Dipl.-Ing. Michael Karner Bakk.techn.

Co-Simulation of Cross-Domain Automotive Systems

Dissertation

vorgelegt an der
Technischen Universität Graz

zur Erlangung des akademischen Grades
Doktor der Technischen Wissenschaften
(Dr.techn.)

durchgeführt am Institut für Technische Informatik
Technische Universität Graz
Vorstand: O. Univ.-Prof. Dipl.-Ing. Dr. techn. Reinhold Weiß

Graz, im April 2011

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am ....
(Unterschrift)

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

....
date (signature)

Kurzfassung

Simulation ist heutzutage eines der Schlüsselgebiete während der Entwicklung von Systemen im Fahrzeug. Allerdings kann mit den bisher gängigen Simulationsmethoden (eine Modellierungssprache, ein Simulator) die hohe Zahl an beteiligten unterschiedlichen Domänen (z.B. Mechanik, Elektronik, Software) mit ihren vielfältigen Anforderungen nicht mehr vernünftig beherrscht werden. Dadurch gewinnt Co-Simulation auch im Automobilbereich immer mehr an Bedeutung. Durch den Einsatz von spezialisierten Modellierungssprachen und Simulationsprogrammen für die jeweilige Domäne bietet Co-Simulation Flexibilität bei gleichzeitig gesteigerter Realitätsnähe. Allerdings hat Co-Simulation, gerade im domänenübergreifenden Bereich, gewisse Einschränkungen.

Diese Arbeit beschäftigt sich mit domänenübergreifender Co-Simulation von Fahrzeugsystemen. Co-Simulation wird um Run-Time Co-Simulation Model Switching erweitert, einer neuen und flexibleren Variante von Co-Simulation. Die Idee besteht darin, die Co-Simulation nicht nur wie bisher üblich einmalig statisch zu definieren, sondern anhand bestimmter Kriterien wie beispielsweise Zeit oder Signalverhalten zur Laufzeit die verwendeten Simulationsmodelle zu variieren. Durch Flexibilitäts- als auch Effizienzsteigerungen wird dadurch eine verbesserte Analyse von domänenübergreifenden Systemen möglich. Basierend auf diesem Ansatz wird eine Methodik zur leistungsfähigen Analyse von domänenübergreifenden Systemen im Fahrzeug entwickelt. Ausgehend von spezifizierten Systemanalysezielen werden geeignete Simulationsmodelle entwickelt und durch den Einsatz von modifizierten Verfahren aus dem Gebiet der Safety Analyse eine geeignete Run-Time Co-Simulation Model Switching Konfiguration erstellt. Dies ermöglicht eine effiziente und ganzheitliche Simulation und Analyse von domänenübergreifenden Systemen im Fahrzeug.

Anhand von Beispielen aus dem Bereich domänenübergreifender Systeme im Fahrzeug wird die vorgestellte, auf Run-Time Co-Simulation Model Switching basierende Methodik angewandt und validiert. Im Rahmen der Beispiele werden sowohl FlexRay als auch AUTOSAR und Simulationsmodelle für komplette Fahrzeuge (mit Fokus auf den Antriebsstrang) in eine gemeinsame Co-Simulation integriert, und deren Zusammenspiel wird analysiert.

Abstract

Today, simulation is a key issue for the development of state-of-the-art automotive systems. However, due to the increasing number of interacting domains (e.g. mechanics, electronics, software) and their very different requirements, standard simulation cannot handle the challenges arising anymore. It is limited to the features of a single modelling language and tool which do not allow to cover all the required details for an extensive simulation-based system analysis. As a result, co-simulation is gaining importance within the development of automotive systems. By using specialised modelling languages and tools for each domain, co-simulation features enhanced flexibility and real-world behaviour. However, especially for cross-domain interaction, co-simulation bears a number of challenges.

This work focuses on the co-simulation of cross-domain automotive systems. Run-time co-simulation model switching, a flexible novel co-simulation methodology, is developed. The main idea is to perform co-simulation not only as a pre-defined static setup like in traditional co-simulation approaches. Instead, according to defined criteria like time or signal behaviour, simulation model variations are performed during co-simulation execution time. This enhanced co-simulation methodology provides the system developer with a new degree of freedom in simulating automotive systems. It enhances co-simulation efficiency, flexibility and cross-domain analysis capabilities. Based on this new methodology, an approach for the efficient analysis of cross-domain automotive systems is developed. According to specified system analysis goals, suitable simulation models are developed and, by using adapted techniques from the area of safety analysis, a suitable run-time co-simulation model switching configuration allowing to fulfil the system analysis goals is created.

The proposed run-time co-simulation model switching methodology is successfully applied and validated within the co-simulation of several cross-domain automotive systems, including FlexRay, AUTOSAR and power-train focused simulation of a complete vehicle.

Acknowledgements

In the beginning, I would like to thank my supervisor Prof. Dr. Reinhold Weiß for his helpful advice during the conduction of my dissertation at the Institute for Technical Informatics at Graz University of Technology. I especially want to thank Dr. Christian Steger for his exceptional support and extraordinary mentoring during the last years. Additionally, I am grateful to Dr. Eric Armengaud, project leader of TEODACS, whose continuing enthusiasm and motivation were really priceless.

I would like to thank all people and companies involved within the TEODACS project, they are too numerous to list them all. My special thanks go to the Virtual Vehicle Competence Center for providing the framework for the TEODACS project, especially to Univ.-Doz. Dr. Daniel Watzenig as head of vehicle E/E and software. Furthermore, I would like to thank Dr. Markus Pistauer and DI Andreas Schuhai from CISC Semiconductor for providing valuable support and financing after the completion of TEODACS.

Moreover, I would like to thank all my students. They performed outstanding work. I especially want to thank Martin Krammer and Stefan Krug. Both provided a valuable contribution to the TEODACS FlexRayXpert.Sim framework. Furthermore, my thanks go to the entire staff at the Institute for Technical Informatics for their support, constructive comments and helpful suggestions during my work.

Last but not least, I would like to thank my parents for their enduring support throughout all my academic studies. With their ongoing motivation, they helped me to successfully handle even critical situations. I would not have got that far without them.

Graz, April 2011 Michael Karner

Extended Summary

Driver assistance systems are integrated into today's cars in an increasing number. As a result, also the X-by-wire approach becomes more and more important. To handle this overwhelming demand for computational power, the number of electronic control units (ECU) within a car is growing. For example, more than 70 different ECUs distributed all over the car are nothing exceptional anymore. Most of them are safety critical systems with the respective real-time requirements. These few examples demonstrate that currently, most of the innovation potential within cars comes from electronics. The introduction of new functionalities like driver assistance systems, the enhancement of mechanic solutions by electronic components and even the replacement of mechanic components by their electronic counterparts (X-by-wire) are typical challenges and goals during the development of tomorrow's cars. Additionally, due to the close interactions within mechanics, electronics and software (e.g. adaptive cruise control systems), the analysis and handling of cross-domain effects are gaining importance.

To get rid of this enormous amount of complexity, new concepts have been introduced into the automotive industry. For example, due to the need for a fast but also reliable communication system, FlexRay has been developed. FlexRay is a time-triggered communication system, featuring both high bandwidth and reliability. However, FlexRay bears a couple of challenges like the complex configuration and the effects of internal and external influences on the FlexRay system. To handle the increasing demand and complexity of software within the car, AUTOSAR has been developed as an industry standard for managing distributed automotive applications. However, with FlexRay, what seems to be simple on paper, becomes extremely complex in practice.

As a result, simulation during the development of automotive systems is highly required. Typical simulation approaches use one modelling language within one simulation tool. However, the complex cross-domain interaction and the enormous number of different systems interacting with each other lead to very different requirements for the modelling language and simulation tool. These wide-spread needs in automotive system simulation simply cannot be handled anymore by using standard simulation. Hence, the topic of co-simulation is gaining more and more importance within the development of automotive systems. In co-simulation, different modelling languages and simulation tools are used within a common simulation (= co-simulation), see Fig. 1. The most suitable tool can be used for each automotive component, allowing for a more efficient simulation approach. There are, however, still some drawbacks, especially for cross-domain co-simulation. For example, as the co-simulation performance is determined mainly by the slowest simulator involved, coupling simulation tools from different domains with very different time constants has so far not been practical.

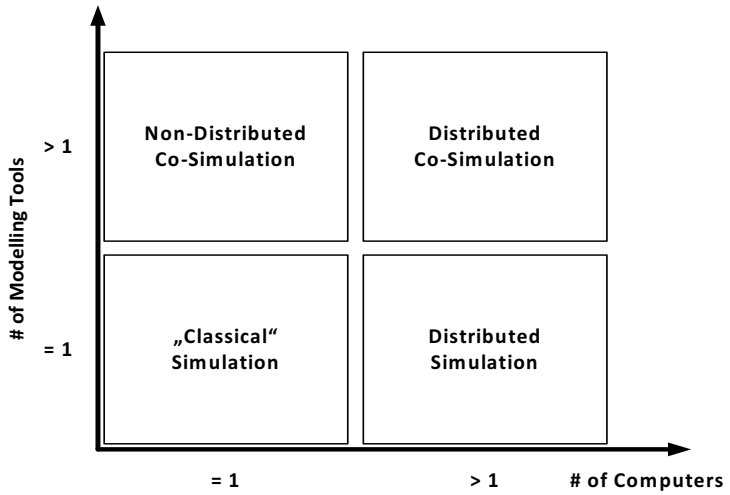


Figure 1: Differentiation between Simulation and Co-Simulation (based on [1])

This work is part of the TEODACS project (Test, Evaluation and Optimisation of Dependable Automotive Communication Systems). The project aims at understanding the different internal and external effects influencing a dependable communication network (especially FlexRay) and the applications using this network. For this, a co-simulation environment is created containing FlexRay simulation models, models of distributed applications by using AUTOSAR concepts and even a power-train focused simulation of a complete car. Additionally, the same setup is also available in real hardware, allowing for interaction and comparison between the co-simulation and reality.

Within this work, a new methodology for the efficient co-simulation of cross-domain automotive systems is described (see Fig. 2 and Fig. 3). This methodology is based on runtime co-simulation model switching (RCMS). In RCMS, standard static co-simulation is enhanced to have dynamic behaviour. As a result, the simulation models used within the co-simulation can be changed during the execution of the co-simulation. Basically, there are two different types of RCMS. The first one is time-triggered RCMS. Here, co-simulation model switching is defined in advance by selecting the points during co-simulation execution when a model switch should take place. The other type of RCMS is adaptive RCMS. A signal analysis is performed during the execution of the co-simulation, comparing the signal behaviour to a user-defined set of criteria. If the signal behaviour matches a criterion, a co-simulation model switching to a specific simulation model is initiated. An important point within both types of RCMS is the correct synchronisation of the upcoming model to allow for seamless switching between the models. By using RCMS, the developer gains an enormous amount of flexibility compared to standard static co-simulation while at the same time improving co-simulation performance, cross-domain interaction and analysis possibilities.

Based on RCMS, this work presents a new methodology for the co-simulation of cross-domain automotive systems. It can be split in two main parts: the development and the execution of the cross-domain automotive co-simulation. The development of a co-

simulation for cross-domain automotive systems is shown in Fig. 2. Based on a specification of automotive system analysis goals, an integrated development of both RCMS configuration and simulation models is done. Starting from an initial simulation model design, the design of a suitable RCMS configuration to fulfil the specified system analysis goals is performed. This RCMS configuration design process uses methodologies from the area of safety analysis (failure mode and effects analysis (FMEA), subsystem hazard analysis (SSHA), system hazard analysis (SHA)) which are adapted for the goal of designing a suitable RCMS configuration. The feedback generated during this process can be used to enhance the simulation model design and implementation process, finally leading to a suitable RCMS configuration and the according simulation models targeted for fulfilling the specified system analysis goals. The second part of the new methodology for the co-simulation of cross-domain automotive systems is the actual execution of the RCMS configuration (see Fig. 3) developed during the first part. Here, after the RCMS configuration and simulation models have been integrated into the cross-domain co-simulation environment, the co-simulation is started and RCMS (both time-triggered and adaptive) is executed according to the configuration. By doing so, results suitable for fulfilling the specified system analysis goals are gained.

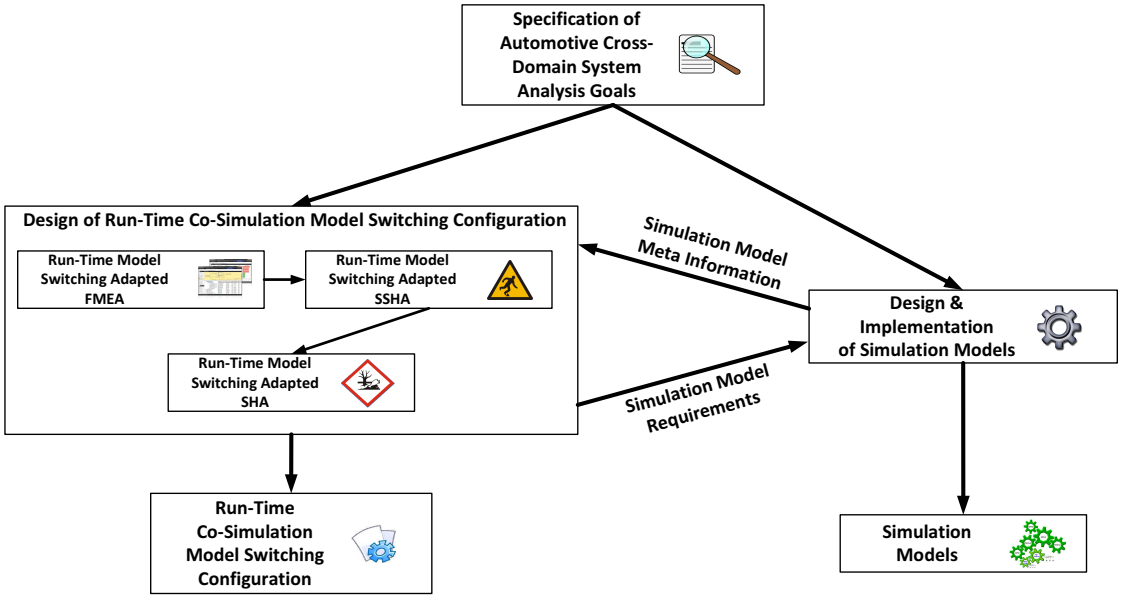


Figure 2: Development of a Co-Simulation of Cross-Domain Automotive Systems

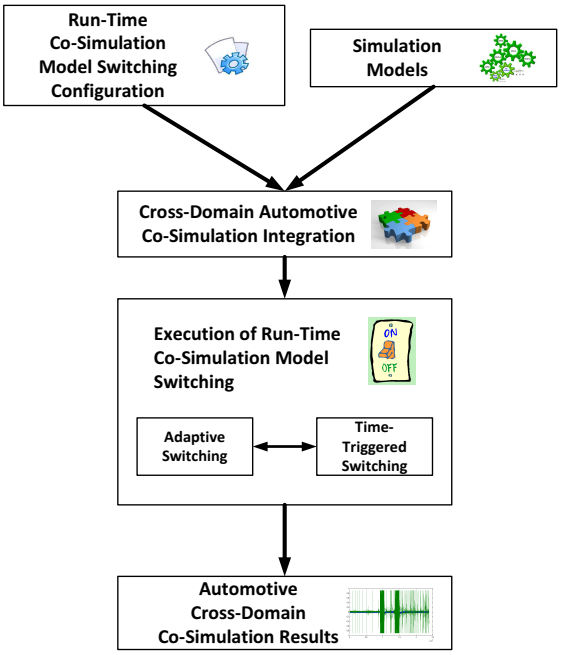


Figure 3: Execution of a Co-Simulation of Cross-Domain Automotive Systems

In conclusion, this dissertation allows for an efficient and flexible co-simulation of cross-domain automotive systems within the TEODACS project. The main benefits of the new RCMS-based methodology are its improved co-simulation performance, flexibility and cross-domain analysis capabilities compared to standard co-simulation. It allows for a consistent and efficient way of analysing cross-domain automotive systems by using co-simulation. RCMS provides the developers of automotive systems with new degrees of freedom in analysis, allowing for the examination of effects which until now have not been able to be modelled at the required level of detail or simulated within a reasonable amount of time. Essential contributions within this dissertation are the development of RCMS to introduce dynamic into co-simulation, the development of a methodology to design a suitable RCMS configuration for specified system analysis goals and the new methodology for the co-simulation of cross-domain automotive systems by using RCMS. Another contribution is the development of a cross-domain co-simulation consisting of FlexRay (electronics), AUTOSAR concepts (software) and the powertrain-focused simulation of a complete car (mechanics).

Contents

List of Figures

List of Abbreviations

AUTOSAR	Automotive Open System Architecture
ECU	Electronic Control Unit
EPL	Electrical Physical Layer
FMEA	Failure Mode and Effects Analysis
HDL	Hardware Description Language
ICOS	Independent Co-Simulation
IEEE	Institute of Electrical and Electronics Engineering
RCMS	Run-Time Co-Simulation Model Switching
RTL	Register Transfer Level
SAE	Society of Automotive Engineers
SHA	System Hazard Analysis
SoC	System on Chip
SSHA	Sub-System Hazard Analysis
swFMEA	Switching-Adapted Failure Mode and Effects Analysis
swSHA	Switching-Adapted System Hazard Analysis
swSSHA	Switching-Adapted Sub-System Hazard Analysis
SyAD	System Architect Designer
TDMA	Time Division Multiple Access
TLM	Transaction Level Modelling
TEODACS	Test, Evaluation and Optimisation of Dependable Automotive Communication Systems
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VHDL-AMS	VHDL-Analogue and Mixed Signal

Chapter 1

Introduction to Co-Simulation of Cross-Domain Automotive Systems

Automotive systems have evolved rapidly during the last decade, especially in the area of electronics and software. Due to the enormous number of new functionalities, often founded by safety critical real-time systems, automotive electronics in today's cars are organised as complex, distributed systems. However, at the same time, the level of complexity reaches new heights, requiring new methodologies for the development of automotive systems. A possible answer to this question is simulation, and especially co-simulation.

1.1 Motivation

In today's cars, over 80% of the innovation potentials come from electronics [2]. Additionally, the 2009 world market for automotive embedded electronics represents about 3 billion units delivered per year for a budget encompassing approximately 160 billion Euros with an annual growth of 9 percent [3]. For example, within current upper-class cars like the Volkswagen Phaeton (see Fig. 1.1), the number of electronic control units (ECU) can be over 80, while at the same time, the software running on these ECUs is comprising more than 100 million lines of code [4], [5] and the components are connected using more than 2000 cables with a total length of nearly 4 kilometres.

This enormous complexity is mainly caused by the introduction of new functionalities and the enhancement of existing components within the car. For example, existing mechanic solutions are enhanced or even replaced by electronic components (e.g. X-by-wire, engine management, car dynamics), or new advanced functionalities like radars and distance sensors for active safety are introduced. For dealing with this increased level of complexity, the automotive industry introduced several new standards. For example, to fulfil the demand for a fast but also reliable communication system, FlexRay has been developed [7]. It is based on a time-triggered approach and features both high performance and reliability. Another example is the Automotive Open System Architecture (AUTOSAR) [8], an automotive middleware for the development of automotive distributed applications. However, new concepts like FlexRay or AUTOSAR on the one side help to deal with the increased complexity and requirements within safety critical real-time automotive systems, but on the other side, they introduce even more complexity because they

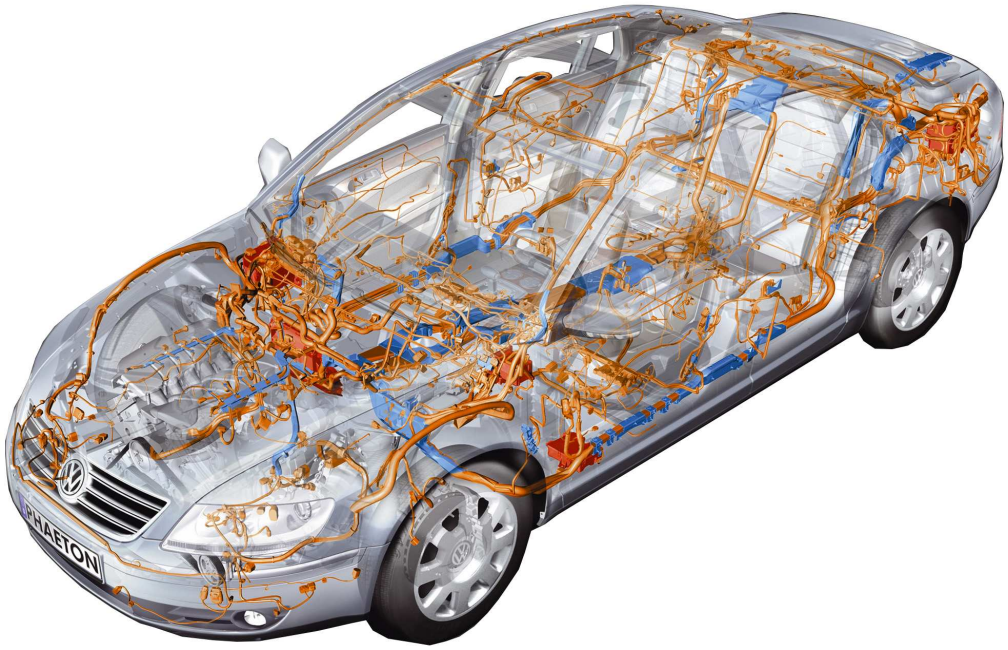


Figure 1.1: Volkswagen Phaeton: Automotive Electronics as Distributed System [6]

have to be managed too. Additionally, it can be seen that not only the overall complexity is rising, but at the same time, the cross-domain interactions (e.g. mechanics, electronics, software) become more and more important for the development of automotive systems [9]. As a result, simulation nowadays is a must during the development of automotive systems. Otherwise, the system complexity could not be managed anymore.

1.1.1 Simulation of Automotive Systems

As demonstrated before, due to the wide range of requirements, the exploding complexity and the different components involved, simulation is a great necessity for the development process of automotive systems. Typically, there are two different kinds of simulation performed, the simulation of single components (e.g. [10], [11]) and the simulation of a multitude of (different) components interacting with each other (e.g. [12], [13]). Both kinds of simulation are practically used throughout the industry and academia. Simulation allows to have a closer look at the component(s), eliminating the black box behaviour of real hardware. The functionality can be verified in advance, hence reducing development effort due to avoiding costly redesigns and leading to faster time to market. Both time and money are crucial within the automotive industry.

However, there are some important drawbacks. When simulating only single automotive components, for example by using SystemC [14], there is no real interaction with other components and the environment. As a result, the system behaviour cannot realistically be simulated. It is reduced to the theoretical behaviour of the component within a sterile environment. The simulation of a single component can definitely help the developer during the design and implementation of this component. Yet, in most environments, it cannot be used to demonstrate whether a component works correctly under all conditions and

within a distributed system. The missing interaction with other components and the environment, especially the missing cross-domain interactions, massively restrict the usability of single-component simulations of automotive systems.

Performing a multi-component simulation (e.g. an automotive multi-ECU network) overcomes some of the problems arising during single-component simulation as an interaction between the different components of the simulation is possible. There are several ways to perform such a simulation. For example, a multi-ECU network simulation is typically performed either by using specially designed hardware (residual bus/network simulation) or by using high abstraction level simulation models of the network components. Another possibility is to only simulate a single layer out of the multitude of layers building a network. All of these three possibilities have several disadvantages, lacking either observability (hardware-based solutions) or accuracy for the overall network (high-level simulation, simulation of single network layers). Additionally, the typical multi-ECU network simulation is missing cross-domain interaction required for the analysis of the overall system behaviour.

In summary, simulation of single components and multi-component simulation both have several individual drawbacks. Hence, simulation as often performed until now is not enough to handle the complexity and challenges arising during the development process of today's automotive systems.

1.1.2 Automotive Cross-Domain Simulation

Standard simulation is not enough for the holistic simulation of automotive systems. Due to the limitations to the features of one modelling language and the according simulation tools, the exploding complexity is still not manageable. Additionally, there are only a few modelling languages and tools supporting multiple domains. Well-known examples are VHDL-AMS [9], [15], [16], Saber MAST [17], Modelica [18] and SystemC-AMS [19]. However, cross-domain simulation of automotive systems is clearly required [20], [21], [22], [23].

A possible solution to achieve an efficient cross-domain simulation of automotive systems is by using co-simulation [24]. Co-simulation means to simulate a system using different modelling languages, abstraction levels and tools in one common simulation [25]. While co-simulation already is a relatively well-known approach for the development of automotive systems (see e.g. [25], [26]), it still has some important drawbacks. For example, to maintain system integrity, the overall co-simulation performance is determined by the slowest simulator involved. If connecting simulation tools from different domains having very different requirements (e.g. microelectronics with simulation time steps in the area of picoseconds and typical time intervals under observation of several milliseconds, and mechanics with time steps in the area of milliseconds and time intervals under observation of several seconds or even minutes), this definitely is a problem. Within the typical time intervals for microelectronics, nothing important may happen within the mechanic simulation. By contrast, the relevant time intervals for mechanic simulation cannot be simulated with microelectronics simulation within a reasonable amount of time. Hence, standard co-simulation is not always a suitable solution to perform cross-domain simulation of automotive systems.

1.2 Co-Simulation of Cross-Domain Automotive Systems

1.2.1 The TEODACS Project

This thesis is part of the TEODACS project [27]. TEODACS means *Test, Evaluation and Optimisation of Dependable Automotive Communication Systems*. The goal of this project is to understand the interactions between the layers building a dependable network like FlexRay and to evaluate the different effects influencing the communication (e.g. topology, EMC, cross-domain interaction). The approach is based on the development of a cross-domain co-simulation framework (FlexRayXpert.Sim) tightly interfaced to a realistic prototype (FlexRayXpert.Lab), see Fig. 1.2. The simulation framework provides observ-

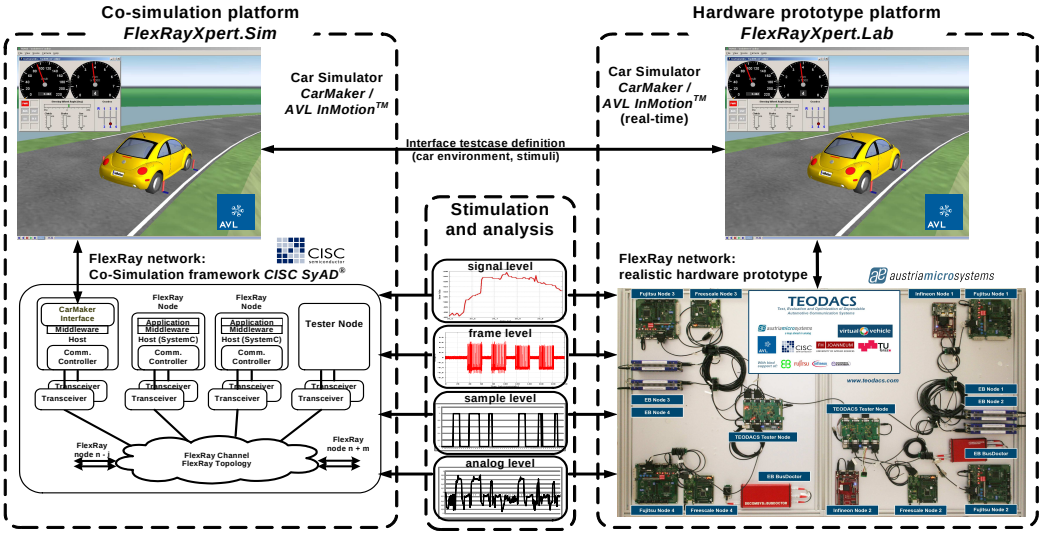


Figure 1.2: TEODACS Overview

ability of the internal components, hence enables the efficient analysis of the cross-domain system. It features high flexibility for testing new configurations and the possibility to validate the system during earlier phases of the development process. The prototype presents a real behaviour both in the time and value domain. The coupling of these two platforms combines the advantages of the two approaches: It enables for example the validation of the simulation models and the analysis of real scenarios within the simulator. Within this thesis, the TEODACS FlexRayXpert.Sim co-simulation framework has been developed, see left side of Fig. 1.2.

1.2.2 Problem Description

As shown in section 1.1, several limitations and drawbacks in the simulation of automotive systems exist.

- Simulation of single components: no real interaction with other components and the environment
- Simulation of multi-ECU networks: missing observability (hardware-based solutions) or high-level and/or single-layer simulation only (missing details)

- Missing of cross-domain interactions which are required for detailed analysis of system behaviour
- Co-Simulation: restrictions in cross-domain systems

The main challenge of this thesis is to solve the problem that at the same time, both high accuracy and acceptable simulation performance are required to enable a holistic analysis of cross-domain automotive systems within a reasonable amount of time. Based on this, a further challenge is the creation of an efficient cross-domain co-simulation environment for automotive systems. Another important challenge is to define a co-simulation-based system analysis flow to design a suitable co-simulation configuration for a specified system analysis goal, including simulation models fulfilling the analysis goals.

1.2.3 Contribution and Significance

This thesis comprises the following major contributions:

1. Design of a flexible co-simulation methodology based on run-time co-simulation model switching (RCMS). By exchanging the simulation models used during run-time of the co-simulation, it is possible to get high co-simulation performance while at the same time having the necessary result details when required. For this, time-triggered RCMS and adaptive RCMS are specified in detail. RCMS allows for the efficient integration of multiple domains into a common co-simulation. [28], [29], [30], [31]
2. A RCMS-based methodology for the analysis of cross-domain automotive systems is developed. Based on specified system analysis goals, suitable simulation models are designed and, by using adapted techniques from the area of safety analysis, a suitable RCMS configuration allowing to fulfil the system analysis goals is created. [28]
3. Development of a comprehensive case study: TEODACS FlexRayXpert.Sim, a cross-domain automotive co-simulation framework for the analysis of dependable automotive communication systems. [32], [33], [34]

1.2.4 Organisation of the Thesis

The remainder of this thesis is organised as follows. Chapter 2 describes related work in the areas of simulation of automotive systems, co-simulation and run-time simulation model switching. In Chapter 3, the methodology of run-time co-simulation model switching and its application for the co-simulation of automotive systems are explained. The case study comprising the TEODACS FlexRayXpert.Sim automotive co-simulation framework based on RCMS is described in Chapter 4. Chapter 5 concludes this thesis and provides an outlook on possible future work.

Chapter 2

Related Work

Due to the rising demand for safety critical real-time systems within cars, the complexity of automotive systems is exploding. With up to 80 different ECUs forming a complex, distributed system the development process cannot be handled anymore without the support of simulation. This chapter evaluates typical methodologies for the simulation of automotive systems and depicts their advantages and limitations. Additionally, it is evaluated if state-of-the-art co-simulation and run-time model switching techniques are able to overcome these limitations.

2.1 Simulation of Automotive Systems

With more than 100 million lines of code [4], more than 80 different ECUs [5] and extensive cross-domain interaction (software, electronics, mechanics) [9], simulation plays a key role during the development process of automotive systems. However, the enormous system complexity cannot be handled easily even by using simulation. By introducing new technologies like FlexRay [7] [35] and AUTOSAR [8] [36], the demand for simulation is enhanced even more. Simulation allows to have a closer look at the system, eliminating the black box behaviour of real hardware. The functionality can be verified in advance, hence reducing development effort due to avoiding costly redesigns and leading to faster time to market.

Generally, there are two main concepts for the simulation of automotive systems: the simulation of single components and the simulation of a multitude of (different) components interacting with each other. In sections 2.1.1 and 2.1.2, an overview of both types of simulations is provided. Due the enormous number of different components and possible combinations within a car, the focus is set to safety critical real-time systems (esp. FlexRay and AUTOSAR) and the simulation of automotive multi-ECU networks (as an example for simulating interacting components).

For the communication between automotive safety critical real-time systems, FlexRay fulfils the demand for dependability and high communication performance. It is based on a time-triggered approach resulting in deterministic communication behaviour and provides a bandwidth of up to 10 Mbit/s per channel [35]. However, the challenges when using FlexRay are lying in its complexity. For example, there are more than 10^{48} different ways to configure a FlexRay network [37]. Additionally, the FlexRay signal integrity on

the electrical physical layer (EPL) plays an important role for the dependability of the network. Hence, the development of simulation models for FlexRay components is of great interest.

AUTOSAR [8] [36] is an architecture for automotive embedded software. By using different abstraction levels, component-based design and by defining processes and methodologies for the development and tool-supported generation of software, it is aiming to achieve enhanced reusability, maintainability, transferability of functions throughout the network, scalability and the integration of safety requirements. However, the enormous complexity when performing AUTOSAR-based development cannot be handled easily. As a result, the simulation of AUTOSAR-based systems is an important topic within industry and academia.

2.1.1 Simulation of Single Automotive Components

The simulation of single automotive components is a well-known approach used by both industry and academia. By focussing on a clearly defined component, it is possible to generate a detailed simulation model covering the complete behaviour of the component. This is exceptionally helpful during the development process of the modelled component as the intended functionality can be easily tested and verified. Additionally, the effects of changes in e.g. important algorithms or functionalities can be evaluated very quickly. High-level (e.g. Simulink, SystemC) and low-level modelling languages and tools (VHDL(-AMS), Saber) can be used for the simulation of single automotive components, depending on the simulation purpose.

As illustrated before, the development of simulation models for FlexRay components is of great interest. In the following, several examples for simulation models of FlexRay components are given.

The logical FlexRay protocol is handled by the communication controller, hence, this component is one of the most interesting to simulate, but also one of the most complex. Several different modelling languages are used for this task, typically high-level ones (e.g. SystemC, C, Stateflow). In [38], Lari et al. present a Verilog model of a communication controller. It is modelled at behavioural level, hence, at a relatively abstracted level. The authors introduce errors like bit flips into the simulation model to analyse which type of error occurring within which part of the communication controller leads to a missing FlexRay frame. As a result, the most fragile parts of the FlexRay controller are detected to be the controller host interface and the clock synchronisation process. The least sensitive part of the controller according to Lari et al. is the coding and decoding unit which, by specification, is designed to be more fault-tolerant. Another simulation model of a communication controller is shown by Kim et al. [11]. It is implemented at system level and uses SystemC as hardware description language (HDL). The authors use the simulation model for verification purposes before building a correspondent register transfer level (RTL) model.

While the communication controller is responsible for handling the logical FlexRay protocol, the EPL is handled by the FlexRay transceiver. It is responsible for conversion between digital high/low values and their corresponding representation on the FlexRay bus. Additionally, the transceiver performs tasks regarding power saving, bus wakeup etc. Several simulation models are available. In [39], a generic transceiver model implemented

using VHDL-AMS is presented. Another VHDL-AMS-based transceiver model is developed by Muller et al. in [40]. It is a mixed-mode behavioural model and the authors perform several tests to demonstrate effectiveness and specification conformance of the developed model. Another behavioural simulation model of a specific NXP transceiver is shown in [10]. It is implemented using Saber. All of these transceiver models focus on the validation of the single component (e.g. conformance test).

There are some important drawbacks when simulating only single components as there is no real interaction with other components and the environment. As a result, the system behaviour cannot be realistically simulated. It is reduced to the theoretical behaviour of the component within a sterile environment. The simulation of a single component can definitely help the developer during the design and implementation of this component. However, in most environments, it cannot be used to demonstrate whether a component works correctly under all conditions and within a distributed system. The missing of interaction with other components and the environment, especially the missing cross-domain interactions, massively restrict the usability of single-component simulations for automotive systems.

2.1.2 Simulation of Automotive Multi-ECU Networks

A possible way to overcome the limitations of simulating single components only is to simulate a multitude of (different) components interacting with each other. This has the advantage that the simulation behaviour is more realistic. Hence, effects propagated throughout the system can be examined in detail. There are several different approaches to perform such a simulation, each having different advantages and limitations.

High-level simulation is a very common way to simulate an automotive network of multiple ECUs. For example, in [12], Krause et al. present a timing simulation of interconnected AUTOSAR software components. It is implemented using SystemC, and AUTOSAR concepts like ports and runnables are mapped to their corresponding SystemC equivalents. Another high-level simulation of a large heterogeneous system including several ECUs communicating via a network is described by Buchmann et al. [41]. The implementation is done by using TLM. A hardware abstraction layer is used for high-level access to the hardware components. The simulation is carried out at a very high abstraction level resulting in high simulation performance, but also losing many important details about the communication. In [42], another TLM approach for communication networks is presented. The transaction level model of a FlexRay network is used for an evaluation of time performances according to different protocol parameters. The FlexRay interface is described in a time-behavioural way using TLM techniques in which selected components of the FlexRay protocol are modelled. Another high-level approach for the simulation of automotive multi-ECU networks is shown by Albert et al. [43]. The ECUs are simulated by using Simulink and the time-triggered busses (FlexRay, CAN) by using TrueTime. Within an example, the effects of communication on an active front steering are examined. Obermaisser et al. [44] demonstrate a simulation framework for distributed real-time systems based on time-triggered control. Here, the emphasis is placed on accurate reproduction of the temporal behaviour of the platform. A significant drawback of high-level approaches is the loss of details. Even minor changes within the system configuration parameters may lead to notably different network behaviour. By using only high-level modelling like TLM,

lots of these subtle details may be abstracted away, making the automotive system simulation not realistic enough for trustworthy in-detail verification of the complete network. However, such simulation approaches can be very useful for the analysis of e.g. scheduling at application level.

The simulation of components within a single FlexRay layer is shown in [45]. Bollati et al. present a Saber-based simulation of the FlexRay EPL components, namely transceiver, termination, cable and passive star. An analysis of the FlexRay signal integrity is performed for a network consisting of several nodes. While the simulation itself is highly detailed, it is missing interaction with other components and FlexRay layers except the EPL. Hence, its usability is limited regarding overall automotive system analysis.

A further approach for analysing automotive multi-ECU networks is residual bus simulation, which enables the emulation of an entire sub-network. Industrial solutions from e.g. Elektrobit [46], dSPACE [47] and Vector [48] are available. All of these examples are a combination of hardware and software components: The software emulates the functionalities of the missing ECUs, and the application messages are transmitted to a real network using dedicated hardware components. Hence, the advantages of simulation (observability, traceability, flexibility...) are lost for many parts of the system. This makes residual bus simulation feasible only for very specific and clearly defined requirements.

It can be seen that several different approaches for the simulation of automotive multi-ECU networks exist, especially of dependable communication networks. However, these solutions usually cover only a specific part of the network (missing interaction), are hardware-based (missing advantages of simulation) and/or are working on a very high abstraction level (missing required details for in-depth system analysis).

2.2 Co-Simulation

Co-simulation is a well-known step during the development of embedded systems. The idea behind co-simulation is to simulate a system by using different HDLs and/or multiple abstraction levels within a single common simulation (co-simulation) [25], [24], [49]. By using co-simulation, it is possible to handle the increased system complexity, as not all simulation models involved have to be implemented using the same abstraction level, modelling language and tool. This allows for an iterative refinement process during system development in which selected parts of the system are already modelled using highly detailed simulation models, while other parts are still modelled in a very abstract way. As a result, the flexibility of the system development is enhanced, allowing verification already in early phases of the design and shortening the overall development cycle.

Due to the rising complexity of automotive systems which cannot be handled anymore by using standard simulation, co-simulation is also increasingly used in the automotive area by industry and academia [26], [9], [20], [21], [22], [23], [50]. It is often seen as possible answer to allow for a cross-domain simulation of complex automotive system, resulting in enhanced analysis and development capabilities.

2.2.1 Co-Simulation Features & Frameworks

Several co-simulation frameworks have been proposed in the past. Some of them are part of scientific projects (e.g. [24], [51]), while others are available as commercial solutions

(e.g. [52], [26]). However, they all basically have to face and solve similar challenges:

- Establishing a connection between different simulation tools (data exchange)
- Integrating different HDLs based on different concepts
- Maintaining co-simulation consistency by synchronised execution of the different simulation tools
- Providing an interface for co-simulation configuration definition (graphical or textual)

Due to the amount of different co-simulation frameworks, only a few selected ones can be shown here. In [24], Bouchhima et al. propose a co-simulation framework that enables coupling of SystemC and Simulink models, thus connecting the worlds of continuous and discrete-event simulation. Another co-simulation framework is shown by Birrer et al. in [51]. Here, the integration of SystemC models into a Verilog-A/AMS simulation is performed. By doing so, the gap between system level description and hardware implementation can be bridged. An example for a commercial co-simulation framework is CISC SyAD [52]. Based on the work described by Kajtazovic et al. [25], it features the coupling of several simulators from both discrete-event and continuous time simulation. Supported HDLs are for example SystemC, VHDL(-AMS) and Simulink. An example for a dedicated automotive co-simulation environment is ICOS (Independent Co-Simulation) [26]. This co-simulation framework enables the coupling of different domains within a co-simulation (e.g. powertrain, thermodynamics, electrics).

Common ground for typical co-simulation frameworks is the problem of cross-domain co-simulation support. As the overall co-simulation performance is determined by the slowest simulator involved, the integration of simulation models coming from different domains having very different time constants presents a huge problem. While most of the frameworks simply neglect the problem, generating implicit restrictions on the cross-domain co-simulation capabilities, some frameworks try to deal with it. For example, in ICOS [26], different advanced coupling algorithms are used to minimise the error due to the different time constants of the different domains involved. However, this methodology is not applicable for all types of domains and simulation tools (e.g. microelectronics simulation). As a result, cross-domain co-simulation like strongly required by automotive systems is still an important problem to be solved.

A very important fact within current co-simulation frameworks is that the co-simulation configuration has to be defined prior to the actual co-simulation execution. Hence, co-simulation as defined until now is a static approach with which the connections between the different models and their level of detail are statically defined in advance.

2.2.2 Design of Co-Simulation Configurations

The task of hardware/software partitioning in general is a well-known approach [53], [54]. However, this is different for the task of designing a suitable co-simulation configuration. For standard static co-simulation, this task is quite manageable and mainly determined by the already existing simulation models. As a result, only some model-specific solutions exist, dealing with the generation of a co-simulation configuration for standard static co-simulation.

In [55], Bragagnini et al. present a model-specific approach to determine the co-simulation partitioning between SystemC and the NS-2 network simulator for a sensor node. A very similar approach is performed by Bombieri et al. [56] with a SystemC/NS-2 partitioning performed by just comparing advantages and limitations of four different partitioning variants against each other. A more general approach is described by Fummi et al. [57]. The authors describe a two-phase partitioning process followed by several refinement steps with the target of finding a suitable co-simulation configuration for the co-simulation of a networked-embedded system. For the individual components of the simulation model, it is decided whether they should be modelled using SystemC, NS-2 or an instruction set simulator. However, again this approach is relatively tailored to networked embedded systems.

In summary, there is a lack in methodologies for designing suitable co-simulation configurations. This is due to the fact that standard static co-simulation usually does not have such a high complexity and is mainly driven by already existing simulation models which are about to be connected within a co-simulation.

2.3 Run-Time Simulation Model Switching

Very often during a system development process, an iterative refinement of simulation models is performed. High abstraction level models are replaced by accordingly low abstraction level models showing more details but also having less simulation performance. While this exchange is performed between two consecutive simulation runs, in run-time simulation model switching, the simulation model exchange is performed during the simulation process. The main idea is to have different implementations of the same module and to change the simulation model of a module while the simulation is executed. The idea behind run-time simulation model switching has been demonstrated within different areas, for example the simulation model of an MPEG4 encoder [58], for road traffic simulation [59], VLSI circuit simulation and power estimation [60] and even for dynamically adaptive software [61].

2.3.1 Run-Time Simulation Model Switching in Homogeneous and Heterogeneous Environments

The idea of run-time simulation model switching has been the topic of some research. Existing works mainly deal with switching between simulation models developed within the same modelling language (homogeneous environment) and at high abstraction levels. For example, in [62], [63], [64], [65] and [66], the authors propose switching between different SystemC TLM models. Within their implementation, the goal is to change the timing accuracy of TLM-based bus transfers during simulation. The selection of the level of accuracy can be performed either statically at elaboration time or dynamically during the simulation by using timing annotation macros in the source code. A similar approach is presented by Beltrame et al. [58]. Here, SystemC TLM is extended to support multi-accuracy models and power estimation with the possibility to switch between different model accuracies at run-time. This switch has to be performed manually. Hines et al. propose in [67] and [68] run-time simulation model switching for modelling of inter-module communication of real time systems. Their tool Pia allows the designer to specify at

multiple levels of detail the communication between components. The designer has to use the Pia modelling language proposed by the authors to describe the necessary interfaces and components for the various abstraction levels. During experiments, a speed-up in the area of 10-100 is achieved. Run-time simulation model switching is also proposed by Yoo and Jerraya [49]. They suggest dynamic switching between several abstraction levels of a processor simulation to improve co-simulation performance. However, no results or an actual implementation are available. Run-time simulation model switching is also used for the simulation of road traffic [59]. Here, Claes et al. implement several road traffic models using the same modelling language. Another example is given by Rao et al. [60]. They perform VLSI circuit simulation and power estimation by using run-time simulation model switching. Within their work, the simulation models are implemented using the same modelling language.

Current run-time simulation model switching focuses on switching within the same modelling language (homogeneous environment) and on high abstraction levels. No work is available regarding run-time co-simulation model switching in heterogeneous environments. As a result, run-time simulation model switching at its current level is not suitable to support the co-simulation of cross-domain automotive systems.

2.3.2 Adaptive Run-Time Simulation Model Switching

Run-time simulation model switching requires a command to actually change the simulation model. This command can be given manually (like in some of the examples within the previous section) or given automatically according to the current simulation model condition. This adaptive run-time simulation model switching is currently only supported in a few implementations. However, it allows the switching to be more flexible, and hence, more effective in terms of simulation performance.

In [66], Radetzki et al. automatically adapt a SystemC TLM simulation model to cycle accuracy if bus transfers are initiated or completed. In [62], the same authors use annotation macros within the source code to adapt the model accuracies during the simulation execution depending on the current state. Within [59], Claes et al. perform a run-time simulation model switching within a road traffic simulation. They use the traffic density as criterion and define a lower and upper bound. If the traffic density passes the boundary, a switch to the high or low detailed simulation model is triggered. If the traffic density stays between the two bounds, the current model is continued to be used. Another example for adaptive run-time simulation model switching is presented in [60] for VLSI circuit simulation including power estimation. Within this work, two different types of adaptive switching are described. Proactive switching is used if the power values stay the same for some time. This causes a change of the simulation model used at the next cycle. The second type is reactive switching. If power estimates are missing, an immediate simulation model change is performed. This includes a roll-back of the simulation and reprocessing of data, causing massive simulation overhead but preventing the loss of power estimates.

It can be seen that there are several ways to perform adaptive run-time simulation model switching. Typically, a set of criteria is required which, however, within all the examples, is extremely tailored for a specific use case. Hence, no generic criteria-based adaptive run-time simulation model switching is available which would be required to apply adaptive run-time simulation model switching to a multitude of applications.

2.4 Summary

With the exploding complexity in automotive systems, it is obvious that simulation gains more and more importance within the development process. Simulation allows to have a closer look at the system, eliminating the black box behaviour of real hardware. The functionality can be verified in advance, hence reducing development effort due to avoiding costly redesigns and leading to faster time to market. However, as demonstrated within this chapter, standard simulation is not enough to handle the requirements for a holistic analysis of automotive systems due to several limitations (e.g. simulation performance, level of detail, interacting components, cross-domain interactions). A possible solution would be co-simulation, the simulation of different simulation models implemented by using different HDLs and abstraction levels within a common simulation. However, current co-simulation is not able to sufficiently deal with the integration of simulation models coming from domains with very different time constants. But the effective simulation of complex cross-domain systems is required for the analysis of automotive systems of today. A possible solution would be run-time simulation model switching. By changing the simulation models used for a module during simulation, it would be possible to overcome several of the challenges of cross-domain co-simulation within a heterogeneous environment. However, current run-time simulation model switching is only available on high abstraction levels within homogeneous environments, and the criteria to perform the more effective adaptive simulation model switching cannot be defined in a generic way.

The goal of this thesis is to develop a methodology for the co-simulation of cross-domain automotive systems based on run-time simulation model switching. The main objectives are:

- Design of a flexible co-simulation methodology based on RCMS. Specification of time-triggered RCMS and adaptive RCMS to allow for the efficient integration of multiple domains into a common co-simulation.
- Development of a RCMS-based methodology for the analysis of cross-domain automotive systems: design of suitable simulation models and a suitable RCMS configuration to fulfil specified system analysis goals.
- Development of a comprehensive case study: cross-domain automotive co-simulation framework for the analysis of dependable automotive communication systems.

Chapter 3

Novel Methodology for Co-Simulation of Cross-Domain Automotive Systems

Overview

As previously explained, the challenges for the co-simulation of cross-domain automotive systems cannot be simply handled by using existing techniques. In the following, a new methodology for the co-simulation of cross-domain automotive systems is proposed. Standard static co-simulation is enhanced by introducing RCMS. This allows to efficiently bringing together different domains and simulation models having very different time constants in a single co-simulation. By providing a methodology for the design of a suitable RCMS configuration for given system analysis goals, acceptable co-simulation performance and the required accuracy of the simulation results can be achieved at the same time.

In Fig. 3.1, an overview of the proposed methodology for the co-simulation of cross-domain automotive systems by using RCMS is given. For specified system analysis goals, this RCMS-based methodology allows the design and implementation of suitable simulation models for obtaining the required accuracy of the results while at the same time optimising co-simulation performance by using RCMS. While some steps are performed by using already existing state-of-the-art approaches (e.g. implementation of simulation models), other steps are completely new or largely adapted (e.g. all steps regarding RCMS). Within Chapter 6, individual parts of the methodology are discussed in more detail. Section 6.1 provides an overview of the complete RCMS-based methodology for the co-simulation of cross-domain automotive systems. Additionally, the topics of adaptive RCMS and design of a suitable RCMS configuration are discussed in detail. Within Sections 6.2, 6.3 and 6.4, the fundamentals of RCMS are presented in detail and a validation of RCMS is performed. Sections 6.5-6.7 show the design and implementation of simulation models for a cross-domain automotive system. Additionally, cross-domain automotive co-simulation integration within an RCMS enabled framework is discussed and results are presented.

