frontiers | Frontiers in Future Transportation

TYPE Review
PUBLISHED 17 February 2025
DOI 10.3389/ffutr.2025.1519390

OPEN ACCESS

EDITED BY
Dongyao Jia,
Xi'an Jiaotong-Liverpool University, China

REVIEWED BY
Ilaria Matteucci,
National Research Council (CNR), Italy
A-Long Jin,
Xi'an Jiaotong-Liverpool University, China

*CORRESPONDENCE
Konstantinos Koufos,
konstantinos.koufos@warwick.ac.uk

RECEIVED 29 October 2024
ACCEPTED 22 January 2025
PUBLISHED 17 February 2025

CITATION
Guo Z, Koufos K, Dianati M and Woodman R (2025) State-of-the-art virtualisation technologies for the centralised automotive E/E architecture.
Front. Future Transp. 6:1519390.
doi: 10.3389/ffutr.2025.1519390

State-of-the-art virtualisation technologies for the centralised automotive E/E architecture

Zixuan Guo[1], Konstantinos Koufos[1]*, Mehrdad Dianati[1,2] and Roger Woodman[1]

[1]WMG, University of Warwick, Coventry, United Kingdom, [2]School of Electronics, Electrical Engineering and Computer Science (EEECS), Queen's University of Belfast, Belfast, United Kingdom

The automotive industry is undergoing profound changes, driven by the need for safer, more environmentally friendly, and more accessible future mobility and transport systems for goods and people. Enabling technologies include electrification, digitalisation, and automation of future vehicles. These technologies are powered by a multitude of onboard Electronic Control Units (ECUs). A typical modern vehicle has about 100 physical ECUs to enable various aspects of its function. These legacy many-ECU electronic/electrical (E/E) architecture models, known as distributed E/E architecture, are deemed inefficient as the number of ECUs and their processing power requirements keep increasing. In contrast, emerging centralised E/E architectures propose using fewer physical high-performance onboard processors on which an almost unlimited number of virtual ECUs can be created to handle various legacy and modern applications. As a result, virtualisation techniques, which enable multiple virtual ECUs with different operating systems to run concurrently on a single hardware platform, are promising models for modern centralised E/E architectures. Motivated by this trend, this paper provides a structured and comprehensive state-of-the-art review of virtualisation techniques for automotive applications, covering areas such as resource allocation, AUTOSAR, peripheral I/O interfaces, and in-vehicle communication networks. We comprehensively review the literature and identify research gaps in virtualisation techniques for cache management, paravirtualsation, software-defined networking for in-vehicle networks, and virtualisation for enhanced prototyping and testing in the context of modern E/E architectures for modern vehicles.

KEYWORDS

automotive E/E architectures, AUTOSAR, software-defined vehicle, virtualisation, paravirtualisation

1 Introduction

With the growing demand for innovative automotive functions like Advanced Driving Assistance Systems (ADAS) and the progression to higher levels of autonomy (Rödel et al., 2014), the number of required ECUs continues to rise due to the one-to-one mapping of vehicle functions and ECUs. Furthermore, as vehicles become more sophisticated, peripheral devices such as perception sensors and communication transceivers are integrated to support these functions. This increases the number of ECUs required and adds to the size and complexity of the communication network, among many other

problems (Kampert et al., 2020). To address these challenges, a centralised E/E architecture has been proposed to reduce the complexity of future vehicles.

Compared to the distributed architecture, the centralised approach integrates multiple functions onto a single multi-core ECU, thereby reducing the overall number of ECUs, simplifying system design and lowering power consumption. However, due to the current performance limitations of ECUs, an immediate shift to centralised architecture is not yet feasible, leading to the adoption of domain-based architecture as a viable alternative. In this approach, vehicle functions are first grouped according to their specific functional domain, such as engine or body control (Navale et al., 2015). Several ECUs (called "Domain ECUs") are then assigned to handle tasks under a specific domain.

High-performance ECUs, specialised GPU processors, and high-bandwidth in-vehicle communication networks are expected to drive the industry's transition towards centralised (or zone-based) architectures. These advancements will enable vehicles to tackle challenges associated with automated driving, such as managing large volumes of data and meeting high computational demands (Di Natale and Sangiovanni-Vincentelli, 2010; Bandur et al., 2021). A centralised E/E architecture facilitates more sophisticated vehicle design (Maul et al., 2018), reduces the wiring harness (ASA, 2023), and scales down by 20% the total number of ECUs, as reported by BOSCH (Solutions B. M., 2023a). Several OEMs have already built prototypes, including Tesla's Autopilot and Nvidia DRIVE Thor. However, the automotive industry remains in the early stages of adopting the centralised E/E architecture.

In a centralised architecture, multiple vehicle functions share the same ECU and may need to access concurrently shared computing resources, the memory, the cache, or the communication network. Consequently, an efficient embedded system becomes crucial for adequately managing, scheduling, and isolating these functions. Virtualisation techniques, commonly used in computing to allow multiple operating systems to run efficiently in parallel on the same hardware (Wulf et al., 2021), offer a promising solution in this context. By implementing virtualisation, resources such as ECUs, memory, communication networks, and peripheral I/O interfaces can be abstracted as Virtual Machines (VMs) and efficiently managed by a Virtual Machine Monitor (VMM), also known as a hypervisor (Smith and Nair, 2005). However, adopting virtualisation techniques in the centralised architecture introduces several technical challenges. These challenges are further complicated because Automated Vehicles (AVs) are safety-critical systems and must adhere to safety standards such as ISO 26262.

An automotive architecture utilising VMs allows multiple tasks with varying dependencies to run simultaneously on the same ECU. In a centralised architecture, computing resources and peripheral devices become shared elements, allowing various tasks to request access to these elements simultaneously. This can lead to message collisions over the communication network when multiple VMs try to communicate with peripheral components concurrently. To optimise performance and mitigate safety concerns in these environments, specific virtualisation technologies like I/O virtualisation, and virtualisation partition approaches have been proposed. The potential of virtualisation techniques in the context of modern automotive E/E architectures has motivated a growing number of studies on the topic in recent years. However, to our knowledge, a comprehensive literature review is lacking to help researchers gain deep insight into the state-of-the-art. To this end, the contributions of this review article are multifaceted. First, by examining the evolution of the E/E architecture from distributed to centralised systems, we highlight the benefits virtualisation can offer in the automotive field and outline the requirements for incorporating virtualisation technologies into vehicle software architectures. Next, we review the state-of-the-art (SOTA) virtualisation applications in key areas such as resource allocation, peripheral I/O interfaces, communication networks, and AUTOSAR. These areas are crucial for effective task processing, connectivity, data transfer, and operation of embedded systems. In particular, we discuss how AUTOSAR, the primary framework the automotive industry uses for managing hardware resources, can be deployed alongside other operating systems on the same ECU (Becker et al., 2015a). Lastly, building on the SOTA virtualisation techniques from fields such as computer science and considering the evolving features of AVs, this review provides a forward-looking perspective on the potential use of virtualisation technologies in the automotive sector. Specifically, we discuss how virtualisation techniques can be applied to cache management, software-defined networking, paravirtualisation, and prototyping during the early development stages of AVs. Paravirtualisation, an emerging technology that reduces the overhead associated with virtualised architectures, will be examined in detail.

To ensure this survey article is comprehensive, we began by reviewing surveys on both vehicle E/E architectures and virtualisation technologies used in other fields of research such as CPU virtualisation in data centres or servers. These reviews helped us identify well-known keywords, enabling us to uncover as many relevant papers as possible on the use of virtualisation in the automotive domain. Additionally, we examined the references list of the identified papers to ensure the available literature is adequately covered.

The remainder of this article is organised as follows. Section 2 introduces the E/E automotive architecture and explores the challenges in implementing a centralised solution. Section 3 presents the concept of virtualisation and reviews the SOTA in the automotive field. Section 4 examines virtualisation technologies well-established in other scientific fields and outlines their potential application in the automotive sector. Finally, Section 5 concludes this survey.

2 Background of the E/E architecture

Major components like ECUs, the Bus communication network, memory, and sensors compose the vehicle's hardware system, supporting rapidly growing innovative features like ADAS (Brunner et al., 2017). In the traditional distributed architecture, each automotive function, like adapting cruise control and autonomous emergency braking, has its own ECU. Bus communication networks like CAN, FlexRay, and Ethernet are responsible for transferring data between the ECUs and actuators (Askaripoor et al., 2022). The Bus communication technology connects several ECUs through the Central Gateway cable. A central processor, the Central Gateway Controller, is also

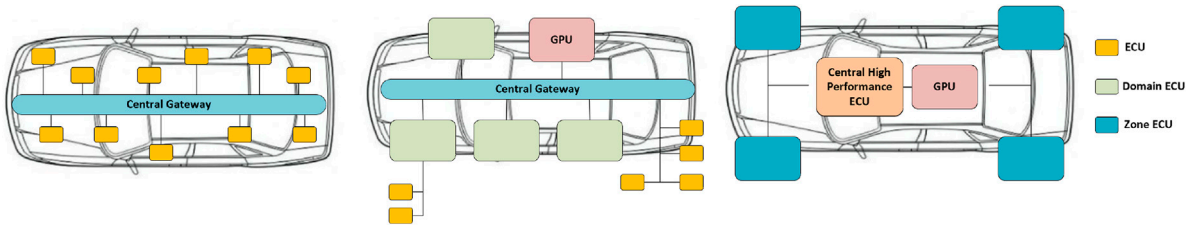

FIGURE 1
Serial evolution of the automotive E/E Architecture (from left to right). Distributed E/E architecture (left), Domain-based E/E architecture (middle), and Zone-based or Centralised E/E architecture (right).

responsible for message and data scheduling. The central gateway plays a fundamental role in the distributed system, managing data exchange between internal interfaces like ECUs and perception sensors through the Data Bus.

In the distributed architecture, electrical components can be easily maintained. However, the growing demand for more intelligent and automated functions has significantly increased the number of ECUs. To ensure design redundancy, in compliance with the vehicle safety standard ISO 26262 (Salay et al., 2017), modern vehicles typically use over 100 ECUs (Bandur et al., 2021). That raises serious concerns about the complexity of the wiring harness, the increased vehicle weight, higher power consumption, and the large volume of data flowing through the CAN Bus, potentially increasing communication delays and blocking information exchange between ECUs and actuators.

