Hindawi Publishing Corporation
EURASIP Journal on Embedded Systems
Volume 2010, Article ID 726286, 11 pages
doi:10.1155/2010/726286

Research Article

A Systematic Development Methodology for Mixed-Mode Behavioral Models of In-Vehicle Embedded Electronic Systems

Candice Muller,[1] Maurizio Valle,[1] William Prodanov,[2] and Roman Buzas[3]

[1] *Department of Biophysical and Electronic Engineering, University of Genoa, Opera Pia 11A, 16145 Genoa, Italy*
[2] *Department of Product Development, Chipus Microelectronics, Lauro Linhares 589, 88036-001 Florianopolis, SC, Brazil*
[3] *Department of Product Development, Automotive IVN (In vehicle networking), ON Semiconductors Inc., Videnska 125, 619 00 Brno, Czech Republic*

Correspondence should be addressed to Candice Muller, candice.muller@unige.it

Received 28 May 2009; Accepted 26 October 2009

Academic Editor: Luca Fanucci

The rising demands for safety, power-weight reduction, and comfort make the in-vehicle network of embedded electronic systems very complex. In particular system reliability is essential, especially because of the safety requirements. Test and verification of the entire in-vehicle network by means of behavioral simulations are each time more widely adopted. To this aim, behavioral models that faithfully represent the behavior of mixed-mode-embedded systems are essential for achieving reliable simulation results. This paper presents a systematic development methodology for mixed-mode behavioral models of in-vehicle-embedded systems. The methodology allows achieving accurate models, which provide reliable system simulations. The model development methodology is described and the results of the methodology applied to two case studies are presented: (1) the mixed-mode behavioral model of a generic Flexray physical layer transceiver and (2) the mixed-mode behavioral model of a CAN bus transceiver-integrated circuit. The simulation results show that behavioral simulations are much faster than transistor level simulations. Moreover, behavioral simulations are flexible, which allows quickly changing and verifying the communication network topology if compared with hardware prototypes.

1. Introduction

The amount of electronics used in vehicle systems is growing fast with the replacement of purely mechanical or hydraulic systems for electronic ones. Each function is implemented by an Electronic Control Unit (ECU), that is, an embedded system; ECUs communicate between them through a fieldbus communication network.

The powertrain and chassis control systems are directly related with the safety of the vehicle's behavior and consequently, with the safety of the occupants [1]. The rising demands for safety, power-weight reduction, and comfort makes the in-vehicle-embedded electronic systems very complex. In particular system reliability is essential, especially because of the safety requirements. Moreover, the in-vehicle-embedded system asks for suitable techniques to assess the system dependability.

Design verifications are compulsory even during the early stages of the system design. Verifications can be done through prototypes tests or circuit simulations. System prototypes are expensive and time consuming. Furthermore, it is difficult to represent the worst case scenarios, because it is usually not possible to set all system parameters in order to reproduce the worst case conditions. Time and investments are necessary to implement hundreds of different topologies and analyze their behaviors. On the other hand, transistor level simulations of such complex systems, like Spectre, are often practically impossible because of the enormous computational time required due the many interactions of all nonlinearities [2]. A possible and efficient solution to the verification problem is to use behavioral simulations. Behavioral simulations can be used to guarantee the correct system behavior, avoiding unneeded hardware development, forecasting design problems, and, consequently, reducing

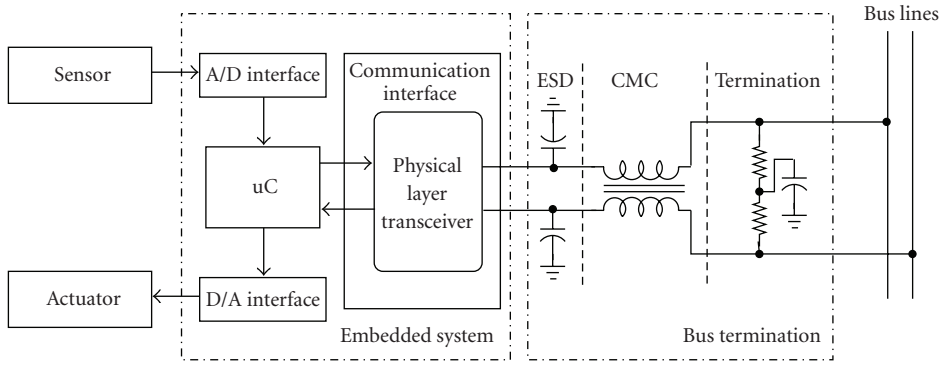

Figure 1: Generic embedded system block diagram.

considerably cost and time to market. In addition, behavioral simulations allow the total environment controllability, which makes very simple to set the boundary conditions.

Some works on behavioral modeling related to in-vehicle communication systems are reported in literature. The challenge of networks developers on dealing with the signal integrity of the communication system physical layer implementation is exposed in [3], where a validation methodology of in-vehicle protocol networks topologies, based on behavioral models, is presented.

A virtual environment of a complete CAN network model and the importance of simulations of in-vehicle networks at the early stage of the design process, in order to reduce the number of prototypes and cost and time to market, are highlighted in [4].

The work in [5] presents the development of the physical layer and signal integrity analysis for Flexray communication systems, while [6] introduces an automated simulation-based methodology based on the guidelines and criteria defined in the Flexray physical layer specification [7], focusing on the network design verification methodology.

In order to achieve reliable results on networks test and verification through the use of techniques and methodologies based on behavioral simulations (e.g., the ones presented in the previews paragraphs), it is necessary to have faithful behavioral models, which accurately represent the behavior of the real ECUs.

A generic in-vehicle electronic embedded system (i.e., ECU) block diagram is presented in Figure 1. It is composed by a by microcontroller, communication interface, and the A/D and D/A interfaces to sensors and actuators. The communication interface connects to the bus line and the bus termination, which can contain common mode choke (CMC) and electromagnetic discharge protection elements (ESD).

