MpSoC in Safety Related Applications

R. Ernst
TU Braunschweig

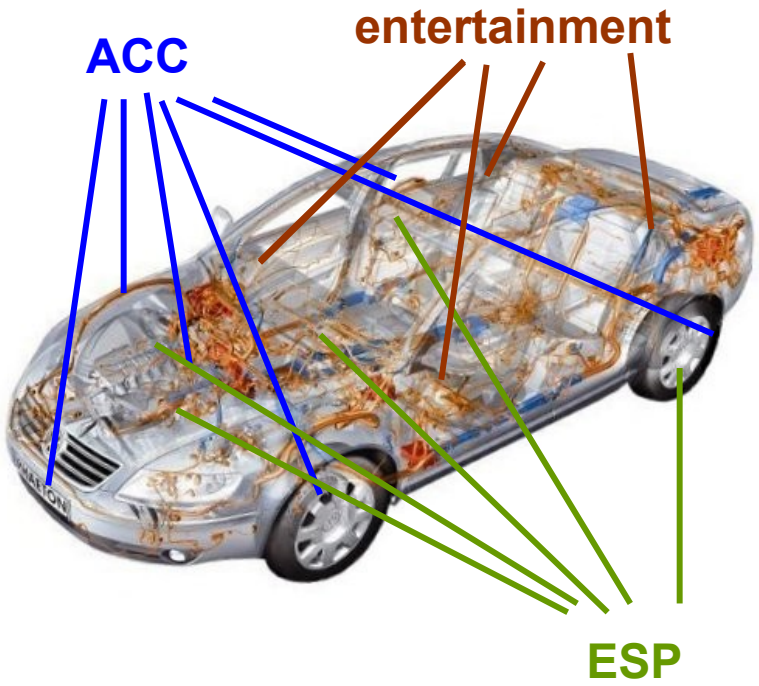
MpSoC 2009, Savannah

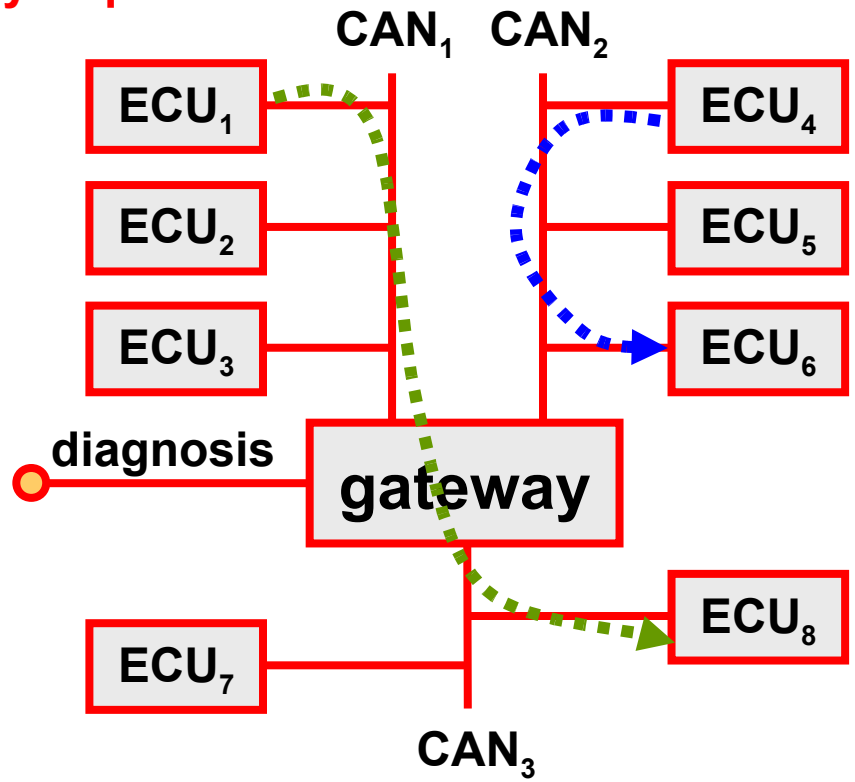
Motivation

- MpSoC are reaching safety critical applications
  - evolutionary transition from single core distributed systems
- two different approaches
  - case 1: merging previously distributed ECUs on MpSoC architectures
  - case 2: load distribution for higher performance/lower power
- complex impact on timing

