

System Architecture Evaluation Using Modular Performance Analysis - A Case Study

Marcel Verhoef (Chess Information Technology, NL)
Ernesto Wandeler, Lothar Thiele (ETH Zürich, CH)
Paul Lieverse (Siemens VDO Automotive, NL)



Contents of this talk

- System Architecture Evaluation
- Modular Performance Analysis
- Case Study – In Car Radio Navigation System
- Conclusions & Future Work

What is performance analysis? (1)

Performance* :

An indicator of how well a system meets its requirements for timeliness.

Timeliness :

An indicator of system response time and / or throughput

Caveat:

There are many different performance indicators, most of them express *non-functional* properties

* Smith & Williams, "Performance Solutions – A Practical Guide to Creating Responsive, Scalable Software", 2002

Business perspective (1)

- Why System Architecture Evaluation?
 - continuous increase in functionality demands
 - continuous drive to reduce cost price
 - tighter time-to-market demands
 - rapidly evolving technology
- over dimensioning not longer viable (\$)
- need for early design choice impact analysis

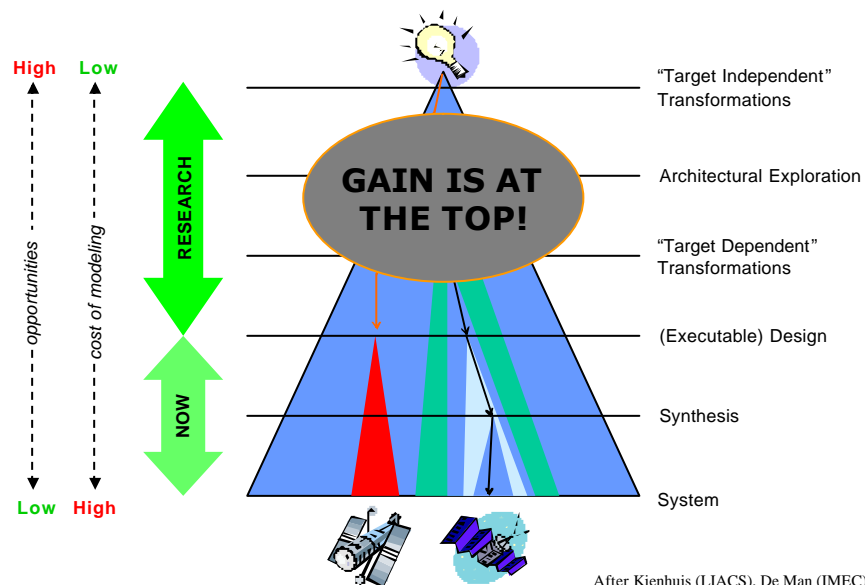
Business perspective (2)

“Does The Product Work?”



“Does The Product Work Given a Set of Hard Resource Constraints?”

Fighting The Complexity Battle

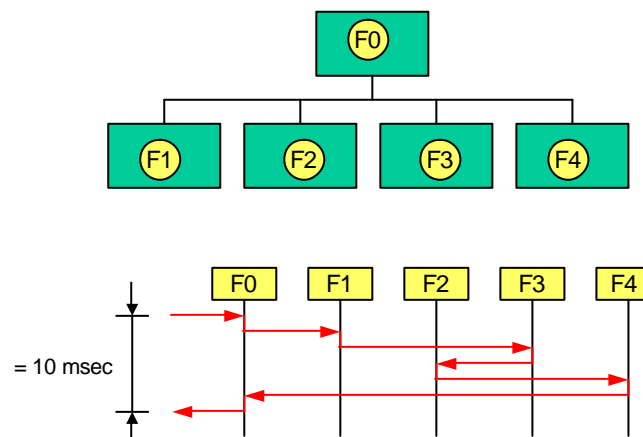


Business perspective (4)

- finding quantitative answers in the early life cycle is very hard, there are many unknowns
- “shooting at a moving target”
- need for a light-weight approach that can deal with highly interactive nature of the design process

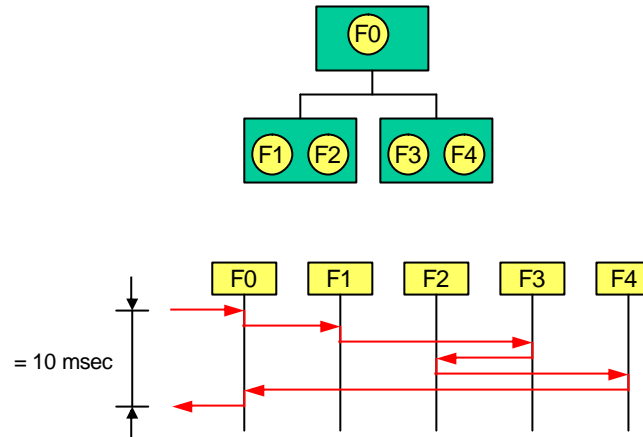
Target practice (1)

development or engineering prototype



Target practice (2)