To address the drawbacks associated with distributed architectures, the E/E architecture should be inevitably modified. Key technical enablers of these modifications include recent breakthroughs in the performance of contemporary processors and the growing maturity of software technologies. Modern ECUs are becoming capable of processing and controlling several functions simultaneously. Additionally, innovative communication technologies like Ethernet, FlexRay, and CAN FD offer higher bandwidth, lower latency, and more flexibility and security. These developments have led researchers to gradually propose centralised architectures in the automotive industry, such as domain-based and zone-based architectures (Ondrej Burkacky and Apostu, 2019). Figure 1 presents an exemplary illustration of the ongoing trend towards centralising the E/E automotive architecture.

In the domain-based architecture, the central gateway and its controller are maintained, while the vehicle system is divided into four primary domains: Body and Cabin, Infotainment, Vehicle Motion and Safety, and Powertrain (Stolz et al., 2010). Each domain contains one or two high-performance multicore controllers called Domain Control Units (DCUs), as well as several sub-domain ECUs responsible for handling light tasks within each domain (Bandur et al., 2021). Furthermore, GPUs are becoming increasingly crucial for executive computationally intensive tasks such as deep learning algorithms for perception and prediction. Domain-based architecture is gradually replacing distributed architecture in the mainstream industry. Major OEMs like Audi (Times, 2016) and BMW (Tuohy et al., 2014), along with suppliers like Nvidia, TI and NXP (Research and Markets, 2022), are developing and commercialising their solutions based on the domain-based architecture.

In a zone-based architecture, the vehicle is divided into four zones determined by the physical layout of the chassis rather than by the functional domain. This architectural shift allows for the decoupling of hardware from software, meaning that ECUs are no longer dedicated to specific functions or domains. Instead, functions are treated as services, facilitating the implementation of a Service-Oriented Architecture (SOA), where functions can be dynamically called as needed (Vetter et al., 2020). An ECU known as the Zone Control Unit (ZCU) is responsible for managing the data flow and processing the functions within its respective zone. Additionally, a GPU combined with a high-performance central ECU creates a powerful central platform to meet the requirements of ADAS. The automotive industry has recently been keenly interested in the shift towards zone-based solutions. JLR has announced a partnership with Nvidia (ROVER, 2022) to utilize DRIVE Orin's centralised AV processor to build its zone-based architecture. Similarly, BMW has started creating an E/E architecture based on a central computing platform for its next-generation of vehicles (Traub et al., 2017). A zone-based architecture offers several significant advantages, including centralised control, reduced wiring complexity, fewer ECUs, and enhanced compatibility with functions of different dependencies. However, there are still two main limitations to overcome before its widespread commercialisation, which are detailed below:

- Resource Allocation: In distributed E/E architectures, computational resources such as ECU and memory are allocated to specific functions. With the shift towards centralised architectures, more functions can be executed onto a single high-performance multicore ECU. Consequently, methods for managing and allocating computational resources and scheduling tasks on the same ECU become critically important.
- Consolidation: In a centralised architecture, multiple functions are processed simultaneously on the same ECU, a design that enhances system efficiency. However, issues with one function may negatively impact other functions running on the same ECU, leading to system failures. To mitigate this risk, segregating functions running on the same ECU becomes paramount in improving the vehicle's safety.

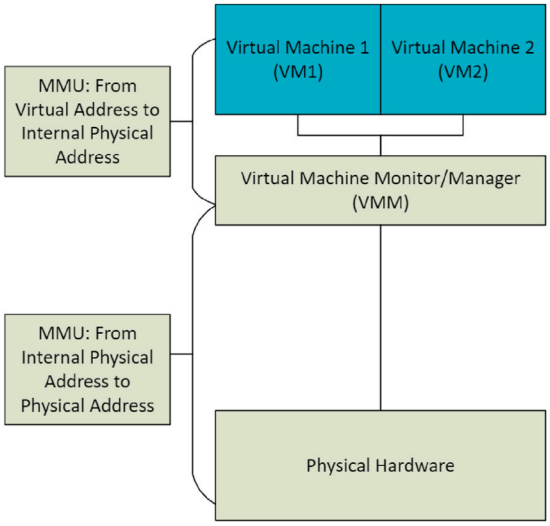

FIGURE 2
Overview of virtualisation architecture.

As the following section explains, virtualisation appears to be a promising technological solution for effectively addressing the above limitations.

3 Virtualisation techniques for automotive

Several innovative technologies like the SOA (Kugele et al., 2017) and the AUTOSAR Adaptive (AUTOSAR, 2024a) have been recently introduced by the automotive industry to improve further the system flexibility and the ability to accept new features (Navale et al., 2015). These functions and features may use different operating systems, have various hardware dependencies, or be supplied by different vendors. Therefore, it becomes important to design an environment compatible with multiple operating systems running on a shared hardware platform while being flexible enough to incorporate new functionalities and features. Virtualisation technology has been proposed as a promising solution to develop such an environment (Kabir et al., 2021).

Virtualisation technology was developed as an abstraction layer to allow functions with various system dependencies to simultaneously access computing resources like processors, peripheral I/O, and memory. Traditionally, the virtualisation layer contains three major components: Virtual Machines (VMs), the Virtual Machine Manager/Monitor (VMM), also known as the hypervisor, and the hardware resources they manage, as illustrated in Figure 2 (Kleidermacher, 2013). Within this architecture, a VM operates as a sandbox, providing an isolated operating environment independent of other VMs. A VM can process tasks or emulate a hardware's behaviour, such as handling sensor data or managing communication networks, depending on the hardware to which virtualisation is applied. The VMM oversees allocating and separating resources, such as memory, to ensure that each VM receives sufficient resources. Finally, a Memory Management Unit (MMU) is required to support the translation between the virtual memory and the physical memory so that a VM can access the hardware through the VMM without directly interacting with the hardware itself, enhancing both isolation and security (Sriramakrishnan et al., 2022).

While the concept of virtualisation remains consistent across various application scenarios, such as personal computing, servers and automotive, the application of virtualisation in automotive must be optimised to meet stringent safety and real-time performance requirements. The rest of this section discusses the application of virtualisation in four key areas of vehicle architecture design: Resource allocation, AUTOSAR, communication networks and input/output (I/O) interfaces.

3.1 Virtualisation for resource allocation

Computational resources in vehicles include hardware components such as memory, cache, processor cores, and the ECU, all essential for processing various vehicle functions. With the rapid advancement and increasing complexity of vehicular functions, managing these computational resources has become a significant challenge. This section reviews the SOTA research on how virtualisation can be utilised for effective resource allocation within multicore ECUs, including support for real-time automotive services and the AUTOSAR standard.

3.1.1 Background

At a time when ECUs are predominantly single-core, a requirement to accommodate an increasing number of vehicle functions is to increase the number of ECUs. While this approach addresses the immediate need for additional computing capacity, it also adds complexity to the E/E architecture and makes it challenging to coordinate the allocation of computing resources among different functions. Failure to do so may result in inefficient utilisation of computing resources or even lead to system failures. To tackle this issue, the authors in (Macher et al., 2015) review the operating systems used in the automotive domain and discuss the migration from single-core to multicore ECU processors as a way to reduce the number of ECUs by allowing different functions to run over multicore ECU. They also highlight the potential of using cloud computing as a viable solution to offload part of the computations and reduce the computational load within the vehicle.

While multicore ECUs offer the advantage of integrating more functions into a single unit and executing more than one function simultaneously, they also introduce new challenges. Due to the restrictions posed by safety standards, the coordination policy of computing resources must ensure that the vehicle functions can always access sufficient resources to meet their latency and safety requirements. Researchers have investigated the optimisation of task scheduling under multicore ECUs to reduce the hardware cost, improve the runtime performance and decrease the latency between different tasks (Kampmann et al., 2022; Vasu and Ramaprasad, 2020; Maticu et al., 2016; Monot et al., 2012). However, their scheduling algorithms run within a single operating environment, which poses a limitation since vehicle functions can be provided by different suppliers or belong to different domains, such as driving or entertainment, and potentially use different operational

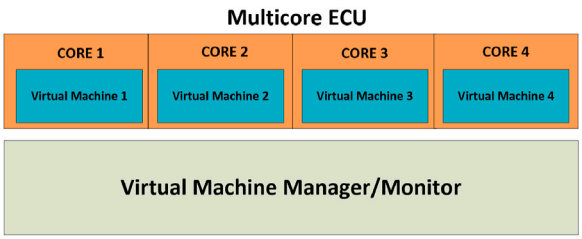

FIGURE 3
An illustration of virtualisation within a multicore ECU.

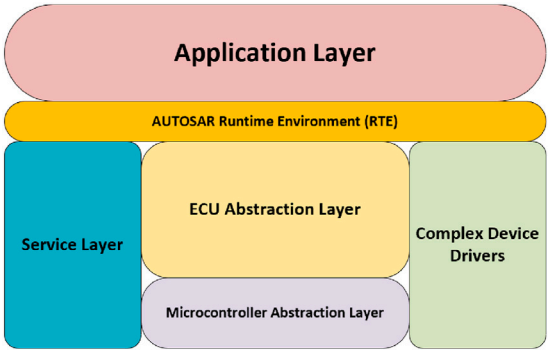

FIGURE 4
An illustration of the high-level architecture of AUTOSAR.

environments. Virtualisation can help address this challenge as explained next.

3.1.2 Virtualisation with multicore ECU

As illustrated in Figure 3 the VMM can create several VMs, up to the number of ECU cores available. Each VM becomes an independent, isolated system, enabling developers to deploy functions, possibly using different operating systems, on the same multicore ECU. Moreover, each VM can have a dedicated task scheduler for its use case, for example, real-time or non-real-time. This architecture enables parallel computing, where multiple VMs can run concurrently on the same multicore ECU, enhancing the efficiency of hardware utilisation. For example, the authors in (Savithry et al., 2019) have built a partitioning model based on virtualisation to determine and isolate the tasks with different critical levels inside the ADAS. They proposed a real-time algorithm called Criticality-Aware Scheduling, which allows high-priority tasks to receive sufficient resources without interference from low-priority tasks. As a result, the runtime performance of tasks with different priority levels on the same multicore ECUs can be improved.