The main difficulty on modelling the embedded system is on modelling the mixed-mode communication interface. The electronic device that implements this block is the physical layer transceiver. It is responsible for converting the digital microcontroller instructions in analog signals on the bus lines and vice versa. The difficulty on modelling the transceiver is mainly due the fact that it is a mixed-mode circuit. The voltage on the bus lines can achieve high levels

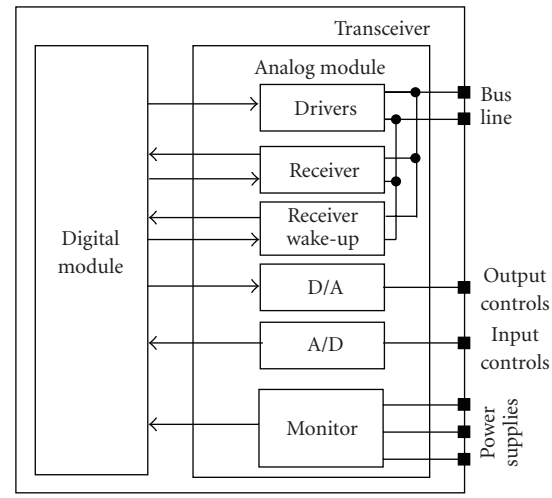

Figure 2: Generic block diagram of the physical layer transceiver.

and silicon transceiver bus drivers must be developed in a technology which allows such high voltages. The digital blocks are implemented with CMOS technology and work with low voltage levels.

The accuracy of the mixed mode behavioral model of the physical layer transceiver directly influences the bus line signal integrity, once the transceiver is the responsible for writing and reading the analog data on the bus lines. Figure 2 shows a generic block diagram of the fieldbus physical layer transceiver.

Another important issue is the modelling of the electromagnetic interference (EMI). The communication systems based on electrical buses are important sources of electromagnetic emissions. Furthermore, the in-vehicle network immunity against the EMI produced by the other electronic equipments placed in the vehicle must be investigated.

The behavior of the physical layer transceiver is defined according to physical layer communication protocol. The most widely used in-vehicle communication protocol is the Controller Area Network (CAN) [8]. CAN is a serial communication protocol that defines a differential voltage to represent recessive and dominant states on a wired line and allows bit rates up to 1.0 Mbps.

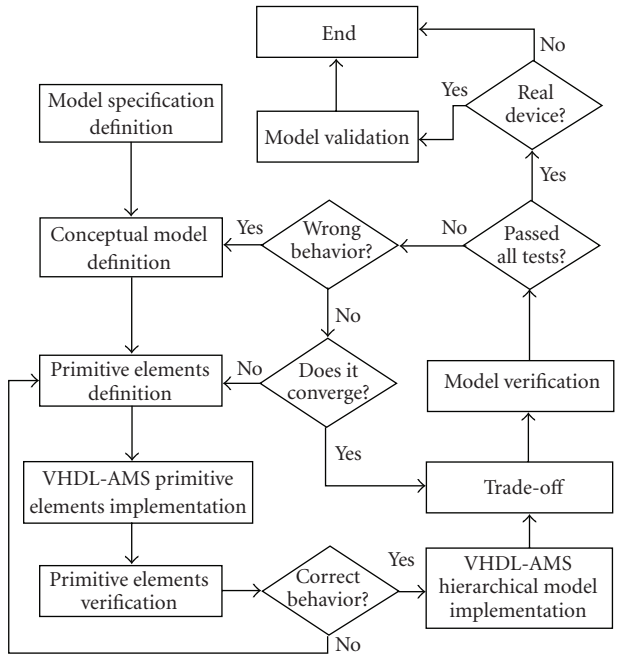

Figure 3: Model development methodology flow chart.

However, the fast growth in automotive control systems demands data rates and reliability not achieved by CAN communication buses. Requirements for actual in-vehicle control applications include the combination of higher data rates, deterministic behavior, and the support of fault tolerance. The Flexray communication system [9] can handle these requirements. It is an automotive standard hybrid protocol that combines time-triggered and event-triggered messages, it is fault-tolerant, and it supports high-speed data communication, up to 10.0 Mbps. Trends point Flexray as the communication protocol of these new in-vehicle applications.

The aim of this paper is to present a systematic development methodology for mixed-mode behavioral models of in-vehicle-embedded electronic systems. The goal of the methodology is to develop accurate models, which provides reliable system simulations. Two case studies are presented, in order to demonstrate the methodology results: the physical layer CAN transceiver and the physical layer Flexray transceiver.

We have used VHDL-AMS hardware description language for the behavioral models implementation, due to the fact that it is an industrial standard modelling language and it is widely supported by the available mixed-mode circuit simulators. Furthermore, it provides features for modelling analog, digital, and mixed-mode systems and it allows to use multiple energy domains, such as electromechanical, electro-optical, and thermal-electrical systems, which allows the complete embedded system modelling (including sensors and actuators).

The paper is organized in four sections, including this introduction. Section 2 presents the model development methodology. Section 3 shows two case studies, gives examples of the model utilization, and presents the advantages of the behavioral simulations compared with hardware prototypes and transistor level simulations, while Section 4 presents the conclusions.

2. Model Development Methodology

The behavioral simulation of complex systems, for example, automotive mixed-mode-embedded system, requires reliable model implementation. In order to achieve this requirement, we present a systematic development methodology, which is divided in four steps, as follows:

(A) model specification definition,

(B) model design and implementation:

(1) conceptual model definition,
(2) primitive elements definition,
(3) VHDL-AMS primitive elements implementation,
(4) primitive elements verification,
(5) VHDL-AMS hierarchical model implementation,

(C) tradeoff,

(D) conformance test:

(1) model verification,
(2) model validation.

Figure 3 shows the model development methodology flow chart.

The model accuracy is very important for signal integrity investigation. Simulation speed-up is mandatory for simulating complete networks in a reasonable CPU usage time. At last but not least, it is necessary to care about convergence issues. It does not matter how fast and accurate the model is if it is not able to converge in all operating conditions [10]. The proposed development methodology faces these aspects, finding a compromise between them.

The previews steps are detailed in the next subsections.

2.1. Model Specification Definition. The model specification can be divided in two different categories, according to the kind of model that must be implemented:

(i) generic device model,

(ii) real silicon device model.

The generic device model is the model of a generic device, that is, a model that fulfills the protocol specification (e.g., the Controller Area Network protocol (CAN)). It means that the model specification is based on the specifications defined by the protocol, respecting the protocol requirements as operation modes, timing characteristics, electrical parameters as voltages levels, I/O signals characteristics, I/O pins impedances, and so forth. It is also compliant with the functionalities defined in the protocol.

