Real-time Software Enablement in Zonal Controllers

In future zonal E/E architectures, dedicated ECUs will handle the interzonal communication of data as well as its uplink to the central compute. Arm describes the consolidation of real-time software as an important requirement – also for Safety Islands in heterogeneous system-on-chip designs, such as in cockpit assemblies or ADAS.

© top images | adobe.stock.com | ATZelektronik

WRITTEN BY

Bernhard Rill
is director of automotive partnerships in the EMEA region of Arm in Munich.

James Scobie
is director of product management within Arm`s Automotive Line of Business in Cambridge (UK).

The automotive industry is currently experiencing a disruptive period. New ADAS features – and the general move towards autonomous driving – are significantly increasing the number of functions in vehicles. Consumer electronics, such as the cell phone, are influencing consumer needs in the car, which, like the smartphone, is expected to update and improve over its lifetime from the end user's perspective. This is creating the stimuli for the automotive world to move towards the Software-Defined Vehicle (SDV). The rise of SDVs and ever-evolving consumer demands are going hand-in-hand with – and often driving – the changes in E/E architecture in the vehicle.

These architectures are migrating from the traditional distributed approach for ECUs through the current state-of-the-art domain controller architecture to, ultimately, a zonal architecture with high-performance central processing systems. One clear goal of this architectural change is to minimize the absolute number of ECUs. The software-defined functions will be located in the central compute units, and the zonal controller platforms implement the signal-to-service conversion in order to abstract sensor and actuator data.

Arm addresses the automotive sector with a dedicated IP product line. This includes the Cortex-A computing cores for the highest performance requirements, -R for real-time and safety applications, and -M for highly efficient functions, particularly in microcontrollers. They are used in current ECUs but are also capable of enabling both domain and zonal architectures.

From a software enablement perspective, there is a broad trend towards development in the cloud, often referred to as cloud-native, but always aimed at deployment on an embedded system in the vehicle. Software enablement in vehicles is addressed by the SOAFEE initiative in collaboration between automotive and semiconductor manufacturers, open source and independent software and cloud providers [1].

REAL-TIME SOFTWARE INTEGRATION OPTIONS

Automotive systems are frequently built with real-time operating systems using Autosar APIs, both in classic and adaptive forms as their standard software stack. However, there are also other real-time operating systems that are used and deployed in automotive systems (e.g. Green Hills Software, uVelosity, Integrity, Blackberry QNX, ESol, Windriver VXworks, Zephyr, etc.). Furthermore, there are simple bare metal applications that directly run without any abstraction level.

Automotive real-time software integration can be deployed across a diverse range of compute platforms. These range from simple MCUs to highly complex Systems-on-Chip, **FIGURE 1**.

However, automotive compute scalability faces some challenges. Due to the recent chip shortage, it has been difficult for OEMs to maintain supply of various parts needed in vehicles today. In particular, sourcing existing 8-bit/16-bit compute platforms has been difficult, and if available, they have been relatively expensive. To ensure supply stability and the continued availability of the silicon, OEMs are migrating their legacy software to power-efficient 32-bit compute based on Arm Cortex-M processors, for example.

This is part of a wider trend to uplift legacy software to 32-bit compute, as it provides both access to a common compute platform and security of supply through availability from multiple semiconductor vendors that offer Arm-based MCUs. Integration of 8-bit/16-bit applications can be done with either simple proprietary Real-Time Operating Systems (RTOS) or classic Autosar software stacks.

Additionally, OEMs want to design their system in a way that gives them freedom to deploy software functions variably depending on available compute resources and according to their timing behavior. This is summarized under the term portability/consolidation. When it comes to these real-time software integration principles, cohesion and separation have to be distinguished:

- Cohesion refers to applications that are closely related, rely on strong coordination and share a common development base, with example implementations including Classic Autosar featuring the platform flexibility principle – the binary update of software components – and Autosar CP-Flex with Logical Timing Extension (LTE) integration.
- Separation, like Exception Level 2 (EL2) separation, applies to distinct workloads that are developed separately and therefore built on independence as a principle. The overall aim is to reduce ECU count. The software stacks used in the separate integrations may still use common implementations like Classic Autosar, but must remain independent.

