Software-Defined Vehicles Shift the Future of Automotive Electronics Into Gear

Texas Instruments

Donovan Porter
Systems Manager
Body Electronics and Lighting

Yannik Muendler
Systems Engineer
Advanced Driver Assistance Systems

In this paper we discuss how software-defined vehicles with zone architectures are enabling development of smarter, safer, and more energy-efficient vehicles. By centralizing software and decoupling hardware from software, they allow for easier updates, reduced costs, and new features.

At a glance

1 **Domain-based and software-defined vehicles**
Explore the differences between domain-based and software-defined vehicle architectures.

2 **New technologies enabled by software-defined vehicles**
Learn how software-defined vehicles are enhancing technologies like digital twins, optimizing vehicle performance.

3 **Variations in software-defined vehicle and zone architecture approaches**
Understand the different approaches to centralizing software in vehicles based on specific design requirements.

Introduction

Automotive original equipment manufacturers (OEMs) are continuously working to improve the occupant experience, simplify over-the-air updates, reduce design and manufacturing costs, collect more vehicle data, and create new revenue streams. However, today's domain-based vehicle architectures are not equipped to easily and effectively meet these needs, leading to a shift toward software-defined vehicles and zone architectures. By centralizing software and decoupling hardware from software, software-defined vehicles are the next step in the development of smarter, safer and more energy-efficient vehicles.

Domain-based and software-defined vehicles

Today's domain-based architectures are inefficient at providing scalable software that automakers can maintain easily through over-the-air updates. Domain architectures segment the control of vehicle functions into domains such as in-vehicle infotainment and advanced driver assistance systems (ADAS), as shown in Figure 1.

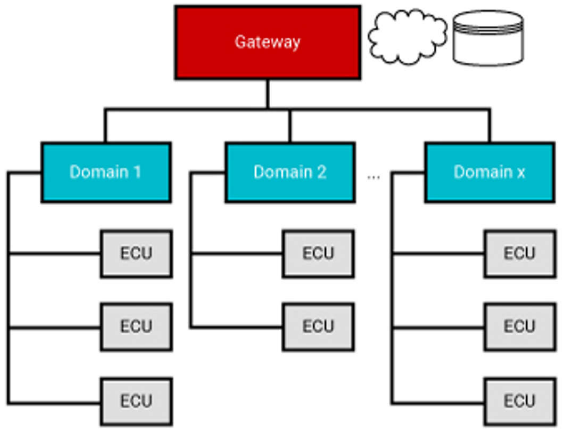


***Figure 1.** Diagram of domain-based architecture in vehicles.*

Dividing control of vehicle functions complicates software development for features that may require communication and control across multiple domains. Updating software for these systems is challenging, as they are designed and manufactured by different Tier-1 suppliers that all use various processors and microcontrollers from different semiconductor suppliers.

The software for controlling vehicle functions is also closely coupled to the hardware. OEMs will install electronic control units (ECUs) to perform specific functions (seat adjustments, parking assistance), with application-specific firmware running on each ECU microcontroller. These ECUs will also vary across vehicle models and trims, leading to higher manufacturing and design costs. Thus, managing software across all vehicle models, trims and individual ECUs is a significant effort, requiring that OEMs work with multiple Tier-1 and potentially even semiconductor suppliers to implement new software updates.

In contrast, software-defined vehicles employing a zone architecture simplify over-the-air updates by centralizing software, enabling the flexibility to add new features through software by decoupling the vehicle hardware from higher-layer application software and offering more cost-effective scalability across vehicle models and trims.

Figure 2 shows an example of a zone architecture that centralizes software in the central computing system and implements zone control modules to aggregate data, actuate loads, and distribute power locally. For more information about zone architectures, see "**How a Zone Architecture Paves the Way to a Fully Software -Defined Vehicle**."

