From a Federated to an Integrated Automotive Architecture

R. Obermaisser, C. El Salloum, B. Huber, H. Kopetz

Vienna University of Technology

***Abstract*—This paper describes an integrated system architecture for automotive electronic systems based on multi-core System-on-a-Chips (SoCs). We integrate functions from different suppliers into a few powerful ECUs using a dedicated core for each function. This work is fueled by technological opportunities resulting from recent advances in the semiconductor industry and the challenges of providing dependable automotive electronic systems at competitive costs. The presented architecture introduces infrastructure IP cores to overcome key challenges in moving to automotive multi-core SoCs: a time-triggered network-on-a-chip with fault isolation for the interconnection of functional IP cores, a diagnostic IP core for error detection and state recovery, a gateway IP core for interfacing legacy systems, and an IP core for reconfiguration. The paper also outlines the migration from today's federated architectures to the proposed integrated architecture using an exemplary automotive E/E system.**

***Index Terms*—Real time systems, Computer architecture, Fault tolerance, Road vehicle electronics**

I. Introduction

Over the past twenty years, many of the conventional mechanical or hydraulic control systems within a car have been replaced or enhanced by electronic control functions. Additionally, new functions, such as an electronic navigation system or sophisticated multi-media systems have been introduced in order to support the driver and enhance the driving experience. The current automotive electronics architecture has thus evolved in a *bottom-up fashion* driven by the technical capabilities and economics of the available hardware and by the organizational constraints of the automotive industry. In a typical car of today, different suppliers develop the different distributed application subsystems (DAS), such as the engine control system, the braking system or the multimedia system in a nearly autonomous manner. In order to establish clear *spheres of liability* the OEM (original equipment manufacturer, i.e., the automotive company) tries to allocate the system responsibility of a DAS to a single supplier. As a consequence, every major DAS is implemented on a nearly autonomous distributed hardware base, consisting of electronic control units (ECUs), networks and the sensors and actuators that are required to implement the functions. As a consequence we find more than fifty ECUs and five different local area networks in a premium car of today [1]. We call such an architecture, where every major function is implemented in a dedicated hardware unit a *federated architecture*.

In addition to this clear allocation of responsibility, the federated hardware approach has remarkable advantages from the point of view of complexity management, fault-isolation and error containment. In a federated architecture the fault-containment units, the ECUs, are clearly defined and it is hardly possible that implicit dependencies cause error propagation from a faulty ECU to another physically distant ECU that is not directly affected by the fault. However, there are many cost, dependability and weight arguments in favor of reducing the number of ECUs and cables by integrating different functions, developed by different suppliers, into a single ECU. We call such an architecture, where a single integrated distributed hardware base for the execution of jobs from different DASs is provided, an *integrated architecture*. *Hammett* describes aptly the technical challenge in the design of an integrated architecture in the avionics domain: [2], p.32: *The ideal future avionics systems would combine the complexity management advantages of the federated approach, but would also realize the functional integration and hardware efficiency benefits of an integrated system.*

In the recent past, a number of efforts have been made to develop such an integrated architecture e.g., *Integrated Modular Avionics (IMA)* [3] in the aerospace domain and *AUTOSAR* [4] in the automotive domain. The key idea in these approaches is the provision of a partitioned operating system for a computer with a single powerful CPU. This operating system is intended to provide in each partition an encapsulated execution environment for a single job of a DAS and eliminate any run-time dependency and error propagation path among the jobs of different DASs. However, the required encapsulation of each job, particularly w.r.t. temporal properties and transient failures, is difficult to achieve in such an architecture.

Ideally, we would like to see an *integrated system architecture* that has the same composability, fault-isolation and error-containment properties than today's *federated* solution, but still supports the integration of multiple functions, developed by different organizational entities, into a single ECU. The

Manuscript received June 16, 2008. This work has been supported in part by the European IST project ARTIST2 under project No. IST-004527 and the national FITIT project TT-SoC under grant number 813299/7852.

advent of properly designed multi-core Systems-on-a-chip (SoCs) offers new possibilities to reach this ambitious goal. This paper explores this design alternative.

The rest of this paper is structured as follows. In Section II we describe the requirements that must be met by an integrated automotive architecture. Section III discusses the recent technological developments in the field of computer architecture and dependability that support the implementation of an integrated ECU. Section IV presents a blueprint of an integrated electronics architecture of the future. Section V is devoted to an explanation of state and state recovery mechanisms in the architecture. The issue of legacy integration is the topic of Section VI. The differences between today's industrial chain and the future industrial chain using the proposed architecture is the focus of section VII. After a discussion in section VIII, the paper terminates with a conclusion in Section IX.

II. Requirements

In this Section we elaborate on the requirements that must be met by an integrated automotive architecture. The first and foremost requirement for the electronic control systems onboard a car is the provision of a dependable service at competitive costs. The service dependability is impaired by physical failures of the hardware and by residual design errors in the software. The cost is composed of the non-recurring engineering and testing effort in the design phase, the recurring manufacturing and installation cost of the ECUs, the sensors, actuators and the cabling in every car and the maintenance cost of the car electronics over the lifetime of the car.

The non-recurring cost of design and testing can be reduced if a component-based architecture is deployed that supports the straightforward implementation and the massive reuse of existing well-tested hardware/software components from different suppliers. The architecture must ensure that the component services can be well-specified in the domains of time and value. Furthermore, component failures must be unambiguously identified and error propagation among components is detected before fault-free components are infected by erroneous input messages from faulty components. Furthermore, the fast restart of a component caused by a transient fault must be supported.

The recurring costs for hardware, cables and sensors can be cut if the number of ECUs and cables is reduced by integrating functions from the different DASs into a few powerful ECUs. Safety-critical functions must be replicated in physically separated ECUs such that the total loss of any ECU can be tolerated without any effect on the safety-critical services of the electronic system (e.g., the brake service).

