Masterarbeit

FlexRay Physical Layer Simulation and Validation Based on Active Star Networks

Martin Krammer

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

Begutachter: Ass.-Prof.Dipl.-Ing.Dr.techn. Christian Steger

Betreuer: Ass.-Prof.Dipl.-Ing.Dr.techn. Christian Steger
Dipl.-Ing.Michael Karner

Graz, Februar 2010

Kurzfassung

Das Bordnetzwerk eines modernen Fahrzeugs stellt die Interaktion der verschiedensten elektronischen Steuergeräte sicher und implementiert dadurch ein verteiltes Kommunikationssystem. Auf Grund zahlreicher Einflüsse kann dieses Kommunikationssystem jedoch nicht als vollständig frei von Fehlern betrachtet werden. FlexRay™ ist ein neuer Standard für automotive Kommunikationsnetzwerke, welcher jene Probleme lösen soll, die während der letzten Jahre im Bereich der Automobilentwicklung aufgetaucht sind.

Die komplexen Strukturen und Architekturen dieser automotiven Kommunikationsnetzwerke verlangen nach neuen Möglichkeiten für Entwicklung, Design, Integration und Test solcher Systeme. FlexRay™ vollzieht dabei einen Paradigmenwechsel, weg von einer ereignisgesteuerten und hin zu einer zeitgesteuerten Kommunikationsarchitektur. Dabei müssen harte Echtzeit-Anforderungen erfüllt werden. Ein derartiges System wird als höchst sicherheitskritisch betrachtet, entsprechende Maßnahmen begleiten daher oftmals den gesamten Entwicklungsprozess. Simulation als Werkzeug hilft, auftretende Probleme bereits so früh wie möglich erkennen zu können und gleichzeitig Aufwand, Entwicklungszeit sowie Kosten auf einem akzeptablen Niveau zu halten.

Im Kontext des TEODACS Projektes, einer Kooperation zwischen dem Kompetenzzentrum *Das Virtuelle Fahrzeug* und der Technischen Universität Graz, Institut für Technische Informatik, wurde die Simulation der elektrisch-physikalischen Ebene des FlexRay™ Kommunikationssystems fertiggestellt. Diese Arbeit beschreibt die Entwicklung von neuen, sowie die Verbesserung von bereits vorhandenen Modellen, und geht dabei im Besonderen auf den aktiven Sternkoppler eines FlexRay™ Netzwerkes ein. Auf Grund der Zielarchitektur von TEODACS sind sämtliche Stimuli und Ergebnisse vollständig zwischen der Simulationsumgebung und dem Labor-Prototypen austauschbar. Ein Schichten-System und dessen Schnittstellen ermöglicht die effiziente Generierung von Stimuli sowie die Verifikation und Validierung der resultierenden Gesamtsysteme. Mit diesem gesamtheitlichen Ansatz werden mehrere relevante Netzwerktopologien untersucht und deren Ergebnisse einer exemplarischen Signalintegritätsanalyse unterzogen.

Abstract

Modern vehicles are equipped with distributed communication networks to enable the interaction between various electronic control units. Due to the large number of possible interferences, the overall communication system cannot assumed to be completely fault-free. FlexRay™ is a new standard for automotive communication networks, addressing many problems which arose over the last few years of automotive developments.

The complex structure and architecture of such automotive communication networks demands new possibilities for development, design, integration and test. FlexRay™ introduces a paradigm change from event-triggered to time-triggered communication architectures. Hard real-time requirements need to be observed. Since such a system is considered to be highly safety-critical, additional safety requirements go along with the entire development process. Simulation helps to detect possible problems as early as possible, and keep efforts, development time and cost at an acceptable level.

In context of the TEODACS project, a joint cooperation of the Virtual Vehicle Competence Center and Graz University of Technology, Institute for Technical Informatics, the FlexRay™ electrical physical layer simulation and validation was completed. This thesis describes the development of new, and enhancement of various present models, with special focus on FlexRay™'s active star. Due to the target architecture of TEODACS, all occurring stimuli and results are completely interchangeable between the simulation environment and the laboratory prototype, on a number of layers. These layers and their defined interfaces are utilized for efficient stimulus generation, verification and validation of the resulting system. This holistic approach is used to investigate several relevant network topologies. These simulation results are used for an exemplary signal integrity analysis.

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)

Acknowledgement

First of all, I want to thank my advisor Michael Karner for his great helpfulness and support. His scientific compentence and social skills were of great value in order to reach the high aims of this thesis. I would also like to thank my supervisor Christian Steger, which is always open for his students' concerns.

Special thanks go to the researchers of the *Virtual Vehicle Competence Center*, in particular Eric Armengaud and Allan Tengg. It was a pleasure to experience their enthusiasm on the topics of automotive communication systems and beyond. However, I am thankful to all other employees of *Virtual Vehicle*'s *Area E* as well, because of their lovely acceptance.

Martin Kohl of *FH Joanneum Kapfenberg* and Harald Gall of *austriamicrosystems AG* deserve thanks for their friendly support. Thanks go to CISC *Semiconductor Design+Consulting GmbH* as well for their valuable assistance.

This thesis would not have been possible without the enduring support of my parents Sylvia and Karl, throughout my whole studies. They always provided me proper conditions to graduate from university. I also want to thank Bettina for her understanding during these times.

I also consider this to be an appropriate place to say thanks to a number of fellow students. It was a pleasure for me, to walk this part of my life's path together with you. I really enjoyed the discussions and studies with you! Thanks to Günther, Hannes, Johannes, Michael, Robert and Thomas.

Finally, I want to thank everyone else involved in my thesis and preceding studies. At the same time, I am asking for your apology, it would be impossible to mention every single one.

Graz, January 2010 Martin Krammer

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

1.1 Thesis Environment

Modern cars are much more than simple means of individual transportation. Sophisticated electronics and electrics offer a high level of comfort, increased safety and better environmental sustainability. Due to the close relation of such embedded systems to the automotive business, this thesis was arranged in connection with "The Virtual Vehicle Competence Center" Graz[1]. The Austrian COMET K2 research program is an international institution in the field of application-oriented vehicle development. It is organized in five research areas, dealing with system design and optimization, thermodynamics, noise, vibration, harshness and friction, mechanics and vehicle electrics/electronics and software.

This work was done in cooperation with *Virtual Vehicle*'s "Area E", *Vehicle Electrics/Electronics and Software*. The project acting as host is named TEODACS, which is an abbreviation for "Test, Evaluation and Optimization of Dependable Automotive Communication Systems". The aim of the TEODACS research project is to gather expertise for the deployment and validation of FlexRay™ based distributed systems [Cen09]. Special attention is paid to the exploration and interaction of communication services and layers. Internal and external mechanisms influencing communications are subject to efficient and realistic modeling within the simulator and laboratory setup. Therefore, three project branches are involved:

FlexRay™ Xpert.Research: Development of new concepts and methods for efficient test, evaluation and optimization of automotive communication systems.

FlexRay™ Xpert.Lab: Prototype development for evaluation and validation of communication architectures and simulation models. Simulator and prototype both require defined interfaces between them.

FlexRay™ Xpert.Sim: The overall goal of this branch is to develop a simulator for the entire communication system at different abstraction levels. This thesis belongs to that branch.

The TEODACS project environment was set up in cooperation with a number of Austrian industry partners. With *austriamicrosystems AG* a widely known semiconductor manufacturer joined the project, their focus is on automotive integrated circuits. Another project partner in the area of semiconductor design and consulting is *CISC Semiconductor Design+Consulting GmbH*, based in Klagenfurt, Austria. CISC has competences in the areas of system design, modeling, simulation, verification and optimization of heterogeneous embedded microelectronic systems. Finally, *AVL List GmbH* rounds off the board of industry partners. AVL is the world's largest privately owned and independent company for the development of power train systems with internal combustion engines as well as instrumentation and test

[1] See `http://www.virtualvehicle.at` for more information.

systems. AVL also develops simulation methods, necessary for development work. Besides *The Virtual Vehicle Competence Center* and the *Institute of Technical Informatics, Graz University of Technology*, the *University of Applied Sciences FH Joanneum Kapfenberg* completes the board of scientific partners in the TEODACS project.

1.2 Motivation and Problem

As already touched in the previous section, distributed automotive communication systems are somewhat difficult to handle over the entire product life cycle. The following requirements are made to such systems. The development process should be well-defined, understandable, and should allow the easy generation of test cases. Ideally, the processes of verification and validation cover all layers involved. The verification process follows closely to all related standards and specifications. Validation techniques, on the other hand, need to be specified individually according to the engineer's situation. All these measures share a common set of objectives: The resulting system should be highly available and stable under all circumstances possible. External and internal sources are influencing the overall system: Temperature variability, EMI, EMC or simply deterioration are most common causes. Furthermore, the system should deliver reliable results at any time and within a fixed time limit. The latter is called a real-time requirement. Various industry standards (IEC 61508, ISO 26262 for example) are dealing with the definition of safety, many additional requirements are derived thereof. Furthermore, from an economic point of view, an automotive communication system needs to be scalable with respect to different vehicle variants. Additionally, fault and error diagnosis systems are necessary to ensure maintainability throughout the whole product life cycle.

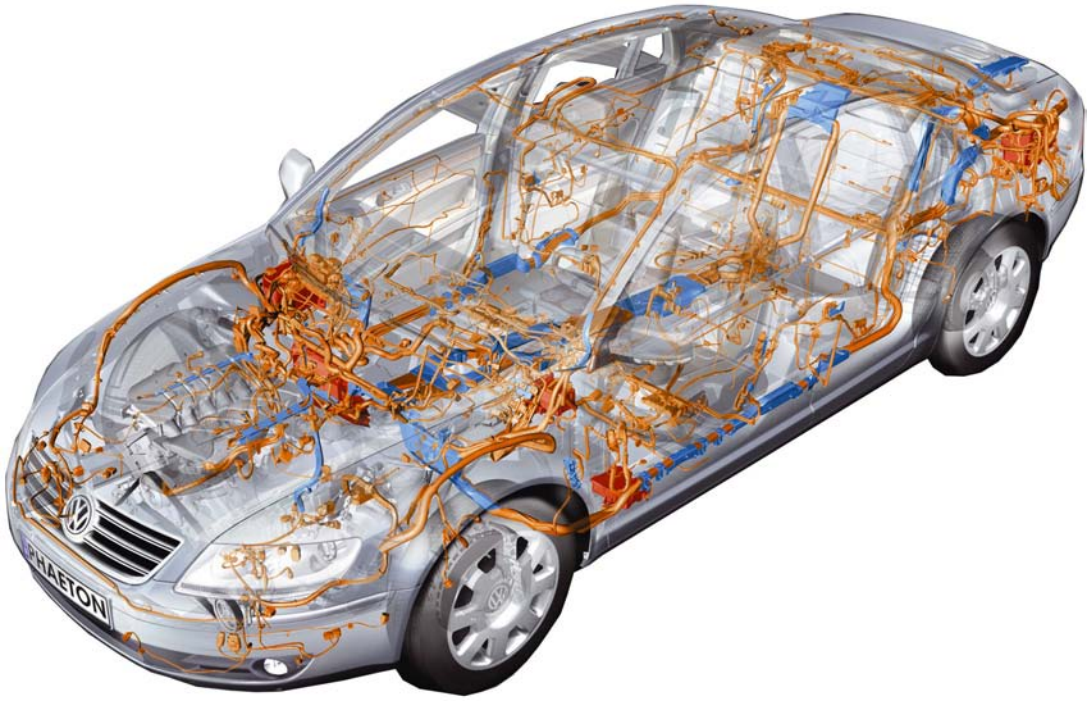

Figure 1.1: Distribution of embedded electronic communication systems inside the Volkswagen Phaeton [Akt09].

Leaps ahead in technology and therefore constantly growing requirements lead to more complex overall systems. This fact requests new methods and strategies. Figure 1.1 shows the distribution of an embedded electronic communication system within a typical upper class car. This way, a total cable length of up to 4 kilometers and a total cable harness weight of up to 64 kilograms is achieved. Around

70 ECUs are necessary to connect each sensor or actuator to the network. Cost is *the* overall criteria in the automotive industry, so new technologies helping to reduce cost while keeping up with latest requirements are desperately sought after. [Vas04]

The TEODACS project [AWS+08] strives for a "New Vision for Testing Dependable Automotive Communication Systems". The project's vision is illustrated using a V-diagram, as shown in figure 1.2. One of the main goals of TEODACS is the establishment of integrated interfaces between the left and right hand side of the V-diagram: The left hand side represents a co-simulation framework, whereas the right hand side represents a laboratory setup. Furthermore, both sides are made up of several layers, which allow a concrete definition of the previously mentioned interfaces.

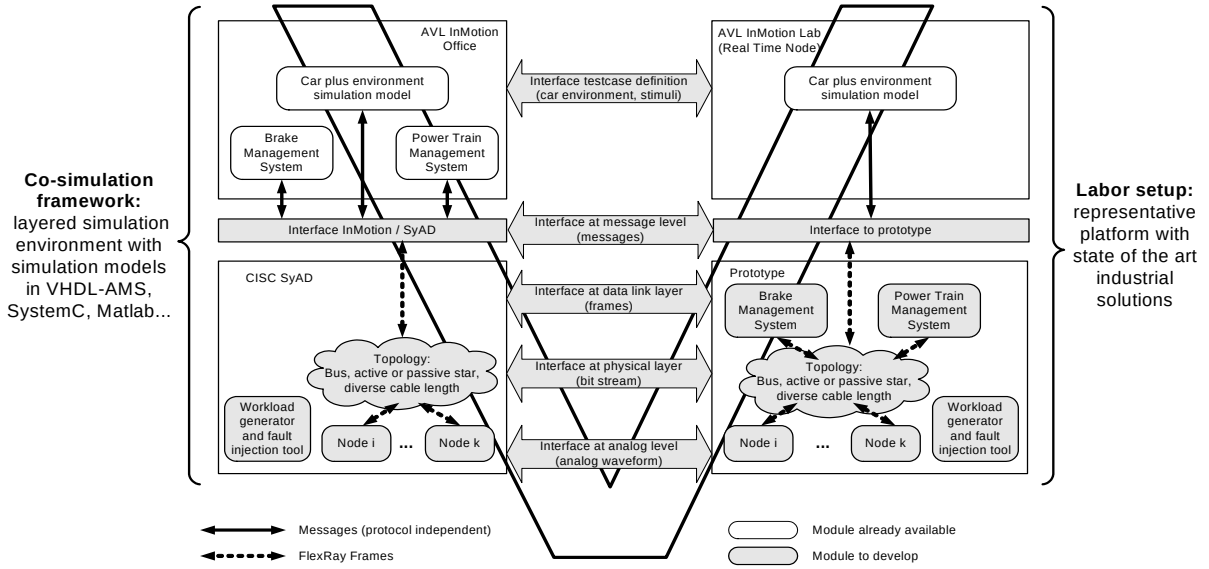

Figure 1.2: TEODACS vision [AWS+08].

As described in the upcoming section, the goal of this thesis is to add necessary components to the TEODACS V-diagram at the lower left-hand side. This means the completion of required models and interfaces at the co-simulation platform, but also the verification and validation with the corresponding laboratory setup, exploiting the proposed layer model.

1.3 Objectives

This section introduces the main objectives of this thesis. Superior to all other goals is research in the fields of automotive co-simulation, simulation of automotive components, fault-injection and -investigation in the automotive area, and finally research of factors which have strong impacts on automotive communication networks. Within these overall objectives, the following tasks and work packages were supposed to accomplish.

First of all, a complete and simulatable active star model should be created. Within the TEODACS project, a number of FlexRay™ network component models is already available, but this network-extending component is still missing. An existing active star digital logic model written in VHDL should be used together with other (analog) models to create a functional model of an active star, compliant to FlexRay™ specifications. A demonstrator should be used to verify the correctness of the resulting overall model. In order to accomplish this task, the presence of a bus driver model is necessary. Therefore, the existing analog bus driver intellectual-property (IP) model written in VHDL-AMS by CISC is subject to integration into the simulation environment. That bus driver model allows the inspection of the analog layer, necessary for signal integrity analysis for example.

Additionally, there is demand for efficient model configuration. This has various reasons. First of all, the effects of differently parameterized components should be investigated. Second, this is necessary to validate a specific component against its hardware counterpart and emulate its behavior. All models should allow the simplest possible methods for parameterization and should integrate seamless into the TEODACS project environment. Moreover, a fault injection possibility should be provided. This means that comprehensive options should be available to modify system inputs and system internal variables, together with the possibilities to observe the effects of these influences. This is aimed at components on network and logic layers and allows the identification of critical points within the communication system.

A specific requirement is the use of CISC's "System Architect Designer" [sya09, KSSP06] co-simulation software platform. All available models should be ported to SyAD®. Additionally, SyAD® compatibility has to be provided throughout the entire TEODACS project and in particular within this thesis, to ensure an integrated co-simulation environment.

Basic methods for test, verification and validation of various network component models should be developed. A FlexRay™ network contains many active and passive elements. Their models were partially created earlier within the TEODACS project. The existing and all newly developed elements should be evaluated carefully in order to ensure sustained reliability of the TEODACS simulation framework.

Different appropriate network topologies should be identified in order to generate efficient test cases for the verification and validation processes.

FlexRay™'s specifications include rules to obey when building network topologies. However, a valid network configuration is no guarantee to achieve a working network. That is why FlexRay™ gives a lot of recommendations concerning topologies. Based on those assumptions, the overall problem of signal integrity analysis is subject to investigation. Another thesis written in parallel to this one at ***The Virtual Vehicle Competence Center*** deals with the detection, interpretation and presentation of signal integrity relevant data. The results of that work will most likely be helpful for automated signal integrity analysis. The shift of signal integrity along with modifications of network topology should be explored as well.

1.4 Overview of Chapters

Chapter 2 introduces related work on the topics of automotive communication systems, modeling and simulation, verification and validation as well as selected topics on FlexRay™. Chapter 3 outlines the general approach and nominates software packages and tools to be used throughout the work. The most important components, models and properties of a typical FlexRay™ network are also given in this chapter. Chapter 4 includes a complete description of all implemented models and scripts. Finally, chapter 5 provides a thorough view on significant results achieved during the work on this thesis. Chapter 6 closes this document, summarizing the gained knowledge and pointing out some aspects for possible future work.

Chapter 2

State of the Art in Modeling and Automotive Engineering

This chapter introduces literature, which is considered relevant for the accomplishment of this thesis. First, publications from the area of modeling and simulation are covered in section 2.1, including a short historical outline in section 2.1.1. Section 2.1.2 is dealing with modeling rules and recommendations. Some examples of modeling automotive systems and applications are recited as well in section 2.1.3. Basics of model verification and validation are introduced in section 2.3.3. Some selected topics concerning the FlexRay™ communication system are given in section 2.3. This includes modeling of FlexRay™ components (section 2.3.2), verification and validation of FlexRay™ bus systems (section 2.3.3) and FlexRay™ signal integrity analysis (section 2.3.5).

2.1 Modeling and Simulation

2.1.1 The Early Days of VHDL-AMS

The first draft language reference manual for VHDL-AMS was accepted by the IEEE ballot in October 1997. Besides the well-known electronic domain, VHDL-AMS suddenly offered the possibility to create new models in other areas of technology. Especially in automotive engineering, where hydraulic and mechanical components, together with their electronic control units, play important roles in the area of modeling and simulation. VHDL-AMS acted as a unified modeling language for the first time. [MM98]

From a mathematical point of view, physical components can be described using differential equations. In the past of 1997, all of the most commonly known languages did not support such an approach naturally. At that time and today, the introduction of new tools is expensive: Selection, licensing, training and support engulfs money in no time. Additionally, the model exchange between a company's internal departments and external suppliers is posing another problem: Besides technology related issues, time and financial constraints have to be met. VHDL-AMS is able to fulfill almost all of the discussed requirements. It has a mathematical orientation and allows the description of differential algebraic equations. Since VHDL established a widely accepted standard in the electronic domain in 1987, VHDL-AMS lives on the good reputation of VHDL. The MSR consortium[1] was founded these days in order to find some unified methods for the easy exchange of product specifications. Based on a study, their choice fell on VHDL-AMS.

The primary problem in the area of unified modeling in continuous domains is the concept of discontinuities or the detection of non-convergence. Proprietary language extensions used questionable constructs to overcome those problems. However, in most situations they worked out. VHDL-AMS was the first language to address those difficulties in a native way, by issuing the break-statement as well as non-conservative communication links.

[1] See `http://www.msr-wg.de`, October 2009, for further information.

VHDL-AMS unified digital and analog electronics for the first time, including system design aspects. The concept of module development with well-defined interfaces simplified the design of system architectures and communication between modules. The availability of co-simulation environments has increased today, so the need for manual coupling of simulator tools is mostly avoided. VHDL-AMS covers all missing concepts for multi-domain modeling and unified simulation. Most important for the automotive industry, VHDL-AMS supports the electronic and hydraulic domain, as well as multi-body systems and system-level model generation. Mixed-domain simulators are available today, to remedy the deficiencies from the early days of VHDL-AMS. From the authors point of view, all their wishes and requirements are entirely fulfilled today.

A few basic examples demonstrating the modeling of electronic and mechanical systems are shown in [HD04]. Considering an electronic system comprising discrete elements like voltage sources, capacitors, inductances and resistors, a number of basic component equations are set up. Depending on these specific components, the resulting equations may contain a derivation after time, leading to differential equations. All equations are simplified and the contained terms are sorted by their corresponding state variables. This way, a mathematical system is achieved, ready for simulation and analysis of the assumed circuit. VHDL-AMS is natively able to deal with such component equations, leaving it to the engineer to describe the system's structure.

Those fundamental features make VHDL-AMS an appropriate tool for the development of automotive components. This was recognized by many car makers and suppliers. It has shown that VHDL-AMS is able to fill the gap for a unified modeling language very well [Mos96].

In the early 1990s VHDL-AMS became interesting for the automotive industry. An inspection of typical VHDL-AMS examples was published by [Mos96]. The MSR consortium, founded by german car makers and suppliers, powered investigations to find a set of minimal necessary capabilities of unified methods to exchange product specifications and models. To derive language requirements, all participating companies and individuals were asked to provide possible sample candidates. However, the feedback they received was insufficient, due to strong legal model restrictions, high complexity of models, and poor documentation standards. Despite those circumstances, a number of requirements for a unified modeling language was derived. Following a short excerpt of those requests as mentioned in [Mos96].

- Structural decomposition: Decomposed model parts should be independent entities. The reasons are obvious: First, understanding of models is important. Second, partitioning increases the reusability of decomposed parts.
- Discrete modeling: Modeling of digital circuits and event driven systems require appropriate mechanisms, where changes only happen at discrete points in time. System state changes only happen at discrete time points, so it is reasonable to request an event-driven mechanism. VHDL already supports event-driven and discrete-time modeling.
- Continuous modeling: The behavior of any mathematically described physical system leads to differential algebraic equations. Given initial conditions, equations describe values of each involved state variable at any time after the initial setting.
- Discontinuities: Physically complicated behavior can be described in an abstract way.
- Interaction between discrete and continuous: Describing mixed systems, containing analog and digital variables, the difference of simulated behavior on different simulators should be within some defined margins.
- Network: The language should offer the capability to model energy-flows or conservative communication facilities. For example, the simulator needs to be aware of Kirchhoff's law.

- Block diagram: To illustrate the cause and effect between different blocks, signal-flows are used. Most control system designers prefer signal-flow diagrams, since directed communication facilitates the understanding of cause and effect within systems.
- Piece-wise linear: Many non-linear properties of physical systems are given as characteristic tables.
- Programming features: To support well-known design patterns, the use of functions and self-defined data types should be possible.
- Schematic view: Graphical representations exist in almost every domain. They are useful for documentation and general model understanding. VHDL-AMS however, does not support such a graphical representation in a native way.

The MSR study group compared VHDL-AMS and other candidate languages for modeling and simulation with respect to the given set of features above. VHDL-AMS supports block oriented and component modeling, has support for discontinuities and already supports event-driven or discrete time modeling. The study group also emphasized the efficiency of VHDL-AMS and noted that it is easy to learn because of its user driven approach.

The only requirement VHDL-AMS completely misses, is a graphical representation or schematic view. However, most development environments used today offer at least the capability of block-wise graphical design. VHDL-AMS does not support the definition of over-determined systems, since those might be ambiguous.

All in all, VHDL-AMS supports almost all of the given requirements. Problems may arise when working with over-determined systems of differential algebraic equations, for example when dealing with multi-body-systems. Additionally, an example model is given, comprising a simple, discrete controller. This controller interacts with an electric driver using a signal-flow mechanism. The electric driver is connected with a plant, consisting of a motor, transmission and load. The author shows that modeling and simulation of the given system is possible using VHDL-AMS, despite the implicitly present discontinuities.

2.1.2 Design Space Exploration

VHDL-AMS offers a wide range of modeling possibilities. Despite the fact, that a model is syntactical correct, there may still arise some problems: The resulting system of differential equations may have zero or infinite solutions. Todays simulators and compilers give hints to the modeling engineer, pointing at critical code sections. Early products, after the introduction of VHDL-AMS as an international standard in 1999, had a not so sophisticated feature set, making it hard for the developer to find the cause for these problems.

As a reaction to this difficulties, [Haa03] proposed a number of possible problems and derived a set of rules to overcome those issues. Three different kinds of problems were identified:

- Insufficient considerations of solutions properties.
- Insufficient considerations of VHDL-AMS standards.
- Inadequate considerations of simulator specifics.

Based on mathematical system descriptions, the author shows the solvability of differential algebraic equations using simple matrix operations. First of all, initial values are analyzed. Each simulator needs to calculate initial values before running the entire simulation in order to get correct results. After that, the simulator needs to solve the resulting nonlinear equations to get values for each analog point in simulation time. Again, matrix operations (i.e. the Jacobi-matrix) are used to verify the existence and uniqueness of solutions.

Following, a short overview of the derived set of rules. Common rules affect the uniqueness of node- and branch-voltages and -currents, as well as free quantities.

- There needs to be a path from each node to the corresponding reference node.
- Using ideal voltage sources, the branch voltage is independent from branch currents.
- Avoid meshes of ideal voltage sources and sections of ideal current sources.
- Do not re-express Kirchhoff's law in simultaneous statements.

Most important for the initialization phase is the consistency of all given information. Further rules are affecting the used VHDL-AMS standard. For example, the initialization of signals is critical. If signals aren't initialized properly in the declaration section, they will earn a default initialization, depending on their type. Analog quantities are initialized using `Null`. Further rules are:

- Proper initialization of real valued signals, which are read in simultaneous statements, is necessary.
- Simultaneous statements need to be solvable even for analog input signals, which are no solutions to the given simulation task.
- Note that signals are only updated during events.

Keeping these basic rules in mind helps to avoid the most common problems during modeling and simulation using VHDL-AMS. Of course, that publication does not refer to any simulator specific details.

The creation of a mixed-signal design model is shown in [GAEC06]. The authors emphasize the importance of multi-domain modeling, requiring new challenges. Additionally, in contrast to other publications mentioned before, the transformation of HDL models into VHDL-AMS is investigated. The abstraction level of a model depends on the goals to reach. Specification validation may be necessary at the early beginning of new projects. At this stage, a high level model could suffice in the system design process. The authors show a clear procedure of methodology, valid for analog and mixed signal models, from specifications over model creation to final validation through simulation.

The market for VHDL-AMS simulation tools has evolved: Language standard support, library management, sufficient accuracy, and high speed simulation are common features nowadays. One important point is intellectual property (IP) management, which is used to enforce the reuse of standard design blocks. Thus, the need to redesign an entire block is redundant. This saves time and money, additional measures can be taken to protect the block from unlicensed modifications. In order to switch to VHDL-AMS technology, the authors show a proper way of automating the translation of HDL to VHDL-AMS language. First of all, parameters and functionalities have to be extracted from specifications to distinguish three different strategies: Pure analog, pure digital and mixed model types written in MAST language receive different treatments. ***Synopsis Paragon*** takes care of analog and mixed signal model types. Analog model code is handled automatically, whereas mixed signal code needs manual translation. Pure digital model parts are subject to be ported manually as well [2].

After VHDL-AMS model creation, a proper test bench is needed for model validation. A head to head comparison between MAST and VHDL-AMS simulator outputs showed good correlations, however, sufficient model documentation is needed in order to identify important parts and transform them correctly. Finally, that publication demonstrates its approach with the modeling of a high frequency amplifier device, comprising multiple simulation domains.

[2] The current version of Saber/MAST already supports analog, digital and mixed domains

2.1.3 Automotive Applications

An early work in the era of bus system simulations was written by Hale [Hal94]. When that publication was released back in 1994, the author was "eagerly awaiting" analog extensions to VHDL. Nevertheless, the entire document describes the modeling of an automotive data bus system using VHDL language. Interestingly, the author has foreseen the arising complexities in the area of automotive communication systems. The basic advantages of bus systems are discussed with respect to several aspects. Many features mentioned in this context are quite common nowadays. Sophisticated diagnostics for example, a typical property of modern, highly networked automobiles, was considered to be wishful thinking.

The concept of automotive wiring harness is explained, together with the occurring difficulties due to multiple bus access. A priority managed bus access scheme is described to ensure reliability. It is recognized that more advanced mechanisms may be necessary to deal with the upcoming complexity of network design. At that time, VHDL was chosen as modeling language, mainly because of its discrete and non-proprietary nature. The modularity of VHDL allows a flexible and re-usable design. But also the sole existence of an analog extensions working group tightened the choice of VHDL.

An ECU model was developed for network modeling, featuring a protocol model, buffering and an interface to sensors and actuators. Sensors provide data for the bus, whereas actuators are recipients. A data message itself does not contain a time stamp, but time stamps are attached to messages in order to trace their timing. The system modeling is based on the widely known OSI model. Each layer involved is represented using an appropriate VHDL entity. During the process of simulation preparation a network topology needs to be set up. The modular aspects of VHDL support this step.

The primary purpose of that research project was to discover message latencies in dependence from bus load. A special logging entity was developed to maintain an ASCII log file. Additionally, a bus-monitor entity was used to watch the bus. The resulting text files are subject to evaluation, generating graphical plots was and is still very popular. VHDL was considered to be an ideal language for modeling data bus networks. From today's point of view, this might appear outdated. Due to the large number of simulation tools available today, VHDL faces the competition when it comes to system level modeling. At hardware level, VHDL is still very popular.

An interesting work, comprising the numerous domains VHDL-AMS is aware of, demonstrates the design of a virtual prototype for an automotive throttle [PFSLP07]. It shows the interaction of mechanical, electronic and thermal processes. The resulting model was investigated through simulation and verified by comparison of measurements.

Due to short development cycles in the automotive business, the assessment of virtual prototypes has become important. Real prototypes are not always available instantly because of cost or time constraints. Models can be delivered much easier, giving a quick overview of a systems behavior. Much more, the use of models within simulation allows a deep insight to values, which may be hard or impossible to observe without any falsification of other values. Additionally, virtual prototypes allow the evaluation of worst-case scenarios, which may put real prototypes at high risk.

That publications shows the modeling of a complete throttle valve, which is installed in every combustion engine powered car today. A throttle valve consists of a power bridge, an electric motor, a gearbox, and a mechanical load. To meet the requirements for multi-domain modeling and simulation, VHDL-AMS is used to describe all necessary parts. Thorough simulation and comparison of output values with real measurements ensure proper model validation. One special aspect regarding this work is the consideration of component-related properties, like deviations caused by production or wearout. The given power bridge for example, consists of 18 relevant parameters, which may have an immediate effect on the output. With respect to those 18 parameters and their minimum and maximum boundaries, 2^{18} simulations (260 000 runs) would be necessary to cover every limit case. However, limit case does not necessarily mean worst case, and such worst-case situations are not covered within this work. Multi-dimensional parameters are not yet taken into account here as well. The modeling of wearout poses an additional problem.

The whole throttle valve model consists of one electric and five mechanical components, modeled entirely in VHDL-AMS. The simulation results show good correlation with measured values. However, small deviations can be explained due to varying motor brush and winding resistances, which lead to oscillations of the measured motor current. The simulated current consumption does not show these oscillations, because this kind of effect was not taken into account inside the motors model. Related to limit value considerations, some additional tests were performed, demonstrating the dependence of battery voltage, temperature, and torque constant on throttle displacement.

Another work in the area of automotive modeling shows the fault simulation of a CD player [GRR08]. Just like the previous publication the authors are using VHDL-AMS to describe the electrical and mechanical structures of the CD player. To achieve a sufficient source of faults, the road profile and audio input are modeled in VHDL-AMS as well. The resulting model is used for fault analysis and failure cause inspection. The final system model is able to predict reading errors due to road profile variations. The proposed methodology to minimize fault, reliability, and performance problems is applied throughout the whole model development process.

The given model uses the ability of VHDL-AMS to describe arbitrary blocks on different levels of abstraction. This way, critical components may include more detailed descriptions, whereas less interesting blocks may contain only basic approximations of behavior. This saves valuable time during simulation without compromising accuracy. Classical shock tests were executed, to determine the impact on output audio quality. The following components are modeled with VHDL-AMS:

- CD player electric and mechanical structures.
- CD player connections to the vehicles dashboard, body and suspension system.
- Irregular roadway profile.

Additionally, a real audio waveform is included in the simulation setup, to stimulate the resulting system with binary information.

Semiconductor manufacturer Atmel has published a document about simulation methodology for integrated mixed-signal circuits in the area of automotive electronics [AW03]. In that work, VHDL-AMS was considered in their development process for the first time. Goal of their work was the efficient modeling and examination of integrated circuits, with respect to simulation. A bottom-up design flow was used to accommodate boundary conditions.

