Received March 22, 2021, accepted April 14, 2021, date of publication April 21, 2021, date of current version April 30, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3074813

Automotive Architecture Topologies: Analysis for Safety-Critical Autonomous Vehicle Applications

ALESSANDRO FRIGERIO[1], **BART VERMEULEN**[2], **(Member, IEEE), AND KEES G. W. GOOSSENS**[1], **(Member, IEEE)**

[1]Department of Electrical Engineering, Eindhoven University of Technology, 5612 AZ Eindhoven, The Netherlands
[2]NXP Semiconductors, 5656 AE Eindhoven, The Netherlands

Corresponding author: Alessandro Frigerio (a.frigerio@tue.nl)

This work was supported in part by the Eindhoven University of Technology (TU/e) Impuls Program, a strategic cooperation between NXP Semiconductors and the TU/e, and in part by the Pan European partnership in micro and Nano-Technologies and Applications (PENTA) under Project HIPER181004.

ABSTRACT Safety-critical systems such as Advanced Driving Assistance Systems and Autonomous Vehicles require redundancy to satisfy their safety requirements and to be classified as fail-operational. Introducing redundancy in a system with high data rates and processing requirements also has a great impact on architectural design decisions. The current self-driving vehicle prototypes do not use a standardized system architecture but base their design on existing vehicles and the available components. In this work, we provide a novel analysis framework that allows us to qualitatively and quantitatively evaluate an in-vehicle architecture topology and compare it with others. With this framework, we evaluate different variants of two common topologies: domain and zone-based architectures. Each topology is evaluated in terms of total cost, failure probability, total communication cable length, communication load distribution, and functional load distribution. We introduce redundancy in selected parts of the systems using our automated process provided in the framework, in a safety-oriented design process that enables the ISO26262 Automotive Safety Integrity Level decomposition technique. After every design step, the architecture is re-evaluated. The advantages and disadvantages of the different architecture variants are evaluated to guide the designer towards the choice of correct architecture, with a focus on the introduction of redundancy.

INDEX TERMS ADAS, ASIL decomposition, AV, functional safety, redundancy, safety-critical systems.

I. INTRODUCTION

The automotive industry is researching Autonomous Vehicles (AVs) as the next revolution for their products. AVs and Advanced Driver Assistance Systems (ADAS) have a large number of requirements, related to performance, safety, and costs. These requirements impact the design choices related to the system architecture. Safety requirements lead to the necessity of redundant and backup systems. Redundant elements require more complex networks and architecture decisions that impact the final cost of the vehicle. Moreover, introducing redundancy affects different architecture topologies differently.

Fig. 1 shows a domain-based architecture topology, in which the functionalities are divided into 6 domains as in [1]. The domain distribution is not standardized between OEMs. For a more complete description of different domains and automotive functionalities, we refer to [2]. Similarly, also the architecture topologies are not standardized. In this work, we analyse domain-based and zone-based topologies.

We focus on the safety-oriented design process behind fail-operational automotive systems, which guarantee full operation of a function even in presence of a fault. To achieve this, we introduce redundancy in the functionality and

The associate editor coordinating the review of this manuscript and approving it for publication was Lorenzo Ciani.

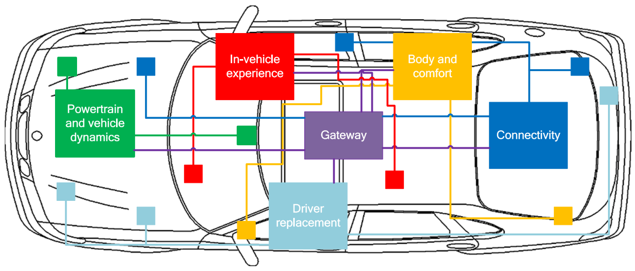


FIGURE 1. A domain based architecture topology [1].

hardware resources and we quantitatively evaluate the resulting architecture.

The two main contributions of this work are:

- *We extend our safety analysis framework [3], [4] by adding additional parameters that can be calculated on a specific set of applications and hardware, allowing for a more complete evaluation of an automotive design. Moreover, we improve our model transformation tool that is used to introduce redundancy in a described system with effects on application and hardware resources layers*. The evaluation consists of a calculation of the following five parameters: cost of the hardware resources, failure probability of safety-critical applications, total communication cable length, total communication load, and total functional load. The cost metric used in this work uses the Automotive Safety Integrity Level (ASIL) of the resources. The failure probability is calculated by a quantitative fault tree analysis. The total communication cable length, which strongly impacts the total weight of the vehicle, is calculated as the Euclidean and Manhattan distances between the two resources that are connected by each communication resource, in two dimensions in the physical space of the vehicles. We do not model power lines. The functional load and communication load distributions are calculated by assigning a load parameter to each application node and observing their mapping on the hardware resources. Introducing redundancy is an automated process that starts by selecting an application node that requires redundancy and automatically modifies the other application nodes and the hardware resource layer accordingly.
- *The analysis of the same three illustrative applications mapped to four different architecture topologies in one non-redundant and two redundant scenarios.* These applications have typical ADAS and AV characteristics, such as high functional and communicational requirements. We select two different application nodes in separate experiments to become redundant, and the framework automates the process that modifies the affected application nodes and the hardware resources accordingly. The ISO26262 standard's ASIL decomposition technique can be applied to the resulting redundant architecture to decompose the original safety requirements into less critical safety requirements. Each redundancy scenario is evaluated by its impact in terms of the five parameters on the resulting architecture.

The performed evaluation helps during an initial design phase in which the architecture topology is chosen, and can be reused during more advanced phases when accurate application details will result in exact quantification of the evaluation parameters or introducing redundancy in the system is necessary.

The rest of this paper is organized as follows: Section II describes the related work. Section III describes the architecture topologies that we analyse in this work. Section IV describes the model and the process used to introduce redundancy in the architecture. Section V describes the application and hardware examples used for the experiments. In Section VI we show the results of the evaluations, followed by a final discussion in Section VII and conclusions in Section VIII.

II. RELATED WORK

The topics of ADAS and AV are widely studied from many points of view. In this work, we focus on the functional safety aspects, mostly described by the *ISO2626 Road Vehicles - Functional Safety* [5] in terms of electronic hardware system reliability during the vehicle lifecycle. The ISO26262 standard presents an ASIL decomposition technique to reduce the safety requirement of parts of the system into redundant components. The authors of [6] improve the technique by adding additional checking elements to prove that the original Functional Safety Requirements (FSRs) are met. These additional checks can be found also in [3] and [4], where we define *splitter* and *merger* functionality to manage the redundant parts of the application. A *splitter* replicates its input to multiple output ports that are connected to redundant parts of the application; a *merger* decides which of its inputs, connected to the redundant parts of the application, should be forwarded on its output port. In this paper we utilize these two definitions and extend them to hardware resources as well: the resources with the *splitter* or *merger* functionality will have the FSR of ensuring correct behaviour of the redundant parts of the system.