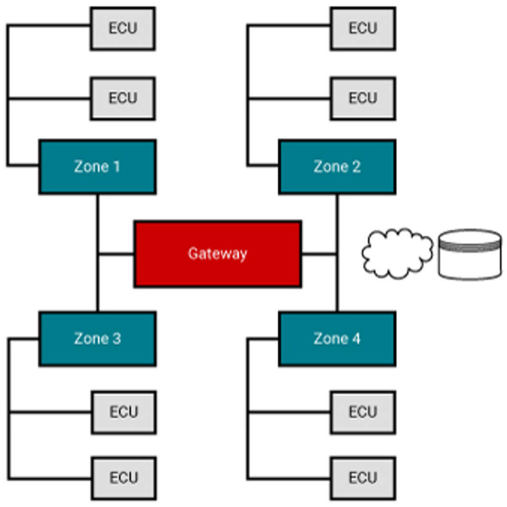


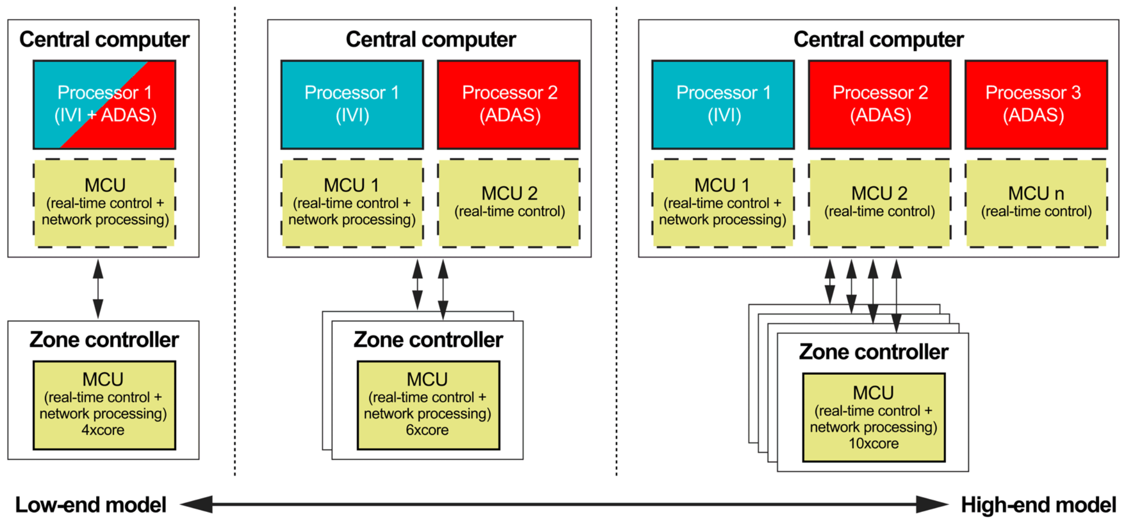
Figure 2. *Diagram of zone architecture in vehicles.*

The main advantage of centralized software in software-defined vehicles is the reduction in ECUs that host application software, simplifying over-the-air updates by reducing the number of processors and microcontrollers requiring firmware changes. Adding new features and applications requires updating only the central computer or zone control module software, as the downstream sensors and remaining ECUs controlling the mechanical actuation (headlights, door modules, audio amplifiers) are abstracted from the application software. Therefore, the ECUs performing mechanical actuation and sensors at the edge of the vehicle network require less complex firmware and may completely shift real-time control to the central computer in the future.

Furthermore, it is possible to repurpose sensors and actuators originally designed for specific applications to create new features. An example is adding new applications for the in-cabin radar sensor, initially designed for occupancy monitoring, to provide intruder or theft detection and seat belt reminder functions. Essentially, OEMs have more flexibility to implement new features with hardware and sensors that already exist within the vehicle.

Lastly, software can scale across all vehicle platforms, as shown in **Figure 3**, further reducing development costs. An economy-level vehicle can implement the same software as a luxury brand for functions such as remote keyless entry, window lifts and rearview cameras.