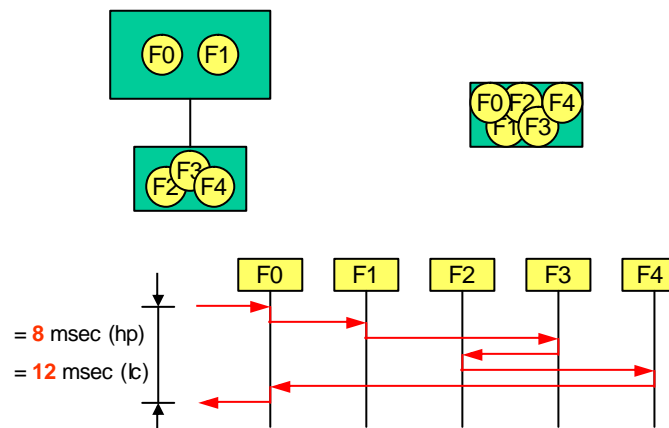
final engineered product



Target practice (3)

high performance

low-cost

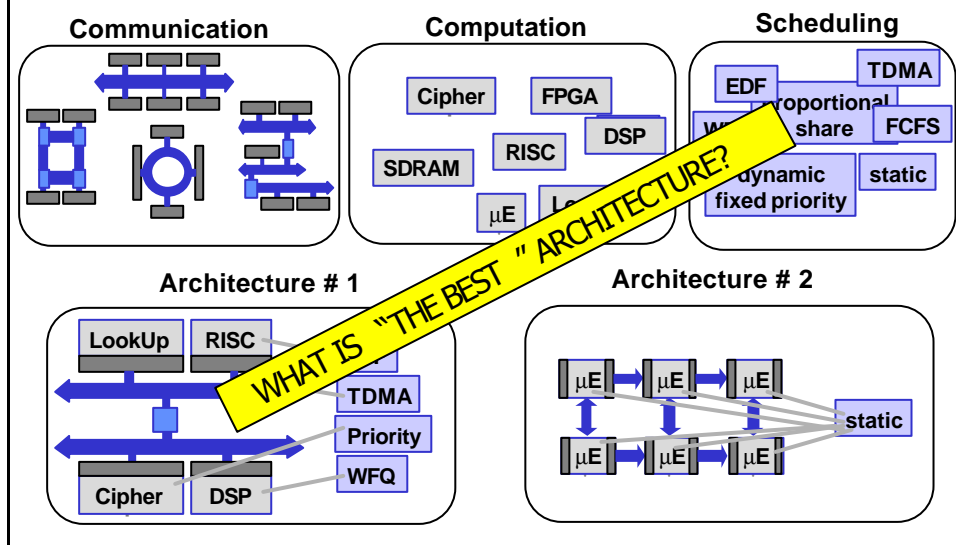


Is performance analysis important?

Clear need to assess the impact of
architecture alternatives during
the **whole** life-cycle

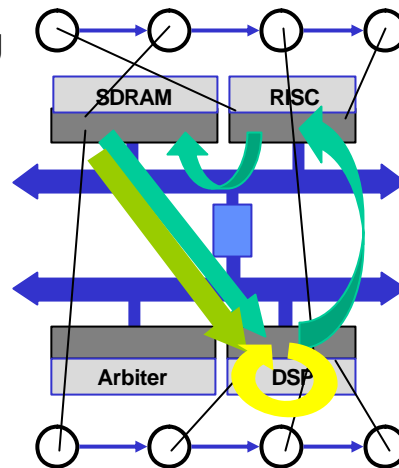
but how do we do this?

System Architecture Evaluation (1)



System Architecture Evaluation (2)

- Distributed processing on different resources
- Interactions



System Architecture Evaluation (3)

- Simulation Based Methods
 - Detailed, explicit, models are required to generate sufficiently useful results
 - Building models is in general costly, as is their evaluation and interpretation
 - No hard guarantee to find the best- or worst case
- Queuing Networks
 - Probabilistic approach causes state space explosion if level of model detail is increased
 - Exponential distributions used in Markov chain analysis does not accurately reflect real-life systems properties

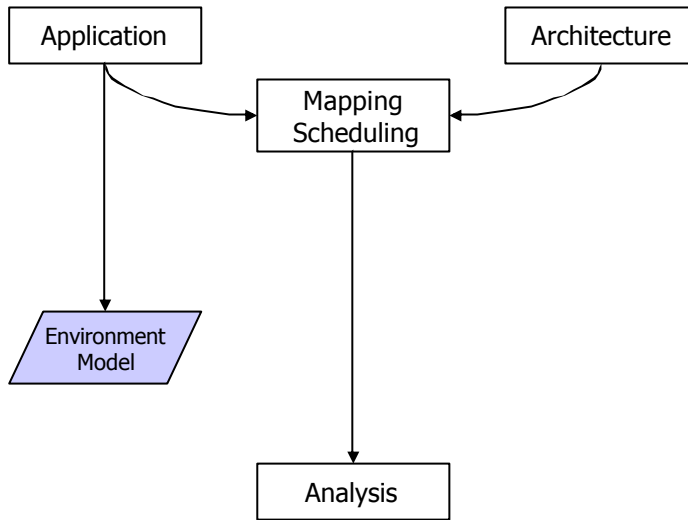
Modular Performance Analysis (1)

- Based on Real-Time Calculus [Thiele-2000/2004]
- Real-Time Calculus extends the well-known Network Calculus [LNCS2050-2001] and early work by [Cruz-1991]
 - RTC deals with both **computation** and **communication** in a single mathematical framework
 - Implements standard event models: periodic, periodic with jitter, periodic with bursts and sporadic
- Network Calculus uses max-plus algebra to compute the results [Bacelli-1992]

Modular Performance Analysis (2)