Atmel is developing automotive integrated circuits in CMOS (or similar) technology. Their integrated circuits consist of sensor- and actuator interfaces, driver stages, microcontrollers, analog-to-digital and digital-to-analog converters, digital and analog subsystems, and a number of wireless components. Following a bottom up approach, a low level simulation is performed with focus on single components. Once these simulation results are satisfying, the development process advances to system-level simulations. Circuits containing a large portion of digital circuitry are primary candidates for mixed-signal simulation. Only small digital parts may lead to a performance decrease because of the computationally expensive analog-to-digital and digital-to-analog conversions. Additionally, behavioral modeling can increase simulation speeds dramatically. That publication clearly points out the differences between several layers of modeling.

According to Atmel, mixed-signal modeling is characterized by the quality of analog-to-digital and digital-to-analog conversion. Atmel is using input capacitances and threshold voltages for analog-to-digital conversion. The partitioning of analog and digital submodels is important: Atmel recommends analog modeling along the critical path, when increased accuracy is an issue.

That publication by Atmel also shows three examples of integrated circuits, modeled using mixed-signal techniques. Using these hands-on case studies, the basic ideas of that work are emphasized. Simulation speeds are dependent on the level of modeling, and behavioral modeling can reduce the time

necessary for simulation. Intelligent partitioning of submodels may also help to improve simulation performance. Furthermore, the authors take the time necessary for model creation into account and conclude that behavioral modeling definitely can pay.

2.1.4 Modeling of Non-Automotive Components

The work of [SHM01] describes the development of an efficient VHDL-AMS cable model. Based on the telegrapher's equation, that work provides a fast and stable simulation model, used to describe wave propagation in electrical and fluidic systems. The authors demonstrate basic equations of the lossless transmission line, by showing a simple voltage/current source system. Time is delayed by parameter τ, so this basic model takes care of a lines delay, capacitance and inductance. The resulting differential equation has no solution and represents only a small section of the overall line. One possible solution is the segmentation of the entire line, however, this approach leads to a huge number of differential equations, which might be computationally intense to solve. Another solution referenced in that publication is based on theories by Bergeron, which are also described in [Ami00]. It represents the transmission line just through a single line element. Two quite complex equations are described, taking seven parameters. These equations and parameters were subject to modeling in VHDL-AMS.

First, a short part of the implementation of a lossless transmission line is given. This early model takes the line's characteristic impedance, length and delay into account. This implementation is extended to a lossy transmission line. The simulation step size is adjustable through a single parameter setting. However, this is actually used to calculate the required simulation step size. The first version of the lossy transmission line model includes a so called ring queue, to access calculated past values. Due to performance reasons, this ring queue was replaced by a static, constant delay, to gain advantage of using VHDL-AMS's built-in `delayed` attribute. This is achieved through mathematical optimization, eliminating a computationally intense sum expression.

The authors also introduce a structural model of the lumped element transmission line, made up of discrete components. Presumable, this type of model is used for comparison and efficiency calculations. Finally, a direct comparison between this newly created and optimized model against the widely known lumped element transmission line model and the PSPICE transmission line model reveals advantages for the new modeling solution. The authors claim that the precision of the model is sufficient for most popular applications, however, there are some limitations mentioned. Those are affecting the line's series-resistance, parallel-admittance, length and characteristic impedance. Since the evolved model is going to be used in automotive network models with comparatively short cable lengths, this should not be a problem at all.

There are several approaches on how to represent a cable within a simulation environment. No matter which modeling language is used, the fundamentals are roughly the same. As described in [Joh02] and [JG03], the well known "Telegrapher's equations" are used in this work to reproduce a cable's behavior within the simulation. Originally derived from a coaxial cable, "Telegrapher's equations" are made up of small segments or blocks. Each segment represents a discrete equivalent circuit model for a small part of the line. Typically, such a segment consists of four discrete elements:

- Resistor R in series (H/m)
- Inductance L in series (Ω/m)
- Conductance G in parallel (S/m)
- Capacitor C in parallel (F/m)

Most important fact to note about these values is that they are measured per unit of length. Depending on the required length, a number of line segments are arranged in series. Figure 2.1 shows such a

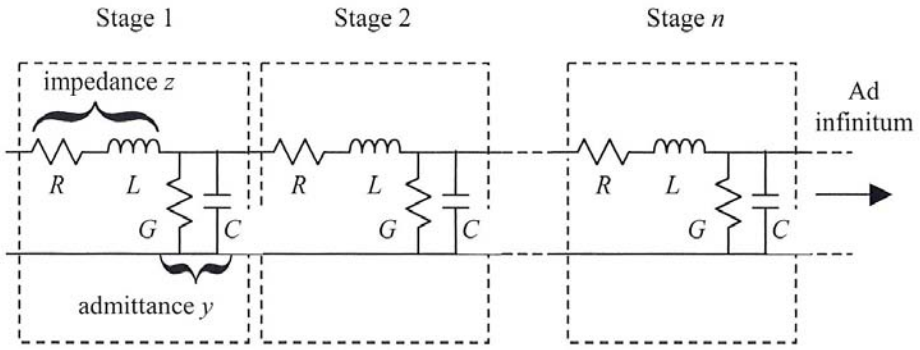

Figure 2.1: Telegrapher's equations discrete equivalent circuit model (RLGC model) [JG03].

basic transmission line model. Some publications refer to a "lumped line" when talking about RLGC transmission line models.

This RLGC cable model already implies the effect of losses through the modeling of resistance and conductance (which can be omitted to get a lossless model). Since one segment generates a set of equations to be solved by the simulator, many segments are only bearable using optimizations, like SPICE implements with its shifting operations [JG03, p. 682]. Straight VHDL-AMS implementations of line segments typically lead to extended simulation times. Together with the previously introduced behavioral model, both modeling techniques are investigated within the scope of this work, to find an ideal model to match the cables used in the laboratory's setup.

2.2 Model Verification & Validation

These two terms are fairly common in the area of computer science nowadays. Originating from other domains like scientific theory or statistics, verification and validation has made it into software engineering and informatics. Since the meaning of these terms is interpreted slightly different depending on the context used, this section defines a terminology for this document. Two documents have proven to be very helpful, the following definitions valid for this document are derived from [Mac05] and [Sar98].

Basically, the purpose of all models used in this work is to reproduce their dedicated real-world device's behavior. The better a model matches its real-world hardware counterpart, the more precise the simulation output will be. This strengthens the engineer's confidence and increases the model's credibility. This practice is used during various processes today, not just the solving of problems, but also decision making.

2.2.1 Model Verification

Model verification ensures that the implementation of the conceptual model is correct. This correctness includes a number of requirements, which are explained as follows.

The model is programmed correctly: The syntax of the used programming or modeling language does not contain any errors. This requirement is mostly supported by the use of programming paradigms, like structured programming, object oriented design or program modularity. According to [Sar98], higher order languages tend to have more errors than low-level or simulation languages. Due to the narrow design space, programming time is reduced. On the other hand, flexibility is reduced as well.

Algorithms have been implemented properly: This property is more difficult to achieve. Among others, mathematical or formal methods may be used to show proof. Mind that testing only does not

demonstrate correctness, because a test case may be successful, although a given algorithm was not implemented properly.

The model does not contain errors, oversights or bugs: This aspect is treated in modern software quality management. An algorithm may be implemented correctly from a formal point of view, but may for example contain typing errors.

Verification ensures that the model's specification is complete and no implementation mistakes were made. Testing is a common technique to verify a model. The process of model verification advances, the more tests are performed and errors are identified. Subsequently, a model becomes more trustworthy with number of passed tests. According to this statement, full verification is almost impossible to achieve: With the number of passed test cases, statistical certainty increases. Strictly spoken, the end of the verification process is not a fully verified model, but rather a model that has passed all verification tests.

The field of verification and testing has become fairly popular over the last few years and gained huge interest in many software-related branches. This includes a lot of various research topics, like

- requirements and system engineering,
- test case generation,
- automated testing, or
- testing methods.

However, verification does not answer all arising questions concerning a given model's behavior. It does not ensure, that a given model

- solves an important problem,
- meets a specified set of model requirements, or
- correctly reproduces the workings of a real world process.

Some of these questions are answered during the validation process, which is described in the upcoming section 2.2.2.

2.2.2 Model Validation

The process of model validation assures, that a given model represents and correctly reproduces the behaviors of the corresponding real-world system. Additionally, validation ensures that the model meets its intended requirements in terms of the methods employed and the results obtained. Quoting [Mac05], "the ultimate goal of model validation is to make the model useful in the sense that the model addresses the right problem, provides accurate information about the system being modeled, and to make the model actually used". Therefore, validation is closely related to the purpose of a given model, and its intended use. The result of the validation process delivers a model that has passed all validation tests. Such a model allows a better understanding of its capabilities, limitations and appropriateness for addressing a range of important questions.

Similar to social modeling, where human decision making leads to fuzziness in the validation process, non-modeled effects, unknown interactions or simply complexity lead to the problem of credibility establishment. Based on a preferable large number of significant experiments, a probably incomplete verification and validation process, and the experiences gained thereby, an engineer obtains confidence in his model. The relationship of cost (in time, money, resources, etc.) and value as a function of model confidence is illustrated in figure 2.2. In common, high model confidence is considered to be very costly. However, after a sufficiently large number of investments (time, efforts), the model's costs amortize and start to deliver even greater value to the engineer.

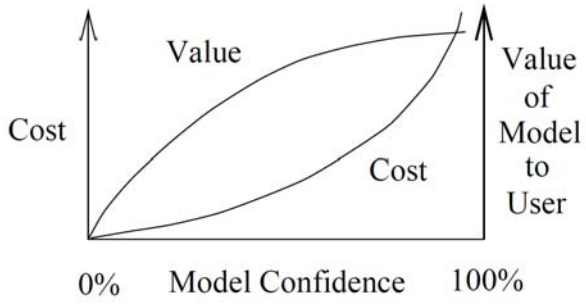

Figure 2.2: Model confidence [Sar98].

2.2.3 Model Validation Techniques

In general, the techniques listed by [Sar98] are summarized here. These can be considered as guidelines, because a combination of techniques is used in most cases. Although many methods have a slightly different background, they can easily be adapted for specific needs. For this work, the most relevant ones are picked and explained shortly.

Comparison to Other Models: The results of a given simulation model are compared to the results of another model, which is known to be valid. This is especially useful if the candidate model was built on different assumptions than the reference model.

Event Validity: The number of occurrences of a specific event within a certain time interval is counted and compared to the corresponding number of events of the real system.

Extreme Condition Tests: The models output should be plausible for any extreme or unlikely system states or inputs.

Face Validity: This method is used by simply asking people knowledgeable about the system, whether the model's behavior and output are reasonable.

Fixed Values: Setting inputs or state variables to fixed values, allows the checking of model results against expected calculated or quickly estimated values.

Operational Graphics: Values of performance measures are shown graphically over the advance of time. This helps to better understand the dynamic behavior of a given model.

Predictive Validation: The model is used to predict the system behavior, after that, comparisons are made between the system's behavior and the model's forecast to determine if they are the same.

2.3 Selected Topics on FlexRay™

This section gives basic information about the FlexRay™ automotive communication system. In connection with this thesis, specification version 2.1, revision "B" applies exclusively. The modeling of FlexRay™ related components is discussed in section 2.3.2. Some publications regarding verification and validation are reviewed in section 2.3.3. Most important aspects of test and diagnosis are discussed in section 2.3.4. A highly relevant publication about FlexRay™ signal integrity analysis is also discussed in section 2.3.5.

2.3.1 Basic Introduction to FlexRay™

This section is a brief introduction to the FlexRay™ automotive communication system. Its most important aspects are summarized here. All information found here is mainly based on the book by Rausch [Rau08] and the FlexRay™ electrical physical layer specification [Fle06a].

2.3.1.1 Interesting Facts

FlexRay™ is powered by the equally named consortium, founded in the year 2000. At the time of foundation, *BMW*, *DaimlerChrysler*, *Philips/NXP* and *Motorola/Freescale* were participating in the consortium's *Core Members* group. The consortium irregularly publishes specifications and specification-related documents, available via the web[3].

The FlexRay™ consortium named a few official goals to achieve during the contract period. An economic goal is the establishment of a new license-free industry standard. Standard components are manufactured in large quantities at low cost, drawing attention to services, tools and efficient engineering. Some of the technical requirements are higher bandwidth, increased redundancy, real time capabilities, topology independence, and the management of sporadic versus continuous communication. FlexRay™'s areas of application currently include backbone connections, distributed real-time applications and control systems. Especially, the replacement of the well known CAN bus communication system is an issue. Some automotive original equipment manufacturer's (OEM) are currently deploying FlexRay™ in the chassis area of serially produced vehicles.

FlexRay™ is based on layer 1 and 2 of the OSI layer model. Therefore a physical layer specification and a protocol specification exists. Higher layers are targeted by software: AUTOSAR for example, currently seems to establish a new industry standard. FlexRay™'s main hardware components are electrical bus drivers, active stars and bus lines.

2.3.1.2 Basic Network Anatomy

A FlexRay™ network consists of several network participants (nodes) and communication channels. One single node contains a microcontroller (host), a communication controller implementing the logical protocol itself, and one or two bus drivers, accessing the communication channel. Image 2.3 shows such an architecture.

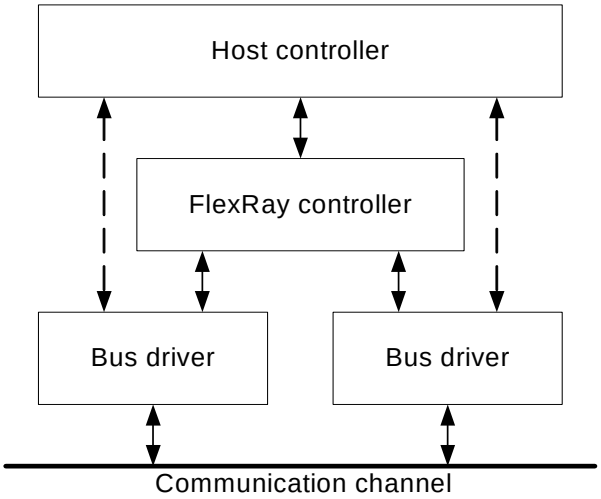

Figure 2.3: Typical FlexRay™ node architecture [Rau08].

[3] See `http://www.flexray.com` for details.

Basic access of the host to the communication channel is shown in image 2.4. The interface between host and communication controller is application specific, whereas the interface between the communication controller and bus driver is exactly specified. As can be seen in the graphics, `RxD`, `TxD` and `TxEN` signals are used to send and receive data streams to and from the bus driver. The latter one has access to the communication channel, which is realized using `BP` and `BM` electrical lines. FlexRay™ is not limited to electrical channels. Optical channels are possible as well, however, there is no specification for optical channels available yet.

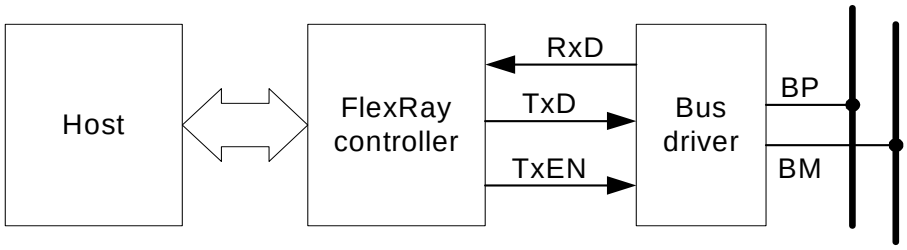

Figure 2.4: Bus channel access [Rau08].

The physical arrangement of nodes is called topology. These nodes are interconnected using a shared communication channel building a network. FlexRay™ supports up to two bus channels per node, which can be used to increase redundancy and/or bandwidth. A cluster is considered to be a subnetwork, which is able to operate independently. FlexRay™ supports two basic topologies, which can be used to build clusters: Bus- and star-topologies. By using two channels, each has its own, fully independent topology. Star topologies can be made up of either active or passive stars. An active star works like a hub, it amplifies received signals and distributes them on the network. A passive star has no active components and is built by simply connecting a number of electrical lines at a single point.

There are various rules to obey in order to build valid FlexRay™ topologies. See [Rau08], [Fle06a] and [Fle06b] for formal descriptions and examples of valid and invalid topologies.

2.3.1.3 FlexRay™ Communication Protocol

FlexRay™ belongs to the family of TDMA (time division multiple access) protocols. All nodes in such a network are equal in the sense that there is no central bus arbiter. Each node has one or more time slots available for communication over the channel. The static distribution of time slots has to take place in advance prior to using the network. This is called scheduling, and requires a common time base between all nodes. The clock synchronization process necessary to achieve synchronized nodes is described in [Rau08] and is not discussed anymore in this thesis.

Messages are sent in frames over the communication channel. A frame contains the data to be sent plus additional data regarding the communication process itself. One frame consists of:

- 5 bytes of header data
- 0-254 bytes of payload data
- 3 bytes of checksum data (cyclic redundancy check, CRC)

The frame header consists of:

- 5 control bits
- 11 bits frame identifier (frame-ID)
- up to 16 bits of payload-length, indicating the number of 16 bit words of payload

- 4 bytes of header CRC
- 1 byte of cycle counter

Non return to zero (NRZ) encoding is used for FlexRay™. Each data transmission starts with a ***transmission start sequence*** (TSS), which has a length between 3 and 15 bits, and is used to activate active stars and receivers. The activation of these components takes some time, therefore an active star does not transmit the entire TSS, but a truncated one. After the TSS, a high-bit called ***frame start sequence*** (FSS) follows. Another subsequent high- and low-bit indicates the so-called ***byte start sequence*** (BSS), which precedes every transmitted byte of the payload data. Finally, after the transmission of the last byte, the ***frame end sequence*** (FES) follows, composed of one low- and one high-bit. Image 2.5 illustrates the previous explanations.

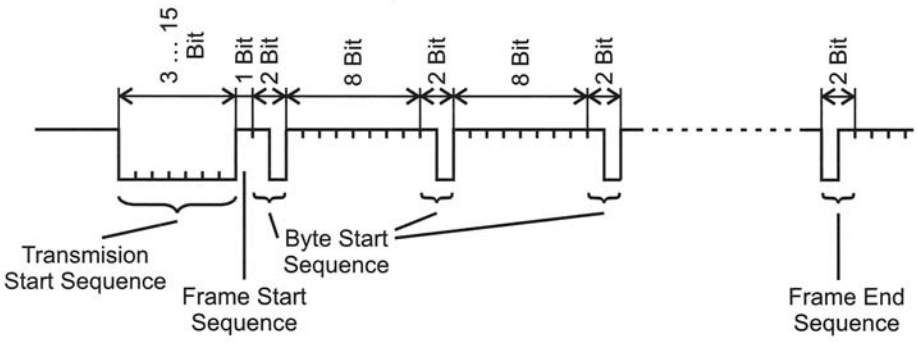

Figure 2.5: Typical FlexRay™ frame structure [Rau08, p. 42].

2.3.2 Modeling of FlexRay™ related Components and Systems

A fairly relevant publication utilizing VHDL-AMS to create a behavioral model of a FlexRay™ compliant transceiver is given with [CVRS09]. The model designers recognized the need for a behavioral model based design, in order to verify such complex and safety critical systems thoroughly. Errors are found more quickly, because of the increased controllability and observability. Behavioral modeling avoids the need for complex hardware development, circumvents low-level design problems, and therefore reduces cost and time to market. From that work, a mixed-signal FlexRay™ transceiver emerged.

A top-down approach was used to break the overall system description down to primitive elements. These single elements can be used to compose different subsystems. Additionally, focus was on convergence problems and discontinuities. For modeling of input and output interfaces, analog domain descriptions were used in order to comply with other transistor level circuit simulators. The transceiver's core module is modeled digital style. The resulting transceiver model was subject to verification. In accordance with the remarks in section 2.2, this means the model was checked for correct implementation and compliance with specifications. Therefore, the transceiver model was connected to a 45Ω resistor and $100pF$ capacitor in parallel. This method is also suggested by [Fle06a, p. 35], to investigate the transceiver's output waveform. To evaluate the resulting voltage levels and slew rates, a simple eye diagram is considered.

One of the main goals was to verify the transceiver's functionality in connection with different bus states. By stimulating the analog input ports, the transceiver generated all requested bus states successfully. After that, a power supply failure was subject to investigation. The transceiver model passed this test as well, showing correct behavior at any time. Finally, a short circuit bus line was introduced, making the differential bus signal stuck at `Vcc` or `GND`. During this test, the transceiver showed expected behavior.

Finally, the transceiver model's performance is evaluated by comparing it to a Spectre model. It shows that the VHDL-AMS model works quite fast while delivering desired results during the described

simulations. However, no insight is given to implementation details throughout that work. The results of that work are fairly generic and focused on performance issues. The validation against a real hardware device is missing.

This thesis is clearly aimed at behavioral and structural modeling, including hardware related aspects. The work by Wang et al. [WSCW10] shows the development of a transceiver frontend for ECUs in FlexRay™ based automotive communication systems. It is included at this point for a number of reasons. First of all, it is fairly difficult to find out details of hardware bus driver implementations. Manufacturers are very sparing with information, concrete implementation details are kept secret because of various optimizations. Second, that document shows basic concepts of a FlexRay™ bus driver design. This thesis does not deal with concrete hardware implementations, because the proposed concept should apply for a wide number of integrated circuits by various manufacturers.

In the first part of that work, basic FlexRay™ properties are explained, with respect to LVDS technology in common. The transmitter's MOS transistors dimensions are calculated on the basis of the specification's load given with $40\Omega||100pF$. The transmitter is made up of two AND and two OR gates, which take care of the enable and data signaling. Those signals are usually utilized by a communication controller, implementing the FlexRay™ communication protocol. They are controlling two p-channel and two n-channel metal-oxide-semiconductor (MOS) transistors, building a voltage bridge. Two $10k\Omega$ resistors connected to the reference voltage are used to achieve a differential voltage between `BP` and `BM` ports of the circuitry. The receiver is made up of three comparators, implementing a threshold based digital signal recovery. Implementation details about that comparators and the used voltage regulator are given. Two $10k\Omega$ resistors ensure high ohmic input ports.

Besides these important components, there are details given: Schematics and dimensioning of clock generator, the contained ring-oscillator and delay cell. Additionally, focus is put on temperature compensation circuits. Finally, the resulting chip was tested using an oscilloscope. The authors even applied thermo chamber tests, ensuring functionality between $-40°C$ and approximately $+125°C$. It has shown that the resulting $0.18\mu m$ manufactured chip operates stable under all tested conditions and delivers FlexRay™ compliant signaling, when loaded as described by the specification. However, there are no results available, demonstrating a larger working FlexRay™ network using the developed bus driver.

The deployment of a practical FlexRay™ based application is shown by [XKC+08]. The engineers used SDL (Specification and Description Language) in the development process. Verilog hardware description language was used to process the emerged SDL source model. Finally, a demonstrator on the basis of an FPGA is shown, the application involved is some sort of distance warning system.

A dual FlexRay™ bus topology was considered. The architecture of one single FlexRay™ node consists of host, communication controller, bus guardian, and up to two bus drivers. Software is used to control the communication process of one single node.

Their approach by using SDL is interesting. Protocol design using this formal language ensures the compatibility between the requirements of the initial design and final implementation. However, SDL was standardized by ITU (International Telecommunication Union) in order to model, simulate and verify communication protocols. Within that work, FlexRay™ communication controller and bus guardian models are presented, described in SDL. For this purpose, a total number of 32 submodels is required. The communication controller and bus guardian are realized in a finite state machine using Verilog. After successful synthesis, a working prototype was available for test.

The designed system is able to operate at speeds up to 76 MHz, for verification it was stimulated using traffic generated by an automotive distance warning system. A host PC was used to verify the correctness of all submitted frames.

2.3.3 Verification and Validation of Automotive Bus Systems

The focus of [GB09] is on verification methodology. That paper shows how to design and verify the physical layer implementation of FlexRay™ systems through an automated and robust simulation based engineering method. Its authors have recognized that signal integrity is an important point, not only during the development stage, but also during the entire product life cycle. Manufacturing tolerances and environmental impacts are disturbing the network's reliability. Even if all guidelines given through FlexRay™'s specifications and suggestions are met, there is no guarantee that the resulting network is going to behave as expected. Since building prototypes is not always possible, simulation is the only choice to evaluate such high-dynamic systems. As evaluation criteria, the following points were identified:

- Asymmetric Delay
- Propagation Delay
- Truncation of the Transmission Start Sequence
- Bit deformation due to ringing and reflections
- Message frame stretching due to ringing after transition from active to idle

The publication proposes a way to evaluate the given delays and bit deformations, as well as an automated process for such evaluations. All kind of analyses are heavily dependent on the used network topology. FlexRay™ is a digital system, however, all data transmission is done in an analog way. Deformed analog signals are hiding behind the bus driver's all-digital interfaces. Circuit ringing, reflections, and other parasitic effects are common causes. FlexRay™ provides robust mechanisms like majority voting to overcome these issues, nonetheless the network developer has to ensure correct operation, so that communication among all ECUs is not inhibited.

The behavior of network components is never ideal, the authors mention falling and rising edges with different slew rates, different transmission delays through the bus driver and varying threshold levels. When all these effects collude, an asymmetric delay may arise, which is critical for other functionality, like clock synchronization.

In general, [GB09] makes use of the SABER mixed-signal simulator, which supports MAST and VHDL-AMS, among a few other modeling languages. The models considered for simulation were a bus driver, a transmission line, a split termination, and various ESD and EMC related components.

The simulated networks included a passive star topology, to show the previously described effects. Additionally, the second part deals with analog-to-digital conversion. A software interface for automated report creation is shown, which interconnects simulation output with spreadsheet software.

Interestingly, a low versus high temperature analysis was performed, considering worst-case limit values as model parameters. Between both limits, a differential voltage drift of around 0.4V was measured, which poses a significant impact on signal integrity. Undesired switching of RxD output due to sampling errors caused by varying thresholds is shown as well to underline the complexity of all parameters used.

The given work demonstrates a modeling and simulation design flow, in order to build and analyze FlexRay™ networks. The differences to this work are diverse, although the goals are similar. The authors introduce tools and methods for simulating FlexRay™ conform networks, whereas this work has its place in the huge TEODACS infrastructure and integrated environment, allowing multi-level design and test operations.

The development and verification of in-vehicle networks is described in [GS05]. The authors emphasize the multiple design layers of an automotive network. Early verification of such multi-layer systems has shown to be critical. First, rather generic models are used for simulation, which might be replaced

by more accurate models in the development process later on. Throughout that work, a CAN network is build using the SABER simulation environment.

The authors subsequently describe the anatomy of a CAN network including corresponding nodes. From a high level view, CAN and FlexRay™ networks are quite similar: Both exist on layer one and two of the OSI model (compare [Rau08, p. 11]) and both utilize the low-voltage differential signaling technique. This leads to a fairly similar problem space when developing and verifying network topologies.

The verification process proposed in that work is based on worst case parameters and tolerances. Since such boundary values are usually not available in real hardware models, verification only by measurements is considered to be insufficient. Furthermore, changing existing prototypes would dramatically raise costs and time. Simulation is assumed to be the only choice for this verification technique. The SABER simulation environment was used for analog and mixed-signal simulations. The models used for simulation include numerous CAN nodes, EMC protection circuitry, a transceiver and a CAN controller model. All CAN nodes are interconnected using a twisted pair transmission line model.

This transmission line model was recognized to be critical, since effects like reflection and crosstalk are essential for signal integrity. The proposed model consists of two conductors, with capacitances between them and to ground, respectively. The inductance along both conductors is modeled as well. The wire model itself was validated against measurements and showed good overall behavior. In order to filter common mode voltages on the bus, a common mode choke is proposed, which is build by two coupled inductances. This model is subject to parameterization, in order to characterize a specific common mode choke.

The used transceiver model was obtained directly from the semiconductor manufacturer. A CAN transceiver model by Infineon is used to generate bus signals. A simplified CAN controller is used to control the transceiver. Since CAN is an event-triggered communication protocol, special attention is payed to the timing behavior and state machine of the controller. Finally, the controller, transceiver, and bus line models are integrated to a test bench. By stimulating the network with selected bit streams, the resulting waveforms on the bus are visualized and measured. Besides basic signal properties like rise and fall times, the correct sampling of bits is thoroughly investigated. The authors are concluding that their methodologies are successful in the areas of early problem detection, topology changes, and prototype reduction. Furthermore, their approach could be used for other in-vehicle network technologies as well.

Opposed to CAN, the following publication describes the development of physical layer and signal integrity analysis of FlexRay™ design systems [GB07]. The goals described are quite similar to those named in [GS05], with the exception of a different technology. FlexRay™ was chosen in order to fulfill upcoming requirements to in-vehicle communication systems. The document gives basic information about FlexRay™ and highlights some economic aspects, which are not commented at this point.

We will turn over to their aspects of simulation instead. According to them, the following components need to be considered when developing physical layer networks:

- Signal filters
- Active stars
- Bus drivers
- Transmission lines
- ESD protection elements
- Topology types
- Termination

These elements offer great flexibility in network design, however, it is very difficult to draw a substantial conclusion about the network's resulting signal integrity. Because of that, simulation is once more considered to be the only solution of dealing with complexity. The following aspects of signal integrity are critical:

- Signal propagation delay
- Asymmetric delay
- Bit deformation
- Truncation of TSS
- Frame stretching

Based on the identification of these critical points, a robust development flow for physical layer implementations is derived. The developer provides information about topology and components, together with their associated parameter values. The simulation reveals the feasibility of the proposed system, a number of iterations may be necessary to re-design or optimize the assumed topology.

For demonstrative purpose, a six node passive star is shown. Each node is made up of the following components:

- Transceiver (bus driver)
- Split termination
- Common mode stabilization unit
- Common mode choke
- ESD protection

As in other publications (for example, see [GS05]) the bus driver model was delivered by an integrated circuit manufacturer. In that case, NXP provided a MAST model for the SABER simulation environment. It is a functional kind of model, to ensure proper simulation speed, at the cost of accuracy. That transceiver model was validated through measurements against a real system implementation. A similar publication by Bollati [Bol06] released earlier shows interesting details of a receiver and transmitter model. Those models are based on a physical layer, yet contain some functional elements. Simple switches are used to generate the differential bus voltage, together with various voltage sources. The receiver is made up of discrete components, including diodes, resistors and bipolar transistors, realizing a threshold based analog-to-digital converter. The transmission line is a pure behavioral model, demonstrating only delay and voltage to current ratio. Eye diagrams are used to judge signal integrity, however, aspects like asymmetric delay or even In the later work by Bollati and Gerke the transmission line model was identified as a critical component in the simulation framework. Three basic requirements are made to the model:

- Wire length
- Frequency dependent losses
- Differential and common mode behavior

SABER provides a transmission line model fulfilling all of these requirements. However, the model equations are defined in frequency domain, convolution operations are necessary to get back to time domain. A field solver transforms transmission line parameters to an RLGC matrix, which is integrated

into the simulation environment. The authors prefer that type of model over a lumped element model. Namely, they try to avoid unnecessary oscillations and computationally intense model instances.

Obviously, no active star model was available for that work, but passive star networks are investigated. By utilizing a round robin communication scheme, basic signal transmission over cables with different lengths is realized. The received waveforms on terminated and non-terminated nodes are compared. On non-terminated nodes, oscillations and reflections are observed. Passive ferrite core filters are used to reduce these effects, which has shown to work out to a certain degree. SABER supports parameter variations very well, so different sized ferrite cores are evaluated.

To assure proper bit sampling during transmission of data streams, the asymmetric delay needs to be watched carefully. The first negative edge after the TSS in the binary data stream helps to determine propagation delay. Asymmetric delay occurs due to limitations of the FlexRay™ decoder module and bus driver [Fle06a, p. 13], and shortens or prolongs the nominal bit length, with reference to `TxD` of the sending node. During simulation of the given passive star network, the asymmetric delay was measured for each pair of nodes, using `DATA_1` and `DATA_0` bits, respectively. No significant variances were detected.

That publication concludes that simulation is the method of choice for FlexRay™ based network design. Simulation helps to keep costs at a low level, because it becomes possible to predict the behavior of physical layer implementations.

In [Ger07] Gerke describes the development of a robust physical layer implementation for in-vehicle networks on the basis of CAN technology. CAN as de-facto industry standard, is pushed to its limits. The number of nodes per vehicle is increasing, and there is always the need for more bandwidth. The only way to verify prototype topologies is through model based design flow. Simulation is used to discover possible issues before they arise in early hardware prototypes. The robustness of the resulting implementation is in focus of that publication.

Future in-vehicle networks may comprise a number of different network technologies, where FlexRay™ might be used as a backbone for other networks and protocols. CAN technology however, is reaching its physical limits, 30 ECUs and cumulated wire length up to 50 meter per network are still possible nowadays. For CAN, these items are sensitive for signal integrity:

- Topology
- Transmission line (type and length)
- Bus driver
- Interface between transceiver and transmission line

System simulation is recognized as a fundamental requirement in early stages of the network design process, in order to achieve robust network architectures. For modeling, SABER was used as primary tool.

The transmission line was identified as one of the most critical parts in the simulation model.

Reflection and crosstalk are mentioned as key features, next to simulation speed. Additionally, the bus driver must provide good accuracy since it has fundamental impact on signal integrity. It is recommended for network developers, to obtain their simulation models directly from semiconductor manufacturers.

An application example is given in order to point out the practical use of the proposed models. A network consisting of 14 nodes was set up, to analyze signal integrity relevant effects, like ringing, superposition of reflections, and stray inductances, to name a few. In CAN networks, timing is important for sampling bit values. Therefore, sample point investigations are possible, even with the simplified CAN controller model used. A second application scenario is given, demonstrating the simultaneous access of five nodes to the bus. This case is not relevant for FlexRay™, because of its time-triggered access scheme.