The generic device model can be used as proof of concept for silicon design development, as support tool during the digital block design and verification and it also can be tuned for representing a real silicon device.

On the other hand, the real silicon device model is the model of a real device. In this case, the model specification is based on the device data sheet (that is compliant with the protocol specification).

The real silicon device model can be used for testing and verifying the behavior of the in-vehicle network topologies.

The model specification should address the following:

(i) the model requirements, for example, bus operating modes, power supplies validity ranges, timings, electrical characteristics, and so forth;

(ii) the model parameterization, for example, typical behavior and corner cases parameters;

(iii) the features, for example, statistic simulation, monitoring system, diagnosis, power consumption estimation, and so forth;

(iv) the model evaluation definition, that is, how the model accuracy must be evaluated.

2.2. Model Design and Implementation. The model design and implementation goes from the conceptual model definition until the code implementation.

2.2.1. Conceptual Model Definition. The conceptual model is defined considering all parameters described in the model specification. It consists of describing how the model requirements are broken down into a collection of components, how the components fit together and interact, and how they work together to meet the specifications. It is a mathematical/logical representation of the problem [11]. Furthermore, each component has to be decomposed until the level of primitive elements, that is, basic circuit cells.

In practical terms, it consists of taking the transceiver and dividing it in a hierarchical way, until the primitive elements level (also called basic cells). The result is a hierarchically composed model. The advantage of the methodology is that the primitive elements can be reused for modelling different subsystems (i.e., reusability) [12].

2.2.2. Primitive Elements Definition. The definition of how the primitives elements are implemented is one of the most important steps of behavioral modelling. It is necessary to pay special attention to the possible discontinuities in the analog domain of the mixed-mode circuit, because the discontinuities can cause convergence problems and instability. Therefore, smooth transitions of the analog variables should be used.

The primitive elements contain one or more definitions. Each definition is implemented by an architecture in the VHDL-AMS primitive elements implementation (please refer to Section 2.2.3). The behavior of the primitive elements can be described at different abstraction levels. One can use mathematical expressions in a very high level of abstraction, while the other can use a low level of abstraction modeling approach resulting in a description very close to the electronic circuit.

As example of primitive element definition lets us consider the model which represents a MOS transistor. It has three terminals (G, D, S) and the voltage level at the terminals defines the region of operation (cutoff, linear, and saturation). Four different architectures using different abstraction levels were defined and are presented as follows.

(a) *First Definition.* The first definition uses a low level of abstraction, considering the physical parameters of a real silicon transistor for modelling the complete behavior of all operation regions. The mathematical expressions are defined as [13]

$$K_n = \mu_n C_{\mathrm{OX}} \frac{W}{L}, \tag{1}$$

Cutoff region, $V_{\mathrm{GS}} \leq V_{\mathrm{TN}}$:

$$I_{\mathrm{DS}} = 0.0A, \tag{2}$$

Linear region, $V_{\mathrm{GS}} - V_{\mathrm{TN}} \geq V_{\mathrm{DS}} \geq 0.0V$:

$$I_{\mathrm{DS}} = K_n \left(V_{\mathrm{GS}} - V_{\mathrm{TN}} - \frac{V_{\mathrm{DS}}}{2} \right) V_{\mathrm{DS}}, \tag{3}$$

Saturation region, $V_{\mathrm{DS}} \geq V_{\mathrm{GS}} - V_{\mathrm{TN}} \geq 0.0V$:

$$I_{\mathrm{DS}} = \frac{K_n}{2} (V_{\mathrm{GS}} - V_{\mathrm{TN}})^2 (1 + \lambda V_{\mathrm{DS}}), \tag{4}$$

where μ_n is electron mobility, C_{OX} is oxide capacitance per unit area, W is channel width, L is channel length, I_{DS} is drain-source current, V_{GS} is gate-source voltage, V_{TN} is threshold voltage, and V_{DS} is drain-source voltage.

The result is a description similar to the one used in transistor level simulators, as Spice, for example.

(b) *Second Definition.* Using a higher abstraction level, the transistor can be implemented describing each operation region by means of a linear approximation. The result is a piecewise linear model. The behavior of the model is controlled by the voltage between G and S terminals. If V_{GS} is smaller than V_{TN}, the transistor is in cutoff region; otherwise it behaves as follows:

Saturation region, $V_{\mathrm{DS}} > V_{\mathrm{SAT}}$:

$$I_{\mathrm{DS}} = I_{\mathrm{SAT}}, \tag{5}$$

Linear region, $V_{\mathrm{SAT}} \geq V_{\mathrm{DS}} \geq V_{\mathrm{REV}}$:

$$I_{\mathrm{DS}} = \frac{V_{\mathrm{DS}}}{R_{\mathrm{ON}}}, \tag{6}$$

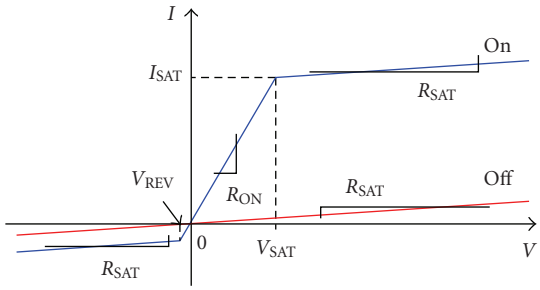

Figure 4: $V_{DS} \times I_{DS}$ piecewise linear approximation.

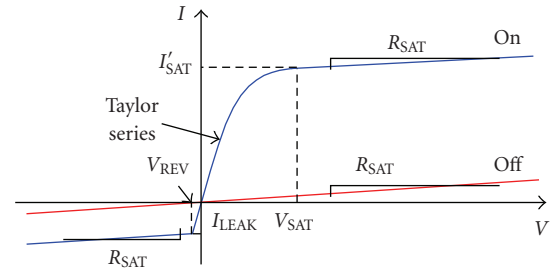

Figure 5: $V_{DS} \times I_{DS}$ Taylor series expansion piecewise approximation.

