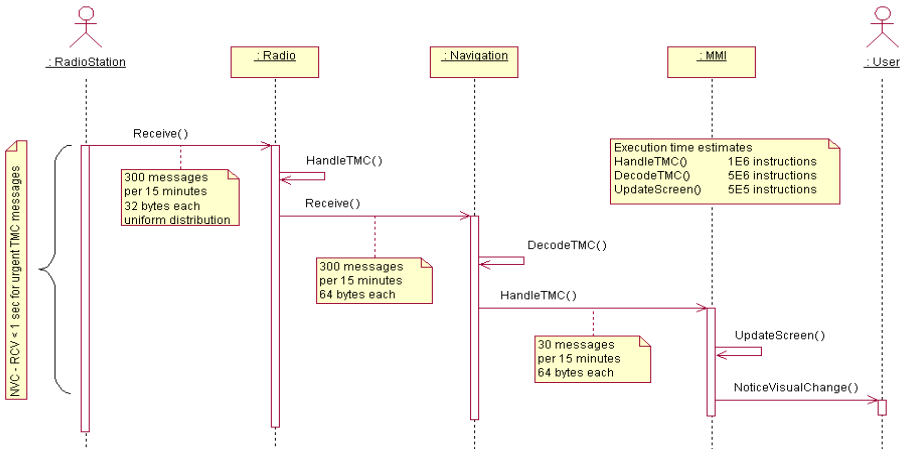
Application 2: Lookup Destination Address



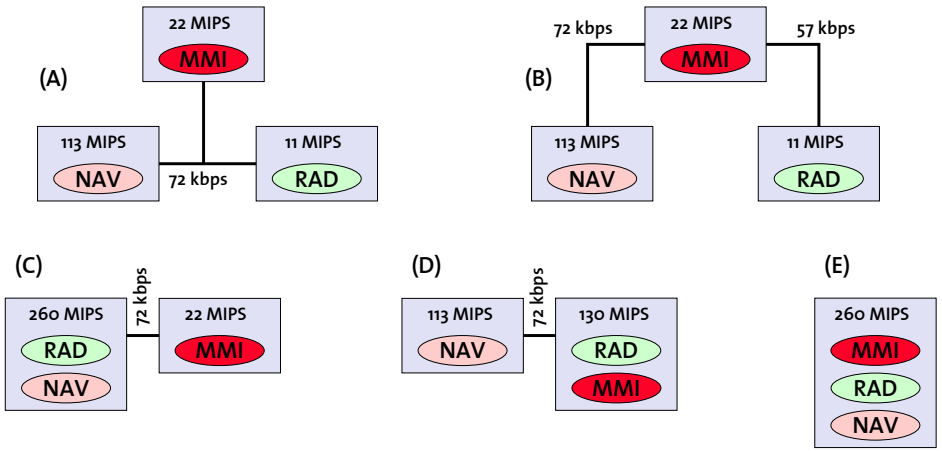
Application 3: Receive TMC Messages



Application 3: Receive TMC Messages



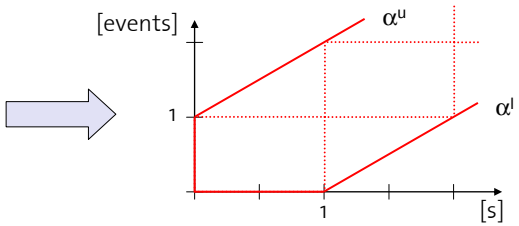
Proposed Architecture Alternatives



Step 1: Environment (Event Steams)

Event Stream Model

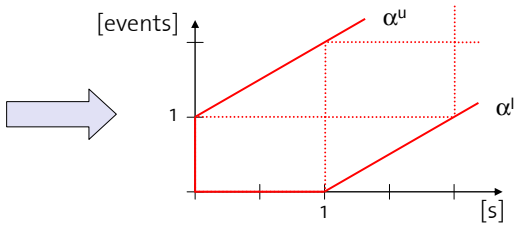
e.g. Address Lookup
(1 event / sec)