The study (Savithry et al., 2019) focuses primarily on task requirements and execution. However, the optimisation of task scheduling remains limited by the capabilities of the ECU and its peripheral computing resources, such as the cache. Therefore, it is essential not only to provide an optimised scheduling algorithm but also to design a comprehensive system that accounts for the available computing resources. This becomes especially important when the computing resources are accessed simultaneously by multiple VMs. As a result, developing an effective interface for managing, isolating, and allocating these resources becomes vital. In this direction, the study in Xu et al. (2019) introduces an innovative method known as vC2M, designed to enhance both task scheduling and memory allocation management. This method focuses on shared memory allocation and isolation by integrating memory bandwidth control and cache partitioning. By dividing each task into a virtual CPU synchronized by vC2M, this approach facilitates better isolation of concurrent computing tasks, addressing the challenges of memory allocation and task scheduling in the context of centralized automotive architectures.

However, a highly integrated multicore ECU often contains functions at different levels of criticality and in various operating environments, such as AUTOSAR and entertainment systems. Due to the vehicle safety standard ISO 26262, it is necessary to guarantee the isolation between functions with distinct ASIL (Automotive Safety Integrity Level) and ensure that the safety functions get priority in using computing resources.

3.1.3 Real-time virtualisation

There are two primary types of virtualisation based on how they handle time sensitivity: real-time and non-real-time. Real-time virtualisation is designed to exhibit deterministic behaviour with predictable response times to meet stringent timing requirements. Virtualisation platforms, such as Xen, usually contain several “domains” with different levels of authority over computing resources. Domain0 is the privileged domain responsible for managing VM creation, destruction, and hardware access. It is the first VM to be started when building the virtualisation environment and usually runs the management tools and VMM. In contrast, DomainU refers to the unprivileged domains that host normal VM instances. These domains are managed by Domain0 and do not have direct access to the hardware. Based on the Earliest Deadline First (EDF) principle, the authors in Masrur et al. (2010) proposed a specialised domain known as domRT that focuses on real-time tasks to improve their response time and prevent them from exceeding their worst-case execution time (WCET). A new scheduler called PSEDF (Priority-Based Scheduling plus EDF) is introduced to determine whether a task should be handled by domRT, depending on its priority.

Finally, the authors in Rajan et al. (2018) tested and compared various commercial virtualisation systems compatible with automotive environments and described their pros and cons in different applications, such as core allocation and I/O interfaces. The authors pointed out that the VMM, as an intermediate layer, will increase the overhead of the overall operating system and slow down processes such as runtime and access to shared resources. Paravirtualisation has been proposed recently to mitigate this issue and will be discussed in Section 4.

3.1.4 AUTOSAR

AUTOSAR (AUTomotive Open System ARchitecture) is an embedded system standard developed to manage and abstract hardware platforms by providing application programming interfaces (APIs) that enable access to hardware resources and system services. Founded in 2003 through a partnership among automotive industry manufacturers, AUTOSAR currently includes

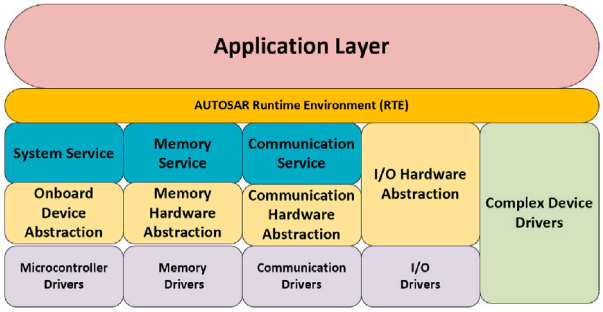

FIGURE 5
An illustration of AUTOSAR modules with a detailed Base Software (BSW) layer description.

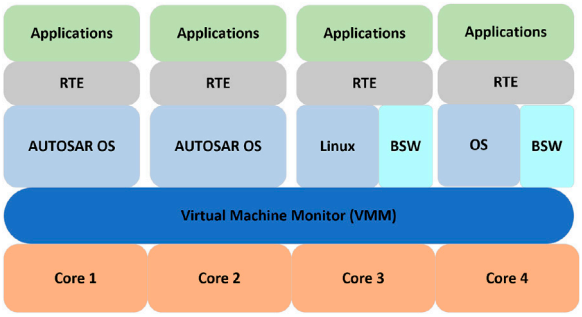

FIGURE 6
An illustration of AUTOSAR with virtualisation.

362 member companies (AUTOSAR, 2024b). It is now the mainstream development standard for in-vehicle architecture, providing a framework to develop modular, scalable, and interoperable automotive software (AUTOSAR, 2024b).

AUTOSAR consists of three main layers: The application layer, the Runtime Environment (RTE), and the Base Software (BSW) layers, as shown in Figure 4. The application layer contains the applications that rely on the AUTOSAR framework and are usually customised by the vehicle manufacturer. RTE acts as the communication layer between the application and the BSW layers, providing a uniform and standardised interface to communicate with hardware components such as sensors and ECUs through the abstraction layer. Thus, the RTE layer facilitates modular design, making migrating applications to different hardware platforms easier. Finally, the BSW layer is the functional combination of several layers, as illustrated in Figure 5, which include:

- Service layer: It defines the abstraction interfaces of the service, including the embedded system functions and the interface of the peripheral I/O devices.
- ECU abstraction layer: It provides a uniform standard to the service and application layers to access the ECU and its peripherals. It also communicates application requests, memory access, and separation to the suitable ECU.
- Microcontroller abstraction layer: It contains device drivers to ensure compatibility of several hardware components, providing a solid embedded layer for the architecture.

With AUTOSAR, OEMs can easily migrate their functional designs across multiple hardware platforms without requiring significant redesign efforts or considering compatibility aspects, as long as the hardware is on the AUTOSAR support list. However, multiple hardware suppliers in the market may lead to potential compatibility issues. For instance, when an ECU is not AUTOSAR-compatible, migrating and running functions already designed for AUTOSAR can be problematic (Becker et al., 2015b). In such cases, virtualisation can offer a viable solution by enabling the operation of heterogeneous operating environments on the same hardware platform. Recall that in a full virtualisation environment, the commands from guest operating systems do not directly communicate with hardware. It is the role of the VMM to translate and direct them to the hardware.

In the multicore ECU architecture shown in Figure 6, multiple operating environments, including AUTOSAR and Linux, run on the same ECU using virtualisation. However, as pointed out by the authors in Evripidou (2016), AUTOSAR has shortcomings in separating applications of different criticality levels, which could potentially lead to the violation of the Freedom From Interference (FFI) requirement, as defined by the ISO26262 standard. To address this issue, the authors leverage the isolation functionality provided by the VMM and propose a novel design method that can efficiently account for applications of different safety levels. According to that, a separate VM within the ECU helps improve the ability to identify and manage mixed-critical applications. By isolating tasks based on their criticality levels, the model ensures that the critical tasks do not exceed their WCET, preventing them from negatively impacting the execution of the subsequent tasks.

The authors in Mounir et al. (2019) demonstrate the implementation of AUTOSAR and Linux on the same multicore ECU and assess the ECU performance to ensure that the latency of safety-critical applications meets the vehicle safety standards. They propose using VirIO to guarantee further temporal and spatial separation between VMs. Similarly, the authors in Reinhardt and Morgan (2014) evaluate the performance of multicore ECU running multiple operating systems, including AUTOSAR, based on the full virtualisation and paravirtualisation technologies. They point out that while paravirtualised systems reduce communication overheads between the hardware and the operating system by bypassing certain management layers, such as the VMM, they suffer from key drawbacks compared to virtualised systems. Specifically, paravirtualisation lacks isolation between the modified operating systems and requires better hardware compatibility.

3.2 Virtualisation for I/O interfaces

In a centralised E/E architecture, a great deal of computing resources and functions are integrated into the hardware platform, including connections to peripheral devices and I/O interfaces. Vehicle functions often need to request data from peripheral devices like onboard sensors. However, a peripheral device can typically serve only one function at a time, which is inefficient and can raise safety concerns. For instance, if a function crushes and holds control of the device indefinitely, it could prevent other critical

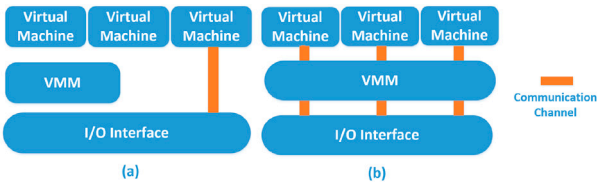

FIGURE 7
Schematic comparison between the Direct and Trap-emulation approaches for I/O interface virtualisation. **(A)** Direct Access and **(B)** Trap-emulation.

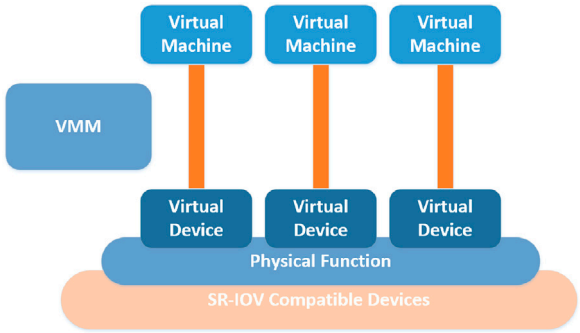

FIGURE 8
Schematic diagram of the Single-Root I/O Virtualisation (SR-IOV).

FIGURE 9
The role of I/O Memory Management Unit (IOMMU) in translating the virtual address requested by the peripheral device to the physical address.

functions from accessing the device, compromising the overall system's operation and safety. Virtualisation can address this issue by creating multiple virtual devices that simulate the behaviour of the physical peripheral device, allowing multiple vehicle functions to share an I/O interface simultaneously. Furthermore, virtualisation can enhance the system's robustness through isolation mechanisms that maintain independent communication channels between VMs and virtual devices. These isolated channels prevent data collisions and ensure that a malfunction of one function does not affect others. This approach not only improves the utilisation of peripheral devices but also boosts the safety and reliability of the AV.

