Real-time Systems Engineering @ Bosch

Dr. Arne Hamann, Robert Bosch GmbH

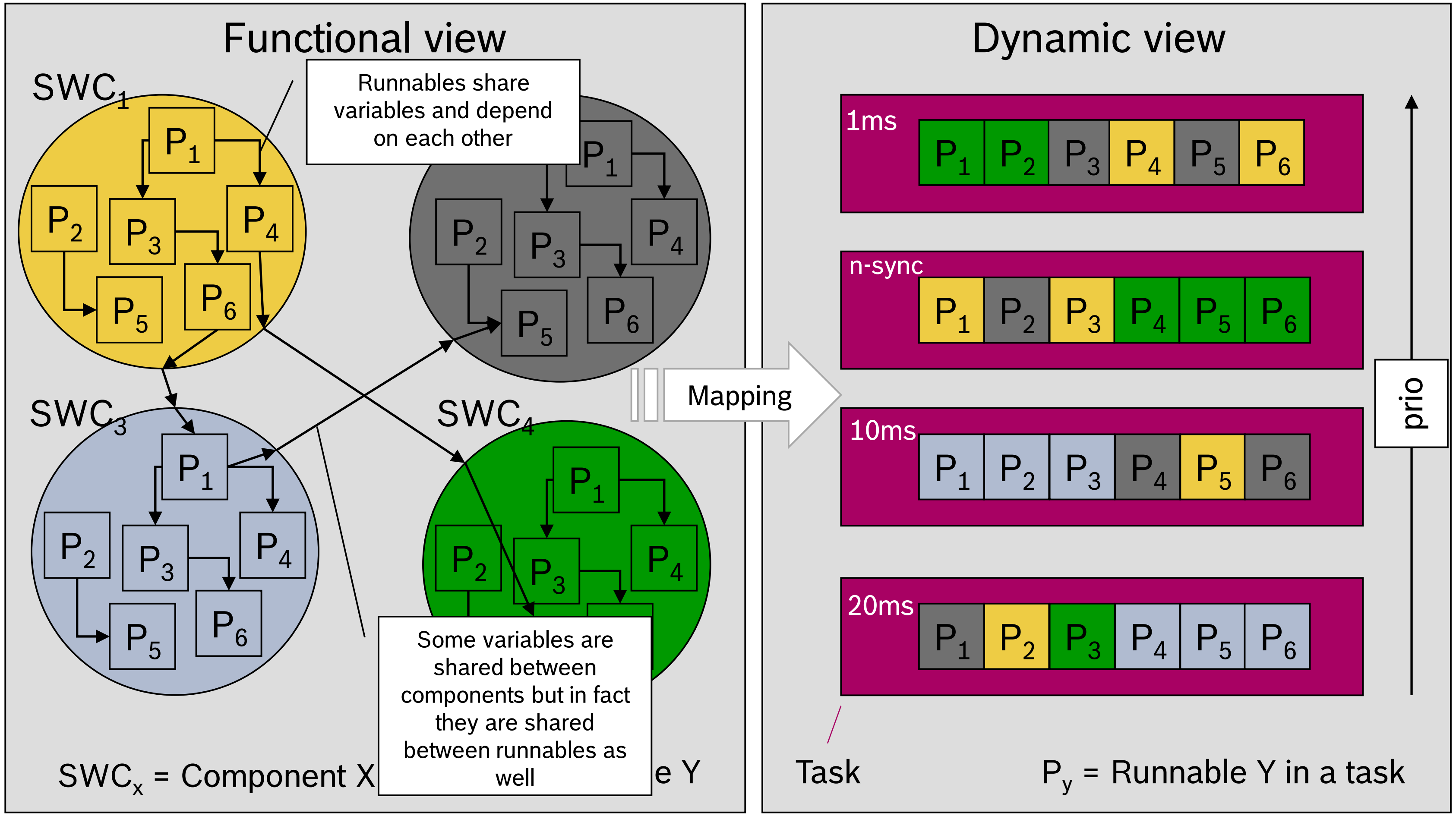
Real-time Systems Engineering @ Bosch
Outline

- Performance modeling & analysis of classical automotive systems
  - Motivation ... or the real complexity
  - Amalthea performance model
  - Current usage @ Bosch
  - Upcoming challenges
- Communication centric design in multi-core systems
  - Importance of cause-effect chains
  - Issues with concurrent execution in multi-core systems
  - Communication mechanisms as solution & impact on latencies
  - Experiments
- Timing-aware control design
  - Control and real-time systems engineering – two worlds collide
  - Co-engineering approach
  - Example

Real-time Systems Engineering @ Bosch
Outline

- **Performance modeling & analysis of classical automotive systems**
  - **Motivation … or the real complexity**
  - Amalthea performance model
  - Current usage @ Bosch
  - Upcoming challenges
- Communication centric design in multi-core systems
  - Importance of cause-effect chains
  - Issues with concurrent execution in multi-core systems
  - Communication mechanisms as solution & impact on latencies
  - Experiments
- Timing-aware control design
  - Control and real-time systems engineering – two worlds collide
  - Co-engineering approach
  - Example

Performance modeling & analysis of classical automotive systems
Typical Classic Automotive Software Architecture Pattern

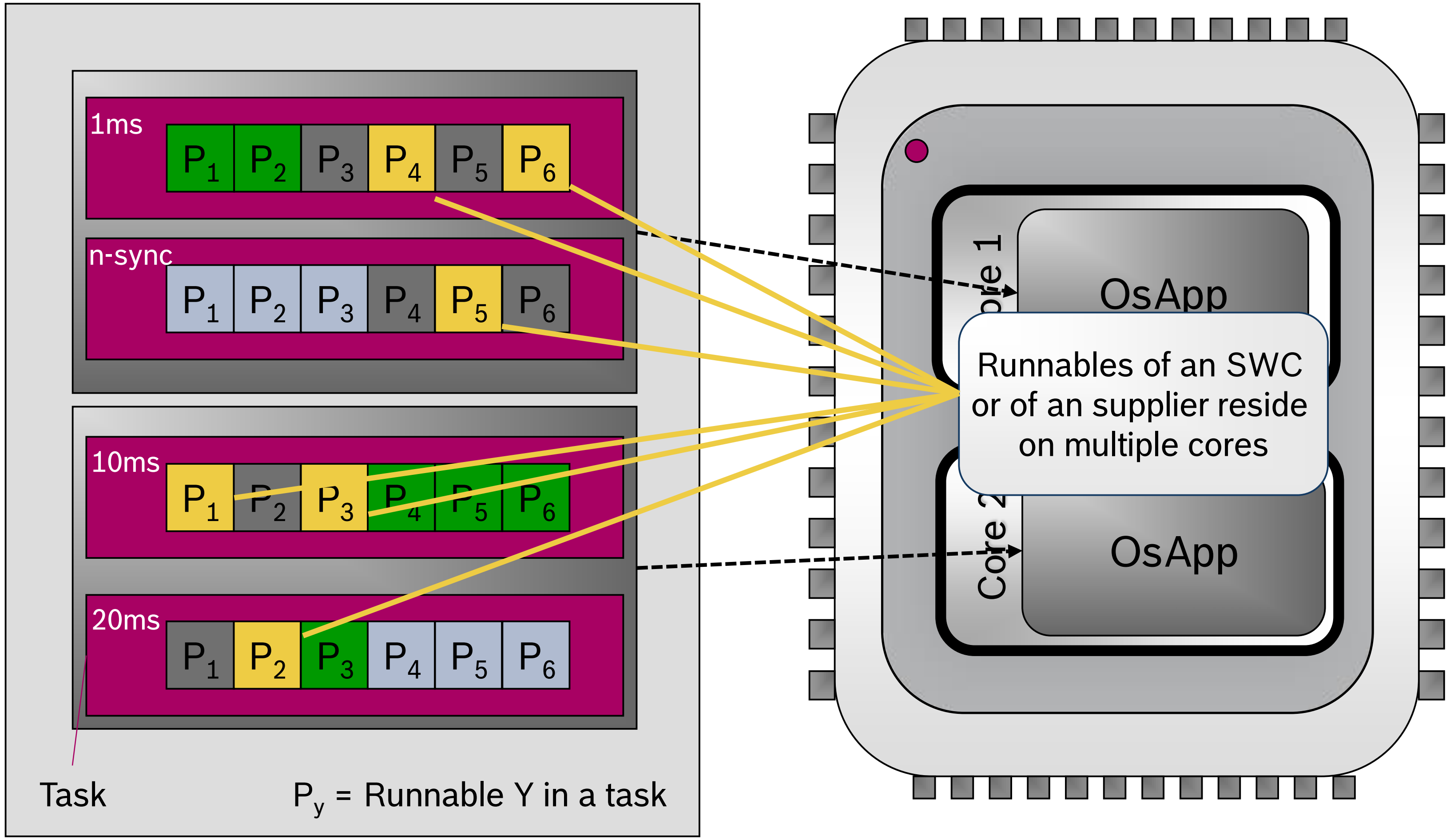

Functional view
SWC_1
SWC_3
SWC_4
P_1 P_2 P_3 P_4 P_5 P_6
Runnables share variables and depend on each other
Some variables are shared between components but in fact they are shared between runnables as well
SWC_x = Component X
e Y
Mapping
Dynamic view
1ms
n-sync
10ms
20ms
prio
Task
P_y = Runnable Y in a task

Dr. Arne Hamann | 2019-01-31

BOSCH

Performance modeling & analysis of classical automotive systems

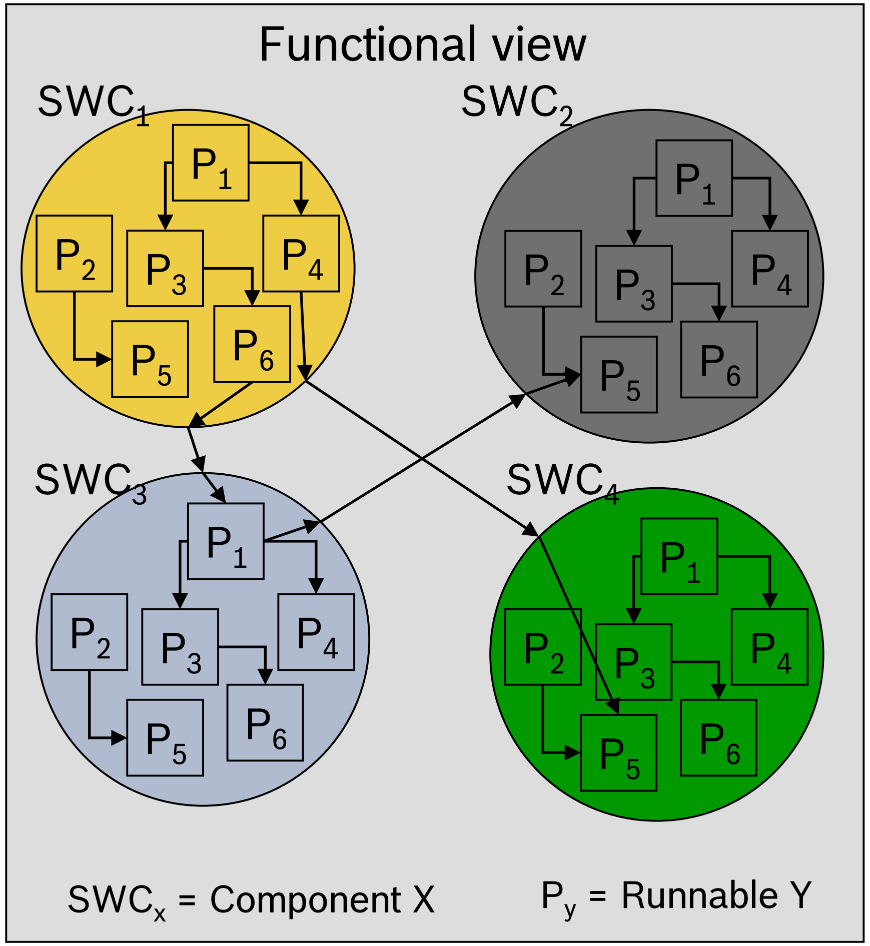
Typical Distribution Pattern – Task-Level Parallelism

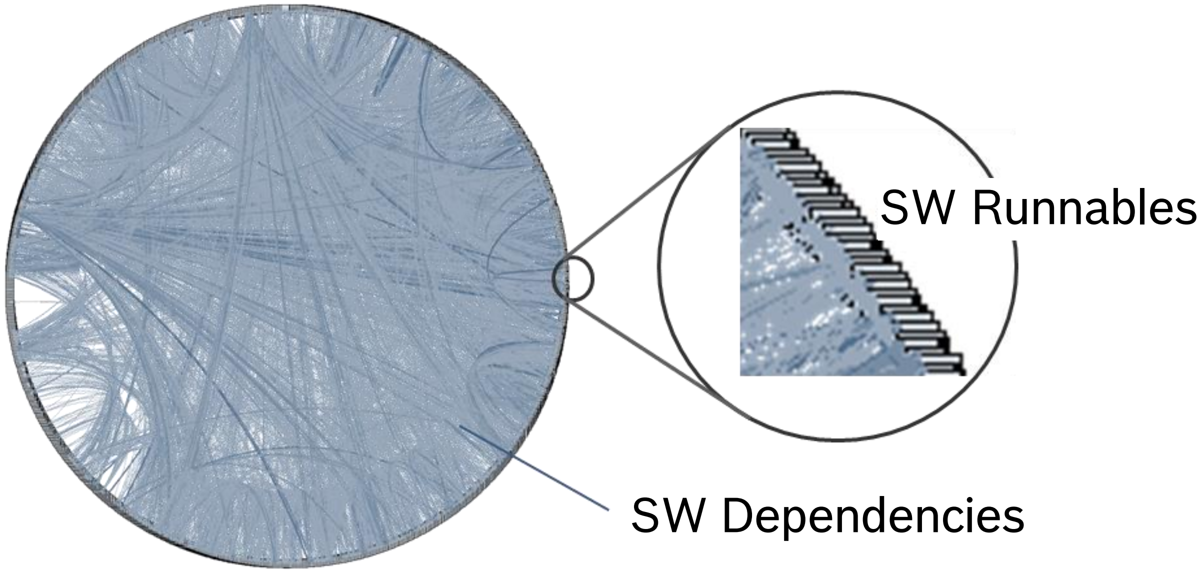


1ms
n-sync
10ms
20ms
P_1 P_2 P_3 P_4 P_5 P_6
Task
P_y = Runnable Y in a task
Core 1
Core 2
OsApp
OsApp
Runnables of an SWC or of an supplier reside on multiple cores

Performance modeling & analysis of classical automotive systems

The Real Complexity…



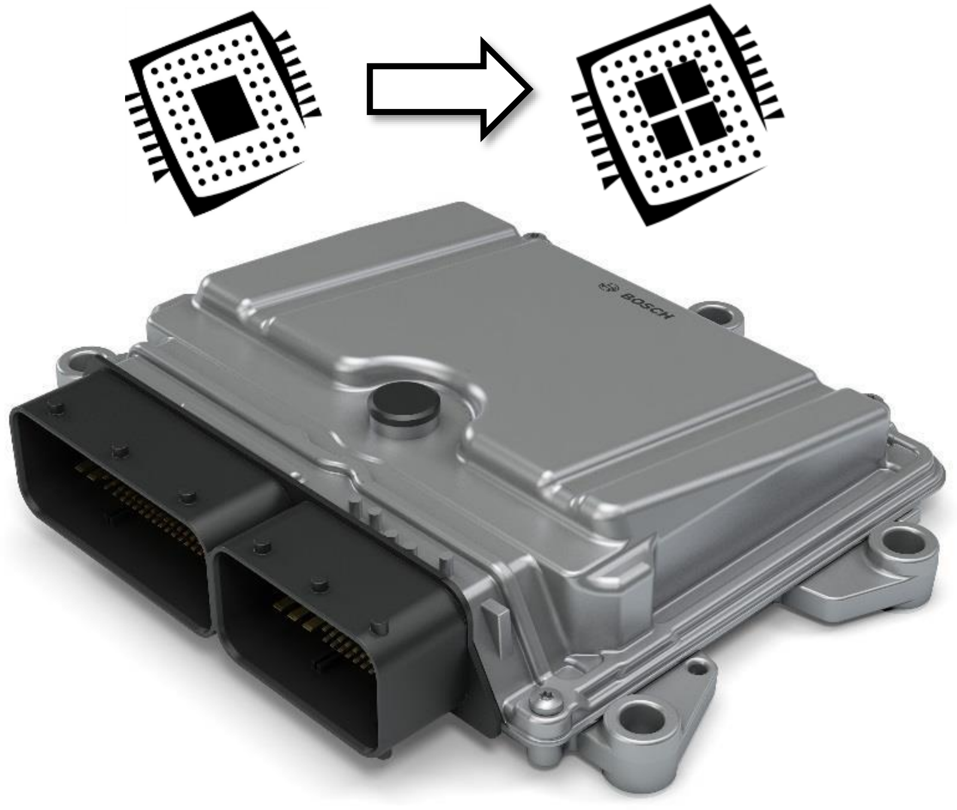


Fine-grain, legacy SW sharing between OEM and Tier1 with multiple dependencies

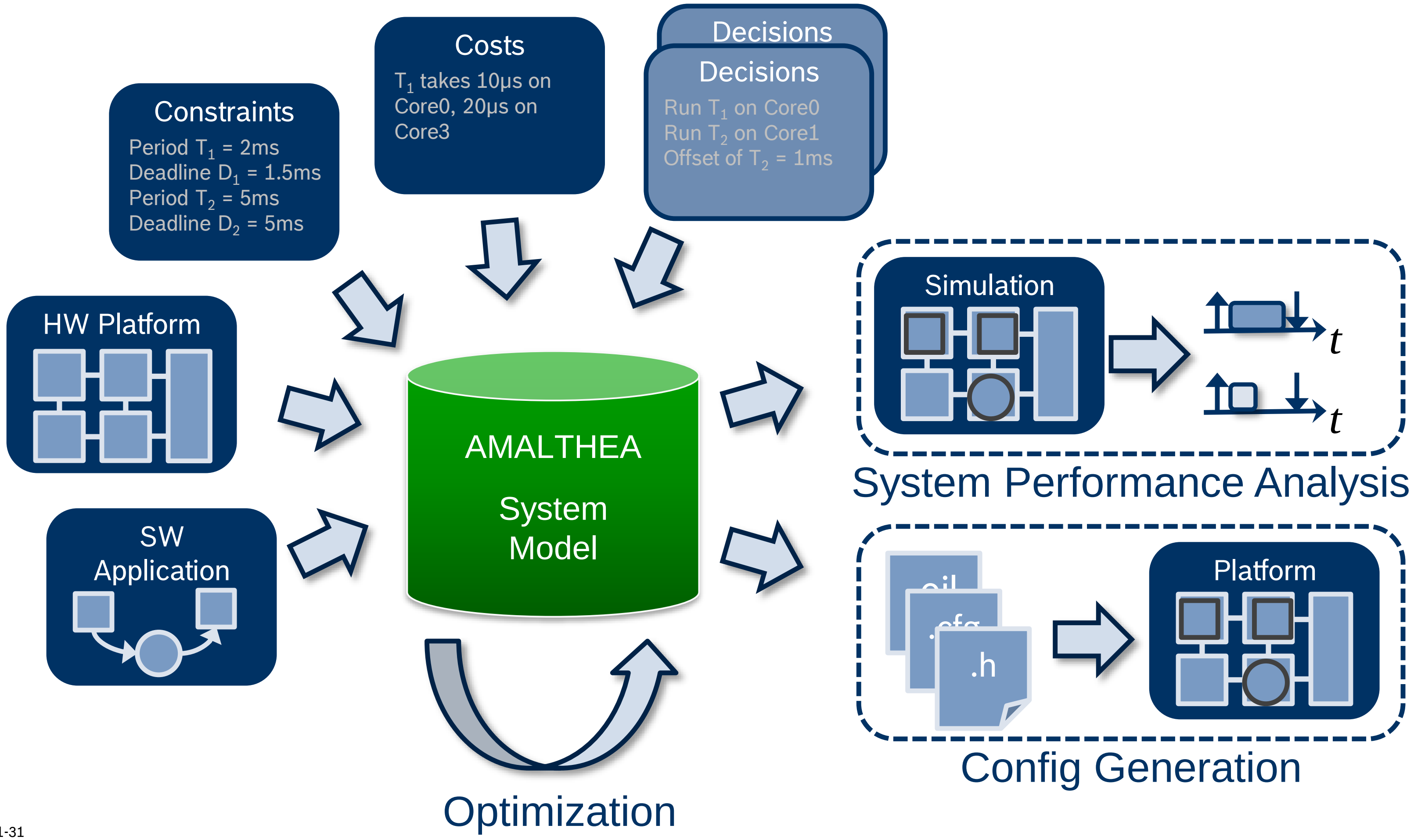
Performance modeling & analysis of classical automotive systems
Tasks to solve during migration to multi-core…

