# Co-simulation Framework for AUTOSAR Multi-Core Processors with Message-based Network-on-Chips

Moisés Urbina, Hamidreza Ahmadian, Roman Obermaisser
Institute of Embedded Systems - University of Siegen
{moises.urbina, hamidreza.ahmadian, roman.obermaisser}@uni-siegen.de

***Abstract*—Simulation environments play a very important role in the development of embedded systems helping system architects in exploring design decisions. However, the simulation of AUTOSAR multi-core processors with Network-on-Chips (NoCs) for inter-core communication is still a significant research problem. Message-based NoCs provide significant advantages for real-time embedded systems as in the case of the automobile industry. Such a simulation would provide early insights into the real-time behavior of the AUTOSAR application on the message-based multi-core chip. This paper presents as a novel contribution a co-simulation framework supporting the integration of the AUTOSAR architecture with NoC-based platforms. We describe a simulation model for application cores playing the role of virtual AUTOSAR ECUs on the MPSoC platform. The framework introduces an interface for the co-simulation of simulation models for the AUTOSAR-based software (virtual ECUs), the natural environment and the NoC behavior. This co-simulation interface combines a Functional Mock-up Unit (FMU) and a local coordinator for the synchronization and the data exchange between the simulators hosting the simulation models. The implementation is performed using the VEOS simulator for the AUTOSAR-based software and physical environment models, and the GEM5 simulator for the on-chip communication level. An anti-lock braking use case serves for the evaluation of the co-simulation framework.**[1]

## I. Introduction

Multi-core processors have become a preferred option for the development of real-time embedded systems. They provide the possibility to execute different software components in parallel on different cores. At present, in the automotive domain, Multi-Processors System-on-a-Chips (MPSoCs) are widely implemented that use the paradigm of shared memory for the interaction between the cores. The AUTOSAR (Automotive Open System Architecture) standard introduces a multi-core version of the AUTOSAR architecture [1] since its version 4, defining a multi-core Operating System (OS) that controls the execution of AUTOSAR Software Components (SWCs) allocated to different cores with a shared memory for the inter-core communication.

However, message-based Network-on-Chips (NoCs) offer significant advantages [2] for real-time embedded systems as in the case of the automotive applications. In particular, a message-based NoC is superior to a shared memory with respect to fault isolation and temporal predictability [3] [4], which have been identified as weak points in AUTOSAR [5]. In [6] the advantages of the instantiation of AUTOSAR on top of a message-based MPSoC platform are discussed. Nevertheless, the implementation of message-based NoCs for inter-core communication is not supported yet by the AUTOSAR standard.

In previous work [7] we presented a novel MPSoC architecture for AUTOSAR based on a Time-Triggered Network-on-Chip (TTNoC) for the communication between the cores. This architecture introduces application cores that are configured based on the AUTOSAR architecture with extended communication modules for supporting NoC communication. Based on the benefits obtained from the NoC, each application core represents an independent unit of abstraction and plays the role of an autonomous Virtual AUTOSAR Electronic Control Unit (ECU) with its own application software (i.e.,SWCs), Run-Time Environment (RTE) and Basic Software (BSW), using the TTNoC for the interaction with other virtual ECUs in the same MPSoC. This TIme-triggered MEssage-based multi-core architecture for AUTOSAR is called the "TIMEA" platform.

The focus of this paper is a simulation environment for AUTOSAR applications on a multi-core platform with NoCs as introduced in TIMEA. Simulation environments are essential to gain early insights into the real-time behavior of MPSoC platforms and are being highly used also in the automotive domain. In the last years several simulation frameworks have been presented in industry and research communities, many of them performing a co-simulation of simulators for different application levels. For example, [8], [9] and [10] focus on the co-simulation of the communication behavior, while [11],[12] and [13] offer full system co-simulations. These publications present different approaches for the coordination and coupling of the different simulation levels. However, a simulation framework supporting the simulation of TIMEA and the application behavior for virtual validation scenarios is still missing. There is not any method that supports the simulation of AUTOSAR software running on a message-based MPSoC platform. Such a simulation framework is required to quantify in an early state the advantages obtained by the integration of the AUTOSAR architecture with a message-based NoC and to evaluate the impact of the hardware choices (e.g. multi-core topology, NoC configurations) on the AUTOSAR software.

The contribution of this paper is a co-simulation framework supporting the simulation of time-triggered message-based multi-core processors hosting AUTOSAR-based software that can be easily adapted and implemented based on existing AUTOSAR simulators and on-chip simulation tools. We present the coordination of three different simulation levels: the AUTOSAR software, the physical environment and the NoC behavior. The coordination occurs using the Functional Mock-up Interface (FMI) [14] standard in combination with the TCP/IP protocol for the synchronization and the data exchange

[1]This work has been supported in part by the European FP7 project DREAMS under grant agreement 610640. Furthermore, gratitude to the dSpace Company for their software support.

between the respective simulators. Additionally, the extension of AUTOSAR BSW simulation modules for supporting NoC communication is presented.

The proposed framework is implemented using an AUTOSAR-based system simulator (VEOS) at the on-chip application level and a NoC simulator (GEM5) at the on-chip communication level. An Anti-lock Braking System (ABS) is considered as an example use case for the evaluation of the co-simulation framework. It is not the purpose of this paper to provide a dependability analysis of the integration of AUTOSAR with message-based NoCs or of the specific AUTOSAR implementation in the presented use case. To the best of our knowledge, this represents the first published attempt to provide a simulation framework for message-based MPSoCs for AUTOSAR.

The remainder of this paper is structured as follows. Section II presents the simulation framework for the TIMEA platform. The implementation of the simulation framework is the topic of section III. In section IV the evaluation of the framework using a validation scenario is presented. The paper concludes with a discussion in section V.