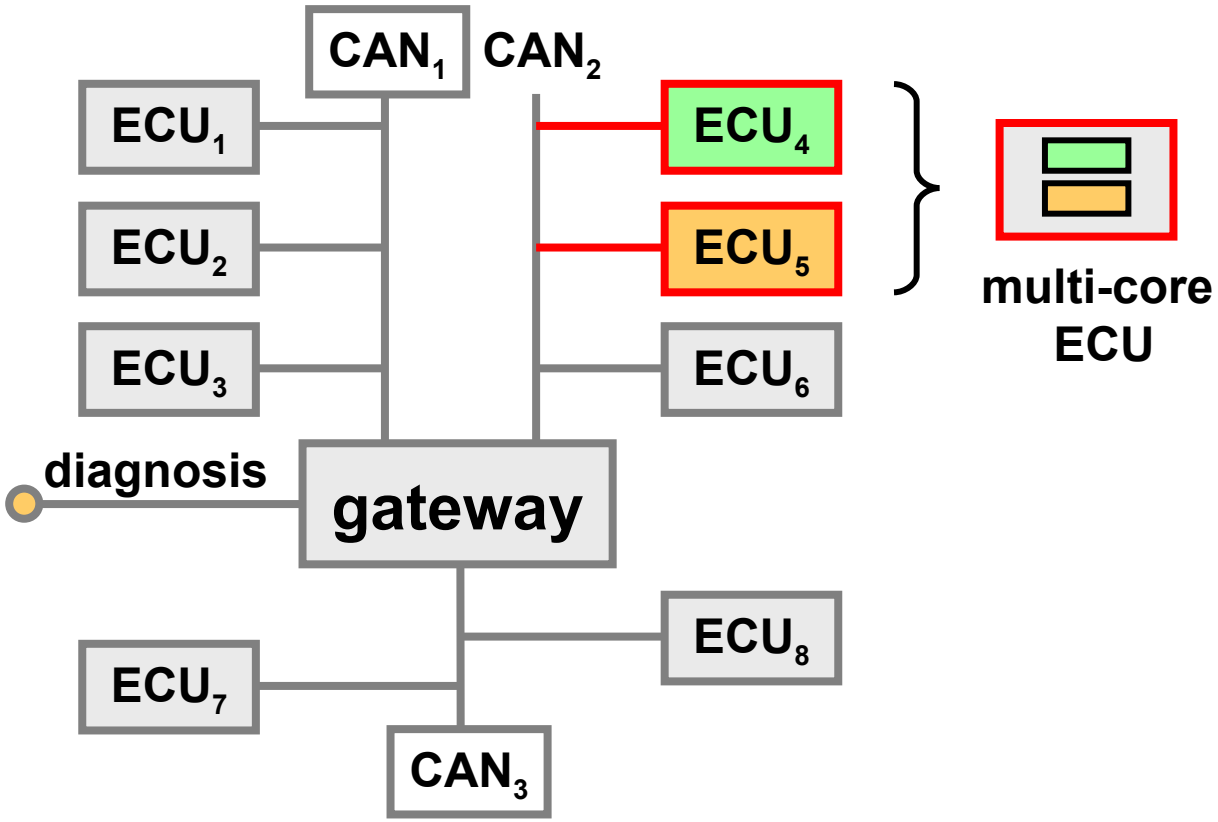
Example: Automotive Networked Systems

- distributed networked system with complex end-to-end time constraints and numerous integrated functions on shared resources
  - example (Daimler): 55 ECUs, 7 buses with gateways
- different service levels and safety requirements





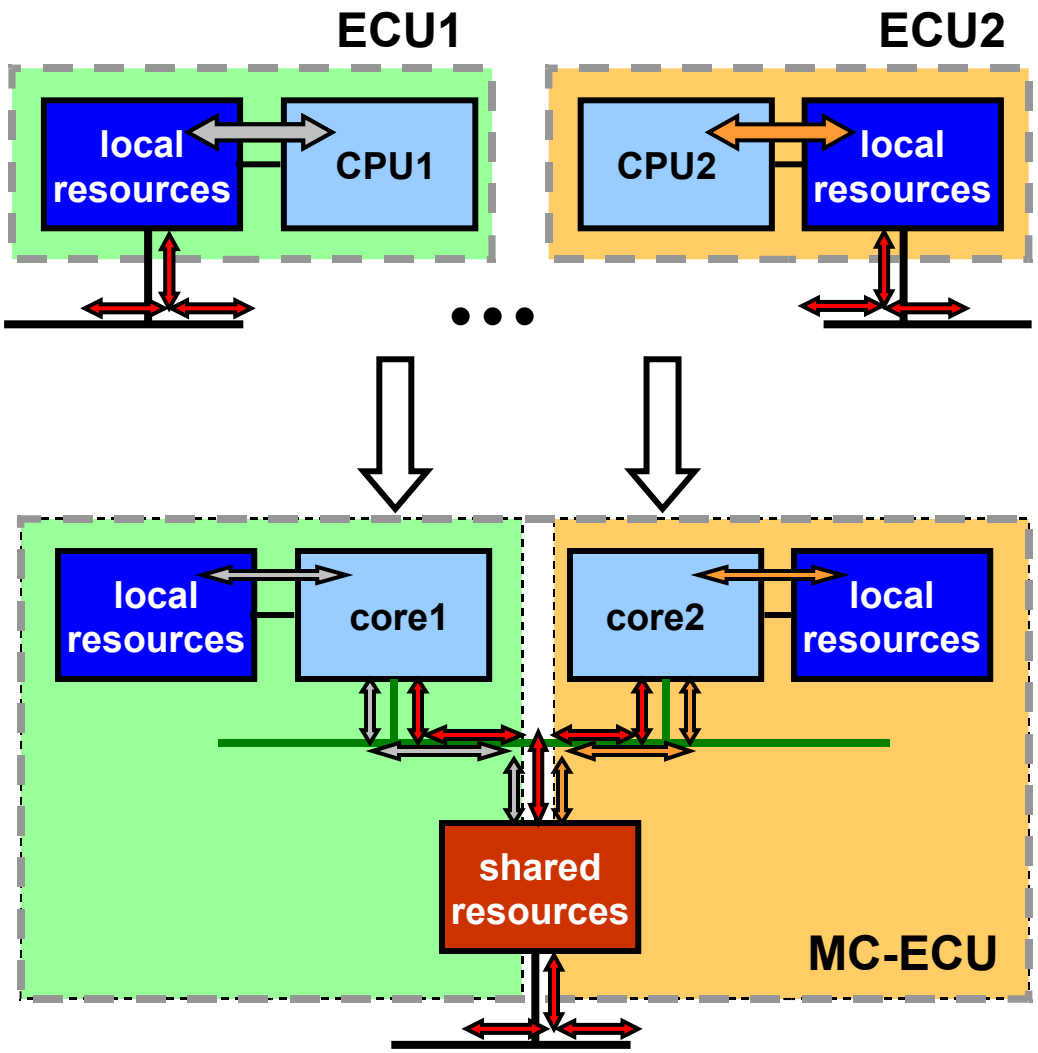
Case 1: Merging ECUs on MpSoC-Architectures



