Analysis of the physical layer using virtual vehicle networks

Carsten Schanze, Volkswagen AG

Thorsten Gerke, Synopsys Inc.

Development and verification of in-vehicle networks include multiple design layers. These layers include the logical layer represented by the software application, the associated data link layer, and the physical connection layer containing bus interfaces, wires, and termination. Verification of these systems in the early stages of the design process (before a physical network is available for testing) has become a critical need. As a result, the need to simulate these designs at all their levels of complexity has become critically important. In-vehicle networks can be simulated on many different abstraction levels using various model levels and modeling technologies. Early in the development process, analyses can be performed without having available any detailed models from the chip manufacturer or component supplier. Later in the development process, more accurate models can be integrated into the simulation process, including those provided by suppliers and chip manufacturers. This paper demonstrates a portion of the development and verification process of the physical layer of an in-vehicle CAN Bus at Volkswagen using the Saber simulation environment. This paper also demonstrates integrating portions of the logical layer into the simulation so both logical and physical layers can be simulated together.

Introduction

Today's automotive networks are very complex and combine a couple of networks together into one heterogeneous network environment. Each sub- network might use different protocols or similar protocols with different transmission rates. Over the past 10 years CAN has become the standard in the automotive industry for applications like powertrain, comfort and infotainment. A complete CAN bus network contains the following components:

- CAN node
- Transmission line
- Termination

Usually the applications require additional circuitry in order to ensure immunity against EMC effects. As shown in Figure 1 every CAN node includes four main components:

- CAN transceiver
- CAN controller
- µ-controller
- Interface to sensors and actuators
- EMC protection circuit

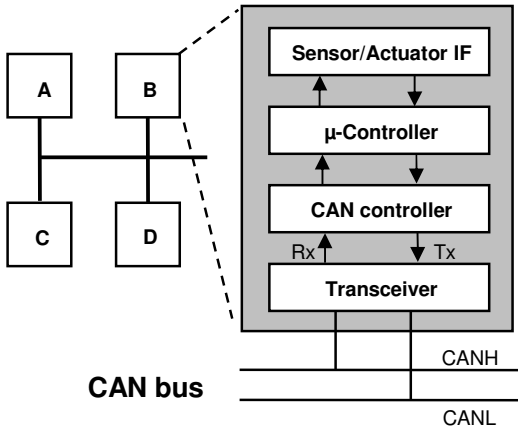


Figure 1 ***CAN node***

The physical layer of a CAN bus network contains the transmission medium (wire harness), transceiver, the physical signaling of the CAN controller and any

additional circuitry necessary to achieve acceptable EMC behavior. This article describes how to model and simulate this physical layer.

Problems when verifying the physical layer of a CAN network

The problem of verifying the physical layer of a CAN bus network is to guarantee sufficient signal integrity in order to ensure that the CAN controller always samples the correct bit value representing the current state on the bus. Unlike the digital controller signals (Tx, Rx), signals on the physical layer (CANH, CANL) are analog quantities. The integrity of these analog signals depend on several factors like network topology, network interface, transceiver etc. The topology includes the structure of the network (e.g. a star architecture or a linear bus with stubs), the number of ECUs and the length of the transmission lines. The minimum and maximum number of ECUs are fixed for the specific vehicle platform. However the number between minimum and maximum depends on the configuration of the individual vehicle. The interface between the ECUs and the transmission lines consists of passive elements like capacitors, ESD protection (e.g. varistors), termination resistors and common mode chokes. Each of these elements is affected by tolerances due to manufacturing inaccuracies or temperature dependencies. The bit timing configuration is one of the important tasks of the verification process. It specifies when the CAN controller samples the bus and determines the transmission rate in the network. It significantly affects the performance of the CAN bus network since a poorly configured bit timing can force a CAN node to go into the error passive state during an arbitration phase. Unlike the digital signals of the controller the signal behavior on the bus is analog.

