Multicore platform towards automotive safety challenges

Romuald NOZAHIC
European Application Engineer

mentor.com/automotive
Agenda

Multicore Consolidation
- Market Trends
- Different way to take advantages
- Complexity overview

Safety
- 3 solutions
Multicore is not new concept
Multicore is not new concept
Consolidation on the SoC level
Consolidation on a system level

ECU

IVI

DRIVER INFO

ADAS
Multicore Configurations

Homogeneous uAMP
Multicore Configurations

Homogeneous uAMP

- Linux (SMP)
- RTOS

Cortex A
Cortex A
Cortex A
Cortex A
Multicore Configurations

Homogeneous uAMP

Linux (SMP)

RTOS (SMP)

Linux (master)

RTOS (master)

RTOS

Cortex A

Cortex A

Cortex A

Cortex A

Cortex A

Cortex A

Cortex A

Cortex A

Cortex A
Multicore Configurations

Homogeneous uAMP
Multicore Configurations

Homogeneous uAMP

- Linux
- RTOS
- Bare Metal Env.

Heterogeneous uAMP

- Linux
- RTOS
- Bare Metal Env.
Multicore Configurations

Homogeneous uAMP

Heterogeneous uAMP
Multicore Configurations

Homogeneous uAMP

Heterogeneous sAMP
Multicore Configurations

Homogeneous uAMP

Heterogeneous sAMP
Multicore Configurations: Heterogeneous

- Linux or RTOS (master)
- Linux or RTOS or BME
- RTOS or BME
- RTOS or BME
- RTOS or BME

Hypervisor

- Cortex A
- Cortex A
- Cortex M
- Cortex M
- Soft Core

14 use cases
* N use cases
* M use cases
Complexity Skyrockets

Extreme complexity is introduced with general purpose development

- System architecture
- Configuration
- Booting
- Debugging
- Separation
- Device sharing
- Inter-processor communication
- Security
Multicore Framework

Comprehensive solution for heterogeneous multicore development that enables:

» Configuring and deploying multiple operating systems and applications across heterogeneous processors

» **Boot**ing multiple operating systems efficiently and in a coordinated manner across heterogeneous processor cores

» **Commun**icating between isolated sub-systems on a multicore processor or between heterogeneous processors

» **Visualizing** interactions between heterogeneous operating systems on heterogeneous multicore for debugging and optimization

» **Proprietary functionality that allows interoperability** of open source and proprietary environments with all the above capabilities
Configuring and deploying multiple operating systems and applications across heterogeneous processors

- Booting multiple operating systems efficiently and in a coordinated manner across heterogeneous processor cores
- Communicating between isolated sub-systems on a multicore processor or between heterogeneous processors
- Visualizing interactions between heterogeneous operating systems on heterogeneous multicore for debugging and optimization
- Proprietary functionality that allows interoperability of open source and proprietary environments with all the above capabilities

Comprehensive solution for heterogeneous multicore development

Multi Core/OS Technologies

- Tools
- Shared Graphics
- Trusted Supervised Native

Proprietary functionality that allows interoperability of open source and proprietary environments with all the above capabilities
Security and Safety via Separation

**Safety**: Protecting the world from the device

**Security**: Protecting the device from the world

**Mixed criticality**: Protecting of security or safety critical parts of the device from other parts of the device

ISO26262-6 requires “freedom from interference”. If two systems can interfere with each other, they must be certified to the highest ASIL level of the two. Secure separation aims to eliminate such interference.
Use Case 1:

Physical Separation aka AMP
What the system looks like today

Multicore Device running one Operating System

- Migrating to multicore device for the next generation or project
- Need to consolidate applications that require real time and determinism with applications requiring Linux networking or graphics services
- Addressing performance constraints of existing design
Multicore Device running multiple Operating Systems

- Single user interface for Configure, Edit, Debug, Optimize work
- Framework to configure, boot, execute and communicate across cores and Operating Systems
- Take full advantage of the underlying ‘silicony goodness’ 😊
How this could be accomplished
Use Case 2:

Separation using Software Enforcement
What the system looks like today

Multiple boards running various Operating Systems and dedicated applications

- Migrating to multicore device for the next generation or project
- Need to consolidate applications that require real time with Linux
- Must share displays and other resources
Consolidation to a single Heterogeneous Multicore SoC running multiple Operating Systems and Applications

- Virtualizing GPU to either control multiple displays per application or layer multiple applications on a single display (1:1, 1:N, N:1)
- Framework to configure, boot, execute and communicate across domains in safe and reliable matter
How this could be accomplished

- **Infotainment Display**
  - USB 2CAN
  - CAN BUS
  - BusMaster CAN Vehicle Simulator
  - CAN Linux
  - AUTOSAR & CAN stack on M4

- **Cluster Display**
  - Hypervisor (2xA15) + GPU sharing
  - FPD-Link Display 12” (1280x480)
  - FPD-Link Touch Display 10” (1280x800)

- **FPD-Link Display**
  - 12” (1280x480)
  - 10” (1280x800)
Use Case 3:

Separation using Hardware enforcement
What the system looks like today

One or more cores running applications of various security or robustness levels
- Migrating to multicore or more powerful device for the next project
- Need to consolidate applications that require secure and non secure apps
What the system will look like

One or more cores running applications of various security or robustness levels

- Migrating to multicore or more powerful device for the next project
- Need to consolidate applications that require secure and non secure apps
Using the Hardware Separation features of ARM architecture to isolate secure or robust applications from the rest of the system

- Control only flows from Secure World to Normal World
- Data could flow either way
How this could be accomplished: Example

Non-certified
- Complex Instrument Logic
- Complex Graphics Render

Certified
- Safe Instrument Logic
- Safe Graphics Render
- Safe Graphics Driver

planes blended in hardware, also managed by safe driver

draws content to separate graphics plane managed by safe driver

Single Core
- Cluster (unSafe)
  - Nucleus
  - Hardware Separation
- Cluster (Safe)
  - Nucleus SafetyCert
  - Safe graphics

Dual Core
- Cluster (unSafe)
- Nucleus
- Hardware Separation
- Cluster (Safe)
  - Nucleus SafetyCert
  - Safe graphics

A9
A9
## Mentor Embedded

**Safe • Secure • Multi-OS • Heterogeneous Multicore Platforms**

<table>
<thead>
<tr>
<th>Multi-OS</th>
<th>Android • AUTOSAR • Bare metal • Linux • Nucleus RTOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secure Multicore Framework</td>
<td>Type 1 Hypervisor • AMP • SMP • TrustZone Enabled</td>
</tr>
<tr>
<td>Safety Certs*</td>
<td>ISO 26262 • DO-178 • IEC-61508 • IEC-62304</td>
</tr>
<tr>
<td>Tooling</td>
<td>Sourcery CodeBench • Analyzer • AUTOSAR Virtual Prototyping • Requirements Tracing</td>
</tr>
<tr>
<td>Reference Platforms</td>
<td>SOP Ready • Automotive Design Rules • Flexible FastBoot • Services</td>
</tr>
</tbody>
</table>

*Note: Safety Certifications are an ongoing process.*

---

33

**Mentor Automotive**
Summary

- Analyze your specific requirements to determine which use case outlined in this session makes sense for your device.
- Mentor has technologies and expertise to help you address automotive consolidation use cases.