- Maintain single-core dependencies
- Ensure data consistency
- Balance core load
- Optimize memory placement of variables
- Bound latency of cause-effect chains
- …

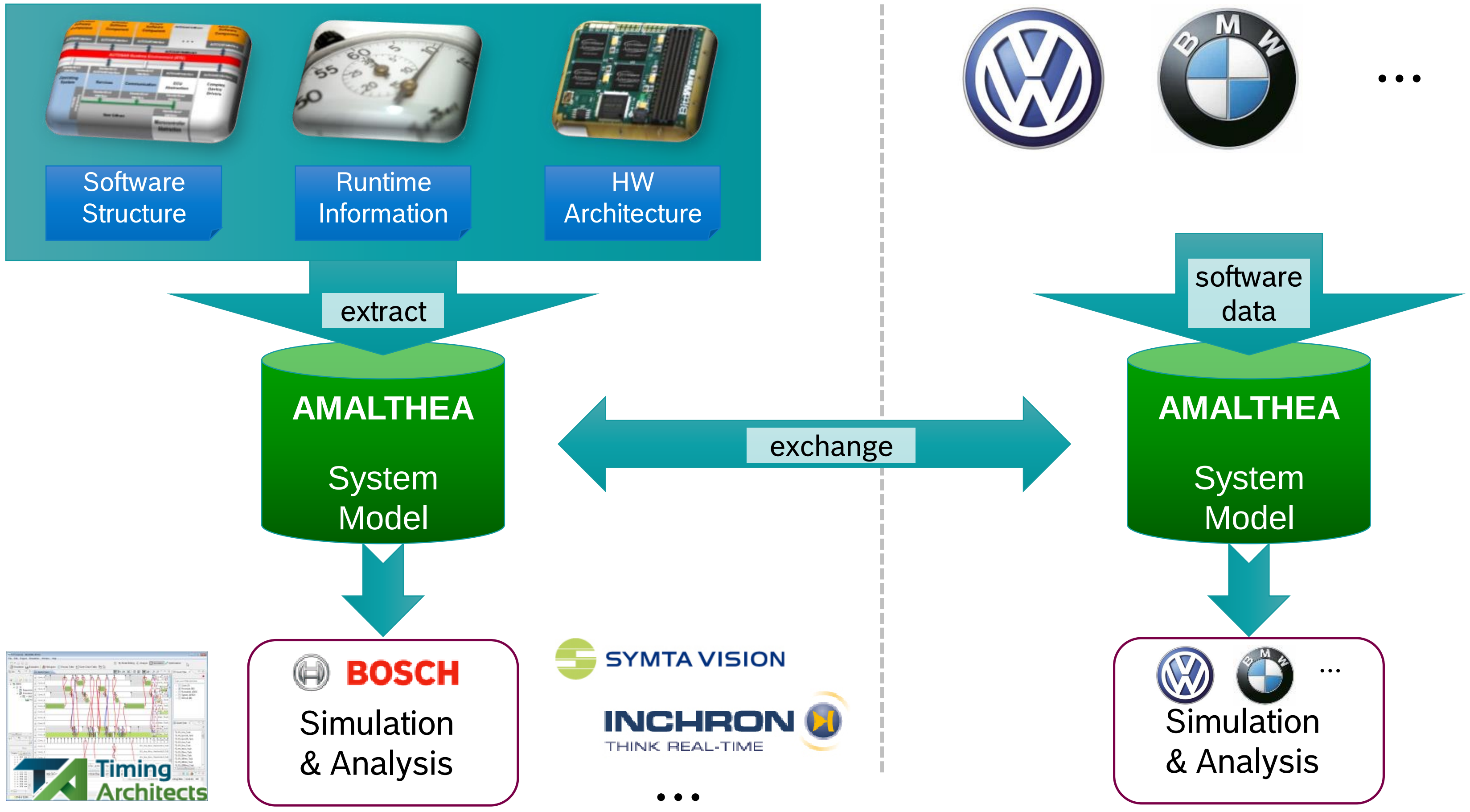
- **Need a model capturing the system aspects required to solve those tasks**



Performance modeling & analysis of classical automotive systems
The Basic Idea



Performance modeling & analysis of classical automotive systems
Suitable abstraction level needed



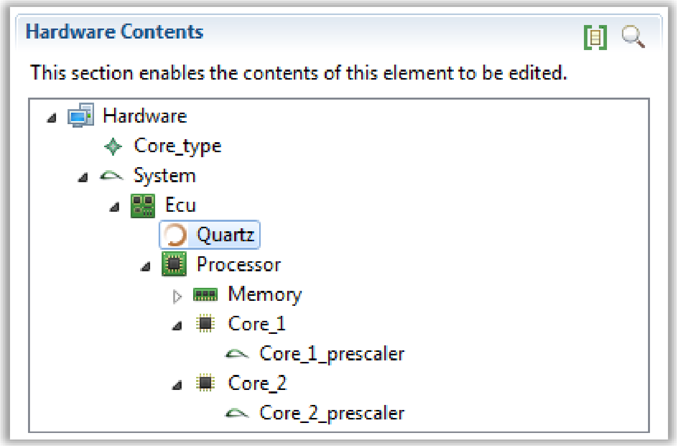
Real-time Systems Engineering @ Bosch

Outline

Performance modeling & analysis of classical automotive systems
AMALTHEA Model – Hardware

HW Platform

- **Hardware elements**
  - ECU
  - Microcontroller
  - Core
  - Memory
  - Network

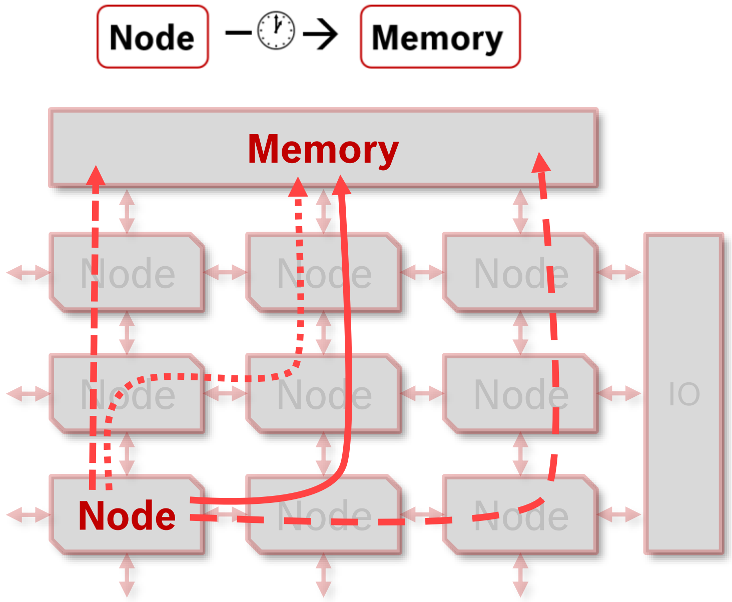


Hardware Contents

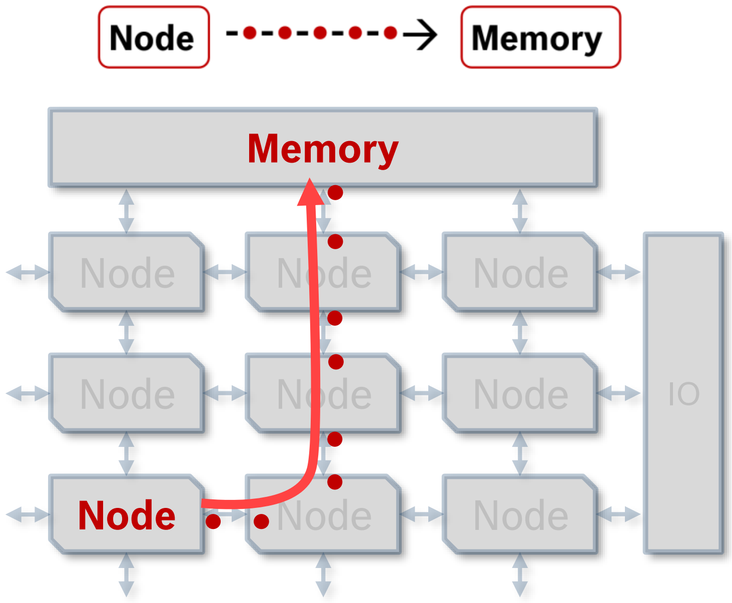
This section enables the contents of this element to be edited.

```
Hardware
  Core_type
  System
    Ecu
      Quartz
      Processor
        Memory
        Core_1
          Core_1_prescaler
        Core_2
          Core_2_prescaler
```

- Latency Access Path



- Hardware Access Path



Performance modeling & analysis of classical automotive systems

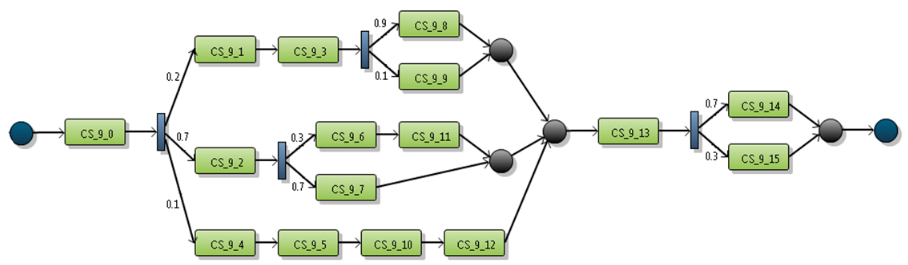
AMALTHEA Model – Software

SW Application

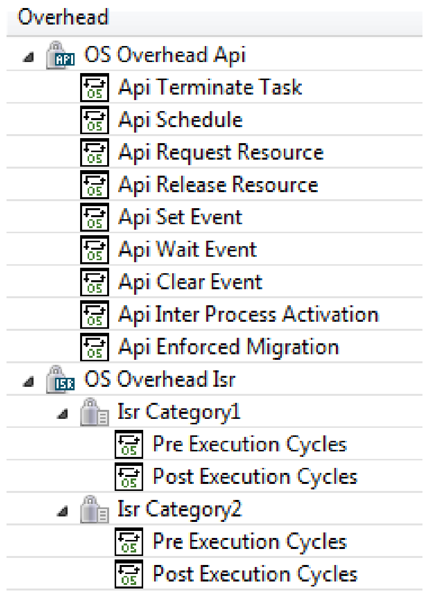
Costs

T_1 takes 10µs on Core0, 20µs on Core3

- **Software behavior**
  - Tasks, runnables, schedulers, …
  - Description on different levels of abstraction
  - Runnables characterized by
    - Execution time (distribution)
    - Variable access (distribution)
  - Detailed (probabilistic) call sequences possible



- **Operating System behavior**

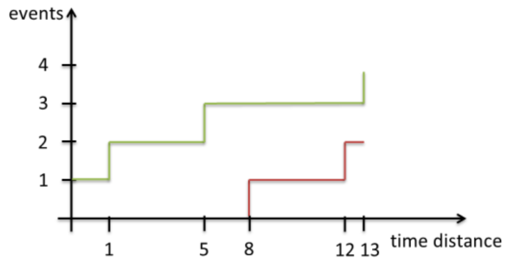


Overhead
- OS Overhead Api
  - Api Terminate Task
  - Api Schedule
  - Api Request Resource
  - Api Release Resource
  - Api Set Event
  - Api Wait Event
  - Api Clear Event
  - Api Inter Process Activation
  - Api Enforced Migration
- OS Overhead Isr
  - Isr Category1
    - Pre Execution Cycles
    - Post Execution Cycles
  - Isr Category2
    - Pre Execution Cycles
    - Post Execution Cycles

Performance modeling & analysis of classical automotive systems
AMALTHEA Model – Constraints

Constraints
Period T_1 = 2ms
Deadline D_1 = 1.5ms
Period T_2 = 5ms
Deadline D_2 = 5ms

- **Runnable Sequencing Constraints**
- **Timing Constraints**
  - Order Constraint
  - Synchronization Constraint
  - Repetition Constraint
  - Delay Constraint
  - Age Constraint
  - Reaction Constraint
- **Data Age Constraints**
- **Arrival Curves**



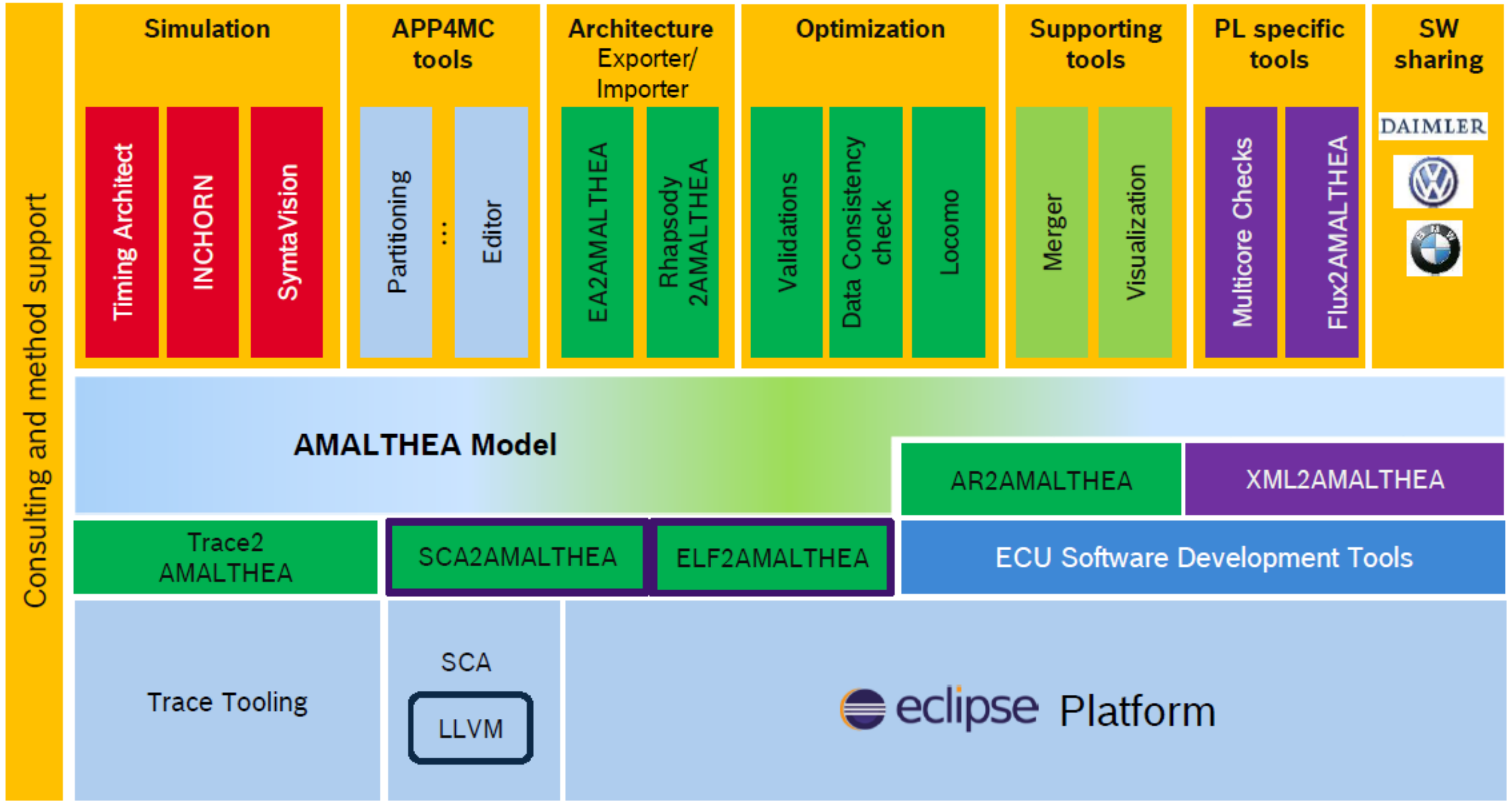
- **Mapping Constraints**
  - Pairing Constraints
  - Separation Constraints
- **Property Constraints**

BOSCH

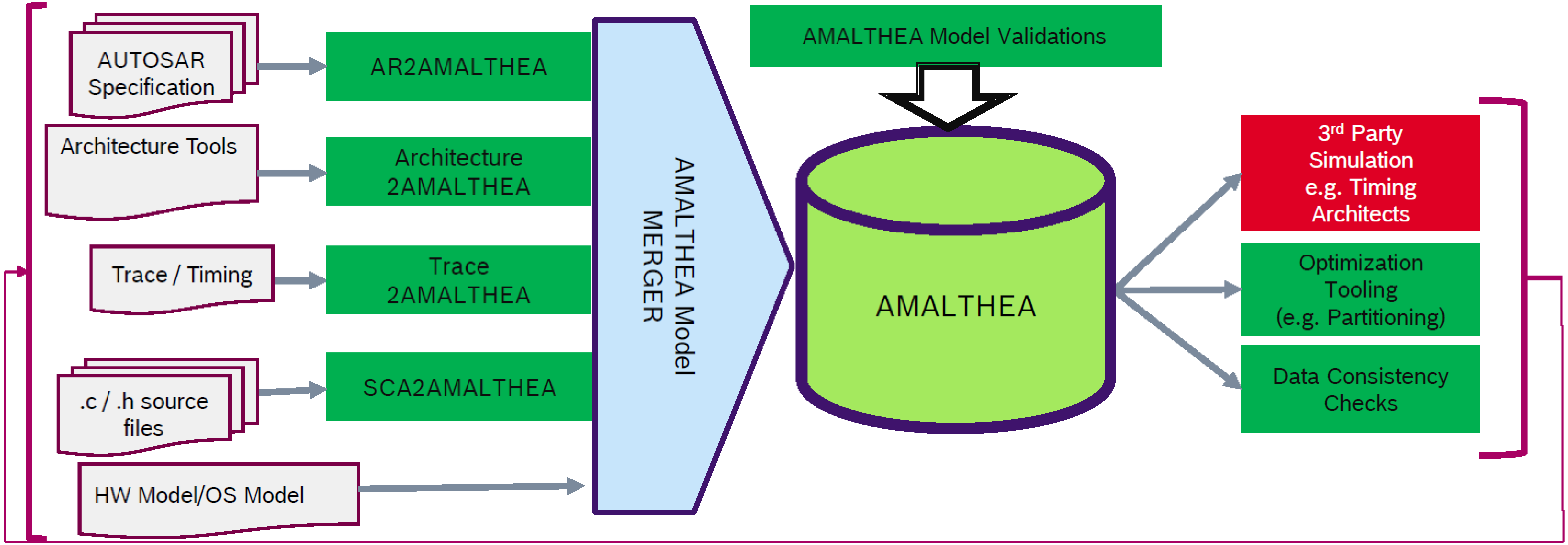
Real-time Systems Engineering @ Bosch
Outline