## II. Co-simulation Framework for TIMEA

The TIMEA platform introduces application cores playing the role of virtual AUTOSAR ECUs using a TTNoC for the communication between each other. To achieve this, the AUTOSAR ECU architecture is extended for supporting message-based NoC communication as illustrated in figure 1.

The proposed simulation framework consists of one AUTOSAR-based system simulator hosting the simulation of the virtual ECUs and the physical environment simulation, one on-chip simulator hosting the NoC simulation, and the coordination of the simulation tools.

In the rest of this section we present the simulation model for the NoC, as well as the simulation models for the virtual AUTOSAR ECUs and the physical environment. Additionally, the co-simulation and coupling of the simulation tools is presented. We introduce a socket-based coordination of simulation tools, which uses the FMI standard and TCP/IP for interfacing and synchronizing the simulation tools.

### A. Simulation Model of Network-on-Chip

The presented simulation framework gives support for different timing models in a message-based multi-core processor running AUTOSAR software. Due to many different and partially contradicting requirements in real-time embedded systems, there is a considerable trend towards multiple timing models in on-chip communication systems to overcome trade-offs in terms of predictability and flexibility [15].

*Time-Triggered (TT) communication of messages* [15] ensures predictability and resource adequacy by *a priori* scheduled transmission times of periodic messages. By dedicating defined time slots to these messages, timely delivery of all messages is guaranteed.

*Rate-Constrained (RC) communication* [15] guarantees a sufficient bandwidth allocation for each message with bounded delays. Priorities determine how contention with other RC

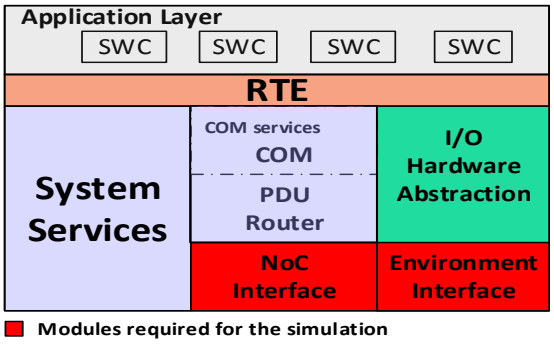


*Fig. 1: Architecture of a simulated Virtual AUTOSAR ECU*

messages is resolved. RC messages provide flexibility and high resource utilization.

The simulated Multi-Processor System-on-Chip (MPSoC) interconnects several simulated cores, using the Network Interfaces (NIs). The interface between the application, running on the core, and the NoC is realized by configurable *Ports*. Based on the configuration, each port contains a FIFO (for event messages) or a buffer with update-in-place semantics (for state messages) to store messages.

Ports are used at the sender core as well as the destination core. At the sender core, the NI dequeues the port at the right instant, which is determined by the schedule, the rate-constraints and the priority. At the destination core, ports keep the arrived messages until the core dequeues them. Alternatively, an interrupt can be used to notify the application, once the message arrives at the port.

### B. Environment Simulation

Environment models act as simulation models representing the natural environment of an ECU in a virtual validation scenario. These environment models emulate the physical process resulting from the interaction of the virtual ECUs with the physical environment. They are controlled by the AUTOSAR-based system simulator and can be integrated as discrete simulation models as well as continuous simulation models into the AUTOSAR system simulation.

### C. Simulation Model of Virtual AUTOSAR ECUs

As described by TIMEA, virtual ECUs host the AUTOSAR SWCs and each one of them is provided with a RTE and BSW. In the proposed framework, virtual ECUs are configured as depicted in figure 1 and simulated by the AUTOSAR system simulator.

Since the AUTOSAR standard does not provide support for NoC communication, we extend the BSW on the simulated virtual ECUs with special BSW modules to support the communication through the NoC. An NoC interface module for the hardware abstraction layer is integrated for accessing the NoC. Extended COM service modules are integrated for routing the messages from the NoC to the application software and vice versa. The integrated NoC interface module allows the access from the BSW of the virtual ECU to the NoC. This module is in charge of the exchange of data between the BSW on the virtual ECU and its corresponding NI in the NoC simulation.

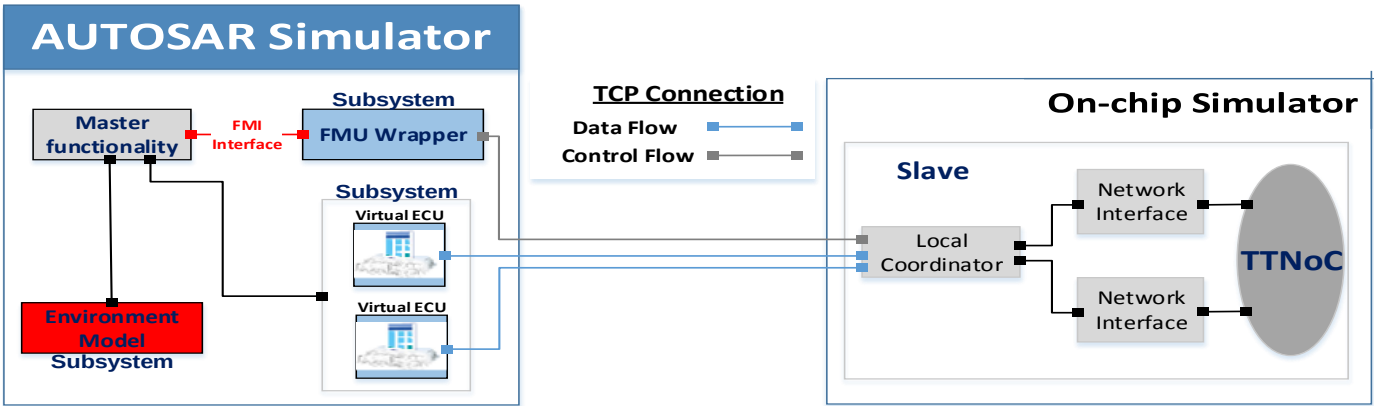


*Fig. 2: Co-simulation coupling*

Moreover, as defined by the AUTOSAR standard, the RTE provides signals that are attached to the data handled by the SWC ports, depending on the communication interface hosted by the port (server/client or sender/receiver). The integrated COM module matches these RTE signals to one of the communication types provided by its NI (RC or TT) in the NoC simulation. This is done by mapping the RTE signals to the sender/receiver ports on the NI. Furthermore, the integrated PDU router [7] is in charge of passing the data from the NoC interface module to the COM module and vice versa.

