Effects of Arbitrary Hardware Faults on Multicore Scheduling in Safety-critical Applications

Evaluation by enhanced Markov models and discrete event simulation

Stefan Krämer

12th June 2014

Contents

1. Motivation - Background
2. Reliable Multicore System Architecture
3. Analysis by Markov Model
4. Analysis by Discrete Event Simulation
5. Conclusion & further work

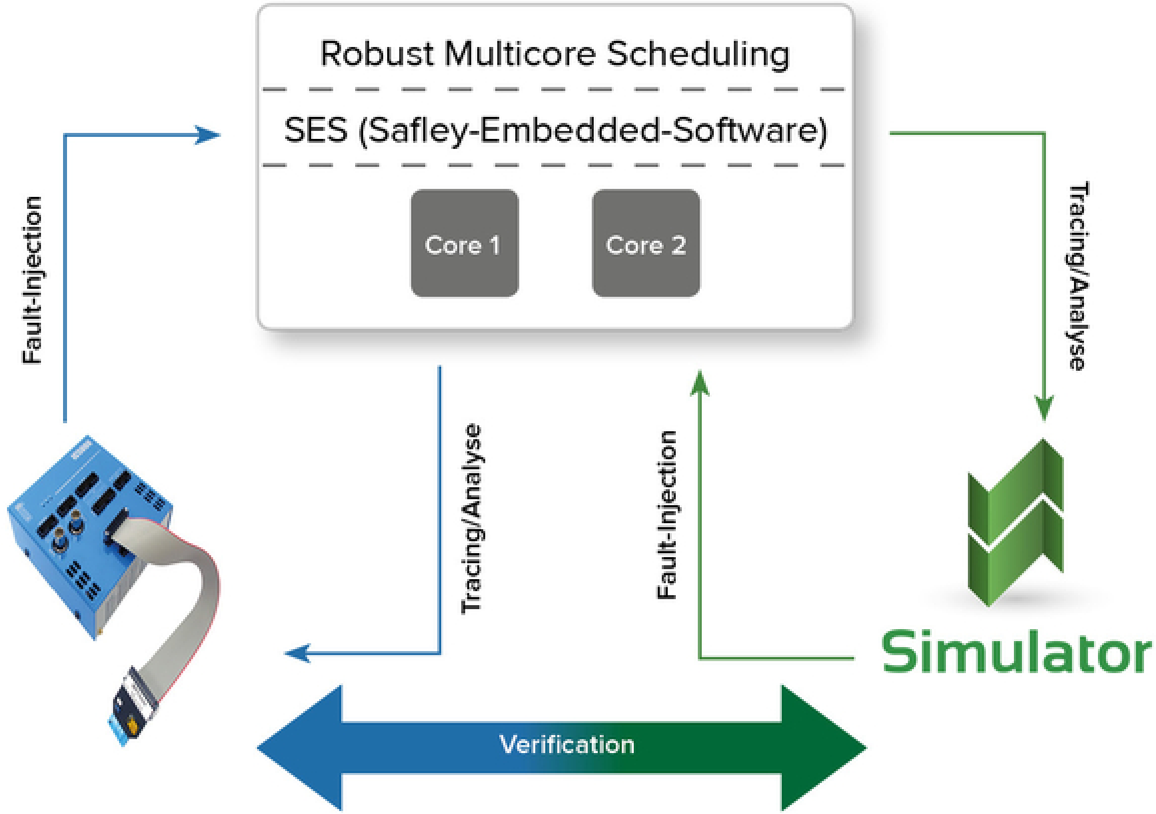
Motivation

- Increased performance demand in real-time systems
- Harder requirements for safety, reliability and availability
- Decreasing feature size on silicon results in more probable transient hardware faults
- Integration of different applications on one ECU

➡ Reliable, fault-tolerant, multi-core, real-time operating system for mixed criticality embedded applications