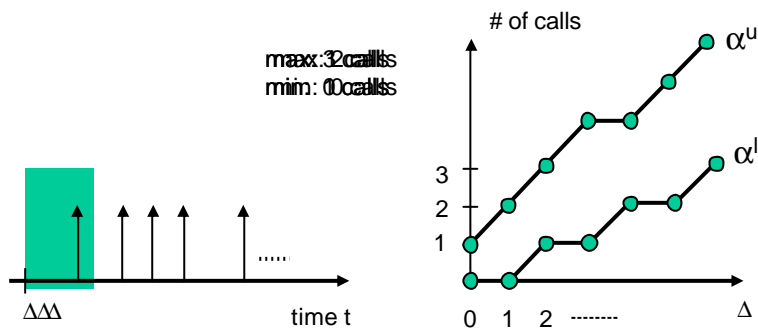
- Characterizes a system by describing
 - How many resources are needed to fulfill a function?
 - How often are functions needed?
 - When are resources available?
- Compositional, heterogeneous
 - Compose networks of MPA elements
 - Decompose MPA elements into MPA element networks
- Deterministic, analytic, algorithm (no simulation)
- Hard upper- and lower bounds are always found

Modular Performance Analysis (3)



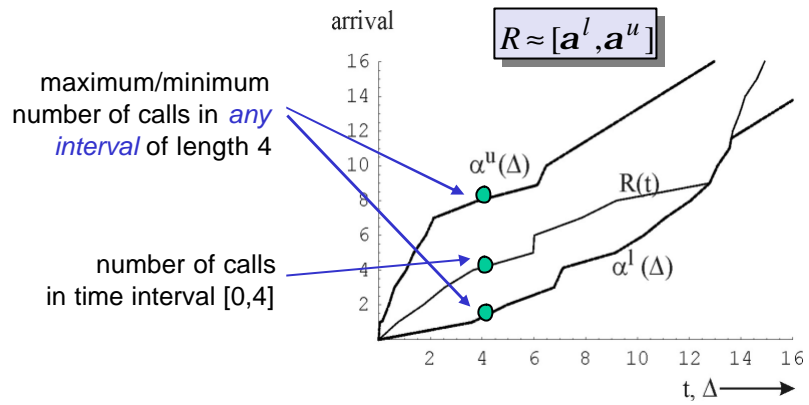
MPA – Environment model (3.1)

We use so-called **arrival curves** to describe abstract event streams

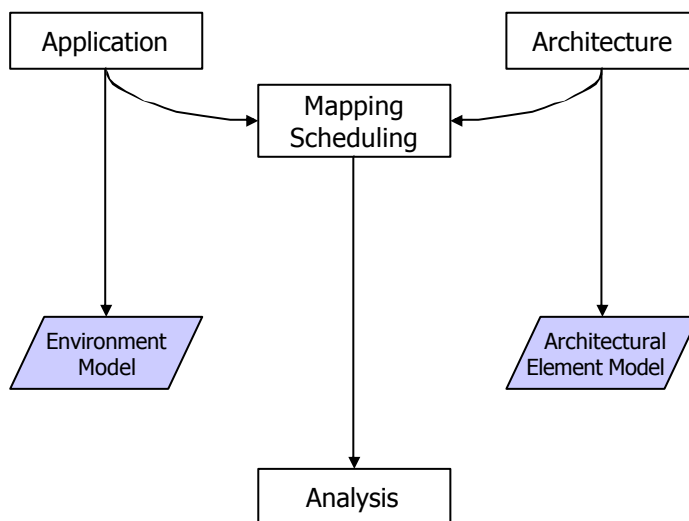


MPA – Environment model (3.2)

Upper and lower arrival curves

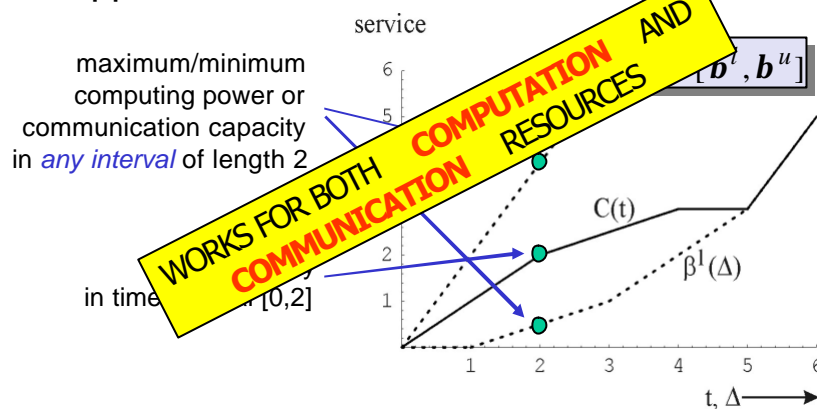


Modular Performance Analysis (4)

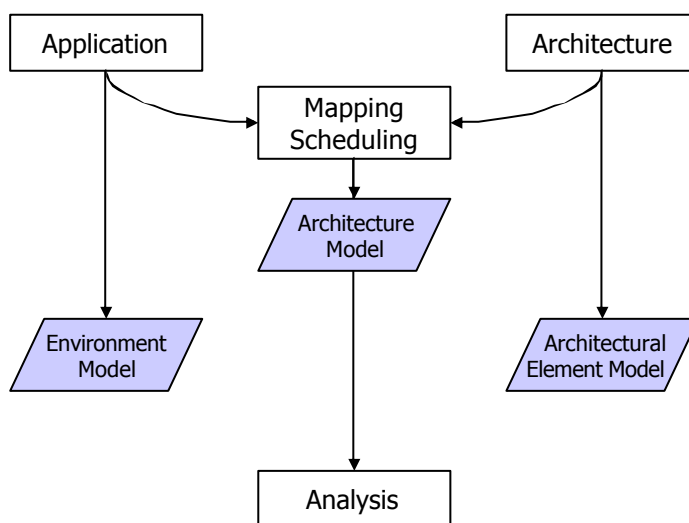


MPA – Environment model (4.1)

