From WCET to System Level Analysis

Rolf Ernst, TU Braunschweig

Overview

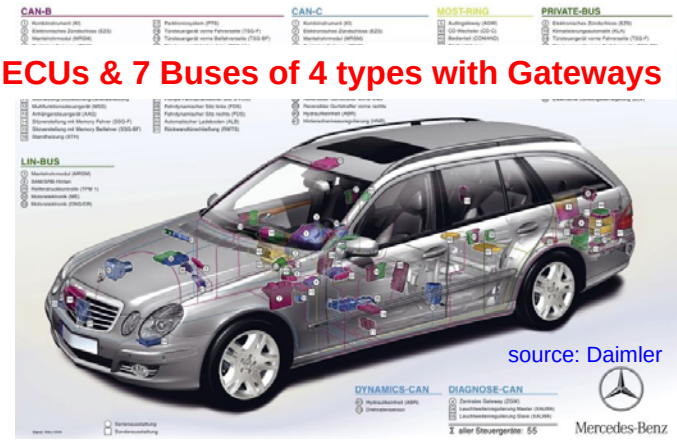
- performance challenges in automotive embedded system
- timing models and compositional analysis
- embedding in design process
- some recent enhancements
- conclusion

Overview

- performance challenges in automotive embedded system
- timing models and compositional analysis
- embedding in design process
- some recent enhancements
- conclusion

Formal Performance Analysis Applications

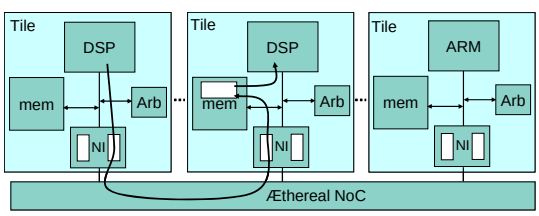
- formal models for performance analysis optimization are in use for very different types of embedded system
  - distributed networks



55 ECUs & 7 Buses of 4 types with Gateways

source: Daimler

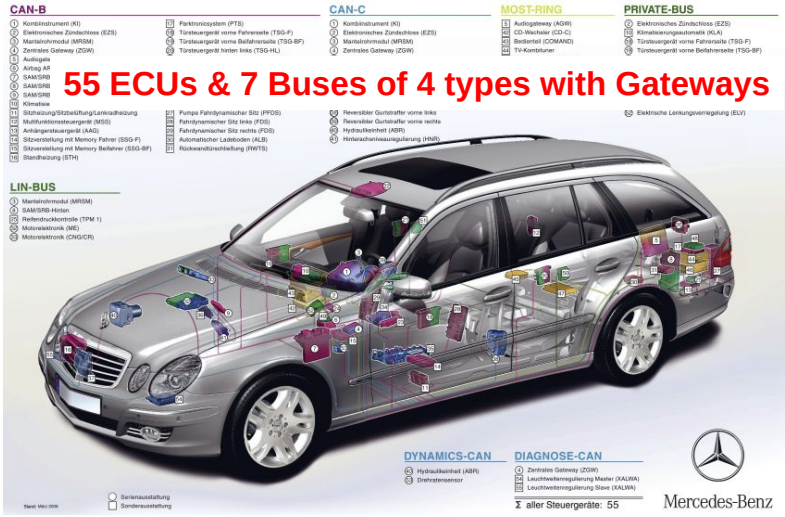
  - MpSoC



source: Bekooj – this tutorial

Example: Automotive Embedded Systems

- **Complexity challenge**
  - **Hundreds of functions**
  - **Networked control**
  - **Many suppliers**
  - **Heterogeneous**
- **Design challenges**
  - **supply chains**
  - **systems integration**
  - **verification**

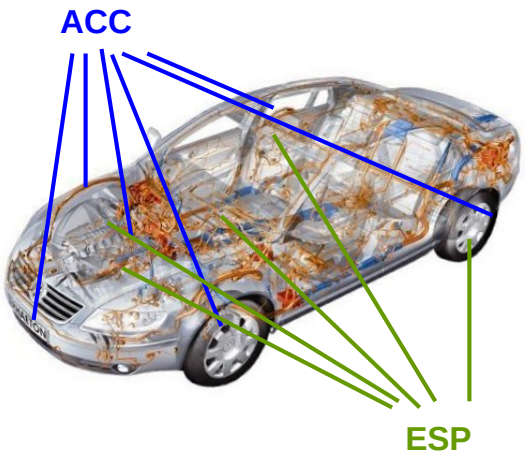


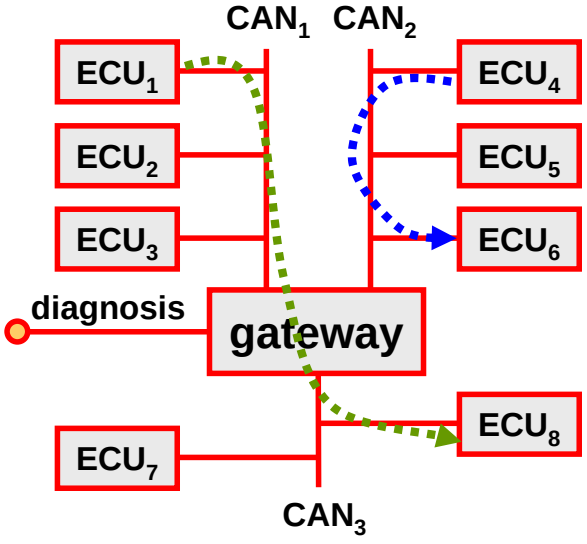
source: Daimler-Chrysler

- **Software standardization: OSEK → AUTOSAR**

Automotive Networked Systems

- **distributed networked system with complex end-to-end time constraints and numerous integrated functions on shared resources**