Luxury models can offer premium features through software on top of baseline features. Although hardware changes may still be required, the overall approach is modular and scalable across vehicles. Adding or removing processors and microcontrollers can scale computing power up or down in the central computer or zone control modules.



Central computer
Processor 1 (IVI + ADAS)
MCU (real-time control + network processing)
Zone controller
MCU (real-time control + network processing) 4xcore
Central computer
Processor 1 (IVI)
Processor 2 (ADAS)
MCU 1 (real-time control + network processing)
MCU 2 (real-time control)
Zone controller
MCU (real-time control + network processing) 6xcore
Central computer
Processor 1 (IVI)
Processor 2 (ADAS)
Processor 3 (ADAS)
MCU 1 (real-time control + network processing)
MCU 2 (real-time control)
MCU n (real-time control)
Zone controller
MCU (real-time control + network processing) 10xcore
Low-end model
High-end model

Figure 3. *Comparison of computing power across low- and high-end vehicle models.*

Implementing features like seat massage, steering wheel heating, and road noise cancellation in vehicles requires additional hardware. However, only software updates for the central computer or zone control modules are required to control these additional features. Emerging MCU-less technology will help simplify or enable designers to remove software in the ECU that typically manages sensing and/or mechanical actuation. For example, a temperature sensor that uses serial peripheral interface (SPI) could directly communicate with a SPI-enabled MCU-less communication PHY. The MCU-less PHY in this scenario would replace the MCU and feature an integrated CAN or Ethernet transceiver, removing the need for an MCU to translate SPI to CAN signals and the software typically needed to communicate with the sensor.

Hardware abstraction layers to enable software defined vehicles

Different abstraction layers are required to enable decoupling of hardware from the software in vehicles. Standardized application programming interfaces (API) enable communication between the different abstraction layers and allow applications source code to be reused in multiple distributed ECUs. The lowest abstraction level is the Microcontroller Abstraction Layer (MCAL).

The MCAL, which plays an important role in SDV, provides APIs that abstract the complexities of the underlying hardware peripherals. It acts as a bridge between the hardware integrated in the central compute SoC like a **TDA4VH-Q1** processor, including timers, ADCs, Ethernet subsystem, and the higher-level software layers. The MCAL ensures that application software can interact with the hardware without being tied to specific hardware details. This abstraction is crucial for achieving software portability across different vehicle platforms, enabling OEMs to reuse software components across multiple models and variants with minimal modifications.

To interface between the higher-level software and the MCAL, there is the ECU Abstraction Layer (ECUAL). The ECUAL provides all the available ECU hardware, including the MCU and peripheral devices (for example, CAN transceiver, Ethernet PHY and SerDes devices) access to higher-level software via standardized APIs.