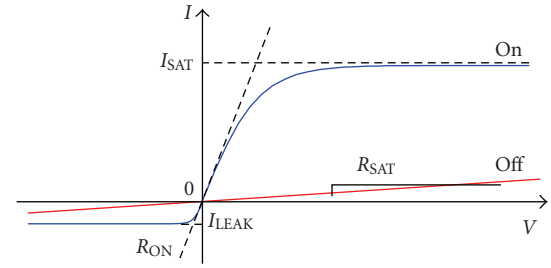

Figure 6: $V_{DS} \times I_{DS}$ Hyperbolic tangent piecewise approximation.

Cutoff region, $V_{REV} > V_{DS}$:

$$I_{DS} = 0.0A, \tag{7}$$

where I_{SAT} is saturation current, R_{ON} is output resistance, $V_{SAT} = I_{SAT} \cdot R_{ON}$, and R_{ON} and I_{SAT} are model parameters.

In order to avoid a null $\partial i/\partial v$ derivative, a linear current term is added in all operation regions. Therefore, the output current will be

$$I'_{DS} = I_{DS} + \frac{V_{DS}}{R_{SAT}}. \tag{8}$$

The *v-i* characteristics of the piecewise linear transistor are shown in Figure 4.

(c) *Third Definition.* The piecewise linear architecture has two points of first-order derivative discontinuities: the transition between cutoff and linear regions and the transition between linear and saturation regions. Substituting the piecewise linear equation of the linear region by means of a Taylor series expansion polynomial equation we obtain a smoothed transition between the linear and saturation regions. The model definition is given by

Saturation region, $V_{DS} > V_{SAT}$:

$$I_{DS} = I_{SAT}, \tag{9}$$

Linear region, $V_{SAT} \geq V_{DS} \geq V_{REV}$:

$$I_{DS} = I_{SAT} \cdot \left(kV_{DS} - \frac{(kV_{DS})^3}{6} + \frac{(kV_{DS})^5}{120} \right), \tag{10}$$

Cutoff region, $V_{REV} > V_{DS}$:

$$I_{DS} = 0.0A, \tag{11}$$

where

$$k = \sqrt{6 \cdot \left(1 - \sqrt{\frac{1}{3}} \right)} \cdot V_{SAT}^{-1}. \tag{12}$$

Adding the linear current to all operation regions we have

$$I'_{DS} = I_{DS} + \frac{V_{DS}}{R_{SAT}}. \tag{13}$$

This definition has a DC characteristic closer to the real transistor, increasing the model accuracy, and a smoother transition on V_{SAT} point. Figure 5 shows the *v-i* characteristics of the third definition.

(d) *Fourth Definition.* The fourth definition makes use of the hyperbolic tangent approximation, which has a natural asymptotic characteristic. Moreover, only two regions must be defined to describe the whole transistor operation, as follows:

Active region, $V_{DS} > 0.0V$:

$$I_{DS} = I_{SAT} \cdot \tanh\left(\frac{V_{DS}}{\tau_1} \right), \tag{14}$$

Cutoff region, $V_{DS} \leq 0.0V$:

$$I_{DS} = I_{LEAK} \cdot \tanh\left(\frac{V_{DS}}{\tau_2} \right), \tag{15}$$

where $\tau_1 = I_{SAT} \cdot R_{ON}$, $\tau_2 = I_{LEAK} \cdot R_{ON}$, and I_{LEAK} is leakage current.

The output characteristics are shown in Figure 6.

Further information about the primitive cells definition can be found in [10].

The availability of primitive element with more than one definition allows to build higher hierarchies with different performances, which can be used according to the user needs. Please refer to Section 3.

2.2.3. VHDL-AMS Primitive Elements Implementation. After defining the mathematical expressions which represent each one of the primitive elements, it is time to convert it to the VHDL-AMS language description. A VHDL-AMS code file is written for each primitive element/architecture.

2.2.4. Primitive Elements Verification. Before using the primitive element it is necessary to verify if it is correctly

implemented. Test cases are generated in order to verify the primitive elements behavior. If the element has more than one architecture, the same test cases are used. All the element architectures must be verified.

If the verification tests are successful, the basic cell is ready to be used. Otherwise, it is necessary to review the primitive element definition.

2.2.5. VHDL-AMS Hierarchical Model Implementation. The VHDL-AMS hierarchical model implementation is based on the conceptual model definition. Usually the implementation is divided in steps, according to the number of hierarchical levels. Further test cases are generated in order to verify the correct model behavior at the different levels. This procedure improves the model reliability and helps to solve design problems, once undesirable behaviors are detected early at lower hierarchical levels.

2.3. Tradeoff. One of the main difficulties on developing mixed-mode behavioral circuits is on finding a tradeoff between speed, accuracy, and convergence. The accuracy is strictly related with the level of abstraction used during the primitive elements implementation. The lower the abstraction level is, the more accurate is the model. Low abstraction levels usually use mathematical equations which require high computational effort. The first important point is on finding a good relationship between accuracy and speed. The second one is related to the convergence issue. Discontinuities between piecewise regions must be avoided as well as null first-order derivatives. Refer to Section 2.2.2.

The tradeoff between speed, accuracy, and convergence is faced by applying the available architectures implemented in the basic cells. Architectures with different abstraction levels give different performances in terms of accuracy and speed as well as different levels of convergence stability.

The methodology allows implementing different architectures for the same behavioral model, each one presenting different abstraction level, accuracy, speed, and convergence performance. The use of one or other architectures depends on the user requirements. Accurate models are important for signal integrity analysis, while faster models can be used for simulations which focus on, for example, the functionalities verification.

In addition, different architectures can include or not some supported features. It means that, with the same accuracy, the model can be faster if some features are not implemented. Please refer to Section 3.2.

2.4. Conformance Test. The conformance test is divided in two parts: verification and validation. These two concepts are often considered as a single process, but actually there is a distinct focus on each one: verification focuses on the model capability and validation focuses on the model credibility [14].

2.4.1. Model Verification. Verification is the process of determining if the model implementation accurately represents the conceptual description and specifications [14]. It consists of verifying all the model requirements (operating modes, validity ranges, timings, electrical characteristics, etc.), the correct parameterization, and the implementation of the supported features.

The model verification is the main purpose of the conformance test. A set of test cases are defined, in order to fully verify the model. The test cases are independent of each other. For each test case the following must be defined:

(i) the purpose,

(ii) the configuration setup,

(iii) the execution steps,

(iv) the pass/failure criterion.

The pass/failure criterion is defined according to the model specification. Examples are reported in Section 3.