EL2 SEPARATION WITH MICROCONTROLLER HYPERVISOR

Cohesion and separation integration options commonly support Freedom

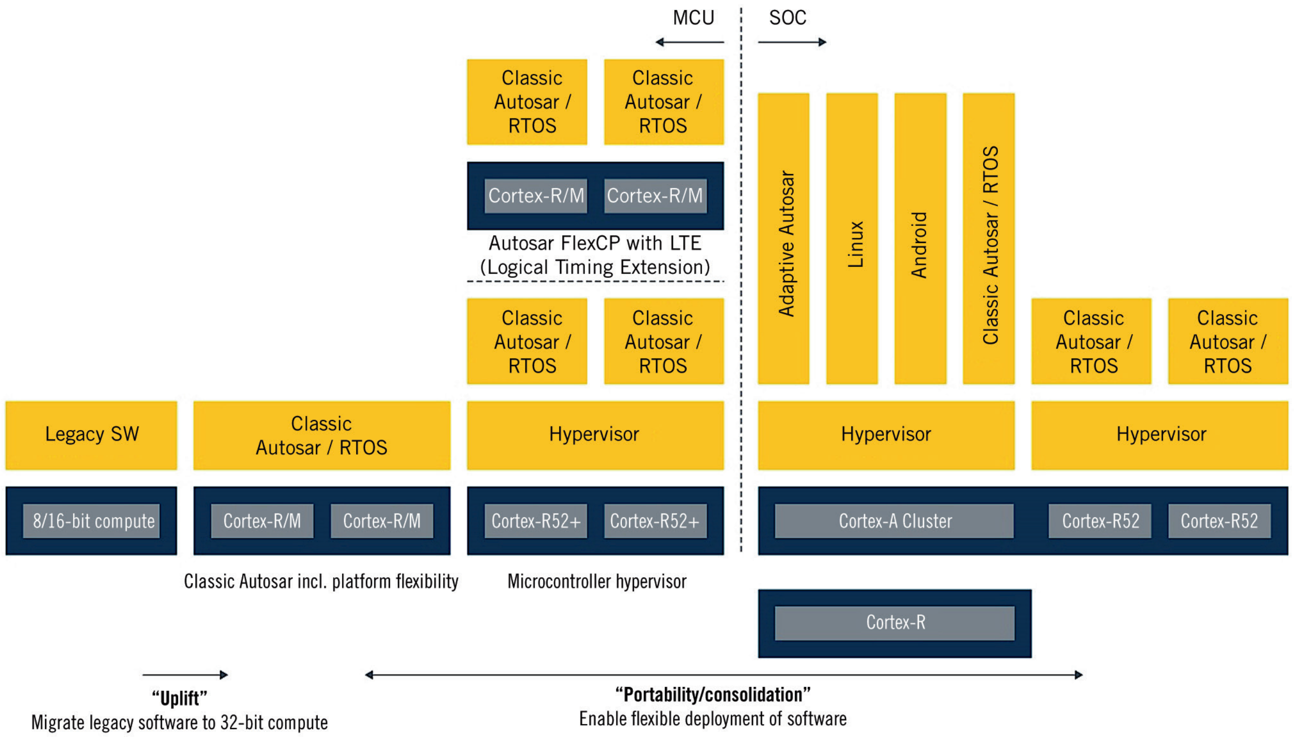

FIGURE 1 Automotive compute scalability (© Arm Ltd)