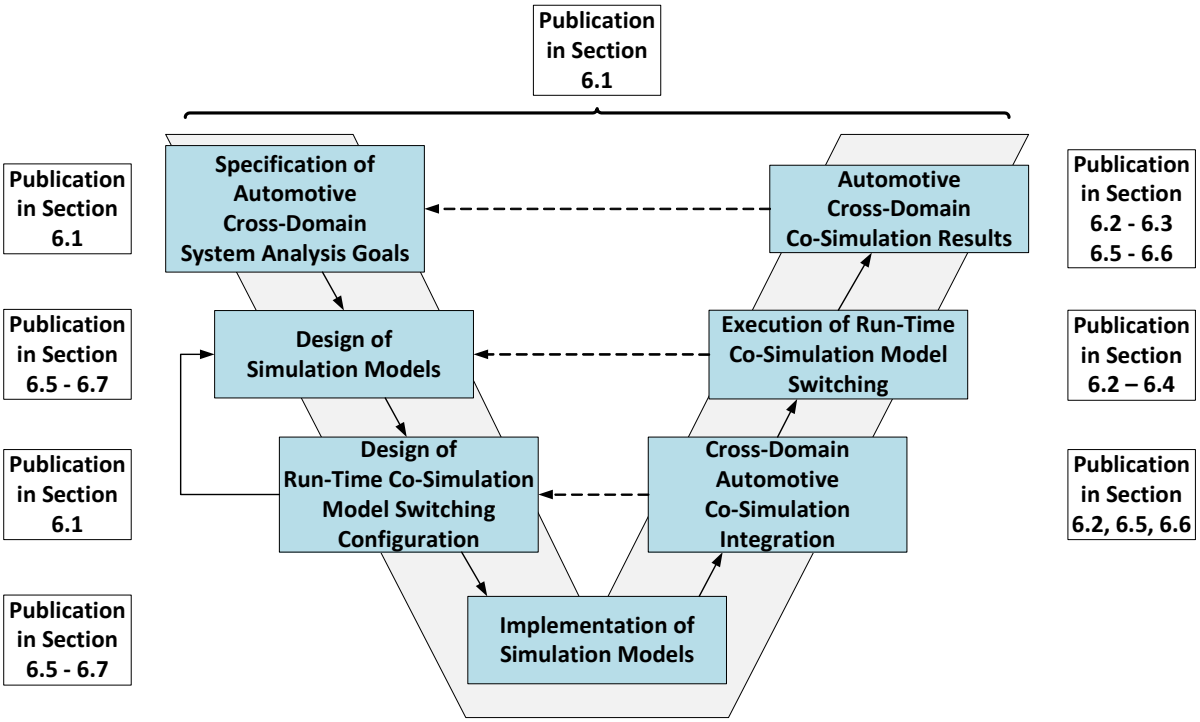


Figure 3.1: Co-Simulation of Cross-Domain Automotive Systems

Run-Time Co-Simulation Model Switching

The idea behind RCMS is to change the simulation models that are used for a specific part of the co-simulation (a module, see Fig. 3.4) at run-time. This allows to speed up the co-simulation while at the same time providing high accuracy when required. It provides the developer with a new degree of freedom for defining the intervals for each module for which a computational expensive high-detail simulation model is required and when a simpler and faster model suffices ('shifting of complexity', see Fig. 3.2). Within this thesis, the existing co-simulation framework CISC SyAD® has been enhanced to support RCMS.

The possibilities of RCMS can be compared with to oscilloscope, having the capability to zoom into significant parts of the simulation, while performing a fast (less accurate) run for the less interesting part of the simulation. And like an oscilloscope has a trigger condition, a trigger condition is also required for RCMS. The following criteria can act as a possible trigger source.

- Time (time-triggered RCMS)
- Events generated by an online signal analysis performed by the provided RCMS implementation (adaptive RCMS)
- Events generated by an external analysis logic (complex system specific criteria, adaptive RCMS)

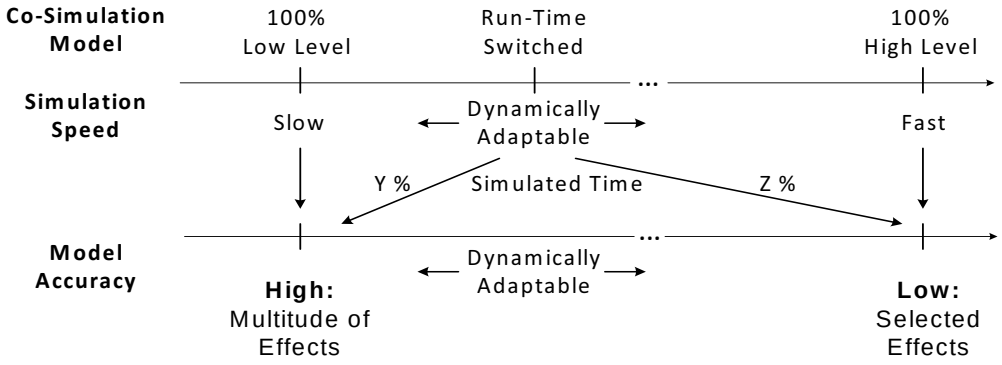


Figure 3.2: Shifting of Complexity

Time-triggered RCMS uses a-priori defined switching points, with time acting as trigger source. Hence, it may not be applicable to all types of systems or simulation models. However, it has one important advantage: Based on the principle of shifting of complexity (see Fig. 3.2), the developer has the possibility to simply adjust the trade-off between the simulation performance and the model accuracy achieved. This is a great benefit, as in fact the developer can specify the simulation performance in advance, hence adjusting the simulation time needed. By specifying the simulated time (including concurrent simulation time) for each simulation model of the module, the developer can select the relation between simulation performance and accuracy according to the requested needs. Thus, the simulation performance improvement factor in comparison with standard static co-simulation methodologies is more or less freely selectable by the developer. This is of great advantage when simulating complex systems. It is possible to estimate the resulting overall co-simulation performance for time-triggered RCMS in advance.

While time-triggered RCMS is rather simple to handle, it has some major drawbacks. It is a relatively inflexible approach as it has to be defined before the simulation starts. It is not able to react to spontaneous events and also requires predictable system behaviour for efficient switching. It also introduces additional overhead as switching takes place according to the current time and not if the current situation would really require it. More information about time-triggered RCMS can be found in Sections 6.2 (*Optimizing HW/SW Co-Simulation based on Run-Time Model Switching*) and 6.3 (*Holistic Simulation of FlexRay Networks by Using Run-Time Model Switching*).

In **adaptive RCMS**, the current system behaviour acts as trigger source for switching. The idea behind it is to perform a run-time analysis of the behaviour of defined signals within the co-simulation. Depending on the analysis results, a switch to the most suitable simulation model for a module is initiated. The run-time signal analysis is performed by comparing the current signal behaviour to a set of criteria defined by the developer. If a criterion matches during the analysis, a switch to the according simulation model is initiated. This principle is shown in Fig. 3.4. The actual process of run-time signal analysis is performed by using a sliding window. Hence, not only the current value but also values from the past can be included within the analysis process. Within the current implementation, the following signal analysis criteria are supported for adaptive RCMS.

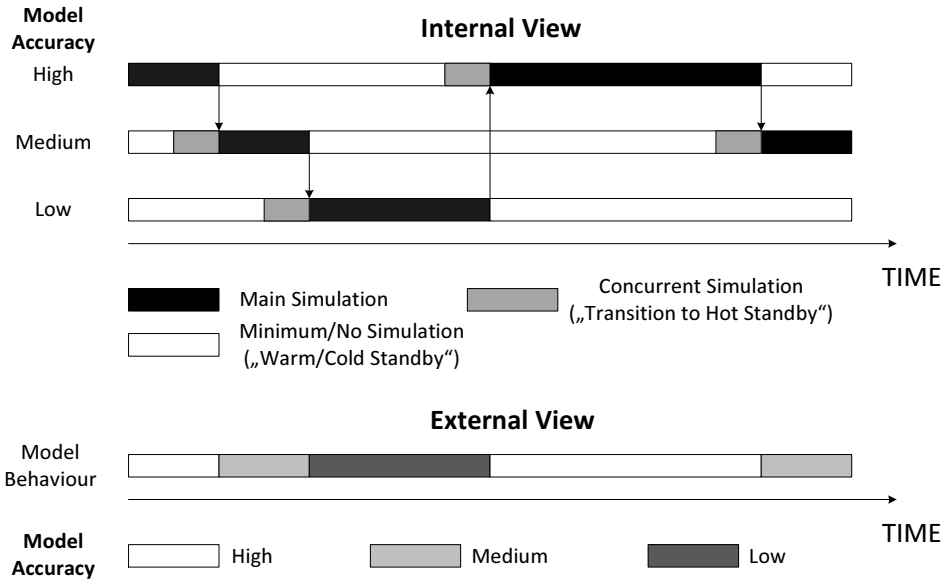


Figure 3.3: Run-Time Co-Simulation Model Switching Synchronisation

- Arithmetic mean of the signal values within the sliding window
- Average frequency of signal value changes within the sliding window
- Average value change between two consecutive signal values within the sliding window
- Standard deviation within the sliding window

Another way for adaptive switching is to use an external analysis logic as a trigger source. This may be required for triggers caused by very complex and system-specific conditions which cannot be mapped to the criteria supported by the standard adaptive RCMS.
More information about adaptive RCMS can be found in Section 6.1 (*Run-Time Co-Simulation Model Switching for Efficient Analysis of Embedded Systems*).

In order to assure correct continuous service delivery of a module during switching, RCMS requires **synchronisation** between the different simulation models of a module. There are several ways of how the synchronisation of the upcoming model can be performed. Typically, it is performed via concurrent simulation upon which the upcoming simulation model is fed with the current data but without actually using its calculated results (see Fig. 3.3, concurrent simulation time). This allows for a synchronisation of the upcoming model so that switching can take place seamlessly. During this synchronisation process, no data is exchanged between the current and the upcoming model. A trade-off exists between, on the one hand, setting the concurrent simulation time long enough in order to allow for a proper model initialisation, and on the other hand, minimising concurrent simulation in order to reduce simulation time. To make the process of synchronisation more clearly, a comparison to the area of redundancy can be made. The currently active model is the main model. If, due to a switching request, the concurrent simulation phase is about to start, the upcoming model is equal to cold standby (or in some cases warm

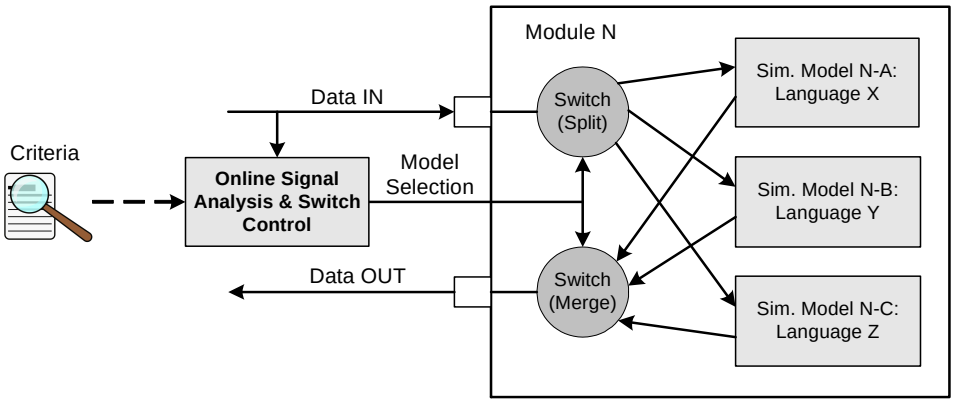


Figure 3.4: Run-Time Co-Simulation Model Switching

standby). The synchronisation process ensures that the upcoming simulation model gets into hot standby by supplying it with the simulation data flow for a certain amount of time (concurrent simulation time). By doing so, seamless switching between current and upcoming model can be performed. More in-depth information about the synchronisation process during RCMS can be found in Section 6.2.

The actual execution of RCMS is performed by using SystemC-based switches. These switches are responsible for splitting and merging the data flow during RCMS. Fig. 3.4 demonstrates the switching principle for RCMS. The switches are controlled by a switch control unit. The switch control unit is responsible for reacting to trigger sources accordingly and to perform the online signal analysis. Additionally, the switch control unit handles the required actions for switching like start of synchronisation or configuration of the switches. More information about the RCMS implementation and execution can be found in Sections 6.1 and 6.2.

A comprehensive overview of all aspects of RCMS can be found in Sections 6.1 - 6.4 (*Verfahren zum Umschalten von heterogenen Simulationsmodellen zur Laufzeit*).

Specification of Automotive Cross-Domain System Analysis Goals

The specification of the automotive cross-domain system analysis goals is essential to make sure that the following steps develop in the right direction. For this step, there are no parameters on how to perform it. It can be done by producing a simple text, but also by using special description languages and tools. However, it is of great importance that at least the following information is specified exactly.

- What is the system to be analysed?
- What are the goals of the analysis?
- Are there special operations that have to be examined in detail?
- What are the requirements on the simulation process itself (e.g. high simulation performance because of the long time interval of interest)?

After these questions will have been answered in detail, the following steps will be much easier to complete and the results of the methodology are greatly improved. Examples for system analysis goals specification are shown in Section 6.1.

Design and Implementation of Simulation Models

Based on the specified system analysis goals, the initial design of simulation models is performed. The simulation model developer specifies how the simulation models should look like, what their properties are and so on. For this, well-known standard tools and methodologies can be used. This step can be seen as functionally linked to the step of designing a RCMS configuration (see later on) as there is a feedback-loop connecting them. After finishing the design of simulation models and RCMS configuration, it is required to perform the implementation of the simulation models according to the created design. This is quite a straight-forward process as there are lots of well-known guidelines for this. Hence, no special directions are given and it is up to the simulation model developer to use any suitable technique for implementation.

Within Sections 6.5-6.7, the design and implementation of simulation models for a cross-domain automotive system is shown by the example of the TEODACS project.

Design of a RCMS Configuration

When using RCMS, there are several challenges to be met. One of the most important challenges is the definition of a suitable RCMS configuration for a given system analysis goal with respect to the available simulation models. Especially in complex systems structured in several subsystems, it is not an easy task to define an adequate configuration: There are different switching strategies possible (standard static co-simulation, time-triggered RCMS and adaptive RCMS). The strategies can be mixed and have to be parameterised correctly to produce the desired result quality. Additionally, it has to be assured that the simulation models are able to fulfil the specified system analysis goals and requirements.

To overcome these problems, a new methodology for the design of run-time co-simulation model switching configurations with respect to given system analysis goals and requirements is proposed. Additionally, this methodology provides feedback for the design of simulation models to enhance their capabilities if required (see Fig. 3.1). The main idea is to apply adapted safety-analysis techniques like failure mode and effects analysis (FMEA), subsystem hazard analysis (SSHA) and system hazard analysis (SHA) to the available simulation models and analysis requirements. Further on, these adapted techniques are called switching-adapted FMEA (swFMEA), switching-adapted SSHA (swSSHA) and switching-adapted SHA (swSHA). As the methodologies of FMEA, SSHA and SHA are adapted for the design of a RCMS configuration, also the according terms may have different meanings. A *failure mode* is a 'risk' caused by a simulation model. A possible failure mode would be for example 'missing of disruptive effects compared to reality'. A *hazard* is defined as the potential for harm, hence a switching configuration not suitable for the desired system analysis purpose. An example for such a hazard is a simulation performance way below the required level or that important effects are not covered by the current simulation model(s) in use.

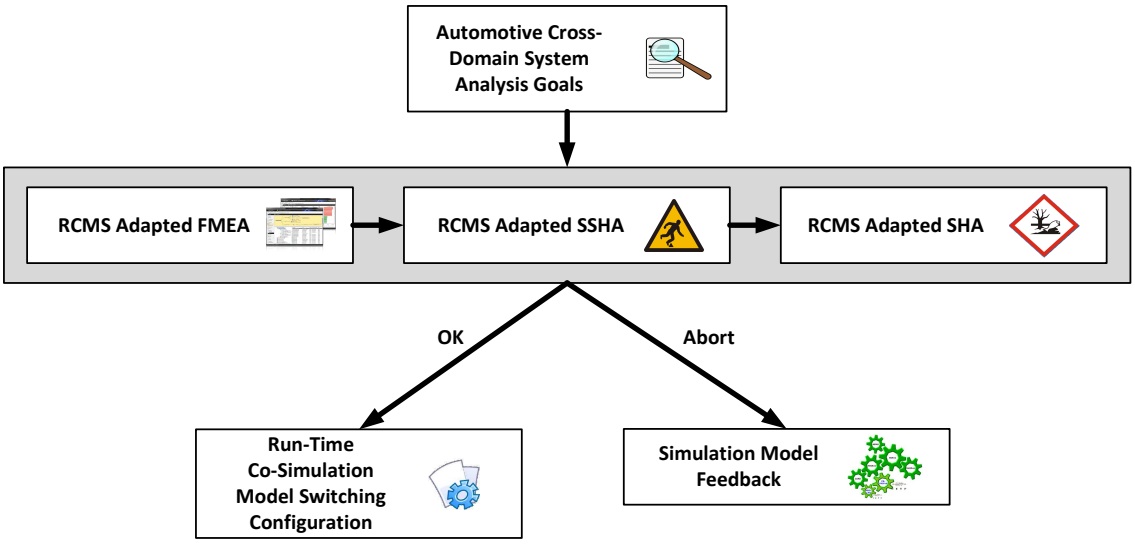


Figure 3.5: Design of a Suitable RCMS Configuration

The methodology supports the developer in finding the most suitable RCMS configuration for the specified system analysis goals. If no suitable switching configuration can be found, information is given why the current analysis goals cannot be fulfilled. This information can be used to adapt the simulation models. The main advantage of this new methodology is that it provides a structured procedure to design the RCMS configuration based on well-known techniques. Additionally, the developer is able to reuse information gathered during product FMEA for the swFMEA (e.g. clustering in subsystems and modules, system information, possible failure modes, use cases...).

Fig. 3.5 shows the proposed methodology for the design of RCMS configurations with respect to given system analysis goals and requirements. It is structured into three main steps. The first step is to execute the *swFMEA*. In this step, all simulation models are analysed for the failure modes that they may introduce into the system if they are used (e.g. 'faulty simulation results compared to reality in case of using incorrect topology') and how it can be dealt with these failure modes. This step is rather generic and if performed once, the data can be reused for other system analysis goals. Additionally, data gathered during a standard product FMEA can also be reused during the swFMEA, hence reducing workload for the developer.

For each subsystem, the *swSSHA* has to be performed. The goal of the swSSHA is to create an RCMS configuration within the subsystem by identifying hazards caused by simulation models and the possible model switching configuration which would avoid the achievement of the system analysis goals. If there are irresolvable hazards, feedback for the design of simulation models is created.

The last important step is to execute the *swSHA*. It is applied to the overall system and identifies hazards that apply to more than a single subsystem and cannot be identified during the swSSHA. The main focus of the swSHA is, according to the possible hazards at system level, to adapt the run-time switching configuration derived during the swSSHA and generate the final RCMS configuration. Upon successful completion, a suitable RCMS

configuration for the specified system analysis goals is created. If this step cannot be completed, detailed feedback for the design of simulation models is provided by the list of unresolved hazards. For performing swFMEA, swSSHA and swSHA, typical work sheets as used in the industry have been adapted and extended. They are now tailored to the specific requirements of the adapted techniques.

In-depth information about the methodology of designing a RCMS configuration for given system analysis goals, especially about the individual steps of swFMEA, swSSHA and swSHA, is included in Section 6.1.

Cross-Domain Automotive Co-Simulation Integration

This step deals with the integration of the simulation models, simulation tools and the developed co-simulation configuration into a common co-simulation. Several co-simulation environments are available in academia and industry. However, by default, none of them supports RCMS. We decided to use the commercial co-simulation environment SyAD® by CISC Semiconductor and enhanced it to support RCMS. The co-simulation integration is a tool-specific task; hence, no general guidelines are given. However, the developed RCMS configuration basically can be used within every co-simulation environment supporting RCMS. As shown in Fig. 3.1, in case of detected problems, it is possible to go back within the methodology and modify the co-simulation configuration accordingly.

Sections 6.2, 6.5 and 6.6 contain information about how the cross-domain automotive co-simulation integration is performed (RCMS, multiple domains). As example, the integration of a car simulator (focus mechanics) into the existing co-simulation framework (focus microelectronics) is demonstrated.

Execution of Run-Time Co-Simulation Model Switching

Here, after the desired RCMS configuration has been set up, the simulation models are executed. The actual RCMS takes place within this step. In case the behaviour of the simulation models and/or the overall co-simulation does not fulfil the expectations (because e.g. the developer made some mistakes during the previous steps or the simulation performance has been overestimated), it is necessary to go back to the phase of design of simulation models, perform the required adaptations and repeat the following steps.

Automotive Cross-Domain Co-Simulation Results

Within this step, the results gained during the execution of the co-simulation are analysed. For this, again, there is no special direction on how to do it. It mainly depends on the output and the result files produced by the simulation models. In case of results not fulfilling the specified system analysis goals (e.g. important effects are missing), it is required to go back to the first step of the methodology (see Fig. 3.1). There, the system analysis goals have to be specified more precisely. During the following steps, the data has to be changed to fit the adapted system analysis goals.

In Sections 6.1 - 6.3, 6.5 and 6.6, the analysis of results gathered during a co-simulation of cross-domain automotive systems is shown in detail.

Chapter 4

Methodology Evaluation and Case Studies

To demonstrate the proposed RCMS-based methodology, it is applied within a co-simulation of a FlexRay-based cross-domain automotive system (see Sections 6.5-6.7). This co-simulation has been developed during the TEODACS project (see Section 1.2.1 for more information). FlexRay is a time-triggered automotive communications protocol operating at data rates up to 10 Mbit/s. Its reliability is highly dependent on the signal integrity. The accurate modelling of the causative effects leads to extensive physical level simulation times even for very short simulated times (e.g. several days or weeks of simulation time for a simulated time of just a few milliseconds). The time window of interest at system level can cover several minutes (typical automotive control applications), however accurate physical models are required to analyse low-level effects and interferences. Because of the wide range of requirements and the enormous complexity, co-simulation and especially RCMS are required to handle this topic.

After presenting two examples, the accuracy of the simulation results achieved by using RCMS is compared to the accuracy of the simulation without using RCMS. Finally, the accuracy of the FlexRay simulation models compared to the reality is analysed.

4.1 Co-Simulation of FlexRay-based Cross-Domain Automotive Systems

Two examples will demonstrate the RCMS-based methodology for the co-simulation of cross-domain automotive systems. In the first example, a distributed application running within a car is to be analysed for delays because of the placement of functionality on different ECUs. In the second example, the effects of FlexRay signal integrity are under observation.

4.1.1 Example 1: Distributed Application Delay Analysis

Supported by standardised methodologies like AUTOSAR, distributed applications are gaining more and more importance within cars. The functionality of the distributed application is placed on different ECUs according to factors like computational power, data

availability and even costs. However, due to this distribution, the impact of the communications architecture is not negligible anymore. For example, the resulting delay within the distributed application is largely influenced by the architecture and configuration of the underlying communications network.

Specification of System Analysis Goals Fig. 4.1 shows an example of a distributed control application which is to be analysed by the RCMS-based methodology shown in Fig. 3.1. The application uses AUTOSAR concepts and FlexRay as communication system. It is embedded within a powertrain-focused simulation model of a complete car. For this system, the following system analysis goals and requirements are specified.

- Goal: analysis of the delay within an automotive distributed application (cause/effect, placement of functionality within the network, sender/receiver) during critical situations (e.g. rapid change of application input data).
- Goal: analysis of the effects of different FlexRay communication configurations on the data transmission within the distributed application.
- Requirement: best possible simulation performance should be achieved as it will be required to perform several simulations (time interval under observation: up to several minutes of simulated time).
- Requirement: data transmission delay & quantisation within the distributed application have to be modelled correctly during critical situations to allow for a realistic analysis of the application behaviour.

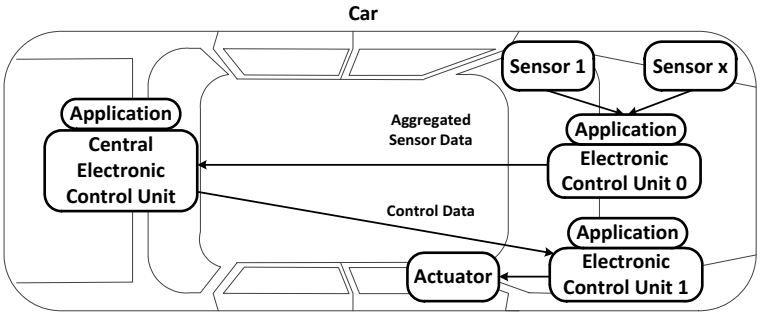


Figure 4.1: Case Study: Automotive Data Transmission for Analysis of Delays within a Distributed Application

After gathering information about the possible simulation models (e.g. several SystemC and VHDL-AMS-based models for the components of a FlexRay network, SystemC-based simulation of AUTOSAR concepts) and defining possible simulation tools (e.g. for the realistic simulation of the car), the next step is to design the RCMS configuration according to the system analysis goals.

Design of RCMS Configuration The design of the RCMS configuration is executed as shown in Fig. 3.5. As the system analysis goals have been already specified, the swFMEA

is executed. For a module like the FlexRay cable, the resulting swFMEA for one simulation model looks as follows:

- *ID, Module*: 3, FlexRay cable
- *Simulation model*: SystemC
- *Simulation performance*: very high
- *Use case*: transmission of a digitised representation of an analogue signal within a correct topology
- *Failure mode*: faulty simulation results in comparison to reality in case of using incorrect topology
- *Cause of failure*: only length-based delay and attenuation taken into account
- *Low level effects*: signal transmitted performed correctly even if it should be faulty
- *Subsystem level effects*: data transfer correct within a flawed FlexRay system
- *Compensating actions*: do not use within a network having a faulty topology
- *Value range limitations*: none

Next, for each subsystem, the swSSHA is performed to determine possible hazards within the subsystem and to create a preliminary RCMS configuration. Within this example, the result for the module FlexRay communication controller is shown:

- *Use case*: execution of the logical FlexRay protocol for data transmission
- *Requirements*: high simulation performance, correct protocol execution, support of different communication configurations
- *Hazards for subsystem*: simulation performance too low
- *Cause for hazard*: internal high-frequency clock of the communication controller in the SystemC model
- *Effects caused by hazard*: only medium simulation performance
- *Simulation model*: SystemC
- *Switching parameters*: SystemC model for critical data (as defined by the application), parameters: concurrent simulation, 30.0 ms concurrent simulation time, hysteresis 200.0 ms;
- *Possible subsystem hazards*: missing of any delay information for non-critical data can make the distributed application work unexpected; internal clock of communication controller always running even if no data is transmitted
- *Recommended switching strategy*: use SystemC only if critical data (as defined by the application) is to be transmitted, otherwise switch to a simple fixed-delay model for data transmission (not FlexRay-based)

- *Open Hazards*: none, if simple fixed-delay model is implemented and internal clock of the communication controller can be halted
- *Feedback to simulation model design*: temporarily support halt of the high-frequency clock of the controller (SystemC); create a simple fixed-delay transmission model for data transmission of non-critical data

It can be seen that also feedback for the design of simulation models is created in this example. A simple fixed-delay model for transmission would be required to fulfil the system analysis specification, but this model does not yet exist. Hence, it has to be implemented so that the methodology can continue. This has been done during a second iteration, so the methodology can continue for this example.

After there will be no more open hazards, the next step is the execution of the swSHA to generate the final RCMS configuration. For example, an entry within the swSHA database looks as follows:

- *Use case*: data transfer between a distributed application within a car using AUTOSAR concepts and FlexRay
- *Requirements*: high simulation performance, accurate delay modelling
- *Hazards for system*: critical data transmitted via simple transmission model instead of FlexRay model
- *Cause for hazard*: no classification of critical data done yet
- *Effects caused by hazards*: analysis of the behaviour of the distributed application delivers wrong results
- *Recommended adaptation to switching strategy*: Implement analysis for critical application data (engine torque). If engine torque changes rapidly (= critical data, $|\Delta torque| > 0.3$ Nm per millisecond), switch to FlexRay model. During intervals of low data variability (= non-critical data), switch to the simple fixed-delay transmission model.
- *Open hazards*: none
- *Feedback to simulation model design*: none

In the end, the following RCMS configuration has been designed: The application is simulated using a SystemC-based model implementing AUTOSAR concepts. The car is simulated using a suitable car simulator integrated into the co-simulation environment. For the transmission of data within the distributed application, adaptive RCMS is used. If critical data is to be transmitted (rapid change in engine torque, $|\Delta torque| > 0.3$ Nm per millisecond), a complete SystemC FlexRay model is used (controller, transceivers, cables). For non-critical data (low variability in engine torque), a simple fixed-delay transmission model is enough. The window size for the run-time signal analysis is of size 15 and updated by a sample clock every millisecond. This RCMS configuration allows for precise simulation of delay and configuration effects during critical areas of the simulation while at the same time having a relatively good simulation performance. The definition of critical data is done in a way to allow for a correct synchronisation of the upcoming model in time (slightly lower thresholds as would be required by the application).

RCMS Integration and Execution After integration into a common co-simulation and execution of the RCMS the results can be analysed. Compared to the standard static approach, the RCMS simulation performance is about 4 times faster. This factor, however, depends on the amount of critical data transmitted as adaptive RCMS is used. In Fig. 4.2, the detected areas of critical data are shown during the first 15 seconds of simulation. The 'high-detail' signal set to 1 indicates that the FlexRay simulation models are used instead of the simple delay-based simulation model. It can be clearly seen that no critical data is detected during areas of low engine torque variability.

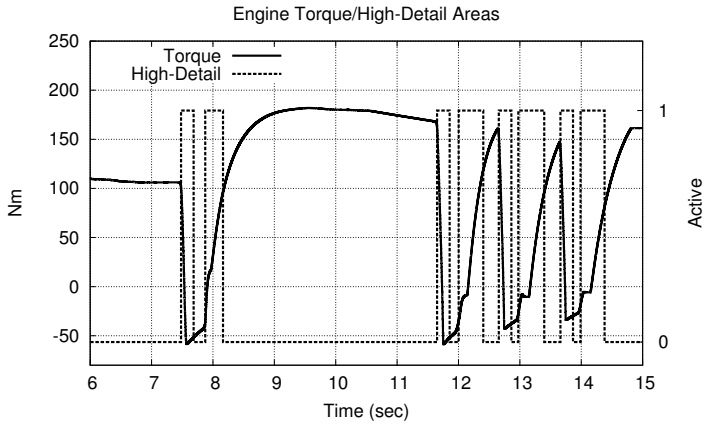


Figure 4.2: Adaptive Switching Criterion (Engine Torque) Mapped to Detected Critical Areas

RCMS Results Analysis Based on the designed RCMS configuration, it is now possible to analyse the delay behaviour of the distributed application, fulfilling the specified system analysis goals. As the functionality of the application is placed on different ECUs, it is interesting what the resulting delay is compared to a non-distributed application placed on a single ECU. The overall delay is mainly affected by the number of ECUs the application is distributed over and the FlexRay communication configuration used. As FlexRay is a time-triggered communication system, its configuration specifies the transmission slots that can be used by each FlexRay controller. By varying the number of slots used per FlexRay controller and the position of the slots within a FlexRay cycle, the update behaviour of the data transmitted can change drastically. This, of course, also massively impacts the resulting overall delay of the distributed application. It is not a good solution to just enlarge the number of slots used by a certain controller (and, in fact the ECU connected to the controller) as the total number of available slots is limited. Hence, the possible slots for other controllers connected to the network would be reduced, resulting in delay penalties for the applications running on the according ECUs. In fact, a good compromise for the FlexRay configuration has to be found.

Within the developed RCMS setup, this analysis can be easily done during a reasonable amount of time. As an example, Fig. 4.3 shows the resulting delay for a control signal sent within the distributed application. The sensor data is collected by ECU0, aggregated and sent via the FlexRay network to another part of the distributed application running on the

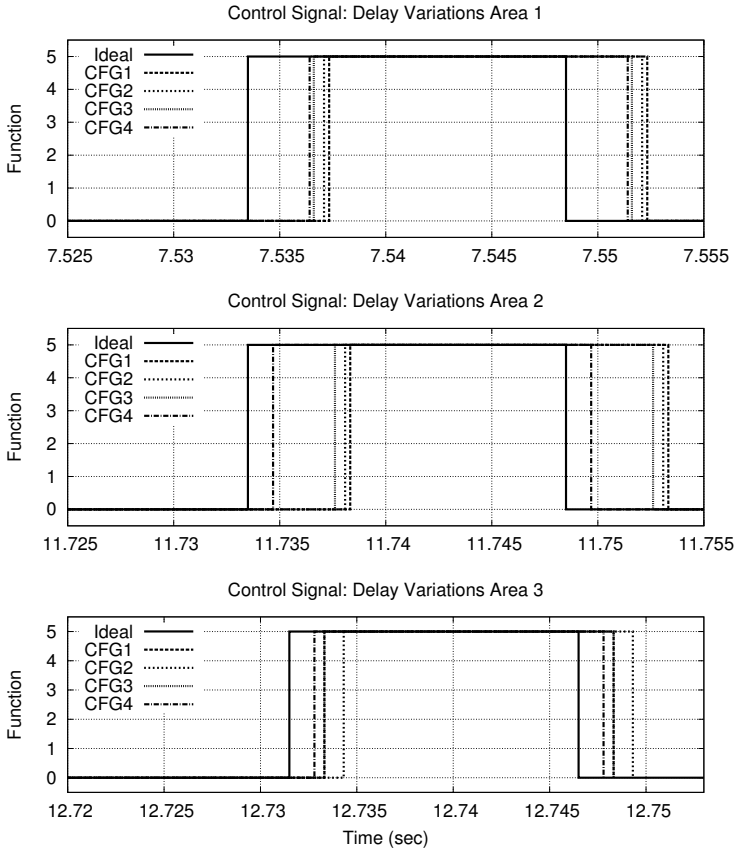


Figure 4.3: Control Signal Delay for Different FlexRay Configurations

central ECU. There, the incoming data is analysed, and for certain conditions, a control signal is generated and sent to ECU1 via FlexRay. The ideal control signal shown in Fig. 4.3 is created by a non-distributed application running completely on ECU0. It is compared to the control signal generated by the distributed application where data is transmitted using FlexRay with different communication configurations. This is done for three different areas during the simulation. All of these areas are within the specified critical regions (rapid change in engine torque), as the control signal is only triggered during these areas. It can be seen that, generally spoken, with FlexRay configuration 4, the application shows the best delay behaviour. It can also be seen that the absolute value of the delay between the ideal signal and the different FlexRay configurations is not constant. While in area 2, the delay of FlexRay configuration 4 is by far shorter than the delay of the other configurations, the delay difference is not that extreme in areas 1 and 3. This is because of the different update behaviour and slot allocations within the FlexRay cycle of the four configurations. Hence, in certain situations, a specific FlexRay configuration may lead to low delay, while in other situations (data available only some moments later), the same configuration may cause a much larger delay within the distributed application.

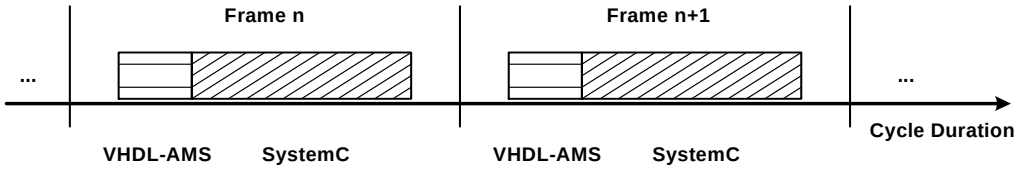


Figure 4.4: Time-Triggered Switching within a FlexRay Frame

4.1.2 Example 2: Analysis of FlexRay Signal Integrity Effects

The second example deals with application behaviour for a borderline FlexRay topology. The reliability of FlexRay is highly dependent on the signal integrity during data transmission. Hence, it would be interesting to have a closer look at the influence of signal integrity effects on the behaviour of a distributed application, for example, the investigation of effects occurring at EPL on higher layers like data link layer (e.g. shortening/lengthening of bits, misinterpretations because of low signal integrity) and application (e.g. missing data because of frame corruption, problems due to start-up/synchronisation errors etc.). As stated during the first experiment, several simulation models for the different FlexRay components are available, featuring different levels of accuracy and implemented using different modelling languages like SystemC and VHDL-AMS. Again, by applying the RCMS analysis and design methodology, a suitable RCMS configuration should be created to allow for an analysis with respect to the specified system analysis goals and requirements.

Specification of System Analysis Goals For this example, the following system analysis goals and requirements are specified.

- Goal: analysis of the behaviour of a distributed application for a borderline FlexRay topology
- Goal: analysis of the effects of signal integrity problems (e.g. reflections) on the application and the network behaviour
- Requirement: simulation performance has to be acceptable (no extremely low simulation performance) because the time interval of interest can be up to several seconds

RCMS Configuration Design, Integration and Execution Again, the methodology shown in Fig. 3.1 is used. For this example, adaptive RCMS would lead to nearly no improvement of the co-simulation performance. However, as FlexRay is a time-triggered communication system, time-triggered RCMS can be used. There are pre-defined points for which the state of the system is completely known. Hence, it is the most advantageous solution to select the switching points accordingly with the concurrent simulation time large enough to allow proper initialisation of the simulation model to be switched to.

It was derived that most of the time, the fast but less detailed SystemC model for topology and transceivers should be used. However, in every frame, the header (6μs) should be transmitted via the more detailed but slow VHDL-AMS simulation models for cable and transceivers (concurrent simulation time = 2μs). This is shown in Fig. 4.4. The signal integrity is extremely unlikely to change within the duration of a FlexRay frame

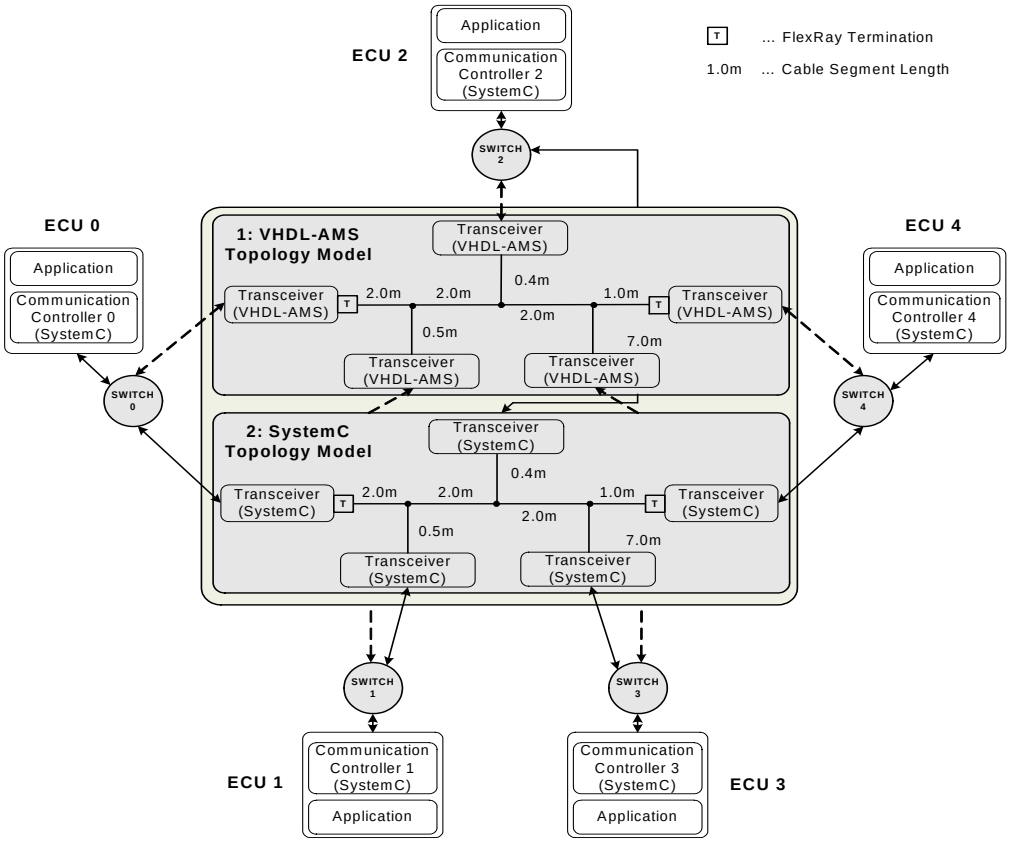


Figure 4.5: Co-Simulation Setup for FlexRay Signal Integrity Effects Analysis

	VHDL-AMS	Switched	SystemC
Seconds	88012	31757	307
Factor	287	104	1

Table 4.1: Example Simulation Time Comparison for one FlexRay Cycle with 12 Frames

(26μs). For the analysis of signal integrity effects, mainly the header of a FlexRay frame is relevant, as all protocol-relevant is contained within the header. All other parts of the network (FlexRay controller, AUTOSAR-based distributed application...) are simulated using SystemC without switching. Hence, the configuration is able to fulfil the specified system analysis goals.

The final experimental setup is shown in Fig. 4.5. The network consists of 5 ECUs connected via a FlexRay network. A distributed application runs on these ECUs using AUTOSAR concepts. Within this network, ECU3 uses a long cable which is, against the FlexRay specification, not equipped with the required termination. This may lead to disruptive reflections which influence the signal integrity.

To get the simulation performance improvement by using RCMS instead of standard co-simulation, experiments with using the high-detailed VHDL-AMS models throughout the complete simulation were performed additionally. Another simulation run included only

```
Node 3 - codec: BSS_ERROR!
Time: 285747500 ns

Node 2 - WARNING CODEC CHANNEL A: HEADER CRC FAILED!
Time: 285751650 ns

Node 1 - WARNING CODEC CHANNEL A: HEADER CRC FAILED!
Time: 285751662500 ps

Node 0 - WARNING CODEC CHANNEL A: HEADER CRC FAILED!
Time: 285751675 ns
```

Figure 4.6: Communication Controller Error Detection Output Excerpt

SystemC models, without switching to the VHDL-AMS models for the FlexRay header. Table 4.1 shows a comparison of the results. While SystemC-only simulation is clearly the fastest, it features no details required for the analysis of the signal integrity effects on the distributed application. Hence, it is fast but does not attain the specified system analysis goals. By contrast, the VHDL-AMS only configuration has all the required details but at the cost of an extremely low simulation performance. Compared to the VHDL-AMS only configuration, the RCMS configuration simulating only the header by using VHDL-AMS is about three times faster. The effects on the behaviour of the FlexRay network and the distributed application are the same. Hence, the designed RCMS configuration presents an acceptable solution for the specified system analysis goals and requirements.

RCMS Results Analysis The executed RCMS demonstrated that, depending on the transmitting ECU, different ECUs have problems in receiving correct frames. For example, the data sent by ECU 2 can be received without any problems by ECUs 0, 1 and 4, while ECU 3 reports a header error check failure. However, if ECU 1 transmits, the ECUs 0, 2 and 3 receive the data correctly, ECU 4 reports an error in the frame header. These effects lead to reception asymmetry that can transfer the system into inconsistent states and should thus be avoided.

In Fig. 4.7, it can be seen that for a frame transmitted by ECU 4 (TxD4), ECU 3 using the VHDL-AMS model is not able to receive any meaningful data on RxD3 because of reflections etc. Hence, the communication controller very early reports an error that it is not able to find a byte start sequence (BSS), see Fig. 4.6. The problems can easily be detected by looking at the resulting differential voltage waveform at ECU 3. It is far from any meaningful state. In Fig. 4.7, the SystemC topology model starts concurrent simulation at around 285.749ms before switching at 285.751ms. During concurrent simulation, the output of the SystemC model is not used. It can be seen that the SystemC model does not show any distortion (because it only includes length-based delay and attenuation). Hence, no error would be detected by using the SystemC model all the time. The situation is a bit different for ECUs 0, 1 and 2. In fact, using the VHDL-AMS model, they are able to receive the header data (transmitted by ECU 4) correctly most of the time. But at around 285.750ms, one bit-sequence becomes altered by a reflection. The high bit-sequence is shortened, and the following low bit-sequence is stretched, leading to a header checksum error detected by the controllers, see Fig. 4.6. By looking at the SystemC data, it is obvious that this error does not happen there. These errors definitely influence the application as,

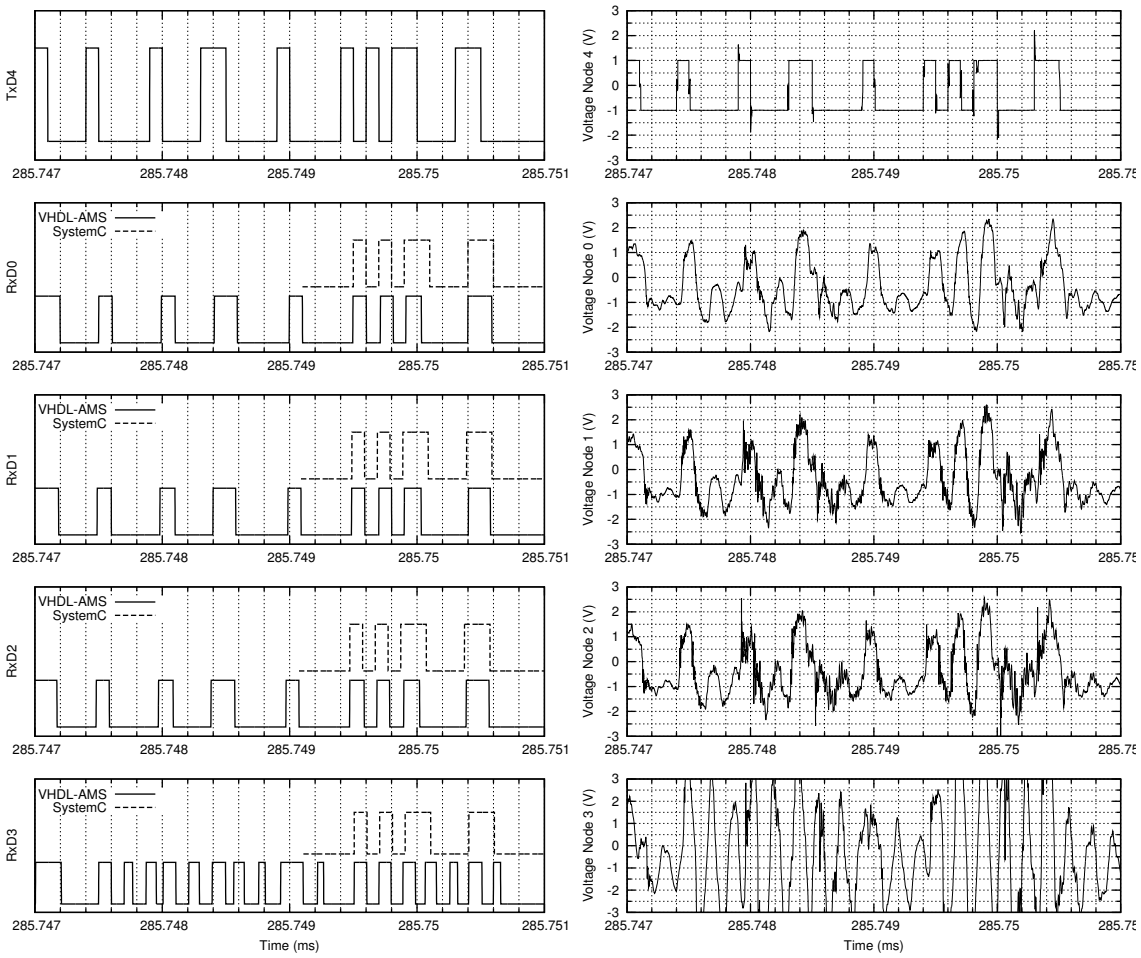


Figure 4.7: TxD/RxD Excerpt of a Switched FlexRay Frame

depending on the sender ECU, an ECU may or may not receive frames correctly. At the worst, the FlexRay network may lose its synchronization and be split apart in different clusters of ECUs that are not able to communicate with each other (cliques). The RCMS detected all of the higher-level errors as shown in Fig. 4.6 which were also detected during the full VHDL-AMS only simulation run. The proposed RCMS configuration was correct, achieving the system analysis goals while improving simulation performance in comparison to the non-switched approach.

4.2 Validation of RCMS Accuracy

An important topic when using the RCMS methodology is the accuracy of the results. In Fig. 4.8, some bits of a FlexRay frame are shown. The frame was transmitted over the line by using the low level VHDL-AMS model during the whole co-simulation. It is compared to the same bits of the same frame but now, only this frame was transmitted via the VHDL-AMS models. For the rest of the simulated time, the high-level SystemC model

was used for frame transmission. It can be seen that both results are nearly identical, only showing some minor differences. Since the same models were used, the good correlation shows that the simulation model initialisation during RCMS was performed correctly. The differences have no effect on the higher levels of the FlexRay system as they are negligibly small. However, the use of the more accurate model is justified since its behaviour affects the higher levels (e.g. deformed and shortened bits) and cannot be simulated by using SystemC.

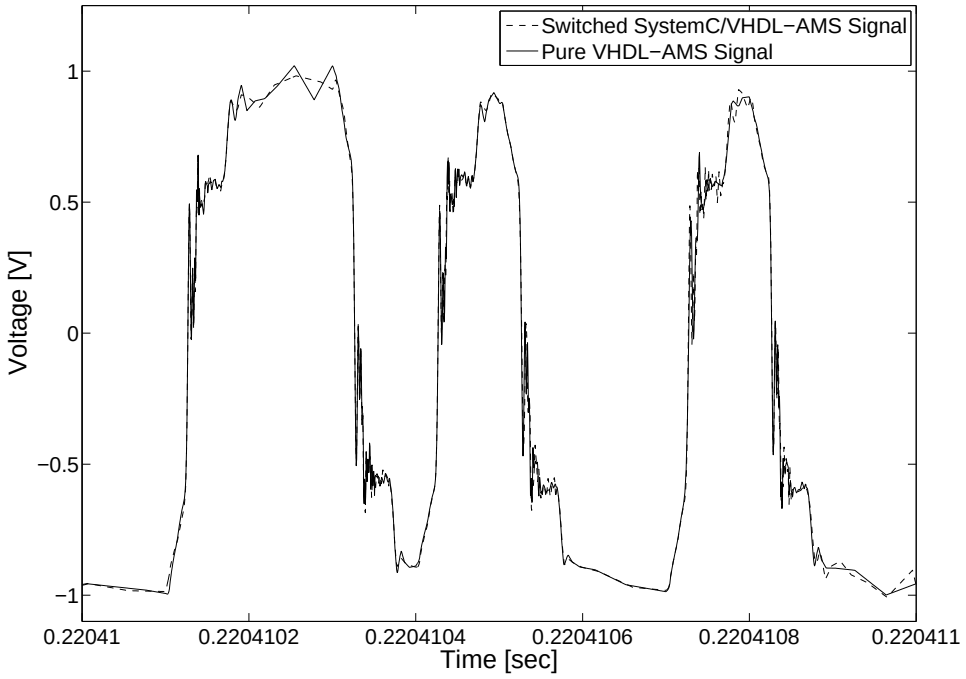


Figure 4.8: VHDL-AMS Simulation vs. Synchronised SystemC/VHDL-AMS RCMS

The following calculations were done by exporting the simulated waveforms from the VHDL-AMS simulator (Mentor AdvanceMS) as comma-separated-value files (n samples with a sample time of $0.5ns$) and importing them into MATLAB. The arithmetic mean $\overline{x}$ of a number of samples is calculated as

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

and the standard deviation s_x is

$$s_x = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2}$$

For the difference ($z_i = x_i - y_i$) between the pure (x) and the switched (y) waveform, the arithmetic mean and the standard deviation during the switched frame are

$$\overline{z} = -1.528 * 10^{-4}V, s_z = 3.08 * 10^{-2}V$$

The Pearson product-moment correlation coefficient (also sample correlation coefficient) between the pure VHDL-AMS waveform and the switched VHDL-AMS waveform is calculated as

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \sum_{i=1}^{n}(y_i - \overline{y})^2}}$$

The sample correlation coefficient is

$$r_{xy} = 0.9992$$

during the switched FlexRay frame (a factor of 1.0 means perfect correlation). It can be seen that the pure and the switched waveform do fit in a very good way.

4.3 Validation of FlexRay Simulation Model Accuracy

To determine the accuracy of the developed FlexRay simulation models compared to the reality, simulation results produced within the TEODACS FlexRayXpert.Sim framework were compared with measurements performed within the FlexRayXpert.Lab environment. Within several test campaigns based on a two-level space exploration, different FlexRay topologies were evaluated with respect to termination-related issues and reflections in both reality and simulation.