There are three approaches for I/O virtualisation (Sundar Rajan and Nirmala Devi, 2021): 1) Direct access to the peripheral device through a Peripheral Component Interconnect Express (PCIe) tunnel (Architecture Specification, 2023). Instead of utilising the VMM as an intermediate layer, direct connection with the peripheral device increases the data transfer rate and reduces the system's overhead. However, only one VM can access the physical device at a time. As a result, this mode of operation could be the root cause of severe latencies and data collisions when multiple VMs request the same device simultaneously. 2) Trap-emulation is another access mode from a VM to the physical peripheral device (Sugerman et al., 2001). Once a privilege instruction (i.e., an instruction that can only be processed by the kernel) arrives in the VMM, a trap will be raised and a handler will interpose the procedure. Based on the instruction, the VMM will emulate the behaviour of the peripheral devices (Varanasi and Heiser, 2011). The trap-emulation mode ensures isolation and prevents collisions when multiple functions request the same peripheral device simultaneously, but it also increases the execution time. A schematic comparison between Direct and Trap-Emulation approaches is shown in Figure 7. 3) Single-Root I/O Virtualisation (SR-IOV). The direct connection and the trap-emulation approaches are not fully virtualised solutions. The trap-emulation approach introduces significant overheads, while the direct-access mode is limited by the number of connected components that can connect simultaneously to the device. To overcome these limitations, hardware-based I/O virtualisation (also known as hardware-assisted virtualisation) has been proposed (Challa, 2012). According to it, the peripheral device is divided into several virtual devices that emulate the device behaviour, allowing heterogeneous functions to access the same I/O device simultaneously without involving the VMM. Building on the direct access mode and the PCIe, the SR-IOV shown in Figure 8 has been proposed to improve system management and achieve higher data transfer rates between VMs and virtual devices (or virtual functions). The SR-IOV contains Physical Functions (PF) and Virtual Functions (VF) as two modes of system management (Muench et al., 2013). The PF is a physical device found as a PCIe device, which the system can directly manage or modify. The VF is a virtualised partition that is managed by a PF. The number of VFs that can be created depends on the configuration of the I/O interface.

To compare I/O virtualisation approaches, the trap-emulate approach for I/O virtualisation will add 26.77% computational overhead compared to the non-virtualised I/O interface and SR-IOV can reduce this overhead to 1.76% (Dong et al., 2012). Moreover, compared to the trap-simulation approach, SR-IOV reduces the communication latency by 40% for small messages (1 KB) and by 12%–16% for large messages (4 MB) according to Lockwood et al. (2014). However, SR-IOV requires support from I/O devices, which is a major issue in the current development of the vehicle. Nevertheless, the core motivation for implementing SR-IOV is to leverage a single or a small number of ECUs to consolidate different domains to achieve functional isolation, security compliance, efficient resource reuse, and rapid evolutionary development. Even though there exist several limitations, SR-IOV is an important tool for in-vehicle systems to move towards a software-defined vehicle.

Isolation is the key feature of I/O interface virtualisation to prevent data collisions when multiple functions attempt to access the same peripheral device simultaneously. Traditionally, the MMU is responsible for translating virtual addresses accessed by the user into physical addresses for the CPU to access. Furthermore, the MMU manages the memory partition and control of access rights to the virtual addresses to facilitate the management of access rights

and the scope of user programs. Similarly, the I/O Memory Management Unit (IOMMU), shown in Figure 9, provides and enhances the support for the direct memory access (DMA) of the peripheral device. Specifically, the IOMMU remaps the device's DMA requests so that the VM accesses the virtual device's allocated memory instead of the physical device memory. This enables device-level memory isolation, ensuring that each VM can only access its allocated memory space, enhancing isolation between VMs and between the VMs and the physical device. The authors in Jiang et al. (2019) present a system solution for on-board real-time I/O virtualization. They utilize a virtualized I/O management interface called Blue I/O. It allows new peripheral devices to easily connect to this virtualized I/O interface, thus enhancing scalability. A built-in memory management module called BlueTree enables DMA-based functionality. Moreover, the IOMMU translates the virtual address requested by the peripheral device to the physical address using the translation table and directs the request to the dedicated physical address. The translation table contains several levels, and each level includes the partial virtual address corresponding to the next dedicated index in the page table; normally, the page table includes three to five levels. Therefore, it requires the IOMMU to access the table in the memory multiple times to obtain the final physical address. To address this issue, a Translation Lookaside Buffer (TLB) has been proposed. The TLB is a cache to store the translation tables, which are frequently accessed by VMs to translate from virtual addresses to physical addresses. The IOMMU will receive the request from the TLB and find whether it hits the map in the cache without entering the page table. The authors in Sriramakrishnan et al. (2022) point out that this method cannot handle large volumes of real-time data from devices like cameras. Therefore, they propose a combined architecture that provides two parallel paths for real-time and non-real-time peripherals, based on a Peripheral virtualisation Unit that flexibly maps and segregates different types of data paths. Physical Address Translation Tables contain one table for large data paths and an alternative table specifically designed to deal with small data to avoid wasting computational resources.

3.3 Virtualisation of communication networks

In-vehicle data networks serve as the communication highway of modern vehicles and play an essential role in achieving vital functions such as electro-mechanical passenger/driving assistance, body functions, and other features such as infotainment etc. They are used to transmit digital data between two or more communication nodes, such as ECUs and sensors. There are various in-vehicle communication technologies available in the market, such as CAN, FlexRay, LIN, and Ethernet, as well as emerging technologies, such as CANFD and new variants of Ethernet-based systems (Zeng et al., 2016).

In a distributed E/E architecture, multiple communication controllers are employed to coordinate the transmission of data streams of potentially different safety levels. These controllers may use different policies to support secure and reliable data exchange across multiple communication channels. However, as explained in Section 2, this results in a complicated and power-hungry communication network with wiring lengths often exceeding 4 km (Auzanneau, 2013). In contrast, a centralised E/E architecture typically features a single multicore ECU and one communication controller to support data exchange between the ECU and other devices via a single communication bus. As a result, while centralisation reduces wiring complexity, it may lead to data collisions and increased latency of critical functions, thereby jeopardising compliance with safety standards such as ISO 26262.

To mitigate the safety risks in a centralised architecture, it is crucial to minimise the inter-dependencies and interactions between functions of mixed-criticality. For example, within a multicore ECU, the communication controller should prioritise data transmissions related to safety-critical functions, as these are vital for the stable operation of the vehicle. In this context, virtualisation emerges as a powerful technical solution. By isolating different functions, virtualization enables them to operate independently while they can simultaneously access shared resources, such as the communication controller. This isolation ensures that safety-critical tasks can maintain their priority and are not delayed or interrupted by less critical functions, thus enhancing both safety and efficiency.

3.3.1 CAN virtualisation

The Controller Area Network (CAN) is a multi-master bus communication system developed by BOSCH in 1986, which remains the mainstream bus communication technology in the automotive industry. Unlike a master-slave communication system, the CAN bus operates as a protocol controller that allows each node to communicate or broadcast its messages. In the idle state, any node on the CAN bus can initiate a message transmission and enter the writing queue. To determine message priorities and resolve conflicts, there is an arbitration segment in the first bit of the message of each function, with smaller values indicating higher priority.

In a centralized architecture, multiple VMs running mixed-criticality functions on the same multicore ECU, accessing the CAN bus and a single CAN bus controller in parallel, can pose challenges in a virtualized environment. A scheduling mechanism is required to avoid the collisions of messages generated by VMs that share the same physical CAN bus controller. To address this issue, the authors in Herber et al. (2013) proposed the virtualised CAN controller shown in Figure 10 based on the SR-IOV mechanism discussed in Section 3.2. They implemented various VMs as virtual CAN controllers to process requests from dedicated VMs running on multicore ECUs. Each virtual CAN controller is guaranteed access to independent resources, such as memory, to send and receive messages. Messages need to be buffered in case the CAN bus is occupied by another VM, preventing immediate access. Consequently, a dedicated buffer is required for each virtual controller to prevent system failure in the event of a VM malfunction. The virtualised CAN controller also includes the following virtual modules to ensure further isolation and safety while transferring data from the VMs to the CAN bus:

- Host Controller Interface: It coordinates data flow, ensuring that data from each VM can be correctly transferred to the corresponding Virtual Controller. It also receives and

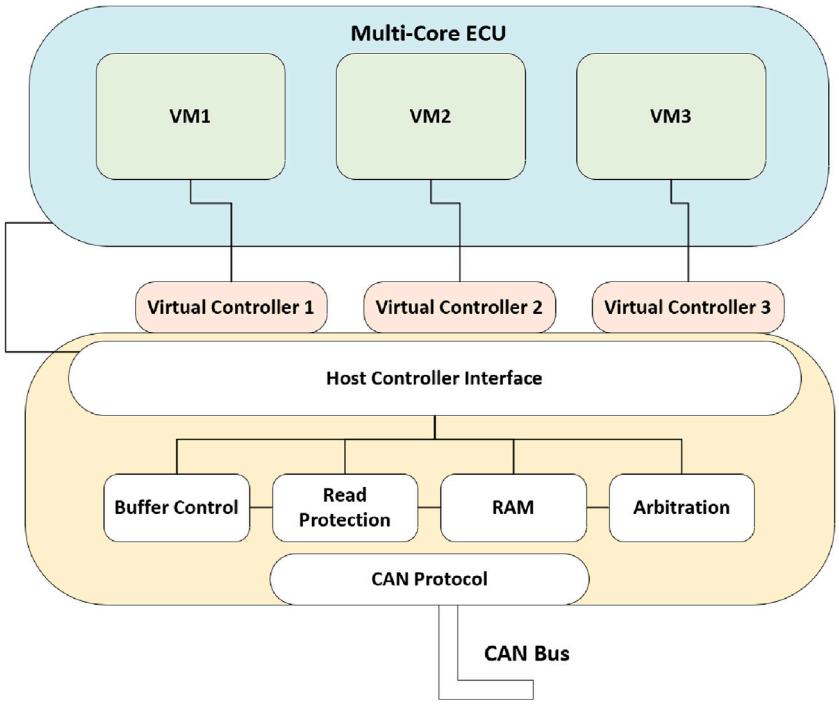

FIGURE 10
The architecture of the CAN Controller leveraging self-virtualisation according to (Herber et al., 2013).

processes commands from Virtual Controllers to ensure that these commands are correctly executed.
- Buffer control: This module manages the storage and scheduling of packets to ensure they do not interfere with each other while forwarding them to the CAN bus.
- Read protection: This module prevents unauthorized VMs from accessing other VMs' data, ensuring data security.
- Arbitration: It is a control mechanism that selects the messages with the highest priority when receiving the sending request from the VMs.