Motivation - Background

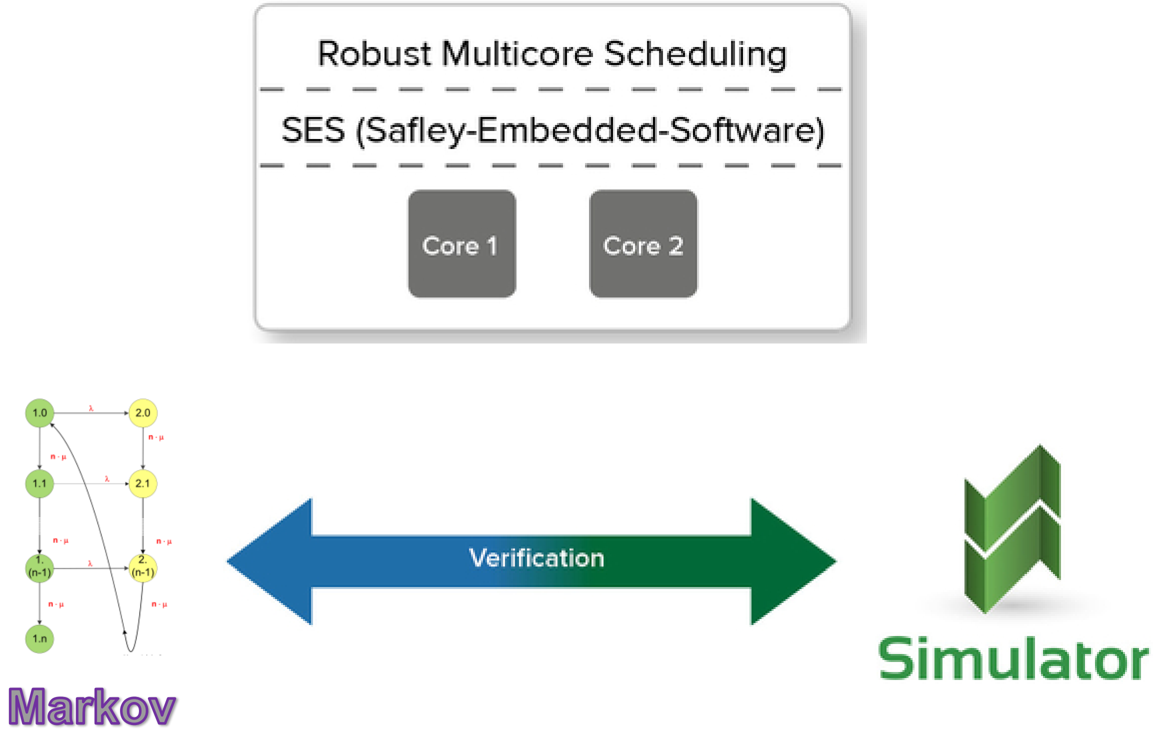
- Project - Goal



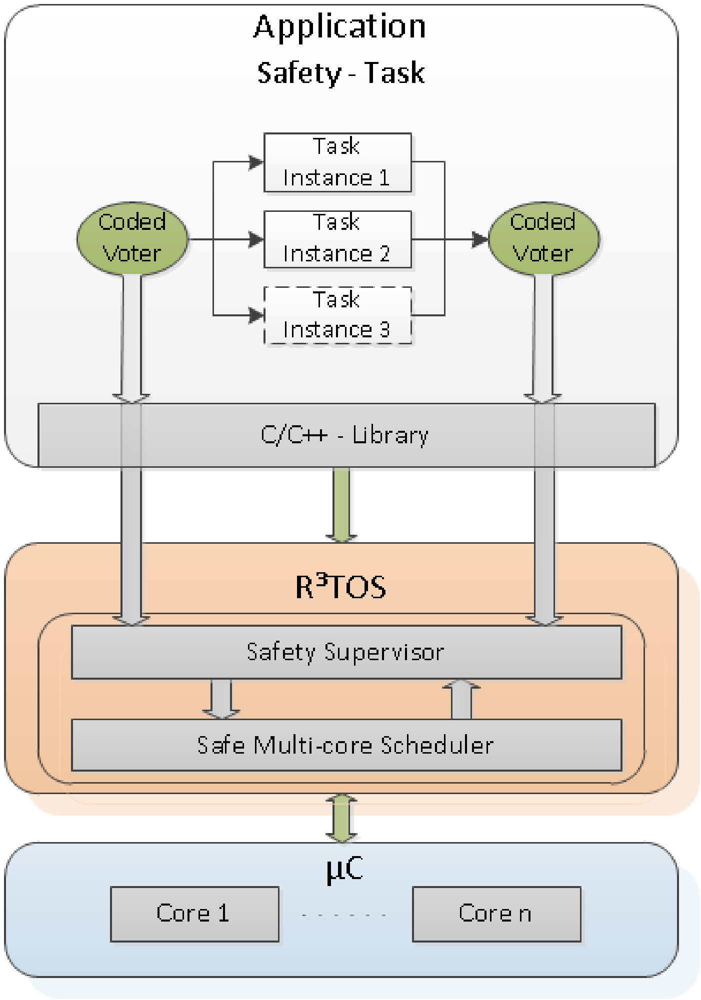
© Timing Architects Embedded System GmbH, ZeloS³

