artist

ARTIST Summer School in Europe 2010

Autrans (near Grenoble), France

September 5-10, 2010

Formal Performance Analysis and Optimization of Safety-related Embedded Systems

Invited Speaker: Rolf Ernst

TU Braunschweig

SEVENTH FRAMEWORK PROGRAMME

Overview

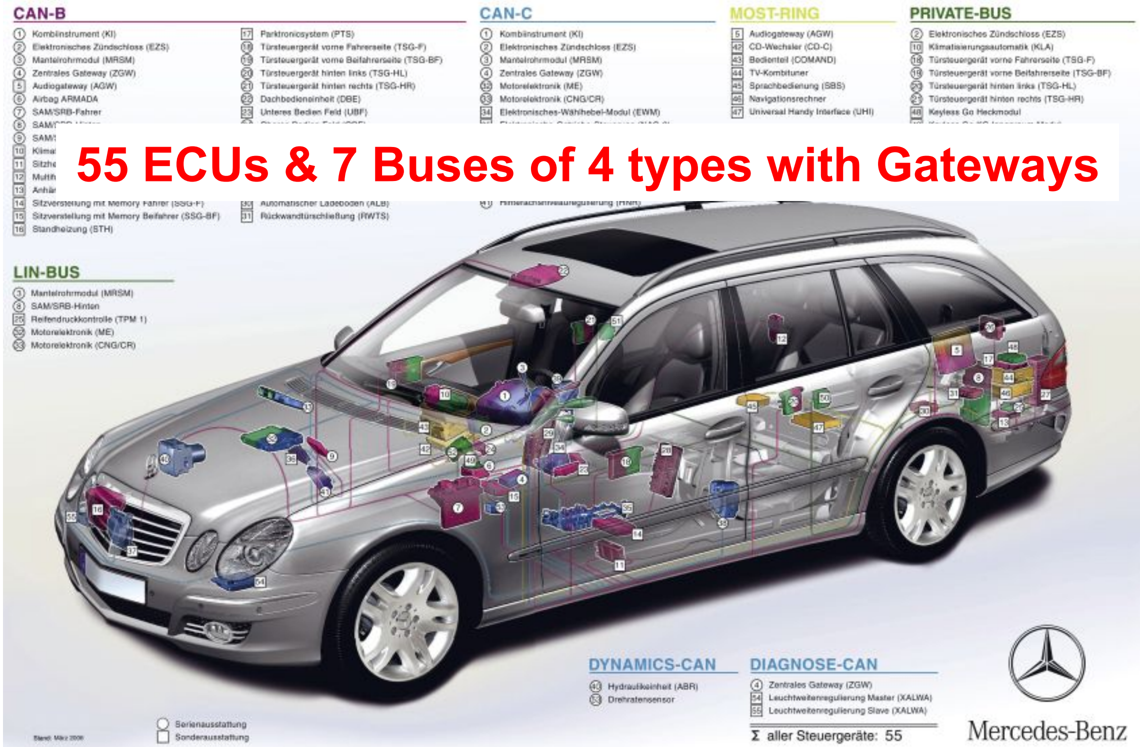
- motivation
- compositional performance analysis
- impact of performance on safety and analysis
  - networks
  - ECUs
- application to safety related system design
- system performance self-protection
- conclusion

Overview

- **motivation**
- compositional performance analysis
- impact of performance on safety and analysis
  - networks
  - ECUs
- application to safety related system design
- system performance self-protection
- conclusion

Motivation

- today's embedded systems use complex networks
  - hundreds of functions
  - thousands of tasks
  - 50+ ECUs
  - networked control
  - many suppliers
  - heterogeneous

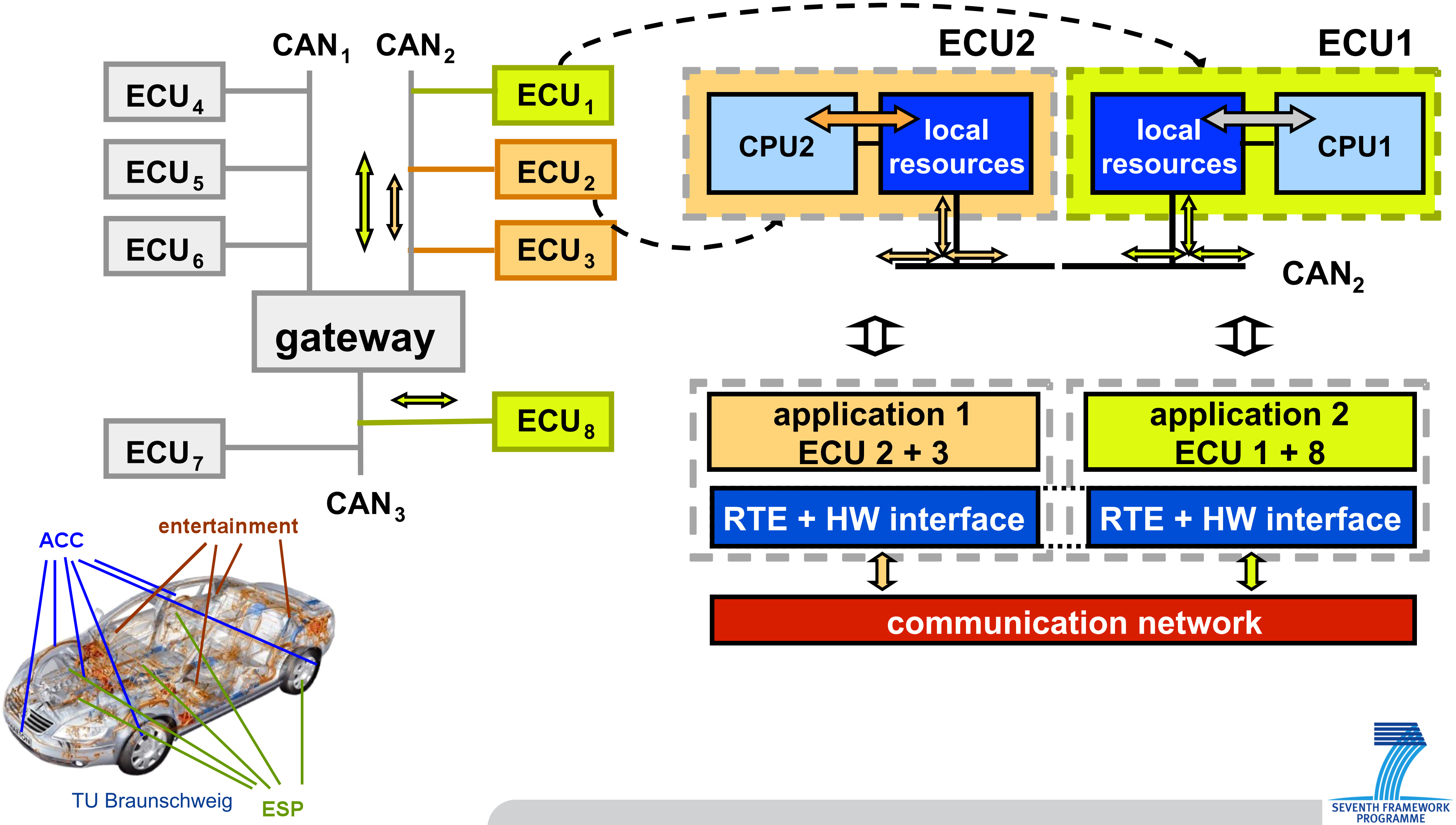


source: Daimler-Chrysler

- networks are an efficient platform for systems integration

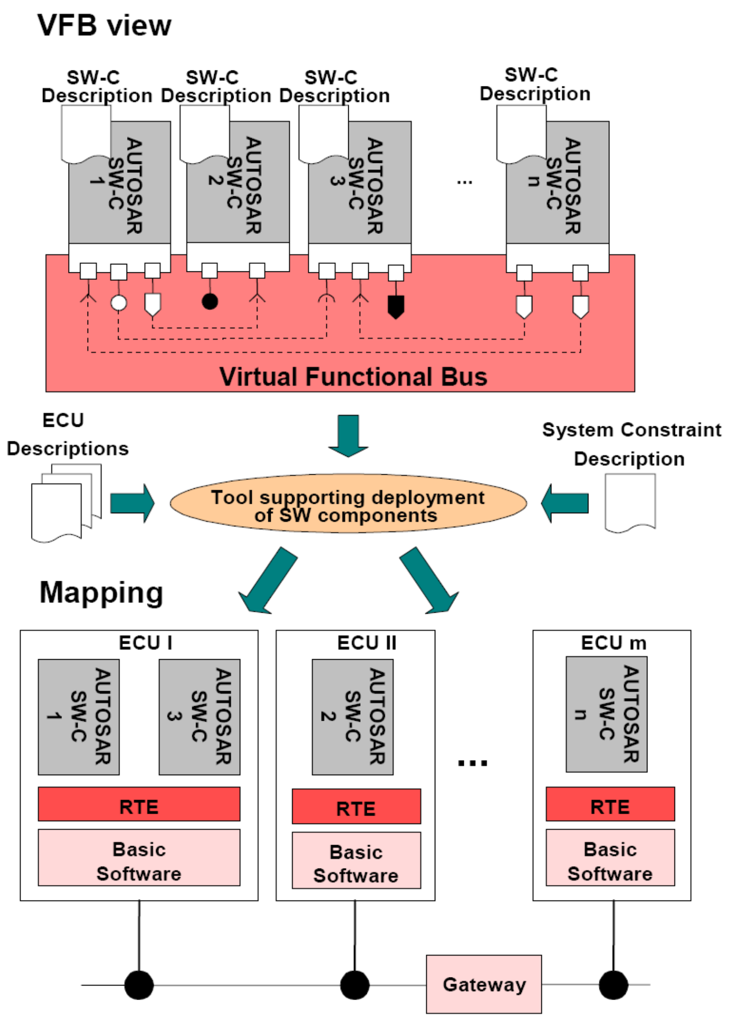
Motivation

- **networks lead to sharing of components and networks for different functions**



Software standardization

- goals
  - reuse and portability of applications
  - system optimization
  - defined interfaces for supply chain with standardized methods and tools
- example AUTOSAR
  - automotive standard software architecture
  - virtual functional bus for integration
  - run time environment (RTE) for specific ECUs



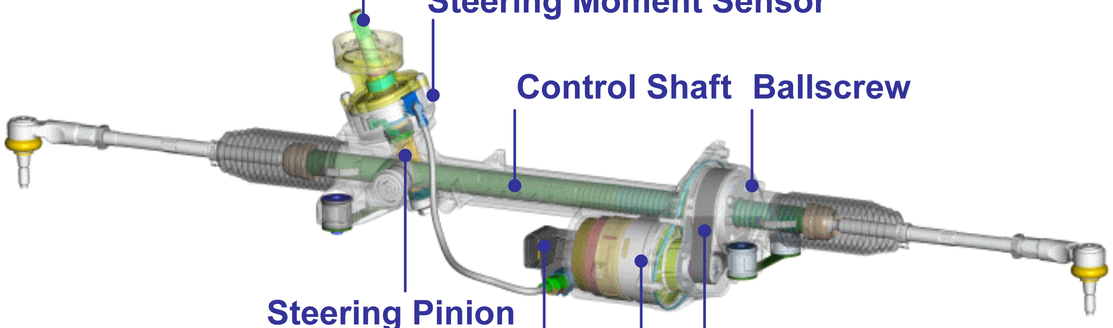
The safety challenge

- embedded systems are increasingly used to
  - implement advanced system features
  - improve safety
- in such cases, the embedded system inherits the safety and dependability requirements of the system function
  - safety related embedded systems
- such functions are no longer simple
- example: automotive electronics
  - electronic steering
  - camera based object recognition and tracking

Example 1: Electronic steering

- standard equipment functions
  - steering power support, speed dependent
  - active centering and dampening
  - straight-running function …
- upgrade equipment functions
  - park assist
  - lane-keeping assist
  - customizable adaptivity - from sportive to an emphasis on comfort

source: Volkswagen

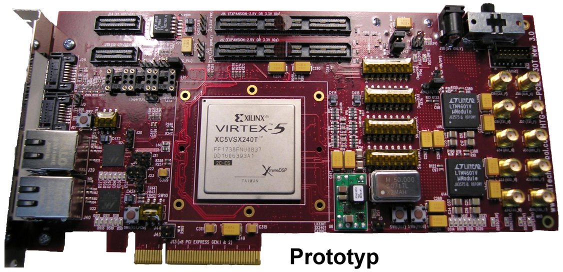


Embedded systems architecture

- two-computer system of the steering control unit
  - steering functions, motor control, and I/O handling are implemented on the main computer
  - the second computer monitors the main computer
  - communication via digital interface
  - exchange of high-frequency question-answer-sequences
  - both computers have an independent clock and energy supply
- classification: fail-safe system function – SIL 3 (more later)

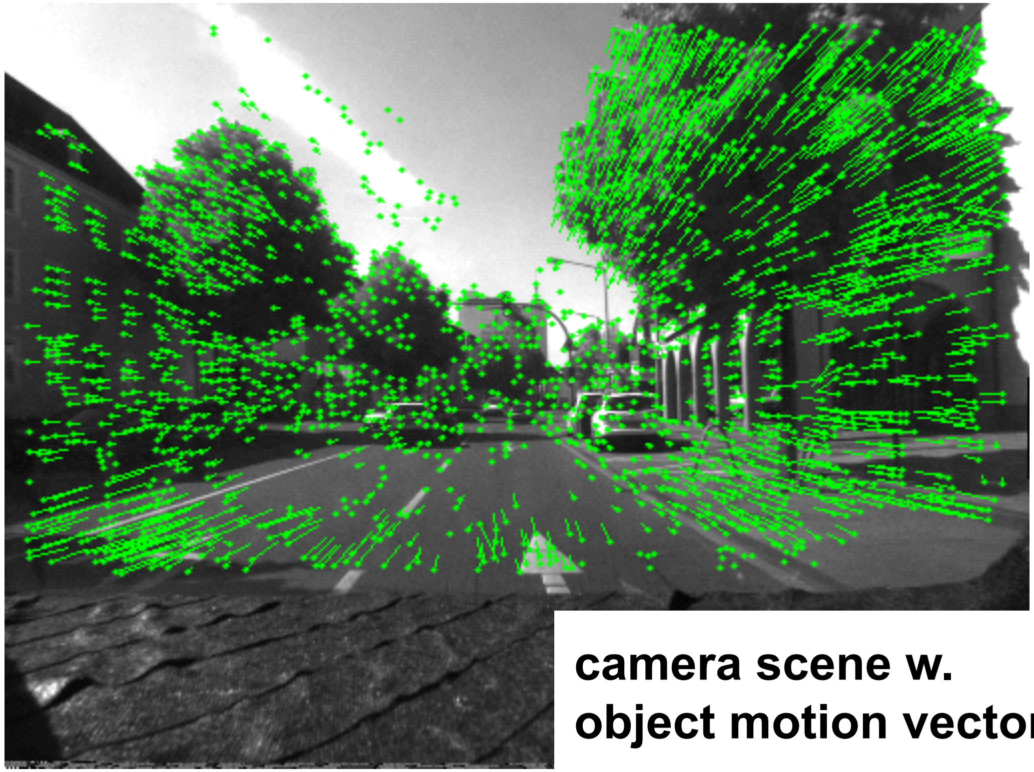
Example 2: Object recognition and tracking

- safety feature (collision avoidance) – SIL3?
- FPGA (or multi-core DSP)
- more than 100 GOp/s (algorithmic)
- power constrained (temperature)

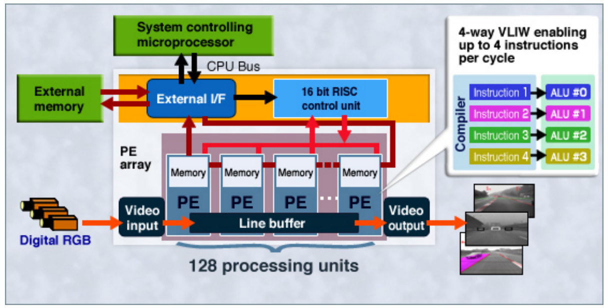


Prototyp
Hardware-Beschleuniger

FPGA prototype
(source: NFF)



camera scene w.
object motion vectors



IMAPCAR DSP
(source: Renesas)

Merging functions with different criticality levels

- integration on one platform leads to mixed (safety) criticality systems
- mutual dependency via platform and sensors/actuators requires safety concept and qualification/certification for all functions
  - data often missing
  - high cost for qualification process
  - significant limitation and costs for updates

→ safety is highly relevant aspect in networked embedded systems design

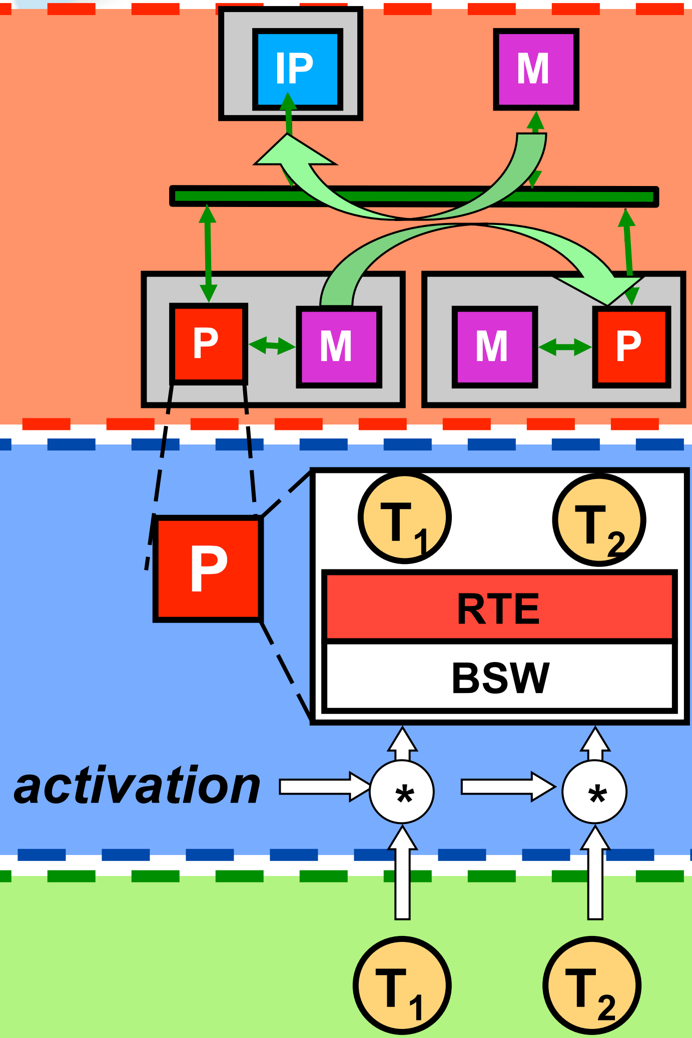
This talk

- what is the role of timing in safety-related systems?
- how to determine timing and performance of networked embedded systems?
- how to derive safety metrics from timing and performance data?
- how to protect a system in case of change?

Overview

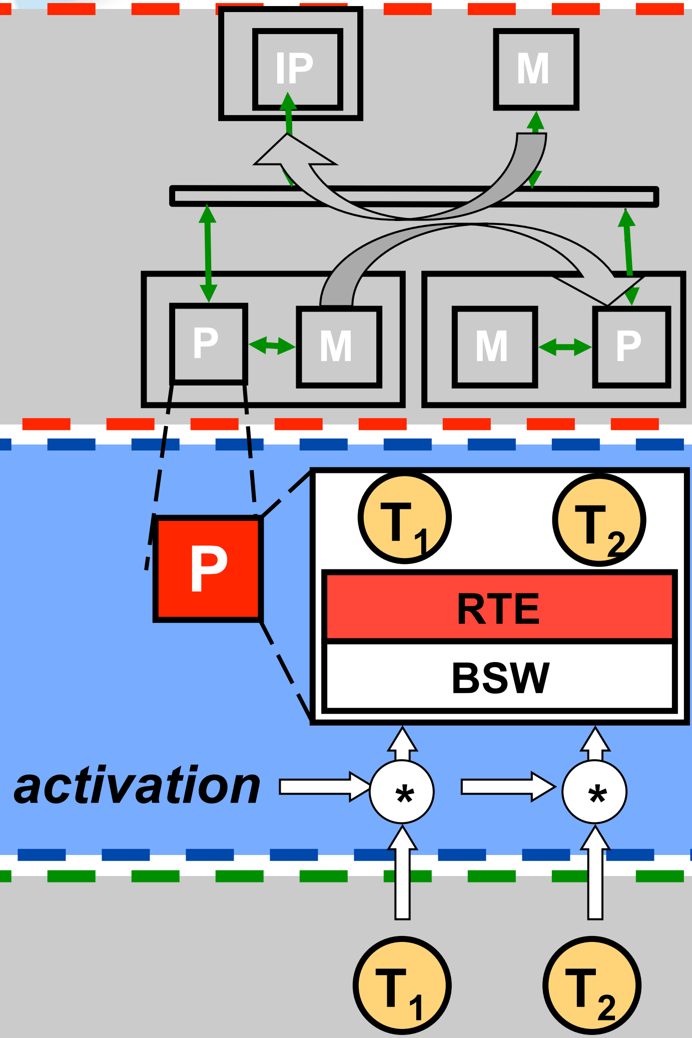
- motivation
- **compositional performance analysis**
- impact of performance on safety and analysis
  - networks
  - ECUs
- application to safety related system design
- system performance self-protection
- conclusion

Performance analysis – 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

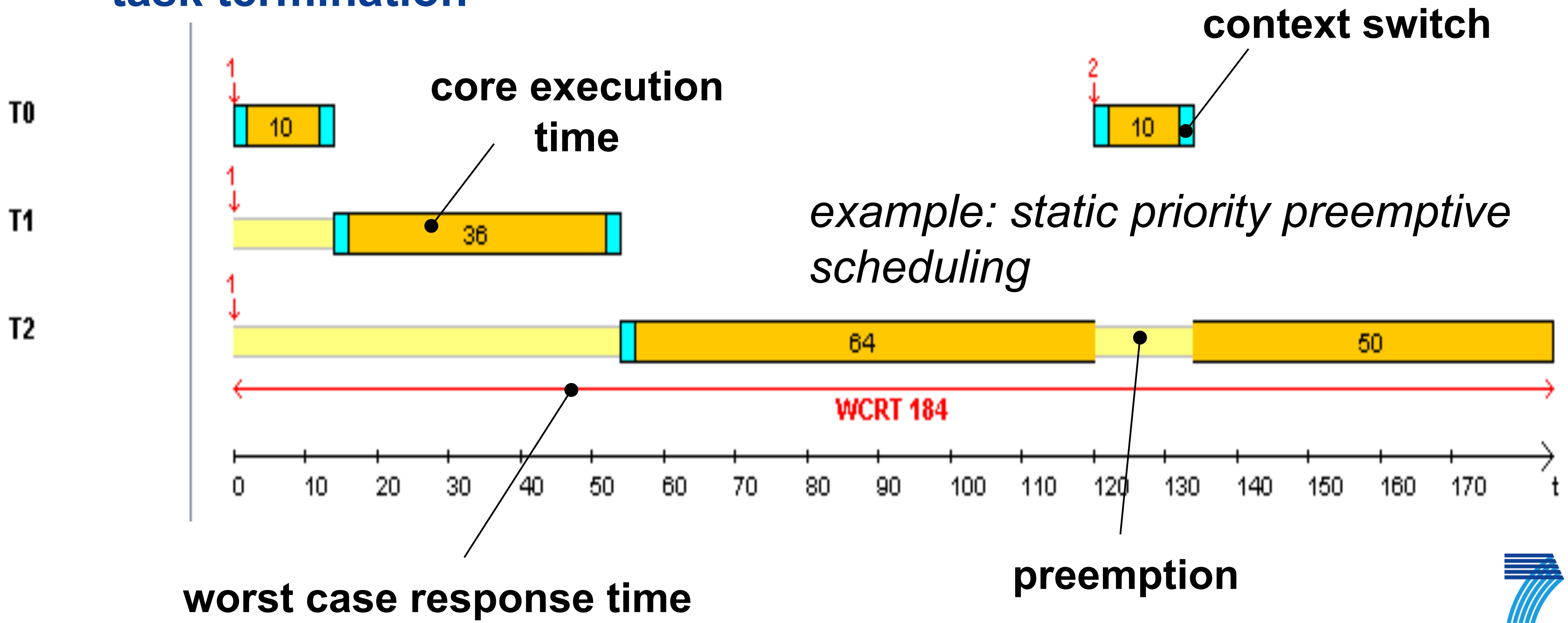
Timing model hierarchy - component timing



- 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 effects of scheduling/arbitration

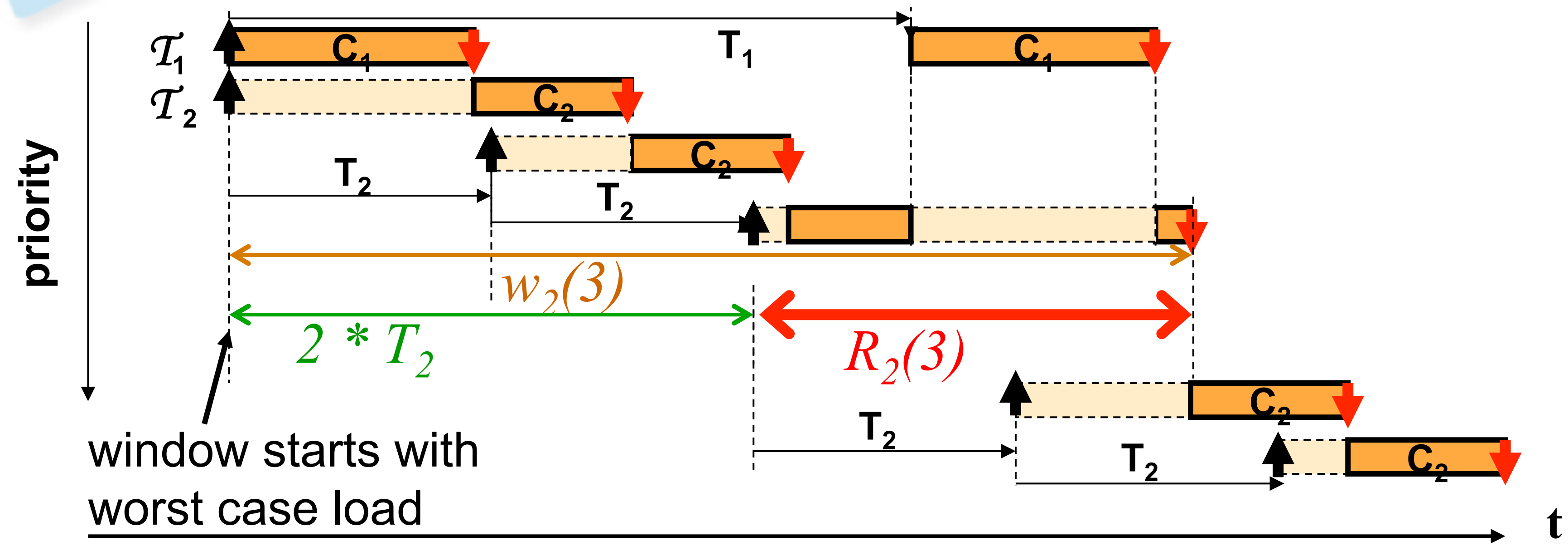
- tasks execute longer than their core execution time
  - time assigned to other tasks
  - operating system overhead
  - context switch, blocking, …
- response time of a task is maximum from time of activation to task termination



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

“Busy window” analysis



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

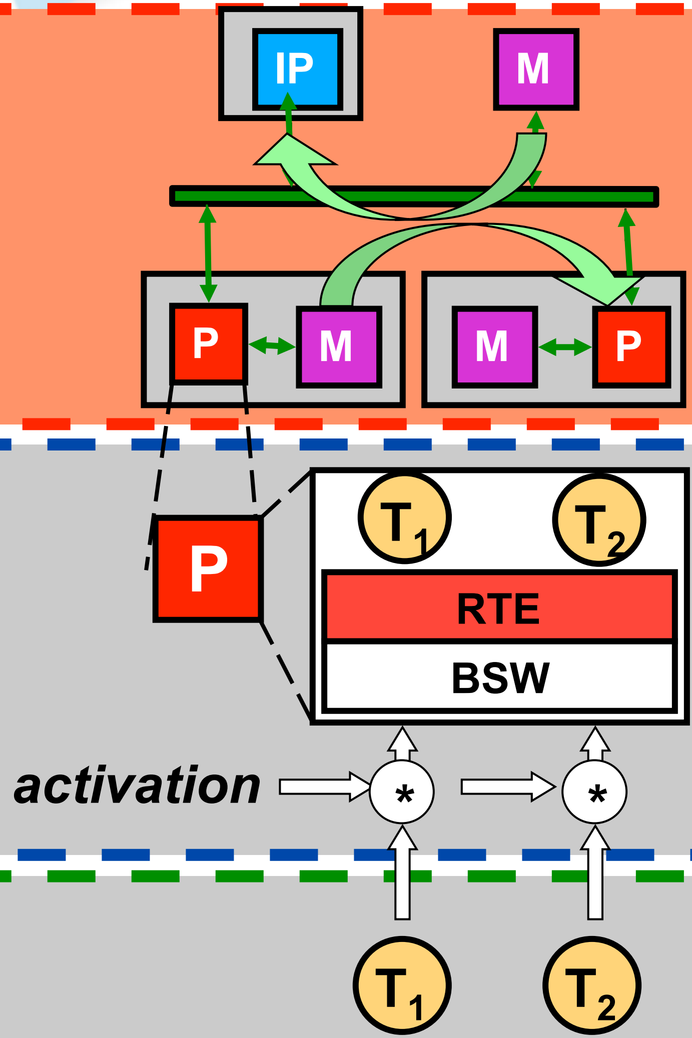
$$R_i(q) = \; w_i(q) \; - \; (q-1)\,T_i$$

increase w_i until fix point found where equations hold!