The verification of the physical layer of the CAN network must ensure that the worst case combination of parameters and tolerances assures sufficient signal integrity so that each allowed bit timing configuration leads to a correct sampled value. Verification by measurements on prototype vehicles is insufficient because the worst case configuration is usually not available during the early stage of the development process and the values of the tolerances are distributed randomly. Changing an existing prototype topology would significantly increase the development costs and time. Therefore the system simulation is a fundamental requirement for the verification of this kind of application in the early stages of a design process.

Simulation models

The complete CAN network and its components were modeled and simulated in the Saber Simulation environment. Saber is an analog/mixed signal simulator that provides a complete environment for modeling, simulation and post processing in order to analyze a wide range of applications. Designs can be created very rapidly via schematic entry. Saber provides both basic analysis types like transient (time domain) and AC (small signal) analysis as well as advanced analysis methods like Monte Carlo, which is very powerful for analyzing the effects of parameter tolerances. Saber's automation methods have also been used in this example in order to automate the simulation and post processing flow. The following sections describe the models required for the simulation.

Overall simulation model of a CAN network

The overall simulation model contains five CAN nodes including all required components in order to simulate the physical behavior of the communication

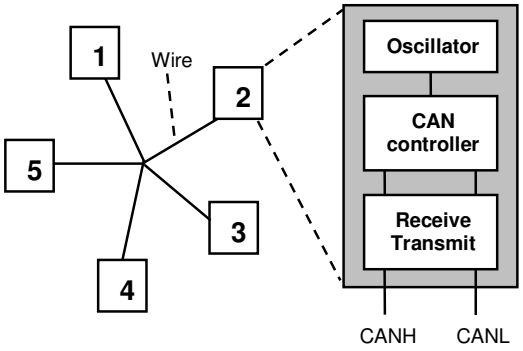


Figure 2 ***Simulation model***

system. Every node includes EMC protection circuitry, transceiver and CAN controller. All nodes are connected via physical wires modeling the behavior of a twisted pair transmission line. Figure 2 shows the architecture of the overall simulation model.

Transmission line

The transmission line is one of the critical parts in the simulation model. It must include effects like reflection and crosstalk but must also achieve good simulation speed as the overall CAN simulation model is intended to be used in analyses that are computationally intensive, e.g. Monte Carlo. Figure 3 compares the transient behavior of the simulated wire

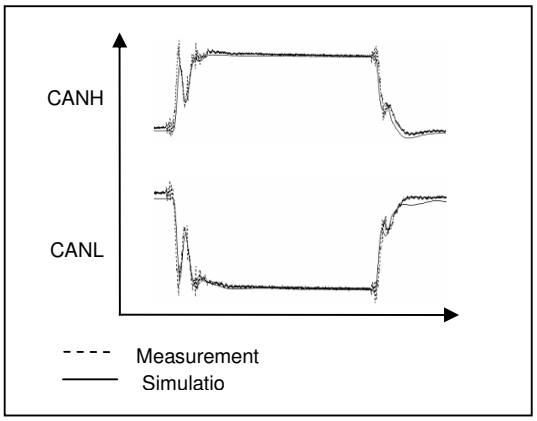


Figure 3 Measurement vs. simulation

model with experimental measurements. It can be seen that the difference between measurements and simulation is sufficiently small and the model maps very accurately the dynamic behavior of the transmission line.

Common mode choke

The common mode choke is modeled hierarchically with coupled inductors and their resonance behavior. Characterization of the model parameters is performed by measurements in the frequency and time domain. The input resistance (odd and even mode) was determined in the frequency domain by measuring the complex resistance values (magnitude and phase). Different common mode chokes

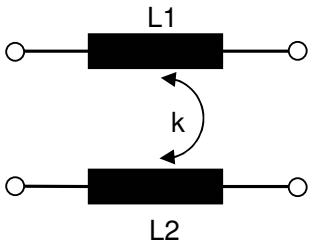


Figure 4 Common mode choke

can be modeled by characterizing this generic model using different parameter sets obtained from measurements. The model provides accurate AC and transient behavior while exhibiting excellent simulation speed.

