The road from GPU-powered prototypes to production ECUs
Status Quo – Cars as computers with wheels

1:1 relation between functionality and embedded device is standard

Since Years:
• Up to 100 small ECUs (Electronic control unit) serving a dedicated function like window lifter or climate control
• Several and highly specialized communication channels between ECUs (CAN, CANFD, FlexRay, LIN, Ethernet)

Today:
• Multi-core processors in use on ECUs serving multiple functions
• Software for ECUs is statically configured and flashed with a single image containing all applications and basic software

Challenge:
Integration of software of several parties cross company
Status Quo – AUTOSAR Standard enabling work split

Decouple application software from hardware
Standardize software interfaces
Standardize configuration concepts
Design the complete vehicle application software over all ECUs
Status Quo – Vehicle infrastructure

Solutions for Classic AUTOSAR ECUs

- Basic software and services
- Tooling for standardized configuration
- Single- and multi-core operating systems
- **Functional Safety solutions**
  ISO 26262 certified up to ASIL-D
- **Embedded Security solutions**
- **Certifiable, e.g. Onboard Diagnostics**
- Automotive Ethernet software
Status Quo – Today’s Vehicle infrastructure

GPUs only in use in the infotainment domain!
The road from GPU-powered prototypes to production ECUs
How can we achieve the next level of mobility?
The road from GPU-powered prototypes to production ECUs

Requirements and challenges

Software over-the-air-updates
- New vehicles features
- Updates and patches
- Silent testing
- Security

Dynamic deployment
- User-driven updates
- Network accessible sensors & actuators

Remote analytics and diagnostics
- Car as a sensor
- Remote diagnostics
- Fleet campaigns

Developer oriented, target independent environment
- Environment independent software
- Easy qualification and deployment
- Small, encapsulated and exchangeable software services (microservice)

Dependable systems
- Safety
- Security
- Availability
- Reliability
- Maintainability
Possible next generation vehicle infrastructure architecture

- Mix of: classic microcontrollers and powerful “cellphone” processors
- Mix of: classic automotive networks and cloud connectivity
- Mix of: safety-relevant functions and apps installed by the user
Sensor fusion system requirements spanning across

Sensor Data Fusion
- Positioning
- Object Fusion
- Grid Fusion
- Road and Lane Fusion
- Vehicle Database

Function Specific Views

Situative Behavior Arbitration
- Situation Analysis
- Path Planning
- Behavior

Motion Management
- Trajectory Control
- Longitudinal Control
- Lateral Control
- HMI Management

Vehicle Abstraction - Sensors

Vehicle Abstraction - Actuators

Safety Management
Grid Fusion in action

Bayes based Grid Fusion
- Traditional occupancy fusion
- Storing probability of occupancy
- Needs less memory due to smaller state vector

Height Map
- Storing height value and confidence
- Processes 3D data
- Use-case: ramp/hole detection, preprocessing for other fusion algorithms

Demster-Shafer based Grid Fusion
- State of the art occupancy fusion
- Storing evidence of occupancy/free
- Various sensor models
- Use-case: free space detection, path planning
Sensor fusion system

The road from GPU-powered prototypes to production ECUs

Scalable inputs

Grid Fusion

Algorithm core

For such application we need (parallel) computing power!

Scalable, standardized interfaces

Interface 1
high volume
for central ECU use

Interface 2
mid volume
for FlexRay / Ethernet

Interface 3
low volume
for CAN

Sonar Sensor Interface
Sonar Sensor Adapter

Lidar Sensor Interface
Lidar Sensor Adapter

Radar Sensor Interface
Radar Sensor Adapter

Scalable inputs

Sonar
Sonar

Lidar
Lidar

Radar
Radar

Sonar Sensor Interface
Sonar Sensor Adapter

Lidar Sensor Interface
Lidar Sensor Adapter

Radar Sensor Interface
Radar Sensor Adapter

For such application we need (parallel) computing power!
The road from GPU-powered prototypes to production ECUs

Sensor fusion system

- Positioning
- Object Fusion
- Grid Fusion
- Lateral Control
- Longitudinal Control
- Trajectory Control
- Sensor Data Fusion
- Motion Management
- Behavior
- Situation Analysis
- Path Planning
- Safety Management
- Sensor fusion system

How to achieve system abstraction?