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 unkown worst case (e.g. release offsets – time table)
- can handle stream queues and register communication
- window size increases with load (limited by deadline)
- window „unrolling“ processes can be considered as symbolic simulation

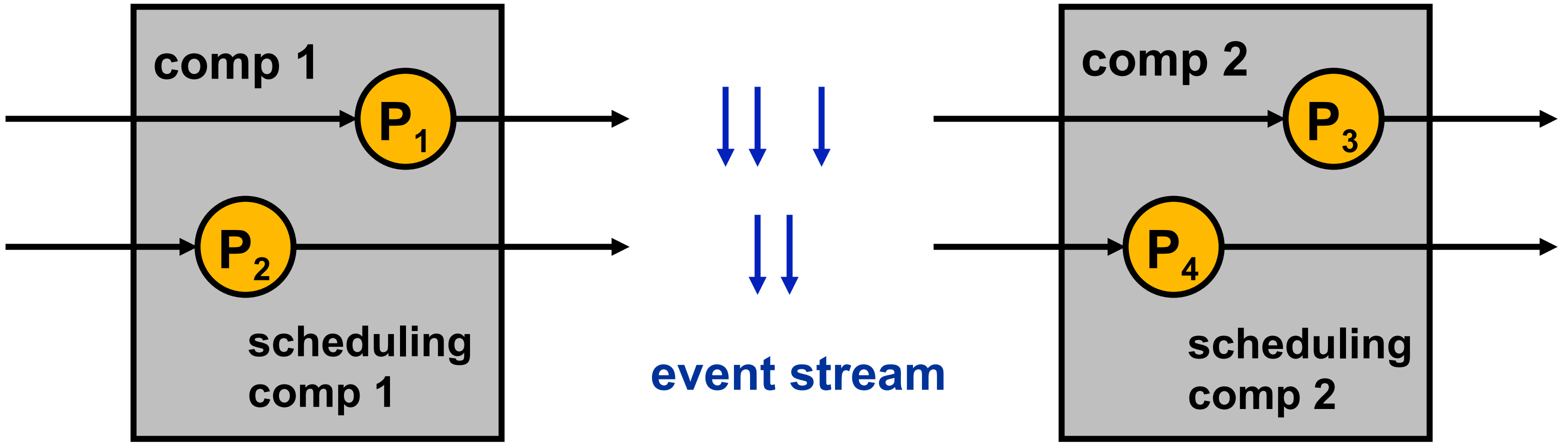
Timing model hierarchy – system timing model



- **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 compositional approach

- independently scheduled subsystems are coupled by data flow

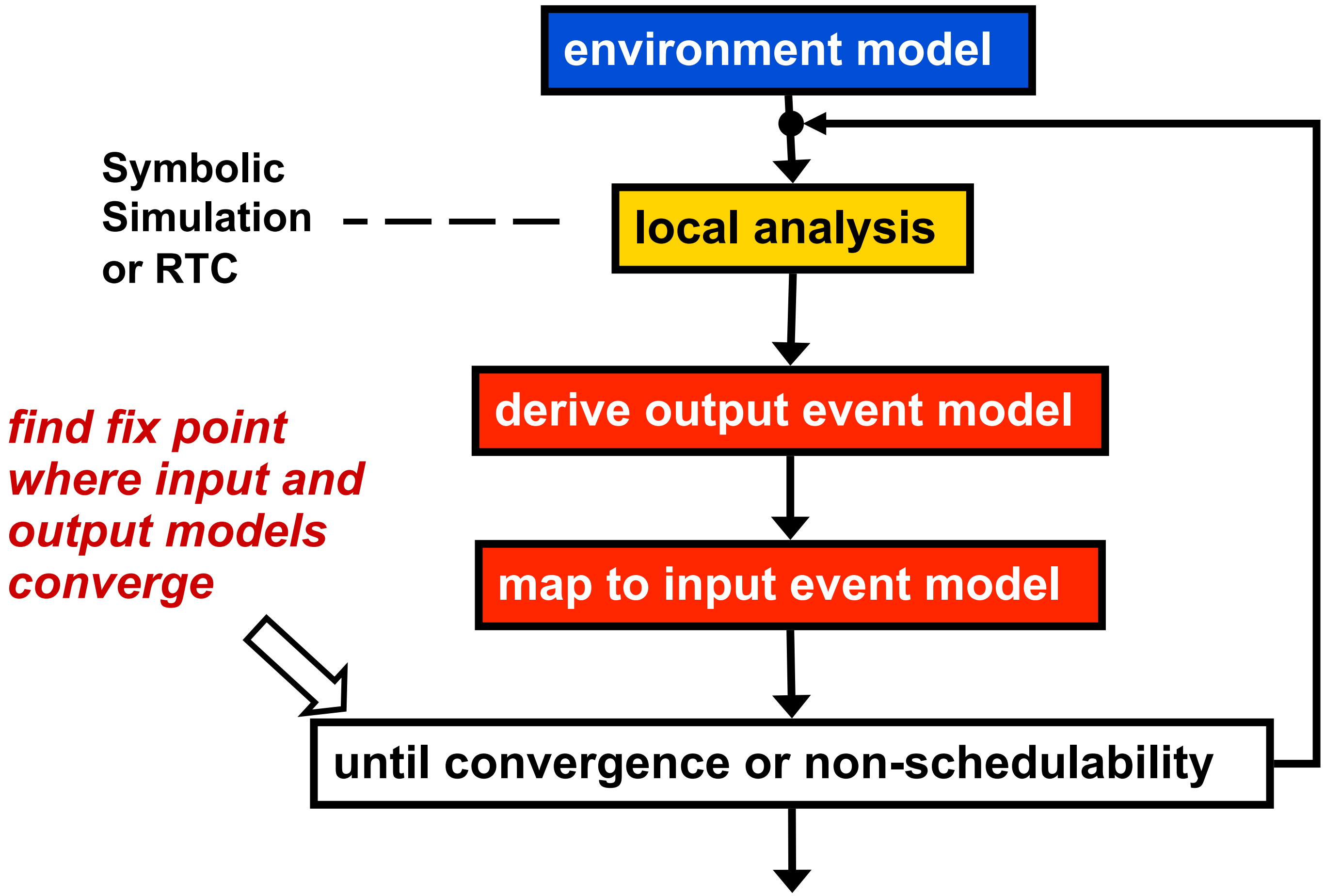


⇒ subsystems coupled by streams of data

⇒ interpreted as activating events

⇒ coupling corresponds to event propagation

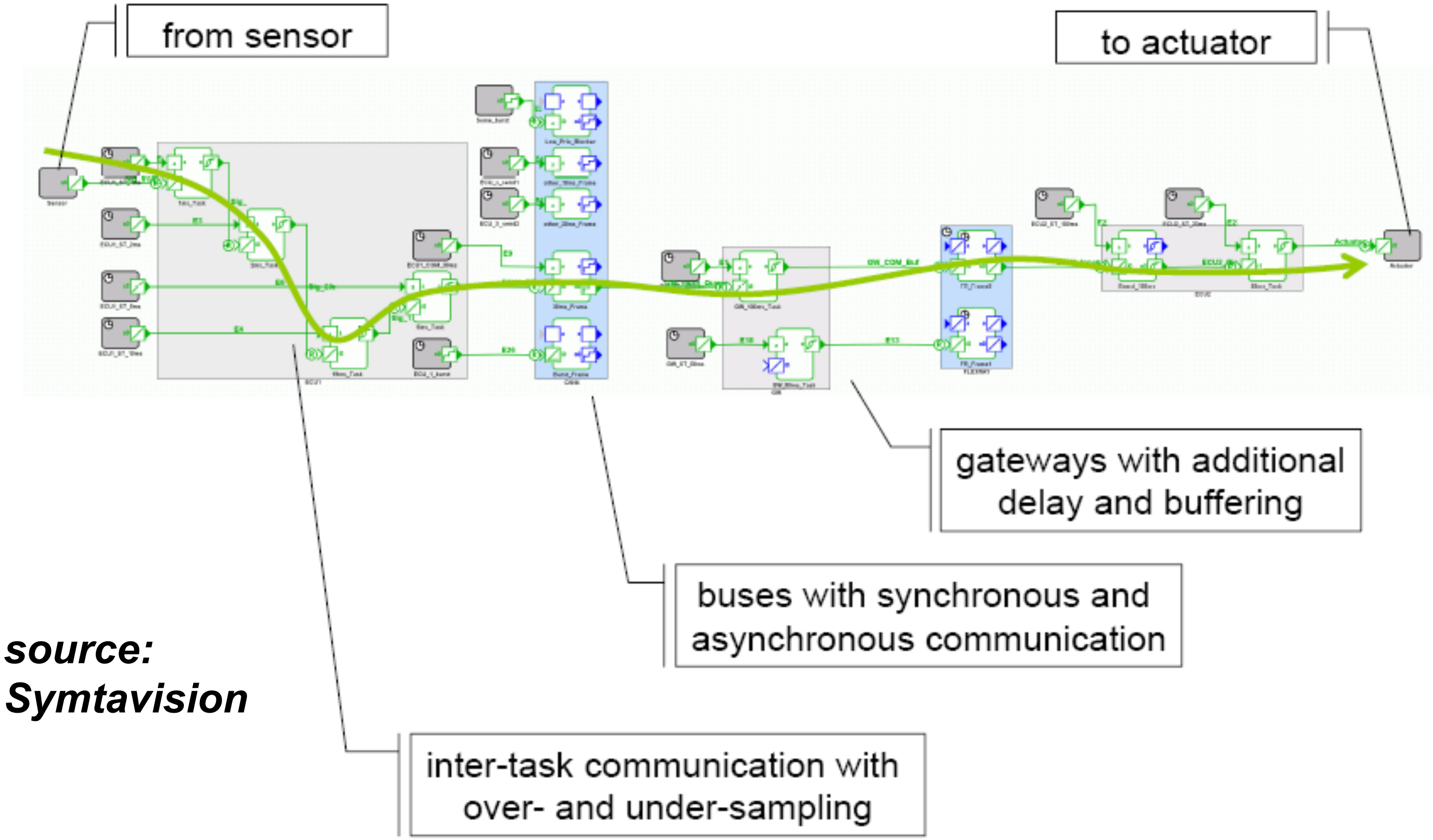
Compositional analysis principle



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 and applications

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

- commercial version of tool SymTA/S used in industrial practice
  - Daimler, Volkswagen, GM, Bosch, Conti, …

SYMTA VISION

Overview

- motivation
- compositional performance analysis
- **impact of performance on safety and analysis**
  - networks
  - ECUs
- application to safety related system design
- system performance self-protection
- conclusion

Technology trends – Reliability issues

- **reliability is an important challenge in future technology generations**
  - **growing system complexity combined with continuous technology downscaling → increasing error rates**
- **appropriate techniques necessary to prevent failures**
  - ***fault isolation***
  - ***error detection and correction***
    - ***bus/network:*** **message retransmission, forward error correction**
    - ***CPU/ECU:*** **redundancy, rollback techniques, microarchitectural measures**
- ***problem:*** **predictability of system reliability**
  - **how does the system behave in case of errors?**
  - **what are consequences for the user / for the environment?**
  - **what is the failure probability?**

Safety standards

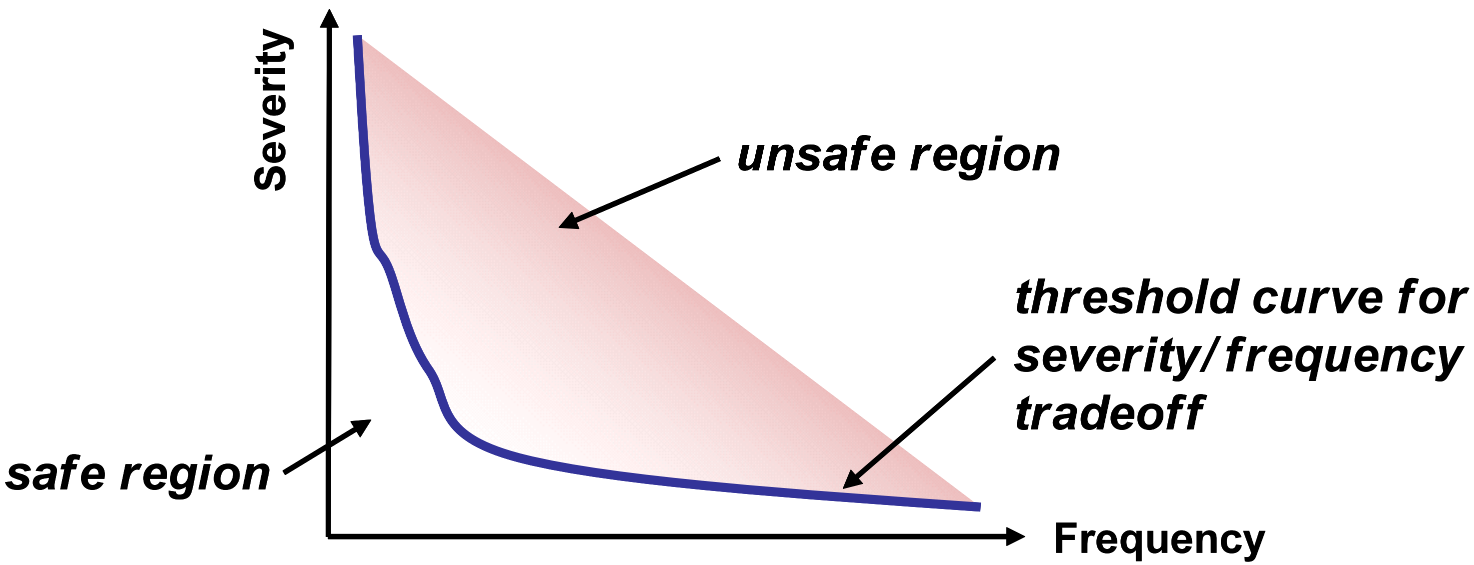
- the design of safety-related systems is driven by safety standards
- safety standards contain
  - rules and regulations for all design system
  - recommended guidelines for the development process
- safety standards cover all stages of the complete development process
  - specification
  - design
  - implementation
  - test
  - maintenance
- *objective of safety related design*
  - avoid unacceptable risk
  - assure functional safety

Functional safety

- **safety*:* Freedom from unacceptable risk of physical injury or of damage to the health of people**
- **functional safety*:* refers to the safety of system functions**
- **risk is characterized by two properties**
  - ***frequency* of hazardous events**
  - ***severity* of hazardous events**

Frequency and severity

- **The idea: *frequency-severity tradeoff***



Functional safety – a short overview

- **safety standards (IEC 61508, ISO 26262) classify systems according to frequency and severity of functional failures**
- **a safe system can handle faults without causing severe functional failures**
- **terminology**

Safety Standards - Overview

- IEC 61508
  - generic standard for safety-related systems
- ISO 26262
  - safety standard for automotive domain
- DO 178B, DO 254
  - safety standards for aerospace domain
- IEC 61511, IEC 62061
  - safety standards for factory automation domain
- EN 50126, EN 50128, EN 50129, EN 50159-1, EN 50159-2
  - safety standards for rail domain

IEC 61508 - Overview

- provides methods to assess the risk of functions
  - based on metrics of severity and frequency of failures
- introduction of safety the lifecycle, which consists of
  - management of functional safety, e.g. enforcement of independent review processes of safety-related components
  - enforcement of verification and evaluation methods to assure functional safety
  - dedicated hardware and software development methods and processes
- further parts of IEC 61508
  - glossary
  - application examples and guidelines

Example: IEC 61508

- reference standard that is used to derive other standards (e.g. ISO26262)
- metric: "Safety Integrity Level" – SIL
  - defines four degrees of safety: from 1 (lowest) to 4 (highest)
  - specification of maximum failure rates for each level

SIL	*Low demand mode: average probability of failure on demand*	*High demand or continuous mode: probability of dangerous failures per hour*
1	$> 10^{-2}$ to $< 10^{-1}$	$> 10^{-6}$ to $< 10^{-5}$
2	$> 10^{-3}$ to $< 10^{-2}$	$> 10^{-7}$ to $< 10^{-6}$
3	$> 10^{-4}$ to $< 10^{-3}$	$> 10^{-8}$ to $< 10^{-7}$
4	$> 10^{-5}$ to $< 10^{-4}$	$> 10^{-9}$ to $< 10^{-8}$

Reliability Analysis and Functional Safety: IEC 61508

- *basic principle:* apply reliability analysis to verify that safety requirements are satisfied
  - *assumption:* required safety level is known a priori → hazard analysis and risk assessment not considered
- IEC 61508 does not directly support mixed criticality systems

  "An E/E/PE safety-related system will usually implement more than one safety function. If the safety integrity requirements for these safety functions differ, unless there is sufficient independence of implementation between them, the requirements applicable to the highest relevant safety integrity level shall apply to the entire E/E/PE safety-related system."

- reliability analysis can help to close this gap!
  - *more later*

Functional Safety – ISO 26262

- **ISO 26262 basically similar to IEC 61508**
  - **includes risk classification**
  - **defines development processes and method for saftey-critical automotive system**
    - **FMEA (failure mode and effect analysis), FTA (fault tree analysis)**
- **ISO 26262 defines ASIL 1-4 (automotive SIL) analogous to IEC 61508 SIL**
- **includes risk analysis and ASIL assessment process according to parameters severity, exposure and controllability**
  - **risk as a function of frequency f and severity S: R = F (f, S)**
  - **frequency as a function of exposure E and controllability C: f = E x C**

Functional Safety – Risk Assessment Matrix

		C1	C2	C3
S1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	A
	E4	QM	A	B
S2	E1	QM	QM	QM
	E2	QM	QM	A
	E3	QM	A	B
	E4	A	B	C
S3	E1	QM	QM	A
	E2	QM	A	B
	E3	A	B	C
	E4	B	C	D

note: **the class QM (Quality Management) denotes "no requirement" according to ISO 26262**

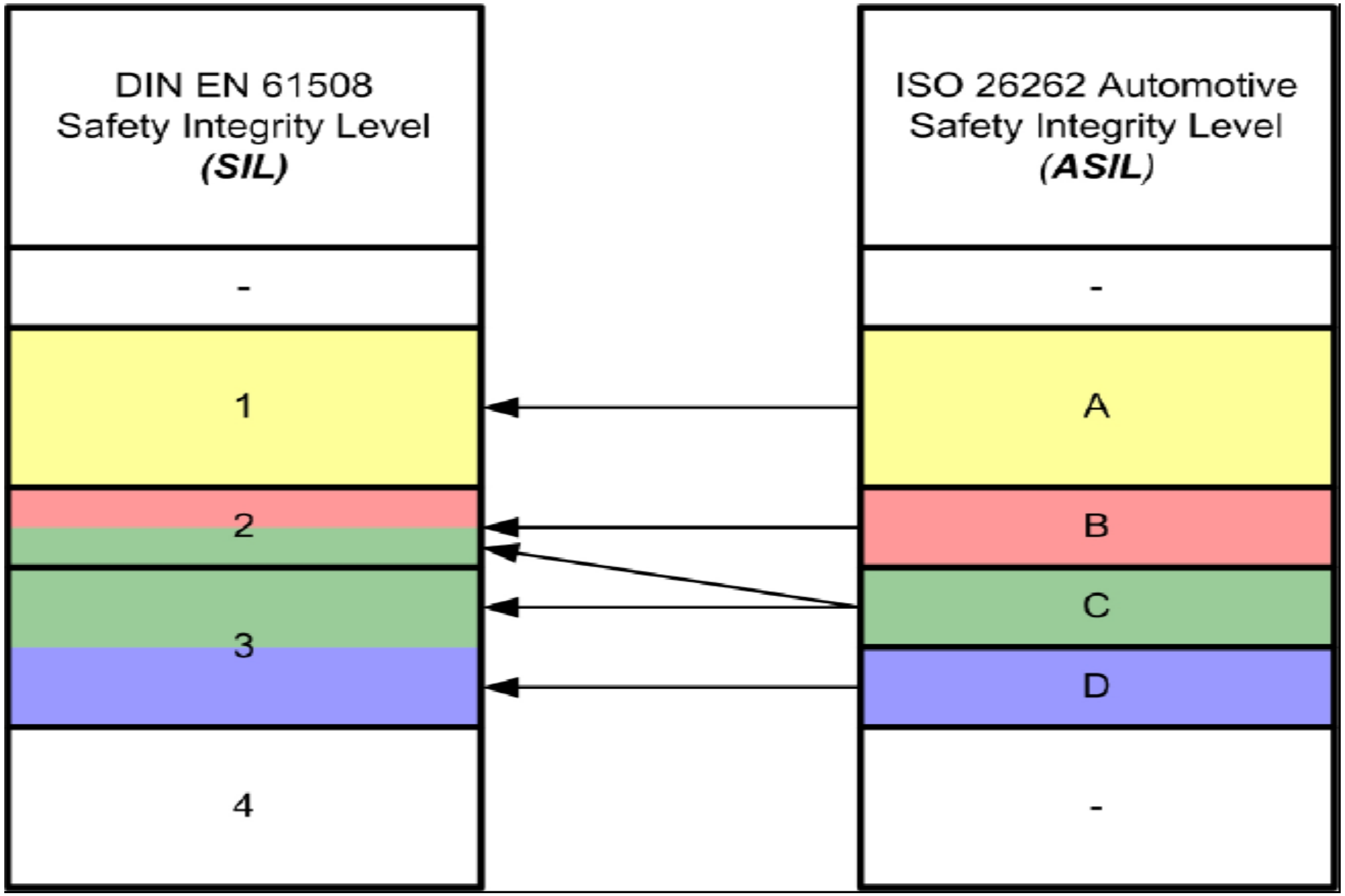
Functional Safety – ISO 26262

- **gap to IEC 61508: ISO 26262 provides no formal failure rate specification such as 61508**
- **however: approximate mapping is possible based on the term of "observable incident rate" introduced in ISO 26262**

ASIL	Observable incident rate
D	$<10^{-8}/h$
C	$<10^{-7}/h$
B	$<10^{-7}/h$
A	$<10^{-6}/h$

- **the observable incident rate is based on relevant field data**
- **basically observable incident rate is used for the proven in use argument**
- ***"Proven in use argument is an alternate means of compliance with ISO26262 requirements that may be used in case of reuse of existing items or elements when field data is available."***