Feature	Classic Autosar	EL2 separation
Architectural freedom	Integration options are based on classic Autosar options (e.g. FlexCP), normally a software stack is pinned to (a) CPU(s)	EL2 separation allows to host several virtual machines within a cluster or even a CPU (with additional scheduling mechanisms). The toolchain needs to be enhanced though (no standardization as of now)
System supervision/ exception handling	Supervision opportunities minimized as there is no EL2 exception level	EL2 separation allows hierarchical supervision (e.g. independent shutdown/restart of VMs)
Heterogenous SW platforms support	All software artefacts should at least support similar Autosar architecture features. Preferably they are of the same classic Autosar version	Different software stacks (e.g. classic Autosar with different versions and different suppliers, bare metal code, FreeRTOS, etc.) can be integrated
Memory requirements	Every stack maintains the existing memory footprint as part of the Basic Software (BSW) stack architecture	Potential optimization of duplicated BSW software components (e.g. communication stacks)
Software suppliers	Integration is expected to be limited to one classic Autosar vendor (detailed analysis needs to follow)	Flexible model allows sourcing from different software vendors
Toolchain	Classic Autosar has a well-established mature tool chain. Testing is done as a single monolithic system	EL2 separation integration toolchain needs to be established,- further standardization is beneficial. Integration and Testing can be done at VM level granularity

TABLE 1 Cohesion and separation – comparing the characteristics of consolidation options (© Arm Ltd)

From Interference (FFI) from both a spatial and temporal perspective, with individual software updates possible for dedicated building blocks.

TABLE 1 helps understand some of the differences between the cohesion and the separation integration options. It is up to the integrator to decide which consolidation option is most suitable. In the next section, we focus on EL2 separation, which is supported by the Armv8-R architecture.

CORTEX-R IN AUTOMOTIVE SYSTEMS

The introduction of the Armv8-R architecture was the first to enable the hardware separation of tasks into Virtual Machines (VMs), which can represent multiple independent guests. The approach is based on Physical Memory System Architecture (PMSA) using additional exception levels and a two-stage MPU ensuring the real-time characteristics of the processor, **FIGURE 2**.

In a fully virtualized system with Virtual Memory System Architecture (VMSA), the software can be developed without any knowledge of other VMs. However, this approach can result in the loss of determinism. Within the Armv8-R architecture, the separation offered with the PMSA solution can maintain real-time determinism but requires users to be aware of other VMs. Furthermore, access to shared resources should be orchestrated.

Armv8-R CPUs support this real-time virtualization capability, with Cortex-R52+ offering additional visibility to a system of more transactions in order to enhance control of the resources in a virtualized system.

Within state-of-the-art automotive silicon designs, the existing Armv8-R products (Cortex-R52 and Cortex-R52+) are integrated in two different types:
– Cortex-R class, which is the primary processor within designs that can be referred to as MCUs
– Armv8-R CPUs, which are used for the real-time compute and within the so-called Safety Island of heterogenous SoC (System-on-Chip) designs, with another CPU block in the design that is responsible for the high-performance compute tasks.

Based on the versatility of the Armv8-R CPUs, a very flexible deployment of software artefacts is possible. Since the Arm IP is widely used in various SoC designs, there are now many software ecosystem partners that have developed products for Armv8-R CPUs. These include Etas, Elektrobit, OpenSynergy, Green Hills Software, Kernkonzept, Vector, and Sysgo.