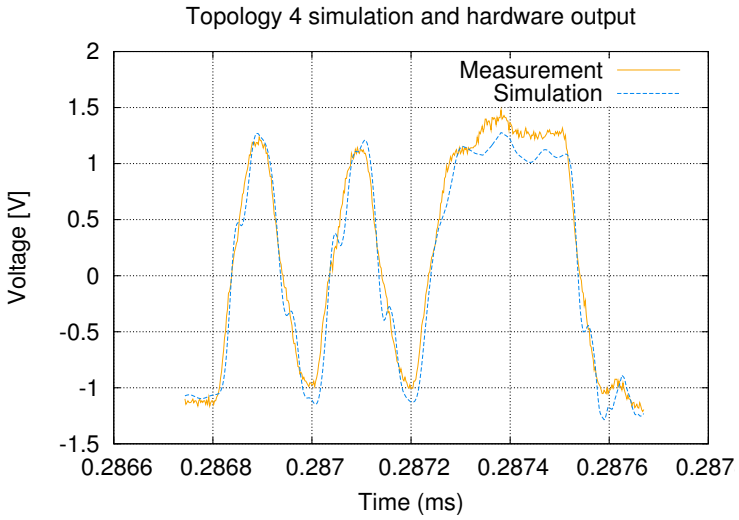


Figure 4.9: Comparison of Measurement and Simulation Waveforms

In Fig. 4.9, a sample result for the comparison of measurement and simulation is shown. Throughout the experiments, a good correlation between simulated and measured waveforms was achieved. The sample correlation coefficient was usually above 0.98, thus showing a good fitting between simulation and hardware measurement.

Section 6.7 presents the performed validation of the FlexRay simulation model accuracy in detail.

4.4 Summary

The RCMS-based methodology was applied to two automotive cross-domain examples: delay analysis within a distributed application and the analysis of FlexRay signal integrity effects on higher levels of the network. For specified system analysis goals, a suitable RCMS configuration fulfilling these goals was designed and executed, and the results were analysed. It was demonstrated that RCMS can be used for the efficient co-simulation of cross-domain automotive systems. By allowing the system developer to define a compromise between results accuracy and co-simulation performance, it is a useful methodology to bring together the different requirements of the various domains involved within the co-simulation. Additionally, the accuracy of the RCMS-based methodology in comparison to standard simulation was validated, showing nearly identical results, while at the same time, the simulation performance was strongly enhanced. Furthermore, the results gained during the simulation were compared with to according measurements within a real hardware setup, showing very good correlation between simulation and reality.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Driven by the strong demand for holistic simulation of automotive systems, this work focuses on the co-simulation of cross-domain automotive systems. For this, standard co-simulation is enhanced by the new methodology of RCMS. In RCMS, the simulation models used for a module can be changed during the execution of the co-simulation. The change of a simulation model can be triggered either by time (time-triggered RCMS) or by an online signal behaviour analysis according to defined criteria (adaptive RCMS). RCMS introduces dynamic into co-simulation, allowing to efficiently introducing different domains and simulation models with very different time constants into a single co-simulation. This enhanced co-simulation methodology provides the system developer with a new degree of freedom in simulating automotive systems. By using RCMS, the system developer gains an enormous amount of flexibility compared to standard static co-simulation, while at the same time improving co-simulation performance, cross-domain interaction and analysis possibilities.

Based on RCMS, this work presents a new methodology for the co-simulation of cross-domain automotive systems. The goal of this methodology is to attain simulation results according to specified system analysis goals in an efficient way. For this, system analysis goals are specified and simulation models are designed. During the next step, a suitable RCMS configuration for fulfilling the specified system analysis goals is created. For this, techniques from the area of safety analysis (FMEA, SSHA, SHA) were adapted to suit the purpose of RCMS configuration design. In this step, a trade-off between co-simulation performance and required simulation accuracy is developed. Additionally, detailed feedback for the design of simulation models is created if the specified system analysis goals cannot be fulfilled by using the existing simulation models. After the implementation of the simulation models, cross-domain co-simulation integration is performed. Within this step, the designed RCMS configuration is created within the RCMS-enabled co-simulation framework by using the simulation models. After the execution of the integrated RCMS configuration, the results can be analysed.

The benefits of this work are as follows. A flexible co-simulation methodology for the efficient co-simulation of cross-domain automotive systems is developed. It is based on

the idea of run-time co-simulation model switching. By exchanging the simulation models used during run-time of the co-simulation, it is possible to achieve high co-simulation performance, while at the same time having the necessary result details when required. For this, time-triggered and adaptive RCMS are specified in detail. RCMS allows for the efficient integration of multiple domains with very different time constants into a single co-simulation. By using the RCMS-based co-simulation methodology, based on specified system analysis goals, suitable simulation models and a suitable RCMS configuration allowing to fulfil the specified goals can be designed.

The developed methodology was evaluated within a comprehensive case study. In this case study, a cross-domain automotive co-simulation for the analysis of dependable automotive communication networks was created (TEODACS FlexRayXpert.Sim). By two different examples, it was demonstrated that by using the RCMS-based methodology for the co-simulation of cross-domain automotive systems, the required level of detail of the co-simulation results could be achieved, while at the same time the co-simulation performance compared to standard co-simulation was strongly enhanced. Additionally, the results gained by using RCMS were compared to results gained by using standard co-simulation. Both results were nearly identical, hence showing the effectiveness of RCMS. Furthermore, a comparison between simulation results and measurements within a real hardware setup demonstrated the high accuracy of the developed FlexRayXpert.Sim framework for the co-simulation of cross-domain automotive systems.

5.2 Future Work

Currently, the number of useful (and sometimes also useless) new functionalities, new driver assistance systems and more that are introduced into a new car generation are enormous. As these systems are often seen as unique selling points by the car manufacturers, this trend will continue in the future. A special issue with such high numbers of interacting systems is their compatibility. How can be guaranteed that the systems do not influence each other in an undesired way? Due to the nearly endless number of possible system combinations, simple testing cannot be the answer. This is a question that also cannot be answered by simulation only. While there are already efforts to handle this problem, still a lot of work will be required to receive a feasible answer for this problem. An additional challenge is the introduction of more and more safety critical systems into cars, for example due to the progressive trend to perform functionality by wire instead of the traditional way but also due to the integration of driver assistance systems as mentioned before. This requires to use more formal methods for the development of automotive systems, which, however, means an additional layer of complexity in the development process.

Middleware for distributed automotive software like AUTOSAR and dependable communication networks like FlexRay are key components for future cars. However, for example, the AUTOSAR development process is extremely complex to handle. While there are already lots of supporting AUTOSAR tools and simulation solutions, still huge effort will be required to make the development of distributed applications by using AUTOSAR more intuitive. And the same amounts for FlexRay with its enormous number of possible con-

figurations. Additionally, also fast, dependable communication networks like FlexRay will sooner or later reach their limits as the amount of data to be transferred grows by the number of ECUs and functionalities distributed throughout the car. As a result, already the application of dependable networks with more bandwidth, like time-triggered Ethernet with a data rate of up to 1Gbit/s, is investigated. However, with the rising data rate, again the complexity of the deployment and configuration grows. As a result, new techniques for handling this challenge will be required.

Within this work, run-time co-simulation model switching is especially used for the co-simulation of cross-domain automotive systems. However, system complexity and cross-domain interaction are rising not only in the automotive area, but also throughout a various number of other areas (e.g. development of embedded systems in general, System-on-Chips (SoCs)...). As a result, within these areas a strong demand for holistic co-simulation of the complete system exists as well. By applying the developed RCMS-based methodology, it could be possible to fulfil these demands. Due to an enhanced possibility of first-time-right by providing an efficient co-simulation of the complete system at the required level of detail, this could lead to cost reduction during development and faster time to market, which is crucial for systems of all kinds to be competitive.

A promising approach for future development of RCMS is the integration of hardware into the RCMS-based co-simulation. Within the TEODACS project, it was barely scratched on the surface of what is possible by the combination of RCMS and hardware. For example, the development of test strategies for various kinds of systems could be massively enhanced with the integration of hardware-simulation-interaction based on RCMS.

Chapter 6

Publications

This chapter contains publications by the author of this thesis which explain the approach presented in Chapter 3 and the case studies presented in Chapter 4 in greater detail. The following publications are included within this chapter:

Publication 1: *Run-Time Co-Simulation Model Switching for Efficient Analysis of Embedded Systems*, International Journal of Embedded Systems, submitted for publication

Publication 2: *Optimizing HW/SW Co-Simulation based on Run-Time Model Switching*, 2009 Forum on Specification and Design Languages, FDL '09, Sophia Antipolis, France, 22–24 September 2009

Publication 3: *Holistic Simulation of FlexRay Networks by Using Run-Time Model Switching*, 2010 Design, Automation & Test in Europe Conference & Exhibition, DATE '10, Dresden, Germany, 8–12 March 2010

Publication 4: *Verfahren zum Umschalten von heterogenen Simulationsmodellen zur Laufzeit*, Patent Application AT 1479/2009, EP 2 299 376, Austrian Patent Office, 2009, European Patent Office, 2010

Publication 5: *A Cross-Domain Co-Simulation Platform for the Efficient Analysis of Mechatronic Systems*, SAE World Congress 2010, SAE '10, Detroit, USA, 12–15 April 2010

Publication 6: *Heterogeneous Co-Simulation Platform for the Efficient Analysis of FlexRay-based Automotive Distributed Embedded Systems*, 8th IEEE International Workshop on Factory Communication Systems, WFCS '10, Nancy, France, 18–21 May 2010

Publication 7: *Exploration of the FlexRay Signal Integrity using a Combined Prototyping and Simulation Approach*, 13th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems, DDECS '10, Vienna, Austria, 14–16 April 2010

In Fig. 6.1, the mapping between the individual publications and the proposed RCMS-based methodology for the co-simulation of cross-domain automotive systems is shown. For specified system analysis goals, this RCMS-based methodology allows the design and implementation of suitable simulation models for obtaining the required accuracy of the results, while at the same time optimising co-simulation performance by using RCMS.

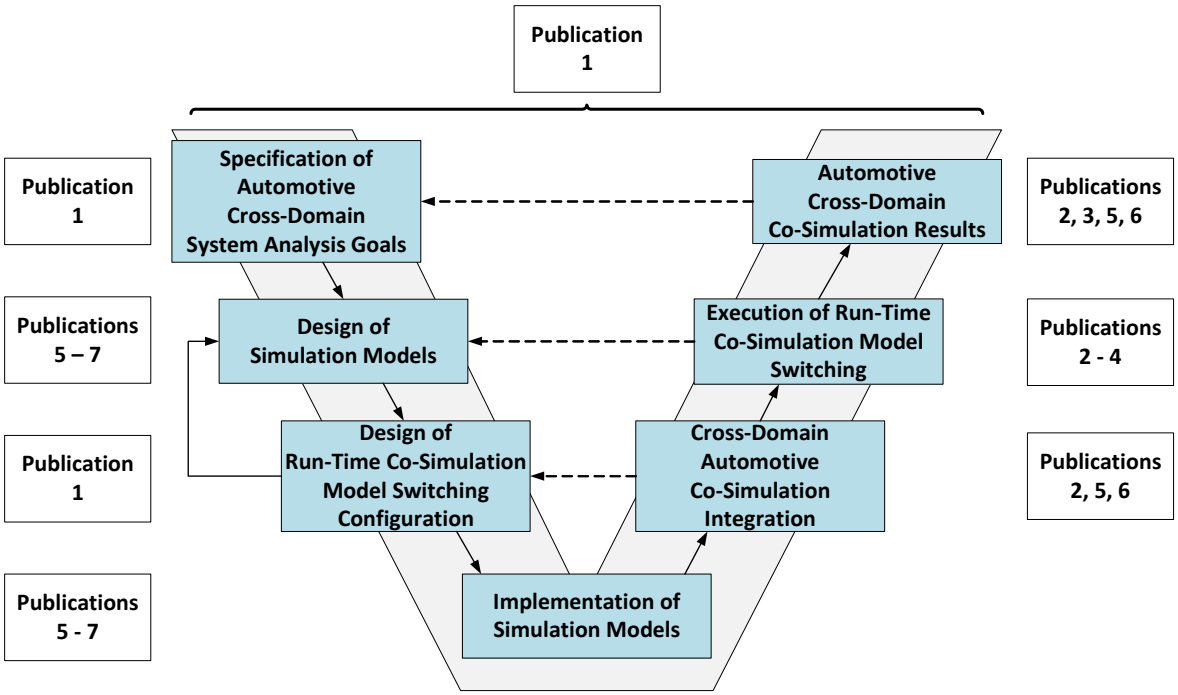


Figure 6.1: Co-Simulation of Cross-Domain Automotive Systems

The methodology as shown in Fig. 6.1 consists of several steps. A comprehensive overview is given in Publication 1. In the first step, the automotive cross-domain system analysis goals have to be specified. This is required to make sure that within the following steps, suitable simulation models and RCMS configuration are developed. Examples for this are included in Publication 1. Next, the initial design of simulation models based on the specified system analysis goals has to take place. This is shown in Publications 5 to 7. This step is closely linked to the design of a suitable RCMS configuration, which is described in Publication 1. Both steps are connected via a feedback loop, hence, simulation model deficiencies can be corrected. The next step is the implementation of the simulation models. For a cross-domain automotive system, this is shown in Publications 5 to 7. The integration of simulation models, RCMS configuration and simulation tools into a cross-domain automotive co-simulation is done during the following step. Details are provided in Publications 2, 5 and 6. The actual RCMS takes place within the next step, when the developed automotive cross-domain co-simulation is executed. More information about the RCMS fundamentals and the cross-domain co-simulation execution is included in Publications 2 to 4. Last but not least, the obtained automotive cross-domain co-simulation results according to the initially specified system analysis goals can be analysed. This is shown in Publications 2, 3, 5 and 6.

Run-Time Co-Simulation Model Switching for Efficient Analysis of Embedded Systems

Michael Karner, Christian Steger, Reinhold Weiss
Institute for Technical Informatics
Graz University of Technology
Graz, Austria
{michael.karner,steger,rweiss}@tugraz.at

Eric Armengaud
The Virtual Vehicle Competence Center
Graz, Austria
eric.armengaud@v2c2.at

Abstract

Simulation is widely used for the early validation and evaluation of embedded systems. In case of complex systems consisting of highly heterogeneous components, a trade-off occurs between the simulation accuracy and time interval under observation. In this work, we propose a new approach for run-time co-simulation model switching in order to enable the test engineer to define the simulation intervals of high accuracy and the intervals of high performance. We further propose a framework for the systematic decision of the component models to be used according to the goals and current status of the simulation. The proposed approach is illustrated by the simulation of automotive distributed embedded systems.

Keywords

Co-simulation; run-time switching; co-design; failure mode and effects analysis; FlexRay; automotive; adaptive switching; time-triggered systems.

1 Introduction

Simulation enables the execution of the system being developed in a simplified (fully controllable and observable) environment. It enables developers to integrate and validate their development before the entire system is available, thus finding and correcting potential errors earlier in the development process and consequently saving costs. This early execution further enables the evaluation of the system and supports early design decisions before the system exists, thus saving costly re-design (first time right).

In the case of embedded systems, the system consists of different, heterogeneous components such as analog and digital hardware (sensors, microcontrollers) and software (middleware, application). The simulation of such heterogeneous systems presents a strong challenge in order to combine all the components in a single environment. The challenges result from the different simulation languages and tools required to model the different components, as well as from the different simulation scopes of the components. Hence, low-level components (e.g. analog sensors) have high dynamics and require simulation resolution below the picoseconds. High level components such as application software, on the contrary, require a long execution time (several seconds, minutes, hours) for efficient validation. It is evident that simulating complex systems for several seconds with a simulation accuracy below the picoseconds is not practicable.

Co-simulation (Yoo and Jerraya (2005)) is an approach to combine different simulation tools within a single framework, thus enabling the concurrent simulation of heterogeneous components within the same environment. However, the overall simulation performance depends on the slowest simulator involved, slowing down the other simulators to guarantee synchronization. The challenge here is to reduce the *simulation time* (thus increasing computation speed of the simulation) while extending the *simulated time* (time interval under observation within the simulation). To that aim, methods for trading the accuracy against the computation speed for a given component within a given modelling language exist (e.g. Hines and Borriello (1997a), Claes and Holvoet (2009)).

The aim of this paper is to present a framework enabling the exchange of a component model during simulation execution. The motivation is to provide the test engineer the possibility to switch between different models of the same component (e.g. high-speed and low accuracy against low-speed and highly accurate) in order to reduce the simulation

time while keeping the required simulation accuracy for the time interval of interests. This paper presents two main contributions. First, the approach of run-time co-simulation model switching (RCMS) is presented. The proposed method enables switching of the simulation models during run-time without limitation regarding the simulation language. Second, a framework for systematic analysis of simulation models and generation of an adequate RCMS configuration is provided. With this methodology, the task of getting the most suitable co-simulation configuration for a given system analysis goal is solved by applying safety-analysis based strategies.

The document is structured as follows. In Section 2 an overview about the related work is provided. The generic approach for run-time co-simulation model switching is described in Section 3. Section 4 describes the framework for RCMS analysis and configuration. The application of this methodology within case studies is illustrated in Section 5. Finally, Section 6 concludes this document.

2 Related Work

For the design and simulation of embedded systems, co-simulation is a common methodology used throughout the community. Several co-simulation frameworks have been proposed to allow the coupling of different development languages and abstraction levels. Bouchhima et al. (2006) propose a co-simulation framework that enables the coupling of SystemC and Simulink models, thus connecting the worlds of continuous and discrete-event simulation. In another work (Birrer and Hartong (2005)) the authors deal with the integration of SystemC models into a Verilog-A simulation to bridge the gap between system level description and hardware implementation. An example for a commercially available co-simulation framework is CISC SyAD (CISC Semiconductor Design+Consulting GmbH (2011)). This framework features the coupling of several simulators from both discrete-event and continuous time simulation.

Also the idea of run-time simulation model switching has been topic of some research. Existing work mainly deals with the dynamic switching within models developed in the same modelling language and not with run-time co-simulation model switching as discussed in this paper. In Hines and Borriello (1997b) and Hines and Borriello (1997a), the authors present their approach for the usage of dynamic communication models by presenting a tool named Pia. It allows the designer to specify communication between components at multiple levels of detail. The designer has to use the Pia language proposed by the authors to describe the necessary interfaces and components for the various abstraction levels. Run-time simulation model switching is also proposed by Yoo and Jerraya (2005). They suggest to dynamically switch between several abstraction levels of a processor simulation. However, no results are available and the authors only suggest dynamic simulation model switching as a possible way to improve co-simulation performance. Other authors deal with run-time simulation model switching of different transaction level models (TLM). In Salimi Khaligh and Radetzki (2010) and Salimi Khaligh and Radetzki (2009), the authors present a SystemC based solution for switching between different TLM abstraction layers during simulation. The approach adapts the simulation accuracy automatically depending on the state of the model. A similar approach is presented in Beltrame et al. (2007). They extend SystemC TLM to support multi-accuracy models with the possibility to switch between different model accuracies at run-time. Additional works deal with accuracy adaptive model switching, see for example Claes and Holvoet (2009), Rao and Wilsey (2006). However, within these papers adaptive switching is based on simple criteria (e.g. model is producing no more results) and focused to a very special use case. Additionally, it is only switched within the same modelling language or tool.

The topic of run-time co-simulation model switching has been previously demonstrated in Karner et al. (2009), Karner et al. (2010a). However, there the core issue is the approach of time-triggered run-time co-simulation model switching. The more general adaptive run-time co-simulation model switching and the extension for the design of a suitable run-time co-simulation model switching configuration presented within this paper are not dealt with.

Within the area of co-simulation, the topic of getting a suitable co-simulation configuration for a given system analysis goal is often neglected. This is the case because if no run-time co-simulation model switching is used, the possible number of configurations reduces drastically. However, this comes with the price of reducing co-simulation efficiency. There are selected papers dealing with this problem. For example, in Claes and Holvoet (2009) the authors describe how they selected the parameters for the run-time simulation model switching. Their approach is very specific for the selected example and it cannot be applied in a generic way. Fummi et al. (2004) gives another example of deriving a possible configuration for co-simulation. There, the authors describe a two-phase partitioning process followed by several refinement steps to find a suitable setup for the simulation of a networked embedded system. For the individual components of the simulation model, it is decided if they should be modelled using SystemC, NS/2 or an instruction set simulator. However, this approach is relatively tailored to networked embedded systems.

It can be seen that there exist several stand-alone solution for specific problems. However, no holistic methodology is available covering all relevant aspects regarding the efficient application of run-time co-simulation model switching. Especially missing are (1) a generic approach for run-time co-simulation model switching and (2) a framework for the identification and design of run-time co-simulation model switching.

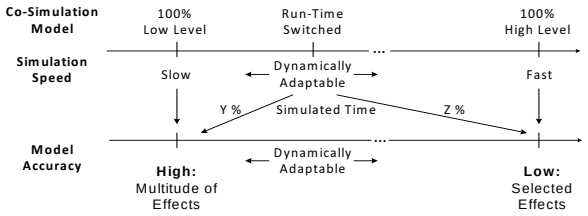
3 A Generic Approach for Run-Time Co-Simulation Model Switching

Co-simulation is a well known step during the development of embedded systems. The simulation of a system by using different hardware description languages (HDL) and multiple abstraction levels within a single co-simulation is of great help for system designers and developers. However, standard co-simulation is defined statically. The simulation models used are fixed throughout the co-simulation. This leads to the problem that, especially when performing a co-simulation with simulation models having very different typical simulated times and simulation performance, the overall co-simulation performance can get really poor or the results are missing required details. As possible answer to this problem, run-time co-simulation model switching (RCMS) can be used. In the following section a generic approach for RCMS is presented.

3.1 Run-Time Co-Simulation Model Switching Overview

The idea behind RCMS is to change at run-time the simulation models that are used for a specific part of the co-simulation. This allows to speed up the co-simulation while at the same time providing high accuracy when required. It provides the developer with a new degree of freedom for defining for each module the intervals for which a computational expensive high-detail simulation model is required and when a simpler and faster model suffices ("shifting of complexity", see Fig. 1). This method can be compared with an oscilloscope having the capability to zoom into significant parts of the simulation, while performing a fast (less accurate) run for the less interesting part of the simulation.

Figure 1. Shifting of Complexity



To have a common vocabulary, it is required to introduce some terms. The overall *system* is composed of different subsystems ("clustering"). *Subsystems* are composed of one or more modules and/or other subsystems. Subsystems are only a composition and do not directly implement e.g. an algorithm. *Modules* within subsystems do not have a "visible structure" to the outside any more, they are forming the lowest level of structuring (e.g. hardware or software components). Modules are implemented by simulation models. A *Simulation model* is implemented using one specific modelling language. There can be one or more different simulation models for one module. These different simulation models for one module can be implemented using different modelling languages, different abstraction levels and executed using different simulators. However, they are basically representing the same functionality and hence representing the same module. Run-time model switching takes place between the different simulation models of one module.

3.2 Concept for Generic Run-Time Co-Simulation Model Switching

To perform effective run-time co-simulation model switching the decision about the switching point and which model to switch to is essential. However, in existing works this decision is done solely based on the time (time-triggered RCMS). Within complex systems showing very dynamic behaviour time-triggered RCMS easily reaches its limits. In the following an extension into a more generic RCMS approach is described, comprising not only time-triggered but also criteria-based adaptive RCMS.

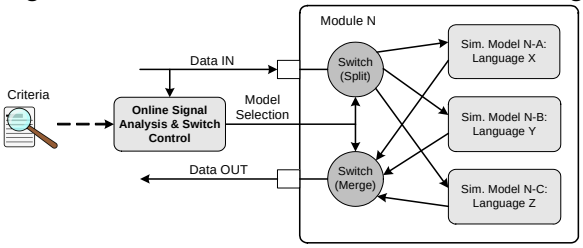
The possibilities of RCMS can be compared with an oscilloscope, having the capability to zoom into significant parts of the simulation, while performing a fast (less accurate) run for the less interesting part of the simulation. And like an oscilloscope has a trigger condition, a trigger condition is also required for a generic RCMS. The following criteria are able to act as a possible trigger source for the generic RCMS approach presented within this document.

- Time (time-triggered RCMS like until now)
- Events generated by an online signal analysis performed by the provided RCMS implementation (adaptive RCMS)
- Events generated by an external analysis logic (complex system specific criteria, adaptive RCMS)

The actual process of RCMS is performed by using switches. These switches are responsible for splitting and merging the data flow during the co-simulation. Fig. 2 demonstrates the switching principle for RCMS. The switches are controlled

by a switch control unit. The switch control unit is responsible for reacting to trigger sources accordingly and to perform the online signal analysis. Additionally, the switch control unit handles the required actions for switching like start of synchronisation or configuration of the switches.

Figure 2. Run-Time Co-Simulation Model Switching



Time-triggered RCMS uses a-priori defined switching points, with time acting as trigger source. Hence, it may not be applicable to all types of systems or simulation models. However, it has one big advantage: based on the principle of shifting of complexity (see Fig. 1), the developer has the possibility to simply adjust the trade-off between the simulation performance and the model accuracy achieved. This is of great advantage, as in fact the developer can specify the simulation performance in advance, hence adjusting the simulation time needed. By specifying the simulated time (including concurrent simulation time) for each simulation model of the module the developer can select the relation between simulation performance and accuracy according to the requested needs. Thus, the simulation performance improvement factor in comparison with standard static co-simulation methodologies is more or less freely selectable by the developer. This is of great advantage when simulating complex systems. As demonstrated in Karner et al. (2010a), it is possible to estimate the resulting overall co-simulation performance for time-triggered RCMS in advance. More details about time-triggered RCMS can be found in Karner et al. (2009).

While time-triggered RCMS is rather simple to handle, it has some important drawbacks. For example, it is a relatively inflexible approach as it has to be defined before simulation start. It is not able to react to spontaneous events and also requires a predictable system behaviour for efficient switching. It also introduces additional overhead as switching takes place according to the current time - and not if the current situation would really require it.

In **adaptive RCMS**, the current system behaviour acts as trigger source for switching. The idea behind it is to perform a run-time analysis of the behaviour of defined signals within the co-simulation. Depending on the analysis results, a switch to the most suitable simulation model for a module is initiated. The run-time signal analysis is performed by comparing the current signal behaviour to a set of criteria defined by the developer. If a criterion matches during the analysis, a switch to the according simulation model and the required synchronisation are initiated. This principle is shown in Fig. 2.

The actual process of run-time signal analysis is performed by using a sliding window. Hence, not only the current value but also values from the past can be included within the analysis process. It is possible to specify the size of the window (e.g. 10 entries) and also its update behaviour. It can be selected between event-triggered (window content changes only after change of input signal) or time-triggered window update (the input signal is sampled with a specific frequency).

Within the current implementation, the following signal analysis criteria are supported for adaptive RCMS.

- Arithmetic mean of the signal values within the sliding window (current signal value)
- Average frequency of signal value changes within the sliding window (only useful for event-triggered window update behaviour; current signal frequency value)
- Average value change between two consecutive signal values within the sliding window (signal dynamics)
- Standard deviation within the sliding window (signal dynamics)

For each criterion it is possible to specify several value intervals for matching. If the calculated criterion value is within the defined interval, a switch to the associated simulation model is initiated. To avoid a fluttering behaviour of the analysis, a hysteresis time can be specified. After a criterion match, all additional matches within the hysteresis time will be ignored. A typical criterion definition contains the following entries: the type of criterion (e.g. standard deviation), lower and upper limit for criterion match, window size, window update behaviour, simulation model to be used in case of criterion match, synchronisation type for upcoming model, concurrent simulation time and hysteresis time. An example for this is shown in section 5.

By using a sliding window for analysis, it is obvious that there is some reaction time until a criterion matches. By reducing the window size the reaction time can be reduced, but at the cost of losing information about past signal behaviour. Additionally, the reaction time is extended by the concurrent simulation time required for synchronisation of the

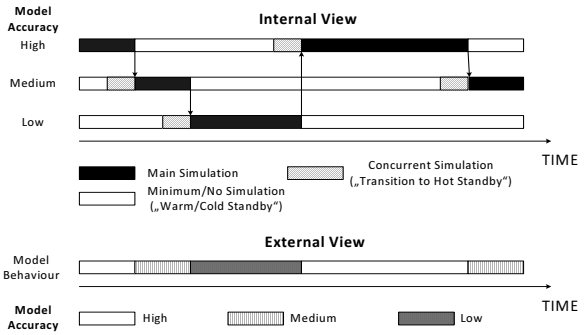
upcoming simulation model. For most systems this short reaction time does not present a problem. However, in case it is a problem there are several possibilities to deal with it. One would be to use a co-simulation environment supporting simulation roll-back. However, the options in this case would be very limited. Another possibility is to adjust the criteria definitions to match more early. But this causes additional overhead and reduces the efficiency of the RCMS. Another possible solution is to introduce artificial delay into the data flow within the co-simulation by including data buffers. In case of a criterion match the buffer contents of the current model are duplicated into the buffer of the upcoming model and further on, new values are inserted into both buffers. If the buffer size is chosen accordingly, the upcoming model can synchronise itself by using the buffer contents. When the value that triggered the criterion match is the one to be sent out of the buffer, a switch between the synchronised models can take place. By doing so, the reaction time can be reduced to zero but at the cost of inserting an additional permanent artificial delay. Hence, the idea of using buffers to reduce the criterion reaction time is not suitable for all systems or analysis goals.

Another way for adaptive switching is to use an external analysis logic as a trigger source. This may be required for triggers caused by very complex and system specific conditions which cannot be mapped to the criteria supported by the standard adaptive RCMS. While this external analysis logic is very system specific, on the other hand it can be exactly tailored to the requested needs, allowing for most efficient generation of the trigger signal.

3.3 RCMS Synchronisation

RCMS requires synchronisation between the different simulation models of a module in order to assure correct continuous service delivery of this module during switching. There are several ways of how the synchronisation of the upcoming model can be performed. Typically it is performed via concurrent simulation where the upcoming simulation model is fed with the current data but without actually using its calculated results (see Fig. 3). This allows for a synchronisation of the upcoming model so that switching can take place seamlessly. During this synchronisation process, no data is exchanged between the current and the upcoming model. Notice here that a trade-off exists between on one hand setting the concurrent simulation time large enough in order to allow for a proper model initialisation, and on the other hand minimising concurrent simulation in order to reduce simulation time.

Figure 3. Typical Synchronization Process during Run-Time Co-Simulation Model Switching



To make the process of synchronisation more clearly, a comparison to the area of redundancy can be done. The currently active model is the main model. If, due to a switching request, the concurrent simulation phase is about to start, the upcoming model is equal to cold standby (or in some cases warm standby). The synchronisation process ensures that the upcoming simulation model gets into hot standby by supplying it with the simulation data flow for a certain amount of time (concurrent simulation time). By doing so, seamless switching between current and upcoming model can be performed.

More in-depth information about the synchronisation process during RCMS can be found in Karner et al. (2009).

4 Run-Time Co-Simulation Model Switching Framework for Efficient Analysis of Embedded Systems

In the following section a guidance for the efficient configuration and use of run-time co-simulation model switching (RCMS) is presented.

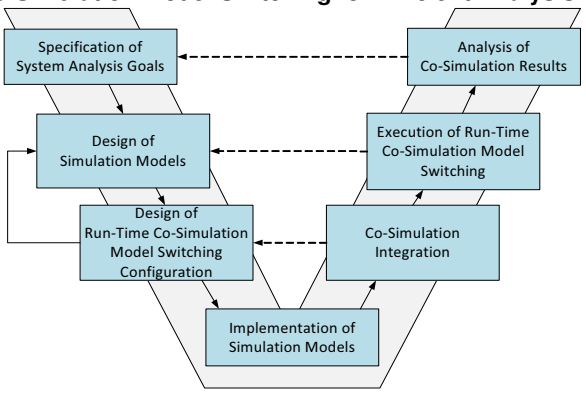
4.1 RCMS-based Analysis of Embedded Systems

If a system is to be analysed by using co-simulation there is a certain set of questions that have to be solved in order to produce suitable analysis results.

- What simulation models are required?
- What is the most suitable modelling language and abstraction level for each simulation model?
- What is the best co-simulation setup to fulfil the analysis goals?

The proposed run-time co-simulation model switching based methodology tries to answer these questions in a systematic way. An overview is shown in Fig. 4.

Figure 4. Run-Time Co-Simulation Model Switching for Efficient Analysis of Embedded Systems



First it is required to *specify the system analysis goals and requirements*. This step is essential to make sure that the following steps work into the right direction. Hence, the specification should be performed with special care. For this step there are no parameters on how to perform it. It can be done by producing a simple text, but also by using special description languages and tools. However, it is of great importance that at least the following information is specified exactly.

- What is the system to be analysed?
- What are the goals of the analysis?
- Are there special operations that have to be examined in detail?
- What are the requirements on the simulation process itself (e.g. high simulation performance because of the long time interval of interest)?

If these questions are answered in detail, the following steps will be much more easier to complete and the results of the methodology are greatly improved. Examples for such a system analysis goals specification are shown within the case studies in Section 5.

The next two steps in Fig. 4 can be seen as functionally linked as there is a feedback-loop connecting them. Based on the system analysis goals, the initial *design of simulation models* is performed. The simulation model developer specifies how the simulation models should look like, what their properties are and so on. For this, well-known standard tools and methodologies can be used. The second of these two linked steps is the *design of a run-time co-simulation model switching configuration* that fulfils the specified system analysis goals and requirements. In this step the basic idea is to apply adapted well-known safety-analysis techniques like failure mode and effects analysis (FMEA), subsystem hazard analysis (SSHA) and system hazard analysis (SHA) (see e.g. Hillenbrand et al. (2010), Goel and Graves (2007), Suganthi and Kumar (2010), Federal Aviation Administration (2010) and Ericson (2005) for details on these techniques) to the available simulation models and analysis requirements. By using these techniques, it is possible to get a valid RCMS configuration for the specified analysis goals while at the same time providing valuable feedback for the design of the simulation models. This allows for an iterative refinement process resulting in optimised simulation models and co-simulation configuration. The design of the RCMS configuration is explained in detail in section 4.2.

After finishing design of simulation models and RCMS configuration it is required to perform the *implementation of the simulation models* according to the created design. This is quite a straight-forward process as there exist lots of well-known guidelines for this. Hence, no special directions are given and it is up to the simulation model developer to use any suitable technique for implementation.

The following step deals with the *integration of the simulation models and the developed co-simulation configuration into a common co-simulation*. Several co-simulation environments are available in academia and industry. However, by

default none of them supports RCMS. We decided to use the commercial co-simulation environment SyAD by CISC Semiconductor (CISC Semiconductor Design+Consulting GmbH (2011)) and enhanced it to support RCMS. The co-simulation integration is a tool-specific task, hence, no general guidelines can be given. However, the co-simulation configuration developed during the previous step basically can be used within every co-simulation environment featuring run-time co-simulation model switching. As shown in Fig. 4, in case of detected problems it is possible to go back and modify the co-simulation configuration.

Execution of run-time co-simulation model switching follows as next step. Here, after the desired RCMS configuration has been set up, the simulation models are executed. The actual act of run-time model switching takes place within this step. In case that the actual behaviour of the simulation models and/or the overall co-simulation does not fulfil the expectations (because e.g. the developer made some mistakes during the previous steps, or the simulation performance has been overestimated) it is required to go back to the design of simulation models and perform the required adaptations.

Last but not least is the *analysis of the co-simulation results*. The results gained during the execution of the co-simulation are analysed. For this, again there is no special direction on how to do it. It mainly depends on the output and the result files produced by the simulation models. In case of results not fulfilling the specified analysis goals and requirements it is possible to go back to the first step of the methodology (see Fig. 4). Then, specify the requested system analysis goals and requirements more precisely and perform the required adaptations during the following steps.

After this general overview about the proposed methodology, within the following section the design of a suitable RCMS configuration is described in detail to give the reader a greater knowledge on how these steps work and how they can be applied.

4.2 Design of Run-Time Co-Simulation Model Switching Configuration

When using run-time co-simulation model switching (and co-simulation in general) there are several challenges to be solved. One of the most important is the definition of a suitable RCMS configuration for a given system analysis goal with respect to the available simulation models. Especially in complex systems structured into several subsystems, it is not an easy task to define an adequate configuration: There are different switching strategies possible (standard static co-simulation, time-triggered RCMS and adaptive RCMS). The strategies can be mixed and have to be parametrised correctly to produce the desired result quality. Additionally, it has to be assured that the simulation models are able to fulfil the specified system analysis goals and requirements.

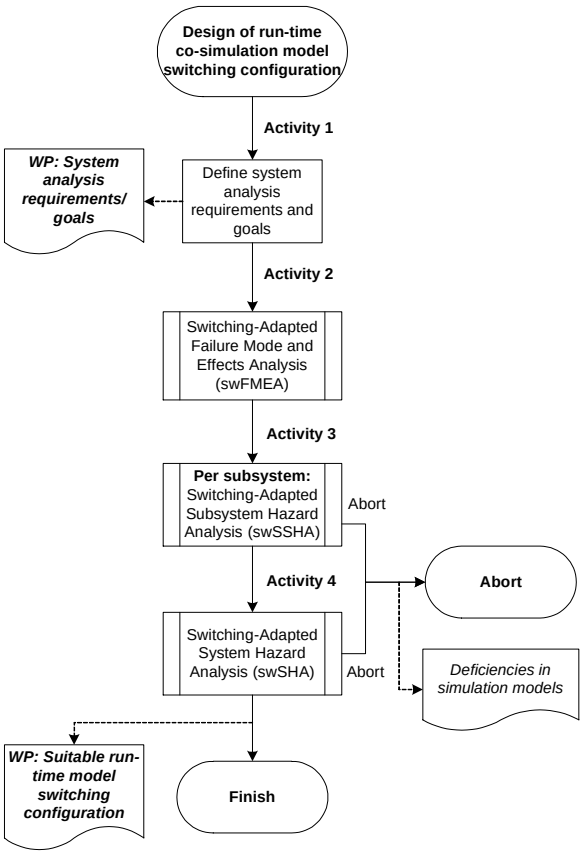
4.2.1 Overview

To overcome these problems, we propose a new methodology for the design of run-time co-simulation model switching configurations with respect to given system analysis goals and requirements. Additionally, this methodology provides feedback for the design of simulation models to enhance their capabilities if required (see Fig. 4). The main idea is to apply adapted safety-analysis techniques like failure mode and effects analysis (FMEA), subsystem hazard analysis (SSHA) and system hazard analysis (SHA) to the available simulation models and analysis requirements. Further on, they are called switching-adapted FMEA (swFMEA), switching-adapted SSHA (swSSHA) and switching-adapted SHA (swSHA). This methodology supports the developer in finding the most suitable RCMS configuration for the specified system analysis goals. If no suitable switching configuration can be found, information is given why the current analysis goals cannot be fulfilled. This information can be used to adapt the simulation models. The main advantage of this new methodology is that it provides a structured procedure to design the RCMS configuration based on well-known techniques. Additionally, the developer is able to reuse information gathered during product FMEA for the swFMEA (e.g. clustering into subsystems and modules, system information, possible failure modes, use cases,...). The terms like defined in the following vocabulary are used throughout the next sections. As the methodologies of FMEA, SSHA and SHA are adapted for the design of a RCMS configuration, also the according terms may have different meanings. A *failure mode* is a "risk" caused by a simulation model. A possible failure mode would be for example "missing of disruptive effects compared to reality". A *hazard* is defined as the potential for harm, hence, a switching configuration not suitable for the desired system analysis purpose. An example for such a hazard is a simulation performance way below of the required level or that important effects are not covered by the current simulation model(s) in use.

Fig. 5 shows the proposed methodology for the design of RCMS configurations with respect to given system analysis goals and requirements. It is structured into four activities and has defined work-products. The first step is to *define the system analysis requirements and goals*. If executed within the overall RCMS methodology as shown in Fig. 4, the according specification can be reused. After this step, activity 1 is completed and the work-product "system analysis requirements and goals" is produced.

Following step is to execute the *swFMEA* like detailed in section 4.2.2. In this step, all simulation models are analysed for the failure modes that they may introduce into the system if they are used (e.g. "faulty simulation results compared to reality in case of using incorrect topology") and how it can be coped with these failure modes. This step is rather

Figure 5. Design of a Suitable Run-Time Co-Simulation Model Switching Configuration



generic and if performed once, the data can be reused for other system analysis goals. Additionally, data gathered during a standard product FMEA can also be reused during the swFMEA, hence reducing workload for the developer. By the end of this step, activity 2 is completed.

Now for each subsystem, the *swSSHA* like described in section 4.2.3 has to be performed. The goal of the swSSHA is to create a RCMS configuration within the subsystem by identifying hazards caused by simulation models and the possible model switching configuration which would avoid the achievement of the system analysis goals. If there are unresolvable hazards, feedback for the design of simulation models is created. After successful completion of this step, activity 3 is completed.

The last important step is to execute the *swSHA* (see section 4.2.4 for details). It is applied to the overall system and identifies hazards that apply to more than a single subsystem and cannot be identified during the swSSHA. The main focus of the swSHA is, according to the possible hazards at system level, to adapt the run-time switching configuration derived during the swSSHA and generate the final RCMS configuration. By successful completion, the work-product ”suitable RCMS configuration” is created. If this step cannot be completed, by the list of unresolved hazards detailed feedback for the design of simulation models is provided.

For performing swFMEA, swSSHA and swSHA typical work sheets like used in the industry (see e.g. Arabian-Hoseynabadi et al. (2010), Ookalkar (2009)) have been adapted and extended. They are now tailored to the specific requirements of the adapted techniques.

4.2.2 Switching-Adapted Failure Mode and Effects Analysis

FMEA is a bottom up approach used to identify how systems fail and to identify the effects of failure. Based on a failure in one module, the effects of this failure are tracked on local and higher levels (Leveson (1995)). Due to the limitations of the standard FMEA approach (especially the interactions within the system are not sufficiently considered) it has been adapted for the special requirements of RCMS analysis to the switching-adapted FMEA (swFMEA) and is supported by swSSHA and swSHA. Several evaluation numbers have been removed (e.g. occurrence rating, detection

Table 1. Switching-Adapted Failure Mode and Effects Analysis Work Sheet Structure

1st step	2nd step	3rd step	4th step	5th step
ID, Module	Simulation Model	Simulation Performance		
		Use Case	Failure Mode	Cause of Failure
				Local Level Effects
				Subsystem Level Effects
				Compensating Action
				Value Range Limitation
			Failure Mode	...
	Simulation Model	...		
ID, Module	...			

Table 2. Switching-Adapted Subsystem Hazard Analysis Table Structure

1st step	2nd step	3rd step
ID, Module	Use Case	
	Requirements	
	Hazard for Subsystem	Cause for Hazard
		Effects caused by Hazard
	Simulation Model	Switching Parameters
		Possible Subsystem Hazards
	Recommended Switching Strategy	
	Open Hazards	
	Feedback to Simulation Model Design	

rating, risk priority number,...) while other criteria have been added, e.g. simulation performance, value range limitations and simulation model information. All simulation models are analysed for the failure modes that they may introduce into the system if they are used (e.g. "faulty simulation results compared to reality in case of using incorrect topology") and how it can be coped with these failures. The complete structure of the swFMEA work sheet is shown in Table 1. According to Table 1, for each module identified by an ID, there exist one or more simulation models. For each simulation model, there is an entry for the simulation performance. Additionally, each simulation model has one or more specific use cases. Each use case has one or more failure modes. Each failure mode has a specific cause of failure, local level effects etc.

The swFMEA is quite a straight-forward process. First, the swFMEA work sheet has to be filled by using already existing FMEA data and simulation model information, but without defining possible compensating actions. During the next steps, for each specified failure mode it has to be defined if there are possible compensating actions (e.g. "do not use for high transmission rates") and if there is a value range limitation by the compensating action (e.g. "max. signal frequency 5 MHz"). If there is no compensating action possible for a failure mode, this has also to be noted. All data generated during this step is stored into the swFMEA work sheet for further use. It has to be noted that this step is rather generic and independent of the system analysis goals. Hence, if performed with care, the results produced here can be reused during future applications of the methodology.

4.2.3 Switching-Adapted Sub-System Hazard Analysis

After the swFMEA, for each subsystem the switching-adapted subsystem hazard analysis (swSSHA) has to be done (see Table 2). Generally spoken, subsystem hazard analysis is a top-down approach that identifies hazards and their effects on subsystem level. Contrary to the swFMEA, this step makes direct references to the specified system analysis goals and requirements. If the system analysis goals change, most likely it will be required to perform the swSSHA (and the following swSHA) again because of the change in focus.

The swSSHA can be split in two main parts: the actual swSSHA and the creation of a recommended RCMS configuration for this module as subtask of the swSSHA. The swSSHA (Table 2) has to be done for each subsystem. For each module of the subsystem, the general use case is defined (e.g. "transmit data via a cable from one point to another"). Next, the general system analysis goals and requirements have to be tailored for the subsystem under evaluation. Based on these requirements, possible hazards for the subsystem are defined (e.g. simulation performance too low, required accuracy not achieved). Possible hazards also include dependencies between modules within the subsystem (e.g. if module A is implemented using HDL X, then module B also has to use HDL X). The cause for the hazard (e.g. SystemC model fast but lacking accuracy, VHDL-AMS model very slow but producing highly accurate results) and also the effects caused by

Table 3. Switching-Adapted System Hazard Analysis Table Structure

1st step	2nd step	3rd step
ID, Use Case	Requirements	
	Hazard for System	Cause for Hazard
		Effects caused by Hazard
		Recommended Adaptations to SSHA Switching Strategy
	Open Hazards	
	Feedback to Simulation Model Design	

the hazard (e.g. results either very detailed but with extremely low simulation performance or with too less detail but high simulation performance) are determined.

Based on the information gathered until now during the swSSHA, the definition of a suitable RCMS configuration for the module is started. Here, the data of the swSSHA and the swFMEA is used to create a preliminary RCMS configuration for the module. For every simulation model of the module, supported by the swFMEA data it is decided if adaptive RCMS can be used (and what the according parameters are). If adaptive RCMS is not required or cannot be used, it is defined if time-triggered RCMS should be used (and its according parameters), if no RCMS is required (there is no need to use it if a single simulation model is perfect for the specified system analysis goals!) or if the simulation model should even not be used at all. Based on this decision, possible hazards for the subsystem created by using this simulation model with the derived configuration are defined. After this has been done for all simulation models of the module, the actual RCMS configuration for this module is created. By the RCMS data for each simulation model, the module RCMS configuration is developed in a way to solve the hazards found during the previous step. The preliminary module RCMS configuration is stored into the run-time model switching work sheet holding the overall system RCMS configuration. In case that after creation of the most suitable module RCMS configuration there are still open hazards, the RCMS configuration methodology has to be aborted. However, it is no final stop. By the data gathered until now, especially the exact list of open hazards, it is possible to create feedback for the simulation model design process. With this information, the simulation models can be modified to deal with the open hazards. Afterwards the methodology can be restarted.

4.2.4 Switching-Adapted System Hazard Analysis

The last step during the design of the RCMS configuration is the switching-adapted system hazard analysis (swSHA). The standard SHA is basically performed like SSHA but on system level and not within subsystems. It analyses the interactions and hazards between different subsystems of the system. It identifies hazards that apply to more than a single subsystem and that are not identified during the SSHA (Ericson (2005)). Within the swSHA, the RCMS configuration derived during the individual swSSHAs is analysed for possible hazards at system level due to subsystem interaction, and, if required, adapted accordingly to overcome these hazards. Like in the swSSHA, if in the end there are still open hazards the methodology is aborted and extensive feedback for the simulation model design is created.

Like the swSSHA also the swSHA can be split into two main parts. The actual swSHA and the creation of adaptations to the RCMS configuration (see Table 3). After the general use case of the system is defined, possible hazards for the system due to subsystem interactions are identified (e.g. level of detail of results delivered by a subsystem to another subsystem is too low for the receiving subsystem to work correctly). Then, the cause of each hazard (e.g. level-of-detail mismatch between subsystem A1 and subsystem B3) and the effects the hazard creates (e.g. ESP algorithm not working correctly any more, hence it cannot be acceptably analysed) are determined.

After the hazards for the overall system are defined, the preliminary RCMS configuration derived during the previous steps has to be adapted to eliminate the hazards. If there are hazards for the overall system, for each of the hazards it is tried to eliminate it by adapting the preliminary RCMS configuration. If this is successful for all hazards, the final RCMS configuration for fulfilling the specified system analysis goals and requirements is created. Otherwise, like before, the methodology is aborted and according feedback based on the unresolved hazards for the system is given back to the design of simulation models. After modifying the simulation models, the design of the RCMS configuration can be started again.

5 Case Study & Experimental Results

To demonstrate the proposed run-time co-simulation model switching based approach a co-simulation of a FlexRay communications network was developed (see Karner et al. (2010b)). FlexRay (FlexRay Consortium (2005)) is a time-triggered automotive communications protocol operating at data rates up to 10 MBit/s. Its reliability is highly depending on the signal integrity. The accurate modelling of the causative effects leads to extensive physical level simulation times

even for very short simulated times (e.g. several days or weeks of simulation time for a simulated time of just a few milliseconds). The time window of interest at system level can cover several minutes (typical automotive control applications), however accurate physical models are required to analyse low level effects and interferences. Because of the wide range of requirements and the enormous complexity, co-simulation and especially run-time co-simulation model switching are required to handle this topic.

Two examples are demonstrating the RCMS based system analysis and development methodology. In the first example, a distributed application running within a car is to be analysed for delays because of the placement of functionality within different electronic control units (ECU). In the second example, the effects of FlexRay signal integrity are under observation.

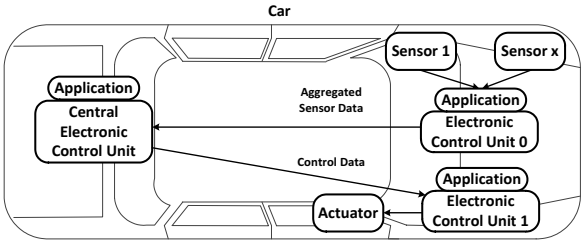
5.1 Example 1: Distributed Application Delay Analysis

Supported by standardised methodologies like AUTOSAR (AUTOSAR GbR (2008)), distributed applications are gaining more and more importance within cars. The functionality of the distributed application is placed on different ECUs, according to factors like computational power, data availability and even costs. However, due to this distribution the impact of the communications architecture is not negligible any more. For example, the resulting delay within the distributed application is largely influenced by the architecture and configuration of the underlying communications network.

Fig. 6 shows an example of a distributed control application which is to be analysed by the RCMS based methodology shown in Fig. 4. The application is using AUTOSAR concepts and FlexRay as communication system. It is embedded within a powertrain-focused simulation model of a complete car. For this system, the following system analysis goals and requirements are specified.

- Goal: analysis of the delay within an automotive distributed application (cause/effect, placement of functionality within the network, sender/receiver) during critical situations (e.g. rapid change of application input data).
- Goal: analysis of the effects of different FlexRay communication configurations on the data transmission within the distributed application.
- Requirement: best possible simulation performance should be achieved as it will be required to perform several simulations (time interval under observation: up to several minutes of simulated time).
- Requirement: data transmission delay & quantisation within the distributed application have to be modelled correctly during critical situations to allow for a realistic analysis of the application behaviour.

Figure 6. Case Study: Automotive Data Transmission for Analysis of Delays Within a Distributed Application



After gathering information about the possible simulation models (e.g. several SystemC and VHDL-AMS based models for the components of a FlexRay network, SystemC based simulation of AUTOSAR concepts) and defining possible simulation tools (e.g. for the realistic simulation of the car) the next step is to design the RCMS configuration according to the system analysis goals.