Another publication about validation of network topologies was written by Lawrenz and Bollati [LB07]. In their work, they recognized the difficulties arising by deploying multiple automotive bus systems in one single vehicle. Throughout that work, their focus is on network topology as well as modeling of single components. By modeling all relevant network components, the electrical signal behavior is investigated.

First of all, the authors introduce a number of bus systems and how they are used in modern vehicle installations. Each of these bus systems has its own specification and requires a different physical layer implementation. An excerpt of those bus systems including their field of application is recapitulated in the following list.

- Infotainment: Most™, D2B
- Power train: CAN, FlexRay™
- Safety systems: FlexRay™
- Body electronics: LIN, CAN

As widely known, a vehicle represents a harsh environment for any electronic system, therefore the given bus systems are designed to succeed in their area of application. However, each system requires individual adaptation and careful design. Various effects may endanger those communication systems: Ground shift, battery drop, electrostatic discharge, or electromagnetic interference (EMC) lead to incorrectly sampled bits or even disturbed communication links. Furthermore, a typical CAN network architecture is introduced, from physical layer bus signals, transceiver and microcontroller up to ECU software. Ringing and line reflection are identified as main problems concerning the physical layer and signal integrity. Individual bits may be sampled wrongly.

It is recognized, that complexity is a major issue, and that new methods are required to deal with it. Different levels of abstraction provide deep insight into systems and their submodels. The ultimate goal is to find new methods, in order to cope with different types of models, whether they are of digital, analog or mixed-signal style. According to [LB07], C&S group has developed a simulation environment to analyze signal integrity in several vehicle networks.

Using the example of CAN, a node's architecture is shown, comprising a transceiver, common mode choke, coupling circuits, termination and ESD protection circuitry. A transmission line model interconnects all available nodes. Modeling is done in three ways: Schematic modeling enforces a very low level and component related view, where physical properties are relevant. Next, VHDL-AMS is used for functional modeling, where component behavior is described in a simplified way. Third, MAST is used to describe the overall communication system.

The authors explain that simulation is a valuable means to identify the valid area of operation of a given topology. In such a complex system, many parameters are involved. It is useful to know about the most critical tolerances, in order to achieve significant results. This publication however, does not tell anything about specific verification or validation techniques.

2.3.4 Test and Diagnosis

This section describes related work about the test and diagnosis concept within the TEODACS project.

2.3.4.1 Layer Model Concept

A layer model for the systematic test of time triggered automotive communication systems is introduced in [ASHP04]. This work is targeted at problems arising during the test phase of such systems. A layer model generally helps to distinguish the different services and mechanisms which are available on the network. Such a layer model is in close relation to the level of abstraction. Especially communication systems appear much more transparent, if they are properly organized in levels of abstraction. This eases

the generation of test cases, and allows unerring diagnosis. Additionally, the propagation of potential faults can be traced more easily.

On the basis of the FlexRay™ protocol, a layer model is introduced. This model is divided in four different layers, along a transmit and receive path, taking time synchronization issues into account. The four layers are based on the well known OSI layer model. On top, the application layer generates and processes arbitrary application messages. The presentation layer below sends and receives payload data, which is converted to frames on the data link layer. In FlexRay™, this layer also deals with time synchronization issues. At the bottom, the physical layer transforms between simple raw bit streams and physical streams. Several parameters are affecting each conversion process at every layer. For example, during the conversion of a physical stream to a raw bit stream threshold values, the used oversampling rate, glitch filter parameters and decoder parameters are affecting the performance of the conversion process. The entire layer model is presented in detail in [ASHP04].

Each of the previously explained conversion processes may produce erroneous output, because of three possible reasons: The input may be faulty, something could be wrong with the configuration parameters, or the mechanism itself produces erroneous output. The proposed model has the ability to track down faults and their resulting errors along different propagation paths. Sending an receiving paths in a model behave differently, since in the sending path, information is appended whereas information is stripped in the receiving path. Finally, that model is applied to fault diagnosis, where a tester node helps to identify local and remote node faults.

2.3.4.2 Test and Diagnosis of FlexRay™ Communication Systems

A relevant piece of work evolved from the TEODACS project describes efficient means for test and diagnosis of FlexRay™ communication systems [ATK+09]. The strength of TEODACS is the common development of a real hardware prototype together with a co-simulation framework. This approach covers the communication system at multiple layers of abstraction, and therefore defines special requirements for test and diagnosis. An important key-feature introduced in that work is the creation and evaluation of so called log files. These log files represent an interface between the simulation setup and hardware laboratory.

In order to diagnose a FlexRay™ communication system at different layers of abstraction, appropriate network access is required. Several scenarios are possible: Cyclic debugging using deterministic replay, model and system validation using simulation results of both hardware and simulation environment, automated stimulus and test case generation, or even cross-analysis methods. The latter allow the replay of hardware-monitored signals in the simulation environment and vice-versa.

The proposed log files allow the efficient data exchange between hardware and simulation environment. Five different log file formats do exist, in order to describe all five different levels of abstraction: The analog level provides the highest accuracy with the largest data amount. The sample level is located above the analog level, providing digitized values. Several samples may represent a single standardized FlexRay™ bit, so the bit level is above the sample level. A FlexRay™ frame regroups several bits, so the frame level is next. Finally, the signal level totally abstracts the underlying communication architecture and is composed of application variables.

Log files may be replayed on a given network using a testernode, available on both platforms. Four import- and export-modules take care of the conversion between simulation environment and laboratory setup. The software testernode is written in SYSTEMC and described in section 2.3.4.4. The hardware testernode available to the authors is based on a FPGA platform. Additionally, to fulfill the requirements above, a log file generator, supporting all previously listed levels of abstraction, was developed to create arbitrary bus traffic scenarios from scratch. This advantage is also underlined in [Arm06], as it is important for system verification. The performance of error detection and correction mechanisms can be reviewed by repeated fault-injection. After all, the authors demonstrate an industrial use case, which validates a recently developed active star integrated circuit by austriamicrosystems AG.

2.3.4.3 Hardware Testernode Model

To fulfill the validation requirements in the TEODACS project, the concept of a hardware testernode was invented [AWK+09]. The basic idea is to gain access to the real hardware bus communication system for a large number of scenarios. To manage the arising complexity, the hardware testernode is build on the layer concept introduced earlier in section 2.3.4.1. A dedicated PCB provides low level access to the network and offers different physical test points. FlexRay™'s TxD and RxD pins are routed straight through an FPGA platform, which enables traffic monitoring and replay. Additionally, two *AS8224* active star devices are implemented, each with four different branches equipped with *AS8220/AS8221* bus drivers. The functionality of an active star device is discussed in section 3.1.3.3. Each branch offers the possibility of external termination, allowing various experiments with different cable termination solutions.

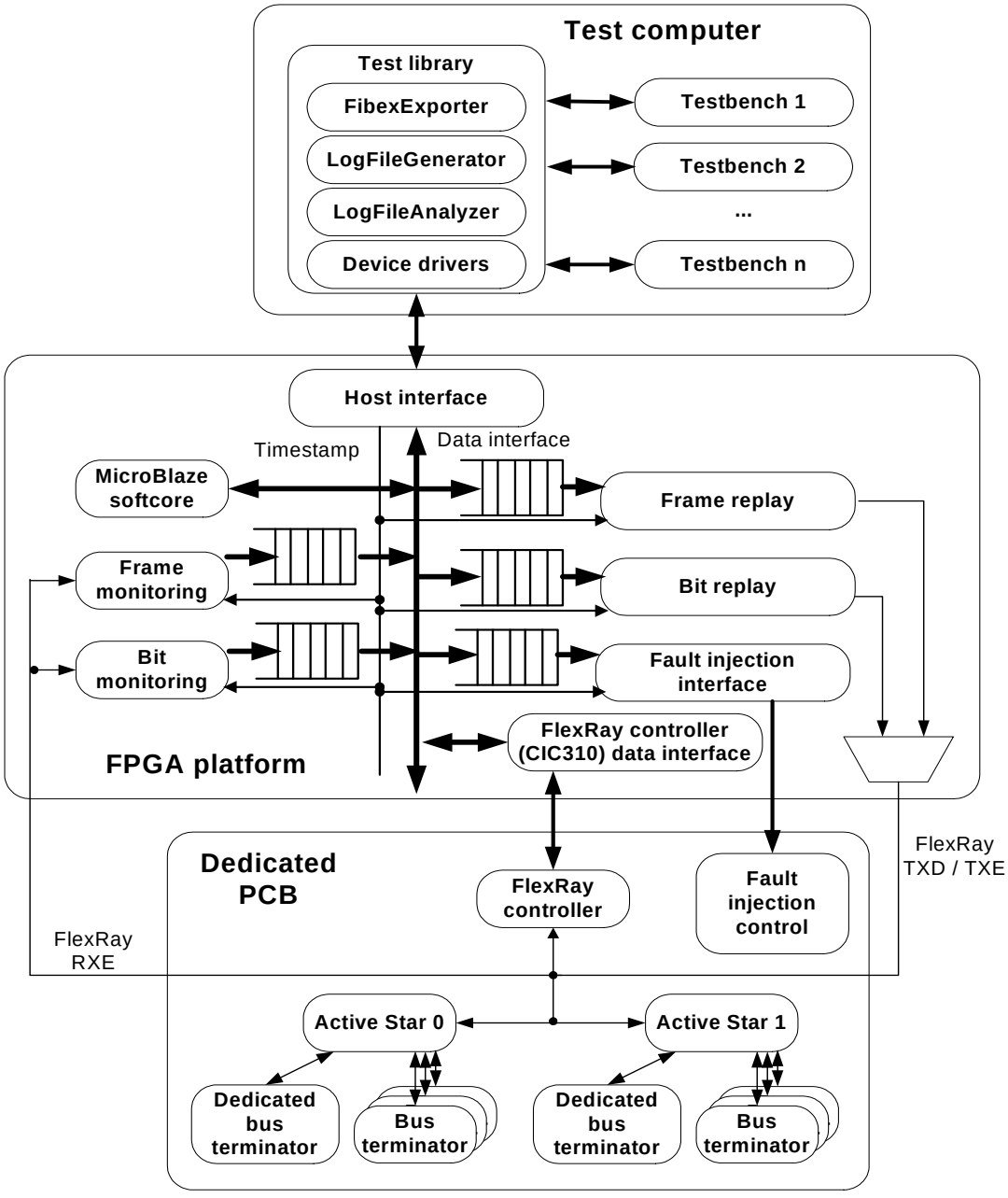

Figure 2.6: FlexRay™ Xpert.Lab testernode concept [AWK+09].

```
1 ;PACKET_NR|Ch |Time            | Length      | DATA
2 ;         |   | [s]           | [samples] |  [hex]
3       1   | A |  0.2210030000 |         360 | FFFFF FFFFF 0FFFF
            FFFFF FFFFF FFFFF 03FFF FFFFF FFFFF FFFFF 00FFF FFFFF
            FFFFF FFFFF 003FF FFFFF FFFFF FFFFF
4       2   | A |  0.2210095000 |         360 | 00000 00000 F0000
            00000 00000 00000 FC000 00000 00000 00000 FF000 00000
            00000 00000 FFC00 00000 00000 00000
```

Listing 2.1: Sample level log file containing raw data

The previously mentioned FPGA platform implements a host interface to map the user's requests to the different modules. Besides timed replay functionality, the FPGA allows traffic monitoring at bit and frame level, as well as fault injection. The host interface talks to a connected PC platform, containing an extensive test library. This library consists of various tools and software programs, which are common to both FlexRay Xpert.Sim and Xpert.Lab environments. Data exchange between both platforms is possible thanks to the log file format discussed in section 2.3.4.5. Figure 2.6 illustrates the described architecture. Note the different signal naming convention, where RXE equals `RxEN`, and TXE equals `TxEN`.

2.3.4.4 Software Testernode Model

To provide efficient means for testing in the TEODACS simulation environment as well as in the hardware setup, a software testernode model was developed by Krug [Kru08]. This testernode is the software counterpart to the TEODACS hardware testernode, which is described earlier in section 2.3.4.3.

The goal of the work by Krug was to provide methods for network traffic monitoring, traffic processing and network stimulation in the FlexRay™ Xpert.Sim environment. To comply to the TEODACS concept and to get an optimal view on the whole communication system, this implementation supports the various levels of abstraction introduced in section 2.3.4.1 as well.

The software testernode model is entirely written in SYSTEMC and integrates seamlessly into CISC's SyAD®. Therefore it is perfectly usable in the TEODACS co-simulation environment. Basically, the testernode model is able to convert so called log files to timed real value output. In SyAD® it is represented by a composite model having two real type output ports in order to supply two FlexRay™ channels. By using a real to analog converter, electrical quantities can be used to stimulate a physical layer model, like the FlexRay™ bus driver for example. Two important configuration files need to be specified for the software testernode to work: First, the FlexRay™ network configuration as described in [Fle06a], and second, the TEODACS log file containing information about the signaling itself.

2.3.4.5 The Log File Format

The log file format can be seen as common language between the FlexRay™ Xpert.Sim and Xpert.Lab environments. All recorded or generated bus traffic data are saved in log file format. The structure of this file format is defined in [Arm08]. Depending on the different levels of abstraction, the file format's fields differ slightly from level to level. To recapitulate the format on sample level, a small example is given in listing 2.1.

This rather short file is interpreted as follows: The first two lines are of informative type and commented out by use of a semicolon. From this point on, the file is organized in columns, separated by vertical bars. The first column denotes the packet number and is represented by an increasing integer value. The second column includes either "A" or "B", indicating the FlexRay™ channel to use. The third column contains a real value, specifying the exact moment of simulation time in seconds, the data packet is started to send. The length of the given data packet is given in column number four, 360 samples at 12.5 nanoseconds each means that this packet will last for exactly 4.5 microseconds. Finally, the

data itself is given in column five. It is expressed using hexadecimal values. Every ***high*** bit representing one ("1") results in `Data_1` on the FlexRay™ bus, whereas every ***low*** bit representing zero ("0") results in `Data_0`.

Therefore, the log file presented here contains simple pulses. In data packet one, the first one appears exactly $10*4*12.5 = 500$ nanoseconds after timestamp 0.221003, lasting for 50 nanoseconds. There are another three pulses following with durations of 75, 100 and 125 nanoseconds. Data packet two contains the same pulses as well, but their polarity is inverted compared to data packet one.

2.3.4.6 Log File Generator

A dedicated log file generator written in Perl language is able to produce arbitrary FlexRay™ compliant log files according to a specific FlexRay™ configuration [Vir08]. This includes parameters like sample clock period, bit length, communication cycle length, slot durations, etc. All in all, there are 16 parameters specifying the correct protocol behavior.

The log file generator is invoked via command line, its output is a log file containing the requested signal, according to the given parameters.

2.3.5 Signal Integrity Analysis in TEODACS

This section mainly describes the work by Clazzer [Cla09], emerged as part of the TEODACS project. The author developed a software tool for the automatic analog layer signal integrity analysis of the FlexRay™ communication bus. It helps to analyze a given network topology, with respect to electrical signal characteristics. That tool integrates seamlessly into the TEODACS project environment: It is capable of processing data files recorded by an oscilloscope, as well as simulation output files. It separates bus activity from non-activity, it is able to generate sample level log files (as discussed later in section 3.2.1) and furthermore, it performs a signal integrity analysis based on FlexRay™ characteristics and eye diagrams. All signal parameters were chosen in accordance with the FlexRay™ physical layer specification [Fle06a]. The following sections provide an overview of the mentioned capabilities.

2.3.5.1 Bus Activity Identification

In order to separate noise from signals, a voltage threshold is necessary, $\pm 30mV$ was determined as a default choice for the software application. The program is able to distinguish FlexRay™ compliant data from erroneous glitches, which are disturbing bus communication. To overcome certain memory limitations, the sliding window approach is used to process huge amounts of input data. The process of searching for bus activity is performed using a dedicated function, which basically implements the flow charts given in [Cla09, p. 30-31]. Finally, a complex splitting function [Cla09, p. 36] manages the breakdown of the input file into several files containing ***bus activity elements***. The result of the bus activity identification process is a hierarchy of files and folders, containing the found bus activity elements and a short report giving details about the overall process.

2.3.5.2 Signal Integrity Analysis

With respect to this work, that function seems to be the most interesting one. It relies on the bus activity identification step introduced in the previous section. Within this signal integrity analysis step, the bus voltage waveform is analyzed. A well known and proven method of signal integrity analysis is the creation of an eye diagram. Basically, an eye diagram graphically superimposes all positive and negative edges of a digital signal in a single diagram. In general, the larger the diameter of the resulting eye, the better the signal integrity. FlexRay™ clearly specifies the voltage levels necessary for correct signal recovery, so it is possible to draw an eye diagram depicting the minimum requirements assumed for a received signal. A so-called six-function approach is used to visualize this minimum eye. Figure

2.7 shows a minimum-requirements eye diagram for received signals at the bus driver, conforming to FlexRay™'s specification.

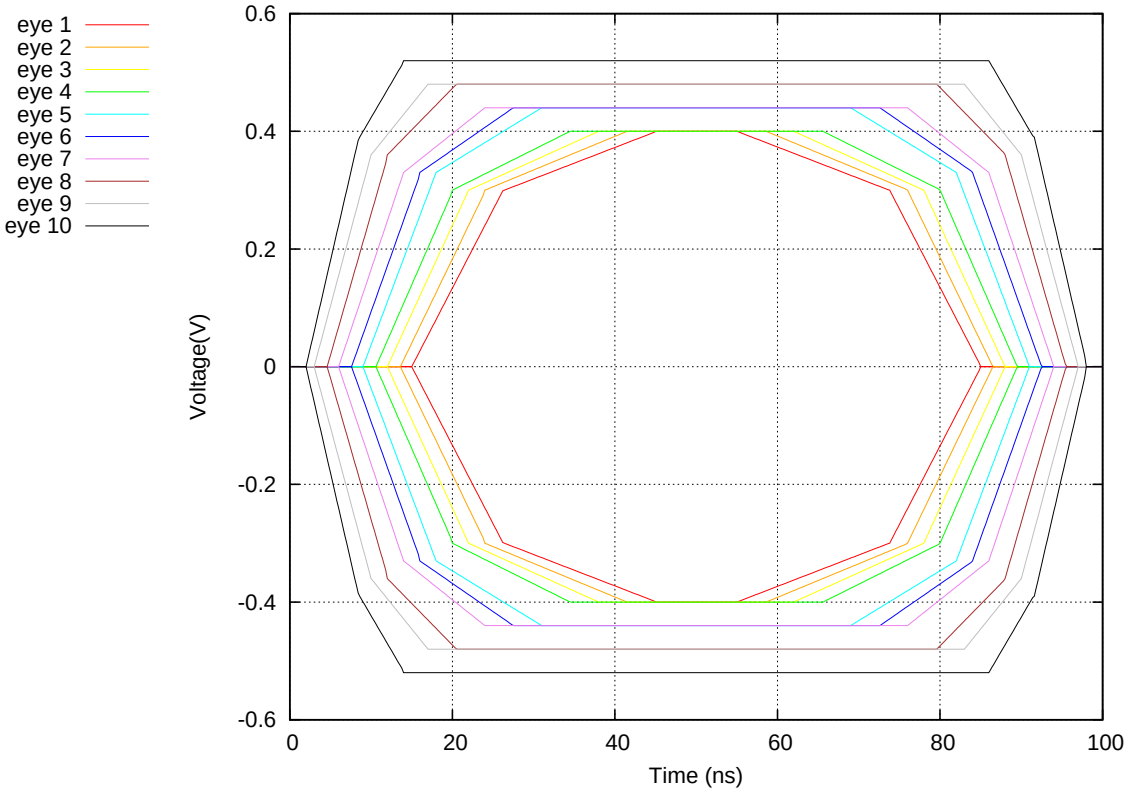

Figure 2.7: Minimum requirements eye diagram.

However, this single eye diagram is not very meaningful in the context of FlexRay™:

- It can be difficult to distinguish between asymmetric delay and larger durations of `DATA_0` or `DATA_1`
- The chronological synchronization between communication and eye diagram might be a problem, because there is no fixed point of reference in time available. It may be difficult to determine effective rising or falling edges of a single FlexRay™ bit.

To overcome these problems, a 10 bit eye diagram is used to analyze signal integrity. A standard compliant FlexRay™ frame consists of several parts, and is decomposed by the software application accordingly. The "transmission start sequence" (TSS), "frame start sequence" (FSS), "byte start sequence" (BSS) and "frame end sequence" (FES) are not in focus of the 10 bit eye diagram, but the communication part is. This way, each data byte, together with the last bit of the preceding BSS and the first bit of the subsequent BSS or FES, can be depicted using a single 10 bit eye diagram. Besides the graphical representation, a textual report is generated by the software application as well, pointing out problems and possible violations of the minimum eye diagram.

This approach is considered as a hybrid kind of, since it covers two levels of abstraction. First, the eye diagram is clearly targeted at the analog layer, mainly dealing with voltage thresholds. Second, there is the BSS synchronization, which involves the sample layer.

The minimum requirement eye diagram mentioned earlier can easily be created using the evolved software application. Based on five time-value pairs a 10 bit eye diagram is created, representing a minimum requirement for all 10 bits of the waveform under investigation. There is also the possibility to create a 10 bit eye diagram containing 10 individual curves. Based on the user's information, an output

file is created containing three columns: The first column contains points in time, and the second and third columns contain `BP` and `BM` voltage levels, at a given sample rate. This eye diagram file will be used to mark the minimum voltage levels during signal integrity analysis.

In context of that work a third function is available, representing an additional interface to the TEODACS framework. It allows the generation of sample layer log files (see 2.3.4.5).

Based on the first step, which extracted bus activity information out of an oscilloscope or simulation data file, the second step decomposes a FlexRay™ frame and generates a hierarchy of files and folders containing information about each part. After all, an interface to `gnuplot` exists, allowing to generate graphical output and visualize the discussed eye diagrams. The whole software application is available as a package of command line tools.

Chapter 3

Modeling Concept

The following chapter introduces concepts to address the problems described in earlier parts of this work. First of all, the overall TEODACS strategy is outlined in section 3.1.1. Then, a general approach is given in section 3.1.2, naming the tools and software products to be used. The most important FlexRay™ components involved are described in section 3.1.3. After that, concrete modeling and systems engineering concepts are described in section 3.2, covering the aspects of stimulus generation, modeling of automotive components using hardware description languages, and model verification and validation.

3.1 General Approach

3.1.1 TEODACS Strategy

Covering the left hand side of the TEODACS V-diagram, CISC's "System Architect Designer" allows easy integration of several system and hardware description languages using co-simulation. Low level automotive network simulations can be achieved this way, featuring a wide range of topologies, effective stimulus generation and fault injection possibilities. The right hand side of the TEODACS V-diagram incorporates a comprehensive laboratory setup, including FlexRay™ nodes from various hardware vendors, like Fujitsu, Freescale or Infineon. The software's co-simulation models are used for validation against the laboratory setup. Several specified interfaces connect both domains on different levels of abstraction: The analog level provides insight to continuous state variables over time, whereas the physical layer shows bit streams. Completing this picture, the data link layer represents entire frames. On top, the message and test case levels can be found. Those are not covered in particular within this thesis.

This integrated environment is applicable to accomplish the tasks described in section 1.3. A number of software modules and models already exist and are ready for use within the scope of this work. On the other side, a FlexRay™ hardware demonstrator is available, allowing the construction of arbitrary topologies. The interface between laboratory and co-simulation setups is represented by a special FlexRay™ communication node, the TEODACS active star, which can be controlled through a number of software scripts. All in all, TEODACS project's vision draws a complete picture of typical development stages within the area of automotive network design. As this approach relies on a number of hardware and software tools, the following sections are going to introduce these in a more detailed way.

3.1.2 Software

3.1.2.1 VHDL-AMS

VHDL-AMS is a language for describing digital, analog and mixed-signal systems. Its roots are based in the United States government's "Very High Speed Integrated Circuits" (VHSIC) program, which led to the first VHDL standardization in 1987. In the early nineties, a need for mixed signal modeling became apparent, and an IEEE working group was established, which initially defined an extension to

VHDL, named VHDL-AMS. The emerging draft was approved in 1999, becoming IEEE Standard 1076.1 "Definition of Analog and Mixed Signal Extensions to IEEE Standard VHDL" [APT03].

VHDL-AMS was chosen because it fills the gap for a standardized and convenient hardware description language allowing simulations on the analog level, with a seamless interconnection possibility to the digital abstraction level. All other advantages of VHDL-AMS, mentioned earlier in section 2.1, apply as well.

3.1.2.2 System Architect Designer (SyAD)

CISC's "System Architect Designer" is currently available in version number three. It is jointly developed by CISC and Graz University of Technology, Institute for Technical Informatics. SyAD® is a system design, partitioning, simulation and verification tool, which allows to process highly heterogeneous systems [sya09].

The current version supports a number of industry standard simulator tools, which can be configured through SyAD®'s user interface:

- Cadence Design Systems: Virtuoso® AMS Designer
- The Mathworks: Matlab® Simulink
- The Mathworks: Matlab® Stateflow
- Mentor Graphics®: ADVanceMS®
- Mentor Graphics®: Modelsim®
- Synopsys®: Saber®
- OSCI: SystemC
- Open Modelica: Modelica

SyAD® allows hierarchical designs through creation of basic and composite models. Each model seats within a block, which is parameter driven and has configurable properties. Besides the blocks name, its parameters and ports can be set individually. One symbol per block can be drawn in order to support SyAD®'s schematic view, which offers the possibility to setup complex systems just by inserting blocks and establishing interconnections between them.

A single block can be of different types, written in different languages, or even represent a composite model block, which houses other interconnected blocks. A block communicates with the outside world entirely through its ports, which can also be of different data types. The third type of models in SyAD® is called a "Testbench", which accommodates the system under test. A test bench is on top of each hierarchy and requires the insertion of at least one block onto the drawing pane.

Within the scope of this work, SyAD® is used to setup and co-simulate models in VHDL, VHDL-AMS and SYSTEMC languages. The simulator used to accomplish this, is described in the next section. Detailed descriptions of necessary prerequisites to simulation, the simulation itself and the subsequent data processing are shown in a corresponding application note [KK10], covering all tools involved, including SyAD®.

3.1.2.3 ADVANCEMS™

ADVANCEMS™ by Mentor Graphics®is used as primary simulator tool. It offers mixed-signal and mixed-domain analog/digital simulation using VHDL-AMS and Verilog-AMS languages, pure digital simulation through VHDL, Verilog, SDF, and VITAL languages. ADVANCEMS™ is IEEE compliant and matches several important standards. A graphical user interface is available.

ADVanceMS also supports multiple domain simulation, therefore not only electrical systems, but also mechanical or fluidic systems can be modeled and simulated. ADVANCEMS™ has the ability to include pre-compiled binary models, also known as IP blocks. ADVanceMS integrates seamlessly into SyAD®, and is available through SyAD®'s user interface. Special configuration parameters are accessible via the ADVANCEMS™ user interface. [Men08]

3.1.3 FlexRay Components

The following sections provide an overview of FlexRay™'s main components, used throughout this work. These include the communication controller, the bus driver, the active star, and the cables used to link the previous components in order to build networks.

A common FlexRay™ network consists of several participants plus one or more communication channels. The participants are also called communication nodes or simply nodes. The communication channel transmits information by electrical or optical means, however, only the physical layer has been specified. [Fle06a]

A typical FlexRay™ node includes a microcontroller or host, a FlexRay™ controller or communication controller (CC) as well as one or two bus drivers. The bus drivers are establishing a physical connection to the communication channel. The communication controller implements the logical FlexRay™ protocol. [Rau08]

This section heavily relies on FlexRay™'s specifications, defined in [Fle06a].

3.1.3.1 Communication Controller

There are two different types of communication controllers: The stand-alone type is an independent circuit and connected to the host via address- and databus or serial interface. The integrated type is placed next to the host controller in the same package. Depending on flexibility, performance and cost factors, one or the other type is used.

The communication controller has a number of interfaces. The interface to the host is implementation specific and has not been standardized. The interface to the bus driver is standardized and consists of three ports. "Receive Data" or `RxD` transmits the bus driver's received data to the communication controller. In the opposite direction, from the communication controller to the bus driver, data is transmitted using "Transmit Data" or `TxD`. Using the third low-active port "Transmit Data Enable-Not" or `TxEN` the communication controller signals when to send the data on `TxD`. [Rau08]

For the tasks given in section 1.2, a communication controller is not strictly necessary in general. However, there needs to be a mechanism, which is able to send compliant data via `TxD` and set `TxEN` accordingly.

3.1.3.2 Bus Driver

The bus driver is an important element when it comes to build a FlexRay™ system. It operates in a bi-directional way and converts binary signals to a trivalent differential signal. Its possible logical levels include high, low, and idle. The other way round, it converts a trivalent differential signal to a binary signal. Besides sending and receiving data, the bus driver has a lot more to offer.

The bus driver protects the communication controller an other components from electrical surge on the communication channel. Furthermore, it is capable of detecting errors on the communication channel. According to the FlexRay™ specification [Fle06a] it has to support supply voltage monitoring. There are a number of additional, yet some optional, functions defined.

Figure 3.1 shows a block diagram of a FlexRay™ bus driver. On the left side, the digital logic interface can be seen. "Stand-By Not" (`STBN`) and "Error Not" (`ERRN`) are mandatory ports for the host interface, "Enable" (`EN`) is optional. As mentioned in section 3.1.3.1, the communication controller has three ports (`RxD`, `TxD` and `TxEN`). The optional "Bus Guardian Interface" has two ports, named "Bus

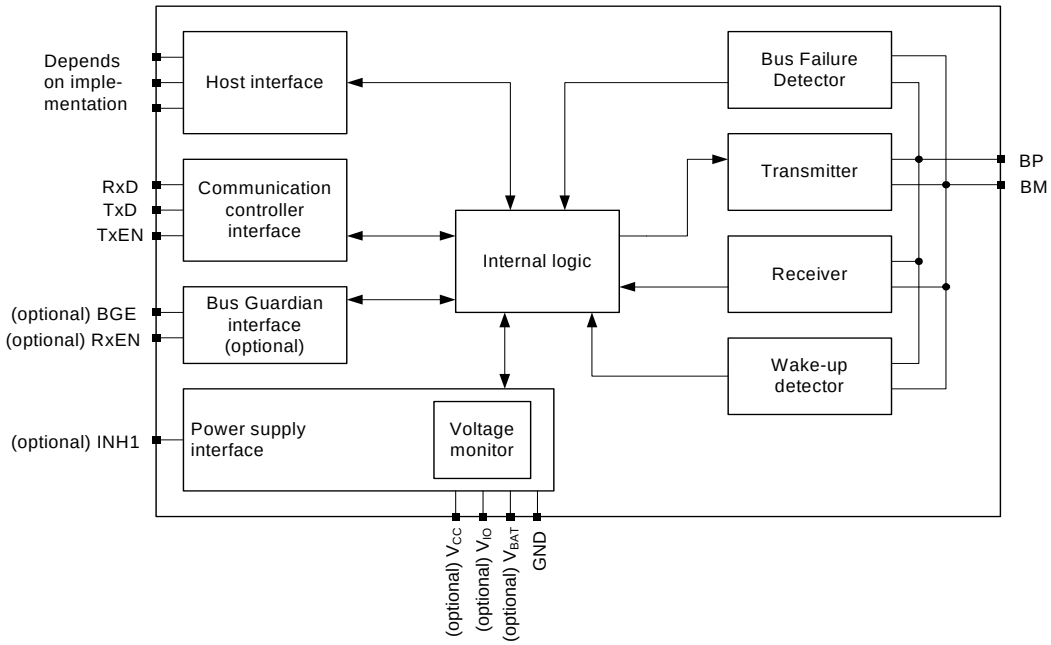

Figure 3.1: FlexRay^{TM} bus driver block diagram [Fle06a, p. 37].

Guardian Enable" (BGE) and "Receive Enable Not". The supply voltage interface has an optional port named "Inhibit" (INH), next to four additional, partial optional, ports (Vcc, Vio, Vbat and GND) which realize the voltage input. A "Wakeup" port (WAKE) is optional as well. Finally, the differential bus lines are connected to the bus driver's "Bus Plus" (BP) and "Bus Minus" (BM) ports. [Rau08]

All mentioned interfaces, the transmitter and receiver modules, wake-up detector and bus error detection are controlled by an internal logic module. A temperature monitor protects the system from blistering heat. It's always possible to add new features to the bus driver, but doing so is not specification conform and therefore not standardized at all.

The bus driver runs an internal state machine, which switches between four states, two of them being optional.

BD_NORMAL : The bus driver is able to send and receive data streams on the bus. A high level on INH (if available) signals, that the bus driver is not in BD_SLEEP state. The bus wires are supplied with a specified bias voltage.

BD_STANDBY : BD_STANDBY is a so-called low power mode. The bus driver is not able to send data streams to the bus, nor receive data streams from the bus. If applicable, the bus driver is able to detect wakeup events. The power consumption is reduced compared to BD_NORMAL. A high level on INH (if available) signals that the bus driver is not in BD_SLEEP state. Additionally, the bus wires are terminated to GND via receiver common mode input resistance.

BD_SLEEP : BD_SLEEP is an optional and so-called low power state. Therefore, the bus driver is not able to send data streams to the bus, nor receive data streams from the bus. All present wakeup functions are operational. The power consumption is reduced compared to BD_NORMAL and INH is floating.

BD_RECEIVEONLY : The bus driver is able to receive data streams from the bus, but is unable to transmit data streams to the bus. A high level on INH (if available) indicates, that the bus driver is not in BD_SLEEP. All bus wires are supplied with a specified bias voltage.