Result: SIL – ASIL Mapping



Embedded systems functional failures

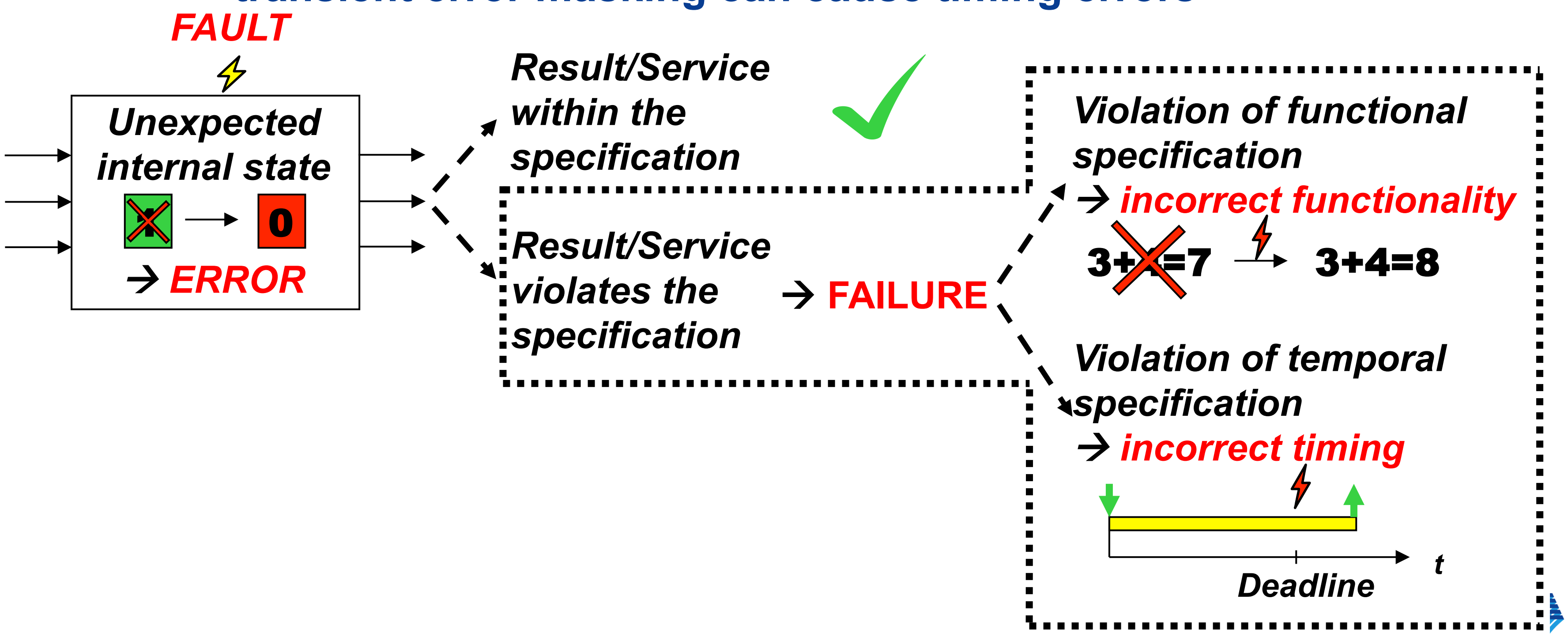
- embedded system (ES) functional failures are not necessarily catastrophic
- effect depends on the importance of the failing function for the overall system
  - function criticality
- depends on the overall system functionality
  - fail safe:
    if the ES function fails there is a safe function backup or a safe system state that avoids severe consequences
    (mechanical steering, hydraulic brake, emergency stop)
    - ES is not critical but important for quality
  - fail operational (fault tolerant):
    the function continues based on system redundancy or turns to an error mode with reduced functionality (graceful degradation)
    - ES function is critical, but possibly only needs a specific function

Embedded system functional failures and timing

- ES functions have different criticality
  - depending on the overall system
- where timing is specified, it becomes part of the function criticality
  - ES timing failures are ES functional failures
- switching to error modes is time critical
  - switching needs hard deadlines to guarantee overall system function

From ES faults to ES failures

- **distinguish static and transient ES errors**
  - **static errors have permanent effects requiring redundancy for repair**
  - **transient errors are more frequent (EMC, …) but can often be masked when detected**
    - **transient error masking can cause timing errors**



Overview

- motivation
- compositional performance analysis
- impact of performance on safety and analysis
  - networks
  - ECUs
- application to safety related system design
- system performance self-protection
- conclusion

Functional safety of CAN networks

- functions of different criticality meet on the automotive network
- use CAN bus as an example
- CAN has error detection capabilities (CRC)
  - repeats message in case of transmission error using defined protocol
  - *CAN functional fault tolerance increases timing and load!*

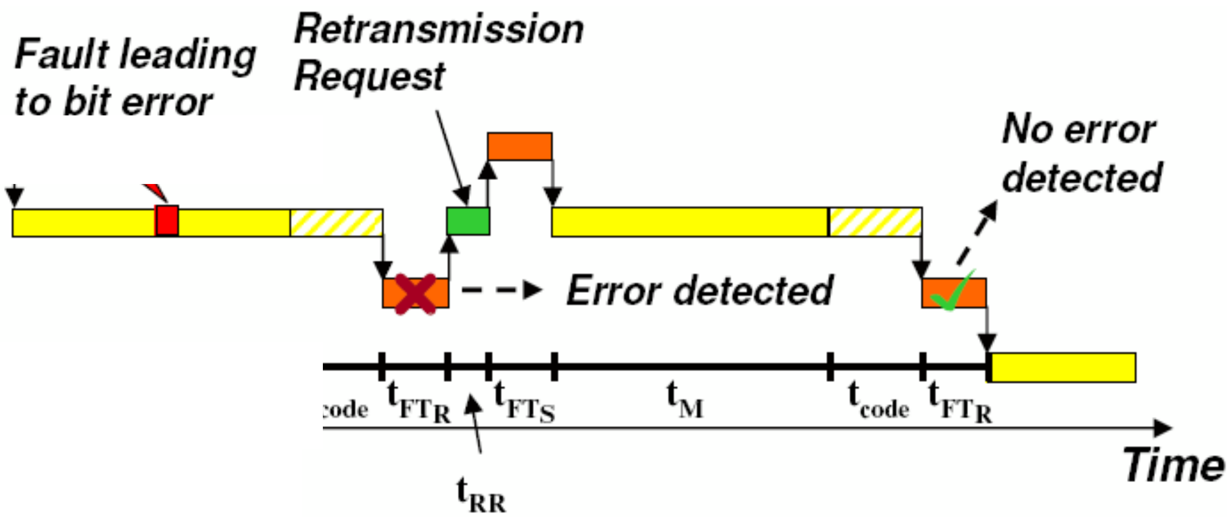
ECU1

fault detection (HW/SW)

bus

fault detection (HW/SW)

ECU2

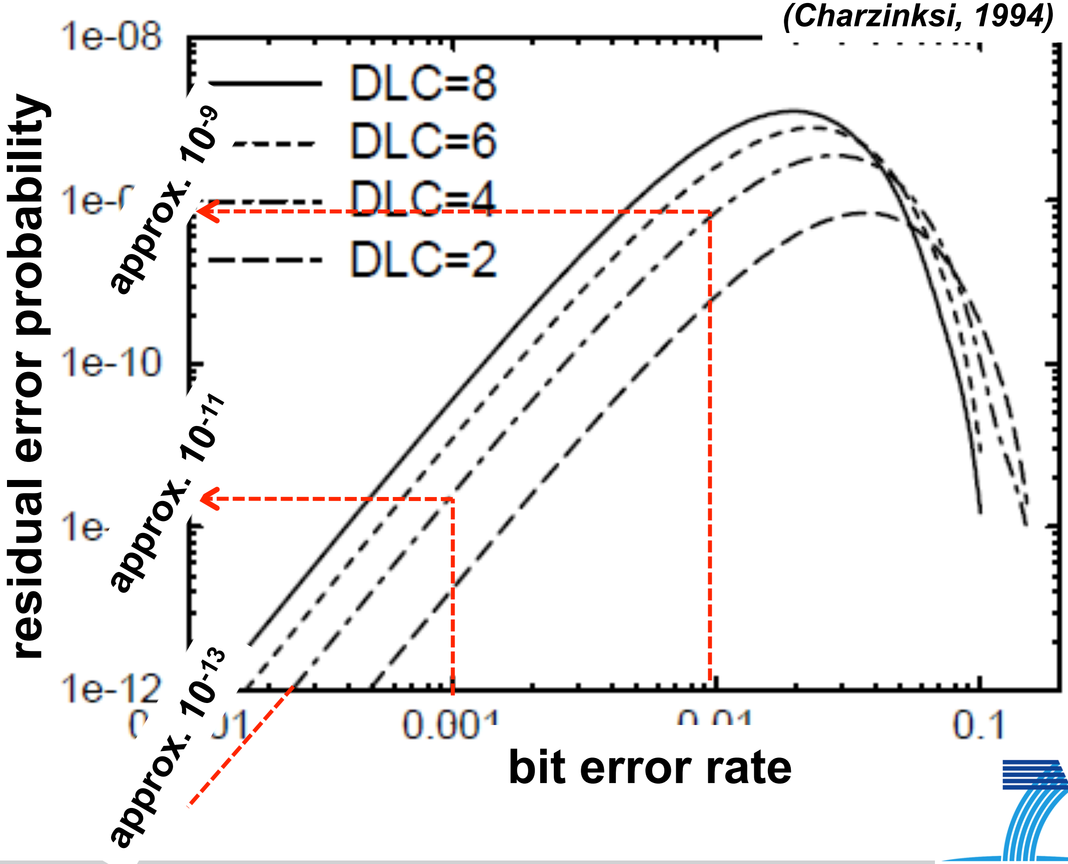


SEVENTH FRAMEWORK PROGRAMME

Timing errors and residual errors

- **undetected transmission errors lead to ES function failure - *residual errors***
- **detected and corrected errors might lead to timing errors**
  - ***what is practically more frequent and important?***
- **study on residual error probability**
  - **functional CAN failures for different data length code (DLC)**
  - **study assumes extremely high bit error rates > 10^{-4} single error, uniform dist**
  - **practical bit error rates are ≤ 10^{-6}**

(Charzinksi, 1994)



ES timing vs. functional failures - Comparison

- *assumption 1:* SAE benchmark frame set
  - approximately 5 x 10^6 activations per hour
  - approximation of the mean time to functional failure *$MTTF_{func}$* based on Charzinski's results (DLC = 4)
- *assumption 2:* bus load of approx. 70 % (CAN at 150 kbit/s)
  - calculation of the mean time to timing failure *$MTTF_{time}$* based on SymTA/S reliability analysis, same error model as Charzinski
  - deadline end of period

Bit error rate:	0,01	0,001	0,0001	
$MTTF_{func}$	*2 x 10^2 h*	*2 x 10^4 h*	*2 x 10^6 h*	SIL 1
$MTTF_{time}$	*< 1 s*	*1,8 min*	*1,4 x 10^3 h*	no SIL

- *NOTE:* data functional failure probability only available for very high bit error rates

Comparison with practical error rates

- study on realistic bit error rates (BER) by Ferreira, 2004
  - measured for aggressive environments: BER = 10^{-7}
- only studies available for higher residual error probability
  - approach: extrapolate residual error probability for BER = 10^{-7}
  - assume linear relation between residual error probability and BER (valid assumption for low BER)
    - BER = 10^{-7} → residual error probability ≈ 10^{-19}
  - corresponding $MTTF_{func}$ satisfies all SIL requirements

	Bit error rate = 10^{-7}	
$MTTF_{func}$	2×10^{12} h	SIL 4
$MTTF_{time}$	$1{,}8 \times 10^{5}$ h	SIL 1

7 orders of magnitude more likely to miss end-of-period deadline!
(for single error fault model)

SEVENTH FRAMEWORK PROGRAMME

Analysis background

- SymTA/S extension for timing failure analysis
- analysis goal
  - given a practically relevant fault model
  - determine the resulting frequency of embedded system errors
    - include all side effects and mutual dependencies of different transmissions
  - classify error probability of each logic channel individually to allow for efficient mixed critical systems
- potential approaches
  - simulative, based on random event generation → Monte Carlo simulation
    - very flexible but prohibitively time consuming for realistic systems
      used for tool validation
  - analytical, based on probability calculus → formal analysis
    - new technology needed

Formal timing failure analysis - principle

- treat faults as exceptions from worst case behavior rather than include it
  - fault statistics lead to a generally unbounded timing
- use compositional analysis
  - failure analysis can focus on single components
  - derivation of system level impact as separate concern
- apply to SymTA/S

Formal analysis approaches – related work

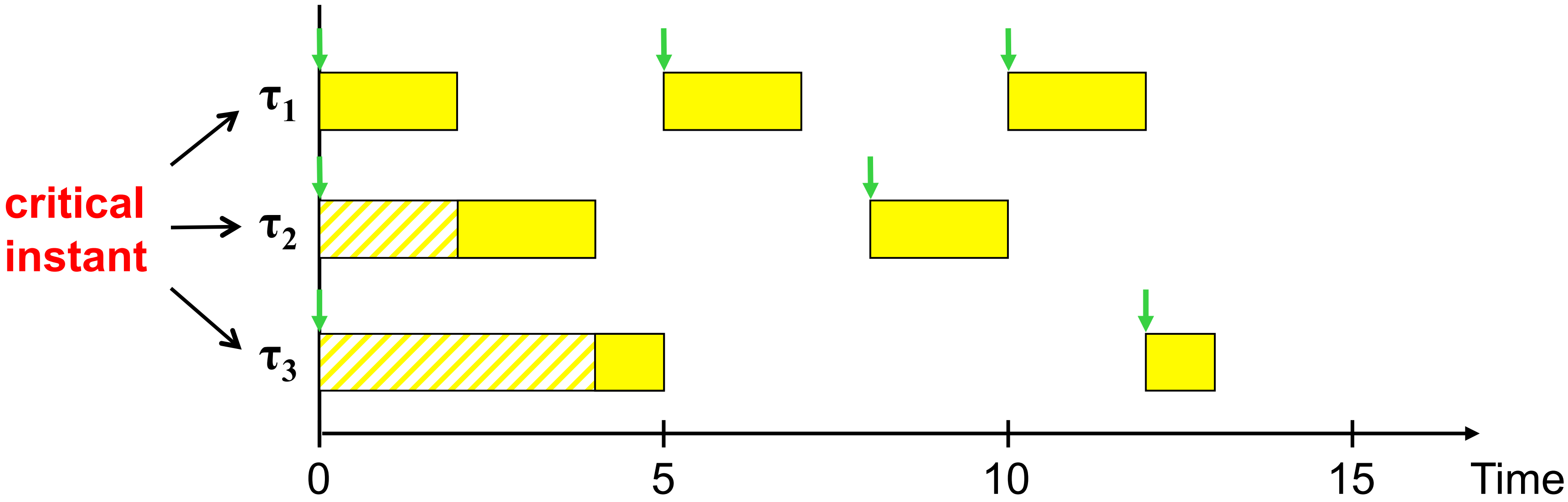
- Burns et al. (1999)
  - computation of deadline failure probability (DFP) for computational components
  - considering periodic task systems
  - assumption: worst-case situation in static-priority based environments
    - (1) critical instant (2) max error penalty of all hpe tasks
- Broster et al. (2002)
  - extension of Burns' approach: probability distributions of task response times on a CAN bus
  - assuming the same worst-case, but closer probability assumption
- Izosimov et al. (2005)
  - MPSoCs with static execution order scheduling only
  - computation and optimization of number of tolerable errors
  - statistical considerations added in 2009

Computation of DFP (Burns, 1999) – some insights

- two step algorithm
  - compute the number of tolerable error per task
  - compute the probability that this number is exceeded
    → deadline miss probability (DFP)
- assumptions
  - periodically activated tasks with constant workload per activation
  - static priority based scheduling policy supporting task preemptions
  - deadline = period → task execution has to finish before it is restarted
  - tasks are activated simultaneously → critical instant (CI)
  - error penalty: workload maximum over all tasks

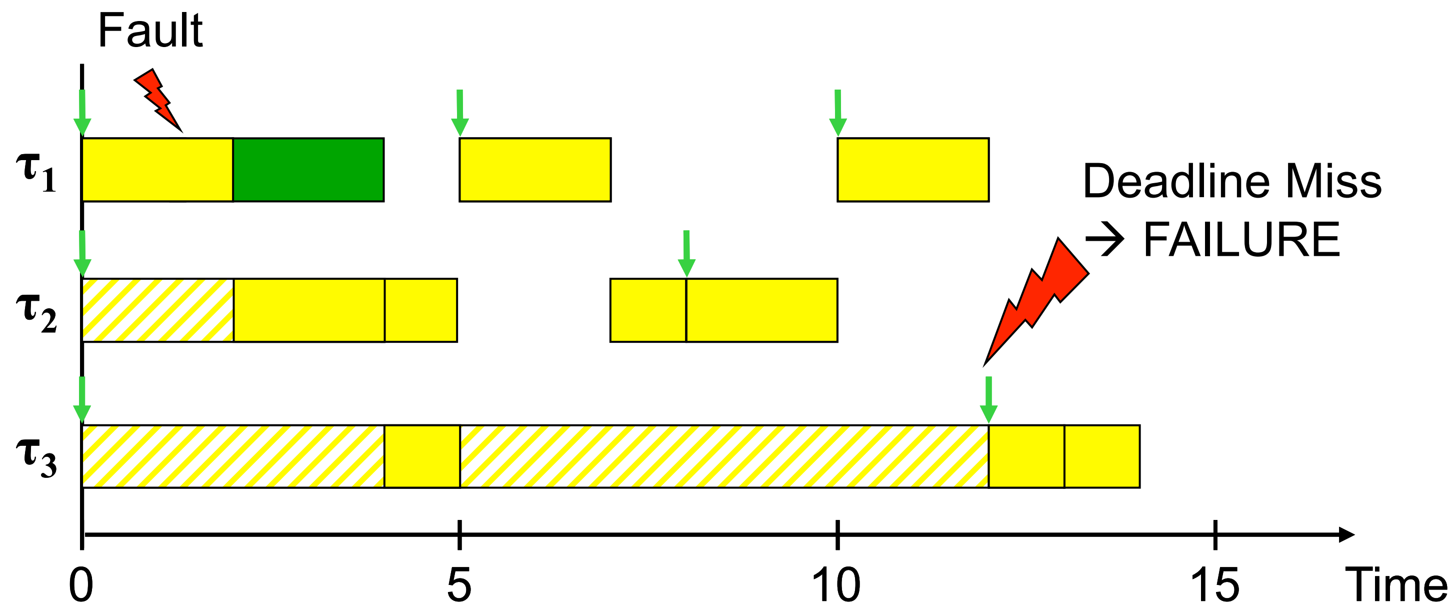
Computation of DFP (Burns) – example

Task	*activation period*	*workload*	*priority*
τ_1	5	2	1
τ_2	8	2	2
τ_3	12	1	3

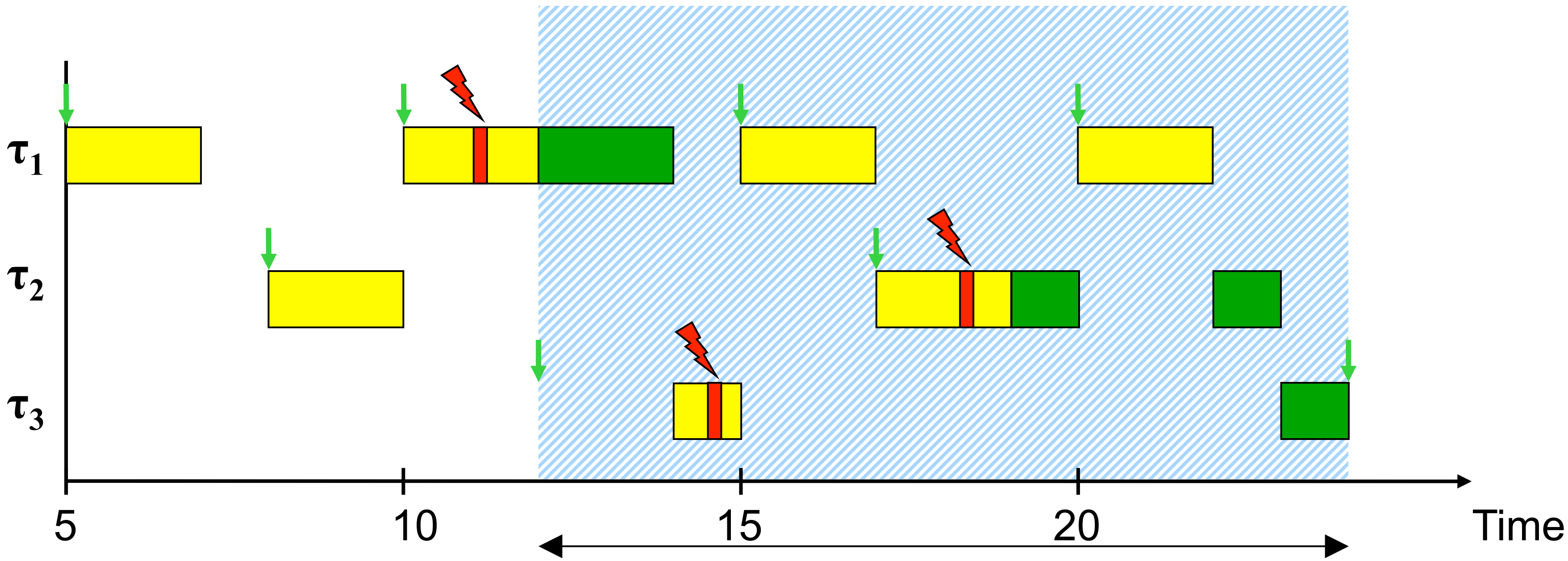


Computation of DFP (Burns) – Principle

- computing the number of tolerable errors is based on the critical instant → each task activated at time 0
- maximum error penalty assumption
  - error either in τ_1 or τ_2
  - result of analysis: *no tolerable error at all for τ_3*



Computation of DFP (Burns) – Example



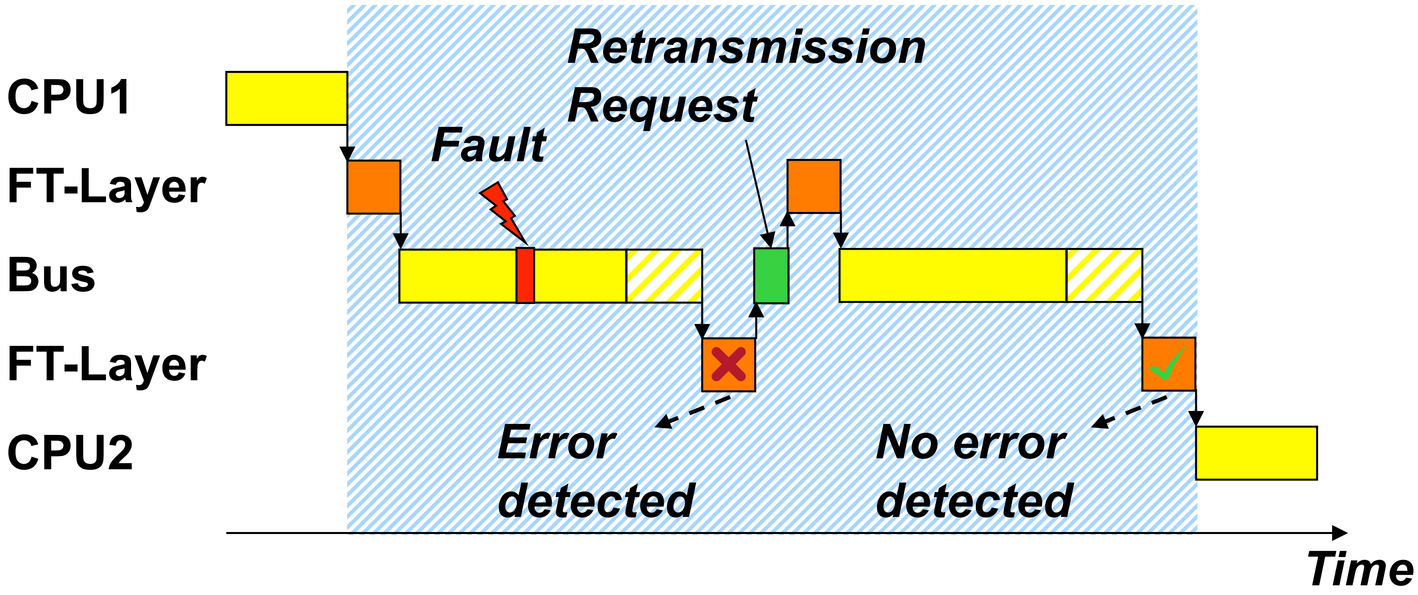
Response time of τ_3 = 12 ✓
→ 3 correction tolerable

- exclusive consideration of the CI induces too much pessimism
- analysis of other situations than the CI may result in more favorable results
- try approach for buses

Communication protocol: basic assumptions

- single bus with sequential data transmission
- periodically triggered data transmission
  - periodic activation pattern generates an infinite sequence of transmission jobs
  - each job is associated with exactly one message to be transmitted
  - response time of a job = time between job release at the sender and the correct delivery of the corresponding message at the receiver
- static-priority based arbitration policy to resolve bus access conflicts (cf. CAN)
  - channel with highest priority gains bus access
  - transmission of messages is non-preemptive
- message transmission may be disturbed by errors
  - *assumption:* single bit errors, characterized by bit error rate (BER)

Communication protocol: fault tolerance



- fault tolerance due to adaption of error detection codes
  - redundant information to detect errors
  - if an error has occurred: *retransmission* of affected message
- transmission time:

$$t_{com,EDC}(n_e) = t_{com,FT} + n_e \cdot (t_{com,FT} + t_{RR})$$

Formal reliability analysis - Overview

- **limitation of known approach: pessimistic worst-case consideration**
  - reliability as probabilistic measure should address the average case
  - use multiple event busy window approach
- **new approach: consider every individual message transfer separately**
  - incorporate individual interference situation of each transmission job
  - use differentiated error penalty
- **two-step algorithm (cp. Burns, but different function)**
  - step 1: identify all tolerable error scenarios for a dedicated transmission without any deadline miss
  - step 2: calculate the probability that no deadline will be missed
- **translate to R(t) and MTTF**

Formal reliability analysis – some basic definitions

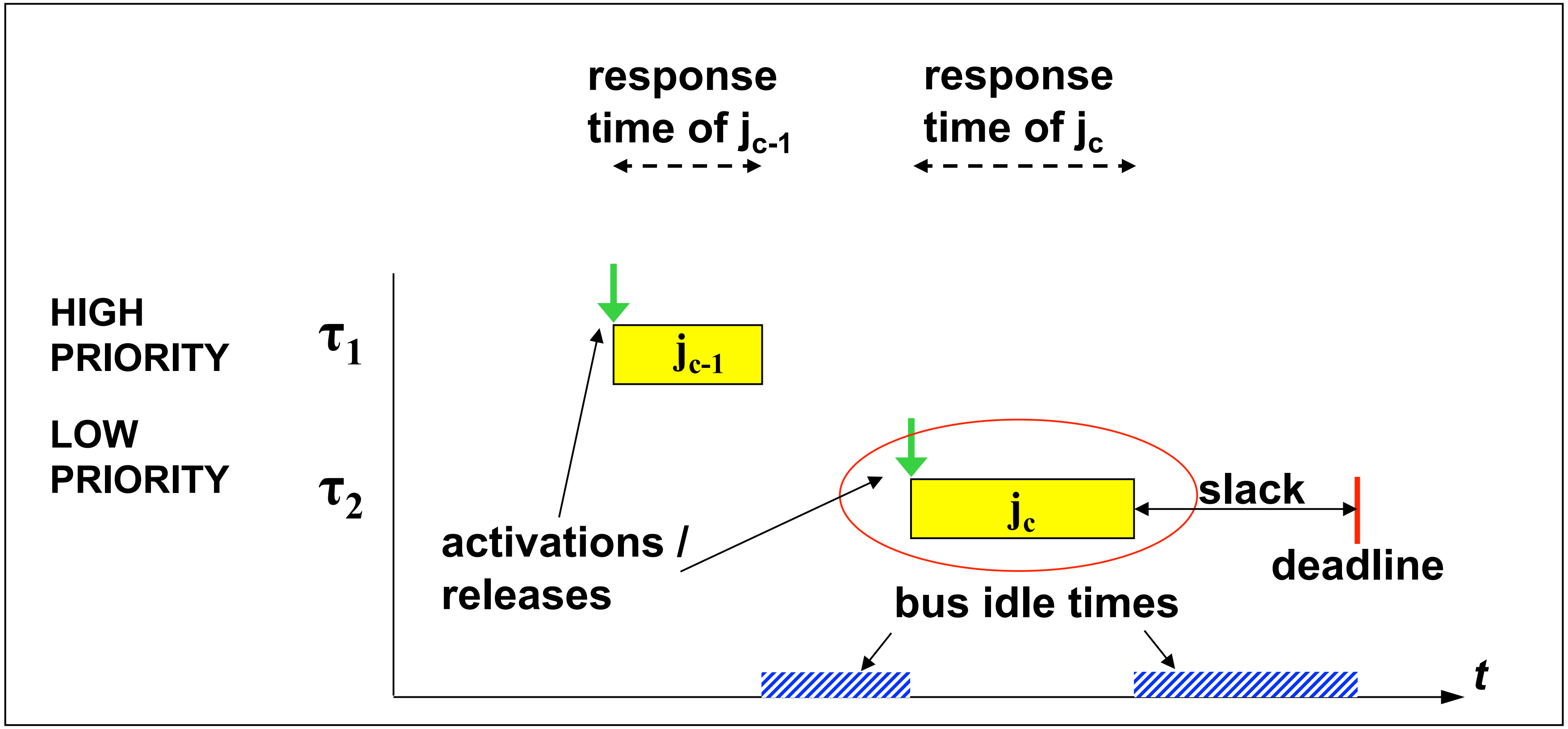
- ***Definition***: An error scenario $\mathbf{s_{i,k}}$ is a dedicated error situation for which the response time of a job $\mathbf{j_i}$ is calculated
- ***Definition***: The working set $\mathbf{W_i}$ of a job $\mathbf{j_i}$ is the set of all error scenarios $\mathbf{s_{i,1}, \ldots, s_{i,Ki}}$ for which $\mathbf{j_i}$ is not violating any timing constraint
- ***Definition***: An error event $\boldsymbol{\varepsilon_i}$ is defined as the occurrence of an error during the message transmission of $\mathbf{j_i}$.