Automotive Design Chain

OEM

- BMW, Daimler, GM, PSA, Toyota, …
- global system, integration and network

specs ↓ ↑ ECUs

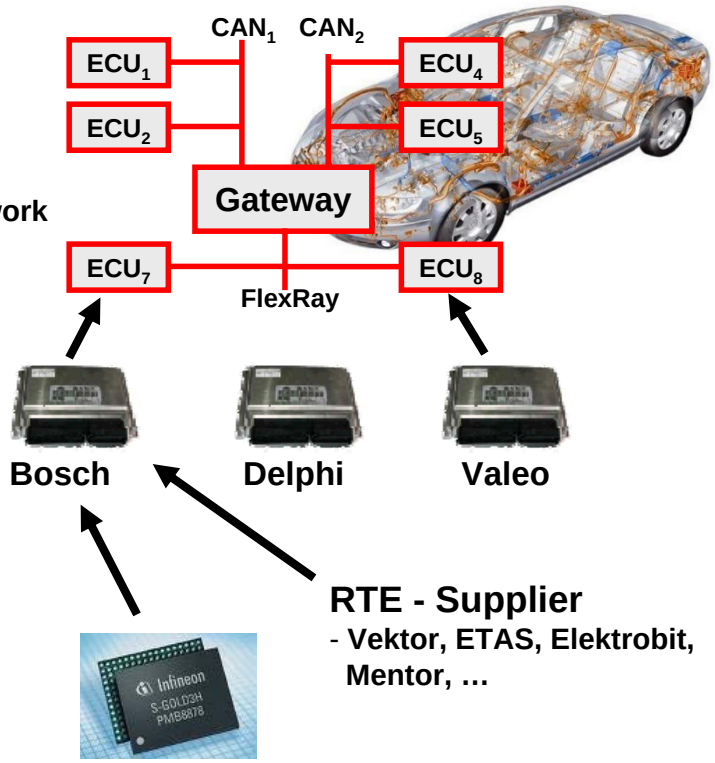
ECU - Supplier

- Bosch, Delphi, Valeo, …
- ECU responsibility

specs ↓ ↑ SoCs

HW Component - Supplier

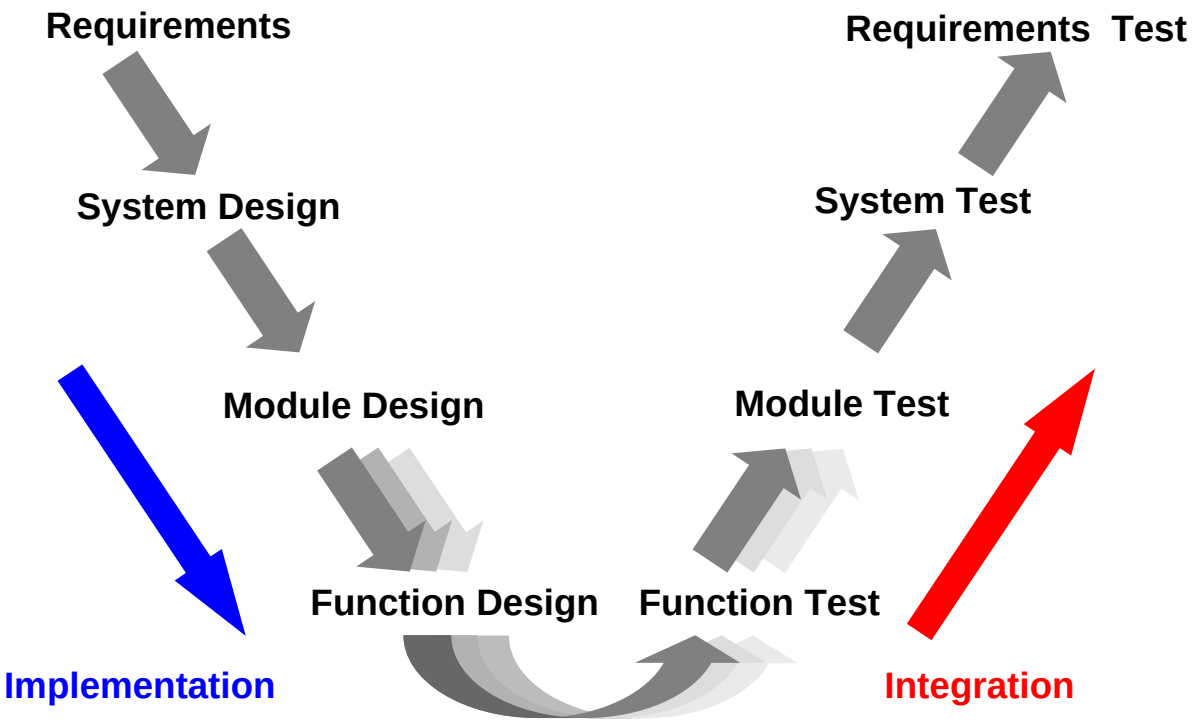
- Infineon, Freescale, ST, Toshiba, …



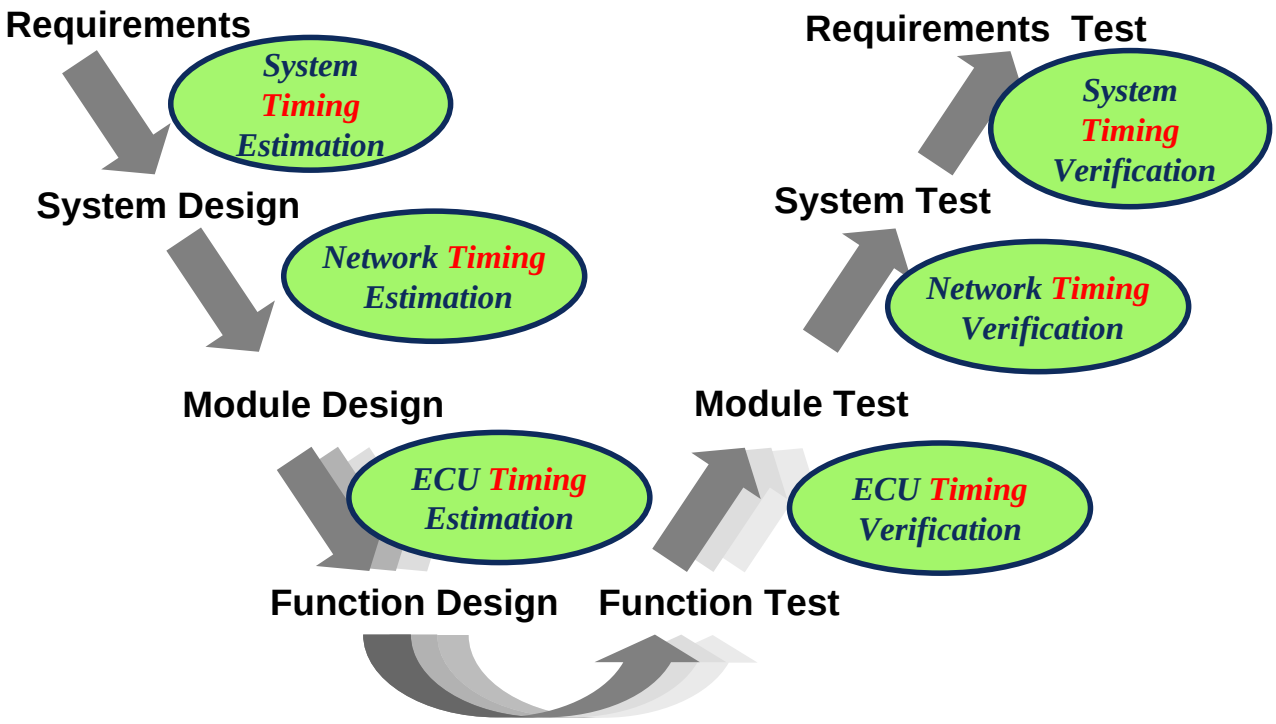
RTE - Supplier

- Vektor, ETAS, Elektrobit, Mentor, …

V Model



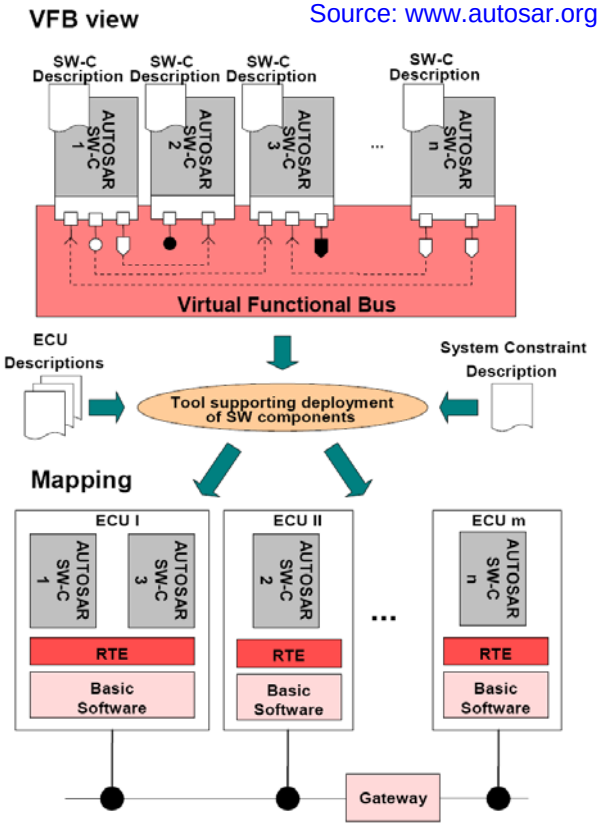
V – Model and Timing



AUTOSAR - Automotive Software Architecture

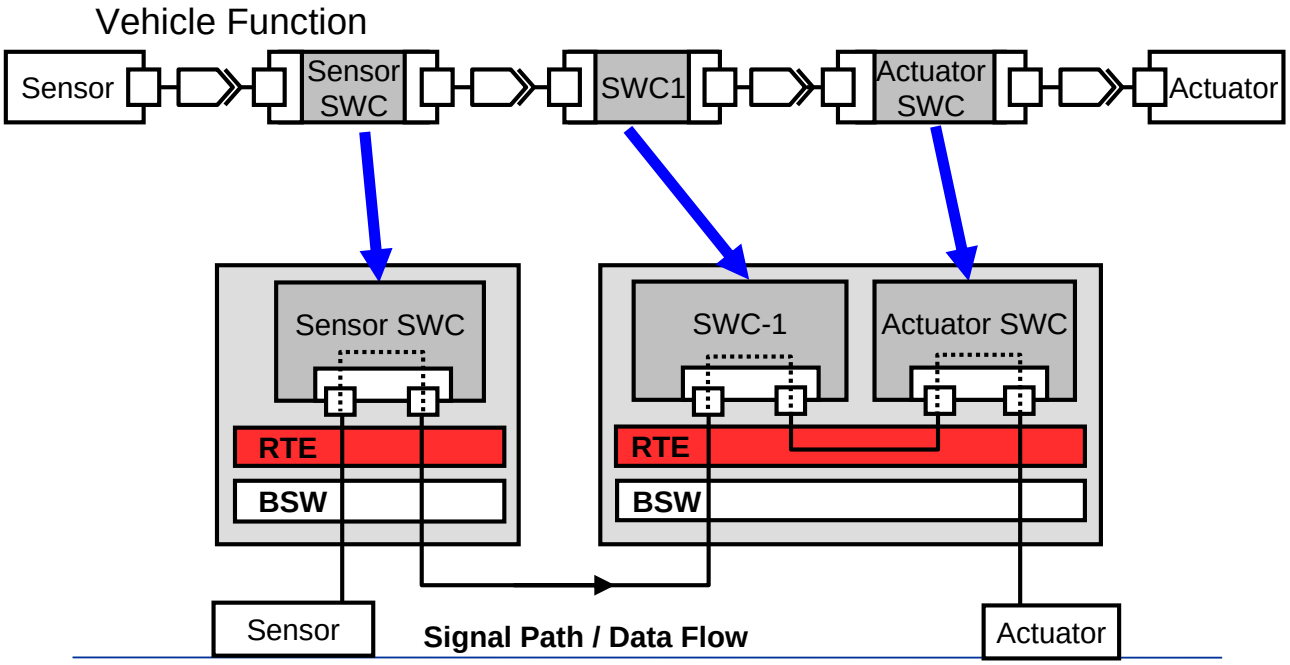
- SW-Components (SW-C)
  - encapsulate the applications
- Virtual Functional Bus (VFB)
  - communication mechanisms
  - interface to Basic SW
- Mapping
  - configuration and generation of RTE and Basic SW
- Runtime Environment (RTE)
  - VFB implementation on a specific ECU
- Basic Software (BSW)
  - infrastructural functionality on an ECU

Source: www.autosar.org



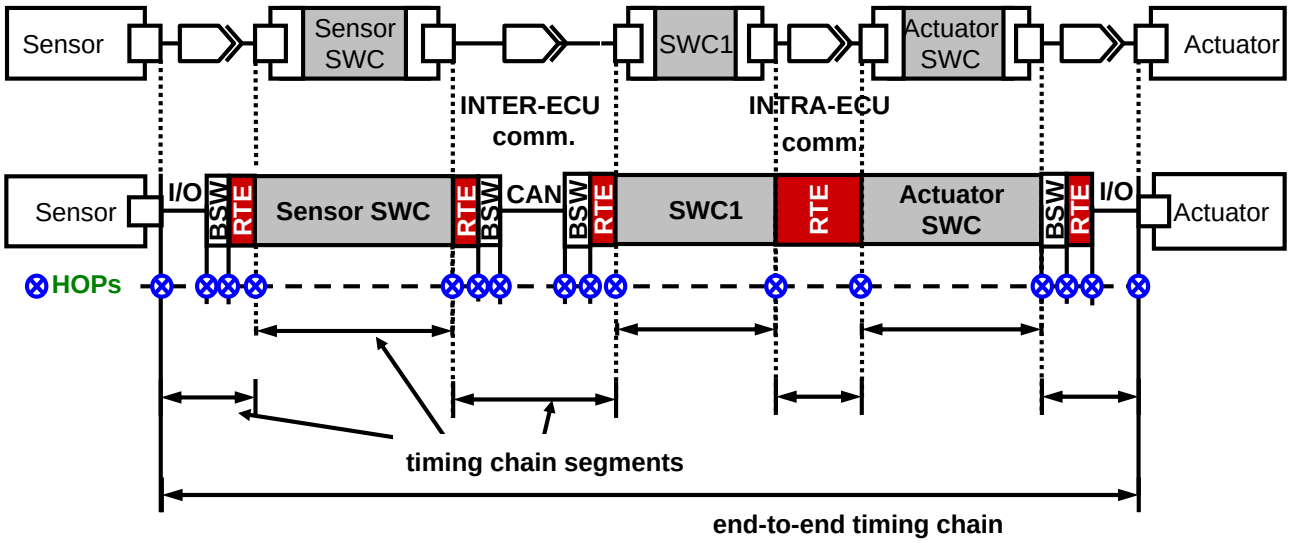
SW Component Execution

- **Standardized RTE eases compiling & linking together several SW components from different teams/vendors**



courtesy: K.Richter, Symtavision

Timing Chains and Hand-Over Points



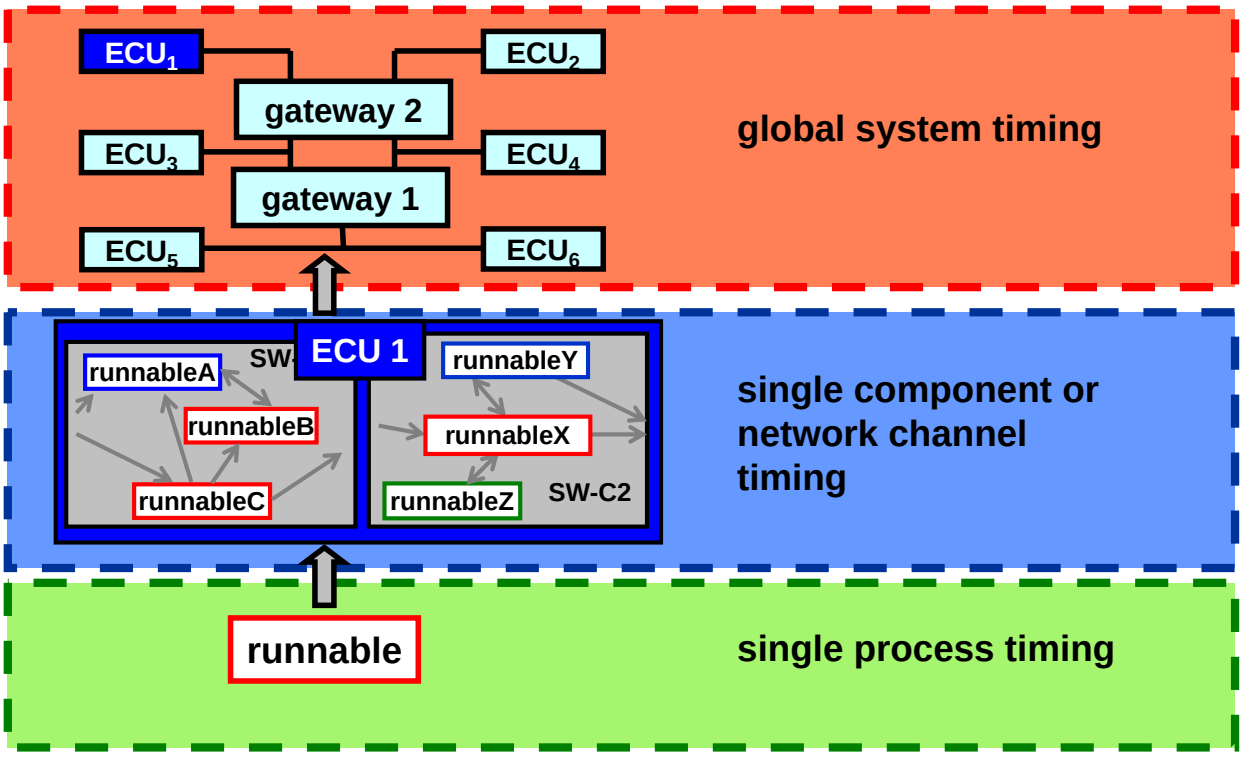
- **Hidden timing chains and non-functional dependencies challenge predictability**
- **Timing verification required to support design process**

courtesy: K.Richter, Symtavision

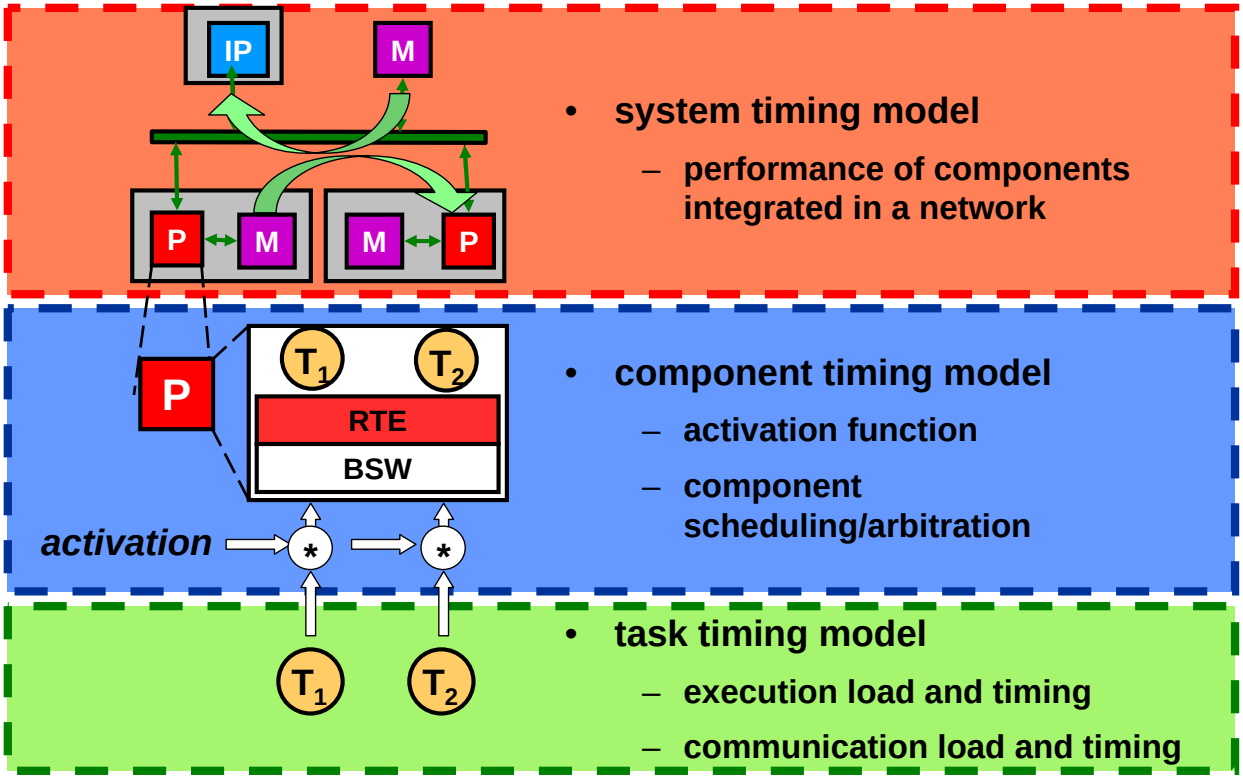
Overview

- performance challenges in automotive embedded system
- timing models and compositional analysis
- embedding in design process
- some recent enhancements
- conclusion