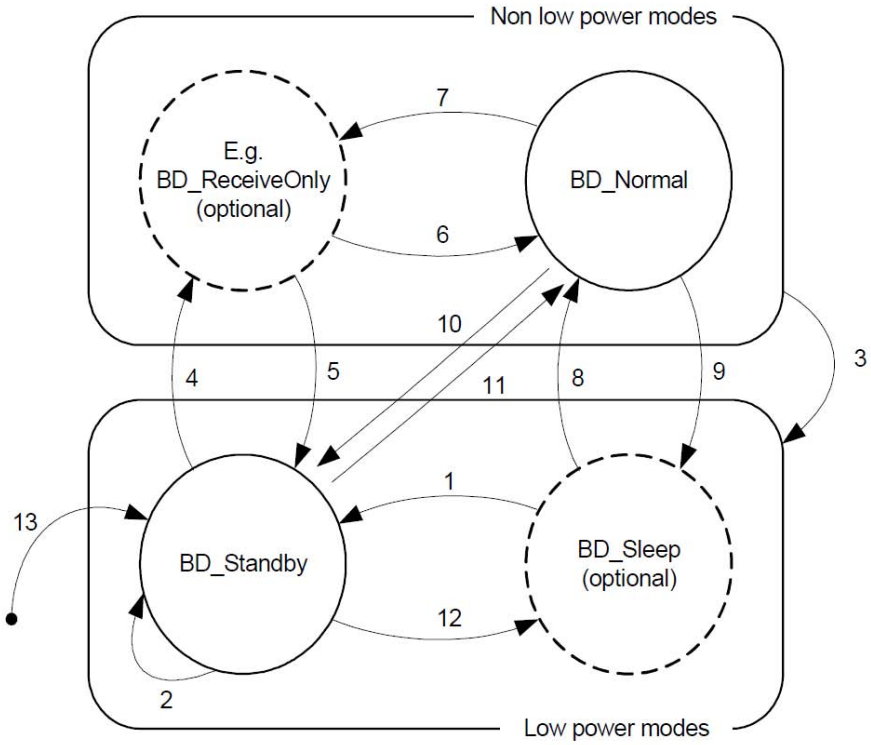

Figure 3.2: Electrical bus driver state diagram [Fle06a].

Transition	Condition
1-3	Detection of Wake-up event or undervoltage condition
4-11	Host command
12	Host command or detection of V_{BAT} or V_{IO} undervoltage condition
13	Power on wake-up

Table 3.1: Simplified bus driver state transitions [Fle06a].

State transitions are happening upon commands from the host via the bus driver-host interface, detection of wakeup events or due to undervoltage conditions. A detected undervoltage situation forces the bus driver from any non low power mode to a low power mode. In regular use, the bus driver switches back and forth between `BD_STANDBY` and `BD_NORMAL`. The exact undervoltage conditions are skipped within this context, since those are not strictly necessary for this work's purpose. Figure 3.2 shows the bus driver's states and table 3.1 shows its corresponding state transitions.

Bus Driver – Communication Controller Interface As already mentioned in section 3.1.3.1, the interface between the bus driver and the communication controller comprises three signals (`TxD`, `TxEN` and `RxD`), which are used to transfer binary data streams to the bus driver and also read binary data streams from the bus driver. According to the specification, a timeout needs to be implemented to ensure that `TxEN` is not permanent on logical low level. Table 3.2 shows the bus drivers logic used to transmit data to the communication channel.

If the communication controller wants to read data from the bus channel, it is confronted with the following situation. Depending on the bus wire signal, `RxD` changes the way defined in table 3.3.

operation mode	TxEN	BGE	TxD	resulting bus signal
BD_NORMAL	high	X	X	Idle
	X	low	X	Idle
	low	high	low	Data_0
	low	high	high	Data_1
Low power modes	X	X	X	Idle_LP

Table 3.2: Signaling on bus wires in dependency of bus driver input states [Fle06a, p. 41].

operation mode	signal on bus wires	wakeup event	RxD
	Idle_LP	X	high
BD_NORMAL &	Idle	X	high
BD_RECEIVEONLY	Data_0	X	low
	Data_1	X	high
BD_STANDBY &	X	detected	low
BD_SLEEP	X	not detected	high
All other	X	X	Product specific

Table 3.3: Resulting RxD signal from bus driver to communication controller [Fle06a, p. 41].

3.1.3.3 Active Star

The active star poses an important element in extended FlexRay™ networks. Basically, it acts as a kind of signal distributor between a number of branches. There is a general set of functions and components specified, which every active star has to obey, in order to fulfill FlexRay™ standards. Concerning the implementation, a variety of possibilities exists.

The basic functionality of the active star is the active data transfer, that means, a data stream received on one branch of the active star is re-sent immediately on all other branches of the active star [Fle06a, p. 68]. The active star not only works on a pure physical layer level, it merely processes the incoming data stream and distributes it. Additionally, the signal gets a refresh in terms of quality, at the expense of a slight delay.

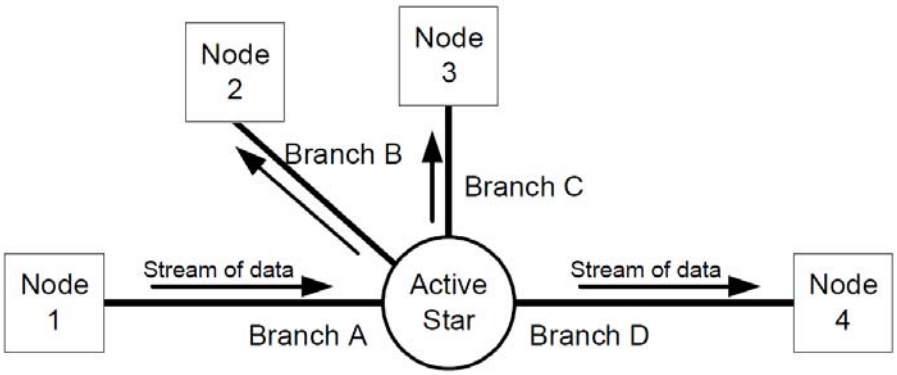

Figure 3.3: Active star basic functionality [Fle06a, p. 69].

Operation Modes Just as the bus driver, the active star has operation modes which support energy saving functions. When the active star is in AS_NORMAL mode, it is able to send and receive data, as described earlier in this section. There is a variable timeout between 640 and 64000 milliseconds

Transition	Condition
1	Wake-up
2	Sleep timeout
3	Undervoltage condition
4	No undervoltage on V_{CC}
5	Power on (V_{BAT} or V_{CC})

Table 3.4: Active star state transitions [Fle06a].

specified, which puts the active star to AS_SLEEP mode when no activity on all branches occurs. In AS_SLEEP mode, the active star is not able to send and receive data, but its wakeup functions are active. The detection of a wakeup symbol forces the active star to enter AS_NORMAL. AS_SLEEP is a low power mode, where power consumption is reduced to a minimum level. In case of an undervoltage condition, the active star remains in AS_STANDBY. As soon as voltages increase to a specified level, the undervoltage state is left immediately. Figure 3.4 shows the active star's states and table 3.4 shows the corresponding state transitions.

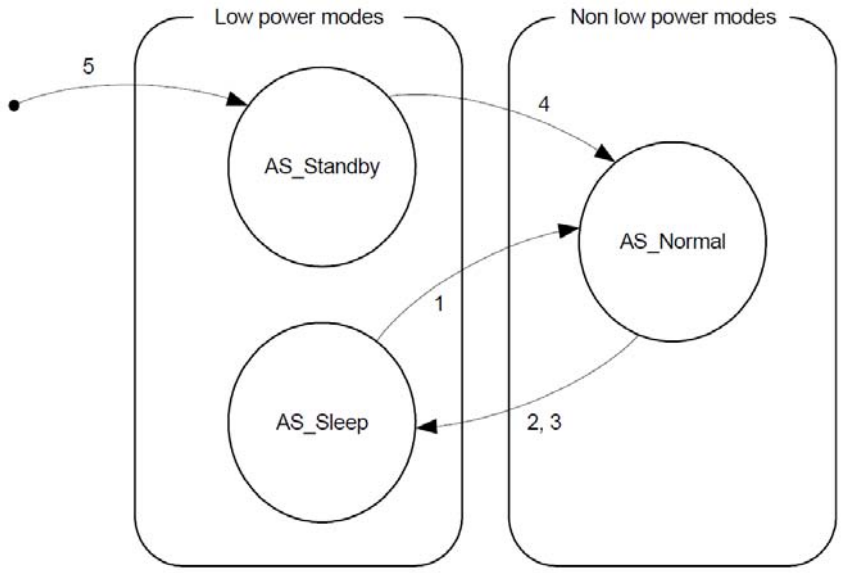

Figure 3.4: Active star state diagram [Fle06a, p. 73].

Active Star Timings The specification advises a maximum propagation delay from idle to active mode of 450 nanoseconds, the opposite direction from active to idle is given with 400 nanoseconds. Both transitions may last up to 30 nanoseconds. Minimum requirements are not defined. Concerning the signaling, the propagation delay of positive edges is specified with up to 250 nanoseconds, as well as the negative edges. Since the propagation delay of positive and negative edges may differ, a so called asymmetric delay is defined as the absolute value of both delays difference. Additionally, the active star may truncate a data stream in length, which results due to shortening of the Transmission Start Sequence (TSS). The Transmission Start Sequence is used to activate active stars and receiving nodes and is cut off as soon as the devices are activated. The difference between the incoming and outgoing Transmission Start Sequence is called truncation, and is specified with a maximum of 450 nanoseconds.

Branch Operating States Similar to the active star itself, each branch can enter a different operating state. `AS_IDLE` can be reached on detection by the central logic unit or by the branch itself. As soon as any activity is detected, the branch enters `AS_ACTIVE`. In this state, a received data stream is passed on to the central logic unit. If `AS_ACTIVE` was reached on request by the central logic unit, the received data stream is transmitted to the branch. When entering `AS_ACTIVE`, a timeout starts to run, which is specified between 1500 and 15000 microseconds. If the branch runs into this timeout, it is excluded from any further communication. This state prevents defective nodes from disturbing the whole networks communication and is called `AS_FAILSILENT`.

3.1.3.4 Wiring Harness

Besides the bus driver, the wiring harness is critical in terms of signal integrity. The market offers a range of cables with different properties, some of them being FlexRay™ certified. FlexRay™ does not define any specifications for cables, however, in order to pass a standardized conformance test, a number of requirements made to the cables must be fulfilled. Therefore, FlexRay™ only makes recommendations about cables. Mainly, FlexRay™ makes use of copper cables. According to the specification, other bus channels like fiber are not prohibited, yet lack of standardization. The communication over fiber optical lines in vehicles has been approved earlier to be working flawlessly [Ros98]. Most™ for example, relies on an optical physical layer and is used for multimedia applications in current upper class vehicles. However, there is an increased inherent cost factor, which is currently not bearable for all car manufacturers or even car variants.

Bus Cables First of all, FlexRay™ recommends shielded cables for bus cable use. The most important property is impedance, which is given with 90Ω at 10 MHz signal frequency. The specific line delay may last up to 10 nanoseconds per meter. Using a 5 MHz signal, an attenuation of up to 82 dB per kilometer is allowed.

Other Cables There are recommendations for power supply cables and connectors as well. Those are not recapitulated at this point, because they are not relevant for this thesis' goals.

3.2 Concept Overview

The upcoming section outlines the integration and interconnection of the previously described elements used to simulate a FlexRay™ automotive network. Special attention is drawn to the early implementation and integration to VHDL-AMS and SyAD®.

3.2.1 Stimulus Generation and Data Exchange

Whenever a network engineer plans a communication network, the design process is driven by iterations. Unforeseen effects, design mistakes and specification defiances are causing network malfunctions. In order to track down those issues and locate the sources of faulty behavior, a repeated test pattern as network stimulus is very valuable. Usually, many iteration steps are necessary during development, verification and validation of models written in VHDL, VHDL-AMS, and SYSTEMC. Hence, during model integration and simulation phases, some well defined component- and network-stimuli are needed. In the TEODACS project, hardware and software testernodes were developed to solve this issue. The emerged SYSTEMC software testernode model is used in this work, giving the possibility to inject pre-defined FlexRay™ compatible data. Key functionality of this software model is its multi-level capability. Signal quality and integrity analysis requires low-level data access – so this work will mostly cope with the testernode's sample level capabilities.

```
architecture wrapper of tr_ip_block is
begin
        tr: entity flextrans_cisc(epl)
        generic map(
                t_uv_recover => t_uv_recover,
                t_sleep => t_sleep,
                r_closed => r_closed,
                r_open => r_open,
                fall_time => fall_time,
                rise_time => rise_time,
        )
        port map(
                GND=>wr_gnd, BAT=>wr_bat, CC=>wr_cc, IO=>wr_io,
                    WAKE=>wr_wake, BP=>wr_bp, BM=>wr_bm,
                    STBN=>wr_stbn, EN=>wr_en, ERRN=>wr_errn,
                    RxD=>wr_rxd, TxD=>wr_txd, TxEN=>wr_txen,
                    BGE=>wr_bge, RxEN=>wr_rxen, INH=>wr_inh,
                    t_case=>wr_t_case
        );
end architecture wrapper;
```

Listing 3.1: Bus driver IP block wrapper.

3.2.2 Bus Driver Model

The bus driver model provided by CISC semiconductors is entirely written in VHDL-AMS and available as IP block, which needs to be provided as an external library through the simulator setup. The implemented features in accordance with CISC's documentation [CIS08] are:

- Device status management
- Supply voltage monitor
- Read-out capability
- Temperature monitor
- Local and remote wake-up
- Transmitting and Receiving
- Bus-error and state flags
- Signal timings
- Thermal behavior

Concerning the use within VHDL-AMS, a so called wrapper encapsulates this IP block. This requires a mapping of all wrapper ports to the IP blocks ports. All properties and configuration parameters, known as "generics" in VHDL-AMS, need to be mapped to the IP block as well. Listing 3.1 shows the architecture of such a wrapper, mapping generics and ports to the IP block.

As stated in [KKS+10], there is a FlexRay™ communication controller model available in context of TEODACS, written in SYSTEMC. It would be possible to use this model to control the electrical bus drivers, in order to support the holistic multi-layer approach of TEODACS. However, this thesis' focus is on physical layer and does not make use of this model. Due to the absence of a communication controller realizing the FlexRay™ protocol and controlling the connected bus drivers accordingly, all

bus drivers are set up and controlled in a manual way. Dedicated blocks generating piece-wise linear output waveforms[1] are used to control the analog input ports of every bus driver.

3.2.3 Active Star Model

The full active star model is created of two major components. First, the active star's logic modules, written in VHDL and provided by project partner University of Applied Sciences FH Joanneum Kapfenberg [Net08]. This code is used to generate a SyAD® compliant active star model, comprising basic- and composite-models. All its internal and external interfaces are of digital kind. Second, the bus driver model is used to build an active star model according to section 3.1.3.3. To connect the bus driver's `terminal electrical` type ports to the active star's `std_logic` type ports, analog-to-digital and digital-to-analog conversion respectively, needs to be performed.

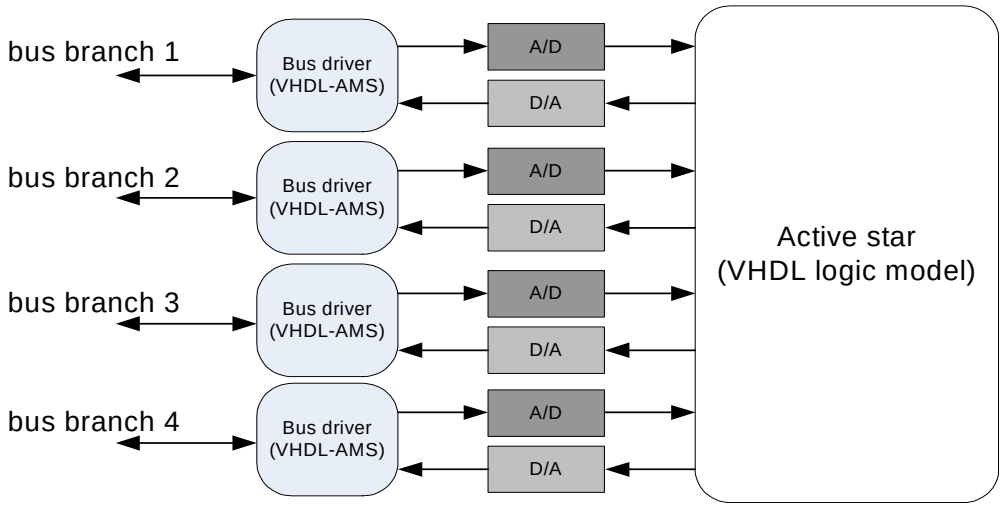

Figure 3.5: Active star analog/digital conversion scheme.

To get the VHDL logic modules up and running, some additional modules need to be created. Every digital simulation model requires a clock signal to change its state and advance in time. The ideal or maximum clock speed to be determined according to all digital or analog modules involved, depending on their critical path and rise or fall times respectively. Immediately after simulation start, the entire model needs to be reset in order to enter a defined state. Therefore, the possibility to signalize an asynchronous reset needs to be created. After those initial tasks, it has to be ensured that the included state machines work as desired.

The available VHDL model consists of nine modules: one top-module with three sub-modules below it, and another five sub-sub-modules on the second level. Table 3.5 [Net08] gives a short overview. `struct` denotes a module only containing instances of sub-modules. `fsm` denotes a finite state machine module. `bhv` stands for a behavioral model and `rtl` identifies a register transfer level module.

As declared earlier within this thesis, one of the major objectives is to provide SyAD® compatibility. Therefore, these modules are ported to basic blocks and composite models. This allows easy multiple instantiation and dedicated module tests. Additionally, better traceability and fault-injection possibilities are achieved.

3.2.4 Wiring Harness & Termination Models

As discussed in section 3.1.3.4, two cable modeling techniques are available and subject to evaluation within the TEODACS project. Besides proper cabling, the concept of termination is important in every

[1] See VDA FAT-AK30 library `FUNDAMENTALS_VDA` available at `http://fat-ak30.eas.iis.fraunhofer.de` for details.

entity name	architecture name	VHDL file name	hierarchy
active_star_device	struct	active_star_device_struct.vhd	Top level
flexray_driver	fsm	flexray_driver_fsm.vhd	1
voltage_monitor	bhv	voltage_monitor_bhv.vhd	1
active_star_core	struct	active_star_core_struct.vhd	1
as_mode	fsm	as_mode_fsm.vhd	2
voltage_event_ctrl	fsm	voltage_event_ctrl_fsm.vhd	2
signal_handler	fsm	signal_handler_fsm.vhd	2
branch_mode	fsm	branch_mode_fsm.vhd	2
sleep_timer	rtl	sleep_timer_rtl.vhd	2

Table 3.5: VHDL active star module list.

electrical network containing long lines compared to the signal wavelength. The term ***electrical length*** is often used to refer to this problem. Furthermore, additional measures for network stabilization are taken in the laboratory setup, which need to be modeled in order to achieve good simulation results.

3.2.4.1 Transmission Line Modeling

Basic possibilities to model a transmission line have been introduced in section 2.1.4. The creation of an arbitrary length cascaded lumped-element equivalent circuit model in VHDL-AMS is possible using a `for`-loop and the capability to instantiate components at elaboration time. Using such a technique, the output ports of a previous line element are mapped to the input ports of the subsequent line element. The resulting transmission line has two input- and two output-ports, just as a single line element. In order to answer the question, how many line elements are necessary to produce a meaningful result, a tradeoff between simulation time, accuracy and parameter availability is required. A large number of line elements containing discrete components modeled through quantities take long time to simulate in VHDL-AMS. On the other hand, the number of line elements should not be too small, compared to the length of line, to ensure correct simulation of wave propagation. Third, as described in 2.1.4, the parameters measured in units per length need to be determined in a sufficient way. Usually, cable manufacturers provide data sheets together with their cables. If those are not available for some reason, values have to be determined by measurement. In terms of error propagation it is advantageous to measure long lines and calculate shorter line sections, than vice-versa.

A few interesting concepts about twisted-pair telephone transmission lines are given in [Lao02]. First of all, all resistances within a conductor are frequency-dependent, meaning that the RLGC model's "R" is frequency dependent because of the skin-effect, and that the contained "G" is frequency dependent because of dielectric loss. For this thesis, these effects are considered to be negligible. However, the existence of these are kept in mind to explain possible deviations of the simulation result. Next, that article describes the RLGC model as a balanced line model, which means that the modeling engineer needs to be aware of interactions between both conductors (capacitances, conductances). This may be difficult to achieve, due to lack of information or accurate measurement possibilities. Concluding, to model any twisted pair cable, a two-conductor approach is applied and explained in detail in section 4.4.

3.2.4.2 Termination Modeling

The concept of termination is crucial for stable operation of any electrical network. A bus driver's sender is likely to be of low-impedance output type, whereas receivers are commonly designed to have a high-impedance input. A high impedance termination equals an open circuit, and a low impedance termination equals a short circuit. According to [Joh02, p. 28] or [JG03, p. 39], the characteristic impedance Z_0 needs to match the lines load Z_l in order to appear of infinite length to the sender. In this special case, the energy generated by the sender is completely absorbed at the receiver's end of the cable.

Otherwise, in case of a termination mismatch, a so called reflection occurs on the line, which makes the energy arriving at the receivers end returning to the sender. The amount of energy traveling backwards can be determined by calculating the reflection coefficient ρ.

FlexRay™ does not specify a dedicated termination type, but does advise some useful structures. In order to achieve better EMC performance, FlexRay™'s specification points out an application hint in shape of a split termination [Fle06b, p. 10], consisting of two matched resistors R_{T_1} and R_{T_2}, one resistor R_1 and one capacitor C_1. Figure 3.6 shows the arrangement of these components. R_1 is suggested to be smaller or equal to 10Ω, C_1 should have $4700pF$ and the matched resistors should coordinate with the used cables characteristic impedance Z_0.

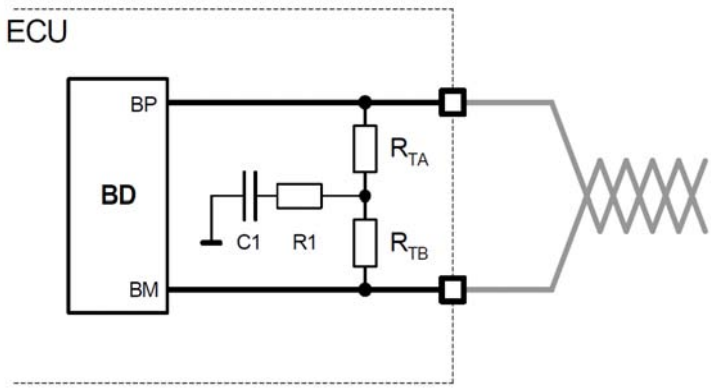

Figure 3.6: Split termination as suggested by [Fle06b].

This termination model is used within this work wherever necessary, the split termination is modeled as basic block in SyAD®.

3.2.4.3 Common Mode Choke Modeling

Additionally to the termination explained in section 3.2.4.2, a common mode choke is modeled to reach a close-to-reality-simulation result, as suggested by the specification. The function of the common mode choke is to force the current in both signal wires to be of the same strength [Fle06a, p. 11], using two coupled inductances in series to the bus wires. Therefore, a high impedance arises for common mode signals.

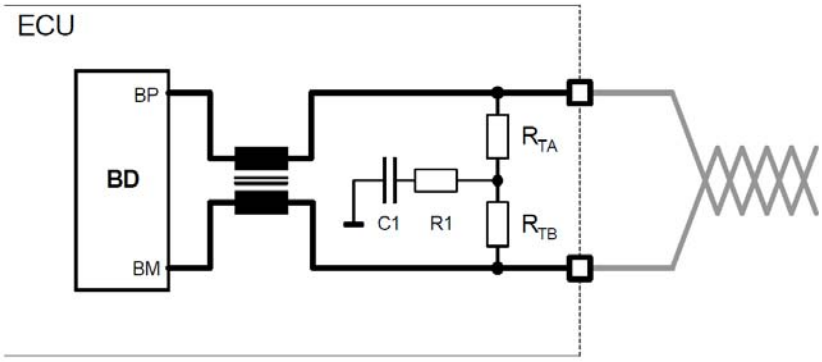

Figure 3.7: Split termination with common mode choke [Fle06b, p. 11].

Figure 3.7 shows the common mode choke between the ECU and the previously explained split termination structure. According to FlexRay™'s specifications, the resistance per line should be smaller or equal to 1Ω, the main inductance should be larger or equal to $100\mu H$ and the stray inductance should

be within a $1\mu H$ limit. Equation 3.2 shows the formulas of both inductances involved, used to model this common mode choke in VHDL-AMS. They may also be recognized as "transformator equations".

$$u_1 = L_1 \frac{di_{L_1}(t)}{dt} + M \frac{di_{L_2}(t)}{dt} + i_1 R_L \tag{3.1}$$

$$u_2 = L_2 \frac{di_{L_2}(t)}{dt} + M \frac{di_{L_1}(t)}{dt} + i_2 R_L \tag{3.2}$$

where M denotes the mutual inductance:

$$M = k\sqrt{L_1 L_2} \tag{3.3}$$

k represents the coupling coefficient. Basically, k tells about the quality of coupling between both inductances. R_L equals the ohmic resistance per bus line.

This common mode choke model is used within this work wherever necessary in conjunction with the split termination model from section 3.2.4.2. The common mode choke is modeled as a basic block in SyAD®, and builds a composite block together with the termination model.

Additional electrostatic protection elements, as suggested in [Fle06a, p. 19], are available in the context of TEODACS.

3.2.5 Model Verfication and Validation

Model verification ensures the operability of a given model by asking if the model was built in a right and correct way. This includes the model's parameters as well. By comparing the model with its real system counterpart, a calibration process is carried out and allows a revision of the considered model. By constantly comparing and revising the created model, validation is performed. Therefore, model validation can be seen as an iterative process.

In this work, verification is based on best practice solutions. There is a lot of expertise knowledge available, as described in chapter 2. Using these basic principles together with the TEODACS approach, new models evolve quickly. Once the models are available, validation through simulation is performed. By building small and simple topologies, or just segments respectively, quick validation is possible since simulation times will be kept on a low level. Some simple topologies may include:

- A single bus driver model and optional termination.
- A single bus driver connected to short circuit line of different length and optional termination.
- A single bus driver connected to open circuit line of different length and optional termination.
- A simple point-to-point connection using two bus drivers and a line of different length and optional termination.

After successful validation of these small network segments, they are used to build larger networks. Assuming, that simulation time increases with the number of differential equations, representing state variables, these networks unavoidably lead to longer simulation times. If these models are successfully validated against the laboratory's hardware setup, it is expected that composite higher order systems validate against the hardware setup as well.

The validation process utilizes two different tools. Both are based on the physical layer, but include interfaces to higher layers as well. First of all, the waveform viewer ***EZwave*** by Mentor Graphics offers a wide range of possibilities to analyze signal metrics [Men07]. Besides well known time domain metrics like rise and fall times, slopes and levels, frequency- and statistical-domain values can be measured as well. These metrics and functions can also be used to compare different signals. However, simple signal metrics deliver no information about the logical correctness of the signal. This is where the second tool

comes into play. It is introduced in section 2.3.5, and is based on the automated creation of ten bit eye diagrams and textual reports.

To achieve a credible validation result, both simulation and hardware platforms ought to be stimulated using the same input data, yielding significant output data. This requirement is entirely fulfilled thanks to the modeling of the SYSTEMC software testernode [Kru08] and the TEODACS active star [Net08]. Once more, both systems are on the same level but on opposite sides of the TEODACS V-diagram, ensuring integrated analysis possibilities of the resulting system.

Chapter 4

Implementation of Models and Tests

The next sections provide deep insight into the modeling process of each component. Now that all needed models are known from a behavioral point of view, let's have a closer look on implementation details. Besides the internal components of each model, parameterization has shown to be a very important task which can dramatically improve simulation results and increase the degree of validation. Therefore, the process of correct parameter retrieval requires our full attention. Once all models and their parameters are set up properly, larger FlexRay™ network structures are constructed and simulated. A special measurement campaign together with TEODACS project partners supported these tasks in a very valuable way.

4.1 Bus Driver Implementation

This chapter allows a deeper insight into the bus driver's implementation, as far as the provided intellectual property module allows. Section 4.1.1 gives an overview of all implemented components. The only components introduced in detail are transmitter and receiver, since these have been identified as critical modules, where more information about internals are available courtesy by CISC semiconductor.

4.1.1 Internal Organization

The bus driver's model is made up of eight components, four of them are strictly necessary for data transmission to and from the bus channel. Others include additional functionality, like supply voltage monitoring or wake up detection.

Input-Output Interface This block basically provides an interface for the communication controller. Based on a number of voltage levels and ratios, given as parameters, this block provides analog type external and digital type internal ports. Therefore, its main functionality is limited to analog-to-digital and digital-to-analog conversion. External analog visible ports are: `EN`, `TxD`, `TxEN`, `RxD`, `INH`, `BGE`, `RxEN`, `STBN`, and `ERRN`. Exactly the same ports are available internally as of digital kind: `EN_dig`, `TxD_dig`, `TxEN_dig`, `RxD_dig`, `INH_dig`, `BGE_dig`, `RxEN_dig`, `STBN_dig`, and `ERRN_dig`. Power supply voltages are connected via separate ports, `Vbat`, `Vio` and necessarily `GND`.

Digital Logic The bus driver's heart in general includes a state machine and several processes, necessary to model the output logic and the next state logic.

4.1.2 Transmitter

The transmitter's model primarily consists of two analog ports accessing the bus `BP` and `BM` lines, another three analog ports for supply voltage `Vcc`, thermal control, and ground `GND`. Digital interfaces

```
1  r_bp1_eff == r_bp1'ramp(rise_time, fall_time);
2  r_bp0_eff == r_bp0'ramp(rise_time, fall_time);
3  r_bm1_eff == r_bm1'ramp(rise_time, fall_time);
4  r_bm0_eff == r_bm0'ramp(rise_time, fall_time);
5
6  if (domain = quiescent_domain) use
7          i_high_BP == v_high_BP/r_open;
8          i_low_BP == v_low_BP/r_open;
9          i_high_BM == v_high_BM/r_open;
10         i_low_BM == v_low_BM/r_open;
11 else
12         i_high_BP == v_high_BP/r_bp1_eff;
13         i_low_BP == v_low_BP/r_bp0_eff;
14         i_high_BM == v_high_BM/r_bm0_eff;
15         i_low_BM == v_low_BM/r_bm1_eff;
16 end use;
```

Listing 4.1: Current calculations using variable resistors as switches

connect to the digital logic block using BGE, TxEN, and TxD. The transmitter's internal structures and their parameterization are most important for close-to-reality simulation results, especially with respect to signal integrity.s BP and BM lines, another three analog ports for supply voltage Vcc, thermal control, and ground GND. Digital interfaces connect to the digital logic block using BGE, TxEN, and TxD. The transmitter's internal structures and their parameterization are most important for close-to-reality simulation results, especially with respect to signal integrity.

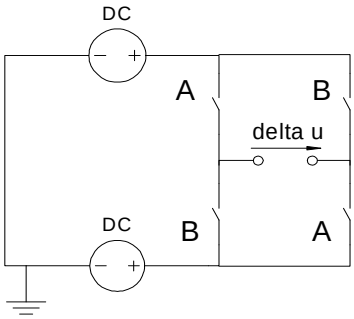

Figure 4.1: Transmitter's voltage bridge schematic.

As shown in figure 4.1, the transmitter's behavior is reproduced through four switches, which are modeled using a simple variable resistor. Since FlexRay™ requires a differential voltage signal on the bus channel, the resistors are operated pairwise. This way, either a positive or negative differential voltage is applied to the bus line, between BP and BM. However, FlexRay™ uses a trivalent bus signal, therefore two idle states are defined as well, Idle and Idle-LP, where the latter stands for "low-power".

The parameterization of the given voltage bridge initially posed a problem, due to the modeling of transient characteristics. The VHDL-AMS attribute 'ramp increases or decreases a signal value in a linear way with constant slope, with respect to a given time interval. Therefore, it is just instinctive to model a switch as a variable resistor. This variant is also proposed in [Hes03]. The disadvantage of this solution becomes imminent when using the resistors resulting value for voltage or current calculations. An open switch can be expressed using any high-resistive value, a closed switch can not be written using an arbitrary small number.

In lines 12 to 15 of listing 4.1 the appearing quantity voltage values are divided by the calculated

resistor values. For very small values of `r_bxx` numerical problems arise due to the constant integer increase of `'ramp`, leading to unwanted steps during state transitions. There may be a number of solutions available to improve the switches transition function. Nevertheless, we try to cope with a slightly larger integer value for closed switch states anyway, because of the required internal resistance of the bus driver when sending data to the bus channel. More on this in section 4.1.4.

4.1.3 Receiver

The bus driver's receiver has four analog and three digital ports. `BP` and `BM` are used to connect to the bus channel, `Vcc` and `GND` are used for power supply and reference, respectively. `RxEN`, `RxD`, and `enable_Rx` are of digital type and connect the receiver to the digital logic unit. The receiver makes heavy use of VHDL-AMS' attribute `'above`, to detect threshold value exceedances. These thresholds are given as parameters, allowing individual adaptation to replicate different receiver behaviors.