- Upper and lower service curves

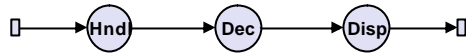


Modular Performance Analysis (5)



MPA – System Architecture Model (5.1)

Application



MPA – System Architecture Model (5.2)

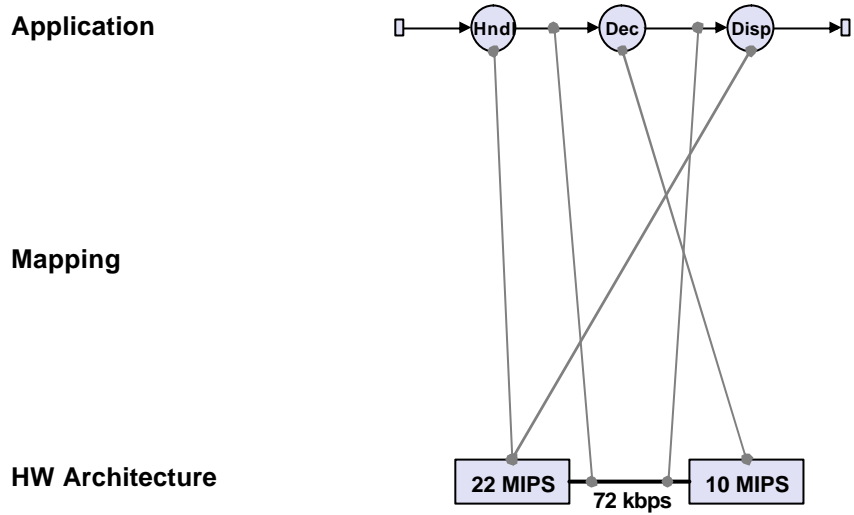
Application



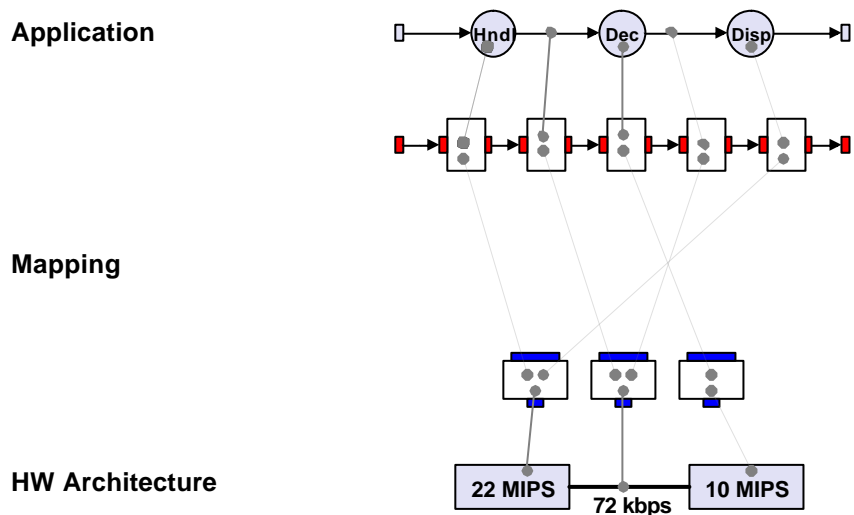
HW Architecture



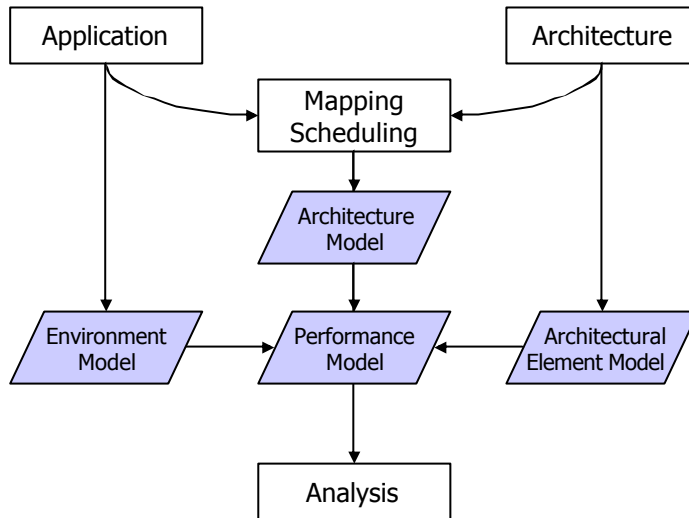
MPA – System Architecture Model (5.3)



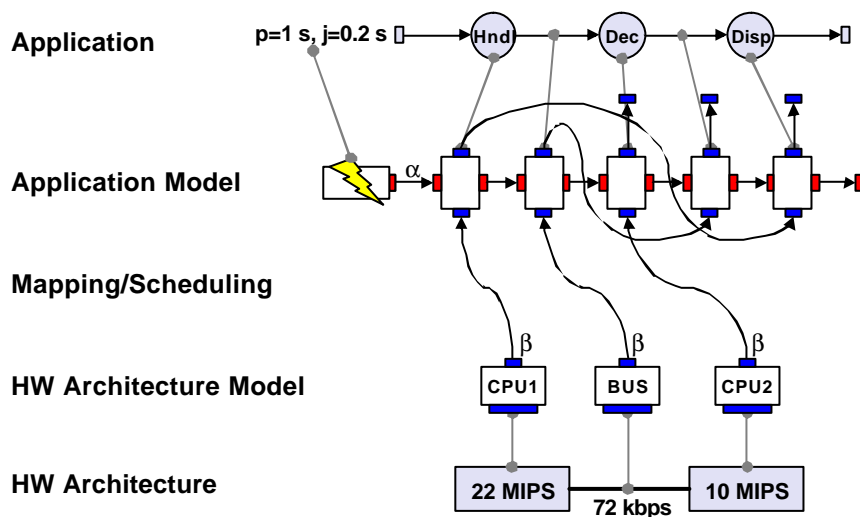
MPA – System Architecture Model (5.4)