The model robustness is also verified, considering stress conditions during the test cases.

The test cases are basically defined according to the conformance test of the communication protocol, for example, *Flexray physical layer conformance test specification* [15] for the Flexray communication system.

If the model fails during the verification tests, it is necessary to return to the preview steps. In this case, it is necessary to analyze the failure results to decide where to act. If the model implements wrong behavior, it is necessary to review the conceptual model definition. If the model behaves correctly, but it is not sufficiently accurate, it may be necessary to use another primitive element architecture (tradeoff step). If the simulation does not converge, it is necessary to review the primitive elements definition (see Figure 3).

The model verification must be done for the generic and for the real device models. Examples of model verification are reported in Section 3.1.

2.4.2. Model Validation. Validation process checks if the model accurately represents the real device from the perspective of the intended use of the model [14]. It consists of comparing the model simulation results with the device measurements.

The model validation can be done just for real device models. Refer to Section 3.2.

3. Results

In this section we present the application of the methodology described in Section 2, showing two case studies: the generic mixed-mode behavioral model of Flexray physical layer transceiver and the real silicon device mixed-mode behavioral model of CAN bus transceiver.

Through the case studies we demonstrate some steps of the model development methodology, giving special attention to the model tradeoff and conformance test. In Section 3.1 we present some results of the model verification and in the Section 3.2 we present how the model tradeoff was done and also some results of the model validation.

Section 3.3 presents the advantages of using behavioral simulations.

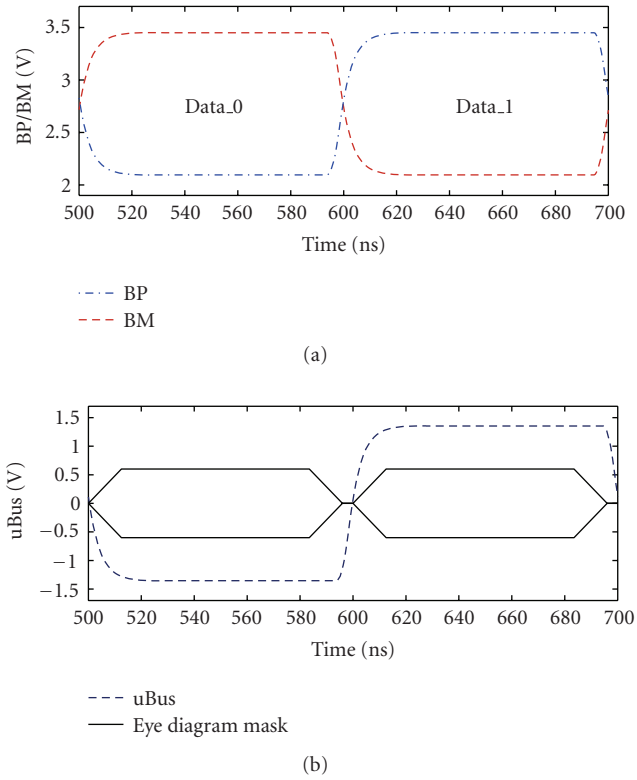

Figure 7: BP and BM bus lines signal integrity simulation.

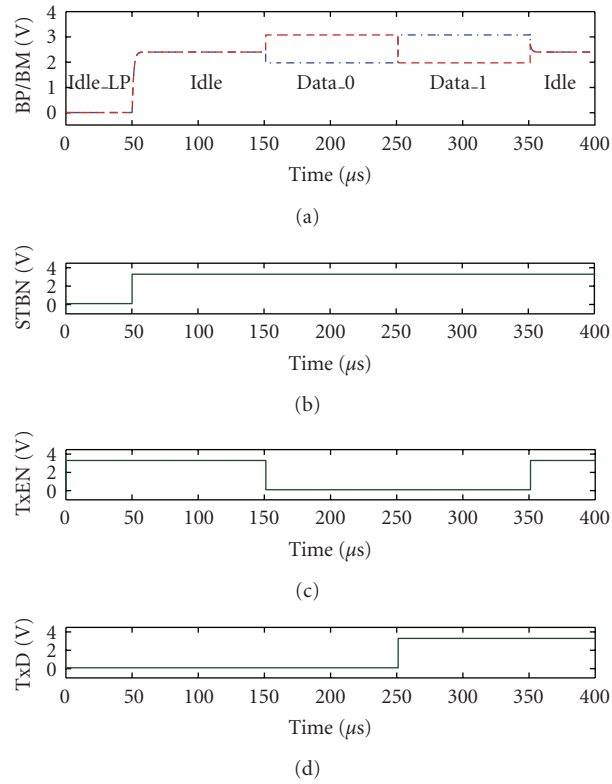

Figure 8: Bus state transitions simulation.

3.1. Flexray Physical Layer Transceiver. The mixed-mode behavioral model of Flexray physical layer transceiver reported in [16] is fully compatible with the Flexray standard [9].

Some examples of model verification are reported. Figure 7 shows the bus signal integrity verification. The verification is done comparing the differential voltage on the bus (uBus) with the eye diagram mask. The pass/failure criterion defines that, for passing the test, the uBus must respect the minimum protocol requirements represented by the eye diagram mask.

BP and BM represent the bus line voltages (Figure 7(a)) and uBus the differential voltage on the bus (Figure 7(b)). From 500 nanoseconds to 600 nanoseconds the transceiver transmits Data_0 (TxD = Low). During this time uBus is negative. At 600 nanoseconds the transceiver is set to transmit Data_1 (TxD = High) and uBus goes to a positive value. In both data transmissions, uBus respects the minimum protocol requirements.

Figure 8 shows the verification of the model behavior during bus states transitions. The transceiver input control signals STBN, TxEN, and TxD were set in order to generate the following bus state sequence: Idle_LP, Idle, Data_0, Data_1, and Idle. Figures 8(b)–8(d) show the input control signals and Figure 8(a) shows the state transitions on BP/BM bus lines.

The pass/failure criterion defines the following behavior for the different bus states.

(i) Idle_LP: no current is driven either to BP or to BM and the bus drivers bias both BP and BM to GND level.

(ii) Idle: no current is driven to BP and BM and the bus drivers bias both BP and BM to a certain level of voltage (around Vcc/2).

(iii) Data_0: at least one bus driver forces a positive differential voltage between BP and BM.