Step 2: Architectural Elements

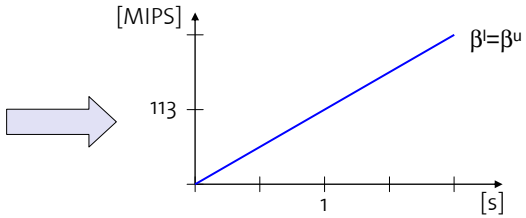
Event Stream Model

e.g. Address Lookup
(1 event / sec)



Resource Model

e.g. unloaded RISC CPU
(113 MIPS)

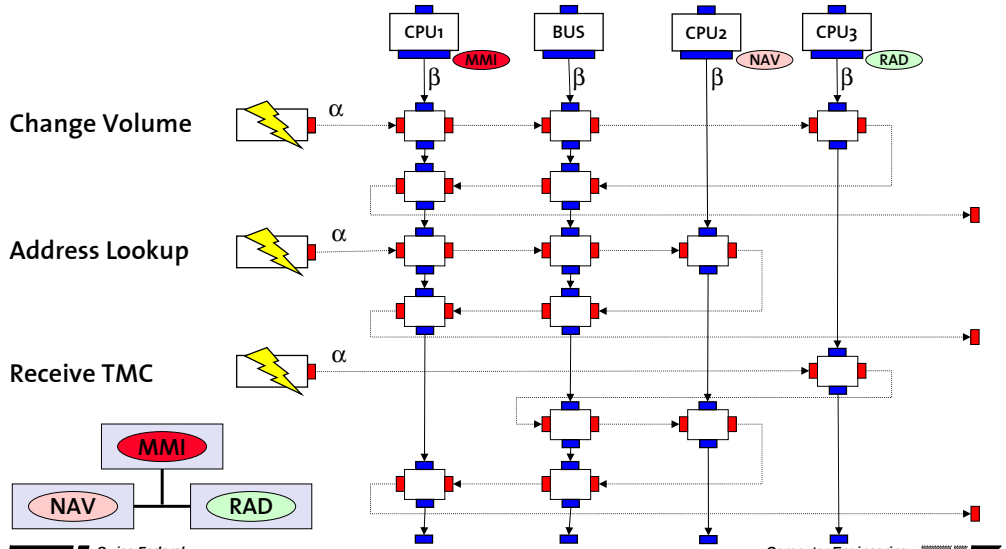


Step 3: Mapping / Scheduling

Rate Monotonic Scheduling
(Pre-emptive fixed priority scheduling):

- Priority 1: Change Volume (p=1/32 s)
- Priority 2: Address Lookup (p=1 s)
- Priority 3: Receive TMC (p=6 s)

Step 4: Performance Model

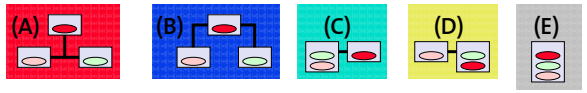


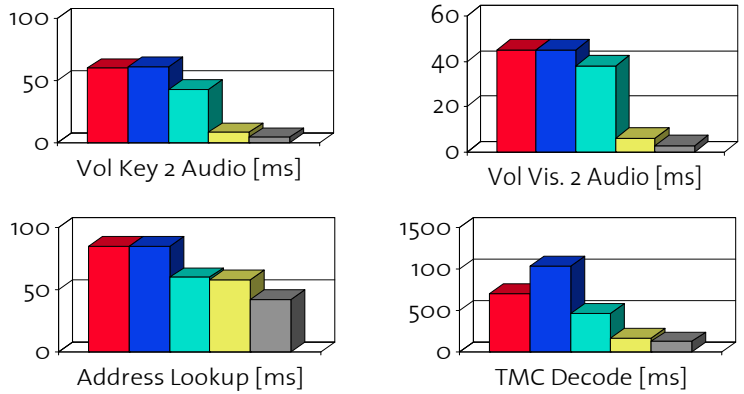
Analysis – Design Question 1

How do the proposed system architectures compare in respect to end-to-end delays?