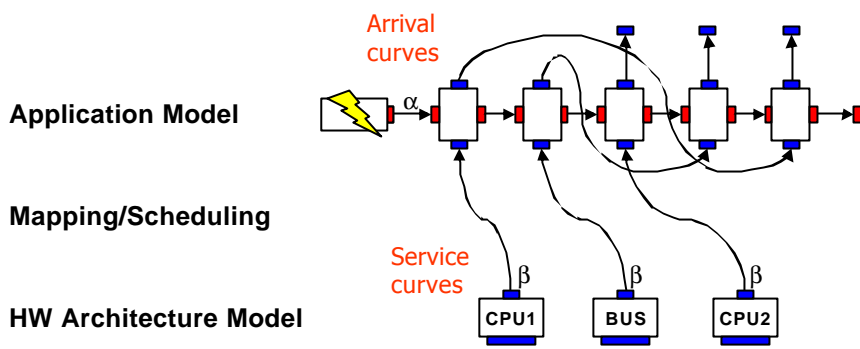
Modular Performance Analysis (6)



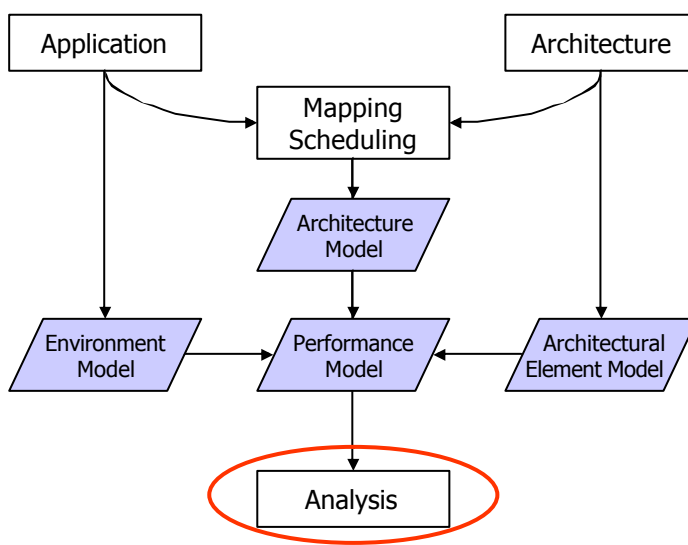
MPA – Performance Model (6.1)



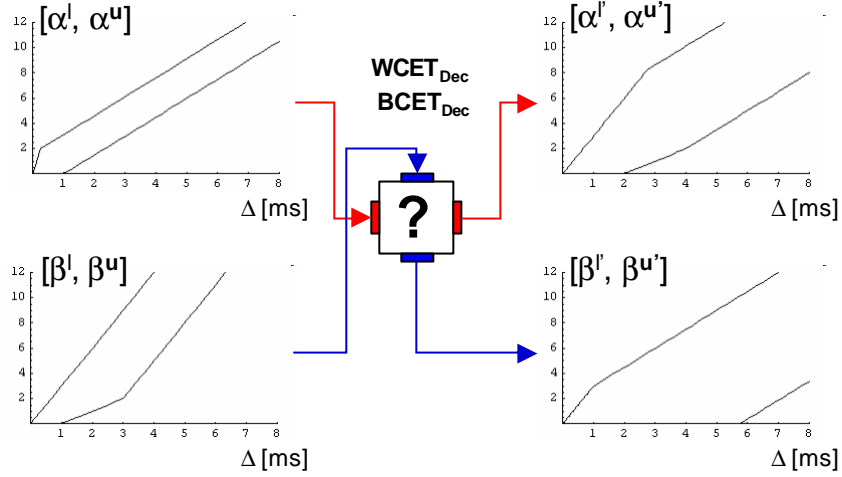
MPA – Performance Model (6.2)



Modular Performance Analysis (7)



MPA – Analysis (7.2)

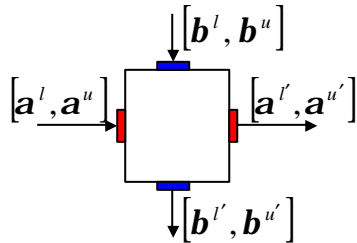


MPA – Analysis (7.3)

$$v(\Delta) \wedge w(\Delta) = \min \{v(\Delta), w(\Delta)\}$$

$$v(\Delta) \oplus w(\Delta) = \inf_{0 \leq I \leq \Delta} \{v(I) + w(\Delta - I)\} \quad v(\Delta) \otimes w(\Delta) = \inf_{0 \leq I} \{v(\Delta + I) - w(I)\}$$

$$v(\Delta) \bar{\oplus} w(\Delta) = \sup_{0 \leq I \leq \Delta} \{v(I) + w(\Delta - I)\} \quad v(\Delta) \bar{\otimes} w(\Delta) = \sup_{0 \leq I} \{v(\Delta + I) - w(I)\}$$