(iv) Data_1: at least one bus driver forces a negative differential voltage between BP and BM.

While STBN signal is at low level, the transceiver is in Idle_LP state and BP/BM is biased to GND. When STBN goes to high level (TxEN is still set high), the transceiver goes to Idle state and BP/BM goes to a level of voltage around Vcc/2. In both bus states, Idle_LP and Idle, the differential bus voltage value (uBus) is almost zero and no current is driven either to BP or to BM. However, when the transmitter is enabled (TxEN set low), BP/BM voltages go to different directions, generating a differential voltage on the bus. During Data_0 state uBus is negative and during Data_1 uBus is positive. The simulation result shows that the behavioral model correctly represents all the states and passes the test.

In addition to the protocol specifications, the model supports some additional features.

Table 1: CAN transceiver model architectures characteristics.

Description	Fast model	Moderate model	Accurate model
Output drivers	Piecewise linear	Taylor series	Transconductor
EME control	No	No	Yes
Internal voltage reference	Piecewise linear	Exponential	Exponential
Power consumption estimation	No	Yes	Yes
Thermal shutdown	No	Yes	Yes

(i) Statistical Analysis: corner and Monte Carlo analyses allow the user to run simulations where the devices are exposed to different environment conditions. The influence of each device on the entire network, in worst case scenarios, can be evaluated even during the network design stage and before any prototype being developed.

(ii) Fault diagnosis: the detection of network faults is implemented by the behavioral model. The model is able to detect and signal bus line short circuits and power supplies failures.

(iii) Monitoring system: all the power supplies as well as the voltages at all input/output pins are constantly monitored. If any voltage is out of the defined valid range, error or warning messages are generated.

Figure 9 shows an example of fault diagnosis, where the power supply voltage (Vcc) failure is simulated. The simulation starts with Vcc at typical value (5.0 V), and then Vcc goes to unpowered (0.0 V) and returns to typical value (Figure 9(b)).

The pass/failure criterion defines that, if Vcc undervoltage is detected, the undervoltage monitoring flag *porb_vcc* must signals the failure and the bus drivers should autonomously switch to low-power mode; that is, no current must be driven either to BP or to BM and the bus drivers must bias both, BP and BM, to GND level.

While Vcc is at typical value, BP and BM bus lines represent the bus state defined by the input control pins (STBN, TxEN, TxD), that, in the simulation case, is Data_0 (Figure 9(a)). When Vcc goes to unpowered, BP and BM go to Idle_LP state and *porb_vcc* signals the Vcc power supply failure (Figure 9(c)). The transceiver remains in Idle_LP state until no power supply failure is detected. The model presents the correct behavior during power supply failure and also the correct power supply failure detection and signal (*porb_vcc*).

When statistical analysis is implemented, it is also necessary to verify the model behavior in the corner cases. Figure 10 shows the bus signal integrity analysis not just for the typical behavior but also for the corner cases (best and worst cases). The corner cases parameters must be provided by the silicon manufacturer (foundry).

The simulation results demonstrate that the model does not present any signal integrity problem (neither in the corner cases).

3.2. CAN Bus Transceiver. The mixed-mode behavioral model of CAN bus transceiver is fully compatible with the ISO CAN standard and with the NCV7341 transceiver [17].

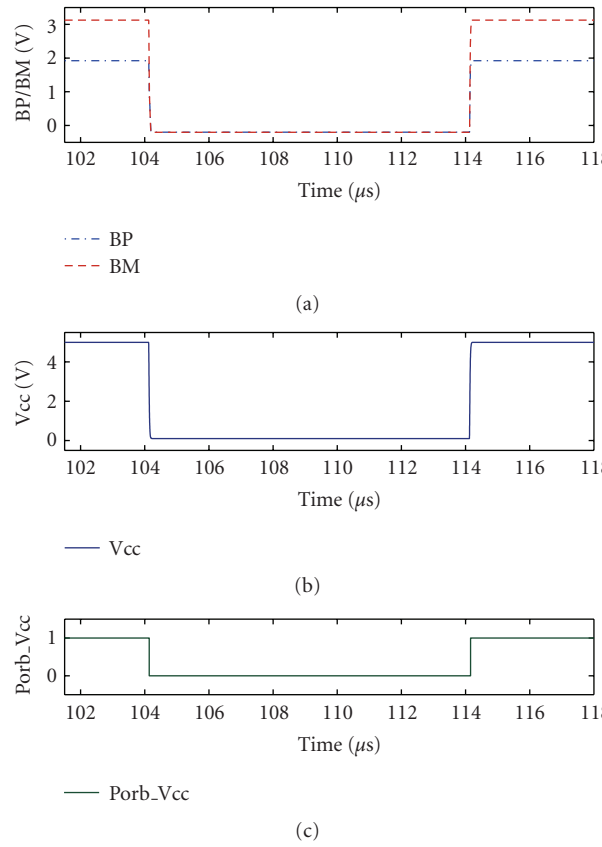

Figure 9: Power supply voltage Vcc failure simulation.

In this case study, we show how the tradeoff between accuracy and speed was managed. Three different CAN transceiver model architectures were implemented. Each architecture uses different primitive elements descriptions, which directly affect the model accuracy and speed. On the other hand, some model architectures implement the thermal protection, the instantaneous power consumption estimation based on the operation modes, and the electromagnetic emission (EME) control. The model architecture characteristics are presented in Table 1. Further information about the implemented architectures can be found in [18].