Merging ECUs on MpSoC-Architectures

Current distributed system

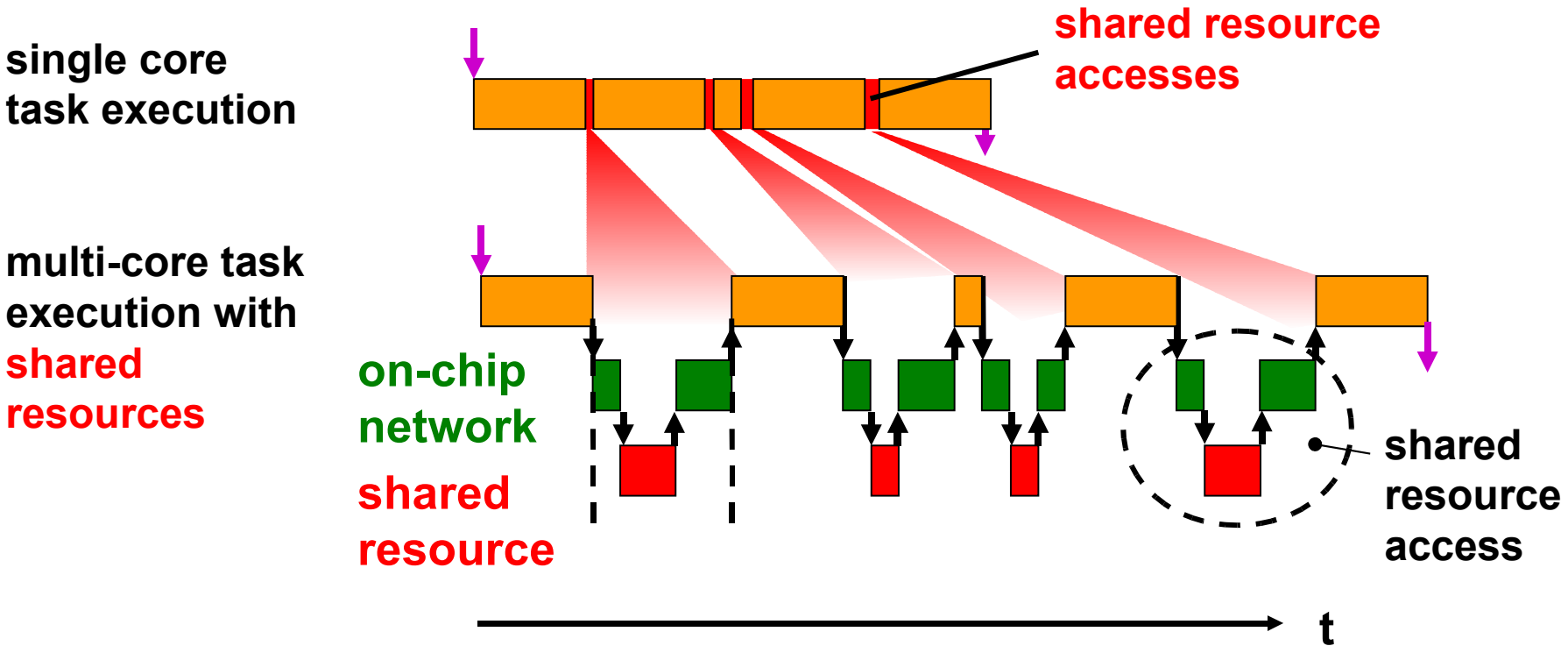
- **all accesses to local resources**
- **bus communication clearly specified and systematic**



Multi-core system

- keep task sets and functions separate
- accesses to local and shared resources
- complicated, interleaved and less systematic communication timing