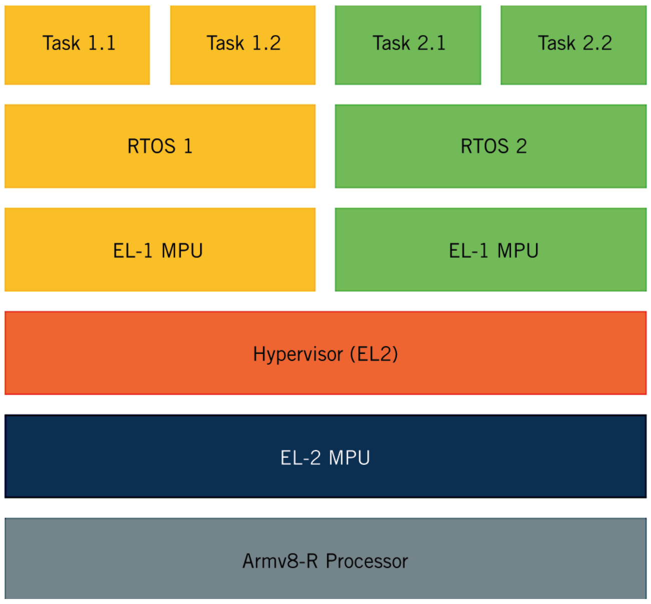

FIGURE 2 Armv8-R Cortex-R exception levels (© Arm Ltd)

Application type	Interrupt virtualization	Virtual processor cores	Para-virtualization/ trap-and-emulate	Example applications
Ultra-low latency hard real-time	No	No	Yes, if the application allows	Powertrain, braking
Low-latency hard real-time	Yes, dynamic scheduling may be needed	Yes, dynamic scheduling may be needed	Yes, if the application allows	Other chassis
Latency focused real-time	Yes	Yes	Yes, if the application allows	General body functions (e.g. immobilizer, ambient lighting, etc.)
Best effort	Yes	Yes	Yes	Convenient body functions (e.g. road noise suppression)

TABLE 2 Integration of real-time software functions based on EL2 separation. Applications with the respective Armv8-R separation technology (© Arm Ltd)

To better understand the EL2 separation options, Arm has released a white paper titled 'Best practices for Armv8-R Cortex-R52 software consolidation' with ETAS. The white paper is available for download at [2].

The paper conveys an overview of how the integration of real-time software functions based on EL2 separation can be achieved. **FIGURE 4** shows relevant applications and the Armv8-R separation technique used in each case.

Arm has published another white paper that focuses on device assignment, including how on-chip modules - such as communications peripherals, memory or hardware security modules - can be assigned to individual independent guests and virtual machines [3].

USE CASES 2025 AND 2028 OUTLOOK

After considering all the integration options for software consolidation, the question is which use cases and scenarios OEMs want to consolidate. Starting from 2025, many leading OEMs will move to a zone-based EE architecture. This move will be driven by the fact