Timing Hierarchy - Automotive

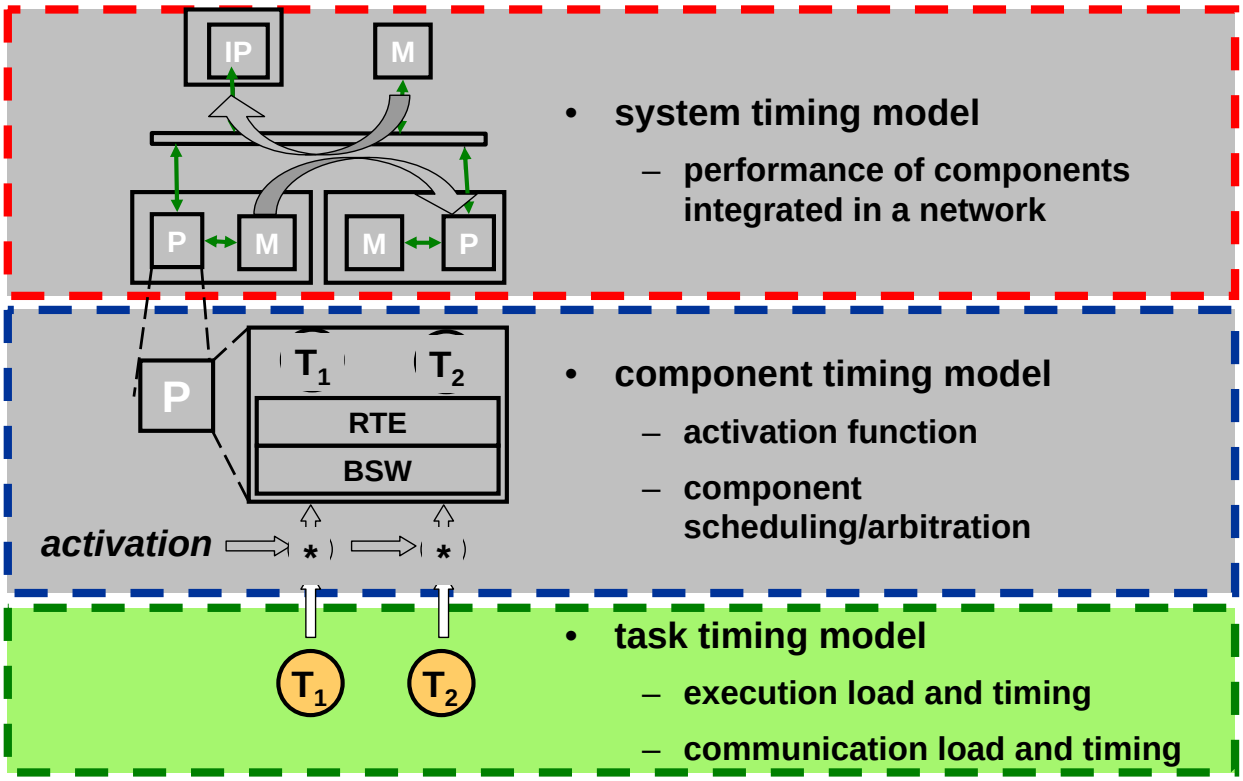


Timing Model Hierarchy



- system timing model
  - performance of components integrated in a network
- component timing model
  - activation function
  - component scheduling/arbitration
- task timing model
  - execution load and timing
  - communication load and timing

Timing Model Hierarchy - Task



- system timing model
  - performance of components integrated in a network
- component timing model
  - activation function
  - component scheduling/arbitration
- task timing model
  - execution load and timing
  - communication load and timing

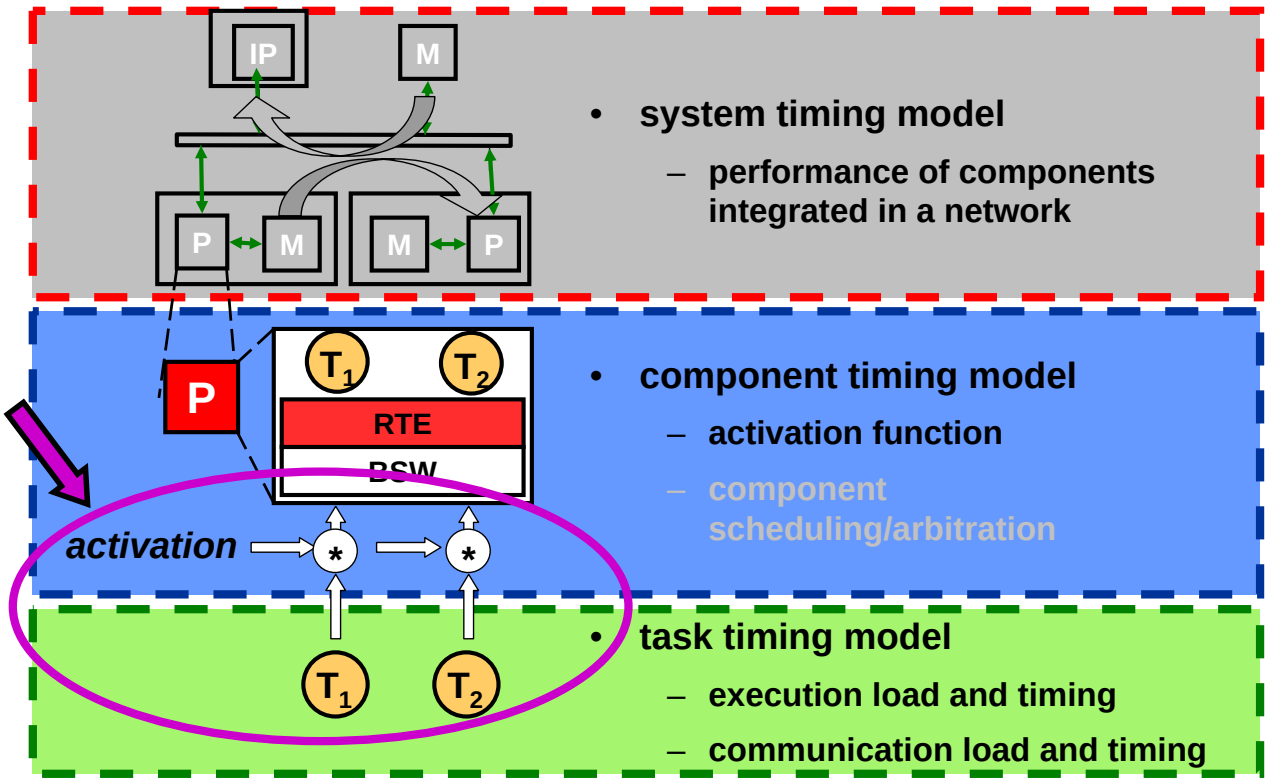
Formal Modeling Fundamentals – Task Execution

- *task core execution time* is the time needed to execute a given task when running alone on a processor
- task core execution time does *not* include
  - operating system overhead
  - the influence of other tasks
  - waiting and synchronization times for global resources
  - shared cache and memory access times (L1 cache often included)
- task core execution time is determined in different ways
  - estimated in early design phases
  - measured with a cycle accurate simulator (e.g. CoWare, Vast)
  - measured with instrumented code on a prototype (e.g. dspace)
  - formally analyzed using WCET analysis

Formal Modeling Fundamentals – Communication

- *core communication time* is the transmission time for a given message to be communicated over a link when no other communication is active
- core communication time does *not* include
  - arbitration (scheduling)
  - buffering
  - gateway, multi-hop or MIN timing overhead
- core communication time is determined in different ways
  - simulation or prototyping
  - communication analysis based on formal model of communication protocol
    - individually adapted to communication protocol
    - typically simpler than WCET analysis

Timing Model Hierarchy - Activation



Formal Modeling Fundamentals – Activation

- total task load, also called utilization of task i, U_i, depends on activation function

  total task load = load/task execution * task activation requency

  = task core execution time * task activation frequency

  – example: periodic task i with core execution time C_i and period T_i

  $U_i = C_i/T_i$

- what defines the task activation function ?
  - application model (Simulink, SPW, LabView, …)
  - environment model (reactive systems)
  - service contracts (max no of requests per time, …)

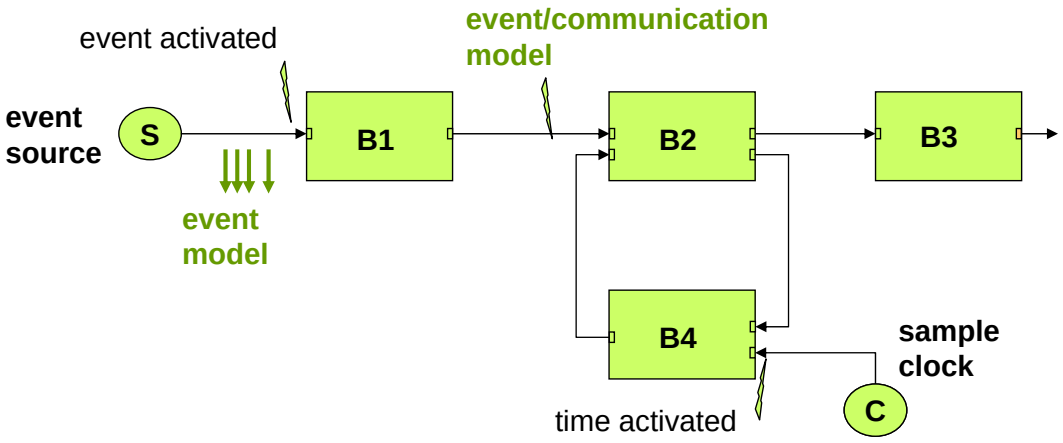
  → typically application rather than platform dependent

  → platform can „modulate“ activation timing to avoid malfunction (e.g. traffic shaping, back pressure)

- two classes of activation – time activation, event activation

Activation Functions

- two classes of activation
  - time activation – tasks are periodically activated by clock
    - example: periodic sample in signal processing / control eng.
  - event activation – tasks are activated when event arrives
    - example: automata, flow graph

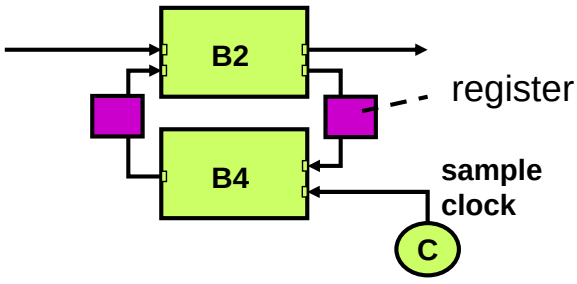


activation functions - example

Events and Activation Dependency

- event activation requires event buffering
  - event queues
  - no event lost (under normal operation)

- time activation does not always require buffering
  - buffering requirements only depends on the application semantics
    - may allow to drop data
    - dropping data may be required to obtain latest sample

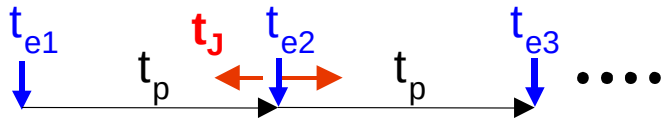


Modeling Events as Streams

- in system level formal performance models, events are modeled as *streams* rather than as sequences of individual events
- examples
  - a clock is given by its period rather than as a sequence of clock ticks
    → clock can be modeled as an event stream
  - a sampled sensor signal is modeled by the sample period and the sample jitter
- the event streams are defined as functions or as parameter tuples

Popular Event Stream Models – PJD

- standard event model used in classical schedulability analysis
  - event sequences are modeled by three parameters, period *p*, jitter *j*, and minimum time interval *d* between 2 events
  - important models that can easily be described
    - strictly periodic events (typically clock released)
    - periodic events with jitter
    - sporadic events
    - sporadically periodic events

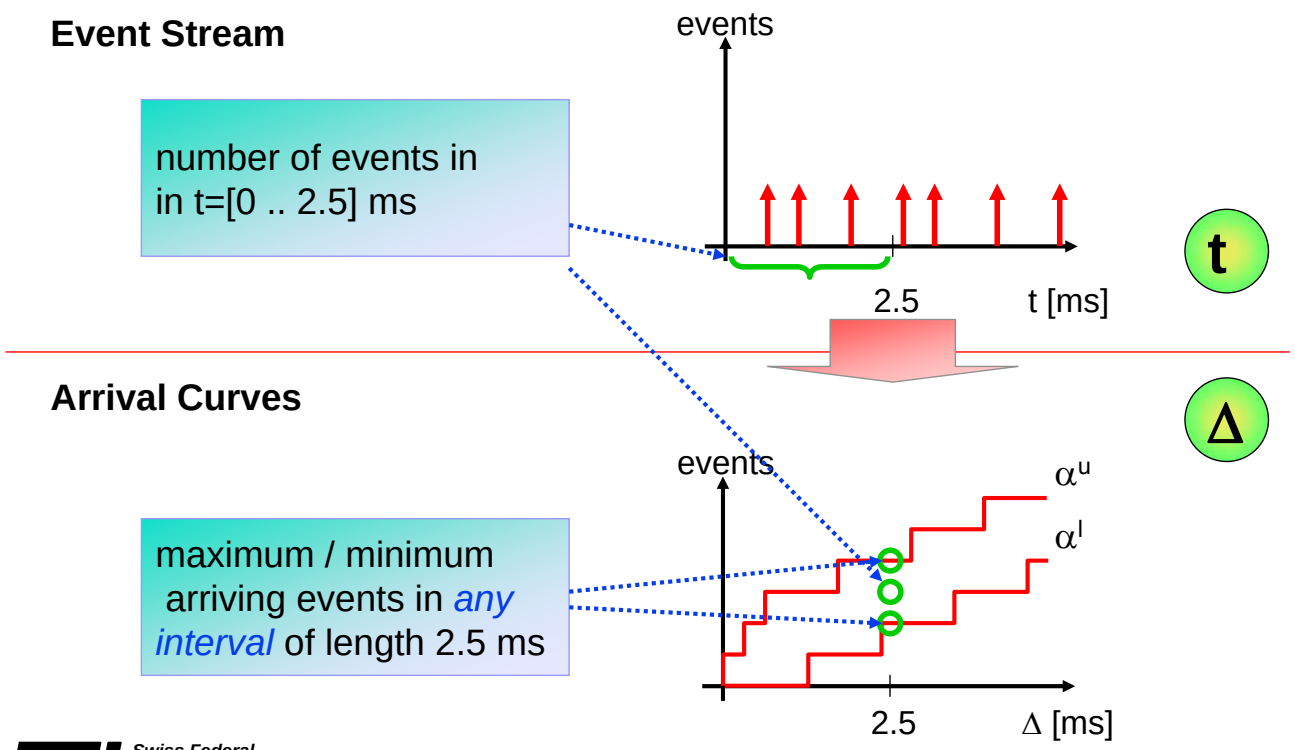


  - covers a large class of applications
  - conservatively approximates more complex functions