Transceiver

The simulation model of the transceiver must be very accurate as it has a fundamental impact on the signal integrity. Therefore it is recommended to obtain this model directly from the semiconductor manufacturer. This application uses the TLE 6250 transceiver chip, a high speed CAN transceiver from Infineon. The model was especially developed for system simulation and is approximately 16 times faster than the transistor circuit model of the entire IC. Figure 5 shows the

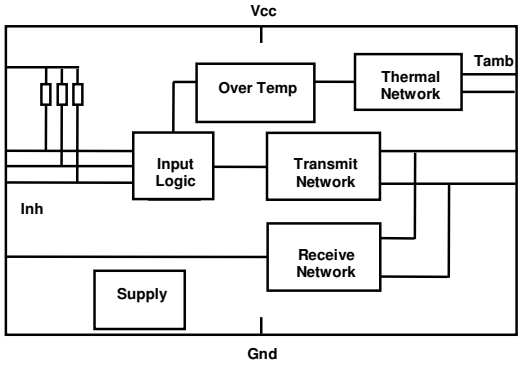


Figure 5 TLE 6250 model

hierarchical architecture of the CAN transceiver model. The model considers all required functions like over temperature, low supply voltage, short circuit and real current consumption. It covers the complete range of input and output voltages, supply voltage and

temperature limits specified in Infineon's data sheet for the TLE 6250. Applied in various test benches the simulation model has delivered both very accurate results and good simulation speed.

Simplified CAN controller

For the verification of the physical layer only a reduced set of the CAN controller's functionality is required. The simplified model contains some functions of a CAN controller according to CAN specification 2.0 and ISO 11989 and is intended to simulate the timing and acknowledgement behavior of a CAN controller. Figure 6 shows the infrastructure of the simplified CAN controller model. The model has four connection pins:

- Osc (Oscillator)
- CAN_state
- Tx
- Rx

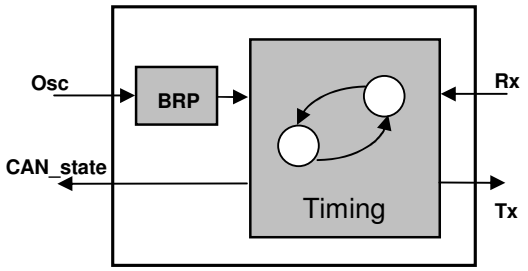


Figure 6 CAN controller model

The pin CAN_state shows the state that has been sampled by the CAN controller at its Rx pin. This determines whether the correct value has been sampled or if the bit timing configuration is incorrect. The BTR parameters (TSeg1, TSeg2, Sample mode, BRP and SJW) can be specified by the user as model arguments of the controller model. The timing behavior determines the bit time (TBit), the sample point (SP), the sample mode (single or triple) and the SJW. In single sample mode sampling proceeds between two tQ at the programmed value. In triple sample mode sampling proceeds in the previous three tQ from the programmed value. The valid value is calculated by majority vote.

One of the major functions of the timing behavior is the resynchronization as shown in Figure 7. The resynchronization occurs at the edges from recessive to dominant state when the CAN controller