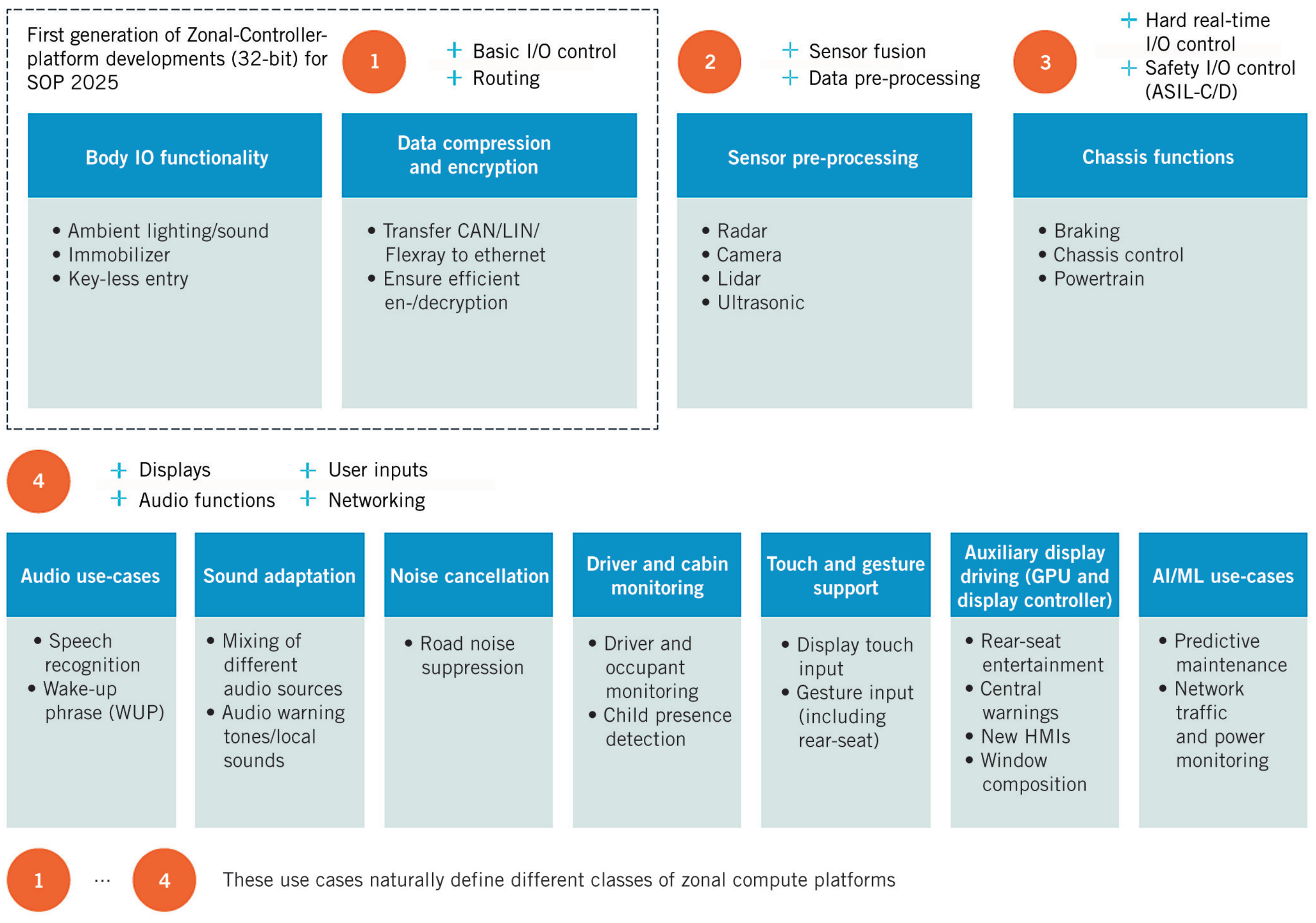
First generation of Zonal-Controller-platform developments (32-bit) for SOP 2025
1
+ Basic I/O control
+ Routing
2
+ Sensor fusion
+ Data pre-processing
3
+ Hard real-time I/O control
+ Safety I/O control (ASIL-C/D)
Body IO functionality
• Ambient lighting/sound • Immobilizer • Key-less entry
Data compression and encryption
• Transfer CAN/LIN/ Flexray to ethernet • Ensure efficient en-/decryption
Sensor pre-processing
• Radar • Camera • Lidar • Ultrasonic
Chassis functions
• Braking • Chassis control • Powertrain
4
+ Displays
+ Audio functions
+ User inputs
+ Networking
Audio use-cases
• Speech recognition • Wake-up phrase (WUP)
Sound adaptation
• Mixing of different audio sources • Audio warning tones/local sounds
Noise cancellation
• Road noise suppression
Driver and cabin monitoring
• Driver and occupant monitoring • Child presence detection
Touch and gesture support
• Display touch input • Gesture input (including rear-seat)
Auxiliary display driving (GPU and display controller)
• Rear-seat entertainment • Central warnings • New HMIs • Window composition
AI/ML use-cases
• Predictive maintenance • Network traffic and power monitoring
1 ··· 4
These use cases naturally define different classes of zonal compute platforms

FIGURE 3 Potential zonal-controller use cases for SOP 2028 (© Arm Ltd)