Virtualisation allows multiple VMs to emulate the behaviour of the CAN controller and share the same physical CAN controller. This resource sharing reduces the need for multiple physical CAN controllers, cutting down hardware costs. Each VM with various operating systems or environments can independently access CAN bus resources through a virtual CAN controller. This flexibility makes it easier to expand and upgrade the system, simplifying the addition of new functionalities. Moreover, virtualisation provides isolation mechanisms that make the operations and data generated by a VM independent of others. This isolation improves system security and stability by preventing errors or malicious behaviour in one VM from interfering with the operation of other VMs. In Herber et al. (2014a), exploring potential safety issues related to virtualised CAN controllers, explicitly focusing on Denial-of-Service (DoS) attacks. For example, such an attack can manipulate a VM to send message requests to the physical CAN controller at an unexpectedly high rate, preventing other VMs from maintaining real-time communication. To address this issue, the authors proposed a scheduling approach based on temporal isolation, which minimises the time between when a VM preempts the CAN bus and when it releases control. This approach aims to mitigate the risk of one VM overwhelming the CAN controller, even in the event of a potential attack.

The authors further developed their work in Herber et al. (2015) compare the paravirtualisation approach with hardware-assisted virtualisation applied to CAN controller virtualisation schemes. They point out that hardware-assisted virtualisation can achieve lower latency than paravirtualisation. However, since CAN controllers do not support hardware-assisted virtualisation, there is still a long way to go before they are commercially available on automotive platforms.

Instead of virtualising the CAN controller, the study in Herber et al. (2014b) introduces a novel approach called Virtual CAN (VCAN) that focuses on time-based isolation of multiple simultaneous communications on the same physical bus. Their method virtualises the physical CAN bus into multiple independent VCAN instances, with each instance allocated its own resources, time slices and bandwidth. The authors suggest dividing the time of the physical CAN bus into multiple time slices using Time Division Multiplexing, a multiplexing technology commonly applied in wireless communication systems. Each VCAN is granted exclusive access to the physical bus during its designated time window, ensuring that mission-critical tasks can transfer data on time in time-sensitive situations. Additionally, the VCAN system utilises a token bucket mechanism to prevent any single VCAN instance from continuously dominating the bus, as well as a traffic shaping mechanism to ensure that the transmission rate for each VCAN instance does not exceed its assigned bandwidth limit. The proposed approach not only

ensures that network bandwidth can be dynamically allocated to mixed-criticality functions but also enhances isolation between functions with different security levels. This is particularly crucial in a centralized E/E architecture, where mixed-criticality functions share the same ECU and communication data bus. However, as the number of virtual CAN channels (vCANs) increases—especially beyond five—additional latency arises, potentially causing missed runtime deadlines. Moreover, this approach requires adding an extra module to the existing CAN controller, which in turn necessitates additional procedures to demonstrate compliance with the ISO 26262 safety standard.

Ethernet has higher bandwidth than traditional in-vehicle buses, such as CAN, to meet modern vehicles' data transfer rate demands, especially when dealing with applications like high-definition cameras and ADAS. Moreover, Ethernet supports a wide range of communication protocols like Time-Sensitive Networking (TSN) based on IEEE 802.1, allowing it to flexibly adapt to different types of data transmission requirements while being easily expandable to accommodate future technological developments. Therefore, the industry is gradually adopting Ethernet to vehicle communication networks. In Reinhardt et al. (2015), the authors discussed the migration approach of mapping the CAN network to the Ethernet and allowing the virtual ECUs with the Ethernet protocol to connect with the current CAN interface. They utilize the CANnelloni software bridging tool, and CAN signals are aggregated into a single Ethernet frame and transmitted between VMs via the User Datagram Protocol (UDP). A buffer has been set to store the incoming CAN frame and will be transferred once the buffer size meets the Maximum Transmission Unit (MTU) of the Ethernet. Furthermore, the authors define the scheduling policy and CAN frame order for transmission from VMs to the Ethernet interface. However, the evaluation process does not account for real-time performance, particularly concerning mixed-criticality functions—a crucial consideration for time-sensitive systems like those found in automotive applications. Furthermore, although industry standards like Time-Sensitive Networking (TSN) have been established to ensure Ethernet performance and security in real-time contexts, automotive systems often combine multiple communication network types and varying switch protocol configurations, posing significant challenges in centralised architectures. Therefore, Software-Defined Network has been proposed as a solution which we will discuss in Section 3.3.

Before requirement concluding this section, it shall 4.3 be noted that besides in-vehicle communication networks, vehicles can also connect over the air to external systems such as the cloud, other road users, and roadside infrastructures using a Vehicle-to-everything (V2X) communication technology such as ITS-G5 or C-V2X. A review of virtualisation techniques of the radio air interface of V2X communication technologies does not fall in the scope of this review article.

4 Identification of research gaps

Following the literature review of SOTA research in automotive virtualisation technologies, we identify promising areas for future work. Our investigation is informed by exploring virtualisation technologies successfully applied in other relevant domains, such as servers and desktop applications, and assessing their potential adaptation in automotive. Specifically, key areas for research and development in automotive virtualisation include cache partitioning, paravirtualisation technologies, software-defined networking for in-vehicle communication networks, and the use of digital twins for prototyping.

4.1 Resource allocation

We have discussed in Section 3.1 how virtualisation can be used to increase the flexibility and efficiency of allocating computing resources to various automotive functions within multicore ECU environments. However, shared resources may also include ECU caches, which can be contested by multiple VMs simultaneously. Specifically, when multiple concurrent running tasks request the memory from the same cache set, they may interfere with each other and cause the cache miss, as we mentioned in Section 3.2, which will lead to a significant performance overhead and a possibility to miss the task's running deadline. Cache partitioning techniques leveraging isolated virtualised partitions have been investigated for desktop applications in Sheikh and Pasha (2021), Gifford et al. (2021), Sohal et al. (2022), but their methods are not optimised for the real-time system like software inside automotive. Furthermore, the existing research for automotive applications is limited, including, to the best of our knowledge, only the following two references. Firstly, the study in Xu et al. (2017) introduces dynamic cache allocation under the strict timing-sensitive real-time system. This approach separates the correspondence between the ECU cores and the cache and is reassigned to the dedicated VMs dynamically by the VMM based on the timing requirements. It provides a better isolation strategy between mix-critical or real and non-real tasks, which is extremely well suited to the operating environment required by vehicles (i.e., vehicle entertainment system and driving system). Secondly, the study in Vasu and Ramaprasad (2020) proposes a method for mapping AUTOSAR tasks to multicore ECUs by building a model that considers cache usage. We believe that shared cache access in automotive virtualised environments is an important research area to pursue. A well-managed sharing mechanism for the cache will prevent the ECU from reading data frequently from memory, and that can reduce up to seven times the task's execution time (Xu et al., 2017).

Furthermore, recall from Section 3.1 that the VMM can compromise the real-time communication performance between the operating system and the hardware. Therefore, paravirtualisation has been introduced as a light-optimised virtualisation layer based on the full-virtualisation technology, see Figure 11. Paravirtualisation modifies the operating system by adding a dedicated API to optimise the commands issued by the operating system and sent to the VMM (Babu et al., 2014). Table 1 shows the comparison between paravirtualsation and full-virtualisation. Compared to a system without virtualisation, the performance overhead of paravirtualisation is only up to 4% (Li et al., 2017). Therefore, researchers and engineers have gradually applied this innovative technology within the vehicle software architecture.

Recent research on this topic has focused on in-vehicle infotainment, including (Karthik et al., 2018; Sinha et al., 2020).

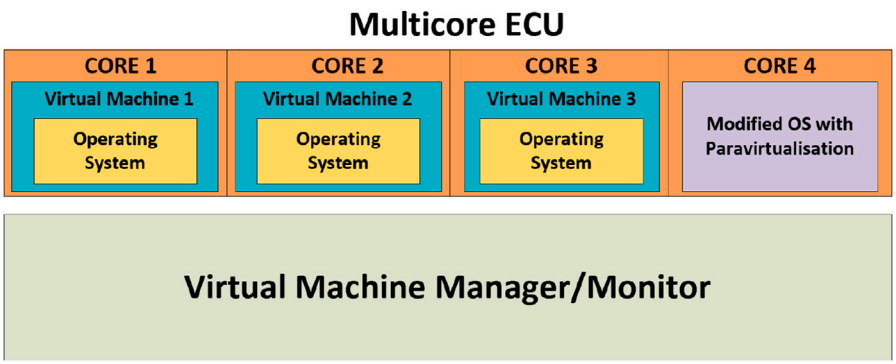

FIGURE 11
An illustration of multicore ECU with paravirtualisation applied in one of the cores.

TABLE 1 Comparison of Paravirtualisation vs. Full Virtualisation.

	Paravirtualisation	Full virtualisation
Setup	Requires modification of the Guest OS to utilize paravirtualisation	No modification is needed in the Guest OS. The VMM provides a fully emulated hardware interface
Performance overhead	Lower overhead compared to full virtualization because fewer instructions need to be trapped/emulated	Higher overhead due to full hardware emulation
Hardware emulation	Partial or minimal; the VMM exposes simplified or paravirtualized drivers/devices	Extensive; the VMM emulates full hardware devices
Compatibility	Guest OS must support paravirtualisation	Broad compatibility; virtually any unmodified OS that runs on the underlying ECU architecture can be installed
Examples	Xen PV, VMware Paravirtual	VMware

By modifying the system kernel to make it suitable for real-time applications, the authors propose a paravirtualisation-based solution that allows systems like Android, which are not specifically designed for automotive, to be compatible with and communicate with virtualised real-time systems. This approach reduces the overhead caused by multi-layer virtualisation architectures and provides an intuitive interface for users to monitor the status of the vehicle's real-time functions. Furthermore, it hides the hardware device driver information from the operating system, thereby enhancing vehicle security. However, customising the system kernel and API functionality when updating the hardware platform—especially when transitioning to a fully virtualised platform—sometimes requires rewriting significant portions of the code, as each paravirtualisation implementation has a dedicated driver for compatibility with the VMM. Moreover, paravirtualisation requires developers to modify the system before using the VMM, which leads to increased development costs. These shortcomings indicate there's still a long way forward before the widespread commercial adoption of paravirtualisation technologies, which is a promising area for further research.