- Performance modeling & analysis of classical automotive systems
  - Motivation ... or the real complexity
  - Amalthea performance model
  - **Current usage @ Bosch**
  - Upcoming challenges
- Communication centric design in multi-core systems
  - Importance of cause-effect chains
  - Issues with concurrent execution in multi-core systems
  - Communication mechanisms as solution & impact on latencies
  - Experiments
- Timing-aware control design
  - Control and real-time systems engineering – two worlds collide
  - Co-engineering approach
  - Example

Performance modeling & analysis of classical automotive systems
PLAT4MC - Multicore Tool Platform @ Bosch Series Production

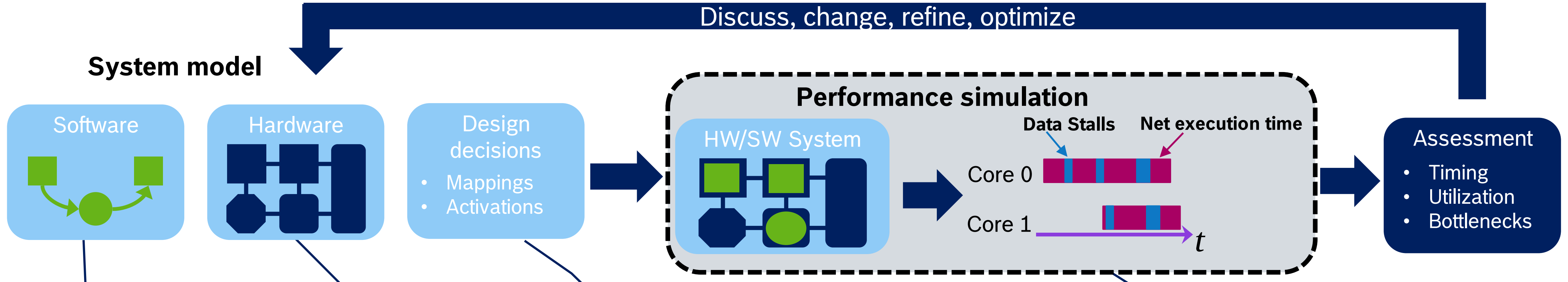


Performance modeling & analysis of classical automotive systems
PLAT4MC - Automated AMALTHEA Model Generation



Performance modeling & analysis of classical automotive systems

Status Quo of Performance Simulation



Application structure, execution cycles, data sizes, communication,

Components, frequency, performance factor, access latencies

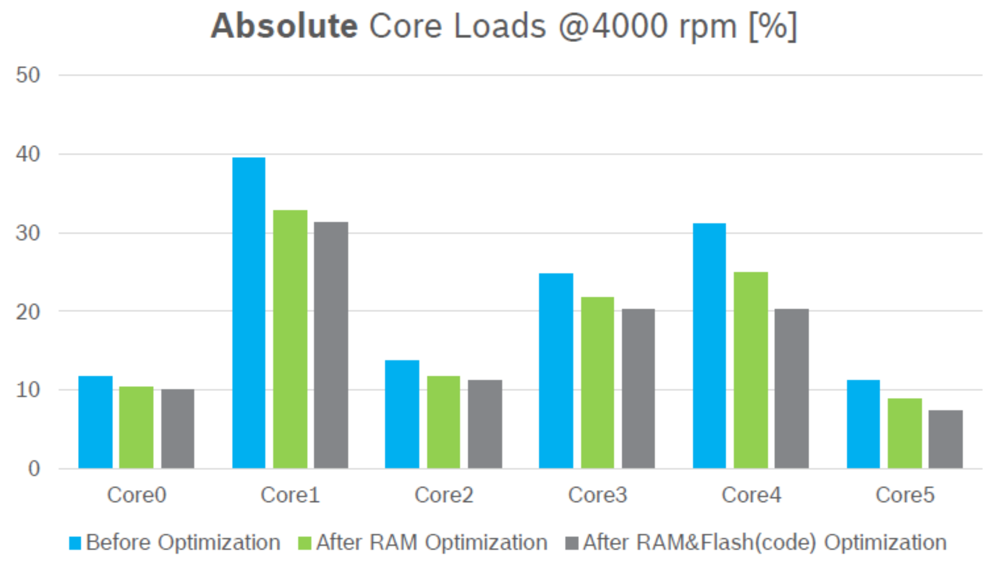
Mapping of scheduler to execution components, task activations

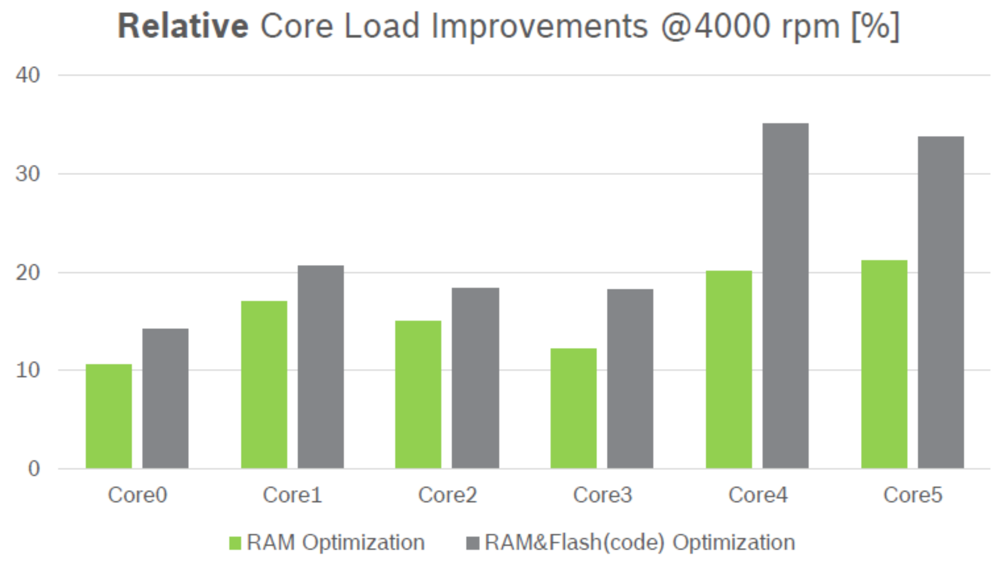
Event based simulation of dynamic effects, using net execution times of functions + data stalls

Current modelling approach suited for (homogeneous) many-core architectures

Performance modeling & analysis of classical automotive systems
PLAT4MC - Example Use Case: Memory Optimization

- Optimize placement of variables and code (in system and core local memories) in order to improve execution time load of cores (for a fixed task to core mapping)
- Considers allocation constraints (White List, Black List) as well as call/access statistics

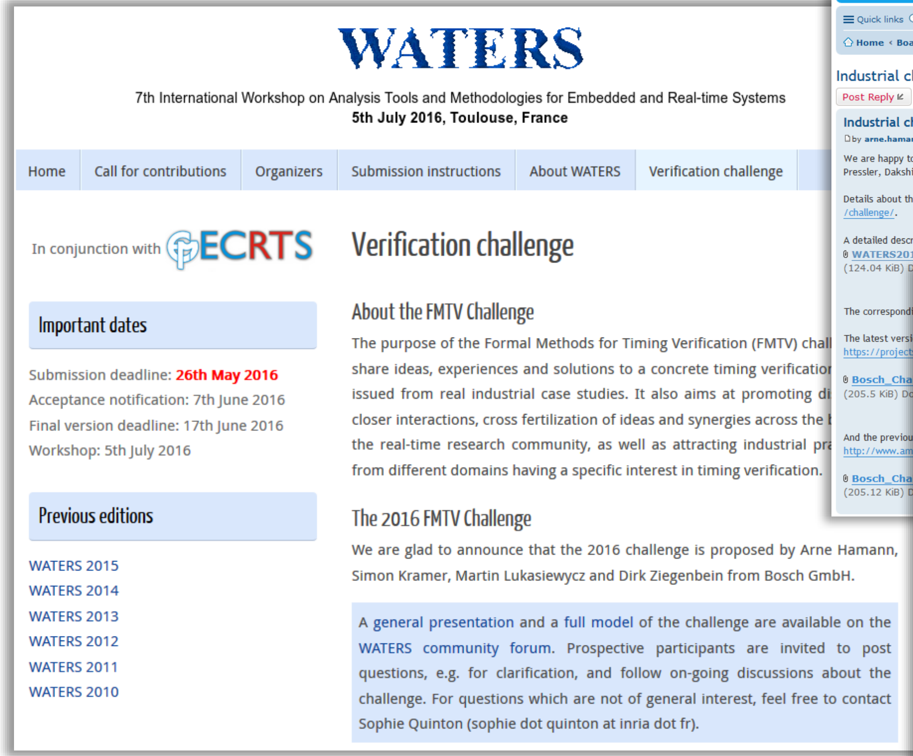




Aurix2G 6 cores

Performance modeling & analysis of classical automotive systems
Basis for WATERS Industrial Challenges 2016/17

https://waters2017.inria.fr/challenge/



WATERS

7th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems
5th July 2016, Toulouse, France

Home | Call for contributions | Organizers | Submission instructions | About WATERS | Verification challenge

In conjunction with ECRTS

Important dates

Submission deadline: 26th May 2016
Acceptance notification: 7th June 2016
Final version deadline: 17th June 2016
Workshop: 5th July 2016

Previous editions

WATERS 2015
WATERS 2014
WATERS 2013
WATERS 2012
WATERS 2011
WATERS 2010

Verification challenge

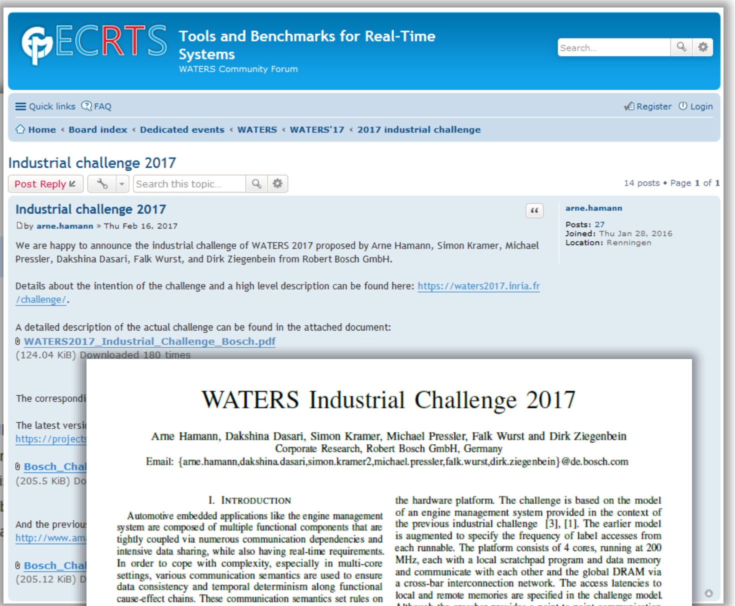
About the FMTV Challenge

The purpose of the Formal Methods for Timing Verification (FMTV) chall
share ideas, experiences and solutions to a concrete timing verificatio
issued from real industrial case studies. It also aims at promoting di
closer interactions, cross fertilization of ideas and synergies across the
the real-time research community, as well as attracting industrial pra
from different domains having a specific interest in timing verification.

The 2016 FMTV Challenge

We are glad to announce that the 2016 challenge is proposed by Arne Hamann, Simon Kramer, Martin Lukasiewycz and Dirk Ziegenbein from Bosch GmbH.

A general presentation and a full model of the challenge are available on the WATERS community forum. Prospective participants are invited to post questions, e.g. for clarification, and follow on-going discussions about the challenge. For questions which are not of general interest, feel free to contact Sophie Quinton (sophie dot quinton at inria dot fr).



ECRTS Tools and Benchmarks for Real-Time Systems
WATERS Community Forum

Search…

Quick links FAQ Register Login

Home ‹ Board index ‹ Dedicated events ‹ WATERS ‹ WATERS'17 ‹ 2017 industrial challenge

Industrial challenge 2017

Post Reply | Search this topic… | 14 posts • Page 1 of 1

Industrial challenge 2017
by arne.hamann » Thu Feb 16, 2017

arne.hamann
Posts: 27
Joined: Thu Jan 28, 2016
Location: Renningen

We are happy to announce the industrial challenge of WATERS 2017 proposed by Arne Hamann, Simon Kramer, Michael Pressler, Dakshina Dasari, Falk Wurst, and Dirk Ziegenbein from Robert Bosch GmbH.

Details about the intention of the challenge and a high level description can be found here: https://waters2017.inria.fr/challenge/.

A detailed description of the actual challenge can be found in the attached document:
WATERS2017_Industrial_Challenge_Bosch.pdf
(124.04 KiB) Downloaded 180 times

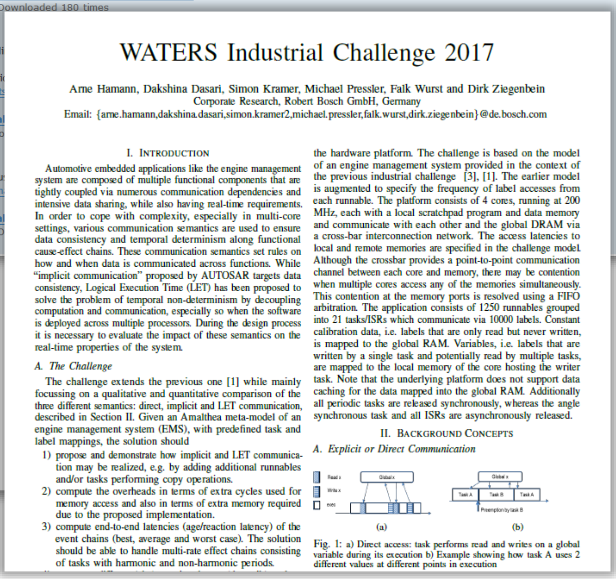
The correspondi

The latest versi
https://projects

Bosch_Chal
(205.5 KiB) Do

And the previou
http://www.am

Bosch_Chal
(205.12 KiB) D



WATERS Industrial Challenge 2017

Arne Hamann, Dakshina Dasari, Simon Kramer, Michael Pressler, Falk Wurst and Dirk Ziegenbein
Corporate Research, Robert Bosch GmbH, Germany
Email: {arne.hamann,dakshina.dasari,simon.kramer2,michael.pressler,falk.wurst,dirk.ziegenbein}@de.bosch.com

I. Introduction

Automotive embedded applications like the engine management system are composed of multiple functional components that are tightly coupled via numerous communication dependencies and intensive data sharing, while also having real-time requirements. In order to cope with complexity, especially in multi-core settings, various communication semantics are used to ensure data consistency and temporal determinism along functional cause-effect chains. These communication semantics set rules on how and when data is communicated across functions. While "implicit communication" proposed by AUTOSAR targets data consistency, Logical Execution Time (LET) has been proposed to solve the problem of temporal non-determinism by decoupling computation and communication, especially so when the software is deployed across multiple processors. During the design process it is necessary to evaluate the impact of these semantics on the real-time properties of the system.

A. The Challenge

The challenge extends the previous one [1] while mainly focussing on a qualitative and quantitative comparison of the three different semantics: direct, implicit and LET communication, described in Section II. Given an Amalthea meta-model of an engine management system (EMS), with predefined task and label mappings, the solution should

1) propose and demonstrate how implicit and LET communication may be realized, e.g. by adding additional runnables and/or tasks performing copy operations.
2) compute the overheads in terms of extra cycles used for memory access and also in terms of extra memory required due to the proposed implementation.
3) compute end-to-end latencies (age/reaction latency) of the event chains (best, average and worst case). The solution should be able to handle multi-rate effect chains consisting of tasks with harmonic and non-harmonic periods.

the hardware platform. The challenge is based on the model of an engine management system provided in the context of the previous industrial challenge [3], [1]. The earlier model is augmented to specify the frequency of label accesses from each runnable. The platform consists of 4 cores, running at 200 MHz, each with a local scratchpad program and data memory and communicate with each other and the global DRAM via a cross-bar interconnection network. The access latencies to local and remote memories are specified in the challenge model. Although the crossbar provides a point-to-point communication channel between each core and memory, there may be contention when multiple cores access any of the memories simultaneously. This contention at the memory ports is resolved using a FIFO arbitration. The application consists of 1250 runnables grouped into 21 tasks/ISRs which communicate via 10000 labels. Constant calibration data, i.e. labels that are only read but never written, is mapped to the global RAM. Variables, i.e. labels that are written by a single task and potentially read by multiple tasks, are mapped to the local memory of the core hosting the writer task. Note that the underlying platform does not support data caching for the data mapped into the global RAM. Additionally all periodic tasks are released synchronously, whereas the angle synchronous task and all ISRs are asynchronously released.