Motivation - Background

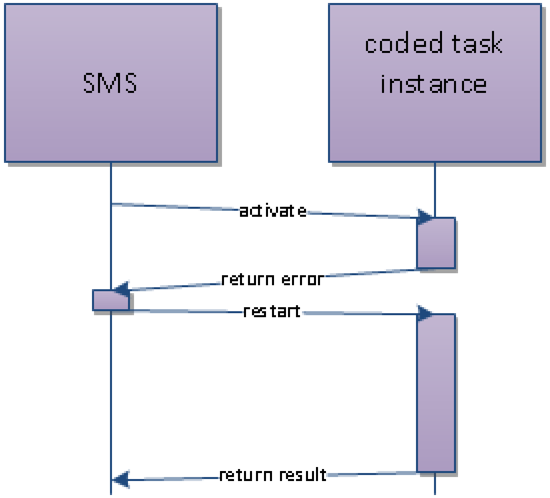
- Sub – Goal: Verification of Discrete Event Simulation



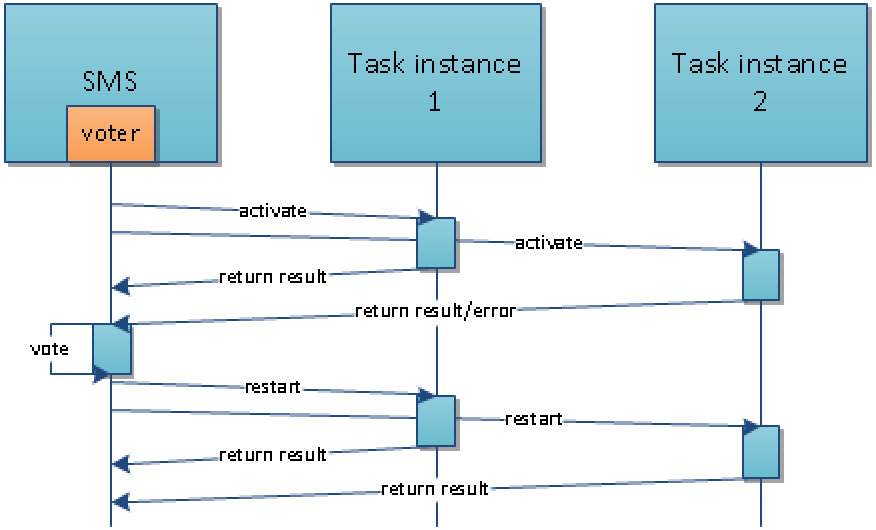
- Scalable, generic System Architecture



Coded task processing



Redundant task processing



- Analytic Approach
  - Reliability analysis:
    - abstracted to simplified model
    - e.g. Markov models or network modeling
  - Proof of feasibility of Scheduling:
    - on a single core analytically possible
    - for global multicore scheduling often impossible