In case of virtual ECUs hosting sensor/actuator SWCs, the I/O BSW module for the hardware abstraction layer is integrated. Additionally, an environment interface module is integrated to connect the virtual ECU with a physical environment model imported into the AUTOSAR-based system simulator.

The communication hierarchy is as follows:

- Communication between SWCs on the same simulated virtual ECU is performed by the Run-Time Environment (RTE) (as defined by the AUTOSAR standard).
- Communication between SWCs on different simulated virtual ECUs is performed by the extended BSW modules for NoC communication (as introduced by TIMEA).

### D. Co-simulation Coordination

The co-simulation of the simulation models described in sections II-A, II-B and II-C results in the proposed simulation framework for AUTOSAR message-based multi-core processors. The co-simulation coordination is based on the FMI standard and a local coordinator for the coupling of the AUTOSAR simulation level and the NoC simulation level. Figure 2 depicts an architecture picture of the co-simulation. The TCP/IP protocol was selected for the communication between the simulation tools. The socket-based communication makes the co-simulation framework independent from the location of the simulation tools and their operating systems. With this approach an existing AUTOSAR-based system simulator is extended for supporting the simulation of virtual ECUs as application cores in a message-based MPSoC.

FMI is a tool-independent standard that provides support for both model exchange and co-simulation of dynamic models. These models are shipped as Functional-Mock-up Units (FMUs) containing an xml description file and the compiled application C-code. FMI defines interfaces for the data exchange and the time synchronization which are provided by the FMU and used by a master algorithm implemented by the hosting simulator. Although the FMI standard is not just limited to the automotive domain, it is currently highly used by the most popular AUTOSAR simulation tools (e.g. CANoe, AUTOSAR builder, VEOS, etc) [16] for the exchange of simulation models between different suppliers and OEMs. This allows us to introduce an interface according to FMI as a key part in the definition of the co-simulation coordination, using it for the synchronization between the simulation tools. However, since most of the on-chip simulators (e.g. GEM5 [17] and SystemC [18]) do not give support for FMI, the presented co-simulation coordination is not just limited to an FMI implementation, but also introduces additional coordination mechanisms for the synchronization of the on-chip simulator with the AUTOSAR simulation.

The FMI standard for co-simulation defines discrete communication points $tc_i$ for the data exchange between simulation models running in different simulation tools. Each communication point is known as a synchronization point between simulation models. The time between two consecutive communication points represents the communication step size $hc_i = tc_{i+1} - tc_i$. A communication step is defined as $tc_i \longrightarrow tc_{i+1} = tc_i + hc_i$. During this time the simulation of the models is performed independently in the different simulation tools.

In the presented co-simulation interface, we use the FMI standard for the coupling of the AUTOSAR simulation with the NoC simulation. A FMU is implemented as a wrapper and integrated into the AUTOSAR-based system simulator for the time synchronization with the NoC simulation model running in the on-chip simulator. Additionally, a local coordinator implemented in the on-chip simulator controls the NoC simulation and uses TCP for the communication with the FMU wrapper.

The AUTOSAR-based system simulator realizes the master functionality defined by the FMI standard. The FMU wrapper is controlled by the simulator using functions to synchronize the FMU simulation with the simulation of the

virtual ECUs and the environment models, proceeding in communication steps from the initial time $tc_0 = t_{start}$ to the finish time $tc_N = t_{stop}$. Likewise, the FMU wrapper uses the TCP connection to trigger the NoC simulation in the on-chip simulator. It is important to note that FMI functions for data exchange between the virtual ECUs and the NoC model are not provided by the defined FMU wrapper. The data exchange is realized directly between the virtual ECUs and the NoC simulation when a communication point is reached by the FMU wrapper. For this, different types of messages are defined to distinguish between the control flow and the data exchange of the simulation tools.

The following messages are used for the communication between the two simulation tools:

- ***Data Message.*** It represents a message sent from the virtual ECUs to the GEM5 local coordinator and vice versa. It contains the following fields:
  1) *Message Status:* This parameter indicates whether the data message is empty or the number of PDUs that it contains.
  2) *Sender ID:* This field contains the ID of the virtual ECU sending the message.
  3) *Receiver ID:* This field declares the destination virtual ECU ID.
  4) *SWC Port:* It indicates for which RTE signal the PDU is matched.
  5) *Payload:* It contains the user data.

  Empty data messages are needed because of the time-triggered behavior for the data exchange between simulation systems and the event-triggered execution of the simulation systems. Fields 4 and 5 represent a PDU. Fields 3, 4 and 5 are repeated in the same order depending on the number of PDUs that the data message contains.
- ***Synchronization Message.*** It is used for the time synchronization of both simulation systems. It is defined as follows:
  1) *Creation Time:* It represents the creation time of the synchronization message.
  2) *Indication:* In a synchronization message sent from the AUTOSAR-based system simulator this field indicates that a communication point was reached. Furthermore, if the synchronization message is sent from the NoC simulation this field indicates that the data exchange has finished and the next communication step can be performed.

The virtual ECUs implement the TCP connection for the data exchange with the NoC simulation. Once a communication point is reached in the AUTOSAR-based system simulation, a data message is sent by the NoC interface from each virtual ECU to the NoC simulation. In case of outgoing PDUs from the PDU router on the virtual ECU, these PDUs are integrated into the data message, otherwise the data message is configured to be empty using the message status. Additionally, an incoming data message from the NoC simulation is received by the NoC interface of each virtual ECU, which verifies its message status. If the data message

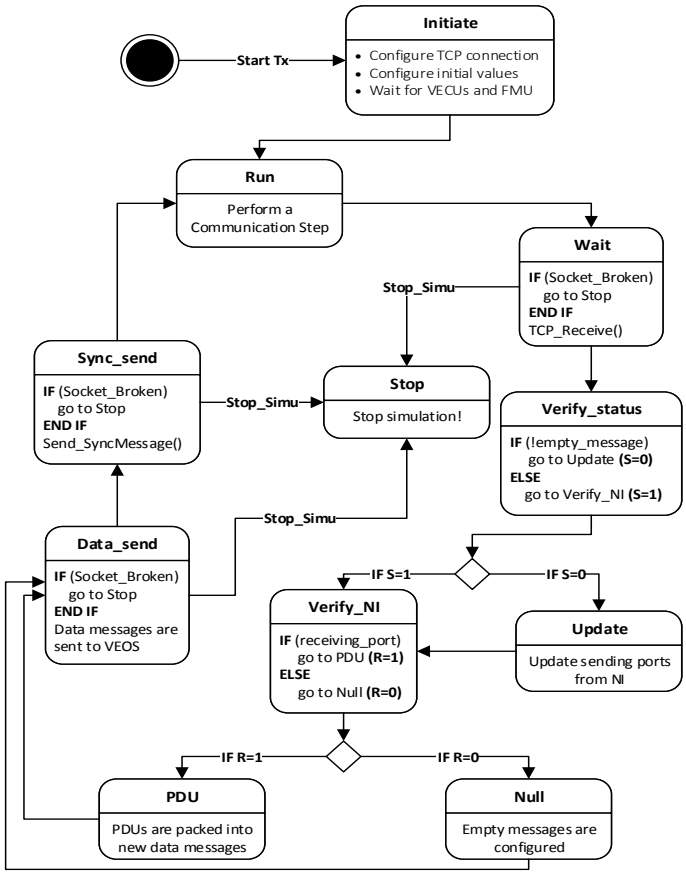


*Fig. 3: State Machine of the NoC Local Coordinator*

contains a PDU, this is made available to be used by the PDU router of the virtual ECU during the next communication step.

As mentioned before, the FMU wrapper uses the socket-based communication for controlling the NoC simulation. Whenever a communication point is reached on the AUTOSAR-based system simulation the FMU wrapper sends a synchronization message to the on-chip simulator and waits for its response. Once an incoming synchronization message from the NoC simulation is received by the FMU wrapper, the AUTOSAR-based system simulator performs the next communication step in the simulation.

Besides the FMU wrapper, the NoC local coordinator constitutes an essential part in the definition of the co-simulation interface. It is responsible for the synchronization of the NoC simulation. Moreover, it provides gateway functionalities to the NoC simulation for the data exchange with the virtual ECUs.

Let us discuss how the co-simulation coordination of the NoC model works using the state machine illustrated in figure 3. After starting the simulation in the *initiate* state, the local coordinator configures the parameters to establish the socket-based connection, configures the initial values for the NoC simulation and waits for the connection of the virtual ECUs and the FMU wrapper. Thereafter the local coordinator performs a communication step on the NoC simulation in the *run* state. In the *wait* state an incoming synchronization message from the FMU is waited for. Once a synchronization

message arrives, the status of the data messages received from the virtual ECUs is verified in the *verify_status* state. Thus, if no empty data messages were received, the respective ports of the NIs are updated in the NoC simulation model in the *update* state, otherwise the next state *verify_NI* is directly accessed. In this state, the receiver ports of the NIs are verified and, in case of new data, the *PDU* state is accessed wherein the data is packed into PDUs and these PDUs are stored into new data messages, otherwise these data messages are configured to be empty in the *null* state. In the *data_send* state data messages are sent to the respective virtual ECUs and thereafter a synchronization message is sent to the FMU wrapper in the *sync_send* state. Finally, the run state is accessed again and the whole procedure is repeated.

Furthermore, if the co-simulation is stopped by the AUTOSAR-based system simulator the stop state is accessed from *wait*, *data_send* or *sync_send* state. In this state the NoC simulation is stopped.

## III. Implementation of the Framework

In the implementation of the co-simulation framework, simulated virtual AUTOSAR ECUs are developed using the dSpace AUTOSAR solution tools [19]. SystemDesk is the dSpace software tool for modeling the architecture of automotive systems based on AUTOSAR SWCs and the interaction between these SWCs, defining AUTOSAR interfaces (e.g. sender/receiver) and AUTOSAR data prototypes handled by these interfaces [20]. Thereafter, the application behavior of each SWC is modeled using the dSpace AUTOSAR module TargetLink in combination with the Simulink-Stateflow graphical environment. Moreover, SystemDesk is used for the configuration of ECU instances based on the architecture illustrated in picture 1 and the generation of the simulated virtual ECUs for virtual validation scenarios.

Furthermore, the virtual ECUs and environment models are simulated with the VEOS simulation tool, while the NoC model is simulated using the GEM5 simulator.

### A. Simulation of Virtual AUTOSAR ECUs and Physical Environment

The VEOS simulator is used for the simulation of the virtual AUTOSAR ECUs and the natural environment. An architecture picture of the VEOS simulation system and its components is depicted in figure 4.

The SystemDesk software tool allows the generation of simulated virtual ECUs and the integration of these in an Offline Simulation Application (OSA) file. This OSA file is used by the VEOS simulator for the simulation of the virtual ECUs. Furthermore, VEOS allows to import environment models into the AUTOSAR simulation which either were generated as FMUs in compliance with the FMI standard for co-simulation or were generated using the Simulink production-coder. In the implementation of the framework we use Simulink for the generation of the physical environment.

Environment models and virtual ECUs are represented as Virtual Processing Units (VPUs) by the VEOS simulator. Virtual ECUs and environment models can be connected to each other using their VPU ports. VPU ports of the environment models are represented by Simulink input and output blocks, while VPU ports of the virtual ECUs are represented by the sensor/actuator SWC ports used by the environment interface module through the I/O hardware abstraction layer.

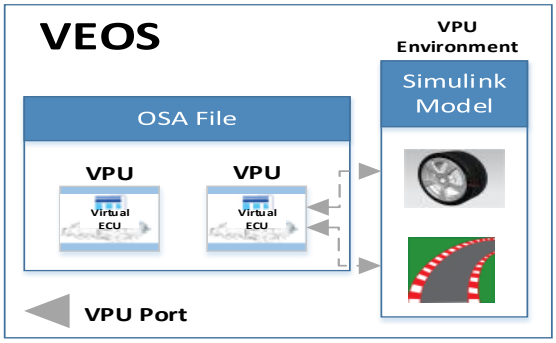


*Fig. 4: VEOS environment*

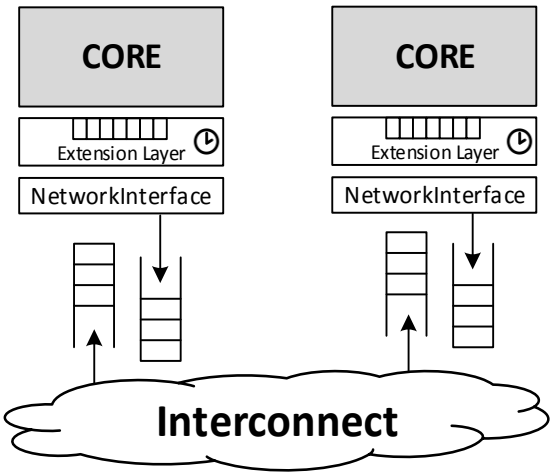


*Fig. 5: Simulated NoC in GEM5*

### B. Simulation System for NoC Simulation

The employed simulation environment is based on the GARNET interconnection network [21] inside GEM5. The simulated MPSoC models a classic five-stage pipelined router, including input buffers, routing logic, allocators and the crossbar switch, with virtual channel (VC) flow control at flit-level. It also supports both mentioned communication paradigms, i.e., TT and RC, using a time-triggered extension layer [22] which is added to the NI of GARNET (cf. Firgure 5).

In the simulated MPSoC, *Ports*[2] provide the interface to the NoC for the cores. They include a *data area*, *port configuration parameters* and *port status flags*. The data area stores the messages and can be a single buffer, which is overwritten in case of messages with state semantics, or a queue in case of messages with event semantics. The port configuration is implemented as a separate class `PortConfig` and is instantiated by CSV configuration files, once the simulation is started. The generated `PortConfig` object is afterwards used for the instantiation of the ports. This class (`PortConfig`)

[2]Different than the ports introduced in GEM5 memory system.

contains the parameters of the communication channel, e.g., the direction of the port, the path to the destination, temporal parameters such as the period and the phase for TT messages and the Minimum INter-arrival Time (MINT) for the RC messages.

Messages are stored at output ports by the sender core. They are dequeued into the NoC (considering the priorities and the schedule) by the extension layer. At the destination side, the messages are stored at input ports to be dequeued by the core. The core can invoke several API routines for sending and receiving the messages. These can be used by the simulated application or by the defined local coordinator which connects two simulation environments in this work.

The time-triggered behavior in GEM5 was technically realized by extending the `Consumer` class and adding a *Scheduled Wake-up* to this class. This class is a virtual base class of all classes that can be the targets of wakeup events. This change enables the classes inherited from `Consumer` to be waked-up not only by the linked consumer, but also by the scheduled wake-up (using `schedule` method).

The NI and the routers in GARNET handle the messages and flits of different types and priorities in the same manner, as the type and the priority of the messages are abstracted from them. This is the task of the scheduler to establish a collision-free communication of the TT messages and to guarantee no impact of the RC message on the TT messages due to contention at the resources (e.g., buffers at routers, physical links).

The scheduler triggers the injection of TT messages according to an *a priori* defined time-triggered communication schedule. In order to avoid collisions of messages at the NoC level, a scheduler uses the concept of *Timely Block*. Timely block guarantees the absence of collisions between TT and RC messages by blocking the injection of RC message during the *guarding windows*. The schedule for opening and closing instants are defined based on the time-triggered communication schedule.

### C. Coordination Implementation

In this section the implementation of the co-simulation coordination at each simulation system is described. The TCP/IP protocol is used by a server in the GEM5 simulation and clients in the VEOS simulation. The blocking socket-based communication is used to suspend and resume both simulation tools.

Since VEOS only allows a fixed communication step size ($hc_i = \kappa$) for the co-simulation of an FMU with an AUTOSAR simulation, its choice becomes in a key parameter for the implementation of the framework. Thus, the choice of the communication step size influences the accuracy and efficiency of the co-simulation framework. Increasing the communication step would reduce the synchronization overhead between the simulation environments and hence reduce the accuracy of their results whilst a short communication step would decrease the simulation performance hence making it less efficient. Since the RTE and the AUTOSAR BSW can only react to an event occurring in the NoC simulation within the interrupt detection latency, we use the minimum interrupt detection latency as a fixed communication step size for the co-simulation with GEM5. We assume $hc_i = 1\mu s$ as the minimum interrupt detection latency. The reason behind this assumption is that the RTE and most of the AUTOSAR BSW are designed for latency and bandwidth requirements that are orders of magnitude higher than $1\mu s$.

*1) FMU wrapper for NoC co-simulation:* VEOS allows the integration of FMUs for co-simulation of subsystem models which have been imported together with their solver and do not require additional tools for their simulation. In the presented co-simulation framework the FMU wrapper is implemented based on the FMI standard for co-simulation. Using FMI the VEOS simulator is able to control the FMU simulation. A TCP client is integrated into the FMU to establish the communication with the GEM5 simulator. This approach extends the VEOS simulator for co-simulation of FMUs requiring an extra simulation tool (in this case GEM5).

The initialization mode of the FMU wrapper is used to configure the socket-based communication. In the start function of the FMU, the socket-based communication parameters are configured to setup a TCP client connection with the GEM5 simulator.

In the step mode of the FMU wrapper the blocking mechanism provided by the socket-based communication is used to suspend and resume the VEOS simulation with synchronization messages. Whenever a communication point is reached, the function for the sending synchronization messages is called and thereafter the function that receives a synchronization message from the NoC simulation. Thus, the VEOS simulation is suspended till a synchronization message arrives from the GEM5 simulator. During this time the exchange of data messages between the virtual ECUs and the NoC simulation is performed. Once a synchronization message arrives, the FMU simulation is unblocked and the next communication step can be performed on the whole VEOS simulation.

*2) Local Coordinator for NoC simulation:* The local coordinator is realized in GEM5 using Python. It is implemented as a task controlling the NoC simulation based on the state machine presented in figure 3.

A TCP server is integrated into the local coordinator for the data exchange with the virtual ECUs and to establish the co-simulation coordination with the FMU wrapper. The blocking mechanism of the TCP communication is used in the same way that is implemented in the FMU wrapper in VEOS. Once a communication step was performed on the NoC simulation, the local coordinator invokes the function for receiving synchronization messages from the FMU wrapper, blocking temporally the GEM5 simulation. Thus, the GEM5 simulator is regularly suspended and the data exchange is possible during the co-simulation.

## IV. Use Case and Evaluation

An ABS use case serves for the evaluation of the co-simulation framework. The use case was modelled as an AUTOSAR-based system consisting of 4 virtual AUTOSAR ECUs, three of them hosting one SWC each and the other one

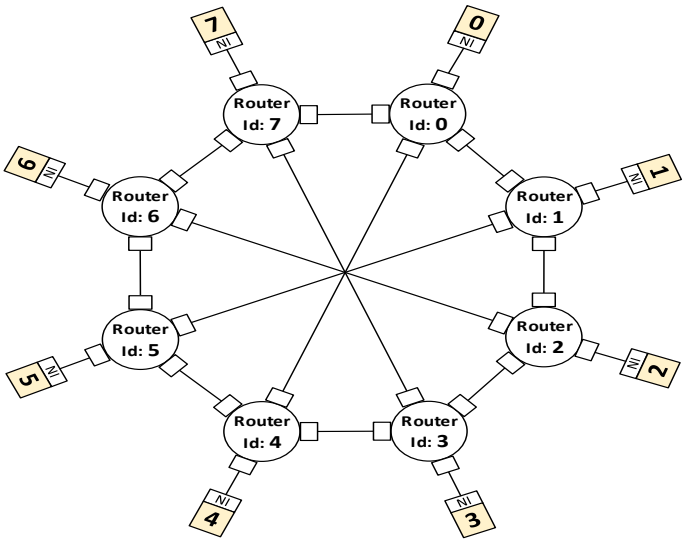


*Fig. 6: Spidergon Topology*

| NoC Configuration 2 | | | | | Results |
|---|---|---|---|---|---|
| **Message** | **Sender Core** | **Receiver Core** | **Message Size (bytes)** | **Temporal Configuration** | **Delay** |
| Angular speed | 0 | 1 | 60 | TT (period: 1ms) | 17ns |
| Braking force | 4 | 1 | 100 | RC (MINT: 3 ms) | 232ns |
| Relative slip | 1 | 3 | 80 | RC (MINT: 3 ms) | 232ns |
| Slip comparison | 3 | 4 | 70 | RC (MINT: 3 ms) | 217ns |

*TABLE I: NoC configuration*

hosting two SWCs. The virtual ECUs are implemented within an MPSoC with 8 cores in a Spidergon topology (see figure 6), where each virtual ECU is mapped to a single core in the MPSoC. This allows us to use different NoC configurations and to evaluate the ABS performance when mapping the virtual ECUs to different cores in the MPSoC. Additionally, since the minimum task period in the ABS application is $5ms$ the accuracy of the co-simulation is guaranteed having a communication step size of $hc_i = 1\mu s$.

A Simulink model was integrated into the VEOS simulation, which represents the human braking behavior, road characteristics, dynamics of the brake system hydraulic component and a physical wheel. In our simulation scenario a hard braking is implemented by the driver having an initial angular speed of $70.4 rad/sec$.

A simulation time of $18s$ was selected. Results are obtained using the experimental tool ControlDesk for the application behavior on the virtual ECUs and environment models, and GEM5 for the inter-core communication behavior. Table I presents the first NoC configuration implemented for the evaluation of our ABS muli-core system. In the configuration table, the sender and the receiver cores represent the cores which host the virtual ECUs.

In order to validate the correct operation of the simulation environment, the simulation scenario is analyzed having enabled and disabled the ABS multi-core processor. Figure 7 compares the wheel behaviors during the simulation scenario while figure 8 shows a braking distance reduction of $26m$ when the ABS is enabled. Also, the configuration table provides results of the communication behavior (end-to-end delay) of

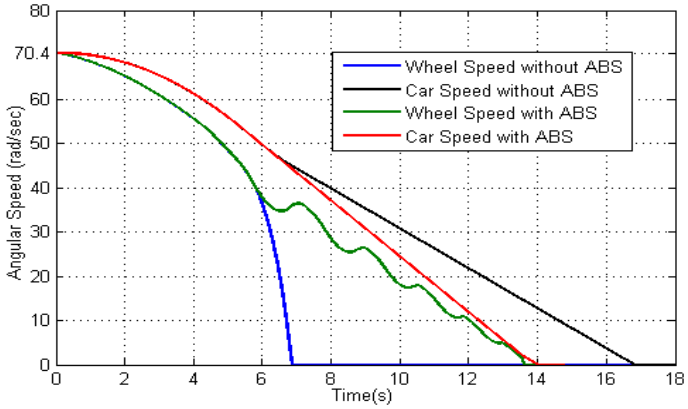


*Fig. 7: Car speed and wheel speed*

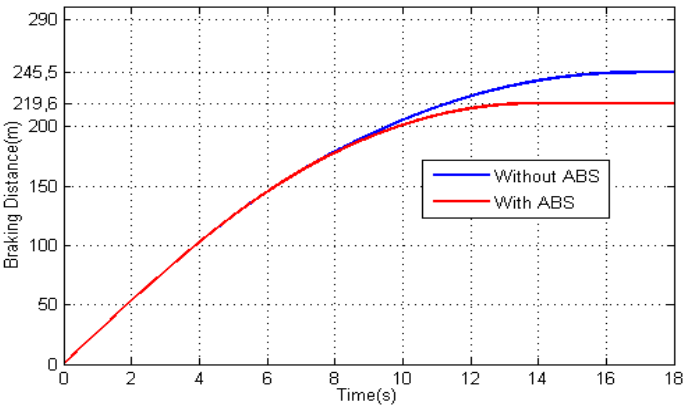


*Fig. 8: Braking distance*

each message sent through the NoC. The use case simulation took an overall of 5 minutes having the VEOS simulation on a 64-bit Windows PC with 4GB of RAM, while the Gem5 simulation was executed on a 64-bit Linux PC with 2GB of RAM.

Furthermore, different NoC configurations were implemented in the mapping of the virtual ECUs to different cores in the MPSoC in order to evaluate the influence of the NoC simulation on the ABS performance. Table II presents three different NoC configurations and their communication results. Figure 9 compares the angular speed of the wheel of 4 different NoC configurations, while figure 10 compares the different braking distances. These results demonstrate the ability of the simulation environment to evaluate the impact of different hardware choices (e.g, different NoC configurations) on the high-level system behavior.

## V. Discussion and Conclusion

Simulation environments are significant for the design exploration and early validation. In this paper a co-simulation framework supporting the integration of the AUTOSAR architecture with NoC architectures was presented. The framework consists of one AUTOSAR-based system simulator, for the simulation of the AUTOSAR software and the physical environment, and one on-chip simulator for the simulation of the NoC communication, implementing the FMI standard in combination with additional coordination mechanisms to

| NoC Configuration 1 | | | | | Results |
|---|---|---|---|---|---|
| **Message** | **Sender Core** | **Receiver Core** | **Message Size (bytes)** | **Temporal Configuration** | **Delay** |
| Angular speed | 7 | 4 | 60 | RC (MINT: 3 ms) | 232ns |
| Braking force | 5 | 4 | 100 | TT (period: 8ms) | 17ns |
| Relative slip | 4 | 1 | 80 | RC (MINT: 3 ms) | 264ns |
| Slip comparison | 1 | 5 | 70 | RC (MINT: 3 ms) | 217ns |
| **NoC Configuration 3** | | | | | **Results** |
| **Message** | **Sender Core** | **Receiver Core** | **Message Size (bytes)** | **Temporal Configuration** | **Delay** |
| Angular speed | 0 | 3 | 60 | RC (MINT: 3 ms) | 232ns |
| Braking force | 6 | 3 | 100 | RC (MINT: 3 ms) | 232ns |
| Relative slip | 3 | 2 | 80 | TT (period: 3ms) | 17ns |
| Slip comparison | 2 | 6 | 70 | RC (MINT: 3 ms) | 217ns |
| **NoC Configuration 4** | | | | | **Results** |
| **Message** | **Sender Core** | **Receiver Core** | **Message Size (bytes)** | **Temporal Configuration** | **Delay** |
| Angular speed | 3 | 4 | 60 | RC (MINT: 3 ms) | 217ns |
| Braking force | 1 | 4 | 100 | RC (MINT: 3 ms) | 232ns |
| Relative slip | 4 | 6 | 80 | RC (MINT: 3 ms) | 249ns |
| Slip comparison | 6 | 1 | 70 | TT (period: 12ms) | 32ns |

*TABLE II: Three different NoC configurations*

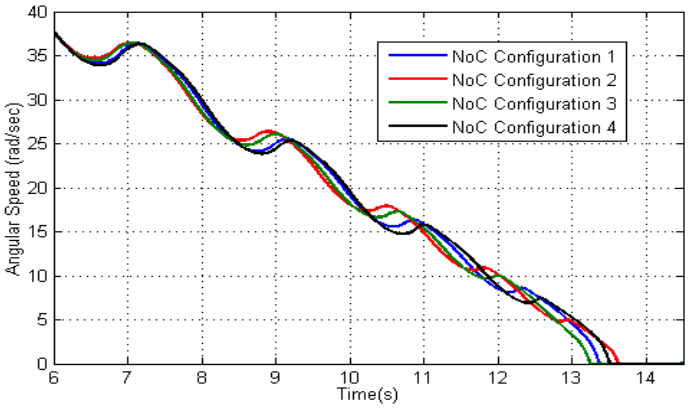


*Fig. 9: Wheel speed with different NoC configurations*

provide an interface for synchronization and data exchange between the two simulation tools. Furthermore, simulation building blocks for the extended AUTOSAR BSW for NoC communication were introduced. It was not the focus of this work to provide a fully quantitative analysis of the benefits obtained by the realization of AUTOSAR on a message-based MPSoC platform but to provide a co-simulation framework that can be used in conjunction with the available simulation tools.