4.2 Software-defined networking

To achieve the vision of fully autonomous driving, various hardware components need massive upgrades and growth. Specialised hardware devices may come from different manufacturers with private communication and network control protocols (Kreutz et al., 2014). Therefore, automotive manufacturers are required to statically define and connect multiple hardware platforms with dedicated protocols during the development stage. Moreover, multiple types of communication data buses, such as CAN, LIN and Ethernet, exist in modern vehicles. This distributed architecture increases the complexity of the communication network system and prevents the introduction of new hardware after the vehicle design phase is complete. Thus, Software Defined Networking (SDN) has been proposed as an alternative software-based solution to make communication networks more flexible and centralised.

According to the Open Networking Foundation, SDN separates the control from the data plane in the distributed architecture and uses an SDN controller to manage the different types of communication networks (Ope, 2024). The data plane broadcasts and forwards data required by various functions or applications. It consists of devices like network protocol routers or switches, which are the connecting layers between the SDN controller and the network. SDN abstracts the architecture of the communication network into programmable APIs and centralises the network control into one flexible and programmable interface that resides within the SDN controller. This approach decouples the hardware from software development and reduces costs in the early design stages.

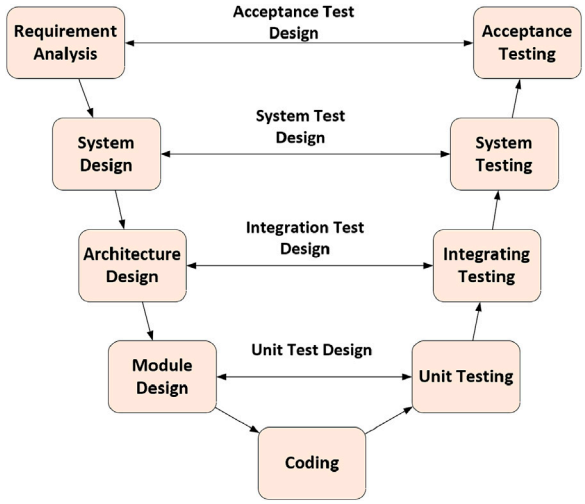

FIGURE 12
An illustration of the Software Development Life-Cycle (SDLC)-V model.

In the automotive domain, the authors in Halba and Mahmoudi (2018) have utilised SDN to enable interoperability between different communication networks within the vehicle. In (Hackel et al., 2019; Haeberle et al., 2020), the time-sensitive networking (TSN) is integrated within the SDN architecture to improve the performance of the vehicle's real-time system. The authors define the data flow and the data traffic control between the SDN controller and the network under the requirements of the TSN standard. Moreover, the study in Mariño et al. (2022) reviews the current in-vehicle networks and points out that SDN can play a fundamental role in developing software-defined vehicles.

Using virtualisation, several virtual SDN (vSDN) controllers can be established. A virtual controller consists of communication network virtualisation, a virtual data switch for forwarding data to the destination network, and a VMM for isolation between different vSDN controllers (Blenk et al., 2015). The vSDN controller allows multiple clients to share the same SDN controller. This sharing scheme improves network efficiency and isolation between multiple dependencies. In the automotive domain, current research on SDN and vSDN focuses on the wireless V2X and 5G communication aspects (Liu et al., 2019), while the existing literature on the application of vSDN within the in-vehicle communication network is limited.

4.3 Prototyping

Virtualisation not only improves the efficiency of in-vehicle hardware and software architectures but can also benefit the functions development phase of the vehicle. Traditionally, the Software Development Life-Cycle (SDLC)-V model is used to guide the vehicle functions development process (Goyal and Mistry, 2020). This model details and expands the milestones for each development stage, linking the prototyping and validation phases, as shown in Figure 12. Sometimes, compatibility issues between hardware and software, as well as errors not foreseen during the simulation and software development phases, are only discovered during testing, e.g., during the field trials every automaker must conduct. Unfortunately, when these issues arise, the redesign cost can be substantial, especially for vehicle functions related to hardware coordination. Virtualisation can offer a potential solution to this problem by abstracting hardware devices. Developers can use virtualised hardware to simulate how the application will behave on the hardware, thus helping to identify and mitigate problems before the physical hardware is involved.

Automotive functions are traditionally divided into multiple domains according to the SDLC-V procedure, and each domain involves various testing and integration hardware platforms, incurring a high cost in embedded system development and a lack of flexibility. Assisted by virtualisation technology, a virtual ECU (vECU) model has been proposed to develop, test, and validate automotive control software without relying on the actual hardware. A vECU is implemented by simulating the ECU functionality on the developer's PC, which improves development efficiency and reduces costs and reliance on the physical hardware. In Goyal and Mistry (2020), the authors propose an improved SDLC-V model that integrates the various functional development domains into a single vECU test platform, reducing the dependence on hardware during testing and optimising the development procedure in the early stage. Furthermore, the authors in Franco et al. (2016) demonstrate how developers can prototype and test according to the AUTOSAR standard without actual hardware. They combine tools such as dSPACE and MathWorks and emulate the communication mechanisms by utilising the Virtual Functional Bus.

The references above focus mainly on vehicle hardware components' functional level and partial modelling. However, the fact that the vehicle contains a multi-level hardware architecture, where all connected hardware will interact with each other, is critical in the early development stages. A systematic virtual model is required to explore the system-level design and further improve the compatibility and scalability of the peripherals and communication buses as well. For this purpose, a technology called digital twins has been proposed as a virtual model that can simulate the system-level behaviour. It refers to the creation of an accurate digital replica of a physical object or system, which not only can be synchronized with the actual physical object but can also be monitored, simulated, and optimised in real-time. The authors in Kabir et al. (2023) propose a system-level model called ViVE based on digital twins. They utilise vECUs as an interface to connect with other virtual hardware components, which contain various virtual models to prototype and emulate the behaviour of the physical hardware components. The model supports developers in configuring the specific virtual components and testing the function flow, which includes the interaction between hardware systems. Their model expands the testing view from a single component towards the full system and supports extensibility for future incoming use cases. Therefore, in Kabir and Ray (2023), depending on the previous ViVE model, they further propose a framework that allows developers to explore the impact of vehicle component failure and detect potential safety issues with functional designs during the prototyping phase. Similarly, authors in Cinque et al. (2022) propose a fault injection framework based on a VMM called "Jailhouse" that can be used to test the isolation and integrity between each VM

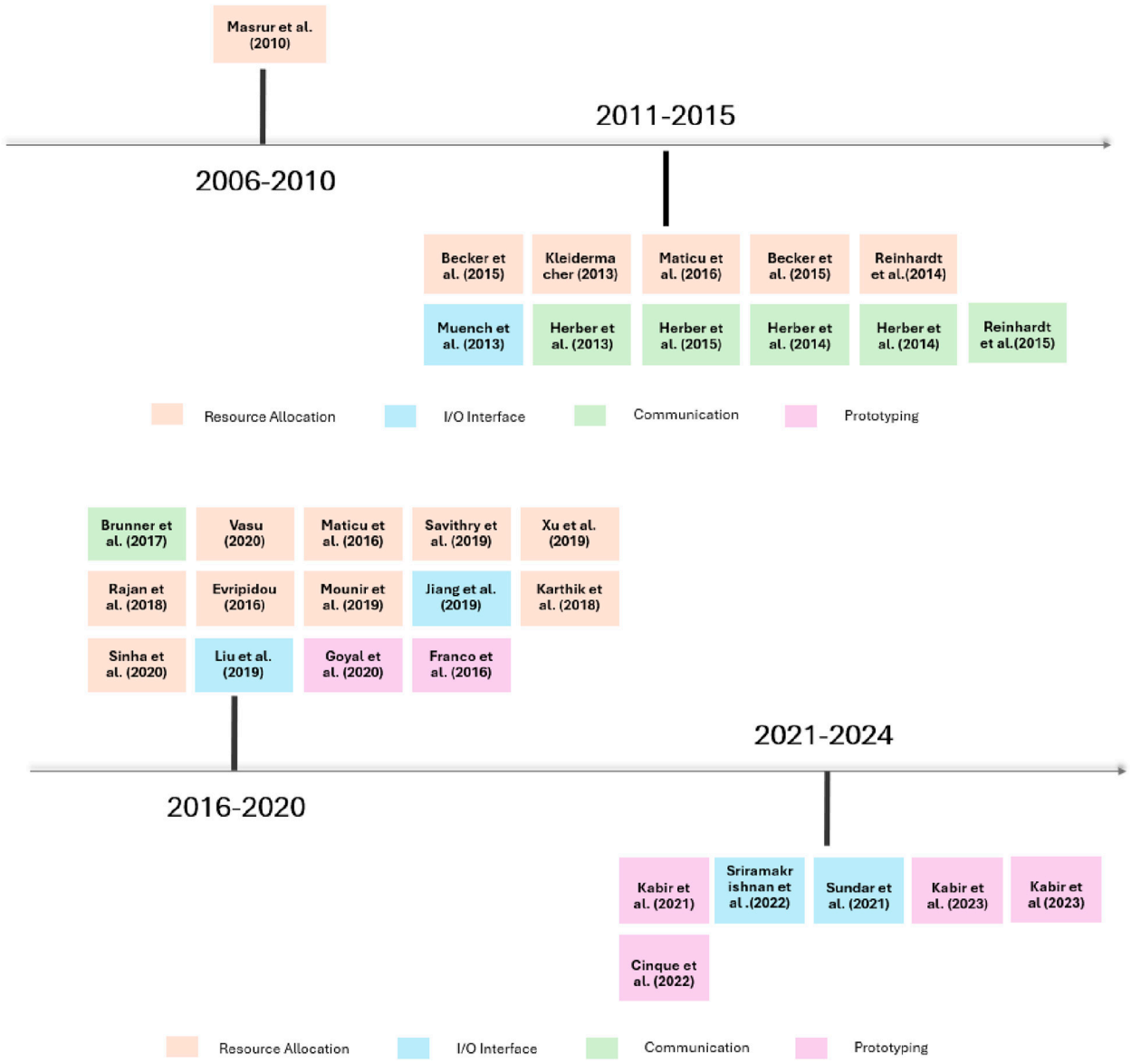

FIGURE 13
Timeline of published research in automotive virtualisation technologies.

and further test the vehicle functions running on the VMs for compliance with the ISO 26262 standard.