II. Background Concepts

A. Explicit or Direct Communication

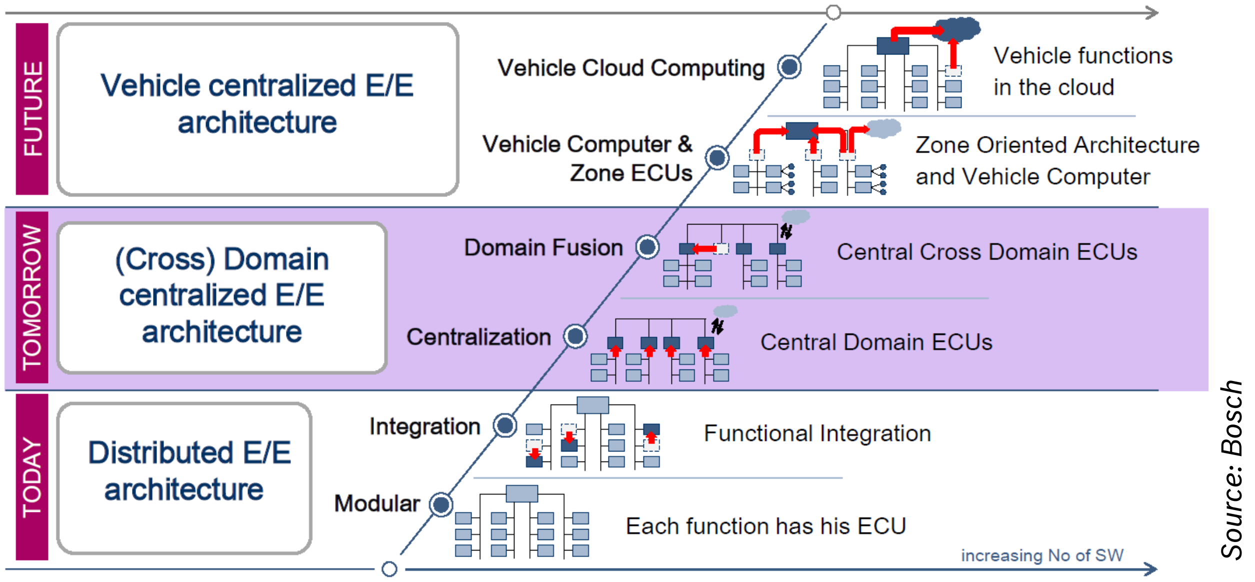
Fig. 1: a) Direct access: task performs read and writes on a global variable during its execution b) Example showing how task A uses 2 different values at different points in execution

BOSCH

Real-time Systems Engineering @ Bosch
Outline

- Performance modeling & analysis of classical automotive systems
  - Motivation … or the real complexity
  - Amalthea performance model
  - Current usage @ Bosch
  - **Upcoming challenges**
- Communication centric design in multi-core systems
  - Importance of cause-effect chains
  - Issues with concurrent execution in multi-core systems
  - Communication mechanisms as solution & impact on latencies
  - Experiments
- Timing-aware control design
  - Control and real-time systems engineering – two worlds collide
  - Co-engineering approach
  - Example

Performance modeling & analysis of classical automotive systems
Trends in automotive E/E systems



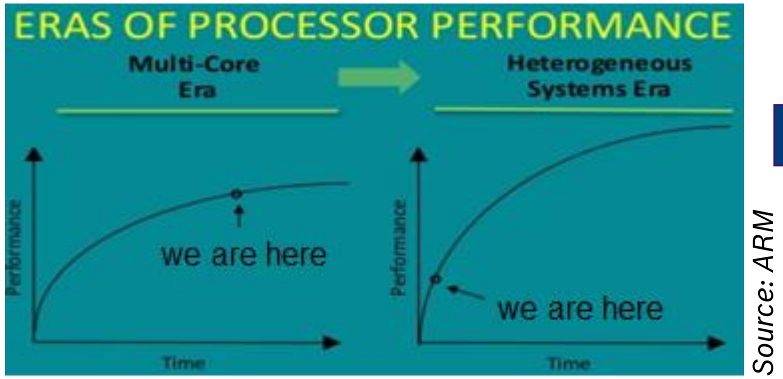
FUTURE
Vehicle centralized E/E architecture
TOMORROW
(Cross) Domain centralized E/E architecture
TODAY
Distributed E/E architecture
Vehicle Cloud Computing
Vehicle functions in the cloud
Vehicle Computer & Zone ECUs
Zone Oriented Architecture and Vehicle Computer
Domain Fusion
Central Cross Domain ECUs
Centralization
Central Domain ECUs
Integration
Functional Integration
Modular
Each function has his ECU
increasing No of SW
Source: Bosch

Large-scale integration of heterogeneous applications on (Cross)-Domain & Vehicle Centralized E/E Architectures

Computing Power Demand

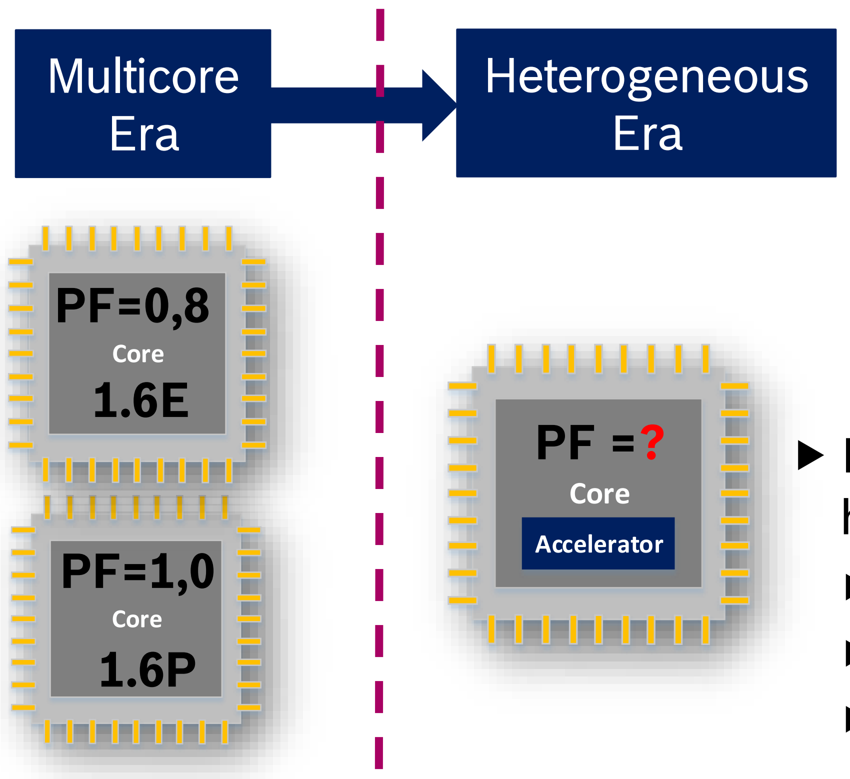
Serial computing in embedded systems is hitting the **techno-logical limits**

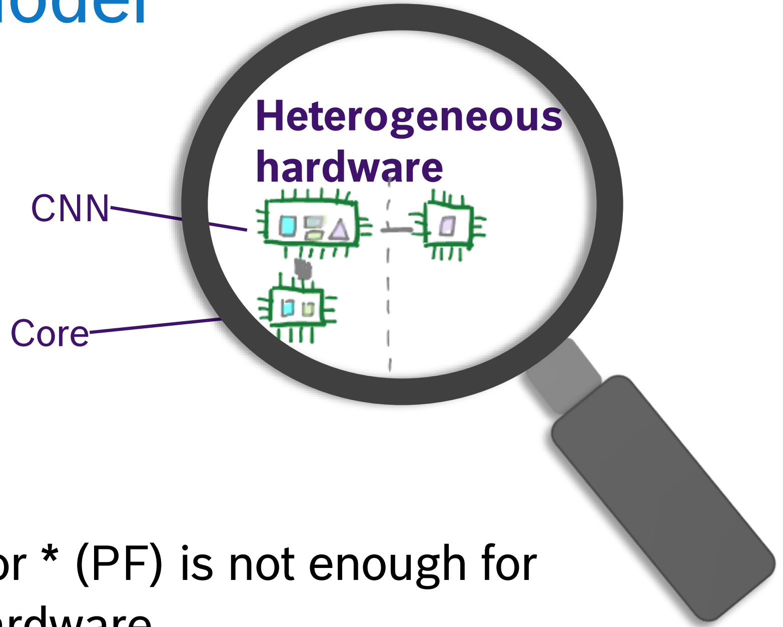
2025 ⇧ 2015 **Factor 2 - 20**



Heterogeneous HW platforms to satisfy tremendous need for computing power

Performance modeling & analysis of classical automotive systems
Extending the AMALTHEA Hardware Model





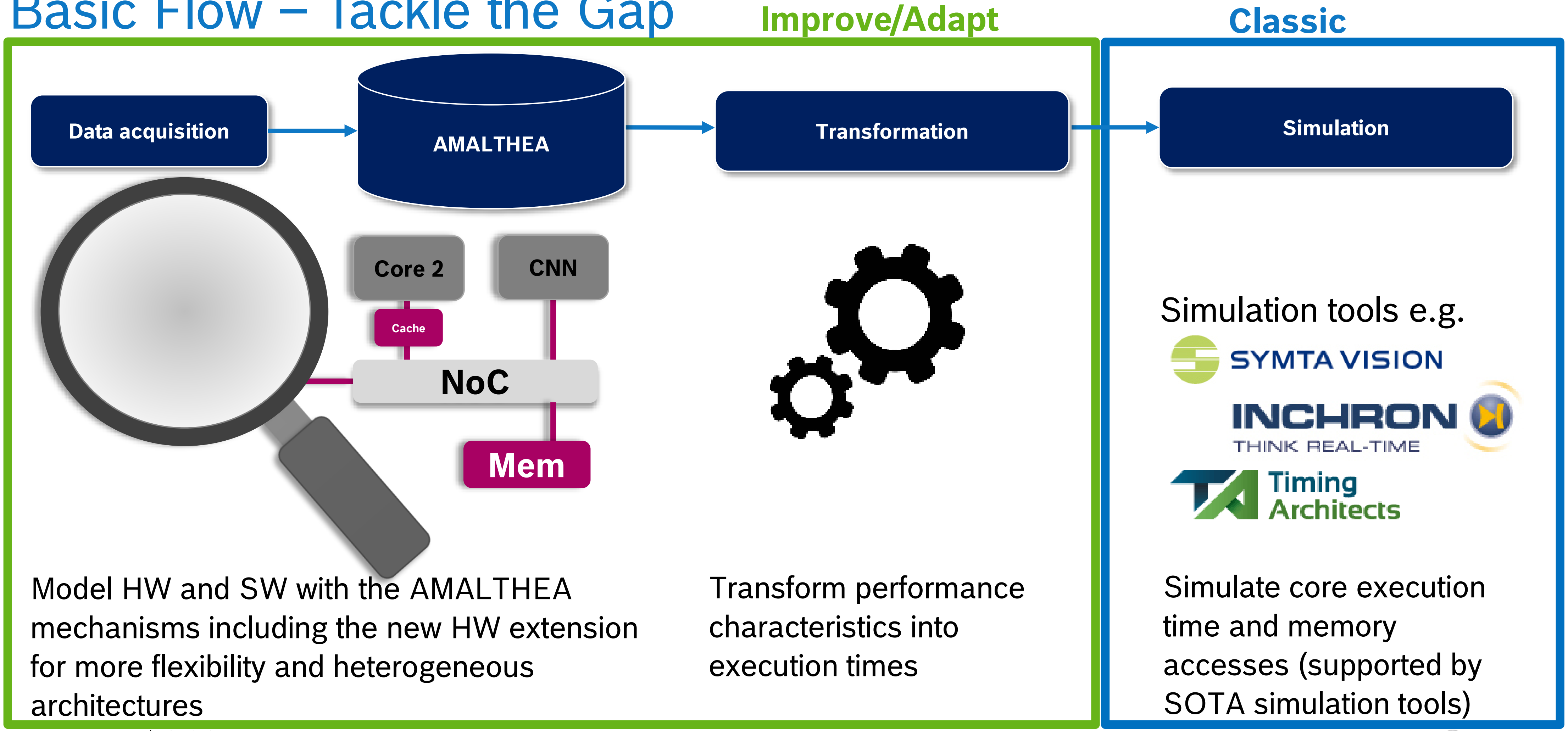
- Performance factor * (PF) is not enough for heterogeneous hardware
  - Non linear performance effects due to specific acceleration
  - PF is not transparent regarding heterogeneous effects
  - Infeasible to compare different ISAs

* E.g., IPC (Instructions Per Cycle)

Goal: Enable our models & simulation for heterogeneous software and hardware

Performance modeling & analysis of classical automotive systems

Basic Flow – Tackle the Gap



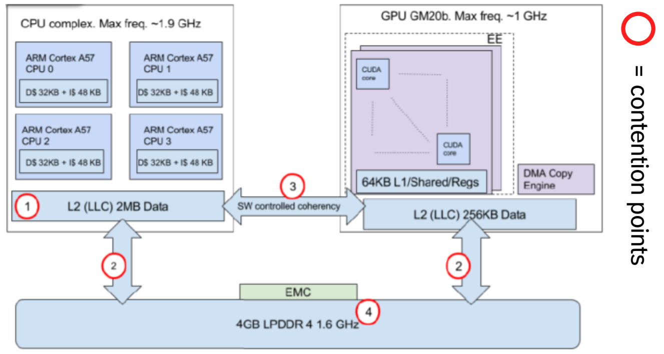
Model HW and SW with the AMALTHEA mechanisms including the new HW extension for more flexibility and heterogeneous architectures

Transform performance characteristics into execution times

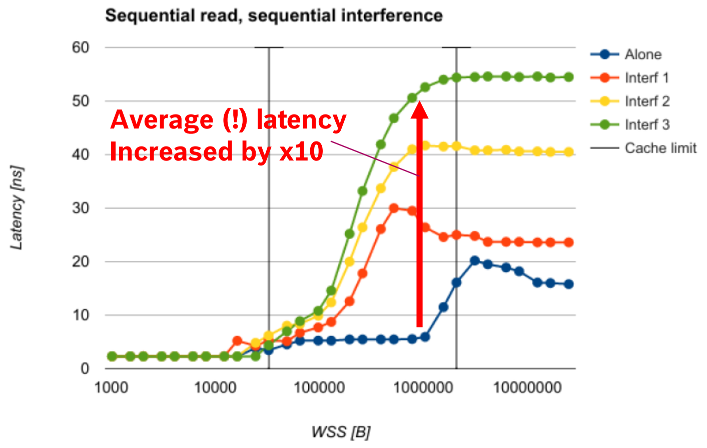
Simulation tools e.g.

Simulate core execution time and memory accesses (supported by SOTA simulation tools)

Performance modeling & analysis of classical automotive systems
Predictability on High-Performance Platforms



NVIDIA Tegra X1 Platform



Avg. memory access latencies per word

Source: Roberto Cavicchioli, Nicola Capodieci, Marko Bertogna, Memory interference characterization between CPU cores and integrated GPUs in mixed-criticality platforms. ETFA 2017

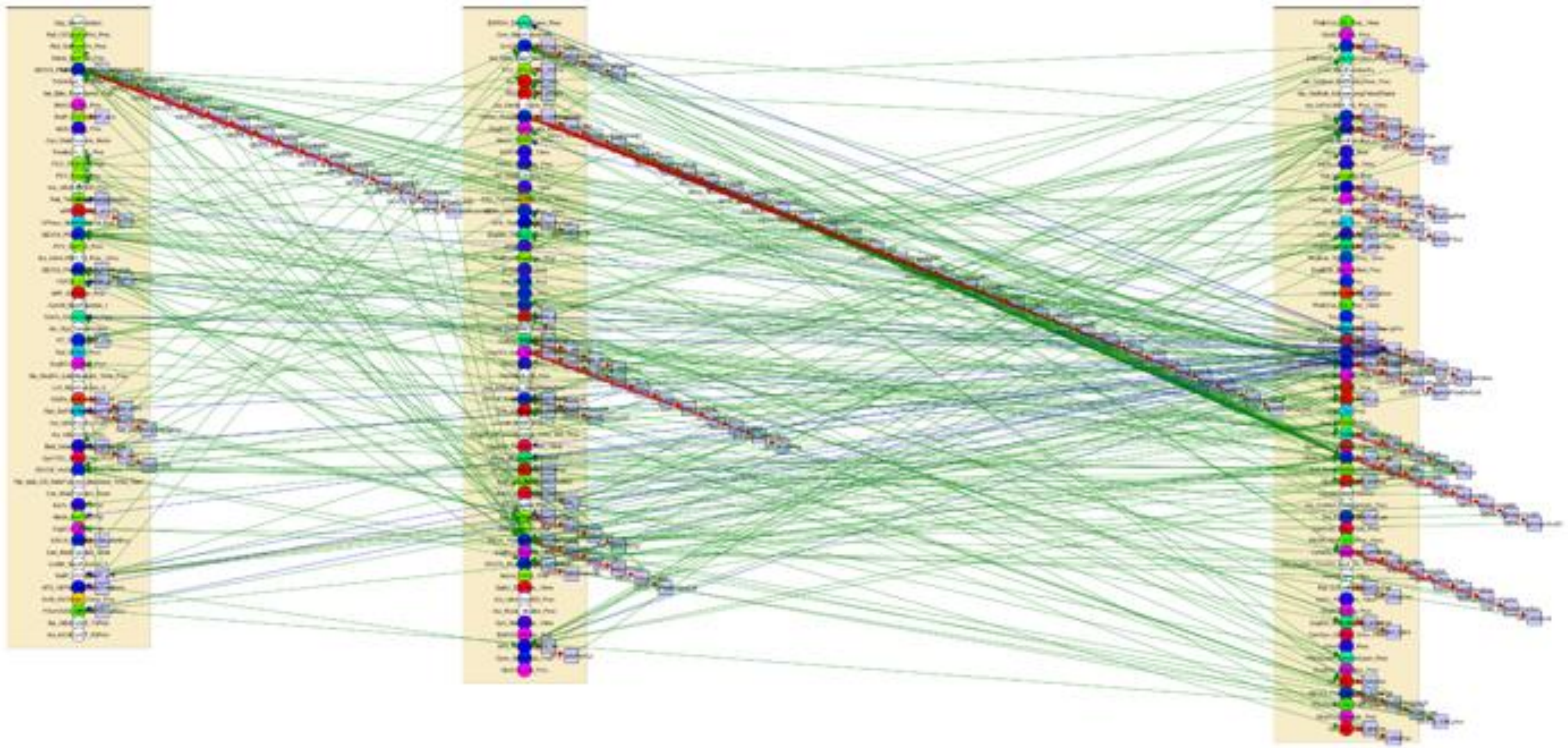
- Shared memory is a big bottleneck in high-end µP based real-time platforms
  - **Interference effects are more severe by orders of magnitude compared to µC platforms**
- Support systems engineering with performance analysis for high-performance platforms
- Goal: predictable real-time behavior

Real-time Systems Engineering @ Bosch
Outline

- Performance modeling & analysis of classical automotive systems
  - Motivation ... or the real complexity
  - Amalthea performance model
  - Current usage @ Bosch
  - Upcoming challenges
- **Communication centric design in multi-core systems**
  - **Importance of cause-effect chains**
  - Issues with concurrent execution in multi-core systems
  - Communication mechanisms as solution & impact on latencies
  - Experiments
- Timing-aware control design
  - Control and real-time systems engineering – two worlds collide
  - Co-engineering approach
  - Example

Communication Centric Design

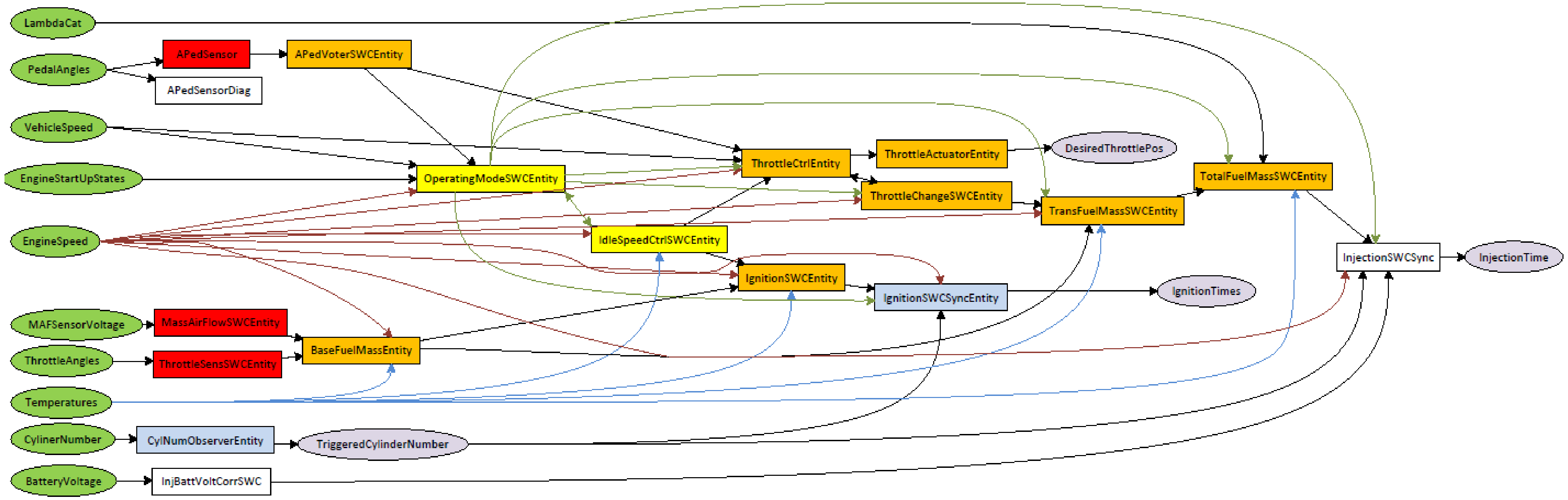
Introduction 1/2: Complexity of communication dependencies