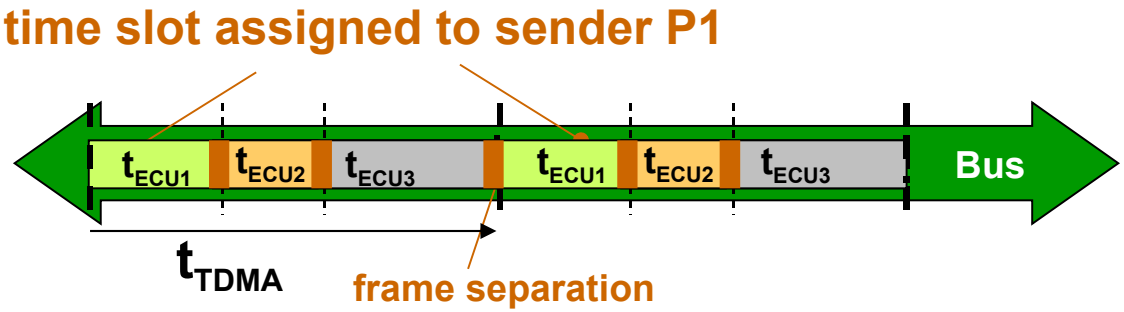
Multi-core Task Execution



- mapping to multi-core changes timing
  - leads to new timing dependencies between applications!
- function separation and virtualization do not include timing
  - verification becomes more complicated than for distributed systems communication

One Approach: Conservative Design

- conservative design principle – strict separation of resource access



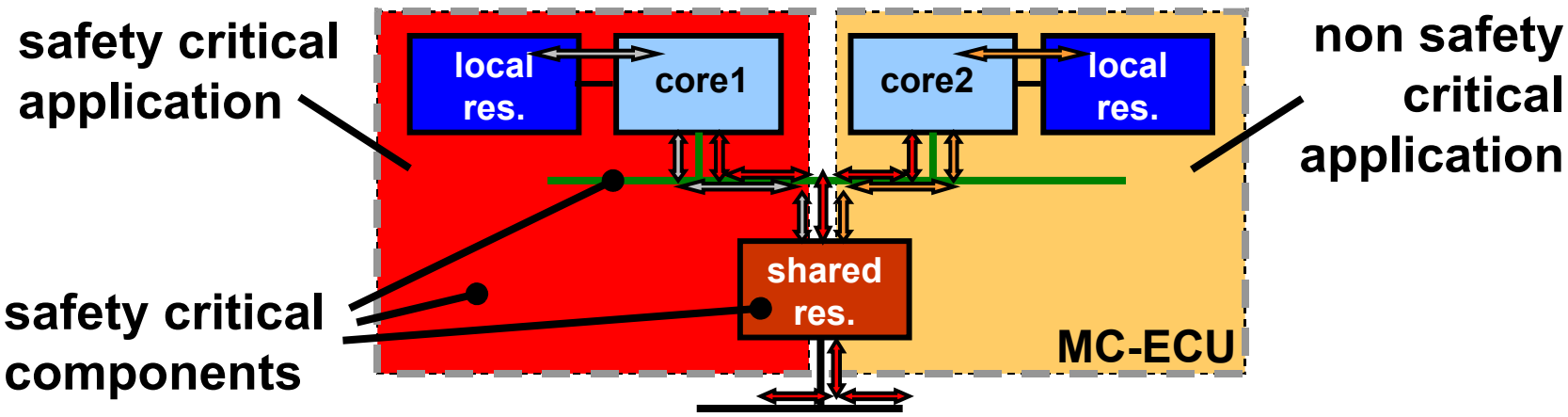
Time triggered architecture – TDMA

- periodic assignment of fixed time slots for resource access
  - unused time slots remain empty - *efficiency*
  - can be extended to system-level time triggered architecture
- main advantage is predictability in integration
  - limited by state dependent resource behavior (memories)

⇒ *no silver bullet in system timing separation*

MpSoC in Safety Critical Applications

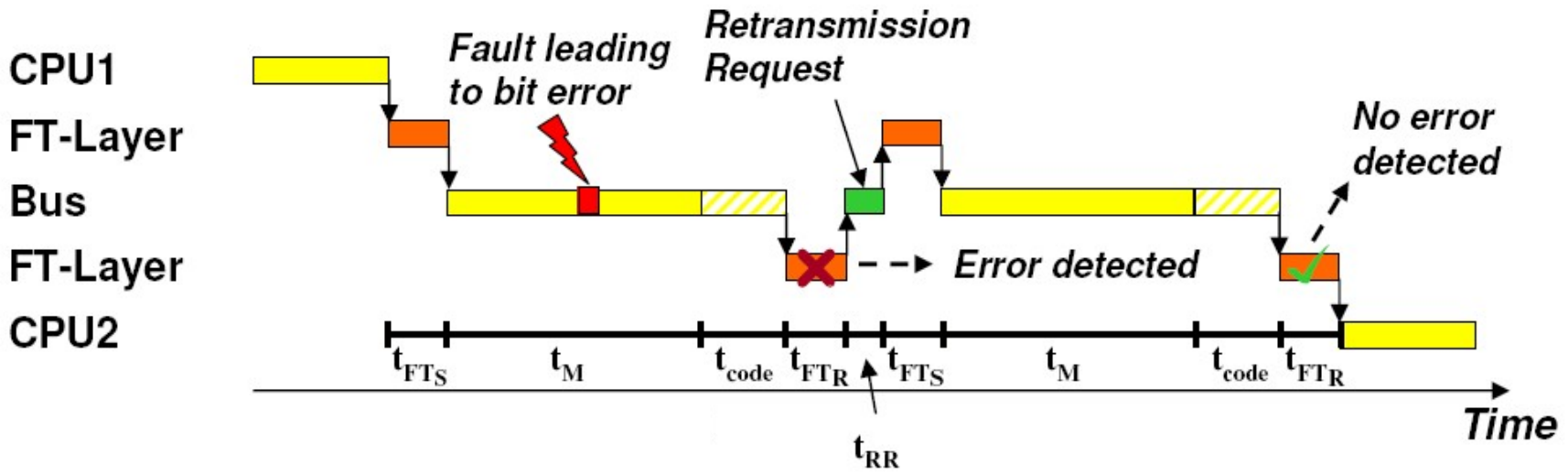
- higher safety requirements require physically separate channels
  - ECUs can still be merged as long as redundancy is preserved
- merging results in MpSoC with functions of different criticality *mixed criticality*
  - resulting ECU subject to highest safety standard involved