The maintenance cost of the car electronics can be reduced if the behavior of field-replaceable units is continuously monitored in order to detect an increasing failure rate of the hardware which is a good indication for an upcoming permanent failure. Support for diagnostics and maintenance at the architecture level is thus an important requirement.

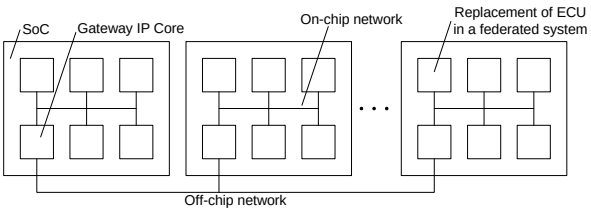

Fig. 1. Integrated Architecture

III. Technological Developments

Over the past forty years, the semiconductor industry has experienced an exponential growth (Moore's law) that has also been of high relevance for the field of automotive electronics—in the domains of dramatic performance improvements, reliability enhancements and cost reductions. At present it is possible to build a Multiprocessor-System-Chip (MPSoC) comparable to the size of a Cell chip [5] in 90nm automotive qualified silicon. Considering the pace of progress in the semiconductor industry, such MPSoCs could become commodity products within a few years.

The first generation of the Cell chip from IBM, Sony and Toshiba contains a master processor and eight 32-bit wide slave processors, each with a local memory of 256 kbyte. Such a slave processor with its local member can provide a performance that is comparable to many of the present-day automotive ECUs. However, the interconnect of the Cell processor is not free of temporal interference and can thus lead to emergent failures in case independently developed subsystems are integrated on a single die. What is needed is a new type of interconnect that supports the constructive composition of components.

A negative effect of the further miniaturization of devices is the expected increase in the transient failure rate of chips. Although the transient failure rate of a single transistor is expected to decrease, this decrease will not be large enough to compensate for the increase in the numbers of transistors on a die [6]. It is thus necessary to provide system-level mechanisms to enhance the robustness and fault-tolerance of automotive applications.

IV. A Blueprint for an Integrated Automotive Architecture

An overview of the integrated automotive architecture is depicted in Fig. 1. The overall electronic system consists of SoCs that are interconnected by an off-chip network. Each SoC contains functional IP cores that are interconnected by an on-chip network. Each functional IP core can serve as a replacement for an ECU of today's automotive architectures. In addition, architectural IP cores provide infrastructure services for diagnosis, startup/reconfiguration and the connection to off-chip networks.

A. Functional IP cores as replacements for ECUs

The proposed architecture follows a strict component orientation from the hardware/software point of view. Functional IP cores provide application services that can be

independently developed and used as a building block in the design of the automotive electronic system. An IP core can provide the functionality of a complete ECU of a federated architecture.

An IP core is integrated with other IP cores to an SoC based on the specification of its *linking interface (LIF)*. The LIF of an IP core abstracts over its internal structure. The LIFs are technology independent in the sense that a LIF does not incorporate implementation details of an IP core. A technology independent LIF ensures that different implementations can be integrated on a single chip (e.g., general purpose CPU, FPGA, ASIC).

The LIFs are precisely specified in time and value at the operational level for the purpose of interoperability. In addition, an interface model assigns meaning to the syntactic structure of the operational specification. A precise LIF specification includes the following information [7]:

- Input and output assertions
- Syntactic properties
- Temporal properties
- Dependability properties
- Semantic specification
- Relevant state for reintegration

Due to the availability of the LIFs, an IP core can have a complex internal structure that is neither visible, nor of concern, to the user of the IP core at the architecture level. Thereby, an IP core offers an appropriate unit of abstraction for the design. Each IP core forms a stable intermediate form that can be studied in isolation and exhibits aggregating properties, when integrated with other IP cores to a SoC. As expressed in [8], '*complex systems will evolve from simple systems much more rapidly if there are stable intermediate forms than if there are not.*'

Particular emphasis in the proposed architecture lies on ensuring that the role of IP cores as stable intermediate forms in an SoC does not break up in the presence of faults. To accomplish this goal, the architecture provides the following fault isolation mechanisms that ensure that each IP core forms a fault containment region [9]:

- ***Separation of computational resources through the physical segregation of functions in dedicated cores:*** Different IP cores cannot perform implicit sharing of computational resources (e.g., memory). If an IP core requires the services of another IP core in order to provide its own services, it has to exploit the services of the other IP core via LIFs.

  For example, the proposed architecture rules out use cases where multiple IP cores access directly a common memory that would belong to all IP cores and no specific IP core in particular. Such a use case would bring in the potential for unintended interference between IP cores that cannot be anticipated from the LIF specifications. By prohibiting any use of common resources by-passing the LIF, we ensure that the interactions between IP cores can be fully understood by solely looking at the LIFs. Even before the deployment on the target platform, the behavior of an SoC can be fully understood given the LIF specifications.

  IP cores are assumed to have local memories. In addition, common memories can be realized in the proposed architecture through an IP core that provides a *storage service*. Via its LIF, such a storage IP core can handle read and write requests to a memory. The fundamental difference to the above mentioned implicit resource sharing is that the storage IP core makes the dependencies and interaction patterns between IP cores explicit via the LIF specifications. Thereby, all the information which is required to understand the interactions at the chip-level is incorporated in the LIF specification.
- **Temporal and spatial partitioning through the NoC:** The on-chip network (cf. Section IV.C) of the proposed architecture employs a time-triggered communication schedule in order to allocate dedicated spatio-temporal communication resources to each functional IP core. The NoC enforces this schedule like in time-triggered off-chip networks with guardians such as FlexRay [10]. Hence, an IP core can neither affect the temporal properties nor the contents of messages exchanged by other IP cores.
- **Monitoring of side channels through the diagnostic IP core:** Side channels are the major obstacle to restricting the interactions between IP cores to the exchange of messages via LIFs. Examples of side channels are an IP core's power consumption or the thermal properties. In theory, an IP core could consume so much power that other IP cores of the SoC need to be clocked down as a compensation. Therefore, the diagnostic IP core retrieves information from on-chip sensors (e.g., thermal on-chip sensors [11]) that monitor side channels and perform a shutdown of misbehaving IP cores.

B. Infrastructure IP Cores

In addition to the functional IP cores, the proposed SoC contains IP cores with predefined roles:

The *diagnostic IP core* is responsible for monitoring the behavior of the functional IP cores at the LIFs and via on-chip sensors (e.g., thermal status, power dissipation). When detecting an error of a functional IP core, the diagnostic IP core resets the functional IP and restores its state (cf. Section V). If specific functional IP cores repeatedly exhibit errors over time, the diagnostic IP core indicates the need for a maintenance decision (e.g., new software to resolve a design fault, replacement of the chip to resolve a permanent hardware fault). The diagnostic IP core can employ threshold techniques, such as the alpha count mechanism [12], to detect permanent faults leading to transient failures.

Another IP core with a predefined role is the *Trusted Network Authority (TNA)*. The TNA is the only IP core in the architecture that can write the time-triggered communication schedule of the on-chip network. The TNA is employed during startup in order to establish the initial schedules. At run-time, the TNA can write a schedule to the on-chip network to perform a restart or enable reconfiguration.

Gateway IP cores are the third type of infrastructure IP cores. These IP cores serve for the interconnection between on-chip and off-chip networks (cf. Section IV.D).

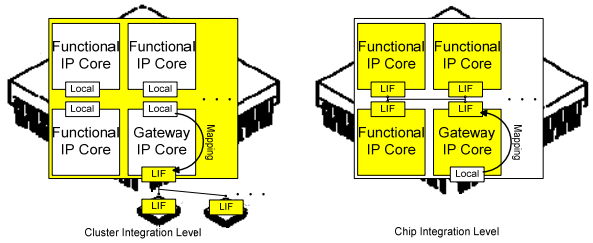

Fig. 2. Integration Levels

C. *Message-Based On-Chip Network*

A message-based on-chip network serves for the interconnection of the functional IP cores and the infrastructure IP cores. Due to the advantages of the time-triggered communication paradigm with respect to predictability, determinism and fault isolation, the on-chip network is based on this paradigm. These advantages have also lead to the introduction of time-triggered communication protocols for automotive off-chip networks [10]. In particular, the determinism of the time-triggered paradigm offers the basis for the implementation of fault-tolerance through active redundancy.

In contrast to time-triggered off-chip networks, the technological constraints at the chip-level (e.g., shorter signal lines, low cost of additional signal lines) enable the on-chip network to offer higher bitrates and massively redundant links between IP cores.

The on-chip network mediates the access to the on-chip network with a so-called Trusted Interface Subsytem (TISS) that is deployed in each IP core. The TISS is equipped with a table-driven communication schedule that encodes slots in a spatio-temporal allocation of the communication resources.

Spatial allocation: The NoC can provide redundant links between IP cores. Hence, multiple messages can be transmitted in parallel. The spatial allocation determines which ones of these links are available to an IP core at a specific point in time. The actual availability of the links depends on the physical topology of the NoC (e.g., mesh, torus, tree).

Temporal allocation: Based on a chip-wide global notion of time, time is divided into slots that are allocated to the IP cores. An IP core gets exclusive access to a communication link during a time slot.

Based on the time slots on specified links, the on-chip network offers the following three communication modes:

- **Periodic exchange of messages:** Using multicast, a message is sent with a specific period and phase from one sending IP core to one or more receiving IP cores.
- **Sporadic exchange of messages:** A sending IP core produces sporadic messages with a minimum message interarrival time. These messages are placed in an outgoing message queue and transported to the receiving IP cores where the messages are stored in incoming message queues.
- **Streaming information:** A sending IP core produces a continuous bit-stream of information that is enqueued. Using the slots of the communication resource allocation, blocks of this stream are transported to the receiving IP cores. In contrast to sporadic messages, receivers do not wait for the reception of complete messages, but can perform the processing of data on-the-fly while the transmission is in progress.

The allocation of time slots on the specified links is also used for fault isolation. The tables in the TISSes can only be written by the TNA. Thus, the application in a functional IP core cannot interfere with the use of slots of other IP cores. As a consequence, the slots provide to a sending IP core an encapsulated communication channels. The communication channel is encapsulated against interference from other IP cores in the temporal and value domain.

D. *SoCs interconnected by Off-Chip Networks*

Using an off-chip communication network (e.g., FlexRay [10]) multiple SoCs can be interconnected to a cluster. Thereby, applications can be supported that need more resources than are available on a single SoC. In addition, a distributed system with multiple SoCs is a prerequisite for implementing safety-critical applications, because today's semiconductor technology does not support the manufacturing of chips with a reliability that is suitable for ultra-dependable applications [6]. In addition, the physical proximity of the IP cores on an SoC results in a probability of common mode failures that must be considered in safety-critical applications.

To accomplish this goal, the proposed architecture employs gateway IP cores. A gateway IP core contains two LIFs, each of which is only visible at a distinct integration level. In the proposed architecture we distinguish two integration levels: At the *chip-level integration level* IP cores are integrated using the on-chip network. The integration of chips with an off-chip communication network occurs at the *cluster integration level* (see Fig. 2).

From the point of view of integrating IP cores on the SoC, the gateway IP core provides a LIF to the on-chip network. The interface to the chip-external network can be considered as a local interface. A local interface at the chip-level is an interface to a sensor, an actuator, or any network other than the on-chip network. The connection at the local interface to sensors and actuators can occur with standardized fieldbuses such as LIN [13] or CAN [14]. The interface to the chip-external network is not directly visible for the functional IP cores of the SoC. Relevant properties of this interface must be mapped by the gateway IP core to its LIF. Relevant properties include, e.g., the timing and syntax of the messages that are redirected between the off-chip network and the on-chip network.

Inversely, from the point of view of integrating chips at the cluster level, the LIF of the SoC is the off-chip interface provided by the gateway IP core. In this case, properties of the on-chip interfaces need to be mapped to the off-chip interface, because the on-chip interfaces are not visible at the cluster level.

V. State Awareness and Recovery

The proposed integrated automotive architecture facilitates the definition, the establishment, and the observation of a consistent system state of distributed IP cores. For this reason,

we exploit the consistent notion of state for system diagnosis and recovery.

A. Consistent Notion of State

State is a key concept for the design of embedded systems. The Webster's New World Dictionary defines state as [15] "*a set of circumstances or attributes characterizing a person or thing at a given time*." When considering a computer system, only a subset of the entire state is relevant for the system's future behavior. We call this part of the overall state the *declared state*. If the system is distributed, the overall state is partitioned among multiple components executing concurrently and mostly independently (e.g., IP cores in an SoC). In distributed systems it is a major challenge to establish and to maintain the *consistency* of the different parts of the state (e.g., the declared state of replicated IP cores has to be identical).

The proposed integrated automotive architecture facilitates the definition, the establishment and the observation of a consistent system state by relying on a *sparse time base* [16]. In the sparse time model the continuum of real-time is partitioned into a sequence of alternating intervals of *activity* of duration ε and intervals of *silence* of duration Δ. All events that are in the *sphere of control* of the system (i.e. are generated by the system itself like sending a message) are restricted to occur within the intervals of activity. The intervals of activity are consecutively numbered by positive integers and the number assigned to an activity interval is called the *global timestamp* of events happening in that interval. Events that occur within the same interval of activity have the same global timestamp and are thus considered as having occurred *simultaneously*. The sparse time base enables a *chip-wide consistent temporal order* of all events. The chip-wide consistent temporal order of events facilitates the definition of a consistent system state between selected activity intervals without the need to execute costly agreement protocols.

B. State and Determinism

According to Mesarovic, the concept of state is closely related to the concept of *determinism* which is a significant property for many problems and challenges of distributed computing: [17], p.45 "*The state enables the determination of a future output solely on the basis of the future input and the state the system is in.*" An example where determinism is required is fault tolerance by active redundancy. Consider for example a future brake-by-wire function using a Triple Modular Redundancy (TMR) scheme with three replicated IP cores on physically separated chips. In order to be able to compare the outputs of the replicas systematically on a bit-per-bit base the replicas have to show *replica deterministic* behavior. Replica determinism requires that all correct replicas produce exactly the same output messages that are at most an interval of *d* time units apart, as seen by an omniscient outside observer. This can only be guaranteed if the state of the replicas is consistent. As mentioned before, the sparse time base of the integrated automotive architecture facilitates the establishment of a consistent state between selected activity intervals.

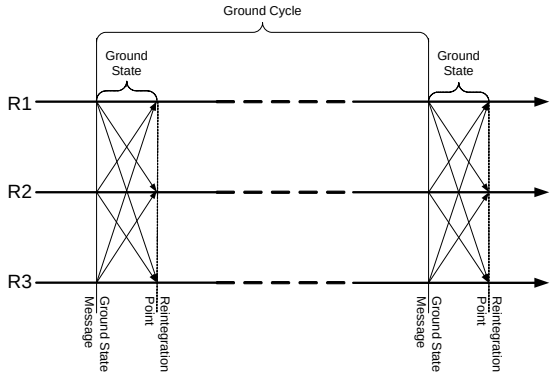

Fig. 3. Ground Cycle in a TMR Configuration

C. Diagnosis and Recovery

A consistent notion of state is highly relevant for system diagnosis and recovery. In general we can interpret the correctness of the state of one IP core only in relation to the state of other IP cores. Think of the electronic brakes of a car. We can only decide whether their actual state is valid or not by considering also the current and recent position of the brake pedal that indicates the desired position of the brakes.

With respect to recovery, a consistent notion of state is required for the reintegration of IP cores that have failed due to transient faults. The key issue during reintegration in a real-time system is to find a future point in time when the state of the IP core is in synchrony with the IP core's environment [18]. In order to facilitate the reintegration of failed IP cores it is beneficial to introduce periodic instants where a IP core's state is *minimal*. This strategy is followed in our architectural approach where we call such a minimal state a *ground state*, and the duration of the interval between two successive ground states the IP core's *ground state cycle.* During system design the precise specification of the ground state at the planned periodic *reintegration points* is required. To simplify the reintegration it is suggested to divide the ground state into three parts [18]:

i. The first part of the ground state consists of information that can be retrieved by sensors from the environment. If the sensors deliver state information, then a complete rescan of all involved sensors is sufficient to resynchronize this part of the IP core's ground state with the state of the environment.
ii. The second part of the ground state consists of output data that is in the sphere of control of the system and can be enforced on the environment. This set of output data is called a *restart vector*. In order to resynchronize this part of the IP core's ground state with the state of the environment the state vector can be enforced on the environment. An example where a restart vector could be applied is a traffic control system where all traffic lights could be set first to yellow and then to red in order to resynchronize the IP core's internal state to the state of the environment without knowing the environment's state.
iii. The third part of the ground state is the most problematic part since it contains all the data that does not fall into

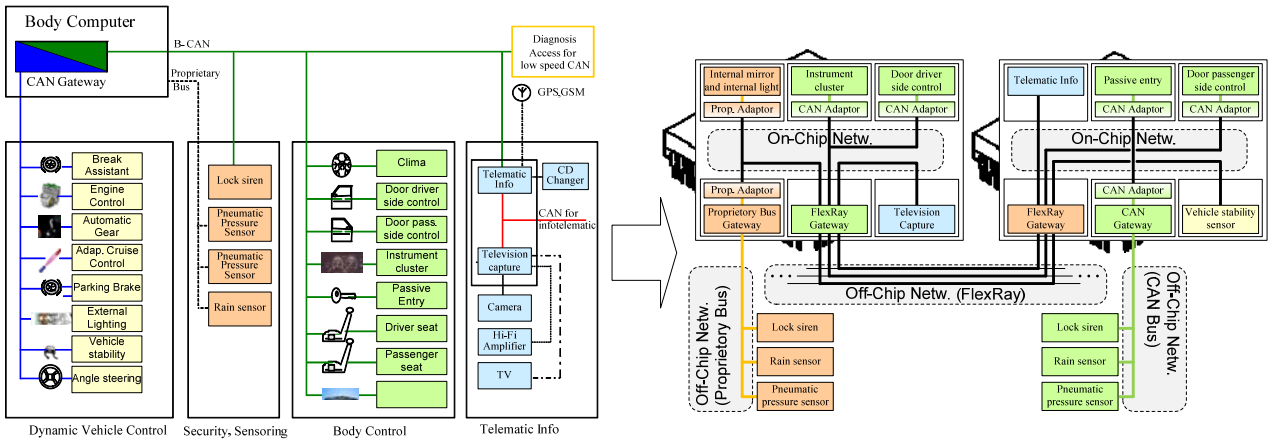

Fig. 4. Mapping of present day automotive E/E system to the proposed architecture (exemplified for eight selected ECUs: internal mirror and internal light, instrument cluster, door driver side control, television capture, telematic info, passive entry, door passenger/side control, vehicle stability sensor)

category (i) or category (ii). This part of the state must be recovered from an external source of the IP core.

To solve the third challenge, each IP core periodically sends out the ground state via a *ground state message*. Thereby, we also simplify diagnosis and recovery. The ground state messages externalize the state of the components and make it observable. The advantage of having the ground state available for diagnosis instead of monitoring only the IP core's outputs is that *dormant faults [19]* in the internal state can be detected that have not shown up yet at the outputs but will potentially cause an IP core failure at a later point in time. Also for recovery and reintegration, the periodic dissemination of the ground state has significant benefits. In a system that employs active redundancy to tolerate IP core failures (e.g., IP cores in a TMR configuration) the ground state message of a correct IP core can serve as a *restart message* of a replicated IP core after is has failed due to a transient fault. Fig. 3 depicts three replicated IP cores in a TMR configuration where each IP core periodically sends its ground state message to the other two IP cores. At every reintegration point each replica votes over the three ground states (over its own ground state and the ground state of the other two replicas) and overwrites its internal state with the majority result of the vote. Thus, the replicas use the ground state messages of the other replicas for diagnosis as well as a restart message in case a replica has failed.

If an IP core is not redundant, the ground state message of the last ground cycle can be stored by the diagnostic IP core described in Section IV.B. In many application scenarios it is sufficient to use the ground state of the last ground cycle as a restart message for the next ground cycle. Of course this state is not in perfect synchrony (i.e., delay of one ground cycle) with the actual state of the environment but, depending on the length of the ground cycle, the restart time and the dynamics of the controlled system or process, this can often be tolerated.

VI. Migration Path and Legacy Integration

This section describes a migration path for switching from present day federated architectures to the proposed integrated architecture. The left hand side of Fig. 4 depicts an E/E system of a present day Fiat car, while the right hand side shows a realization with the proposed architecture.

ECUs from the left hand side are replaced by functional IP cores in the realization on the right hand side. The communication activities in the federated architecture occur with different off-chip networks, such as CAN buses with different bandwidths or proprietary buses. The integrated solution combines on-chip and off-chip networks.

A. Time-Triggered On-Chip and Off-Chip Networks

The IP cores in each SoC are interconnected with a time-triggered on-chip network as introduced in Section IV.C. Each IP core is assigned time slots that are used to transport messages to other IP cores of the SoC. These time slots serve as replacements for the different off-chip networks in the federated architecture.

Furthermore, the integrated system in Fig. 4 employs FlexRay as an off-chip network to interconnect multiple SoCs. Each SoC contains a FlexRay gateway as an infrastructure IP core. Due to the use of the time-triggered paradigm, the gateway can relay the contents of slots from the NoC to slots on the FlexRay network.

In case ECUs of an application subsystem have been assigned to IP cores of different SoCs (e.g., body control in Fig. 4), a communication activity within an application subsystem involves several steps. First, a message is transported from a producing functional IP core to the FlexRay gateway. The gateway sends the message on the FlexRay network, where the message is transported to the receiving SoC. There, another FlexRay gateway relays the message to the NoC where it is transported to the receiving IP core.

Since the latencies of an NoC are generally orders of magnitude lower than the latencies of an off-chip network (e.g., due to physical characteristics such as short distances or the low price of many parallel wires), the additional overhead of these communication steps is tolerable for the majority of automotive applications.

B. Gateways to Legacy Networks

Due to existing investments, it cannot be assumed that all ECUs are instantly replaced by the new architecture. Therefore legacy ECUs and legacy networks can be reused in the integrated architecture.

In addition to the FlexRay gateway, further infrastructure IP cores can be used to establish connections to different types of off-chip networks. In Fig. 4, several ECUs of the security/sensing subsystem and the body control subsystem are reused without being integrated into SoCs.

C. Network Adaptors

Network adaptors serve for the reuse of application software from ECUs of the federated system within IP cores of the SoC. A network adaptor establishes the Application Programming Interface (API) of a legacy network. Thereby, the on-chip network appears to the legacy application software within an IP core like a legacy network. For example, in Fig. 4 network adaptors for CAN and a proprietary bus are depicted. The network adaptor implements the protocol of the legacy network as a higher protocol on top of the time-triggered on-chip network. The establishment of higher protocols on top of a time-triggered communication service has already been shown in previous work, e.g., CAN [20], TCP/IP [21] or a transport protocol for CORBA [22]. Since the on-chip network provides a time-triggered communication service, the solutions can be applied in the proposed integrated architecture.

D. Compatibility to AUTOSAR

The proposed architecture is compatible to AUTOSAR. The *Automotive Open System Architecture* (AUTOSAR) [4] is an attempt to exploit the benefits of integrated system architectures in the automotive domain. It is the main objective of this initiative to facilitate the reuse of AUTOSAR *Software Components* (SW-Cs) between different vehicle platforms, OEMs, and suppliers. For this purpose, AUTOSAR defines a standardized software architecture for each ECU that provides a technology-independent infrastructure for SW-Cs. This uniform environment is provided by the AUTOSAR Runtime Environment (RTE), which abstracts from implementation details of the ECU. For instance, it provides standardized communication services to the application software, which are defined independently whether the communication manifests after the integration of the system in inter-ECU or intra-ECU information exchange [23]. Thus, a SW-C need not be aware of its physical location and the physical location of other SW-Cs.

The proposed architecture forms an ideal hardware platform for realizing these objectives of AUTOSAR.

Encapsulated execution platform for SW-Cs. AUTOSAR SW-Cs are atomic components, which means that each instantiation of a SW-C is allocated to exactly one ECU and cannot be distributed across several ECUs. Thus, the IP cores of the proposed SoC form a suitable execution platform for SW-Cs. In addition, if we restrict the allocation of SW-Cs to IP cores to a 1-to-1 mapping, i.e. each IP core executes exactly one SW-C, the problems of spatial and temporal partitioning between SW-Cs are ruled out by design. This way, the need for certified operating system can be eliminated.

Emulation of the AUTOSAR RTE. The on-chip network of the integrated automotive architecture provides three basic communication modes, namely periodic messages, sporadic messages, and streaming information. On top of these communication modes, an implementation of the RTE can be established using a network adaptor that provides the AUTOSAR functionality to the SW-C within an IP core.

Support of the AUTOSAR communication patterns. The Virtual Function Bus (VFB) of AUTOSAR specifies explicitly two different communication patterns for SW-Cs: client/server communication and sender/receiver communication. The sender/receiver pattern is used for the distribution of data from one sender to one or more receivers via ports. This communication pattern of AUTOSAR is natively supported by the message-based communication infrastructure of the proposed architecture. Furthermore, data can be relayed with minimal latency to different SoCs using gateways. In the client/server communication, one SW-C acts as server that provides its service to one or more clients, which use the service. The transmission of a service request in addition with an optional parameter set from the client to the server as well as the transmission of the response of the server back to the client can be realized with little effort on top of the message based on-chip network using middleware.

VII. INDUSTRIAL CHAIN

In today's automotive development chain, car manufactures submit the specification of an automotive function (e.g., ABS) to a supplier, including constraints concerning technical properties and cost [24, 25]. Today, the supplier typically delivers the entire function in the form of one or more ECUs. According to these requirements the supplier is responsible for the function including hardware and software of both the platform and the application. In this process, the role of the OEM is limited, because the supplier delivers entire system parts according to the requirements of the OEM. However, most of the important development technologies belong to the supplier and in case any technical problems take place after the ECUs are delivered, they must be solved in cooperation with the supplier.

In this development process, the OEM would generally be unable to combine different functions using shared ECUs (as proposed in this paper). The OEM receives self-contained supplier-specific ECUs and is not aware of the separation between platform and application.

The proposed MPSoC architecture enables an AUTOSAR conformant industrial chain, where it becomes possible to cleanly differentiate between platform developers and application developers. Functions from multiple application developers can be combined on a shared platform. The coordination between the platform and application developers could be performed by the OEM and is simplified by encapsulation mechanisms of the proposed MPSoC, which

aim at avoiding unintended interference between functions from different application developers.

The proposed MPSoC represents thus an important technology for an AUTOSAR-compliant industrial chain. By exploiting the temporal and spatial partitioning of the proposed MPSoC, application developers can realize application functions independently of other application developers and the system integration performed by the OEM is facilitated.

VIII. Discussion

This section discusses the major benefits of the proposed integrated automotive architecture.

A. Complexity Reduction

The inherent complexity and comprehensive functionality in present automotive applications enforce the optimal support for complexity management as one of the major design drivers for future automotive architectures. As stated in the report on "Software for Dependable Systems: Sufficient Evidence?" from the National Academies of July 2007 "*…one key to achieving dependability at reasonable cost is a serious and sustained commitment to simplicity, including simplicity of critical functions and simplicity in system interactions. This commitment is often the mark of true expertise [26]*." Following this recommendation, the proposed integrated automotive architecture employs the following *simplification strategies* [27]:

Abstraction. Abstraction is a deliberate simplification capturing only those properties that are relevant for a particular purpose, while disregarding detail that is irrelevant for the given purpose. The support for abstraction in the integrated automotive architecture is three-fold: Firstly, the architecture provides a message-based network interface at the TISS that abstracts over the concrete implementation of the communication infrastructure. Secondly, on top of this interface additional middleware layers (either realized as software or as dedicated hardware block within an IP core) can be stacked, which expose only the relevant properties of the platform to the application layer at appropriate level of detail. Finally, IP cores interact solely via the exchange of message over a well-specified LIF. Thereby, IP cores can have a complex internal structure that is neither visible nor of concern for the user of the IP core.