Communication Centric Design

Introduction 2/2: Importance of cause-effect chains

- Very simple SW structure of an engine control system

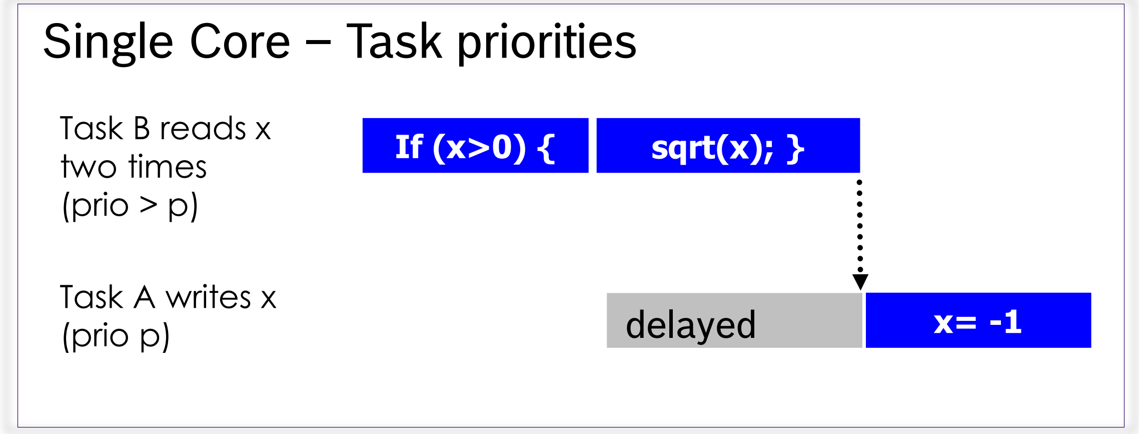


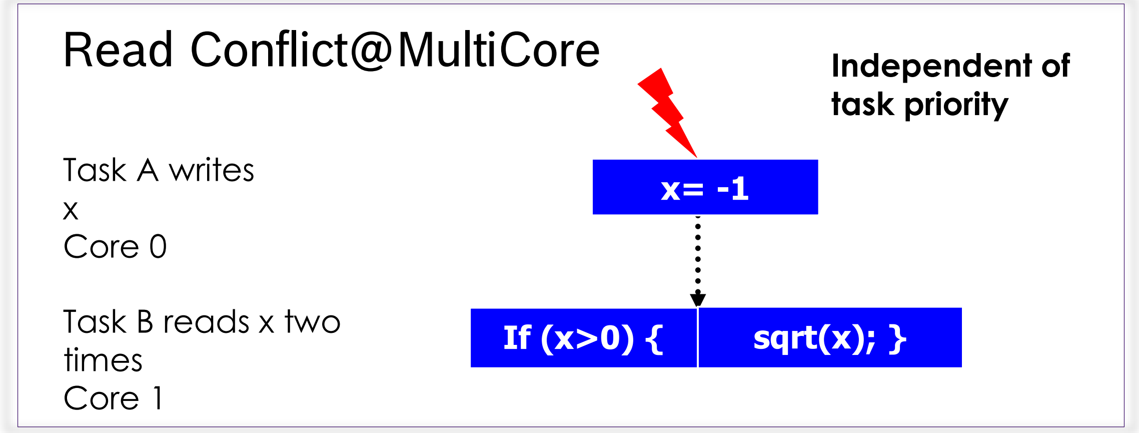
Benchmarking, System Design and Case-studies for Multi-core based Embedded Automotive Systems
Piotr Dziurzanski, Amit Kumar Singh, Leandro S. Indrusiak, Björn Saballus

BOSCH

Real-time Systems Engineering @ Bosch
Outline

Communication Centric Design
Data inconsistency problem


Single Core – Task priorities
Task B reads x two times (prio > p)
If (x>0) {
sqrt(x); }
Task A writes x (prio p)
delayed
x= -1

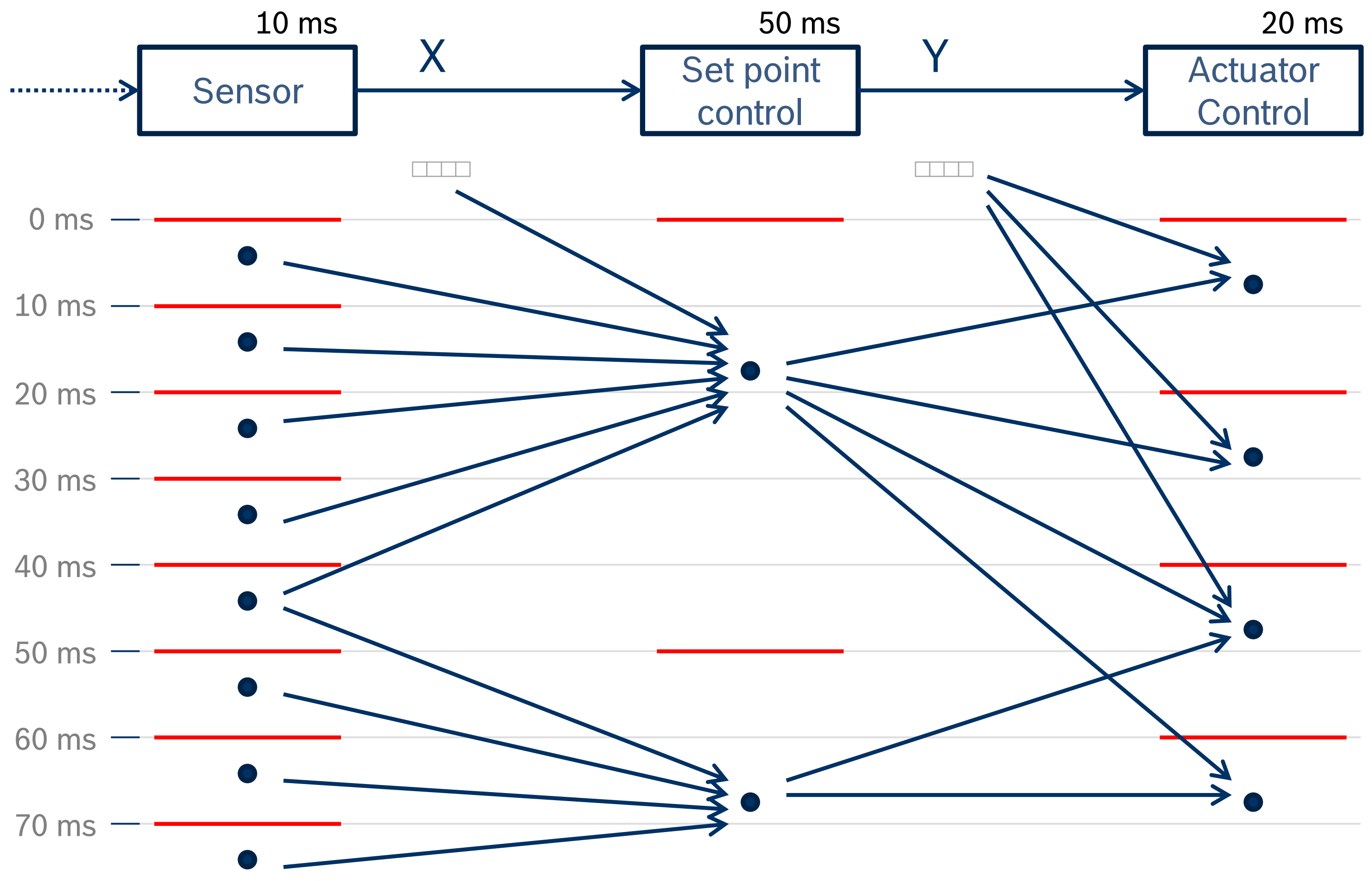


- Single core: Legacy code contains implicit assumptions about priorities and thus execution sequences
- Multi-core: These assumptions often break the functionalities and require lots of debugging of race conditions

→ Need for data consistency

Communication Centric Design

Distribution and load dependent end-to-end latencies

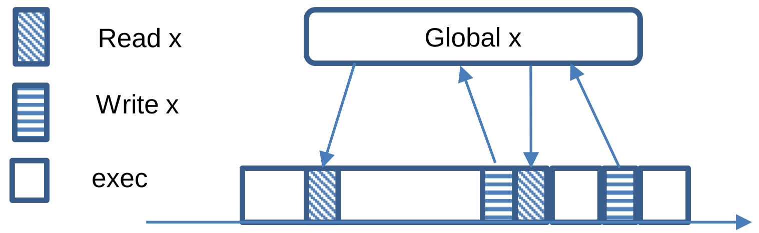


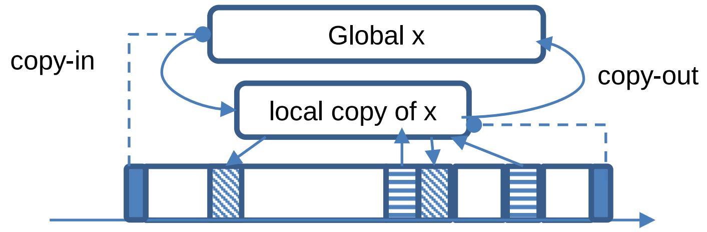
- End-to-end behaviour along cause effect chains is non deterministic
- Heavily depends on distribution & scheduling
- 188 possible chains
- Prohibitive for “large scale engineering” where we need to handle thousands of variants
- It’s not about optimization !
- **Determinism needed**: distribution and load independent timing behaviour

Real-time Systems Engineering @ Bosch
Outline

- Performance modeling & analysis of classical automotive systems
  - Motivation ... or the real complexity
  - Amalthea performance model
  - Current usage @ Bosch
  - Upcoming challenges
- **Communication centric design in multi-core systems**
  - Importance of cause-effect chains
  - Issues with concurrent execution in multi-core systems
  - **Communication mechanisms as solution & impact on latencies**
  - Experiments
- Timing-aware control design
  - Control and real-time systems engineering – two worlds collide
  - Co-engineering approach
  - Example

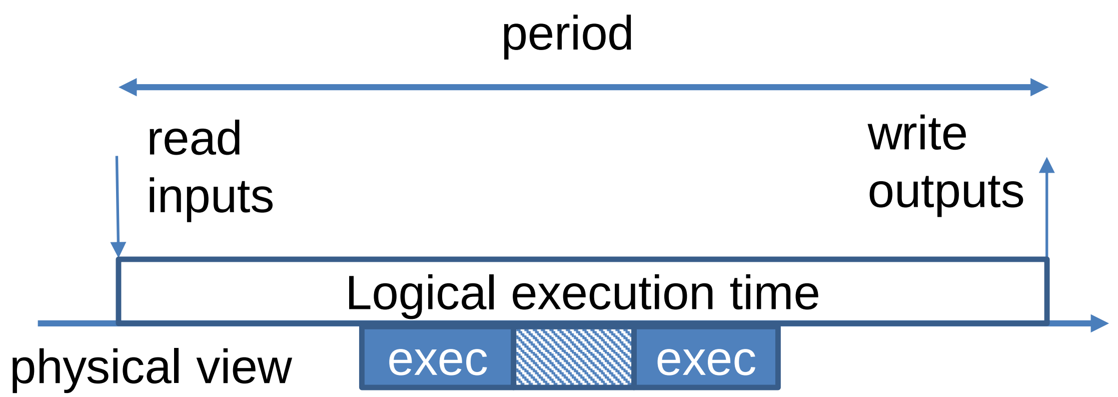
Communication Centric Design
Implicit communication to achieve data consistency





- Automotive embedded systems are organized in tasks containing functions that communicate over shared memory (using labels)
- Explicit communication
  - No regulations in place, each function directly reads and writes labels
  - Possible races are handled using locks by the developers
- Implicit communication
  - Local copies are created for each read label at the beginning of the task
  - All computations work on the local copies
  - The local copies are written back to the shared memory at the end of the task
  - Result: data consistency on task level: all functions operate on the same data set

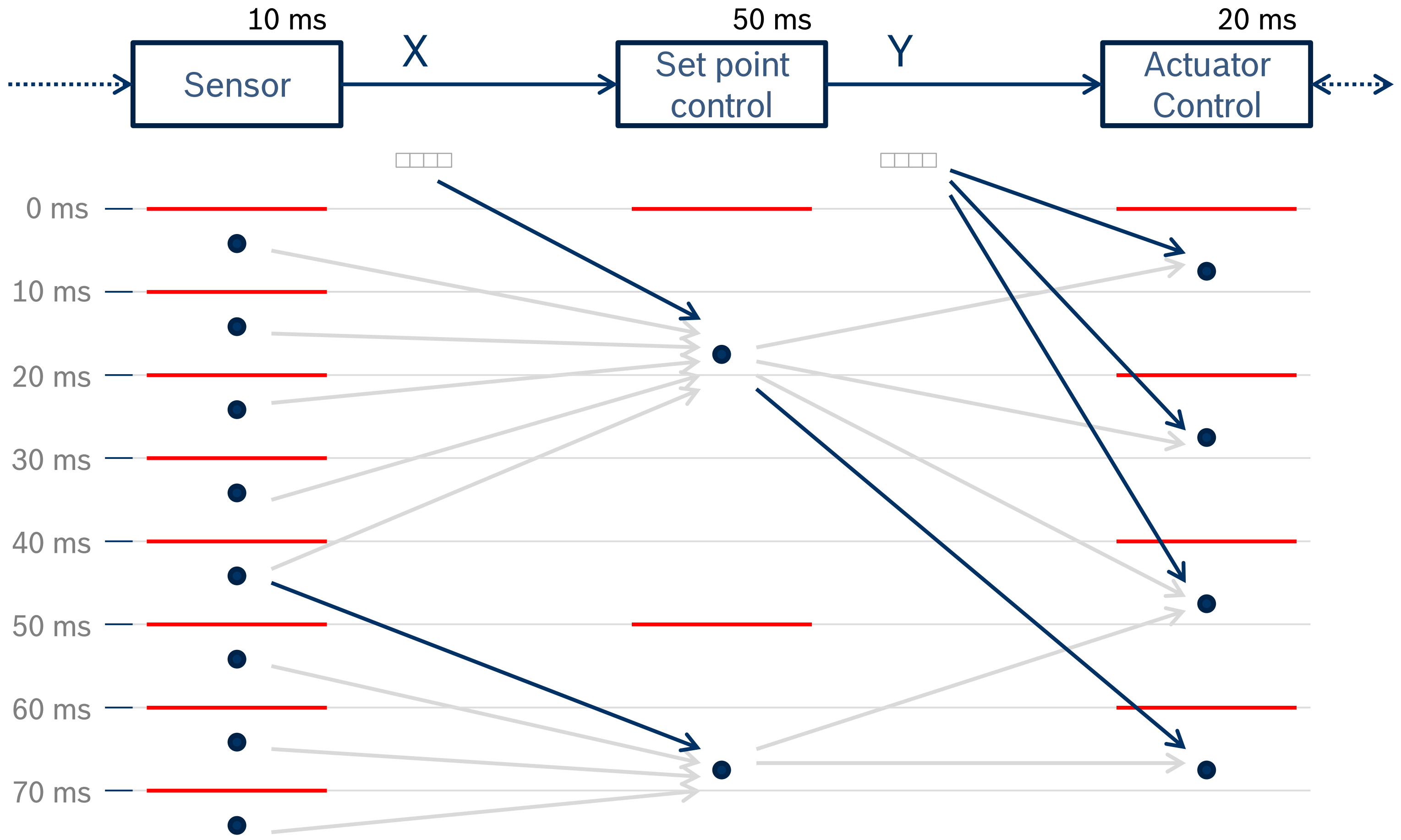
Communication Centric Design
Logical Execution time (LET) communication



- Mechanism to ensure determinism and data consistency
- Data is communicated at the beginning and end of the period (activation interval)
- Deterministic availability of data irrespective of where the task executes
- Decouples communication and execution
  - Also independent of where data is mapped
- Incurs longer latency
- Simplified event chain timing analysis for complex event chains with multi-rate tasks

Communication Centric Design

Cause-effect chain revisited using LET

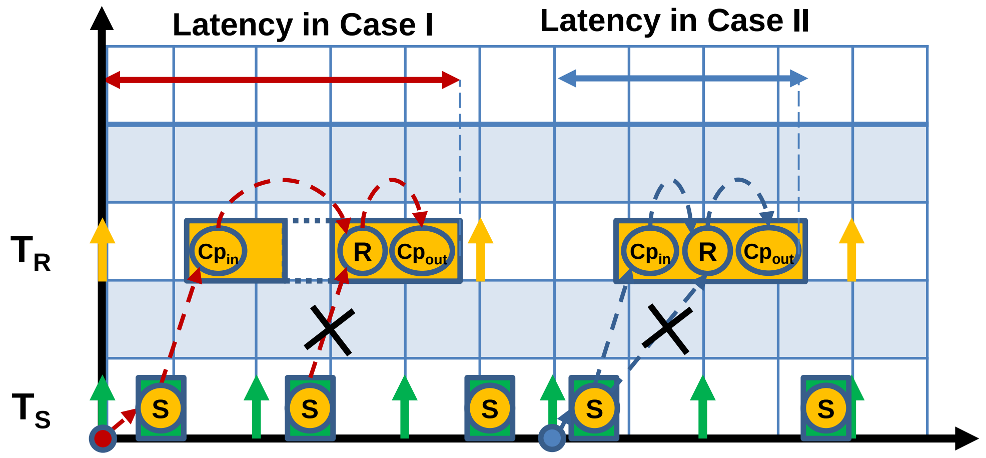


Communication Centric Design
Analysis of end-to-end latencies

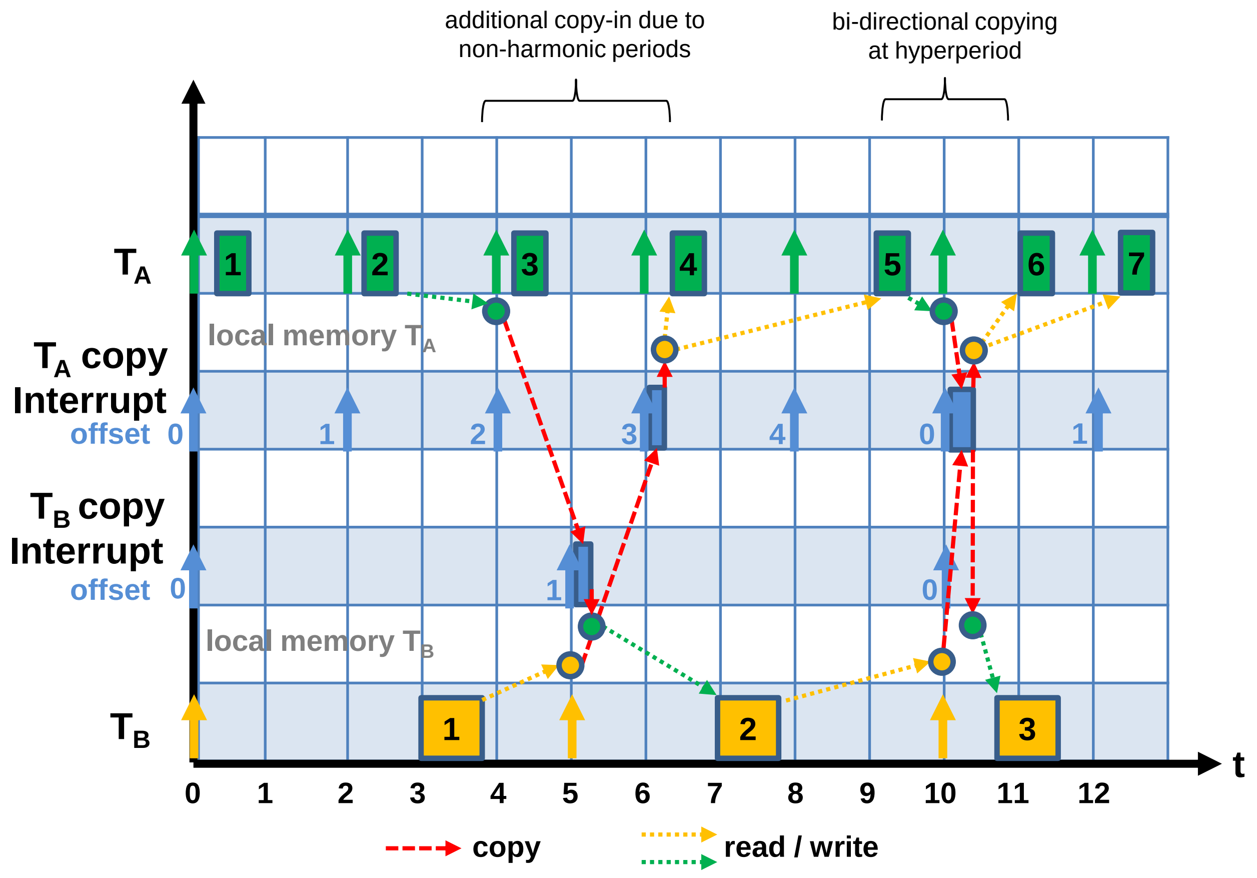
- For real-world systems **implicit & LET communication** need to be taken into account
- End-to-end latency analyses & simulation approaches are available for **direct communication**
  - MAST, SymTA/S, pyCPA, Prelude, Timing Architects, Real-time Calculus, ... (name it)
- However, these tools generally ignore communication semantics or focus on schedulability analysis considering task deadlines only
- Idea: **transform the performance model** to take into account the different communication semantics