- faults require special attention
  - includes deadline violations caused by transient errors
  - scheduling can be exploited to improve robustness of systems with different criticalities

Example: Transient Faults in On-chip Communication

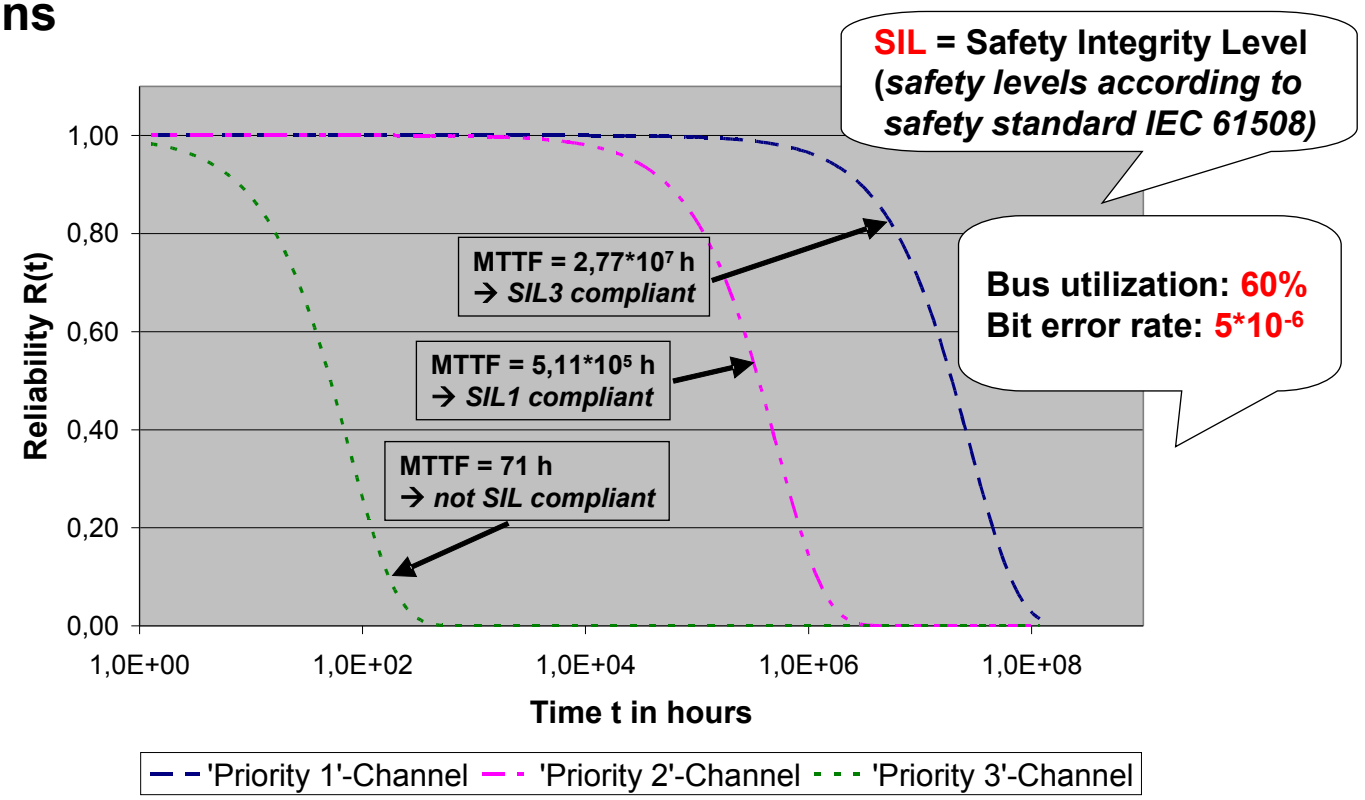
- example: retransmission



- retransmission affects response times, threatens deadlines
  - deadline violation turns transient fault into *failure*
- fail safe system (e.g. automotive steering, brake)
  - single failure *enforces* switch to fall back solution to avoid hazard