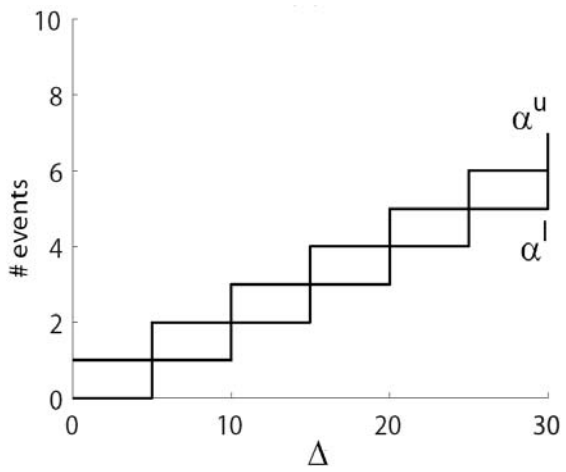
Popular Event Stream Models - Arrival Curves

- arrival curves of the network calculus
  - captures the no. of event in a time interval Δt
  - $\alpha^l(\Delta t)$ is lower bound
  - $\alpha^u(\Delta t)$ is upper bound
- can be used to describe the standard event models
- reaches infinite values for $\Delta t \rightarrow \infty$
  - must be approximated or extended by periodic function for $\Delta t \rightarrow \infty$
- is approximated when event sequences become very complex, e.g. as a result of operations on event sequences

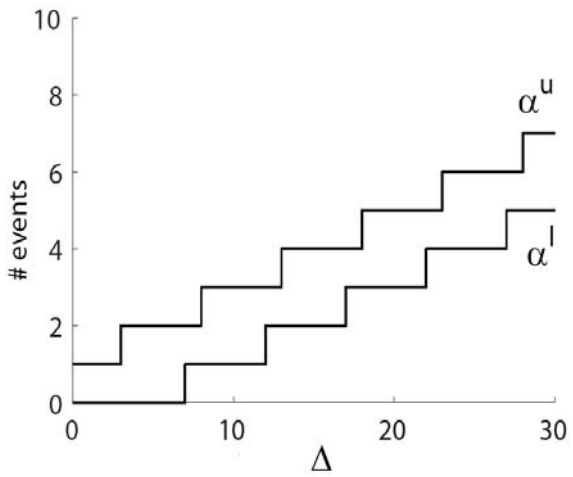
Arrival Curves - Example



Example 1: Periodic with Jitter



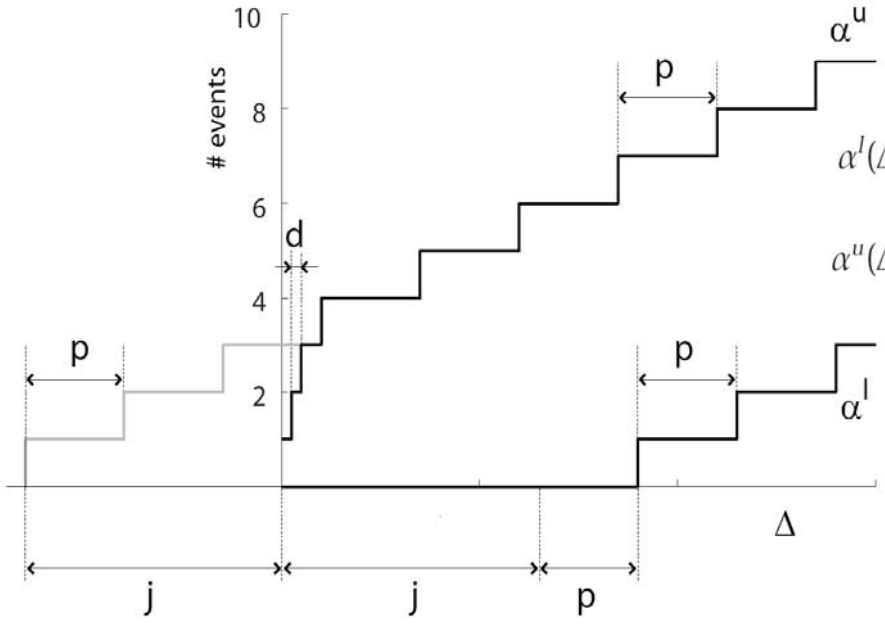
periodic



periodic with jitter

Example 2: Periodic with Jitter and Minimum Distance d

- ***Arrival curves***:

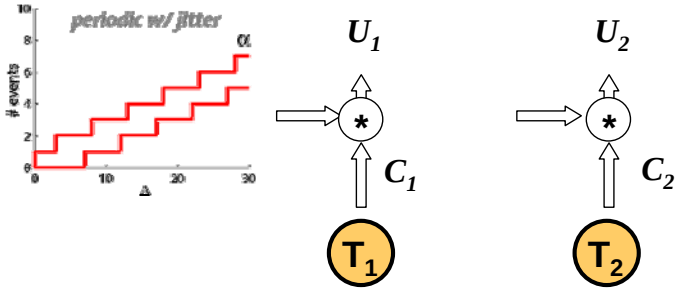


$$\alpha^l(\Delta) = \left\lfloor \frac{\Delta - j}{p} \right\rfloor$$

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

Total Load of a Task

- with activation model and core execution time or (core communication time), we can now derive the total load of a task

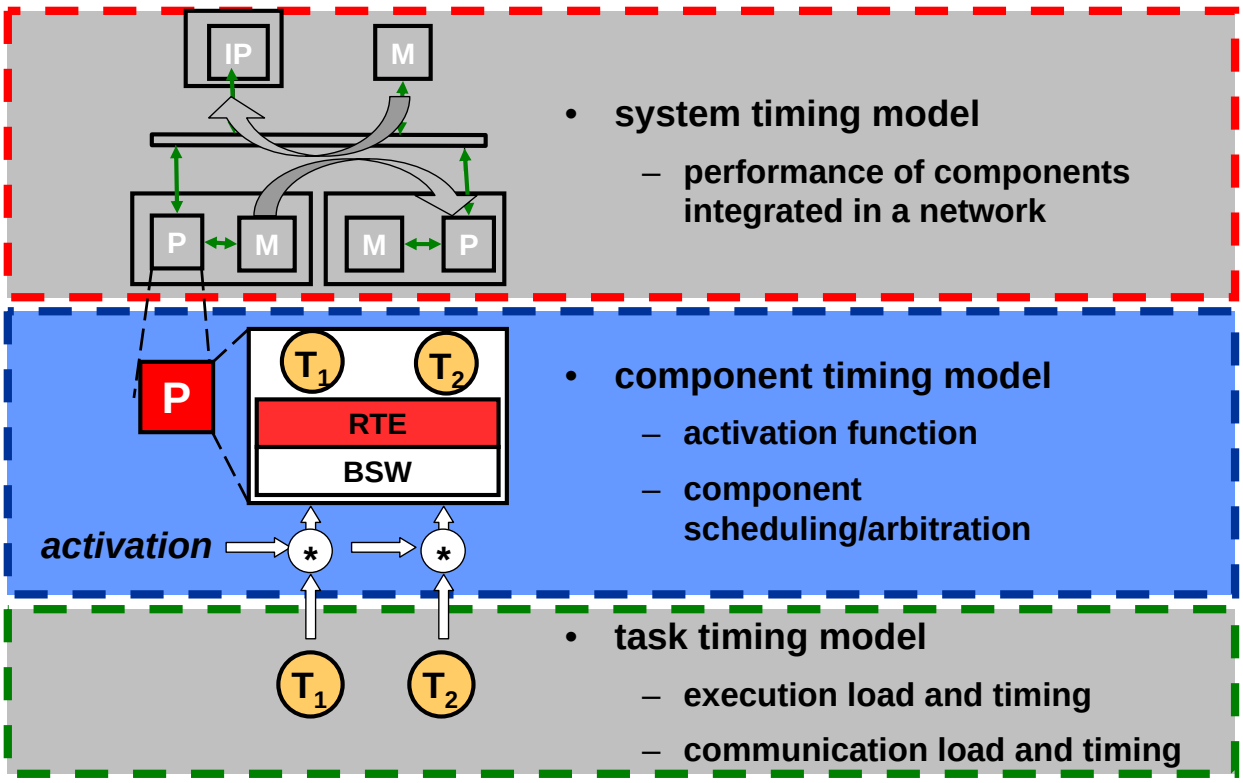


- the resource is not fully available to one task or communication, but is shared with others

Overview

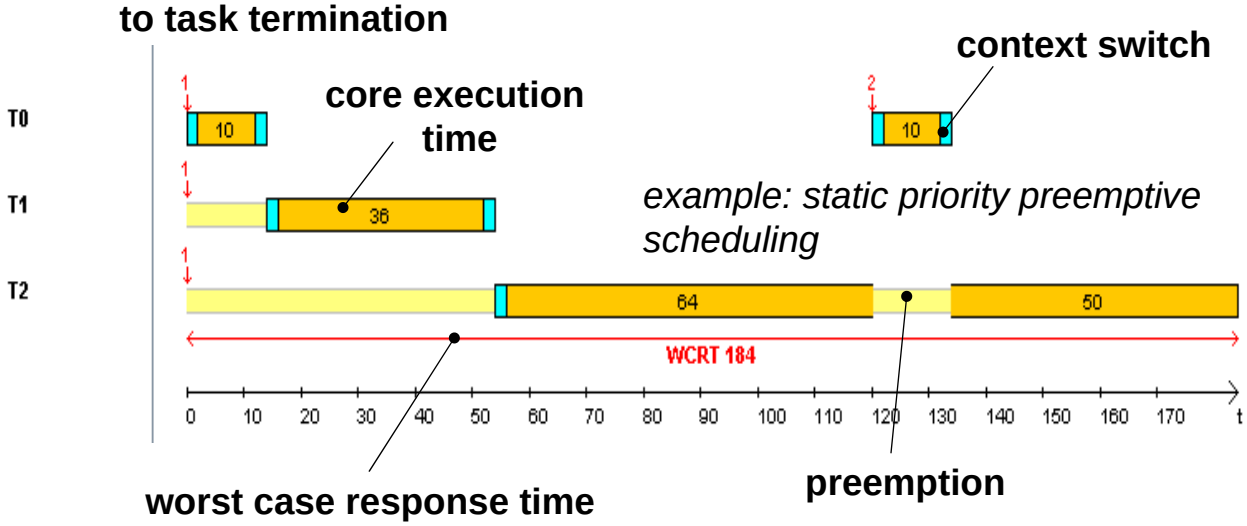
- applications for formal performance analysis methods
- formal performance modeling and analysis principles
- modeling activation and event streams
- component analysis
- system analysis
- enhancements to the basic analysis
- summary and comparison
- conclusion

Timing Model Hierarchy - Component Timing



Timing Effects of Scheduling/Arbitration

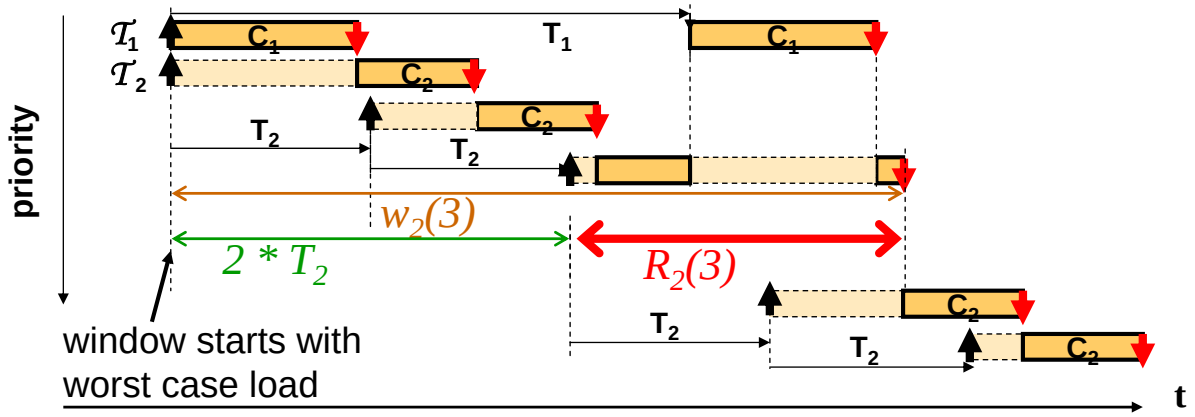
- tasks execute longer than their core execution time
  - time assigned to other tasks
  - operating system overhead
  - context switch, blocking, …
- response time of a task is maximum from time of activation to task termination



Scheduling Analysis

- **different analysis algorithms**
  - **generalization of busy window algorithm (Lehoczky, Tindell) to fit general event model (Richter, Jersak, Henia, Racu, Ernst, Schliecker, et al.)**
    - **Tool SymTA/S**
  - **extension of Network Calculus to Real-time Calculus (Chakraborty, Wandeler, Künzli, Thiele, et al.)**
    - **Tool MPA**