The design of the RCMS configuration is executed as shown in Fig. 5. As the system analysis goals have been already specified the swFMEA is executed. For a module like the FlexRay cable the resulting swFMEA for one simulation model looks as follows (according to Table 1):

- *ID, Module*: 3, FlexRay cable
- *Simulation model*: SystemC
- *Simulation performance*: very high

- *Use case*: transmission of a digitised representation of an analog signal within a correct topology
- *Failure mode*: faulty simulation results against reality in case of using incorrect topology
- *Cause of failure*: only length-based delay and attenuation taken into account
- *Low level effects*: signal transmitted performed correctly even if it should be faulty
- *Subsystem level effects*: data transfer correct within a flawed FlexRay system
- *Compensating actions*: do not use within a network having a faulty topology
- *Value range limitations*: none

Next, for each subsystem the swSSHA is performed to determine possible hazards within the subsystem and to create a preliminary RCMS configuration. Within this example, the result for the module FlexRay communication controller is shown according to Table 2:

- *Use case*: execution of the logical FlexRay protocol for data transmission
- *Requirements*: high simulation performance, correct protocol execution, support of different communication configurations
- *Hazards for subsystem*: simulation performance too low
- *Cause for hazard*: internal high frequency clock of the communication controller in the SystemC model
- *Effects caused by hazard*: only medium simulation performance
- *Simulation model*: SystemC
- *Switching parameters*: SystemC model for critical data (as defined by the application), parameters: concurrent simulation, 30.0 ms concurrent simulation time, hysteresis 200.0 ms;
- *Possible subsystem hazards*: missing of any delay information for non-critical data can make the distributed application behave unexpected; internal clock of communication controller always running even if no data is transmitted
- *Recommended switching strategy*: use SystemC only if critical data (as defined by the application) is to be transmitted, otherwise switch to a simple fixed-delay model for data transmission (not FlexRay based)
- *Open Hazards*: none, if simple fixed-delay model is implemented and internal clock of the communication controller can be halted
- *Feedback to simulation model design*: support temporarily halt of the high frequency clock of the controller (SystemC); create a simple fixed-delay transmission model for data transmission of non-critical data

It can be seen that also feedback for the design of simulation models is created in this example. A simple fixed-delay model for transmission would be required to fulfil the system analysis specification, but this model does not yet exist. Hence, it has to be implemented so that the methodology can continue. This has been done during a second iteration, so the methodology can continue for this example.

After there are no more open hazards the next step is the execution of the swSHA to generate the final RCMS configuration. For example, an entry within the swSHA database looks as follows (according to Table 3):

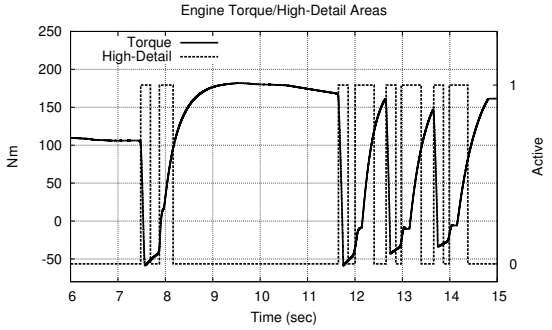
- *Use case*: data transfer between a distributed application within a car using AUTOSAR concepts and FlexRay
- *Requirements*: high simulation performance, accurate delay modelling
- *Hazards for system*: critical data transmitted via simple transmission model instead of FlexRay model
- *Cause for hazard*: no classification of critical data done yet
- *Effects caused by hazards*: analysis of the behaviour of the distributed application delivers wrong results
- *Recommended adaptation to switching strategy*: Implement analysis for critical application data (engine torque). If engine torque changes rapidly (= critical data, $|\Delta torque| > 0.3$ Nm per millisecond) switch to FlexRay model. During intervals of low data variability (= non-critical data) switch to the simple fixed-delay transmission model.

- *Open hazards*: none
- *Feedback to simulation model design*: none

In the end, the following RCMS configuration has been designed: The application is simulated using a SystemC based model implementing AUTOSAR concepts. The car is simulated using a suitable car simulator integrated into the co-simulation environment. For the transmission of data between the distributed application, adaptive RCMS is used. If critical data is to be transmitted (rapid change in engine torque, $|\Delta torque| > 0.3$ Nm per millisecond), a complete SystemC FlexRay model is used (controller, transceivers, cables). For non-critical data (low variability in engine torque), a simple fixed-delay transmission model is enough. The window size for the run-time signal analysis is of size 15 and updated by a sample clock every millisecond. This RCMS configuration allows for precise simulation of delay and configuration effects during critical areas of the simulation while at the same time having a relatively good simulation performance. The definition of critical data is done in a way to allow for a correct synchronisation of the upcoming model in time (slightly lower thresholds as would be required by the application).

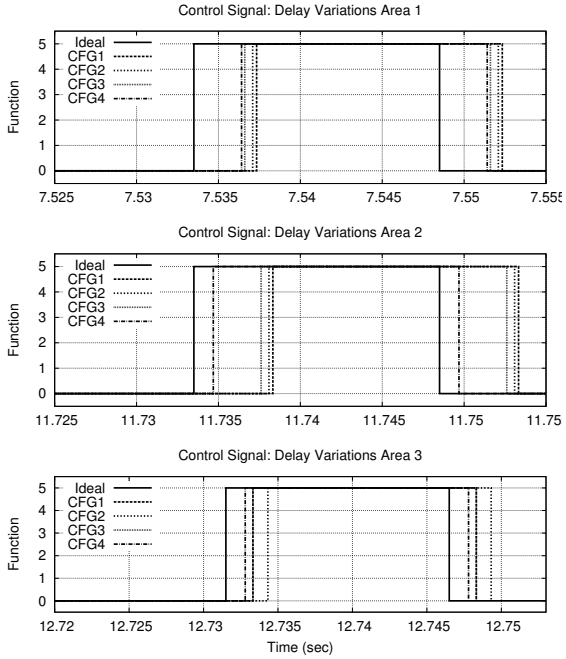
After integration into a common co-simulation (we used the co-simulation tool CISC SyAD) and execution of the RCMS the results can be analysed. Compared to the standard static approach, the RCMS simulation performance is about 4 times faster. This factor however depends on the amount of critical data transmitted as adaptive RCMS is used. In Fig. 7, the detected areas of critical data are shown during the first 15 seconds of simulation. The "high-detail" signal set to 1 indicates that the FlexRay simulation models are used instead of the simple delay-based simulation model. It can be clearly seen that no critical data is detected during areas of low engine torque variability.

Figure 7. Adaptive Switching Criterion (Engine Torque) Mapped to Detected Critical Areas



Based on the designed RCMS configuration, it is now possible to analyse the delay behaviour of the distributed application, fulfilling the specified system analysis goals. As the functionality of the application is placed on different ECUs, it is interesting what the resulting delay is compared to a non-distributed application placed on a single ECU. The overall delay is mainly affected by the number of ECUs the application is distributed over and the FlexRay communication configuration used. As FlexRay is a time-triggered communication system, its configuration specifies the transmission slots that can be used by each FlexRay controller. By varying the number of slots used per FlexRay controller and the position of the slots within a FlexRay cycle the update behaviour of the data transmitted can change drastically. This, of course, also massively impacts the resulting overall delay of the distributed application. It is not a good solution to just enlarge the number of slots used by a certain controller (and, in fact the ECU connected to the controller) as the total number of available slots is limited. Hence, the possible slots for other controllers connected to the network would be reduced, resulting in delay penalties for the applications running on the according ECUs. In fact, a good compromise for the FlexRay configuration has to be found. Within the developed RCMS setup this analysis can be easily done during a reasonable amount of time. As an example, Fig. 8 shows the resulting delay for a control signal sent within the distributed application. The sensor data is collected by ECU0, aggregated and sent via the FlexRay network to another part of the distributed application running on the central ECU. There, the incoming data is analysed, and for certain conditions a control signal is generated and sent to ECU1 via FlexRay. The ideal control signal shown in Fig. 8 would be created by a non-distributed application running completely on ECU0. It is compared with the control signal generated by the distributed application where is data is transmitted using FlexRay with different communication configurations. This is done for three different areas during the simulation. All of these areas are within the specified critical regions (rapid change in engine torque) as the control signal is only triggered during these areas. It can be seen that, generally spoken, with FlexRay configuration 4 the application shows the best delay behaviour. It can also be seen that the absolute value of the delay between the ideal signal and the different FlexRay configurations is not constant. While in area 2 the delay of FlexRay configuration 4 is way shorter than the delay of the other configurations, this is not that extreme in areas 1

Figure 8. Control Signal Delay for Different FlexRay Configurations



and 3. This is because of the different update behaviour and slot allocations within the FlexRay cycle of the four configurations. Hence, in certain situations a specific FlexRay configuration may lead to low delay, while in other situations (data available only some moments later) the same configuration may cause a much larger delay within the distributed application.

5.2 Example 2: Analysis of FlexRay Signal Integrity

The second example deals with application behaviour for a borderline FlexRay topology. The reliability of FlexRay is highly depending on the signal integrity during data transmission. Hence, it would be interesting to have a closer look at the influence of signal integrity effects on the behaviour of a distributed application. For example, the investigation of effects occurring at physical layer on higher layers like data link layer (e.g. shortening/lengthening of bits, misinterpretations because of low signal integrity) and application (e.g. missing data because of frame corruption, problems due to start-up/synchronization errors etc.). As stated during the first experiment, several simulation models for the different FlexRay components are available, featuring different levels of accuracy and implemented using different modelling languages like SystemC and VHDL-AMS. Again, by applying the RCMS analysis and design methodology a suitable RCMS configuration should be created to allow for an analysis with respect to the specified system analysis goals and requirements.

For this example, the following system analysis goals and requirements are specified.

- Goal: analysis of the behaviour of a distributed application for a borderline FlexRay topology
- Goal: analysis of the effects of signal integrity problems (e.g. reflections) on the application and the network behaviour
- Requirement: simulation performance has to be acceptable (no extremely low simulation performance) because the time interval of interest can be up to several seconds

Again, the methodology shown in Fig. 4 is used. For this example, adaptive RCMS would lead to nearly no improvement of the co-simulation performance. However, as FlexRay is a time-triggered communication system the more simple time-triggered RCMS can be used. There exist pre-defined points where the state of the system is completely known. Hence, it is the most advantageous solution to select the switching points there with the according concurrent simulation time large enough to allow proper initialisation of the simulation model to be switched to.

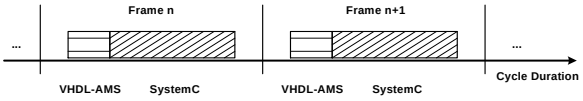
It was derived that most of the time the fast but less-detailed SystemC model for topology and transceivers should be used. However, in every frame the header (6μs) should be transmitted via the more-detailed but slow VHDL-AMS

Table 4. Example Simulation Time Comparison for one FlexRay Cycle with 12 Frames

	VHDL-AMS	Switched	SystemC
Seconds	88012	31757	307
Factor	287	104	1

simulation models for cable and transceivers (concurrent simulation time = 2μs). This is shown in Fig. 9. The signal

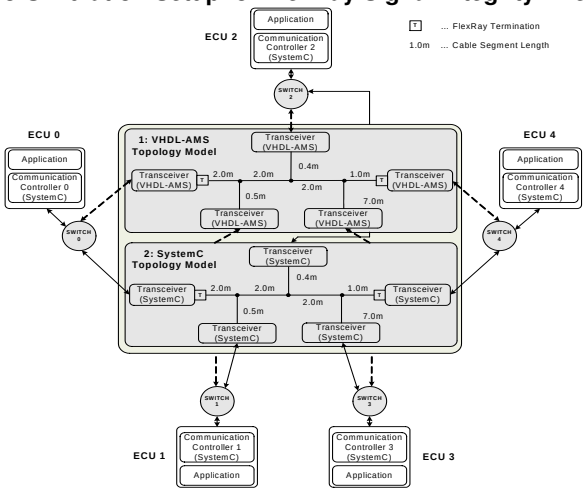
Figure 9. Time-Triggered Switching within a FlexRay Frame



integrity is extremely unlikely to change within the duration of a FlexRay frame (26μs). For the analysis of signal integrity effects, mainly the header of a FlexRay frame is relevant as all protocol-relevant is contained within the header. All other parts of the network (FlexRay controller, AUTOSAR based distributed application,...) are simulated using SystemC without switching. Hence, the configuration is able to fulfil the specified system analysis goals.

The final experimental setup is shown in Fig. 10. The network is consisting of 5 ECUs connected via a FlexRay

Figure 10. Co-Simulation Setup for FlexRay Signal Integrity Effects Analysis



network. A distributed application is running on these ECUs using AUTOSAR concepts. Within this network, ECU3 is using a long cable which is, against the FlexRay specification, not equipped with the required termination. This may lead to disruptive reflection which influence the signal integrity.

To get the simulation performance improvement by using RCMS instead of standard co-simulation additionally experiments with using the high-detailed VHDL-AMS models throughout the complete simulation were performed. Another simulation run included only SystemC models, without switching to the VHDL-AMS models for the FlexRay header. Table 4 shows a comparison of the results. While SystemC-only simulation is clearly the fastest, it features no details required for the analysis of the signal integrity effects on the distributed application. Hence, it is fast, but does not fulfil the specified system analysis goals. On the other hand, the VHDL-AMS only configuration has all the required details but at the cost of an extremely low simulation performance. Compared to the VHDL-AMS only configuration, the RCMS configuration simulating only the header by using VHDL-AMS is about three times faster. The effects on the behaviour of the FlexRay network and the distributed application were the same. Hence, the designed RCMS configuration presents an acceptable solution for the specified system analysis goals and requirements.

The executed RCMS demonstrated that, depending on the transmitting ECU, different ECUs have problems in receiving correct frames. For example, the data ECU 2 is sending can be received without any problems by ECUs 0, 1 and 4, while ECU 3 reports a header error check failure. However, if ECU 1 is transmitting the ECUs 0, 2 and 3 receive the data correctly, while ECU 4 reports an error in the frame header. These effects lead to reception asymmetry that can move the system into inconsistent states and thus should be avoided.

In Fig. 12 it can be seen that for a frame transmitted by ECU 4 (TxD4), ECU 3 using the VHDL-AMS model is not

Figure 11. Communication Controller Error Detection Output Excerpt

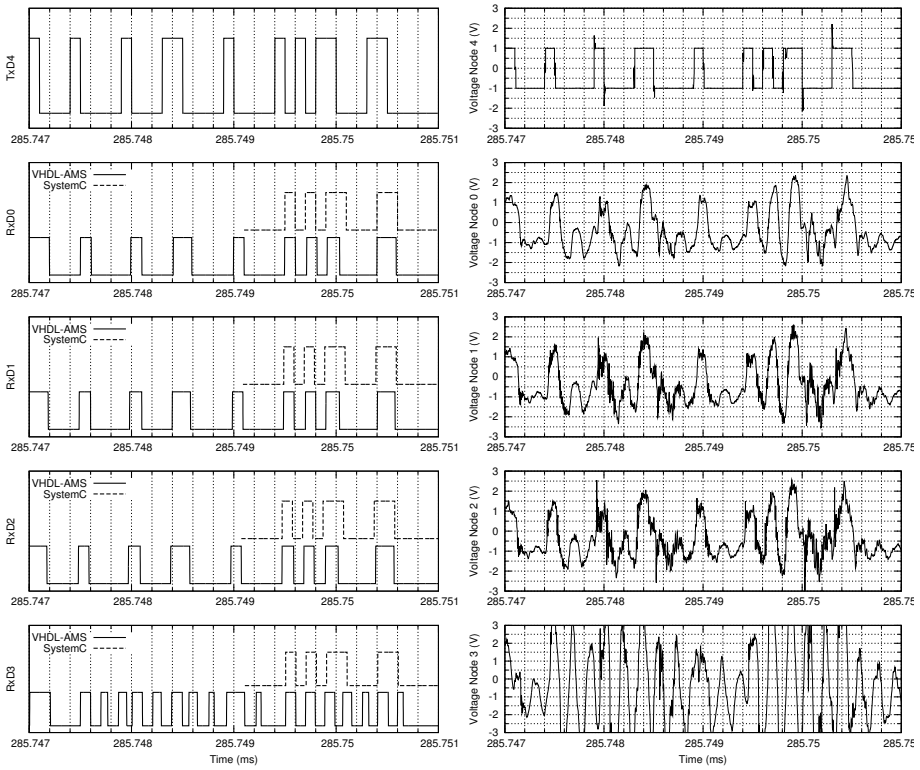
```
Node 3 - codec: BSS_ERROR!
Time: 285747500 ns

Node 2 - WARNING CODEC CHANNEL A: HEADER CRC FAILED!
Time: 285751650 ns

Node 1 - WARNING CODEC CHANNEL A: HEADER CRC FAILED!
Time: 285751662500 ps

Node 0 - WARNING CODEC CHANNEL A: HEADER CRC FAILED!
Time: 285751675 ns
```

Figure 12. TxD/RxD Excerpt of a Switched FlexRay Frame



able to receive any meaningful data on RxD3 because of reflections etc. Hence, the communication controller very early reports an error that it is not able to find a byte start sequence (BSS), see Fig.11. The problems can be easily detected by looking at the resulting differential voltage waveform at ECU 3. It is far away from any meaningful state. In Fig. 12, the SystemC topology model starts concurrent simulation at around 285.749ms before switching at 285.751ms. During concurrent simulation the output of the SystemC model is not used. It can be seen that the SystemC model does not show any distortion (because it only includes length-based delay and attenuation). Hence, no error would be detected by using the SystemC model all the time. The situation is a bit different for ECUs 0, 1 and 2. In fact, using the VHDL-AMS model they are able to receive the header data (transmitted by ECU 4) correctly most of the time. But at around 285.750ms, one bit sequence gets altered by a reflection. The high bit sequence is shortened and the following low bit sequence is stretched, leading to a header checksum error detected by the controllers, see Fig.11. By looking at the SystemC data it is obvious that this error does not happen there. These errors definitely influence the application as, depending on the sender ECU, an ECU may or may not receive frames correctly. At the worst, the FlexRay network may lose its synchronization and get split apart in different clusters of ECUs that are not able to communicate with each other (cliques). The RCMS detected all of the higher level errors like shown in Fig.11 that were also detected during the full VHDL-AMS only simulation run. The proposed RCMS configuration was correct, achieving the system analysis goals while improving simulation performance in comparison to the non-switched approach.

6 Conclusion

The proposed run-time co-simulation model switching (RCMS) approach enhances co-simulation in providing the possibility for the test engineer to define simulation intervals of high speed and other intervals of high accuracy within the same co-simulation run. This approach is relevant to reduce computing resources while keeping simulation accuracy for the intervals into consideration. The configuration of the RCMS (when to switch from a model to another for a given component) is a complex task due to the number and complexity of components involved. For that, we have proposed a framework based on switching-adapted FMEA and hazard and risk analysis in order to systematically identify the purpose of the simulation as well as the risks of using an inappropriate simulation model. This guidance leads to the definition of a suitable configuration for run-time co-simulation model switching for minimizing computation resources while using adequate simulation models to fulfil the specified analysis goals. Two uses cases with focus on automotive distributed applications have been discussed to illustrate the benefits of the approach. In comparison to the non-switched approach, the experimental results are equal while at the same time the simulation performance could be improved.

Acknowledgement

The authors wish to thank the "COMET K2 Forschungsförderungs-Programm" of the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT), the Austrian Federal Ministry of Economics and Labour (BMWA), Österreichische Forschungsförderungsgesellschaft mbH (FFG), Das Land Steiermark and Steirische Wirtschaftsförderung (SFG) for their financial support.

Additionally we would like to thank the supporting companies and project partners austriamicrosystems, AVL List and CISC Semiconductor as well as Graz University of Technology and the University of Applied Sciences FH Joanneum.

References

S. Yoo and A. Jerraya, "Hardware/software cosimulation from interface perspective," *Computers and Digital Techniques, IEE Proceedings* -, vol. 152, no. 3, pp. 369–379, 2005.

K. Hines and G. Borriello, "Dynamic communication models in embedded system Co-Simulation," in *Design Automation Conference, 1997. Proceedings of the 34th*, 1997, pp. 395–400.

R. Claes and T. Holvoet, "Multi-model traffic microsimulations," in *Winter Simulation Conference (WSC), Proceedings of the 2009*, 2009, pp. 1113 –1123.

F. Bouchhima, M. Briere, G. Nicolescu, M. Abid, and E. Aboulhamid, "A SystemC/Simulink Co-Simulation framework for Continuous/Discrete-Events simulation," in *Behavioral Modeling and Simulation Workshop, Proceedings of the 2006 IEEE International*, 2006, pp. 1–6.

P. Birrer and W. Hartong, "Incorporating SystemC in Analog/Mixed-Signal Design Flow," in *Forum on Specification and Design Languages, Proceedings of the 8th International*, 2005, pp. 173–178.

CISC Semiconductor Design+Consulting GmbH, *SyAD Online Documentation*, Klagenfurt, Austria, February 2011. [Online]. Available: http://www.cisc.at/syad

K. Hines and G. Borriello, "Selective focus as a means of improving geographically distributed embedded system co-simulation," in *Rapid System Prototyping, 1997. ' Shortening the Path from Specification to Prototype'. Proceedings., 8th IEEE International Workshop on*, 1997, pp. 58–62.

R. Salimi Khaligh and M. Radetzki, "Modeling constructs and kernel for parallel simulation of accuracy adaptive tlms," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, 2010, pp. 1183 –1188.

——, "Adaptive interconnect models for transaction-level simulation," in *Languages for Embedded Systems and their Applications*, ser. Lecture Notes in Electrical Engineering, M. Radetzki, Ed. Springer Netherlands, 2009, vol. 36, pp. 149–165.

G. Beltrame, D. Sciuto, and C. Silvano, "Multi-Accuracy power and performance Transaction-Level modeling," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, no. 10, pp. 1830–1842, 2007.

D. Rao and P. Wilsey, "Applying parallel, dynamic-resolution simulations to accelerate vlsi power estimation," in *Simulation Conference, 2006. WSC 06. Proceedings of the Winter*, 2006, pp. 694 –702.

M. Karner, C. Steger, R. Weiss, and E. Armengaud, "Optimizing hw/sw co-simulation based on run-time model switching," in *Specification Design Languages, 2009. FDL 2009. Forum on*, 2009, pp. 1 –6.

M. Karner, E. Armengaud, C. Steger, and R. Weiss, "Holistic simulation of flexray networks by using run-time model switching," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, 2010, pp. 544 –549.

F. Fummi, M. Poncino, S. Martini, F. Ricciato, G. Perbellini, and M. Turolla, "Heterogeneous co-simulation of networked embedded systems," in *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, vol. 3, 2004, pp. 168 – 173 Vol.3.

M. Hillenbrand, M. Heinz, N. Adler, J. Matheis, and K. Muller-Glaser, "Failure mode and effect analysis based on electric and electronic architectures of vehicles to support the safety lifecycle iso/dis 26262," in *Rapid System Prototyping (RSP), 2010 21st IEEE International Symposium on*, 2010, pp. 1 –7.

A. Goel and R. Graves, "Using failure mode effect analysis to increase electronic systems reliability," in *Electronics Technology, 30th International Spring Seminar on*, May 2007, pp. 128 –133.

S. Suganthi and D. Kumar, "Fmea without fear and tear," in *Management of Innovation and Technology (ICMIT), 2010 IEEE International Conference on*, 2010, pp. 1118 –1123.

Federal Aviation Administration, *System Safety Handbook*. Washington D.C., United States: Federal Aviation Administration, 2010.

C. A. Ericson, *Hazard analysis techniques for system safety*. Wiley, Hoboken, NJ, 2005.

H. Arabian-Hoseynabadi, H. Oraee, and P. Tavner, "Failure modes and effects analysis (fmea) for wind turbines," *International Journal of Electrical Power and Energy Systems*, vol. 32, no. 7, pp. 817 – 824, 2010.

A. Ookalkar, "Quality improvement in haemodialysis process using fmea," *International Journal of Quality and Reliability Management*, vol. 26, pp. 817–830(14), 2009.

N. G. Leveson, *Safeware: system safety and computers*. New York, NY, USA: ACM, 1995.

M. Karner, M. Krammer, S. Krug, E. Armengaud, C. Steger, and R. Weiss, "Heterogeneous co-simulation platform for the efficient analysis of flexray-based automotive distributed embedded systems," in *Factory Communication Systems (WFCS), 2010 8th IEEE International Workshop on*, May 2010, pp. 231 –240.

FlexRay Consortium, *FlexRay Communications Systems – Protocol Specification Version 2.1*, 2005. [Online]. Available: http://www.flexray.com

AUTOSAR GbR, *AUTOSAR: Specification of the Virtual Functional Bus*, AUTOSAR GbR, February 2008. [Online]. Available: http://www.autosar.org

Optimizing HW/SW Co-Simulation based on Run-Time Model Switching

Michael Karner, Christian Steger, Reinhold Weiss
Institute for Technical Informatics
Graz University of Technology, Austria
{*michael.karner,steger,rweiss*}*@tugraz.at*

Eric Armengaud
The Virtual Vehicle Competence Center (ViF)
Graz, Austria
eric.armengaud@v2c2.at

***Abstract*—The development of embedded systems nowadays is strongly supported by simulation in order to reduce development time and improve product quality. However, effects occurring e.g. on the physical level may impact the whole system and cannot be captured by using only high abstracted models. Co-simulation is a possible solution for this problem. It enables the combination of an abstracted, system level view with highly accurate models of different components, and thus supports the analysis and validation of the embedded system. In this work we present an approach based on the inter-language run-time switching of simulation models during co-simulation. This new method enables long-time system level simulation with dynamic (user defined) switches to more accurate models for the punctual analysis of low level effects. The approach is applied for the co-simulation of an automotive distributed network and enables the analysis of the system at application level with a dynamically selectable accuracy down to the signal integrity within the physical cables.**

I. Introduction

Modelling at system level is required for the validation and analysis of the entire embedded system and especially for the analysis of the interactions between the different components. However, providing a view that can be simulated for typical system operations in a reasonable amount of time usually requires models that abstract away lots of details. These abstracted behaviours hide low level effects that might propagate and influence the system, thus making the modelled system not realistic enough for validation.

To solve this problem hardware/software co-simulation can be used, allowing the developer to simulate different parts of a system by using different languages and/or different abstraction levels in one common co-simulation. This allows considering e.g. physical level effects also in system level models. However, the overall simulation performance depends on the slowest simulator involved, slowing down the other simulators to guarantee synchronization. This is a problem when mixing accurate models (e.g. physical level) with system level simulation in a common environment. At system level, the focus is set to short *simulation time* (computation speed of the simulation) in order to obtain longer *simulated time* (time interval under observation). At physical level the simulation times are typically much higher because of the simulation of complex analogue components. Hence, relatively short simulated times are explored here ($\mu s, ms$). The problem is quite obvious: A simulated time of several seconds at system level leads to huge simulation times at physical level. On the other way, during the simulated times manageable for physical level simulation not much may happen at system level, rendering system level simulation futile.

In the following we present an approach that combines the advantages of both worlds: (1) the possibility to have the long simulated time with high simulation speed of system level models and (2) the high accuracy of low physical/analogue level models. The approach is based on the modelling of a single component by using different hardware description languages (HDL) and abstraction levels. Here, dynamic switching between the abstraction levels during co-simulation is proposed, thus reducing simulation time. This enables the system to be simulated using a fast high level model under normal circumstances and to switch to high-detailed models for accurate simulation of the component for the time intervals of particular interest (e.g. simulation of EMI/ESD).

II. Related Work

Several co-simulation frameworks have been proposed to allow the coupling of different development languages and abstraction levels. Bouchhima et al. propose in [1] a co-simulation framework that enables the coupling of SystemC and Simulink models, thus connecting the worlds of continuous and discrete-event simulation. In [2] Birrer et al. deal with the integration of SystemC models into a Verilog-A/AMS simulation to bridge the gap between system level description and hardware implementation. An example for a commercially available co-simulation framework is CISC SyAD [3]. This framework features the coupling of several simulators from both discrete-event and continuous time simulation.

The idea of run-time simulation model switching has been topic of some research. Existing works mainly deal with the dynamic switching within models developed in the same (discrete-event) hardware description language and not with cross-language (continuous/discrete-event) simulation model switching as proposed in this paper. In [4] and [5], Hines and Borriello present their approach for the usage of dynamic communication models by presenting a tool named Pia. It allows the designer to specify communication between components at multiple levels of detail. The designer has to use the Pia language proposed by the authors to describe the necessary interfaces and components for the various

abstraction levels. During experiments, a speed-up in the area of 10-100 is achieved. Run-time simulation model switching is also proposed by Yoo and Jerraya [6]. They suggest to dynamically switch between several abstraction levels of a processor simulation. However, no results are available and the authors only suggest dynamic simulation model switching as a possible way to improve co-simulation performance.

Other authors deal with dynamic simulation model switching of different transaction level models (TLM). Radetzki et al. [7] present a SystemC based solution for switching between different TLM abstraction layers during simulation. The approach adapts the simulation accuracy automatically depending on the state of the model. A similar approach is presented by Beltrame et al. [8]. They extend SystemC TLM to support multi-accuracy models with the possibility to switch between different model accuracies at run-time.

III. Run-Time Model Switching in HW/SW Co-Simulation

The existing works typically perform within a specific HDL and a discrete-event based simulation. Hence, achievable enhancements are limited to the capabilities of one specific HDL. In this paper we propose an approach to allow switching between different simulation models for the same component in a dynamic and inter-language (different HDLs, continuous and discrete-event based simulators) way. This greatly enhances the simulation speed by using computational expensive simulation models only in a (by model and time) clearly defined area.

A. Methodology

The approach is described as follows: On the time axis, the duration of time T is split into (virtual) non-uniform time stamps t_i with $t_i \in T | i = 1..m$ and $t_0 = 0$. At every time stamp t_i it is possible to switch between which model abstraction level should be used. A multiple-abstraction-level model M consists of several different models built at different abstraction levels (realizing basically the same functionality). If L denotes the space of implemented abstraction levels for a model M a single abstraction level is written as l_x with $l_x \in L | x = 1..n$. So the model for a specific abstraction level is expressed as $M(l_x)$. Following this, the model space M of implemented abstraction levels for a specific model is summarized as $M = M(l_1) \cup M(l_2) \cup M(l_3) \cup ... \cup M(l_n)$.

In order to synchronize the models and enhance the accuracy before switching both models involved should be running in parallel for a certain amount of time. Only the output of the primary model is used during this phase. The principle is shown in Figure 1. The internal view at the top of the figure is depicting how the parallel simulation is working. If one abstraction level model currently is not required ("minimum/no simulation"), depending on the type of model it can be either completely disabled or it is set to a level of minimal operation required to allow continuation of the simulation in the future (e.g. for some types of continuous-time simulation). This will be explained later on in this section in more detail. The external view at the bottom displays what the other parts of the simulation see as output of the model. To describe the operation of parallel simulation before switching the factor p_i is introduced. It describes the fraction of an interval of time τ after which the parallel computation starts. For this, τ is defined as $\tau_i = t_i - t_{i-1}$, the difference between two virtual time stamps. Hence, the interval of time where both models simulate in parallel is calculated as τ_{p_i} where $\tau_{p_i} = \tau_i - \tau_i \cdot p_i = \tau_i \cdot (1 - p_i)$. The simulated time for one abstraction level for a specific amount of time including parallel processing is

$$\tau_{R_i} = \tau_i + \tau_{p_{i-1}} = \tau_i + \tau_{i-1} \cdot (1 - p_{i-1})$$

and

$$\tau_{R_A} = \sum_{i=1}^{m} \tau_{R_i}$$

is the overall simulated time for one model abstraction level. The total amount of simulated time T_S for all abstraction levels of a model including parallel processing is written as

$$T_S = \sum_{A=1}^{n} \tau_{R_A}$$

The total simulation time T_R is $T_R = \frac{1}{C} \cdot \mathit{CALC}(T_S)$ where C is the achievable speedup in the system during parallel computation possibilities and $\mathit{CALC}(T_S)$ determines the simulation time needed for the simulated time T_S.

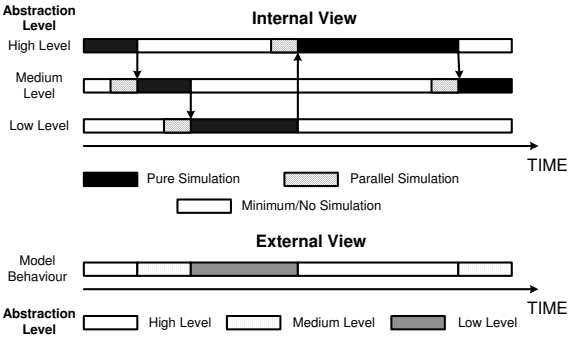


Figure 1. Internal and external view of the behaviour of a switched model built by several models at different abstraction levels

Depending on the model M additional properties have to be considered. For example in models realizing a communication protocol the model is typically consisting of several internal states. The states S of M are written as $S = S_1, S_2, S_3, ..., S_k$. Before switching to another abstraction level l_x occurs it may be required that $M(l_x)$ is in a condition of states S_D with $S_D \subseteq S$ to work properly. For example, S_D can be the condition of states a controller model requires to show a specific behaviour. These required states S_D can be the same for all abstraction

levels, $S_D = S_D(L)$, or may be specific for a certain level, $S_D = S_D(l_x)$. Depending on $M(l_x)$, one way for synchronization of $S_D(l_x)$ is to set the phase of parallel computation to a value that the following model is able to get into the correct internal condition of states S_D until the end of parallel computation. Choosing the duration of parallel computation is not an easy task. However, a quick estimation can be done by performing simple switching experiments to approximate to a suitable duration. Another way for synchronization is to provide the possibility to access the states S_D of $M(l_x)$ from the outside to initialize them with the correct values. Here, the correct state initialization values have to be provided by the developer. Another way would be to always provide the models M with the required information that they are able to get into S_D at the right time but do not have to do the main simulation effort. So e.g. $M(l_x)$ does get the necessary state information S_D all the time, but computational intensive data is only provided during τ_R. For systems with periodic behaviour the state synchronization is rather simple: In these models, the history is by default limited in time. There exist synchronization points where the system is in a defined/well-known state and switching of the simulation models can take place without problems by selecting an appropriate parallel processing duration. Of course there exist systems where the run-time switching approach may not be easily applicable. For example, a very complex VHDL-AMS system containing state information and history in both digital and analogue domain. In this case in-depth knowledge is required to define (1) the time durations where which abstraction level model should be used and (2) the parameters to properly initialize the model before switching. For the following analysis we presume that either (a) the model does not have dependent internal states or (b) is able to be initialized during parallel processing, or (c) it allows to externally set the internal model states.

All factors mentioned before influence several system properties. We focus on the simulation speed ω and on the external accuracy α of the encapsulated model. The external accuracy α at a given simulated time depends on the abstraction level simulation model $M(l_x)$ used as output. The external accuracy also depends on the factor p_i before the model is switched on the output. Hence, the external accuracy is defined as

$$\alpha = \alpha(M(l_x), p_i)$$

There is an interconnection between simulation speed ω and α, so $\omega = \omega(\alpha)$. If two models at different abstraction levels run in parallel this has a noticeable effect on the simulation speed as more computational power is required. Hence, the overall simulation speed ω is written as

$$\omega = \omega(T_R) = \omega\left(\frac{1}{C} \cdot \mathit{CALC}\left(\sum_{A=1}^{n} \tau_{R_A}\right)\right)$$

The simulation speed during the simulation of a single abstraction level l_x is

$$\omega_p = \omega_p(\alpha(M(l_x)))$$

During parallel processing of models at different levels of abstraction l_x and l_y the simulation speed is

$$\omega_s = \omega_s(\alpha(M(l_x), M(l_y)), p_i)$$

and hence $\omega = \omega(\omega_s, \omega_p)$. For further evaluation of the effects the factor p is stated as $p = \sum_{i=1}^{m} p_i$. As p_i describes the fraction of an interval of time after which the parallel computation of different abstraction levels starts it is apparent that if p is minimized the following relations are valid: $p = min \Rightarrow [\alpha = max, \omega_s = min] \Rightarrow p \propto \alpha^{-1}, p \propto \omega_s$. And typically an abstraction level with a high accuracy leads to low simulation speeds. These effects are shown in Figure 2. The simulation speed depends on both abstraction level and parallel simulation done before switching. During parallel processing the lower of the two abstraction levels dominates the simulation speed. The simulation speed drops even a bit below the speed that could be achieved by simulating only the lower of the two abstraction levels. This drop is depending on the system speedup factor C (multiprocessor usage, distributed simulation environment,...).

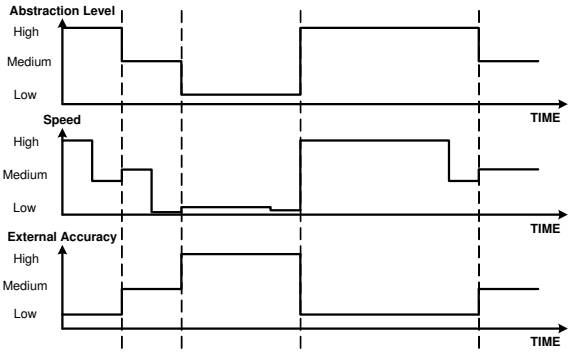


Figure 2. Relation between abstraction level, simulation speed and achieved accuracy in a run-time switched co-simulation

B. Implementation

The proposed run-time inter-language simulation model switching approach relies on existing co-simulation techniques. Hence it is required to use a co-simulation framework that supports the implementation of simulation models at different abstraction levels and by using different HDLs. We decided to use the commercially available co-simulation framework CISC SyAD [3]. It supports the co-simulation of simulation models at different abstraction levels and of simulation models implemented by using different HDLs like SystemC, VHDL-AMS, etc. The framework fulfils all the requirements to implement the proposed simulation model switching approach.

The main idea of our approach is to have implementations of a given model at different abstraction levels (see

Figure 3), ideally by using different suitable HDLs. As the switched multiple abstraction level model should appear to the outside as one consistent model we have to take care of assigning the simulation input to the correct abstraction level implementation. And only the output of the currently active abstraction level simulation model should be used as output of the switched model (see Figure 1, external view).

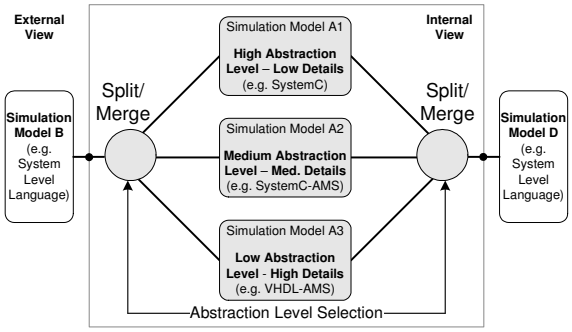


Figure 3. Run-time simulation model switching during co-simulation

For this, SystemC blocks for data splitting and merging have been implemented. The splitters/mergers decide through which abstraction level model the main simulation data flow is directed. They feature multiple ports to connect the switched models and one or more ports as interface to the non-switched simulation part. The principle is shown in Figure 3. These SystemC 'switches' can be parameterized to define for which simulated time which abstraction level model should be used. Additionally, the parallel computation time can be specified. It is possible to connect a simulation model to the splitter that acts as 'idle data' generator. Its output is provided to the different abstraction level models that are currently not involved in the main co-simulation and keeps them in the correct internal states so that no initialization problems may occur. However, this idle data generation is only required for certain kinds of models (e.g. communication protocol models).

The splitters/mergers are available in SyAD as ready-made blocks and can be drag-and-dropped into the co-simulation workspace. They can be connected in a graphical user interface to the other co-simulation models (e.g. SystemC models, Simulink models,...) and to the different abstraction level models the run-time switched model should consist of. Between some types of ports it is necessary to perform a digital to analogue (DAC) or analogue to digital conversion (ADC). If one of the abstraction levels of the switched simulation model has VHDL-AMS terminal as interface port there needs to be a AD/DA convertor between the splitter/merger and the VHDL-AMS model. The splitters/mergers can control the ADC/DAC so that the conversion only operates at full speed if the abstraction level model connected to the convertors is really active. This reduces the simulation time overhead that arises due to the integration of the AD/DA convertors.

IV. Experimental Results

To demonstrate the proposed co-simulation based run-time model switching approach a co-simulation of a FlexRay communications network was developed. FlexRay [9] is a time-triggered automotive communications protocol operating at data rates up to 10 MBit/s. Its reliability is highly depending on the signal integrity. The accurate modelling of the causative effects leads to extensive physical level simulation times even for very short simulated times (e.g. several days or weeks of simulation time for a simulated time of just a few milliseconds). The time window of interest at system level can cover several minutes (typical automotive control applications), however accurate physical models are required to analyze low level effects and interferences (e.g. EMI/ESD).

A. Use Case: FlexRay Network

The developed FlexRay co-simulation [10] consists of several simulation models implemented at different abstraction levels and by using different HDL. The car simulator CarMaker is used to generate the data transported via the FlexRay network. The FlexRay communication controller (SystemC) is responsible for the logical FlexRay protocol. The FlexRay transceiver (VHDL-AMS) converts between bits and electrical signals. The signals are received/transmitted via FlexRay cable and topology models implemented in VHDL-AMS. The co-simulation is realized by using CISC SyAD as co-simulation framework.

For our experiment we used a simplified version of this network: instead of the communication controllers we used FlexRay tester nodes developed in SystemC. A tester node is able to produce a FlexRay bit stream out of a frame description and can store and analyze a bit stream it receives via the network. The model of the physical level (transceivers, topology plus cables) is the model to be switched within: $M = M(l_1) \cup M(l_2)$. Hence we implemented two different abstraction levels l_1 and l_2 for it. The first one is a high level SystemC model $M(l_1)$ which basically just adds length-based delay and attenuation to the signals transferred between the FlexRay tester nodes. The second one is a high detail low level VHDL-AMS model $M(l_2)$ of both FlexRay transceiver and topology including cables. This experimental setup is shown in Figure 4, including the length of the different cables (from 0.2m to 3.5m in this example). The low level analogue model enables the analysis of effects such as EMI/ESD, reflections, wrong cable termination etc.

We transmitted a short data sequence consisting of two FlexRay communication cycles via the network. A cycle consists of 8 frames of FlexRay data. Each communication cycle is $5ms$ of simulated time and a FlexRay frame is about $40\mu s$ long. We performed three different tests. The first was the co-simulation without the proposed run-time switching approach. Here the low level physical VHDL-AMS model $M(l_2)$ was used all the time to transfer the data between

the SystemC tester nodes. The second test included run-time switching. Most of the time the high abstraction level SystemC model $M(l_1)$ was used to transfer the data between the nodes. For the fifth frame of the first cycle we transmitted the data stream via the realistic VHDL-AMS model $M(l_2)$ of FlexRay transceiver, topology and cables. In the third test the simple SystemC based topology model $M(l_1)$ was used throughout the simulation.

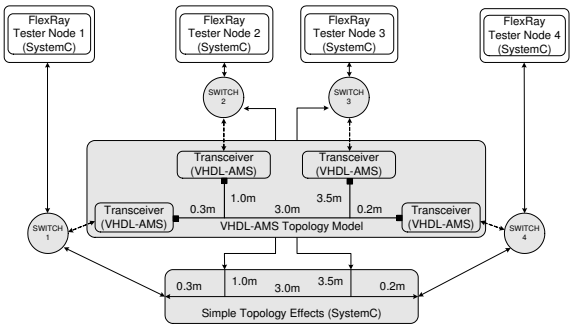


Figure 4. FlexRay switched co-simulation example

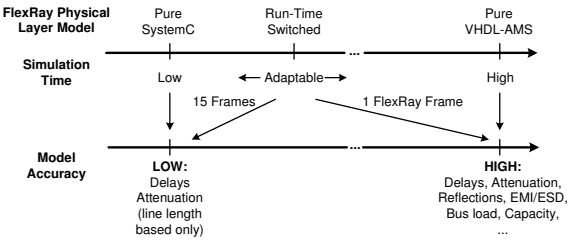


Figure 5. Shifting of simulation time and model accuracy by using run-time switching

B. Results

In Table I the simulation results are summarized. As expected the pure SystemC simulation is the fastest but at the price of losing nearly all details and effects generated by the transmission. On the other hand, the pure VHDL-AMS model is several magnitudes slower than the SystemC model, but providing all the required details for the system analysis. It can be observed that the run-time switched co-simulation approach is over a magnitude faster than the pure VHDL-AMS model. For the selected time interval, both pure and switched VHDL-AMS models feature the same high accuracy, thus allowing to explore the influence of the signal integrity on the reliability of the FlexRay system.

Table I
SIMULATION TIME COMPARISON FOR PURE/SWITCHED FLEXRAY TOPOLOGY AND TRANSCEIVER CO-SIMULATION

	Pure VHDL-AMS	**Switched**	**Pure SystemC**
Seconds	51453	3835	0.152
Factor	338506	25230	1

One of the main advantages of the run-time switching approach is the designer's possibility to easily adjust the trade-off between simulation time and model accuracy. Figure 5 demonstrates this for the test setup used. The designer is able to specify how long the simulation time T_R should be by defining the simulated time τ_{R_A} for which each abstraction level model $M(l_x)$ should be used. Hence, the simulation time improvement factor compared to standard static co-simulation can be influenced by the designer according to the requested needs. For this example, we used a rather small improvement factor: out of a total number of 16 FlexRay frames, 1 frame was transmitted via the high accuracy VHDL-AMS model. This led to a simulation time improvement through the run-time switching approach by a factor of over 13. By the relation of 15:1 of transmitted frames, a larger improvement factor could be expected. However, due to the parallel processing overhead there are practical limits in improvement potential. For more complex simulations or for other configurations, larger improvements can be expected. Especially if the difference between total simulation time and switched simulation time is high the overhead due to switching gets negligible small.

An important topic when using the run-time switched co-simulation approach is the accuracy of the results. In Figure 6 a zoom to some bits of a FlexRay frame is shown. The frame was transmitted over the line by using the low level VHDL-AMS model during the whole co-simulation. It is compared to the same bits of the same frame but now only this frame was transmitted via the VHDL-AMS models. The rest of the simulated time the high level SystemC model has been used for frame transmission. It can be seen that both results are nearly identical, only showing some minor differences. Since we are using the same models, the good correlation shows that the model initialization was performed correctly. The differences have no effect on the higher levels of the FlexRay system as they are negligible small. However, the use of the more accurate model is justified since its behaviour affects the higher levels (e.g. deformed and shortened bits), and cannot be simulated by using SystemC.

The following calculations have been done by exporting the simulated waveforms from the VHDL-AMS simulator (Mentor AdvanceMS) as comma-separated-value files (n samples with a sample time of $0.5ns$) and importing them into MATLAB. The arithmetic mean $\overline{x}$ of a number of samples is calculated as

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

and the standard deviation s_x is

$$s_x = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2}$$

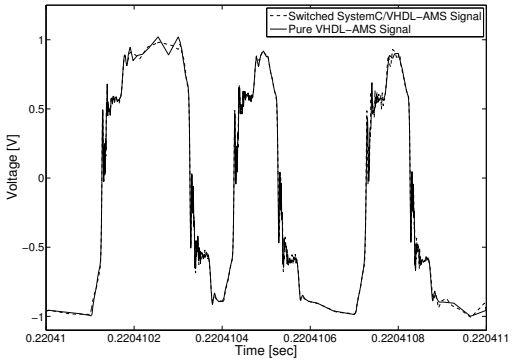


Figure 6. Simulation results of some bits of one frame (pure VHDL-AMS compared with switched SystemC/VHDL-AMS co-simulation)

For the difference ($z_i = x_i - y_i$) between the pure (x) and the switched (y) waveform the arithmetic mean and the standard deviation during the switched frame are

$$\overline{z} = -1.528 * 10^{-4} V, s_z = 3.08 * 10^{-2} V.$$

The pearson product-moment correlation coefficient (also sample correlation coefficient) between the pure VHDL-AMS waveform and the switched VHDL-AMS waveform is calculated as

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \sum_{i=1}^{n}(y_i - \overline{y})^2}}$$

The sample correlation coefficient is

$$r_{xy} = 0.9992$$

during the switched FlexRay frame (a factor of 1.0 means perfect correlation). It can be seen that the pure and the switched waveform do fit in a very good way.

V. Conclusion and Outlook

Inter-language co-simulation is a well known approach in order to provide selectable accuracy for the different components, thus increasing simulation accuracy while reducing validation efforts and computation resources. We go a step further and propose a run-time switching approach in order to modify the accuracy level of selected components during the co-simulation. This approach is especially interesting for the analysis of short time intervals during long total simulated times. A typical application for this approach is the analysis of distributed automotive embedded systems: The application runs for several seconds or minutes and the effects of interests (e.g. EMI/ESD) require detailed simulation models at the analogue level. For an example of an embedded automotive system, the proposed method reduced the simulation time by more than a magnitude while providing simulation accuracy nearly identical compared to standard static co-simulation. The improvement potential in simulation time mainly depends on the relation between total and switched simulated time. Hence, the simulation time improvements can be much higher than just a magnitude.

Future steps will include the fully automated integration of the run-time co-simulation switching approach into the design as well as the optimization of the switching parameters (trigger conditions, parallel processing time).

Acknowledgment

The authors wish to thank the "COMET K2 Forschungsförderungs-Programm" of the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT), the Austrian Federal Ministry of Economics and Labour (BMWA), Österreichische Forschungsförderungsgesellschaft mbH (FFG), Das Land Steiermark and Steirische Wirtschaftsförderung (SFG) for their financial support.

Additionally we would like to thank the supporting companies and project partners austriamicrosystems, AVL List and CISC Semiconductor as well as Graz University of Technology and the University of Applied Sciences FH Joanneum.

References

[1] F. Bouchhima, M. Briere, G. Nicolescu, M. Abid, and E. Aboulhamid, "A SystemC/Simulink Co-Simulation Framework for Continuous/Discrete-Events Simulation," in *Behavioral Modeling and Simulation Workshop, Proceedings of the 2006 IEEE International*, 2006, pp. 1–6.

[2] P. Birrer and W. Hartong, "Incorporating SystemC in Analog/Mixed-Signal Design Flow," in *Forum on Specification and Design Languages, Proceedings of the 8th International*, 2005, pp. 173–178.

[3] S. Kajtazovic, C. Steger, A. Schuhai, and M. Pistauer, "Automatic generation of a coverification platform," *Applications of Specification and Design Languages for SoCs: Selected papers from FDL 2005*, pp. 187–203, 2006.

[4] K. Hines and G. Borriello, "Selective Focus as a Means of Improving Geographically Distributed Embedded System Co-Simulation," in *Rapid System Prototyping, 1997. Proceedings., 8th IEEE International Workshop on*, 1997, pp. 58–62.

[5] ——, "Dynamic Communication Models in Embedded System Co-Simulation," in *Design Automation Conference, 1997.*, 1997, pp. 395–400.

[6] S. Yoo and A. Jerraya, "Hardware/Software Cosimulation from Interface Perspective," *Computers and Digital Techniques, IEE Proceedings* -, vol. 152, no. 3, pp. 369–379, 2005.

[7] M. Radetzki and R. Khaligh, "Accuracy-Adaptive Simulation of Transaction Level Models," in *Design, Automation and Test in Europe, 2008. DATE '08*, 2008, pp. 788–791.

[8] G. Beltrame, D. Sciuto, and C. Silvano, "Multi-Accuracy Power and Performance Transaction-Level Modeling," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, no. 10, pp. 1830–1842, 2007.

[9] "Flexray Communications Systems – Protocol Specification Version 2.1 Rev A, available at http://www.flexray.com," FlexRay Consortium, 2005.

[10] M. Karner, C. Steger, R. Weiss, E. Armengaud, D. Watzenig, and G. Knoll, "Verification and Analysis of Dependable Automotive Communication Systems Based on HW/SW Co-Simulation," in *IEEE International Conference on Emerging Technologies and Factory Automation. ETFA '08*, 2008, pp. 444–447.