$$a^{u'} = [(a^u \oplus b^u) \bar{\otimes} b^l] \wedge b^u$$

$$a^{l'} = [(a^l \bar{\otimes} b^u) \oplus b^l] \wedge b^l$$

$$b^{u'} = (b^u - a^l) \bar{\otimes} 0$$

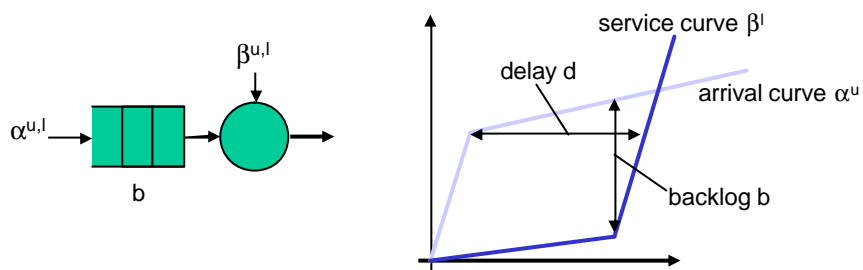
$$b^{l'} = (b^l - a^u) \bar{\oplus} 0$$

MPA – Analysis (7.4)

Delay and Memory

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

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

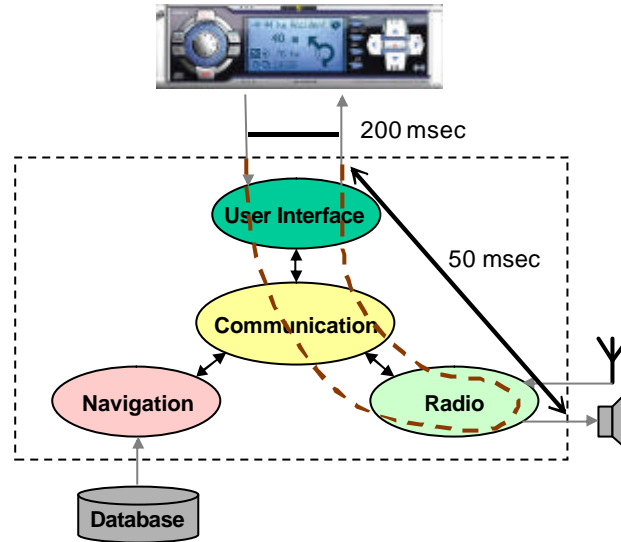


Case Study: In-Car Navigation System

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

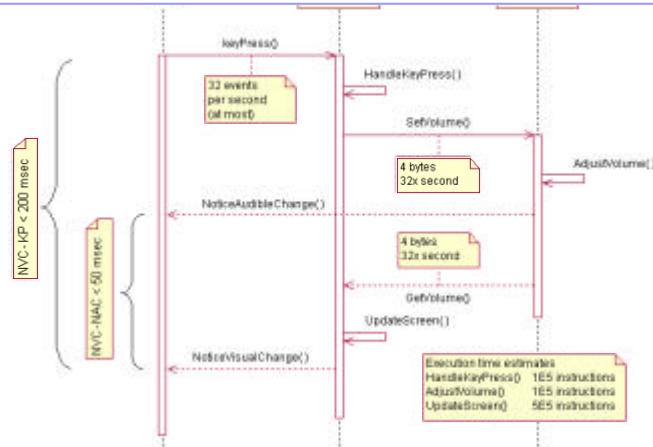


System Overview – Change Volume

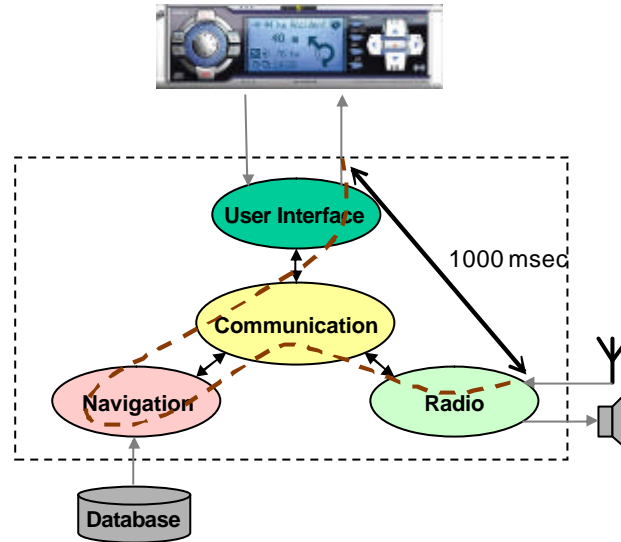


Application A: Change Audio Volume

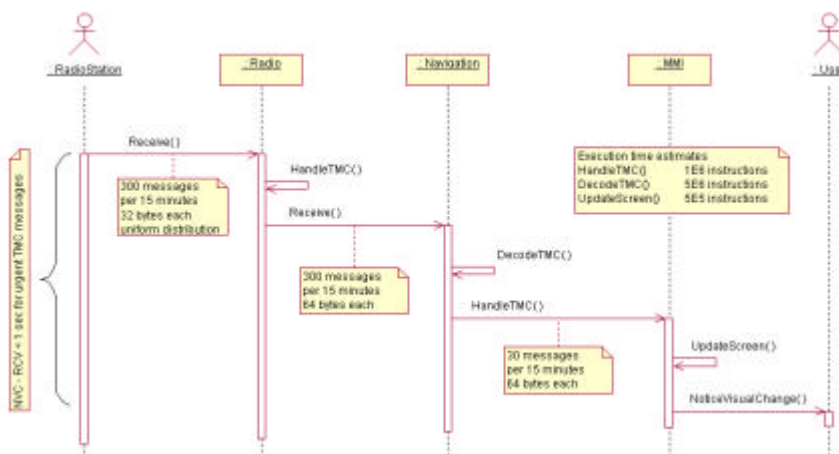
Computation Resource Demand



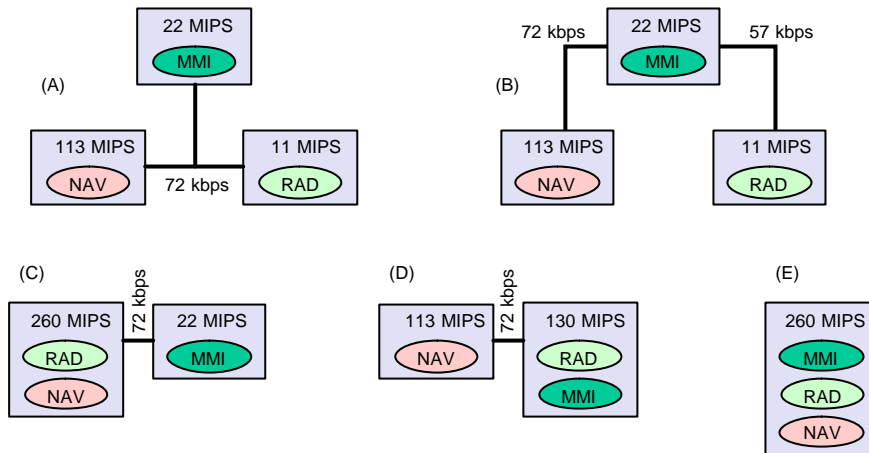
System Overview – Handle TMC



Application B: Handle TMC



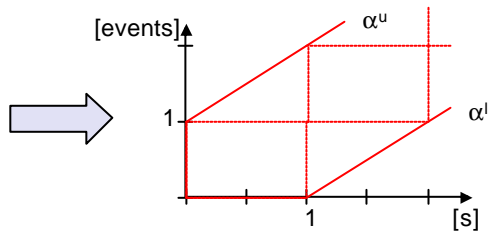