The ASIL decomposition technique is widely studied, for example, the authors in [7] provide tools for the automatic assignment of decomposed ASIL values to a particular set of hardware resources. Since the standard provides only guidelines to implement the decomposition, the technique can be misinterpreted, as explained in [8] and [9].

More safety-oriented automotive standards exist, for example, the *ISO/PAS 21448 Road Vehicles - Safety of the intended functionality* focusses on the intended functionality of an autonomous vehicle or an ADAS, in which situational awareness is critical to safety. Another example is the *ANSI/UL 4600 Standard for Safety for the Evaluation of Autonomous Products*, which focuses on the safety of autonomous systems such as self-driving cars, addressing changes required from traditional safety practices to accommodate autonomy. However, in this work, we will focus on ASIL-based safety analysis, and we will not discuss further implications of these standards.

Architecture patterns for functional safety are analysed in [10] and [11], where specific solutions are proposed for automated driving functions. Each of them involves a certain level of redundancy and can be implemented with different topologies. The authors in [12] present a centralized topology for the RACE project, while [13] present the distributed architecture used in the prototype vehicle that won the 2012 AVC in Korea. The authors in [14] mention domain-based and zone-based architectures and the advantages of Ethernet networks in future automotive systems. However, no previous literature compares different possible architecture topologies such as Domain-based versus Zone-based from a safety-oriented system-level perspective as we do in this work.

III. AUTOMOTIVE ARCHITECTURE TOPOLOGIES

In this work, we define an architecture topology as a combination of a category and a mapping. We consider two possible architecture categories: *Domain-Based (D)* and *Zone-Based (Z)*. In each scenario, the sensors and actuators have a fixed position related to their functionality (e.g. front radar), while the rest of the hardware resources can be positioned freely in the vehicle. The two categories follow these rules to connect the sensors and actuators to the rest of the system:

- **Domain-Based (D)** groups system components according to their functionality. Each domain has a domain controller. The sensors and actuators of a specific domain are connected to the corresponding domain controller either with a direct connection or by using a domain network.
- **Zone-Based (Z)** groups system components according to their physical position in the vehicle. A number of zones have to be selected and a zone controller is positioned in every zone. Each sensor or actuator is connected to the nearest zone controller either with a direct connection or a zone network.

In both categories, the controllers are then connected to an optional central unit and/or to the other controllers via a backbone network. For each of the two categories, we consider two possible mapping rules of the application to the hardware resources:

- **Vehicle-Centralized (VC)**: all the computational nodes are mapped to the central unit. The sensors feed their data directly to the central unit, which directly feeds the actuators. The domain or zone controllers perform only networking functions between the domain or zone network and the central unit.
- **Controller-Based (CB)**: when possible, the functions are mapped to the controllers in the domain or zone, which now require computational capabilities. The central unit performs only the tasks that require data from or provide data to multiple domains or zones and thus are executed centrally.

The combination of the two categories and the two mapping rules gives us the four architecture topologies that we will analyse in this work: *D-VC*, *D-CB*, *Z-VC*, *Z-CB*.

We always assume that no inter-domain or inter-zone communication is allowed. When inputs from or output to multiple domains or zones are necessary, a function must be executed in the central unit, which communicates with all the domains or zones via the backbone network.

While more architecture topologies are possible, the four selected in this paper correspond to the far ends of the possible architectures spectrum, allowing us to analyse the extreme scenarios [15]. It is not yet clear to the industry which topology will prevail, but with the proposed framework we can analyse any different architecture topology. Our goal is to quantitatively evaluate these cases to be able to guide the architectural choices.

IV. PROPOSED METHODOLOGY

A. SYSTEM MODEL

The system is modeled by a three-layer model: *application*, *resources*, and *physical* layers. Each layer is described by a graph: $G_a = (V_a, E_a)$ contains the application nodes and the logical connection between them, $G_r = (V_r, E_r)$ contains the hardware resources and their connections, and $G_p = (V_p, E_p)$ contains the physical locations in which resources can be placed and their connections. $E_{ar} : V_a \rightarrow V_r$ and $E_{rp} : V_r \rightarrow V_p$ are the mapping edges that complete the system description with the relationships between the three layers. The model is based on [3], and we reuse our definitions of splitter and merger nodes, mentioned in Section II, to describe redundancy patterns.

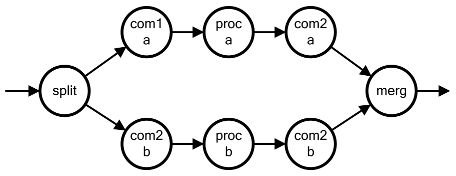


FIGURE 2. Redundant pattern in the application layer.

TABLE 1. Failure rates metric (failures/hour) [4].

Resource Type	QM	A	B	C	D
Splitter or Merger	10e-6	10e-7	10e-8	10e-9	10e-10
Other	10e-5	10e-6	10e-7	10e-8	10e-9

Fig. 2 shows a redundancy pattern in the application layer. The node *proc* is implemented in two redundant ways, *proc a* and *proc b*, which receive their data from a splitter. The outputs are analysed by the merger to choose which data path to forward. This representation can describe multiple ways of implementing redundant applications: *proc a* and *proc b* could be implemented with diverse or same functionality, *proc a* could implement the nominal functionality and *proc b* a reduced set of operations, etc.

To perform our quantitative evaluation, we extend the model of [3] (which considered only failure probability, cost, and cable length related to an application) by adding the functional load and the communication load to the application nodes, and the two-dimensional coordinates to the physical nodes. To calculate the failure probability and the cost of the system we use the algorithms presented in [4]. Fault tree analysis is used to calculate the system failure probability, where the failure rates of the hardware resources are related to their ASIL specification on a logarithmic scale, as seen in Table 1. The failure rate values we use for the experiments are in line with the system failure rates requirements based on its ASIL defined in the ISO 26262 standard.

The cost metric we use is shown in Table 2. It is similar to the scale used in [16], but we group the ASIL values A-B and C-D since the safety practices that an organization must perform according to the ISO26262 standard for these pairs are similar, and will therefore likely have similar development costs.

B. INTRODUCING REDUNDANCY IN THE SYSTEM MODEL

Redundancy is a necessary technique to develop fail-operational systems. Our framework supports the system designer to analyse different levels of redundancy by

TABLE 2. Resources cost metric.