Holistic Simulation of FlexRay Networks by Using Run-Time Model Switching

Michael Karner*, Eric Armengaud†, Christian Steger* and Reinhold Weiss*
*Institute for Technical Informatics, Graz University of Technology, Austria
†Virtual Vehicle Competence Center, Austria
{michael.karner,steger,rweiss}@tugraz.at, eric.armengaud@v2c2.at

***Abstract*—Automotive network technologies such as FlexRay present a cost-optimized structure in order to tailor the system to the required functionalities and to the environment. The space exploration for optimization of single components (cable, transceiver, communication controller, middleware, application) as well as the integration of these components (e.g. selection of the topology) are complex activities that can be efficiently supported by means of simulation. The main challenge while simulating communication architectures is to efficiently integrate the heterogeneous models in order to obtain accurate results for a relevant operation time of the system. In this work, a run-time model switching method is introduced for the holistic simulation of FlexRay networks. Based on a complete modeling of the main network components, the simulation performance increase is analyzed and the new test and diagnosis possibilities resulting from this holistic approach are discussed.**

I. Introduction

The FlexRay technology [1] is being introduced as a new wired network for automotive high-speed control applications. This communication protocol provides features like increased data rates (a factor of 10 faster than the CAN protocol) and the simultaneously support of time-triggered and event-triggered communication schemes for both deterministic network behavior and the efficient transmission of single events. This protocol, in comparison to predecessors (e.g. CAN), presents a very large flexibility that results in a huge number of implementation variants. Parameters such as topology (passive line, active star, hybrid topologies), bus schedule (e.g. slot / cycle length) and communication matrix (mapping between ECUs, slots and frames) are as much variables that can influence the quality of the communication.

One important aim of the TEODACS project is to understand the interactions between the layers building a dependable network and to evaluate the different effects influencing the communication (e.g. topology, EMC). The approach is based on the development of a heterogeneous co-simulation model of a FlexRay network tightly interfaced to a realistic FlexRay prototype. This combines the good observability and diagnosability provided by the simulation environment with the realistic system behavior provided by the hardware prototype.

The holistic simulation of a FlexRay network represents a challenge due to the heterogeneous nature of the system. Hence, a typical network consists of analog components (physical layer, transceivers), digital components (data link layer, communication controllers), basic software components (middleware, AUTOSAR) and software components (application). An important problem is to obtain a long *simulated time* (time interval under observation) for the analysis of the entire system, while at the same time keeping a reasonable *simulation time* (computation speed of the simulation) and a reasonable simulation accuracy for each component. This is especially difficult for lower layers that require more performance for the simulation of complex analog components.

Within this work, the run-time model switching method introduced in [2] is integrated within our FlexRay co-simulation environment. The main concept is to provide different simulation models of the same component (e.g. one for high speed, another one for high accuracy) that can be switched during the simulation run. This approach provides the test designer with a new degree of freedom for defining for each component the time intervals for which a computational expensive high-detail simulation model is required and when a simpler and faster model suffices. This method can be compared with an oscilloscope having the capability to zoom into significant parts of the simulation, while performing a fast (less accurate) run for the less interesting part of the simulation.

This document presents two main contributions. First, the performance increase resulting from the run-time model switching method is analyzed (both theoretically and experimentally). Second, the resulting heterogeneous co-simulation model of the FlexRay network is presented, and the resulting cross-layer analysis possibilities (e.g. analysis of the influence between the physical and data link layer) are discussed. The document is organized as follows: Section II provides an overview of the state of the art regarding simulation methods of automotive networks. In Section III, the concepts of the run-time model switching method are illustrated and the rationale for the performance analysis presented. The focus of Section IV is set to the experimental validation of our approach and finally Section V concludes this work.

II. Simulation of FlexRay Communication Networks: State of the Art

For the simulation of different parts of a (FlexRay) communication network several solutions and simulation models have been developed. For example, in [3] a SystemC-based FlexRay communication controller is presented for the timing analysis of interconnected AUTOSAR components. Furthermore, a communication controller model developed using Verilog is proposed in [4] for the assessment of message missing failures. In [5], another implementation of a communication controller

using SystemC is described. Yet another simulation model of a FlexRay communication controller is demonstrated in [6]. Here, the controller is implemented using standard C language and simulated on a PC used to interact with real hardware components like engines.

Moreover, different FlexRay transceiver models are available: A generic model implemented using VHDL-AMS and including physical effects like thermal power is presented in [7], while the model presented in [8] is a specific behavioral Saber model of an NXP transceiver. The simulation setup described in [9] incorporates several simulation models for the FlexRay physical layer including cable, passive star and transceiver simulation models using Synopsis Saber. All of these approaches are focused on one single component or layer, thus the interactions with the other components and protocol layers involved are difficult to analyze (e.g. interaction between physical layer and data link layer).

In [10], the simulation of large heterogeneous systems including several ECUs communicating via a network is implemented by using transaction based modeling. The authors use a hardware abstraction layer for high level access to the hardware components and simulate at a very high abstraction layer, hence achieving a high simulation speed but losing many important details about the communication.

A further approach for analyzing dependable communication networks is residual bus simulation which enables the emulation of an entire sub-network. Industrial solutions from e.g. Elektrobit, dSPACE and Vector are available. All of these examples are a combination of hardware and software components: The software emulates the functionalities of the missing ECUs and the application messages are transmitted to a real network using dedicated hardware components. Hence, the advantages of simulation (observability, traceability, flexibility,...) are lost for many parts of the system.

It can be seen that there exist several solutions for the simulation of dependable communication networks, especially of FlexRay networks. However, these simulations usually cover only a specific part of the network and/or are working in a very abstract way. No holistic approach covering all parts of the network (from physical layer up to the application layer) with sufficient accuracy is available, making a comprehensive in-depth analysis nearly impossible to achieve by using existing setups.

III. Run-Time Co-Simulation Model Switching

A. Run-time switching

Co-simulation is a well known approach for the integral simulation and analysis of different simulation models implemented at different abstraction levels and/or by using different hardware description languages (HDL). However, when combining highly heterogeneous simulation models, standard co-simulation presents limitations: the overall simulation performance depends on the slowest simulator involved, slowing down the other simulators to guarantee synchronization and data exchange. The drop in simulation performance is especially a problem when mixing accurate models (e.g. physical/analog level) with system level simulation in a common co-simulation. At system level, the focus is set to short *simulation time* (computation speed of the simulation) in order to obtain longer *simulated time* (time interval under observation). At physical level the simulation times are typically much higher because of the simulation of complex analog components. Hence, relatively short simulated times are explored here ($\mu s, ms$). The guidelines of the different abstraction level models are quite opposite: For system level models, the simulated time of several seconds can not be handled by the low level physical simulation model. This would lead to enormous simulation times for the low level model that may not be acceptable or even accomplishable within a reasonable amount of time. On the other hand, the very short simulated times for the low level physical simulation model are too short for the abstracted system level model. During this very short time interval, nothing of interest may happen at system level, thus rendering the system level simulation futile.

To overcome this problem, the methodology of dynamically switching the co-simulation models used is proposed in [2]. The basic idea is to change at run-time the simulation models that are used for a specific part of the co-simulation and for given components. For example, while most of the time a fast less-detailed SystemC simulation model is used for the cable, for special intervals a more-detailed low level VHDL-AMS model is utilized. This speeds up the co-simulation (compared to low level simulation for the whole simulation) while at the same time providing the high accuracy when required. The principle is shown in Figure 1. Splitters and mergers (switches) are used to perform the co-simulation model switching and to ensure a correct data flow within the system. For more details about the basic methodology and the implementation, see [2].

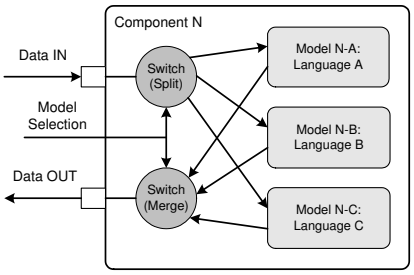


Fig. 1. Methodology of Run-Time Co-Simulation Model Switching

The run-time switching approach requires the synchronization between the different simulation models of a given component in order to assure correct continuous service delivery of this component. This is performed via parallel processing where the upcoming simulation model for the component is fed with the current data. This allows for an initialization of the model so that switching can take place seamlessly. Notice here that a trade-off exists between on one hand setting the parallel processing time large enough in order to allow for a proper model initialization, and on the other hand minimizing parallel processing in order to reduce simulation time. The appropriate parallel processing time has to be determined either by in-depth system knowledge or by experiments.

A further challenge is to set the point in time where the

switching occurs. In the case of FlexRay, its periodic and a-priori known behavior can be used to find synchronization points where the system is in a known state (e.g. at cycle, frame or even bit borders). Hence, switching can take place at such points without problems. The definition of switching points is a rather system specific task with the basic goal of determining either periodic or beneficial points during system execution.

B. Performance analysis

In the run-time simulation model switching approach, the developer has the possibility to simply adjust the trade-off between the simulation performance and the model accuracy achieved. This is of great advantage, as in fact the developer can specify the simulation performance in advance, hence adjusting the simulation time needed. By specifying the simulated time (including parallel processing) for each abstraction level of the switched component the developer can select the relation between simulation performance and accuracy according to the requested needs. This is shown in Figure 2. Thus, the simulation performance improvement factor in comparison with standard co-simulation methodologies is more or less freely selectable by the developer. This is of great advantage when simulating complex networks.

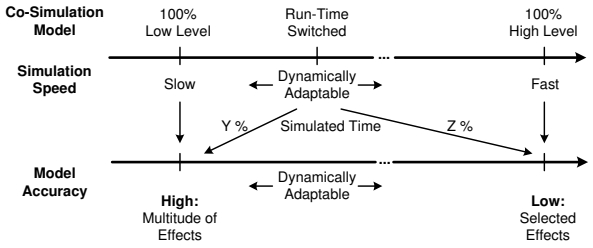


Fig. 2. Shifting the Complexity of Co-Simulation Models

To estimate the achieved simulation performance by applying the run-time co-simulation model switching to our FlexRay network simulation we extended the theory presented in [2]. The total simulated time T in seconds (s) for each switched co-simulation model consisting of M different abstraction levels is written as $T[s] = \sum_{n=1}^{M} T_n[s]$. This time already includes the defined amount of parallel processing. The total simulated time spent on parallel processing per model abstraction level n is

$$T_{P_n}[s] = \sum_{i=1}^{X} t_{P_{n_i}} \tag{1}$$

with $t_{P_{n_i}}[s]$ as the parallel processing time for this abstraction level n before switch number i is activated. Following this, the simulated time where the abstraction level n is running stand-alone because of switch i is stated as $t_{S_{n_i}}[s]$ and the total simulated time where the simulation model for abstraction level n is running stand-alone is

$$T_{S_n}[s] = \sum_{i=1}^{X} t_{S_{n_i}} \tag{2}$$

with X as the total number of switches including this abstraction level. Summing up, the total simulated time for all abstraction models including parallel processing is

$$T[s] = \sum_{n=1}^{M} T_n = \sum_{n=1}^{M} (T_{P_n} + T_{S_n}) \tag{3}$$

Out of this, a factor w for the estimated simulation performance achieved by applying the run-time co-simulation model switching approach can be derived.

$$w = \frac{1}{\sum_{n=1}^{M} T_n \cdot C_n} \tag{4}$$

Here, $C_n[\frac{1}{s}]$ are the implementation dependent simulation costs compared to real-time. This factor has to be derived out of experiments or experience. Typical factors for C_n are e.g. $10^6 - 10^9$ for VHDL-AMS models or $10^2 - 10^4$ for SystemC models. Following this, the simulation performance depending on the switching configuration and the abstraction levels involved can be estimated as

$$w = \frac{1}{\sum_{n=1}^{M} \left[\sum_{i=1}^{X} (t_{P_{n_i}} + t_{S_{n_i}}) \right] \cdot C_n} \tag{5}$$

This allows the designer to compare the estimated performance of different switching configurations. This factor is no absolute value; it only shows the relation between different switching configurations. For example, it can be calculated how a lengthening of the parallel processing time or of the simulated time for one abstraction level affects the total simulation performance of the run-time switched co-simulation. An example calculation will be shown in the next section.

IV. Advanced Co-Simulation of Dependable Communication Architectures: The FlexRay Example

A. FlexRay co-simulation environment

The co-simulation environment developed within the TEODACS project implements a co-simulation of a FlexRay communication network from the physical layer up to the application, see Figure 3. As co-simulation framework we use the commercially available co-design tool CISC SyAD [11]. Additionally, we use the car simulator CarMaker / AVL InMotion^{TM} [12] to generate realistic data to be transferred via the FlexRay network.

Regarding the simulation models of the network components, we combine models at different abstraction levels and implemented using different HDLs. The physical layer (cables, active/passive star, transceiver) is implemented both in more-detailed VHDL-AMS models and also using fast but less-detailed SystemC simulation models. This provides us with the ability to apply the run-time co-simulation model switching approach to the most computational intensive layer: the physical layer. The FlexRay communication controller (data link layer) is implemented using SystemC.

The middleware (selected AUTOSAR [13] concepts) and the software application are also implemented using SystemC

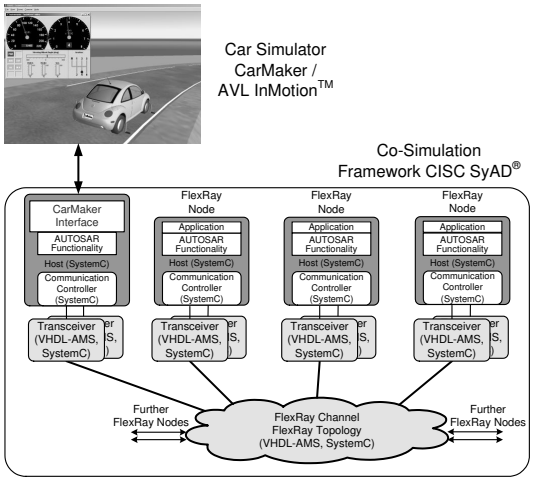


Fig. 3. FlexRay Communication Network Simulation

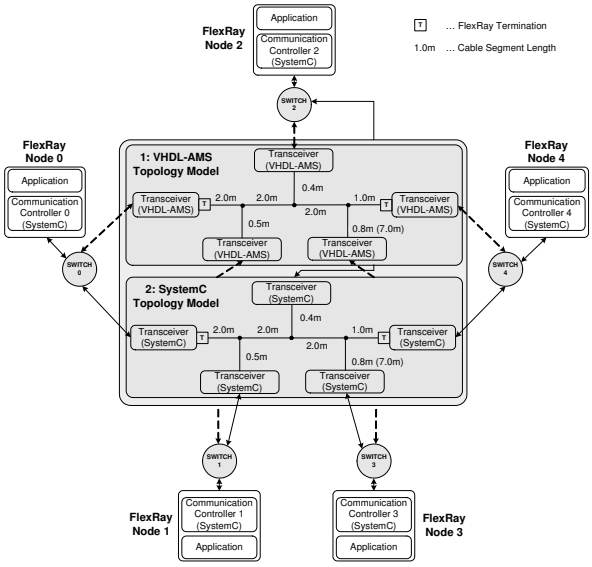


Fig. 4. Run-Time Switched FlexRay Co-Simulation Example

and C/C++. Hence, the same application code can be re-used for real hardware FlexRay controllers. By using the run-time co-simulation switching this setup allows us to analyze the complete network behavior and especially all the interactions between the different layers of the network – from the physical layer up to the application – within a reasonable amount of time.

B. Experimental setup

For our experiments we created the sample FlexRay setup shown in Figure 4. The network is consisting of 5 FlexRay nodes including transceivers, communication controllers and software. The topology with different cable lengths and the FlexRay specific line termination is also shown in Figure 4. We applied the run-time co-simulation model switching by inserting SystemC-based splitters/mergers (switches; see Section III) in the co-simulation to handle the switching process. We switched between fast but less-detailed SystemC simulation models for transceivers and cables (only including length-based delay and attenuation) and very slow but more-detailed VHDL-AMS models (including all cable effects like reflections, termination etc.) for these components. The signals to be switched are at the interface between the communication controller and the transceiver: receive data (RxD) and transmit data (TxD). This allows the investigation of effects occurring at physical layer on higher layers like data link layer (e.g. shortening/lengthening of bits, misinterpretations because of low signal integrity) and application (e.g. missing data because of frame corruption, problems due to startup/synchronization errors etc). Also the other way is possible: effects of the protocol configuration or the bit structure on the physical waveforms can be analyzed.

An important topic when applying the run-time co-simulation model switching is the choice of switching point and parallel processing duration. As FlexRay is a time-triggered communication protocol there exist pre-defined points where the state of the system is completely known. Hence, it is the most advantageous solution to select the switching points there with a parallel processing time large enough to allow proper initialization of the co-simulation model to be switched to. In our experiments, we defined that most of the time the fast SystemC model for topology and transceivers is used. However, in every frame the header is transmitted via the more-detailed but slow VHDL-AMS models.

In the experiments the following FlexRay-related configuration is used: A total of 16 FlexRay communication cycles is simulated, including startup and integration of the nodes. Each cycle lasts 5ms and, if all nodes are integrated, contains 12 FlexRay frames. Each frame is $26\mu s$ long and transmitted at 10MBit/s. In each frame, the first $6\mu s$ (t_{S1_i}) are transmitted via the more-detailed VHDL-AMS ($n = 1$) models of topology and transceiver. The remaining part ($t_{S2_i} = 20\mu s$) of the frame is transmitted using the fast SystemC ($n = 2$) models, see Figure 4. We assume that the attributes of the signal integrity will not change within one frame and therefore the detailed analysis of the first $6\mu s$ will provide relevant results for the entire frame. In this experiment, up to 12 switches occur per cycle. Based on the FlexRay schedule and experiments, the parallel processing time is set to $t_{P1_i} = t_{P2_i} = 2\mu s$.

C. Results

Four different experiments have been performed based on the setup described in Section IV-B. Experiment one was the setup shown in Figure 4 but using SystemC simulation models only, hence, without switching. The second experiment was performed using VHDL-AMS only for the models of transceiver and cables. Here, also no switching was used. The third experiment applied the run-time simulation model switching approach to the co-simulation. The switching configuration as described in Section IV-B was used and switching

TABLE I
EXAMPLE SIMULATION TIME COMPARISON FOR ONE FLEXRAY CYCLE WITH 12 FRAMES

	100% VHDL-AMS	Switched	100% SystemC
Seconds	88012	31757	307
Factor	287	104	1

occurred after the FlexRay header ($6\mu s$) has been transmitted. For these first three experiments, a valid FlexRay network was simulated (correct cable length and termination). In the fourth experiment, the network topology was modified. The line from the bus to node 3 has been altered from 0.8m to 7.0m. The faulty termination concept led to several physical effects (e.g. reflections) that have degraded the signal integrity within the FlexRay network. For this experiment, the same switching setup as in experiment three was used.

In the first three experiments, the switching within the frames worked as expected and the simulation performance was improved compared to pure VHDL-AMS simulation. A comparison is shown in Table I. As the network topology was designed according to the FlexRay specification, the frames were transmitted correctly and the upper layers worked without problems.

However, the situation is a bit different for the fourth example: here, the setup can definitely be regarded as borderline case because of the 7.0m long, unterminated line to node 3, leading to reflections. The simulation demonstrated that, depending on the transmitting node, different nodes have problems in receiving correct frames. For example, the data node 2 is sending can be received without any problems by nodes 0, 1 and 4, while node 3 reports a header error check failure. However, if node 1 is transmitting the nodes 0, 2 and 3 receive the data correctly, while node 4 reports an error in the frame header. These effects lead to reception asymmetry that can move the system into inconsistent states and thus should be avoided.

In Figure 6 it can be seen that for a frame transmitted by node 4 (TxD4), node 3 using the VHDL-AMS model is not able to receive any meaningful data on RxD3 because of reflections etc. Hence, the communication controller very early reports an error that it is not able to find a byte start sequence (BSS), see Figure 5. The problems can be easily detected by looking at the resulting differential voltage waveform at node 3. It is far away from any meaningful state. In Figure 6, the SystemC topology model starts parallel processing at around 285.749ms before switching at 285.751ms. During parallel processing the output of the SystemC model is not used. It can be seen that the SystemC model does not show any distortion (because it only includes length-based delay and attenuation); hence, no error would be detected during the whole simulation if the SystemC model would be the primary model all the time. The situation is a bit different for nodes 0, 1 and 2. In fact, using the VHDL-AMS model they are able to receive the header data (transmitted by node 4) correctly most of the time. But at around 285.750ms, one bit sequence gets altered by a reflection. The high bit sequence is shortened and the following low bit sequence is stretched, leading to a header checksum error detected by the controllers, see Figure 5. By looking at the SystemC data it is obvious that this error does not happen there. These errors definitely influence the application as, depending on the sender node, a node may or may not receive frames correctly. At the worst, the FlexRay network may lose its synchronization and get split apart in different clusters of nodes that are not able to communicate with each other (cliques). The switched simulation detected all of the higher level errors like shown in Figure 5 that were also detected during the full VHDL-AMS only simulation run.

```
Node 3 - codec: BSS_ERROR!
Time: 285747500 ns

Node 2 - WARNING CODEC CHANNEL A: HEADER CRC FAILED!
Time: 285751650 ns

Node 1 - WARNING CODEC CHANNEL A: HEADER CRC FAILED!
Time: 285751662500 ps

Node 0 - WARNING CODEC CHANNEL A: HEADER CRC FAILED!
Time: 285751675 ns
```

Fig. 5. Communication Controller Error Detection Output Excerpt (Exp. 4)

With these experimental results we are able to verify the simulation performance estimation proposed in Section III. We disregard the negligible simulated times with no frame transmission (for easier determination of C_{AMS} and C_{SYSC}) and use the switching configuration described in Section IV-B. Based on experiments we determine $C_{AMS} = C_1 = 10^6, C_{SYSC} = C_2 = 3 \cdot 10^3$ and $i = 12$ (as 12 frames are transmitted in one cycle). By using Formula 5 it is calculated that the simulation performance difference between pure VHDL-AMS simulation and the switched simulation should be about 3.22. Via Table I it is ascertainable that the actual difference is 2.79, a discrepancy of about 15% to the estimation. Considering that the values of C_{AMS} and C_{SYSC} are basic approximations to the reality this is a rather good result. The simulation performance of the switched co-simulation compared to the SystemC only simulation is estimated by Formula 5 as 103 while the results of Table I show a simulation performance difference of 104, hence nearly identical results.

V. CONCLUSION

Holistic simulation of automotive communication architectures is strongly required for an early system integration as well as for an efficient cross-domain design exploration and optimization. The highly heterogeneous nature of the system under consideration strongly limits the simulation (and therefore the analysis) capabilities. In this work, we have integrated a run-time model switching method within our heterogeneous co-simulation model of a FlexRay network. The rationale for the expected performance increase has been presented and further experimentally validated. Another important result of this work is the fault propagation analysis within the communication architecture. Hence, we have pointed out that the same faulty topology might present very different symptoms and lead to different reactions of the system.

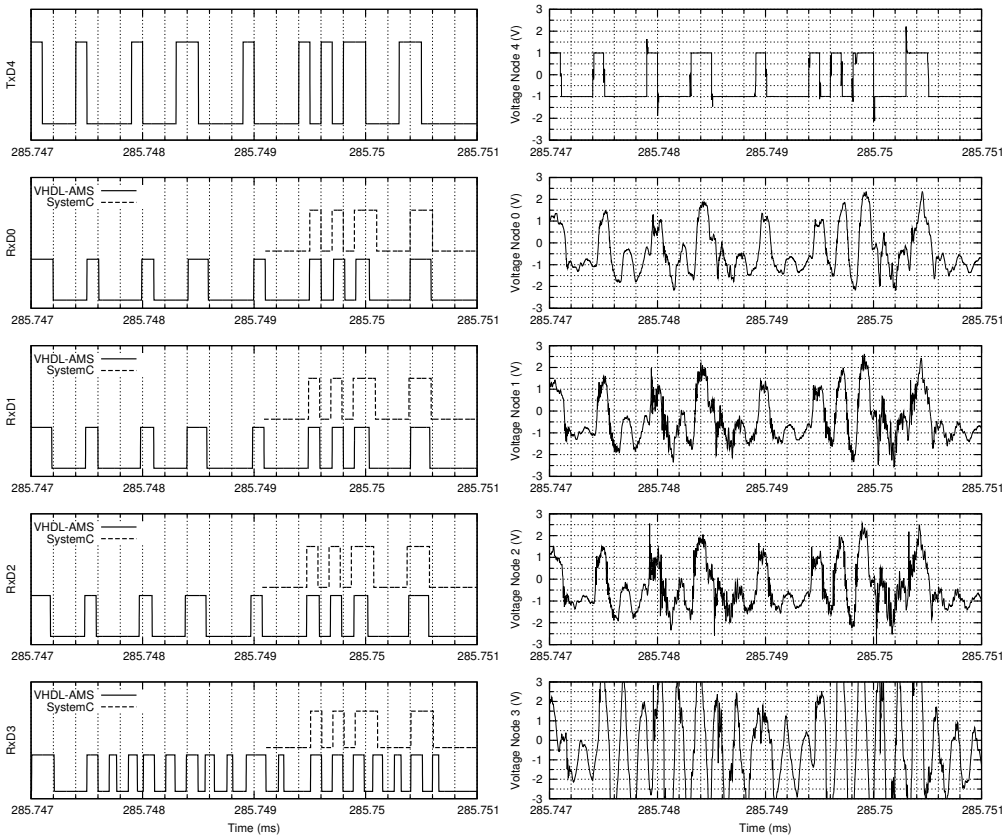


Fig. 6. TxD/RxD Excerpt of a Switched FlexRay Frame

ACKNOWLEDGMENT

The authors wish to thank the "COMET K2 Forschungsförderungs-Programm" of the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT), the Austrian Federal Ministry of Economics and Labour (BMWA), Österreichische Forschungsförderungsgesellschaft mbH (FFG), Das Land Steiermark and Steirische Wirtschaftsförderung (SFG) for their financial support. Additionally we would like to thank the supporting companies and project partners austriamicrosystems, AVL List and CISC Semiconductor as well as Graz University of Technology and the University of Applied Sciences FH Joanneum. Further information about the TEODACS project can be found on www.teodacs.com.

REFERENCES

[1] *FlexRay Communications System – Protocol Specification Version 2.1 A*, FlexRay Consortium, December 2005. [Online]. Available: http://www.flexray.com

[2] M. Karner, C. Steger, R. Weiss, and E. Armengaud, "Optimizing HW/SW Co-Simulation based on Run-Time Model Switching," in *FDL2009, Forum on specification & Design Languages*, September 2009, pp. 1–6.

[3] M. Krause, O. Bringmann, A. Hergenhan, G. Tabanoglu, and W. Rosenstiel, "Timing Simulation of Interconnected AUTOSAR Software-Components," in *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE '07*, 2007, pp. 1–6.

[4] V. Lari, M. Dehbashi, S. G. Miremadi, and N. Farazmand, "Assessment of Message Missing Failures in FlexRay-Based Networks," in *Dependable Computing, 2007. PRDC 2007. 13th Pacific Rim International Symposium on*, 2007, pp. 191–194.

[5] W. S. Kim, H. A. Kim, J.-H. Ahn, and B. Moon, "System-Level Development and Verification of the FlexRay Communication Controller Model Based on SystemC," *Future Generation Communication and Networking*, vol. 2, pp. 124–127, 2008.

[6] C. Xu and Y. Zhang, "Simulation of FlexRay Communication Using C Language," *Computer Science and Computational Technology, International Symposium on*, vol. 2, pp. 272–276, 2008.

[7] *FlexRay Transceiver Model*, CISC Semiconductor Design+Consulting GmbH, Klagenfurt, Austria, Feb. 2007. [Online]. Available: http://www.cisc.at/flexray

[8] *NXP FlexRay network simulations: Safeguard the operation of your FlexRay network architectures*, NXP Semiconductors, The Netherlands, Nov. 2007. [Online]. Available: http://www.nxp.com/acrobat_download/literature/9397/75016200.pdf

[9] T. Gerke and D. Bollati, "Development of the Physical Layer and Signal Integrity Analysis of FlexRay Design Systems," in *Simulation & Modelling Mechatronics (SP-2111)*, 2007.

[10] R. Buchmann, M. Cartron, and Y. Bonhomme, "Transaction-based Modeling for Large Scale Simulations of Heterogeneous Systems," in *Simutools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, pp. 1–2.

[11] *SyAD Online Documentation*, CISC Semiconductor Design+Consulting GmbH, Klagenfurt, Austria, September 2009. [Online]. Available: http://www.cisc.at/syad

[12] *AVL Hybrid Development Platform™*, AVL List GmbH, May 2007. [Online]. Available: http://www.avl.com

[13] *AUTOSAR: Specification of the Virtual Functional Bus*, AUTOSAR GbR, February 2008. [Online]. Available: http://www.autosar.org

EINGEREICHT AM
31. JAN. 2011

(51) IPC: K 14668

AT PATENTSCHRIFT

(11) Nr.

*(Bei der Anmeldung sind nur die eingerahmten Felder auszufüllen - bitte **fett umrandete Felder unbedingt ausfüllen!**)*

(73) Patentanmelder (bzw. –inhaber):
Kompetenzzentrum – Das Virtuelle Fahrzeug Forschungsgesellschaft mbH (ViF), A-8010 Graz, AT
Technische Universität Graz, A-8010 Graz, AT
CISC Semiconductor Design+Consulting GmbH, A-9020 Klagenfurt, AT

(54) Titel der Anmeldung:
Verfahren zum Umschalten von heterogenen Simulationsmodellen zur Laufzeit

(61) Zusatz zu Patent Nr.

(66) Umwandlung von *GM* /

(62) Gesonderte Anmeldung aus (Teilung): *A* /

(30) Priorität(en):

(72) Erfinder:
Steger Christian Dr., A-8045 Graz, AT
Armengaud Eric Dr., A-8045 Graz, AT
Karner Michael DI, A-8200 Hofstätten/Raab, AT

(22)(21) Anmeldetag, Aktenzeichen: 2009-09-18, A 1479/2009

(42)(45) Ausgabetag / Beginn der Patentdauer:

Längste mögliche Dauer:

(56) Ermittelter Stand der Technik:

US 2009/0063120 A1

BOMBANA, M. ET AL. 'System C-VHDL co-simulation and synthesis in the HW domain' Design, In: Automation and Test in Europe Conference and Exhibition, 2003, 19.12.2003, p. 101-105, ISBN: 0-7695-1870-2

EP 0 772 140 A1

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) **EP 2 299 376 A1**

(12) **EUROPÄISCHE PATENTANMELDUNG**

(43) Veröffentlichungstag:
23.03.2011 Patentblatt 2011/12

(51) Int Cl.:
G06F 17/50 *(2006.01)*

(21) Anmeldenummer: **10177698.7**

(22) Anmeldetag: **20.09.2010**

(84) Benannte Vertragsstaaten:
AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO SE SI SK SM TR
Benannte Erstreckungsstaaten:
BA ME RS

(30) Priorität: **18.09.2009 AT 14792009**

(71) Anmelder:
- **Kompetenzzentrum- das Virtuelle Fahrzeug Forschungsgesellschaft mbH (VIF)**
  8010 Graz (AT)
- **Technische Universität Graz**
  8010 Graz (AT)
- **Cisc Semiconductor Design+Consulting Gmbh**
  9020 Klagenfurt (AT)

(72) Erfinder:
- **Steger, Christian**
  A-8045, Graz (AT)
- **Karner, Michael**
  A-8200, Hofstätten/Raab (AT)
- **Armengaud, Eric**
  A-8045, Graz (AT)

(74) Vertreter: **Margotti, Herwig Franz**
Schwarz & Partner
Patentanwälte
Wipplingerstrasse 30
1010 Wien (AT)

(54) **Verfahren zum Umschalten von heterogenen Simulationsmodellen zur Laufzeit**

(57) Diese Erfindung beschreibt ein Verfahren, um zur Laufzeit zwischen verschiedenen Simulationsebenen für dieselbe Komponente umzuschalten und dadurch die Simulationseffizienz der Co-Simulationsumgebung zu steigern. Die Beschreibung der jeweiligen Komponente erfolgt in mehreren Simulationsebenen, welche in einer geeigneten Hardwarebeschreibungssprache und auf einer geeigneten Abstraktionsebene modelliert sind. Eine für die Simulation verwendete Komponente kann dabei in verschiedenen Hardwarebeschreibungssprachen und auf verschiedenen Abstraktionsebenen implementiert werden. Diese Erfindung kombiniert damit die Möglichkeit, auf hoher Abstraktionsebene (z.B. Systemebene) lange simulierbare Zeiten zu erzielen mit der Möglichkeit, für einen definierten Abschnitt der simulierten Zeit die hohe Genauigkeit eines auf tiefer Ebene (z.B. physikalische Ebene) realisierten Simulationsmodells erzielen zu können. Der Hauptvorteil dieser Erfindung ist die Möglichkeit, die Simulation in Intervalle hoher Genauigkeit und Intervalle mit niedriger Genauigkeit einzuteilen.

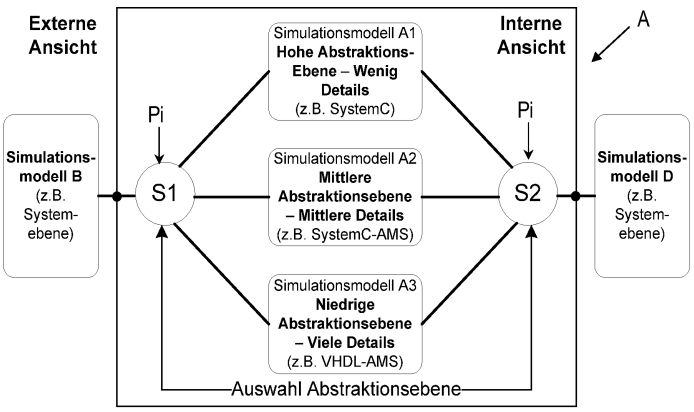


Fig. 1

EP 2 299 376 A1

Beschreibung

[0001] Für Embedded Systems (Eingebettete Systeme) ist die Verhaltenssimulation auf Systemebene (d.h. auf Ebene des gesamten Systems inkl. Informationsverarbeitung und Umwelt des Systems) ein wichtiger Schritt, um sowohl die interne funktionale Architektur des Systems als auch die Schnittstellen zwischen den Komponenten des Systems (d.h. SubSysteme des Gesamtsystems, in der Literatur auch Modul genannt) zu validieren. Dafür werden typischerweise abstrakte Modelle der einzelnen Komponenten verwendet. Diese high-level Modelle abstrahieren allerdings unter Umständen ein bestimmtes Modellverhalten, welches das komplette System beeinflussen kann und die Realität nur schlecht abbildet. Eine besondere Herausforderung ist es, bereits in einem frühen Stadium der Entwicklung ausgewählte Komponenten durch besonders detaillierte Verhaltensmodelle (d.h. ein Modell, welches wesentliche Aspekte des Verhaltens einer Komponente abbildet) darzustellen, um damit schon früh mit ausreichender Genauigkeit die gesamte Systemarchitektur validieren zu können. Unter Verhaltensmodellierungssprachen ist in diesem Zusammenhang allgemein eine Beschreibungssprache (z.B. Hardwarebeschreibungssprache wie SystemC, SystemC-AMS, VHDL-AMS aber auch Softwarebeschreibungssprache wie C/C++) des Verhaltens von Komponentenmodellen zu verstehen. Verhaltensmodellierungssprachen ermöglichen die Dokumentation, die Modellierung, die Simulation, den Entwurf und die Herstellung von Systemen.

[0002] Auf Systemebene liegt das Hauptaugenmerk auf einer kurzen Simulationszeit (hohe Simulationsgeschwindigkeit) um eine lange simulierte Zeit (Zeitintervall unter Beobachtung) zu erzielen. Auf physikalischer Ebene, auf der die elektrischen und mechanischen Eigenschaften des Systems modelliert werden (z.B. Signalpegel, Kontakte) mit diversen komplexen analogen Komponenten dagegen sind die Simulationszeiten wesentlich länger. Die Anforderungen der verschiedenen Ebenen an die Simulationsmodelle widersprechen sich also: Eine simulierte Zeit von mehreren Sekunden oder Minuten auf Systemebene kann nicht mittels low-level physikalischen Modellen erreicht werden. In diesem Fall würden sich enorme Simulationszeiten ergeben und die Simulation könnte nicht bzw. nicht in einer vernünftigen Zeit abgeschlossen werden. Auf der anderen Seite sind die simulierten Zeiten, die mit Modellen auf physikalischer Ebene erreichbar sind (z.B. einige Millisekunden), deutlich zu kurz für das abstrakte Modell auf Systemebene. In diesen kurzen Zeiträumen geschieht auf Systemebene möglicherweise überhaupt nichts, was die Simulation auf Systemebene nutzlos machen würde.

[0003] Ein möglicher Lösungsansatz für diese Problematik ist Hardware/Software Co-Simulation. Bei der Co-Simulation kann der Entwickler unterschiedliche Teile des Systems mit unterschiedlichen Verhaltensmodellierungssprachen bzw. auf unterschiedlichen Abstraktionsebenen modellieren und in einer gemeinsamen Co-Simulation simulieren. Die Festlegung des Simulationsmodells für eine Komponente geschieht statisch vor der Simulation, und das Modell kann nicht während der Simulation geändert werden.

[0004] Diese Erfindung beschreibt ein Verfahren, um zur Laufzeit zwischen verschiedenen Simulationsebenen für dieselbe Komponente umzuschalten und dadurch die Simulationseffizienz der Co-Simulationsumgebung zu steigern. Die Beschreibung der jeweiligen Komponente erfolgt in mehreren Simulationsebenen, welche in einer geeigneten Verhaltensmodellierungssprache und auf einer geeigneten Abstraktionsebene modelliert sind. Eine für die Simulation verwendete Komponente kann dabei in verschiedenen Verhaltensmodellierungssprachen und auf verschiedenen Abstraktionsebenen implementiert werden. Die Idee ist nun, dass der Entwickler durch die Erfindung die Möglichkeit besitzt zu definieren, in welchen Zeitabschnitten der simulierten Zeit welche Simulationsebene für die Komponente zum Einsatz kommen soll, wann also z.B. eine langsame, detaillierte Simulationsebene benötigt wird und wann eine einfache, aber schnelle Simulationsebene für die Komponente ausreicht. Diese Erfindung kombiniert damit die Möglichkeit, auf hoher Abstraktionsebene (z.B. Systemebene) lange simulierte Zeiten zu erzielen mit der Möglichkeit, für einen definierten Abschnitt der simulierten Zeit die hohe Genauigkeit eines auf tiefer Ebene (z.B. physikalische Ebene) realisierten Simulationsmodells erzielen zu können.

[0005] Einige Forschungsgruppen beschäftigen sich mit Co-Simulation im Allgemeinen. Dabei sind diverse Co-Simulationsumgebungen entstanden, die es durch verschiedenste Methoden erlauben, unterschiedliche Abstraktionsebenen, Entwicklungssprachen und Simulatoren miteinander zu verbinden. So schlagen z.B. Bouchhima et al. ("A SystemC/Simulink Co-Simulation Framework for Continuous/Discrete-Events Simulation", Behavioral Modeling and Simulation Workshop, Proceedings of the 2006 IEEE International, pages 1-6, 2006) eine Co-Simulationsumgebung vor, die es erlaubt, SystemC- und Simulink-Modelle miteinander zu koppeln. Damit wird auch eine Verbindung zwischen zeitkontinuierlicher und zeitdiskreter Simulation geschaffen. Birrer et al. ("Incorporating SystemC in Analog/Mixed-Signal Design Flow. In Forum on Specification and Design Languages", Proceedings of the 8th International, pages 173-178, 2005) beschäftigen sich mit der Integrierung von SystemC Modellen in eine Verilog-A/AMS Simulation. Damit versuchen sie die Lücke zwischen Beschreibung des Systems auf Systemebene und seiner Hardwareimplementierung zu schließen. Ein anderer Ansatz für eine Co-Simulationsumgebung ist das HW/SW Co-Design Tool CISC SyAD ("System Architect Designer") (CISC Semiconductor. SyAD. CISC Semiconductor, Klagenfurt, Austria, Online Documentation, http://www.cisc.at/syad, 2009). Diese Umgebung ermöglicht es diverse Simulatoren sowohl aus der zeitdiskreten als

auch der zeitkontinuierlichen Domäne miteinander zu verbinden. Eine ähnliche Umgebung ist CosiMate der Firma ChiasTek (ChiasTek Inc. CosiMate. ChiasTek Inc, Chicago, USA, Online Documentation, http://www.chiastek.com, 2009). Die vorliegende Erfindung erweitert den Stand der Technik dahingehend, dass eine Umschaltung zur Laufzeit auch für Komponentenmodelle in unterschiedlichen Verhaltensbeschreibungssprachen erfolgen kann.

[0006] Es existieren bereits Forschungsprojekte im Gebiet des Umschaltens zwischen verschiedenen Simulationsebenen. Die existierenden Projekte beschäftigen sich allerdings mit dem Umschalten zwischen Simulationsebenen implementiert in derselben zeitdiskreten Hardwarebeschreibungssprache und nicht mit sprachübergreifendem Umschalten (verschiedenen Hardwarebeschreibungssprachen und Abstraktionsebenen, zeitdiskret und auch zeitkontinuierlich) und den sich daraus ergebenden Problemen. Hines und Boriello demonstrieren in ihren Arbeiten ("Selective Focus as a Means of Improving Geographically Distributed Embedded System Co-Simulation", Proceedings of the 8th IEEE International Workshop on Rapid System Prototyping, pages 58-62, 1997) und ("Dynamic Communication Models in Embedded System Co-Simulation" Proceedings of the 34th Design Automation Conference, pages 395-400, 1997) einen Ansatz zur Verwendung von dynamischen Kommunikationsmodellen mittels des Tools "Pia". Dieses Tool erlaubt es dem Entwickler die Kommunikation zwischen einzelnen Komponenten mit unterschiedlichem Detailgrad zu spezifizieren. Dies ermöglicht es, für verschiedene Teile des Kommunikationssystems den verwendeten Detailgrad anzupassen. Es muss allerdings zwingend die von Hines/Boriello entworfene Designsprache "Pia" verwendet werden um die notwendigen Schnittstellen und Komponenten auf den unterschiedlichen Abstraktionsebenen zu beschreiben. Bei diesem Ansatz lassen sich nur die Abstraktionsebenen der Hardwareschnittstellen ändern und nicht komplette Modelle. Ein ähnlicher Ansatz wird auch von Yoo/Jerraya vorgeschlagen ("Hardware/Software Cosimulation from Interface Perspective", IEE Proceedings, Computers and Digital Techniques, 152(3):369-379, 2005) wo zwischen verschiedenen Abstraktionsebenen (in derselben Implementierungssprache) für eine Prozessorsimulation umgeschaltet werden soll. Die Autoren geben allerdings keine konkrete Lösung dafür, sondern definieren das Problem lediglich als mögliches Forschungsgebiet. Andere Autoren beschäftigen sich mit dem Umschalten zwischen verschiedenen Simulationsmodellen implementiert auf Transaktionsebene (TLM) in derselben Hardwarebeschreibungssprache. Radetzki et al. ("Accuracy-Adaptive Simulation of Transaction Level Models", Design, Automation and Test in Europe, 2008. DATE '08, pages 788-791, 2008.) zeigen eine SystemC basierte Lösung zum Umschalten zwischen verschiedenen SystemC-TLM Modellen zur Laufzeit. In Abhängigkeit vom momentanen Modellzustand wird ein Simulationsmodell gewählt. Ein ähnlicher Ansatz wird von Beltrame et al. ("Multi-Accuracy Power and Performance Transaction-Level Modeling. Computer-Aided Design of Integrated Circuits and Systems", IEEE Transactions on, 26(10):1830-1842, 2007.) verfolgt. Hier wird SystemC-TLM erweitert um SystemC-TLM Modelle verschiedener Genauigkeit zur Laufzeit verwenden zu können.

[0007] Das Patent DE 69631278 T2 beschreibt eine allgemeine Co-Simulationsmethodik. Es beinhaltet keinerlei Umschalten zwischen verschiedenen Simulationsmodellen zur Laufzeit. Im Patent US 7,191,111 B2 wird ein Weg beschrieben, einen Simulator zur Laufzeit zu deaktivieren. In der Zwischenzeit wird allerdings kein anderer Simulator für die jeweilige Komponente verwendet, sondern sie ist komplett deaktiviert. Das Patent US 7,146,300 beschreibt eine allgemeine Co-Simulationsmethodik und beinhaltet keinerlei Umschalten zwischen verschiedenen Simulationsmodellen zur Laufzeit. Das Patent US 7,069,204 beschreibt eine Methodik in Hinsicht auf Performance Hardware und Software zu modellieren. Es beinhaltet keinerlei Umschalten zwischen verschiedenen Simulationsmodellen zur Laufzeit. Das Patent US 5,870,588 umfasst eine allgemeine Co-Simulationsmethodik. Es beinhaltet keinerlei Umschalten zwischen verschiedenen Simulationsmodellen zur Laufzeit. In der internationalen Patentanmeldung WO 2007/025491 A1 wird eine Möglichkeit beschrieben, die Synchronisation zwischen verschiedenen Modellen einer Co-Simulation zu optimieren. Hier werden allerdings keine unterschiedlichen Simulationsmodelle für dieselbe Komponente verwendet, und es kann daher auch nicht zur Laufzeit das Simulationsmodell gewechselt werden. Die Patentanmeldung US 2007/0192079 A1 beschreibt eine zweistufige CPU Simulation. Jede Instruktion wird zuerst offline mit hoher Genauigkeit simuliert und das Ergebnis gespeichert. Während der gesamten Simulation wird anschließend ein abstraktes CPU-Verhalten modelliert, das auf die vorher berechneten Ergebnisse zurückgreift mit einer Art von Lookup-Table: effektiv gibt es also nur ein Simulationsmodell für die Komponente und das wird im Vorhinein vollständig "offline" berechnet. Dies ist allerdings nur möglich, weil es eine sehr begrenzte Anzahl verschiedener Instruktionen gibt. Es unterscheidet sich also wesentlich von der vorliegenden Erfindung.

[0008] Die Patentanmeldung US 2009/0063120 A1 beschreibt eine allgemeine Co-Simulationsmethodik bzw. Emulationsmethodik für die Kopplung von echter Hardware und Software. Dabei wird beschrieben, wie die Interaktion zwischen Hardware und Software durchgeführt wird. Weiters wird beschrieben, wie die Hardware zur Unterstützung der Simulation genutzt wird. Es beinhaltet keinerlei Umschalten zwischen verschiedenen Simulationsmodellen zur Laufzeit.

[0009] Das Dokument "SystemC-VHDL co-simulation and synthesis in the HW domain" (Bombana, M. et al.) beschreibt eine allgemeine Co-Simulationsmethodik zur

Co-Simulation zwischen SystemC und VHDL. Es beinhaltet keinerlei Umschalten zwischen verschiedenen Simulationsmodellen zur Laufzeit.

[0010] Die Patentanmeldung EP 0 772 140 A1 beschreibt eine allgemeine HW/SW Co-Design Methodik unter Verwendung von Co-Simulation. Dabei wird ein Entwurfsprozess für ein System unter Verwendung von HW/SW Co-Design und Co-Simulation beschrieben. Es beinhaltet keinerlei Umschalten zwischen verschiedenen Simulationsmodellen zur Laufzeit.

[0011] Die vorliegende Erfindung überwindet die geschilderten Nachteile des Standes der Technik durch Bereitstellen eines Verfahrens zur Simulation von Komponenten unter Nutzung von Co-Simulation mit den Merkmalen des Anspruchs 1. Vorteilhafte Ausgestaltungen der Erfindung sind in den Unteransprüchen dargelegt.

[0012] Der Hauptvorteil dieser Erfindung ist die Möglichkeit, die Simulation in Intervalle hoher Genauigkeit und Intervalle mit niedriger Genauigkeit einzuteilen. Während Intervallen mit hoher Genauigkeit kommen rechenintensive und hochdetaillierte Simulationsmodelle zum Einsatz, während in Intervallen niedriger Genauigkeit einfachere, aber dafür schnellere Modelle verwendet werden. Die Performance der Co-Simulation wird durch die Möglichkeit des Umschaltens zur Laufzeit zwischen verschiedenen Simulationsmodellen für dieselbe Komponente enorm verbessert. Dadurch wird für komplexe Systeme überhaupt erst die Simulation eines ausreichend großen simulierten Zeitraumes mit vertretbarer Simulationszeit möglich. In dieser Erfindung wird daher vorgeschlagen, dynamisch zur Laufzeit der Co-Simulation zwischen verschiedenen Simulationsmodellen und Abstraktionsebenen für eine Komponente umzuschalten. Durch die dadurch wesentlich verkürzte Simulationszeit wird der Entwickler in die Lage versetzt, das gesamte System über eine lange simulierte Zeit zu beobachten und gleichzeitig aber einen Zeitraum zu definieren, in dem die erzielte Simulationsgenauigkeit massiv erhöht wird. Das System kann also mit einem schnellen und ungenauen Simulationsmodell unter normalen Umständen simuliert werden, während zum Zeitpunkt des Auftretens besonderer Effekte (z.B. hinsichtlich elektromagnetischer Verträglichkeit) auf ein hochdetailliertes Simulationsmodell der Komponente umgeschaltet wird, um eine bessere Genauigkeit der Ergebnisse zu erzielen.

[0013] Ein weiterer Punkt der Erfindung umfasst das automatisierte Umschalten zwischen verschiedenen Simulationsmodellen für dieselbe Komponente. Durch Vorgabe bestimmter, einstellbarer Kriterien wie Verwendungsgrad des Komponentenmodells (d.h. wird eine Komponente derzeit überhaupt angewendet) oder vorausberechnete Abweichung der Genauigkeiten der Modelle für eine gegebene Komponente wird automatisiert zwischen den verschiedenen Simulationsmodellen zur Laufzeit gewechselt. Wird also beispielsweise anhand der Parameter erkannt, dass nun eine besonders kritische Phase erreicht wird, wird automatisch auf das dafür geeignete hochdetaillierte Simulationsmodell umgeschaltet. Ein Verlassen des kritischen Bereichs wird ebenso erkannt und damit automatisiert wieder auf ein schnelleres, aber ungenaueres Simulationsmodell gewechselt, um so die Gesamtsimulationszeit so kurz wie möglich zu halten, aber gleichzeitig in besonders interessanten Simulationsintervallen eine möglichst hohe Simulationsgenauigkeit zu erzielen.

[0014] Eine detaillierte Beschreibung der Erfindung wird beispielhaft anhand von Zeichnungsfiguren durchgeführt. In diesen Figuren zeigen:

Fig. 1 eine Darstellung des erfindungsgemäßen Prinzips des Umschaltens zwischen verschiedenen Simulationsmodellen für eine Komponente zur Laufzeit in einer heterogenen Co-Simulationsumgebung;

Fig. 2 die interne und externe Ansicht auf die Komponente gemäß Fig. 1 mit verschiedenen Simulationsmodellen, zwischen denen zur Laufzeit umgeschaltet wird;

Fig. 3 ein Zeitdiagramm, das den Zusammenhang für die Komponente mit verschiedenen Simulationsmodellen zwischen Abstraktionsebene, Simulationsgeschwindigkeit und erzielbarer Simulationsgenauigkeit nach außen hin darstellt; und

Fig. 4 ein Beispiel für eine Co-Simulation eines FlexRay Netzwerks unter Anwendung der vorliegenden Erfindung.