There are two VHDL-AMS processes running within the receiver, and a third one was added to improve the simulation output performance. The first one implements the detection of bus activity in general, and sets a flag accordingly. The second process is in charge of converting the analog bus voltage levels to digital signals. In situations encountering good overall signal integrity with steep edges, this method works as proposed by the authors at CISC. However, problems arise when signal integrity is labile, when heavy reflections occur due to wrong or bad termination, or other voltage fall-offs due to faulty components. In such cases, very short voltage dips lasting around 10 nanoseconds are triggering the threshold values too early in time, causing unwanted modifications of the signal's pulse widths. Therefore, a method to model noise cancellation and signal stability is needed, as they appear more or less naturally in integrated circuits.

This is where the third process comes into play. Basically, it allows to check the incoming differential bus signal for stability over a given time interval. This time interval is specified using the additional `stability_time` parameter of the bus driver, 30.0 nanoseconds have proven to be a reasonable value. At the current simulation time, the process determines if the shifted input signal's last event occurred at least before the given time interval in the past. If that is the case, the signal was stable over the last given time interval, and the digital output port delivers the rising or falling edge of the signal. The same behavior could possibly have been achieved with the VHDL-AMS attribute `'stable`, however, this attribute was found to be not supported by ADVANCEMS™.

4.1.4 Inner Resistance and Termination

FlexRay™'s concept of termination is straightforward to understand, but under certain circumstances hard to implement. It is not enough just to look at the bus driver's `BP` and `BM` ports, it is necessary to know what is going on behind these ports. As already pointed out in section 3.2.4.2, the receiver requires a high impedance input, to ensure that no energy is consumed out of the transmission lines, since transmitters are not designed to deliver energy through their output ports. This requirement is easy to match, because in real-hardware systems amplifier circuits are used, which in general have high impedance input stages. This way, any additional termination measures at the bus driver's input becomes immediately effective.

At the sender's side things are a bit different. The impedance at the output must not be high, since this would mean the ports are undriven. A very low impedance, as any ideal voltage source with reference to ground represents, would be impossible to terminate correctly. Any connected line would always see just the very low impedance at the end, regardless of termination. Concluding from these facts, it is necessary to parameterize the sender in a proper way, to achieve an effectively working termination and close-to-reality signal integrity in succession.

Parameter Extraction: In order to replicate the bus voltage and termination behavior of a hardware bus driver, the TEODACS active star was chosen for investigation. Main parameters to be determined

were `r_open` and `r_closed`, with respect to an applied termination attached to `BP` and `BM`. The measurement setup included the TEODACS active star only, without any lines or termination elements connected. Using an oscilloscope, the bus output voltages of a single port (`BP`, or `BM` respectively) were measured under a variety of different ohmic loads. Table 4.1 lists the collected values.

$R\_l[\Omega]$	$U_{low}[V]$	$U_{high}[V]$	$\Delta U[V]$	$t_{rise}[ns]$	$t_{fall}[ns]$	$T_{high}(bit)[ns]$	$T_{low}(bit)[ns]$
10	2.35	2.59	0.24	na	na	94	106
47	2.0	2.95	0.95	13.0	10.0	98	102
100	1.74	3.22	1.48	11.0	10.5	98	102
150	1.6	3.4	1.8	9.0	11.5	98	102
220	1.52	3.52	2.0	7.5	9.0	99	101
470	1.24	3.76	2.52	7.5	8.5	99	101
1000	1.04	3.96	2.92	7.5	8.0	100	100
3300	0.8	4.08	3.28	7.5	7.5	100	100
10000	0.68	4.16	3.48	7.5	8.0	100	100
Air	0.6	4.16	3.56	8.0	8.5	100	100

Table 4.1: `BP` measurement outputs.

The first column shows the resistor loads available at the laboratory. The TEODACS active star was stimulated using the LogFile generator as proposed in [ATK+09] and demonstrated in [AWK+09]. The resulting FlexRay™ compliant frames were measured with respect to voltage levels, rise- and fall-times and bit length.

The last row shows measurement data for no resistor placed between the bus driver's ports. Without any load R_l, voltage levels were 4.16V for high level and 0.6V for low level. These exact values are used to supply the voltage bridge described in section 4.1.2. To remain compatible to different supply voltage levels, these values are modeled as variable factors in the VHDL-AMS transmitter.

It is commonly known, that (FlexRay™-)bus driver manufacturers are calculating their circuits in a way, such that signal integrity remains good even under tough conditions. According to the FlexRay™ specification [Fle06a], any differential voltage level between 0.6V and 2.0V is valid. Early oscilloscope measurements have shown that high levels of `BP` and `BM` are located around 3.1V to 3.2V, whereas low levels are located around 1.8V to 1.9V. Note that table 4.1 represents `BP` values. Values for `BM` would be similar, with interchanged U_{low} and U_{high} data columns.

During regular operation with a correct network topology attached, the bus driver's steady state output measures around 1.2V. This represents our model's calibration target. Without any load, the voltage drop between positive and negative supply voltage amounts to 3.56V, R_l is known from table 4.1, and the four equal voltage bridge resistors R_{B_x} are connected in series to R_l, therefore it is possible

$R\_l[\Omega]$	$R\_Bx[\Omega]$
10	69.17
47	64.56
100	70.27
150	73.33
220	85.80
470	96.98
1000	109.59
3300	140.85
10000	114.94

Table 4.2: Transmitter's internal resistor-bridge values.

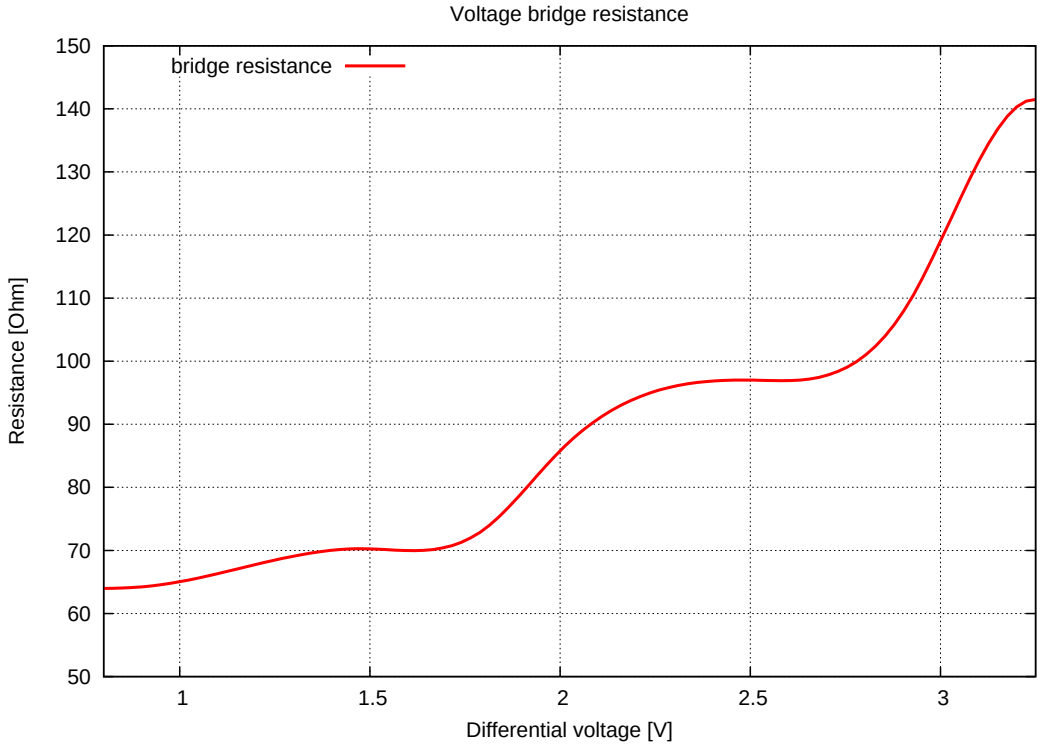

Figure 4.2: Bus driver bridge resistance.

to calculate R_{B_x} values for every R_l, as described in equation 4.2:

$$R_{B_x} = \frac{\frac{3.56-\delta U}{2}}{\frac{\delta U}{R_l}} \tag{4.1}$$

$$R_{B_x} = \frac{R_l(3.56-\Delta U)}{2\Delta U} \tag{4.2}$$

Image 4.2 shows a graph of calculated values for R_{B_x}. The requested calibration target lies in the area around 60Ω to 70Ω. It is worth to note, that the determined value of R_{B_x} is of behavioral nature. Any internal resistance affecting BP and BM outputs, caused by physically present electrical components, is therefore expressed through R_{B_x}.

A second method to interpret these results, is to look at R_l as a termination resistor. The split termination used in the laboratory consists of $2 \cdot 47 = 94\Omega$, which is slightly below the single 100Ω resistor used in the preceding calculations.

Termination is implemented in VHDL-AMS language using a SyAD® basic block model, following the descriptions of section 3.2.4.2. Since every bus line interface is going to be terminated, either matched or highly resistive, the termination is included in a composite model, together with the also required common mode choke model as introduced in section 3.2.4.3. This common mode choke model written in VHDL-AMS consists of two coupled inductances, which are expressed through VHDL-AMS quantities and their appropriate differential equations. The termination and the common mode choke models are electrical quadrupoles, with respect to ground. Therefore the resulting composite model is a quadrupole as well, containing the discussed basic blocks interconnected in series.

The only parameter of the split termination model is its resistive value. The common mode choke requires the two inductances L_1 and L_2, the coupling factor k and the ohmic resistance R_L of the inductances. All parameters are provided via SyAD® as generics to VHDL-AMS.

4.2 Active Star Implementation

4.2.1 Changes to the Original Active Star Model

This section describes the implementation of a specification compliant active star using a combined application of VHDL and VHDL-AMS, as introduced in section 3.2.3.

In the top module `active_star_device`, the entirely digital written `flexray_driver` was replaced by `tr_ip_block` modules, containing the proposed wrapper for access to the intellectual property bus driver module. The `flexray_driver` module contained a finite state machine model, which emulated a simple digital bus driver interface using timed concurrent statements.

The `active_star_core` module included in the `active_star_device` model originally contained all remaining model blocks, together with some logic expressions. These logic expressions are now pulled out of this block and put into a separate one, in order to keep functionality at bottom level. This approach complies with SyAD®'s hierarchical structuring. The four `branch_mode` blocks were used to control the digital `flexray_driver` units, their outputs are now redirected to digital-to-analog converters and in succession to the bus driver models. The `active_star_mode`, `voltage_event_ctrl`, `sleep_timer` and `signal_handler` blocks basically remain in place.

There is an interface to a communication controller available, all related signals were kept despite the absence of such a device model. The `sleep_timer` and `signal_handler` blocks however, needed some adjustments to work properly without valid signaling from a communication controller.

4.2.2 Internal Organization

This section presents the conversion result of VHDL modules to SyAD® blocks, and gives an overview of the achieved model architecture. Figure 4.3 (a) shows the digital model structure before the integration process. The composite model's depth was two. Figure (b) shows the fully integrated mixed signal model. Its depth increased to three levels, due to the insertion of a new composite model at the very top of the model's structure. All necessary blocks to control the active star were added at this point.

Since the `branch_mode` and `signal_handler` blocks, which are controlling the bus drivers, communicate with the `active_star_mode`, `sleep_timer`, and the newly generated `active_star_core_functional` blocks, it was simply a design-driven decision to leave the bus drivers on top level. All signals needed to control the bus drivers are passed on to the topmost level. This way, all analog-to-digital and digital-to-analog conversions are performed on top, and a clear separation between analog and digital models is achieved on top level.

4.2.3 Active Star Block Details

Supply Voltage: The supply voltage blocks V_{IO} and V_{CC} are set to 5 volts DC current, whereas V_{BAT} is set to 10 volts DC current. The supply voltage blocks are powered on just a few milliseconds after simulation start.

Bus Guardian: The bus guardian block's output is set to 4.5 volts, shortly before the beginning of data transmission, to ensure bus access.

Wake: This block provides the possibility to send a wake-up signal to the bus drivers, `as_mode` and `active_star_core_functional` blocks.

One Bit DAC: This block represents a one-bit digital-to-analog converter. Its only parameter specifies the reference output voltage. Any digital type `std_logic` input is converted to either zero or reference voltage on the `terminal electrical` output port.

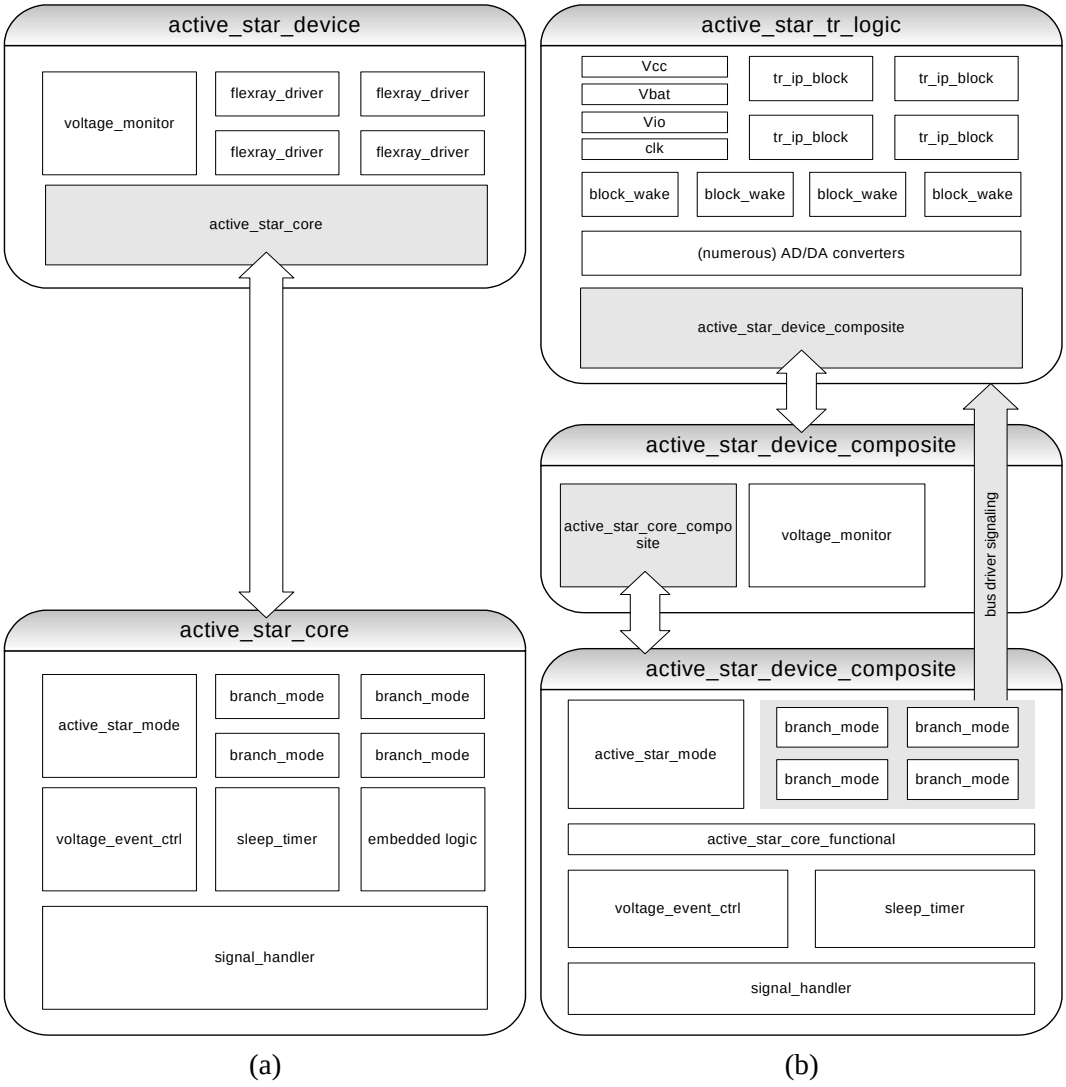

Figure 4.3: Digital (a) and analog (b) interfaces of the active star model.

One Bit ADC: This block represents a one-bit analog-to-digital converter, or comparator respectively. Its only parameter specifies the reference voltage used for conversion. Any analog type `electrical` quantity is converted to either binary zero or binary one on the digital type `std_logic` output port. The conversion process works on the base of threshold detection.

Clock: The system's clock is generated using this block's only parameter, the periodic time. A value of 12.5 nanoseconds therefore corresponds to 80MHz, which allows the active star to operate at a data transmission speed of 10 megabits per second.

Reset: This asynchronous reset puts all connected digital systems to a defined state. Its output voltage is 5 volts.

Host Command, Communication Controller: Since these signals are not available or relevant, dummy blocks are applied to keep the affected ports to logical *low* or ***high***.

Unused output ports: The VHDL model initially contained some `std_logic` type ports marked with the VHDL keyword `open`. Those could have been used to connect further models or simply debug the active star logic model. Since the simulator deployed in this thesis was not able to interpret this statement, a special block was set up to act as a signal sink.

Output Resistors: Due to the absence of a communication controller, analog `electrical` output ports like `INH` must not left unconnected. A simple $1k\Omega$ resistor connected to ground helps to resolve this problem.

Thermal Outputs: Every bus driver model block has a `thermal` type output port, indicating the current estimated temperature of the bus driver during operation. The purpose of this circuitry is the early detection of overheating. A dedicated thermal network, representing an appropriate specific heat capacity, takes care of this output.

4.3 Network Stimulus Generation

As discussed in section 3.2.1, the TEODACS concept enforces direct interfaces between the laboratory setup and the co-simulation environment. Based on common file formats, data exchange for simulation and validation becomes not only possible, but convenient for such tasks.

4.3.1 Simple Test Signal

The primary goal was to prepare a fast simulating, significant stimulus file, in order to investigate basic physical network properties. This simple test signal does not comply to FlexRay™'s standard signaling. The basic idea was to generate a stimulus file comprising "very short" pulses to gain knowledge about the transient behavior of the proposed cables, bus driver, and other models. Figure 4.4 shows this signal captured directly at the software testernode.

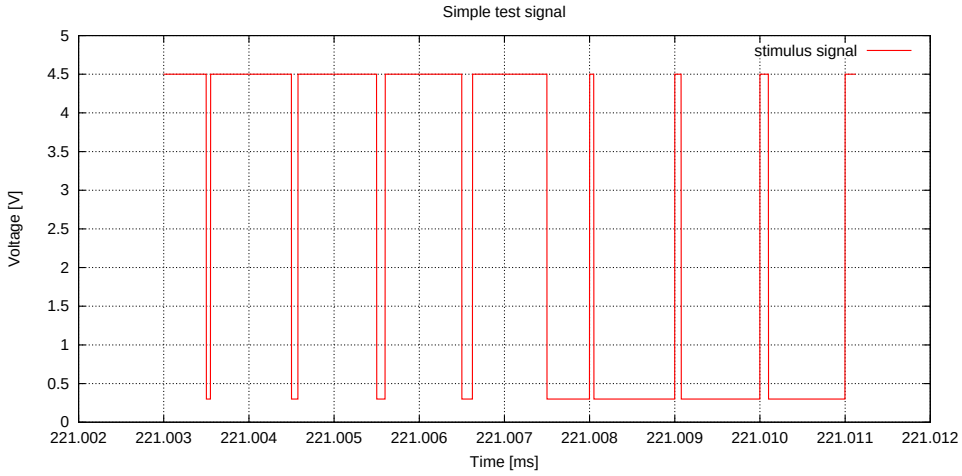

Figure 4.4: Simple test signal.

The chosen pulse lengths are of 50, 75, 100, and 125 nanoseconds length, with reasonable space of time between them. The chosen pattern appears twice in the test signal, normal and inverted, to get an impression of positive and negative switching properties and behavior. Note that 100 nanoseconds conform to one FlexRay™ bit, the shortest piece of information possible, at a data rate of 10 Megabits per second. For detailed information about FlexRay™ timing constraints, refer to [Fle06a, p. 84].

4.3.2 FlexRay™ Frame Test Signal

The FlexRay™ frame test signal contains complete FlexRay™ compliant frames, created using the log file generator as explained in section 4.3. This log file contains several cycles of communication. In most cases, the first few frames are sufficient to draw an early conclusion about signal integrity or run some replay tests. Figure 4.5 shows the first frame out of four simulated ones, captured directly at the software testernode.

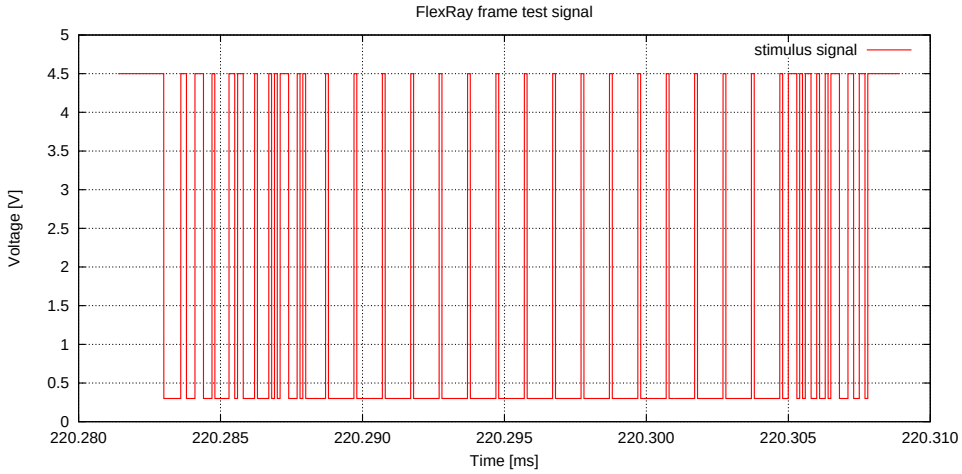

Figure 4.5: FlexRay™ frame test signal.

4.4 Cable Model Implementation

The following section provides a detailed view on the implementation of the proposed cable models. In general, there are two cable models available, featuring different properties. First of all, the cables used in the laboratory are investigated and measured. Then the two models are introduced and discussed. Afterwards a comparison tries to determine one model for further investigations with respect to signal integrity.

4.4.1 RLGC Model

The first model discussed in detail is the widely known, so called "RLGC model". The term "telegrapher's equations" is often used to refer to this model as well. Its basic structure is already described in section 3.1.3.4. However, this basic transmission line model is adapted very often to the engineer's needs. For this reason, many modifications can be found throughout the literature. For modeling the given FlexRay™ cables found in the TEODACS laboratory, a two-conductor transmission line model was established and described with VHDL-AMS using SyAD®.

Section 4.4.3 showed the measurement and calculation of the needed R, L, G, and C parameters, representing a two-conductor cable having a characteristic impedance of 100Ω. Figure 4.6 outlines the layout of the proposed cable model in SyAD®.

Each conductor is modeled using a separate RLGC model described in a SyAD® basic block. Each basic block has five parameters:

- `C_LINE` – The line's capacitance per meter
- `G_LINE` – The line's shunt conductance per meter
- `L_LINE` – The line's inductance per meter

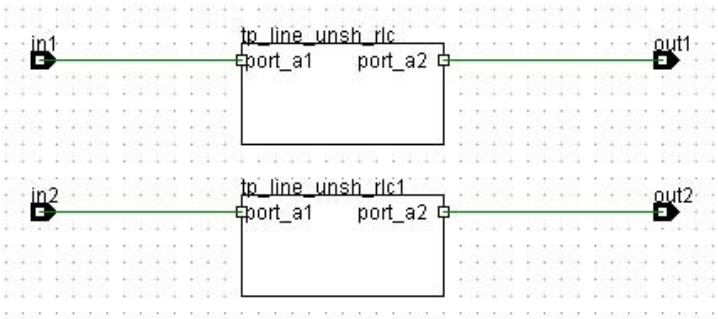

Figure 4.6: Transmission line pair in SyAD®.

- `R_LINE` – The line's resistance per meter
- `LENGTH` – The line's length in meters

Using the lumped element model introduced in section 3.2.4.1, all constants are broken down to $1/100$ of their initial value. Thus, one lumped element of the proposed cable equals 1 centimeter. This poses a good tradeoff between simulation speed and accuracy: Assuming ideally modeled conditions, considering signals of 10-80 MHz and a propagation speed of approximately 0.7 times the speed of light [Joh02, p. 81] the resulting wavelength makes up to a few meters. So if it is possible, to model a transmission line fulfilling these assumptions, a 20 MHz signal traveling along a 10.5 meter transmission line, 1 signal period fits the length of the transmission line and needs to pass 1050 segments of the RLGC model. This is considered enough to observe various effects of signal propagation in FlexRay™ networks. In SyAD®, both RLGC conductor models are placed within one composite model, having two pairs of electrical ports.

4.4.2 Behavioral Model

The second cable model available is based on the work by [SHM01]. One VHDL-AMS model represents one conductor of the twisted pair cable found in the TEODACS laboratory. For this reason, it is based on the same five parameters as the RLGC variant, and has four electrical ports as well. Two conductor models are placed in one composite model, replicating the desired behavior of the twisted pair cable.

Concerning the conductor models' internals, eight quantities are used within the model, which could be described as a reactive kind of model. Initially, thirteen constants are calculated, and during simulation time eight equations are processed whenever necessary.

The cables' length is also required to perform calculations. Similar to the RLGC model, a segmentation parameter is used to adjust simulation step size and precision. This behavioral model is characterized by high simulation speeds, obviously due to the low number of VHDL-AMS quantities.

4.4.3 Parameter Extraction

Subject of this section is to determine all parameters needed for modeling of the given cables, used in the laboratory's setup. According to [JG03, p. 34], most important transmission line properties are characteristic impedance, delay, high frequency loss and crosstalk.

Crosstalk refers to a common phenomenon, by which a signal traveling through one transmission line causes an undesired effect in another transmission line running close to the first one. There are a number of countermeasures to attenuate this effect, like twisting corresponding pairs at a different rate within one cable and using a properly shielded cable. Crosstalk is not in focus of this work, for two reasons. First, the mentioned countermeasures are usually effective enough nowadays, and second, the laboratory's setup does not include any multiple twisted pair cables.

High frequency loss is an umbrella term used for a number of effects arising when higher frequencies are applied to the transmission line. These include the Skin effect, the Proximity effect, surface roughness and dielectric effects. Detailed descriptions of these aspects are available in [JG03], and are not discussed within this work, for a number of reasons.

Nevertheless, the remaining two terms are quite important for modeling a transmission line in VHDL-AMS. Characteristic impedance and delay are even closely related to each other. In order to extract appropriate parameters from the laboratory's cables for use in simulation models, some measurements and calculations are needed. Most eminent parameter of a transmission line is undoubtful its characteristic impedance, which denotes the ratio of voltage to current experienced by a signal traveling in one direction along a transmission line [JG03, p. 39]. Considering the lumped element model introduced in section 4.4.1, the series impedance z and the shunt admittance y are frequency dependent and can be written as

$$z = j\omega L + R \tag{4.3}$$

$$y = j\omega C + G \tag{4.4}$$

Deriving the telegraphers equation using these complex valued variables, the expression for characteristic impedance Z_C becomes

$$Z_C(\omega) = \sqrt{\frac{j\omega L + R}{j\omega C + G}} \tag{4.5}$$

As described in [JG03, p. 43], in modern transmission lines G becomes almost zero, obviously because of the used manufacturing techniques and materials. The R term tends to change with frequency. However, R and G can be neglected at higher frequencies, because $j\omega L$ and $j\omega C$ are leading to a steady plateau of impedance, which is almost constant at high frequencies. This fact allows us to terminate a transmission line with a single resistor. Mathematically written,

$$Z_0 \triangleq \lim_{\omega \to \infty} Z_C(\omega) \approx \sqrt{\frac{L}{C}} \tag{4.6}$$

which is only valid between frequencies above the LC and Skin effect mode, and below the onset frequencies of waveguide mode. In this region, the characteristic impedance is comparatively flat with frequency, and Z_0 becomes a single real value. The variable Z_C is used for characteristic impedance as a function of frequency.

The laboratory's setup included cables designed for FlexRay™ networks, made by ***Kromberg & Schubert*** [Kro07]. Those cables are characterized by a mean impedance of $100\Omega \pm 10\Omega$ from 1MHz to 50Mhz, at 10MHz the impedance is given with $100\Omega \pm 7\Omega$. Delay is given with less than or equal to 6 nanoseconds per meter. Finally the conductor resistance amounts a maximum of 55.5 milliohm per meter. The DC conductance G between both conductors is assumed to be fairly small, since very small currents are driven through the well isolated cable. It is estimated to 1 nanosiemens per meter. Therefore, only L and C are left to determine. Both are critical due to their dependence on frequency. The upcoming calculations were already proposed by [ZK09], but repeated here due to the unknown cables and measuring devices used during their experiments.

As already shown, the (slightly simplified) characteristic impedance Z_0 is

$$Z_0 \approx \sqrt{\frac{L}{C}} \tag{4.7}$$

By knowing one conductors' delay Δt for a given cable length l it is possible to calculate the signals' propagation velocity. Using the longest available cable from the laboratory's setup (11m), the signal propagation delay along this conductor was measured 59.215 nanonseconds. Therefore,

$$\frac{1}{\nu} = \frac{\Delta t}{l} = \frac{59.215ns}{11m} = 5.3832[ns/m] \tag{4.8}$$

which confirms the datasheet by ***Kromberg & Schubert***. Signal propagation velocity equals simply the inverse of the delay, and allows the calculation of L and C, if the cables' characteristic impedance is known. Since the delay referred to one single conductor out of two, the cables' characteristic impedance has to be divided by two. By considering both conductors up and down the line, the overall characteristic impedance works out to 100Ω, as documented.

$$\frac{1}{\nu} = \sqrt{LC} \tag{4.9}$$

$$\nu = \frac{1}{\sqrt{LC}} = 1.8576 \cdot 10^8[m/s] \tag{4.10}$$

$$Z_0 = \sqrt{\frac{L}{C}} = 50\Omega \Rightarrow 2500C = L \tag{4.11}$$

$$\nu = \frac{1}{\sqrt{2500C^2}} = \frac{1}{50C} \tag{4.12}$$

$$C = 107.66363 \cdot 10^{-12}[F] \tag{4.13}$$

$$L = 269.15909 \cdot 10^{-9}[H] \tag{4.14}$$

With the calculation of L and C, the estimation of G, and R out of the manufacturers datasheet, all four parameters needed to setup an RLGC cable model are known.

4.4.4 Comparison RLGC vs. Behavioral Model

To compare both models initially described in section 3.1.3.4, two methods are used. The first method is to compare the modeling itself and possible requirements of both types. Second, the validation against laboratory measurements is important. By applying fairly basic network topologies, a confrontation between both models against hardware measurements is inevitable. Additionally, simulation time is an important criteria, when it comes to business applications.

Model Comparison and Computational Complexity Both cable models are based on the same fundamentals, the "Telegraphers' Equations". The derived RLGC-model poses a convenient and well-understood way of modeling an electrical cable. Whereas the RLGC model is made up of a number of discrete elements, the behavioral model is made up of one single element. The only electrical components representing a devices state are energy storing devices, like inductances or capacitances. VHDL-AMS utilizes quantities to match this fact. Since a single lumped element of a RLGC model includes one capacitance and one inductance per conductor, four quantities per segment are necessary. A twisted pair, 10 meter cable model, featuring 100 segments per meter, therefore counts $2 \cdot 2 \cdot 1000 = 4000$ quantities. Concluding, these will result in 4000 differential equations, which need to be solved over time. The behavioral model consists of a total number of eight quantities and a total number of eight recurring equations, which need to be solved and calculated over time. From the point of computational complexity, the behavioral model causes considerably less efforts than the pure RLGC model.

Both models are represented in time domain, but the pure RLGC model may appear more intuitive to the experienced engineer. Small changes to the model or even derivations of it are definitely easier to accomplish using the pure RLGC model. In such a case, the behavioral model or a frequency-domain based model would need to undergo a complete re-calculation, which might be difficult when using more sophisticated higher-order transfer functions. Again, both models utilize the exact same set of parameters, as described in section 4.4.1.

Validation through Measurement Comparison The pure RLGC and behavioral cable models are available in VHDL-AMS language and were ported to SyAD®. To compare the performance of both models and finally validate them, a few simple test benches were set up. For validation, the same test benches were assembled using the hardware of the TEODACS FlexRay™ Xpert.Lab laboratory.

4.5 Topology and Test Bench Implementations

This section demonstrates and explains the network topologies used in SyAD®'s test benches and the laboratory. The overall idea is to generate simple topologies first, for transceiver behavior and cable model validation. Based on these basic experiments, more sophisticated networks are constructed, using the basic segments as some sort of building blocks.

4.5.1 Simple Topologies

The topologies, or even just network segments, shown in this section were used during the model verification process first. Fast simulating, yet demonstrating elementary physical effects, these structures provided early simulation results.

Bus Driver Only This was used to verify the bus driver's output voltage, in order to validate the calculated results achieved in section 4.1. The only model used in this experiment was the bus driver, with its bus interface connected to a variable load resistor. The stimulus was provided over the bus drivers `TxD` interface, carrying the simple test signal as specified in section 4.3.1.

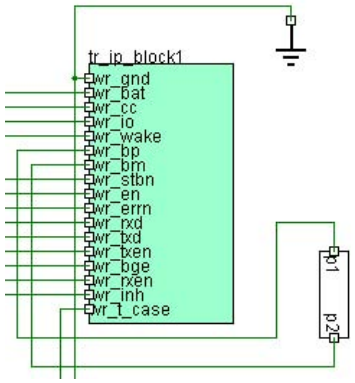

Figure 4.7: Bus driver in SyAD® with a simple resistor load

Bus Driver, Short Circuit Cable In this setup, a single bus driver and one cable is used. One end of the cable is connected to the bus driver, the other end is short circuit. The general idea of this configuration is to ensure the interaction of the bus driver with the cable model. If the cable behaves in a correct manner, any signal traveling down the line is reflected at the far end of the line and arrives back at the bus driver.

