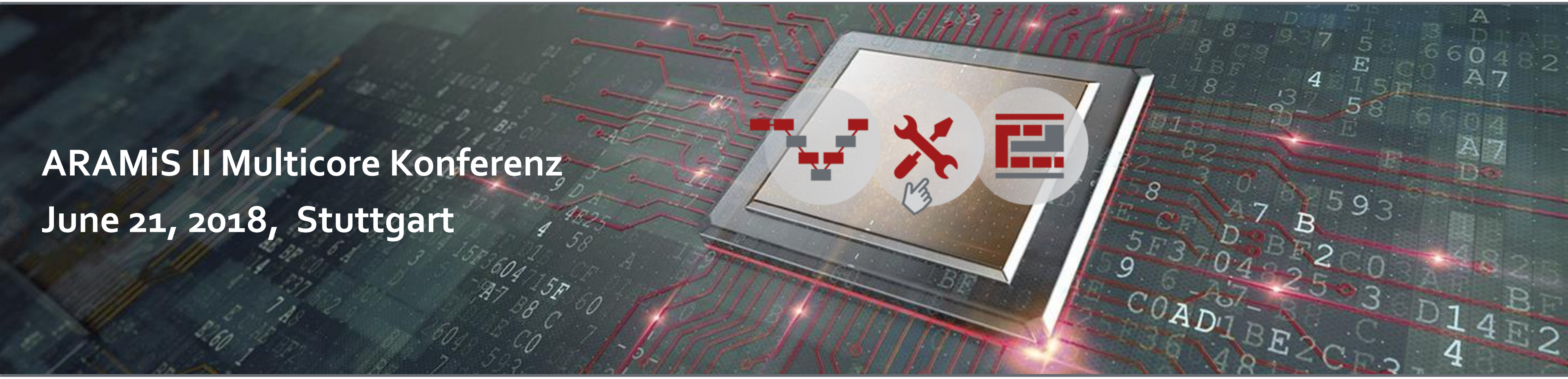
DEVELOPMENT PROCESSES I TOOLS I PLATFORMS
FOR SAFETY-CRITICAL MULTICORE SYSTEMS



ARAMiS II Multicore Konferenz
June 21, 2018, Stuttgart

Automotive Powertrain Demonstrator

Sebastian Kehr, Denso Automotive Deutschland GmbH, Eching

SPONSORED BY THE
Federal Ministry of Education and Research

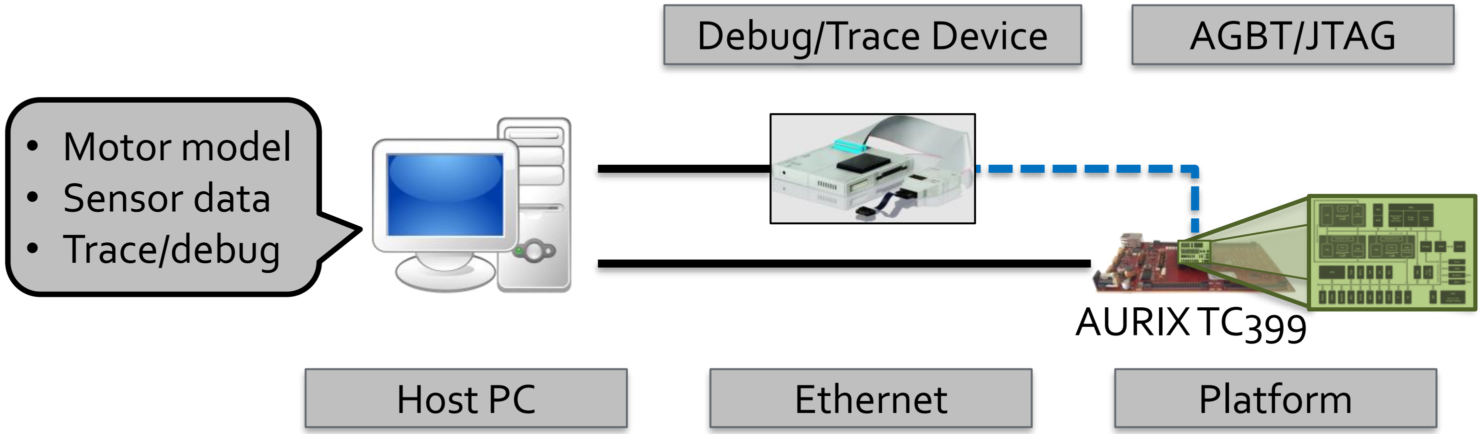
Agenda

- Demonstrator Setup
- Development Process
- Workflow
- Results & Outlook

Demonstrator Setup

aramis II
DEVELOPMENT PROCESSES I TOOLS I PLATFORMS
FOR SAFETY-CRITICAL MULTICORE SYSTEMS