Scalable inputs

Grid Fusion

- Sonar
  - Sensor Interface
  - Sensor Adapter
- Lidar
  - Sensor Interface
  - Sensor Adapter
- Radar
  - Sensor Interface
  - Sensor Adapter

Scalable, standardized interfaces

- Interface 1
  - high volume for central ECU use
- Interface 2
  - mid volume for FlexRay / Ethernet
- Interface 3
  - low volume for CAN

Scalable inputs

Sensor fusion system
Next generation of standardized basic software
Adaptive AUTOSAR

The road from GPU-powered prototypes to production ECUs

Adaptive AUTOSAR Services

Adaptive AUTOSAR Basis

(Virtual) Machine / Hardware
### Possible high performance controller architecture for SOP in 2019

#### Diagram:

- **1002**
- **Performance Partitions for Vehicle & Consumer Functions**
  - App
  - Adaptive AUTOSAR QM
  - Adaptive AUTOSAR Safety
  - LINUX OS
  - LINUX OS
  - Automotive-grade Hypervisor
  - Performance Cores and HW accelerators
- **Safety Partition**
  - App
  - Classic AUTOSAR Safety
  - AUTOSAR OS
  - Safety OS
  - Safety Cores
- **Security Partition**
  - App
  - Classic AUTOSAR Safety
  - Security TEE
  - Trusted OS
  - HSM

---

**Legend:**
- MCU
- Safety
- Safety Core
- Security Core
- Trusted OS
- TEE

---

**Note:**
This diagram represents a logical architecture for high performance controllers, focusing on the transition from GPU-powered prototypes to production ECUs. It outlines the integration of different AUTOSAR variants, Linux OS, and security partitioning to address performance, safety, and security requirements for vehicle and consumer functions.
Worksplit in automotive supply chain

The road from GPU-powered prototypes to production ECUs

Function development: SW supplier
Function integration: Tier 1

Adaptive AUTOSAR QM
POSIX OS
Drivers

Adaptive AUTOSAR Safety
POSIX OS
Drivers

Classic AUTOSAR
AUTOSAR OS
MCAL (Drivers)

Performance Cores and HW accelerators

MCU

Interface Definition: OEM
Network deployment: OEM

Middleware development: Basic software supplier
Middleware configuration: Tier 1

OS development: OS supplier
OS configuration integration: Tier 1

Driver development: Hardware supplier
Driver integration: Tier 1

Hypervisor development: Hypervisor supplier
Resource partitioning: Tier 1

Security Partition

App

App

App

HSM
The road from GPU-powered prototypes to production ECUs

Standardization in AUTOSAR

- Software component description
- Software component configuration

Adaptive AUTOSAR QM
- POSIX OS
- Drivers

Adaptive AUTOSAR Safety
- POSIX OS
- Drivers

Classic AUTOSAR
- AUTOSAR OS
- MCAL (Drivers)

AUTOSAR OS
- Drivers

Performance Cores and HW accelerators

MCU

Automotive-grade Hypervisor

Safety Cores

HSM

POSIX OS

Drivers

Standardized APIs
- Standardized network configuration

Classic AUTOSAR:
- Standardized functionality and configuration

Standard APIs
- Standardized configuration

Safety Cores

TEE

HSM

All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.
How to use GPUs for automotive applications?

- Multiple independently developed applications
- Applications that are developed without focus on specific HW architecture
- Applications that reside in different hypervisor partitions

How to provide reasonable freedom from interference between those applications
Integrating GPUs in software infrastructure
Current integration in (most) prototypes

Performance Partitions for Vehicle & Consumer Functions

- Adaptive AUTOSAR QM
- POSIX OS
- Drivers
- Adaptive AUTOSAR Safety
- POSIX OS
- Drivers
- AUTOSAR OS
- MCAL (Drivers)

Safety Partition

- Classic AUTOSAR Safety
- Safety OS
- MCAL (Drivers)

Security Partition

- Security TEE
- Trusted OS
- Drivers

Performance Cores and HW accelerators

- Safety Cores
- HSM

MCU
The road from GPU-powered prototypes to production ECUs

Current integration in (most) prototypes

- Adaptive AUTOSAR QM
- POSIX OS
- Drivers
- GPU Library + Driver
- Performance Cores and HW accelerators
- It actually works!
  - Application becomes hardware-dependent
  - Applications must explicitly share the GPU – application dependencies
  - No integration into Health Monitoring