Basically, a proper termination matching the cables characteristic impedance avoids any reflections. Put into physical words, a mismatched line is not able to absorb the arriving energy at its end, and the only way dealing with this fact is to reflect the arriving wave. To calculate the proportion of energy being reflected back along the cable we need to know the characteristic impedance Z_0 and the terminating impedance Z_t [Joh02]. Therefore, the so called *reflection coefficient* ρ is expressed through

$$\rho = \frac{Z_t - Z_0}{Z_t + Z_0} \tag{4.15}$$

An ideal short circuit cable has a terminating impedance of $Z_t = 0\Omega$. Thus,

$$\rho = \frac{0 - Z_0}{0 + Z_0} = \frac{-Z_0}{+Z_0} = -1 \tag{4.16}$$

Since ρ has a range of $[-1, +1]$, the entire energy is reflected at the short circuit end of the cable and the occurring voltage drop across the short circuit cramp is canceled. The reflected signal arriving back at the sender appears inverted, without any amplitude loss (except possible losses occurring within the cable). Figure 4.8 shows the given setup. Using SyAD®, a single wire primitive is used to establish a short circuit, which means that VHDL-AMS is just connecting the two ports of the line's end.

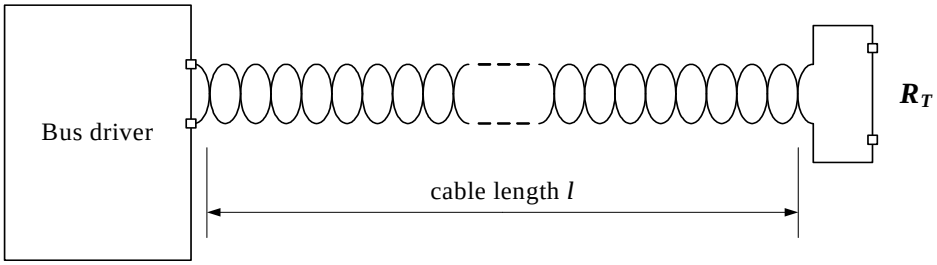

Figure 4.8: Bus driver with short-circuit transmission line.

Bus Driver, Open Circuit Cable This setup includes, just like the previously described one, a single bus driver and one cable. One end of the cable is connected to the bus driver, the other end makes an open circuit with $Z_t = \alpha$ (infinitely high). According to the descriptions in the preceding paragraph, the reflection coefficient ρ becomes

$$Z_t = \alpha\rho = \frac{\alpha - Z_0}{\alpha + Z_0} = \frac{\alpha}{\alpha} = +1 \tag{4.17}$$

In this upper extreme case, the occurring voltage drop across the open circuit resistance works out to a maximum. The reflected signal arriving back at the sender appears unmodified, without any amplitude loss (except possible losses occurring within the cable). Figure 4.9 shows the given setup. Using SyAD®, a high impedance cable end can be expressed through a "high valued" resistor between both bus lines, plus two optional resistors towards ground. Experience has shown that values in the order of magnitude $1.0 \cdot 10^9$ are sufficient.

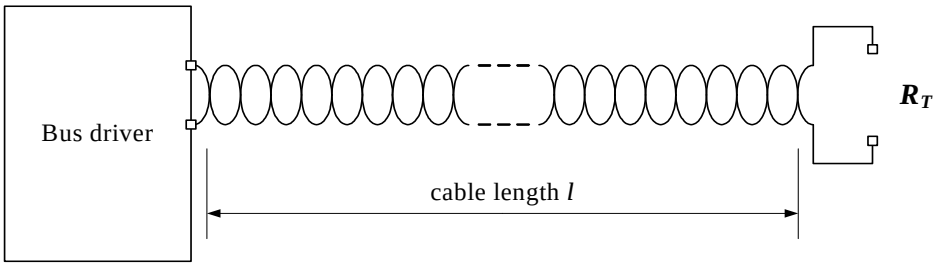

Figure 4.9: Bus driver with open circuit transmission line.

Point to Point Connection The previous two setups described are only of scientific value. They do not make any sense in terms of data transmission and are no valid FlexRay™ topologies at all. The setup described in this paragraph already represents a valid FlexRay™ topology, containing two bus drivers and one cable. According to the suggested termination strategy both cable ends need to be terminated, in order to achieve good signal quality without reflections. Using termination elements matching the cables' characteristic impedance ($Z_t = Z_0 = 100\Omega$), the reflection coefficient ρ becomes

$$\rho = \frac{Z_t - Z_0}{Z_t + Z_0} = \frac{100 - 100}{100 + 100} = 0 \tag{4.18}$$

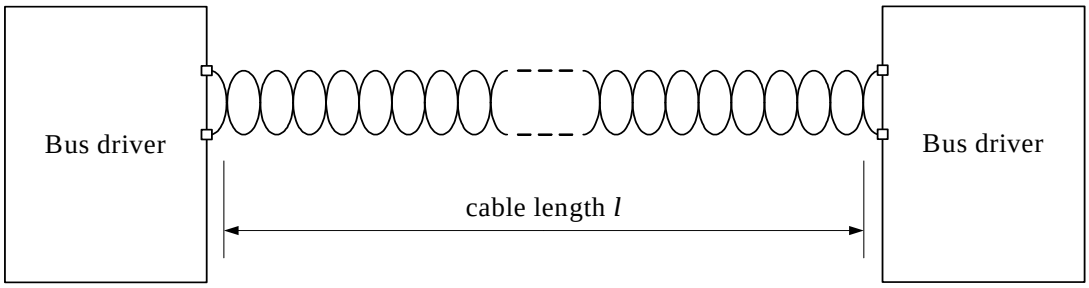

Figure 4.10: Point-to-point connection with two bus drivers.

Thus, no signal reflection occurs at all. Figure 4.10 shows the discussed point-to-point topology. The cables by "Kromberg & Schubert" used in the TEODACS laboratory are designed to meet FlexRay™'s recommendations. Their characteristic impedance is given with 100Ω plus some tolerance margin. The termination elements available in the laboratory are build as introduced in 4.1.4, using two 47Ω resistors. Therefore, the total terminating resistance sums up to 94Ω, which poses a mismatch to the line:

$$\rho = \frac{94 - 100}{94 + 100} \approx 0.03 \tag{4.19}$$

$$R_{loss} = 20 \cdot log(0.03) = -30dB \tag{4.20}$$

According to this solution, the mismatched line should have no severe impact on signal quality, a reflection coefficient of just 0.03 seems to be practically negligible: Only three hundredth of the signals' amplitude are going to be reflected. The return loss (R_{loss}) according to the formula presented in [Joh02] amounts $-30dB$. Equation 4.20 shows the calculation.

4.5.2 Measurement Use Cases

Due to recent activities in the area of bus driver development and topology design, a measurement campaign was organized in context of the TEODACS project. A number of project partners joined researchers from *The Virtual Vehicle Competence Center* and *Institute for Technical Informatics, Graz University of Technology*, like austriamicrosystems AG and *University of Applied Sciences FH Joanneum Kapfenberg*. Goal of this arrangement was the validation of the proposed simulation environment, using the methods described in section 2.2.2. In general, a FlexRay™ topology was set up in the laboratory and in the simulation environment in parallel. By utilizing a FlexRay™ compliant oscilloscope by *LeCroy*, an additional interface between both domains was established, complementing the previously mentioned tester node. This way it was possible to exchange stimuli and outputs as well, allowing a direct comparison between them. As stimulus, the earlier described simple test file and a number of FlexRay™ compliant frames were used. To judge the results, two techniques were applied, explained later in chapter 5. The upcoming paragraphs describe and reason the arranged topologies.

Topology 1: The first topology investigated consists of only two FlexRay™ nodes and five cables of various lengths. Figure 4.11 shows the setup. On the left hand side of the network, the TEODACS active star (number one) is used as a sender. It is terminated according to FlexRay™'s recommendations using a $2 \cdot 47\Omega$ split termination. An 11 meter cable leads to the first 0.2 meter stub line, which is connected using a T-element. Another one meter cable extends to the second T-element, branching out the second stub line of 0.3 meter length. Finally, a 10 meter cable leads to a Fujitsu FlexRay™ node, comprising

the same bus drivers as the TEODACS active star (*AS8220/AS8221*, see section 2.3.4.3). The Fujitsu node (number four) is terminated using the same split termination as the TEODACS active star. Both stub lines remain open circuit in this setup. Besides these open circuit stub lines, this network complies to FlexRay^{TM}'s standards in cable length and termination.

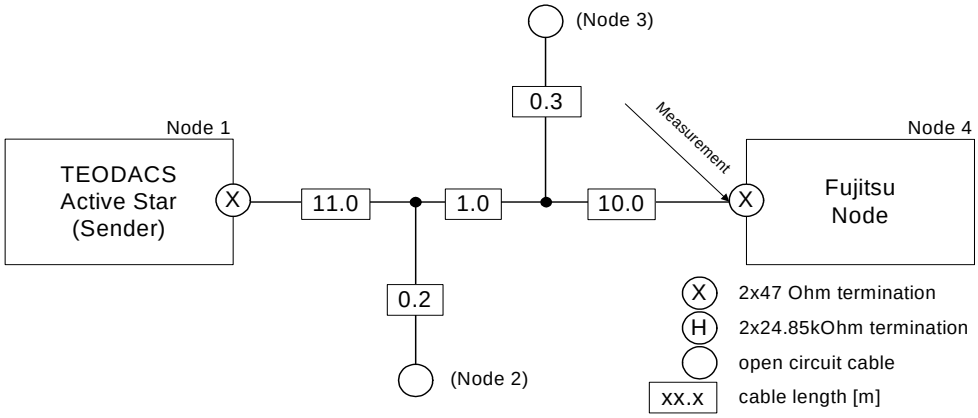

Figure 4.11: Topology 1 schematics.

The signals sent using the TEODACS tester node number one were measured at the Fujitsu node number four. In the simulation environment, sending and receiving nodes are only differing in their drive, however both types include the same bus driver. This way the simulation environment matches the hardware setup.

Since both stub lines are comparatively short, the signal integrity decrease because of reflections should only be minimal. Both cable ends farthest away from each other are terminated, matching the lines characteristic impedance.

Topology 2: The second topology represents a modification of "topology 1". All cable lengths remain the same, including the two stub lines. However, the first stub line, which is of 0.2 meter length, is connected to a second TEODACS active star (node number two). This node is terminated using a $2 \cdot 47\Omega$ split termination. Figure 4.12 shows the network topology.

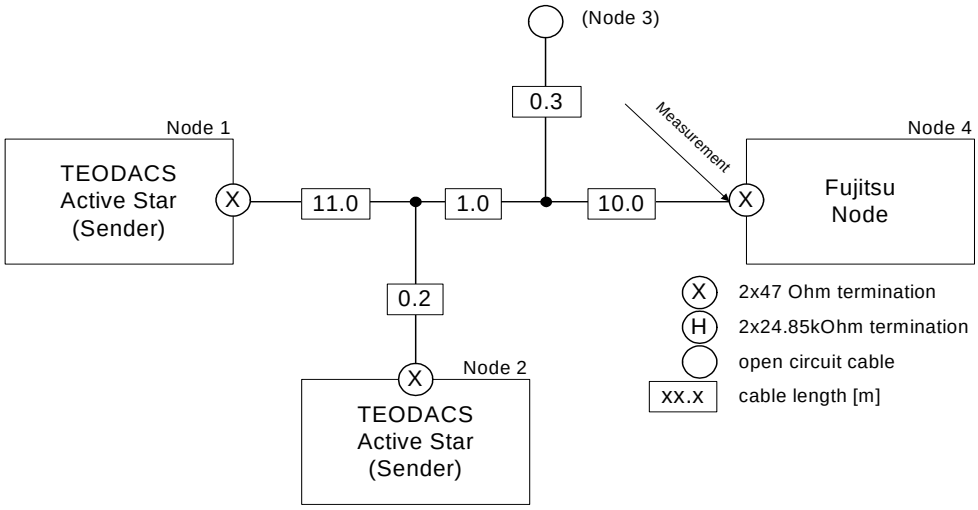

Figure 4.12: Topology 2 schematics.

This network complies to FlexRay™'s standards in cable lengths. It does not comply to FlexRay™'s specification with respect to termination or DC bus load. There are three termination elements present in this topology, lowering the total DC bus load under the acceptable threshold. FlexRay™'s specification [Fle06a, p. 24] only allows a DC bus load between 40Ω and 55Ω. In order to calculate the DC bus load, parasitic resistances are neglected, and the following formula applies for this basic parallel circuit, where R_{t_m} denotes each node's total termination resistance:

$$R_{DCLoad} = \left(\sum_m (R_{T_m})^{-1} \right)^{-1} \tag{4.21}$$

$$= \frac{1}{\frac{1}{94\Omega} + \frac{1}{94\Omega} + \frac{1}{94\Omega}} = 31.33\Omega \tag{4.22}$$

Therefore, we expect to see a decreased voltage level over correctly terminated topology 1. Besides that, no significant decrease in signal integrity should occur. Since the first 0.2 meter stub line is terminated matching the lines characteristic impedance, no reflections should occur from this branch. Again, the leftmost TEODACS active star node number one is used as sender, whereas node number four by Fujitsu acts as a receiver.

Topology 3: From now on, the rightmost part of the network introduced in the previous paragraphs is in focus. The strategy was to modify the network successively, moving more and more towards an invalid topology. By side-by-side comparison the simulation environment was validated against the hardware setup.

This third topology basically represents a standard compliant FlexRay™ network. It is based on topology 2, replacing the $2 \cdot 47\Omega$ split termination with a high resistive $2 \cdot 24.85k\Omega$ split termination. The second 0.3 meter open circuit stub line remains open as in topology 2. Figure 4.13 shows the network topology.

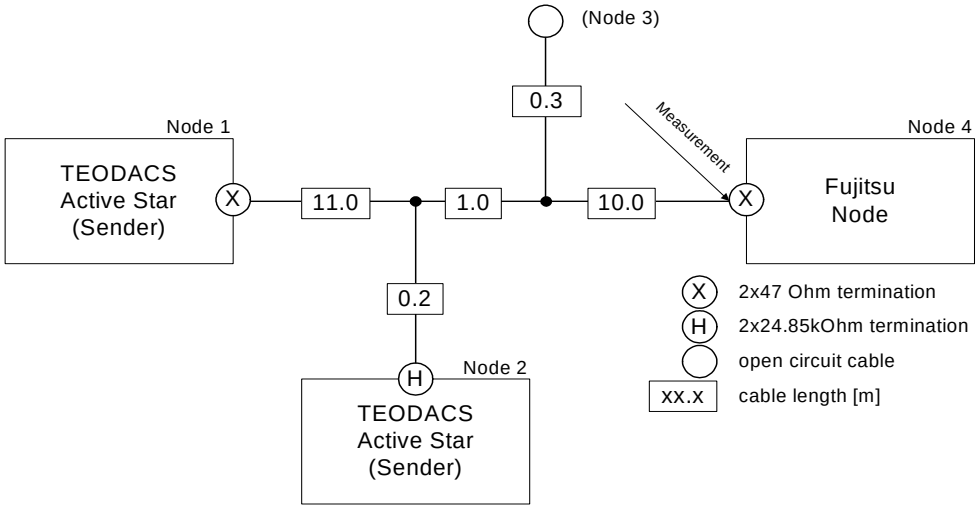

Figure 4.13: Topology 3 schematics.

Because of the high resistive termination at the first 0.2 meter stub line, we expect the network to behave properly, keeping reflections at a minimum and signal integrity at a maximum level. The DC load is within the specified range again:

$$R_{DCLoad} = \frac{1}{\frac{1}{94\Omega} + \frac{1}{94\Omega} + \frac{1}{24850\Omega}} = 46.91\Omega \tag{4.23}$$

Using the second stub line on the right, which poses no correct FlexRay™ network structure of course, the network is modified to disturb signal integrity.

Topology 4: This network topology basically represents a modification of topology 3 and extends the stub line on the right in length but remains open circuit. The leftmost TEODACS active star node number one still acts as a sender, whereas the Fujitsu node number four on the right hand side acts as a receiver. A signal traveling across the network needs to pass a total of 22 meters of cable. The second stub line was increased in length to 3.5 meter. The following considerations are viewed from the corresponding splice or T-element.

Any signal arriving at this point from node one, extends in two directions: First, into direction of node three (open circuit stub line) and second, into direction of node four. Assuming a signal propagation time of approximately 5 nanoseconds per meter, any signal takes 17.5 nanoseconds to travel along the 3.5 meter stub line. At the end of this stub line, the signal will be reflected ($\rho = +1$) due to its open circuit end.

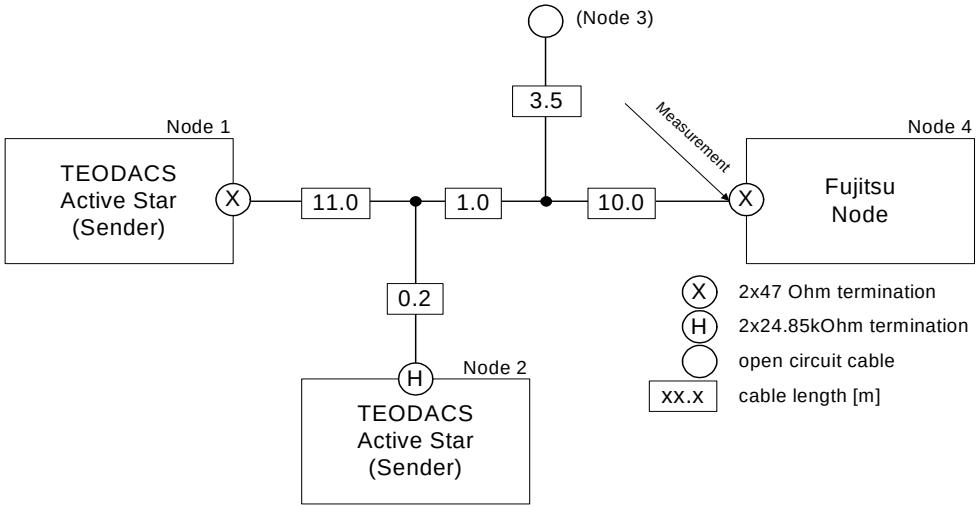

Figure 4.14: Topology 4 schematics.

During that time, the original signal has traveled along 3.5 meter of the 10 meter cable to node four. Now the reflected signal is running backwards along the stub line, whereas the signal traveling along the 10 meter line progresses normally. After another 17.5 nanoseconds, the reflected signal hits the splice again, the original signal has passed 7 of 10 meters of cable at this moment. Concluding from these considerations, a reflected wave is expected to arrive at node four, 7 meters or 35 nanoseconds later than the original wave, disturbing signal integrity significantly more than the previously introduced 0.3 meter stub line, which caused a reflection to arrive at the point of measurement at node four after just 3 nanoseconds. The topology and the discussed behavior is illustrated in figure 4.14.

Topology 5: This final network topology is based on topology 4. Due to the limited choice of cables in the laboratory, the 10 meter cable is exchanged with the 3.5 meter stub cable. All termination measures remain in place, nevertheless, this network does not comply to FlexRay™'s standards: The two nodes farthest away from each other on one bus line are not terminated anymore, but the open circuit stub line became unacceptably long. Figure 4.15 shows the network topology.

Considering the same assumptions as with topology 4, the first signal wave will arrive after 17.5 nanoseconds after passing the second splice at node four. At this moment, the wave propagating along the 10 meter stub line still has 6.5 meter to go until it gets reflected at the open circuit cable end, and has another $10 + 3.5 = 13.5$ meter to go. It will arrive 20 meter or 100 nanoseconds later than the original

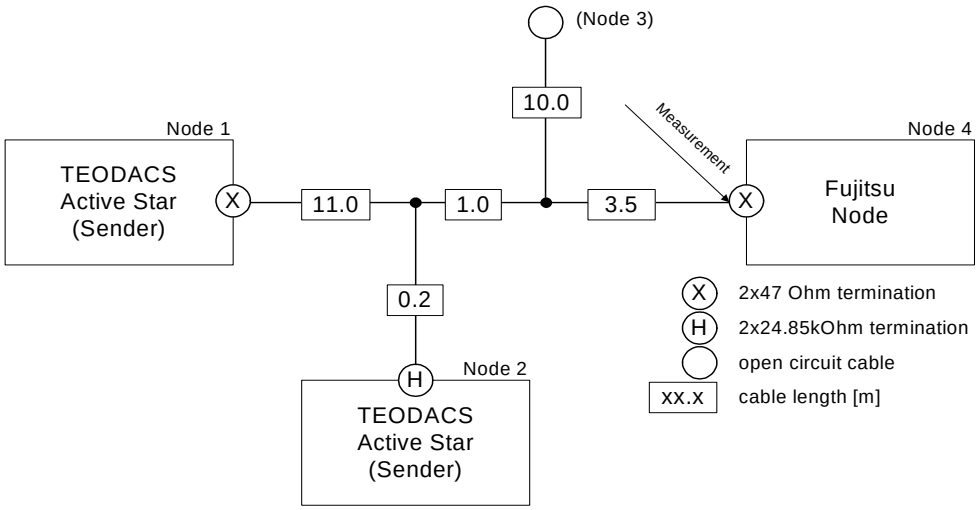

Figure 4.15: Topology 5 schematics.

signal at node four. 100 nanoseconds correspond to one full FlexRay™ bit, so if the reflected wave is superimposed with the original signal, it might have serious effects on signal integrity. Assuming the worst case, it might even destroy the signal and render the network unusable.

Why does it make sense to set up a network topology which is not up to the specification? To answer this question, methods from section 2.2 need to be recalled. We started our verification and validation process with simple topologies, as explained in section 4.5.1. From network topology 1 to 5 we successively worsened the signal quality at node four. For this reason, signal integrity at node four provides a nice indicator for confidence in the simulation environment. Additionally, it helps to answer questions about the reliability of future network models, where a real hardware prototype would be too expensive or even impossible to build due to limited resources.

4.5.3 Topologies with Active Star

The final topology presented in this work includes an active star. This test bench is used for a number of reasons. First of all, it is a proof-of-concept implementation of an active star model, written in VHDL and VHDL-AMS, demonstrating all characteristics of this FlexRay™ network element. Second, the active star model is used to show the advantage of additional hardware efforts regarding signal integrity. Another reason is to temporarily complete the range of available FlexRay™ elements in the simulation environment.

Figure 4.16 shows a network with cable lengths similar to those of topologies 1 to 5. Basically, a network engineer could replace the 1 meter interconnecting bus line cable with an active star, achieving (almost) the same area of coverage. The additional hardware cost and occurring delay is compensated with a refreshment of signal integrity on all branches of the active star. Each of the four point-to-point connection type cables is terminated on both ends, therefore no harsh reflections are expected using this topology.

An important property of an active star's behavior is the delay it causes. This rather simple topology is used to verify the implementation of the active star model. A number of characteristics are subject to investigation, including:

- delays of positive and negative edges
- asymmetric delays
- idle-to-active signal truncation

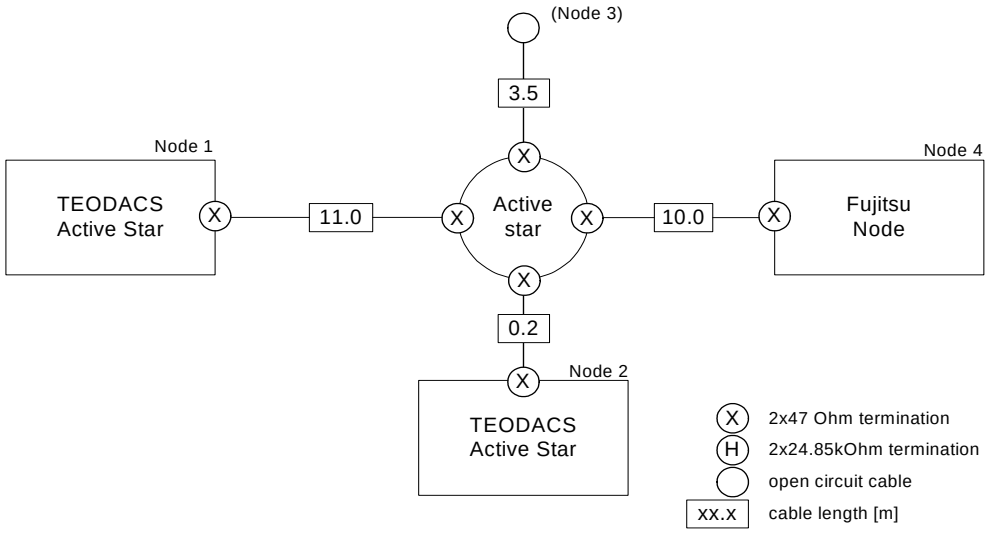

Figure 4.16: Active star topology schematics.

There are a number of additional parameters, however, almost all of them can be configured in a direct or indirect way, primarily using the bus driver's parameter set. According to the terminology of [Rau08], this active star model corresponds to a discrete one, opposing a monolithic active star integrated in one single circuit or model respectively. The timing constraints for monolithic active stars are slightly more relaxed compared to those of discrete ones.

4.5.4 Other topologies

During research activities within the TEODACS environment, a number of more complex topologies have been simulated as well: For example, 11 bus drivers, 17 cables and one active star were not a problem at all, besides the increased simulation time (see section 5.2.1). However, these topologies are just compound of basic topologies or clusters covered in the previous sections. Therefore, these network architectures do not introduce any new concepts and their behavior can be analyzed by looking at single clusters as well.

Chapter 5

Results

This final chapter provides all kind of results emerged from this thesis. Predominantly, these are various electrical physical layer simulation results. Section 5.1 contains such results. The bus driver model is validated in section 5.1.1, together with cable models and some other components. Once all these models are validated, they were used to build whole topologies. The outcome of these experiments is documented in section 5.1.2 for bus line topologies, and in section 5.1.3 for active star topologies. After the introduction of these results, they are discussed in section 5.2: Simulation speed and observability are commented in sections 5.2.1 and 5.2.2, respectively. Network engineers and designers might be interested in section 5.2.3, where the credibility of the simulation framework is explained. In connection with that, the behavior of the simulation's signal integrity is investigated. Finally, some non-modeled properties are discussed in section 5.2.4.

5.1 Simulation & Validation Results

5.1.1 Bus Driver & Simple Topologies

The parameters determined for bus driver simulation are listed in table 5.1. `v_RxD_L` and `v_RxD_H` are used to set digital voltage output levels, which is important whenever a received data stream is passed on across one or more bus drivers, e.g. a discrete active star. Both values were determined by measurement and set accordingly. `v_CC_plus` and `v_CC_minus` are factors, which act as multipliers for `Vcc` to achieve `DATA_0` and `DATA_1` bus voltages. According to [CIS09], `t_uv_recover` controls the "undervoltage recovery detection time", and `t_sleep` specifies the "sleep mode timeout". As described earlier in section 4.1.3, `stability_time` is used to adjust the receiver's sensibility to input changes. Most important, `r_open` and `r_closed` are used for internal voltage bridge dimensioning.

Parameter	Unit
`v_RxD_L`	V
`v_RxD_H`	V
`v_CC_plus`	1
`v_CC_minus`	1
`t_uv_recover`	s
`t_sleep`	s
`stability_time`	s
`r_open`	Ω
`r_closed`	Ω

Table 5.1: Bus driver model parameters.

By adjusting these parameters, the simulation setups introduced in section 4.5.1 led to table 5.2,

showing the resulting output voltages of the bus driver's BP and BM ports, including the differential bus voltage.

$R\_l[\Omega]$	$U_{low}[V]$	$U_{high}[V]$	$\Delta U[V]$
10	2.26	2.49	0.23
47	1.94	2.82	0.88
100	1.66	3.10	1.44
150	1.49	3.27	1.77
220	1.34	3.42	2.08
470	1.05	3.71	2.66
1000	0.86	3.90	3.03

Table 5.2: Bus driver simulation output voltage.

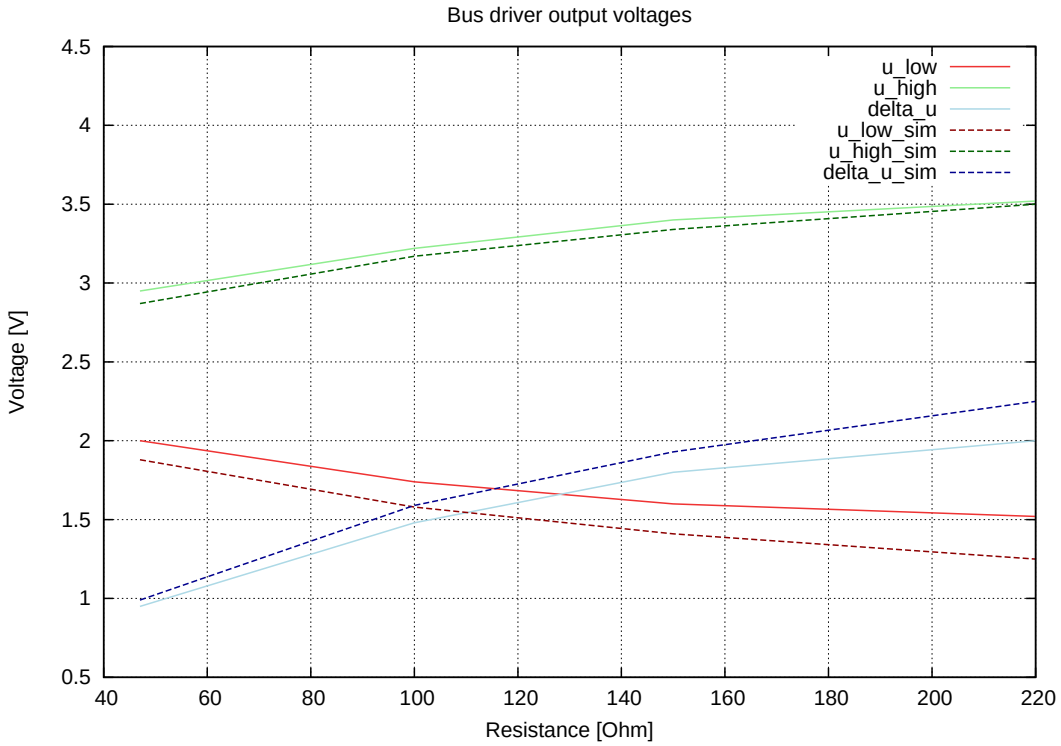

Figure 5.1: Bus driver output voltages.

Comparing these values with table 4.1 from section 4.1.4 leads to figure 5.1. Note the logarithmic scale of the abscissa. The average difference between simulation and measurement of ΔU is just 0.16 percent. The DC load of a properly terminated FlexRay™ network makes up to 47Ω, the deviation at this point is 0.07 volts or 7.47 percent. Additionally, the same setup was used in connection with a $40\Omega||100pF$ load, as suggested by [Fle06a, p. 57]. The absolute value of the differential bus voltage should range between 600 and 2000 millivolts, the simulated absolute value was exactly 776.7 millivolts. All other transmitter characteristics can be configured directly via model parameters and are not revised at this point. The bus driver is considered to be validated sufficiently at this point for further simulation experiments.

At this point the simulation result of one of the previously introduced simple topologies (4.5.1) is given, where a single bus driver is connected to an 11 meter open circuit FlexRay™ cable. The simple test file of section 4.3.1 is used as stimulus for the given setup. Figure 5.2 shows a direct comparison

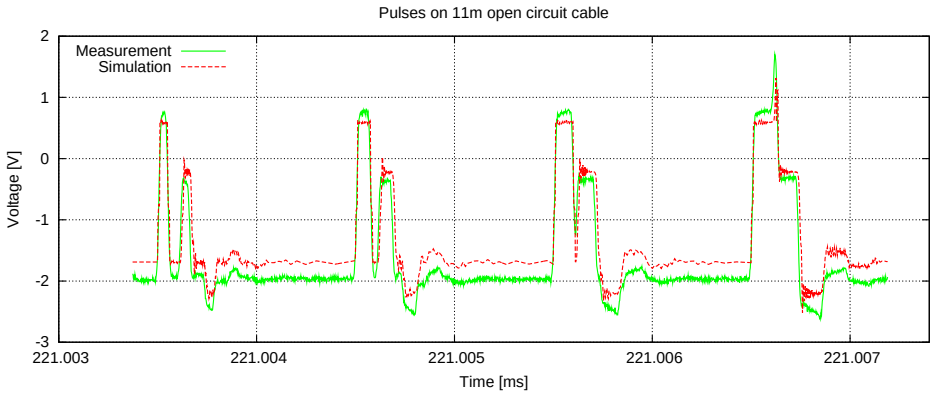

Figure 5.2: Simple test file on an 11m open circuit cable.

between simulation and hardware measurement. The first 50 nanoseconds pulse is traveling from the bus driver to the open circuit cable end, becomes reflected due to the implicit high-impedance termination, and travels back to the bus driver. The last and rightmost 125 nanoseconds pulse has a sufficient length to superimpose the pulses traveling back and forth. This case needs to be avoided at any time.

The resulting waveforms have a correlation coefficient (see next section) of 0.9728. The rising and falling edges are matching very good, however the stable voltage levels are visibly differing. We suspected the hardware bus driver to implement an unknown mechanism causing this effect.

5.1.2 Bus Line Topologies

As introduced in section 4.5.2, five bus line network topologies were simulated and measured in parallel, to observe the changes in signal integrity under various circumstances.

5.1.2.1 Topology 1

Figure 5.3 shows two overlaid signals, one created through simulation, the other one measured using an oscilloscope. It shows that both lines match very well in general. Figure 5.4 shows a crop of approximately $2.5\mu s$ of the frame shown in figure 5.3. Validation through visual inspection shows good correlation between both signals.