Analysis uses "Busy Window" approach

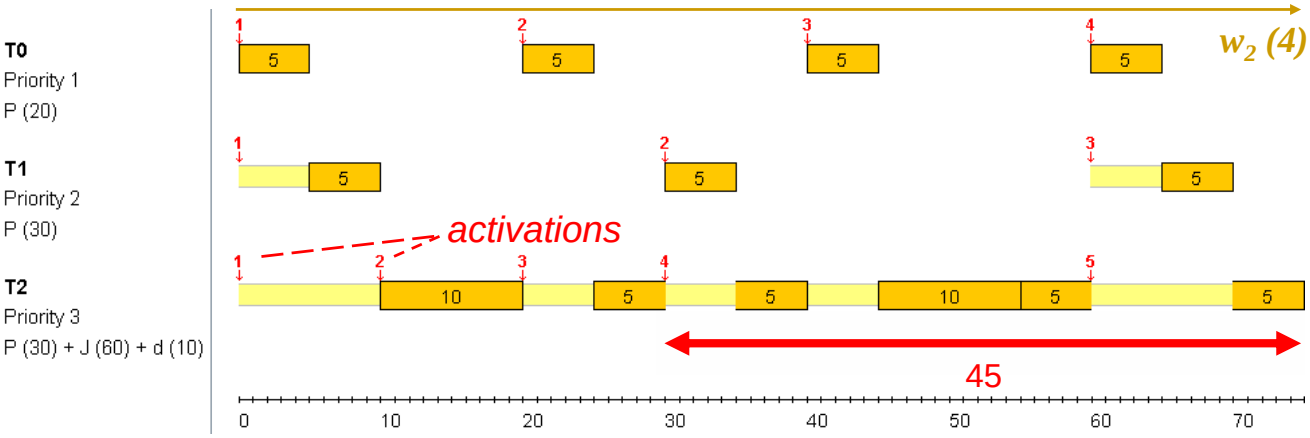


$$w_i(q) = q\,C_i + \sum_{j \in \mathrm{hp}(i)} C_j \left\lceil \frac{w_i(q)}{T_j} \right\rceil$$

increase w_i until fix point found where equations hold!

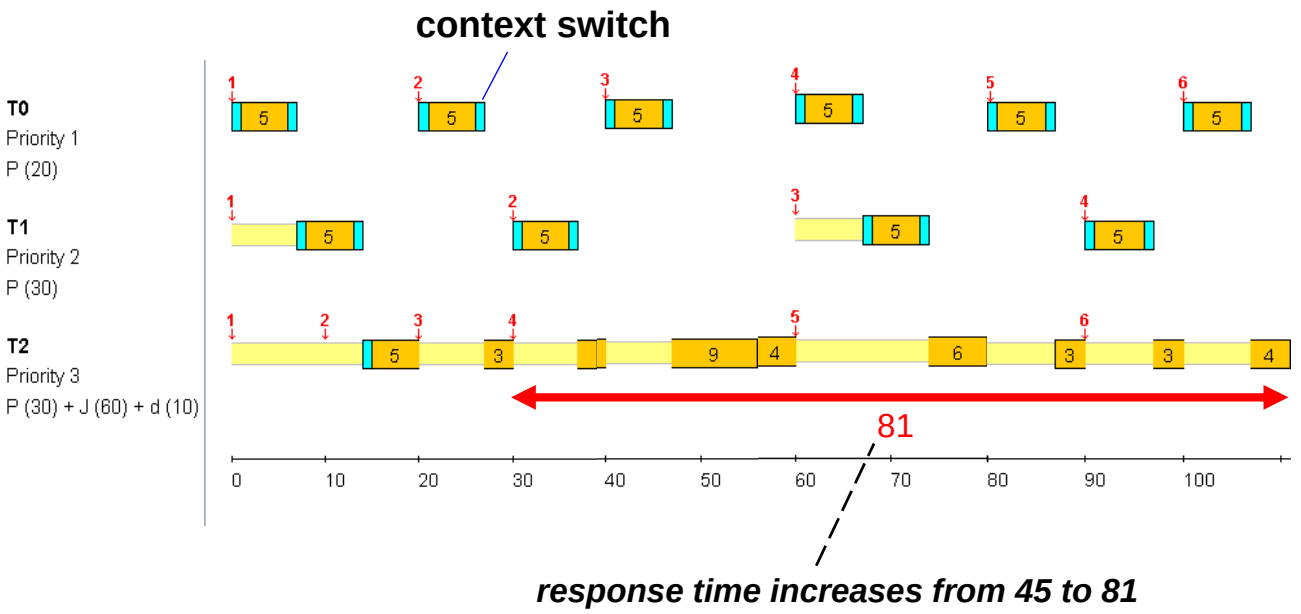
$$R_i(q) = w_i(q) - (q-1)\,T_i$$

Busy Window Analysis



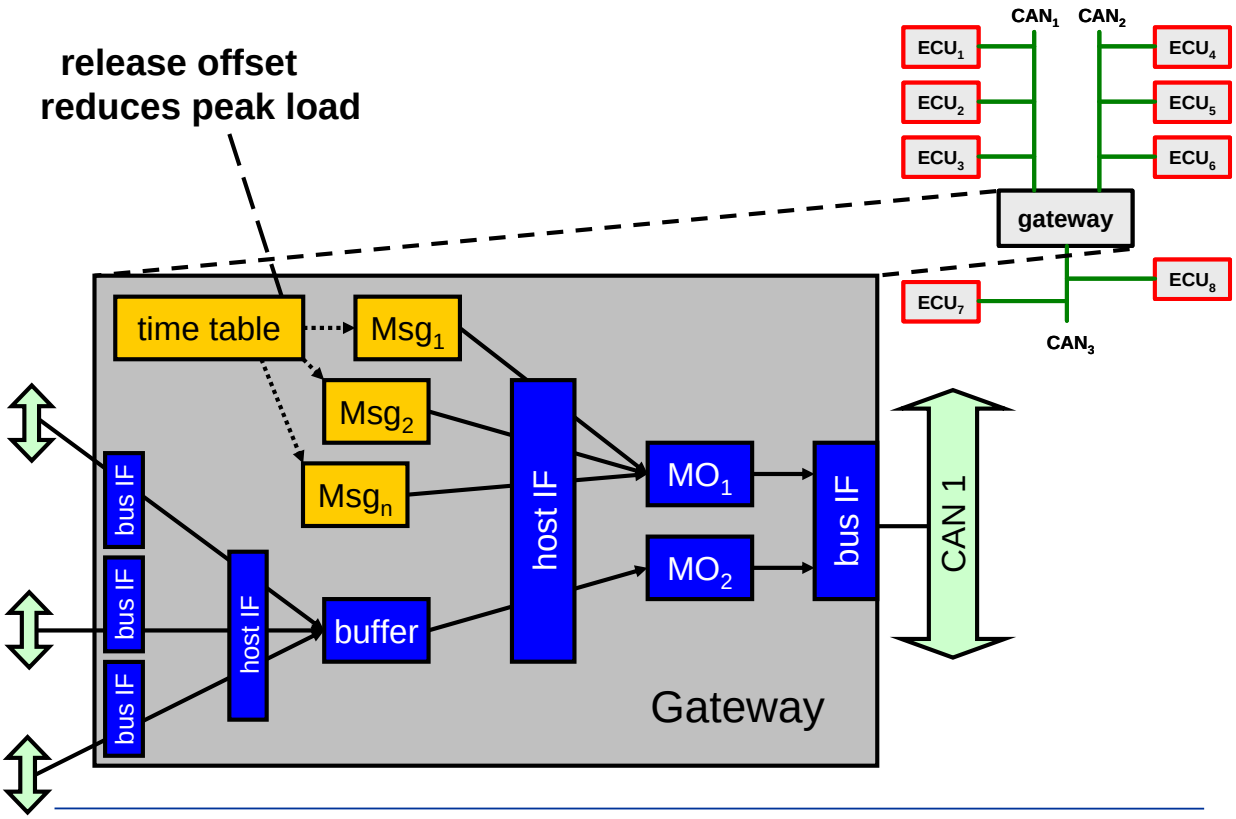
- very versatile approach
- has been extended to analyze even difficult scheduling strategies
  - round-robin, non preemptive, collaborative processes (e.g. OSEK), …
- can handle parameter dependent worst case (e.g. release offsets – time table)
- can handle stream queues and register communication
- window size increases with load (limited by deadline)
- this window „unrolling“ process can be considered as symbolic simulation

Importance of Context Switch Consideration

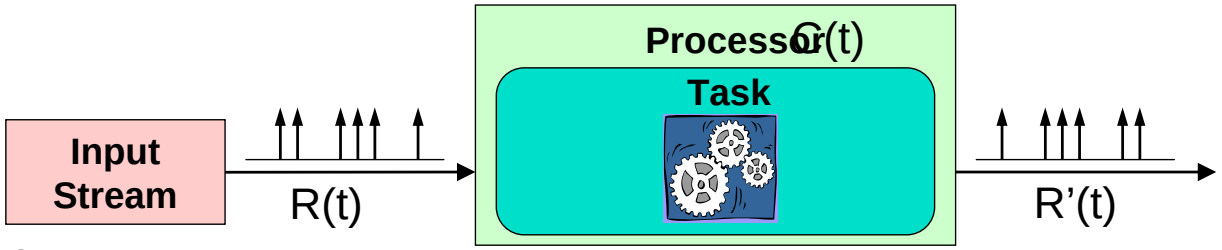


- context switch increases load → non load preserving

Time Table for Release Offset

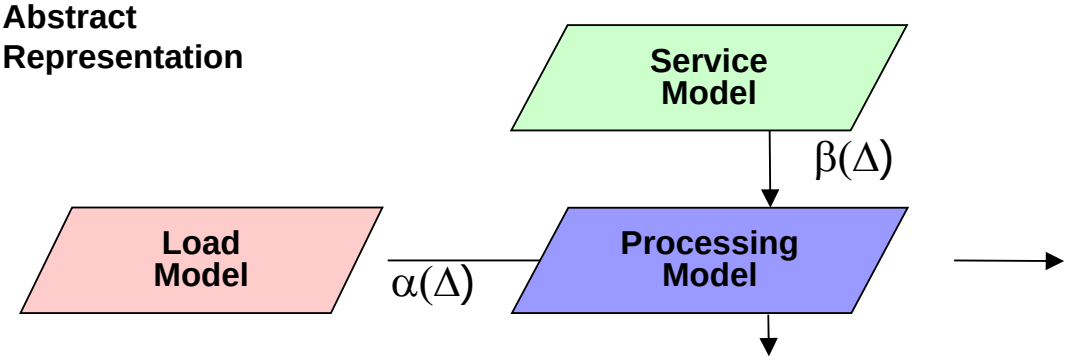


Real-time Calculus

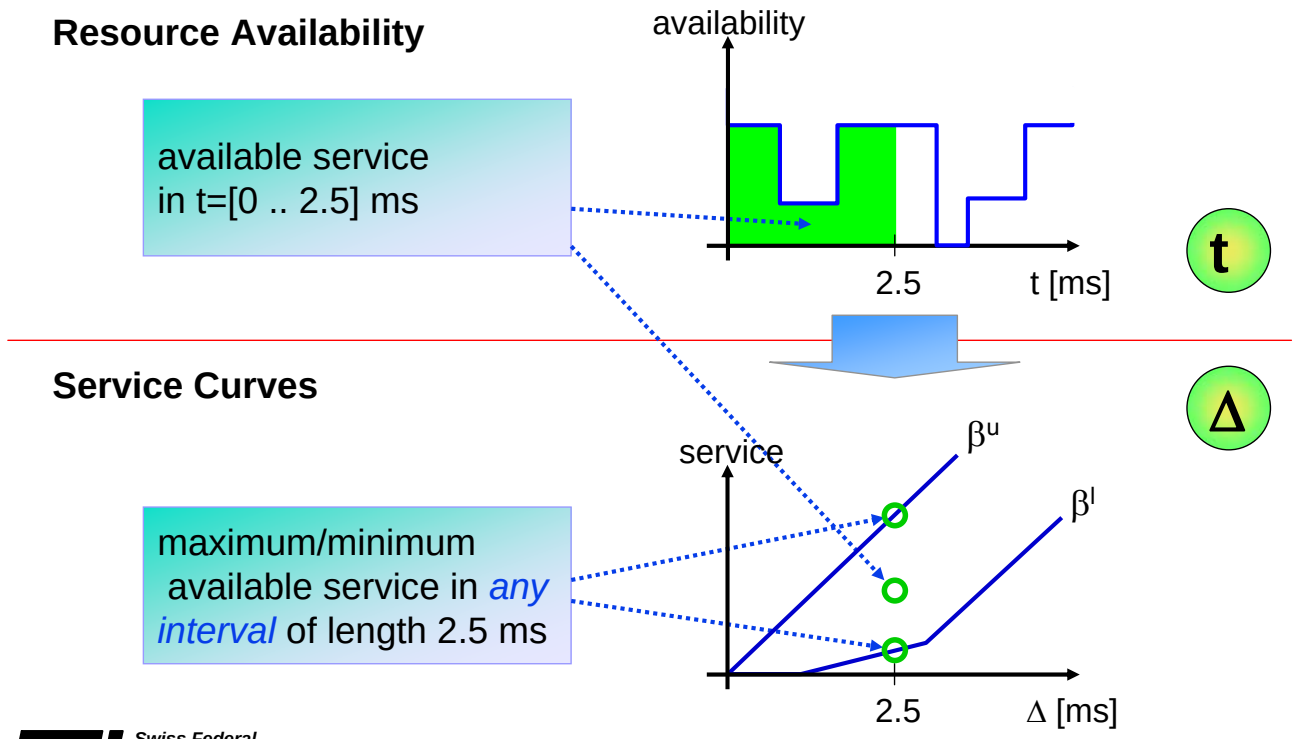


Concrete Instance

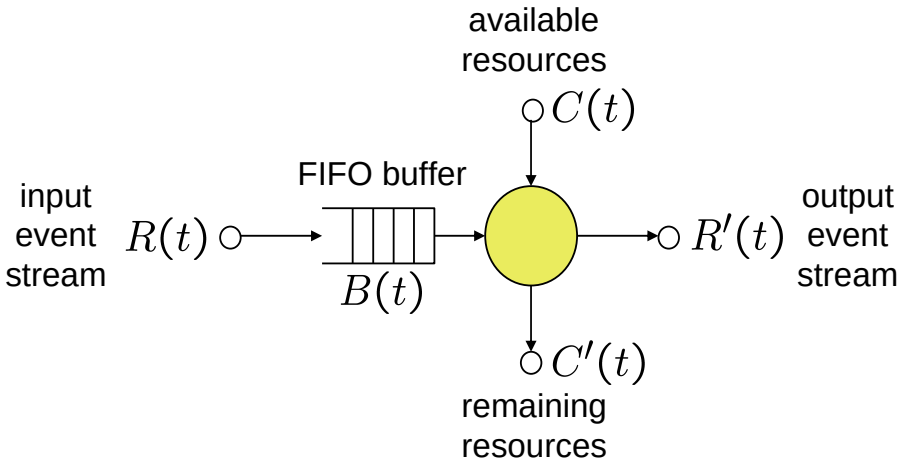
Abstract Representation



Service Model (Resources)