The performance is evaluated comparing the CPU usage time of the three architectures. Some simulation results are presented. Table 2 gives a brief description of the performed tests and Table 3 shows the CPU usage time in seconds. The

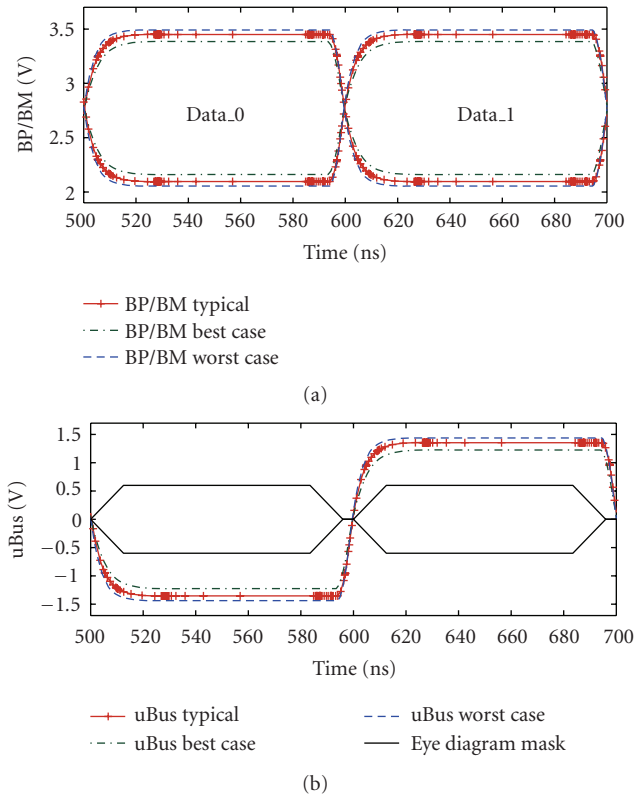

Figure 10: BP and BM bus lines signal integrity simulation.

Table 2: Test cases description.

Test no.	Description	Simulation period
Test 1	TxD dominant clamp timeout	1.1 ms
Test 2	Vio undervoltage timeout	12.3 ms
Test 3	Loop delay TxD-CANBus_RxD	14.0 μs
Test 4	Hoping states	133.1 μs
Test 5	Short circuits	210.1 μs
Test 6	Local and remote wake-up	670.1 μs
Test 7	Power-On detection	10.5 ms

PC workstation used to perform the tests presented in Table 3 is an Intel P4, 2.66 GHz, 512 MB RAM, and Linux O.S.

The results shown in Table 3 demonstrate the CPU usage time variation according to the chosen architecture. It is also possible to verify that CPU usage time variation is strongly dependent on the test case characteristics.

In addition, we present some examples of model validation for the CAN transceiver model. Model simulations in the three architectures are compared with measurements. Figure 11 shows the recessive to dominant and Figure 12 shows the dominant to recessive bus state transitions.

The measured and ACCURATE model data match, mainly regarding to bus signals voltage slope. The FAST and MODERATE model results match on voltage levels, but they exhibit steeper voltage slopes. The architecture choice must consider the user requirements.

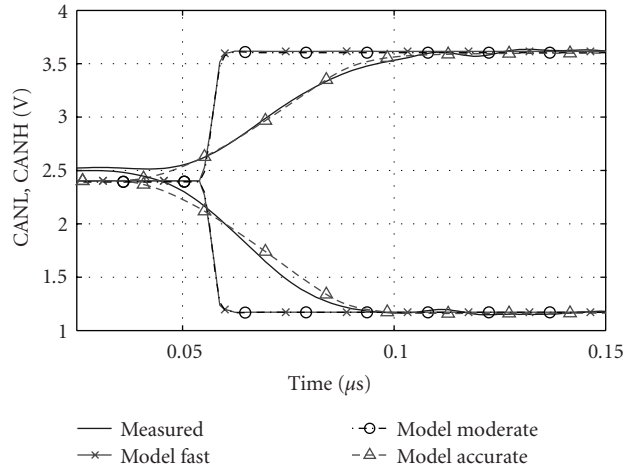

Figure 11: Recessive to dominant state transition.

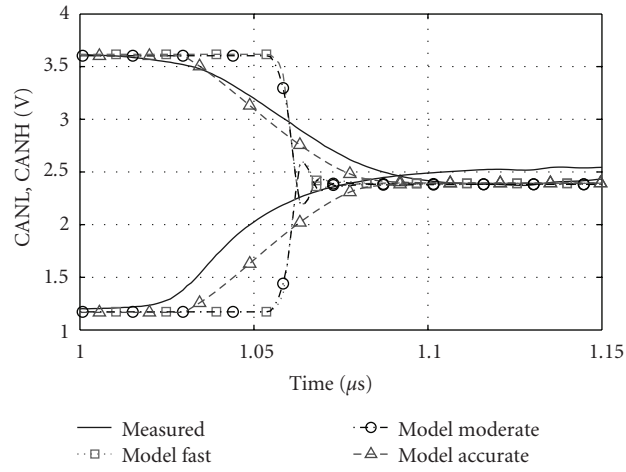

Figure 12: Dominant to recessive state transition.

Table 3: CPU usage time.

Test no.	Fast	Moderate	Accurate
Test 1	46.9 s	49.7 s	59.3 s
Test 2	329.4 s	450.0 s	534.7 s
Test 3	15.2 s	24.2 s	31.3 s
Test 4	18.8 s	28.3 s	35.0 s
Test 5	30.4 s	52.4 s	90.5 s
Test 6	31.8 s	38.0 s	46.5 s
Test 7	339.1 s	390.0 s	461.0 s

3.3. Advantages of Using Behavioral Simulation. The in-vehicle communication network is composed by the ECUs (see Figure 1) and the bus line, which connects all the ECUs of the network. Figure 13 shows an example of the communication network. Accordingly with the application requirements (bandwidth, safety, deterministic/statistic behaviors, etc.), different communication protocols can be used, as, for instance, CAN and Flexray protocols. The

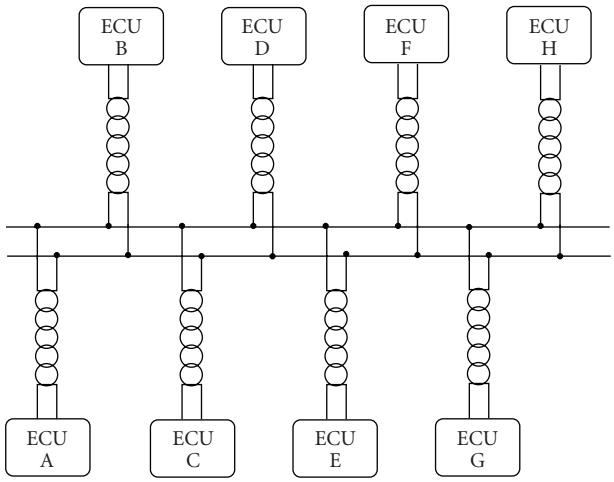

Figure 13: In-vehicle network example.

Table 4: CPU usage time.