New technologies enabled by software-defined vehicles

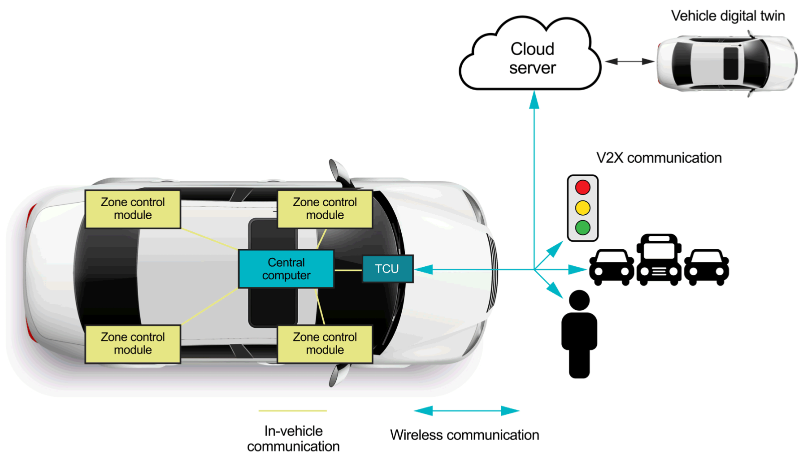


Figure 4. *Software-defined vehicle connectivity to the cloud and V2X.*

Software-defined vehicles enable new technologies and revenue streams for OEMs. As vehicles continue to integrate more electronics and sensors, vehicle performance, fault scenarios and driver preference data is more available than ever before. Software-defined vehicles help further improve digital twin and vehicle-to-everything (V2X) capabilities by simplifying the collection and ability to securely share vehicle data, as shown in Figure 4.

With digital twin capabilities (a virtual representation of a real-world system), software-defined vehicles can share data to the cloud to document real-world performance such as electric vehicle battery health over time, ADAS sensor information throughout various driving conditions, and even vehicle feature usage. This data can help OEMs optimize vehicle capabilities and reduce the time required to solve new challenges, especially for technologies such as ADAS and autonomous driving. Additionally, OEMs would be able to identify common issues across specific vehicle models and provide fixes before major problems arise.

In addition to digital twin technology, vehicle data is valuable for V2X communication because it enables vehicles to share information between vehicles, people and infrastructures to improve safety and traffic flow. Sharing information such as lane departure and vehicle speed securely from the central computer to other vehicles can help improve collision avoidance capabilities.

Lastly, OEMs are continuing to find ways to create new revenue streams. Software-defined vehicles give OEMs complete control of the software within their vehicles and therefore, the capabilities to differentiate their user experience. OEMs could offer subscription models for specific features that can be enabled through software. Features could be simple such as heated seats or they could be more complex like advanced driving safety capabilities. Although subscriptions may not sound attractive to consumers, new features could be added through software updates to existing vehicles instead of requiring consumers to purchase the latest model year.

Over-the-air software update process

Over-the-air (OTA) or firmware-over-the-air (FOTA) software updates must be developed, tested, and uploaded to a secure cloud-based server accessible by the vehicle to be executed. The vehicle, in turn, must be able to download and store the update, whether in the central compute system, zone controllers, or edge ECUs. Given that an ECU restart is typically required for the update to take effect, the update process must occur while the vehicle is in a safe state.

With an update being available, one option is to notify the driver of an available update and let the driver confirm the start of the update once the vehicle has been safely parked. Alternatively, by tracking vehicle usage times, the system can guess the best time to perform a software update without user intervention. The vehicle may be temporarily inoperable during this time, so the update must be completed efficiently to minimize downtime. The ECU must remain powered throughout the update, and consideration must be given to the vehicle's battery capacity. To mitigate risks, the ECU may be designed to store both the current software version and the new update in memory, allowing it to switch to the updated version during the next startup. If the update fails, the system can revert to the previous software version, ensuring continued vehicle functionality.

When vehicle functions are distributed across multiple ECUs, updates must be coordinated through carefully planned update campaigns. These campaigns involve deploying update packages to all affected ECUs to ensure system-wide compatibility and performance.

Variations in software-defined vehicle and zone architecture approaches

Each automotive manufacturer has a unique approach to achieving the software-defined vehicle. The legacy of previous-generation vehicle platforms will force many OEMs to shift gradually to an electrical and electronic zone architecture that better suites their centralized software approach.

While the majority of OEMs are developing zone architectures, there are different approaches when deciding where the software to control vehicle functions resides, as shown in **Figure 5**.

There are three options when centralizing software control: central computer; shared between the central computer and zone control modules; or distribution into a few domain controllers and zone control modules. Some OEMs are centralizing high-performance computing domains such as ADAS and in-vehicle infotainment and adding additional application processing for other domains. Outside of the ADAS and in-vehicle infotainment domains, real-time control is implemented either in the zone control modules or edge ECUs.