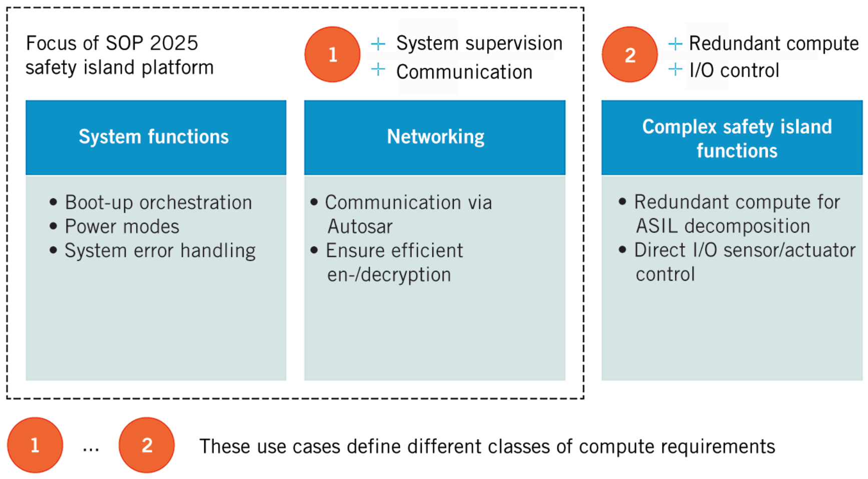
Focus of SOP 2025 safety island platform
1
+ System supervision
+ Communication
2
+ Redundant compute
+ I/O control
System functions
• Boot-up orchestration
• Power modes
• System error handling
Networking
• Communication via Autosar
• Ensure efficient en-/decryption
Complex safety island functions
• Redundant compute for ASIL decomposition
• Direct I/O sensor/actuator control
1 ... 2
These use cases define different classes of compute requirements

FIGURE 4 Potential Safety Island SOP 2028 use cases (© Arm Ltd)

that the wiring harness is too complex and needs to be split into multiple spatial areas. Once this is done, the Zonal-Controller platforms are important to ensure highly efficient networking, such as converting CAN, LIN or Flexray to Ethernet, and to direct intelligent power – via smart fuses – to the appropriate area of the car. Moreover, the platforms are used to drive some simple I/O-related body funtions based on 32-bit classic Autosar applications.

Zonal-Controller platforms are a perfect transition point from signal-based to service-based communications. Currently, the industry is investigating how to use Zonal-Controller platforms for a Start of Production (SOP) 2028. Next to the network and simple IO abstraction applications mentioned above, investigations are focused on three additional areas, **FIGURE 3**:

- Pre-processing of sensor data
- Integration of high-integrity chassis functionality
- Cabin-/interior-driven use cases.

Aside from the SOP-2028 timeframe, the authors see further functionality requirements that need to be hosted in the Cortex-R-based Safety Island of heterogenous SoCs, **FIGURE 4**.

SUMMARY

This outlook on the future of E/E architecture development emphasizes the need for further real-zonal software integration; a focus on zone controllers and the Safety Islands of heterogeneous SoCs. The Armv8-R CPU family is prominently used in this compute class. Investment in this product line will help drive software reuse across current and future platforms over the long term. As the automotive industry moves towards SDVs and ever more complex compute requirements, Arm see their commitment to the development of this product line as being particularly important for future automotive platforms.

REFERENCES

[1] Scalable Open Architecture for Embedded Edge (SOAFEE) project; https://www.soafee.io/; access: April 12, 2023

[2] Paul Austin, Andrew Coombes,Paul Hughes, James Scobie, Bernhard Rill: Best practices for Armv8-R Cortex-R52+ software consolidation; Arm/Etas White Paper, https://armkeil.blob.core.windows.net/developer/Files/pdf/white-paper/best-practices-for-armv8-r-cortex-r52-st2-whitepaper.pdf; access: April 12, 2023

[3] Alexandre Romana: Introducing device virtualization principles for real-time systems; Arm White Paper, https://community.arm.com/arm-community-blogs/b/automotive-blog/posts/device-virtualization-principles-for-real-time-systems; access: April 13, 2023