Evaluation of Performance and Fault Containment in AUTOSAR Micro-ECUs on a Multi-Core Processor

Moisés Urbina and Roman Obermaisser
Institute of Embedded Systems - University of Siegen
Email:{moises.urbina, roman.obermaisser}@uni-siegen.de

***Abstract*—The AUTOSAR standard does not provide an approach for the mapping of its ECU software architecture to a message-based multicore system. In this work we present an analysis of performance and fault containment of a novel TIme-triggered MEssage-based multi-core architecture for AUTOSAR (TIMEA). The TIMEA platform is intended to bring the advantages of network-on-a-chip architectures to the AUTOSAR software, which lead to multiple benefits for fail operational real time systems such as temporal predictability and fault isolation.**

We introduce a fault hypothesis consisting of multiple fault assumptions and the definition of the fault containment regions and we describe the algorithms for the integration of a multicore monitoring service into the AUTOSAR Basic Software. A set of experiments were carried out to evaluate the performance of the system using an anti-lock braking use case in a simulation scenario under failure occurrences. The obtained results demonstrate how the TIMEA platform remains operational in the presence of failures.

I. INTRODUCTION

Since 2012 the automotive industry introduced the use of Multi-Processor System-on-a-Chips (MPSoCs) for the development of automotive embedded systems with the publication of the version 4 of the AUTomotive Open System ARchitecture (AUTOSAR) standard. The last release of the AUTOSAR platform (4.3) defines a multicore operating system [1], where the functionalities provided by the Basic Software (BSW) are separated in multiple master and slave modules to support AUTOSAR Software Components (SWCs) in the application layer on different cores. Thus, in order to allow the communication between the master and the slave BSW modules, and between SWCs located in different cores, an Inter OS-application Communicator (IOC) was introduced as a new service in the BSW of each core using a shared memory as the medium for the inter-core communication.

In previous work [2], a novel AUTOSAR message-based multicore architecture was presented which combines the AUTOSAR software with a multi processor Network-on-Chip (NoC) platform. The so-called TIMEA (TIme-triggered MEssage-based multi-core platform for AUTOSAR) defines a message-based NoC as the only physical medium for the communication between the cores and introduces autonomous application cores which function as AUTOSAR Micro-Electronic Control Units (μECUs) on the MPSoC. Each μECU acts as a unit of abstraction where the SWCs are provided with a Run-Time Environment (RTE) and a lightweight implementation of the AUTOSAR BSW, exploiting the well known advantages of message-based NoCs for real time multicore systems over a shared memory approach [3] [4] (e.g., fault isolation, temporal predictability).

In this work, we carry out an evaluation of the performance and fault containment of the TIMEA platform by implementing an anti-lock braking use case on a multicore simulation environment [5]. Additionally, a health monitoring service was added to the BSW of the μECUs, which provides recovery solutions to the AUTOSAR system in case of SWC failures. To the best of our knowledge, this represents the first published evaluation of the AUTOSAR software running on a message-based multicore Electronic Control Unit (ECU).

II. MESSAGE-BASED MULTI-CORE ARCHITECTURE FOR AUTOSAR

In TIMEA [2] [6], a message-based multi-core architecture is proposed for the mapping of the single-core AUTOSAR ECU software architecture to a message-based multi/many-core system. As depicted in figure 1, this architecture introduces a message-based NoC as the communication interface between the cores. Two types of cores can be distinguished: AUTOSAR μECUs and system cores. While μECUs are in charge of the execution of the automotive software, system cores perform hardware acceleration services needed by the μECUs, as for example I/O services [7] or off-chip network services [8].

A. Message-Based Network-on-Chip

In TIMEA, the message-based NoC provides temporal predictability and fault isolation to the AUTOSAR μECUs and the system cores. We assume that the NoC provides support for multiple timing models to fulfill the different and partially contradicting communication requirements of different application subsystems in real-time embedded systems such as those in the automotive industry. Different available NoCs satisfy this assumption such as AEtheral [9] and DREAMS-NoC [10].

Time-Triggered (TT) communication: this kind of communication specifies messages whose transmission times are stored in a static communication schedule. Time-triggered messages ensure temporal predictability since resource conflicts with other time-triggered messages are eliminated at development time.

Rate-Constrained (RC) communication: A sufficient bandwidth allocation for each message transmission is guaranteed with a defined Minimum INter-arrival Time (MINT) and temporal deviations. Rate-constrained messages provide flexibility and a high utilization of resources. Priorities are used to resolve contention between multiple rate-constrained messages and contention with time-triggered messages. Time-triggered messages are assigned higher priority than rate-constrained ones.

Best-Effort (BE) communication: It supports the transmission of messages that are triggered by the occurrence of significant events in the environment or inside the system. Priorities are also used to resolve contention but in comparison with time-triggered and rate constrained messages, this kind of messages are assigned lower priority.

The message-based NoC is composed of Network Interfaces (NIs) and routers. NIs serve as the interfaces for the μECU and the system cores to access the NoC by injecting the messages from the cores into the NoC as well as delivering the received messages from the NoC to the cores. Routers relay the fixed-length fractions of messages from the sender NI to the destination NI. Physical links serve for the interconnection between the NIs and the routers.

B. AUTOSAR Micro-ECUs

The μECUs host the AUTOSAR SWCs and they are in charge of the automotive application software. As depicted in Figure 1, they are provided with a RTE and a simplified implementation of the AUTOSAR BSW, where BSW modules for I/O functionalities and off-chip communication are replaced by interfaces to the system cores.

Extended BSW modules are required in order to provide support for on-chip communication to the μECUs. The NI is accessed by the Micro-Controller Abstraction Layer (MCAL) of the μECUs providing sending and receiving ports to allow the access to the message-based NoC. A COM service module and a PDU router are added to the communication service layer to store RTE signals in on-chip Protocol Data Units (PDUs) and to map these to one of the communication models supported by the NoC (time-triggered messages, rate-constrained messages and best-effort messages).

Moreover, an I/O proxy module serves for the exchange of messages between the sensor/actuator SWCs and an I/O system core [7]. This module acts as an interface forwarding data handled by sensor/actuator SWC ports to the PDU router, so these SWCs do not need to be aware whether the I/O functionality is executed locally or by a dedicated system core. Furthermore, a health monitoring service module is integrated which enables fault tolerance in the AUTOSAR application. This module provides error detection mechanisms based on AUTOSAR pre-defined parameters, e.g., data constraint element and unit specifications, and recovery mechanisms in case of the failure of a SWC. In addition, BSW modules for memory services are also integrated, which allow the μECU to store data to a local memory.

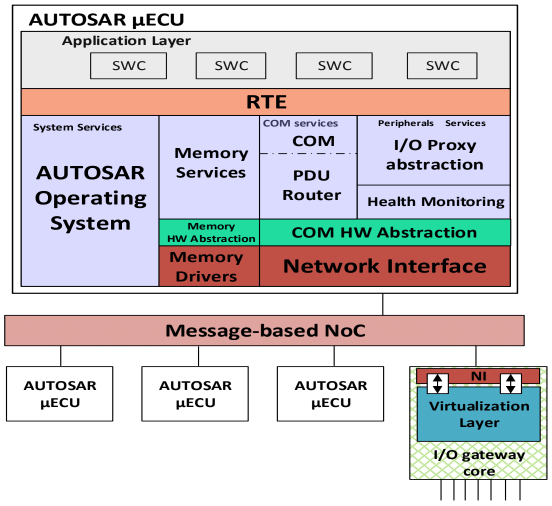


Fig. 1: TIMEA Architecture

The AUTOSAR Virtual Funcitonal Bus (VFB) is kept transparent to the SWCs through a communication hierarchy that is established as follows:

- ***Inner-core communication:*** the exchange of messages between SWCs on the same μECU is performed by the RTE of the μECU.
- ***Inter-core communication:*** the communication between SWCs on different μECUs in the same MPSoC is available through the extended COM service modules for on-chip communication and the NoC, based on the pre-defined on-chip communication schedule of the network.

III. Fault Containment Regions & Failure Assumptions

In this section we define the fault hypothesis for the TIMEA platform. This fault hypothesis specifies the assumptions about the kinds of failures that system components may experience and, as a consequence, could lead to the failure of the entire system. Based on these failure assumptions, fault tolerance mechanisms are defined. In case a potential fault occurs, which has not been included in the failure assumptions, the complete system might fail. The fault hypothesis is comprehended by the specification of the Fault Containment Regions (FCRs) and the failure mode assumptions [12].

A. Fault Containment Regions

The FCRs represent isolated units of failures that operate independently from each and perform correct outputs regardless of any arbitrary logical or electrical fault that may occur in the system but outside of their pre-delimited regions. This independence of the FCRs can be compromised at the design time and at runtime if something happens that was not assumed as part of the fault assumptions (e.g., massive explosion).

We determine AUTOSAR SWCs and AUTOSAR μECUs as FCRs for the AUTOSAR multicore architecture. In each μECU, the AUTOSAR software architecture guarantees a high level of fault independence between SWCs which is assured by the AUTOSAR operating system with memory

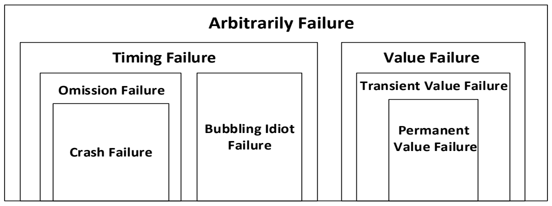


Fig. 2: Failure Modes

protection and temporal partitioning (task scheduler). Furthermore, μECUs are identified as independent units of failures due to the employment of the message-based NoC for the communication between each other, which provides fault isolation to each μECU. Thus, two layers of defense can be distinguished: ***(1)*** SWCs as FCRs with lower containment coverage (only addressing software faults) ***(2)*** μECUs as FCRs with higher containment coverage (also addressing hardware faults).

B. Failure Mode Assumptions

Assumptions of the failures are determined independently of the actual cause of the failure (e.g., logical error, hardware error) but from the perspective of the service user and can be classified as a failure in the time domain or in the value domain (see figure 2). Based on these assumptions, different failure modes are specified which allow to define the degree of redundancy required to provide correct error processing. For the development of the TIMEA platform the following failure assumptions are considered.

1) Time Domain:

- ***Timing Failure.*** A timing failure is a kind of failure wherein a FCR does not comply with its temporal specification. The FCR outputs are delivered too early or too late. In case no prior knowledge about the message periods (time-triggered messages) or the MINT (rate-constraint messages) is available to the system, the detection of a timing failure is not possible.
  A time-triggered SWC that does not send a message according to its temporal run-time configuration would be an example of a timing failure.
- ***Omission Failure.*** This kind of failure represents a transient failure, where a FCR fails sending a message at its respective time slot and, as a consequence, the receiver FCR does not respond to any input. In contrast to a general timing failure in an omission failure the output is never delivered by the FCR.
- ***Crash Failure.*** In contrast to the omission failures, a crash failure represents a permanent failure that can remain undetected by the system. In case a specific FCR is experiencing a crash failure this FCR would stop producing any outputs. In case a SWC suffers a crash failure this SWC would stop sending messages.
- ***Babbling Idiot Failure.*** This failure represents a special kind of timing failure where a FCR starts sending numerous messages without a minimum time interval between each other, so the communication medium is monopolized by the FCR.
  An example would be an AUTOSAR SWC which constantly sends messages generating delays because of contention in the communication network.

2) Value Domain:

- ***Transient/Permanent Value Failure.*** It represents a kind of failure where the content of the message sent by an FCR does not comply with the specification. In case the failure has a sporadic behavior it is called a transient value failure, otherwise if the failure remains it is called a permanent value failure.
  A SWC which sends random message values that do not comply with its AUTOSAR specification (e.g., data constraint element, constant specifications) would be an example of a value failure.

IV. Monitoring Service & Fault Tolerance Mechanisms

In this section we explain the functionality of the introduced Health Monitoring Module. As described in section II, the health monitoring module serves to detect software failures in the time or value domain and to enable recovery solutions in the application software. This module consists of a health monitoring algorithm and a list of recovery actions that cover the kinds of failures described by the fault hypothesis.

In the TIMEA architecture, we use spare SWCs which are activated upon the detection of a fault. This means, safety-critical SWCs are developed with multiple implementations that can be activated in case the pre-set SWC fails. For this, callback functions are used to resume (at run time) operating system tasks that host the replicated SWC. As defined by AUTOSAR [1], callback functions provide the capability to trigger or stop SWCs that are outside of the AUTOSAR BSW. Thus, a replica of the SWC does not increase the operating system overhead, since the tasks are set up in suspended state [1] as defined at compile time.

Omission and crash failure recognition is processed by the health monitoring module using algorithm 1, which defines the crash failure processing procedure. Moreover, for the handling of transient and permanent value failures the value failure processing procedure is presented in algorithm 2. Also, algorithm 1 defines the procedure for activating replicas, which is used by both algorithms for the exploitation of the SWC redundancy.

A. Health Monitoring Algorithm

The following sets and entities are used to explain the health monitoring algorithm:

- $RTEswc_{ij}$ = This represents the obtained status of a SWC data/argument prototype returned by its specific RTE function, with $i \in [1, I]$ and $j \in [1, J]$, where I is the total number of AUTOSAR SWCs in the μECU, i is the ID of the requester SWC, J represents the total number of data/argument prototypes handled by the SWC and j is the data/argument prototype ID.
- $Dswc_{ij}$ = It represents a data/argument prototype of a specific SWC. Each $Dswc_{ij}$ is comprehended by a

data constraint element $DC_{ij} \in [DCij_{min}, DCij_{max}]$ and has a specific unit $Unit_{ij}$, both set up by the data/argument prototype AUTOSAR specification.

- ρ_i = This threshold parameter represents the maximum number of consecutive omission failures of a single SWC that can be tolerated by the μECU.
- ϕ_i = This threshold parameter represents the maximum number of consecutive value failures of a single SWC that can be tolerated by the μECU.
- φ_i = This threshold parameter represents the maximum number of omission failures of a single SWC that can be tolerated by the μECU within an interval of κ_i executions.
- γ_i = This threshold parameter represents the maximum number of value failures of a single SWC that can be tolerated by the μECU within an interval of c_i executions.
- N_i = It represents the actual number of consecutive omission failures of a single SWC.
- R_i = It represents the actual number of consecutive value failures of a single SWC.
- n_i = It represents the actual number of omission failures of a single SWC within an interval of κ_i executions.
- r_i = It represents the actual number of value failures of a single SWC within an interval of c_i executions.
- $SDswc_{ij}$ = This variable serves to indicate whether the data/argument prototype complies with its AUTOSAR specification.
- $Rswc_i$ = It represents the SWC replica of a safety-critical SWC$_i$.
- ξ_i = It represents the actual number of executions of a single SWC within an interval of κ_i executions.
- χ_i = It represents the actual number of executions of a single SWC within an interval of c_i executions.

The following expressions are used to determine the presence of a crash failure occurrence and a permanent value failure occurrence.

$$\begin{cases} N_i \geqslant \rho_i + 1 \text{ Crash failure detected} \\ N_i < \rho_i + 1 \text{ No crash failure present} \end{cases} \tag{1}$$

Permanent Omission Failures

$$\begin{cases} n_i \geqslant \varphi_i + 1 \text{ Crash failure detected} \\ n_i < \varphi_i + 1 \text{ No crash failure present} \end{cases} \tag{2}$$

Multiples Omission Failures within κ_i executions

$$\begin{cases} R_i \geqslant \phi_i + 1 \text{ Permanent value failure detected} \\ R_i < \phi_i + 1 \text{ No permanent value failure} \end{cases} \tag{3}$$

Permanent Value Failures

$$\begin{cases} r_i \geqslant \gamma_i + 1 \text{ Pemanent value failure detected} \\ r_i < \gamma_i + 1 \text{ No permanent value failure} \end{cases} \tag{4}$$

Multiple Value Failures within c_i executions

Once a safety critical SWC runnable is triggered by the AUTOSAR operating system, the status of its specific RTE function for the internal communication is observed by the integrated health monitoring functionality. This functionality uses the algorithm 1 for monitoring the status value of the RTE function.

As explained earlier, algorithm 1 is responsible for the recognition and processing of crash failures. The procedure "$CrashFailureRecognition(RTEswc_{ij})$" is called for each safety critical SWC triggered by the operating system. This procedure distinguishes between two kinds of crash failure possibilities: ***(1)*** Reaching a pre-defined threshold of consecutive omission failures ρ_i tolerated by the μECU, ***(2)*** Reaching a pre-defined threshold of total omission failures φ_i tolerated by the μECU within an interval of κ_i.

The algorithm starts updating the ξ_i counter since a new SWC execution was performed. This counter is checked every single execution and serves to keep the n_i counter updated within a delimited interval of κ_i executions. In case ξ_i reaches κ_i and no crash failure was recognized, this counter, as well as the n_i counter, are set to 0.

The passed RTE function status $RTEswc_{ij}$ is checked for the recognition of omission failure occurrences. In case $RTEswc_{ij}$ is $True$, the N_i counter is set to 0 since no consecutive omission failure was detected and the "$ValueFailureRecognition(Dswc_{ij})$" procedure of algorithm 2 is called for value failure processing. This procedure will be explained later for algorithm 2.

In case the $RTEswc_{ij}$ is $False$, this means that an omission failure occurred and therefore the N_i and n_i counters are increased. After this, the expression 1 serves for determining the existence of a crash failure due to consecutive omission failures and, based on this assumption, the "$ActivateReplica(Rswc_i)$" procedure is called if a crash failure was detected. This procedure is detailed later.

If no crash failure was determined by the expression 1, then expression 2 is used for determining the existence of a crash failure due to multiple omission failures within an interval of κ_i executions. At this moment, in case no crash failure was detected, the value failure recognition is initiated and the "$ValueFailureRecognition(Dswc_{ij})$" procedure of algorithm 2 is called.

The algorithm 1 also defines the "$ActivateReplica(Rswc_i)$ procedure. This procedure implements callback functions to resume the tasks running $Rswc_i$ and pause the tasks hosting the original SWC_i. Callback functions are implemented according to the $Rte\_Call\_<p>\_<o>$ API [11] of the RTE in order to enable safe configuration of the AUTOSAR services.

The algorithm 2 is in charge of the processing of the value failures. It defines the $ValueFailureRecognition(Dswc_{ij})$ procedure, which serves to distinguish a permanent value failure from a transient value failure. Alike the crash failure processing, permanent value failures are differentiated in two types: ***(1)*** Reaching a pre-defined threshold of consecutive value failures ϕ_i tolerated by the μECU, ***(2)*** Reaching a pre-defined threshold of total value failures γ_i tolerated by the μECU within an interval of c_i executions.

The permanent value failure recognition starts by updating the χ_i counter. Comparable to the "$CrashFailureRecognition(RTEswc_{ij})$" procedure, the χ_i counter is checked every single execution to keep the r_i counter updated within a delimited interval of c_i executions. In case χ_i reaches c_i and no permanent value failure was recognized, this counter, as well as the r_i counter, are set to 0.

The passed data/argument prototype $Dswc_{ij}$ is checked for the recognition of a transient value failure. For this, the data constraint element and the data unit linked to this data/argument prototype are used. Since the data/argument prototype must be comprehended by its data constraint element, it is compared with the maximum and minimum values of the constraint element. Additionally, the data/argument prototype unit is compared with its original specification. In case $Dswc_{ij}$ conforms to its AUTOSAR specification, $SDswc_{ij}$ is set to $True$ (no value failure), otherwise it is set to $False$. If a transient value failure occurred, the R_i and r_i counters are increased and expressions 3 and 4 are used to establish the presence of a permanent value failure. When a permanent vailue failures is recognized, the "$ActivateReplica(Rswc_i)$ procedure is called to execute the replica $Rswc_i$ and to turn off the original SWC.

Algorithm 1 HM Algorithms - Crash Failure Processing

```
1:  procedure $CrashFailureRecognition(RTEswc_{ij})$
2:      $\xi_i = \xi_i + 1$
3:      if $RTEswc_{ij} == True$ then
4:          $N_i = 0$
5:          if $\xi_i == \kappa_i$ then
6:              $\xi_i = 0$
7:              $n_i = 0$
8:          end if
9:          $ValueFailureRecognition(Dswc_{ij})$
10:     else
11:         $N_i = N_i + 1$
12:         $n_i = n_i + 1$
13:         if $N_i \geqslant \rho_i + 1 \ \| \ n_i \geqslant \varphi_i + 1$ then
14:             $ActivateReplica(Rswc_i)$
15:         else
16:             if $\xi_i == \kappa_i$ then
17:                 $\xi_i = 0$
18:                 $n_i = 0$
19:             end if
20:         end if
21:     end if
22: end procedure
23: procedure $ActivateReplica(Rswc_i)$
24:     $RTE\_SWC_i(OFF)$
25:     $RTE\_Rswc_i(ON)$
26: end procedure
```

Algorithm 2 HM Algorithms - Permanent Value Failure Processing

```
1:  procedure $ValueFailureRecognition(Dswc_{ij})$
2:      $\chi_i = \chi_i + 1$
3:      if $Dswc_{ij} < DCij_{min} \ \| \ Dswc_{ij} > DCij_{max} \ \|$
        $Dswc_{ij}.Unit \neq [Unit_{ij}]$ then
4:          $SDswc_{ij} = False$
5:      else
6:          $SDswc_{ij} = True$
7:      end if
8:      if $SDswc_{ij} == False$ then
9:          $R_i = R_i + 1$
10:         $r_i = r_i + 1$
11:         if $R_i \geqslant \phi_i + 1 \ \| \ r_i \geqslant \gamma_i + 1$ then
12:             $ActivateReplica(Rswc_i)$
13:         else
14:             if $\chi_i == c_i$ then
15:                 $\chi_i = 0$
16:                 $r_i = 0$
17:             end if
18:         end if
19:     else
20:         $R_i = 0$
21:         if $\chi_i == c_i$ then
22:             $\chi_i = 0$
23:             $r_i = 0$
24:         end if
25:     end if
26: end procedure
```

B. Fault Tolerance Mechanisms

The TIMEA platform provides a set of fault tolerance mechanisms comprising recovery solutions to counteract the impact of the pre-defined failure modes. Additionally, a single-fault hypothesis is selected, which means the maximum number of failures is 1. The maximum number of failures represents the maximum number of FCR failures that the system may handle at the same time. A single failure occurrence is the prevalent assumption in many present-day safety-critical systems [12] (e.g., TTA, FlexRay [13]).

Since a message-based NoC is used for the communication between the cores, communication delays due to the occurrence of a babbling idiot failure can be avoided in safety-critical SWCs. These SWCs use time-triggered messages with pre-defined network scheduling, thus quarantining the core faults since they are isolated by the communication network. Thus, in case a SWC starts sending unlimited messages to the network no delays are perceived at other μECUs.

In case of omission/crash failures, the defined monitoring service allows the TIMEA platform to provide recovery actions based on SWC redundancy on the μECU level. Thus, supposing a safety-critical SWC crashes, the error detection algorithm would determine the presence of a crash failure and the replica of the SWC would be executed.

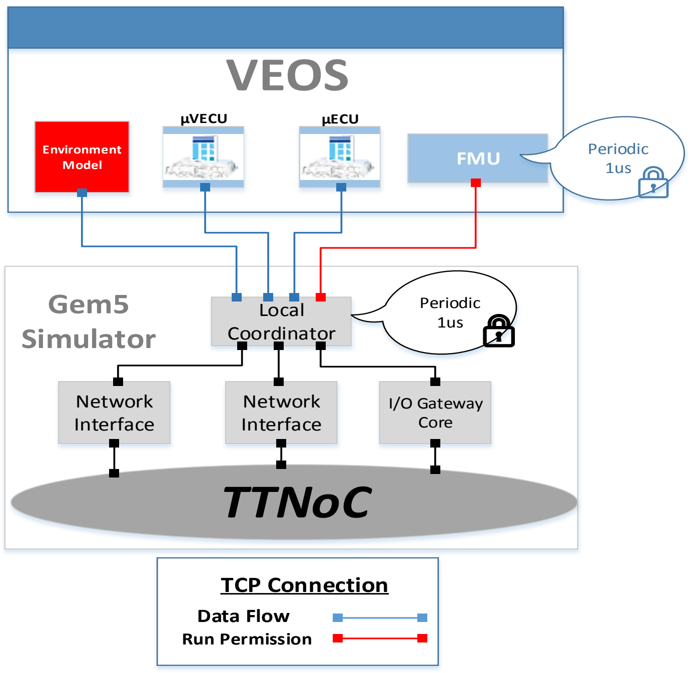


Fig. 3: Co-simulation Platform for TIMEA

For timing failures of the AUTOSAR software, we expect time-triggered messages in the NoC to not be affected. This means, in the scenario that a safety-critical SWC does not obey its temporal specification and delays the sending of a message, the resulting communication jitter would not be affected due to the intrinsic fault isolation property of time-triggered messages.

In the same way that omission/crash failures are prevented, the monitoring service avoids the propagation of value failures through the NoC. For example, in case a safety-critical SWC starts sending messages whose values are not complying with its data constraint element specification, the monitoring service would recognize the fault and use SWC redundancies on the μECU to prevent the persistence of the failure. The data constraint element [14] is an AUTOSAR system specification used to restrict the range of possible values for the data handled by a SWC.

The choice of the defined threshold parameters for crash failure and permanent value failure recognition depends on the specific design and implementation use case.

V. Implementation & Evaluation

Simulation environments are frequently used for the development of automotive electronic systems since they provide a framework for early performance and reliability examination of the platforms under fault occurrences using fault injection. For this reason, we propose the use of the co-simulation framework presented in [5] for the evaluation of the AUTOSAR multi-core platform (see figure 3). This environment consists of the AUTOSAR simulator VEOS, for the simulation of the AUTOSAR software, and the multi-core simulator GEM5 [15] for the simulation of the message-based NoC communication.

Simulated AUTOSAR ECUs running on VEOS are configured according to the software architecture of figure 1, so they serve as the AUTOSAR μECUs on the TIMEA platform. The dSpace AUTOSAR tools SystemDesk and TargetLink are used for the development and the generation of the simulated AUTOSAR μECUs as described in [16]. Additionally, the GARNET interconnection network [17] inside GEM5 is used for the simulation of the NoC.

To support all mentioned communication paradigms (i.e., best-effort, time-triggered and rate-constrained), a Time-Triggered Extension Layer (TTEL) was added to the regular NI of GARNET. This extension layer takes care of the timely injection of the messages and priorities of the time-triggered and rate-constrained. The detailed architecture of the TTEL is elaborated in [10].

The co-simulation technique presented in [5] is used for the coupling of both simulation tools. This technique provides an interface that combines the Functional Mock-up Interface (FMI) standard [18] with additional coordination mechanisms for the synchronization and the exchange of data between the two simulators, where each AUTOSAR μECU is mapped to its corresponding NI on the NoC simulation.

Additionally, for the evaluation of the TIMEA platform and the presented fault tolerance mechanisms, RTE interventions are used to perform fault injections during a simulation scenario. The RTE interventions are defined by the AUTOSAR standard for the testing of ECU code allowing to access the RTE internal communication of sender-receiver and client-server interfaces in order to read and to modify the data elements and operation arguments transmitted by the interfaces. Also, the status return values of RTE API functions can be modified using RTE interventions. Thus, during a simulation SWC ports can be stimulated or error states can be injected to test the behavior of the SWCs.

A. *Use Case*

As a realistic automotive use case, we implemented an Anti-lock Braking System (ABS) consisting of five AUTOSAR SWCs distributed on 4 AUTOSAR μECUs and a Light Indicator System (LIS) of four SWCs distributed on 3 μECUs. Both application systems are implemented within a multi-core platform of 8 cores in Spidergon topology, where each μECU is mapped to a single core in the MPSoC.

Figure 4 presents the described use case. The $\mu ECU4$ contains an AUTOSAR SWC performing the controller of the ABS functionality. Additionally, SWC redundancy is implemented in the $\mu ECU4$. Thus, two different SWC implementations of the controller are used, one based on the open source ABS simulink implementation of MathWorks [19] adapted to an AUTOSAR SWC (**Math_SWC**), and one obtained from the dSpace solution for an AUTOSAR ABS implementation (**dSpace_SWC**). We set the threshold parameters for crash and permanent value failure recognition on $\mu ECU4$ with a granularity of $\rho = 3$, $\phi = 6$, $\varphi_i = 6$, $\gamma_i = 12$, $\kappa_i = 50$ and $c_i = 50$. Furthermore, to analyze the impact of a babbling idiot failures on our MPSoC platform a SWC (**BI_SWC**) is

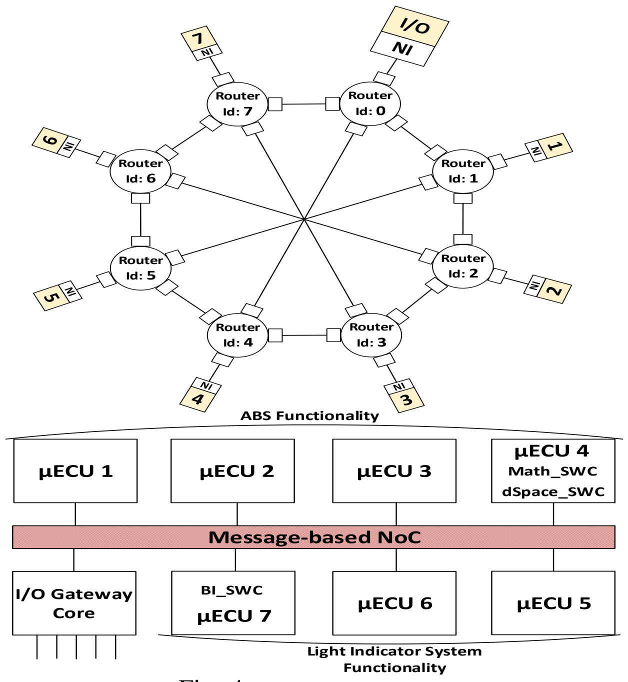


Fig. 4: Automotive Use Case

NoC Configuration				Results					
				Without Fault Injection			Under Fault Injection		
ID	Sender Core	Receiver Core	Temporal Configuration	Min. E2E Delay (ns)	Max. E2E Delay (ns)	Jitter (ns)	Min. E2E Delay (ns)	Max. E2E Delay (ns)	Jitter (ns)
1	I/O core	µECU1	TT (period: 1ms)	17	17	0	17	17	0
2	I/O core	µECU5	RC (MINT: 3 ms)	232	264	32	232	264	32
3	µECU1	µECU2	TT (period: 1ms)	17	17	0	17	17	0
4	µECU2	µECU4	TT (period: 1ms)	32	32	0	32	32	0
5	µECU2	µECU3	TT (period: 1ms)	17	17	0	17	17	0
6	µECU3	µECU4	TT (period: 1ms)	17	17	0	17	17	0
7	µECU5	µECU6	BE	246	272	26	246	286	40
8	µECU6	µECU5	BE	264	284	20	264	284	20
9	µECU6	µECU7	BE	258	286	28	258	304	46
10	µECU4	I/O core	TT (period: 1ms)	17	17	0	17	17	0
11	µECU7	I/O core	RC (MINT: 3 ms)	234	262	28	234	275	41
12	µECU7	I/O core	RC (MINT: 3 ms)	248	289	41	248	298	50

TABLE I: NoC configuration in timing failure experiment

implemented on the $\mu ECU7$ of the LIS functionality sending untimely messages to the network.

Additionally, Automotive Simulation Models (ASMs) are integrated into the AUTOSAR simulation representing the physical environment that interacts with the MPSoC, e.g., road characteristics, dynamics of the brake system hydraulic component, the physical wheel and the human braking behavior of the driver. In the presented simulation scenario, a hard braking is implemented by the driver while the car has an initial speed of $88km/h$. In this scenario the ABS functionality represents a safety critical system while the LIS is not that important in terms of safety (ISO 26262 [20]).

B. Results

1) Timing Failure Experiment: for analyzing the behavior of the AUTOSAR message-based multicore system under a timing failure occurrence, RTE interventions are employed to delay the time-triggered execution of the SWCs which send the messages through the NoC, for both, the ABS and the LIS. In this fault injection experiment we use the NoC

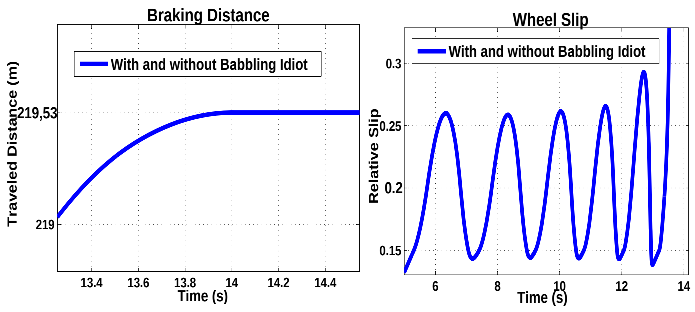


Fig. 5: Braking distance and wheel slip with NoC configuration 1

configuration presented in table I for the setting of the network. In this NoC configuration messages exchanged by the ABS are set as time-triggered messages since this functionality is safety critical, while messages exchanged by the LIS are set as best-effort and rate-constrained messages. Additionally, table I establishes a comparison between the resulting jitter of the NoC messages without faults and in the presence of a timing failure. These communication results demonstrate how time-triggered messages are unaffected by the delay fault occurrence unlike best-effort and rate-constrained messages whose communication jitter is affected.

2) Babbling Idiot Experiment: As a second experiment, we run our simulation scenario having **BI_SWC** on the $\mu ECU7$, so the response of the platform under a babbling idiot failure is tested. The NoC configuration 1 presented in table II is used for the configuration of the multi-core platform. Additionally, we run our simulation use case using the NoC configurations 2 and 3 (table II), where messages exchanged by the ABS functionality are set as rate-constrained messages and best-effort messages respectively, so a comparison can be established.

Figures 5, 6 and 7 represent the distance traveled by the car and the wheel slip with and without babbling idiot failures using each one of the NoC configurations. Figure 5 shows how the behavior of the wheel slip stays even and the distance traveled by the car was not affected, which is due to the time-triggered behavior of the messages sent by the ABS functionality. In contrast, figures 6 and 7 exhibit a difference between the curves when the babbling idiot failure is injected. In figure 6 the braking distance is increased by $2.03m$ ($\simeq 1\%$), while in figure 7 the distance presents a significant increase of $5.37m$ ($\simeq 2.5\%$). Furthermore, table II compares the resulting maximum and minimum delays in all three NoC configurations with and without babbling idiot injection. As presented in this table, NoC configuration 1 shows no difference between maximum and minimum delays on messages sent by the ABS functionality, so a jitter of 0 is kept for all of them. This is not the case for NoC configurations 2 and 3 where the calculated jitter is increased significantly.

3) Omission/Crash Failure Experiment: We run our simulation scenario using RTE interventions for the injection of omission failures in the ABS SWC controller (**Math_SWC**). In this case we expect the monitoring service to recognize the omission failures by receiving error statuses from the RTE

NoC Configuration 1				Results					
				Without Fault Injection			Under Fault Injection		
ID	Sender Core	Receiver Core	Temporal Configuration	Min. E2E Delay (ns)	Max. E2E Delay (ns)	Jitter (ns)	Min. E2E Delay (ns)	Max. E2E Delay (ns)	Jitter (ns)
1	I/O core	µECU1	TT (period: 1ms)	17	17	0	17	17	0
2	I/O core	µECU7	RC (MINT: 3 ms)	232	264	32	232	293	61
3	µECU1	µECU2	TT (period: 1ms)	17	17	0	17	17	0
4	µECU2	µECU4	TT (period: 1ms)	32	32	0	32	32	0
5	µECU2	µECU3	TT (period: 1ms)	17	17	0	17	17	0
6	µECU3	µECU4	TT (period: 1ms)	17	17	0	17	17	0
7	µECU5	µECU6	BE	246	272	26	246	413	167
8	µECU6	µECU5	BE	264	284	20	264	435	171
9	µECU6	µECU7	BE	258	286	28	258	489	231
10	µECU4	I/O core	TT (period: 1ms)	17	17	0	17	17	0
11	µECU7	I/O core	RC (MINT: 3 ms)	234	262	28	234	319	85
12	µECU7	I/O core	RC (MINT: 3 ms)	248	289	41	248	322	74
13	µECU7 BI_SWC	I/O core	BE				Fault Injection		
		µECU1							
		µECU2							
		µECU3							
		µECU4							
		µECU5							
		µECU6							

NoC Configuration 2				Results					
				Without Fault Injection			Under Fault Injection		
ID	Sender Core	Receiver Core	Temporal Configuration	Min. E2E Delay (ns)	Max. E2E Delay (ns)	Jitter (ns)	Min. E2E Delay (ns)	Max. E2E Delay (ns)	Jitter (ns)
1	I/O core	µECU1	RC (MINT: 3 ms)	17	46	29	17	317	300
2	I/O core	µECU7	RC (MINT: 3 ms)	17	46	29	17	373	356
3	µECU1	µECU2	RC (MINT: 3 ms)	17	52	35	17	347	330
4	µECU2	µECU4	RC (MINT: 3 ms)	32	68	36	32	332	300
5	µECU2	µECU3	RC (MINT: 3 ms)	17	52	35	17	386	369
6	µECU3	µECU4	RC (MINT: 3 ms)	17	46	29	17	366	349
7	µECU5	µECU6	BE	138	172	34	138	613	475
8	µECU6	µECU5	BE	146	184	38	146	795	649
9	µECU6	µECU7	BE	184	196	12	184	589	405
10	µECU4	I/O core	RC (MINT: 3 ms)	17	46	29	17	317	300
11	µECU7	I/O core	RC (MINT: 3 ms)	17	52	35	17	319	302
12	µECU7	I/O core	RC (MINT: 3 ms)	17	68	51	17	362	345
13	µECU7 BI_SWC	I/O core	BE				Fault Injection		
		µECU1							
		µECU2							
		µECU3							
		µECU4							
		µECU5							
		µECU6							

NoC Configuration 3				Results					
				Without Fault Injection			Under Fault Injection		
ID	Sender Core	Receiver Core	Temporal Configuration	Min. E2E Delay (ns)	Max. E2E Delay (ns)	Jitter (ns)	Min. E2E Delay (ns)	Max. E2E Delay (ns)	Jitter (ns)
1	I/O core	µECU1	BE	138	176	38	138	917	779
2	I/O core	µECU7	RC (MINT: 3 ms)	17	26	9	17	173	156
3	µECU1	µECU2	BE	32	62	30	32	632	600
4	µECU2	µECU4	BE	64	88	24	64	587	523
5	µECU2	µECU3	BE	32	52	20	32	648	616
6	µECU3	µECU4	BE	17	46	29	17	936	919
7	µECU5	µECU6	BE	64	72	8	64	832	768
8	µECU6	µECU5	BE	52	84	32	52	795	743
9	µECU6	µECU7	BE	32	96	64	32	917	885
10	µECU4	I/O core	BE	17	46	29	17	1089	1072
11	µECU7	I/O core	RC (MINT: 3 ms)	17	22	5	17	119	102
12	µECU7	I/O core	RC (MINT: 3 ms)	17	28	11	17	162	145
13	µECU7 BI_SWC	I/O core	BE				Fault Injection		
		µECU1							
		µECU2							
		µECU3							
		µECU4							
		µECU5							
		µECU6							

TABLE II: Three NoC configurations in bubbling idiot experiment

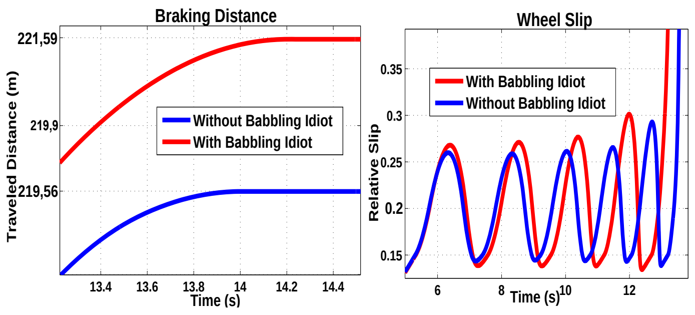


Fig. 6: Braking distance and wheel slip with NoC configuration 2

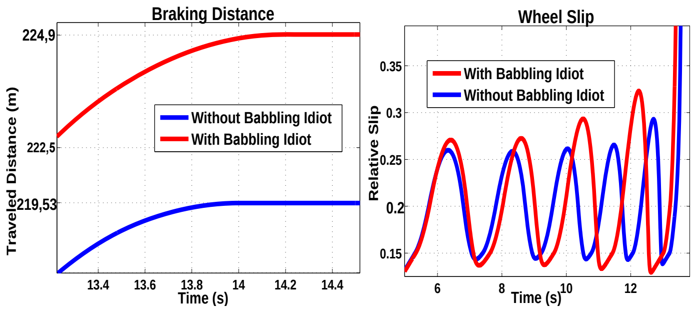


Fig. 7: Braking distance and wheel slip with NoC configuration 3

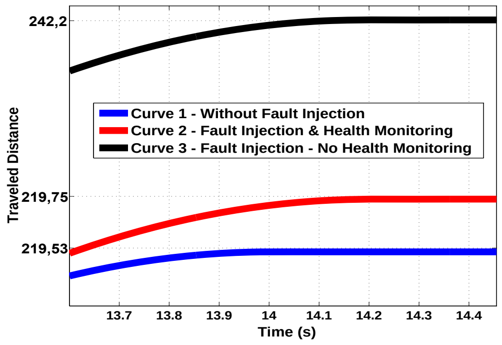


Fig. 8: Braking distance in omission/crash failure experiment

function that triggers this SWC and thus, to determine the presence of a crash failure so **dSpace_SWC** will be activated for fault recovery.

Figure 8 illustrates the comparison of the braking distance behavior when no value failure is injected (curve 1), under fault injection (curve 2) and without health monitoring service (curve 3). This figure shows how the ABS functionality remains operational under a crash failure occurrence when having the monitoring service module. However, a reduction of $\simeq 0.1\%$ $(0.22m)$ in the ABS performance is visible in curve 3 compared with curve 2, which is resulting from the delay in response of the fault recognition algorithm.

4) *Value Failure Experiment:* In order to test the response of the message-based MPSoC platform to a permanent value failure scenario, we use RTE interventions to inject unlimited value failures on **Math_SWC**. The value failure consists of a message value which is not compliant to the data constraint element of the AUTOSAR message.

Just as for the crash failure injection, in this case the expected result is the monitoring service triggering the **dSpace_SWC** once the parameter ϕ is overcome to continue providing the ABS functionality. In Figure 9, curves 1 and 2 depict the distance traveled by the car without faults and in the presence of a permanent value failure. As displayed in this figure, the ABS performance in curve 2 has decreased by $\simeq 0.2\%$ $(0.39m)$ compared with curve 1, but still offering a significant difference of the braking distance in comparison with curve 3 when no health monitoring functionality is provided under a permanent value failure occurrence.

Furthermore, in order to measure the impact of the introduced health monitoring service on the operating system overhead figure 10 compares the number of task invocations

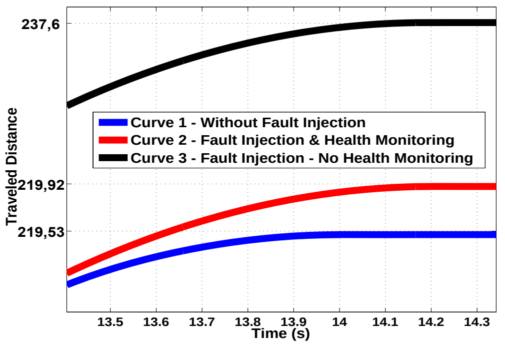
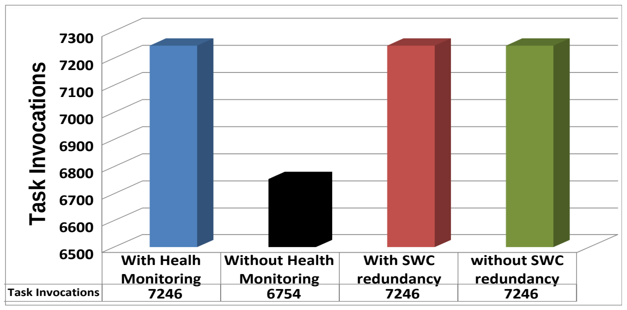


Fig. 9: Braking distance in permanent value failure experiment



	With Healh Monitoring	Without Health Monitoring	With SWC redundancy	without SWC redundancy
Task Invocations	7246	6754	7246	7246

Fig. 10: Comparison of Task Invocations

realized by the AUTOSAR operating system in $\mu ECU4$ with health monitoring implementation, without health monitoring, with SWC redundancy and without SWC redundancy. This figure shows an increase of $\simeq 6.8\%$ of the operating system overhead when a health monitoring service is provided by the BSW in the μECU, which can be justified since the reliability of the system has improved significantly. Additionally, the SWC redundancy does not affect the operating system overhead as expected.

VI. Conclusion

In this work we presented an evaluation of performance and fault containment of the AUTOSAR μECUs in the TIMEA platform. A realistic automotive simulation scenario and multiple fault injection experiments were carried out using a co-simulation environment for AUTOSAR message-based multicore systems.

The obtained results demonstrate how μECUs running safety critical automotive functionalities are isolated from faults occurring on a different core due to the intrinsic fault isolation property of the network, since these functionalities are mapped to time-triggered messages with a static communication schedule. Furthermore, the reliability of the AUTOSAR system has been improved considerably through the integration of the health monitoring service and the SWC redundancy.

Acknowledgment

This work has been supported in part by the European H2020 project SAFEPOWER under grant agreement 687902. Furthermore, gratitude to the dSpace Company for their software support.

References

[1] *AUTOSAR Operating System, AUTOSAR Release 4.3*, AUTOSAR, 2016.

[2] M. Urbina and R. Obermaisser, “Multi-core architecture for autosar based on virtual electronic control units,” in *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*, Sept 2015, pp. 1–5.

[3] R. Obermaisser, H. Kopetz, and C. Paukovits, “A Cross-Domain Multi-Processor System-on-a-Chip for Embedded Real-Time Systems,” *IEEE Transactions on Industrial Informatics*, 2010.

[4] F. Poletti, A. Poggiali, D. Bertozzi, L. Benini, P. Marchal, M. Loghi, and M. Poncino, “Energy-Efficient Multiprocessor Systems-on-Chip for Embedded Computing: Exploring Programming Models and Their Architectural Support,” *Computers, IEEE Transactions on*, 2007.

[5] M. Urbina, H. Ahmadian, and R. Obermaisser, “Co-simulation framework for autosar multi-core processors with message-based network-on-chips,” in *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*, July 2016, pp. 1140–1147.

[6] M. Urbina, “A time-triggered message-based multi-core architecture for autosar: Student research abstract,” in *Proceedings of the Symposium on Applied Computing*, ser. SAC ’17. New York, NY, USA: ACM, 2017, pp. 1488–1489. [Online]. Available: http://doi.acm.org/10.1145/3019612.3019932

[7] M. Urbina and R. Obermaisser, “Efficient multi-core autosar-platform based on an input/output gateway core,” in *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, March 2017, pp. 157–166.

[8] M. Urbina and R. Obermaisser, “A gateway core between on-chip and off-chip networks for an autosar message-based multi-core platform,” in *AmE 2016 - Automotive meets Electronics; 7th GMM-Symposium*, March 2016, pp. 1–6.

[9] K. Goossens, J. Dielissen *et al.*, “Aethereal network on chip: concepts, architectures, and implementations,” *Design Test of Computers, IEEE*, vol. 22, no. 5, pp. 414–421, Sept 2005.

[10] H. Ahmadian and R. Obermaisser, “Time-triggered extension layer for on-chip network interfaces in mixed-criticality systems,” in *2015 Euromicro Conference on Digital System Design*, Aug 2015, pp. 693–699.

[11] *AUTOSAR Specification of Run Time Environment, AUTOSAR Release 4.3*, AUTOSAR, 2016.

[12] R. Obermaisser and P. Peti, “The fault assumptions in distributed integrated architectures,” in *SAE AeroTech Congress and Exhibition*, 2007.

[13] *FlexRay Communications System Protocol Specification Version 2.1*, FlexRay Consortium. BMW AG, DaimlerChrysler AG, General Motors Corporation, Freescale GmbH, Philips GmbH, Robert Bosch GmbH, and Volkswagen AG., May 2005.

[14] *AUTOSAR Software Component Template, AUTOSAR Release 4.3*, AUTOSAR, 2016.

[15] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, “The gem5 simulator,” *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011. [Online]. Available: http://doi.acm.org/10.1145/2024716.2024718

[16] M. Urbina, Z. Owda, and R. Obermaisser, “Simulation environment based on systemc and veos for multi-core processors with virtual autosar ecus,” in *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, Oct 2015, pp. 1843–1852.

[17] N. Agarwal, T. Krishna, L. S. Peh, and N. K. Jha, “Garnet: A detailed on-chip network model inside a full-system simulator,” in *2009 IEEE International Symposium on Performance Analysis of Systems and Software*, April 2009, pp. 33–42.

[18] T. Blochwitz, M. Otter *et al.*, “The Functional Mockup Interface for Tool independent Exchange of Simulation Models,” in *In Proceedings of the 8th International Modelica Conference*, 2011.

[19] “Modeling an Anti-Lock Braking System.” [Online]. Available: www.mathworks.com/help/simulink/examples/modeling-an-anti-lock-braking-system.html

[20] “Road vehicles-functional safety-part 9: Automotive safety integrity level (asil)-oriented and safety-oriented analyses,” *ISO 26262-9:2011, ICS: 43.040.10*, 2011.