5 Discussion and conclusion

This review article reviewed SOTA research (approximately 30 key studies) in automotive virtualisation technologies. As depicted in Figure 13, most of the referenced studies were published in the last decade, which coincides with the rise of AI models and automated driving systems such as Tesla Autopilot, which have benefited from advancements in Neural Process Units and GPUs. The volume of published research in this field has reached a plateau. Centralised automotive architectures and virtualisation are not yet mainstream research topics, likely because the commercially available ECUs are not yet capable of running several virtual machines simultaneously. More importantly, current ECUs are not optimised for virtualisation and this will cause compatibility issues. According to a report by BOSCH Solutions B. M., (2023b), nearly 70% of OEMs still use a distributed architecture, and even by 2029, only up to 30% are estimated to adopt fully centralised architectures. Constrained by the current state of microprocessor technology, most ECUs can only marginally meet the minimum requirements needed to support virtualisation. Moreover, the existing E/E architecture is already mature enough for OEMs to deploy and upgrade their L2 autonomy functions and features without requiring immediate, large-scale changes in the architecture. Additionally, while applying virtualisation within the vehicle software architecture, the following aspects must be considered to ensure the security of the in-vehicle virtual environment: Real-Time Constraints, Certification & Compliance, and Safety Mechanisms & Diagnostics (Bagalini et al., 2017). This will increase the timing cost in the early development

stage, especially for the Certification & Compliance process. Developers need to perform functional safety analyses of the VMM and its configuration based on ISO 26262-compliant analysis and testing tools like Failure Mode and Effects Analysis (FMEA). These procedures will result in short-term cost increases. Nevertheless, centralised architectures and virtualisation hold significant potential for future software-defined vehicles.

Therefore, this survey article offers automotive stakeholders an up-to-date overview of SOTA virtualisation techniques and their associated benefits, including effective isolation of services of mixed-criticality, seamless integration of vehicle functions developed by different vendors, easier migration to new hardware platforms, and the flexibility to add or remove vehicle functions. Furthermore, we outline the most promising application areas of automotive virtualisation in the near term. Specifically, we believe that integrating paravirtualisation with resource allocation and the I/O interface in the vehicle is a promising direction, as it could help reduce the overhead that virtualisation brings.

Author contributions

ZG: Writing–original draft, Conceptualization, Investigation, Methodology. KK: Methodology, Supervision, Writing–review and editing. MD: Writing–review and editing. RW: Writing–review and editing.

Funding

The author(s) declare that no financial support was received for the research, authorship, and/or publication of this article.

Acknowledgments

The authors are with WMG, University of Warwick. Mehrdad Dianati also holds a part-time professorial post at the School of Electronics, Electrical Engineering and Computer Science (EEECS), Queen's University of Belfast. e-mail: {zixuan.guo, konstantinos.koufos, m.dianati, r.woodman}@warwick.ac.uk, m.dianati@qub.ac.uk.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The author(s) declare that Generative AI was used in the creation of this manuscript. To improve the use of English language in some sentences.

Publisher's note

References

Architecture Specification (2023). IntelÂ® virtualization technology for directed i/o architecture specification

ASAM Association (2023). Available at: https://www.asam.net/index.php?eID=dumpFile&t=f&f=798&token=148b5052945a466cacfe8f31c44eb22509d5aad1 (Accessed June 22, 2023).

Askaripoor, H., Hashemi Farzaneh, M., and Knoll, A. (2022). E/e architecture synthesis: Challenges and technologies. *Electronics* 11, 518. doi:10.3390/electronics11040518

AUTOSAR (2024a). Autosar.

AUTOSAR (2024b). Autosar partners.

Auzanneau, F. (2013). Wire troubleshooting and diagnosis: review and perspectives. *Prog. Electromagn. Res. B* 49, 253–279. doi:10.2528/pierb13020115

Babu, S. A., Hareesh, M., Martin, J. P., Cherian, S., and Sastri, Y. (2014). "System performance evaluation of para virtualization, container virtualization, and full virtualization using xen, openvz, and xenserver," in 2014 fourth international conference on advances in computing and communications *(IEEE)*, 247–250.

Bagalini, E., Sini, J., Reorda, M. S., Violante, M., Klimesch, H., and Sarson, P. (2017). "An automatic approach to perform the verification of hardware designs according to the iso26262 functional safety standard," in *2017 18th IEEE Latin American test symposium (LATS)* (IEEE), 1–6.

Bandur, V., Selim, G., Pantelic, V., and Lawford, M. (2021). Making the case for centralized automotive e/e architectures. *IEEE Trans. Veh. Technol.* 70, 1230–1245. doi:10.1109/tvt.2021.3054934

Becker, M., Dasari, D., Nã©lis, V., Behnam, M., Pinho, L. M., and Nolte, T. (2015b). "Investigation on autosar-compliant solutions for many-core architectures," in *2015 euromicro conference on digital system design*, 95–103. doi:10.1109/DSD.2015.63

Becker, M., Dasari, D., Nélis, V., Behnam, M., Pinho, L. M., and Nolte, T. (2015a). "Investigation on autosar-compliant solutions for many-core architectures," in *2015 euromicro conference on digital system design* (IEEE), 95–103.

Blenk, A., Basta, A., Reisslein, M., and Kellerer, W. (2015). Survey on network virtualization hypervisors for software defined networking. *IEEE Commun. Surv. and Tutorials* 18, 655–685. doi:10.1109/comst.2015.2489183

Brunner, S., Roder, J., Kucera, M., and Waas, T. (2017). "Automotive e/e-architecture enhancements by usage of ethernet tsn," in *2017 13th workshop on intelligent solutions in embedded systems (WISES)* (IEEE), 9–13.

Challa, N. R. (2012). "Hardware based i/o virtualization technologies for hypervisors, configurations and advantages-a study," in *2012 IEEE international conference on cloud computing in emerging markets (CCEM)* (IEEE), 1–5.

Cinque, M., De Simone, L., and Marchetta, A. (2022). "Certify the uncertified: towards assessment of virtualization for mixed-criticality in the automotive domain," in *2022 52nd annual IEEE/IFIP international conference on dependable systems and networks workshops (DSN-W)* (IEEE), 8–11.

Di Natale, M., and Sangiovanni-Vincentelli, A. L. (2010). Moving from federated to integrated architectures in automotive: the role of standards, methods and tools. *Proc. IEEE* 98, 603–620. doi:10.1109/jproc.2009.2039550

Dong, Y., Yang, X., Li, J., Liao, G., Tian, K., and Guan, H. (2012). High performance network virtualization with sr-iov. *J. Parallel Distributed Comput.* 72, 1471–1480. Communication Architectures for Scalable Systems. doi:10.1016/j.jpdc.2012.01.020

Evripidou, C. (2016). *Scheduling for mixed-criticality hypervisor systems in the automotive domain*. University of York. Ph.D. thesis.

Franco, F. R., Neme, J. H., Santos, M. M., da Rosa, J. N. H., and Fabbro, I. M. D. (2016). Workflow and toolchain for developing the automotive software according autosar standard at a virtual-ecu, 869–875. doi:10.1109/ISIE.2016.7745004

Gifford, R., Gandhi, N., Phan, L. T. X., and Haeberlen, A. (2021). "Dna: dynamic resource allocation for soft real-time multicore systems," in *2021 IEEE 27th real-time and embedded technology and applications symposium (RTAS)* (IEEE), 196–209.

Goyal, R., and Mistry, P. (2020). Standard process for establishment of ECU virtualization as integral part of automotive software development life-cycle. *Tech. Rep.* doi:10.4271/2020-01-5007

Hackel, T., Meyer, P., Korf, F., and Schmidt, T. C. (2019). "Software-defined networks supporting time-sensitive in-vehicular communication," in *2019 IEEE 89th vehicular technology conference (VTC2019-Spring)* (IEEE), 1–5.

Haeberle, M., Heimgaertner, F., Loehr, H., Nayak, N., Grewe, D., Schildt, S., et al. (2020). "Softwarization of automotive e/e architectures: A software-defined networking approach," in *2020 IEEE vehicular networking conference (VNC)* (IEEE), 1–8.

Halba, K., and Mahmoudi, C. (2018). "In-vehicle software defined networking: an enabler for data interoperability," in *Proceedings of the 2nd international conference on information system and data mining*, 93–97.

Herber, C., Reinhardt, D., Richter, A., and Herkersdorf, A. (2015). "Hw/sw trade-offs in i/o virtualization for controller area network," in *2015 52nd ACM/EDAC/IEEE design automation conference (DAC)* (IEEE), 1–6.

Herber, C., Richter, A., Rauchfuss, H., and Herkersdorf, A. (2013). "Self-virtualized can controller for multi-core processors in real-time applications," in *Architecture of computing systems–ARCS 2013: 26th international conference, Prague, Czech republic, february 19-22, 2013. Proceedings 26* (Springer), 244–255.

Herber, C., Richter, A., Rauchfuss, H., and Herkersdorf, A. (2014a). Spatial and temporal isolation of virtual can controllers. *ACM SIGBED Rev.* 11, 19–26. doi:10.1145/2668138.2668141

Herber, C., Richter, A., Wild, T., and Herkersdorf, A. (2014b). "A network virtualization approach for performance isolation in controller area network (can)," in *2014 IEEE 19th real-time and embedded technology and applications symposium (RTAS)* (IEEE), 215–224.

Jiang, Z., Audsley, N., Dong, P., Guan, N., Dai, X., and Wei, L. (2019). "Mcs-iov: real-time i/o virtualization for mixed-criticality systems," in *2019 IEEE real-time systems symposium RTSS* (IEEE), 326–338.

Kabir, M. R., Mishra, N., and Ray, S. (2021). "Vive: virtualization of vehicular electronics for system-level exploration," in *2021 IEEE international intelligent transportation systems conference (ITSC)* (IEEE), 3307–3312.

Kabir, M. R., Ravi, B. B. Y., and Ray, S. (2023). A virtual prototyping platform for exploration of vehicular electronics. *IEEE Internet Things J.* 10, 16144–16155. doi:10.1109/jiot.2023.3267339

Kabir, M. R., and Ray, S. (2023). "Virtualization for automotive safety and security exploration," in *2023 IEEE 16th Dallas circuits and systems conference (DCAS)* (IEEE), 1–4.