Performance Partitions for Vehicle & Consumer Functions
- Classic AUTOSAR Safety
- Safety OS
- MCAL (Drivers)

Safety Partition
- Safety Cores
Standardized basic software – Adaptive AUTOSAR

Goal:
• Uncouple application design from the hardware it is deployed upon

Functionality to be included:
• Libraries for using GPUs for ADAS applications
  • E.g. Tensorflow, Parallel STL, OpenCV
  • Build on top of e.g. CUDA, SYCL

Likely not standardized before 03/2019
Sharing a GPU across virtual machines

**Emulation**
- VM
- GPU-Drv
- vGPU
- User Space
- Hypervisor
- Hardware
- GPU
- Performance Controller

**Passthrough**
- VM
- GPU-Drv
- User Space
- Hypervisor
- Hardware
- GPU
- Performance Controller

**Driver in Guest**
- VM
- vGPU
- GPU-Drv
- User Space
- Hypervisor
- Hardware
- GPU
- Performance Controller

**GPU Virtualization**
- VM
- GPU-Drv
- GPU Mngmt
- User Space
- Hypervisor
- Hardware
- GPU
- Performance Controller
Are we done yet?

What have we achieved so far:

• Standardization of APIs and abstraction from hardware
  – Allows to develop applications independent of ECU projects

• Enabling GPU virtualization
  – Allows sharing hardware resources in a more efficient manner

Integration should work. Now let’s make things safe and secure.
The reality: Automotive recalls in the US

<table>
<thead>
<tr>
<th>Year</th>
<th>Vehicles affected (Milion)</th>
<th>Vehicles affected (Per day)</th>
<th>Recalls (Total)</th>
<th>Vehicles sold (Million)</th>
<th>Recalls per sold vehicle</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014</td>
<td>51.0</td>
<td>139,726</td>
<td>779</td>
<td>16.45</td>
<td>3.1</td>
</tr>
<tr>
<td>2015</td>
<td>51.2</td>
<td>140,274</td>
<td>868</td>
<td>17.39</td>
<td>2.9</td>
</tr>
<tr>
<td>2016</td>
<td>53.2</td>
<td>145,753</td>
<td>927</td>
<td>17.46</td>
<td>3.0</td>
</tr>
</tbody>
</table>

- Software-related recalls have gone from less than 5 % in 2011 to 15 % by end of 2015\(^1\)
  \(\Rightarrow\) roughly 7.68 Million software related car recalls in 2015
- Software vendors are liable for their products in the automotive sector

\(^1\)Stout Risius Ross (SSR), Automotive Warranty & Recall Report 2016
Software development for automotive (ASPICE)

- Quality- and verification driven
- Common standard: Automotive SPICE v3.1 for development processes
- In addition: most OEMs have further standards
- Process assessments are regularly performed by customers: OEM -> Tier-1 -> Tier-2 -> ...
- A Tier-1 is fully liable for the complete product, including the software
## Functional Safety according to ISO 26262

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO 26262 has</td>
<td>Overall Safety Management</td>
<td>Item Definition</td>
<td>System Dev. Initiation</td>
<td>HW Safety Requirements</td>
<td>Initiation</td>
<td>Distributed Development</td>
<td>Requirement Decomposition</td>
<td></td>
</tr>
<tr>
<td>• 10 parts</td>
<td></td>
<td></td>
<td></td>
<td>HW Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 500 pages</td>
<td></td>
<td></td>
<td></td>
<td>HW Architectural Metrics</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 43 Chapters</td>
<td></td>
<td></td>
<td></td>
<td>HW Failure Rate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 600 Require</td>
<td></td>
<td></td>
<td></td>
<td>HW Integration and Testing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ments</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 100 Work P</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>roducts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 180 Methods</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### ISO 26262 Requirements
- 10 parts
- 500 pages
- 43 Chapters
- 600 Requirements
- 100 Work Products
- 180 Methods

ISO 26262 has

- 10 parts
- 500 pages
- 43 Chapters
- 600 Requirements
- 100 Work Products
- 180 Methods

Only the parts 1-9 are normative
Functional Safety Requirements (ISO 26262)