Analysis – Design Question 1

End-to-end delays:

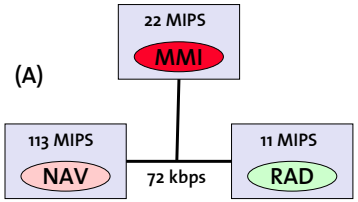




Analysis – Design Question 2

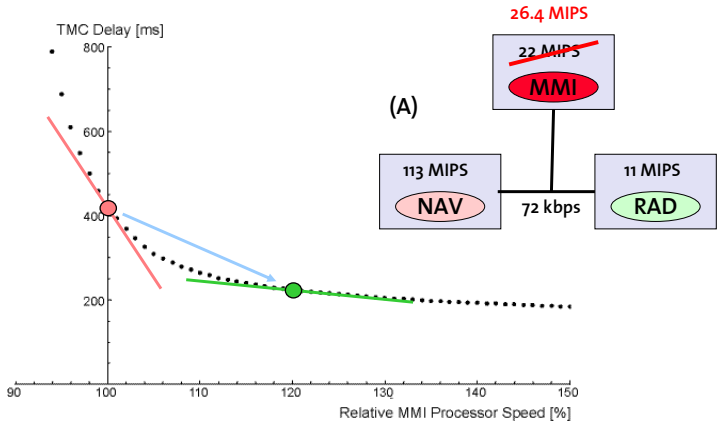
How robust is architecture A?

Where is the bottleneck of this architecture?



Analysis – Design Question 2

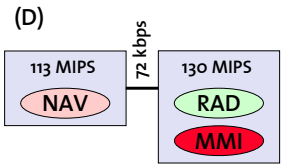
TMC delay vs. MMI processor speed:



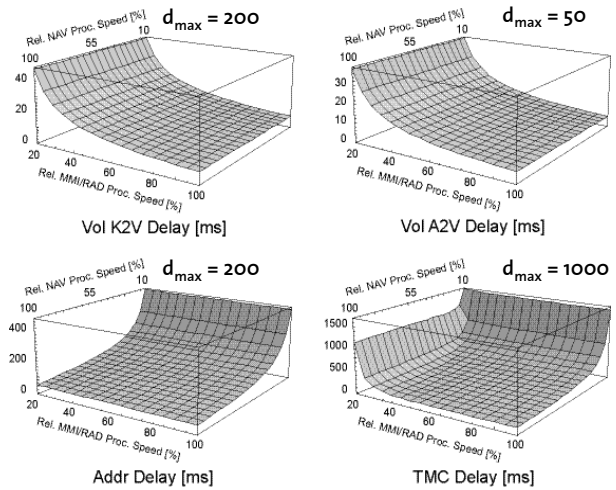
Analysis – Design Question 3

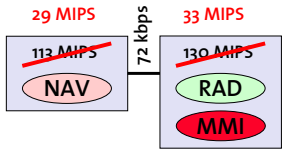
Architecture D is chosen for further investigation.

How should the processors be dimensioned?

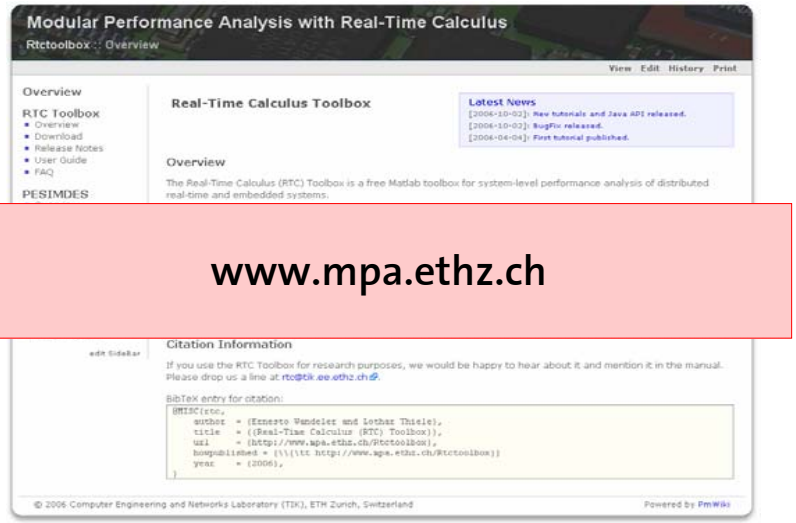


Analysis – Design Question 3





Implementation: RTC Toolbox
Publications on the Subject



Acknowledgement

- Martin Naedele
- Samarjit Chakraborty
- Alexander Maxiaguine
- Simon Kuenzli
- Ernesto Wandeler
- Nikolay Stoimenov
- Wolfgang Haid
- Simon Perathoner