Step 1 – determining tolerable error scenarios

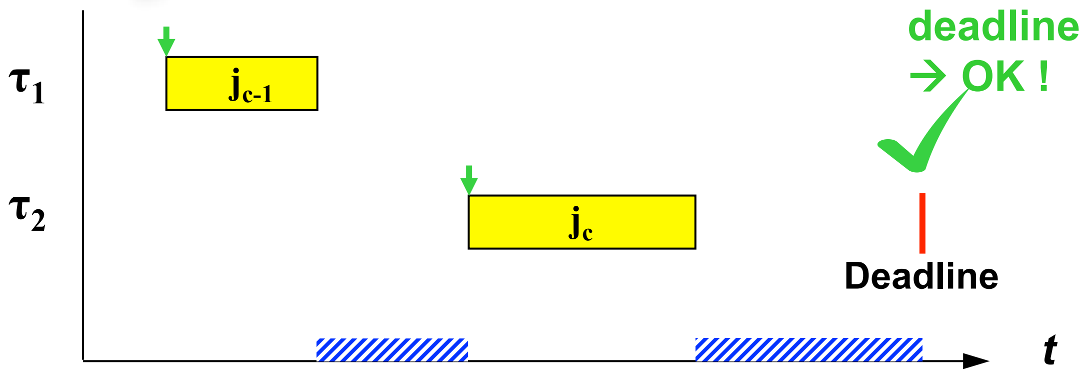
- consider an individual transmission job j_c
- objective: determine all error scenarios without j_c missing a deadline (=tolerable error scenarios)
  - error penalty (= retransmission time) depends on size of affected message
  - exploration of all tolerable error scenario using error tree analysis
- error tree analysis: depth-first graph search for all tolerable error scenarios
  - node = dedicated error scenario
  - edge = error event that increases the overall error number by 1
  - schedulability test for each error scenario:
    response time < deadline ?
    → response time analysis algorithm required

Error tree analysis - example

- example: error tree analysis for message j_c
  - competing job j_{c-1} with higher priority is included into analysis

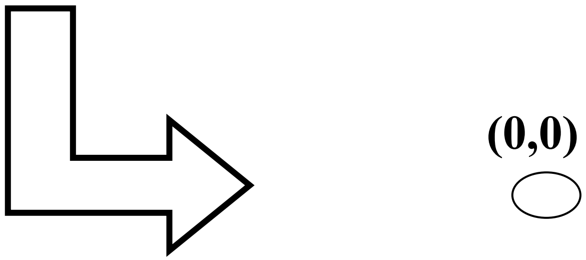


Error tree analysis - example



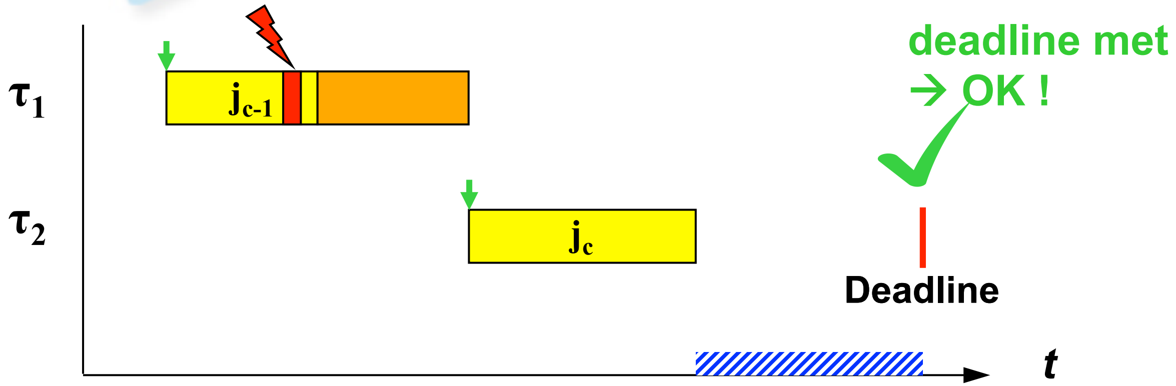
considered error scenario:

(j_c → 0 errors, j_{c-1} → 0 errors)



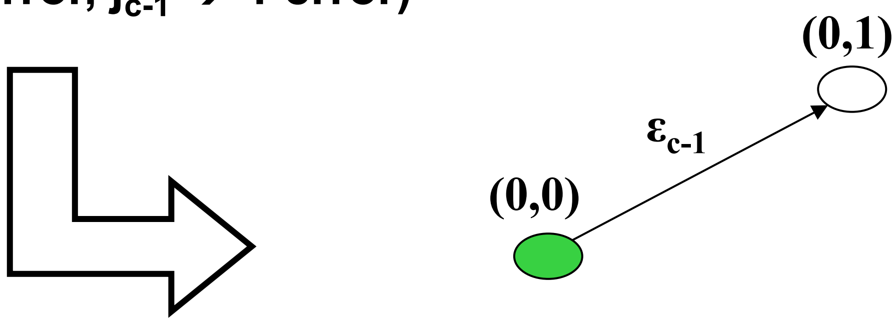
current error tree

Error tree analysis - example



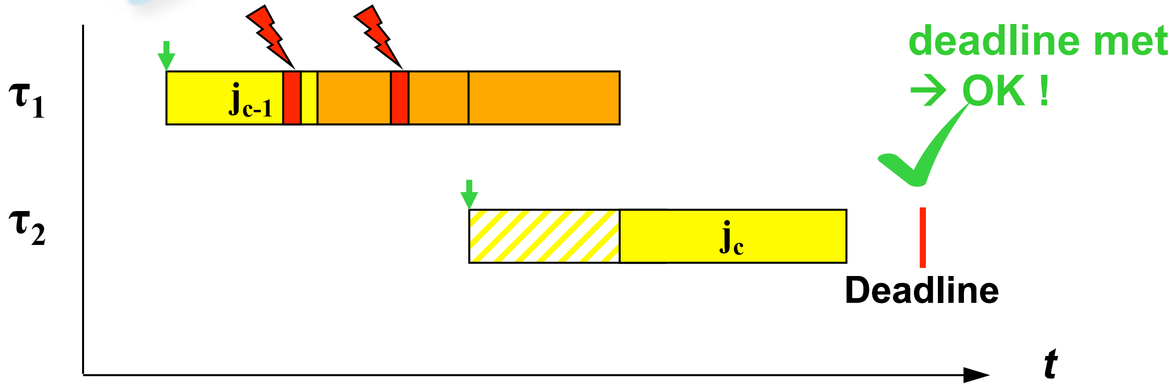
considered error scenario:

(j_c → 0 error, j_{c-1} → 1 error)



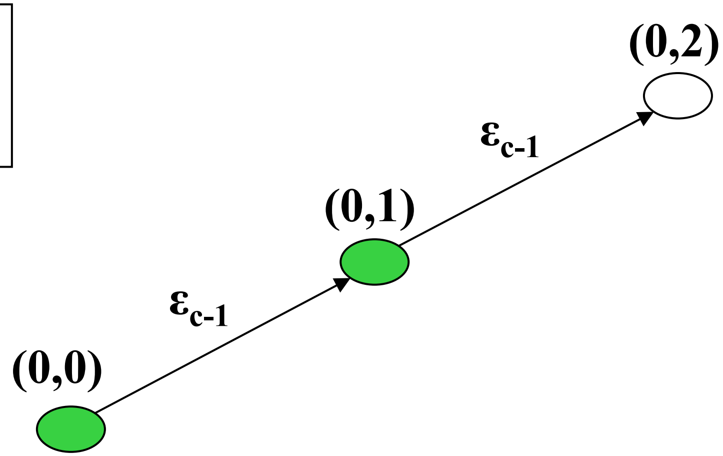
current error tree

Error tree analysis - example



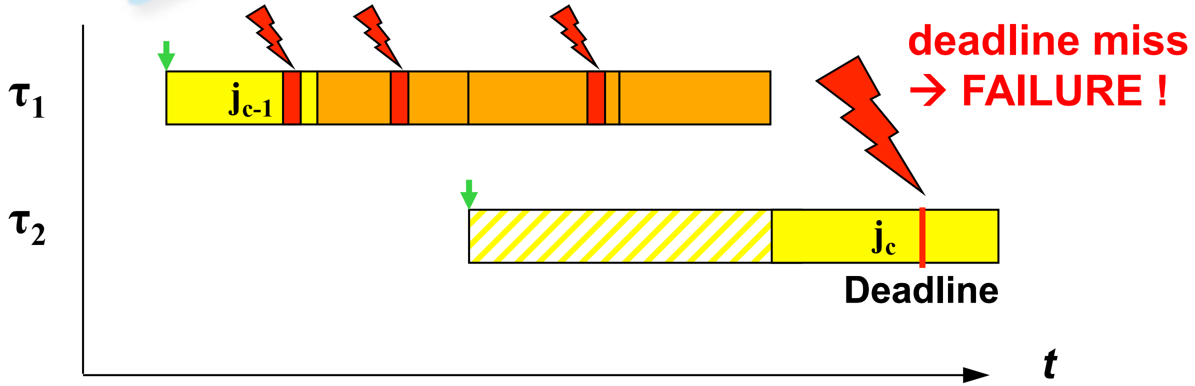
considered error scenario:

(j_c → 0 errors, j_{c-1} → 2 errors)



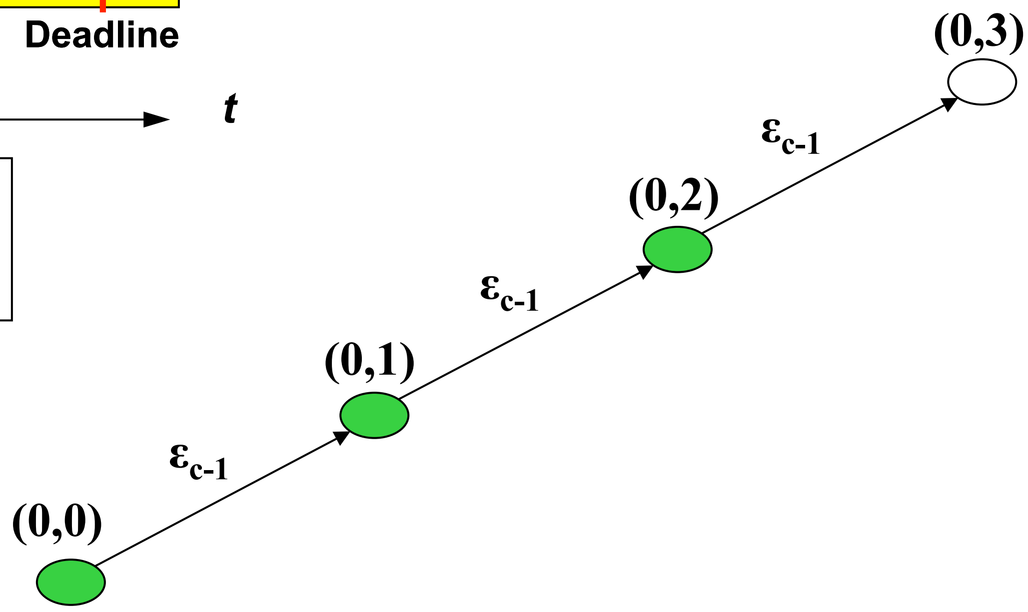
current error tree

Error tree analysis - example



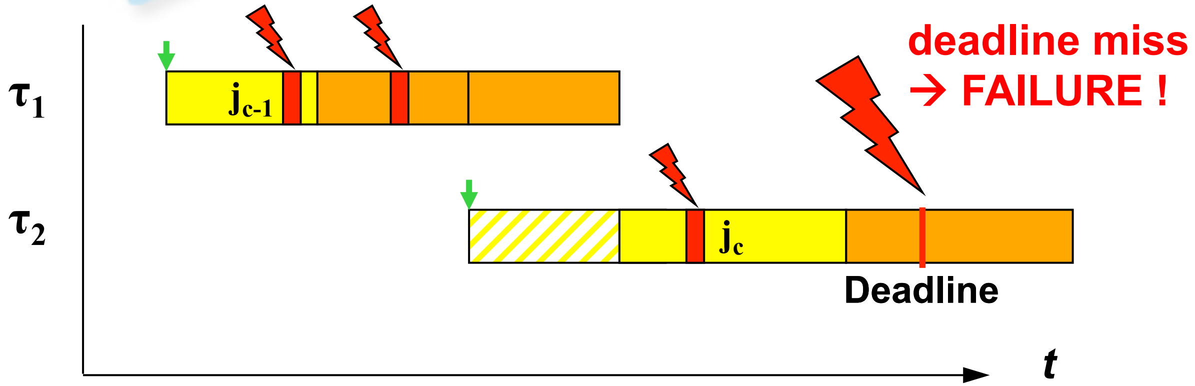
considered error scenario:

(j_c → 0 errors, j_{c-1} → 3 errors)



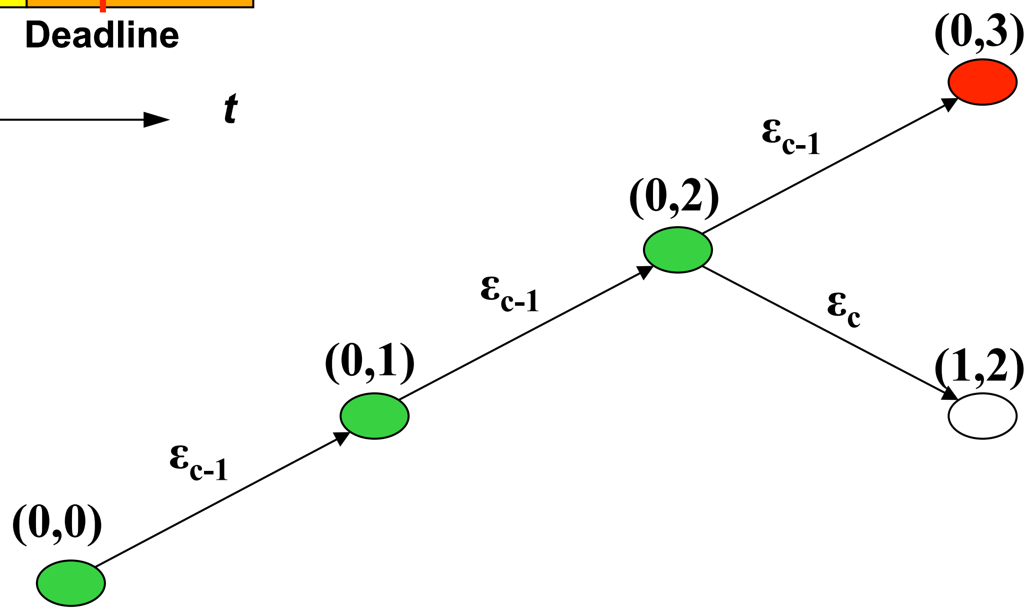
current error tree

Error tree analysis - example



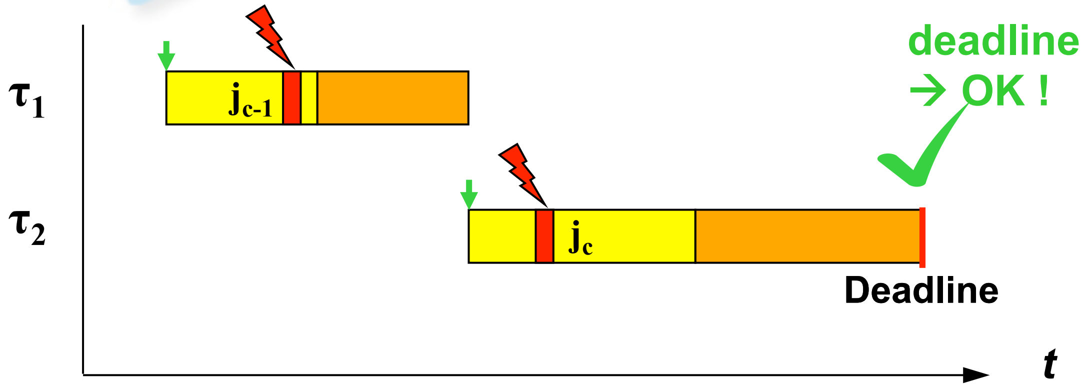
considered error scenario:

(j_c → 1 error, j_{c-1} → 2 errors)



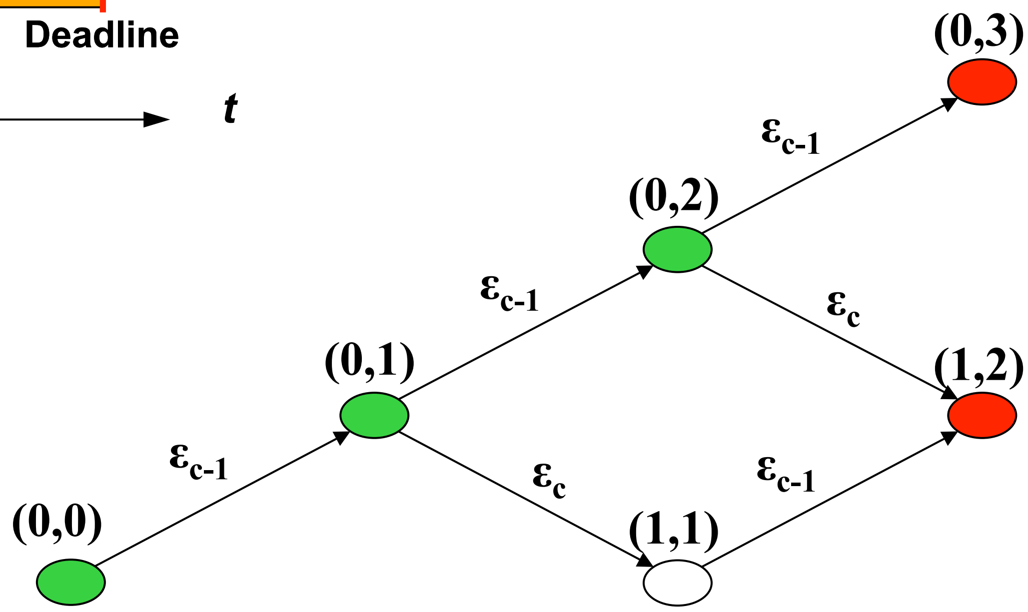
current error tree

Error tree analysis - example



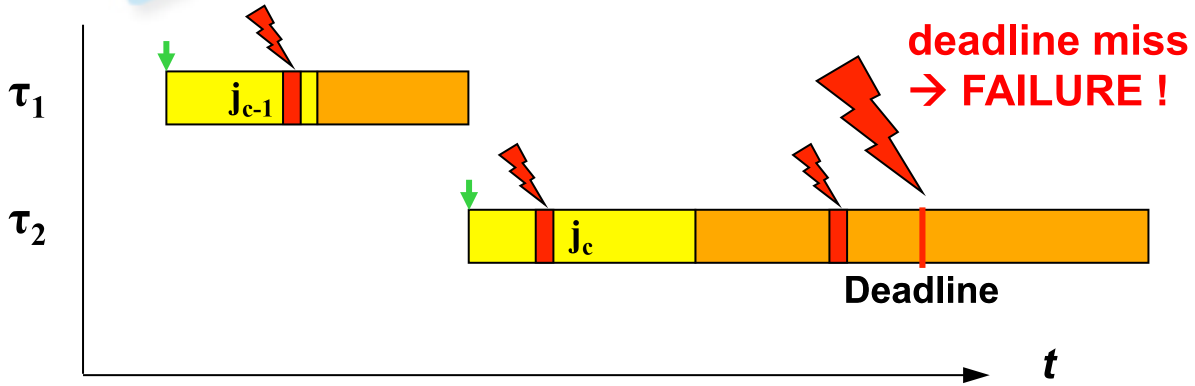
considered error scenario:

(j_c → 1 error, j_{c-1} → 1 error)



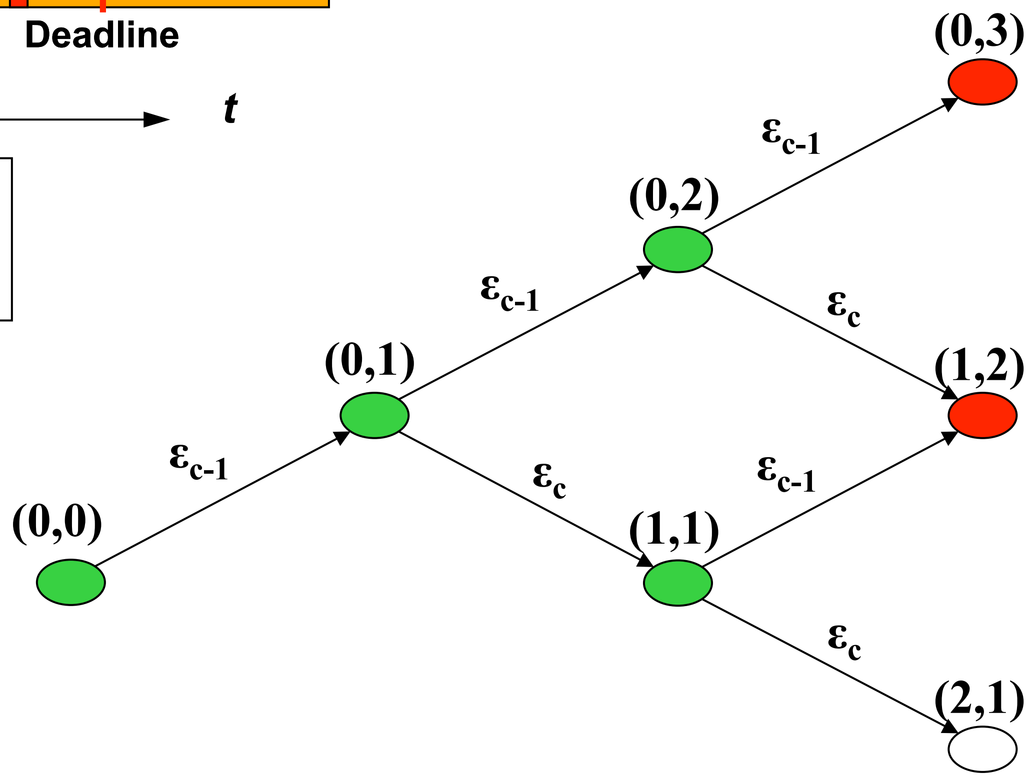
current error tree

Error tree analysis - example



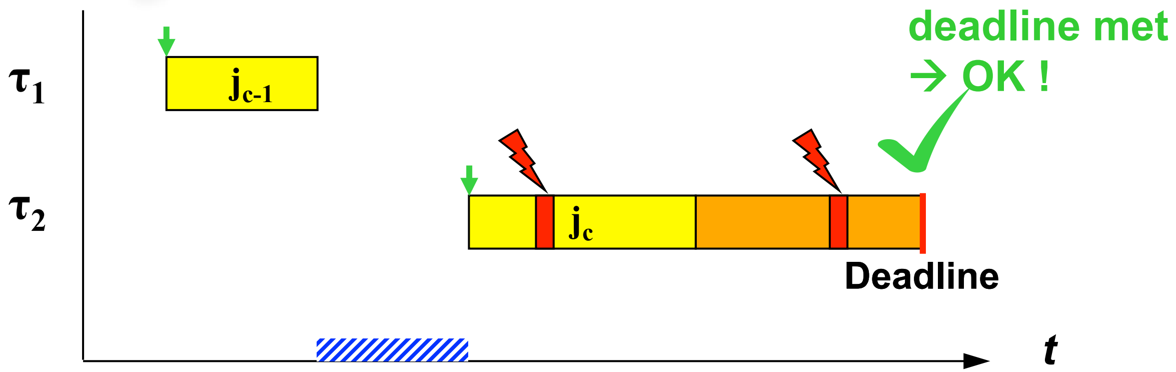
considered error scenario:

(j_c → 2 errors, j_{c-1} → 1 errors)



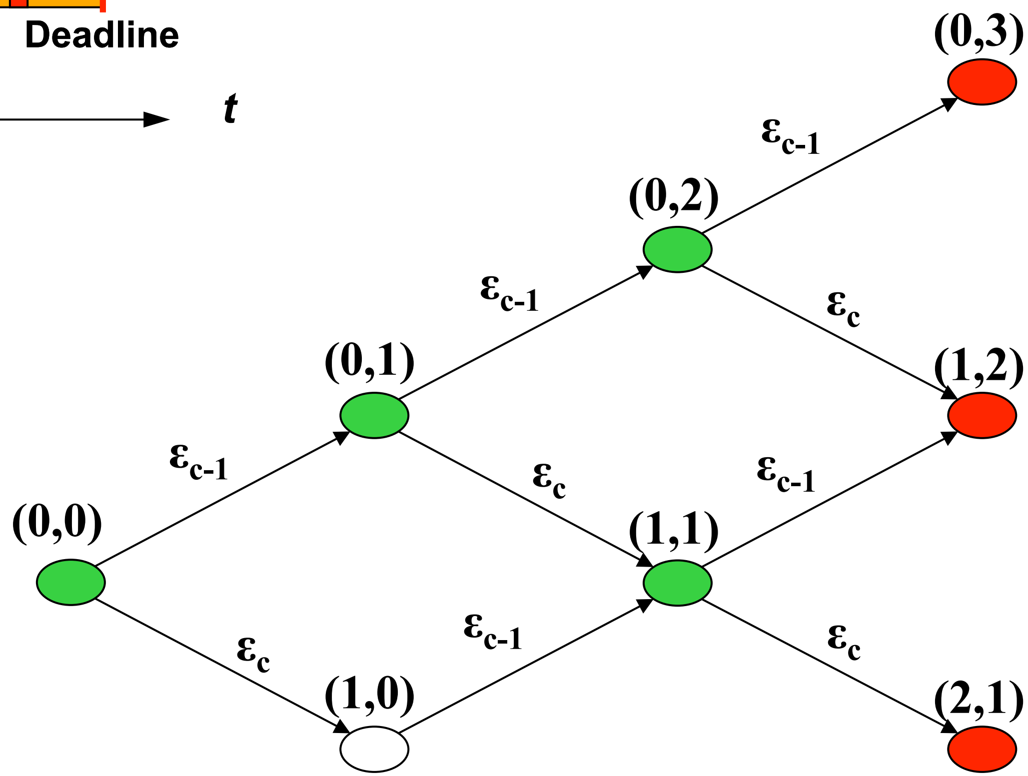
current error tree

Error tree analysis - example



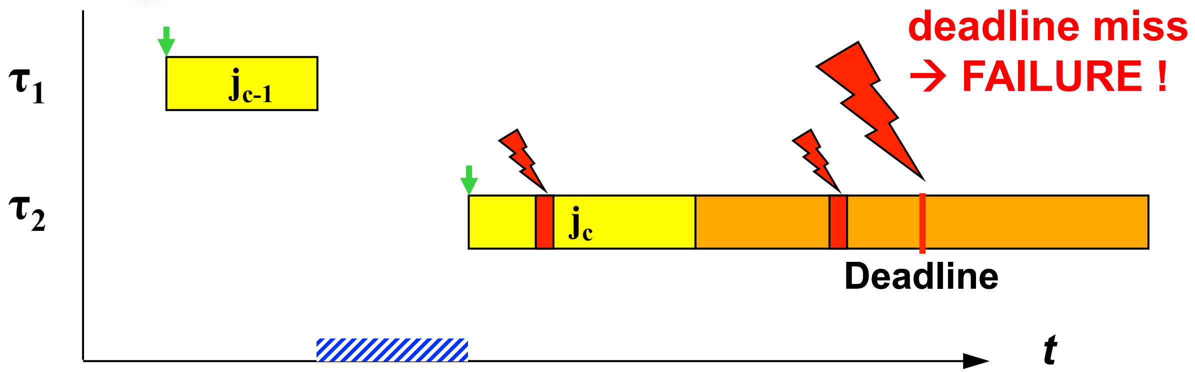
considered error scenario:

(jc → 1 error, jc-1 → 0 errors)



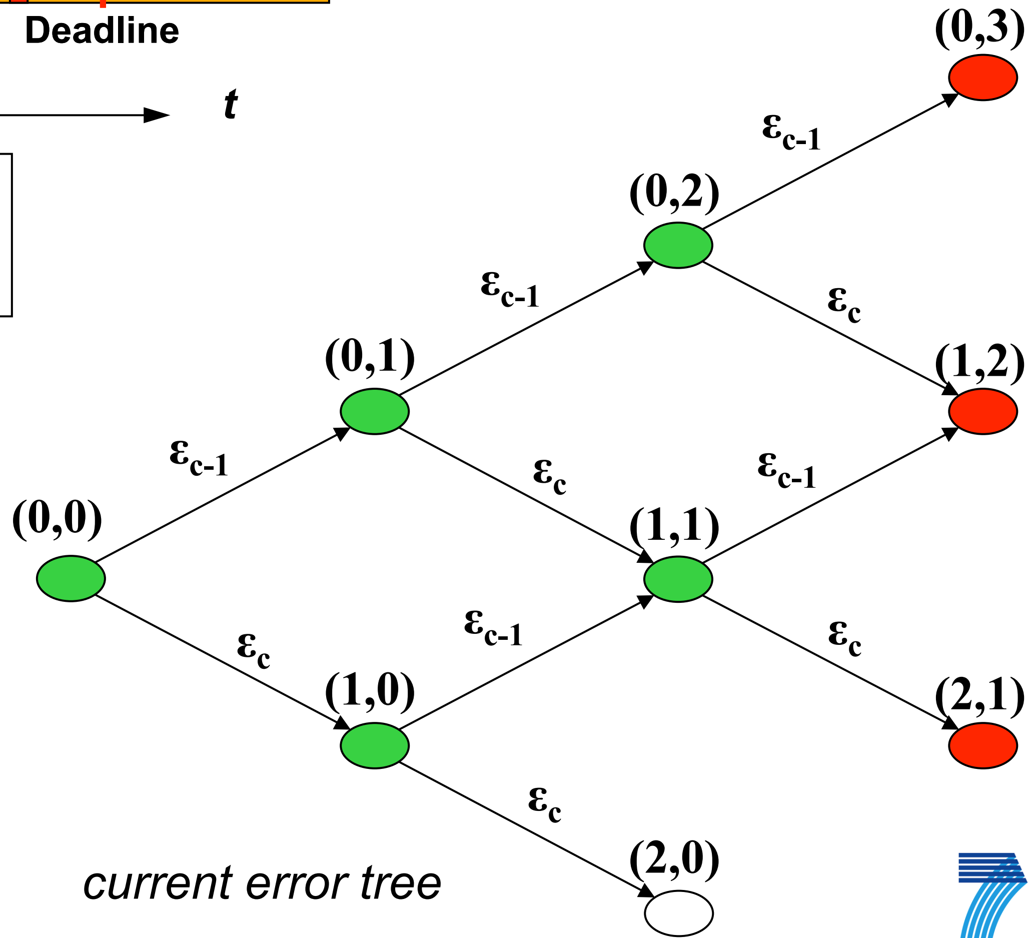
current error tree

Error tree analysis - example



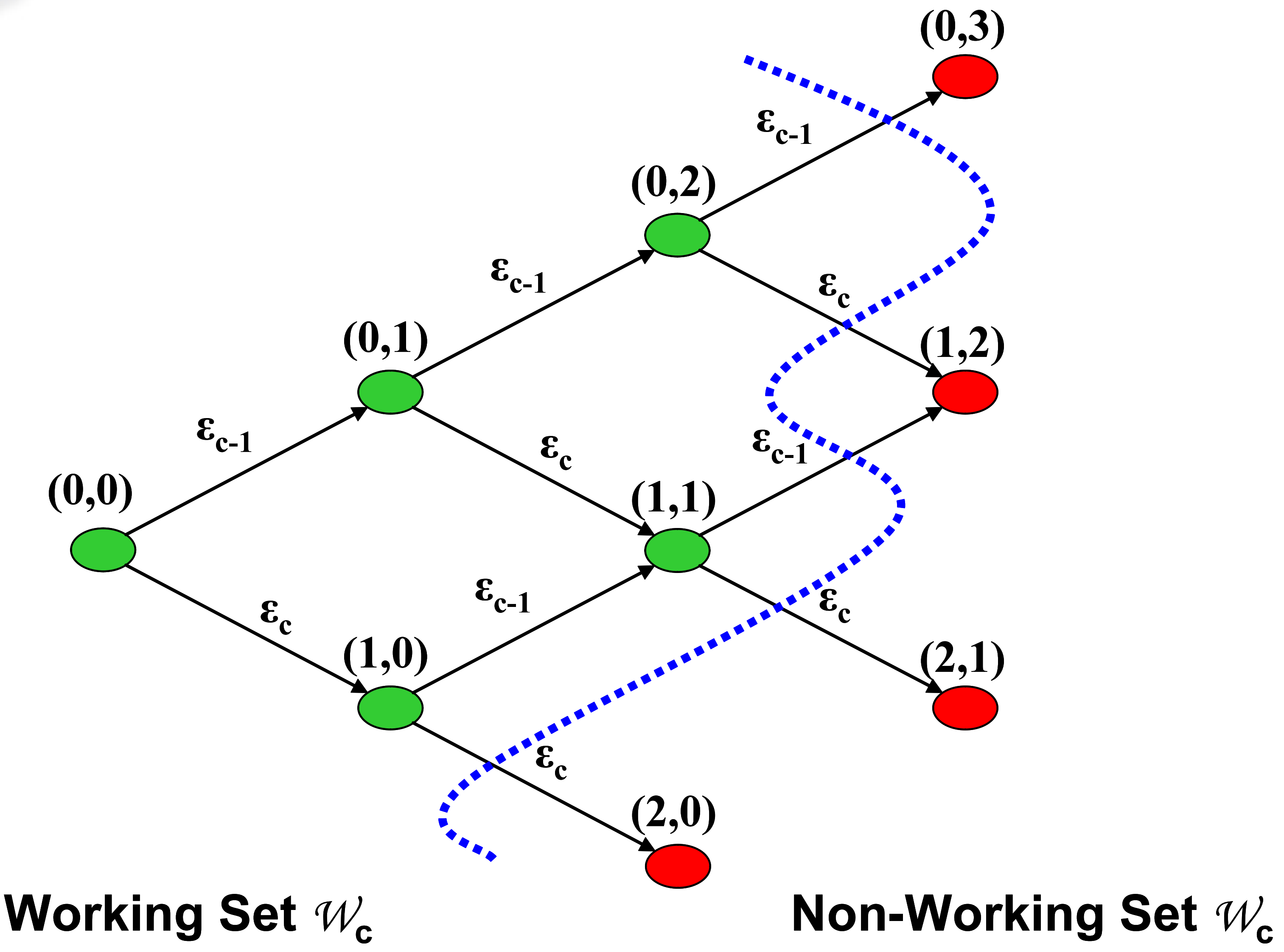
considered error scenario:

(j_c → 2 errors, j_{c-1} → 0 errors)



current error tree

Error tree analysis - result



Error tree analysis - Result

- *problem:* potentially all jobs activated before j_c may influence j_c's response time
  - *solution:* consider only a bounded history → search depth D (*in the example:* D=1)
  - less accurate, but fast approach (lower complexity)
- working set can directly be derived from the error tree analysis result
  - *remember:* perform error tree analysis for each job individually
  - periodic activation causes repetition of error trees after the hyperperiod

    → hyperperiod = lcm of all period
  - restrict analysis interval to the hyper-period

Step 2 – success probability calculation

Definition (**success**): The fact that the job j_i meets its deadline will be referred to with S_i (success of j_i).

Definition (**scenario occurrence**): The fact that an arbitrary error scenario $s_{i,k}$ of a job j_i actually occurs is denoted with $\omega^{i,k}$.

- working W_c set contains all scenarios for which the job j_c will meet its deadline
- success probability of j_c = probability that exactly one the error scenarios contained in W_c occurs:

$$P[S_c] = P[\omega^{c,1} \vee \omega^{c,1} \vee \ldots \vee \omega^{c,K_c}]$$

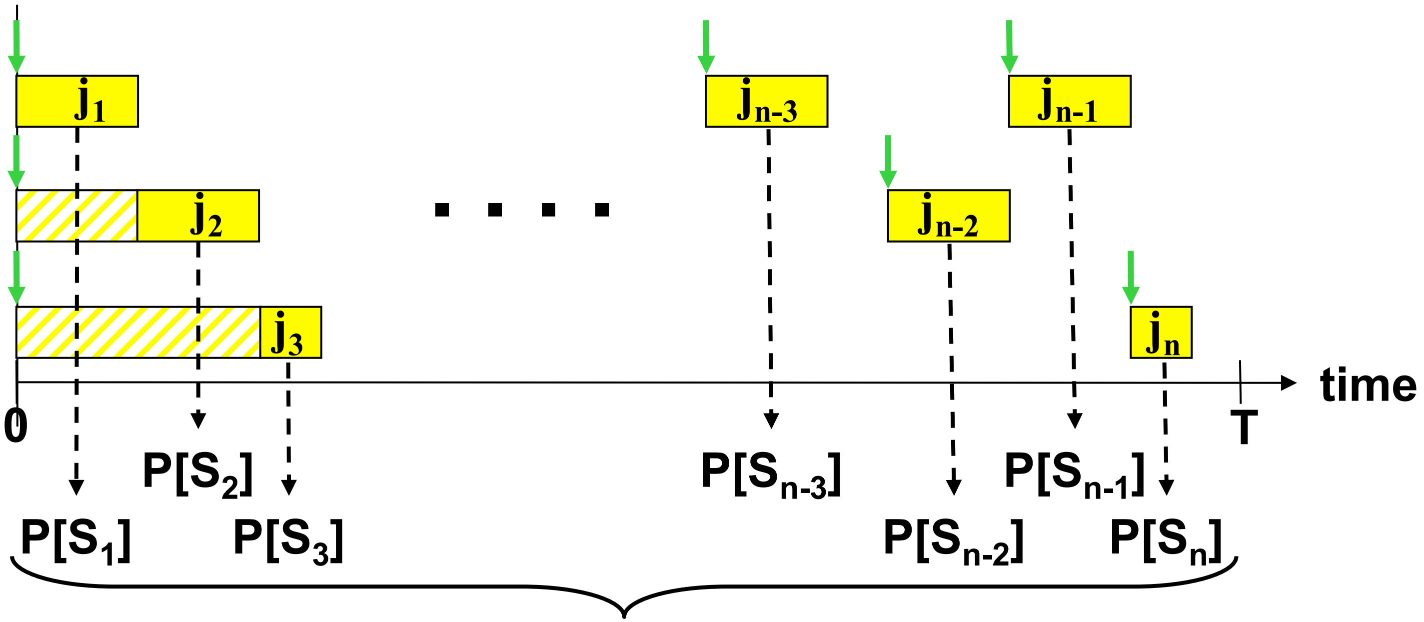
$$= P[\omega^{c,1}] + P[\omega^{c,2}] + \ldots + P[\omega^{c,K_c}]$$

SEVENTH FRAMEWORK PROGRAMME

Step 2 – Derivation of reliability

Remember: **reliability R(t) = probability of no failure in [0,t]**

⇨ **R(t) = probability that no job in [0,t] misses its deadline**

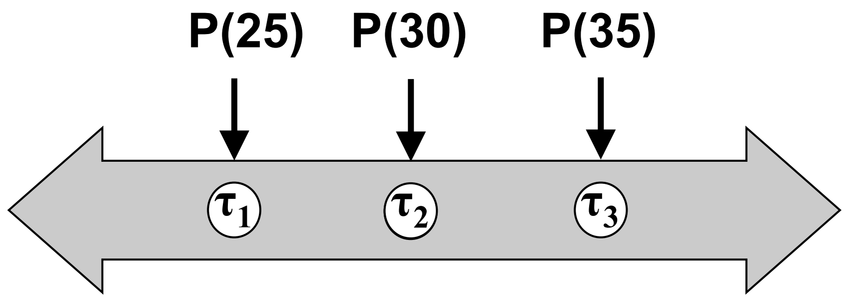


$$R(T) = P[S_1 \wedge S_2 \wedge S_3 \wedge \ldots \wedge S_{n-3} \wedge S_{n-2} \wedge S_{n-1} \wedge S_n]$$

$$= P[S_1] \bullet P[S_2 \mid S_1] \bullet \ldots \bullet P[S_n \mid S_{n-1}, \ldots, S_1]$$

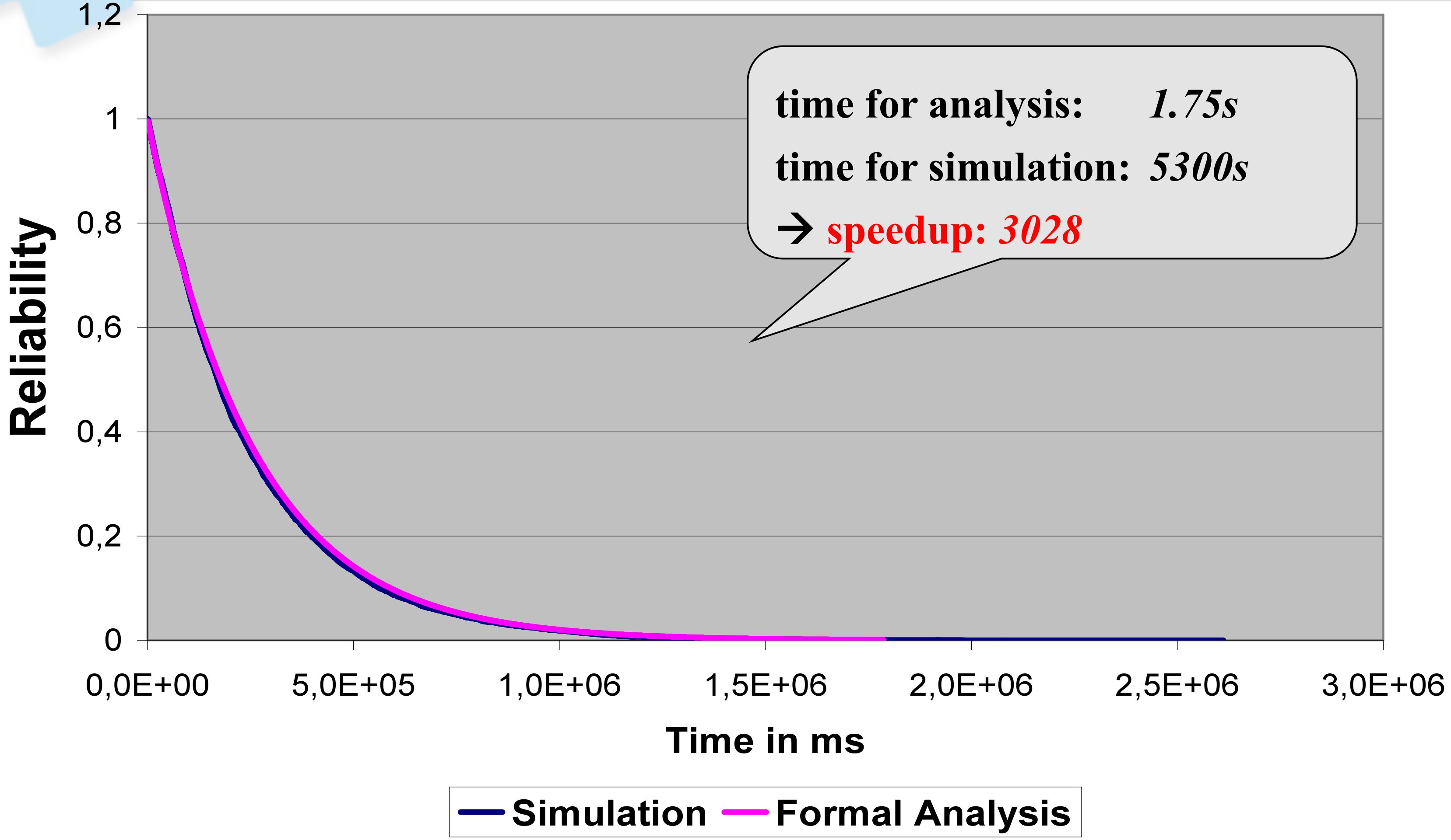
Experiment - Analysis accuracy

- 3 logical communication channels, periodically activated
- deadline of each message equal to the channel's period
- bit error rate: $4 \cdot 10^{-4}$
- Monte-Carlo simulation as reference to determine formal analysis accuracy
- *fault tolerance mechanisms:* CRC-16 EDC with retransmission



Channel	*Period*	*Size*	*Priority*
τ_1	25	144	1
τ_2	30	96	2
τ_3	35	128	3

Analysis accuracy – Results



- *in error case:* message retransmission with the original priority
- *alternative:* message retransmission basically have highest priority (see Literature)

Analysis accuracy – Results

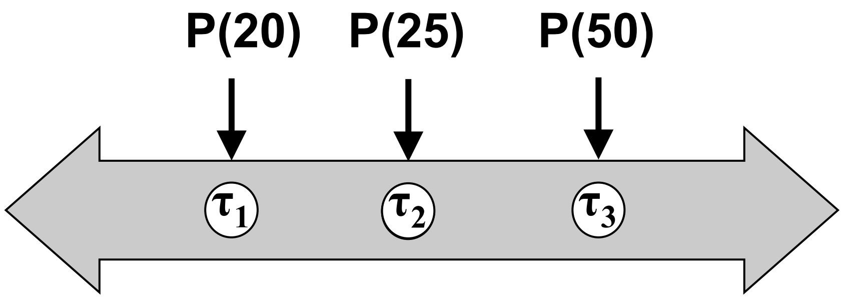
Time t	Formal Analysis $\Re(t)$, $D = 4$	Formal Analysis $\Re(t)$, $D = 3$	Formal Analysis $\Delta_{D=3;4}$ (in %)	Simulation $\Re(t)$	Simulation $\Delta_{Sim,D=4}$ (in %)
10^4	0.96026	0.96032	0.0065	0.96	0.02726
$2 \cdot 10^4$	0.92585	0.92596	0.01237	0.92413	0.18534
$3 \cdot 10^4$	0.88906	0.88923	0.01888	0.88907	0.00101
$4 \cdot 10^4$	0.8572	0.85741	0.02473	0.86	0.327
$5 \cdot 10^4$	0.82313	0.82339	0.03124	0.82747	0.52642
$6 \cdot 10^4$	0.79364	0.79393	0.0371	0.79253	0.13884
$7 \cdot 10^4$	0.7621	0.76243	0.04360	0.75853	0.46769
$8 \cdot 10^4$	0.73479	0.73515	0.04946	0.73253	0.30666
$9 \cdot 10^4$	0.70559	0.70598	0.05596	0.70293	0.37617
10^5	0.68030	0.68072	0.06182	0.67267	1.12231
$5 \cdot 10^5$	0.14513	0.14558	0.30936	0.1336	7.94241
10^6	0.02106	0.02119	0.61776	0.01293	38.5931

differences between formal analysis with search depth of 3 and 4

deviation between simulation and formal analysis

Application example 1: Mixed criticality

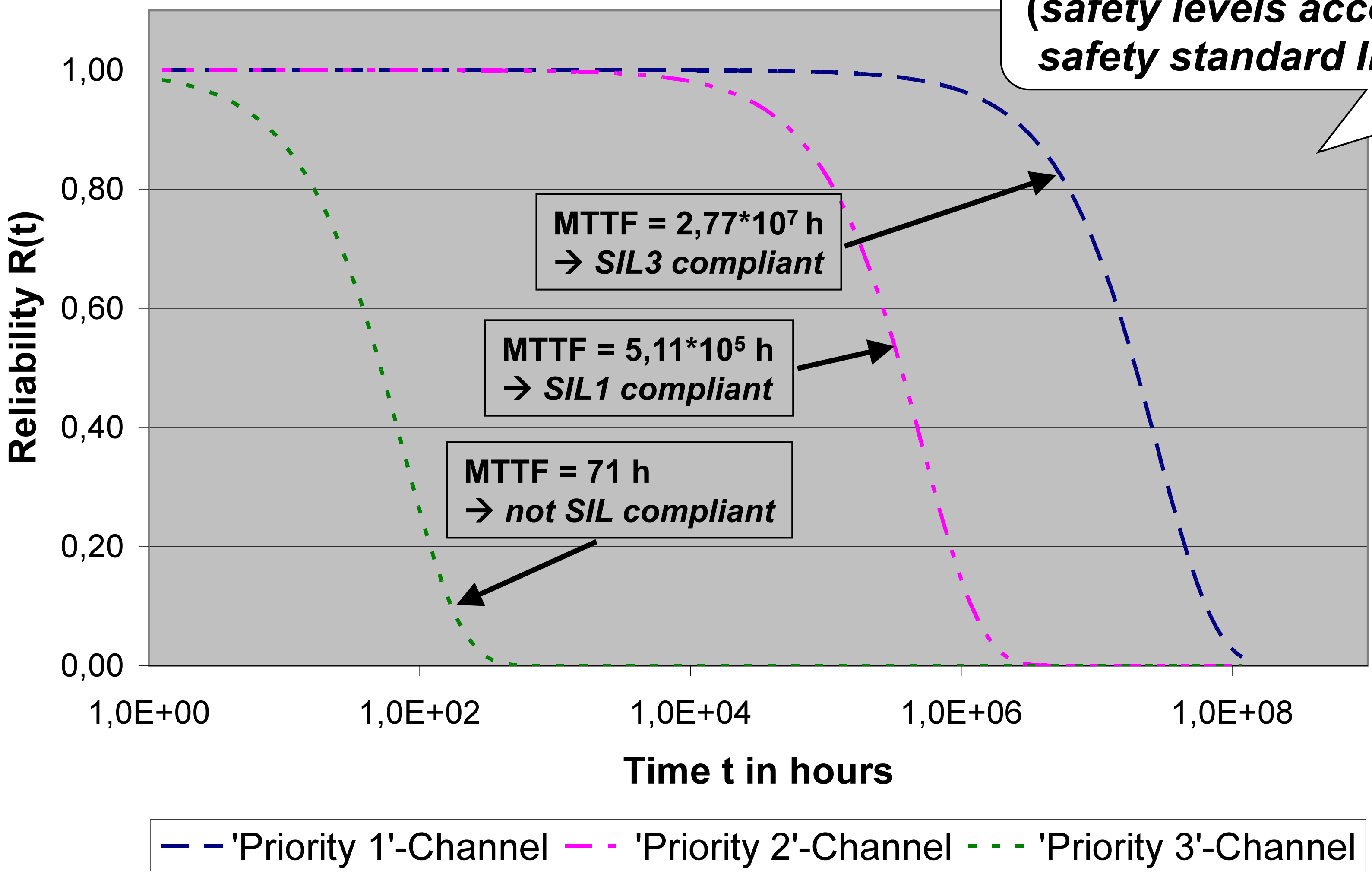
- 3 logical communication channels, periodically activated
- deadline of each message equal to the channel's period
- bit error rate: $4 \cdot 10^{-4}$
  (to be able to use Monte-Carlo simulation as reference)
- *fault tolerance mechanisms:* CRC-16 EDC with retransmission