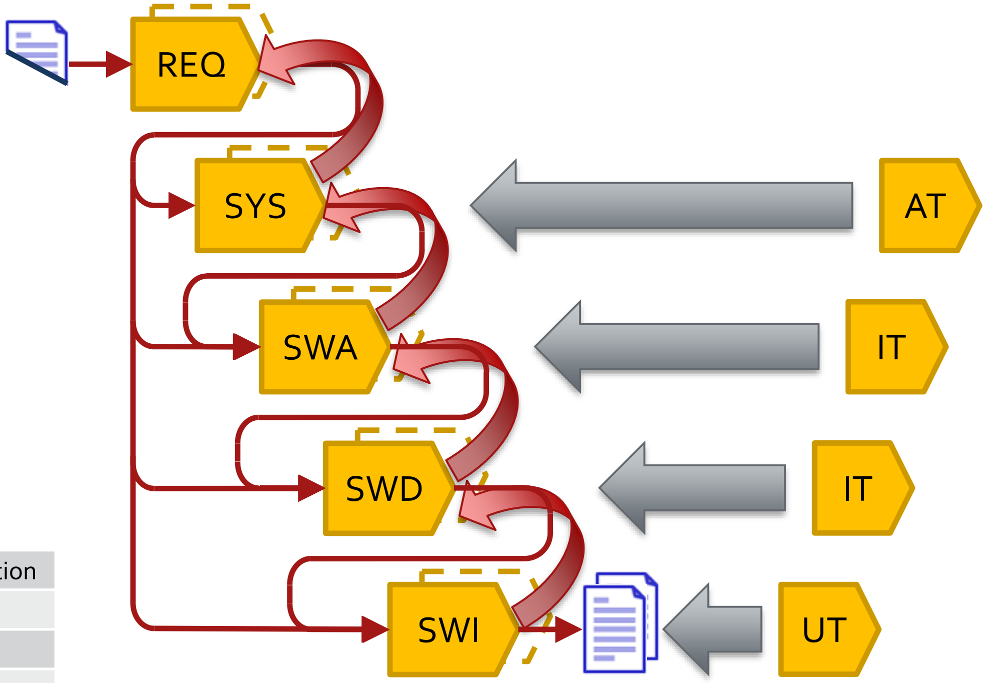
- Targets
  - Automated **migration of legacy software** to a multicore system
  - Demonstration of an efficient tool chain
  - Demonstration of an efficient migration result
- Use Case
  - Diesel Engine Management System



Development Process for Multicore Migration

- Multicore migration from legacy application requires reverse engineering
  - Analysis of data dependencies
  - Timing analysis with trace
  - Documentation ...

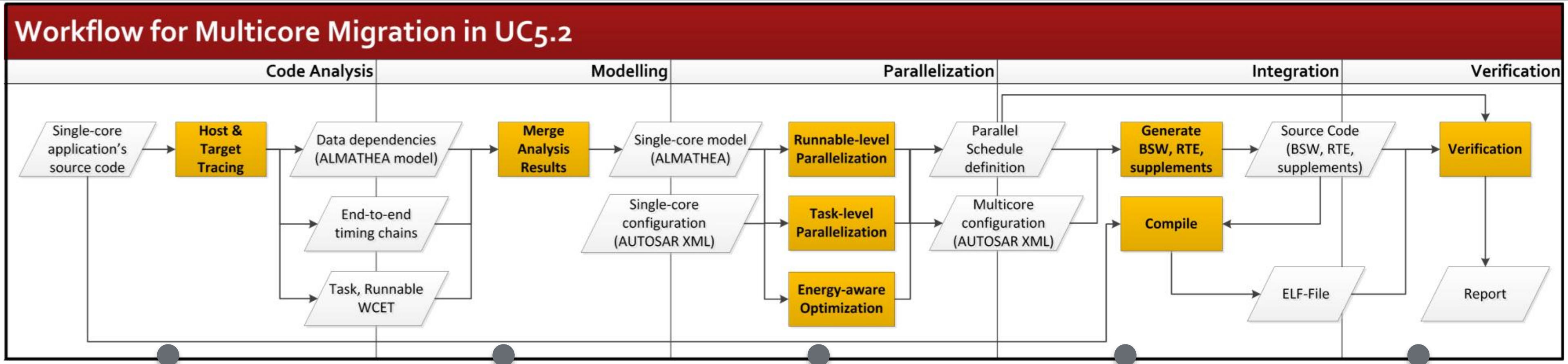
- Right side of the V
  - Schedule feasibility
  - Race conditions
  - System timing ...