Resource Type	QM	A	B	C	D
Functional	5	500	500	50000	50000
Communication	4	400	400	40000	40000
Sensor / Actuator	8	800	800	80000	80000
Splitter / Merger	1	100	100	10000	10000

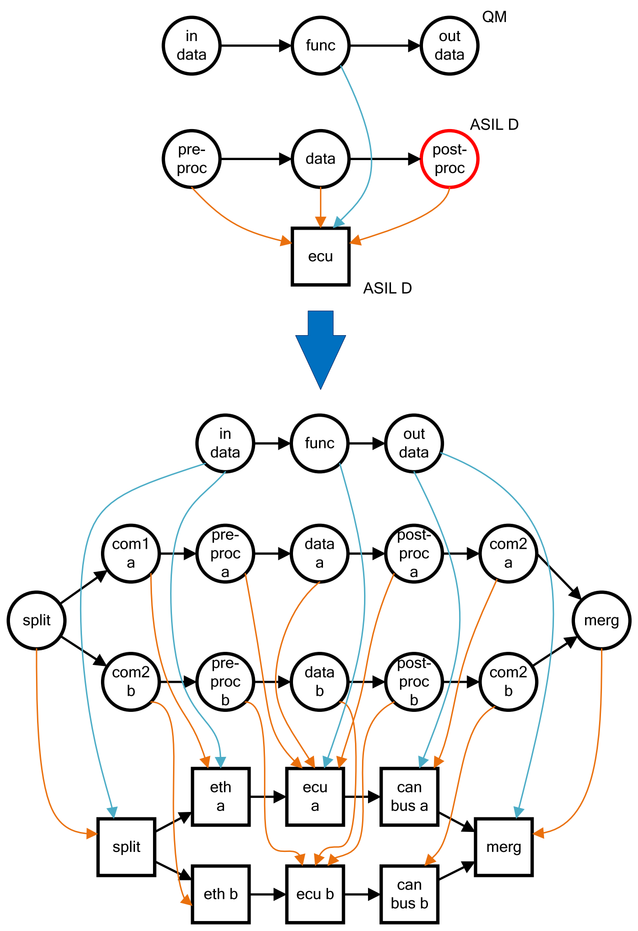


FIGURE 3. Transformation applied on the functional application node *post-proc* to introduce redundancy.

applying a component-based ASIL decomposition technique on selected application nodes. In [4] we define the rules for the transformation and its effect on the application layer, obtaining from a single application node a pattern as shown in Fig. 2. We extend this work by adding transformation rules to modify the resource layer, allowing us to remap the affected application nodes to matching resources. The obtained resource layer will have independent resources to independently remap redundant branches of the application, validating the ASIL decomposition.

Similarly to the application layer, we can identify redundancy patterns in the resource layer with splitter and merger resources, either realised by dedicated units or as part of a multi-purpose resource. Fig. 3 shows the results of the transformation performed after selecting the node *post-proc* to become redundant. We define ASIL(application node) as the ASIL requirement of an application node, and ASIL(resource) as the ASIL specification of a resource. The transformation steps are the following:

1) The node *post-proc* is transformed (as in Fig. 2).

2) The transformed node was mapped on the resource *ecu*. A modification of the resource layer may be required depending on ASIL(post-proc) and ASIL(ecu). In particular, we assume that the resource layer does not require any modification if ASIL(post-proc) $<$ ASIL(ecu). This means that the resulting redundant branches can be remapped to the original resource since it has a high enough ASIL specification to justify the independence of the redundant elements. In case ASIL(post-proc) $\geq$ ASIL(ecu), the resource *ecu* is transformed as shown in Fig. 3. It is theoretically possible to map the redundant application nodes to the original resource when ASIL(post-proc) $=$ ASIL(ecu). However, we decide to transform the resource layer anyway for our experiments. This decision is taken because the input graphs that are used in the experiments are valid, meaning that the ASIL value of an application node cannot exceed the ASIL specification of the resource on which it is mapped. This means that in the input descriptions that are used in Section VI the safety-critical resources always have the highest ASIL specification, while in reality that might not be the case: ASIL D ready resources might not be available for a specific project due to their higher cost or unavailability. By introducing redundancy, we shift the higher safety requirements to the parts that will perform the splitter and merger functionalities, while the redundant application can be processed by less safety-critical resources.

3) If the resource layer is modified the other nodes that were mapped on the original resource must be remapped to the new resources resulting from the transformation. In Fig. 3, the nodes *pre-proc* and *data* are mapped on *ecu*, which is being transformed. Since they are connected to the node that is transformed, they must follow its same transformation. For the other nodes that are mapped on the resource *ecu*, if ASIL(node) $\leq$ ASIL(ecu a) or ASIL(node) $\leq$ ASIL(ecu b) the node can be remapped to only one side of the redundant pattern in the resource layers. In any other case it must follow the same transformation that the node *post-proc* had. The node *func* in the figure has QM requirements, meaning that it can be mapped to only one side, in the example to *ecu a*. In order for the input and output data of node *func* to reach *ecu a*, they must pass through *split*, *eth a*, *can bus a*, and *merg* as shown in the figure, and finally mapped to the already existing external communication resources that are not shown in the figure.

4) The redundant branches of consecutive redundant application nodes can be connected when the ASIL specifications allow it. In Fig. 3 the branches *a* and *b* are connected together, removing the additional intermediate splitter, merger, and communication nodes.

5) Last, the resulting application nodes are remapped to the new resource layers (orange arrows in Fig. 3).

In our example, the nodes *pre-proc* and *data* were part of the same application of the node that is being selected for its transformation, but also nodes belonging to different

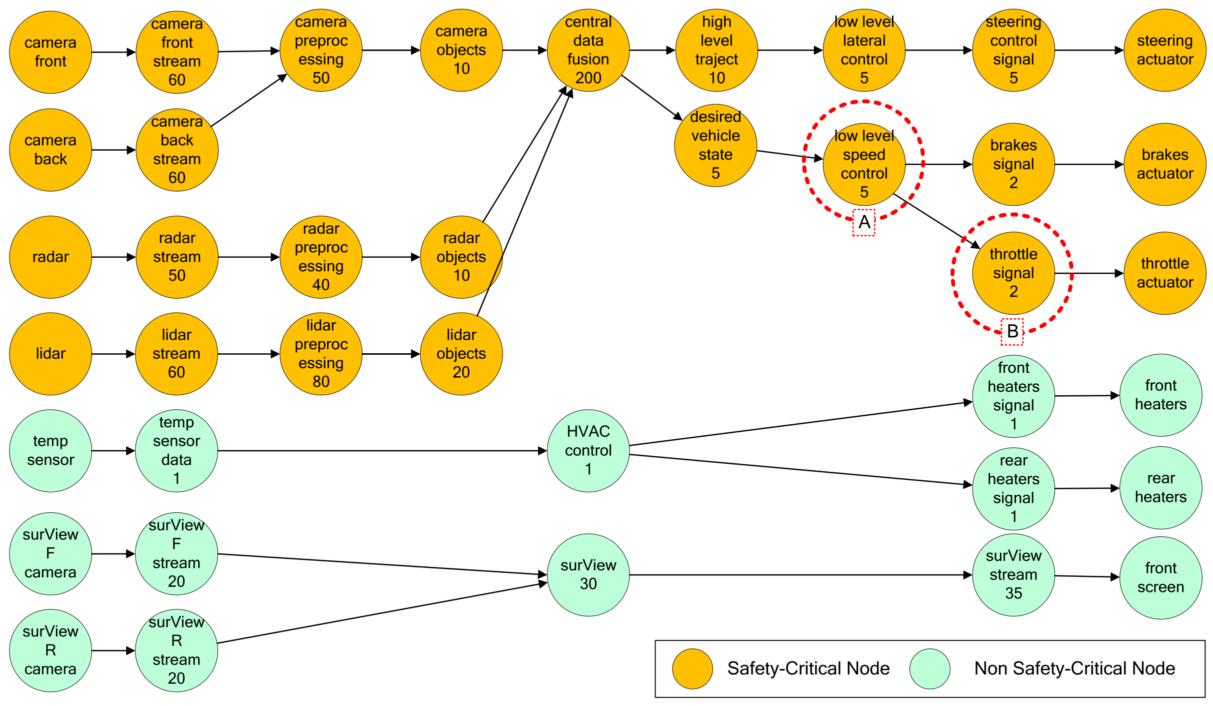


FIGURE 4. The set of applications used for the topologies evaluation.

applications and with different ASIL specifications may be affected by the modification of the resource layer and will follow the same rules.

The new resources are positioned in the physical locations based on their proximity with the neighbour resources. For valid ASIL decomposition, the decomposed elements require independence. In our experiments, we implement this with physical separation of the redundant resources.

V. EXAMPLE SETUP

A. SOFTWARE APPLICATIONS

We define three illustrative applications that we use for our experiments. Each application is described by a separate graph in Fig. 4, showing their functional and communication nodes and their logical connections. For the simpler non-safety-critical application we use the common Sense-Think-Act paradigm [17]. For the safety-critical application, we extend it, dividing the Think block into Preprocessing, Data Fusion, and Postprocessing steps. This enables a realistic function mapping over multiple components. Nodes A and B are selected for redundancy for the experiments of Sections VI-B and VI-C respectively.

We divided the applications into a safety-critical one, in orange, and two non-safety-critical ones, in green. The safety-critical application contains typical operations that a self-driving vehicle performs, such as environmental modeling and vehicle control, while the non-safety-critical applications provide additional information with a surround-view application that shows the back and front camera views to the vehicle passengers, and a comfort application that interacts with the heat, ventilation, and air conditioning system. We assume that only the central unit can meet high computational requirements in our experiments, so that the centralized part of the safety-critical application, which has high computational requirements, is always executed in the central unit. Depending on the actual resources that are used, also the controllers can satisfy high computational application requirements.

The applications have high bandwidth requirements on the sensing side (mostly in the safety-critical application), high computational requirements in the central data fusion node, low bandwidth and computational requirements in the post-processing and actuation sides.

For qualitative analysis, we annotate the functional and communication loads on each node to reflect these requirements, as shown in Fig. 4. In our example, we use realistic proportions between the nodes for the sake of a final comparison and analysis of the load distributions over the different architecture topologies, while real applications details are generally confidential.

Moreover, we assume that the lidar and the radar are *smart sensors* that locally convert raw data into output objects. This means that the pre-processing part of these sensors will be mapped on the sensor resource itself.

B. HARDWARE RESOURCES

We use a representative non-redundant hardware architecture formed by two domains (or zones) and a central unit. In the domain-based architectures, one domain contains all the safety-critical resources, sensors, and actuators, while the other domain contains the non-safety-critical resources, sensors, and actuators. In the zone-based architectures, the two

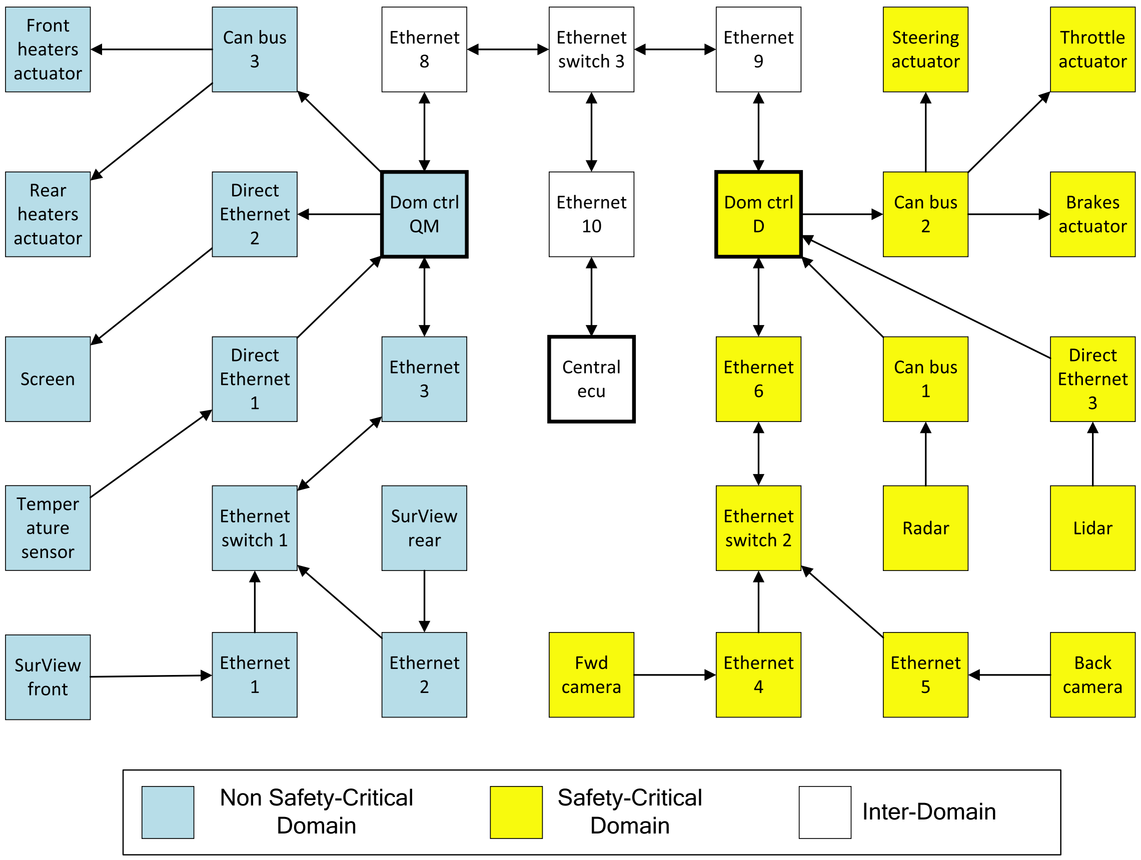


FIGURE 5. Domain-based hardware resources.

zones divide the vehicle into a front and a rear part, and sensors and actuators are connected to the zone controllers based on their position inside the vehicle. Fig. 5 shows the chosen architecture for domain-based categories.

The domains are connected to the central unit via a star switched-Ethernet network, while the domain networks are a combination of buses, switched-Ethernet networks, and direct connections.

Each sensor and actuator has a fixed position in the physical space, while other resources can be placed with some freedom inside the vehicle. The central unit is placed in a central position for the vehicle. The domain controllers are placed in a central position for the domain's sensors and actuators, to minimize the total communication cable length between them. The zone controllers are placed in a central position in the respective zone, again for cable length optimization.

The initial resource layer for the topologies *D-VC* and *D-CB* is identical, while there will be differences in the redundant scenario due to the different mapping of the application. In the same way, the zone-based topologies resource layers will show differences only in the redundant scenarios.

Variations are possible: the backbone network can be a ring network, the internal networks and the number of sensors and actuators can vary, multiple controllers can be used, different topologies can be mixed, etc. Our framework can analyse these, but in this paper, we focus on comparing the four architecture topologies, described in Section III.

VI. EXPERIMENTS

Each architecture topology from Section I is analysed in terms of resources cost, safety-critical application failure probability, total communication cable length, and functional and communication load distributions. As an example, in each topology, we then apply in separate experiments the ASIL decomposition of two specific nodes: the Low-Level Speed Control (LLSC) and the Throttle Signal (TS) nodes. By following the transformation rules described in Section IV-B we obtain a new redundant system that can be compared with the original solution.

A. ANALYSIS OF NON-REDUNDANT ARCHITECTURES

First, we analyse the topologies without introducing redundancy in the system. The three applications of Fig. 4 are mapped to the different architecture topologies and each solution is analysed.

We observe in Fig. 6a that the cost of the domain-based topologies is lower than of the zone-based ones, assuming the cost metric of Table 2. Since the zones have mixed-critical application nodes (both ASIL D and QM), the zone controllers and the dedicated units need to be more

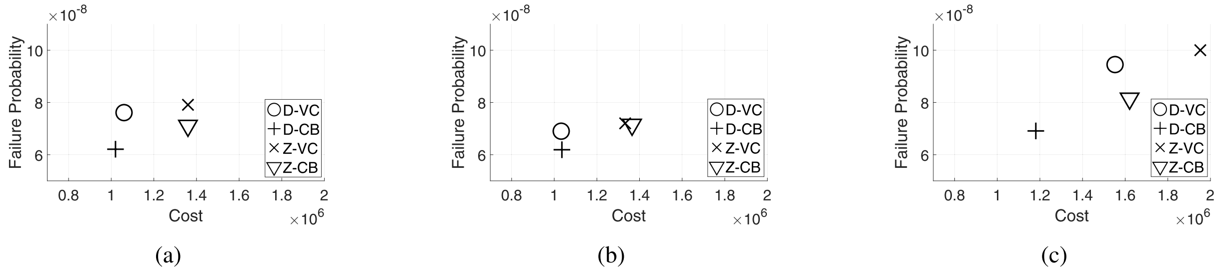


FIGURE 6. Failure probability vs Cost for non-redundant (a), redundant LLSC node (b), redundant TS node (c).

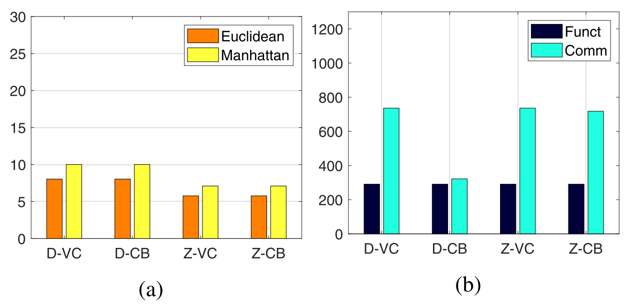


FIGURE 7. Total communication cable length (a) and total communication and functional loads (b) for non-redundant scenario.

expensive ASIL-D ready resources to satisfy the safety-critical requirements. The difference in failure probabilities between domain-based and zone-based topologies is related to how the internal networks are configured (in our case the Z categories have one less communication resource), while VC and CB mappings differ in terms of communication paths and the number of utilized resources. For example, in the D-CB topology, the non-safety-critical part of the application does not reach the *Central ecu* resource, and the *Ethernet 8* resource is not used.

Fig. 7a shows the calculated total communication cable length. It is lower in the zone-based topologies, which is expected since they connect sensors and actuators to the closest controllers. Since the resource layer for VC and CB mappings are identical, there are no differences in total communication cable length between them.

Fig. 7b shows the total functional and communication loads of the topologies. In the non-redundant scenarios, the total functional load does not vary as it is related only to the applications, but it is distributed differently over the architecture as we will observe in Section VII. The communication load varies instead based on the topology. The D-CB topology has the lowest communication load since all the processing steps are executed in the domain controller and the sensor or actuator data is sent only through the local domain network and not through the backbone network. The output data size of the post-processing step, in particular, is highly reduced compared to the initial input raw sensor data. VC mappings instead require all the processing steps to be done in the central unit, which means that the raw sensor data has to be transmitted not only inside the domain or zone network but also through the backbone network. The Z-CB topology instead can perform only part of the functionality locally: since we assume no inter-communication between the zones is possible, most tasks, which require inputs from or send outputs to multiple zones, have to be executed in the central unit.

By combining the results of Fig. 7a and 7b, we observe that a) the D-VC topology has longer cable lengths with higher bandwidth requirements, b) the D-CB topology has longer cable lengths with lower bandwidth requirements, and c) zone-based topologies have shorter cable lengths with higher bandwidth requirements.