Channel	*Period*	*Size*	*Priority*
τ_1	20	80	1
τ_2	25	80	2
τ_3	50	128	3

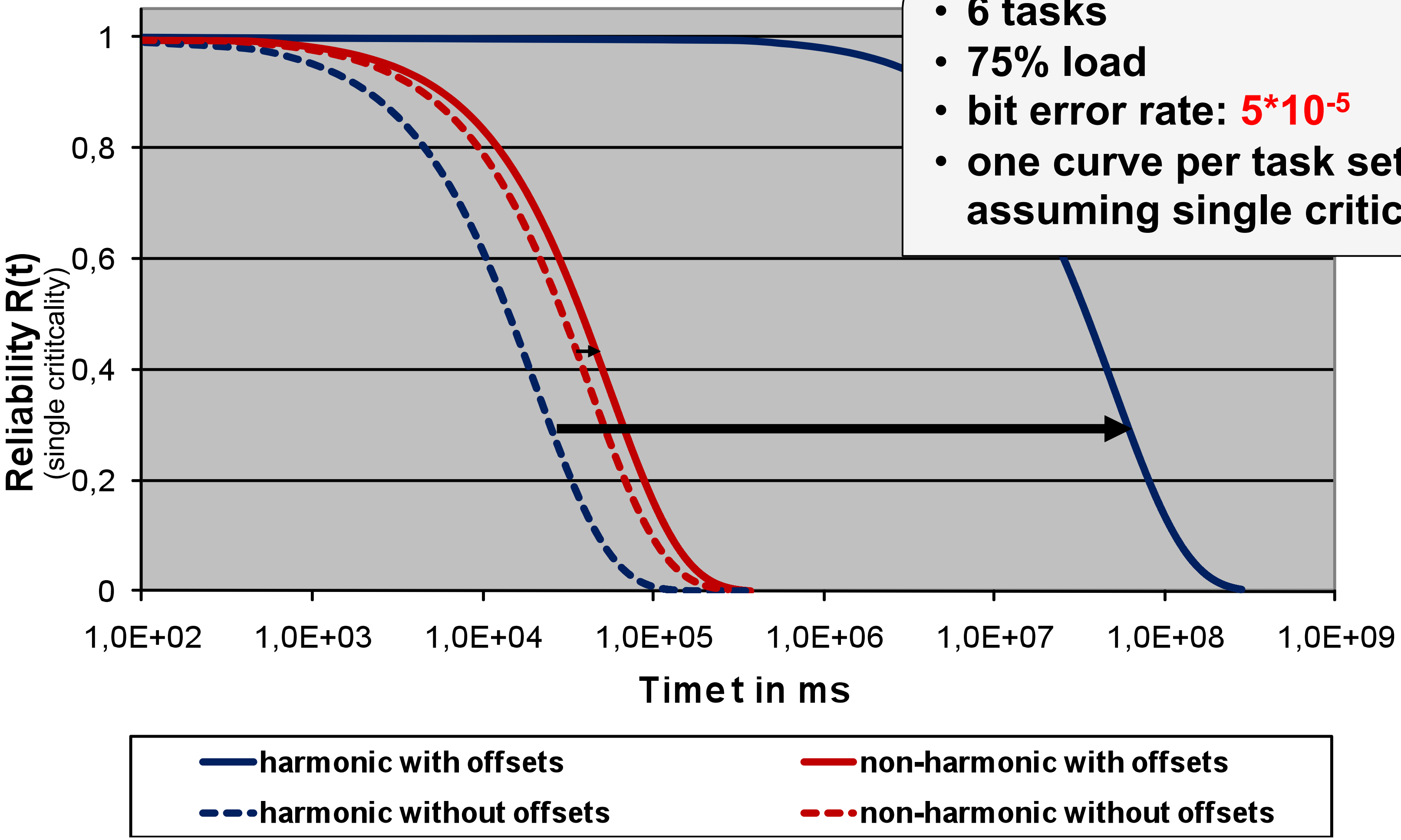
Example 1 – Result

SIL = Safety Integrity Level
(*safety levels according to safety standard IEC 61508)*



Example 2: Offset effect

- 6 tasks
- 75% load
- bit error rate: $5*10^{-5}$
- one curve per task set assuming single criticality



Example 2: Interpretation

- message sets without offsets
  - develop peak load scenarios that are succeptible to faults
  - harmonic sets are even worse than non-harmonic sets due to high and frequent peak loads
- message sets with optimized offsets
  - optimized harmonic message sets have well distributed load with small fault succeptible peak loads
  - non-harmonic task sets don‘t profit much from offsets

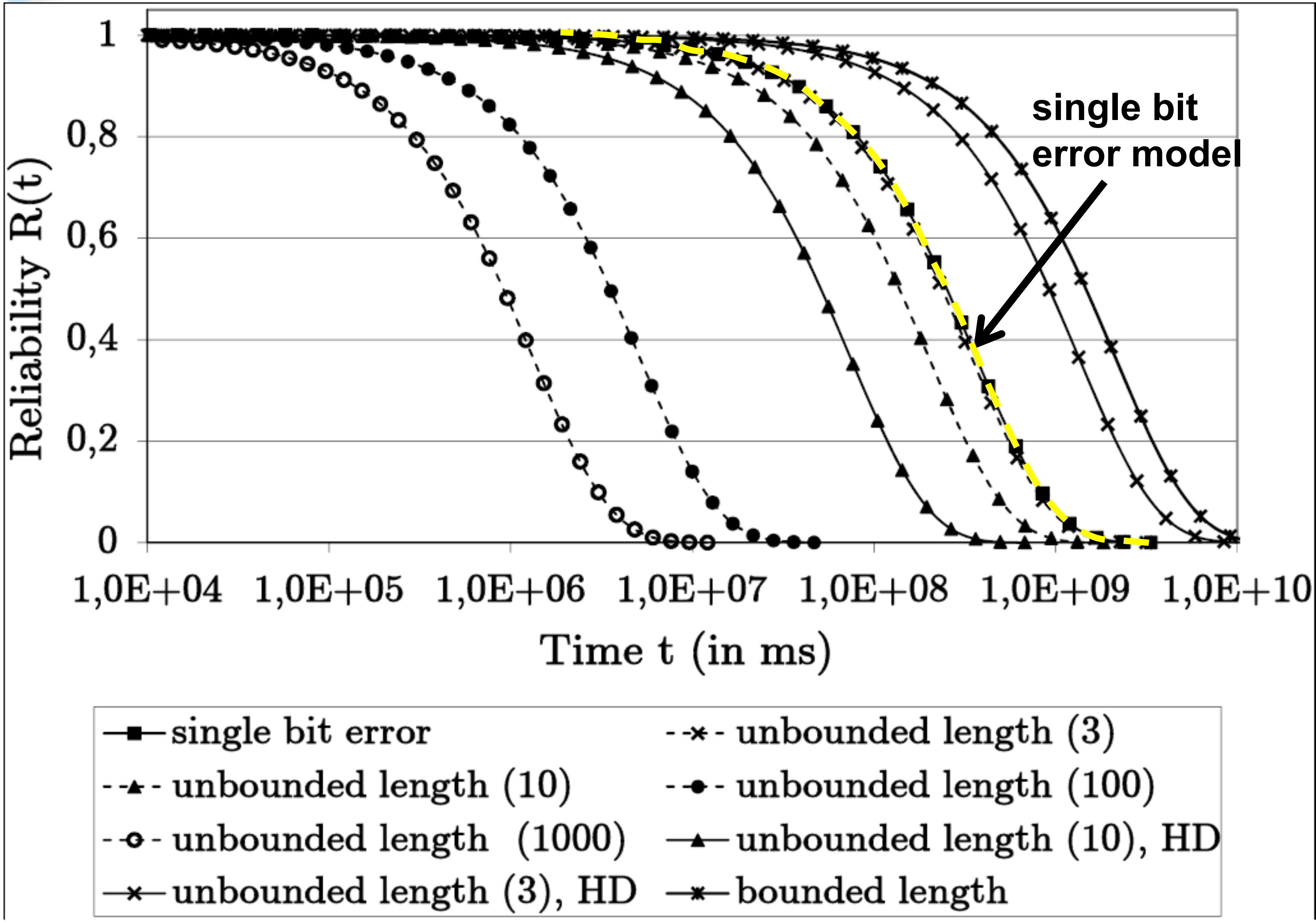
Extending the error model

- *common assumption:* bit errors occur independently
  - each bit might be corrupted with the same probability according to the bit error rate
  - corruption of subsequent messages is even totally independent from each other
  - analysis requires only basic methods of probability theory
- in physical reality bit errors may be highly correlated
  - effect of bursty noise and repeating fault patterns
- accurate models and analysis methods needed to reflect error interdependencies
- use Hidden Markov Models (HMM)

Error Modeling: Hidden Markov Models (HMM)

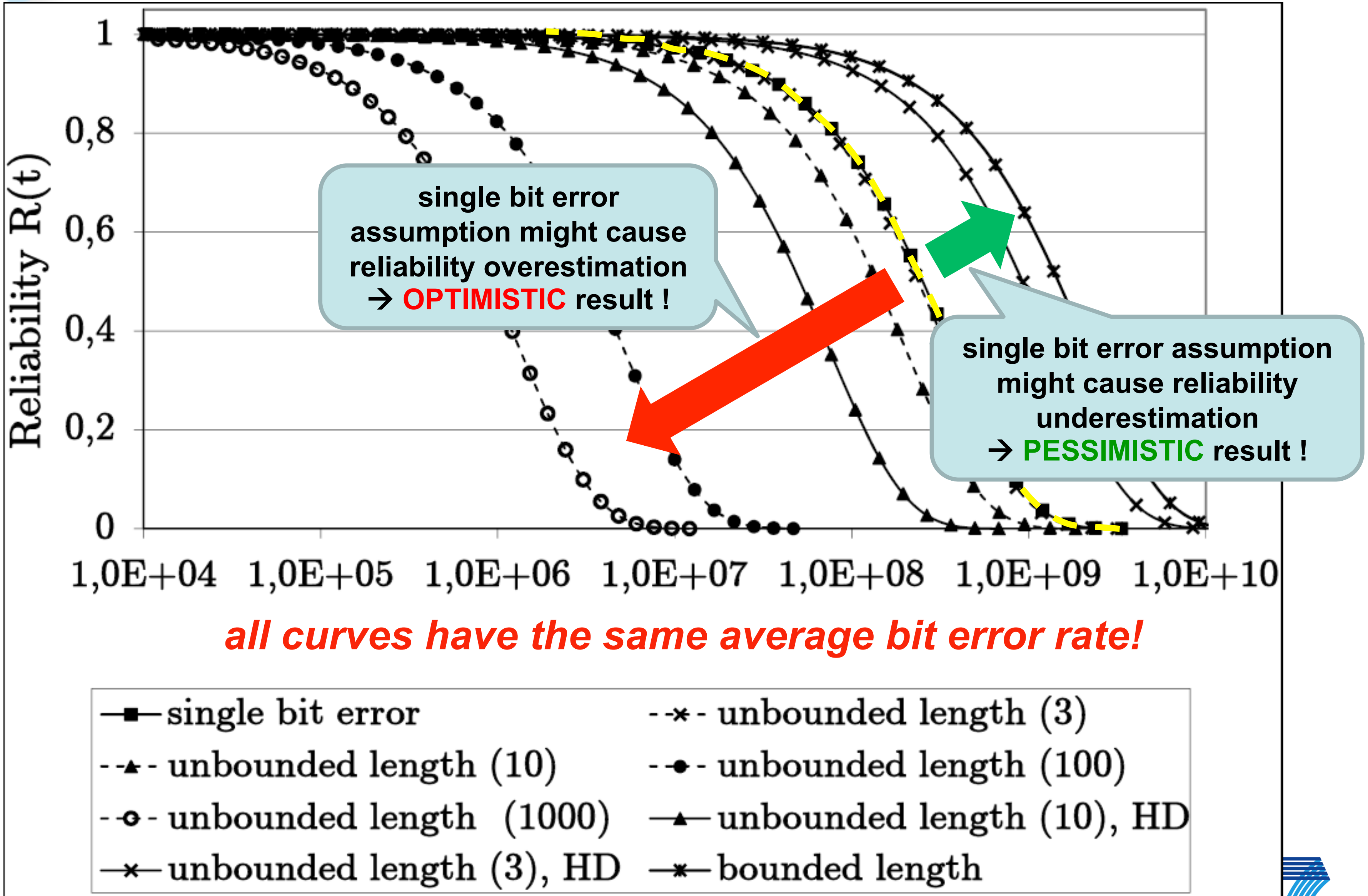
- **hidden Markov models needs to parameterized in a suitable way**
  - **parameter space consists of transition and emission probabilities**
- **parameterized HMM reflect different real-life error parameters**
- ***examples:***
  - **burst errors with bounded length (maximum burst length = 5 bits)**
  - **burst errors with potentially unbounded length (burst length geometrically distributed with mean length = 3,10, 100, 1000 bits)**
  - **burst errors with different bit error density**
- **next slide**
  - ***all curves have the same average bit error rate!***

Error Modeling: Results



artist

Error Modeling: Results

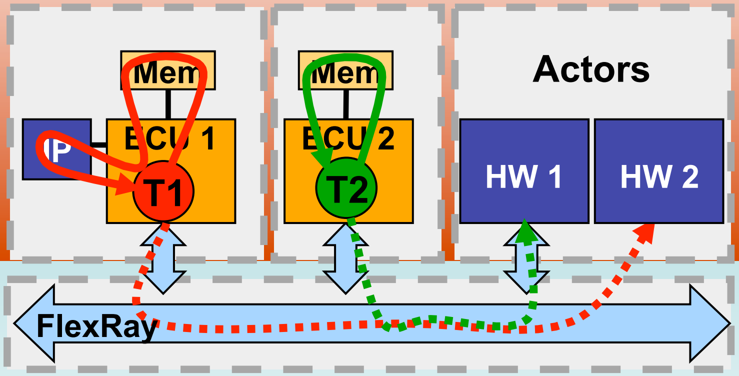


Overview

- motivation
- compositional performance analysis
- **impact of performance on safety and analysis**
  - networks
  - **ECUs**
- application to safety related system design
- system performance self-protection
- conclusion

Motivation: Errors in ECUs

- cosmic radiation
- transistor variation
- shrinking → higher error rates

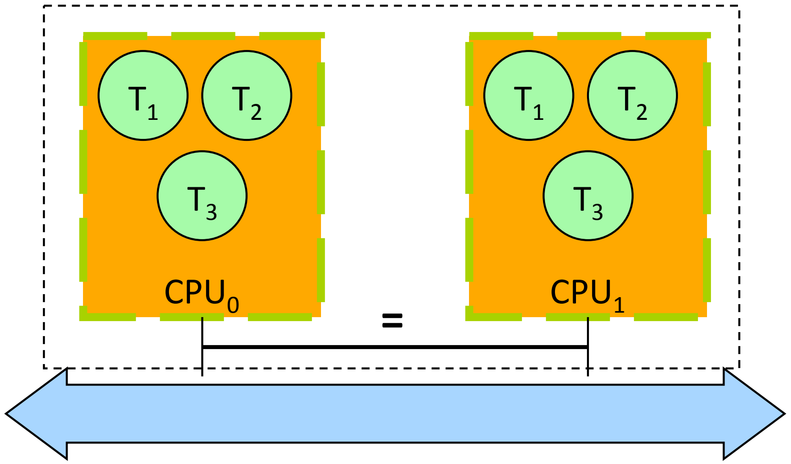


- capacitive/inductive coupling in wiring harness
- HF interference (ignition)
- bad cables/connectors

- **communication can easily incorporate fault-tolerance**
  - **EDC (CRC, parity, hashes)**
  - **ECC (Hamming, Turbo, …)**
  - **→ retransmission, correction**
- **computation it is much harder to protect**
  - **entire processor affected**
  - **control and data flow (including IP cores)**
  - **→ errors can propagate**

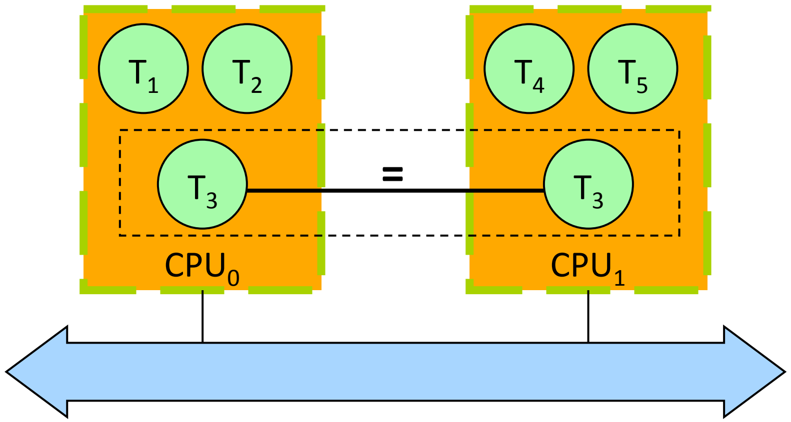
Safety in ECUs

- various methods exist to increase reliability
  - redundancy in various flavors: hot/cold, temporal/spatial
  - acceptance tests (software assertions)
  - vitality checks (watch dog)
  - tradeoff between: detection coverage, reliability, performance overhead, area overhead, predictability
- if not considered ECU/gateway can be single point of failure
- "best practice" in most domains DMR (dual modular redundancy) or TMR (triple modular redundancy) → high cost



Safety ECU – Fine grained approach

- DMR (dual modular redundancy) / TMR (triple modular redundancy) on system level
  - compatible with current real-time performance analysis
- alternative approach DMR/TMR on task level (fine grained)

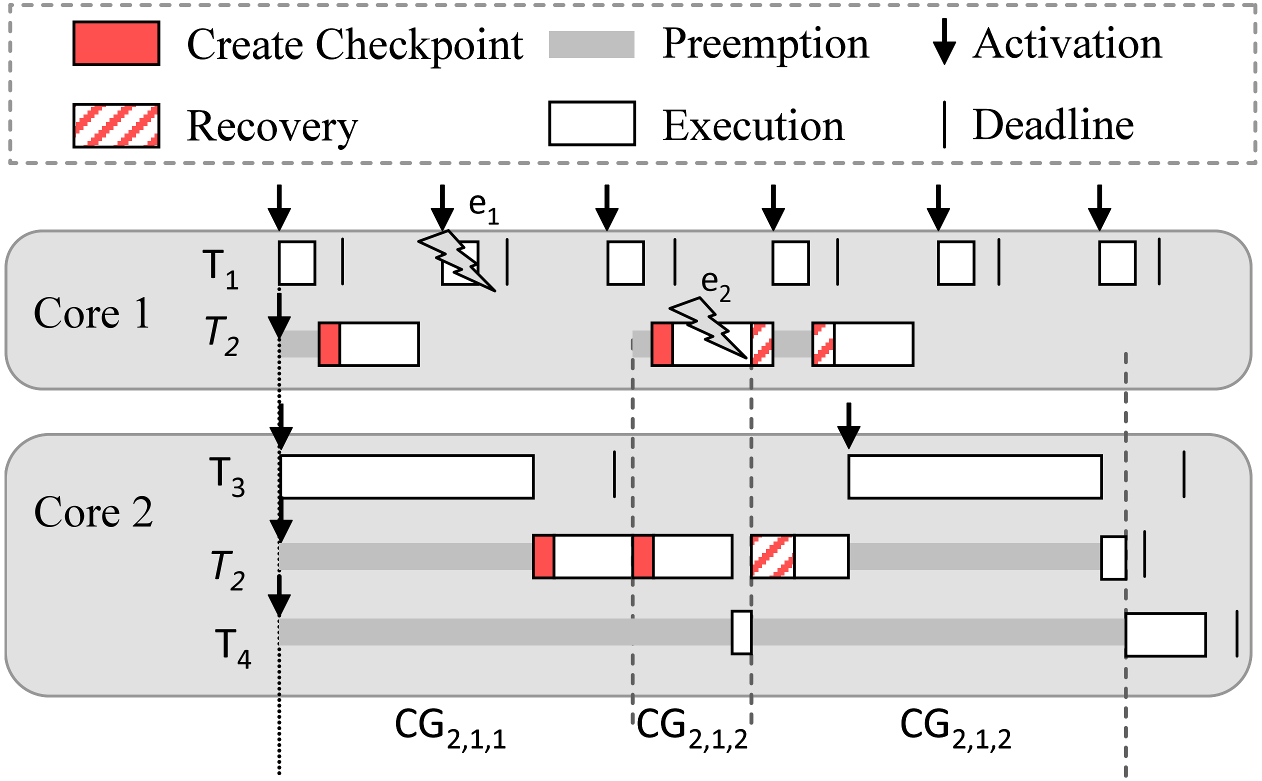


How to compare tasks? How to recover a task?
How to predict timing? How to predict reliability?

