Taking the Right Turn with Safe and Modular Solutions for the Automotive Industry
A Time-Triggered Middleware for Safety-Critical Automotive Applications

Ayhan Mehmet, Maximilian Rosenblattl, Wilfried Steiner

Software Architect
General ADAS Architecture

Sensors
- Front Camera
- Surround View
- Radar
- Laser
- Ultrasonic, Nano Radar

Preprocessing

Fusion
- Map Fusion
- Object Fusion
- e-Horizon

Application
- L3/4 Drive Pilot
- Lane Assist
- Parking
- Function X
- ...

Actuation
- Power
- Train
- Brake
- Steering

Central Fusion ECU (Raw Data Processing)
The ADAS Challenges

• ADAS need safety and performance
  • Demand for high-performance safety-capable μCs
• # Functions > # HW resources
• Demand for integration concept
• Multiple periods
• Demand for Scheduling concept
• End-to-end latency requirements
• Demand for accurate timing model
• OEM`s
  • Demand for fast system development
The ADAS Challenges

• ADAS need safety and performance
  • Demand for high-performance safety-capable μCs
• # Functions > # HW resources
• Demand for integration concept
• Multiple periods
• Demand for Scheduling concept
• End-to-end latency requirements
• Demand for accurate timing model
• OEM‘s
  • Demand for fast system development
The Challenge: ADAS Need Safety and Performance

- Sensor processing and data fusion need highest performance levels
- Steering and braking require up to ASIL-D

Today’s automotive safety controllers do not fulfill the high computing performance and memory requirements of usual ADAS applications

www.tttech.com
Joining Safety and Performance
Multi-μC-ECU

Control, ASIL C/D
Automotive μC

Fusion, ASIL B/C/D
Performance SoC

Sensor processing, ASIL A/B, QM
Performance SoC

Communication, Synchronisation
Ethernet Switch
(Deterministic)

FlexRay CAN CAN Ethernet Ethernet

ASIL C/D, ASIL B/C/D, ASIL A/B, QM
Sensor processing,
Communication,
Synchronisation
Functional Safety

- Memory Protection
- Timing/State Supervision

- Autosar OS
- BSW
- MCAL
- Automotive µC
- RTE
- Middleware
- Linux
- MCAL
- Performance SoC

- Safe Execution Platform: SEooC acc. ISO 26262
- Memory Protection, Communication Protection, Timing and State Supervision, Diagnostics up to ASIL D
The ADAS Challenges

✓ ADAS need safety and performance
  ✓ Demand for high-performance safety-capable µCs
    • # Functions > # HW resources
    • Demand for integration concept
  • Multiple periods
    • Demand for Scheduling concept
  • End-to-end latency requirements
    • Demand for accurate timing model
• OEM‘s
  • Demand for fast system development
Time-Triggered Communication, Deterministic Scheduling

- Jitter- and collision-free communication
- Deterministic, collision-free SWC schedule
- Deterministic data flows and latencies
Integration on an Event-Based Platform

1. Platform configuration
2. Single SWC test without consideration of other SWCs
3. Integration shows conflicts and collisions
4. Iterative rework until system runs stable
Integration Process on a Time-Triggered Platform

Robustness through clean allocation of resources

Parallel integration accelerates SW development

All software runs without jitter or variation

1. Platform configuration and application scheduling
2. Single SWC test within configured schedule
3. SWCs are instantly running together ("composability")
Integration Process on a Time-Triggered Platform

Integration process massively accelerated

1. Platform configuration and application scheduling
2. Single SWC test within configured schedule
3. SWCs are instantly running together ("composability")
The ADAS Challenges

☑ ADAS need safety and performance
  ✔ Demand for high-performance safety-capable μCs
☐ # Functions > # HW resources
  ✔ Demand for integration concept
  • Multiple periods
  • Demand for Scheduling concept
• End-to-end latency requirements
  • Demand for accurate timing model
• OEM‘s
  • Demand for fast system development
TTA Task Scheduling Model

- $t_A$: activation point
- $t_D$: deadline
- $T_R$: runtime; to be allocated

$T_R < t_D - t_A$
TTA Task Scheduling Model

- A Task $\tau_i$ is characterized as following:
  - WCET $C_i$
  - period $T_i$
  - offset (phase) $\phi_i$
  - deadline $D_i$
  - priority $P_i$
  - CPU affinity $A_i$