B. EXPANSION OF THE LOW-LEVEL SPEED CONTROL

Next, we apply the node transformation described in Section IV-B to the LLSC node, marked with the dashed circle A in Fig. 4. This allows us to observe how introducing redundancy in the low-level control part of the safety-critical application impacts the system. Low-level control nodes are the ones that interact directly with the vehicle dynamics, being able to provide signals to the actuators. They are a very critical part of the system, despite being less computationally intensive than higher-level functions. The redundancy transformation affects different parts of the system based on its topology since the LLSC node's mapping varies. Fig. 8 shows the effects of the transformations on the application layer for the D-VC and the Z-VC topologies after following the transformations rules of Section IV-B. The preprocessing, data fusion, and post-processing parts of the safety-critical application are all mapped to the central unit, which is being duplicated because of the redundancy in the LLSC node. All these parts become redundant as well, obtaining the redundant branches *a* and *b* in Fig. 8. The non-safety-critical applications are not affected by the transformation since they can be mapped on one of the two redundant central ECUs because of their lower ASIL requirement. In the D-CB and Z-CB architecture topologies, the transformation modifies a smaller part of the application, since fewer nodes are mapped to the same controllers as the LLSC node.

We observe that the architectures with redundant LLSC have similar cost values compared to non-redundant ones. While non-redundant architectures must use ASIL-D ready resources, redundant architectures can use ASIL-D ready splitters and mergers in combination with parallel lower-level

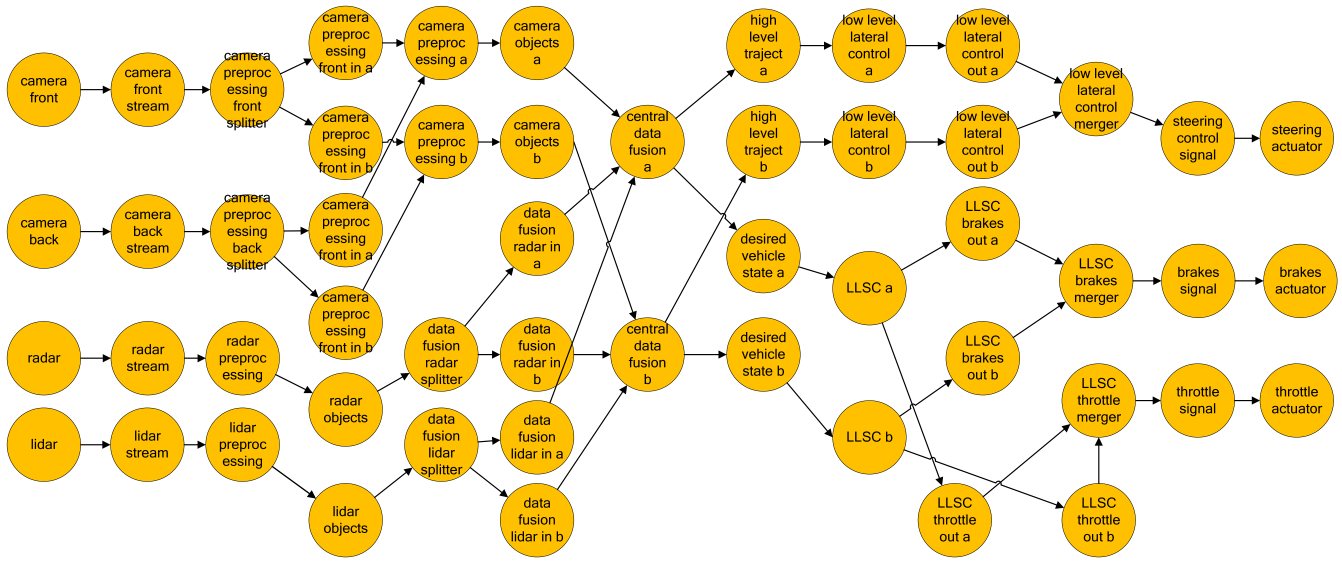


FIGURE 8. Safety-critical application in the *D-VC* and *Z-VC* topologies: application layer after the transformation of the LLSC node (Section VI-B).

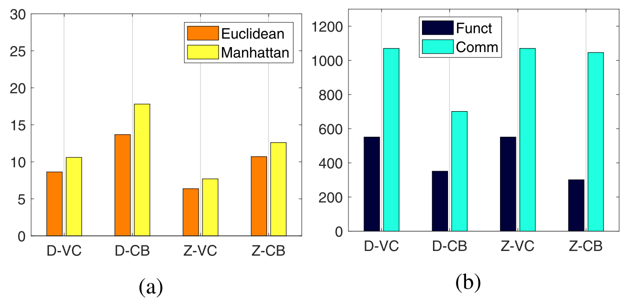


FIGURE 9. Total communication cable length (a) and total communication and functional loads (b) for redundant LLSC node scenario.

resources, e.g. two ASIL-B ready ones. The cost of an ASIL D splitter or merger combined with ASIL B functional and communication resources is similar to the cost of the original ASIL D resource when following Table 2, but can vary with the cost metric that is selected. The failure probability is lower for VC mappings, as shown in Fig. 6b since more nodes are mapped on the now redundant central ECU and can benefit from the lower failure rates of the safety-oriented splitter and merger resources. In CB mappings instead, this effect is hidden by the more complex communication interfaces that are connected to the now redundant controllers. The central unit only has an Ethernet port, meaning that only one splitter and one merger resource will be required. The controllers are connected instead to multiple network components, and on each port, a splitter and/or a merger resource is required. For example, from Fig. 5, the domain controller *Dom ctrl D* will have with the D-CB topology in LLSC-redundant scenarios four splitters (one for each input port) and three mergers (one for each output port).

When analysing the total communication cable length in this redundant scenario, we notice how the CB mappings have a greater impact on this parameter, shown in Fig. 9a.

Redundant controllers lead to an increase in the number of communication resources since the controllers are connected to both the local and the backbone networks.

Fig. 9b shows the total functional and communication loads after introducing redundant hardware for the redundant LLSC nodes. The functional load varies between the topologies because of a different number of application nodes mapped to the original resource. The VC mappings lead to a high communication load when redundancy is introduced in the LLSC node (or in any other processing node, since they are all mapped to the central ecu). In the case of the Z-CB topologies, the expansion of the resources does not involve additional functional node transformations, since the LLSC is the only node mapped to the zone controller while the other nodes are mapped to the central unit.

We observe greater differences between the different mappings compared to the non-redundant topologies. The D-CB topology is not strictly better than the D-VC one with its slightly higher cost, while the zone-based topologies have lower total communication cable length, with increasing requirements for communication load though.

C. EXPANSION OF THE THROTTLE SIGNAL

Our third experiment consists of applying the transformation to the communication node TS, as marked with the dashed circle B in Fig. 4. It is a low-level signal with a small communication load but with critical importance since it controls the throttle actuator. Depending on the topology it is mapped to different communication resources and more than one resource is affected by the transformation of the node. In CB mappings the signal comes from the domain or zone controller, while in VC mappings it comes from the central unit through the backbone and the domain or zone network.