Fine grained approach - Task model

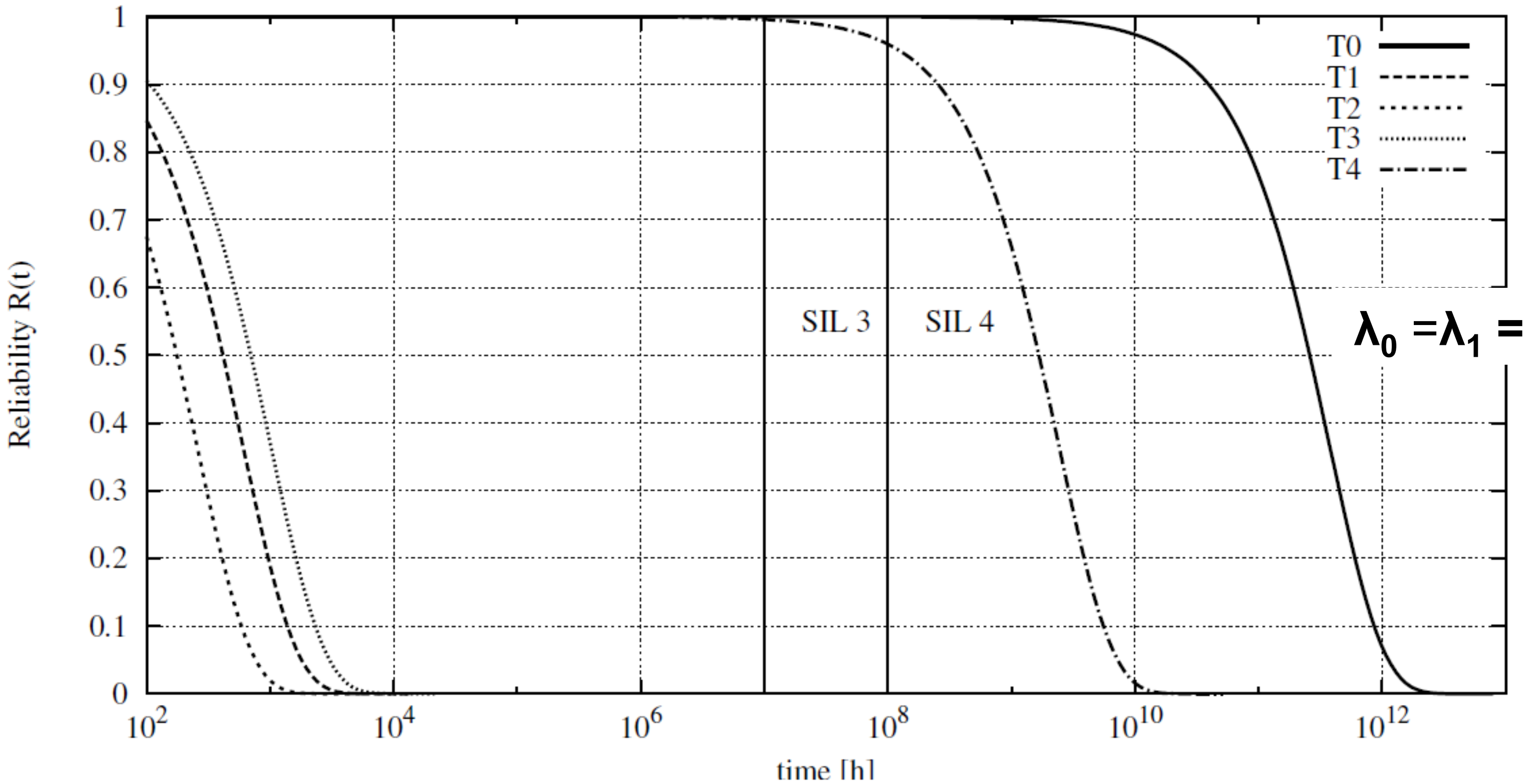
- assuming poisson error model with a given fault rate λ per core – standard in literature
- “regular” tasks
  - periodic, fixed priority, fixed execution time, mapped to exactly one core, partitioned, preemptively scheduled
- “fault tolerant” task
  - redundantly mapped to arbitrary cores
  - partitioned, preemptively scheduled, fixed priority per core
  - employ checkpointing (n-checkpoints per activation)
  - unique fingerprint is calculated for execution stream
    - → detection: comparison of fingerprints (Smolens et al., 2004)
  - in case of errors all redundant copies are reverted to the recent checkpoint
    - roll back and recovery

Example



Task	Mapping	Checkpoints	Type
T1	1	-	regular
T2	1,2	2	ft
T3	2	-	regular
T4	2	-	regular

Formal analysis – Results



$\lambda_0 = \lambda_1 = 1/year$

Task	P_{core1}/P_{Core2}	T	C	CPS	COV	ROV
T0	3 / 2	300 ms	60 ms	2 ms	1 ms	3 ms
T1	4 / -	250 ms	50 ms	-	-	-
T2	2 / -	100 ms	10 ms	-	-	-
T3	- / 1	300 ms	50 ms	-	-	-
T4	1 / 3	600 ms	40 ms	2 ms	1 ms	3 ms

Summary functional failure analysis

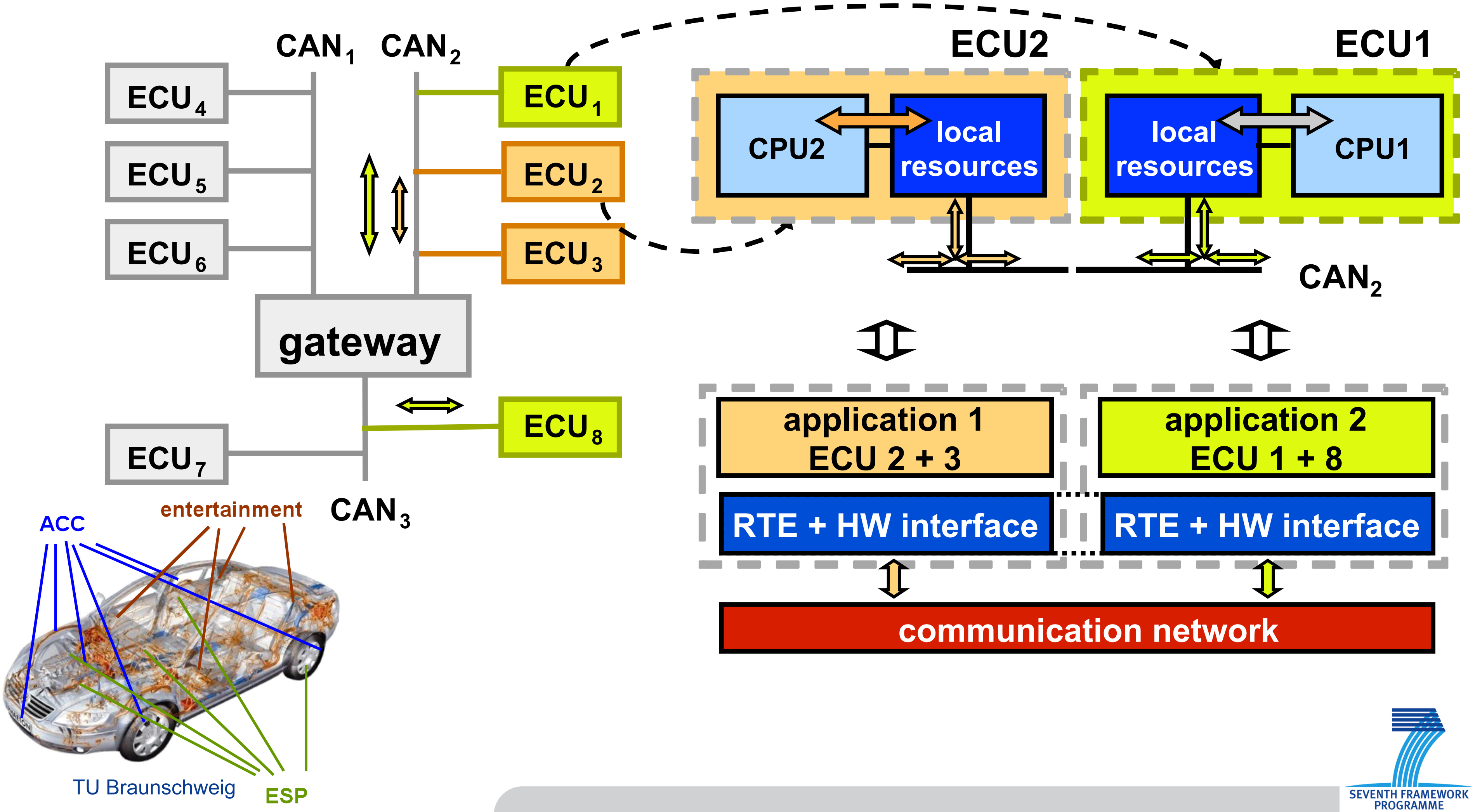
- formal methods for communication and computation component analysis developed for priority based periodic systems
  - method generally applies to other scheduling stratgies
- treat failures as (unbounded) exceptions from WCET behavior
- fault analysis directly maps fault model to safety standard metrics!
- can be used for qualification/certification - key result
  - supports mixed critical system analysis by providing separate results for individual tasks and messages
  - provides probability of fault induced message loss in case of non corrected messages (used for register based communication)
- still missing: complete network and system analysis
  - further work

Overview

- motivation
- compositional performance analysis
- impact of performance on safety and analysis
  - networks
  - ECUs
- **application to safety related system design**
- system performance self-protection
- conclusion

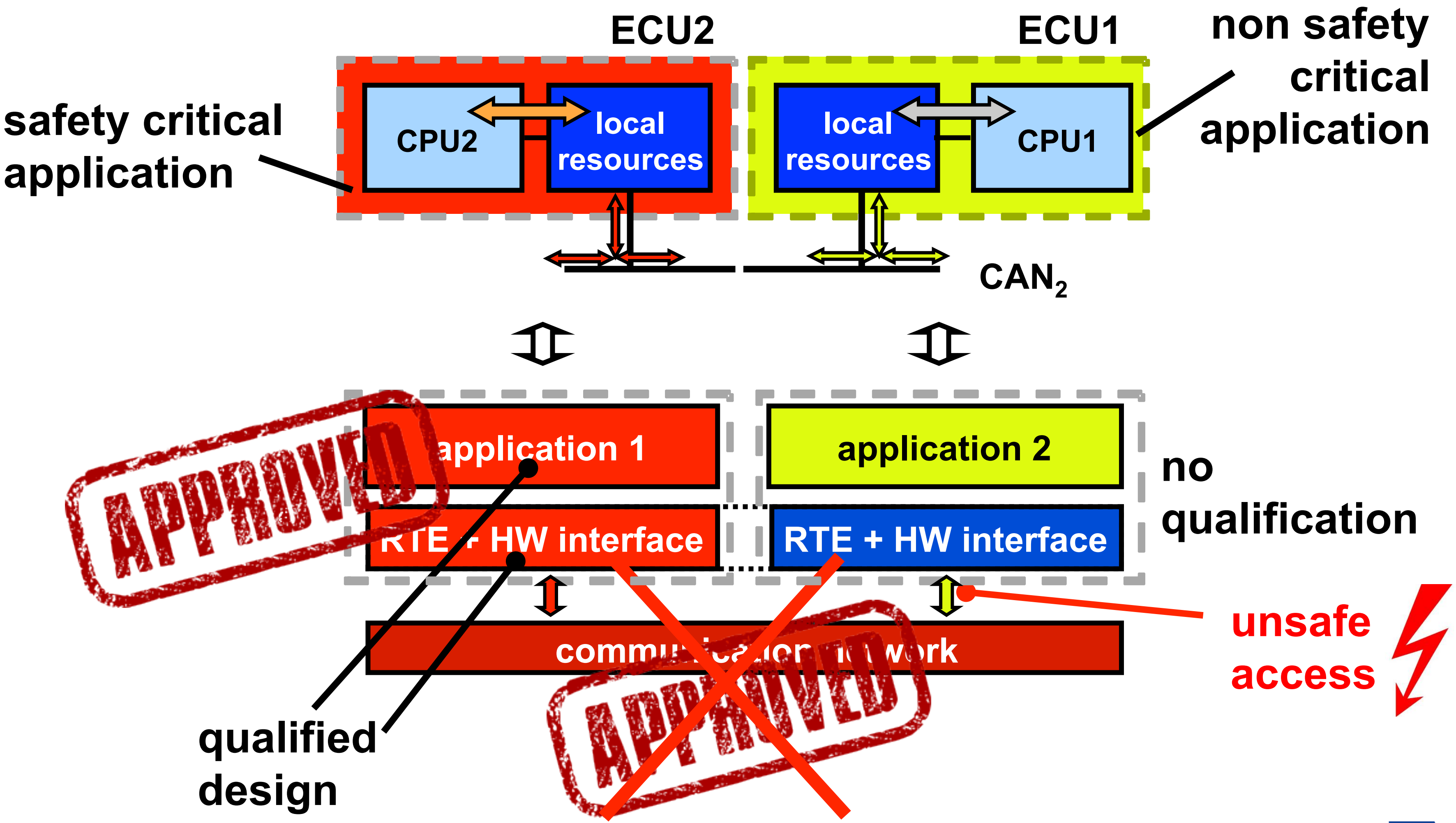
Designing mixed criticality systems

- need **rules** to integrate functions with different criticalities (SIL)
- can be derived from standards



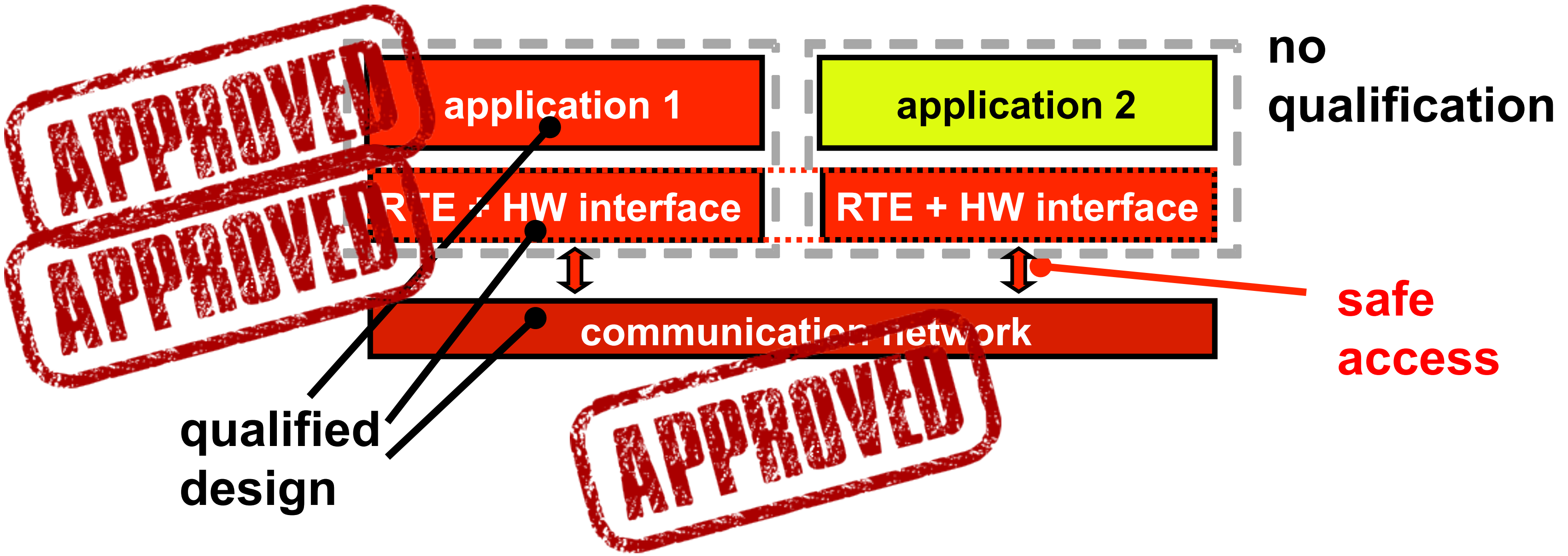
Mixed criticality integration challenge

- integration subject to highest safety standard involved



Correct approach

- qualify network
- isolate and control access of non qualified applications



artist Mixed critical certification/qualification

- **high certification (or qualification, resp.) requirements**
  - **must cover ALL ECUs**
  - **often qualified data of non-critical application not available**
  - **repetion required for *any* update/upgrade (even non-critical)**
- **minimize effort by isolation of different criticalities**
  - **network designed according to highest safety standard of any function involved**
  - **for each change of any function using the network**
    - **determine impact of change on critical network traffic**
    - **validate that change does not violate critical function requirements**
      - **constructive (e.g. FlexRay static segments) and/or tool based - SymTA/S**
      - **include function and timing error handling !**
- **even more complicated for multi-core design – further dependencies**

Overview

- motivation
- compositional performance analysis
- impact of performance on safety and analysis
  - networks
  - ECUs
- application to safety related system design
- **system performance self-protection**
- conclusion

Self-protection - Motivation

- increasing design process complexity

  today

  – increasing number of variants

  – updates of large system parts

  – different mapping of functions in different configurations (AUTOSAR enabled)

  future

  – hardware upgrades

  – migration of functions between cores or ECUs (load distribution, reliability)

  – self-optimizing, self-learning functions

Test Challenge

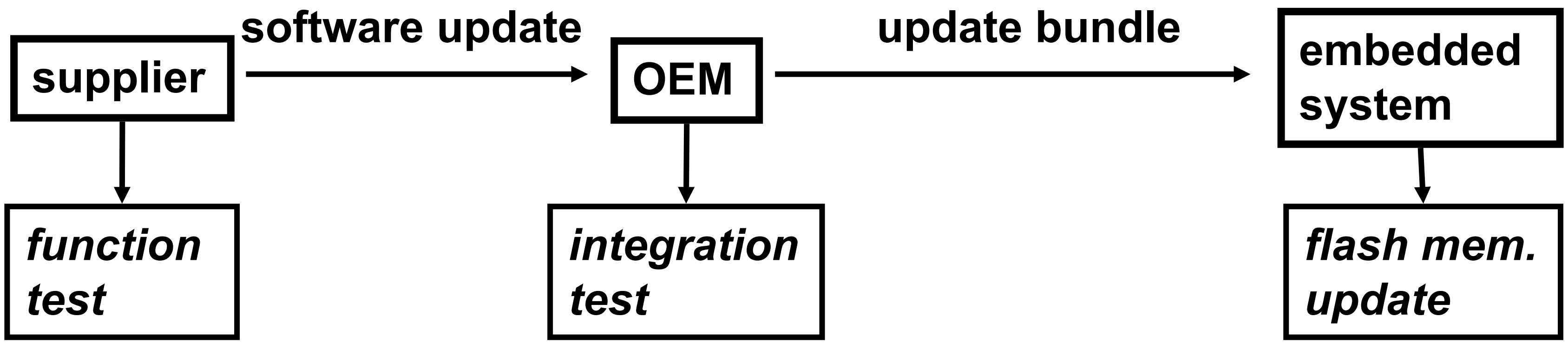
- test problems and cost are likely to become overwhelming
  - large configuration space
    - large data base needed
  - anticipation of change very difficult and will require new test strategies
  - new features, e.g. advanced driver assistance functions (ADAS) require complex test scenarios
    example: object recognition and tracking for emergency break
  - these functions are safety relevant and require repeated formalized design approaches
- *will test cost eventually become the show stopper for automotive innovations?*

SEVENTH FRAMEWORK PROGRAMME

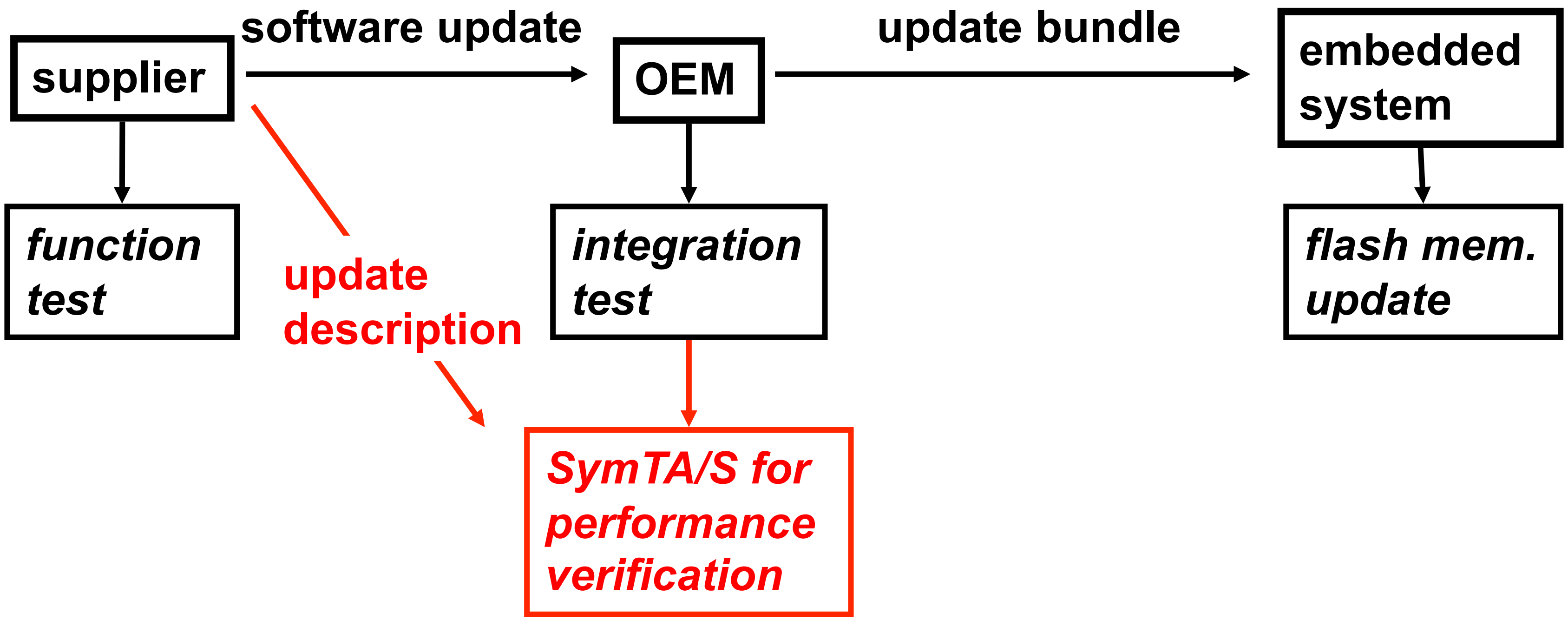
Introducing in-field design support

- alternative: support lab test by system protection mechanisms
  - move part of the design process to the field
  - establish in-field functions for safe updates and reconfigurations
- requirements
  - similar quality as lab based test
  - must be compatible to current design process
  - traceability, clear definition of responsibilities for cost assignment and clear liability

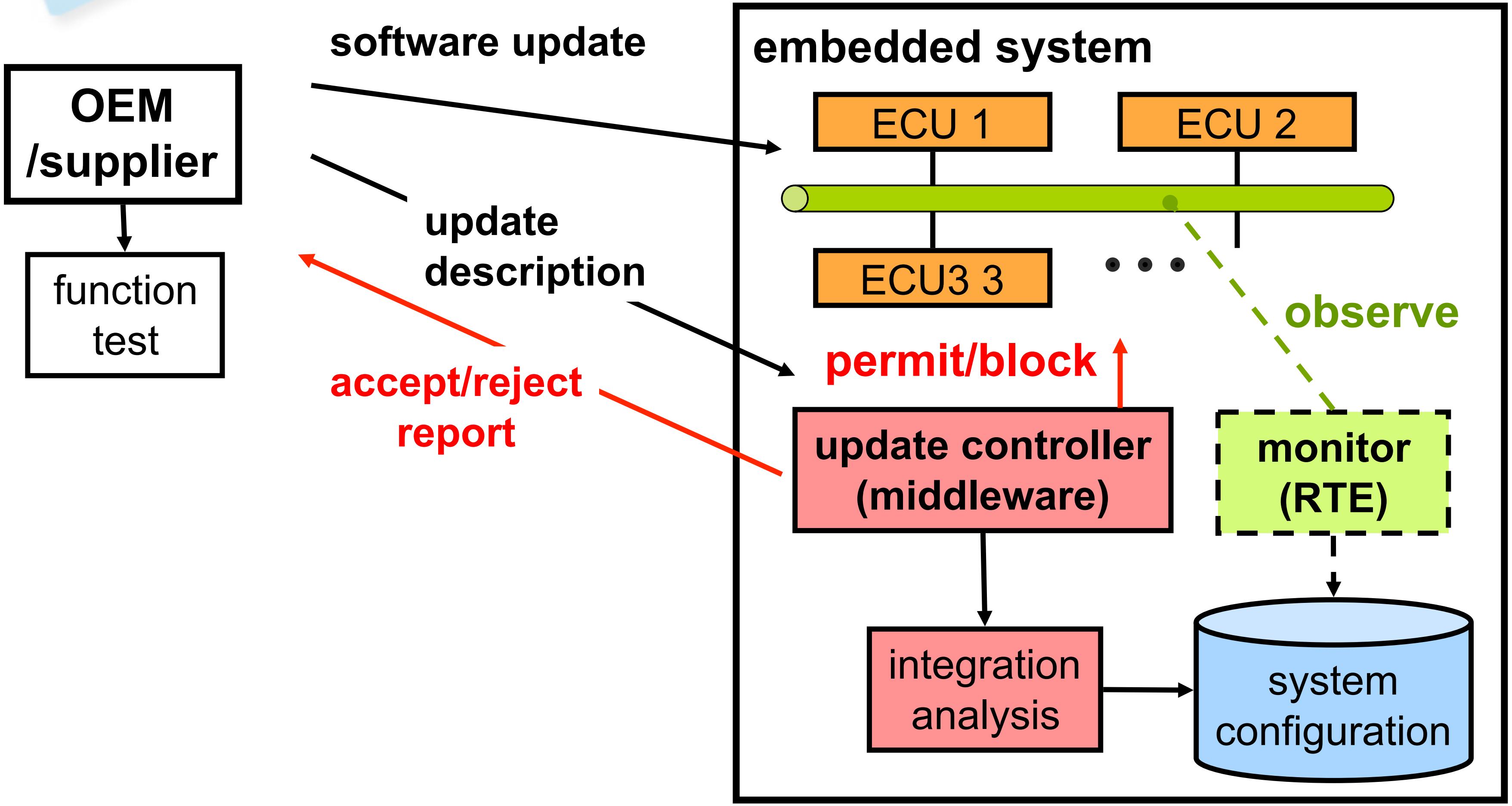
Update process today



SymTA/S enhanced update process



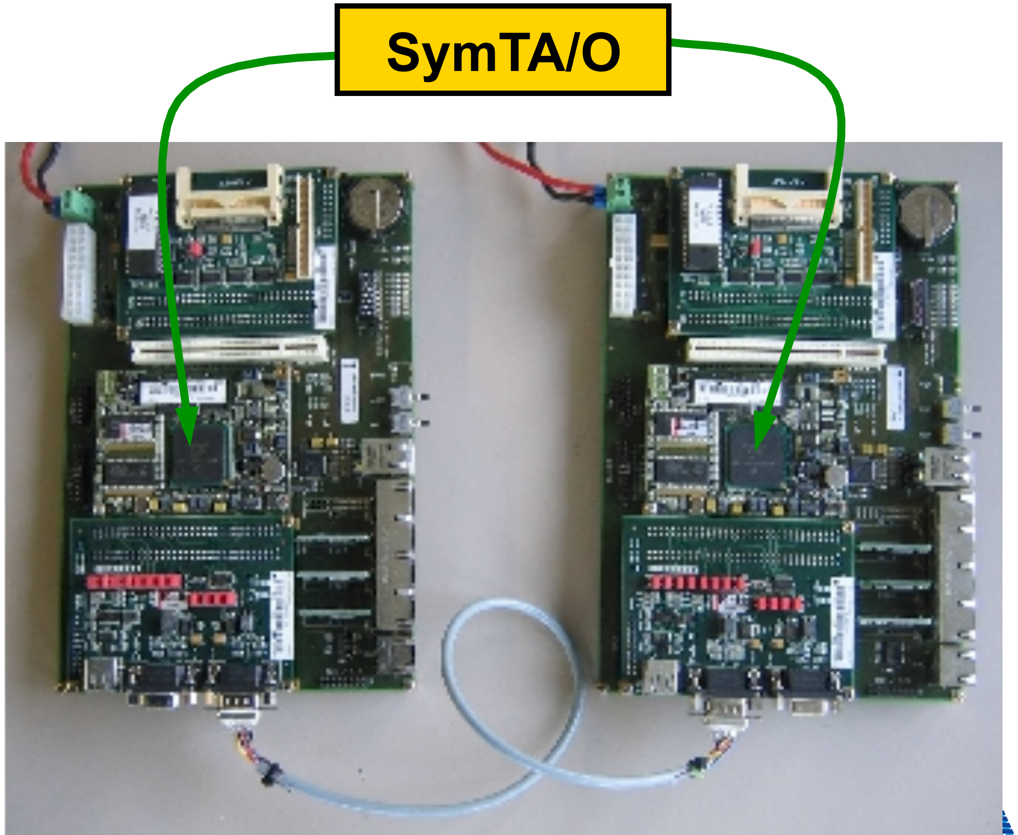
Proposed Update Process



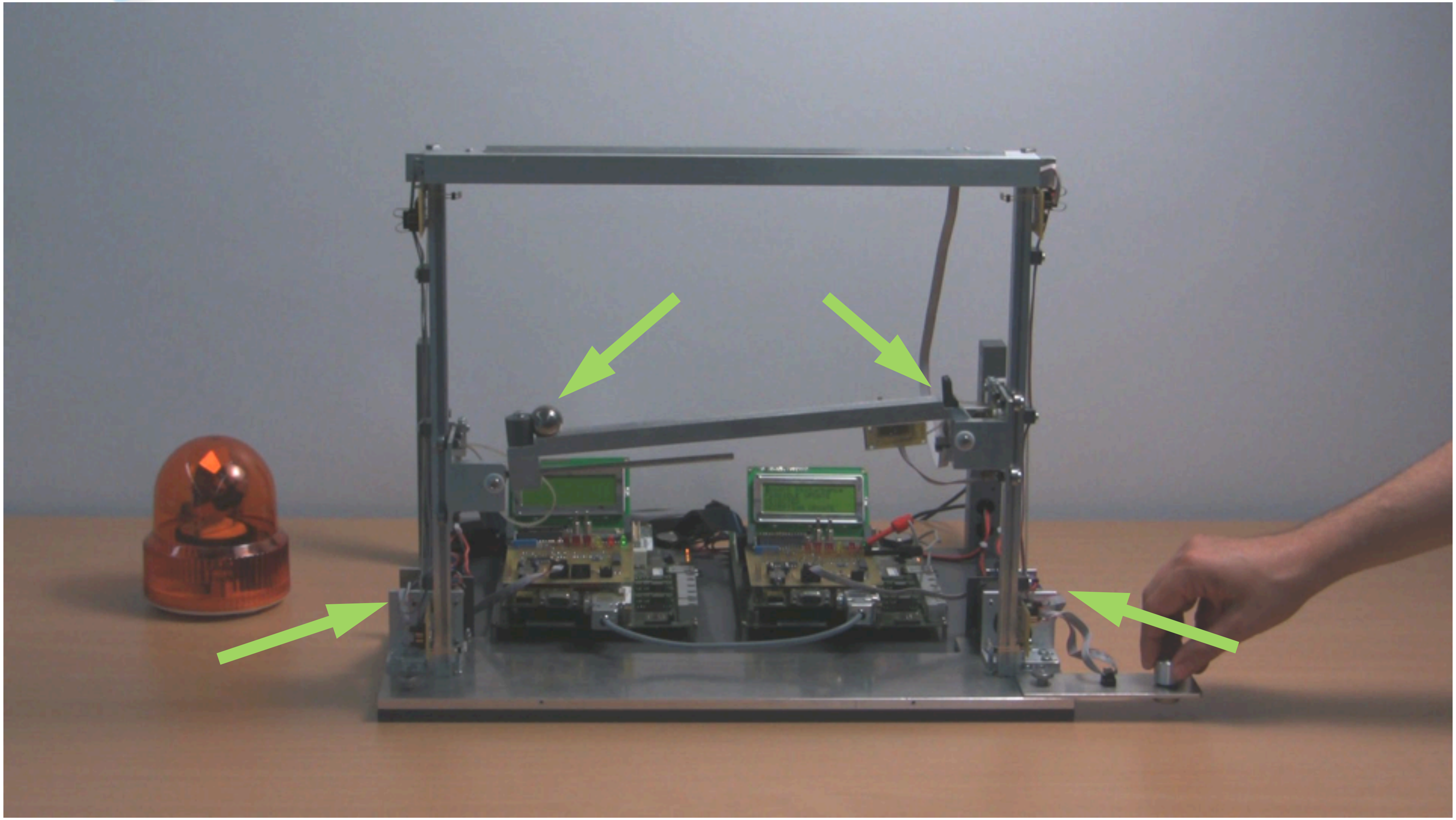
- system stores its configuration, description and applied changes
- enables in-field analysis and update tracing