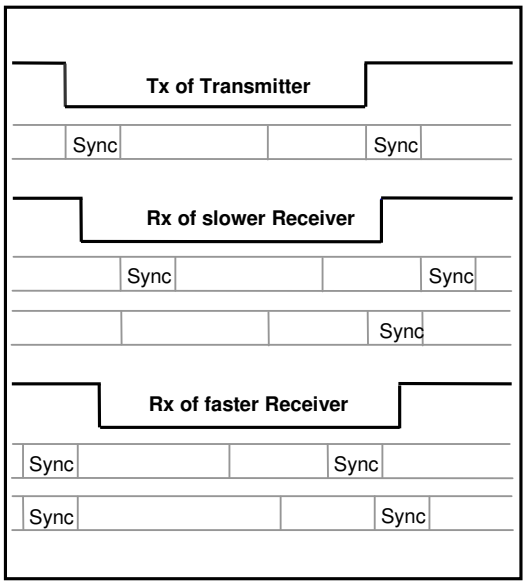


Figure 7 Resynchronization

recognizes a falling edge at its Rx pin. If the edge occurs in TSeg1 (not in the SyncSeg, but before the sample point) the receiver interprets this as a delayed edge from a slower transmitter and TSeg1 of the receiver will be extended. If the edge occurs in Tseg2 (between sample point and SyncSeg), this is interpreted as early edge of a fast transmitter and TSeg2 of the receiver will be shortened. If the phase error is smaller than SJW, the relevant segment is corrected by the value of the phase error, otherwise by SJW. In the case that the phase error is greater then the SJW, the CAN controller cannot completely resynchronize the appropriate CAN node within a single bit timing cycle. All receivers that receive a correct message acknowledge this by transmitting a dominant bit. The transmitter sends a recessive bit in the same time slot. The bits before and after the acknowledgement of the receiver are recessive. The complete algorithm of the timing and the acknowledgement behavior of the CAN controller can be modeled as state machine. StateAMS enables graphical modeling of state machines containing

analog and digital device behavior. The benefit of this technology is that it frees the user from dealing with any modeling languages as the entire behavior is described graphically in conjunction with analog equations and digital assignments. StateAMS creates the simulation model on-the-fly based on the state diagram information, making it easier for the model developer to alter and maintain complex models without hand-coding in modeling languages like MAST, VHDL-AMS or Verilog. Figure 8 shows a portion of

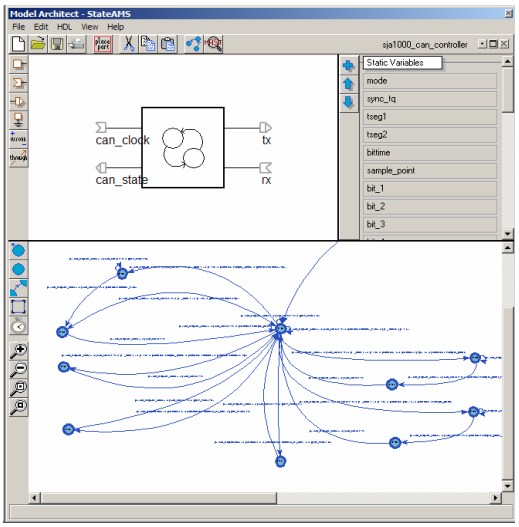


Figure 8 State machine

the CAN controller's state diagram model in StateAMS. States are represented by circles indicating the current operating point of the controller. The controller changes from one operating state to another as soon as the transition that is connecting these states becomes true.

Simulation of the CAN network

The required simulation described previously can be combined into a test bench for the purpose of verifying the behavior of an entire CAN bus network. Figure 9 shows the test bench including five CAN nodes connected together via transmission lines to form a star arrangement. Each CAN node is modeled hierarchically, consisting of the following components:

- CAN controller including baud rate prescaler and bit timing state machine
- CAN physical layer
- Oscillator

The hierarchical controller model contains

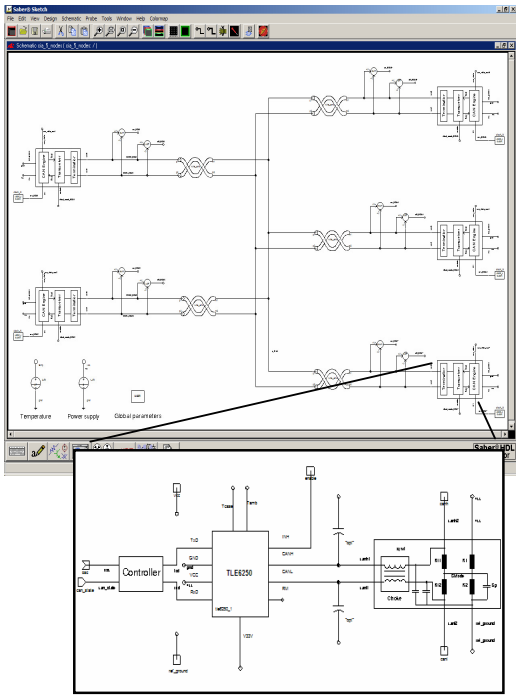


Figure 9 CAN test bench

the BRP and the BTL. The state machine of BTL has been modeled as a state diagram and the BRP has been implemented directly in MAST, