Timing Analysis on Complex Real-Time Automotive Multicore Architectures

Mircea Negrean
Simon Schliecker
Rolf Ernst

Institute of Computer and Network Engineering

Technische Universität Braunschweig

Automotive challenges

- Requirements of automotive E/E platforms and architectures
  - Sufficient computing power and communication bandwidth
  - Avoid unnecessary over-dimensioning of computing resources
  - Low cost

➔ Find sweet spot between **maximum performance** and **minimum costs**

- Requirements of automotive OEMs (for certification)
  - Analytical proof:
    - That systems correctly work under maximum load
    - Sufficient resource availability at any time

→ Formal performance verification techniques

- During different system design phases
- For final product to increase SIL (safety integrity level) compliance

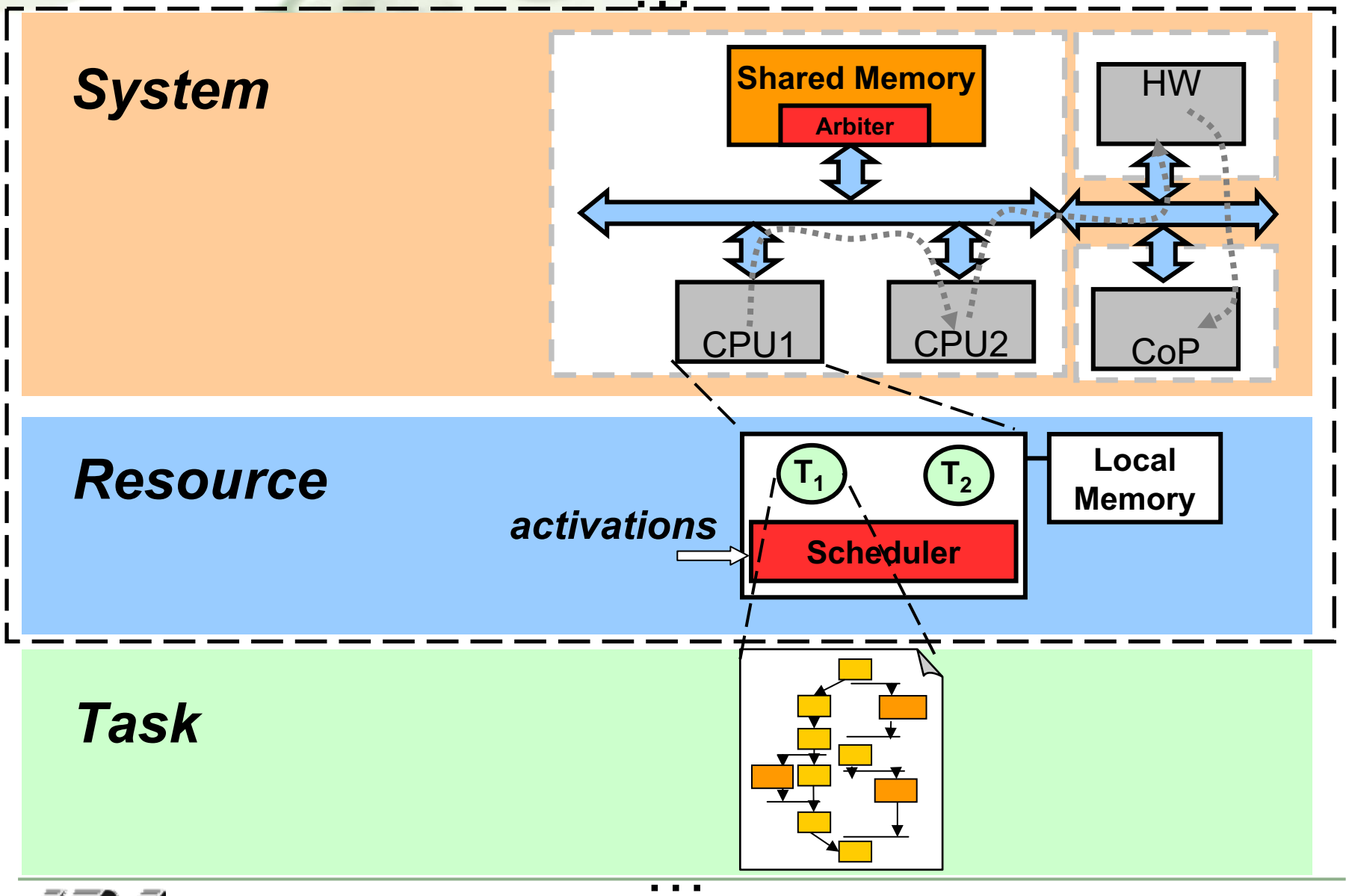
Outline

- Abstractions for the analysis of real-time systems
- Multicore architectures
- Timing implications and countermeasures
- Formal analysis method
- Conclusion

Outline

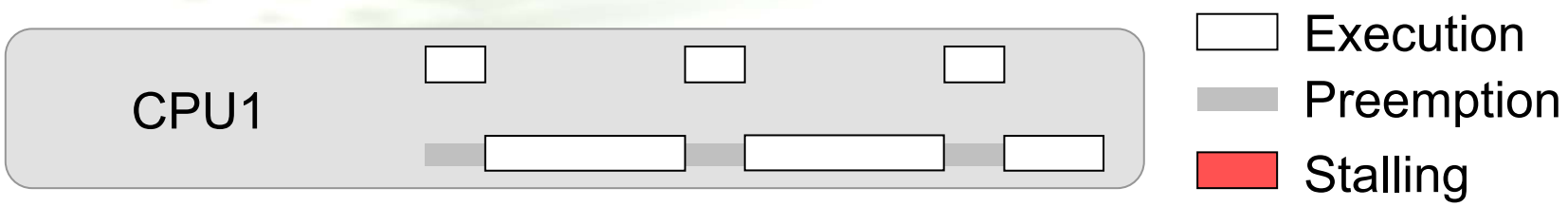
- **Abstractions for the analysis of real-time systems**
- Multicore architectures
- Timing implications and countermeasures
- Formal analysis method
- Conclusion

Software Timing Hierarchy

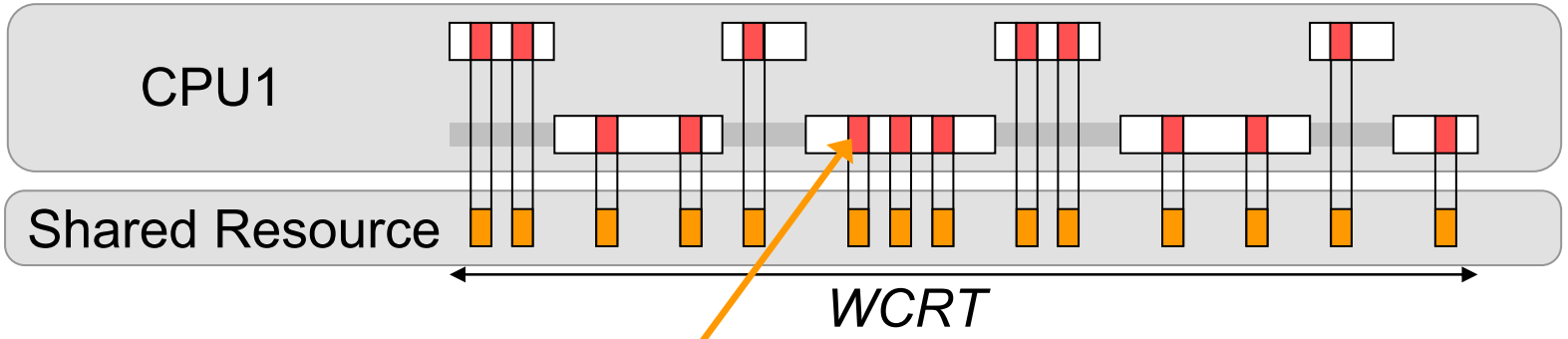


Local Scheduling Analysis

- Single core task execution (classical model)



- Single core task execution (with shared resources)



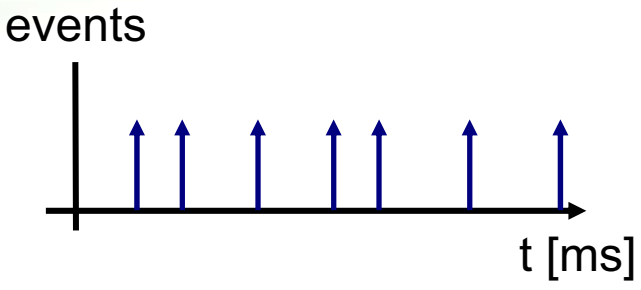
Valid assumption for single processor systems: Shared Resource access times are part of the Core Execution Time C_i

- Large body of methods available to derive WCRTs for different scheduling policies
  - SPP, TDMA, RR, EDF,…
  - considering realistic scheduling effects (context switch times, offsets,…)

Model Task Activation as "*Event Streams*"

- Response Time Analysis requires traffic models:

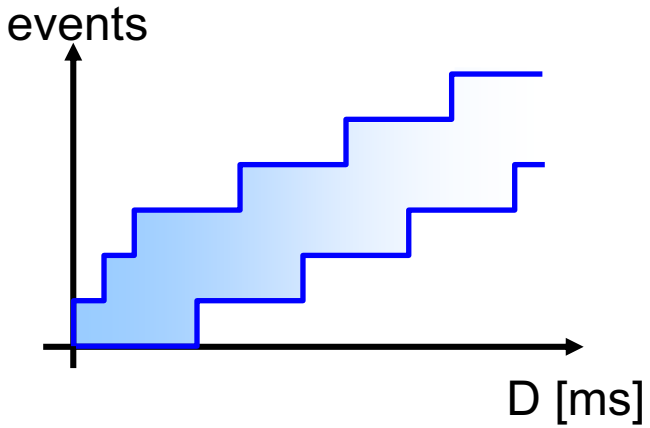
Event Stream



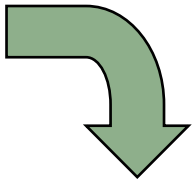
Determined by
- application model (Simulink, LabView, …)
- environment model (reactive systems)
- service contracts (max no of requests per time, …)

Arrival Curves

Derive event bounds

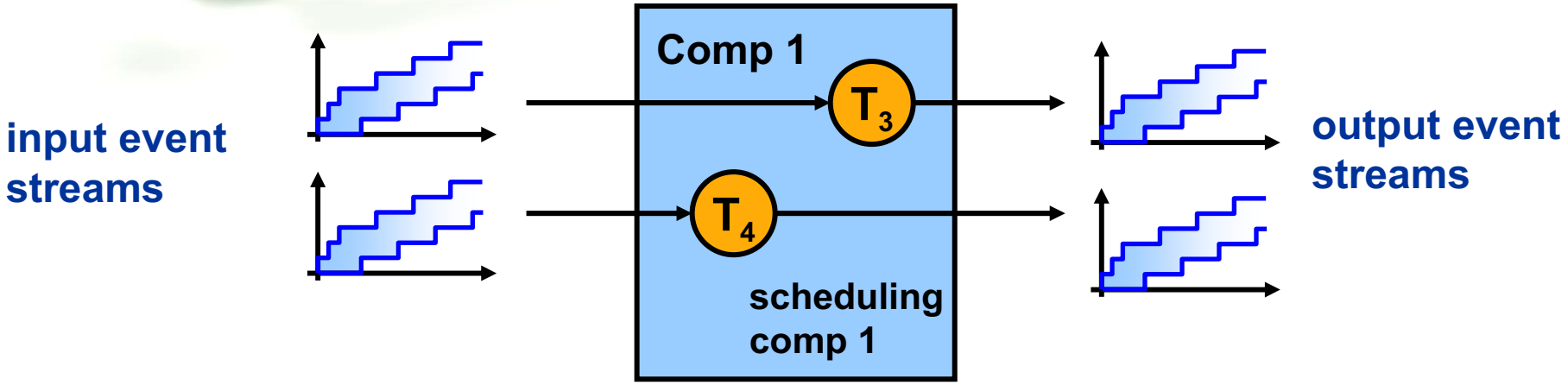


Extract key parameters (optional)



P ***Period***
J ***Jitter***
d_{min} ***Minimum event distance***

Component Performance Analysis



- Output event model of one processor becomes input event model of successor → compose multiple local analyses into system level analysis

- Abstractions for the analysis of real-time systems
- **Multicore architectures**
- Timing implications and countermeasures
- Formal analysis method
- Conclusion

Multicore Architectures

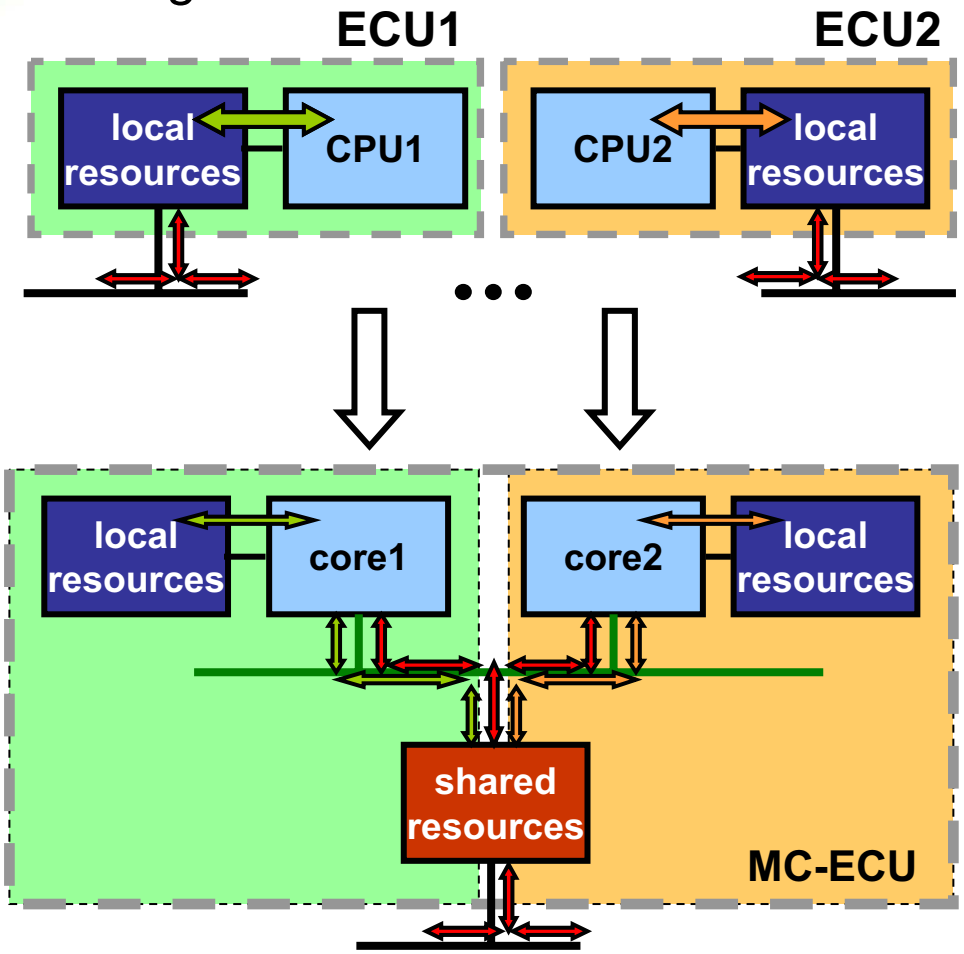
- Expected in near future automotive designs

Current distributed system

- All accesses to local resources
- Bus communication clearly specified and systematic

Multicore system

- Accesses to local and shared resources
- Complicated, interleaved and less systematic communication timing

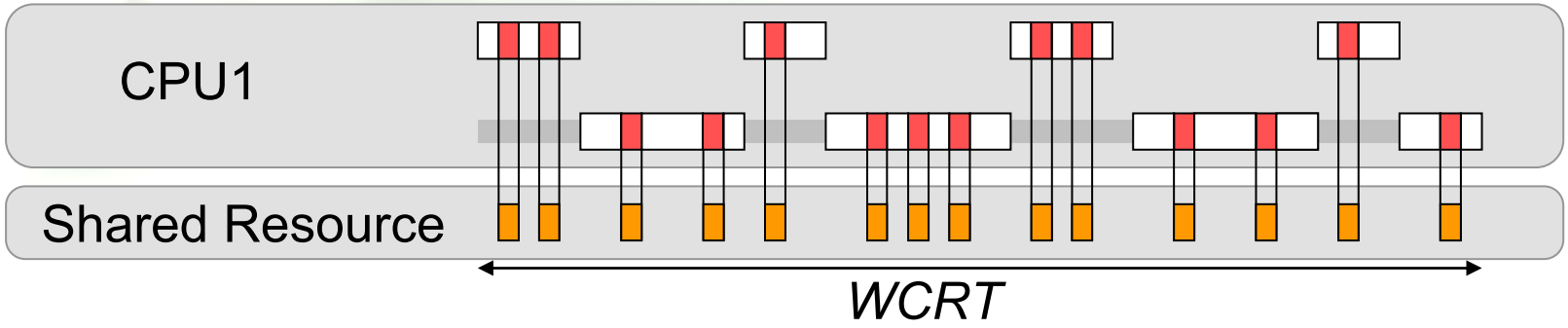


→ **Complex impact on timing**

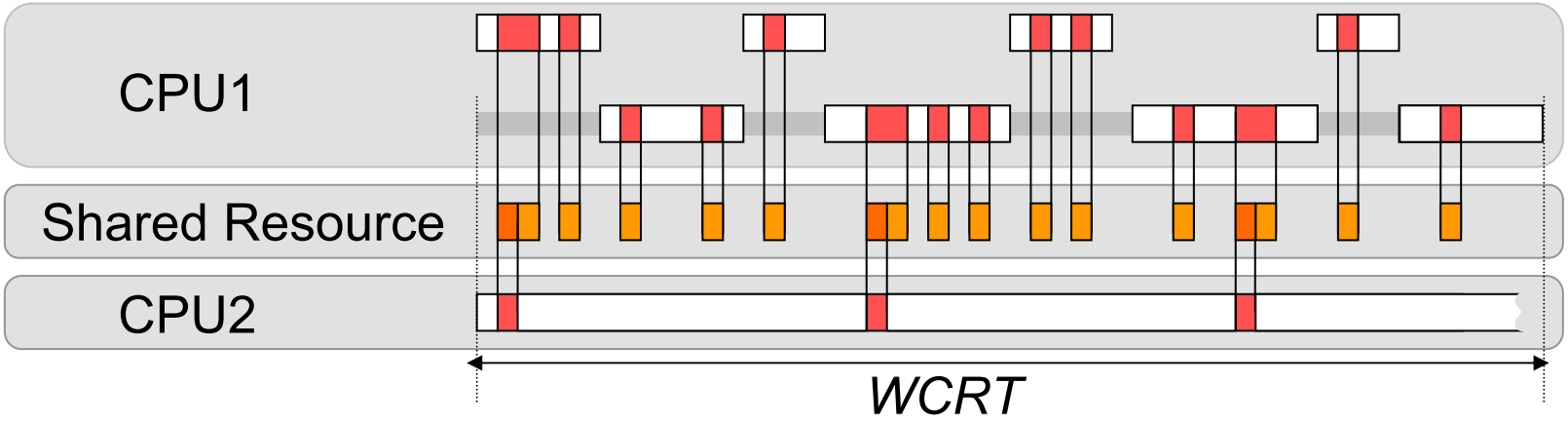
- Abstractions for the analysis of real-time systems
- Multicore architectures
- **Timing implications and countermeasures**
- Formal analysis method
- Conclusion

Task Execution in Multicore

- Single core task execution (with shared resources)

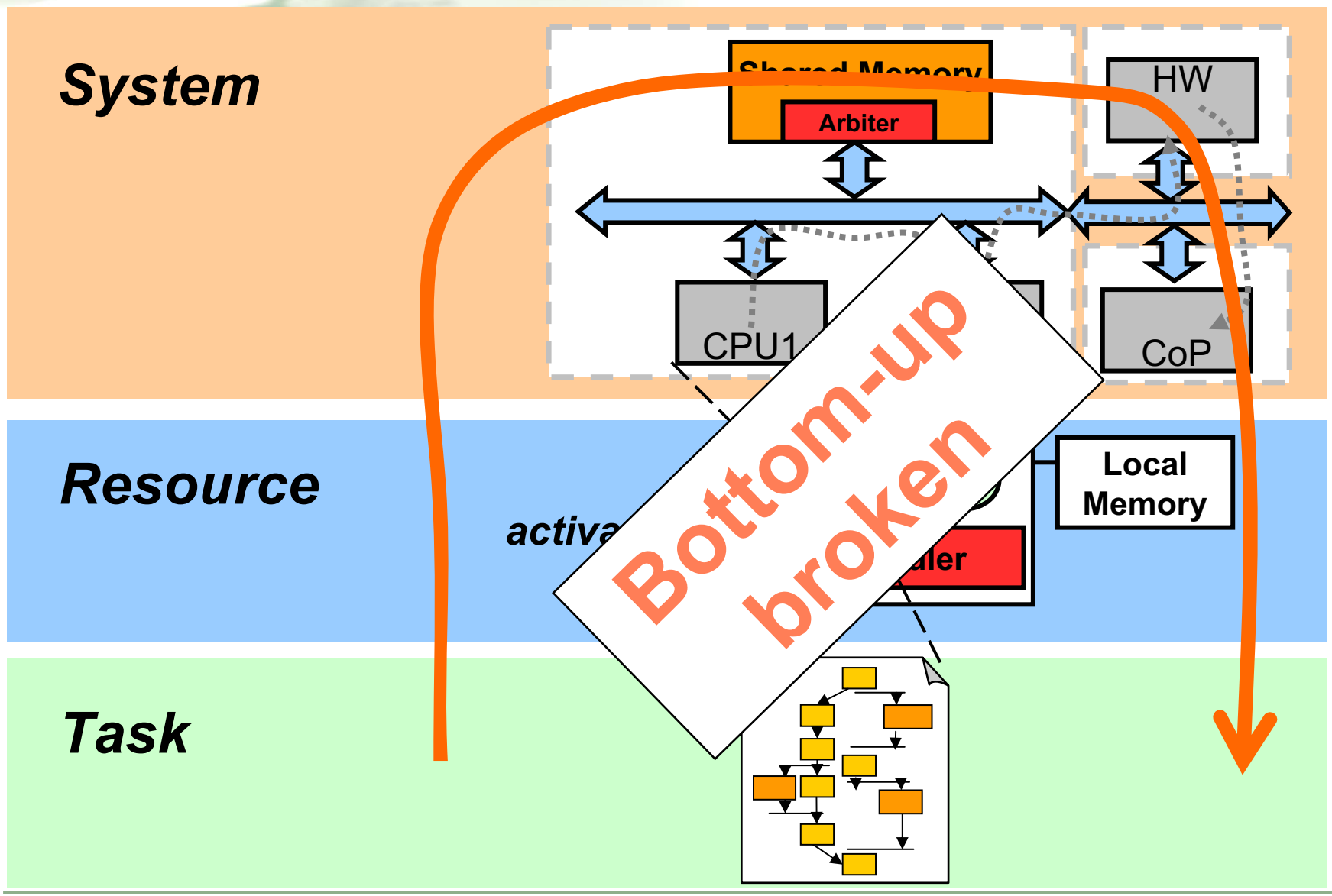


- Multicore task execution with shared resources



- Mapping to multicore architectures **changes timing**
  - Leads to **new timing dependencies between applications!**

Software Timing Hierarchy



Countermeasures

- **Orthogonalize resources**
  - Introduce schedulers that give upper bounds on interference independently of competing streams
  - At least perform traffic shaping
  - → imposes strict hardware guidelines
  - → protection from partially false system specification
  - → prone to over-provisioning (not so much in hard real-time setups)

- **Use formal analysis that covers dynamism**
  - Find realistic upper bounds on application behavior
  - Provide formulas and analysis methods matching actual system
  - → requires comprehensive knowledge of hardware behavior to set up analysis
  - → requires safe assumptions about behavior of the software
  - → allows considering dynamic schedulers and load

- **Mix of the above**

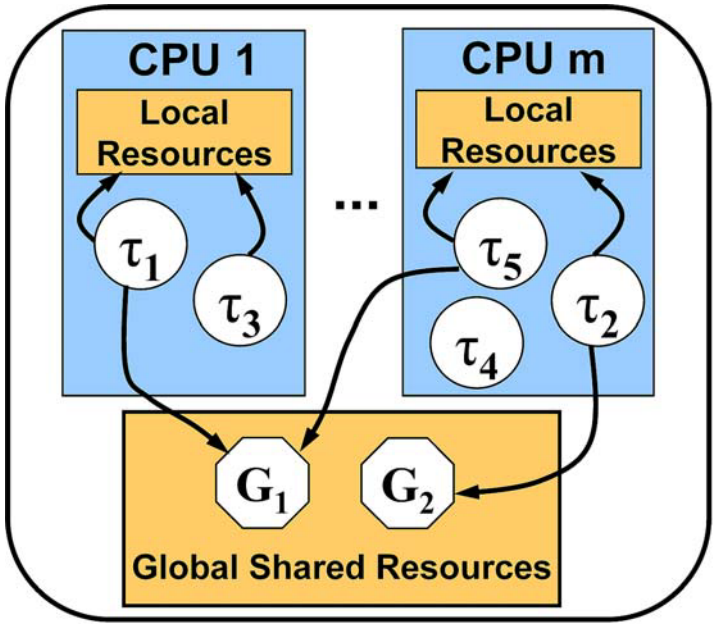
- Abstractions for the analysis of real-time systems
- Multicore architectures
- Timing implications and countermeasures
- **Formal analysis method**
- Conclusion

Formal Analysis Method - Example

- Response Time Analysis for multiprocessor systems (DATE2009)
  - Set of “m” processor systems → each with its own SPP scheduling
  - Static set of tasks $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$
    - statically mapped on the „m“ processors
  - Unique priority space across processors
    - Priority (τ_1) > Priority (τ_2) > ... > Priority (τ_n)
  - Set of shared resources
    - Local and global
    - Arbitration according to MPCP

→ Evolutionary extension of OSEK-AUTOSAR scheduling

→ Matches design practice in automotive domain

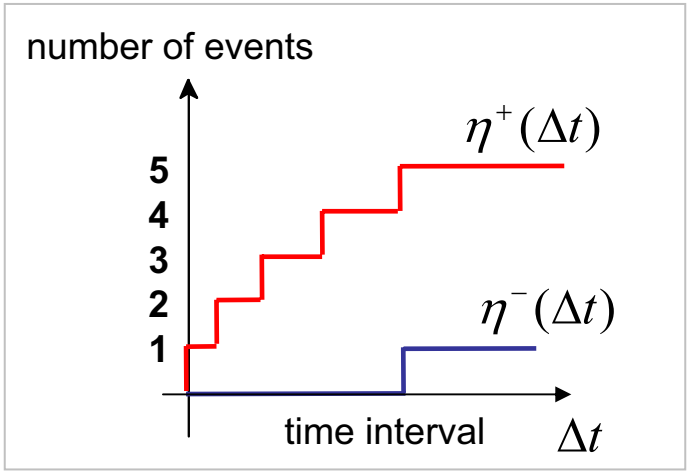


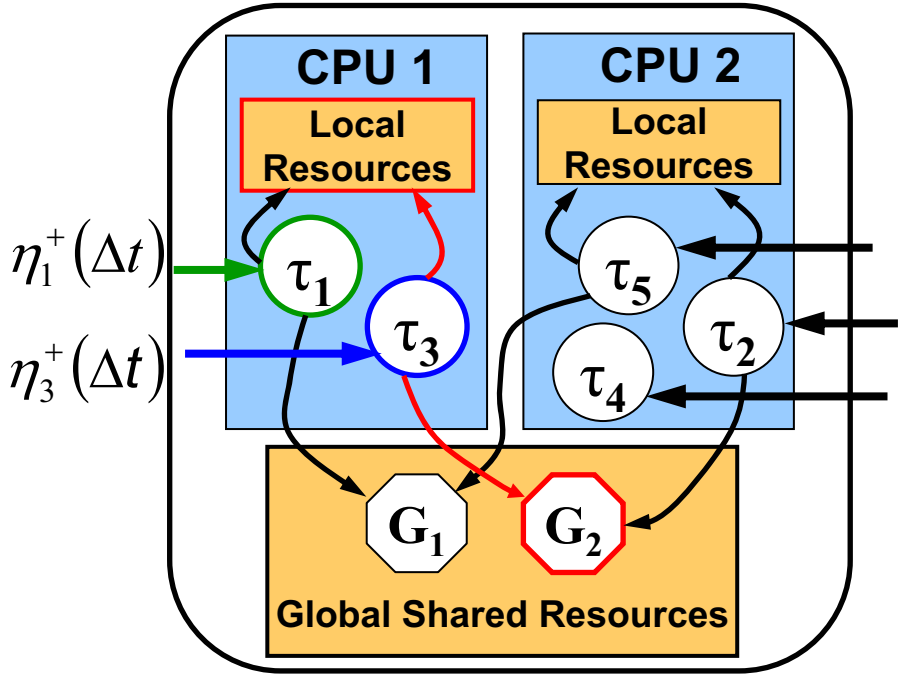
Response Time Analysis

- Worst-case response time R_i of a task τ_i on a processor with SPP
  - **Task own execution**
  - **Interference due to higher priority local tasks**
  - **Blocking time when accessing shared resources**

$$R_i = \eta_i^+(R_i) \cdot C_i + I(R_i) + B_i(R_i)$$

$\eta(\Delta t)$: Number of events per time interval Δt

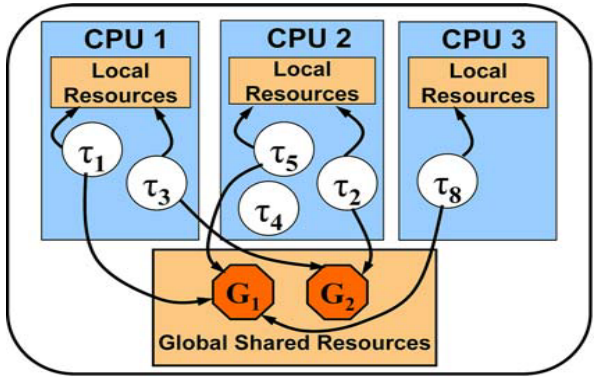


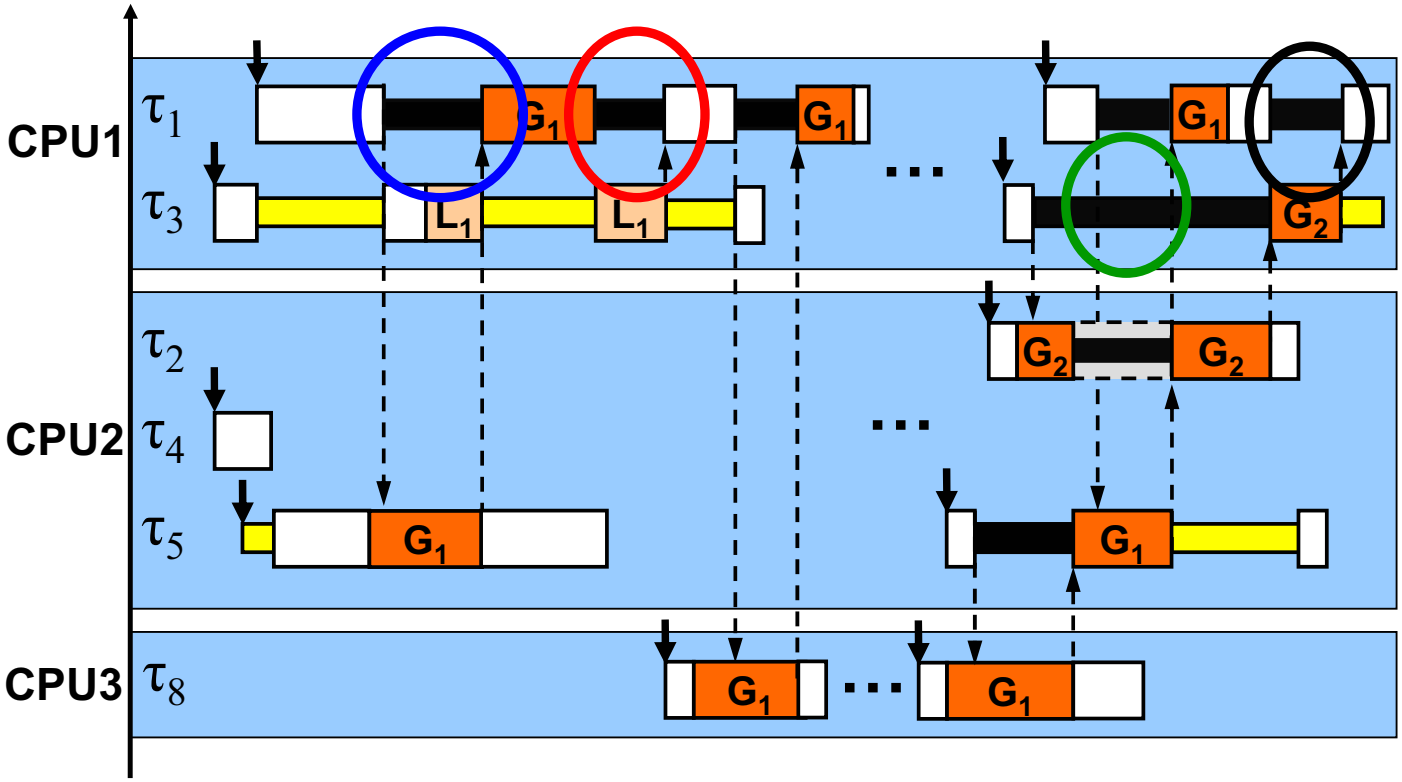


Types of Blocking - according to MPCP

- **Local blocking time**
- **Direct blocking time**
- **Indirect preemption delay**
- **Local preemption delay**

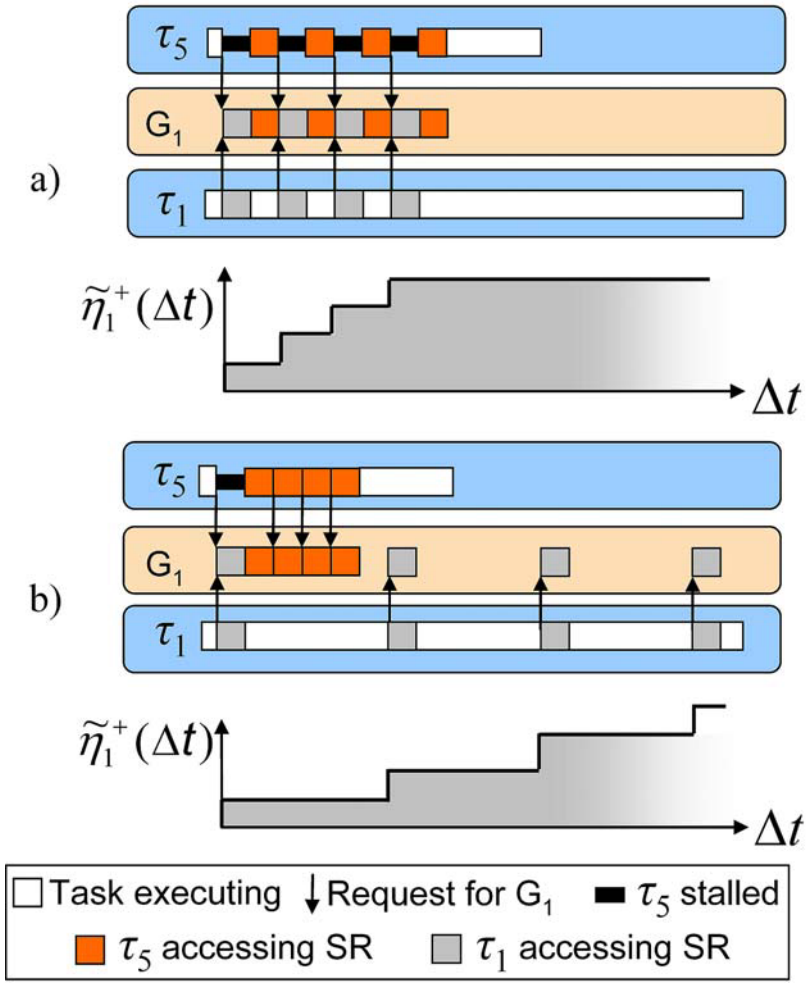
$$B_i(R_i) = B_1(R_i) + B_2(R_i) + B_3(R_i) + B_4(R_i)$$





Derivation of Shared Resource Latencies

- Concept → Use event model concept to capture resource traffic



- Possible upper bounds derivation:

Bound 1: $\tilde{\eta}_1^+(\Delta t) = \eta_1^+(\Delta t + R_1) \cdot n_1$

Bound 2: $\tilde{\eta}_1^+(\Delta t) = \left\lceil \frac{\Delta t}{d_{srr}} \right\rceil$

- Direct blocking time of task τ_5:

$$B_{5,direct}(R_5) = \tilde{\eta}_1^+(R_5) \cdot \omega_1^{G_1}$$

$\omega_1^{G_1}$ - Time duration when G_1 is blocked by τ_5

Multiprocessor Response Time Analysis

- Couple local scheduling analysis with the blocking time analysis

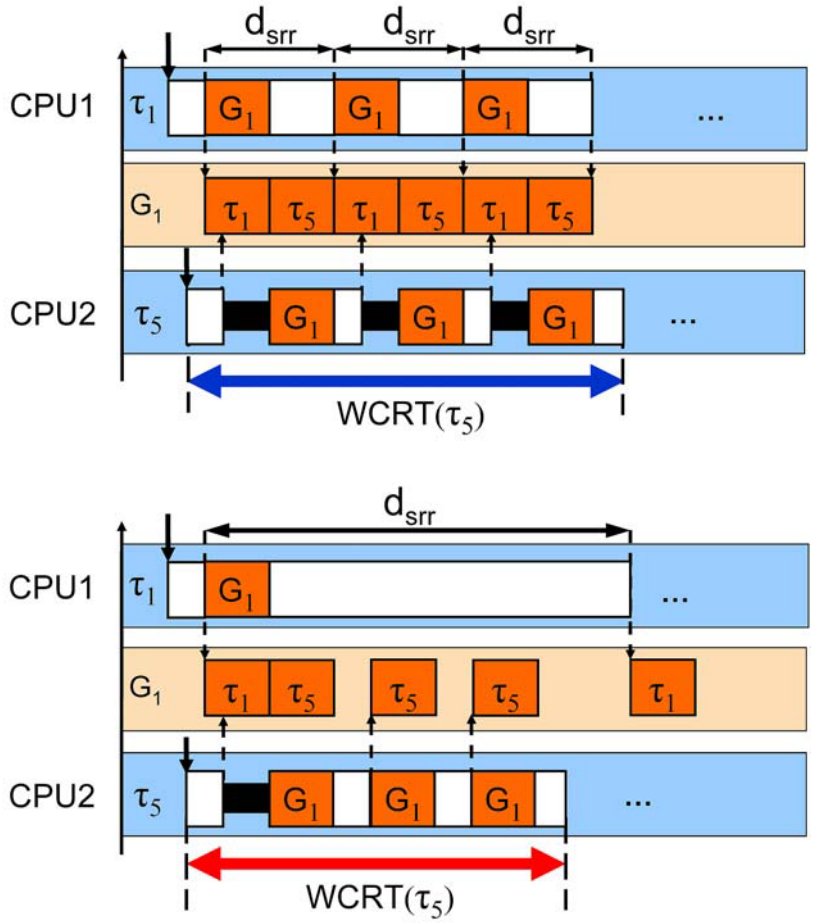
$$R_i = \eta_i^+(R_i) \cdot C_i + \sum_{\forall \tau_j \in hpl(\tau_i)} \eta_j^+(R_i + R_j) \cdot C_j + B_i(R_i)$$

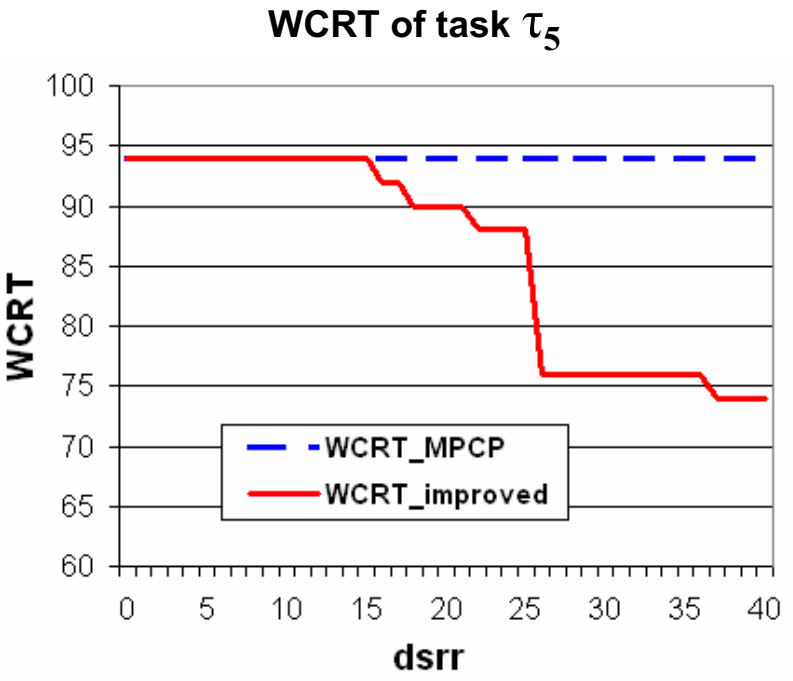
Analysis issues:

- Critical instance scenario not valid due to possible self-suspension of higher priority tasks
  - → compute response times top-down (higher priority first)
- $B_i(R_i)$ depends on the load imposed by other tasks, but their response time has possibly not been calculated
  - → Iterative computation of tasks WCRT until general convergence
  - → For lower priority tasks use a load derivation that is independent of WCRT

Applicability

- Modeling and Analysis of multiprocessor systems implemented in SymTA/S (Symbolic Timing Analysis for Systems)





dsrr - minimum distance between two consecutive shared resource requests

- Abstractions for the analysis of real-time systems
- Multicore architectures
- Timing implications and countermeasures
- Formal analysis method
- **Conclusion**

Conclusion

- Mapping ECU functions on multicore **impacts function timing** due to shared resources → Cross-processor interference

- **New scheduling** and **analysis algorithms** are available for **multicore**
  - Compatible to system level analysis (DATE 2009)
  - Can work with incomplete and estimated task sets

Use formal analysis methods to:

- **Optimize performance and cost**:
  - In **early design stages** to guide towards optimal design choices
  - Refine input data **during design process** to provide verification strength guarantees for final product
- Achieve **compliance to safety standards**

Thank you
for your attention !

Questions ?

Bibliography

[1] Negrean, M., Schliecker, S., and Ernst, R. "Response-Time Analysis of Arbitrarily Activated Tasks in Multiprocessor Systems with Shared Resources." In *Proceedings of Design, Automation and Test in Europe Conference (DATE), Nice, France* (April 2009).

[2] Schliecker, S., Rox, J., Negrean, M., Richter, K., Jersak, M., and Ernst, R. "System Level Performance Analysis for Real-Time Automotive Multi-Core and Network Architectures." *IEEE Transactions on Computer Aided Design* (July 2009).

[3] Richter, K., Jersak, M., and Ernst, R. "Learning Early-Stage Platform Dimensioning From Late-Stage Timing Verification." In *Proceedings of Design, Automation, and Test in Europe (DATE), Nice, France,* (April 2009).

[4] Schliecker, S., Negrean, M., Nicolescu, G., Paulin, P., and Ernst, R. "Reliable Performance Analysis of a Multicore Multithreaded System-On-Chip." *Proceedings of the 6th International Conference on Hardware/Software Codesign and System Synthesis (CODES-ISSS), Atlanta, GA* (October 2008).