Proposed Architecture Alternatives



Step 1: Environment (Event Steams)

Event Stream Model

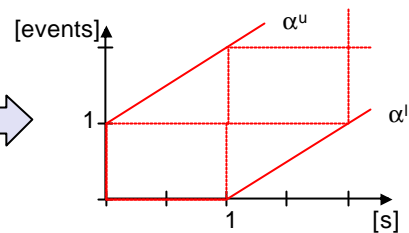
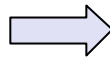
e.g. Address Lookup
(1 event / sec)



Step 2: Architectural Elements (Resources)

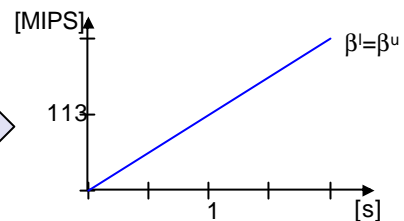
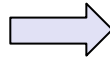
Event Stream Model

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



Resource Model

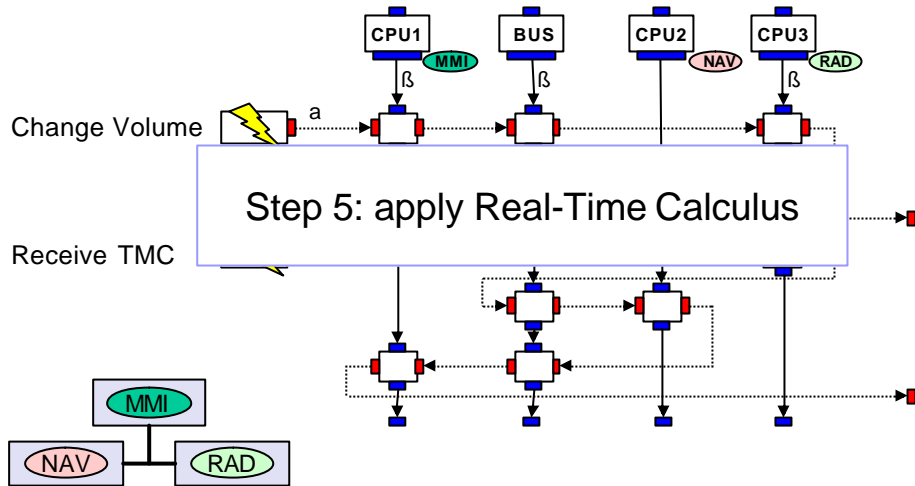
e.g. unloaded RISC CPU
(113 MIPS)



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)
- Consider scenario combinations
 - Change Volume / Receive TMC
 - Address Lookup / Receive TMC

Step 4: Construct performance model

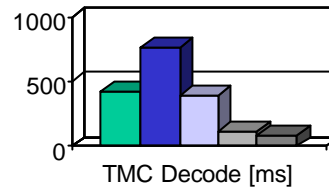
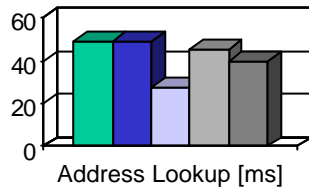
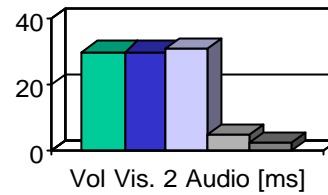
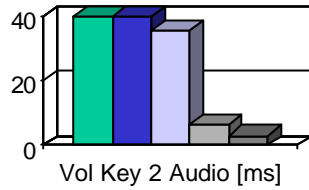