Fig. 6c shows the total cost and failure probabilities of the topologies for a redundant TS node. The VC mappings have

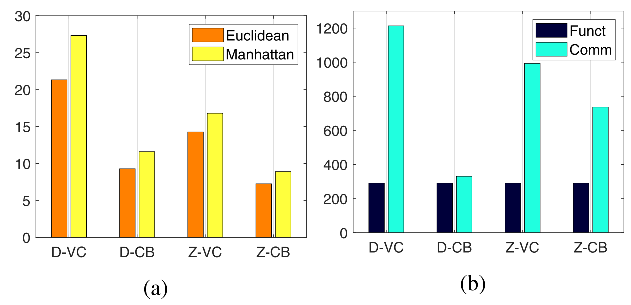


FIGURE 10. **Total communication cable length (a) and total communication and functional loads (b) for redundant TS scenario.**

a significantly higher cost and failure probability compared to the non-redundant scenario: the cost increases by 46% and 43% while the failure probability increases by 24% and 26% for the D-VC and the Z-VC topology respectively. This effect is due to the transformation of all the communication resources that carry the throttle signal, which in VC mappings are both parts of the backbone and the domain or zone network. The CB mappings cost and failure probability also increase, despite the control signals being transmitted locally, but to a lower degree since fewer communication resources are involved.

As shown in Fig. 10a, the zone-based topologies have a lower total communication cable length, and the CB mappings result in lower values compared to the other mappings. This happens because only part of a local network becomes redundant, which is reflected in Fig. 10b in the form of a lower communication load.

The functional load is constant since only communication resources are expanded. The communication load of the TS node is low, but when making it redundant most of the communication nodes that are mapped on the same resources are consequently transformed following the rules of Section IV-B. In the case of the VC mappings and the zone-based topologies, higher-level communication data is mapped on these resources, such as raw camera streams or lidar and radar detected objects, with high communication loads. The transformation of a single low-level control signal leads to the modification of many parts of the system.

VII. DISCUSSION

We observed in the previous experiments how redundancy applied in two different application nodes impacts the system properties. We observed how the D-CB topology has better loads and costs results than the other topologies while being surpassed by some zone-based topologies in terms of failure probabilities and total communication cable length.

Fig. 11a shows the distribution of the functional load over the computational resources.

Note that the different topologies have different requirements: the VC mappings require only the central unit to process the data, while the CB mappings require the central unit to process the data fusion part of the application, but also require the controllers to have computational power for pre

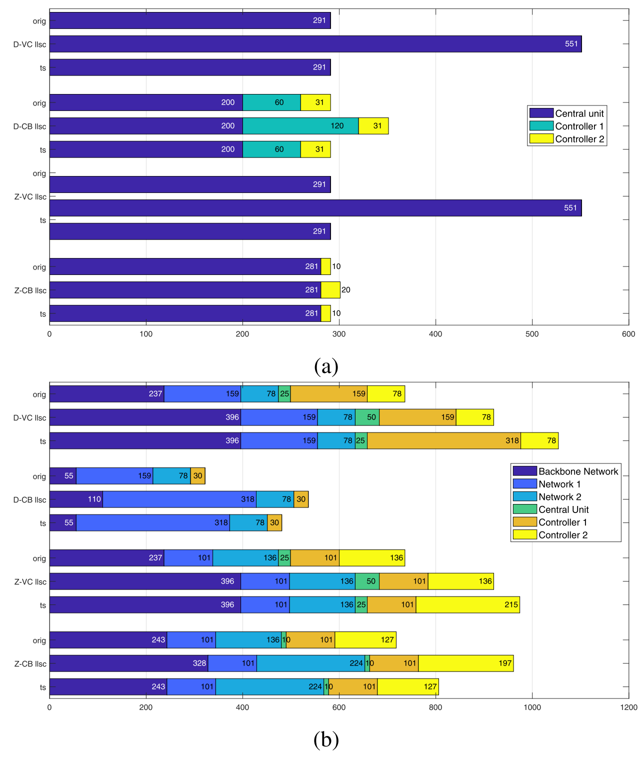
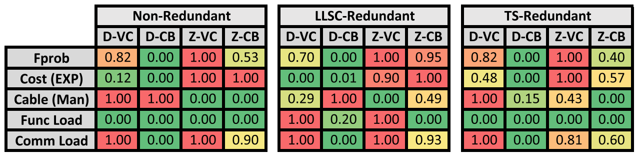


FIGURE 11. **Total functional load (a) and total communication load (b) distributions.**



	Non-Redundant				LLSC-Redundant				TS-Redundant			
	D-VC	D-CB	Z-VC	Z-CB	D-VC	D-CB	Z-VC	Z-CB	D-VC	D-CB	Z-VC	Z-CB
Fprob	0.82	0.00	1.00	0.53	0.70	0.00	1.00	0.95	0.82	0.00	1.00	0.40
Cost (EXP)	0.12	0.00	1.00	1.00	0.00	0.01	0.90	1.00	0.48	0.00	1.00	0.57
Cable (Man)	1.00	1.00	0.00	0.00	0.29	1.00	0.00	0.49	1.00	0.15	0.43	0.00
Func Load	0.00	0.00	0.00	0.00	1.00	0.20	1.00	0.00	0.00	0.00	0.00	0.00
Comm Load	1.00	0.00	1.00	0.90	1.00	0.00	1.00	0.93	1.00	0.00	0.81	0.60

FIGURE 12. **Final comparison tables with normalized parameters over all topologies, separately for each redundancy scenario.**

and post-processing. Fig. 11b shows the communication load distributions in the topologies. The zone-based topologies show more balanced communication loads between the zones (*Network 1* and *Network 2*), as a result of having sensors from a specific domain, for example, the front and back cameras, distributed over the car. In this case, a significant part of the total communication load is placed in the backbone network, since most computation is performed centrally and a great part of the data generated inside a zone is sent via the backbone network to the central unit.

A final comparison between the topologies in the different redundancy scenarios is shown in Fig. 12.

The parameters for each scenario are normalized over the topologies, where a 0.00 corresponds to the lowest parameter value across all topologies and a 1.00 corresponds to the highest (for each analysis parameter, the lower the better). The zone-based topologies show a lower cable length. The good results in terms of the functional load of the Z-CB topology for the LLSC-Redundant scenario are related to the isolation of the LLSC node to the front zone controller, which is a result of the specific configuration and not a general characteristic of these topologies. To conclude, when the application requires redundancy, in the presence of high communication loads (ADAS), and the controllers provide enough computational power to execute the application nodes a CB mapping is highly recommended. Instead, sharing of centralized processing should be considered in the presence of low communication load requirements and when no functional redundancy is necessary.