EPOC project demonstrator

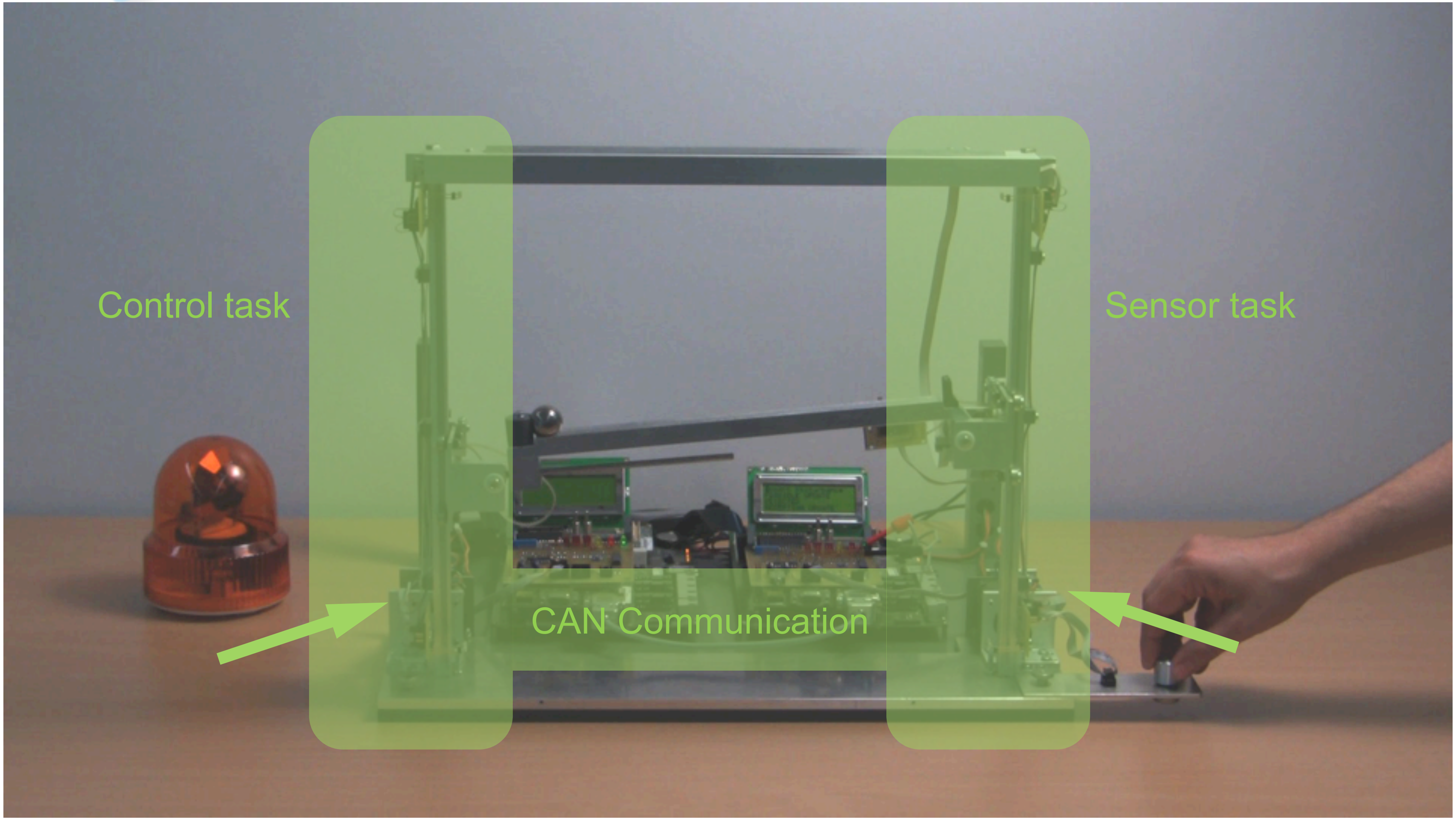
- „organic“ system based on PowerPC CAN-bus boards with Micro Kernel: µC/OS II
- includes network topology analysis, control plane
- supports update mechanism as above
- analysis with distributed SymTA/S (SymTA/O)
  - distributed time constrained fixed point solution alg.
- code size ca. 120 kB



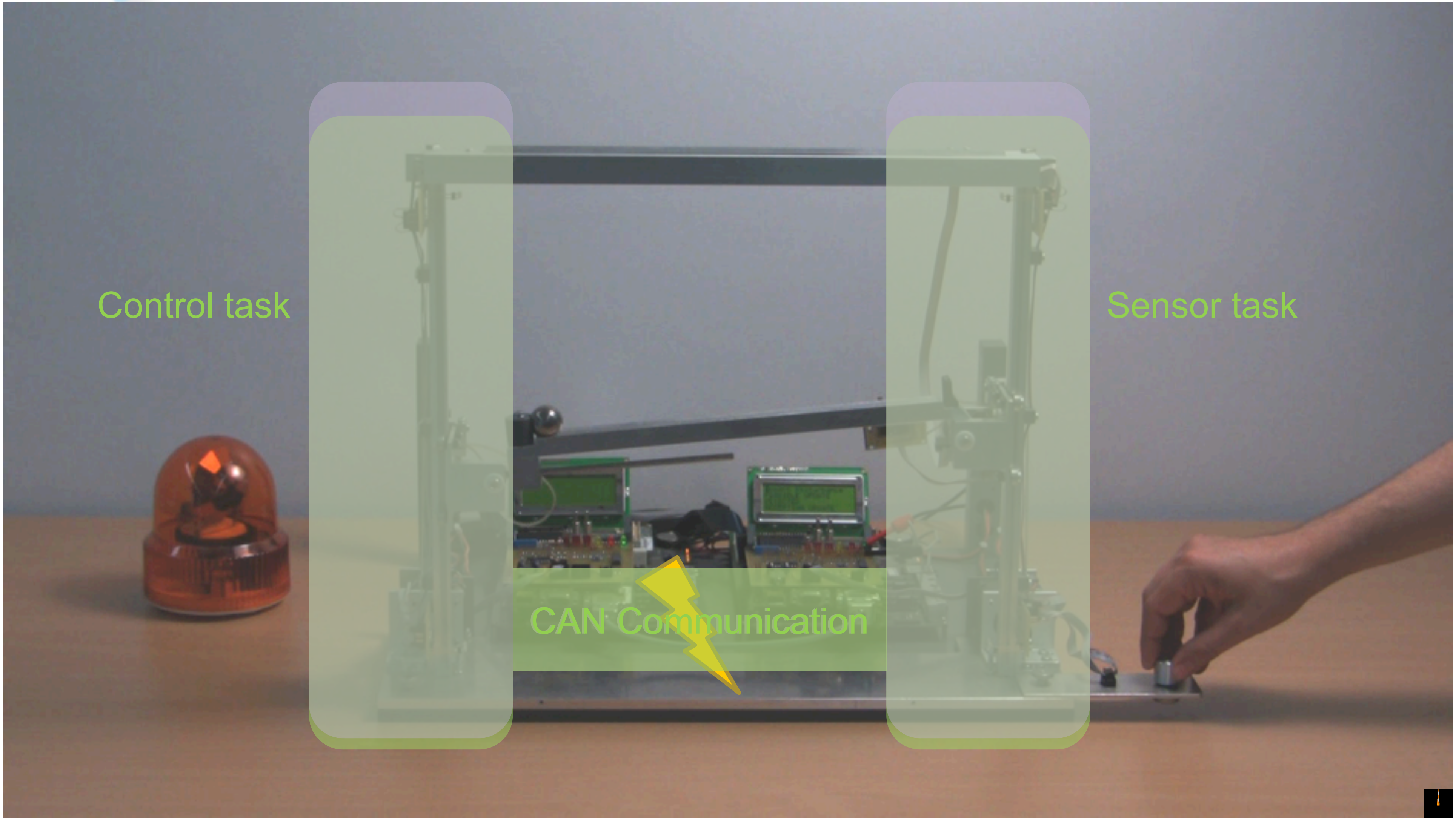
EPOC demonstrator



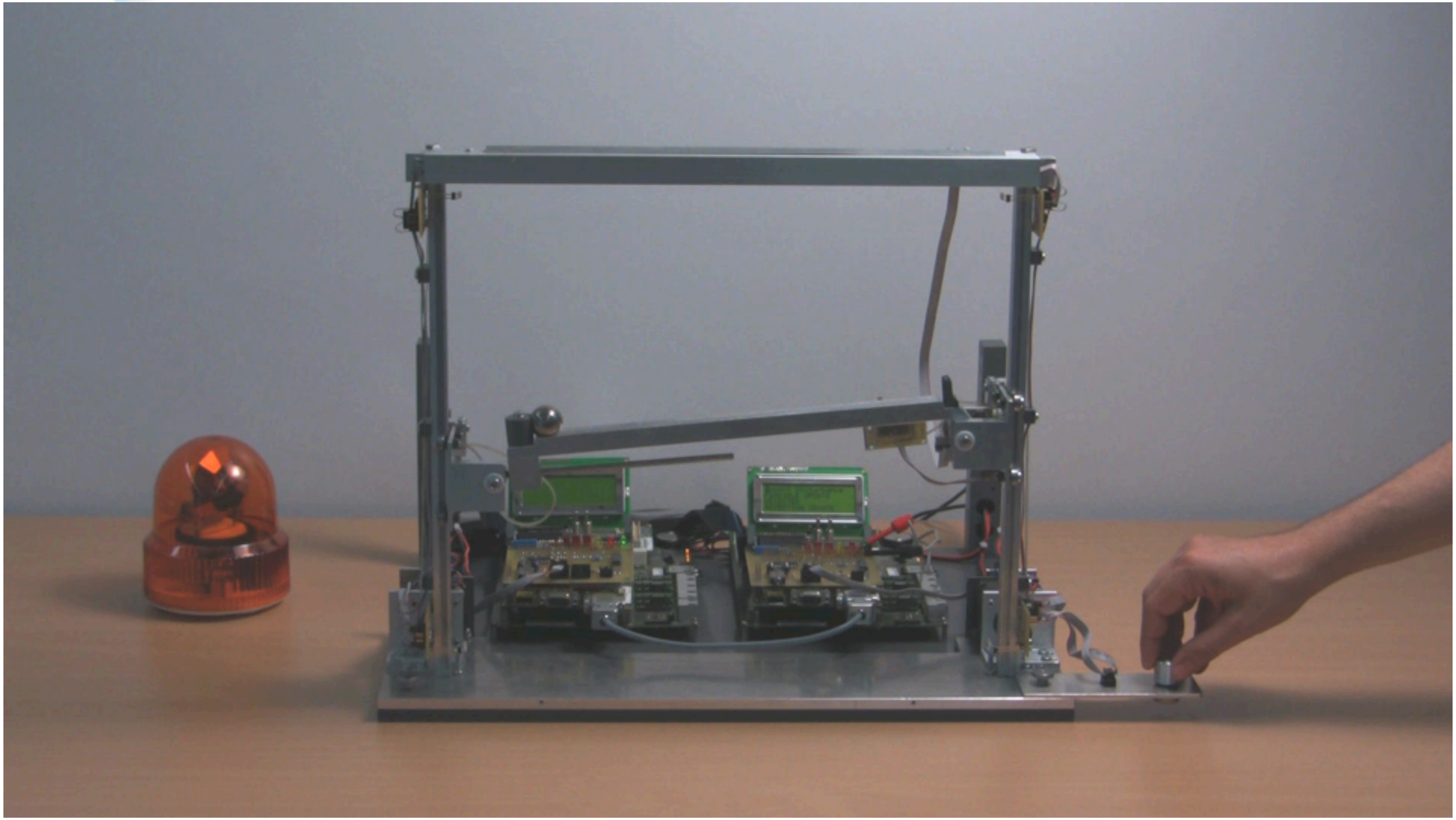
Demonstrator – Task distribution



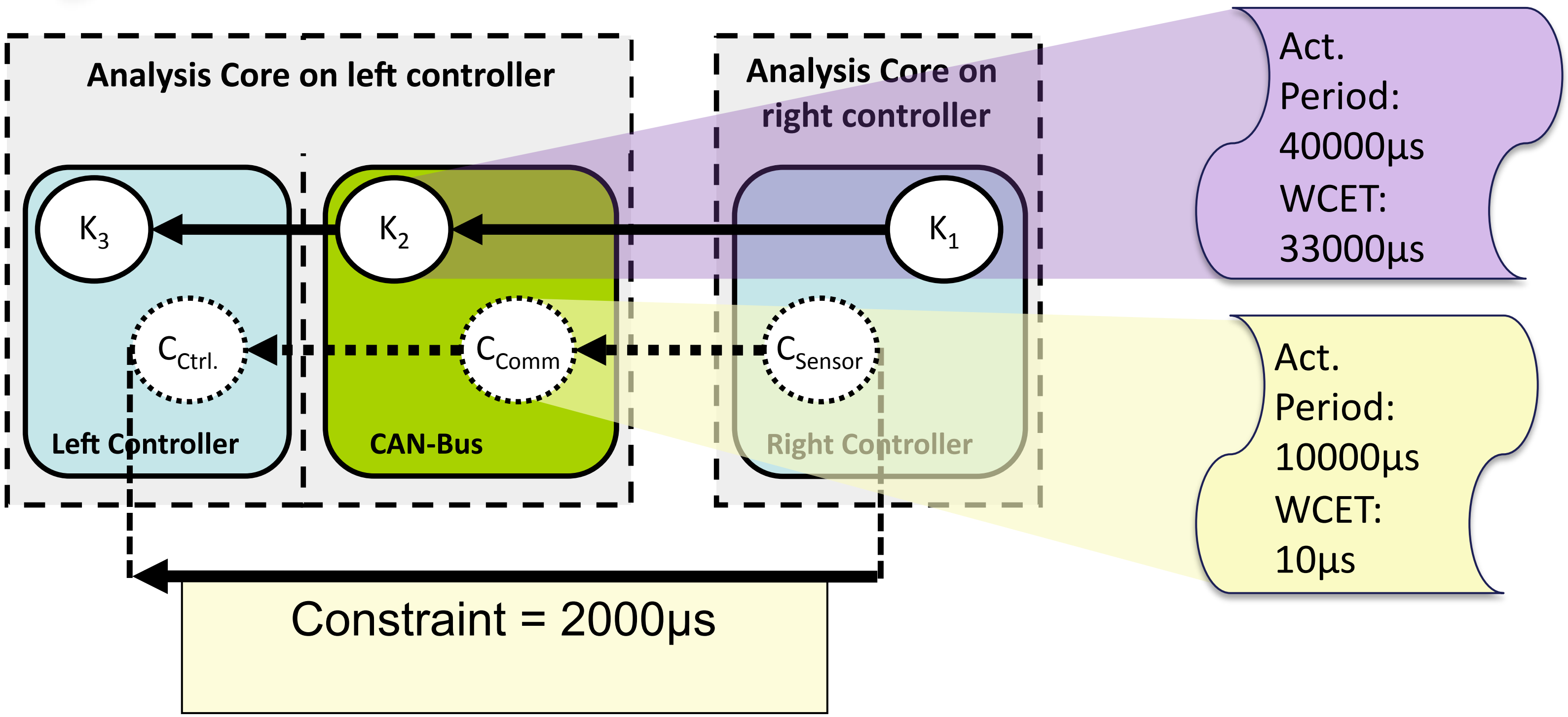
Demonstrator – Critical communication overload



Demonstrator without self-protection

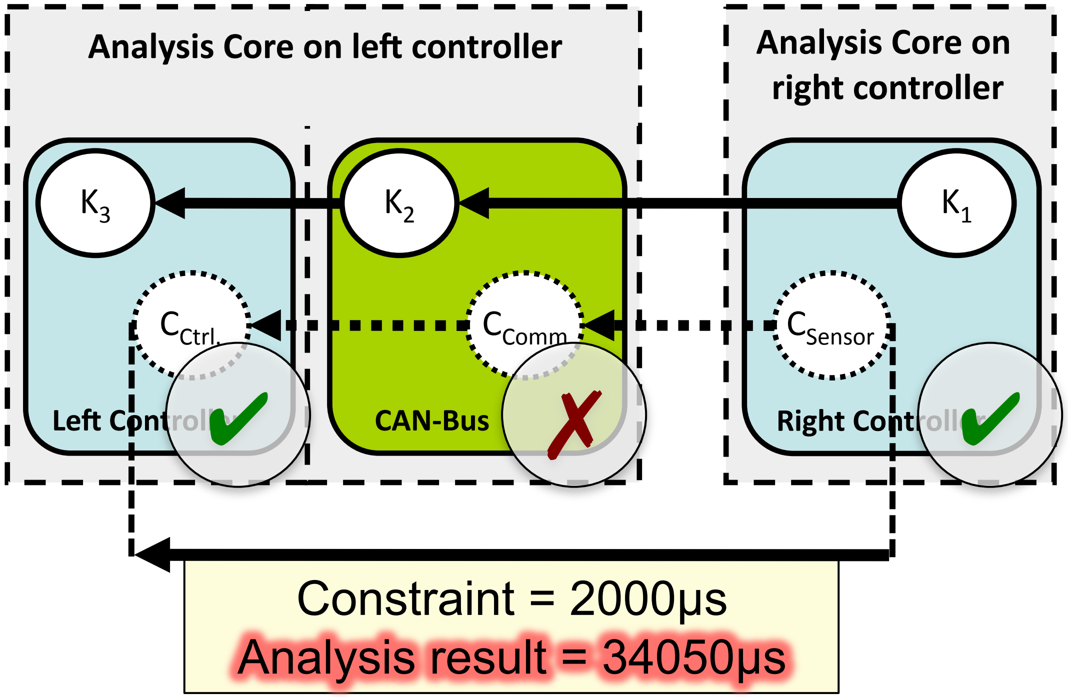


Demonstrator setup



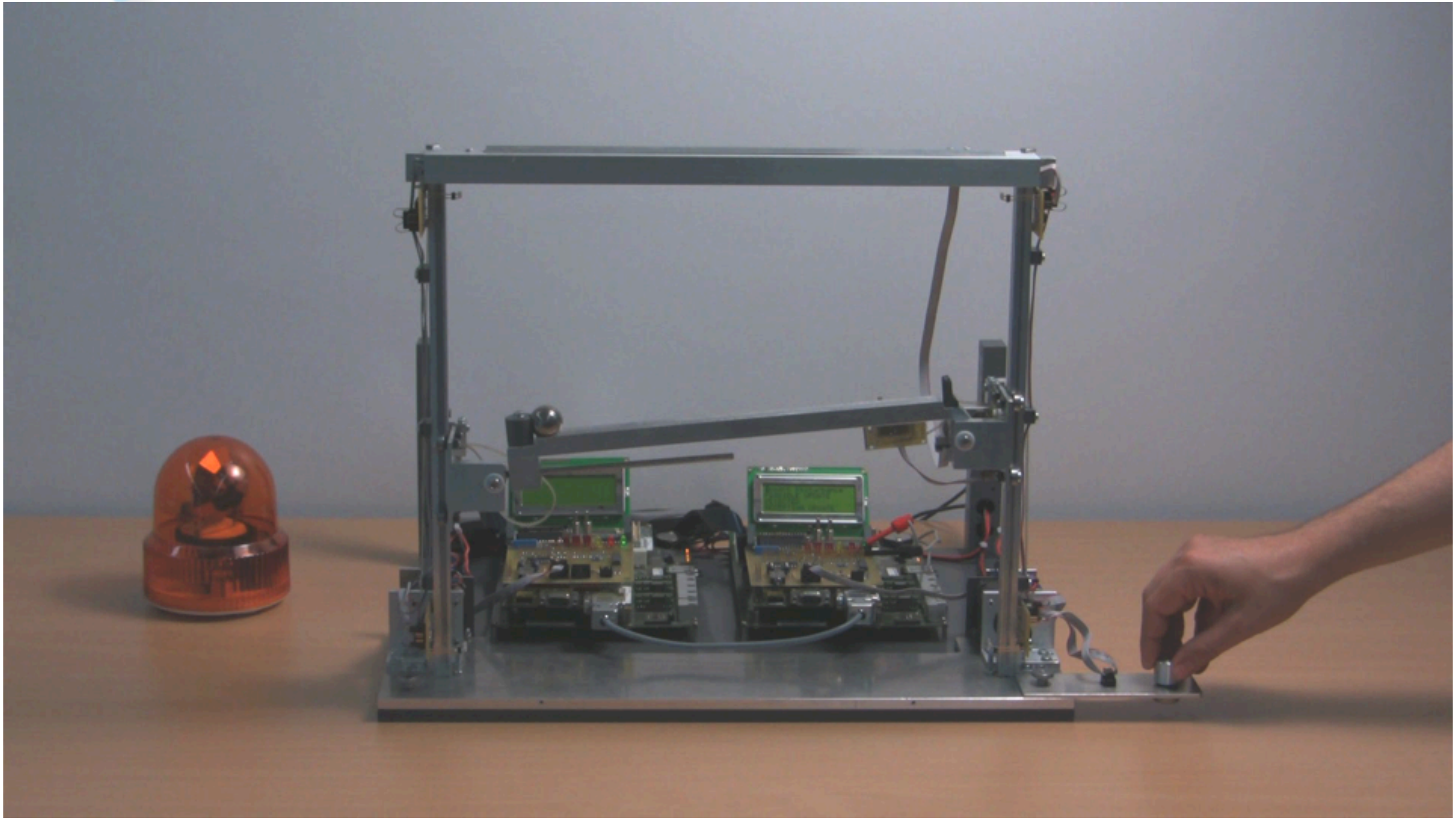
The second application with high priorities and high load on the CAN-Bus is inserted into the system.

Demonstrator with hazardous task



The third application would cause the control application to fail its latency constraint.
The third application is rejected.

Demonstrator with self-protection



Summary and outlook

- increasing design complexity and update processes suggest to move part of the design process to the field
- proposed safe update mechanism for in-field system integrity analysis – currently limited to performance as key aspect
- long-term goal: in-field analysis to support
  - safety critical function update and
  - safety critical systems autonomy
- current target applications: automotive, smart buildings

SEVENTH FRAMEWORK PROGRAMME

Overview

- motivation
- compositional performance analysis
- impact of performance on safety and analysis
  - networks
  - ECUs
- application to safety related system design
- system performance self-protection
- **conclusion**

Conclusion

- safety requirements are of growing importance in embedded system design
- performance is a key factor of safety
- performance analysis must therefore be extended to include error handling
- formal performance analysis with failure analysis is possible and can be compatible to safety standard requirements
  - first approaches have been presented
- system self-protection against performance failures can simplify design and updates of complex systems and can support autonomous safety critical systems

Acknowledgements

- the demonstrator has been developed by Steffen Stein, Moritz Neukirchner and Dr.-Ing. Harald Schrom, supported by several students
  - the video was prepared by Philip Axer
- Maurice Sebastian and Philip Axer have helped to develop the slides