The centralized computing approach may be the most attractive from an OEM perspective, as a single computer controls all vehicle functions. There may be additional challenges with real-time control loop latencies (active suspension, window anti-pinch) and functional safety if the communication link fails.

The distributed computing approach takes a more gradual step toward centralizing software, maintaining some application and real-time control software in the zone control modules or even in separate domain controllers. In all architectures, zone control module requirements will vary even within the same vehicle, depending on the OEM. One zone could handle some real-time control of body; heating, ventilation and air conditioning; and chassis functions, while another handles additional body, lighting and the vehicle control unit application software. Ultimately, OEMs will have to balance hardware and mechanical actuation control latency, in-vehicle network capabilities, functional safety, security, and how to structure software for the architecture that they choose and its specific zone control module requirements.

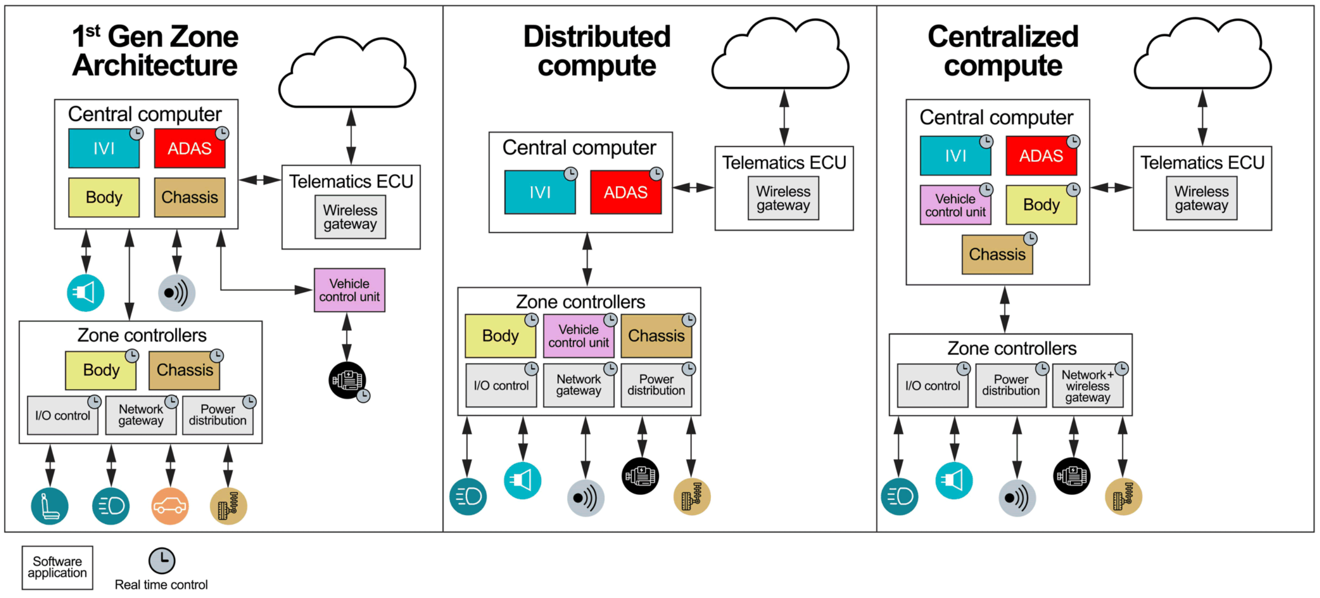


Figure 5. *Comparison of vehicle architecture types.*

Conclusion

Software-defined vehicles are creating new opportunities for automotive manufacturers to decrease the time and cost involved in developing new vehicles and functions, continue improving the driver experience throughout the vehicle's lifetime, and creating new revenue streams. Although there are several approaches, the centralization of vehicle software and the abstraction of vehicle hardware from the software will be the greatest priority. Overall, OEMs will accelerate the development of smarter, safer and more energy-efficient vehicles through zone architectures and the software-defined vehicle.