Analysis – Design Question 1

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

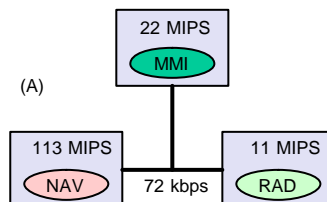
Analysis – Design Question 1

End-to-end delays:



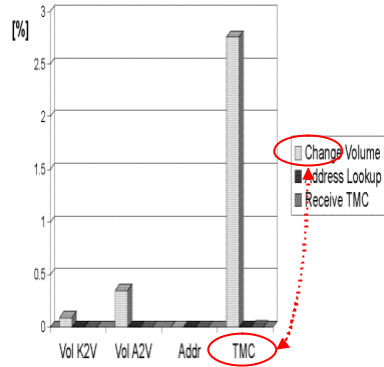
Analysis – Design Question 2

How robust is architecture A?
Where is the bottleneck of this architecture?

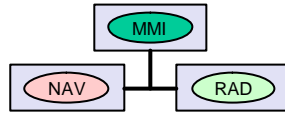
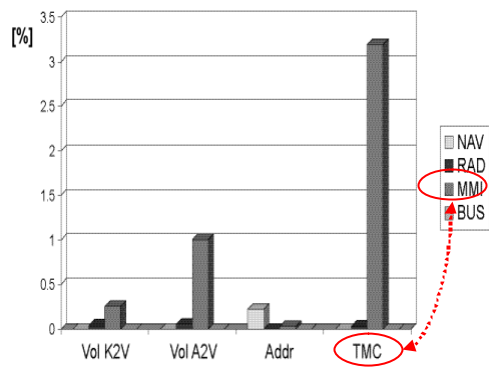


Analysis – Design Question 2

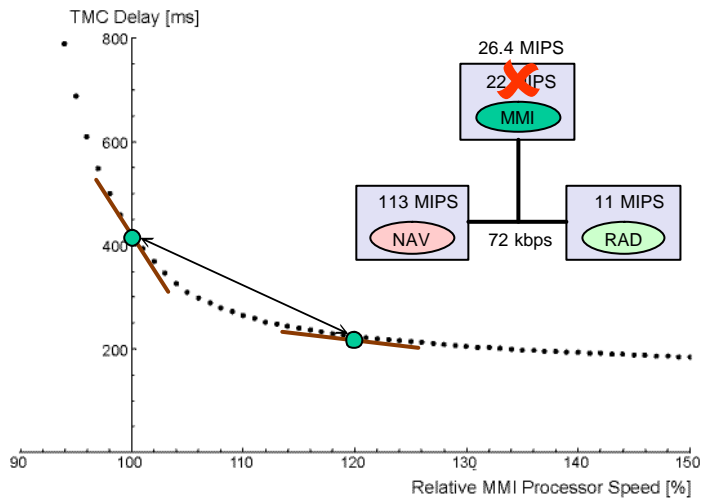
Sensitivity to input rate:



Sensitivity to resource capacity:

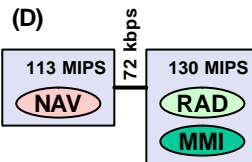


Analysis – Design Question 2

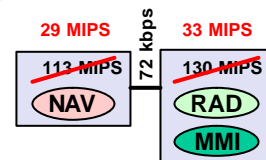
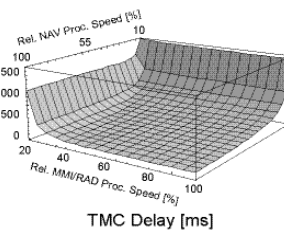
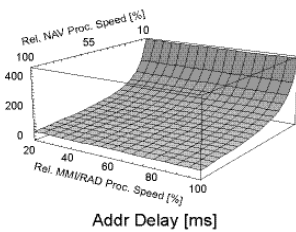
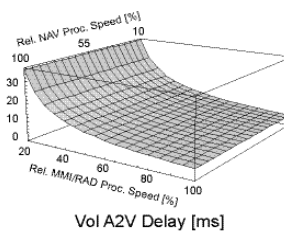
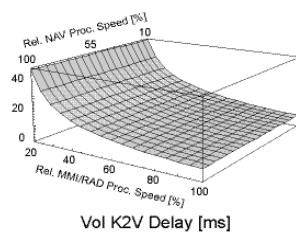


Analysis – Design Question 3

Architecture D is chosen for further investigation.
How should the processors be dimensioned?



Analysis – Design Question 3



Conclusions

- Easy to construct models (~ half day)
- Evaluation speed is fast and linear to model complexity (~ 1s per evaluation)
- Needs little information to construct early models (Fits early design cycle very well)
- Even though involved mathematics is very complex, the method is easy to use (Language of engineers)
- "light weight" formal method
- towards resource aware component models

Future Work

- Explore compositionality in more detail
- Quantitative comparison to other techniques
- Validation (comparison to measured data)
- Enhance Tool Support
- From UML to Performance Model...

Thank you for your attention!

Questions?

Marcel.Verhoef@chess.nl

Full paper available at

<http://www.esi.nl/boderc>