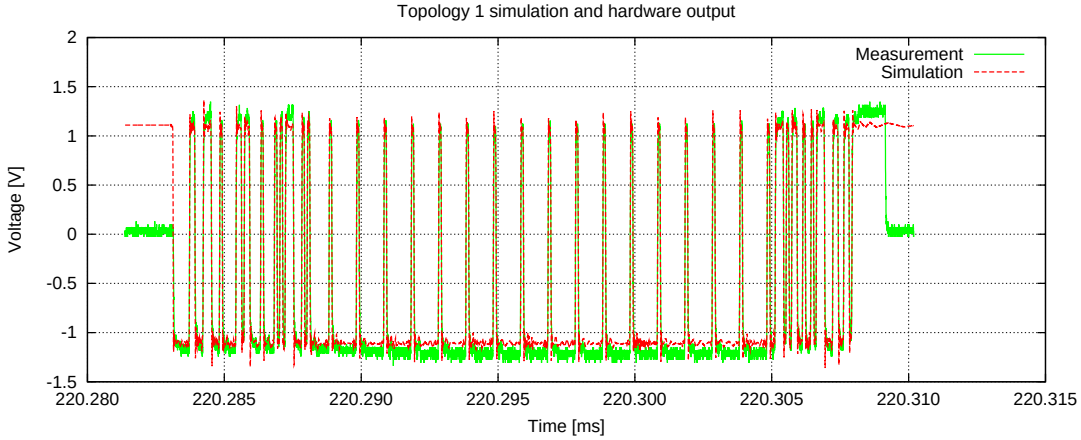

Figure 5.3: Topology 1: Complete FlexRay^{TM} frame

It can be seen, that the simulated signal captured at node number four tends to overshoot slightly, compared to the measured signal. The rising edges of both signals differ slightly ($10ns$ versus $22ns$

at 20% to 80%), with the simulated signal having the steeper rising edge. During the measurement campaign, the hardware setup was expected to deliver worse signal integrity than simulation. This was justified with environmental influences caused by the laboratory's test setup and measurement devices, which are not present in the simulation environment.

Both open stub lines are too short to have an impact on signals measured at nodes number one and four. As explained in [Kup07] and [JG03], these stub lines are considered to be electrically short, because their length is well below one quarter of the signals wavelength:

$$c = 3 * 10^8 m/s \tag{5.1}$$
$$f = 10 * 10^6 Hz \tag{5.2}$$
$$\lambda = c/f = \frac{3 * 10^8}{10 * 10^6} = 20m \tag{5.3}$$
$$0.3 < \frac{1}{4} * \lambda \tag{5.4}$$

The generated 10-bit eye diagram (figure 5.5) clearly shows that signal integrity is very good and all bits within the frame are far above or below their critical threshold values. The average asymmetric delay for `DATA_1` was $-0.86ns$, and $-0.51ns$ for `DATA_0`. This bus line topology has no critical cable lengths and is terminated correctly, therefore it is no surprise to see such satisfying results.

In order to specify a metric for the similarity of two different waveforms, the pearson product-moment correlation coefficient, also known as sample correlation coefficient [Bro87], is calculated as

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \sum_{i=1}^{n}(y_i - \overline{y})^2}} \tag{5.5}$$

Values of r_{xy} close to 1.0 indicate a close correlation between the samples of x and y. On the other hand, values close to 0.0 indicate a low or nonexistent correlation between the samples of x and y. Practically, values smaller than 0.8 already indicate a low correlation. The calculated correlation coefficient of the complete FlexRay™ frame shown in figure 5.3 is 0.9444. The chosen bit sequence of figure 5.4 correlates even better with a value of 0.9909.

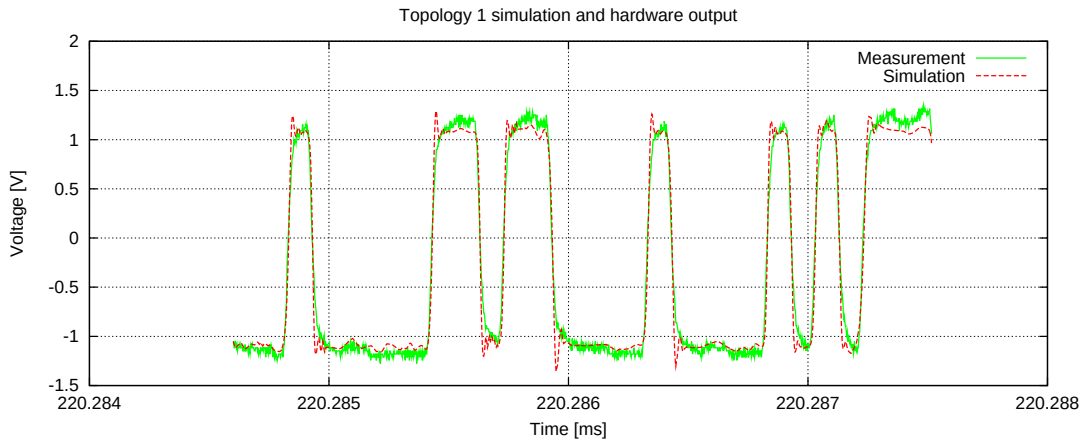

Figure 5.4: Topology 1: FlexRay™ frame crop

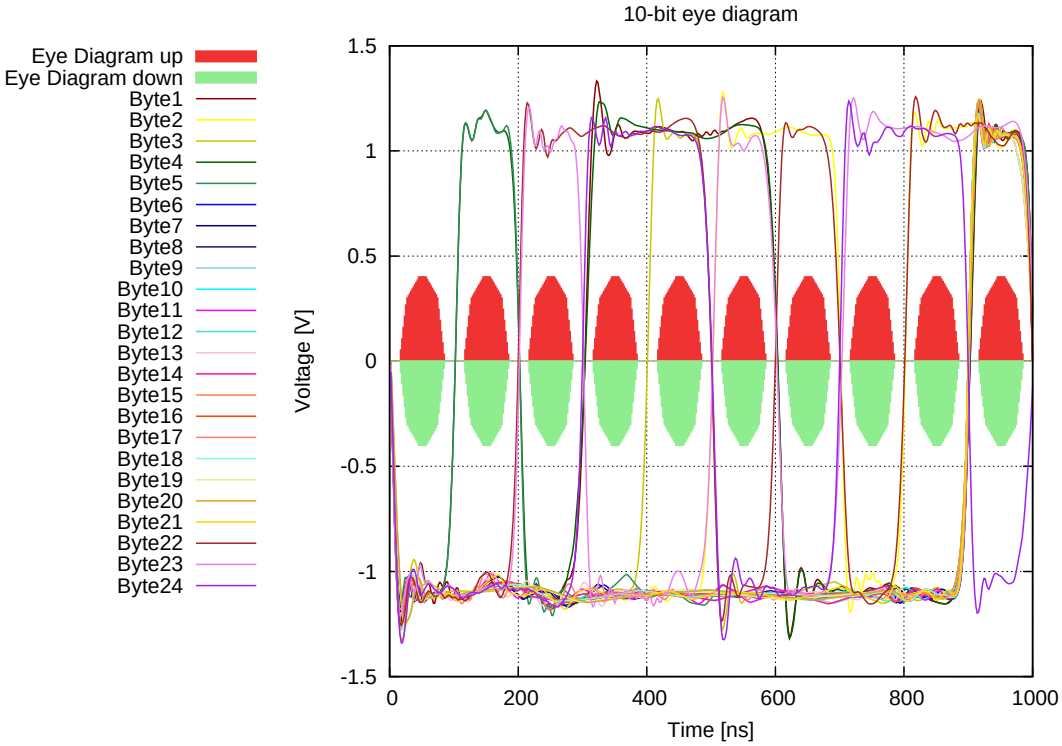

Figure 5.5: Topology 1: Eye diagram

5.1.2.2 Topology 2

This network topology is remarkable because of its redundant termination at node number 2. The comparison of a full frame is shown in figure 5.6, the signals were transmitted from node number 1 to node number 4. The additional split-termination causes the overall DC bus load to decrease. Thereby, bus voltage drops to a visible lower level, as a detailed crop shows in figure 5.7. It is also possible to interpret this behavior as an unexpected low-impedance connection between both bus lines `BP` and `BM`. In this section, the resulting digital signal at the bus driver's `RxD` port is also included. The direct comparison between simulation and measurement (figure 5.8) shows that the voltage drop does not affect the digital signal output yet. The correlation coefficient of the complete FlexRay™ frame shown is 0.92, and of the frame detail even 0.9887. The captured digital signal correlates with a value of 0.994 to the simulated signal.

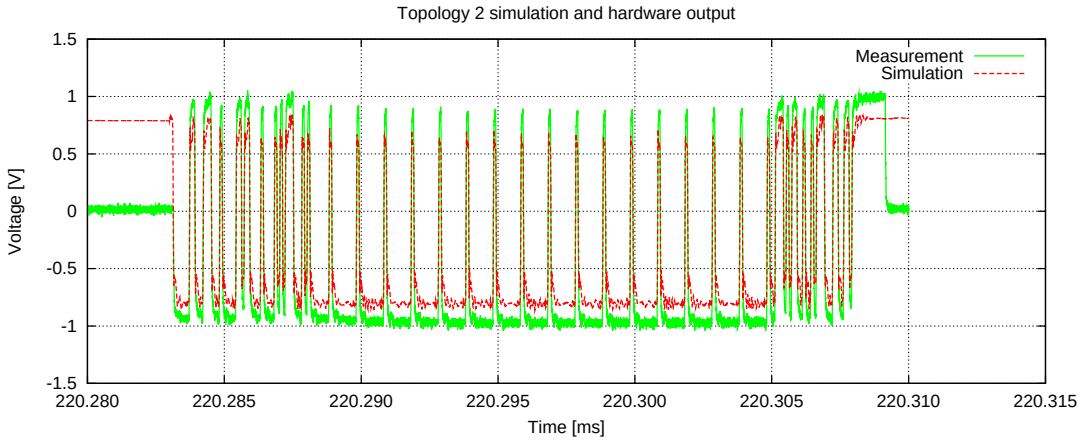

Figure 5.6: Topology 2: Complete FlexRay™ frame

However, the difference between the stabilized `DATA_1` level (around 600 millivolts) and the corresponding FlexRay™ specific threshold (225 millivolts) is rather small. The time between those two critical points is just 6 nanoseconds, so any additional short circuit or unexpected capacitance might have a serious impact on signal integrity of this topology. A look on the 10-bit eye diagram in figure 5.9 underlines this observations. All bits are still transferred and recovered correctly, however, the gap to FlexRay™'s minimum requirements became smaller.

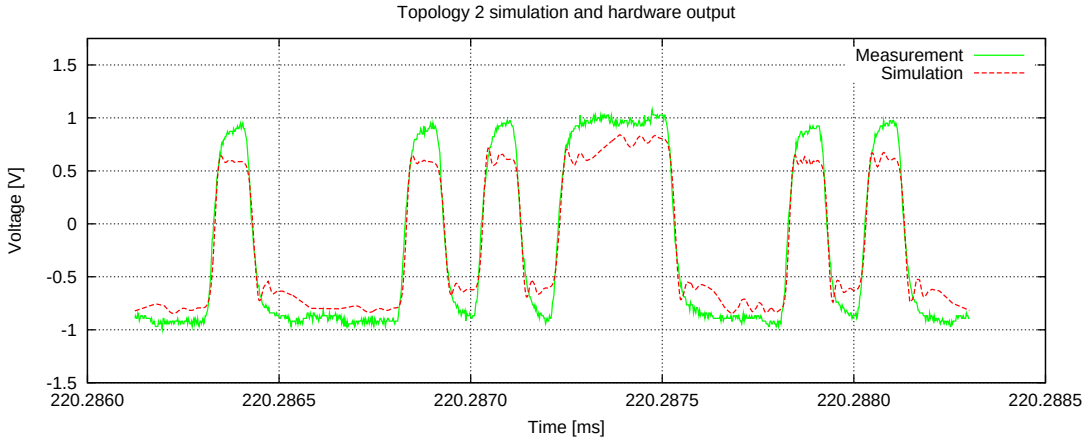

Figure 5.7: Topology 2: FlexRay™ frame crop

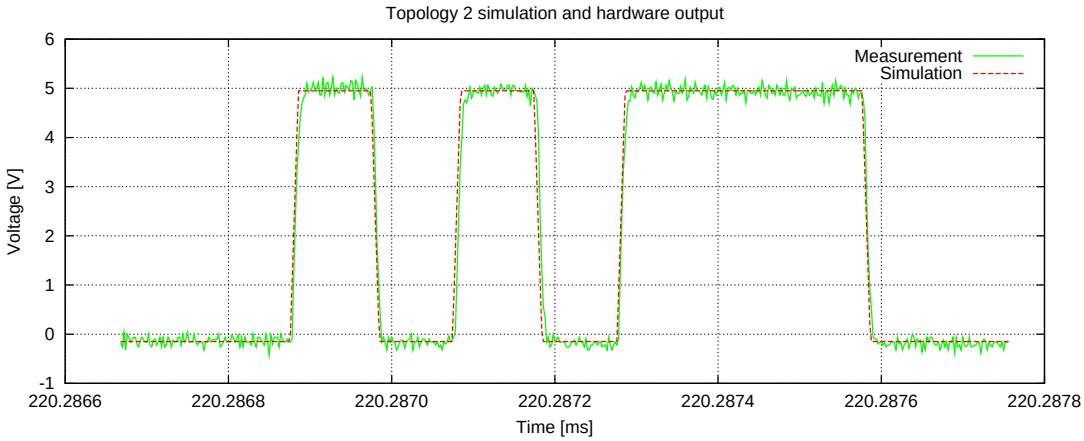

Figure 5.8: Topology 2: Digital signal comparison

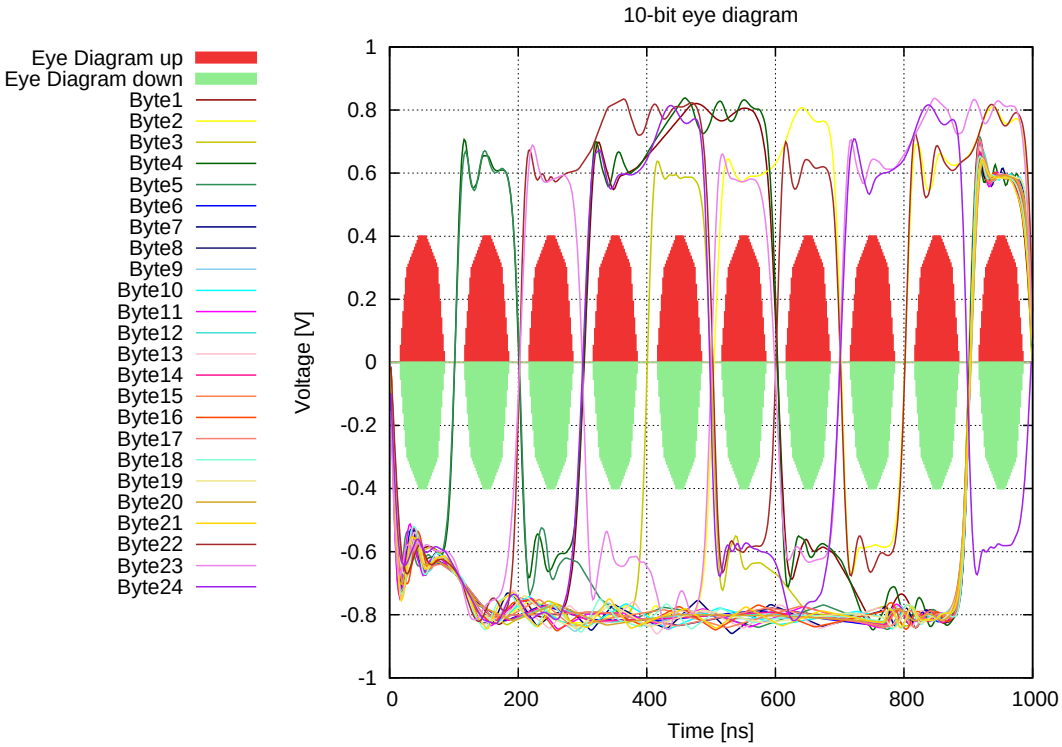

Figure 5.9: Topology 2: Eye diagram

5.1.2.3 Topology 3

A complete FlexRay™ frame captured at node 4 of topology 3 is shown in figure 5.10. A more detailed crop is shown in figure 5.11. Overshooting of the simulated signal decreased. Still, the simulated signal has slightly steeper edges, carrying higher frequencies than the measured signal.

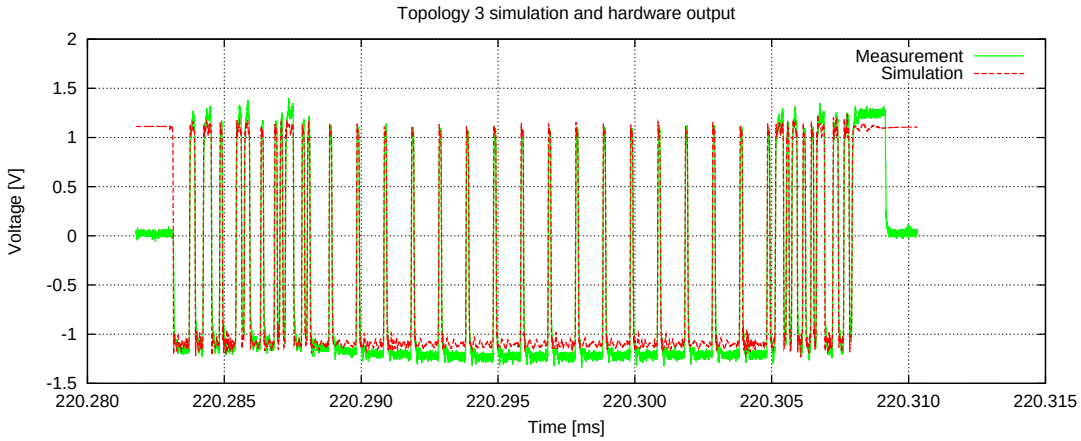

Figure 5.10: Topology 3: Complete FlexRay™ frame

Signal integrity is very good, as the 10-bit eye diagram shows in figure 5.12. The 0.3 meter open circuit stub line is too short to have any visible impact. It is worth to note that these results are very similar to those of topology 1, proving that an open circuit stub line has the same impact on signal integrity than a high-impedance termination or receiving bus driver. The correlation coefficient of the complete FlexRay™ frame is 0.9465, and of the frame detail even 0.9922.

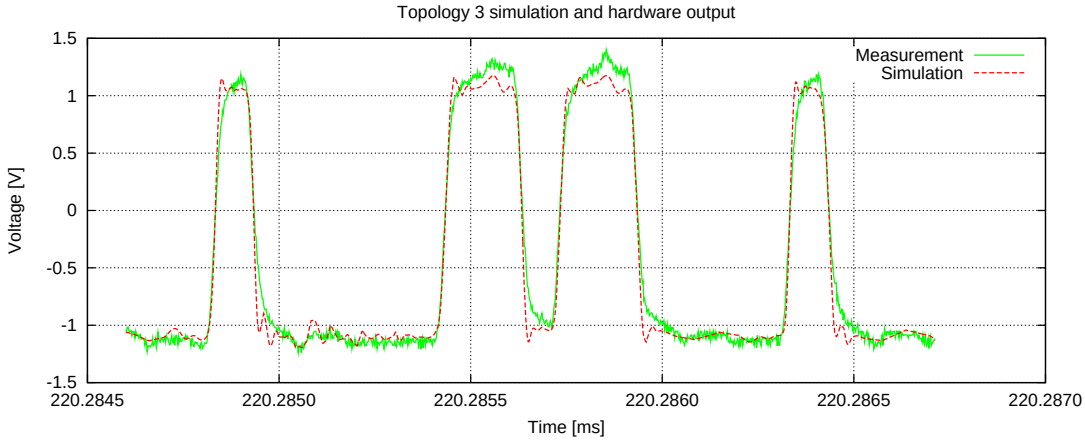

Figure 5.11: Topology 3: FlexRay™ frame crop

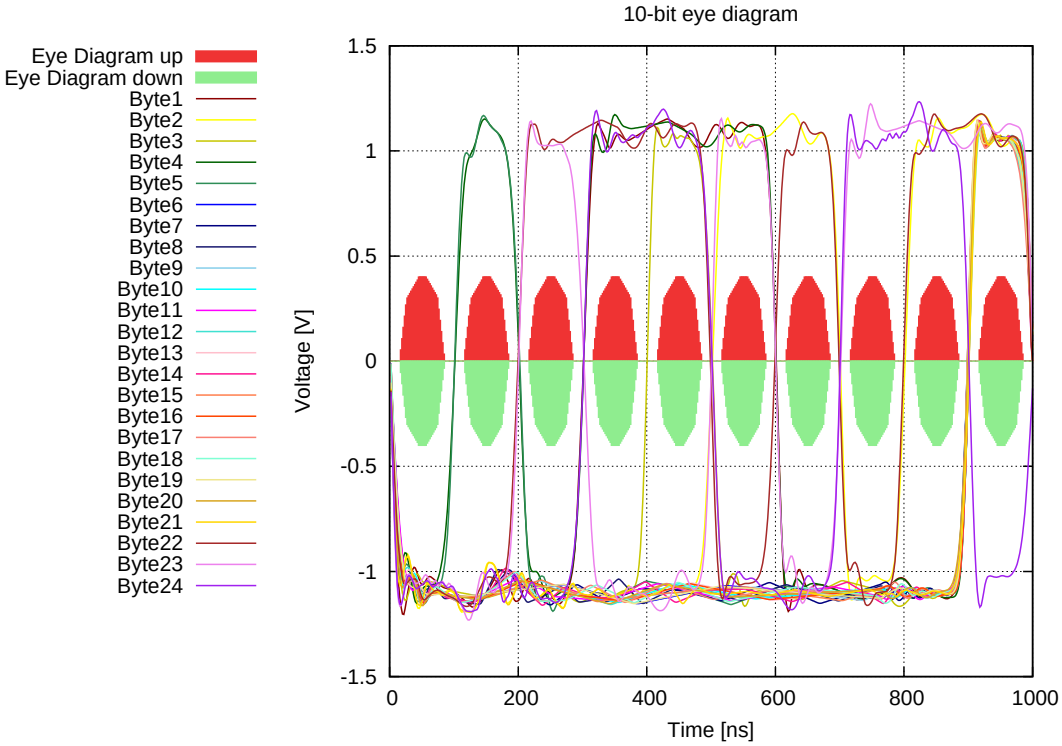

Figure 5.12: Topology 3: Eye diagram

5.1.2.4 Topology 4

The simulation and measurement results of topology 4 show the arising problems with long stub lines. A complete frame transmitted from node 1 to node 4 is shown in figure 5.13. Due to the positive signal reflection ($\rho = 1$) at the end of the 3.5 meter open circuit stub line, the reflected signal reaches node number 4 with a small delay. At a signal propagation time of 5.5 nanoseconds per meter, every reflected signal edge reaches node four $5.5 \cdot 7 = 38.5$ nanoseconds later than the original signal, causing some ripple as figure 5.14 shows. In this topology, the reflected rising edge interferes with the original rising edge in the area of approximately +0.3 volts. The FlexRay^{TM} specification defines the threshold for rising edge analog-to-digital conversion typically at +0.225 volts. Therefore, the bus driver's switchpoint might be delayed or, much worse, the digital signal might contain unwanted peaks. In this example, digital signal recovery is still working at an acceptable level. The digital signal only contains a minor asymmetric delay of 10 nanoseconds. Figure 5.15 shows a direct comparison between the measured and simulated digital signal at the receivers RxD port. The resulting 10-bit eye diagram is shown in figure 5.16. The correlation coefficient of the complete FlexRay^{TM} frame is 0.9712, of the frame detail even 0.988. The simulated and measured digital signals match with a value of 0.9765.

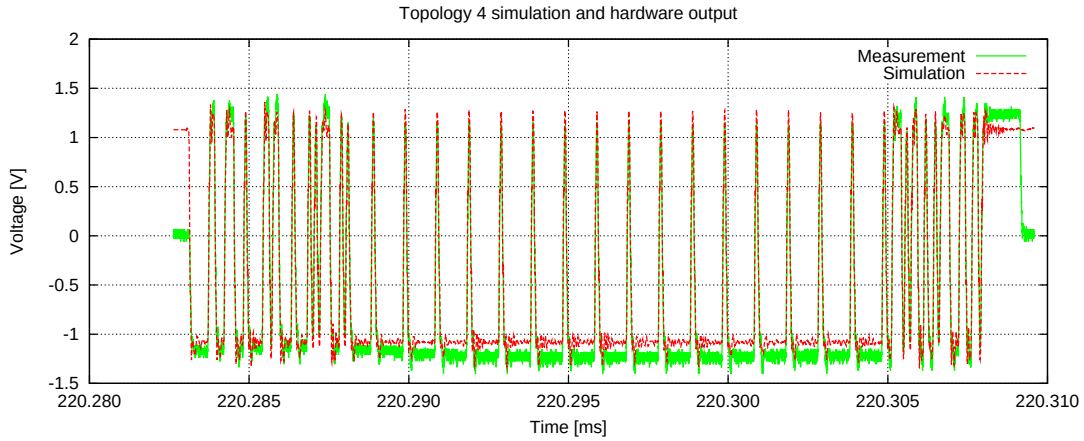

Figure 5.13: Topology 4: Complete FlexRay^{TM} frame

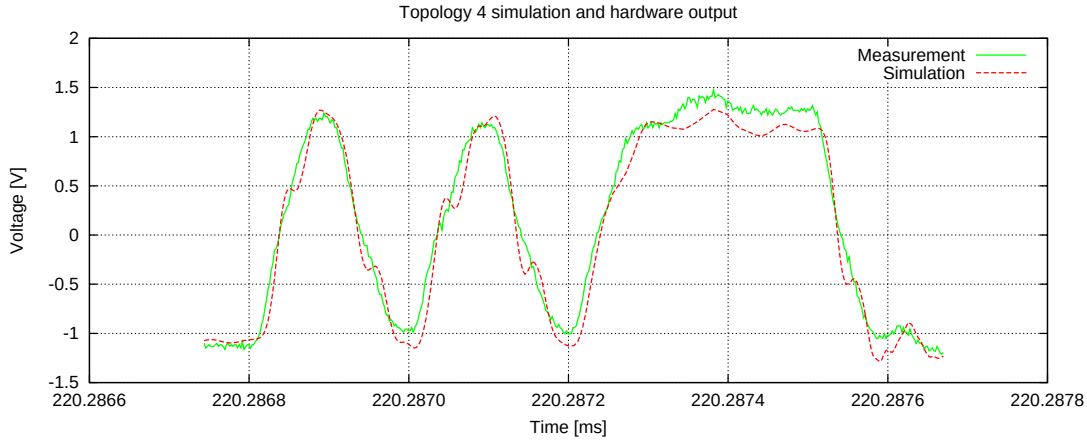

Figure 5.14: Topology 4: FlexRay^{TM} frame crop

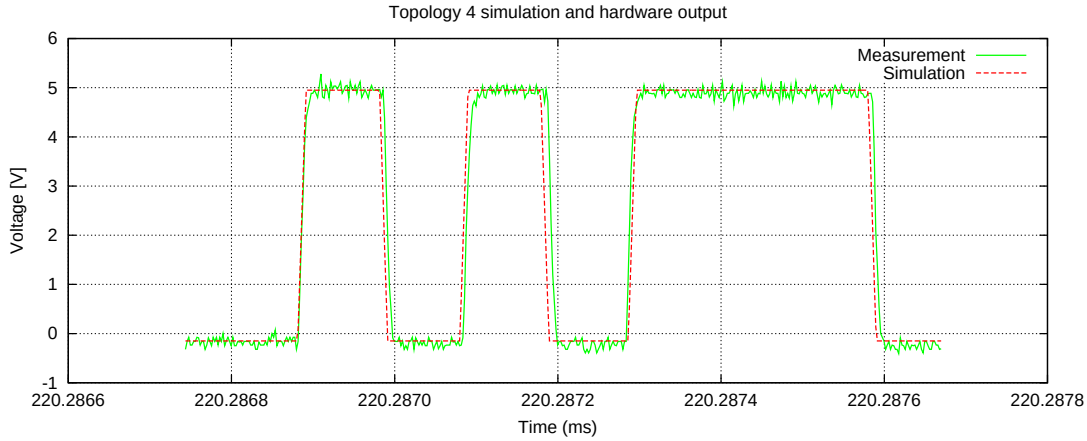

Figure 5.15: Topology 4: Digital signal comparison

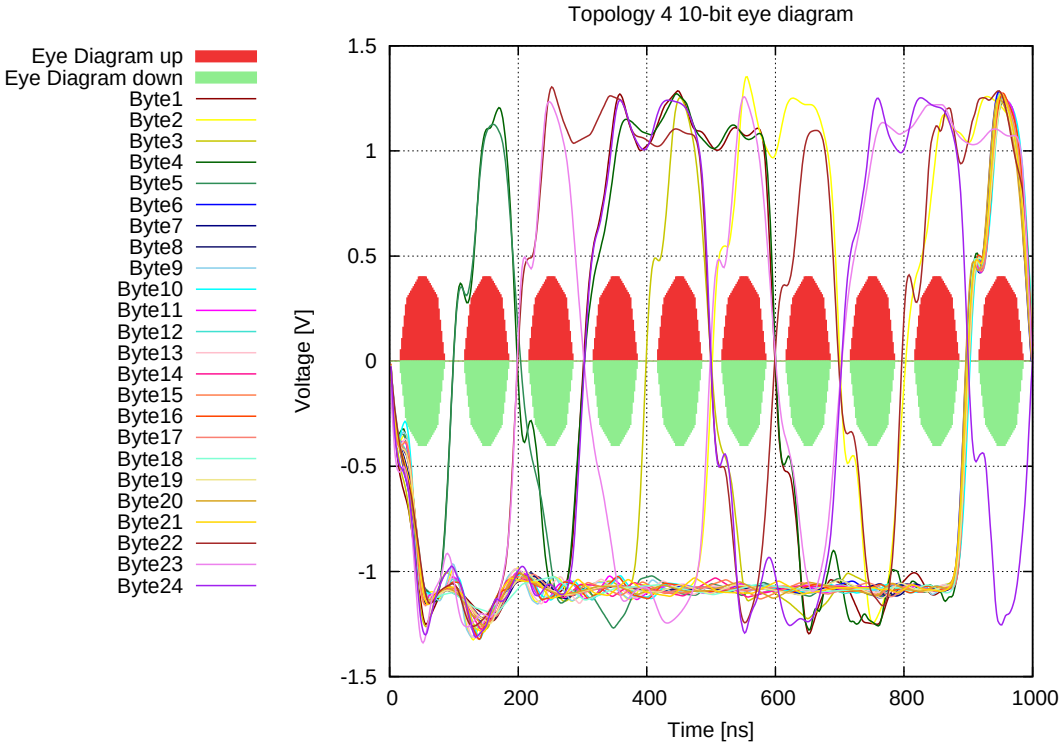

Figure 5.16: Topology 4: Eye diagram

5.1.2.5 Topology 5

Topology 5 represents an invalid network architecture, either by termination or maximum cable length. Similar to topology 4, the signal reflected by the open circuit stub line (node number 3) hits node number 4 with a delay. Again, the signal propagation time is considered with 5.5 nanoseconds per meter, and the additional length caused by the stub line is $2 \cdot 10$ meter. Therefore, the reflected signal is approximately 110 nanoseconds late, which corresponds to more than a single bit duration at 10 Megabits per second of bandwidth.

A complete frame transmitted from node number 1 and captured at node number 4 is shown in figure 5.17. The differential bus signal at node number 4 experiences heavy distortion, as figure 5.18 shows. This has immediate effects on signal integrity. Figure 5.19 shows a direct comparison between the measured signal from the corresponding hardware setup and the simulated signal at node number 4. Additionally, the network stimulus sent at node number 1 is included for reference. Of course, it was shifted in time to eliminate the propagation delay and make an easy comparison by visual inspection possible.

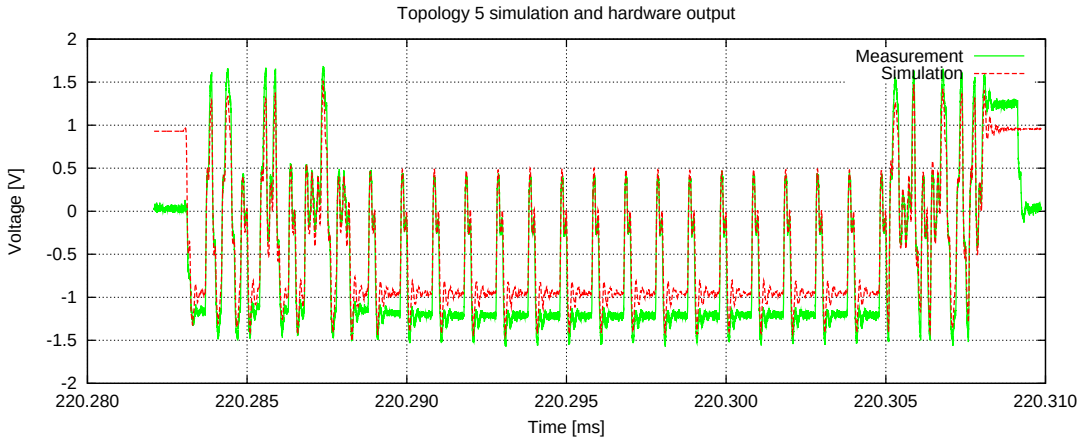

Figure 5.17: Topology 5: Complete FlexRay^{TM} frame

One can see that the measured signal is definitely worse than the simulated signal, leaving out some entire bits. The simulated signal still switches according to the stimulus, where the measured signal remains stable. Nevertheless, the digital thresholds are not correctly triggered anymore and the resulting bits experience asymmetric delays. The worst asymmetric delay within the simulated frame was 44.0 nanoseconds. The generated 10-bit eye diagram in figure 5.20 visualizes these problems as well. As explained earlier, these results match expectations, because the simulated signal is assumed to behave "better" than the measured one.