[0015] In Fig. 1 ist beispielhaft das erfindungsgemäße Prinzip des Umschaltens zwischen verschiedenen Simulationsmodellen (sowohl Hardware, Software, oder gemische Ausführungsform) für eine Komponente A (z.B. physikalische Ebene eines Netzwerks, wie in Fig. 4 dargestellt) zur Laufzeit in einer heterogenen Co-Simulationsumgebung dargestellt. Die drei Blöcke in der Mitte repräsentieren dabei die verschiedenen Simulationsmodelle A1 bis A3 für die Komponente A, zwischen denen zur Laufzeit umgeschaltet werden kann. Die Simulationsmodelle A1 bis A3 repräsentieren unterschiedliche Simulationsebenen. Die Komponenten B und D stehen dabei auszugsweise für die restlichen Komponenten (z.B. Softwareapplikation und Umwelt, wie in Fig. 4 verdeutlicht) des zu simulierenden Systems. Die beiden Blöcke S1 und S2 sind dafür zuständig, dass für die Komponente A jeweils auf das richtige Simulationsmodell A1 - A3 umgeschaltet wird und dienen in weiterer Folge als Schnittstelle zur restlichen Co-Simulation (Funktionsprinzip eines Multiplexers bzw. DeMultiplexers). Darüber hinaus sind sie für die Synchronisierung der verwendeten Simulationsmodelle A1-A3 zuständig (rechtzeitige Aktivierung eines Simulationsmodells einer Komponente vor der Umschaltung auf ein anderes Simulationsmodell zur Synchronisierung des internen Modellzustandes). Sie führen auch die automatisierte Auswahl der jeweils geeigneten Modellabstraktionsstufe, basierend auf defi-

nierbaren Kriterien Pi, wie Verwendungsgrad des Komponentenmodells (d.h. wird eine Komponente derzeit überhaupt angewendet) oder vorausberechnete Abweichung der Genauigkeiten der Modelle für eine gegebene Komponente durch. Damit stellen sie einen wesentlichen Punkt der Erfindung dar.

[0016] Die Simulationsmodelle einer gegebenen Komponente A1 bis A3 können unterschiedliche Komplexität aufweisen und sind unabhängig von der Verhaltensmodellierungssprache. Im Fall von unterschiedlicher Verhaltensmodellierungssprache wird das Prinzip der Co-Simulation angewendet (d.h. Einbindung von mehreren Simulationssprachen in ein Simulationsframework für die gesamtheitliche Simulation eines Systems mit heterogenen Komponenten).

[0017] Fig. 2 zeigt die interne und externe Ansicht auf die Komponente A, die die verschiedenen Simulationsmodelle A1, A2, A3 umfasst, zwischen denen zur Laufzeit umgeschaltet wird. Der obere Teil von Fig. 2 zeigt die internen Abläufe während der Simulation, die das Umschalten betreffen. Es wird gezeigt, wie zwischen den verschiedenen Simulationsebenen (Hoch, Mittel, Tief), in den Figuren als Abstraktionsebenen bezeichnet, gewechselt wird. Dabei ist jeweils eine kurze Phase nötig, in der das als nächstes zu verwendende Simulationsmodell A1, A2, A3 parallel zum momentan laufenden Simulationsmodell A1, A2, A3 ausgeführt wird, bevor ein Umschaltvorgang zwischen den beiden Simulationsmodellen stattfindet. Dies ist notwendig, um die beiden Modelle miteinander zu synchronisieren. Eine andere Synchronisierungsmöglichkeit ist, dass dem nachfolgenden Simulationsmodell der für ein nahtloses Umschalten notwendige interne Zustand von extern eingeprägt wird, bevor das Umschalten zwischen den Simulationsmodellen stattfindet. Dies muss aber durch das Simulationsmodell unterstützt werden. Der untere Teil von Fig. 2 zeigt die externe Ansicht auf die Komponente A, wenn als Synchronisierungsvariante Parallelsimulation eingesetzt wird. Dabei ist ersichtlich, dass der Übergang zwischen den verschiedenen Simulationsmodellen von außen betrachtet nahtlos geschieht, die Parallelberechnung also außerhalb der Komponente A nicht erkennbar ist.

[0018] Fig. 3 stellt ein Zeitdiagramm dar, in dem für die verschiedenen Simulationsmodelle A1, A2, A3 der Komponente A der Zusammenhang zwischen Abstraktionsebene, Simulationsgeschwindigkeit und erzielbarer Simulationsgenauigkeit nach außen hin dargestellt ist. Dabei ist ersichtlich, dass eine niedrige (tiefe) Abstraktionsebene der Komponente A (d.h. es läuft das Simulationsmodell A3 der Komponente A) eine geringe Simulationsgeschwindigkeit und hohe Simulationsgenauigkeit der Komponente bewirkt. Dasselbe gilt auch umgekehrt: eine hohe Abstraktionsebene (d.h. es läuft das Simulationsmodell A1 der Komponente A) führt zu hoher Simulationsgeschwindigkeit, aber niedriger Simulationsgenauigkeit der Komponente. Weiter ist zu sehen, dass durch die Phase der Parallelberechnung (siehe Figur 2) die Simulationsgeschwindigkeit während der Parallelberechnung absinkt.

[0019] Fig. 4 zeigt ein Beispiel für eine Co-Simulation eines FlexRay Netzwerks (Komponente: Netzwerk) unter Einsatz der Erfindung. Dabei wird das Modell M2 für die Komponenten Netzwerktopologie (VHDL-AMS Topologiemodell, inklusive Kabel) und Transceiver (Transceiver VHDL-AMS) je nach Bedarf zur Laufzeit zwischen hochdetaillierten, aber langsamen VHDL-AMS Simulationsmodellen und einem schnellen, aber ungenauen SystemC Simulationsmodell M1 (Einfache Topologieeffekte (SystemC)) umgeschaltet.

[0020] Die FlexRay Testerknoten K1 bis K4 in SystemC liefern dabei die Stimuli für das Netzwerk und analysieren das Ausgabeverhalten des Netzwerkes. Die Schalter S1 bis S4 sind für ein 2-zu-1 Multiplexing zwischen Netzwerk und entsprechende FlexRay Testerknoten K1 bis K4 zuständig und gewährleisten zusätzlich die Synchronisierung der verwendeten Simulationsmodelle M1 und M2 während des Umschaltprozesses.

Patentansprüche

1. Verfahren zur Simulation von Komponenten unter Nutzung von Co-Simulation, **dadurch gekennzeichnet, dass** zumindest eine Komponente (A) in unterschiedlichen Simulationsebenen, die unterschiedliche Detailtiefen aufweisen, modelliert wird, die Modellierung der zumindest einen Komponente (A) durch zumindest zwei verschiedene Verhaltensmodellierungssprachen erfolgt, und die Umschaltung der Simulationsebenen für zumindest eine Komponente (A) zur Laufzeit der Simulation erfolgt.

2. Verfahren nach Anspruch 1, **dadurch gekennzeichnet, dass** eine Zuordnung der zu verwendenden Simulationsebene für zumindest eine Komponente (A) für einen vorgegebenen Zeitabschnitt der zu simulierenden Zeit vom Benutzer definiert wird.

3. Verfahren nach Anspruch 1, **dadurch gekennzeichnet, dass** eine Umschaltung der zu verwendenden Simulationsebene zumindest einer Komponente (A) ereignisgesteuert durch definierte Kriterien, wie z.B. ob eine Komponente derzeit überhaupt angewendet wird bzw. der Verwendungsgrad des Komponentenmodells, oder vorausberechnete Abweichung der Genauigkeiten der Modelle für eine gegebene Komponente, erfolgt.

4. Verfahren nach Anspruch 3, **dadurch gekennzeichnet, dass** der Wechsel zwischen den Simulationsebenen mit Hilfe während der Laufzeit automatisiert ausgewerteter Kriterien erfolgt.

5. Verfahren nach einem der vorangegangenen Ansprüche, **dadurch gekennzeichnet, dass** beim Wechsel zwischen verschiedenen Simulationsebe-

nen vorübergehend das als nächstes zu verwendende Simulationsmodell (A1, A2, A3) parallel zum momentan laufenden Simulationsmodell (A1, A2, A3) betrieben wird.

6. Verfahren nach einem der Ansprüche 1 bis 4, **dadurch gekennzeichnet, dass** beim Wechsel zwischen verschiedenen Simulationsebenen dem nachfolgenden Simulationsmodell (A1, A2, A3) der für ein nahtloses Umschalten notwendige interne Zustand von extern eingeprägt wird, bevor das Umschalten zwischen den Simulationsmodellen stattfindet.

EP 2 299 376 A1

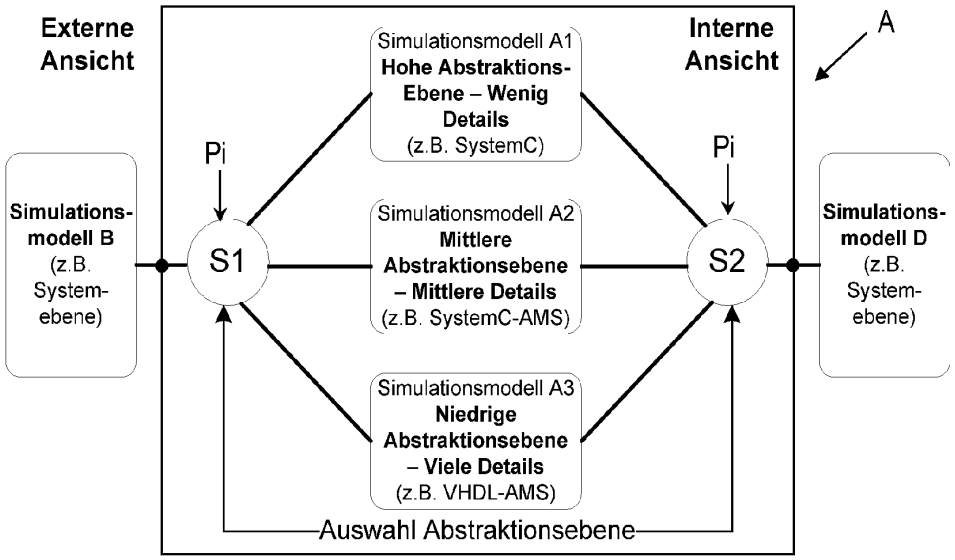


Fig. 1

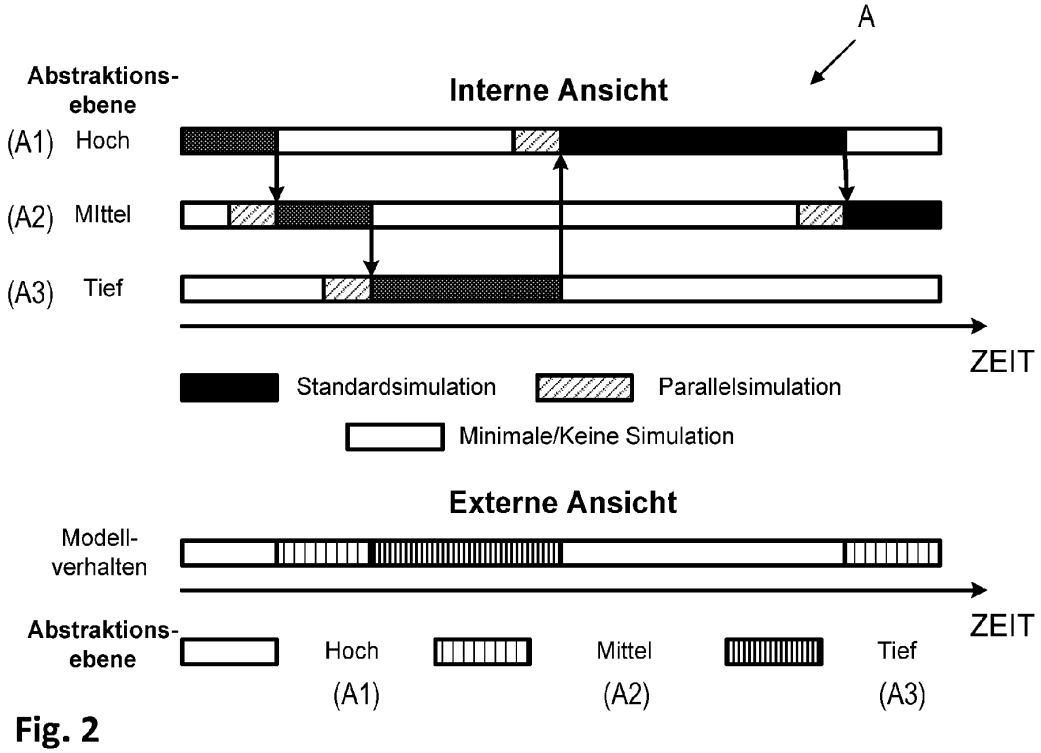


Fig. 2

EP 2 299 376 A1

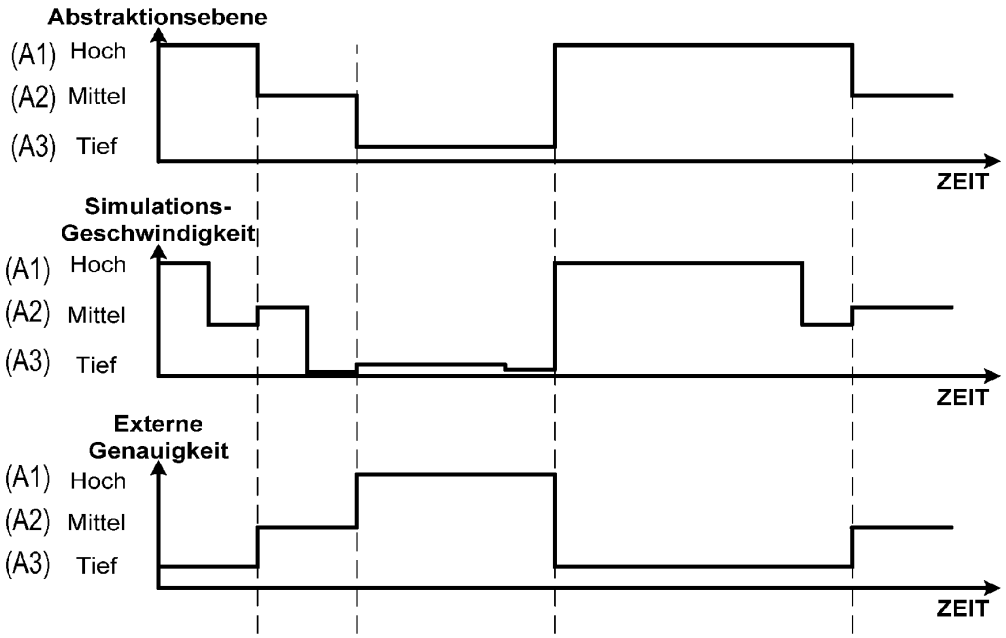


Fig. 3

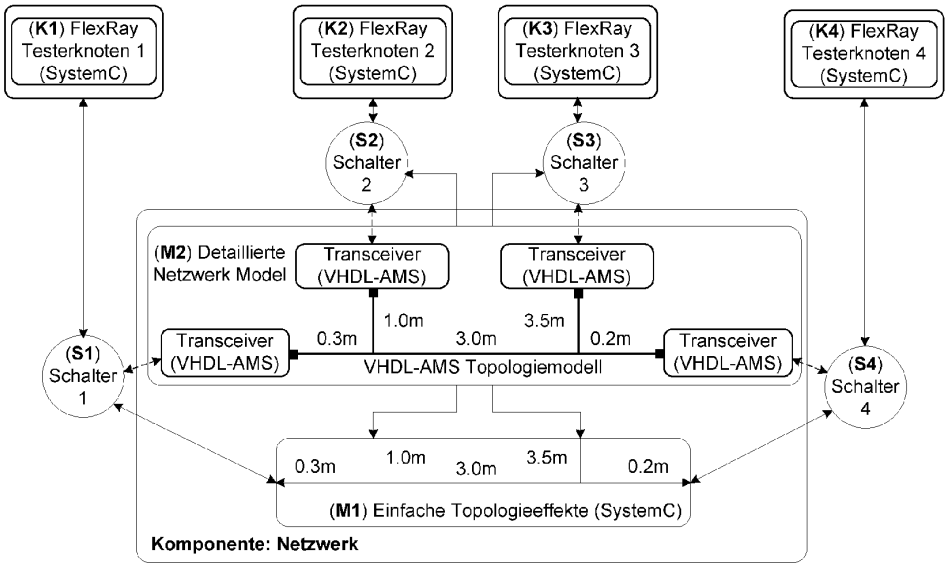


Fig. 4

EP 2 299 376 A1

Europäisches Patentamt
European Patent Office
Office européen des brevets

EUROPÄISCHER RECHERCHENBERICHT

Nummer der Anmeldung
EP 10 17 7698

EINSCHLÄGIGE DOKUMENTE

Kategorie	Kennzeichnung des Dokuments mit Angabe, soweit erforderlich, der maßgeblichen Teile	Betrifft Anspruch	KLASSIFIKATION DER ANMELDUNG (IPC)
X,P	MICHAEL KARNER ET AL: "Optimizing HW/SW Co-simulation based on run-time model switching" SPECIFICATION&DESIGN LANGUAGES, 2009. FDL 2009. FORUM ON, IEEE, PISCATAWAY, NJ, USA, 22. September 2009 (2009-09-22), Seiten 1-6, XP031622942 ISBN: 978-2-9530504-1-7 * das ganze Dokument *	1-6	INV. G06F17/50
X,D	BELTRAME G ET AL: "Multi-Accuracy Power and Performance Transaction-Level Modeling" IEEE TRANSACTIONS ON COMPUTER AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, Bd. 26, Nr. 10, 1. Oktober 2007 (2007-10-01), Seiten 1830-1842, XP011192426 IEEE SERVICE CENTER, PISCATAWAY, NJ, US ISSN: 0278-0070 DOI: 10.1109/TCAD.2007.895790 * Seite 1830 * * Seite 1832 - Seite 1835 * * Seite 1838 *	1-6	RECHERCHIERTE SACHGEBIETE (IPC) G06F
A	SARMENTO A ET AL: "Automatic building of executable models from abstract soc architectures made of heterogeneous subsystems" 15TH IEEE INTERNATIONAL W ORKSHOP ON RAPID SYSTEM PROTOTYPING, 2004. GENEVA, SWITZERLAND, 28. Juni 2004 (2004-06-28), - 30. Juni 2004 (2004-06-30) Seiten 88-95, XP010708391 PISCATAWAY, NJ, USA,IEEE DOI: 10.1109/IWRSP.2004.1311101 ISBN: 978-0-7695-2159-6 * das ganze Dokument *	1-6	

-/--

Der vorliegende Recherchenbericht wurde für alle Patentansprüche erstellt

Recherchenort	Abschlußdatum der Recherche	Prüfer
München	2. Februar 2011	Alonso Nogueiro, L

KATEGORIE DER GENANNTEN DOKUMENTE

X : von besonderer Bedeutung allein betrachtet
Y : von besonderer Bedeutung in Verbindung mit einer anderen Veröffentlichung derselben Kategorie
A : technologischer Hintergrund
O : nichtschriftliche Offenbarung
P : Zwischenliteratur

T : der Erfindung zugrunde liegende Theorien oder Grundsätze
E : älteres Patentdokument, das jedoch erst am oder nach dem Anmeldedatum veröffentlicht worden ist
D : in der Anmeldung angeführtes Dokument
L : aus anderen Gründen angeführtes Dokument

& : Mitglied der gleichen Patentfamilie, übereinstimmendes Dokument

4

EPO FORM 1503 03.82 (P04C03)

EP 2 299 376 A1

Europäisches Patentamt
European Patent Office
Office européen des brevets

EUROPÄISCHER RECHERCHENBERICHT

Nummer der Anmeldung
EP 10 17 7698

EINSCHLÄGIGE DOKUMENTE

Kategorie	Kennzeichnung des Dokuments mit Angabe, soweit erforderlich, der maßgeblichen Teile	Betrifft Anspruch	KLASSIFIKATION DER ANMELDUNG (IPC)
A	MATHIEU DUBOIS ET AL: "Acceleration for a compiled Transaction Level Modeling simulation" 13TH IEEE INTERNAT IONAL CONFERENCE ON ELECTRONICS, CIRCUITS AND SYSTEMS, 2006. ICECS '06. IEEE, PI, 1. Dezember 2006 (2006-12-01), Seiten 1176-1179, XP031111705 ISBN: 978-1-4244-0394-3 * das ganze Dokument * -----	1-6	
X	BIRRER P ET AL: "Incorporating SystemC in Analog/Mixed-Signal Design Flow"[Online] 30. September 2005 (2005-09-30), XP002619170 LAUSANNE, SWITZERLAND FORUM ON SPECIFICATION AND DESIGN LANGUAGES, FDL 2005, PROCEEDINGS. ECSI 2005 Gefunden im Internet: URL:http://lsmwww.epfl.ch/FDL05/AMS/AMS25_final.pdf> [gefunden am 2011-02-02] * das ganze Dokument * -----	1-6	RECHERCHIERTE SACHGEBIETE (IPC)

Der vorliegende Recherchenbericht wurde für alle Patentansprüche erstellt

Recherchenort	Abschlußdatum der Recherche	Prüfer
München	2. Februar 2011	Alonso Nogueiro, L

KATEGORIE DER GENANNTEN DOKUMENTE

X : von besonderer Bedeutung allein betrachtet
Y : von besonderer Bedeutung in Verbindung mit einer anderen Veröffentlichung derselben Kategorie
A : technologischer Hintergrund
O : nichtschriftliche Offenbarung
P : Zwischenliteratur

T : der Erfindung zugrunde liegende Theorien oder Grundsätze
E : älteres Patentdokument, das jedoch erst am oder nach dem Anmeldedatum veröffentlicht worden ist
D : in der Anmeldung angeführtes Dokument
L : aus anderen Gründen angeführtes Dokument

& : Mitglied der gleichen Patentfamilie, übereinstimmendes Dokument

EPO FORM 1503 03.82 (P04C03)

EP 2 299 376 A1

IN DER BESCHREIBUNG AUFGEFÜHRTE DOKUMENTE

Diese Liste der vom Anmelder aufgeführten Dokumente wurde ausschließlich zur Information des Lesers aufgenommen und ist nicht Bestandteil des europäischen Patentdokumentes. Sie wurde mit größter Sorgfalt zusammengestellt; das EPA übernimmt jedoch keinerlei Haftung für etwaige Fehler oder Auslassungen.

In der Beschreibung aufgeführte Patentdokumente

- DE 69631278 T2 **[0007]**
- US 7191111 B2 **[0007]**
- US 7146300 B **[0007]**
- US 7069204 B **[0007]**
- US 5870588 A **[0007]**
- WO 2007025491 A1 **[0007]**
- US 20070192079 A1 **[0007]**
- US 20090063120 A1 **[0008]**
- EP 0772140 A1 **[0010]**

In der Beschreibung aufgeführte Nicht-Patentliteratur

- **Bouchhima et al.** A SystemC/Simulink Co-Simulation Framework for Continuous/Discrete-Events Simulation. *Behavioral Modeling and Simulation Workshop, Proceedings of the 2006 IEEE International,* 2006, 1-6 **[0005]**
- **Birrer et al.** Incorporating SystemC in Analog/Mixed-Signal Design Flow. In Forum on Specification and Design Languages. *Proceedings of the 8th International,* 2005, 173-178 **[0005]**
- Selective Focus as a Means of Improving Geographically Distributed Embedded System Co-Simulation. *Proceedings of the 8th IEEE International Workshop on Rapid System Prototyping,* 1997, 58-62 **[0006]**
- Dynamic Communication Models in Embedded System Co-Simulation. *Proceedings of the 34th Design Automation Conference,* 1997, 395-400 **[0006]**
- Hardware/Software Cosimulation from Interface Perspective. *IEE Proceedings, Computers and Digital Techniques,* 2005, vol. 152 (3), 369-379 **[0006]**
- **Radetzki et al.** Accuracy-Adaptive Simulation of Transaction Level Models. *Design, Automation and Test in Europe, 2008. DATE '08,* 2008, 788-791 **[0006]**
- **Beltrame et al.** Multi-Accuracy Power and Performance Transaction-Level Modeling. Computer-Aided Design of Integrated Circuits and Systems. *IEEE Transactions on,* 2007, vol. 26 (10), 1830-1842 **[0006]**

2010-01-0239

A Cross Domain Co-Simulation Platform for the Efficient Analysis of Mechatronic Systems

Michael Karner, Christian Steger, Reinhold Weiß
Graz University of Technology - Institute for Technical Informatics

Eric Armengaud
Virtual Vehicle Competence Center

Markus Pistauer
CISC Semiconductor Design + Consulting GmbH

Felix Pfister
AVL List GmbH

ABSTRACT

Efficient integration of mechanics and microelectronics components is nowadays a must within the automotive industry in order to minimize integration risks and support optimization of the entire system. We propose in this work a cross domain co-simulation platform for the efficient analysis of mechatronic systems. The interfacing of two state-of-the-art simulation platforms provides a direct link between the two domains at an early development stage, thus enabling the validation and optimization of the system already during modeling phase. The proposed cross-domain co-simulation is used within our TEODACS project for the analysis of the FlexRay technology. We illustrate using a drive-by-wire use case how the different architecture choices may influence the system.

INTRODUCTION

The current world market for automotive embedded electronics represents about 3 billion units delivered per year for a budget encompassing approximately 160 billion Euros with an annual growth of about 9 percent [1]. Already representing an important part of the car budget, this is still expected to grow fast. Hence, experts estimate that more than 80% from the innovation potential in car comes from electronics [2]. This concerns, for example, the introduction of new functionalities (e.g. radars and distance sensors for active safety) as well as the enhancement of mechanics solutions by electronics components (e.g. engine management or car dynamics – ESP), and also the replacement of mechanics components by electronics counterparts (e.g. drive-by-wire).

Contrary to other products such as consumer electronics that are focused to a single domain, an automotive system combines electronics and mechanics components. In this context, new methods are required to support the efficient co-design and co-integration of multidisciplinary systems. Until now, the development of mechanics and microelectronics used to be clearly separated (different design flows, different teams) and the integration work was performed at the end of the development process. This late integration presents two main problems: First the risks of expensive re-design due to incompatible interfaces (e.g. side effects due to the

complex interactions between the two domains), and second the difficulties to perform system wide optimization.

We present in this work a cross-domain simulation platform which interfaces CISC SyAD® [3] and CarMaker/AVL InMotion [5] tools. The System Architect Designer SyAD®, on one hand, provides a co-simulation framework for the system design and simulation of highly heterogeneous microelectronic systems. On the other hand, CarMaker/AVL InMotion targets the simulation, test and calibration of hybrid powertrain systems. The proposed co-simulation framework supports the early integration of both mechanics and microelectronics components, thus (a) minimizing the integration risks and (b) supporting the efficient cross-domain system optimization. Hence, the different models can directly interact within the co-simulation environment, thus providing realistic data from a domain to another. These attributes largely support system integration and optimization.

This cross-domain simulation platform is used within our TEODACS project (Test, Evaluation and Optimization of Dependable Automotive Communication Systems, see www.teodacs.com [6, 7, 8]). The focus of this research project is set to the analysis of the communication architecture and the influence of this component to the distributed system. To that aim, we propose a co-simulation platform tightly interfaced to a realistic prototype for the in-depth analysis of FlexRay networks. CarMaker/AVL InMotion simulates a powertrain system (road, driver, control loops for vehicle dynamics) and generates a realistic bus traffic for the network. SyAD® provides a co-simulation framework for the seamless modeling and integration of the different microelectronics components (e.g. cable model, bus drivers, middleware, software functions). Based on a Drive-by-Wire use case, we evaluate the benefits of the proposed cross-domain co-simulation approach for the early analysis and verification of the assembled system.

SIMULATION OF MECHATRONICS SYSTEMS

Mechatronics means the combination of the worlds of mechanics and (micro-)electronics. In the real world, this is a combination with often very diverse requirements. However, when talking about simulation of mechatronic systems the contrast gets even more obvious. In this section we will analyze the characteristics of the simulation of microelectronics, mechanics and mechatronics.

MICROELECTRONICS SIMULATION

When talking about the simulation of microelectronic systems it has to be distinguished between at least three different types of simulations: (1) simulation of analogue components or often called circuit simulation, (2) simulation of digital components and (3) the simulation of the software components that are running on a certain microelectronic system. Every type of microelectronic simulation has its special requirements, advantages and disadvantages which will now be looked at in more detail.

The analogue part of microelectronic systems is typically simulated by using circuit simulation. A circuit simulation is built by using several different components like capacitors, coils, resistors or diodes which are connected as a circuit, see [9] for more details. In practice, usually an enormous number of components are building the circuit simulation. For example, in complex microelectronic systems the number of components for the circuit simulation model can easily reach the area of several million components. Typical for circuit simulation is that the observed variables are in continuous form, for example currents and voltages. Due to the structure of the system and the components used the simulator has to solve differential-algebraic equation (DAE) systems for the simulation to take place. Regarding time representation, it is obvious that the time has to be accounted as continuous as analogue systems with variables in continuous form are under observation. Typical time ranges in circuit simulation are in the area of microseconds to milliseconds, in complex systems

only this short time interval can be simulated within a reasonable amount of time. Because of the high computation requirements in circuit simulations, the achievable simulation performance typically is very low. Examples for hardware description languages (HDL) supporting the simulation of analogue microelectronic systems are e.g. MAST [15], VHDL-AMS (Very High Speed Integrated Circuit Hardware Description Language – Analog and Mixed Signal) [10] or SystemC-AMS [18].

The simulation of the digital part of microelectronic systems (like e.g. for complex state machines) is another basic type of microelectronic simulation. Here, instead of continuous values like currents or voltages logic values are used. The simulation of digital microelectronics is described in detail in [9]. By making the shift from analogue values to digital values there are already lots of details abstracted and lost. Hence, when developing simulation models for digital components it has to be assured that at the analogue level everything is working as expected. In digital simulation, in difference to circuit simulation, time is seen as discrete or event-oriented. Hence, if a value changes, this means a transition of the according signal with a specific time delay. In discrete time simulation there are specified time stamps where the signals and values are updated. These simulation time steps are typically in the order of picoseconds or nanoseconds. However, in event oriented simulation the signal values are only updated if a value changes. This can be seen as kind of chain reaction. Typical time ranges for the simulation of digital parts of microelectronic systems are in the area of microseconds to milliseconds. Hardware description languages supporting digital modeling are for example VHDL or Verilog.

Another important part in microelectronic simulation is the simulation of the software components that are running on a certain microelectronic system. Here, very complex program behavior of software running on an also very complex CPU architecture has to be evaluated. For this, often so called cycle-accurate instruction set simulators (ISS) are used that allow the timing of the software to be determined in terms of processor cycles. This allows for the optimization of the software for different platforms as both the behavior of the software and the infrastructure on which the program is run is taken into account. The typical simulation focus for software is in the area of milliseconds up to several seconds or even minutes. To achieve the required flexibility for the exchange of the hardware platforms there is a strong need for standardized software architectures (like AUTOSAR in the automotive area). In simulation of software typically development languages like C/C++ or SystemC (www.systemc.org) are used. In SystemC, it is possible to model both worlds of software and hardware by using one common modeling language.

MECHANICS SIMULATION

Following microelectronic systems, the other part in mechatronics simulation is the simulation of mechanics systems. Here, opposite to microelectronics, several different reference systems have to be taken into account which complicates the task of simulation. However, typically only up to some hundred components are involved in the simulation of mechanics systems, which is way below the number of components in microelectronic simulation. Like in microelectronics, there are several different types of simulations of mechanics components. In [9], it is distinguished between (1) multi-body simulation, (2) block diagram simulation and (3) finite element simulation. Typical modeling languages or simulators for mechanics are for example VHDL-AMS [10, 16, 9], MAST or Simulink. For the simulation of the mechanics of cars tools like e.g. CarMaker/AVL InMotion [5] are used. In mechanics simulation, a typical simulation step size is in the order of milliseconds and the time interval under observation can easily reach the area of several seconds or even minutes.

The first type of mechanical simulation is multi-body simulation. According to [9], in multi-body simulation the elements considered are bodies with mass and inertia moments, joints, springs, dampers, actuators, sensors and more. Basically, it can be distinguished between two different types of mechanical simulators supporting multi-

body simulation. In the first type, the mechanics are represented as symbol equations systems that can be handled by applying typical numerical standard solution procedures. In the second type of multi-body simulation, the simulators are interpreting the mechanics as a system of several matrices for e.g. mass, stiffness and damping. These matrices are afterwards processed. An example for a modeling language allowing multi-body simulation is Modelica [17].

Another type of mechanical simulation is block diagram simulation. According to [9], a block diagram of a control flow is created that describes the structure of a system of mathematical equations in graphical form. To perform the simulation process, the simulation derives equations of the structuring of the blocks and the connections between them. A typical example for block diagram simulation is Simulink.

The third commonly used type of mechanical simulation is finite element simulation. Here, finite elements are used for the description of a mechanical continuum [9]. Mass, damping and elasticity matrices are characterizing each finite element and numerical techniques are used for solving the resulting equations.

MECHATRONICS SIMULATION

Bringing together both mechanics and microelectronics simulation results in mechatronics system simulation. In principle, here the different domains are simulated in one simulation; hence, data exchange is possible between the two worlds. The same modeling language is used to model both mechanics and microelectronics. Main focus here is the interaction between the domains. Following [9], the system behavior is considerable determined by the interaction between mechanics and microelectronics domain. However, a problem here is that, as already mentioned before, the time intervals under observation differ between mechanics and microelectronics domain. This has to be taken into account when modeling and simulating mechatronics. An example for a mechatronics simulation of a car, respectively its components, is shown in [11, 12, 13, 14]. Modeling languages typically used for mechatronics simulation are e.g. VHDL-AMS or MAST.

A special enhancement of mechatronics simulation is cross-domain co-simulation. Here, the principle of co-simulation by using different modeling languages and simulators in one common co-simulation [3] is applied to the simulation of mechatronic systems.

As stated before, in standard mechatronics simulation the same description language is used to model both worlds of microelectronics and mechanics, for example VHDL-AMS (see [16, 9, 10] for details) or MAST. This limits the simulation to the possibilities and features of one simulator. Also simulators and languages designed for only one of the domains cannot be used, even if they would produce more accurate results or support modeling techniques that would enhance the mechatronic simulation. For example, CarMaker/AVL InMotion is a good simulator for car dynamics and mechanics, but it lacks support for large parts of the microelectronic domain.

Cross-domain co-simulation is a possible solution to lift the boundaries between the different simulation and model domains. By putting together specialized simulators for both mechanics and microelectronics domain in one common co-simulation it is possible to combine the advantages of the different simulators and modeling languages. This massively improves the simulation and analysis procedure in terms of accuracy, observability, real-world behavior and much more. However, there are specific challenges that arise when dealing with cross-domain co-simulation, such as the integration of the different sub-models within one co-simulation system and especially of the correct configuration of the sub-models involved. Additionally, the data interface and the time synchronization between the simulators have to be handled. Another challenge is the difference in typical time constants of the different domains as they typically differ between mechanics and microelectronics in the order of several magnitudes.

SYAD - CO-SIMULATION PLATFORM FOR MICROELECTRONICS SYSTEMS

The CISC System Architect Designer (SyAD, [4]) is a tool for the design and verification of microelectronic systems. In the area of system design, it allows for an easy creation of heterogeneous multi-modeling-language designs covering several different microelectronic modeling languages like VHDL, VHDL-AMS, SystemC, Matlab/Simulink etc. Also homogenous simulations can be created. SyAD supports a modular and evolutional design approach, hence making different abstraction levels easy to handle.

One advantage of SyAD is the possibility to easily create co-simulations between models implemented in the different supported modeling languages [3]. For this, SyAD implements the design methodology shown in Figure 1.

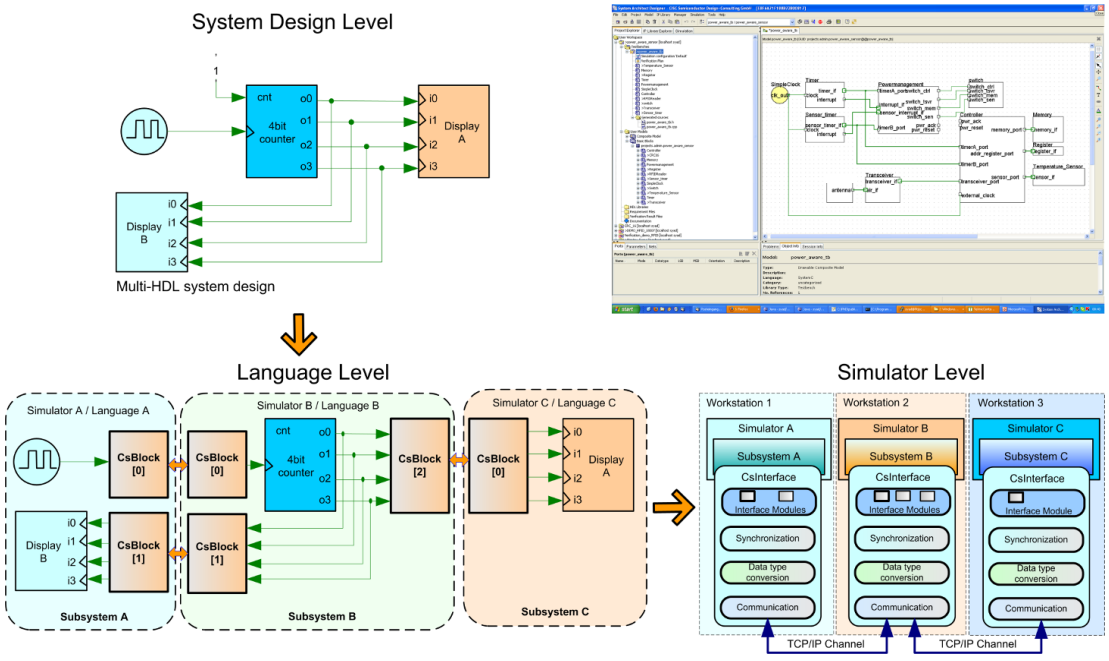


Figure 1: SyAD Design Methodology & Example

The developer creates the heterogeneous system model at system design level: the blocks are put into the workspace and connected according to the specification, representing simulation models of different components modeled by using the same or different modeling languages and abstraction levels. Nevertheless, a connection between them can be established by just connection them.

The following two levels are created by SyAD automatically to set up the co-simulation. In the language level, the system design is divided into groups of simulation models developed within the same language. At the language boundaries, so called co-simulation blocks are automatically inserted into the simulation. These co-simulation blocks represent the interfaces to the other modeling languages used in the co-simulation. The co-simulation blocks act as kind of translation blocks between the different modeling languages.

On the simulator level the connections between the different simulators involved in the co-simulation are established. Here, the co-simulation blocks of the language level are connected to co-simulation interfaces specific for each simulator. These co-simulation interfaces exchange data with their corresponding co-simulation blocks and are responsible for the necessary low-level tasks to establish a co-simulation between several different simulators, for example data exchange and simulator synchronization. In a co-simulation it is required that all the simulators are synchronized to guarantee simulation determinism. Hence, the slowest simulator determines the performance of the complete co-simulation. For synchronization, SyAD uses a decentralized approach where the co-simulation interfaces directly control the time continuation without any central control unit. More details about the technical background of the co-simulation environment can be found in [3].

An example co-simulation model developed using SyAD is also shown in Figure 1. It is a co-simulation of an automotive application. Both analog (like transceiver) and digital (micro-)electronic components (like controller, memory, timers etc.) are modeled within the co-simulation. The modeling languages used are SystemC and VHDL-AMS. With this co-simulation a realistic microelectronic environment can be simulated including effects and models of both analog and digital type. This allows for a system-wide optimization including consideration of effects that cross the boundary between digital and analog microelectronic world. The developer is able to optimize the assembled system much earlier than in standard development flows, thus saving time and money.

SyAD is a very suitable tool for the simulation and analysis of microelectronic systems and designs. In the context of automotive development, however, a link to a mechanics simulator is required for performing cross-domain effects analysis and system-wide optimization of e.g. an ECU controlling car dynamics.

CARMAKER / AVL INMOTION: MANEUVER AND EVENT-BASED TESTING

The automotive agenda strikes an intricate balance between requirements for fuel efficiency, emissions, driveability, reliability and cost-efficiency. CarMaker / AVL InMotion helps to solve this complex issue. This real-time simulation platform is a solution for maneuver and event based testing at the test bed. It supports key business objectives such as hybridization and electrification of powertrain engineering. The key of CarMaker / AVL InMotion is its high-performance virtual vehicle-driver-road-traffic-environment, into which the unit under test (engine, transmission, powertrain, etc.) and the control units are embedded. CarMaker / AVL InMotion enables to evaluate real-world driving behavior at the test bed.

CarMaker/AVL InMotion provides a truly integrated development process. Road-rig-lab-math simulation environments are integrated under one platform with identical GUIs. This is critically important to enable collaborative and close development among distributed teams all along the development process. The model manager integrates Cruise RT or legacy Simulink/Modelica models. Further, CarMaker/AVL InMotion provides an open and powerful real-time development environment, such that new components or interfaces can be efficiently integrated without large efforts. This simulation platform is targeted to maneuver based testing. It helps bringing new powertrain designs to market faster, manage increased regulatory pressures, and maintain cost efficiencies through versatile and reliable maneuver based testing. The simulation results are made available through a powerful online “vehicle-road-environment” visualization, see Figure 2. Typical applications for this simulation platform include:

- Hybrid testing: Electrification is changing automotive industry. CarMaker / AVL InMotion makes sure that three premises remain priority: quality, cost reduction and time-to-market.

- Real World Fuel Economy Testing: Reducing CO2 emissions is now the most important issue of OEMs and TIER1s. CarMaker / AVL InMotion brings the real world to the test bed. The concepts can now be evaluated and turned into tangible products.
- Virtual Proving Ground Testing: The test engineer has the capability of performing corporate test procedures in a reliable simulated environment in lieu of or prior to more expensive experimentation on the proving ground.
- ECU Software Quality Testing: The means of innovation and competitive differentiation is today mainly being created through software. OEMs and TIER1s are infusing more and more control functions into their products. Complexity increases dramatically. Testing with CarMaker / AVL InMotion is a key enabler for quality and efficiency.
- Powertrain Misuse Testing; Handbrake U-turns, bootleg turns, left-foot braking maneuvers, drifting, clutch-kicking, bumping and jumping. Key components to these tests are leading-edge real-time tire-road models, such as Michelin's TaMeTire.

Figure 2: Carmaker / AVL InMotion: Screenshots

This simulation platform enables the efficient analysis of the vehicle from a global and integrated point of view. A current limitation is the efficient co-analysis of microelectronics components required to implement the complex control functions. Hence, subtle effects coming from e.g. intelligent sensors, network technology or software architecture can strongly influence the system. A typical example for a brake-by-wire application based on a CAN network is the influence of end-to-end delays between the brake pedals and the actuators for the overall brake distance. Hence, depending on the current bus utilization and priority assignment, the brake message could be delayed thus leading to a less efficient system reaction. Considering the strong increase of microelectronics in cars, the cross-domain evaluation of the system is strongly required.

EFFICIENT CROSS DOMAIN COUPLING BETWEEN THE SIMULATION PLATFORMS

Cross-domain co-simulation means the co-simulation of component models and simulators implemented in different HDLs and coming from different domains, e.g. microelectronics and mechanics. There are lots of challenges that have to be solved to allow for an efficient cross-domain coupling. For example, one of the biggest problems is the different time bases involved. In the mechanics domain, the simulation step size is typically one millisecond and the time under observation is several seconds or even minutes. In the microelectronics domain we have time steps of down to picoseconds and the observed time interval is in the area of milliseconds. To overcome this problem we use the methodology described in [19]: run-time switched co-simulation. Here, the complexity of the co-simulation models used is changed during the co-simulation. This

allows getting the time base of the microelectronics world much closer to the time base of the mechanics world during normal simulation. Only for the simulation of relevant effects in the microelectronics world it is switched to a complex model for a short, defined amount of time.

A further problem is the difference between the types of models and data formats, and therefore the resulting strongly differing simulator structures between microelectronics and multi-body based mechanics simulators. Hence, an interface is required for the moderation between microelectronics and mechanics simulators in order to allow for a seamless data exchange and synchronization between them. In this section with present how the coupling between CarMaker / AVL InMotion (mechanics domain) and SyAD (microelectronics domain) has been performed based on the existing interfaces of both platforms.

In the case of the car simulator CarMaker / AVL InMotion, there already exists a network-based interface called APO (Application Online, [21]). By using APO, a developer can connect a C/C++ application with the simulator using UDP network communication. For this, CarMaker / AVL InMotion acts as APO server and the application has to implement an APO client. Afterwards it is possible for the client to receive simulation data over the network and send control commands and data to CarMaker. From the SyAD side, which already provides dedicated interfaces to different simulators, the complete development of a further dedicated interface targeted to CarMaker / AVL InMotion was required.

On the CarMaker / AVL InMotion side the simulation kernel was extended in order to support for pausing the simulation process via APO. The simulator now accepts a command via APO that defines how many of CarMaker / AVL InMotion internal time steps (each of 1ms) should be computed until waiting for the next control command and data via APO. This allows the synchronization of the CarMaker /AVL InMotion simulation with the microelectronics simulations of SyAD. On the SyAD side, we decided for a non-traditional way to couple the two platforms. SyAD is already able to integrate SystemC models into the co-simulation. Principally, a SystemC model is a C++ application with an additional library linked to support the modeling of hardware components. Hence, the idea is to implement a SystemC-based APO client that is responsible for both synchronization and data exchange between microelectronics and mechanics domains.

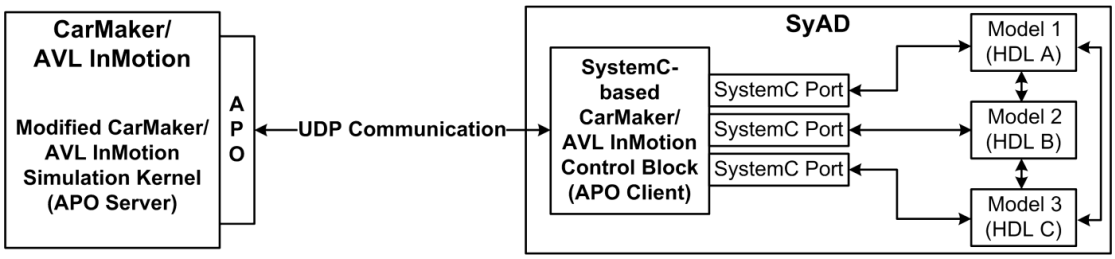


Figure 3: CarMaker/AVL InMotion - SyAD Interface Concept

This principle is shown in Figure 3. The SystemC-based CarMaker / AVL InMotion control block on the SyAD side implements an APO client. It sends the required commands to CarMaker / AVL InMotion for simulation parameterization and to keep mechanics and microelectronics in synchronization. For the data exchange, the SystemC-based control block converts the data between microelectronics and mechanics domain. It maps the mechanics data received via the APO interface to SystemC ports to which all components in the microelectronic domain simply can be connected, even if they are modeled in other languages than SystemC. On the other way, data management from the microelectronics to the mechanics domain is taken via SystemC ports from the control block and sent to CarMaker / AVL InMotion as mechanics data via the APO interface. Hence, both problems of data exchange and synchronization between microelectronics and mechanics domains are handled.

MECHATRONICS DEVELOPMENT WITHIN THE TEODACS PROJECT

THE TEODACS PROJECT

The approach chosen within the TEODACS project is based on the development of a co-simulation framework (FlexRayXpert.Sim) as well as a realistic prototype (FlexRayXpert.Lab) which covers the entire communication architecture and describes the system operation at different levels of abstraction, see Figure 4. The motivation for these two platforms is to combine the advantages of the two environments. This regroups, for the simulation environment, (a) the improved observability of the system, (b) the high flexibility for testing new configurations, and (c) the possibility to validate the system during earlier phases of the development process. Parallel to that, a hardware prototype provides real results both in the value and in the time domain. Moreover, the execution of test cases is usually faster in hardware than in a simulation environment when the system includes microelectronics components. Evidently, these advantages can only be cumulated when a close interfacing between the environments is available, see [20].

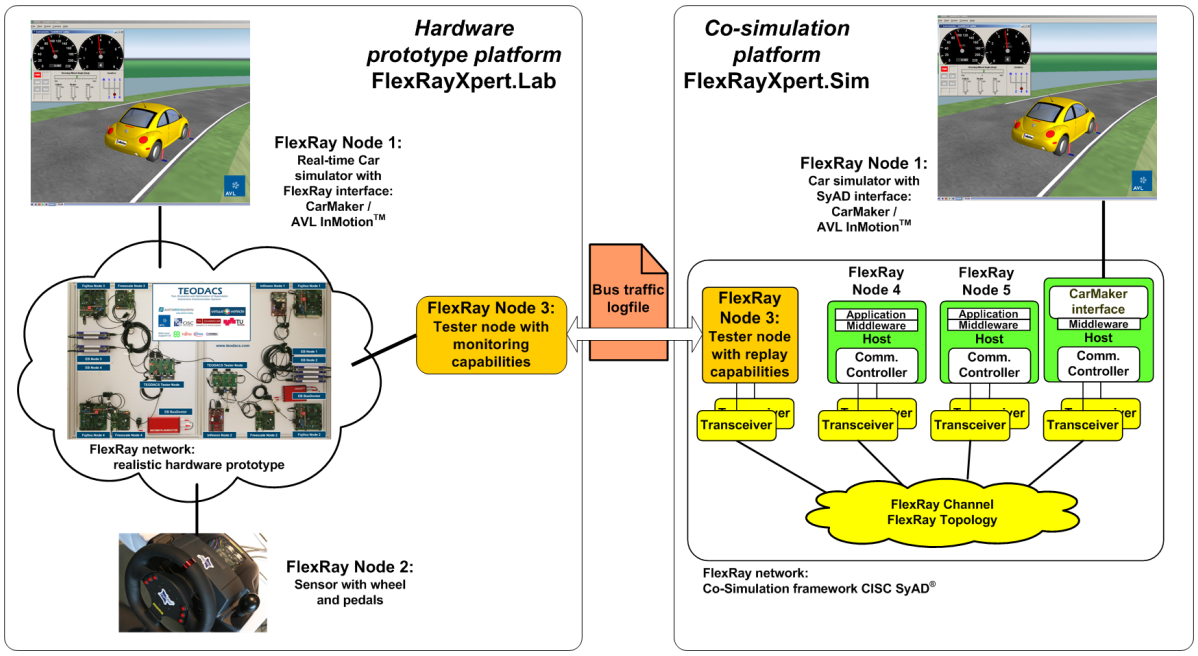


Figure 4: TEODACS approach with Drive-by-Wire use case

FlexRayXpert.Lab

The main aims of the prototype environment FlexRayXpert.Lab are to provide a realistic network reflecting the current car architecture and to understand the typical design, integration and validation challenges a car supplier is confronted to. Our prototype implements different topologies (active star, bus topology with different cable length) and regroups different suppliers: FlexRay transceivers from NXP (TJA1080) and austriamicrosystems (AS8221 and AS8224) as well as standalone FlexRay controllers from Fujitsu (MB88121B) and Infineon (CIC310), and integrated solutions from Freescale (S12XF512 MCU with embedded FlexRay controller).

FlexRayXpert.Sim

The FlexRayXpert.Sim co-simulation platform combines simulation models of the different FlexRay components (cables, transceiver, active/passive star, communication controller and AUTOSAR concepts) in order to simulate and analyze the entire communication architecture from the physical level up to application level. The co-simulation framework creates the possibility to combine different levels of accuracy according to the requested needs (e.g. physical layer modeling using high detailed VHDL-AMS, data link layer modeling using fast and high level SystemC). This largely reduces the processing resources and makes the analysis of such complex systems possible. The co-simulation framework used in FlexRayXpert.Sim is the System Architect Designer tool from CISC. The simulation of the entire architecture supports the analysis of the interactions between the single components and thus the design exploration of the assembled system. The two platforms are stimulated by the CarMaker / AVL InMotion simulator, which simulates the dynamics of a car driving on a road and thus provides a realistic workload for the network.