VIII. CONCLUSION

We presented an analysis of domain and zone-based automotive architecture topologies, with vehicle-centralized or controller-based mappings. The automotive system is modeled with a three-layer model that consists of application, resources, and physical layers. The developed framework allows a system designer to introduce redundancy in selected nodes of the system, with an automated procedure that follows the ISO26262 ASIL decomposition guidelines. An architecture can be analysed in our framework in terms of cost, failure probability, total communication cable length, and communication and functional loads. Our results show how introducing redundancy impacts the chosen architecture topology. The results of this paper are obtained by analysing three illustrative applications. By varying its annotation in terms of functional or communication load of each node, cost, or failure rate parameters the absolute numbers would change and the new inputs can and should be re-evaluated with the proposed framework. When redundancy is required, the domain-based controller-based (D-CB) topology offers the best balance between the analysed parameters. With lower communication and functional loads, the zone-based topologies have lower total communication cable lengths. We expect hybrid solutions to appear in the future with zones for some applications (e.g. body and comfort functions) and separate domains with their isolated domain controller for others (e.g. for safety-critical ADAS).

REFERENCES

[1] L. Reger, "TThe EE architecture for autonomous driving a domain-based approach," *ATZelektronik Worldwide*, vol. 12, pp. 16–21, Dec. 2017.

[2] J. Bach, S. Otten, and E. Sax, "A taxonomy and systematic approach for automotive system architectures-from functional chains to functional networks," in *Proc. VEHITS*, 2017, pp. 90–101.

[3] A. Frigerio, B. Vermeulen, and K. Goossens, "A generic method for a bottom-up ASIL decomposition," in *Proc. SAFECOMP*, 2018, pp. 12–26.

[4] A. Frigerio, B. Vermeulen, and K. Goossens, "Component-level asil decomposition for automotive architectures," in *Proc. SSIV*, 2019, pp. 62–69.

[5] *Road Vehicles—Functional Safety*. Standard ISO 26262-2018, 2018.

[6] C. Lidström, C. Bondesson, M. Nyberg, and J. Westman, *Improved pattern for ISO 26262 ASIL Decomposition With Dependent Requirements*, document QRS-C, 2019.

[7] Y. Papadopoulos, M. Walker, M.-O. Reiser, M. Weber, D. Chen, M. Törngren, D. Servat, A. Abele, F. Stappert, and H. Lönn, "Automatic allocation of safety integrity levels," in *Proc. Workshop CAARS*, 2010, pp. 7–10.

[8] J. G. D'Ambrosio and R. Debouk, "ASIL decomposition: The good, the bad, and the ugly," SAE, Warrendale, PA, USA, SAE Tech. Paper 2013-01-0195, 2013.

[9] D. D. Ward and S. E. Crozier, "The uses and abuses of ASIL decomposition in ISO 26262," in *Proc. IET ICSS*, Oct. 2012, pp. 1–6.

[10] Y. Luo, A. K. Saberi, T. Bijlsma, J. J. Lukkien, and M. van den Brand, "An architecture pattern for safety critical automated driving applications: Design and analysis," in *Proc. SysCon*, Apr. 2017, pp. 1–7.

[11] A. Kohn, R. Schneider, A. vilela, A. Roger, and U. Dannebaum, "Architectural concepts for fail-operational automotive systems," SAE, Warrendale, PA, USA, SAE Tech. Paper 2016-01-0131, 2016.

[12] S. Sommer, A. Camek, K. Becker, C. Buckl, A. Zirkler, L. Fiege, M. Armbruster, G. Spielberg, and A. Knoll, "RACE: A centralized platform computer based architecture for automotive applications," in *Proc. IEVC*, 2013, pp. 1–6.

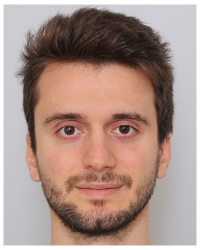
[13] K. Jo, J. Kim, D. Kim, C. Jang, and M. Sunwoo, "Development of autonomous car—Part II: A case study on the implementation of an autonomous driving system based on distributed architecture," *IEEE Trans. Ind. Electron.*, vol. 62, no. 8, pp. 5119–5132, Aug. 2015.

[14] S. Brunner, J. Roder, M. Kucera, and T. Waas, "Automotive E/E-architecture enhancements by usage of Ethernet TSN," in *Proc. ISES*, 2017, pp. 1–8.

[15] GuardKnox. (2020). *Zonal Architecture: The Foundation for Next-Generation Vehicles*. Accessed: Feb. 2, 2021.[Online]. Available: https://learn.guardknox.com/zonal-architecturethe-foundation-for-next-generation-vehicles

[16] A. Murashkin, L. Silva Azevedo, J. Guo, E. Zulkoski, J. H. Liang, K. Czarnecki, and D. Parker, "Automated decomposition and allocation of automotive safety integrity levels using exact solvers," *SAE Int. J. Passenger Cars-Electron. Electr. Syst.*, vol. 8, no. 1, pp. 70–78, Apr. 2015.

[17] M. Siegel, "The sense-think-act paradigm revisited," in *Proc. ROSE*, 2003, p. 5.



ALESSANDRO FRIGERIO received the B.Sc. and M.Sc. degrees in electronic engineering from the University of Pisa, in 2013 and 2016, respectively. He is currently pursuing the Ph.D. degree with the Eindhoven University of Technology.

His research interests include functional safety, autonomous vehicles, and E/E vehicle architectures.

BART VERMEULEN (Member, IEEE) received the M.Sc. and Ph.D. degrees from the Eindhoven University of Technology, in 1997 and 2013, respectively. He joined the Philips Research Laboratories, Eindhoven, the Netherlands, in 1997, to work on the manufacturing test and hardware/software debug of digital systems-on-chip. He is currently a Technical Director with NXP Semiconductors, Eindhoven, investigating the trends in and requirements for in-vehicle networks and architectures. He published one book, more than 40 articles, and over 20 patents. His main research interests include automotive hardware and software systems, including dependable real-time wired and wireless communication, performance analysis, and functional safety and security.

KEES G. W. GOOSSENS (Member, IEEE) received the Ph.D. degree from the University of Edinburgh, in 1993, with a focus on hardware verification using embeddings of formal semantics of hardware description languages in proof systems.

From 1995 to 2010, he worked with Philips/NXP on real-time networks on chip for consumer electronics. He was a part-time Full Professor with Delft University, from 2007 to 2010. Since then, he has been a Full Professor with the Eindhoven University of Technology, researching composable, predictable, low-power embedded systems, and supporting multiple models of computation. His Google Scholar H index is 49. He published four books, more than 200 articles, 17 patents, and the DRAM-power open source software.