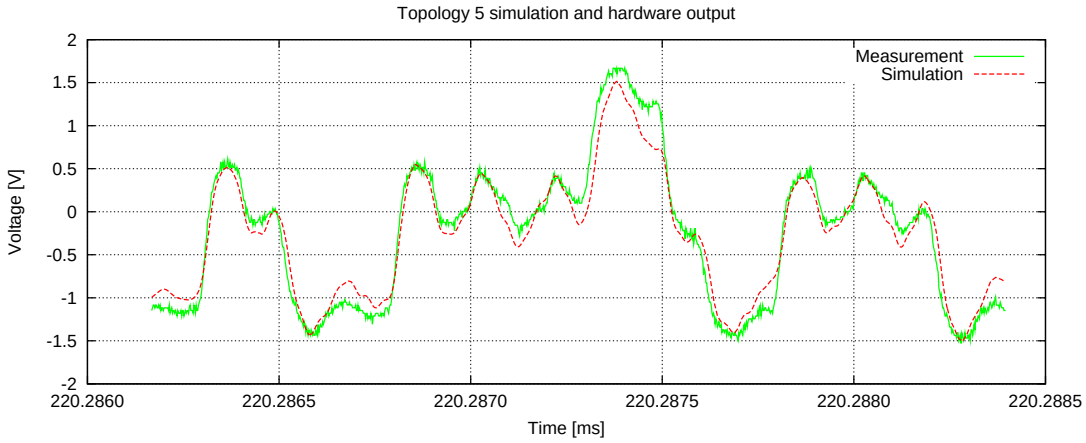

Figure 5.18: Topology 5: FlexRay^{TM} frame crop

As in the previous sections, the correlation coefficients were calculated for this topology as well. The complete FlexRay^{TM} frame simulation matches the measurement at a value of 0.9506, the frame detail reaches even 0.9847. The comparison between the resulting digital signals just reaches 0.5877, for the reasons discussed in the previous paragraph.

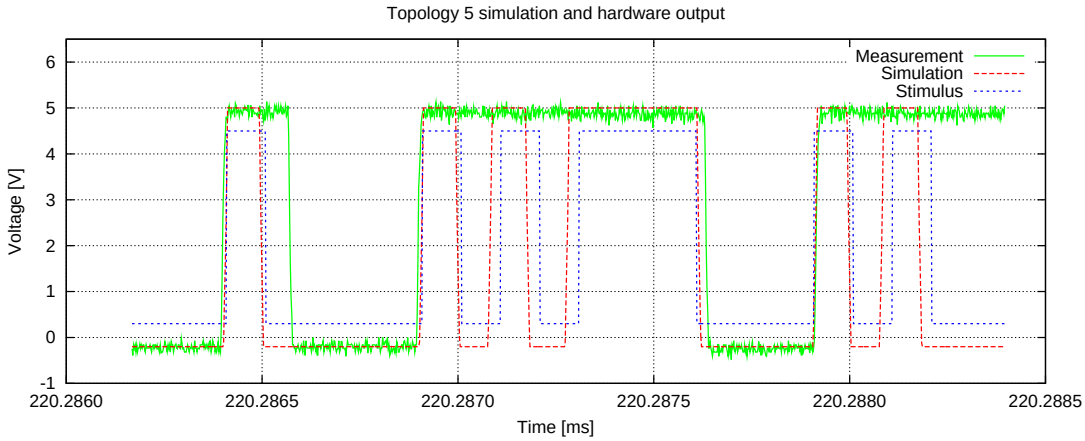

Figure 5.19: Topology 5: Digital signal comparison

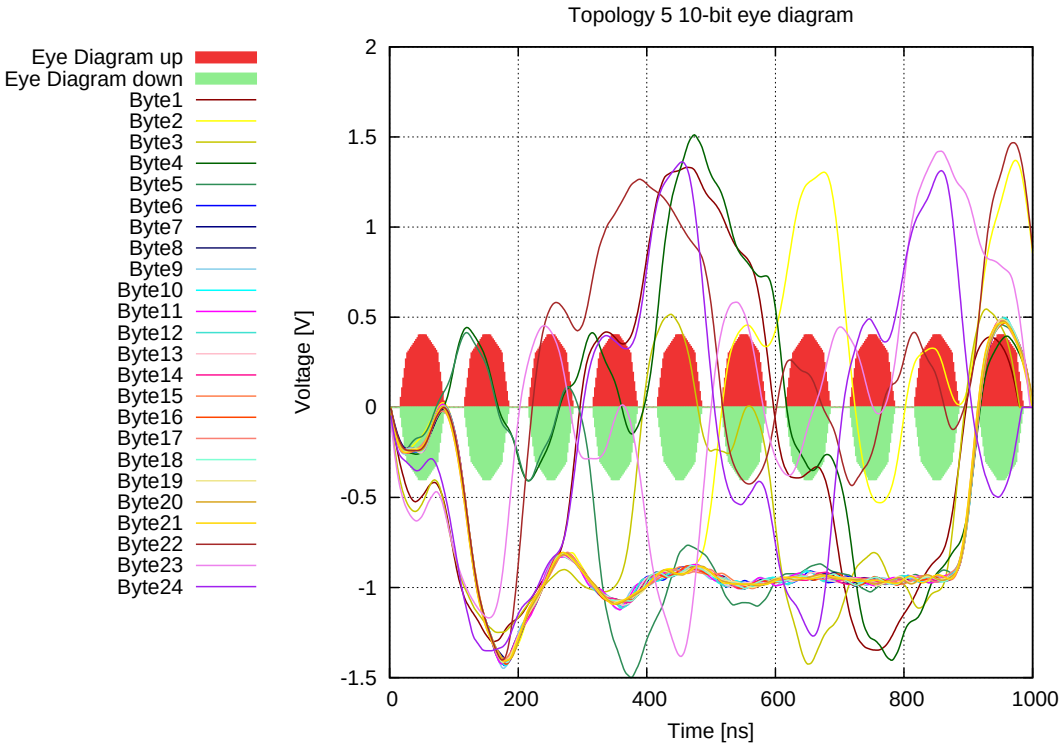

Figure 5.20: Topology 5: Eye diagram

5.1.3 Active Star Topology

In this section the simulation results of the simple active star topology from section 4.5.3 are reviewed. First of all, the proposed active star works as expected. Its internal state changes, depending on the operation phase. All signals received at branch one are passed on to the other three branches. A slight TSS truncation can be observed, its concrete value depends on the bus driver's configuration (analog-to-digital and digital-to-analog delays, threshold voltage configuration, etc.). In connection with that, the `stability_time` bus driver parameter was introduced in order to model the integrated circuit's response time and signal transition behavior.

During experiments with the simple test file introduced in section 4.3.1, it has shown that the receiving bus driver skipped the first low-amplitude $50ns$ pulses. The same behavior was observed in the

laboratory as well. The incoming pulse is simply too short to charge all (parasitic) capacitances and toggle the digital output. For this reason, an active star works as a low-pass or peak filter, and therefore does not distribute such faulty signals. This behavior is illustrated in figure 5.21.

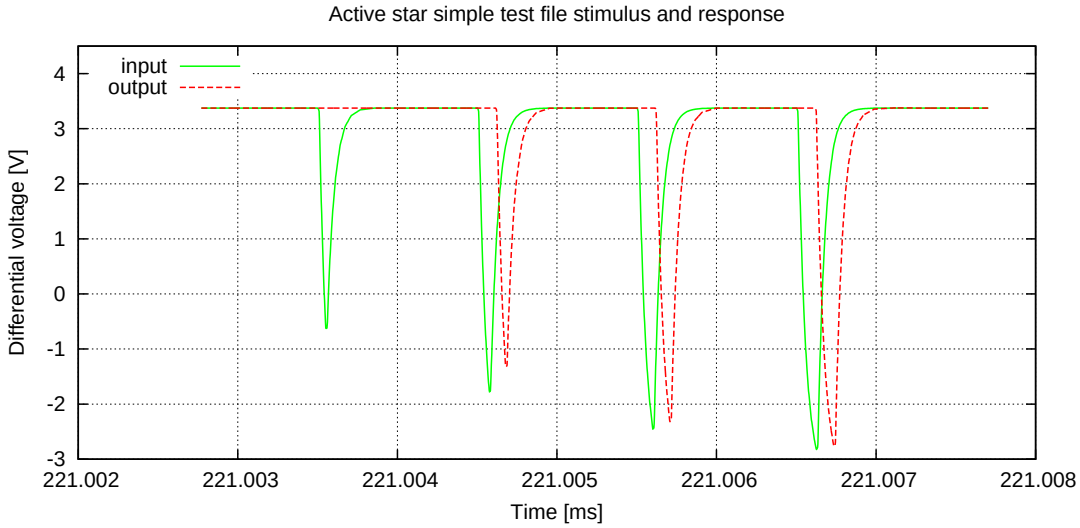

Figure 5.21: Active star simple test file stimulus and response

Similar to disturbing peaks, a network may suffer from erroneous termination, cable lengths, or other unforeseen interferences, as described in previous setups (see section 4.5.2). Thus, it is a good reason to include an active star in a given FlexRay^{TM} communication network, to separate different branches into independent clusters and increase the system's overall reliability.

5.2 Discussion of Results

This section discusses and analyses the achieved results from the previous sections. At this point, meta-information is our matter of interest. Simulation speed, observability, simulation credibility and non-modeled properties are discussed to complete the entire picture.

5.2.1 Simulation Speed

For industrial applications, simulation speed is an important issue. If a measurement task can solve a problem, simulation might represent an unnecessary overhead. However, this requires that specific hardware is available, which is not always the case in research and development departments of the automotive industry, in most cases due to high cost. Reasons for the effectiveness of simulation have been outlined in section 1.2.

The main question is, how fast can a specific problem be analyzed, to get the wanted solution or required information in-time? To answer that question, detailed identification of the problem is necessary. This helps to choose an appropriate layer for simulation tasks. Usually, analog level simulations are slow but provide great level of detail. Considering the same simulated time and input data, sample level simulations are faster, due to the decreased range of values. Bit level, frame level and signal level simulations are even faster, but their level of abstraction increased as well. A related work emerged of the TEODACS project, exploiting this situation and optimizing overall simulation performance is presented in [KAS09]. The models and simulations presented in this thesis were captured mostly on the analog level (waveforms, signal integrity), parts of it on bit level (active star logic). Interfaces to higher levels do exist by design of the TEODACS project 2.3.4.1.

Bus drivers	Active stars	Cable models	Total cable length [m] 1cm/section	Topology notes	Time
1	0	0	0.0	variable ohmic load only	1min 1sec
1	0	1	11.0 (behavioral)	100Ω terminated	1min 14sec
1	0	1	11.0 (structural)	100Ω terminated	1h 35min
4	0	5	25.8 (structural)	bus line topology	7h 10min
4	1	0	0.0	active star testbed	55min
4	1	4	24.8 (structural)	active star topology	1d 5h 31min
11	1	17	44.0	bus lines, passive star, point-to-point	2d 14h

Table 5.3: Simulation speeds of various models and test benches.

Table 5.3 shows typical simulation speeds experienced during various experiments of this thesis. In all cases, the simulated time was 300 milliseconds, where 221 milliseconds were needed for bus driver startup. After that time, the network's data transmission (and desired stimulation) started. The stimulus file contained four FlexRay™ compliant frames, each lasting $25\mu s$. Including idle times, the overall signal lasts approximately $145\mu s$. In case of larger network topologies, the network startup time is not negligible at all and simulation may take longer as expected. In theory, this initialization time could be optimized to reach the beginning of data transmission earlier. Nevertheless, larger intervals allow easier debugging and better observability of bus driver internals. Of course, if only one single frame is enough for replay-based experiments, it is possible to stop the running simulation task much earlier. During these experiments, all bus drivers were equipped with a $2x94\Omega$ split termination and a common mode choke.

All in all, the cable model was found to be the critical factor in this thesis' models and test benches. Due to different cable modeling (see section 4.4.4) differences in simulation time do exist. Although the behavioral model simulates faster in most cases, especially for long lines, the structural model was still preferred. One advantage thereof is the possibility to trace a signal at any point on the line. This segmented approach would also be convenient in fault-injection scenarios.

Concerning the available computational power, the SyAD® server ran on top of a $2 \cdot 2.4$ GHz "Core 2 Duo E6600" machine with 2 GB of RAM, the corresponding client was installed on a Pentium 4 processor with 2.4 GHz clock frequency and 1 GB of RAM. For simulation tasks however, only the server's performance is crucial. A fast client platform may be helpful during the model and test bench composition phase.

5.2.2 Observability

Although using an intellectual property module, its internal state variables and quantities are observable through the simulator. As already described in the previous section, the structural cable model was found to provide better observability, due to the more intuitive segmented modeling approach. The active star's state and digital signals are completely traceable as well. Using ***EZwave*** waveform viewer, it is possible to investigate all variables, signals and electrical quantities in a well-arranged way. All in all, it is safe to argue that all modeled FlexRay™ Xpert.Sim variables, signals and quantities are perfectly observable.

5.2.3 Simulation Credibility

This term was chosen to rate the overall performance of the simulation environment FlexRay™ Xpert.Sim in a qualitative way. All results and experiences learned during the compilation of this thesis are influencing this expression. Therefore, it is not a single snapshot, but much more some kind of long-time observation.

Throughout this thesis, it is shown that all models are written correctly and generate their desired output. This corresponds to the process of model verification. Additionally, all models involved have

shown to be validated, because their output matches a dedicated real-world behavior. However, the interesting question is: How good do these models reproduce overall trends and tendencies, when small changes are made to the system? This can be answered by looking at certain unique events, experienced during simulation runs. For instance, signal reflection in cables is an effect, which occurred on a reliable and predictable basis. Voltage drops due to low DC-load and over-termination occurred in a deterministic way as well. Concerning signal related properties, it has shown that the FlexRay™ Xpert.Lab hardware environment has some additional properties which are currently not present in the simulation environment. Discrimination thresholds for example, acting as some kind of peak or glitch filter, or better signal integrity due to higher signaling frequencies. For a detailed description of non-modeled effect, see section 5.2.4.

This basic idea of simulation credibility was extended in [KCA+10]. In this submitted publication a validation matrix is introduced, opposing two independent, but significant simulation parameters. By executing different experiments, a validation matrix is set up, allowing to draw conclusions from non-validated areas of the matrix.

All in all, simulation credibility has shown to be established. The most significant properties of FlexRay™'s hardware electrical physical layer are mapped to the simulation environment. General trends and tendencies in the area of signal integrity are visible and comprehensible.

5.2.4 Non-modeled Properties

During research in the area of automotive electronics, transmission lines and twisted pair wires, several physical effects were identified which likely contribute to the overall signal integrity behavior. Some of them are mentioned at this point.

- Skin Effect: As described in [SW98] or [ZSF+08], the well known skin-effect appearing during the use of transmission lines cannot be neglected. Basically, high frequencies are attenuated too much, whereas low frequencies are passing conductors in an almost undamped way. This can have severe impacts on signal integrity, as the skin-effect increases with frequency. Its name is derived from the property, that electric current tends to flow at the "skin" of conductors. The authors of [GJDP08] also describe the need for skin-effect modeling in order to do research in the area of automotive electro-magnetic emissions.

- Eddy Currents: The electric alternating current flowing at the conductor's skin is mainly caused by eddy currents. This effect is described in [JG03]. Both effects, the skin-effect and eddy currents were not modeled nor investigated within this thesis. However, both might have an impact on signal integrity, though.

- Environmental Issues: The automotive environment can be harsh, as introduced in section 1.2. Nevertheless, measurements in the FlexRay™ Xpert.Lab environment have shown that any pieces of metal around a transmission line are acting as a natural low-pass filter. A typical car, predominantly built of metal, encloses all cables and electronics. Because of that, simulation models often showed slightly better signal transmission behavior than physical hardware: Steep edges and sharp overshooting are considered to result from these observations. In order to overcome these problems, an additional filter capacitor was applied at the `BP` and `BM` ports of the bus driver simulation model. The specification describes such a filter capacitor as well, but as measure for electromagnetic compatibility. Depending on the networks topology, capacitor values between $150pF$ and $400pF$ trimmed the resulting signal to look like its hardware counterpart. However, this can be considered as a workaround solution, and therefore still remains as a non-modeled property.

- Connectors: The influence of any connectors or plugs involved in the FlexRay™ Xpert.Lab (and therefore Xpert.Sim) environment was completely disregarded in this thesis. An arbitrary connector or plug has different properties than the used conductor. Additional parasitic capacitances,

inductances and resistances considered in the simulation environment might have a small, yet simulation-improving impact on signal integrity analysis.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis deals with modeling, simulation and validation of FlexRay™ physical layer components in the TEODACS environment. Modeling of automotive networks and components is considered to be a challenging topic. Based on a number of problems, TEODACS tries to enhance the process of automotive network engineering and offers a whole new perspective, together with a sophisticated architecture.

Basic principles of modeling and simulation were used to prepare behavioral and structural models. Advanced concepts were introduced to generate larger, more complex models, like the active star. For this particular model, existing modules had to be re-used, extended and enhanced. The FlexRay™ Xpert.Sim co-simulation environment eased the integration of analog and pure digital models. To ensure the credibility and usability of each model evolved, the process of verification and validation gained special attention. The TEODACS platform provides concepts and appropriate interfaces for these tasks. The proposed multilayer approach has proven to work as expected, within the FlexRay™ Xpert.Sim and Xpert.Lab environments, as well as between them. The generation of appropriate stimuli for various tests also exploited this advantage of the TEODACS platform, enabling insightful tests.

SyAD® has proven to be an ideal co-simulation platform software. It supports a number of very important hardware- and system-description languages, which are relevant to the automotive industry. Furthermore, it supports various state-of-the-art modeling paradigms: Structuring, reusability or parameterization to name a few.

All in all, this thesis shows that a reliable and credible simulation of automotive networks is possible using the TEODACS architecture. The gained results underline the importance of physical layer simulation: Many effects happening on the physical layer are becoming evident on higher layers of communication. Thus, extensive knowledge of the underlying communication system is indispensable.

Two publications have emerged from this thesis, which were accepted at two scientific conferences. [KKS+10] deals with the overall FlexRay™ Xpert.Sim simulation environment, from top-level software components down to bottom-level waveforms transmitted on electrical cables. For this reason, that work underlines a holistic approach to manage the fast-increasing system complexity. [KCA+10] has a strong focus on signal integrity. On the electrical physical layer, a large number of parameters is affecting the shape of the resulting waveforms and the interpreted quality thereof. That work discusses the entire TEODACS test approach and validation process, introducing a new method for analysis and evaluation of signal integrity in FlexRay™ networks.

6.2 Future Work

Concerning the TEODACS project, all major goals have been reached and the co-simulation and hardware prototype platforms were completed as intended. Nevertheless, this section mentions some ideas

for further improvement of the entire TEODACS platform. Most of them are aimed at the Xpert.Sim environment.

6.2.1 Workflow Enhancement

This thesis and the overall topic of automotive network simulation has a strong relation to the automotive business. To make these results accessible for business use, the overall workflow, especially the interaction between the used tools, should be enhanced to ensure an integrated development and test environment. The output of this work would be a perfectly usable development process, ready for deployment in the automotive business.

6.2.2 Improve Simulation Accuracy

Currently, the simulation environment provides sufficient accuracy for network analysis and validation tasks. Modeling of currently non-modeled properties, as discussed in section 5.2.4, could improve accuracy, simulation credibility and overall performance of the simulation environment even more.

6.2.3 Improve Simulation Speed

Simulation speed is always an issue, especially for business use. There are many improvements possible: The most obvious one, higher CPU power, might appear naive. But in most cases, a hardware upgrade is a valid, yet expensive, option. ADVANCEMS™ and SyAD® already support parallelization, however, those features have not been fully deployed within TEODACS yet.

Nomenclature

AC	alternating current
ADC	analog to digital converter
AS	active star
BM	bus minus
BP	bus plus
BSS	byte start sequence
CAN	controller area network
CC	communication controller
DAC	digital to analog converter
DC	direct current
ECU	electronic control unit
EMC	electro-magnetic compatibility
ESD	electro-static discharge
FES	frame end sequence
FPGA	field programmable gate array
IP	intellectual property
LIN	local interconnect network
MOST	media oriented systems transport
MSR	Messen-Steuern-Regeln
OEM	original equipment manufacturer
OSI	open systems interconnect
PC	personal computer
PCB	printed circuit board
TSS	transmission start sequence

Bibliography

[Akt09] Volkswagen Aktiengesellschaft. Distribution of embedded electronic communication systems inside the Volkswagen Phaeton. By courtesy of Volkswagen AG, MultimediaCentrum Fotoservice, 2009.

[Ami00] Miri Amir. *Ausgleichsvorgänge in Elektroenergiesystemen: Mathematische Einführung, elektromagnetische und elektromechanische Vorgänge*. Springer-Verlag Berlin Heidelberg New York, 2000.

[APT03] Peter J. Ashenden, Gregory D. Peterson, and Darrell A. Teegarden. *The System Designer's Guide to* VHDL-AMS*: Analog, Mixed-Signal, and Mixed-Technology Modeling*. Morgan Kaufmann, San Francisco, California, 2003.

[Arm06] Eric Armengaud. Low Level Bus Traffic Replay for the Test of Time-Triggered Communication Systems. In *Design and Diagnostics of Electronic Circuits and systems, 2006 IEEE*, pages 153–154, 0-0 2006.

[Arm08] Eric Armengaud. *Log File Format User's manual*. The Virtual Vehicle Competence Center, 2008. Version 0.1.

[ASHP04] E. Armengaud, A. Steininger, M. Horauer, and R. Pallierer. A layer model for the systematic test of time-triggered automotive communication systems. In *Factory Communication Systems, 2004. Proceedings. 2004 IEEE International Workshop on*, pages 275–283, Sept. 2004.

[ATK+09] Eric Armengaud, Allan Tengg, Michael Karner, Christian Steger, Reinhold Weiss, and Martin Kohl. Moving Beyond the Component Boundaries for Efficient Test and Diagnosis of Automotive Communication Architectures. In *Proceedings of the 14th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2009)*, 2009.

[AW03] Mario Anton and Jürgen Weber. Simulationsmethodik für integrierte Mixed-Signal Automotive Schaltkreise. In *ANALOG 2003 - Entwicklung von Analogschaltungen mit CAE-Methoden*, volume 177, pages 103–108. Atmel Germany GmbH, Heilbronn, VDE Verlag GmbH Berlin Offenbach, September 2003.

[AWK+09] Eric Armengaud, Daniel Watzenig, Michael Karner, Christian Steger, Reinhold Weiss, Christian Netzberger, Martin Kohl, Markus Pistauer, Felix Pfister, and Harald Gall. Combining the Advantages of Simulation and Prototyping for the Validation of Dependable Communication Architectures: the TEODACS Approach. In *SAE International Journal of Passenger Cars - Electronic and Electrical Systems, V118-7*, number 2009-01-0763, October 2009. Present at SAE World Congress, Detroit, Michigan, USA, April 2009. 10 pages.

[AWS+08] Eric Armengaud, Daniel Watzenig, Christian Steger, Hubert Berger, Harald Gall, Felix Pfister, and Markus Pistauer. TEODACS: A new Vision for Testing Dependable Automotive Communication Systems. In *Proceedings of the Third International Symposium on Industrial Embedded Systems*, pages 257–260, 2008.

[Bol06] D. Bollati. FlexRay - Simulation of Physical Layer Topologies. In *Synopsis User Group Europe 2006*, 2006.

[Bro87] Il'ja N. Bronstein. *Taschenbuch der Mathematik, 23*. Benedictus Gotthelf Teubner Verlagsgesellschaft, Leipzig, und Verlag Nauka, Moskau, 1987.

[Cen09] The Virtual Vehicle Competence Center. Virtual Vehicle: E03_T01 TEODACS. Online, October 2009, 2009. `http://www.teodacs.com`.

[CIS08] CISC Semiconductor GmbH. *Model Notes CISC FlexRay Transceiver*, 2008.

[CIS09] CISC Semiconductor GmbH. *Top-level Description CISC FlexRay Transceiver*, 2009.

[Cla09] Federico Clazzer. Design of a dedicated software tool for the automatic analog layer signal integrity analysis of the FlexRay communication bus. Technical report, The Virtual Vehicle Competence Center, 2009.

[CVRS09] Muller Candice, Maurizio Valle, Buzas Roman, and Augustin Skoupy. Mixed-Mode Behavioral Model of Flexray Physical Layer Transceiver, 2009.

[Fle06a] The FlexRay Consortium. *FlexRay Communications System Electrical Physical Layer Specification*, 2006. Version 2.1, Revision B.

[Fle06b] The FlexRay Consortium. *FlexRay Electrical Physical Layer Application Notes*, 2006. Version 2.1 Revision B.

[GAEC06] D. Guihal, L. Andrieux, D. Esteve, and A. Cazarre. VHDL-AMS Model Creation. In *Mixed Design of Integrated Circuits and Systems, 2006. MIXDES 2006. Proceedings of the International Conference*, pages 549–554, June 2006.

[GB07] Thorsten Gerke and David Bollati. Development of the Physical Layer and Signal Integrity Analysis of FlexRay Design Systems. In *SAE 2007 Transactions Journal of Passenger Cars: Electronic and Electrical Systems*, pages 665–675, 2007.

[GB09] Thorsten Gerke and David Bollati. An Automated Model Based Design Flow for the Design of Robust FlexRay Networks. *SAE International Journal of Passenger Cars - Electronic and Electrical Systems*, 1(1):457–466, 2009.

[Ger07] Thorsten Gerke. Developing a robust CAN physical layer implementation for In-Vehicle Networks using the Robust Design Method. In *Automotive Electronics, 2007 3rd Institution of Engineering and Technology Conference on*, pages 1–6, June 2007.

[GJDP08] M. Gursoy, S. Jahn, B. Deutschmann, and G. Pelz. Methodology to Predict EME Effects in CAN Bus Systems Using VHDL-AMS. *Electromagnetic Compatibility, IEEE Transactions on*, 50(4):993–1002, Nov. 2008.

[GRR08] Mariagrazia Graziano and Massimo Ruo Roch. An Automotive CD-Player Electro-Mechanics Fault Simulation Using VHDL-AMS. *J. Electron. Test.*, 24(6):539–553, 2008.

[GS05] Thorsten Gerke and Carsten Schanze. Development and Verification of In-Vehicle Networks in a Virtual Environment. In *In-Vehicle Networks and Software*, 2005.

[Haa03] J. Haase. Regeln für die Erstellung von VHDL-AMS-Modellen. In *Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen*, 2003.

[Hal94] Karen Hale. Automotive databus simulation using VHDL. In *EURO-DAC '94: Proceedings of the conference on European design automation*, pages 592–597, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[HD04] Martin Horn and Nicolaos Dourdoumas. ***Regelungstechnik***. Pearson Studium, 2004.

[Hes03] E. Hessel. Kfz-Bordnetz: Einfache Teilmodelle in VHDL-AMS, 2003. Hella KG Hueck & Co., Simulation und Software-Entwicklung.

[JG03] Howard Johnson and Martin Graham. ***High-speed Signal Propagation: Advanced Black Magic***. Prentice Hall Press, Upper Saddle River, NJ, USA, 2003.

[Joh02] Crisp John. ***Introduction to Copper Cabling***. Newnes, 2002.

[KAS09] Michael Karner, Eric Armengaud, and Christian Steger. Run-time switching of heterogeneous simulation models, 2009. Austrian patent application A1479/2009.

[KCA+10] Martin Krammer, Federico Clazzer, Eric Armengaud, Michael Karner, Christian Steger, and Reinhold Weiss. Exploration of the flexray signal integrity using a combined prototyping and simulation approach. Design and Diagnostics of Electronic Circuits and Systems, April 2010, Vienna, Austria, 2010. Accepted for publication.

[KK10] Martin Krammer and Michael Karner. ***Active Star Co-Simulation with SyAD***, 2010.

[KKS+10] Michael Karner, Martin Krammer, Christian Steger, Stefan Krug, and Reinhold Weiss. Heterogeneous Co-Simulation Platform for the Efficient Analysis of FlexRay-based Automotive Distributed Embedded Systems. 2010. Accepted for publication at the 8th IEEE International Workshop on Factory Communication Systems, Communication in Automation.

[Kro07] Kromberg & Schubert. ***Datenblatt FlexRay Datenleitung, Anwendung im Kabelbaum, Typ FLR09YS-YW***, 2007.

[Kru08] Stefan Krug. Multi-Level Replay and Hardware Interfacing in a FLEXRAY Network Simulation. Technical report, Institute of Technical Informatics, Graz University of Technology, 2008.

[KSSP06] Suad Kajtazovic, Christian Steger, Andreas Schuhai, and Markus Pistauer. ***Automatic Generation of a Coverification Platform***, pages 187–203. Springer Netherlands, 2006.

[Kup07] Tony R. Kuphaldt. Lessons In Electric Circuits. Online, January 2009, 2007. `http://www.allaboutcircuits.com`.

[Lao02] R. Lao. The Twisted-Pair Telephone Transmission Line. ***High Frequency Electronics***, November 2002.

[LB07] W. Lawrenz and D. Bollati. Validation of in-vehicle-protocol network topologies. In ***Systems, 2007. ICONS '07. Second International Conference on***, pages 24–24, April 2007.

[Mac05] Charles M. Macal. Model Verification and Validation. Presentation of Workshop on Threat Anticipation: Social Science Methods and Models, April 2005.

[Men07] Mentor Graphics. ***EZwave User's and Reference Manual***, 2007. Software Version 2.8_1.1 Release 2007.2.

[Men08] Mentor Graphics. ***ADVance MS User's Manual***, 2008. Software Version 4.7_2 Release AMS 2007.2a.

[MM98] E. Moser and N. Mittwollen. VHDL-AMS: the missing link in system design - experiments with unified modelling in automative engineering. In ***DATE '98: Proceedings of the Conference on Design, Automation and Test in Europe***, pages 59–65, Washington, DC, USA, 1998. IEEE Computer Society.

[Mos96] Eduard Moser. Automotive model exchange using VHDL-AMS: a benchmark. In *IEEE International Symposium on Computer-Aided Control System Design*. Robert Bosch GmbH, Research Center, September 1996.

[Net08] Christian Netzberger. Design Document for the Active Star VHDL Model. Technical report, FH Joanneum Kapfenberg, 2008.

[PFSLP07] Anton Pirker-Frühauf, Karsten Schönherr, Arnaud Laroche, and Georg Pelz. Worst-Case Modeling and Simulation of an Automotive Throttle in VHDL-AMS, 2007.

[Rau08] Mathias Rausch. FLEXRAY. Carl Hanser Verlag Muenchen Wien, 2008.

[Ros98] Antonio Rosati. Fiber Optics and the High-Speed Data Communication in Vehicles. In *VII International Mobility Technology, Conference and Exhibit*. Delphi Packard Electric Systems, SAE Brazil, Sao Paulo, November 1998. Document Number: 982877.

[Sar98] Robert G. Sargent. Verification and validation of simulation models. In *WSC '98: Proceedings of the 30th conference on Winter simulation*, pages 121–130, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[SHM01] Michael Schlegel, Göran Herrmann, and Dietmar Müller. Entwicklung eines effizienten VHDL-AMS-Modells der verlustbehafteten Leitung. In *Tagungsband 5. Chemnitzer Fachtagung Mikrosystemtechnik - Mikromechanik und Mikroelektronik, Chemnitz, 23./24. Oktober 2001*, pages 12–16, 2001. ISBN: 3-00-008201-8.

[SW98] B.K. Sen and R.L. Wheeler. Skin effects models for transmission line structures using generic SPICE circuit simulators. In *Electrical Performance of Electronic Packaging, 1998. IEEE 7th topical Meeting on*, pages 128–131, Oct 1998.

[sya09] CISC *SyAD* product folder. CISC Semiconductor Design+Consulting GmbH, Lakeside B07, 9020 Klagenfurt, Austria, 2009.

[Vas04] Thomas Vasek. Wird das Auto zu komplex? *Technology Review*, Juli 2004.

[Vir08] The Virtual Vehicle Competence Center. *LogFileGenerator User's manual*, 2008.

[WSCW10] Chua-Chin Wang, Gang-Neng Sung, Po-Cheng Chen, and Chin-Long Wey. A Transceiver Front End for Electronic Control Units in FlexRay-Based Automotive Communication Systems. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 57(2):460–470, Feb. 2010.

[XKC+08] Yi-Nan Xu, Y.E. Kim, K.J. Cho, J.G. Chung, and M.S. Lim. Implementation of FlexRay communication controller protocol with application to a robot system. In *Electronics, Circuits and Systems, 2008. ICECS 2008. 15th IEEE International Conference on*, pages 994–997, 31 2008-Sept. 3 2008.

[ZK09] Carlos Alberto Zerbini and Martin Kohl. Dokumentation für VHDL-AMS Physical Layer Komponenten. Technical report, University of Applied Sciences FH Joanneum, The Virtual Vehicle Competence Center, 2009.

[ZSF+08] H. Zhang, K. Siebert, S. Frei, T. Wenzel, and W. Mickisch. Multiconductor transmission line modeling with VHDL-AMS for EMC applications. In *Electromagnetic Compatibility, 2008. EMC 2008. IEEE International Symposium on*, pages 1–6, Aug. 2008.