Partitioning. Partitioning is the separation of components in order to avoid unintended feature interaction. The integrated automotive architecture employs a time-triggered message-based on-chip communication system that provides temporal and spatial partitioning. By the use of a protected time-triggered communication schedule stored in each TISS and separate memory regions for the storage of each individual message, an IP core can neither affect the temporal properties nor the contents of messages exchanged by other IP cores.

Segmentation. Segmentation is concerned with the introduction of structure into the behavior of components for supporting the temporal decomposition of behavior into smaller parts that can be processed sequentially. The sparse global time model that is the foundation of the architecture is suitable for the support of segmentation. It eases to reason about the chronology of events, since the challenge simultaneity can be simply solved with this model of time.

B. Reduction of Resource Requirements

The state-of-the-art practice in automotive systems to deploy a dedicated ECU to a single job of a distributed application becomes very costly and is reaching its limits (e.g. the BMW 7 series cars contain up to 70 ECUs [1]). Experiences from other applications domains where integrated architectures are a matured technology show that the potential of reducing the number of deployed ECU by using an integrated architecture promises massive cost savings. For instance, in the Boeing 787 Dreamliner the introduction of the IMA architecture has enabled a reduction of components resulting in a weight reduction of about 900 kg compared to previous aircrafts [28].

This challenge is perfectly addressed by the proposed integrated automotive architecture by establishing (in the best case) a one-to-one mapping between ECUs and IP cores. Taking the recent appearance of multi-core SoCs as indicators, a formidable reduction of ECUs can be envisioned. For instance, each of the 8 slave cores (synergistic processing elements) of the Cell broadband engine provides a comparable performance to many present automotive ECUs. Accompanied with the reduction of ECUs is the reduction of wires and cabling. Besides the economic benefits resulting from such a reduction, the dependability of automotive systems is also improved, since in automotive environments more than 30% of electrical failures are ascribed to connector problems [29].

C. Dependability in the Light of Increasing Transient Failure Rates

The reliability of today's electronic devices is significantly impacted by the increasing vulnerability with respect to soft errors. Beyond the 90 nanometer feature size of electronic devices, as it is state-of-the-art today, logic circuits and latches become increasingly vulnerable to cosmic rays and alpha particles. Therefore, the architecture provides mechanisms at the architectural level to handle the resulting increasing transient failure rate of electronic devices: Firstly, the encapsulation of communication activities of individual IP cores by the provision of temporal and spatial partitioning and the explicit definition and establishment of different fault-containment regions for IP cores and TISSs ensure that in case a fault hitting a particular IP core, the remaining set of correct IP cores are not impaired. For safety-related applications, this fault management strategy is too weak, since the correct functionality of the IP cores depends on the correct functionality of the chip's infrastructure such as the on-chip network and the TNA. Such applications require additional fault-tolerance strategies like the replication of functionality using a triple-modular redundant configuration of IP cores located on different SoCs. The sparse time base of the integrated automotive architecture is a key enabler of replica determinism, which is a prerequisite for TMR based on systematic bit-exact voting.

Secondly, the explicit specification of regular reintegration points for IP cores and the periodic externalization of the state of an IP core using ground state messages enable the recovery after a transient failure.

D. Economic Benefits

The proposed integrated system architecture will result in quantifiable cost reductions in the development and deployment of embedded systems in the areas of system hardware cost and maintenance. The replacement of a subset of the chip-external networks with chip-internal NoCs reduces wiring and the number of connectors. In addition, the integration of IP cores as replacements of ECUs on a single chip leads to a reduction of the overall number of ECUs. In order to quantify these savings, we will estimate the implications onto the required number of hardware units on-board a high-end car.

Let us assume that the integration of different functions, developed by different suppliers, into a single ECU will result in a reduction of 20% of the hardware units and a corresponding reduction in the number of wiring points of a car. On the other side, the new hardware units will be more powerful and may thus cost more. Also, the development costs of the new MPSoC and costs for chip masks will be part of the cost of the hardware units.

If we consider a typical high-end distributed system on board a car with 50 ECUs, each ECU costing on average about 35€ and 1000 wires, each wire costing about 0.5€, then the total hardware cost of such a system is about 2250€. If an integrated system is deployed, the number of components will be reduced to 40 ECUs of 40€ each (increase of the node cost by 5€), and the number of wires to 800. The total hardware cost will thus be reduced to 2000€ or 250€ per system. A hardware cost reduction of about 10% can thus be realized. The induced savings in the hardware domain during the production of 100,000 cars amount then to about 25 Mio €.

For estimating the implications with respect to maintenance cost, let us assume that the cost of maintenance of an electronic system onboard a car is about 300 € per car over the lifetime of a car. By reducing the number of ECUs by 20% and the number of wiring points by 20%, a reduction of the maintenance cost by 20% can be expected. The induced savings in the maintenance domain in 100,000 cars amount then to about 6 Mio €, not considering the image gain of the manufacturer due to the improved dependability of its product.

E. Legacy Reuse

Legacy systems often represent major investments. Due to cost and time constraints, a complete system redevelopment is often undesirable and a gradual migration to a new technology is performed. Therefore, many large systems consist of a combination of existing legacy subsystems and newly developed subsystems. This coexistence is supported by the integrated automotive architecture by the provision of a dedicated infrastructure IP core, the gateway IP cores. The gateway IP cores enable the interconnection of an SoC to existing legacy networks such as FlexRay, CAN, or LIN. If the off-chip network is also time-triggered, the deterministic network on-chip in combination with the gateway IP cores enables the temporal coordination of activities within functional IP cores on-chip and existing ECUs.

In addition to the reuse of legacy hardware, network adaptors providing APIs for legacy networks facilitate also the reuse of legacy software components. Furthermore, these network adaptors support application design in switching to the new technology.