Kampert, E., Schettler, C., Woodman, R., Jennings, P. A., and Higgins, M. D. (2020). Millimeter-wave communication for a last-mile autonomous transport vehicle. *IEEE Access* 8, 8386–8392. doi:10.1109/access.2020.2965003

Kampmann, A., Lüer, M., Kowalewski, S., and Alrifaee, B. (2022). "Optimization-based resource allocation for an automotive service-oriented software architecture," in 2022 IEEE intelligent vehicles symposium (IV) *(IEEE)*, 678–687.

Karthik, S., Ramanan, K., Devshatwar, N., Paul, S., Mahaveer, V., Zhao, S., et al. (2018). "Hypervisor based approach for integrated cockpit solutions," in *2018 IEEE 8th international conference on consumer electronics-berlin (ICCE-Berlin)* (IEEE), 1–6.

Kleidermacher, D. (2013). "System virtualization in multicore systems," in *Real world multicore embedded systems* (Elsevier), 227–267.

Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2014). Software-defined networking: a comprehensive survey. *Proc. IEEE* 103, 14–76. doi:10.1109/JPROC.2014.2371999

Kugele, S., Obergfell, P., Broy, M., Creighton, O., Traub, M., and Hopfensitz, W. (2017). "On service-orientation for automotive software," in 2017 IEEE international Conference on software architecture (ICSA) *(IEEE)*, 193–202.

Li, Z., Kihl, M., Lu, Q., and Andersson, J. A. (2017). "Performance overhead comparison between hypervisor and container based virtualization," in *2017 IEEE 31st International Conference on advanced information networking and applications (AINA)* (IEEE), 955–962.

Liu, K., Xu, X., Chen, M., Liu, B., Wu, L., and Lee, V. C. (2019). A hierarchical architecture for the future internet of vehicles. *IEEE Commun. Mag.* 57, 41–47. doi:10.1109/mcom.2019.1800772

Lockwood, G. K., Tatineni, M., and Wagner, R. (2014). "Sr-iov: performance benefits for virtualized interconnects," in *Proceedings of the 2014 annual conference on extreme science and engineering discovery environment*, 1–7.

Macher, G., Höller, A., Armengaud, E., and Kreiner, C. (2015). "Automotive embedded software: migration challenges to multi-core computing platforms," in *2015 IEEE 13th international conference on industrial Informatics (INDIN)* (IEEE), 1386–1393.

Mariño, A. G., Fons, F., and Arostegui, J. M. M. (2022). The future roadmap of in-vehicle network processing: a hw-centric (r-) evolution. *IEEE access* 10, 69223–69249. doi:10.1109/access.2022.3186708

Masrur, A., Drossler, S., Pfeuffer, T., and Chakraborty, S. (2010). "Vm-based real-time services for automotive control applications," in *2010 IEEE 16th international conference on embedded and real-time computing systems and applications* (IEEE), 218–223.

Maticu, F., Pop, P., Axbrink, C., and Islam, M. (2016). Automatic functionality assignment to AUTOSAR multicore distributed architectures. *Tech. Rep.* doi:10.4271/2016-01-0041

Maul, M., Becker, G., and Bernhard, U. (2018). Service-oriented ee zone architecture key elements for new market segments. *ATZelektronik Worldw.* 13, 36–41. doi:10.1007/s38314-017-0092-4

Monot, A., Navet, N., Bavoux, B., and Simonot-Lion, F. (2012). Multisource software on multicore automotive ecus—combining runnable sequencing with task scheduling. *IEEE Trans. Industrial Electron.* 59, 3934–3942. doi:10.1109/tie.2012.2185913

Mounir, M., AbdelSalam, M., Safar, M., and Salem, A. (2019). "Hardware-assisted virtualization for heterogeneous automotive applications," in *2019 14th international conference on computer engineering and systems (ICCES)* (IEEE), 195–200.

Muench, D., Isfort, O., Mueller, K., Paulitsch, M., and Herkersdorf, A. (2013). "Hardware-based i/o virtualization for mixed criticality real-time systems using pcie sr-iov," in 2013 IEEE 16th international Conference on computational Science and engineering *(IEEE)*, 706–713.

Navale, V. M., Williams, K., Lagospiris, A., Schaffert, M., and Schweiker, M.-A. (2015). (r) evolution of e/e architectures. *SAE Int. J. Passeng. Cars-Electronic Electr. Syst.* 8, 282–288. doi:10.4271/2015-01-0196

Ondrej Burkacky, J. D., and Apostu, G. D. S. (2019). Automotive software and electrical/electronic architecture: implications for oems

Open networking foundation (2024). Available at: https://opennetworking.org/ (Accessed May 23, 2024).

Rajan, A. K. S., Feucht, A., Gamer, L., Smaili, I., and M., N. D. (2018). Hypervisor for consolidating real-time automotive control units: its procedure, implications and hidden pitfalls. *J. Syst. Archit.* 82, 37–48. doi:10.1016/j.sysarc.2018.01.001

Reinhardt, D., Güntner, M., Kucera, M., Waas, T., and Kühnhauser, W. (2015). "Mapping can-to-ethernet communication channels within virtualized embedded environments," in *10th IEEE international symposium on industrial embedded systems (SIES)* (IEEE), 1–10.

Reinhardt, D., and Morgan, G. (2014). "An embedded hypervisor for safety-relevant automotive e/e-systems," in *Proceedings of the 9th IEEE international symposium on industrial embedded systems (SIES 2014)* (IEEE), 189–198.

Research and Markets (2022). *China autonomous driving and cockpit domain control unit (dcu) industry report*.

Rödel, C., Stadler, S., Meschtscherjakov, A., and Tscheligi, M. (2014). "Towards autonomous cars: the effect of autonomy levels on acceptance and user experience," in *Proceedings of the 6th international conference on automotive user interfaces and interactive vehicular applications*, 1–8.

Rover, J. L. (2022). Jaguar land rover announces partnership with nvidia.

Salay, R., Queiroz, R., and Czarnecki, K. (2017). An analysis of iso 26262: using machine learning safely in automotive software. *arXiv preprint arXiv:1709.02435*

Savithry, J., Ortega, A. G., Pillai, A. S., Balbastre, P., and Crespo, A. (2019). "Design of criticality-aware scheduling for advanced driver assistance systems," in *2019 24th IEEE international conference on emerging technologies and factory automation (ETFA)* (IEEE), 1407–1410.

Sheikh, S. Z., and Pasha, M. A. (2021). Energy-efficient cache-aware scheduling on heterogeneous multicore systems. *IEEE Trans. Parallel Distributed Syst.* 33, 206–217. doi:10.1109/tpds.2021.3090587

Sinha, S., Golchin, A., Einstein, C., and West, R. (2020). "A paravirtualized android for next generation interactive automotive systems," in *Proceedings of the 21st international workshop on mobile computing systems and applications*, 50–55.

Smith, J., and Nair, R. (2005). *Virtual machines: versatile platforms for systems and processes*. Elsevier.

Sohal, P., Bechtel, M., Mancuso, R., Yun, H., and Krieger, O. (2022). "A closer look at intel resource director technology (rdt)," in *Proceedings of the 30th international conference on real-time networks and systems*, 127–139.

Solutions, B. M. (2023a). Ee architecture.

Solutions, B. M. (2023b). Whitepaper: E/e architecture.

Sriramakrishnan, G., Mody, M., Shurtz, G., Fuoco, C., Chitnis, K., Devshatwar, N., et al. (2022). "Io virtualization for real time automotive systems," in *2022 IEEE international conference on consumer electronics (ICCE)* (IEEE), 1–4.

Stolz, W., Kornhaas, R., Krause, R., and Sommer, T. (2010). Domain control units-the solution for future e/e architectures? *SAE Int.* doi:10.4271/2010-01-0686

Sugerman, J., Venkitachalam, G., and Lim, B.-H. (2001). "Virtualizing i/o devices on vmware workstation's hosted virtual machine monitor," in *Proceedings of the General Track: 2001 USENIX Annual Technical Conference* (General Track), 1–14.

Sundar Rajan, A. K., and Nirmala Devi, M. (2021). "Virtualizing an automotive state-of-the-art microcontroller: techniques and its evaluation," in *Automotive embedded systems: key technologies, innovations, and applications* (Springer), 19–36.

Times, E. (2016). Why audiâ€™s zfas is blueprint for next-gen domain architectures

Traub, M., Maier, A., and Barbehön, K. L. (2017). Future automotive architecture and the impact of it trends. *IEEE Softw.* 34, 27–32. doi:10.1109/ms.2017.69

Tuohy, S., Glavin, M., Hughes, C., Jones, E., Trivedi, M., and Kilmartin, L. (2014). Intra-vehicle networks: a review. *IEEE Trans. intelligent Transp. Syst.* 16, 534–545. doi:10.1109/tits.2014.2320605

Varanasi, P., and Heiser, G. (2011). "Hardware-supported virtualization on arm," in *Proceedings of the second asia-Pacific workshop on systems*, 1–5.

Vasu, A., and Ramaprasad, H. (2020). "Application constraints and safety aware mapping of autosar applications on multi-core platforms," in *2020 IEEE international conference on embedded software and systems (ICESS)* (IEEE), 1–10.

Vetter, A., Obergfell, P., Guissouma, H., Grimm, D., Rumez, M., and Sax, E. (2020). "Development processes in automotive service-oriented architectures," in *2020 9th Mediterranean conference on embedded computing (MECO)*, 1–7. doi:10.1109/MECO49872.2020.9134175

Wulf, C., Willig, M., and Göhringer, D. (2021). "A survey on hypervisor-based virtualization of embedded reconfigurable systems," in *2021 31st international conference on field-programmable logic and applications (FPL)* (IEEE), 249–256.

Xu, M., Gifford, R., and Phan, L. T. X. (2019). "Holistic multi-resource allocation for multicore real-time virtualization," in *Proceedings of the 56th annual design automation conference 2019*, 1–6.

Xu, M., Thi, L., Phan, X., Choi, H.-Y., and Lee, I. (2017). "vcat: dynamic cache management using cat virtualization," in 2017 IEEE real-Time and embedded Technology and applications symposium (RTAS) *(IEEE)*, 211–222.

Zeng, W., Khalid, M. A., and Chowdhury, S. (2016). In-vehicle networks outlook: achievements and challenges. *IEEE Commun. Surv. and Tutorials* 18, 1552–1571. doi:10.1109/comst.2016.2521642