Parallel Source Code

REQ	Requirements		SWI	Software Implementation
SYS	System Architecture		UT	Unit Test
SWA	Software Architecture		IT	Integration Test
SWD	Software Design		AT	Acceptance Test

Use-Case-Specific Workflow



Workflow for Multicore Migration in UC5.2
Code Analysis
Modelling
Parallelization
Integration
Verification
Single-core application's source code
Host & Target Tracing
Data dependencies (ALMATHEA model)
End-to-end timing chains
Task, Runnable WCET
Merge Analysis Results
Single-core model (ALMATHEA)
Single-core configuration (AUTOSAR XML)
Runnable-level Parallelization
Task-level Parallelization
Energy-aware Optimization
Parallel Schedule definition
Multicore configuration (AUTOSAR XML)
Generate BSW, RTE, supplements
Compile
Source Code (BSW, RTE, supplements)
ELF-File
Verification
Report

- Host Tracing
  - Silexica (Automotive Flow: Analyze)
  - Elektrobit (Tresos Studio / AutoCore)
- Target Tracing
  - AbsInt (TimingProfiler + TimeWeaver)
  - Symtavision

- AbsInt / Timing Architects (APP4MC)

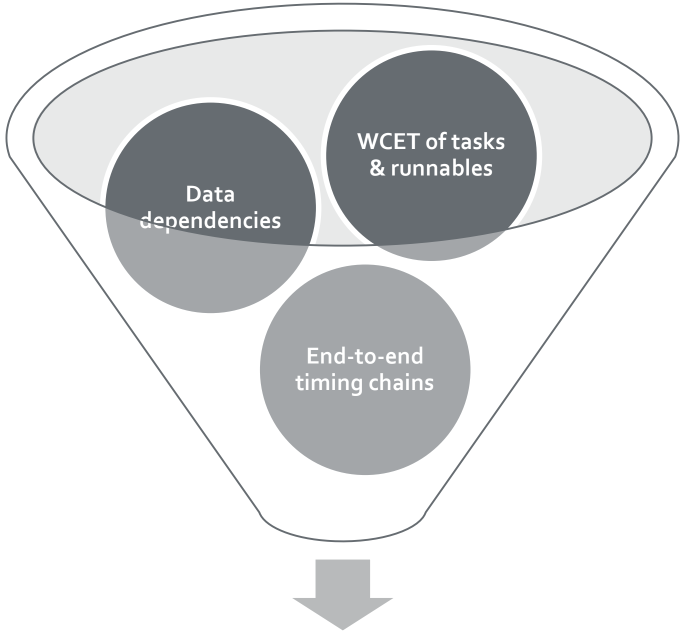
- Silexica (Automotive Flow: Optimize)
- Timing Architects (TA Simulator, TA Optimizer)
- Denso (Parcus)

- Elektrobit (Tresos Studio / AutoCore)
- Silexica (Automotive Flow: Implement)
- TU Braunschweig

- Uni Kiel (Lodin)
- Accemic (CEDAR)
- Uni Lübeck (TeSSLa)
- Fraunhofer IESE (FERAL framework)

Workflow: Code Analysis & Modelling

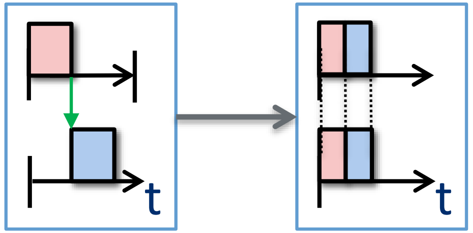
- Host tracing
  - Runtime environment: AUTOSAR on Windows
  - Static and dynamic dependency analysis with compiler-based analysis technology
- Target tracing
  - **Same configuration like host tracing**
  - Traces from execution on target to derive accurate WCET with hybrid WCET analysis
- Output: application model
  1. Data dependencies between runnables
  2. Worst-case latency of end-to-end timing chains
  3. Worst-case execution time (WCET) of runnables and tasks