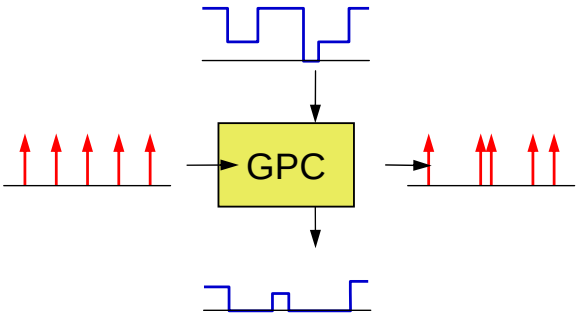


Greedy Processing Component (GPC)



- ***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.

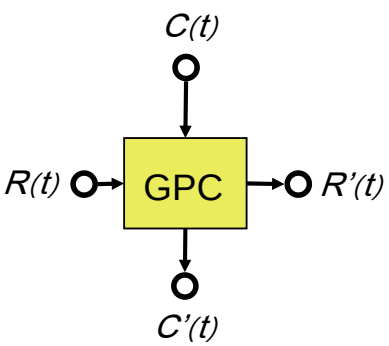
Greedy Processing Component (GPC)

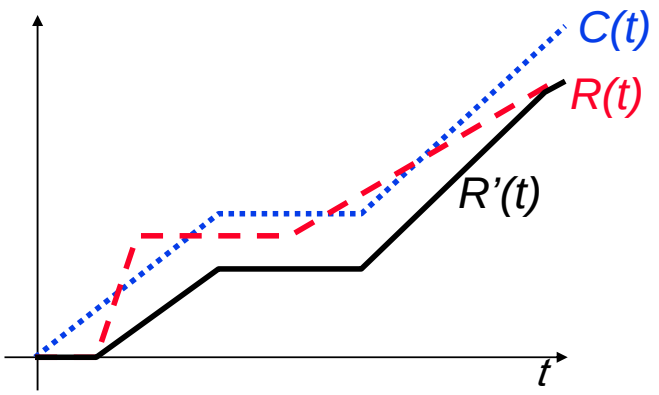
If the resource and event streams describe available and requested units of processing or communication, then

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

Conservation Laws

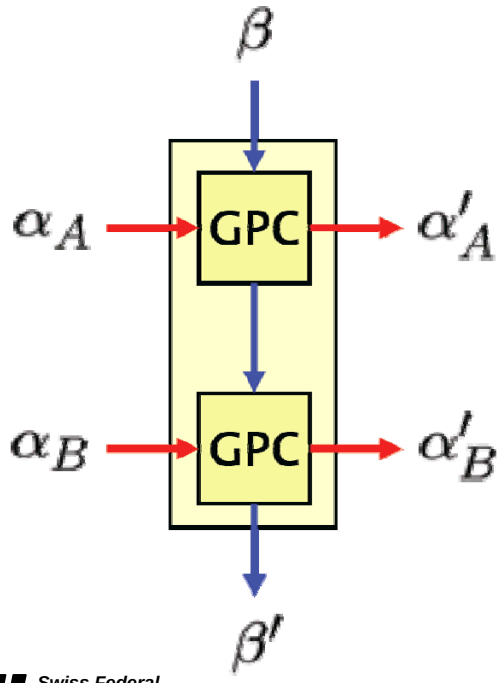
$$R'(t) = \inf_{0 \le u \le t} \{R(u) + C(t) - C(u)\}$$



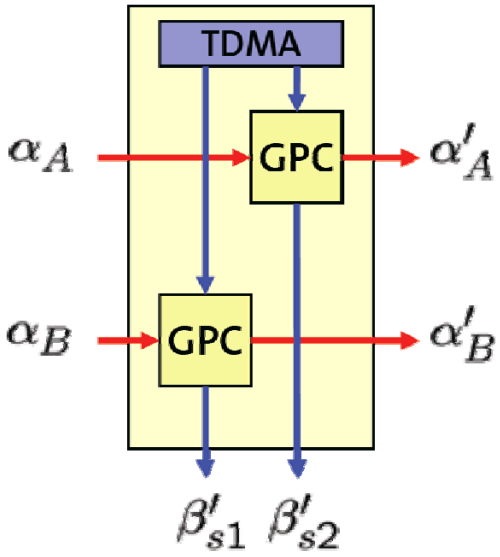


MPA-RTC – Scheduling - Examples

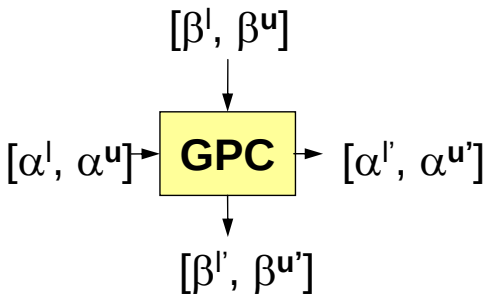
Fixed Priority Preemptive Scheduling

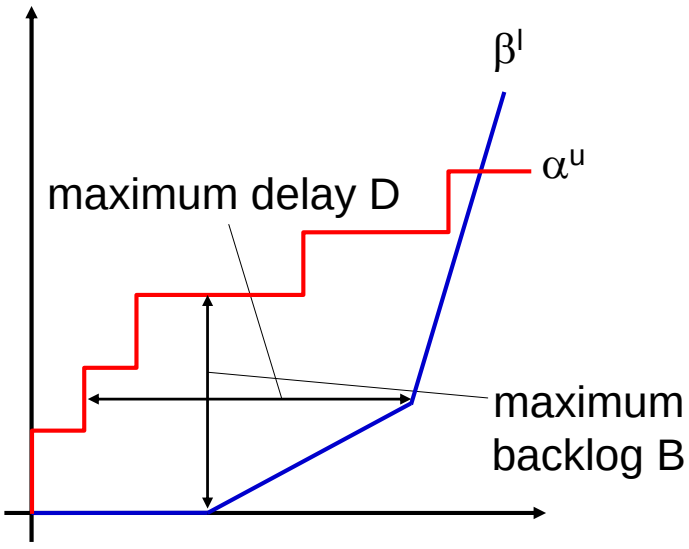


Time Division Multiple Access (TDMA)



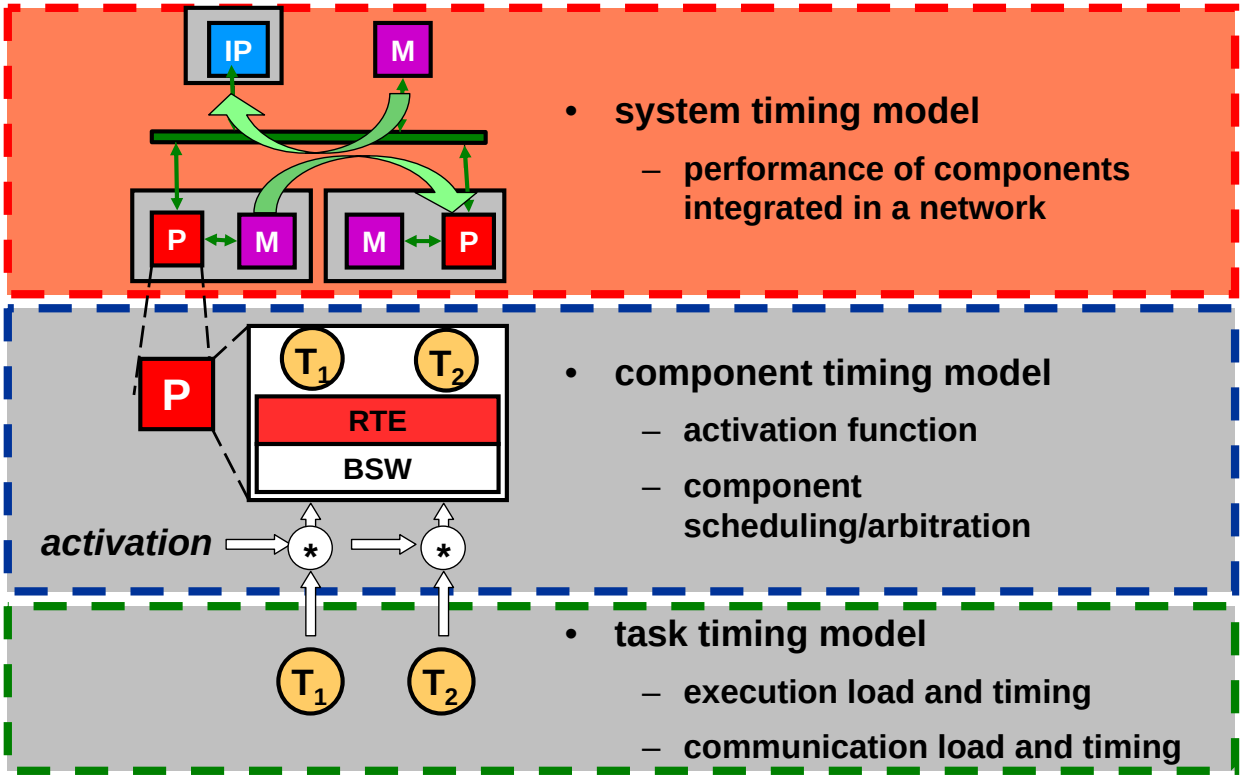
Delay and Backlog





→ *same solution for Busy Window analysis*

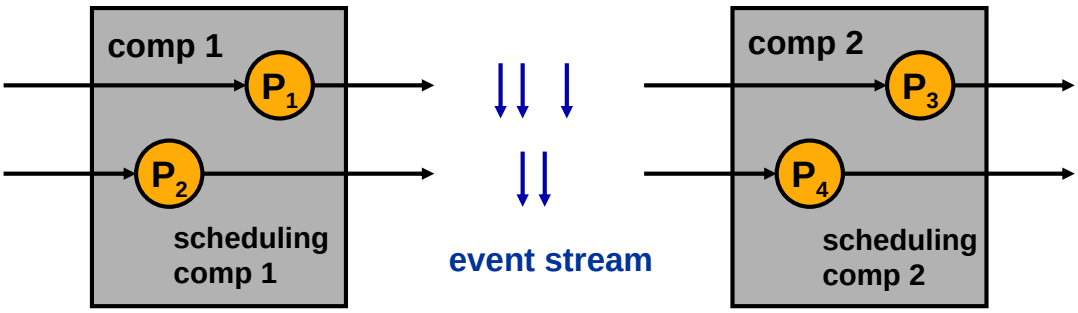
Timing Model Hierarchy – System Timing Model



- system timing model
  - performance of components integrated in a network
- component timing model
  - activation function
  - component scheduling/arbitration
- task timing model
  - execution load and timing
  - communication load and timing