IX. CONCLUSION

In the automotive domain an integrated architecture is the key to continue building more advanced automotive electronic systems with more functions while maintaining a manageable amount of ECUs. The introduced architecture for automotive multi-core System-on-a-Chips supports this goal and provides composability, fault-isolation and error-containment properties. These properties are essential to build dependable automotive electronic systems that are seamlessly integrated out of independently developed functions from different suppliers. The example with a mapping from a present day automotive E/E system to the proposed architecture demonstrates the ability to reuse existing functions despite the change of the architectural paradigm. Also, the proposed architecture is an ideal hardware platform for realizing the objectives of AUTOSAR. As future work it is planned to establish the AUTOSAR RTE within the functional IP cores in order to support the integration of AUTOSAR software components into SoCs.

REFERENCES

[1] A. Deicke, "The electrical/electronic diagnostic concept of the new 7 series," in *Proc. of the Convergence International Congress & Exposition On Transportation Electronics*: SAE, 2002.

[2] R. Hammett, "Flight-Critical Distributed Systems--Design Considerations," 2002, pp. 13-B3.1-13B3.5.

[3] ARINC, "ARINC Specification 651: Design Guide for Integrated Modular Avionics," Aeronautical Radio, Inc., 2551 Riva Road, Annapolis,Maryland 214011991.

[4] AUTOSAR, "AUTOSAR - Technical Overview V2.0.1," AUTOSAR GbR2006.

[5] IBM, Sony, and Toshiba, "Cell broadband engine architecture," 2006.

[6] C. Constantinescu, "Trends and Challenges in VLSI Circuit Reliability," *IEEE Micro,* vol. 23, p. p. 17, 2003.

[7] H. Kopetz and N. Suri, "Compositional design of RT systems: A conceptual basis for specification of linking interfaces.," in *Proc. of the Sixth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, 2003, pp. 51–60.

[8] H. Simon, *The Sciences of the Artificial*: MIT Press, 1996.

[9] J.H. Lala and R. E. Harper, "Architectural principles for safety-critical real-time applications," *Proc. of the IEEE,* vol. 82, pp. 25–40, 1994.

[10] FlexRay Consortium, "FlexRay Communications System Protocol Specification Version 2.1," 2005.

[11] Pablo Ituero Jose, L. Ayala Marisa, and Lopez-Vallejo, "Leakage-based On-Chip Thermal Sensor for CMOS Technology," in *IEEE International Symposium on Circuits and Systems*, 2007.

[12] A. Bondavalli, S. Chiaradonna, F. Di Giandomenico, and F. Grandoni, "Threshold-based mechanisms to discriminate transient from intermittent faults," *IEEE Transactions on Computers,* vol. 49, pp. 230–245, 2000.

[13] "LIN Specification Package Revision 2.0.," LIN Consortium2003.

[14] Robert Bosch Gmbh, "CAN Specification, Version 2.0," 1991.

[15] D. Guralnik, "Webster's New World Dictionary, Simon & Schuster," Second College Edition1982.

[16] H. Kopetz, "Sparse time versus dense time in distributed realtime systems," in *Proc. of the 12th International Conference on Distributed Computing Systems*, 1992.

[17] M.D. Mesarovic and Y. Takahara, "Chapter 3," in *Abstract Systems Theory*, Springer-Verlag, Ed., 1989.

[18] H. Kopetz, "Real-time systems: design principles for distributed embedded applications," Kluwer Academic Publishers, 1997.

[19] C. Scherrer and A. Steininger, "Dealing With Dormant Faults in an Embedded Fault-Tolerant Computer System," IEEE TRANSACTIONS ON RELIABILITY, vol. 52, 2003.

[20] R. Obermaisser and P. Peti, "Comparison of the temporal performance of physical and virtual CAN networks," in *Proc. of the IEEE Int. Symposium on Industrial Electronics* Dubrovnik, Croatia, 2005.

[21] NextTTA, "Project deliverable D2.4. Emulation of CAN and TCP/IP. High Confidence Architecture for Distributed Control Applications - ST-2001-32111," 2003.

[22] M. Segarra, T. Losert, and R. Obermaisser, "Hard real-time CORBA: TTP transport definition," Universidad Politecnica de Madrid, Lunds Tekniska Högskola, Technische Universität Wien, SCILabs Ingenieros 2003.

[23] H. Heinecke, K.-P. Schnelle, H. Fennel, J. Bortolazzi, L. Lundh, J. Leour, J.-L. Mate, K. Nishikawa, and T. Scharnhorst, "AUTomotive Open System ARchitecture - An Industry-Wide Initiative to Manage the Complexity of Emerging Automotive E/E Architectures," in *Proceedings of the 2004 International Congress on Transportation Electronics, SAE/P-387, pp. 325-332*, 2004.

[24] W. H. Oh, J. H. Lee, H. G. Kwon, and H. J. Yoon, "Model-based Development of Automotive Embedded Systems: A Case of Continuously Variable Transmission (CVT)," in *11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2005.

[25] D. Sörensen, *The Automotive Development Process*: Deutscher Universitäts-Verlag, 2006.

[26] D. Jackson, M. Thomas, and L. I. Millett, "Software for Dependa-ble Systems: Sufficient Evidence?," The National Academies Press, 2007.

[27] G.S. Halford, W.H. Wilson, and S. Phillips, "Abstraction, Nature, Costs, and Benefits," Department of Psychology, University of Queensland, 4072 Australia.1996.

[28] J. W. Ramsey, "Integrated Modular Avionics: Less is More," *Avionics Magazine,* 2007.

[29] J. Swingler and J. W. McBride, "The degradation of road tested automotive connectors," in *45th IEEE Holm Conference on Electrical Contacts*, 1999, pp. 146–152.