FLEXRAY-IN-THE-LOOP: DRIVE-BY-WIRE APPLICATION

The use case chosen in the TEODACS project consists of a Drive-by-Wire application, see Figure 4. In this example, a FlexRay node in FlexRayXpert.Lab is continuously sensing steering, gas and brake information (node 2). The raw sensor data is filtered in order to remove the outliers and after that both the raw data and the filtered data are sent over the FlexRay network to control the CarMaker / AVL InMotion simulator (node 1). The FlexRay traffic is traced in parallel into a file using a dedicated tester node (node 3). Parallel to the hardware setup, a similar architecture is provided in FlexRayXpert.Sim: The steering and braking information are replayed by a tester node (node 3) according to the trace performed from the hardware prototype, thus guaranteeing that the system under test is stimulated by the same input (driver information). The component under test in this case is the network infrastructure. Hence, architecture changes such as software component mapping (which function is assigned to which node), FlexRay configuration (e.g. length of the communication cycle) and network configuration (which slots are assigned to which nodes) can be performed in order to observe the changes within the system.

In our case, the sensor node in FlexRayXpert.Lab environment implements a filtering function in order to smooth the sensor results and remove the outliers. In this test campaign, different variants have been implemented in the FlexRayXpert.Sim simulation environment and the following architectures have been compared:

1. Filtering function is implemented directly at the sensor node and the results are sent using the FlexRay network (intelligent sensor).
2. The sensor is sending the raw data to the FlexRay network and another node (node 4) is performing the filtering
3. Different filtering algorithms are used at the remote node (node 4) to smooth the data

We expect naturally the first architecture to perform better since the averaging is performed directly with the sensor inputs (yielding to a higher sampling rate and better smoothing) and processed immediately (without communication delays). The other variants are required when the filtering operation cannot be performed at the sensor node (no processing unit or not enough memory available to process complex filters).

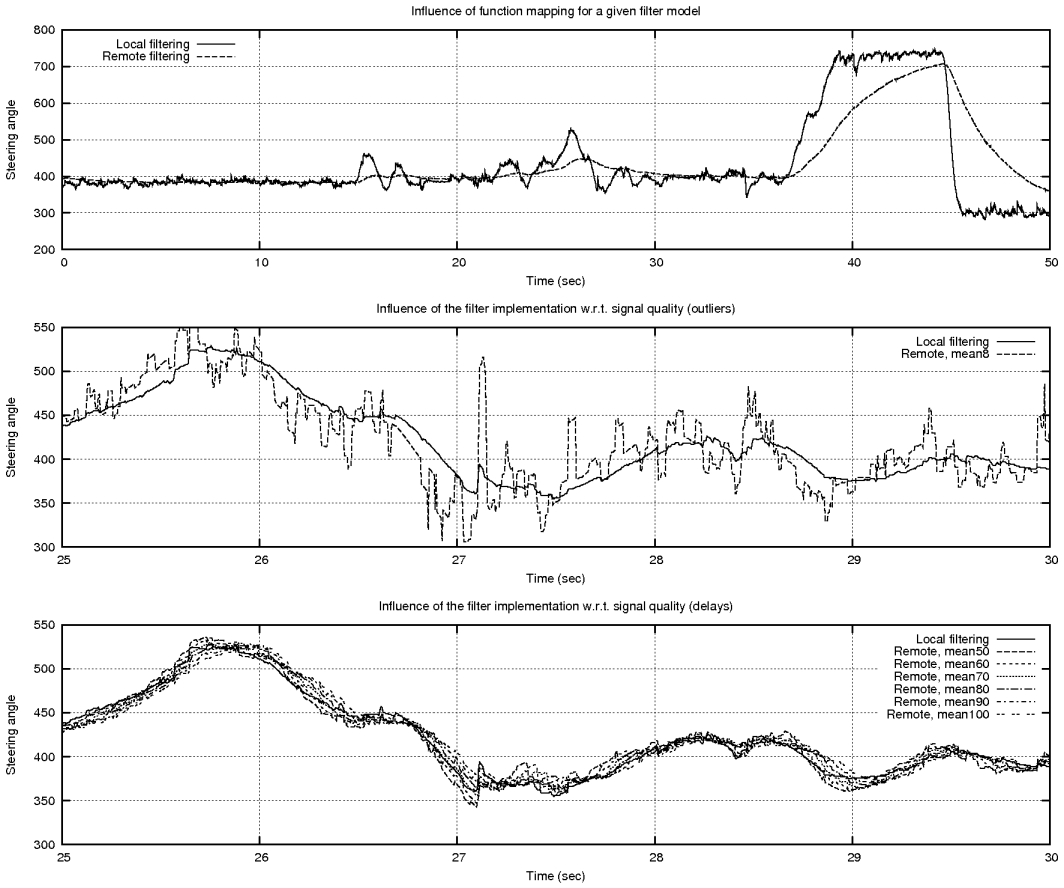


Figure 5: Test case results: (1): influence of function mapping for a given algorithm, (2) influence of the filter model w.r.t. signal quality (outliers) and (3) influence of the filter model w.r.t. signal quality (delay)

Figure 5 illustrates three test cases. The aim of the first test (first graph) is to show the influence of function mapping within the system. The same filtering function was placed once at the sensor node and once at a remote node. For this last architecture, the raw data has been sent to the network before processing. Because of the limited bandwidth, a down-sampling was performed before filtering (only one out of fifty samples was sent). This down-sampling led to an excessive smoothing and made the output not useful for the vehicle. The second test campaign (Figure 5: middle graph) illustrates the influence of the component model for the system. In this example an architecture with local filtering (IIR filter) is compared with an architecture with remote filtering (FIR, mean with 8 samples). The second architecture presents a very bad signal quality (the outliers are not properly removed) because of the poorly adapted filter model. The third test campaign (Figure 5: bottom) illustrates the influence of the filter modeling for the signal quality (delay). During this example, different remote filtering approaches (mean computation with number of samples considered varying between 50 and 100) have been compared. While they all present a good signal quality (low number of outliers), the result availability is delayed up to 100ms. In case of Steer-by-Wire application that presents strong real-time constraints, this delay might not be acceptable.

This test campaign illustrates the influence of the mapping between software functions and Electronic Control Units (ECUs) within the system as well as an evaluation between different filter implementation strategies. The benefit of using a cross-domain co-simulation platform for this analysis is to enable architecture decisions before the implementation of the system, thus saving re-design efforts (verification front-loading). The proposed cross-domain simulation framework provides (1) the flexibility for state exploration (modeling different architecture or filter concepts) and (2) the evaluation of the results within the assembled system (how the car react to the different stimulus).

SUMMARY/CONCLUSIONS

Electronics in cars are strongly integrated within the mechanics components. New approaches are thus required to support the efficient vehicle development and the cross-domain component integration at the very beginning of the design phase. We have proposed such a cross-domain simulation platform that in fact combines two state-of-the-art simulation platforms. Using an exemplary Drive-by-Wire application, we have shown that this approach enables the early verification of the assembled system and efficiently supports design decision and system evaluation at the beginning of the development process. The proposed platform further enables cross-domain system optimization in providing information about the best strategy for a given system and a given application.

REFERENCES

1. Ebert, C. and Jones, C., “Embedded software: Facts, figures, and future”, *IEEE Computer*, **42**(4): 42–52, 2009.
2. Leen, G. and Hefferman, D., "Expanding Automotive Electronic Systems". In IEEE Transaction on Computers: 88–93, January 2002.
3. Kajtazovic, S., Steger, C., Schuhai, A. and Pistauer, M., "Automatic Generation of a Verification Platform for Heterogeneous System Designs", Advances in Design and Specification Languages for SoCs, Kluwer Academic Publishers Boston/Dordrecht/London, 2005.
4. CISC Semiconductor Design+Consulting GmbH, "System Architect Designer (SyAD®)", http://www.cisc.at/syad, September 2009.
5. Schyr, C., Schaden, T. and Schantl, R., "New Frontloading Potentials through Coupling of HiL-Simulation and Engine Test Bed", FISITA 2008 World Automotive Congress, Munich, Germany, September 2008, F2008-12-317. See www.avl.com or www.ipg.de
6. Armengaud, E., Watzenig, D., Steger, C., Berger, H., Gall, H., Pfister, F. and Pistauer, M., "TEODACS: A new Vision for Testing Dependable Automotive Communication Systems", Third International Symposium on Industrial Embedded Systems (SIES 2008), Montpellier, France, June 2008, pages 289 – 292
7. Karner, M., Steger, C., Weiß, R., Armengaud, E., Watzenig, D. and Knoll, G., "Verification and Analysis of Dependable Automotive Communication Systems". Thirteenth International Conference on Emerging Technologies and Factory Automation (ETFA 2008), Germany, September 2008, pages 444 – 447
8. Armengaud, E., Watzenig, D., Karner, M., Steger, C., Weiß, R., Netzberger, C., Kohl, M., Pistauer, M., Pfister, F. and Gall, H., "Combining the Advantages of Simulation and Prototyping for the Validation of Dependable Communication Architectures: The TEODACS Approach", *SAE Int. J. Passeng. Cars - Electron. Electr. Syst.*, **2**(1): 309-318, 2009
9. Pelz, G., Mechatronic Systems: Modelling and Simulation with HDLs, Jon Wiley & Sons Ltd., Chisester, England, ISBN 0-470-84979-7, 2003
10. Ashenden, P.J., Petersion, G.D. and Teegarden, D.A., The System Designer’s Guide to VHDL-AMS, Morgan Kaufmann Publishers, San Francisco, CA, USA, ISBN 1-55860-749-8, 2003

11. Gerke, T. and DeMeis, R., "The Virtual Vehicle: Part 1 – In-Vehicle Network Simulation and Analysis", Automotive DesignLine [Online]: http://www.automotivedesignline.com/ United Business Media, London, United Kingdom, January 2009
12. Gerke, T. and DeMeis, R., "The Virtual Vehicle: Part 2 – Early Validation of Vehicle Electrical Systems and Power Management", Automotive DesignLine: http://www.automotivedesignline.com/ United Business Media, London, United Kingdom, February 2009
13. Gerke, T., Lagerqvist, S. and DeMeis, R., "The Virtual Vehicle: Part 3 – Developing Robust Wiring Harnesses", Automotive Design Line[Online]: http://www.automotivedesignline.com/ United Business Media, London, United Kingdom, March 2009
14. Gerke, T. and Lehmann, F., "The Virtual Vehicle: Part 4 – Automotive Design and Optimization of Electrical Fuel Injection Systems", Automotive DesignLine [Online]: http://www.automotivedesignline.com/ United Business Media, London, United Kingdom, April 2009
15. Synopsis, Inc., "MAST: The Analog, Mixed-Technology and Mixed-Signal HDL for Saber", http://www.synopsys.com/tools/sld/mechatronics/saber/pages/mast.aspx Synopsis, Inc., Mountain View, CA, USA, 2009
16. Ausejo, S., Suescun, A., Celigüeta, J.T., Moser, E. and Charlot, J.J., "A Methodology for Model Interchange and Simulation of Mechatronic Systems", Proceedings of the 9th Mediterranean Conference on Control and Automation, Dubrovnik, Croatia, ISBN 953-6037-34-3, 2001
17. Modelica Association, "Modelica® - A Unified Object-Oriented Language for Physical Systems Modeling Language Specification", Version 3.1, Sweden, 2009
18. Grimm, C., Barnasconi, M., Vachoux, A. and Einwich, K., "An Introduction to Modeling Embedded Analog/Mixed-Signal Systems using SystemC AMS Extensions", Open SystemC Initiative (OSCI), June 2008
19. Karner, M., Steger, C., Weiß, R. and Armengaud, E., "Optimizing HW/SW Co-Simulation based on Run-Time Model Switching", Forum on specification and Design Languages (FDL 2009), 6 pages, September 2009
20. Armengaud, E., Tengg, A., Karner, M., Steger, C., Weiß, R. and Kohl, M., "Moving Beyond the Component Boundaries for Efficient Test and Diagnosis of Automotive Communication Architectures", Conference on Emerging Technologies and Factory Automation (ETFA 2009), 8 pages, September 2009
21. IPG Automotive GmbH, "APO Library Reference Manual", Karlsruhe, Germany, December 2004

CONTACT INFORMATION

Michael Karner, Christian Steger, Reinhold Weiß
Graz University of Technology - Institute for Technical Informatics
Inffeldgasse 16, 8010 Graz, Austria
Email: michael.karner@tugraz.at, steger@tugraz.at, rweiss@tugraz.at

Eric Armengaud
Virtual Vehicle Competence Center
Inffeldgasse 21a, 8010 Graz, Austria
Email: eric.armengaud@v2c2.at

Markus Pistauer
CISC Semiconductor Design + Consulting GmbH
Lakeside B07, 9020 Klagenfurt, Austria
Email: m.pistauer@cisc.at

Felix Pfister
AVL List GmbH
Hans-List-Platz 1, 8020 Graz, Austria
Email: felix.pfister@avl.com

Further information about the TEODACS project is available at www.teodacs.com

ACKNOWLEDGMENTS

The authors wish to thank the ”COMET K2 Forschungsförderungs-Programm” of the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT), the Austrian Federal Ministry of Economics and Labour (BMWA), Österreichische Forschungsförderungsgesellschaft mbH (FFG), Das Land Steiermark and Steirische Wirtschaftsförderung (SFG) for their financial support.

Additionally we would like to thank the supporting companies and project partners austriamicrosystems, AVL List and CISC Semiconductor as well as Graz University of Technology and the University of Applied Sciences FH Joanneum.

Heterogeneous Co-Simulation Platform for the Efficient Analysis of FlexRay-based Automotive Distributed Embedded Systems

Michael Karner[1], Martin Krammer[1], Stefan Krug[1]
Eric Armengaud[2], Christian Steger[1], Reinhold Weiss[1]
[1] Graz University of Technology, Institute for Technical Informatics, Austria
[2] The Virtual Vehicle Competence Center, Austria
{michael.karner, steger, rweiss}@tugraz.at
eric.armengaud@v2c2.at

Abstract

Validation front-loading using simulation is required to save efforts and support decision making during the development process. For automotive distributed embedded systems, the fast increasing system complexity usually prevents the use of simulation encompassing the entire communication architecture. The simulation is then focused to single domain and/or components. We present in this work a heterogeneous co-simulation model of a FlexRay based system comprising simulation models from the software components down to the waveforms within the FlexRay electrical cables. This platform enables the efficient analysis of the assembled system and more especially of the interactions between the components. The description of the models is supported by three test campaigns in order to highlight the importance of holistic simulation as well as the potential of the proposed approach.

1 Introduction

The FlexRay technology [1] is being introduced as new wired network for automotive high-speed control applications. This protocol, in comparison to predecessors (e.g. CAN), presents a very large flexibility that results in a huge number of implementation variants. Parameters such as topology (passive line, active star, hybrid topologies), bus schedule (e.g. slot / cycle length), communication matrix (mapping between ECUs, slots and frames) are as much variables that can influence the quality of the communication.

One important aim of the TEODACS[1] project ("Test, Evaluation and Optimization of Dependable Automotive Communication Systems") is to understand the interactions between the layers building a dependable network and to evaluate the different effects influencing the communication (e.g. topology, EMC) [2]. The approach is based on the development of a heterogeneous co-simulation model of a FlexRay network tightly interfaced to a realistic FlexRay prototype. This aims at combining the good observability and diagnosability provided by the simulation environment with the realistic system behavior provided by the hardware prototype.

The focus of this document is set to the co-simulation platform. The holistic simulation of a FlexRay network represents a challenge due to the heterogeneous nature of the system. A typical network consists of analog components (physical layer, transceivers), digital components (data link layer, communication controllers), basic software components (middleware, AUTOSAR) and software components (application). An important problem is to obtain a long *simulated time* (time interval under observation) for the analysis of the entire system, while at the same time keeping a reasonable *simulation time* (computation speed of the simulation) and a reasonable simulation accuracy for each component. This is especially difficult for lower layers that require more performance for the simulation of complex analog components.

The proposed approach relies on the development of different simulation models in different simulation languages and environments. The models are interconnected within a co-simulation environment that enables the concurrent simulation of the assembled system. However, the overall simulation performance is limited by the slowest simulator. Hence, the co-simulation approach has been enhanced by a runtime switching method [3] in order to make the simulation of the entire communication architecture feasible within a reasonable amount of time.

The contributions of this work regroup (1) the design of a heterogeneous co-simulation model for the efficient simulation of automotive distributed embedded systems, and (2) the illustration of the analysis potential with selected use cases. The document is organized as follows: Section 2 reviews the existing simulation models for FlexRay systems. Section 3 describes the heterogeneous co-simulation platform developed within the TEODACS project and Section 4 illustrates the potential of this platform by using three test campaigns. Finally, Section 5 concludes this work.

[1] www.teodacs.com

2 Simulating automotive communication networks: FlexRay

The simulation of large heterogeneous systems including several ECUs communicating via a network is described in [4]. The implementation is done by using transaction based modeling. The authors use a hardware abstraction layer for high level access to the hardware components and simulate at a very high abstraction layer. Another transaction level modeling approach for communication networks is presented in [5]. Here, the authors describe the creation of a transaction level model for a FlexRay network. The model is used for an evaluation of time performances according to several different protocol parameters. The FlexRay interface is described in a time-behavioral way using transaction level modeling techniques where selected components of the FlexRay protocol are modeled. A significant drawback of both approaches and typically of all transaction level approaches is the loss of details. Even minor changes within the FlexRay configuration parameters may lead to a notably different network behavior. By using only high level transaction level modeling, lots of these subtle details may be abstracted away, making the FlexRay system simulation not realistic enough for trustworthy in-detail verification of the complete network. However, such simulation approaches can be useful for the analysis of e.g. scheduling at application level.

A further approach for analyzing complete communication network systems is residual bus simulation, which enables the emulation of an entire sub-network. Industrial solutions from e.g. Elektrobit[2], dSPACE[3] and Vector[4] are available. All of these examples are a combination of hardware and software components: The software emulates the functionalities of the missing ECUs and the application messages are transmitted to a real network using dedicated hardware components. Hence, the advantages of simulation (observability, traceability, flexibility,...) are lost for many parts of the system. This makes residual bus simulation feasible only for very specific and clearly defined requirements.

A strong standardization focus has been set to the middleware (low level software) with the **AUTOSAR** initiative (AUTomotive Open System ARchitecture) [6]. A methodology for mapping the AUTOSAR design process with SystemC for efficient simulation of AUTOSAR components is presented in [7]. It principally relies on the mapping of AUTOSAR software components to SystemC modules and AUTOSAR ports with SystemC ports. A different approach has been chosen in [8]. An implementation of a RTOS with SystemC is provided and the model is integrated into the AUTOSAR toolchain. SystemDesk[3] is used as offline simulation tool. The integration of the RTOS enables to perform detailed analysis of the timings, performance and the influence of the operating system on the application. Also the approach of simulating the whole AUTOSAR core on an instruction set simulator is possible. The high degree of details of this method comes at the cost of long simulation durations. Also combinations of using instruction set simulation and abstract functional simulation are possible [9].

Several simulation models for the **FlexRay communication controller** are available. For example, a SystemC-based communication controller is presented in [7]. The authors use the model for the timing analysis of selected interconnected AUTOSAR components. However, they do not provide any in-depth details about the implementation of the controller. A simulation model of a FlexRay communication controller using the Verilog hardware description language (HDL) is proposed in [10]. With this model the authors evaluate the effects of so called message-missing-failures on the FlexRay network. Another implementation of the controller using SystemC is described in [11]. The authors are using the simulation model for verification purposes before building a correspondent RTL model. Another simulation model of a FlexRay communication controller is demonstrated in [12]. Here, the controller is implemented using standard C language and simulated on a PC used to interact with real hardware components like engines.

Another FlexRay component with several existing simulation models is the **FlexRay transceiver** (bus driver). In [13] a generic transceiver model implemented using VHDL-AMS is presented by the company CISC. This model is fully compliant to the FlexRay specification (including all optional interfaces) and also models physical effects like thermal power. Another implementation of a transceiver model using VHDL-AMS is presented in [14]. It is a mixed-mode behavioral model and the authors performed several tests to demonstrate effectiveness and specification conformance of the developed model. A specific behavioral Saber model of a NXP transceiver is presented in [15]. However, no in-depth details are available about this implementation. A simulation setup that incorporates several simulation models for the FlexRay physical layer like transceiver, passive star and cables using Synopsis Saber is described in [16]. They demonstrate the capabilities of their implementation by analyzing the signal integrity of a FlexRay network topology.

There also has been some previous work in the area of **cable harness and topology** modeling. Nearly all work relies on the famous "telegrapher's equations", also known as the "RLGC-model". The implementation represents a pure structural model, made up of chained discrete elements [17]. This model primarily consists of cascaded line elements. Each element represents a corresponding section of the cable. One element is made up of four discrete parts: An inductance L, a resistor R, a capacitance C and a conductance G. Another cable model is proposed in [18] that demonstrates an optimized version of the RLGC-model implemented in a behavioral style. Both models consider losses across the given cable length and take the same set of parameters, which have to be determined prior to simulation in order to get adequate results.

The main limitation while simulating single compo-

[2] www.elektrobit.com
[3] www.dspace.com
[4] www.vector-worldwide.com

nents is the missing of inter-layer effects for the holistic analysis of the network. However, the interactions are required to create a realistic workload for the components.

3 FlexRayXpert.Sim co-simulation environment

3.1 Overview of the platform

Obviously, there exist several solutions for the simulation of dependable communication networks and especially of FlexRay networks. However, these simulations usually only cover a specific part of the FlexRay network and/or are implemented in a very abstract way. No holistic approach covering all parts of the network (from physical layer up to the application layer) with sufficient accuracy is available. This fact makes a comprehensive in-depth analysis of the network very hard or even impossible to achieve by using existing simulation setups.

The proposed work provides an accurate heterogeneous co-simulation model of the entire communication architecture that enables in-depth analysis of the network. One main challenge is to find a trade-off between accuracy of the simulation models and processing time. The FlexRayXpert.Sim environment includes simulation models from the physical layer up to the application layer and mechanics data. However, it can be clearly seen that using the same hardware description language (HDL) for all models is an impossible task. This would either lead to a loss in information for the lower layers or extremely complex and slow simulation models for the higher layers.

A possible solution for this challenge is co-simulation: simultaneous simulation of two or more parts of a system, described using different HDLs and/or different levels of abstraction, and/or by using several different simulators concurrently [19]. This creates the possibility to implement selectable levels of detail according to the requested needs within one common co-simulation. The simulation of different layers (e.g. physical up to application) within one common co-simulation is fundamental to the TEODACS approach.

The developed FlexRay co-simulation platform FlexRayXpert.Sim is consisting of several tools and simulation models. The tool CISC SyAD [20] is used as co-simulation framework for the FlexRay network. SyAD automatically generates the required co-simulation interfaces to allow for a synchronized communication between models developed by using different hardware description languages and simulators (e.g. OSCI for SystemC, Mentor AdvanceMS for VHDL-AMS). The FlexRayXpert.Sim co-simulation platform also supports mechanics simulation with the integration of the CarMaker/AVL InMotion [21] simulator. The capability of simulating concurrently microelectronics and mechanics further supports cross-domain mechatronics co-simulation and massively enhances the applicability of the TEODACS approach. The mechanics data is transferred via the FlexRay network as realistic payload and any effects resulting from this transfer (e.g. delay, jitter, loss of data, instability of control loops) can be easily analyzed. Furthermore, a tester node allowing to interface with a real hardware prototype is available. This tester node enables an efficient validation of the simulation models against the hardware, see [22] for further information regarding the test concept. Details about the performed validation of the simulation models against the hardware prototype can be found in [23]. The complete TEODACS FlexRayXpert.Sim platform is shown in Figure 1.

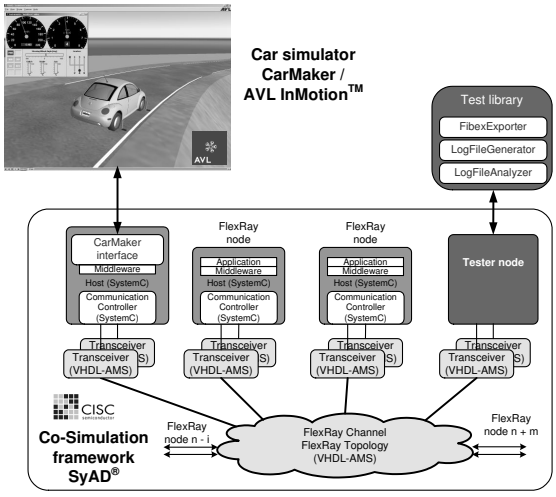


Figure 1. TEODACS FlexRayXpert.Sim co-simulation platform

3.2 Run-time model switching approach

Due to the heterogeneous nature of the simulated FlexRay system (analog components, digital components, software components, mechanics) a special problem rises that can not be handled by using standard co-simulation: Obtaining a long *simulated time* (time interval under observation) for the analysis of the entire system, while at the same time keeping a reasonable *simulation time* (computation speed of the simulation) and a reasonable simulation accuracy for each component. This is especially difficult for lower layers that require more performance for the simulation of complex analog components. For example, at system level a simulated time in the order of several seconds is required and can be easily achieved. However, for low level physical models of analog components (e.g. cables) typical simulated times are only in the area of milliseconds. This discrepancy has to be handled to allow for a holistic simulation of the FlexRay network.

To overcome this problem the run-time simulation model switching approach presented in [3] is applied within the FlexRayXpert.Sim platform. The basic idea behind run-time simulation model switching is to move from standard static co-simulation to a more advanced dynamic co-simulation approach. Here, for a specific part of the co-simulation the simulated models used are changed at run-time. For example, while most of the time a fast high-level

SystemC simulation model is used for cable and topology, for defined intervals a high-detailed analog VHDL-AMS model is used. This approach is somehow similar to an oscilloscope: it is possible to have a high detailed closer look at selected time intervals while the remainder of the simulation is computed in a more abstract, but noticeable faster way. More technical details about the run-time simulation model switching approach are described in [3], [24].

An important feature of the run-time simulation model switching approach is the possibility for the designer to dynamically shift between model accuracy and simulation performance, according to the requested needs. This is shown in Figure 2. By selecting the time intervals when

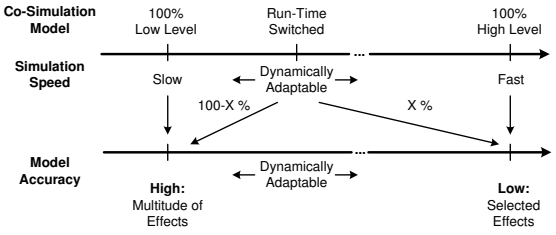


Figure 2. Shifting of complexity by using run-time model switching

a high detailed simulation model should be used the designer is able to specify the resulting simulation performance in advance. For example, for the analysis of the effects of reflections within the FlexRay cables on the mechanics data transferred by the top-level software application the designer may specify that only the first few bits of the FlexRay frame (e.g. the header) should be transmitted via the high detailed VHDL-AMS model of transceiver and topology. The remaining time the fast but low-detailed SystemC model of transceiver and cable harness should be used.

3.3 Simulation models: software & environment

The following two sections present in detail the simulation models developed within the TEODACS project. Basically, all components that are specified within the FlexRay specification (except bus guardians) are available as simulation models within the FlexRayXpert.Sim co-simulation environment. Additionally, there exist models that are implementing concepts of the AUTOSAR specification to support the modeling of software components and models of the complete car with focus to the mechanics. All these models can be used to create complete FlexRay communication networks - from the physical layer up to the application working within a car.

Looking at Figure 1 it can be seen that on top of the FlexRay co-simulation the simulator **CarMaker/AVL InMotion** [21] is placed. This simulator (and its models) provides realistic mechanics data to be transferred via the FlexRay network, hence, bringing additional authenticity into the co-simulation. It includes component models necessary for the detailed simulation of hybrid vehicles (like power train, driver, track, control unit functions, brakes etc.). Interconnections between CarMaker/AVL InMotion components are either internal, assuming an ideal communication backbone, or external, using the developed FlexRay communication network simulation. To bring together the worlds of mechanics (CarMaker/AVL InMotion) and microelectronics (CISC SyAD) a bridge has to be built between both domains. A strong challenge is the difference between the types of models and data formats, and therefore the resulting strongly differing simulator structures between microelectronics and multi-body based mechanics simulators like CarMaker/AVL InMotion. The interface has to support a smooth interaction between both domains by providing the required functionality for data exchange and synchronization between the domains. For this, the developed interface is split into two parts, one part for the mechanics and one for the microelectronics domain. This is shown in Figure 3.

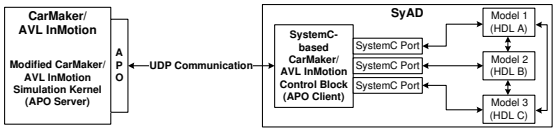


Figure 3. Interfacing concept between CISC SyAD and CarMaker/AVL InMotion

Within the mechanics simulator, the already existing network interface APO (Application Online) was extended to support suspending the next simulation step until the "continue" command arrives via the network interface. The mechanics simulator is acting as APO server and sending its data via the network to the APO client. The APO client is implemented as a SystemC module within the microelectronics co-simulation framework CISC SyAD. This client module controls the APO server running within the mechanics simulator and performs data exchange between the two simulation platforms. Additional information about the interfacing between CISC SyAD and CarMaker/AVL InMotion can be found in [25].

For easier development and simulation of applications (software components, SW-C) using the FlexRay infrastructure, an additional abstraction layer based on **AUTOSAR** on top of the communication system is introduced. This abstraction layer provides an interface to the SW-C similar to the Runtime Environment (RTE) [26]. Basically, it is a realization of the Virtual Function Bus (VFB) [27] that abstracts the communication between the software components (performing the routing either ECU internal or through a network) and performed by calling RTE API methods. The functionality also supports the execution of remote operations according to the AUTOSAR client-server communication.

An object oriented approach is used for the realization of the VFB abstraction layer within our FlexRay co-simulation. In the proposed approach, the entire code for a software component is encapsulated into a SystemC module in order to have a namespace for each software compo-

nent and thus avoids naming conflicts. An additional benefit is that multiple instances of the same software component do not make any problem. The challenge of keeping track of instances by using RTE_Instance references as defined in the AUTOSAR specification is solved by simply instantiating multiple objects of the same class.

The configuration of the proposed middleware consists of two parts: The first part concerns the configuration of the FlexRay cluster. Within TEODACS this configuration is extracted from a FIBEX [28] file. In addition to the FlexRay protocol parameters, it provides a mapping between the signals (application variables), the frames (data containers) and the frame triggering (condition for which the frames are sent on the bus; slot identifier in the case of FlexRay). The second part concerns the configuration of the VFB. It consists of a mapping between (a) the data elements, (b) the operations defined in a software component, and (c) the signals defined in the FlexRay configuration. In the proposed implementation this configuration is specific to the software component and therefore specified directly in the class description of the software component. Whenever a defined data element or operation is mapped to a signal name defined within the FlexRay configuration, the data is considered as remote and shall be transmitted via the FlexRay network. In addition to the transmission via the communication system, the data must also be distributed locally to software components running on the same ECU.

The communication on the VFB is done by calls of RTE API methods. In the performed implementation, generic versions of the methods are included in a class `SimulationSoftwareComponent` that is used as a base class for all other (user defined) software components. Based on the VFB configuration, specific RTE API methods are available in the user defined software components following the AUTOSAR naming convention. They are essentially aliases to the generic methods provided by the base class, which contain the actual functionality. The class diagram in Figure 4 displays the relation between the basic class and the user defined software components with respect to the RTE API for the communication on the VFB.

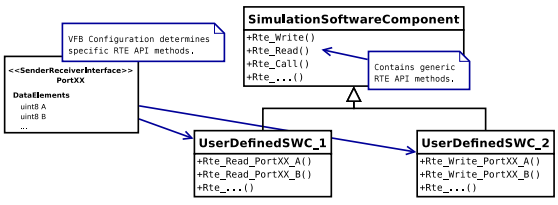


Figure 4. Relationship between basic class and user defined software components

To enable multiple software components to use a single communication controller, an additional `Connector` module is used within the simulation. This module acts as connection point between the communication controller and the software components allowing setups like depicted in Figure 5. It is responsible for handling the communication — local communication between software components on the same ECU as well as remote communication between distributed software components. Remote communication is realized by assembling and transmitting FlexRay frames according to the configuration, local communication is realized using shared memory.

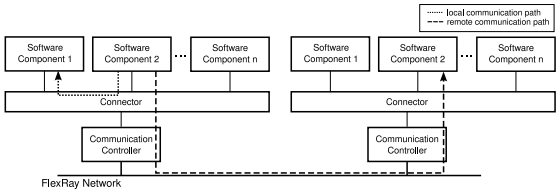


Figure 5. Local and remote communication between software components

3.4 Simulation models: hardware

Another important component within a FlexRay network is the **FlexRay communication controller**. It is responsible for the logical FlexRay protocol as defined within the FlexRay protocol specification [1]. Its main tasks are sending and receiving data via the FlexRay bus and error detection/reporting. Furthermore it is responsible for network wake-up and start-up as well as node re-integration. The controller implements also clock synchronization and further autonomously generates the schedule on which data transmission is triggered. In the TEODACS FlexRayXpert.Sim implementation, the communication controller is modeled by using SystemC, a C++ extension adding hardware modeling support. To gain additional simulation performance, the internals of the controller are modeled in a transaction level way. Hence, the data transfer within the communication controller is interface-method-call based instead of single point-to-point connections, thus speeding up the simulation. The complete communication controller is compliant to the FlexRay protocol specification 2.1 Rev A [1].

The implementation is based on the textual and SDL-based description of the controller within the FlexRay protocol specification. The internal structuring of the controller model can be seen in Figure 6. The SystemC modules are named in correspondence to the specification. For every module there exists a SystemC interface implementing the required functionality for the data exchange with the other modules. Every SDL signal within the specification is substituted by a corresponding interface call to the according module, hence speeding up the simulation. For communication with other components within the FlexRay networks, the controller implements two different types of connections. The interface to the transceiver is consisting of in total six SystemC ports for sending and receiving data. To the application,

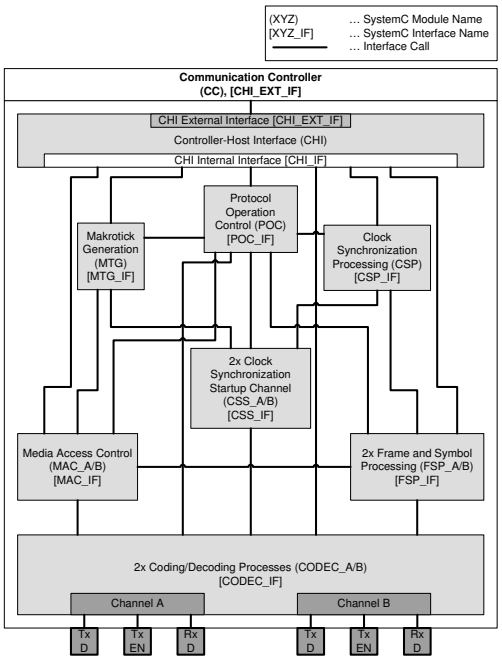


Figure 6. SystemC FlexRay communication controller model

the communication controller features a SystemC interface `CHI_EXT_IF` which allows the application to call the according interface-methods provided by the controller. Hence, there is no need for the application to care about e.g. hardware details of a memory access to some transmit buffer. The application is able to access the controller like it would be some normal piece of software, hence simplifying the application development.

The **FlexRay transceiver model** within FlexRayXpert.Sim was developed by CISC Semiconductor and is entirely written in *VHDL-AMS*. Its design follows closely the suggestions given in FlexRay's electrical physical layer specification [29]. It is a functional model, with emphasis on adequate output levels. Basically, the transceiver behaves as a signal converter, which operates in a bi-directional way: It converts binary signals to a trivalent differential signal, which is used to drive the two available bus wires `BP` (bus plus) and `BM` (bus minus). Possible logical levels therefore include high (`Data_1`), low (`Data_0`) and idle (`Idle`). The other way round, the transceiver converts this trivalent differential signal back to a binary signal. Furthermore, it is capable of additional functions, some being optional, like stand-by and sleep states or the bus guardian interface.

Concerning the overall input/output behavior, the transceiver model is fully compliant to FlexRay's electrical physical layer specification [29]. All in all, there are 51 parameters distributed among eight internal modules, ensuring simple adaptation to emulate a vendor specific hardware circuit. The transceiver's internal logic module is modeled in an all-digital style. An IO-Interface provides the necessary analog to digital conversion, with respect to the given parameters. All levels and thresholds are fully configurable, using the specified parameters. In order to achieve good simulation behavior and signal integrity, two modules are quite important: The first being the receiver module, responsible for converting analog bus signals to digital data. Due to its threshold levels, the receiver is important for digital signal recovery. Second, the sender module has shown to be most critical when it comes to signal integrity. The waveforms delivered by the sender can be seen as stimulus for the entire communication system, so this input variable should be generated as close-to-reality as possible.

The transceiver implicitly models an open circuit termination for bus cables. Therefore, an additional terminating element might be required. Depending on the network's topology, transceivers may be terminated in different ways. FlexRay's electrical physical layer application notes suggests the use of a so-called split termination, made up of three resistors and one capacitor, placed between both bus lines directly at the transceiver. This should prevent physical effects like ringing and reflections that decrease signal integrity, see [30]. Different styles of termination are modeled within the FlexRayXpert.Sim environment by using VHDL-AMS.

In addition to the high-detailed VHDL-AMS model there also exists a *SystemC* **FlexRay transceiver** simulation model. This is required to support the run-time simulation model switching approach within TEODACS FlexRayXpert.Sim. The SystemC model is also implemented according to the FlexRay physical layer specification [29]. However, the external interface to the host and communication controller consists of digital SystemC ports. The analog bus interface is emulated by generating corresponding floating point values instead of real voltages. Here, also rise and fall times are modeled. The SystemC transceiver model features a good trade-off between model accuracy and simulation performance, being suitable for scenarios where there is no in-depth interest on the FlexRay physical layer or the run-time simulation model switching approach is used.

FlexRay also introduces **active and passive stars** for topology structuring. A passive star is built implicitly by connecting all nodes on a linear passive bus to a single point. This structure can be useful, but doesn't extend the reach of a topology that much. This is also implemented within our FlexRay simulation. Additionally, an active star is specified which contains one bus driver per connected branch. Its primary purpose is to split the network into different branches in order to improve signal quality. It might further implement fault detection and containment. An active star has a digital central logic unit, a power supply interface, and an optional bus guardian interface. In our simulation environment, an active star model is available. Based on the work of [31], its digital logic is written in pure VHDL and was ported to SyAD, retaining its multi-level block design. It supports low

power modes on each individual branch and the central logic unit, respectively. The final active star model is capable of connecting with up to four branches.

In chapter 2 two different **cable harness** models were introduced. Both of them are implemented using *VHDL-AMS* within the FlexRayXpert.Sim environment. The required values for R, L, G and C were determined using an oscilloscope by measuring the lines delay per unit length, stimulated with a step function. Afterwards, the simulation models were verified against real FlexRay hardware measurements, showing very good correlation. The parameterization using the RLCG parameters further enables the analysis of different cable types (e.g. dedicated CAN or FlexRay cables) as well as the simulation of environment variations (e.g. temperature). Additional models of cable harness related components are available as well. A common mode choke model comprising of two coupled inductances as suggested by [30] is written in VHDL-AMS. An electrostatic discharge protection, modeling NXP's PESD1FLEX integrated circuit, is available in the same language as well.

Like with the transceiver, there also exists a *SystemC* **cable harness** implementation to support the run-time simulation model switching approach. This SystemC simulation model is rather simple compared to its VHDL-AMS based counterpart. It only features two length-based effects: attenuation and delay. Instead of real voltages, floating point values are transmitted and modified consistently to the parameters based on the cable length. However, this model complexity is more than enough for doing e.g. high-level or timing analysis of the FlexRay network and to be used within the simulation model switching.

4 Use case: FlexRay architecture analysis

The aim of this section is to illustrate the potential of the proposed heterogeneous co-simulation platform for the efficient analysis of the system. One main advantage is the availability of the different abstraction levels and consequently the possibility to analyze the interactions within the assembled system. In the following, we focus on three use-cases: (1) influence of software allocation, (2) influence of FlexRay schedule and (3) influence of the topology for the system.

4.1 Influence of software allocation

This first test campaign illustrates the influence of the mapping between software functions and Electronic Control Units (ECUs) as well as the influence of the implementation choice of a filter algorithm for the system. The system under test consists of a sensor node producing a noisy output that requires to be smoothed in order to remove the outliers. Because of the desired high-level analysis, only the SystemC based FlexRay simulation models (CarMaker/AVL InMotion interface, AUTOSAR/application, controller, transceiver and topology) described in Section 3 are used here. In this test campaign, different variants have been implemented and the following architectures have been compared:

- Filtering function is implemented directly at the sensor node and the results are sent using the FlexRay network (intelligent sensor)
- The sensor is sending the raw data to the FlexRay network and another node is performing the filtering
- Different filtering algorithms are used at the remote node to smooth the data

We expect naturally the first architecture to perform better since the averaging is performed directly with the sensor inputs (yielding to a higher sampling rate and better smoothing) and processed immediately (without communication delays). The other variants are required when the filtering operation can not be performed at the sensor node (no processing unit or not enough memory available to process complex filters).

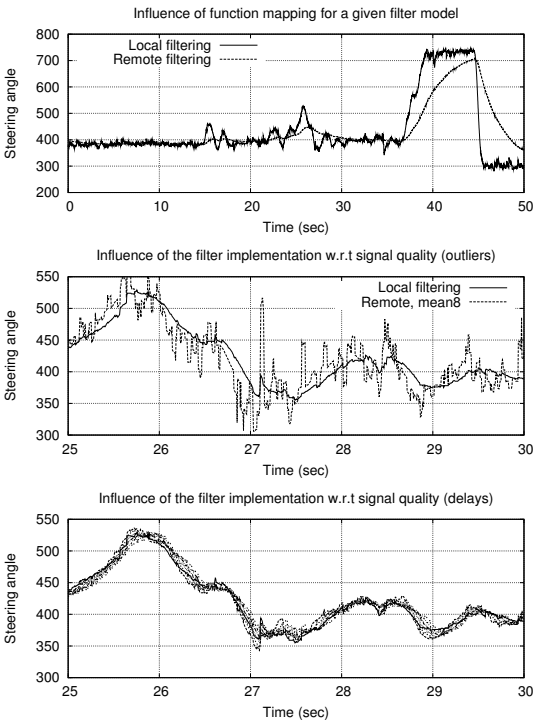


Figure 7. Effects of filter strategies performed within a FlexRay network

Figure 7 illustrates three test cases. The aim of the first test (first graph) is to show the influence of function allocation within the system. The same filtering function was placed once at the sensor node and once at a remote node. For this last architecture, the raw data has been sent to the network before processing. Because of the limited bandwidth, a down-sampling was performed before filtering (only one out of fifty samples was sent). This down-

sampling led to an excessive smoothing and made the output not useful for the vehicle.

The second test case (Figure 7: middle graph) illustrates the influence between component (filter) model and architecture for the system. In this example a local filtering (IIR filter) is compared with remote filtering (FIR, mean with 8 samples). The second architecture presents a very bad signal quality (the outliers are not properly removed) because of the poorly adapted filter model.

The third test case (Figure 7: bottom) illustrates the influence of the filter algorithm for the signal quality (delay). During this example, different remote filtering approaches (mean computation with number of samples varying between 50 and 100) have been compared. While they all present a good signal quality (low number of outliers), the result availability is delayed up to 100ms. In case of Steer-by-Wire application that presents strong real-time constraints, this delay might not be acceptable.

4.2 Influence of the FlexRay schedule

This second test campaign illustrates the influence between communication schedule, task execution and signal dynamics. For that, three functions (sinus, pulse, gas pedal) generated using CarMaker/AVL InMotion have been taken as inputs for transmission (Frame F1). After F1 is received from an application it triggers a runnable within the software component to send the data back using frame F2. The quality of the resulting outputs depends on the refresh period between two successive transmissions, and on the delay between F1, execution of the software task and F2. Due to the high-level analysis, like in the previous test campaign only the SystemC based FlexRay simulation models described in Section 3 are used. For this test campaign eight different schedules with different periods and delays have been defined, see Figure 8 for an overview.

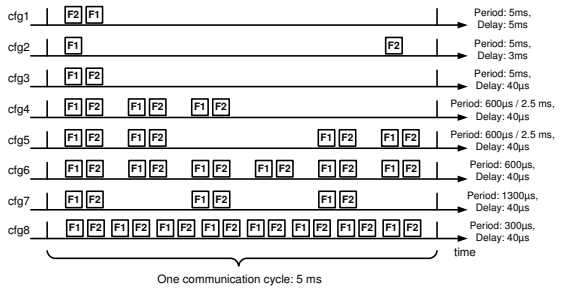


Figure 8. FlexRay configuration under test

Figure 9 illustrates the results for the three test stimuli. It can be observed that the sinus and pulse signals are differently altered depending on the communication schedule. The sinus signal presents a distortion both for the time and for the value components (some values such as extremum are not reached). Regarding the pulse, the distortion is limited to the time component. In general, a higher refresh rate leads to a better correlation between the input signal transmitted in frame F1 and the output signal

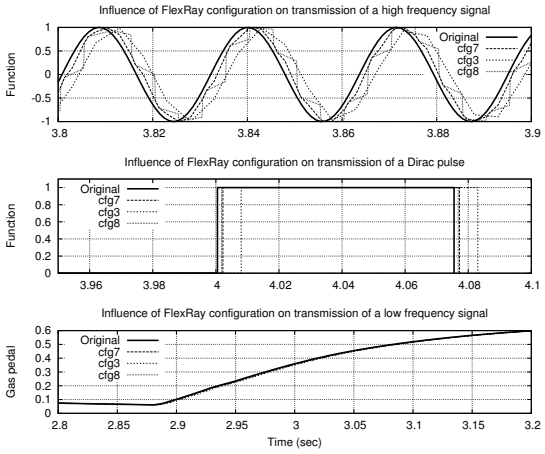


Figure 9. Input stimulus and system reaction

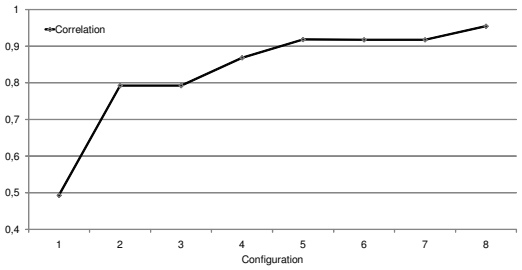


Figure 10. Correlation results

transmitted in F2, see Figure 10. There are however some limitations (e.g. configuration 5, 6, 7) where the sending of additional messages does not lead to a better output signal. In this case, the corresponding software component is executed at the beginning of the cycle and the additional frames are not taken into account. It can be moreover noticed that the different schedules have no visible effects on the gas pedal signal (third waveform). This third stimulus signal does not present a high dynamic and therefore the delays and distortion are quite minimal.

4.3 Signal integrity analysis

This third test campaign illustrates the strong local interactions for the analysis of the integrity of the electrical signal as well as the necessity to analyze the assembled system and not only single components (such as physical layer). In this experiment, a FlexRay network topology slightly out of specification has been defined. This leads to reflections that alter the signal integrity within the cables. To permit the simulation of such effects, for topology (cable harness etc.) and transceiver the VHDL-AMS based simulation models described in Section 3 are used. The higher levels of the FlexRay system are sim-

ulated using SystemC models (communication controller, AUTOSAR/application).

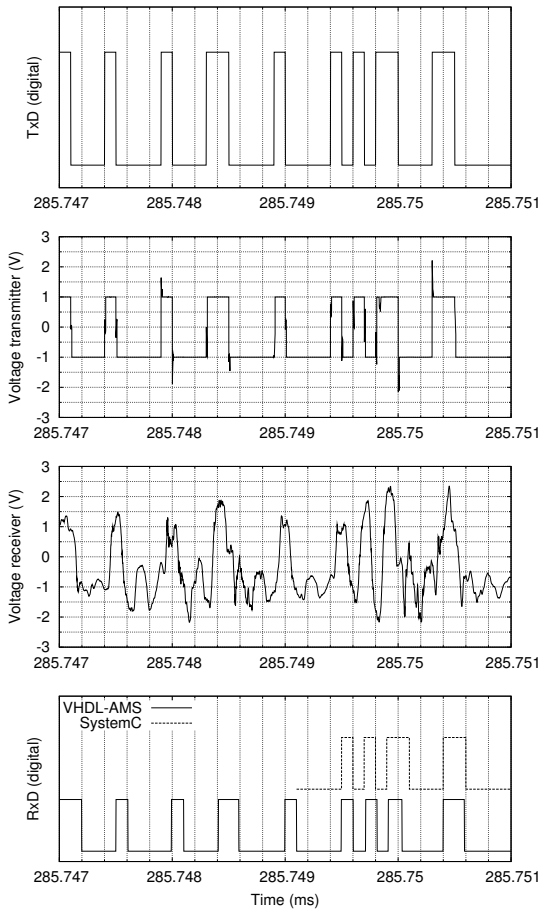


Figure 11. FlexRay run-time switching for cross-layer effects analysis

Figure 11 illustrates part of the FlexRay frame (1) at sample level (digital interface of the transceiver) at transmitter, (2) at analog level (analog interface of the transceiver) at transmitter, (3) at analog level at receiver, and (4) at sample level at the receiver. This exemplary data transfer illustrates the capacity to analyze the interactions between the network topology, the digitalized signal and the FlexRay frames. The analysis was enabled by applying the run-time model switching approach described within Section 3.2, leading to reasonable simulation performance for the effects analysis between physical and data link level. Within this test campaign, after 40 bits (= header) of each FlexRay frame it was switched from the highly accurate VHDL-AMS models of topology and transceivers to the respective fast but low detailed SystemC-based models. This is shown in Figure 11 by the SystemC RxD signal which starts initializing itself before switching takes place. Here, also the different model accuracies of VHDL-AMS and SystemC topology and transceiver models for this slightly out of specification topology can be seen. Using run-time switching lead to an increase in simulation performance compared to VHDL-AMS only simulation by a factor of about 3 while still having the same high accuracy within the selected time interval. More details about the specific run-time model switching setup used can be found in [24].

Two major findings have been performed during this analysis: First, for a given topology and a given transmitter, the data transmitted have an influence on the syntactic correctness of the frame. Hence, the bitstreams b00000000 and b01010101 for example will generate different kind of reflections on the bus that might lead to different appreciation of the frame correctness. Second, within a given topology a frame might be interpreted differently by the different nodes within the system (some nodes see the frame as correct while the same frame is rejected as faulty by other nodes). This effect comes from the relative placement between the sender and receivers and the different effects of the reflections for each single point-to-point connection. This effect might lead to asymmetric behaviors that can be a threat for the system stability.

5 Conclusion

Simulation is an important step that enables architecture decisions before the implementation of the system, thus saving re-design efforts (verification front-loading). In the case of automotive distributed embedded systems, the concurrent co-simulation of complex heterogeneous models is required to model the entire system. The proposed FlexRayXpert.Sim platform developed within the TEODACS project provides (1) the flexibility for efficient state exploration (with quantification of the effects on the system) and (2) the evaluation of the results within the assembled system (interaction between the components). The presented evaluation campaigns have illustrated the different analysis possibilities at different abstraction levels as well as the local interactions between the components within the system.