Test case description	Behavioral model	Spectre model	Simulation period
Bus signal integrity	3.8 s	372.8 s	102 μs
Bus state transitions	3.5 s	512.2 s	400 μs
Vcc Power supply failure	2.5 s	162.9 s	120 μs
Wake-up pattern detection	7.8 s	2622.8 s	235 μs

number of ECUs can also increase since new electronic controlled features are implemented in the vehicle system.

The increasing complexity of in-vehicle communication networks requires big effort on system design verification. It is compulsory to verify if the network behavior is compliant to the protocol physical layer specification and to ensure the interoperability between the ECUs that compose the network.

The transceiver behavioral models presented in Section 3 can be used for evaluating the communication network design and also for forecasting design problems in the early stages of the design process, even before building any hardware prototype.

If all the models of the network component are available, it is possible to analyze the network behavior in real operation conditions, verifying the effects of the network reflections, timings, and so forth.

The use of behavioral simulations has advantages with respect to the use of prototypes as well to the use of transistor level simulations.

When compared with hardware prototypes, behavioral simulations represent a reduction in cost and time to market. In addition, it allows easily and quickly changing and verifying the network topology, which is very hard and costly with prototypes. Moreover, in behavioral simulations the user has the total system controllability, which allows setting, simulating, and verifying the system behavior in the worst case scenarios.

Behavioral simulations are much faster when compared with transistor level simulations. Table 4 shows the comparison of the CPU usage time between the simulations of VHDL-AMS behavioral model and Spectre model of Flexray physical layer transceiver. The first column gives a brief description of the main tests performed. The second and third columns report the CPU usage time of the simulations of behavioral model and Spectre model, respectively. The fifth column shows the simulation period set for the test case transient analysis. The PC workstation used to perform the tests is an Intel P4, 2.66 GHz, 2 GB RAM, and Linux O.S.

4. Conclusion

The paper presents a systematic development methodology for mixed-mode behavioral models of in-vehicle electronic embedded systems. The proposed methodology is divided in four different steps: model specification, model design and implementation, tradeoff, and conformance test.

Two case studies are presented: a real silicon device mixed-mode behavioral model of a CAN bus transceiver and a generic mixed-mode behavioral model of a Flexray physical layer transceiver. The paper presents how the proposed methodology was applied to the models development, reporting some simulation results.

In addition, the paper shows that the behavioral simulations present an expressive reduction of the CPU usage time if compared with Spectre transistor level simulations. Moreover, behavioral simulations are flexible and allow the fast communication network setup and verification if compared with hardware prototypes.

Future work will address the electromagnetic compatibility (EMC) analysis applied to behavioral modelling. It should focus on the electromagnetic emissions effects (caused by the fast transitions of the transmitted bus line differential signal) and the conducted immunity. A deep study of how behavioral models can reliably represent the electromagnetic effects is necessary.

References

[1] N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert, "Trends in automotive communication systems," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1204–1222, 2005.

[2] K. Current, J. F. Parker, and W. Hardaker, "On behavioral modeling of analog and mixed-signal circuits," in *Proceedings of Conference on Signals, Systems and Computers*, pp. 264–268, Pacific Grove, Calif, USA, October 1994.

[3] W. Lawrenz and D. Bollati, "Validation of in-vehicle-protocol network topologies," in *Proceedings of the 2nd International Conference on Systems (ICONS '07)*, pp. 20–24, April 2007.

[4] T. Gerke and C. Schanze, "Development and verification of in-vehicle networks in virtual environment," SAE Technical Paper Series 2006.01.0168, SAE International, Warrendale, Pa, USA, 2005.

[5] T. Gerke and D. Bollati, "Development of physical layer and signal integrity analysis of flexray design systems," SAE Technical Paper Series 2007.01.1636, SAE International, Warrendale, Pa, USA, 2007.

[6] T. Gerke and D. Bollati, "An automoted model based design flow for the desing of robust flexray networks," SAE Technical Paper Series 2008.01.1031, SAE International, Warrendale, Pa, USA, 2008.

[7] "FlexRay Communication System Electrical Physical Layer Specification," FlexRay Consortium, http://www.flexray.com/.

[8] "BOSCH—CAN Specification Version 2.0," 1991, Robert Bosch GmbH, http://www.bosch.com/.

[9] "Flexray Communications System Specifications," FlexRay Consortium, http://www.flexray.com/.

[10] W. Prodanov, M. Valle, R. Buzas, and H. Pierscinski, "Behavioral models of basic mixed-mode circuits: practical issues and application," in *Proceedings of the European Conference on Circuit Theory and Design (ECCTD '07)*, vol. 1, pp. 854–857, Seville, Spain, August 2007.

[11] J. Chew and C. Sullivan, "Verification, validation, and accreditation in the life cycle of models and simulations," in *Proceedings of the Winter Simulation Conference*, vol. 1, pp. 813–818, Orlando, Fla,USA, December 2000.

[12] P. J. Ashenden, G. D. Peterson, and D. A. Teegarden, *The System Designer's Guide to VHDL-AMS*, Morgan Kaufmann, San Francisco, Calif, USA, 2003.

[13] R. C. Jaeger, *Microelectronic Circuit Design*, McGraw-Hill, New York, NY, USA, 1997.

[14] D. Caughlin, "An integrated approach to verification, validation, and accredition of models and simulations," in *Proceedings of the Winter Simulation Conference*, vol. 1, pp. 872–881, Orlando, Fla,USA, December 2000.

[15] "Flexray Physical Layer Conformance Test Specification Version 1.0," FlexRay Consortium, http://www.flexray.com/.

[16] C. Muller, M. Valle, R. Buzas, and A. Skoupy, "Mixed-mode behavioral model of flexray physical layer transceiver," in *Proceedings of the 19th European Conference on Circuit Theory and Design (ECCTD '09)*, pp. 527–530, Antalya, Turkey, August 2009.

[17] NCV7341, "High-Speed Low Power CAN Transceiver," 2007, ON Semiconductors Inc., http://www.onsemi.com/.

[18] W. Prodanov, M. Valle, R. Buzas, and H. Pierscinski, "A mixed-mode behavioral model of a controller-area-network bus transceiver: a case study," in *Proceedings of the IEEE International Behavioral Modeling and Simulation Workshop (BMAS '07)*, pp. 67–72, San Jose, Calif, USA, September 2007.