The VEOS simulation tool for the AUTOSAR application

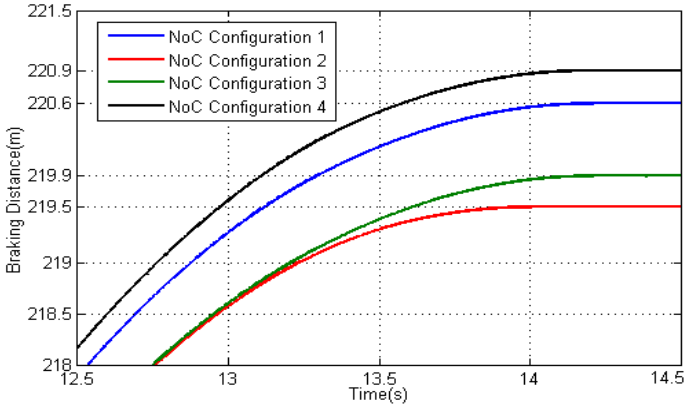


*Fig. 10: Braking distance with different NoC configurations*

and environment models, and the GEM5 simulator for the on-chip communication level were used in the implementation of the framework. A realistic automotive use case providing a virtual validation scenario demonstrates the ability of the framework for early validation of applications as well as its capability to analyze the performance and the real-time behavior of AUTOSAR applications on different multi-core platforms with message-based NoCs.

## REFERENCES


[1] *AUTOSAR Guide to Multi-Core Systems, AUTOSAR Release 4.2*, AUTOSAR, 2014.

[2] F. Poletti, A. Poggiali *et al.*, "Energy-Efficient Multiprocessor Systems-on-Chip for Embedded Computing: Exploring Programming Models and Their Architectural Support," *Computers, IEEE Transactions on*, 2007.

[3] R. Obermaisser, H. Kopetz *et al.*, "A Cross-Domain Multi-Processor System-on-a-Chip for Embedded Real-Time Systems," *IEEE Transactions on Industrial Informatics*, 2010.

[4] J. Leverich, H. Arakida *et al.*, "Comparing Memory Systems for Chip Multiprocessors," in *Proceedings of the 34th Annual International Symposium on Computer Architecture*, ser. ISCA '07. New York, NY, USA: ACM, 2007. [Online]. Available: http://doi.acm.org/10.1145/1250662.1250707

[5] M. Rudorfer, T. Ochs *et al.*, "Realtime system design utilizing AUTOSAR methodology," in *elektroniknet*, 2009.

[6] B. Huber and R. Obermaisser, "An ARTEMIS Cross-Domain Embedded System Architecture and Its Instantiation for Real-Time Automotive Applications," in *Proceedings of the 30th IFAC Workshopon Real-Time Programming and 4th International Workshop on Real-Time Software*, Poland, 2009.

[7] M. Urbina and R. Obermaisser, "Multi-Core Architecture for AUTOSAR based on Virtual Electronic Control Units," in *Emerging Technologies and Factory Automation 2015. IEEE Conf. on*.

[8] Z. Zhang, E. Eyisi *et al.*, "Co-simulation framework for design of time-triggered cyber physical systems," in *Cyber-Physical Systems, 2013 ACM/IEEE Int. Conf. on*.

[9] Z. Owda, M. Abuteir *et al.*, "Co-simulation framework for networked multi-core chips with interleaving discrete event simulation tools," in *Emerging Technologies Factory Automation (ETFA), IEEE 20th Conf. on*, 2015.

[10] B. Muller-Rathgeber and H. Rauchfuss, "A Cosimulation Framework for a Distributed System of Systems," in *Vehicular Technology Conf., 2008. IEEE 68th*.

[11] Z. Fang, Q. Min *et al.*, "Transformer: A functional-driven cycle-accurate multicore simulator," in *Design Automation Conf., 2012*.

[12] M.-C. Chiang, T.-C. Yeh *et al.*, "A QEMU and SystemC-Based Cycle-Accurate ISS for Performance Estimation on SoC Development," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 2011.

[13] M. Urbina, Z. Owda *et al.*, "Simulation Environment based on SystemC and VEOS for Multi-Core Processors with Virtual AUTOSAR ECUs," in *Dependable, Autonomic and Secure Computing 2015. IEEE Conf. on*.

[14] T. Blochwitz, M. Otter *et al.*, "The Functional Mockup Interface for Tool independent Exchange of Simulation Models," in *Proc. of the 8th Int. Modelica Conf.*, 2011.

[15] Editted by: Roman Obermaisser, *Time-Triggered Communication*, ser. Embedded Systems. USA: CRC Press, 2012.

[16] "FMI Support in Tools." [Online]. Available: https://www.fmi-standard.org/tools

[17] N. Binkert, B. Beckmann *et al.*, "The GEM5 Simulator," *SIGARCH Comput. Archit. News*. [Online]. Available: http://doi.acm.org/10.1145/2024716.2024718

[18] "NOXIM : The NoC Simulator." [Online]. Available: www.noxim.org

[19] *Virtual Validation Overview, Release 2014-B*, dSpace, 2014.

[20] *AUTOSAR Layered Software Architecture, AUTOSAR Release 4.2*, AUTOSAR, 2014.

[21] N. Agarwal, T. Krishna *et al.*, "Garnet: A detailed on-chip network model inside a full-system simulator," in *Performance Analysis of Systems and Software, 2009. IEEE Int. Symposium on*, April.

[22] H. Ahmadian and R. Obermaisser, "Time-triggered extension layer for on-chip network interfaces in mixed-criticality systems," in *Digital System Design DSD 2015*.