Acknowledgement

The authors wish to thank the ”COMET K2 Forschungsförderungs-Programm” of the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT), the Austrian Federal Ministry of Economics and Labour (BMWA), Österreichische Forschungsförderungsgesellschaft mbH (FFG), Das Land Steiermark and Steirische Wirtschaftsförderung (SFG) for their financial support.

Additionally we would like to thank the supporting companies and project partners austriamicrosystems, AVL List and CISC Semiconductor as well as Graz University of Technology and the University of Applied Sciences FH Joanneum.

References

[1] FlexRay Consortium, *FlexRay Communications System – Protocol Specification V2.1 Rev A*, December 2005.

[2] E. Armengaud, D. Watzenig, M. Karner, C. Steger, R. Weiß, C. Netzberger, M. Kohl, M. Pistauer, F. Pfister, and H. Gall, "Combining the Advantages of Simulation and Prototyping for the Validation of Dependable Communication Architectures: the TEODACS Approach", in *SAE World Congress, 2009-01-0763*, April 2009.

[3] M. Karner, C. Steger, R. Weiss, and E. Armengaud, "Optimizing HW/SW Co-Simulation based on Run-Time Model Switching", in *Proceedings of the 12th Forum on specification & Design Languages*, September 2009, p. 6.

[4] R. Buchmann, M. Cartron, and Y. Bonhomme, "Transaction-based Modeling for Large Scale Simulations of Heterogeneous Systems", in *Simutools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, 2009, p. 2, ICST, Brussels, Belgium, Belgium. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[5] M. Cheikhwafa, S. Le Nours, O. Pasquier, and J. Calvez, "Transaction Level Modeling of a FlexRay Communication Network", in *Proceedings of the 12th Forum on specification and Design Languages*, Sept. 2009, p. 4.

[6] "AUTOSAR - Technical Overview, v2.2.2", Technical report, AUTOSAR GbR, August 2008.

[7] M. Krause, O. Bringmann, A. Hergenhan, G. Tabanoglu, and W. Rosenstiel, "Timing Simulation of Interconnected AUTOSAR Software-Components", in *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE '07*, 2007, pp. 1–6.

[8] M. Becker, H. Zabel, W. Müller, and U. Kiffmeier, "Integration abstrakter RTOS-Simulation in den Entwurf eingebetteter automobiler E/E-Systeme", in *Proc. MBMV 2009*, March 2009.

[9] M. Krause, D. Englert, O. Bringmann, and W. Rosenstiel, "Combination of Instruction Set Simulation and Abstract RTOS Model Execution for Fast and Accurate Target Software Evaluation", in *Proc. CODES/ISSS 2008*, 2008, pp. 143–148, New York, NY, USA. ACM.

[10] V. Lari, M. Dehbashi, S. G. Miremadi, and N. Farazmand, "Assessment of Message Missing Failures in FlexRay-Based Networks", in *Dependable Computing, 2007. PRDC 2007. 13th Pacific Rim International Symposium on*, 2007, pp. 191–194.

[11] W. S. Kim, H. A. Kim, J.-H. Ahn, and B. Moon, "System-Level Development and Verification of the FlexRay Communication Controller Model Based on SystemC", *Future Generation Communication and Networking*, vol. 2, pp. 124–127, 2008.

[12] C. Xu and Y. Zhang, "Simulation of FlexRay Communication Using C Language", *Computer Science and Computational Technology, International Symposium on*, vol. 2, pp. 272–276, 2008.

[13] CISC Semiconductor Design+Consulting GmbH, (Feb. 2007), Product Information: FlexRay Transceiver Model [Online]. Available: http://www.cisc.at/automotive/flexray.html

[14] C. Muller, M. Valle, R. Buzas, and A. Skoupy, "Mixed-Mode Behavioral Model of a FlexRay Physical Layer Transceiver", in *Proceedings of the 19th European Conference on Circuit Theory and Design*, Aug. 2009, p. 4.

[15] NXP Semiconductors, (Nov. 2007), NXP FlexRay network simulations: Safeguard the operation of your FlexRay network architectures [Online]. Available: http://www.nxp.com/acrobat_download/literature/9397/75016200.pdf

[16] T. Gerke and D. Bollati, "Development of the Physical Layer and Signal Integrity Analysis of FlexRay Design Systems", in *Simulation & Modelling Mechatronics (SP-2111)*. 2007.

[17] H. Johnson and M. Graham, *High-speed signal propagation: advanced black magic*, Prentice Hall Press, Upper Saddle River, NJ, USA, 2003.

[18] M. Schlegel, G. Herrmann, and D. Mueller, "Entwicklung eines effizienten VHDL-AMS-Modells der verlustbehafteten Leitung", Technical report, TU Chemnitz, Fakultaet fuer Elektrotechnik und Informationstechnik, 2001.

[19] S. Kajtazovic, C. Steger, A. Schuhai, and M. Pistauer, "Automatic Generation of a Coverification Platform", *Applications of Specification and Design Languages for SoCs: Selected papers from FDL 2005*, , pp. 187–203, 2006.

[20] CISC Semiconductor Design+Consulting GmbH, (Nov. 2009), SyAD Online Documentation [Online]. Available: http://www.cisc.at/syad/

[21] AVL List GmbH, (May 2007), AVL Hybrid Development Platform™[Online]. Available: http://www.avl.com

[22] E. Armengaud, A. Tengg, M. Karner, C. Steger, R. Weiss, and M. Kohl, "Moving Beyond the Component Boundaries for Efficient Test and Diagnosis of Automotive Communication Architectures", in *Fourteenth IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2009), 8 pages*, Sept. 2009.

[23] M. Krammer, F. Clazzer, E. Armengaud, M. Karner, C. Steger, and R. Weiss, "Exploration of the FlexRay Signal Integrity using a Combined Prototyping and Simulation Approach", in *Proceedings of the 13th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems, 2010. DDECS '10*, April 2010, p. 6.

[24] M. Karner, C. Steger, R. Weiss, and E. Armengaud, "Holistic Simulation of FlexRay Networks by Using Run-Time Model Switching", in *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition, 2010. DATE '10*, March 2010, pp. 544–549.

[25] M. Karner, E. Armengaud, C. Steger, R. Weiss, M. Pistauer, and F. Pfister, "A Cross Domain Co-Simulation Platform for the Efficient Analysis of Mechatronic Systems", in *SAE World Congress, 2010-01-0239*, April 2010, pp. 1–14.

[26] "AUTOSAR - Specification of the RTE, v2.0.1", Technical report, AUTOSAR GbR, June 2008.

[27] AUTOSAR GbR, *AUTOSAR: Specification of the Virtual Functional Bus*, February 2008.

[28] "ASAM MCD-2 NET, Fibex V3.0 - Data Model for ECU Network Systems, available at http://www.asam.net", ASAM – Association for Standardization of Automation and Measuring Systems, January 2008.

[29] FlexRay Consortium, *FlexRay Communications Systems – Electrical Physical Layer Specification V2.1 Rev B*, December 2005.

[30] The FlexRay Consortium, *FlexRay Electrical Physical Layer Application Notes*, 2006, Version 2.1, Revision B.

[31] C. Netzberger, *Design Document for the Active Star VHDL Model*, 2008.

Exploration of the FlexRay Signal Integrity using a Combined Prototyping and Simulation Approach

Martin Krammer*†, Federico Clazzer*, Eric Armengaud*, Michael Karner†, Christian Steger† and Reinhold Weiss†
*The Virtual Vehicle Competence Center, Austria
{martin.krammer, federico.clazzer, eric.armengaud}@v2c2.at
†Graz University of Technology, Institute for Technical Informatics, Austria
{michael.karner, steger, rweiss}@tugraz.at

***Abstract*—Ensuring a correct signal integrity within the entire FlexRay network and for all the possible environmental situations is mandatory for reliable operation of the distributed application. However, this is a goal difficult to reach due to the large number of parameters that influence the signal integrity. The use of simulation is a natural answer to efficiently support space exploration. We discuss in this work how the TEODACS test approach supports the validation process of the simulation models for FlexRay topologies and provides trustfulness for the simulation results even if hardware reference is not available. Further, we introduce a new method for the advanced analysis and evaluation of signal integrity in FlexRay networks.**

I. Introduction

Automotive electronics are organized as complex distributed systems. The availability of the information among the Electronic Control Units (ECUs) leads to a better apprehension of the vehicle environment and therefore is an enabler for new services. Advanced functions such as electronic stability control (ESC), parking support or battery management have been introduced or strongly enhanced thanks to the efficient distribution of the information within the vehicle. In parallel, the introduction of new, complex functions leads to higher bandwidth requirements as well as increased complexity for an efficient deployment and validation of the communication architecture.

The FlexRay technology [1] based on the Time-Triggered concept [2] has been introduced in that context. This network presents a data rate of 10Mbits/s (factor 10 faster than CAN) and supports different topologies such as bus line, stars or hybrid in order to improve the system flexibility and robustness. The challenge, while designing a FlexRay physical network, is to ensure a correct signal integrity (quality of the electrical signal within the cable) within the entire network and for all the possible environmental situations. Different aspects, coming from the topology (e.g. cable length and type, termination, presence of passive or active stars) as well as environmental (e.g. EMC, parameter variation due to temperature, humidity or ageing effects), might influence the signal integrity.

Within the TEODACS[1] project ("Test, Evaluation and Optimization of Dependable Automotive Communication Systems" [3]) a heterogeneous co-simulation platform tightly interfaced to a realistic hardware prototype has been developed. Both development platforms support the analysis of the communication architecture based on the FlexRay technology. Interfacing the two platforms permits to combine the benefits of the two worlds: good observability, diagnosability as well as efficient space exploration provided by the simulation environment with the realistic system behavior provided by the hardware prototype.

The motivation of this work is to describe the approach chosen in TEODACS for validating the simulation models for the FlexRay topology and further for ensuring trustfulness of the simulation results without requiring a reference hardware. The contributions are (1) a combined measurement / simulation test method for fast and trustful space exploration, as well as (2) a 10-bits eye diagram method that enables a better analysis of the signal integrity and furthermore can be used as metrics to compare different topologies.

The paper is divided as follows: Section II provides a state of the art concerning the simulation and analysis of FlexRay topologies. After that, Section III describes the test and analysis methods developed within the TEODACS project. The validation of the simulation models is illustrated in Section IV, and finally Section V concludes this work.

II. Simulation and Analysis of FlexRay Topologies

The FlexRay communication medium [4] is defined as differential bus implemented on twisted cable pairs (similar to CAN technology). Good signal quality is required during the entire product life cycle from system design to normal operation. However, signal integrity strongly depends on different parameters such as network topology, cable length, bus driver types as well as manufacturing tolerances and environmental impacts [5]. Unadapted parameters might lead to circuit ringing, reflections, and other parasitic effects that lead in turn to communication failures. It is evident that the exhaustive test in hardware of all parameter combinations is not feasible. Simulation, due to the good controllability of the different model parameters, efficiently supports design space exploration.

Regarding the FlexRay physical layer, different models of transceivers exists. The models proposed in [6], [7] are implemented in VHDL-AMS while the one proposed in [8] is

[1] www.teodacs.com

a specific behavioral Saber model. The focus of these models is the validation of single components (e.g. conformance test). Regarding the cable harness and topology modeling, nearly all work relies on the "telegrapher's equations" or "RLGC-model" [9]. An optimization (computation speed) is proposed in [10].

The efficient analysis of the signal integrity must include the interactions between the different components building the network and consequently requires the capability to regroup the different models within one (co-)simulation environment. A simulation setup that incorporates several simulation models for the FlexRay physical layer like transceiver, passive star and cables using Synopsis Saber is described in [11]. The authors demonstrate the capabilities of this implementation by analyzing the signal integrity of a FlexRay network topology. The approach chosen in the TEODACS project [12] relies on a co-simulation platform that combines models of the different FlexRay components (cables, transceiver, active/passive star, communication controller, AUTOSAR[2] concepts). It allows the simulation and analysis of the entire communication architecture from the physical level up to application level. The co-simulation framework creates the possibility to combine different levels of accuracy according to the requested needs (e.g. physical layer modeling using high detailed VHDL-AMS, data link layer modeling using fast and high level SystemC).

The authors of [13] discuss the difficulties arising by deploying multiple automotive bus systems (e.g. MOST, D2B, CAN, FlexRay, LIN) in one single vehicle. They identify various effects that may endanger the communication systems such as ground shift, battery drop, electrostatic discharge, or electromagnetic interference (EMC) and that may lead to incorrectly sampled bits or even disturbed communication links. Furthermore, ringing and line reflection are identified as main problems concerning the physical layer and signal integrity. The focus of [14] is set to the analysis of CAN physical layers. The models used for simulation include numerous CAN nodes, EMC protection circuitry, a transceiver and a CAN controller model (all in a Saber environment). The nodes are interconnected using a twisted pair transmission line model. The wire model was validated against measurements. The verification process of the network is based on worst case parameters and tolerances. Besides basic signal properties like rise and fall times, the correct sampling of bits is thoroughly investigated.

The works described in [11], [15] focuses on the development of physical layers and signal integrity analysis for FlexRay networks. The authors paid particular attention to the following components: signal filters, active stars, transceiver, transmission line, ESD protection elements, topology type, and termination. They further identify the following aspects of signal integrity as critical: signal propagation delay, asymmetric delay [4, p.13], bit deformation, truncation of TSS, and frame stretching. The signal integrity analysis is based on eye diagrams. To assure proper bit strobing during transmission of data streams, the asymmetric delay is monitored. No significant variances were detected during simulation of a sample six-node passive star.

The works discussed before present two main limitations. First, the trustfulness of the simulation results is difficult to assess (how to be sure that the different models with a new parameter set will react as its hardware counterpart would do). Second, the signal integrity analysis consists of very different aspects and is difficult to automate. This automation is especially required for the efficient analysis of entire systems (not only sub-systems generating standard stimuli) with real communication content, as well as for objective comparison between different topology concepts.

III. TEODACS Approach

A. TEODACS overview

The approach chosen within the TEODACS project is based on the development of a co-simulation framework (FlexRayXpert.Sim) as well as a realistic prototype (FlexRayXpert.Lab) which covers the entire communication architecture and describes the system operation at different levels of abstraction, see Figure 1. The FlexRayXpert.Sim co-simulation platform (Figure 1, left part) combines simulation models of the different network components (cables, transceiver, active/passive star, communication controller, AUTOSAR concepts) in order to simulate the entire communication architecture. The co-simulation framework [16] creates the possibility to implement selectable levels of accuracy according to the requested needs, thus largely reducing the processing resources and making the analysis of such complex systems possible. In our case we are using the System Architect Designer [17] (SyAD®) tool from CISC. Both platforms are stimulated by the CarMaker / AVL InMotion [18], which simulates the car's power train as well as its environment (road, driver) and thus provides a realistic workload for the network. Further information concerning FlexRayXpert.Sim is available in [12].

The focus of the prototype environment FlexRayXpert.Lab is to provide a realistic network reflecting the current car architecture and to understand the typical design, integration and validation challenges a car supplier is confronted with. Figure 1 (right part) illustrates our prototype. It implements different topologies (active star, bus topology with different cable length) and regroups different suppliers such as austriamicrosystems [19], NXP, Fujitsu, Freescale or Infineon. This prototype is also stimulated by the CarMaker / AVL InMotion simulator, which is executed in this case on a real-time platform. Further information regarding FlexRayXpert.Lab is available in [3].

The test method relies on the description and storage of the network behavior at different abstraction levels. Therefore, logfiles storing traces of the bus traffic and describing the network behavior with different degrees of accuracy are introduced, and exporters are available for interfacing to the .Lab and .Sim platforms (e.g. for data comparison or replay from a recorded scenario). These methods support both efficient monitoring and replay of network scenarios. Further, methods to generate and / or modify logfiles are available to support efficient generation of complex stimuli. Finally, methods are

[2] www.autosar.org

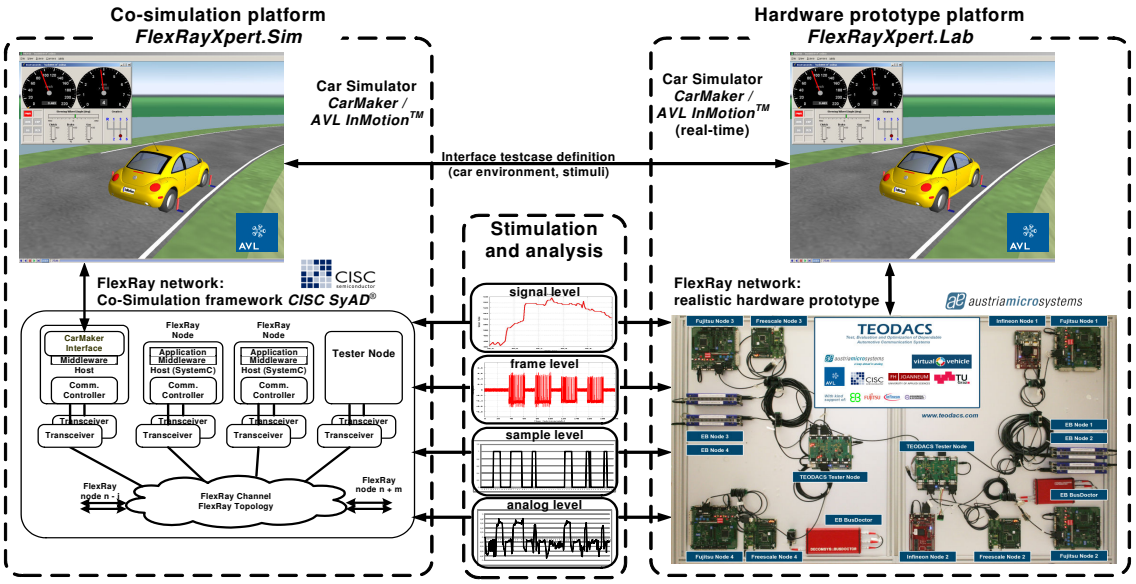


Fig. 1. TEODACS approach

available for the off-line analysis of bus traffic logfiles at different abstraction levels. Additional information regarding the test approach is available in [20].

B. The matrix based test approach

A main challenge for the simulation is the trustfulness of the results: How can it be assured that the real system would present the same reaction. While simulation can be efficiently used for space exploration, the validation of each single model against a hardware counterpart for all possible variants is a tedious task. It means that most of the time a hardware reference of the assembled system for a topology variant under investigation is not available.

We use in this work the tight interfacing between hardware prototype and simulation platform developed during the TEODACS project. Comparison between the two development platforms FlexRayXpert.Sim and .Lab (and therefore validation of the simulation models) is made easy thanks the capability (1) to generate and apply the same stimulus to the system in both platforms, as well as (2) to export a trace of the network's reaction from the two development platforms into a file for further analysis.

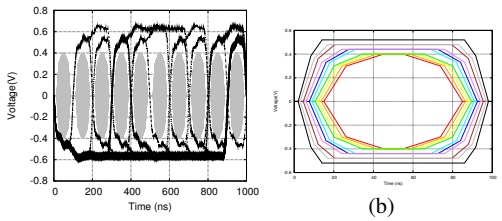
The main idea is to use this dual platform concept to perform space exploration. Using the hardware prototype, a coarse space exploration can be performed (e.g. new node arrangement or cable length modification with large steps). The simulation platform is then used as complement to perform fine-grained exploration (e.g. cable length modification with smaller steps). This combined hardware / simulation space exploration supports validation of the simulated system for different variants (how close are the simulated and the real results) and provides information about the trustfulness of the simulation results (since hardware measurements have been performed on a topology close to the fine-grained simulation).

C. The 10-bit eye diagram

Analyzing signal integrity of serial transmission using eye-diagrams is a well known method implemented in most of high-end oscilloscopes. A classical eye diagram for FlexRay is shown in [15]. Usually, the sender clock is required for triggering bit acquisition. When not available, the zero crossing of each occuring rising edge is detected and triggers the capturing process of the received bit. This way, all bits are drawn into a single diagram, allowing to detect signal integrity violations. The minimum signal requirements are also drawn into the diagram. This enables a fast comparison whether the electrical signal transgresses the specification (too low voltage levels, too slow edges) or not. The main limitation of this approach, however, is to take only one bit into account. Complex bit patterns (series of zeros or ones) and accumulated delays can not be taken into account.

We propose a 10-bit eye diagram tailored for the FlexRay protocol in order to tackle these problems. The idea is to follow the FlexRay bit synchronization which is performed at the beginning of the byte using the rising edge of the "byte start sequence" (BSS), a defined sequence of two bits (zero following a one). In the proposed approach, the FlexRay frame is divided into bytes (ten bits which regroups the BSS and the current byte) that are tested against a series of 10 single eye-diagrams, see Figure 2-a. The single eye-diagrams are centered inside the consecutive bit cells. The other parts of the frame (frame beginning including Transmit Start Sequence

and Frame Start Sequence, as well as end of frame including Frame End Sequence and return to idle) are tested separately.



(a) (b)

Fig. 2. (a) 10-bit eye diagram, (b) eye-diagram based

The advantage of this approach is its ability to take complex multi-bit patterns into account. Hence, all possible bit combinations can be tested (also consecutive zeros or ones) and cumulated timing problems (leading to bit strobing errors) can be taken into account. This includes the capability to measure symmetric and asymmetric delays leading to variation of the bit cell or byte duration (and thus sampling point). Another important benefit of this approach is the frame syntax interpretation for detecting the beginning and end of the frame, which allows to perform additional tests regarding the transition to idle. Our implementation generates both textual and visual reports. Therefore, results can be automatically analyzed and at the same time can be easily interpreted by a test engineer.

An enhancement of this method is to use it as metrics for signal integrity assessment and topology comparison. For that different eye-diagrams references (with different edges and different voltage levels) are defined, see Figure 2-b. For example, one topology variant will satisfy only three out of eight eye diagrams while another topology variant will satisfy five out of eight eye diagrams. It signifies that both topologies are specification conform (since the minimal requirements are satisfied). However, the second variant is more robust (due to the larger distance to the minimal requirements). This method enables automated and objective comparison between topology variants. Further information is available in [21].

IV. Model Validation & Analysis of the Topology

A. System under test

Our approach for model validation was a hierarchical one. The first step was to restrict the verification and validation process to single models, ensuring their proper parameterization and desired behavior. After that, the validated basic blocks were used to build larger composite models that were in turn validated. Finally, complete FlexRay compliant network topologies were built, in order to simulate waveforms within the network.

During early experiments, the electrical bus driver was identified as critical component. Especially the inner resistances of transmitter and receiver modules have a strong impact on the resulting waveforms and digital signal recovery. The transmitter's inner resistance influences the parallel DC load value and characterizes the resulting output voltage levels. The receiver's inner resistance should be as high as possible, in order to avoid any retroactive effects. Further parameters have severe impact on signal integrity validation. It includes signal rise and fall times, size of filter capacitors at BP and BM ports, and voltage multiplication factors, which reduce supply voltage to a given level. We should mention that the overall bus driver has over 50 parameters, so during the model validation process it is very important to select a subset of relevant parameters, where adaptations are necessary or which have the strongest impact on system behavior. For extensive network analysis, an active star composite model was created. It features four branches, where each branch uses the previously introduced bus driver. Its core logic is written in VHDL [22].

The electrical bus cables turned out to be important as well. Two different cable model types that both rely on the RLGC model [9] have been chosen. The first type represents the classical lumped element model. One capacitance, one inductance, one admittance and one resistance are used to describe an arbitrary cable length section (in our case all values are normalized to 1cm per section) and to characterize the line's timing and overall frequency behavior. The second type is a mathematically optimized variant of the RLGC model [10]. However, it does not make use of lumped elements, but of the `'delayed` attribute of VHDL-AMS. The simulation step size is adjustable using a dedicated parameter. The precision and simulation time depend on the number of lumped elements or step size, respectively. As expected, both types deliver similar simulation results while the second model in general computes faster the the first one. The speedup of the optimized model is approximately up to factor 10, and depends heavily on the line's length and step size configuration.

Network termination protects the injected signals from getting reflected at the end of the cable. For this reason, proper termination is necessary and needs to be chosen in accordance with cables and bus drivers. All cables in the FlexRay Xpert.Lab environment were manufactured by Kromberg & Schubert [23]. Their characteristic impedance is given with 100Ω. For proper termination, we relied on the specification's recommendation [24, p.10] and used a split-termination. In our setup, every node received a termination: Either $2 \times 47\Omega$ or $2 \times 24850\Omega$, where the latter basically represents an open end. Together with the split termination, a common mode choke consisting of two coupled inductances was applied to every node, as recommended in the application notes.

B. Test approach

Once various network topologies were set up using the previously described components, we utilized a so called testernode, available in both hardware and software [25], for network stimulation. By using our unique TEODACS sample level log-file format it is possible to stimulate the same network topology in hardware and software with the same input data. The TEODACS concept includes a multi-layer approach, allowing to analyze various aspects and interactions of different levels of abstraction within distributed communi-

cation systems [26]. For this reason, the testernode concept includes all relevant layers of FlexRay communication, from sample layer up to application. By capturing the resulting signals at various interesting points on the network utilizing an export-enabled oscilloscope, and transferring those data right back into our simulation environment, it becomes possible to directly compare the system response from the hardware and simulation platforms.

During the course of the following test campaign, we sought for a common and fast simulating network architecture, suitable for targeted modifications. Our choice fell on a four-node bus line topology, containing various cable lengths. The strategy implied the successive modification of this initial topology while observing the resulting signal integrity at various points on the network. The topology was successively modified above the FlexRay specification in order to intentionally obtain disturbed bits. In this approach, the analysis was performed with the 10-bit eye diagram proposed in Section III-C.

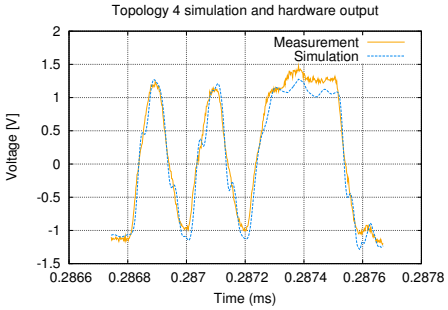


Fig. 3. Measurement and simulation waveform detail

We experienced good correlation between simulated and measured waveforms throughout the experiments. The sample correlation coefficient was usually above 0.98, thus showing a good fitting between simulation and hardware measurement.

The test campaign included the two following parts:

- Termination related issues: An additionally added, redundant split-termination caused the bus voltage to drop significantly in hardware and simulation. The voltage levels were not reached by all transmitters for all bit sequences.
- Reflections: By modification of an open circuit stub line's length, reflection effects were introduced. Depending on the resulting delay of the occurring reflection, the original signal is overlaid to a certain degree with its own copy at receiving nodes. This may lead to unacceptable signal integrity.

The main outcome of this test campaign is the optimization potential behind the FlexRay specification. Hence, the limits are dimensioned for complex worst cases and usually include additional safety margins. When some assumptions can be done (e.g. about the environment, or about the tolerance from specific components), then different topology concepts (faster, lower costs) can be deployed.

C. Matrix-based test campaign

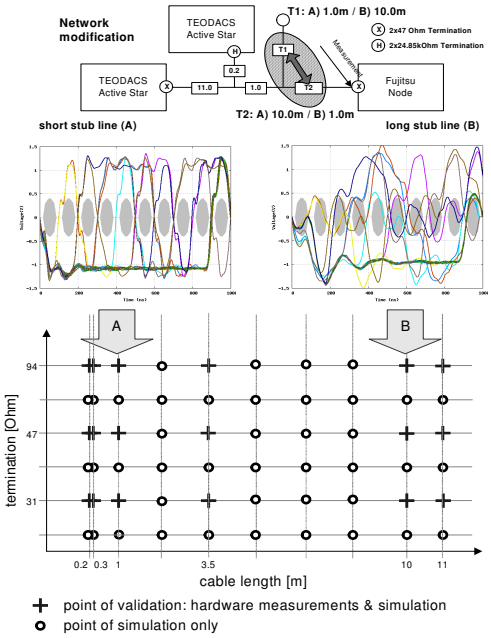


Fig. 4. Sample validation matrix

The matrix-based approach discussed in Section III-B has been use to deal with the resulting model complexity. Figure 4 illustrates our approach, which shows the space exploration for two characteristic parameters, in this case cable length and network termination. During this test campaign, a coarse validation was performed using the hardware platform (plus sign), and enhanced with simulation (circle) for fine-grained analysis. Figure 4 illustrates the distribution between hardware measurements and simulations. The advantage of simulation is the automation capability as well as the fine modification range of the parameters. Hence, modifying a cable length in the simulation takes one click, while on the hardware is requires the assembly of a new cable. The advantage of the hardware measurements is to provide a trustful background for the simulation.

Assuming 30 minutes per simulation run versus one hour to assemble a new cable and perform a new measurement, a test campaign comprising 10 measurements and 30 simulations will take 25 hours instead of 40 hours when performing only hardware measurements. Considering further the automation possibility (simulation campaigns can run over nights or week ends) and the test reproducibility, the advantages of the proposed methods are evident.

The trustfulness of the simulation results are increased, too. Although there is no dedicated hardware validation available for each specific topology variant, we still do expect the simulation to be credible to a certain degree because a given

variant lies around of already validated topology variants. Regarding the evaluation, the eye-diagram based metrics was used to automatically and objectively compare the topology variants and select the best ones.

V. Conclusion

Simulation is a powerful mean for space exploration and thus for supporting design decision at early stage. However, a strong challenge remains the validation of the simulation models and how trustful the simulation results will be (even if a reference hardware is not available yet). We have seen in this work that the efficient interfacing between hardware and co-simulation platform strongly supports the validation process. The two-level space exploration proposed in this work improved results trustfulness while minimizing the exploration efforts. Hence, the simulation models are validated against the hardware for few main topology variants, and the fine-grained space exploration is performed with simulation only. Regarding the result analysis, a 10-bit eye diagram has been introduced. This approach enables the analysis of complex bit patterns (series of zero or ones) as well as the analysis of cumulated timing problems. Further, a method for signal integrity measurement based on the 10-bit eye diagram has been presented. This approach enables the comparison between different topologies and also supports the comparison between different transmission paths within one topology.

Future work will be concerned in defining a method to quantify analytically the lost of trustfulness in the simulation results depending on the modification range of the parameters (e.g. cable length, termination).

Acknowledgement

The authors wish to thank the "COMET K2 Forschungsförderungs-Programm" of the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT), the Austrian Federal Ministry of Economics and Labour (BMWA), Österreichische Forschungsförderungsgesellschaft mbH (FFG), Das Land Steiermark and Steirische Wirtschaftsförderung (SFG) for their financial support.

Additionally we would like to thank the supporting companies and project partners austriamicrosystems, AVL List and CISC Semiconductor as well as Graz University of Technology and the University of Applied Sciences FH Joanneum.

References

[1] *FlexRay Communications System – Protocol Specification Version 2.1 A*, FlexRay Consortium, December 2005. [Online]. Available: http://www.flexray.com

[2] H. Kopetz and G. Bauer, "The Time-Triggered Architecture," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 112 – 126, Jan. 2003.

[3] E. Armengaud, D. Watzenig, M. Karner, C. Steger, R. Weiß, C. Netzberger, M. Kohl, M. Pistauer, F. Pfister, and H. Gall, "Combining the Advantages of Simulation and Prototyping for the Validation of Dependable Communication Architectures: the TEODACS Approach," in *SAE World Congress, 2009-01-0763*, April 2009.

[4] "Flexray Communications Systems – Electrical Physical Layer Specification V2.1 Rev B, available at http://www.flexray.com," FlexRay Consortium, 2005.

[5] T. Gerke and D. Bollati, "An Automated Model Based Design Flow for the Design of Robust FlexRay Networks," *SAE International Journal of Passenger Cars - Electronic and Electrical Systems*, vol. 1, no. 1, pp. 457–466, 2009. [Online]. Available: http://saepcelec.saejournals.org/content/1/1/457.abstract

[6] *FlexRay Transceiver Model*, CISC Semiconductor Design+Consulting GmbH, Klagenfurt, Austria, Feb. 2007. [Online]. Available: http://www.cisc.at/flexray

[7] C. Muller, M. Valle, R. Buzas, and A. Skoupy, "Mixed-Mode Behavioral Model of a FlexRay Physical Layer Transceiver," in *Proceedings of the 19th European Conference on Circuit Theory and Design*, Aug. 2009, p. 4.

[8] *NXP FlexRay Network Simulations: Safeguard the Operation of Your FlexRay Network Architectures*, NXP Semiconductors, The Netherlands, Nov. 2007. [Online]. Available: http://www.nxp.com/acrobat_download/literature/9397/75016200.pdf

[9] H. Johnson and M. Graham, *High-speed Signal Propagation: Advanced Black Magic*. Upper Saddle River, NJ, USA: Prentice Hall Press, 2003.

[10] M. Schlegel, G. Herrmann, and D. Mueller, "Entwicklung eines effizienten VHDL-AMS-Modells der verlustbehafteten Leitung," 2001, 5 pages.

[11] T. Gerke and D. Bollati, "Development of the Physical Layer and Signal Integrity Analysis of FlexRay Design Systems," in *Simulation & Modelling Mechatronics (SP-2111)*, 2007, pp. 665–675.

[12] M. Karner, M. Krammer, S. Krug, E. Armengaud, C. Steger, and R. Weiss, "Heterogeneous co-simulation platform for the efficient analysis of flexray-based automotive distributed embedded systems," in *IEEE International Workshop on Factory Communication Systems (WFCS'10)*, Apr. 2010, accepted for publication, 10 pages.

[13] W. Lawrenz and D. Bollati, "Validation of In-Vehicle-Protocol Network Topologies," in *Second International Conference on Systems 2007 (ICONS '07)*, April 2007, pp. 24–24.

[14] T. Gerke and C. Schanze, "Development and Verification of In-Vehicle Networks in a Virtual Environment," in *In-Vehicle Networks and Software*, 2005, 12 pages.

[15] D. Bollati, "FlexRay - Simulation of Physical Layer Topologies," in *Synopsis User Group Europe 2006*, 2006, 22 pages.

[16] S. Kajtazovic, C. Steger, and M. Pistauer, "A HDL-Independent Modeling Methodology for Heterogeneous System Designs," in *Proceedings of the 2005 IEEE International Behavioral Modeling and Simulation Workshop, 2005 (BMAS 2005)*, 2005, pp. 88–93.

[17] *SyAD Online Documentation*, CISC Semiconductor Design+Consulting GmbH, Klagenfurt, Austria, September 2009. [Online]. Available: http://www.cisc.at/syad

[18] C. Schyr, T. Schaden, and R. Schantl, "New Frontloading Potentials through Coupling of HiL-Simulation and Engine Test Bed," in *FISITA 2008 World Automotive Congress, F2008-12-317*, September 2008.

[19] F. Baronti, P. D'Abramo, M. Knaipp, R. Minixhofer, R. Roncella, R. Saletti, M. Schrems, R. Serventi, and V. Vescoli, "FlexRay Transceiver in a 0.35 μm CMOS High-Voltage Technology," in *DATE '06: Proceedings of the conference on Design, automation and test in Europe*. 3001 Leuven, Belgium, Belgium: European Design and Automation Association, 2006, pp. 201–205.

[20] E. Armengaud, A. Tengg, M. Karner, C. Steger, R. Weiss, and M. Kohl, "Moving Beyond the Component Boundaries for Efficient Test and Diagnosis of Automotive Communication Architectures," in *Fourteenth IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2009), 8 pages*, Sept. 2009.

[21] F. Clazzer, "Design of a Dedicated Software Tool for the Automatic Analog Layer Signal Integrity Analysis of the FlexRay Communication Bus," The Virtual Vehicle Competence Center, Tech. Rep., 2009.

[22] C. Netzberger, "Design document for the active star VHDL model," FH Joanneum Kapfenberg, Tech. Rep., 2008.

[23] *Datenblatt FlexRay Datenleitung, Anwendung im Kabelbaum, Typ FLR09YS-YW*, Kromberg & Schubert, 2007.

[24] "Flexray Communication System - Electrical Physical Layer Application Notes V2.1 Rev B, available at http://www.flexray.com," FlexRay Consortium, 2005.

[25] S. Krug, "Multi-Level Replay and Hardware Interfacing in a FlexRay Network Simulation," Institute of Technical Informatics, Graz University of Technology, Tech. Rep., 2008.

[26] E. Armengaud, A. Steininger, M. Horauer, and R. Pallierer, "A Layer Model for the Systematic Test of Time-Triggered Automotive Communication Systems," in *Proceedings of the 2004 IEEE International Workshop on Factory Communication Systems 2004 (WFCS 2004)*, Sept. 2004, pp. 275–283.

Bibliography

[1] M. Geimer, T. Krueger, and P. Linsel, "Co-Simulation, gekoppelte Simulation oder Simulatorkopplung," *O + P Zeitschrift für Fluidtechnik*, vol. 50, no. 11-12, pp. 572 – 576, 2006.

[2] G. Leen and D. Heffernan, "Expanding automotive electronic systems," *Computer*, vol. 35, pp. 88 –93, Jan. 2002.

[3] C. Ebert and C. Jones, "Embedded software: Facts, figures, and future," *Computer*, vol. 42, pp. 42–52, 2009.

[4] J. Mossinger, "Software in automotive systems," *Software, IEEE*, vol. 27, no. 2, pp. 92 –94, 2010.

[5] P. Hansen, "New S-Class Mercedes: Pioneering Electronics," *The Hansen Report on Automotive Electronics*, vol. 18, pp. 1–2, Oct. 2005.

[6] Volkswagen Aktiengesellschaft, "Distribution of embedded electronic communication systems inside the Volkswagen Phaeton." By courtesy of Volkswagen AG, MultimediaCentrum Fotoservice, 2009.

[7] FlexRay Consortium, *FlexRay Communications System – Protocol Specification Version 2.1 A*, December 2005.

[8] AUTOSAR GbR, *AUTOSAR: Specification of the Virtual Functional Bus*, February 2008.

[9] G. Pelz, *Mechatronic Systems: Modelling and Simulation with HDLs*. Chisester, England: Jon Wiley & Sons Ltd., 2003.

[10] NXP Semiconductors, The Netherlands, *NXP FlexRay network simulations: Safeguard the operation of your FlexRay network architectures*, Nov. 2007.

[11] W. S. Kim, H. A. Kim, J.-H. Ahn, and B. Moon, "System-Level Development and Verification of the FlexRay Communication Controller Model Based on SystemC," *Future Generation Communication and Networking*, vol. 2, pp. 124–127, 2008.

[12] M. Krause, O. Bringmann, A. Hergenhan, G. Tabanoglu, and W. Rosenstiel, "Timing Simulation of Interconnected AUTOSAR Software-Components," in *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE '07*, pp. 1–6, 2007.

[13] A. Hanzlik, "SIDERA - a Simulation Model for Time-Triggered Distributed Real-Time Systems," *International Review on Computers and Software (IRECOS)*, vol. 1, pp. 181–193, Nov. 2006.

[14] Open SystemC Initiative (OSCI), "SystemC." http://www.systemc.org/, 2011. last visited: 06.03.2011.

[15] P. Ashenden, G. Petersion, and D. Teegarden, *The System Designer's Guide to VHDL-AMS*. San Francisco, CA, USA: Morgan Kaufmann Publishers, 2003.

[16] S. Ausejo, A. Suescun, J. Celigueta, E. Moser, and J. Charlot, "A Methodology for Model Interchange and Simulation of Mechatronic Systems," in *Proceedings of the 9th Mediterranean Conference on Control and Automation*, IEEE Control Systems Society, 2001.

[17] Synopsis Inc., "MAST: The Analog, Mixed-Technology and Mixed-Signal HDL for Saber." Synopsis, Inc., Mountain View, CA, USA, http://www.synopsys.com/tools/sld/mechatronics/saber/pages/mast.aspx, 2011.

[18] Modelica Association, "Modelica - A Unified Object-Oriented Language for Physical Systems Modeling Language Specification (Version 3.1)." Modelica Association, Sweden, 2009.

[19] C. Grimm, M. Barnasconi, A. Vachoux, and K. Einwich, "An Introduction to Modeling Embedded Analog/Mixed-Signal Systems using SystemC AMS Extensions." Open SystemC Initiative (OSCI), San Francisco, CA, USA, June 2008.

[20] T. Gerke and R. DeMeis, "The Virtual Vehicle: Part 1: In-Vehicle Network Simulation and Analysis." Automotive DesignLine, United Business Media, London, United Kingdom, http://www.automotivedesignline.com/, January 2009.

[21] T. Gerke and R. DeMeis, "The Virtual Vehicle: Part 2: Early Validation of Vehicle Electrical Systems and Power Management." Automotive DesignLine, United Business Media, London, United Kingdom, http://www.automotivedesignline.com/, February 2009.

[22] T. Gerke, S. Lagerqvist, and R. DeMeis, "The Virtual Vehicle: Part 3: Developing Robust Wiring Harnesses." Automotive DesignLine, United Business Media, London, United Kingdom, http://www.automotivedesignline.com/, March 2009.

[23] T. Gerke and F. Lehmann, "The Virtual Vehicle: Part 4: Automotive Design and Optimization of Electrical Fuel Injection Systems." Automotive DesignLine, United Business Media, London, United Kingdom, http://www.automotivedesignline.com/, April 2009.

[24] F. Bouchhima, M. Briere, G. Nicolescu, M. Abid, and E. Aboulhamid, "A systemc/simulink co-simulation framework for continuous/discrete-events simulation," in *Behavioral Modeling and Simulation Workshop, Proceedings of the 2006 IEEE International*, pp. 1 –6, 2006.

[25] S. Kajtazovic, C. Steger, and M. Pistauer, "A HDL-independent modeling methodology for heterogeneous system designs," in *Behavioral Modeling and Simulation Workshop, 2005. BMAS 2005. Proceedings of the 2005 IEEE International*, pp. 88–93, 2005.

[26] M. Benedikt, H. Stippel, and D. Watzenig, "An adaptive coupling methodology for fast time-domain distributed heterogeneous co-simulation," in *Reliability and Robust Design in Automotive Engineering 2010 (SP-2272)*, 2010.

[27] M. Karner, C. Steger, R. Weiss, E. Armengaud, D. Watzenig, and G. Knoll, "Verification and analysis of dependable automotive communication systems based on hw/sw co-simulation," in *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*, pp. 444 –447, 2008.

[28] M. Karner, E. Armengaud, C. Steger, and R. Weiss, "Run-Time Co-Simulation Model Switching for Efficient Analysis of Embedded Systems," *International Journal of Embedded Systems (submitted for publication)*, 2011.

[29] M. Karner, C. Steger, R. Weiss, and E. Armengaud, "Optimizing hw/sw co-simulation based on run-time model switching," in *Specification Design Languages, 2009. FDL 2009. Forum on*, pp. 1 –6, 2009.

[30] M. Karner, E. Armengaud, C. Steger, and R. Weiss, "Holistic simulation of flexray networks by using run-time model switching," in *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '10, (3001 Leuven, Belgium, Belgium), pp. 544–549, European Design and Automation Association, 2010.

[31] M. Karner, E. Armengaud, and C. Steger, "Verfahren zum Umschalten von heterogenen Simulationsmodellen zur Laufzeit." EP 2299376, published 2011-03-23.

[32] M. Karner, E. Armengaud, C. Steger, R. Weiss, M. Pistauer, and F. Pfister, "A Cross-Domain Co-Simulation Platform for the Efficient Analysis of Mechatronic Systems," in *SAE World Congress & Exhibition, 2010. SAE '10*, p. 14, April 2010.

[33] M. Karner, M. Krammer, S. Krug, E. Armengaud, C. Steger, and R. Weiss, "Heterogeneous co-simulation platform for the efficient analysis of flexray-based automotive distributed embedded systems," in *Factory Communication Systems (WFCS), 2010 8th IEEE International Workshop on*, pp. 231 –240, May 2010.

[34] M. Krammer, F. Clazzer, E. Armengaud, M. Karner, C. Steger, and R. Weiss, "Exploration of the flexray signal integrity using a combined prototyping and simulation approach," in *Design and Diagnostics of Electronic Circuits and Systems (DDECS), 2010 IEEE 13th International Symposium on*, pp. 111 –116, 2010.

[35] M. Rausch, *FlexRay – Grundlagen Funktionsweise Anwendung.* Carl Hanser Verlag München Wien, 2008.

[36] O. Kindel and M. Friedrich, *Softwareentwicklung mit AUTOSAR: Grundlagen, Engineering, Management in der Praxis.* Heidelberg: dpunkt, 2009.

[37] E. Armengaud, A. Steininger, and M. Horauer, "Automatic parameter identification in flexray based automotive communication networks," in *Emerging Technologies and Factory Automation, 2006. ETFA '06. IEEE Conference on*, pp. 897 –904, 2006.

[38] V. Lari, M. Dehbashi, S. G. Miremadi, and N. Farazmand, "Assessment of Message Missing Failures in FlexRay-Based Networks," in *Dependable Computing, 2007. PRDC 2007. 13th Pacific Rim International Symposium on*, pp. 191–194, 2007.

[39] CISC Semiconductor Design+Consulting GmbH, Klagenfurt, Austria, *FlexRay Transceiver Model*, Feb. 2007.

[40] C. Muller, M. Valle, W. Prodanov, and R. Buzas, "A systematic development methodology for mixed-mode behavioral models of in-vehicle embedded electronic systems," *EURASIP J. Embedded Syst.*, vol. 2010, pp. 4:2–4:2, January 2010.

[41] R. Buchmann, M. Cartron, and B. Yannick, "Transaction-based modeling for large scale simulations of heterogeneous systems," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, Simutools '09, (ICST, Brussels, Belgium, Belgium), pp. 33:1–33:2, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.

[42] M. Cheikhwafa, S. Le Nours, O. Pasquier, and J. Calvez, "Transaction level modeling of a flexray communication network," in *Specification Design Languages, 2009. FDL 2009. Forum on*, pp. 1 –4, 2009.

[43] A. Albert, B. Pietsch, and F. Voetz, "Simulation Environment for Investigating the Impacts of Time-Triggered Communication on a Distributed Vehicle Dynamics Control System," in *ECRTS Workshop on Real-Time and Control, 2005. RTC 2005. Proceedings of the 1st International*, pp. 1–6, 2005.

[44] R. Obermaisser and M. Schlager, "A simulation framework for virtual integration of integrated systems," in *EUROCON, 2007. The International Conference on Computer as a Tool*, pp. 2208 –2216, 2007.

[45] T. Gerke and D. Bollati, "Development of the Physical Layer and Signal Integrity Analysis of FlexRay Design Systems," in *Simulation & Modelling Mechatronics (SP-2111)*, 2007.

[46] Elektrobit Corporation, Oulunsalo, Finland, *FlexRay Rest Bus Simulation Solution: EB tresos Busmirror*, 2011.

[47] dSPACE GmbH, Paderborn, Germany, *dSPACE FlexRay Configuration Package*, Feb. 2011.

[48] *CANoe.FlexRay, url = http://www.vector.com, publisher = Vector Informatik GmbH, organization = Vector Informatik GmbH, month = March, year = 2011.*

[49] S. Yoo and A. Jerraya, "Hardware/software cosimulation from interface perspective," *Computers and Digital Techniques, IEE Proceedings* -, vol. 152, no. 3, pp. 369–379, 2005.

[50] P. Le Marrec, C. Valderrama, F. Hessel, A. Jerraya, M. Attia, and O. Cayrol, "Hardware, software and mechanical cosimulation for automotive applications," in *Rapid System Prototyping, 1998. Proceedings. 1998 Ninth International Workshop on*, pp. 202 –206, June 1998.

[51] P. Birrer and W. Hartong, "Incorporating SystemC in Analog/Mixed-Signal Design Flow," in *Forum on specification and Design Languages, FDL 2005, September 27-30, 2005, Lausanne, Switzerland, Proceedings*, pp. 173–178, 2005.

[52] CISC Semiconductor Design+Consulting GmbH, *SyAD Online Documentation.* February 2011.

[53] J. Harkin, T. M. McGinnity, and L. P. Maguire, "Genetic algorithm driven hardware-software partitioning for dynamically reconfigurable embedded systems," *Microprocessors and Microsystems*, vol. 25, no. 5, pp. 263 – 274, 2001.

[54] M. Purnaprajna, M. Reformat, and W. Pedrycz, "Genetic algorithms for hardware-software partitioning and optimal resource allocation," *J. Syst. Archit.*, vol. 53, pp. 339–354, July 2007.

[55] A. Bragagnini, F. Fummi, A. Huebner, G. Perbellini, and D. Quaglia, "Co-simulation framework for the angel platform," in *Electronics, Circuits and Systems, 2007. ICECS 2007. 14th IEEE International Conference on*, pp. 629 –632, 2007.

[56] N. Bombieri, F. Fummi, and D. Quaglia, "System/network design-space exploration based on tlm for networked embedded systems," *ACM Trans. Embed. Comput. Syst.*, vol. 9, pp. 37:1–37:32, April 2010.

[57] F. Fummi, M. Poncino, S. Martini, F. Ricciato, G. Perbellini, and M. Turolla, "Heterogeneous co-simulation of networked embedded systems," in *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, vol. 3, pp. 168 – 173 Vol.3, 2004.

[58] G. Beltrame, D. Sciuto, and C. Silvano, "Multi-Accuracy power and performance Transaction-Level modeling," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, no. 10, pp. 1830–1842, 2007.

[59] R. Claes and T. Holvoet, "Multi-model traffic microsimulations," in *Winter Simulation Conference (WSC), Proceedings of the 2009*, pp. 1113 –1123, 2009.

[60] D. Rao and P. Wilsey, "Applying parallel, dynamic-resolution simulations to accelerate vlsi power estimation," in *Simulation Conference, 2006. WSC 06. Proceedings of the Winter*, pp. 694 –702, 2006.

[61] B. Morin, O. Barais, J.-M. Jezequel, F. Fleurey, and A. Solberg, "Models@ run.time to support dynamic adaptation," *Computer*, vol. 42, no. 10, pp. 44 –51, 2009.

[62] R. Khaligh and M. Radetzki, "A Dynamic Load Balancing Method for Parallel Simulation of Accuracy Adaptive TLMs," in *Forum on Specification and Design Languages. FDL '10*, 2010.

[63] R. Salimi Khaligh and M. Radetzki, "Adaptive interconnect models for transaction-level simulation," in *Languages for Embedded Systems and their Applications* (M. Radetzki, ed.), vol. 36 of *Lecture Notes in Electrical Engineering*, pp. 149–165, Springer Netherlands, 2009.

[64] R. Salimi-Khaligh and M. Radetzki, "A latency, preemption and data transfer accurate adaptive transaction level model for efficient simulation of pipelined buses," in *Specification, Verification and Design Languages, 2008. FDL 2008. Forum on*, pp. 37 –42, 2008.

[65] R. Salimi Khaligh and M. Radetzki, "Modeling constructs and kernel for parallel simulation of accuracy adaptive tlms," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, pp. 1183 –1188, 2010.

[66] M. Radetzki and R. Khaligh, "Accuracy-Adaptive simulation of transaction level models," in *Design, Automation and Test in Europe, 2008. DATE '08*, pp. 788–791, 2008.

[67] K. Hines and G. Borriello, "Selective focus as a means of improving geographically distributed embedded system co-simulation," in *Rapid System Prototyping, 1997. ' Shortening the Path from Specification to Prototype'. Proceedings., 8th IEEE International Workshop on*, pp. 58–62, 1997.

[68] K. Hines and G. Borriello, "Dynamic communication models in embedded system Co-Simulation," in *Design Automation Conference, 1997. Proceedings of the 34th*, pp. 395–400, 1997.