The TTA scheduler solves a variable assignment problem.
TTA Task Scheduling Model

\[ \sum_{i=1}^{n} T_i = T_R < t_D - t_A \]

- \( t_A \): activation point
- \( t_T \): trigger point
- \( t_D \): deadline
- \( T_1, T_2, ..., T_n \): allocated CPU time
- \( T_R \): runtime

www.tttech.com
Utilizing unused CPU time of dedicated slices is the key mechanism of the **Soft TTA Approach**:

**Case 1:** SW-C 1 finishes within the dedicated time window

**Case 2:** SW-C 1 finishes in leftover time

www.tttech.com
Soft-TTA Task Scheduling Model

\[ T_{TYP} < \sum_{i=1}^{n} T_i = T_R < T_{WCET} \]

\( t_A \): activation point
\( t_T \): trigger point
\( t_{SDL} \): Soft Deadline (\( t_{SDL} \leq t_{HDL} \))
\( t_{HDL} \): Hard Deadline (\( t_{HDL} = t_D \))
\( t_D \): deadline
\( T_1, T_2, ..., T_n \): allocated CPU time

\( T_R \): runtime
\( T_{WCET} \): worst case execution time
\( T_{TYP} \): „typical“ execution time
The ADAS Challenges

- ADAS need safety and performance
  - Demand for high-performance safety-capable μCs
- # Functions > # HW resources
  - Demand for integration concept
- Multiple periods
  - Demand for Scheduling concept
  - End-to-end latency requirements
  - Demand for accurate timing model
- OEM‘s
  - Demand for fast system development
End-to-End Communication Latency Guarantees

- Scheduling of runnables with defined maximal end-to-end latency
- Static schedule as timing model allows to compute worst case latency easily
- Timing supervision: effective model verification during runtime

www.tttech.com
Definition: sequence of runnables with a given maximal end-to-end latency

Static schedule tables allow to easily compute worst case latency

End-to-end latency can be optimized by specifying additional constraints for involved tasks

Example:
**E2E Latency Guarantees**

**Scheduling Perspective**

- Definition: sequence of runnables with a given maximal end-to-end latency
- Static schedule tables allow to easily compute worst case latency
- End-to-end latency can be optimized by specifying additional constraints for involved tasks

**Example:**

![Diagram showing sequence of runnables and end-to-end latency](image-url)
Actual Schedule with End-to-End Latencies
The ADAS Challenges

- ADAS need safety and performance
  - Demand for high-performance safety-capable μCs
- # Functions > # HW resources
  - Demand for integration concept
- Multiple periods
  - Demand for Scheduling concept
- End-to-end latency requirements
  - Demand for accurate timing model
  - OEM‘s
    - Demand for fast system development
The Challenge

Demand for more services

Increased number of ECUs

Growth in sharing software and functionality between the ECUs

Diversity of the hardware and communication standards

Re-development of the software

High: Complexity, development time and cost
The solution

Following the AUTOSAR software architecture

- Decoupling of Application SW from HW
- Modularity,
- Scalability,
- Re-usability, …
Conclusions

✅ To fulfill the safety and performance requirements of today’s ADAS systems an integration platform must support seamless integration of safety- and performance microcontrollers.

✅ A time-triggered architecture explicitly models the temporal properties of SW-Cs, which supports the

✅ prediction of temporal characteristics of event chains (no worst-case analysis necessary)

✅ reduction of integration testing efforts (no side-effects caused by SW-C microtiming, re-use of test results)

www.tttech.com
Development Perspective

Good Limitations

- Development is a creative process
- For defined quality, there have to be limitations
- That is the basic idea behind
  - MISRA
  - HIS
  - The V-model (used in ISO26262)
  - Every development guideline
Development Perspective
Good Limitations

Inherent property of limitations in the development context:

Limited development possibilities draw attention to the limits and raise additional thoughts *early in the development process*
Development Perspective
Good Limitations

☑️ In a non-TTA system, all software components together have limited runtime

☑️ TTA introduces limited runtime for every single software component
Typically, the following questions are raised immediately by customers, developers, …

- What about interrupt load?
- What about memory wait states?
- What happens when my SW misses the deadline?
- etc.
Development Perspective

Typical Questions

These problems usually arise late in the development process

- The integration test phase or
- The system test phase

TTA forces the developers (SW-C and system) to think about that early → more time for solutions