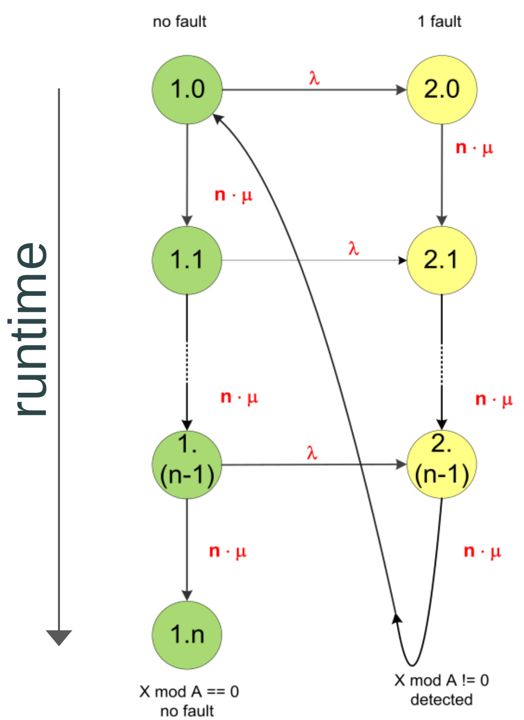
- Simulation based Approach
  - More detailed model for
    - Variance of task execution
    - Influence of transient faults
  - Combined consideration of error and timing model

- Safe Task Execution
  - Analyzed by:
    - Discrete Event Simulation
    - Markov Model
  - Analyzed Scenarios:
    - Coded Processing
    - Symmetric Redundant Processing
  - Task Parameters:

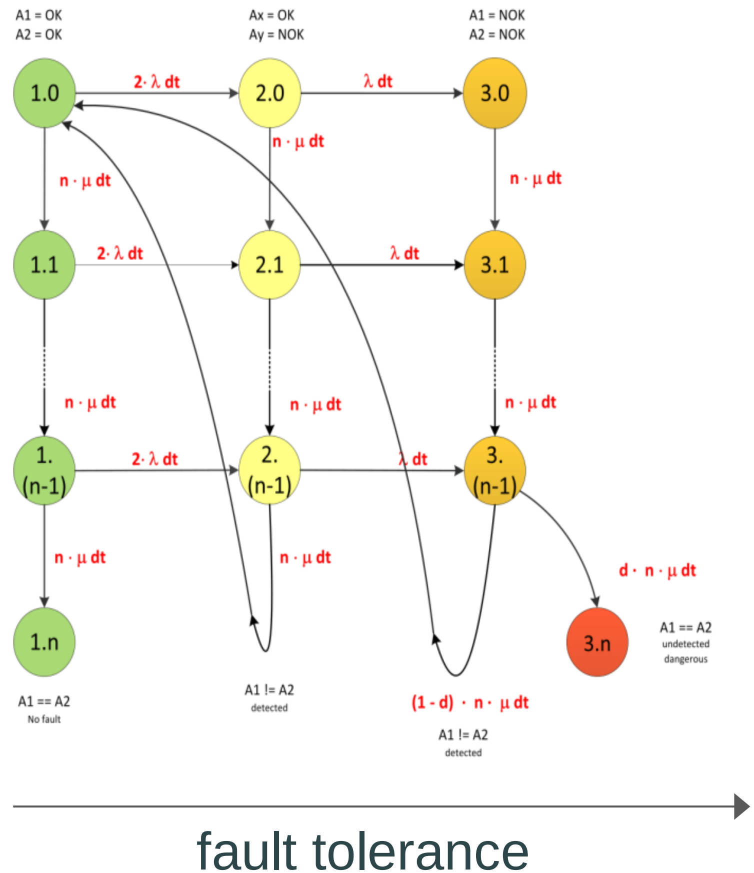
	Execution time [ms]	Period/Deadline [ms]	Fault rate λ [1/ms]
A: Coded	10	100	0.10
B: Redundant	10	100	0.10