The road from GPU-powered prototypes to production ECUs

Safety Requirements

Automotive Safety Integrity Level ASIL
ASIL = Severity x Exposure x Controllability
Functional Safety Requirements (ISO 26262)

Safety Requirements

**Automotive Safety Integrity Level ASIL**

\[ \text{ASIL} = \text{Severity} \times \text{Exposure} \times \text{Controllability} \]
Functional Safety Requirements (ISO 26262)

Safety Requirements

Automotive Safety Integrity Level ASIL
ASIL = Severity x Exposure x **Controllability**

Drive in the city @35mph

Collide with pedestrian @35mph

Functional Description

Hazard & Risk Analysis
Hazard and Risk Assessment in a nutshell

**Hazard**
Potential source of harm, caused by malfunctioning behavior of the item
Example: Not detecting a pedestrian on the street

**Operational situation**
Scenario that can occur during a vehicle’s life, in which the hazard creates a considerable risk
Example: Driving at speed > 35 mph on a street operating with an activated Highway Chauffeur

**Possible avoidance actions**
Things that the driver or other involved person are able to do in order to avoid the harm
Example: Driver immediately brakes with full power

**Severity (S0-S3)**
Impact of the harm
Example: Life-threatening injuries => S3

**Exposure (E0-E4)**
Probability of the operational scenario
Example: High probability => E4

**Controllability (C0-C3)**
Probability that one of the involved persons is able to avoid the harm
Example: Normally not controllable => C3

<table>
<thead>
<tr>
<th></th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>QM</td>
<td>QM</td>
<td>QM</td>
</tr>
<tr>
<td>E2</td>
<td>QM</td>
<td>QM</td>
<td>QM</td>
</tr>
<tr>
<td>E3</td>
<td>QM</td>
<td>QM</td>
<td>A</td>
</tr>
<tr>
<td>E4</td>
<td>QM</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E1</td>
<td>QM</td>
<td>QM</td>
<td>QM</td>
</tr>
<tr>
<td>E2</td>
<td>QM</td>
<td>QM</td>
<td>A</td>
</tr>
<tr>
<td>E3</td>
<td>QM</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>E4</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E1</td>
<td>QM</td>
<td>QM</td>
<td>A</td>
</tr>
<tr>
<td>E2</td>
<td>QM</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>E4</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
</tbody>
</table>
Functional Safety Requirements (ISO 26262)

Safety Requirements
- **Severity 3**: Severe injuries expected.
- **Exposure 4**: Pedestrians in car’s path likely.
- **Controllability 3**: Driver is sleeping.

ASIL = Severity x Exposure x **Controllability**

= **ASIL-D**
Strategies to achieve safety for highly automated driving

- Redundant channels (e.g.: 1oo2D)
- 3 Layer safety solution (E-Gas concept)
- Safety parts fully certified up to ASIL-D
Cybersecurity engineering

• The standard is being worked on:
  ISO/SAE 21434 – Cybersecurity Engineering
• Contains general requirements, will probably include requirements for software updates
• Publication: end of 2019
  → Projects with an **SOP in 2019 / 2020** need to take it into account.

General Outline (Draft)

1. Scope
2. Normative references
3. Terms and Definitions
4. Introduction
5. Cybersecurity Engineering – General Requirements
6. Cybersecurity Management – Govern Process
7. Cybersecurity Management – Engage Process
8. Cybersecurity Engineering – Develop Process
9. Cybersecurity in Production & Post-production
10. Appendixes
Security solutions

- Hardware enhanced cryptography
- Embedded security mechanisms
- Cyber security fleet monitoring (IDS)
- Cyber security analysis & response
- Software update over-the-air

End-to-end security
External interface
In-vehicle network
Summary

Standardization of Middleware to develop application independent for high performance ECUs

Enabling standardized GPU virtualization
Allows sharing hardware resources in an efficient way

EB extended the safety product portfolio to fulfill the requirements of high performance ECUs

EB & Argus are offering an end-to-end-security solution for the future vehicle architecture
EB tresos solution for NVIDIA DRIVE PX 2

Adaptive & Classic AUTOSAR for DrivePX planned to be available at October 2018!

- Integrating technologies from NVIDIA, Infineon (safety processor) and EB
- The EB software provides seamless integration capability of Linux and AUTOSAR applications

Visit our booth at the GTC #532