Communication Centric Design
Transformation for implicit communication

- Goal: data consistency on task level
  - Different tasks might work on different values at the same time instant
- Trivial transformation: For each Task T
  - Adding one copy-in runnable Cp_{in}: Create a local copy for all data that is read or modified
  - Adding one copy-out runnable Cp_{out} : Write back local copies
  - Add these copy runnables Cp_{in} and Cp_{out} to the cause-effect chain



Communication Centric Design
Transformation for LET communication (naïve implementation)



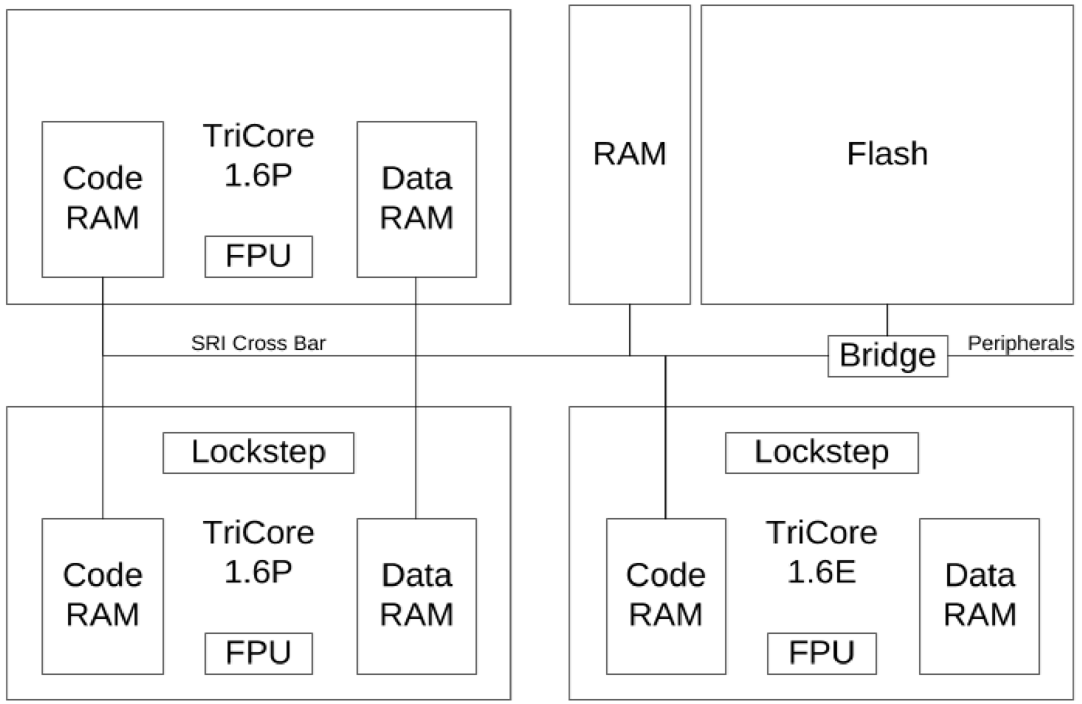
- Many different possibilities to implement LET communication
- Need to perform copy operation between each pair of communicating tasks
- Here: copy operations done by high priority copy interrupts
  - Leads to jitter

Real-time Systems Engineering @ Bosch
Outline

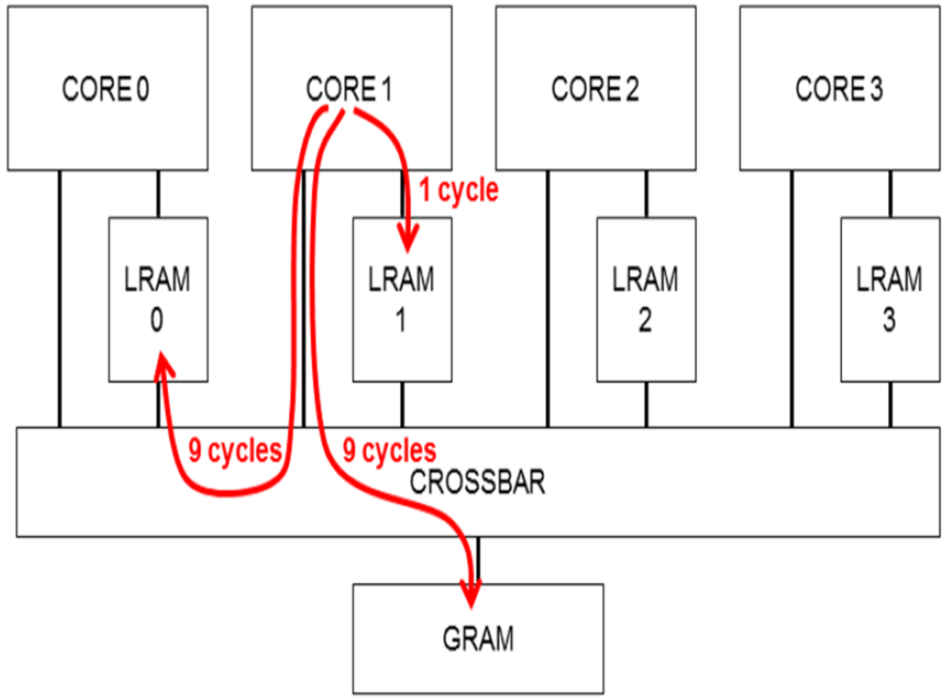
- Performance modeling & analysis of classical automotive systems
  - Motivation ... or the real complexity
  - Amalthea performance model
  - Current usage @ Bosch
  - Upcoming challenges
- **Communication centric design in multi-core systems**
  - Importance of cause-effect chains
  - Issues with concurrent execution in multi-core systems
  - Communication mechanisms as solution & impact on latencies
  - **Experiments**
- Timing-aware control design
  - Control and real-time systems engineering – two worlds collide
  - Co-engineering approach
  - Example

Communication Centric Design
HW Model

- Simplified AURIX Architecture

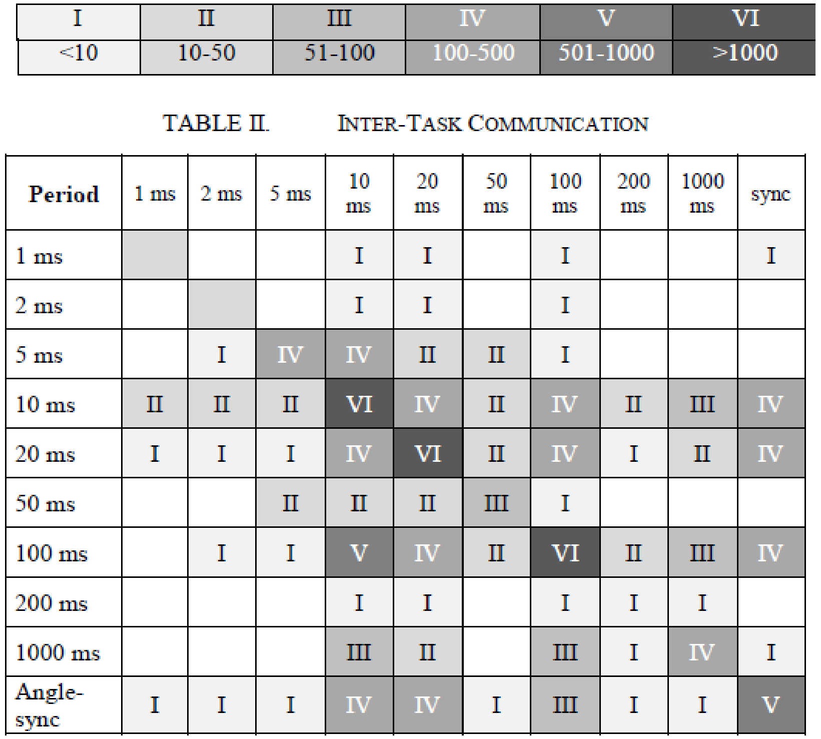


- Memory Access Time



Communication Centric Design
SW Model

- Key data of the model
  - **1250 Runnables** mapped to
  - **21 Tasks & Interrupts** accessing
  - **10.000 Labels** (shared data)
  - Event chains
- Huge amount of data dependencies
  - challenge exact analysis methods

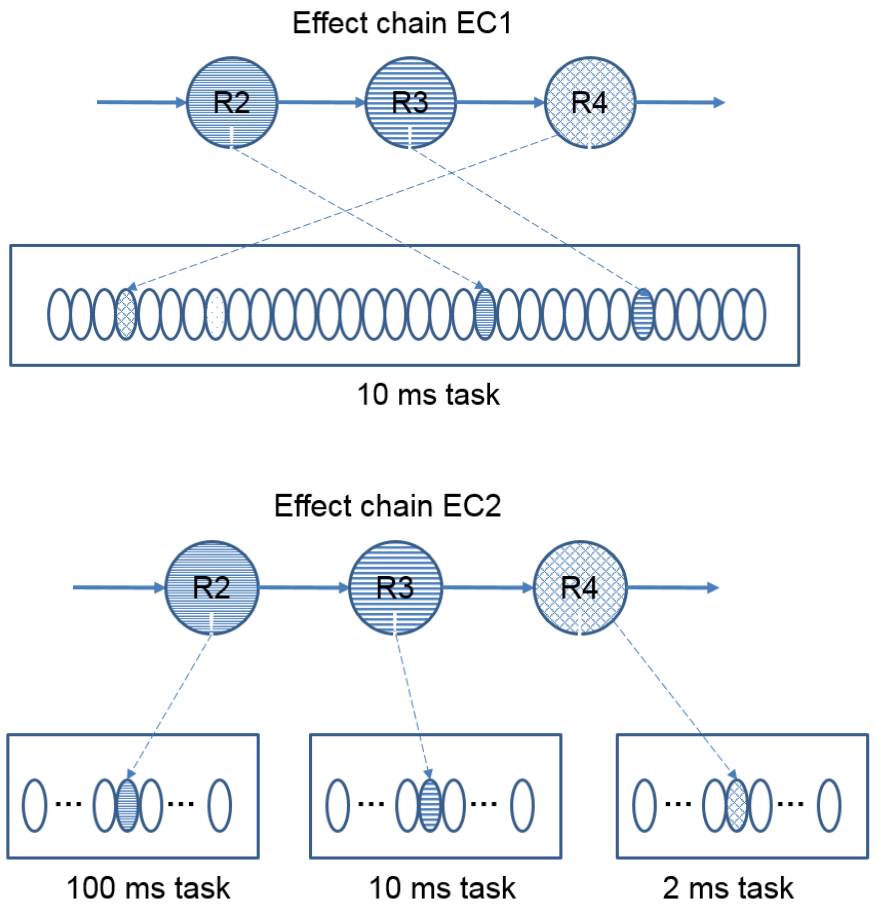


I	II	III	IV	V	VI
<10	10-50	51-100	100-500	501-1000	>1000

TABLE II. INTER-TASK COMMUNICATION

Period	1 ms	2 ms	5 ms	10 ms	20 ms	50 ms	100 ms	200 ms	1000 ms	sync
1 ms				I	I		I			I
2 ms				I	I		I			
5 ms		I	IV	IV	II	II	I			
10 ms	II	II	II	VI	IV	II	IV	II	III	IV
20 ms	I	I	I	IV	VI	II	IV	I	II	IV
50 ms			II	II	II	III	I			
100 ms		I	I	V	IV	II	VI	II	III	IV
200 ms				I	I		I	I	I	
1000 ms				III	II		III	I	IV	I
Angle-sync	I	I	I	IV	IV	I	III	I	I	V

Communication Centric Design
Experiment setup

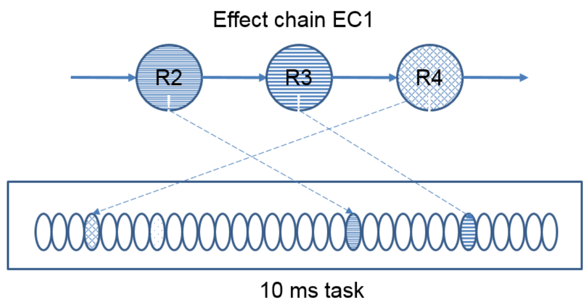


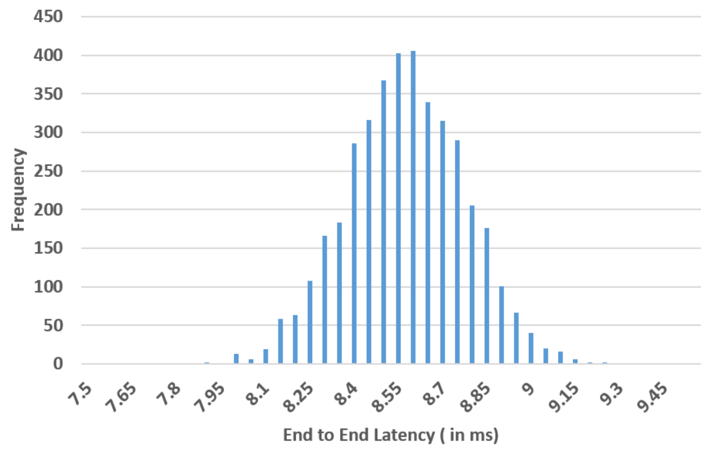
- Analysis of 2 cause-effect chains
- Calculation of end-to-end latency distribution
  - Direct communication
  - Implicit communication
  - LET communication
- Comparison of overhead for copy operations
- Use of scheduling simulation engine of SymTA/S
  - Worst-case end-to-end latency of limited interest

Communication Centric Design

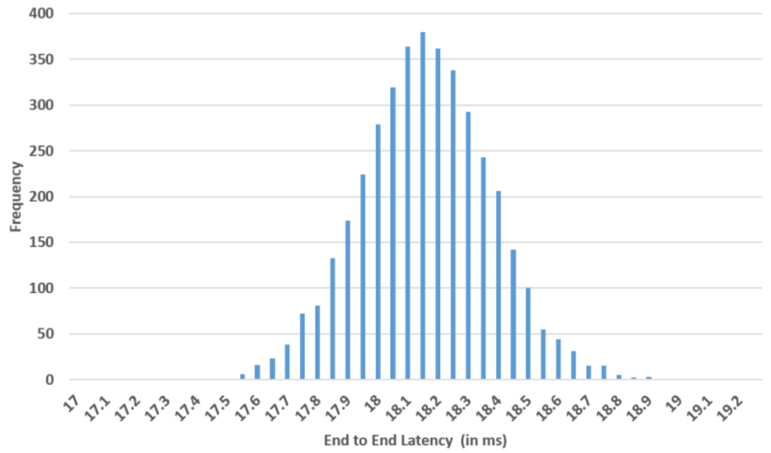
End-to-end latency EC1

- Reaction semantic: 10ms → 10ms → 10ms

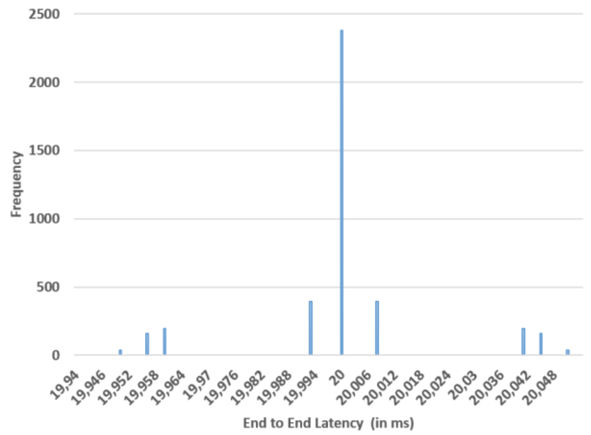




Direct communication

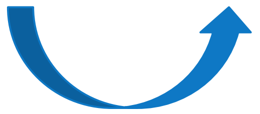


Implicit communication



LET communication

Latency ~ x 2

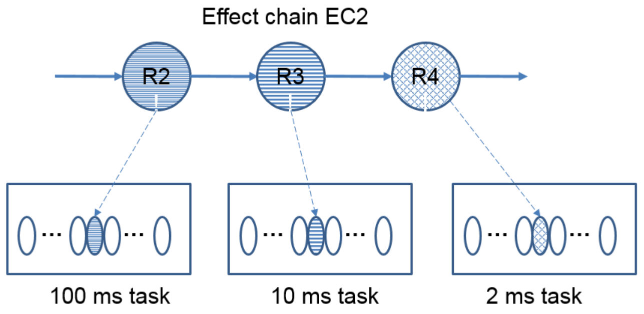


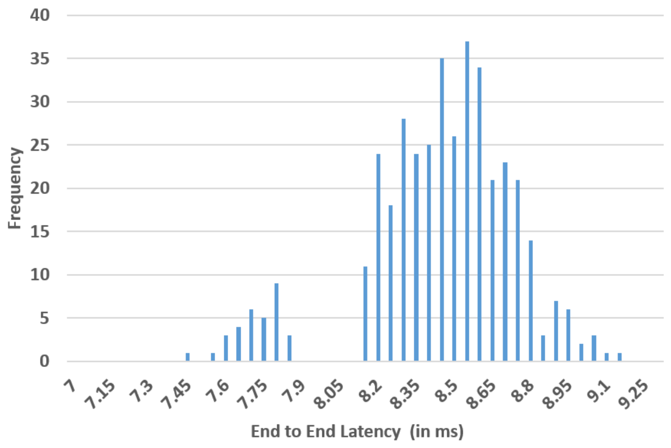
~ same Latency

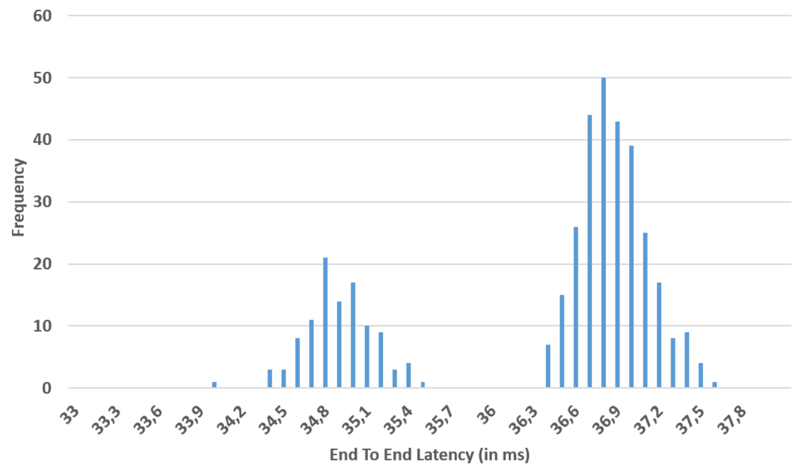
Communication Centric Design

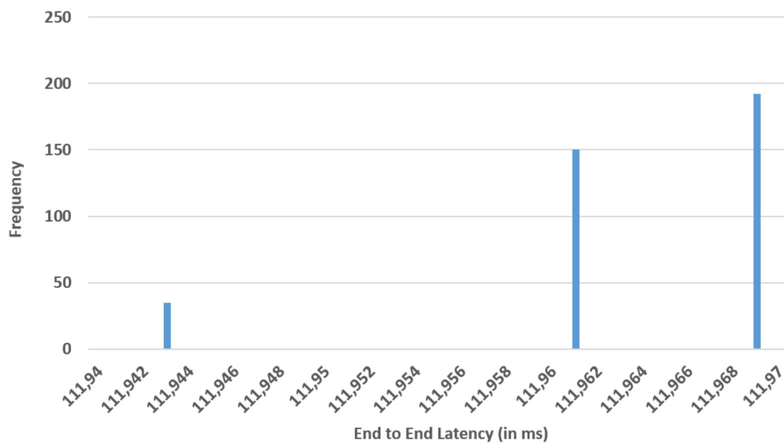
End-to-end latency EC2

- Reaction semantic: 100ms → 10ms → 2ms





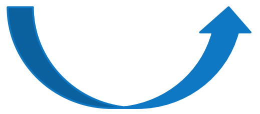




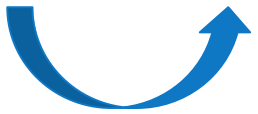
Direct communication

Implicit communication

LET communication



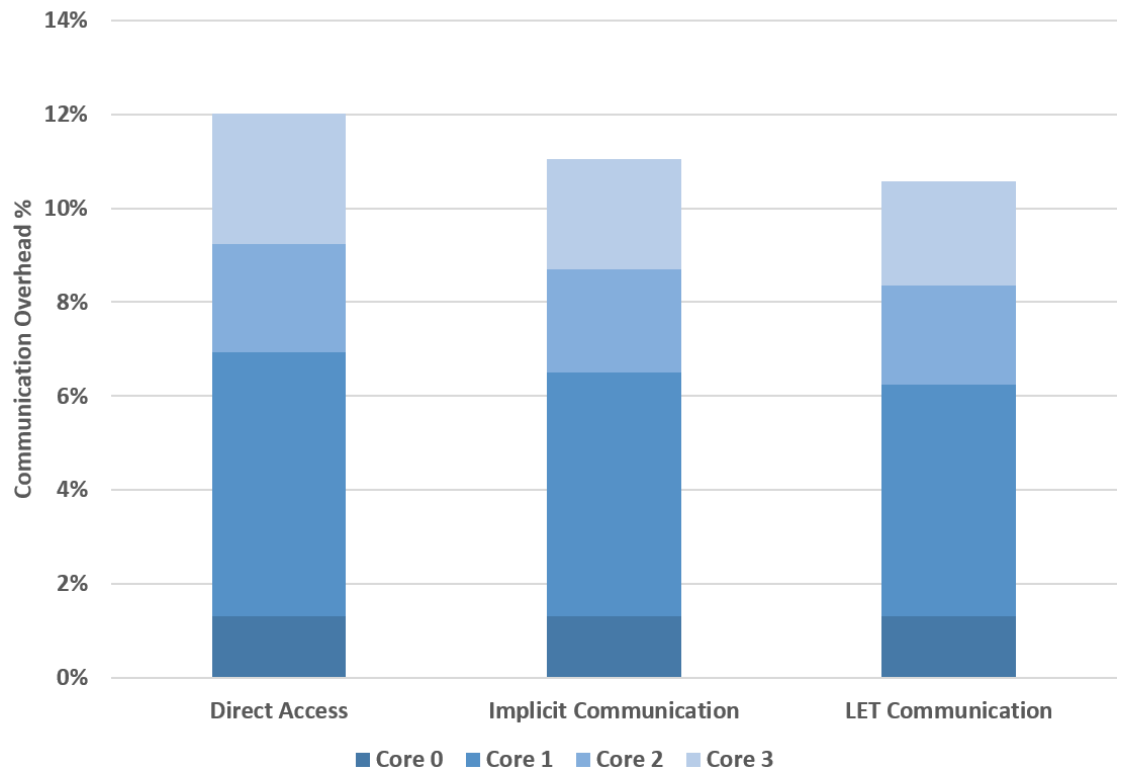
Latency ~ x5



Latency ~ x3

BOSCH

Communication Centric Design
Data access costs for the different communication semantics



- Observation 1: implicit communication reduces data access costs
- Observation 2: LET communication further reduces data access costs since less copy operations are performed
- Room for optimizing the data placement to reduce data access costs

Communication Centric Design
Conclusion

- Large scale engineering requires mechanisms that simplify timing analysis
  - Simplicity, maintainability, composability key principles of robust design

- Benefits offered by Implicit and LET communication in terms of determinism and data consistency outweigh the increase in latency

- Communication semantics need to be accounted for in the timing analysis
  - Impact each stage: Task Formation, Task Mapping, End-to-end Latencies

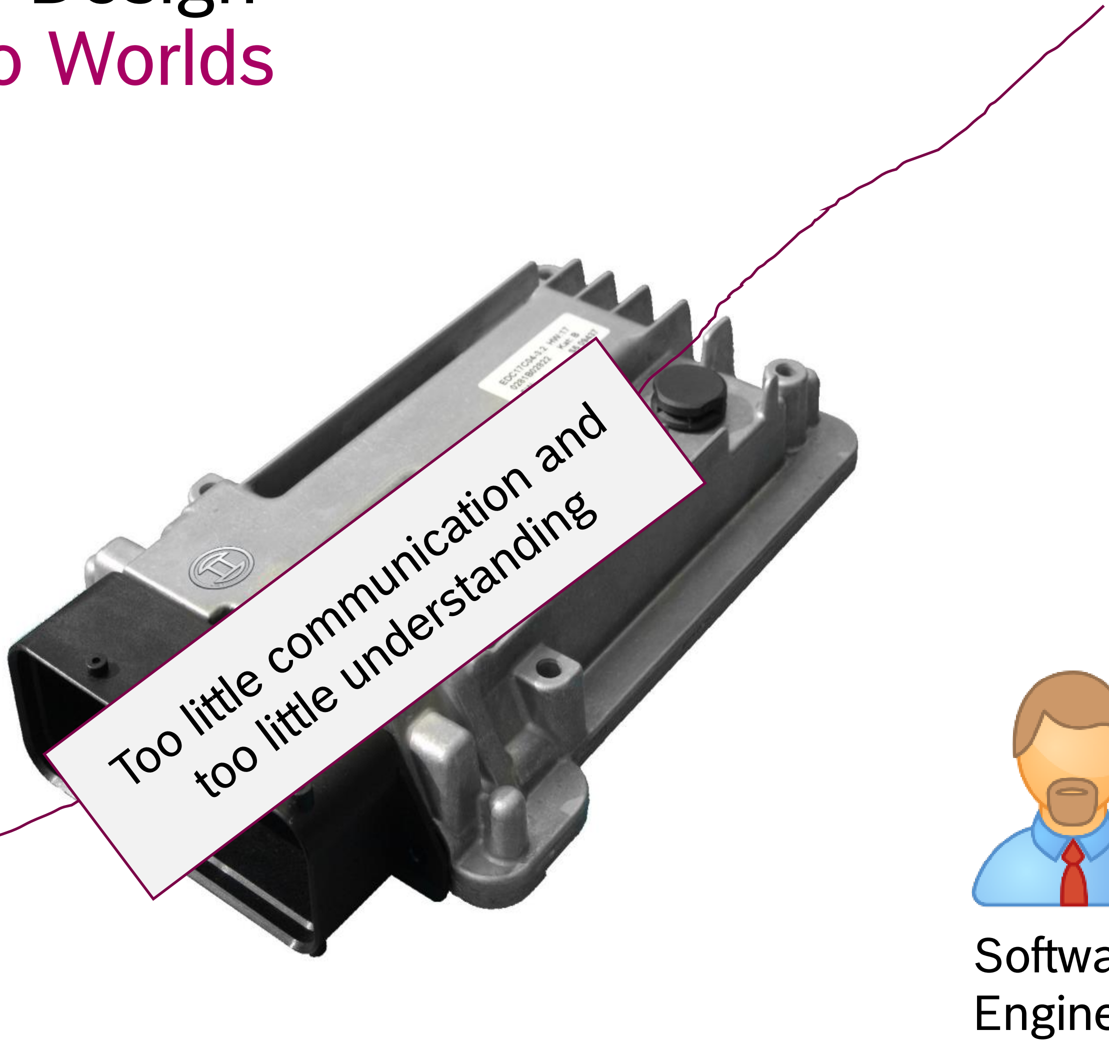
- Existing academic approaches handling co-scheduling of computation/communication should be extended towards meeting the goals of determinism and data consistency

Real-time Systems Engineering @ Bosch
Outline

- Performance modeling & analysis of classical automotive systems
  - Motivation ... or the real complexity
  - Amalthea performance model
  - Current usage @ Bosch
  - Upcoming challenges
- Communication centric design in multi-core systems
  - Importance of cause-effect chains
  - Issues with concurrent execution in multi-core systems
  - Communication mechanisms as solution & impact on latencies
  - Experiments
- **Timing-aware control design**
  - **Control and real-time systems engineering – two worlds collide**
  - Co-engineering approach
  - Example

Timing-aware Control Design
Two Disciplines – Two Worlds

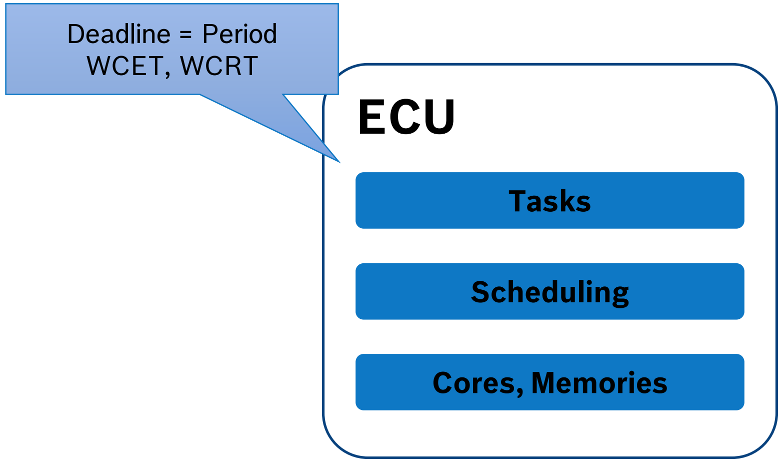
Control Engineer



Software Engineer

Timing-aware Control Design
System as seen by the software engineer

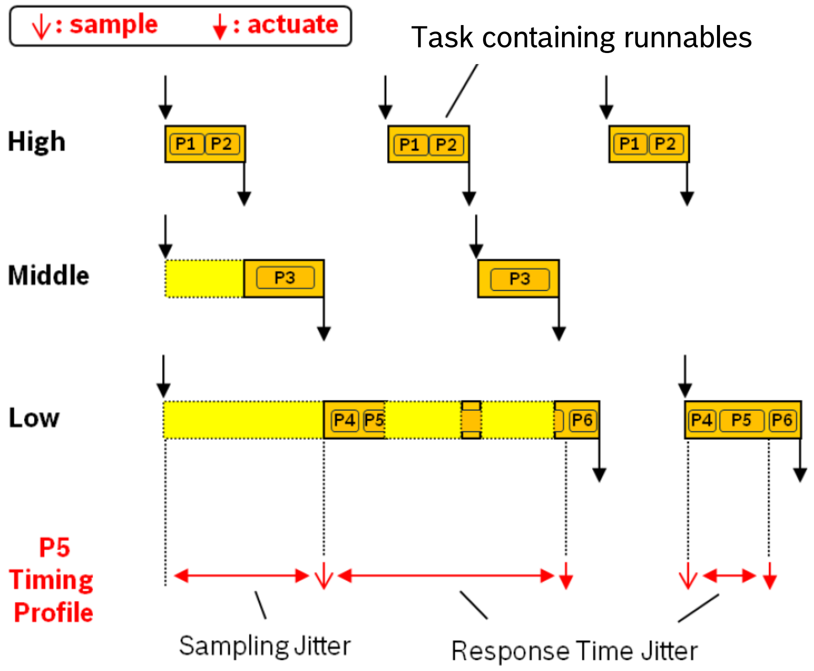
Software Engineer



$$R_i = C_i + \sum_{j \in \mathrm{hp}(i)} C_j \left\lceil \frac{R_i}{T_j} \right\rceil \le D_i = T_i$$

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \le n \cdot \left(\sqrt[n]{2} - 1\right)$$

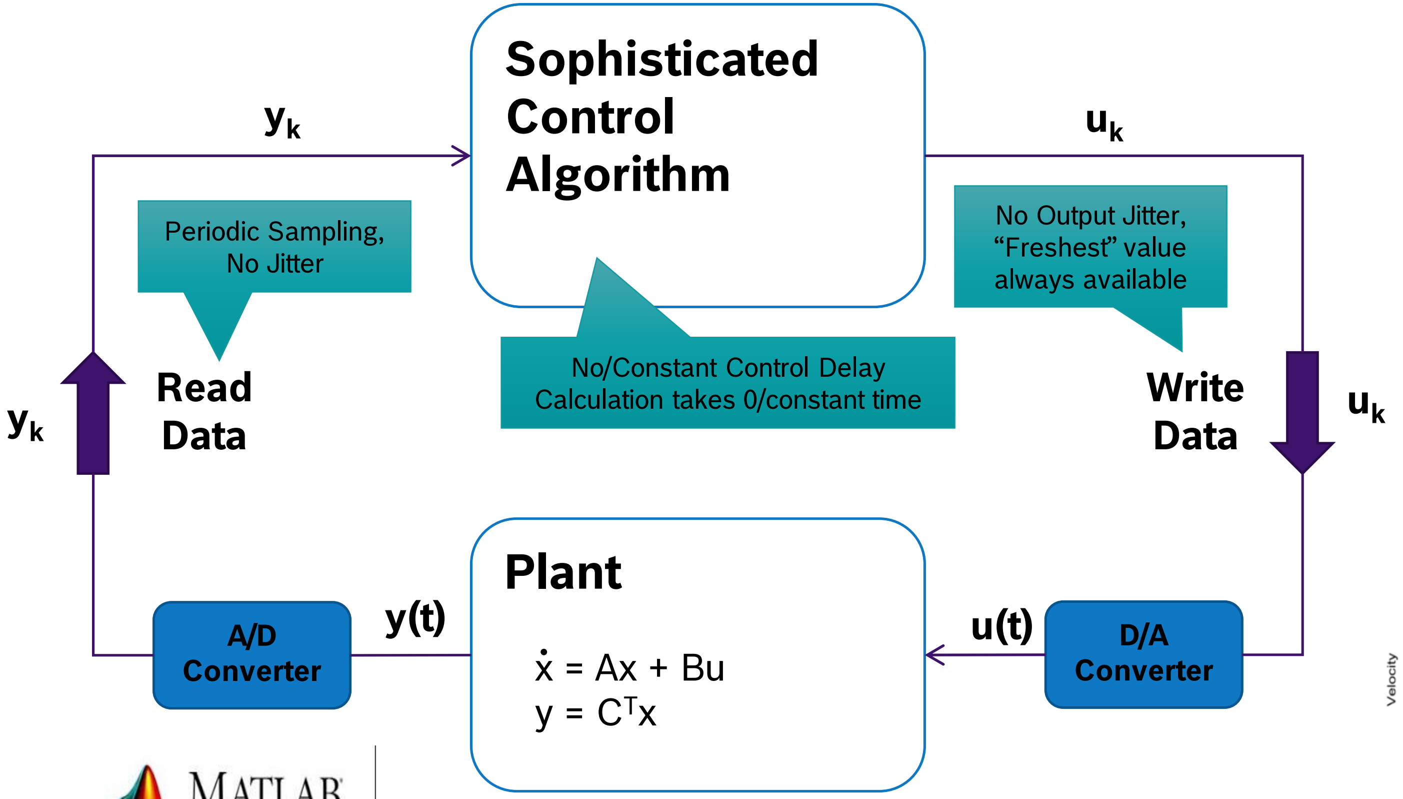
$$\ln 2 \approx 69{,}3\%$$



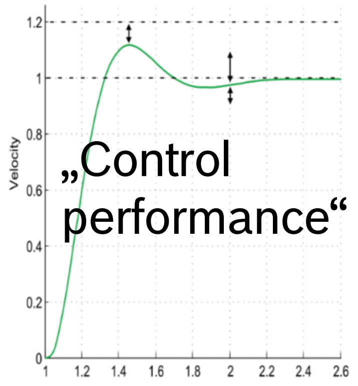
„Real-time performance“

Timing-aware Control Design

System as seen by the control engineer



y_k
Sophisticated Control Algorithm
u_k
Periodic Sampling, No Jitter
No Output Jitter, “Freshest” value always available
No/Constant Control Delay Calculation takes 0/constant time
y_k
Read Data
Write Data
u_k
A/D Converter
$y(t)$
Plant
$\dot{x} = Ax + Bu$
$y = C^T x$
$u(t)$
D/A Converter



Timing-aware Control Design
Consequences

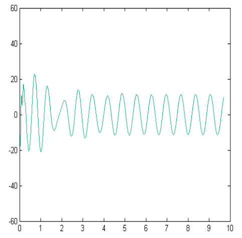
- “That guy has unclear, not implementable requirements -> I’ll optimize resources”
- Guaranteed period, task-wide data consistency, last-is-best (LIB) communication
- Load-dependent behavior & jitter

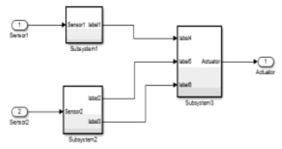
- “My algorithm performs always better in simulation than in the prototype vehicle $\rightarrow$ I’ll test my algorithm only in the vehicle and make it more robust”
- Long design iterations, late rework, ... (adds burden to heavy calibration tasks)
- Wasted HW resources

Real-time Systems Engineering @ Bosch
Outline

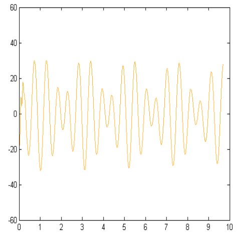
- Performance modeling & analysis of classical automotive systems
  - Motivation ... or the real complexity
  - Amalthea performance model
  - Current usage @ Bosch
  - Upcoming challenges
- Communication centric design in multi-core systems
  - Importance of cause-effect chains
  - Issues with concurrent execution in multi-core systems
  - Communication mechanisms as solution & impact on latencies
  - Experiments
- **Timing-aware control design**
  - Control and real-time systems engineering – two worlds collide
  - **Co-engineering approach**
  - Example

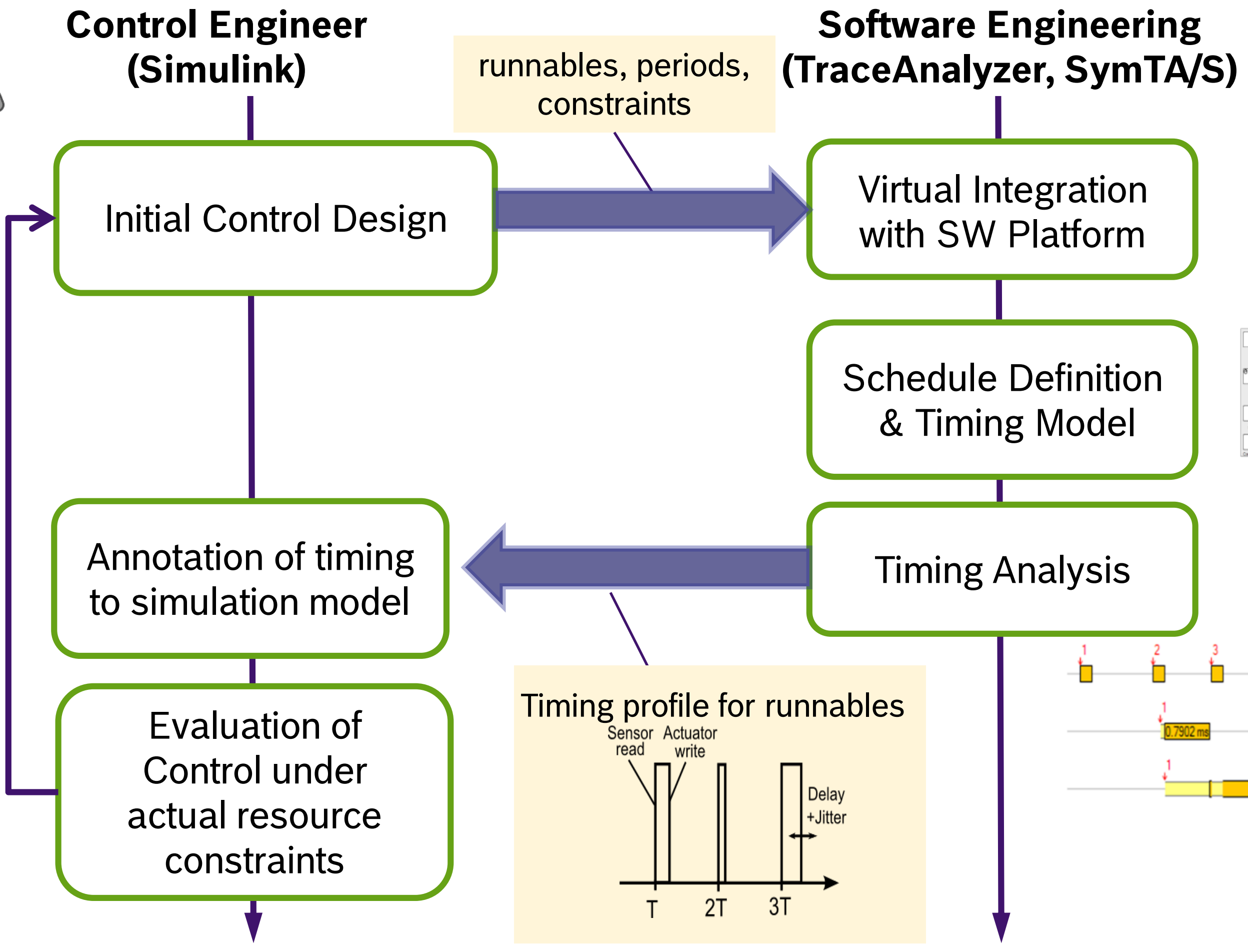
Timing-aware Control Design
Solution: Early Simulation Feedback



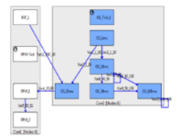


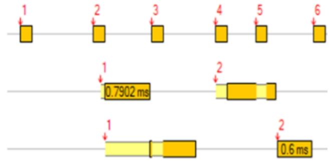
If needed: iterate long before code is actually implemented



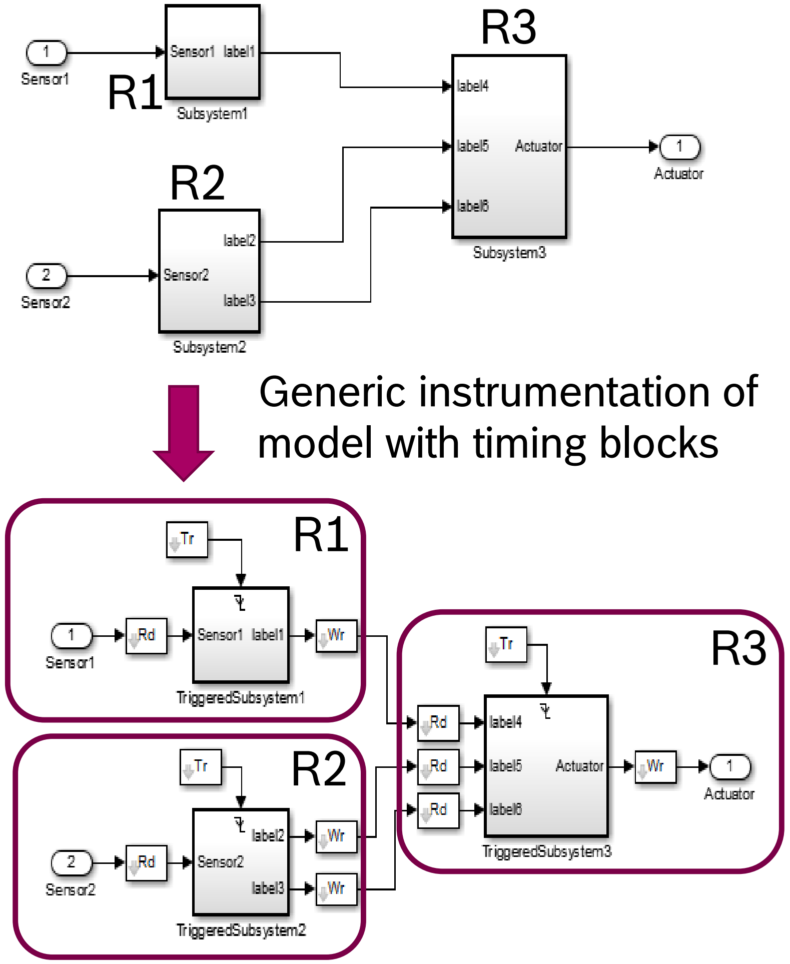


Control Engineer (Simulink)
Software Engineering (TraceAnalyzer, SymTA/S)
runnables, periods, constraints
Initial Control Design
Virtual Integration with SW Platform
Schedule Definition & Timing Model
Timing Analysis
Annotation of timing to simulation model
Evaluation of Control under actual resource constraints
Timing profile for runnables
Sensor read
Actuator write
Delay +Jitter
T
2T
3T

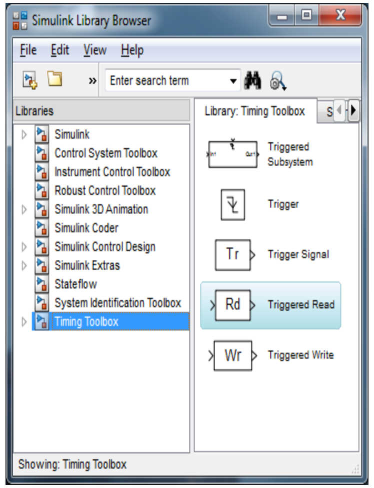




Timing-aware Control Design
Solution Details: Simulink Toolbox



Generic instrumentation of model with timing blocks

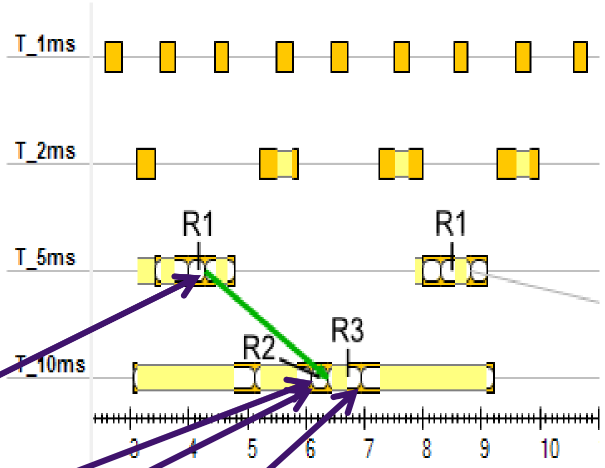


Blocks are configured with timing profiles (i.e. lists of time stamps)

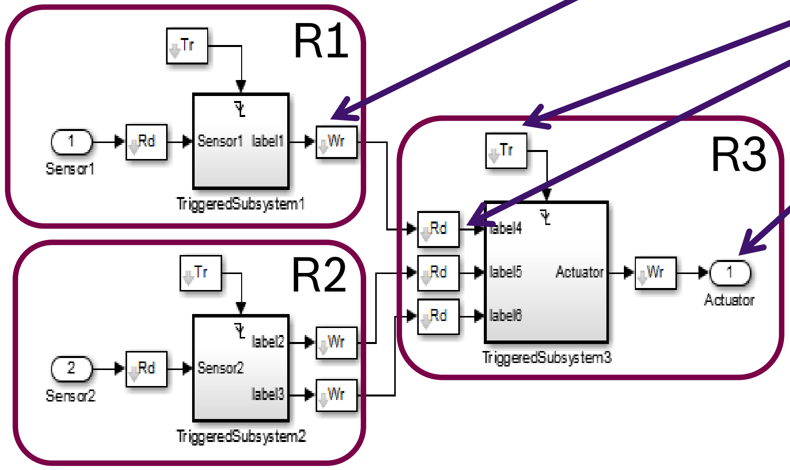
Timing-aware Control Design

Solution Details: Timing Profile Generation

Generates timing traces



& time stamps for instrumented model



More Details, Extensions & Case Study:
Lampke, Schliecker, Ziegenbein, Hamann,
"Model-Based Co-Engineering of Control Algorithms and Real-Time Systems,"
SAE World Congress 2015

Real-time Systems Engineering @ Bosch
Outline

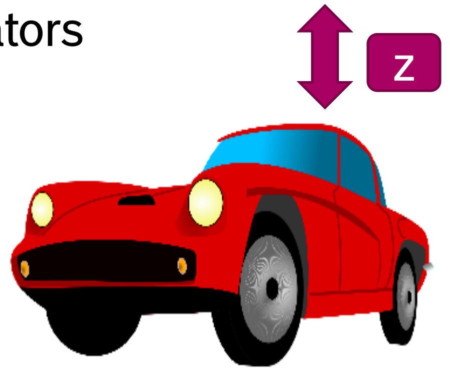
- Performance modeling & analysis of classical automotive systems
  - Motivation ... or the real complexity
  - Amalthea performance model
  - Current usage @ Bosch
  - Upcoming challenges
- Communication centric design in multi-core systems
  - Importance of cause-effect chains
  - Issues with concurrent execution in multi-core systems
  - Communication mechanisms as solution & impact on latencies
  - Experiments
- **Timing-aware control design**
  - Control and real-time systems engineering – two worlds collide
  - Co-engineering approach
  - **Example**

BOSCH

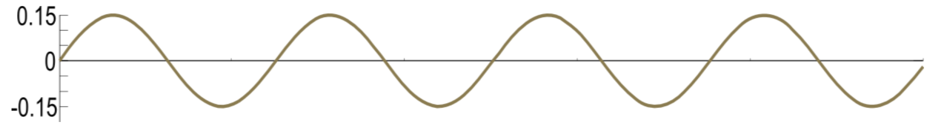
Timing-aware Control Design

Proof of Concept: Car Road Damping (Active Suspension)

- Is tool and concept applicable for complex systems?
- Case study: Damp car body acceleration with body control
- Complex system: 12 runnables, 7 accelerometers, 4 force actuators
- Exercise basic workflow with tool
- Tested different platform timing configurations
- Published SAE 2015 World Congress

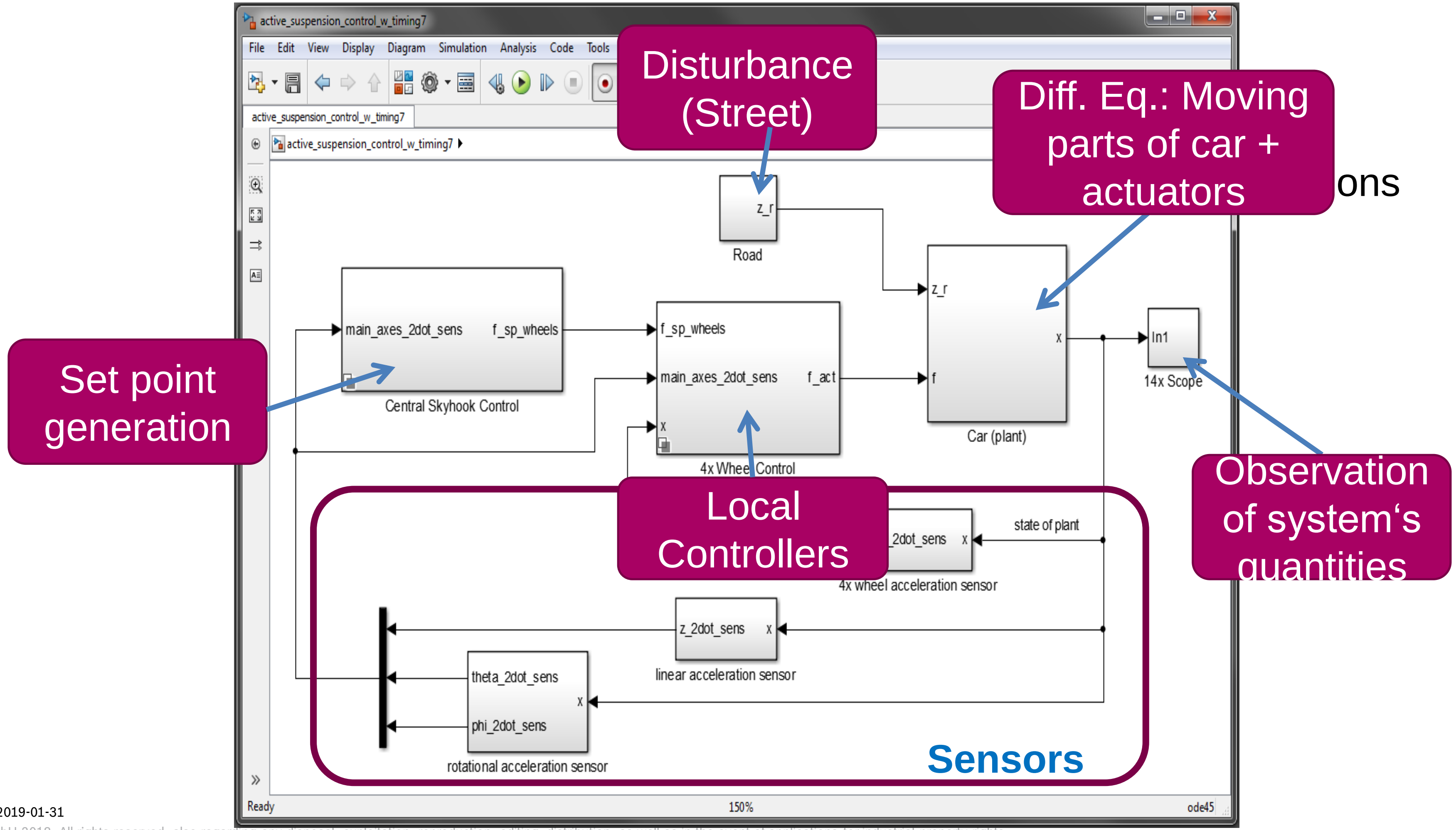


Road unevenness in m



Source of Model:
S. Ikenaga, F. L. Lewis, J. Campos and L. Davis (2000). Active Suspension Control of Ground Vehicle based on a Full-Vehicle Model. In Proceedings of the American Control Conference (ACC). Chicago, USA.

Timing-aware Control Design
Simulink model structure

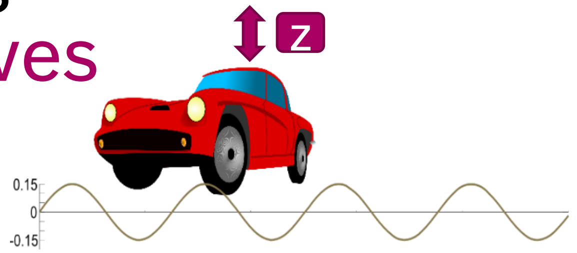


BOSCH

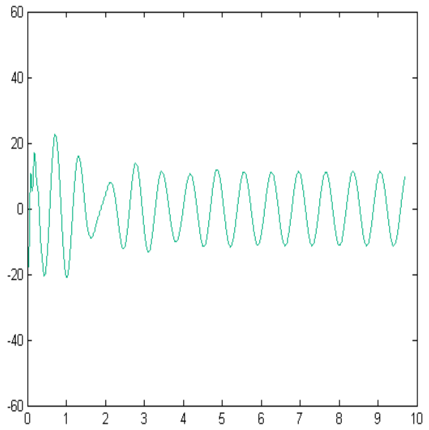
Timing-aware Control Design

Different Solutions Alternatives

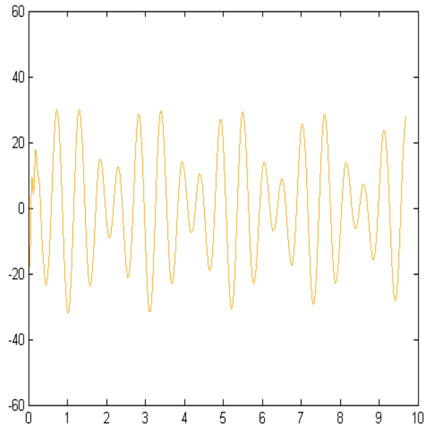
Road unevenness
in m



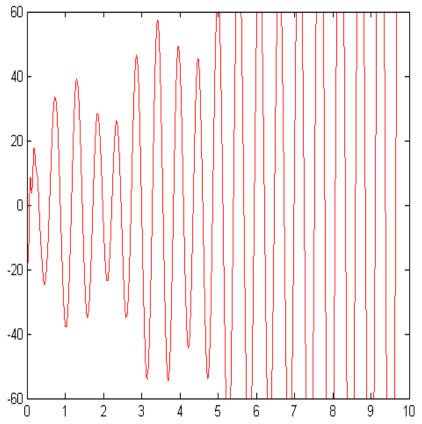
Results for one set of control parameters (Shown: z-Acceleration in m/s²):



Ideal timing



Timing Single Core



Timing Distributed ECUs

- Approach is able to handle complex systems
- Results are plausible and show expected differences in response

BOSCH

Timing-aware Control Design

Co-Engineering can start ...

Control Engineer

- I can see the effects of real-world timing on my control performance already in functional simulation

Software Engineer

- But now I want to redistribute tasks to balance load in multi-core...

THANK YOU

…AND HARALD MACKAMUL, JÖRG TESSMER, FALK WURST, TOBIAS BEICHTER, SYED AOUN RAZA, DIRK ZIEGENBEIN, JENS GLADIGAU, DAKSHINA DASARI, MICHAEL PRESSLER