Enhanced Markov Model

Coded task processing



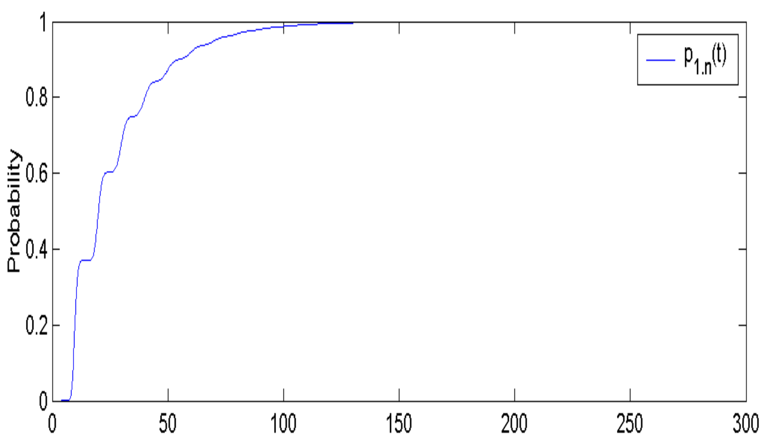
Redundant task processing



Enhanced Markov Model - Results

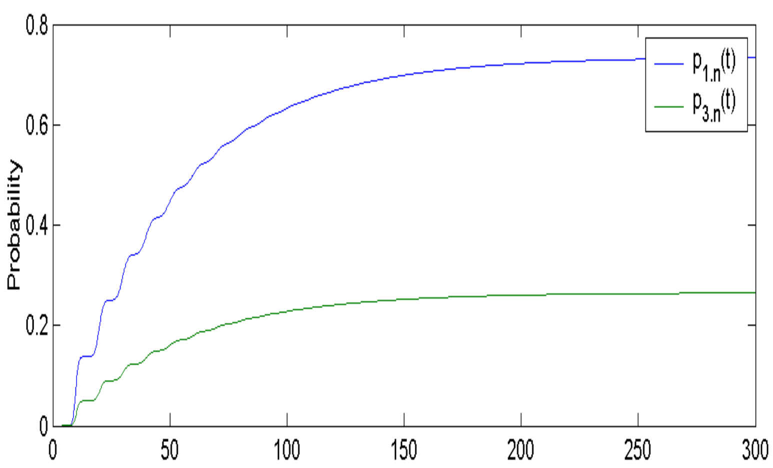
Coded task processing

- expected value of the task execution time: E = 26.81ms
- State Probability:

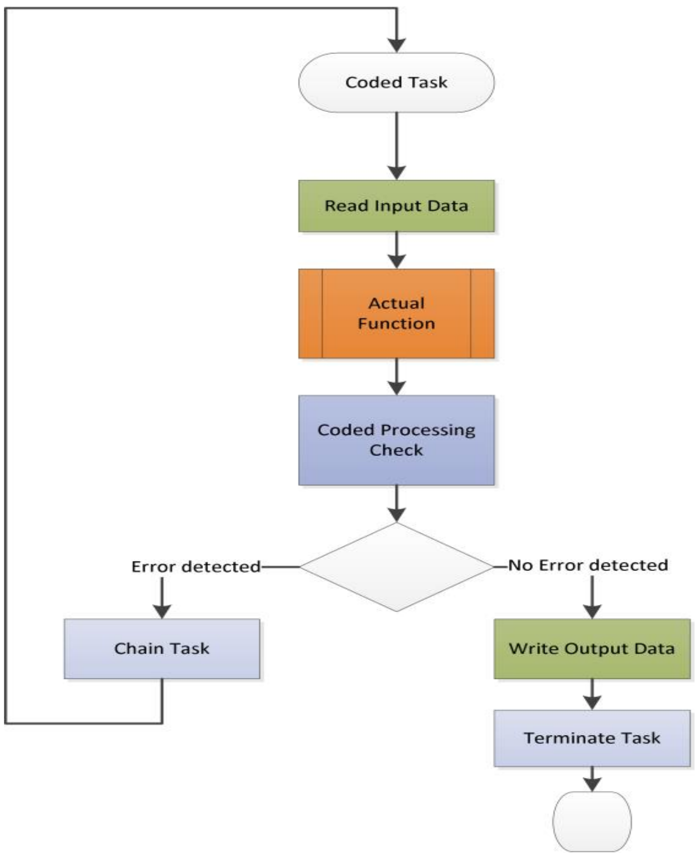


Redundant task processing

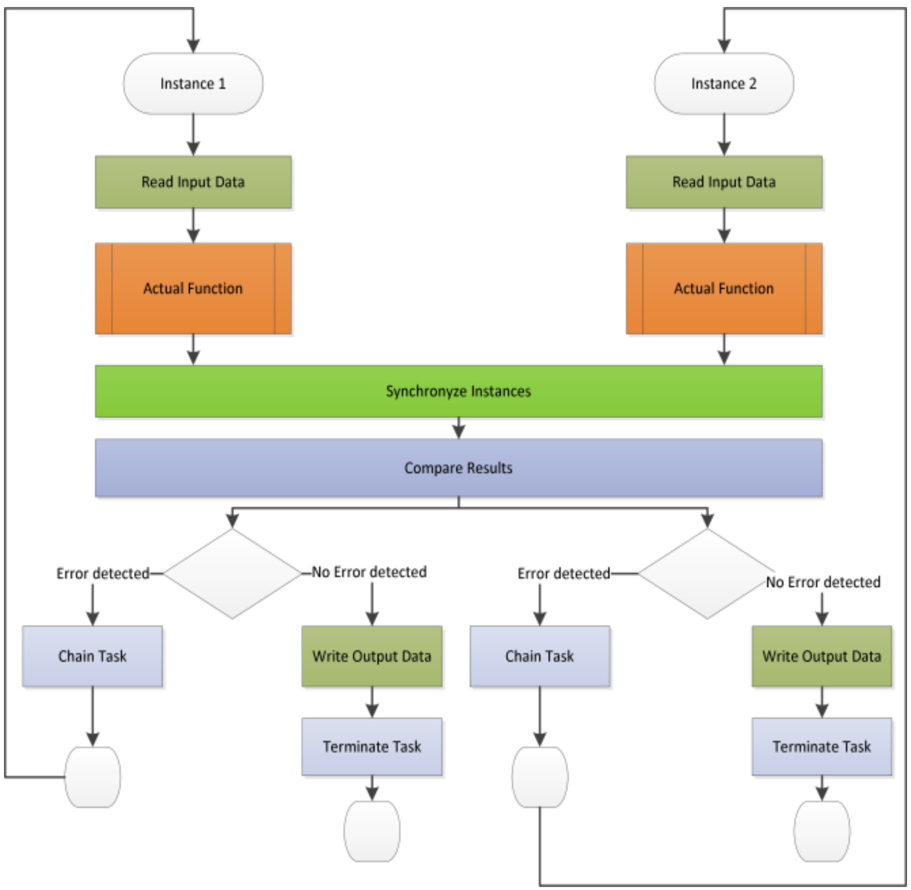
- expected value of the task execution time: E = 52.5ms
- State Probability:



Coded task processing



Redundant task processing



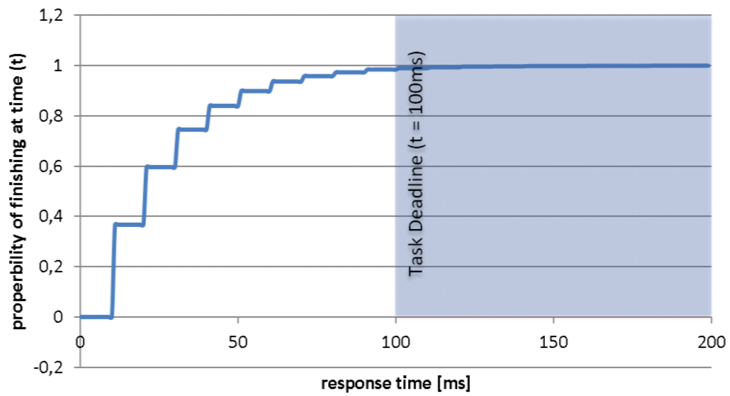
Coded task processing

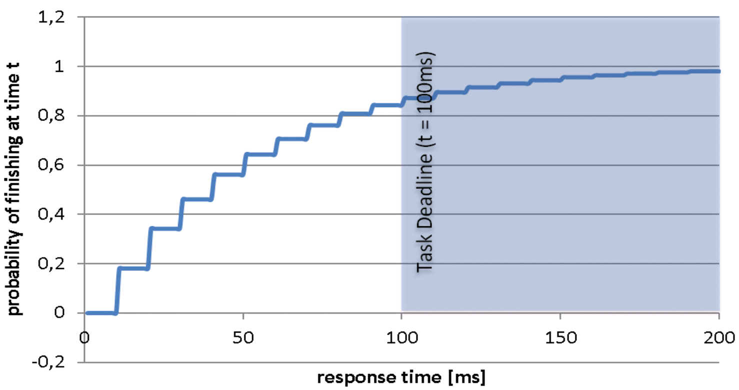
- average value of the task response time:
$$\overline{t_{response}} = 27.28ms$$
- p($t_{response}$ < deadline) = 0.99

Redundant task processing

- average value of the task response time:
$$\overline{t_{response,\,Multiseed}} = 53.95ms$$
- p($t_{response}$ < deadline) = 0.87
- Multi Seed Simulation

	min [ms]	average [ms]	max [ms]
Response time	51.67	53.95	55,82





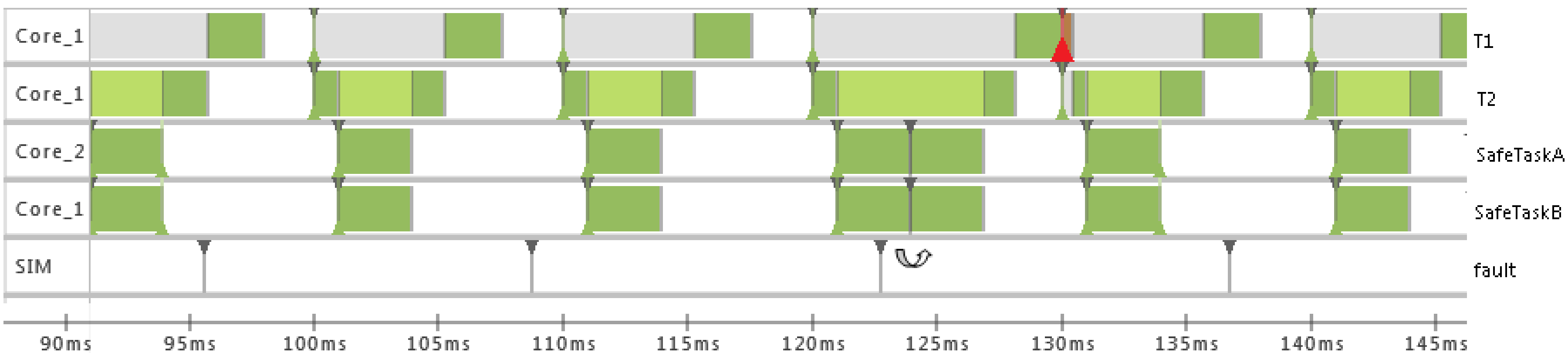
Transition to Automotive Taskset

Additional QM-Tasks

	Execution time [ms]	Period/Deadline [ms]
QM-tasks	2 - 3	10

Results

	Without QM-tasks	With QM-tasks
Saftey-task: response time [ms]	53,95	57,53
Saftey-task: deadline violations [%]	12.8	16.3
QM-task: start to start jitter [ms]	0	0 … 462



Conclusion

- Result of Markov approach for the expected response time within range of the results of the multi seed simulation
- deviation of the simulated results because of consideration of operating system calls (synchronization, scheduling)

- Simulation capable for more complex real-life systems possible
- Discrete event simulation is capable to evaluate safety-critical systems in a holistic timing and reliability view

- Applied scheduling algorithm has to be considered in the whole system analysis which hardly can be achieved by Markov modelling

Thank you for your attention!