System Analysis using Composition

- independently scheduled subsystems are coupled by data flow

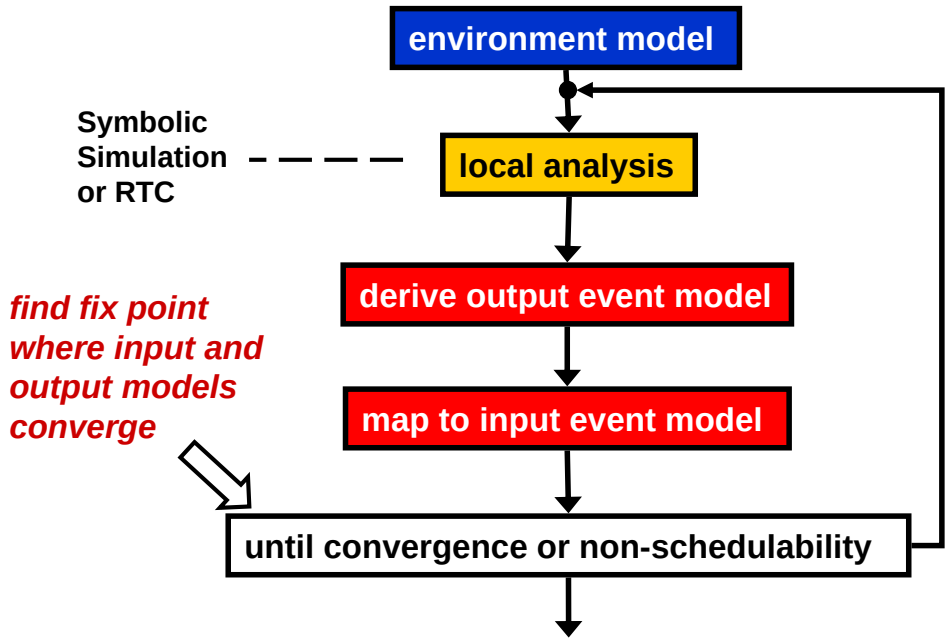


⇒ subsystems coupled by streams of data

⇒ interpreted as activating events

⇒ coupling corresponds to event propagation

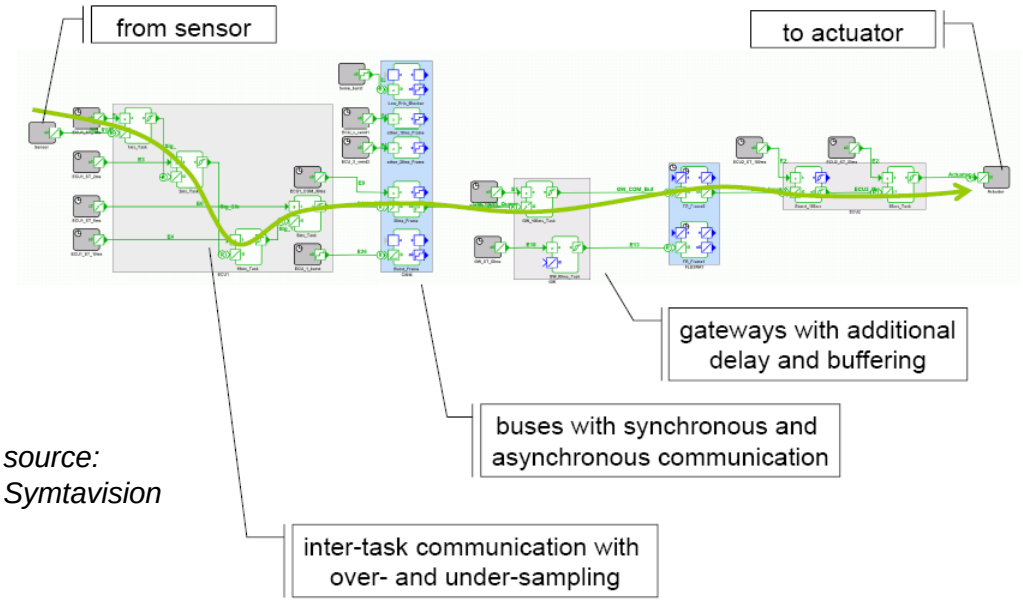
Compositional Analysis Principle



System-level Analysis Results

- end-to-end latencies
- buffer sizes
- system load
- ….

example: complex end-to-end latency analysis w. SymTA/S

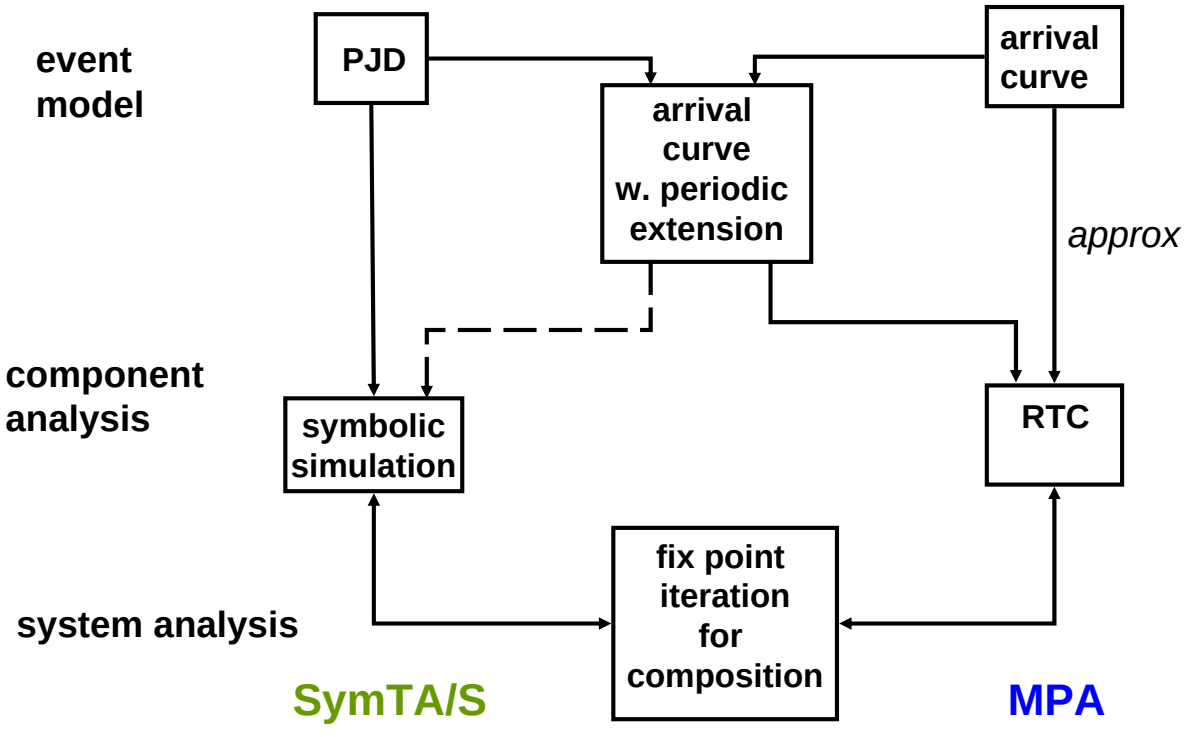


source: Symtavision

Compositional Analysis Properties

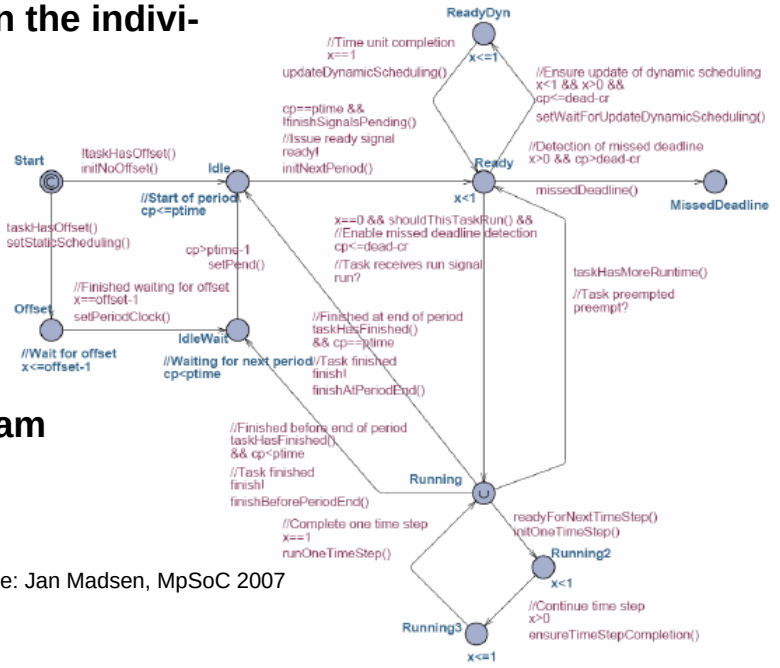
- compatible event stream models allow to couple any number of blocks for local analysis
  - → scalable
- fixpoint iteration automatically adapts to platform topology
  - → easy integration and extension
  - → RTC and SymTA/S analysis blocks have been shown to easily work together [KHT07]
- very short analysis time (few seconds) opens new opportunities in design space and robustness optimization

The Compositional Analysis „Landscape“



Further Performance Models

- timed automata have been used to explicitely model the task scheduling algorithm and OS interactions and then apply model checking to identify deadline violations
- can be more accurate in the individual component model but is computationally far more expensive
- work e.g. Madsen or Johnson
- can potentially be linked via common event stream models

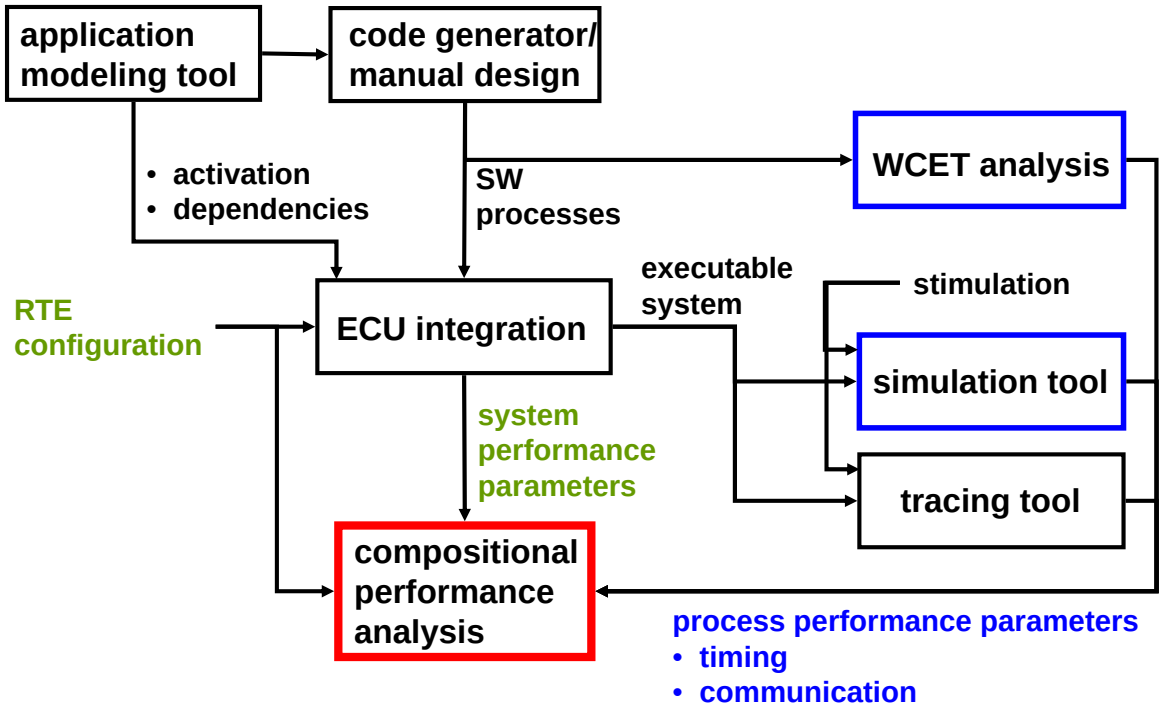


source: Jan Madsen, MpSoC 2007

Overview

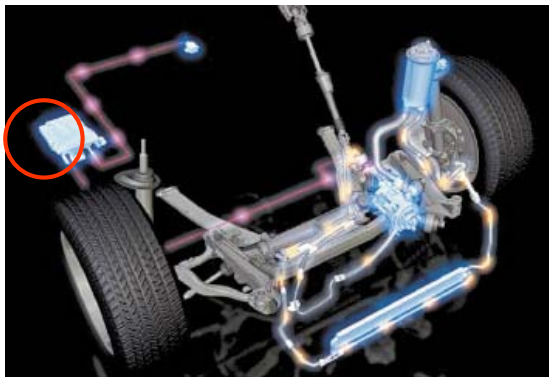
- performance challenges in automotive embedded system
- timing models and compositional analysis
- embedding in design process
- some recent enhancements
- conclusion

Performance Verification Flow- ECU



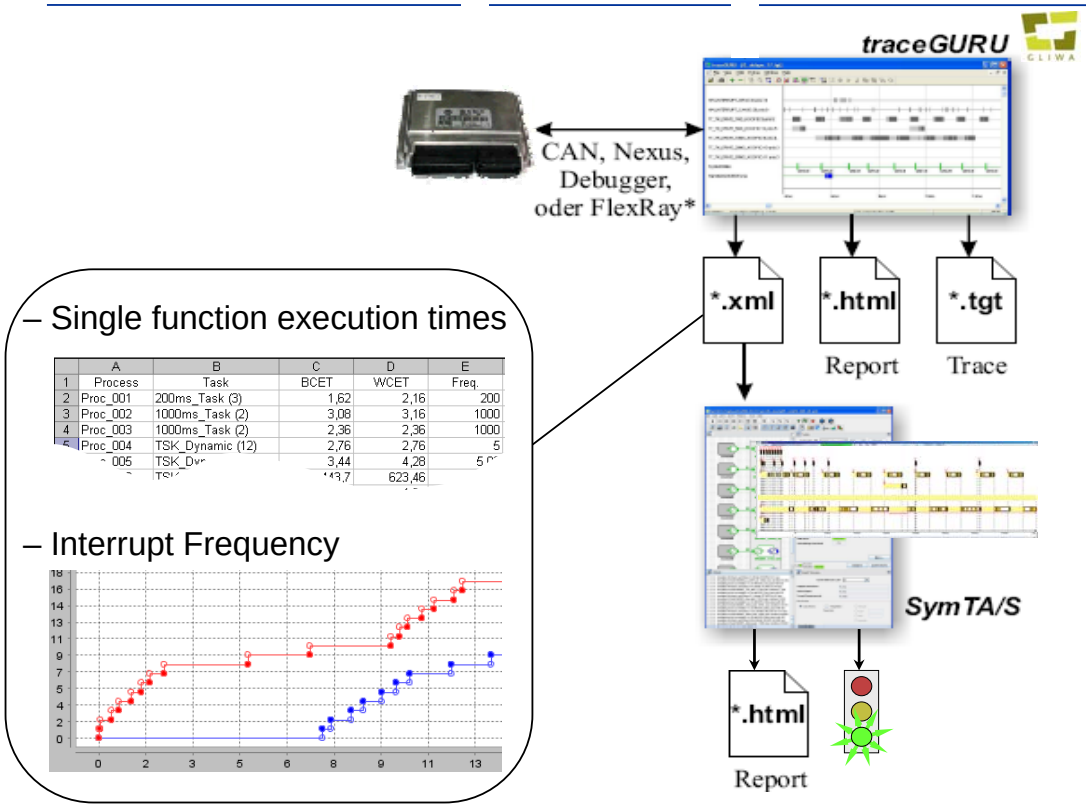
Example: Safety-Critical ECU

- Chassis domain: Active Front Steering (SIL 3)
  - Verifying Performance and Timing for all critical cases
  - Optimizing ECU performance and cost (use of cheaper CPU)
  - Safeguarding against liability claims
- late design phase
- formal analysis used as complement to prototyping to reach higher confidence