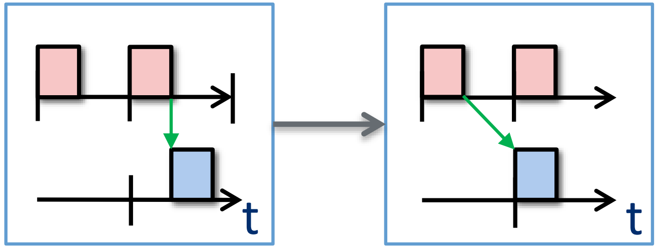
AMALTHEA model

Workflow: Parallelization

1. Runnable-level parallelization: Each task is split into multiple parallel running tasks
   - Data dependencies and WCET are considered
   - Start and termination of split tasks are synchronised



2. Task-level parallelization: Tasks are distributed to cores
   - Data distribution according to the logical execution time (LET) concept

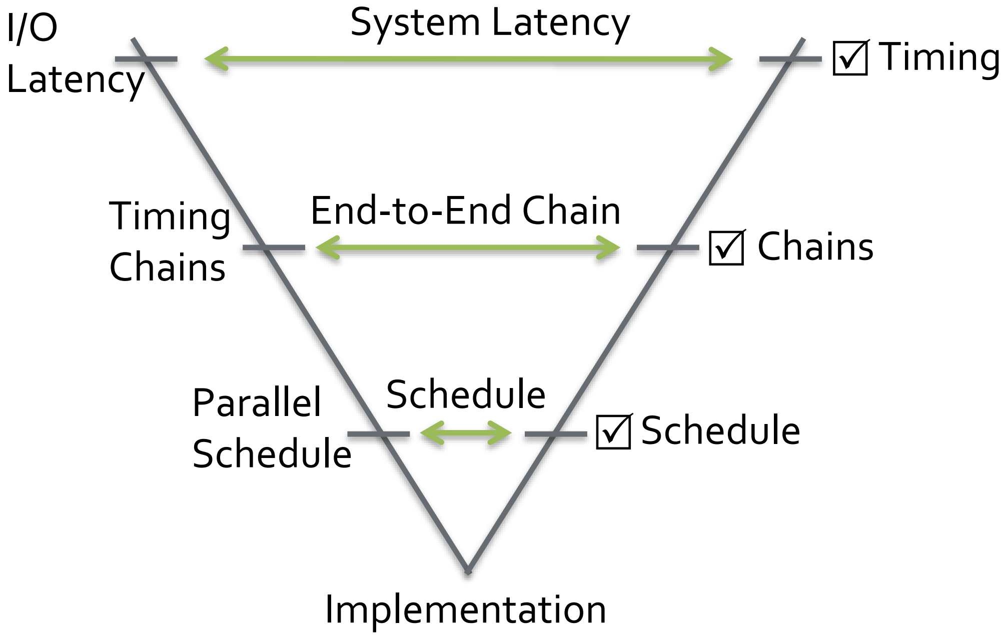


3. Energy Optimization: Combines approach 1 & 2
   - Considers impact on latency and the processor frequency
   - Parallel schedule quality (PSQ) metric quantifies success of parallelization

Workflow: Integration & Verification

- Generate code for AUTOSAR BSW + RTE
  - Add synchronisation primitives and/or communication buffers

- Verification
  - Comparison of timing between single-core and multicore software
  - Offline
    - Offline evaluation of measurements
    - Concurrency bugs
  - Online
    - Measurement on target and real-time evaluation with specialised FPGA



Results & Outlook

- Workflow / interoperability
  - Tool interoperability based on AMALTHEA model
- Partial automated migration
  - Runnable-level parallelization incl. memory mapping

- Next steps
  - Full automation of runnable-level parallelization
  - Detailing of verification methodology
  - Integration of LET concept and its automation

aramis II

ENTWICKLUNGSPROZESSE I WERKZEUGE I PLATTFORMEN
FÜR SICHERHEITSKRITISCHE MULTICORESYSTEME

STRUCTURED MULTICORE DEVELOPMENT

MULTICORE METHODS AND TOOLS

INDUSTRIAL PLATFORMS FOR MULTICORE SYSTEMS

Thank you for your attention!