Scheduling Can Reduce Failures

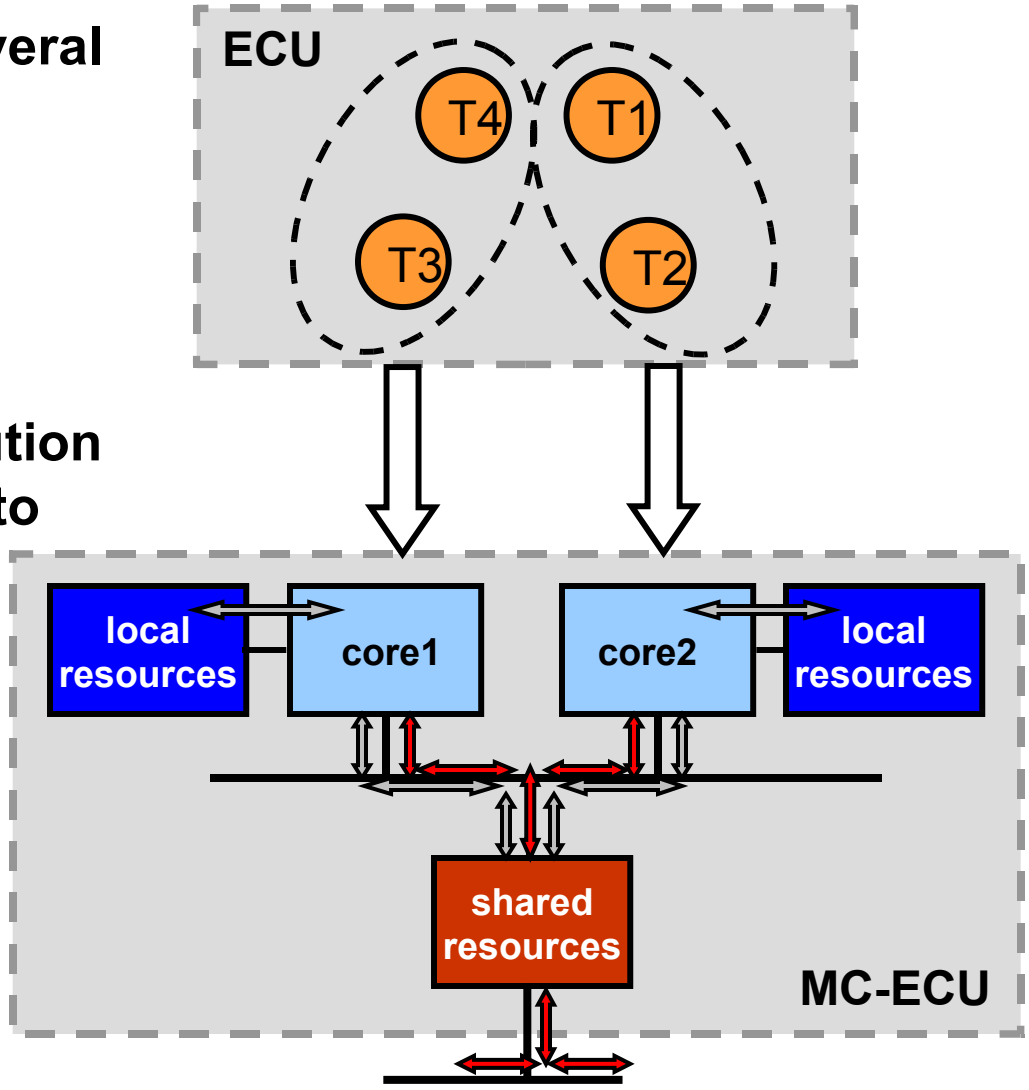
- example: static priority preemptive communication scheduling (cp. CAN bus)
- priority assignment optimized to minimize failures of safety critical functions



- exploitation needs formal guarantees – *research topic*

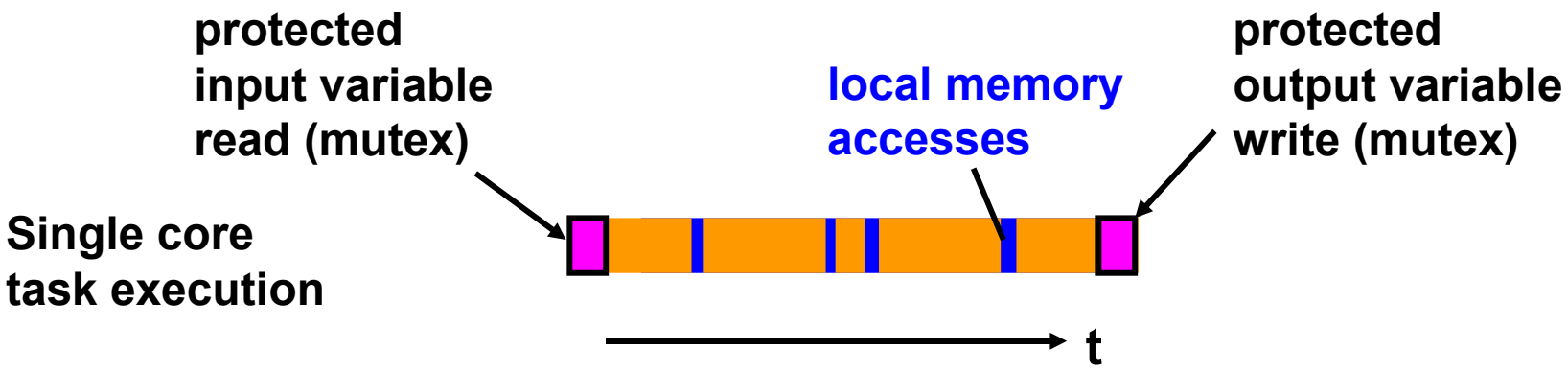
Case 2: Load Distribution (Parallelization)

- distribute task set to several cores
- task communication is mapped to core communication
- preferably static distribution as an evolutionary step to single scheduling

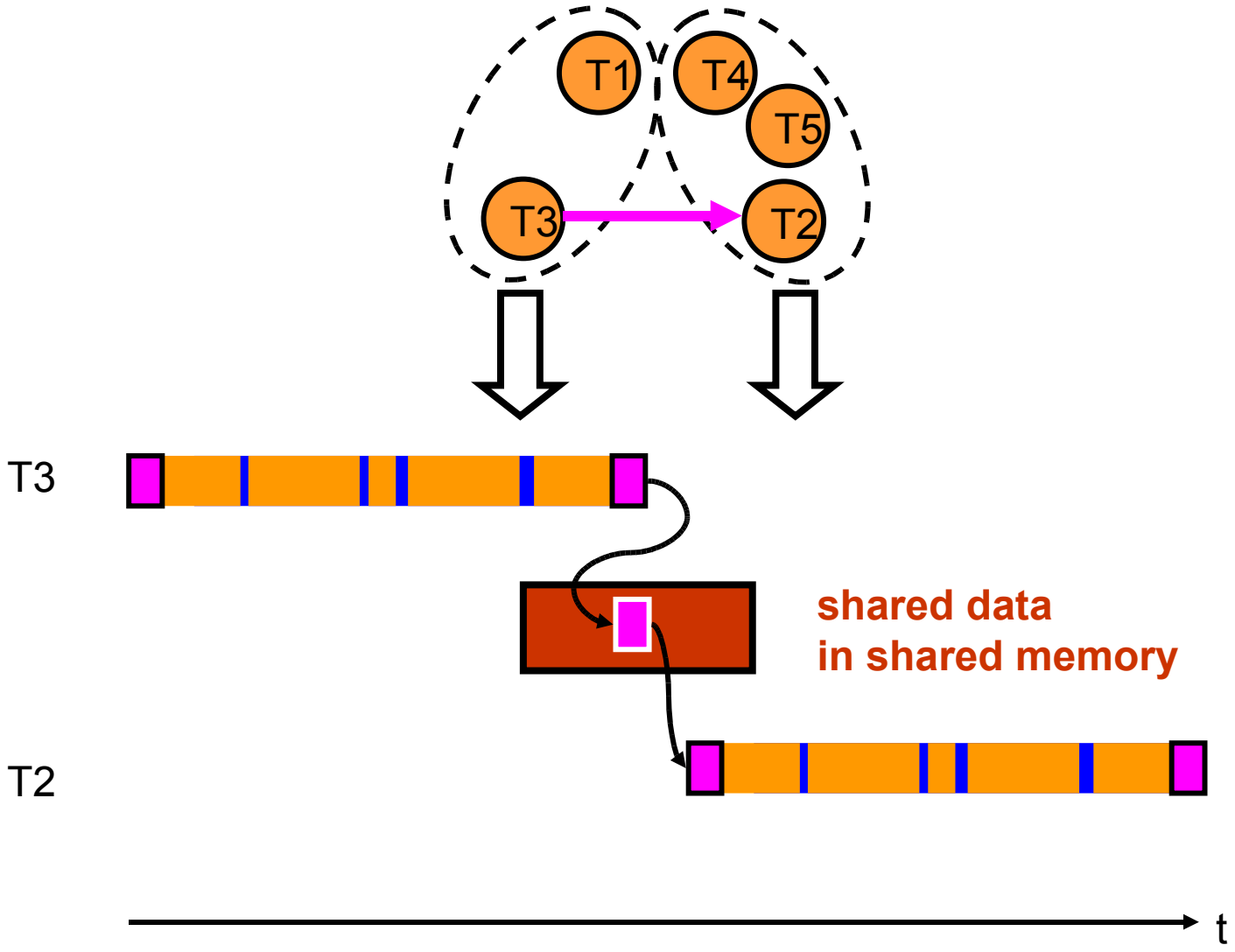


Task Communication in OSEK and AUTOSAR

- single ECU communication uses shared memory
- shared memory access is protected (mutex) for data consistency
- similar approach for other shared resources



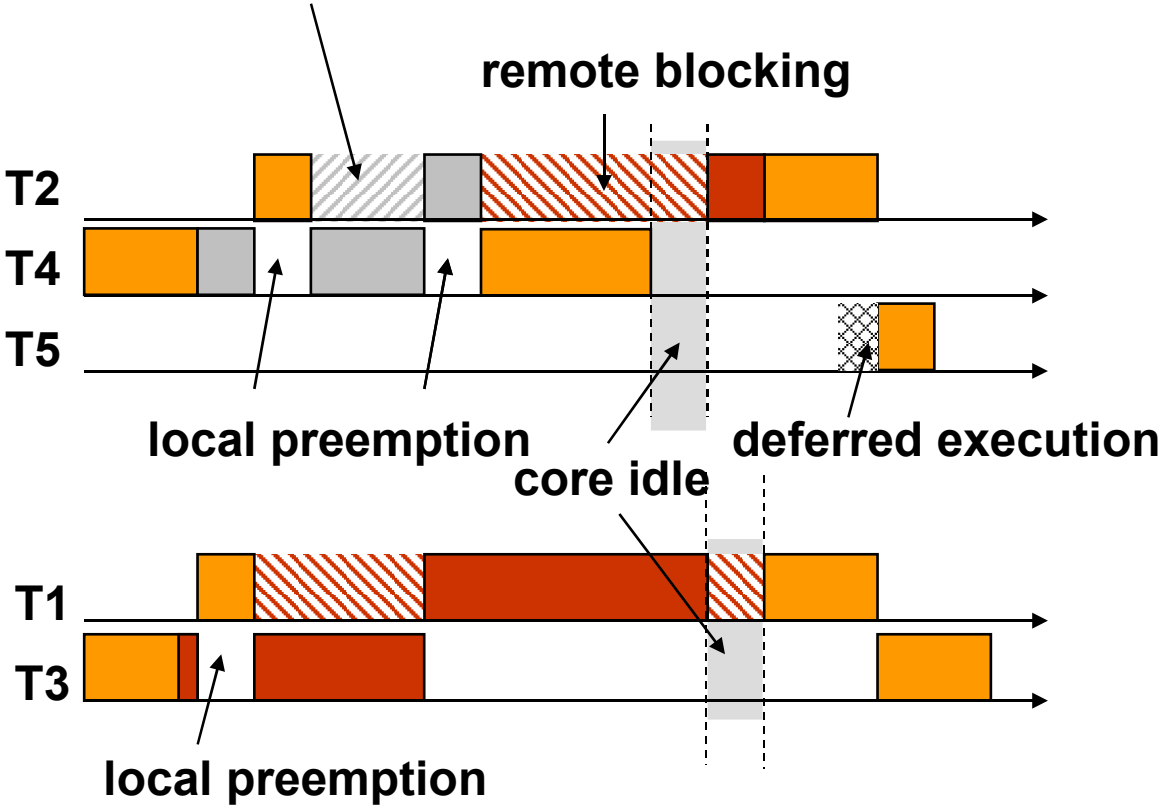
Load Distribution with Shared Memory Communication



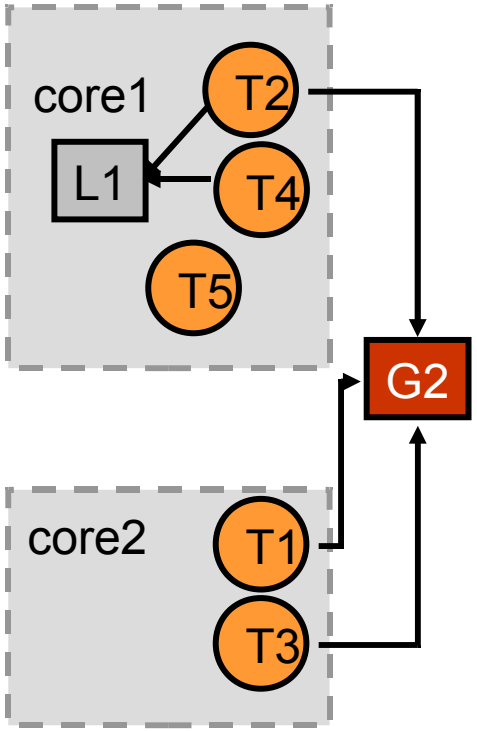
New Effect: Remote Blocking

- remote blocking may increase load and lead to deadlocks

Direct blocking
(and all other single processor blocking components)



L1: local resource
G2: global resource



Conclusion

- merging ECU functions on multiple cores impacts function timing due to resource sharing
- no silver bullets for integration available
- safety critical and mixed criticality systems require new approaches
- load distribution has critical side effects in current software standards that threaten performance
- active research into new scheduling and analysis methods

Literature Formal Methods for Performance Analysis

- DATE 2008 tutorial – overview on performance analysis
  - www.ida.ing.tu-bs.de/~ernst
- this talk
  - Maurice Sebastian, Rolf Ernst. Modeling and designing reliable on-chip-communication devices in MPSoCs with real-time requirements. Proc. ETFA, Hamburg, 2008.
  - Maurice Sebastian, Rolf Ernst. Reliability and Safety-Guarantees in Modern MpSoCs with Real-Time Requirements. In: Proc. edaWorkshop 2009, Dresden.
  - Mircea Negrean, Simon Schliecker, Rolf Ernst. "Response-Time Analysis of Arbitrarily Activated Tasks in Multiprocessor Systems with Shared Resources." In *Proc. of Design, Automation, and Test in Europe (DATE)*, Nice, France, April 2009.