Source: BMW

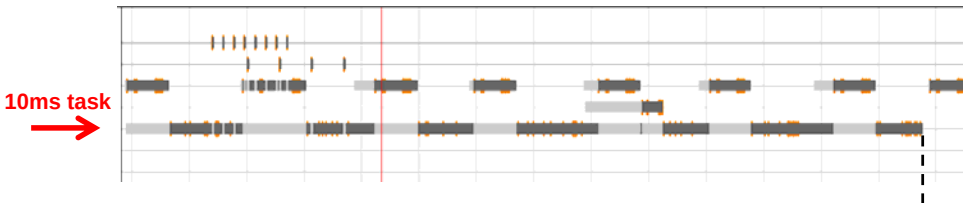
Integration: Tracing + SymTA/S



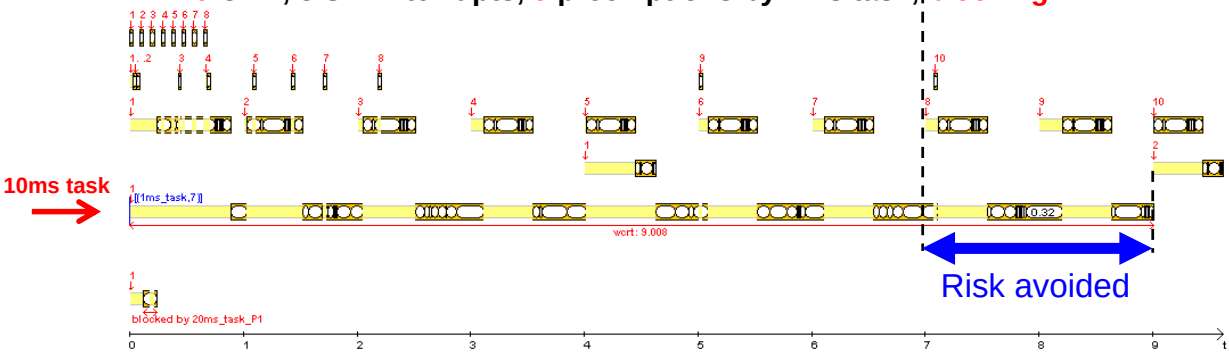
courtesy: K.Richter, Symtavision

Tracing plus SymTA/S Analysis

- **Measured 10ms task: Response time 6,9ms**
  - **4 CAN, 8 SPI interrupts, 7 preemptions by 1ms task**

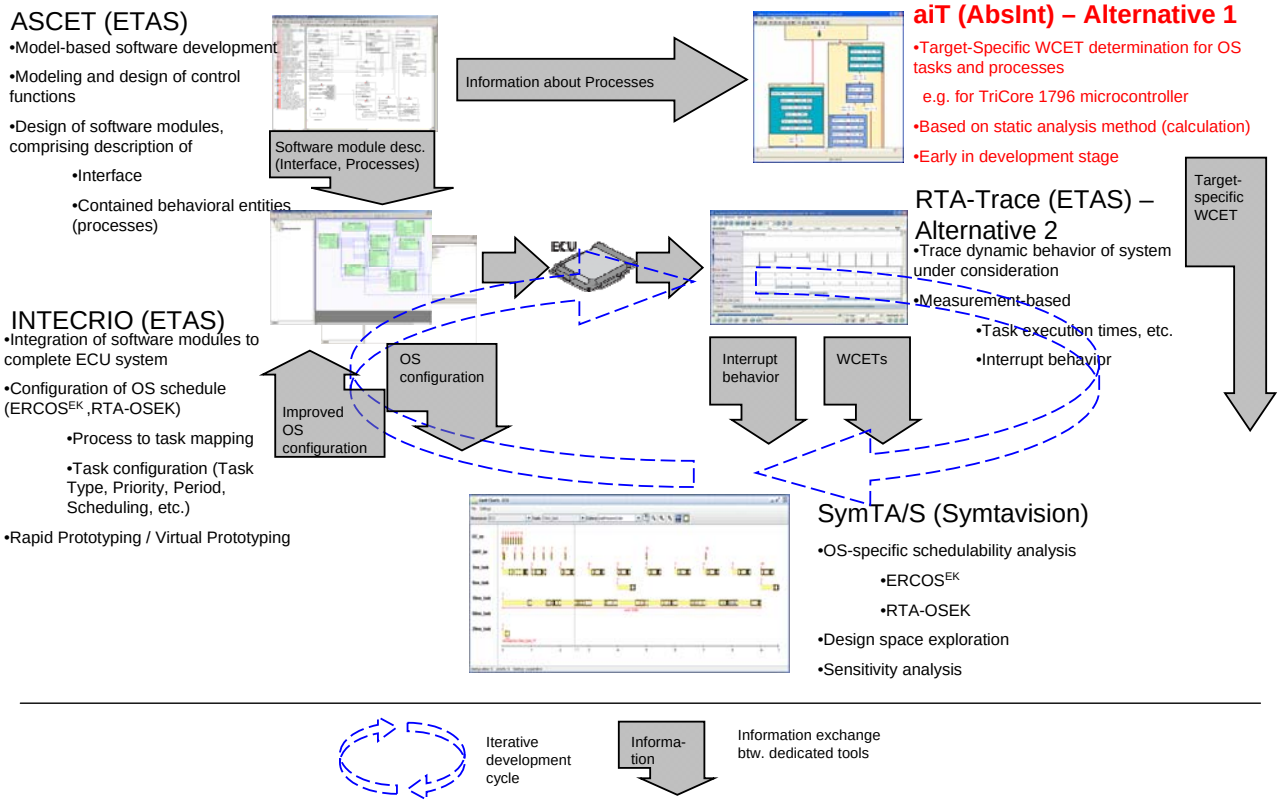


- **SymTA/S Analysis of 10ms task: Worst-case response time 9ms**
  - **10 CAN, 8 SPI interrupts, 9 preemptions by 1ms task, blocking**



courtesy: K.Richter, Symtavision

WCET Tool Integration



Overview

- performance challenges in automotive embedded system
- timing models and compositional analysis
- embedding in design process
- some recent enhancements
- conclusion

Enhancements to the Basic Analysis

- shared memory modeling on multi-core systems
  - „secondary“ traffic from shared memory and coprocessor access in conflict with other traffic
  - many schedulability hazards
  - requires model and analysis extension
  - active research area (see literature and tutorial link)
- robustness analysis
  - identification of „system reserves“ for potential changes
  - can be used for optimization (see literature and tutorial link)
- scenario analysis

Scenario Analysis

- Identify different sets of tasks or deviating core execution times of tasks for different application contexts → scenarios
- Example: engine acceleration/idle, ..
- Interesting is transition between scenarios
  - possibly leading to overloads, lost data, …
- Analysis: analyse scenarios individually + analyse transition

Accelerating on a hill

Holding the speed

Overview

- performance challenges in automotive embedded system
- timing models and compositional analysis
- embedding in design process
- some recent enhancements
- conclusion

Conclusion

- several performance analysis and optimization approaches have have brought system analysis far beyond the stage of toy examples
- the cost of a predictable design has been reduced by higher modeling and analysis precision
- WCET analysis is part of the analysis chain and currently often “substituted” by extensive simulation
- challenges for WCET analysis arise in multi-core systems, scenario analysis and system optimization

References Symta/S 1/4

[RRE03] K. Richter and R. Racu and R. Ernst. "Scheduling Analysis Integration for Heterogeneous Multiprocessor SoC." In *IEEE Real-Time Systems Symposium (RTSS)*, Cancun, Mexico, December 2003.

[HHJ05] R. Henia and A. Hamann and M. Jersak and R. Racu and K. Richter and R. Ernst. "System Level Performance Analysis - the SymTA/S Approach." In *IEE Proceedings Computers and Digital Techniques*, 2005.

[RER07] R. Racu and R. Ernst and K. Richter and M. Jersak. "A Virtual Platform for Architecture Integration and Optimization in Automotive Communication Networks." In *SAE World Congress*, Detroit, USA, April 2007.

[RHE07] R. Racu and A. Hamann and R. Ernst and K. Richter. "Automotive Software Integration." In *Proc. of the 44th Design Automation Conference*, San Diego, CA, USA, June 2007.

[HRE06] A. Hamann and R. Racu and R. Ernst. "Formal Methods for Automotive Platform Analysis and Optimization." In *Proc. Future Trends in Automotive Electronics and Tool Integration Workshop (DATE Conference)*, Munich, March 2006.

[HRE07] A. Hamann and R. Racu and R. Ernst. "Multi-Dimensional Robustness Optimization in Heterogeneous Distributed Embedded Systems." In *Proc. of the 13th IEEE Real-Time and Embedded Technology and Applications Symposium*, April, 2007.

[HJR06] A. Hamann, M. Jersak, K. Richter, R. Ernst. "A framework for modular analysis and exploration of heterogeneous embedded systems." In *Real-Time Systems Journal*, Volume 33, pp 101-137, July 2006.

References Symta/S 2/4

[RHE06] R. Racu and A. Hamann and R. Ernst. "A Formal Approach to Multi-Dimensional Sensitivity Analysis of Embedded Real-Time Systems." In *Proc. of the 18th Euromicro Conference on Real-Time Systems (ECRTS)*, Dresden, July 2006.

[RE06] R. Racu and R. Ernst. "Scheduling Anomaly Detection and Optimization for Distributed Systems with Preemptive Task-Sets." In *12th IEEE Real-Time and Embedded Technology and Applications Symposium*, San Jose, USA, April 2006.

[HE07] R. Henia, R. Ernst. "Scenario Aware Analysis for Complex Event Models and Distributed Systems." In *Proc. Real-Time Systems Symposium*, December 2007.

[SHR07] S. Schliecker and A. Hamann and R. Racu and R. Ernst. "Formal Methods for System Level Performance Analysis and Optimization." In *Proc. of the Design Verification Conference (DVCON)*, San Jose, CA, February 2008.

[SIE06] S. Schliecker and M. Ivers and R. Ernst. "Integrated Analysis of Communicating Tasks in MPSoCs." In *Proc. 3rd International Conference on Hardware Software Codesign and System Synthesis (CODES)*, Seoul, Korea, October 2006.

[SHE06] S. Stein and A. Hamann and R. Ernst. "Real-time Property Verification in Organic Computing Systems." In *Proc. of the 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation*, November 2006.

[RE08] J. Rox and R. Ernst. "Modeling Event Stream Hierarchies with Hierarchical Event Models." In *Proc. Design, Automation and Test in Europe (DATE),* Munich, 2008.

[KHT07] S. Künzli, A. Hamann, L. Thiele, R. Ernst. "Combined Approach to System Level Performance Analysis of Embedded Systems". Codes+ISSS 2007, Salzburg, 2007.

References MPA 3/4

- E. Wandeler. Modular Performance Analysis and Interface-Based Design for Embedded Real-Time Systems. PhD Thesis ETH Zurich, 2006

[WT07a], E. Wandeler and Lothar Thiele, Workload correlations in multi-processor hard real-time systems. Journal of Computer and System Sciences, Mar. 2007

[WTVL06] E. Wandeler and Lothar Thiele and Marcel Verhoef and Paul Lieverse, System Architecture Evaluation Using Modular Performance Analysis - A Case Study, Software Tools for Technology Transfer (STTT), Oct. 2006}

[TWS06] Lothar Thiele and Ernesto Wandeler and Nikolay Stoimenov, Real-time interfaces for composing real-time systems, International Conference On Embedded Software EMSOFT 06, 2006.

[TWC05] Lothar Thiele and Ernesto Wandeler and Samarjit Chakraborty, A Stream-Oriented Component Model for Performance Analysis of Multiprocessor DSPs, IEEE Signal Processing Magazine, special Issue on Hardware/Software Co-design for DSP, May 2006

[WMT05] Ernesto Wandeler and Alexandre Maxiaguine and Lothar Thiele,Quantitative characterization of Event Streams in Analysis of Hard Real-Time Application}, Real-time Systems. Mar, 2005

[CKT03] Samarjit Chakraborty and Simon K\"unzli and Lothar Thiele. A General Framework for Analysing System Properties in Platform-Based Embedded System Designs},. DATE 2003.

[TCN99] Lothar Thiele and Samarjit Chakraborty and Martin Naedele, Real-time Calculus for Scheduling Hard Real-Time Systems, International Symposium on Circuits and Systems ISCAS 2000, Mar. 2000.

References other 4/4

- J. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In Proceedings Real-Time Systems Symposiom, pages 201–209, 1990.
- K. Tindell, A. Burns, and A. Wellings. An extendible approach for analysing fixed priority hard real-time systems. Journal of Real-Time Systems, 6(2):133–152, Mar 1994.
- F. Baccelli, G. Cohen, G. J. Olster, and J. P. Quadrat, Synchronization and Linearity --- An Algebra for Discrete Event Systems, Wiley, New York, 1992.
- 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.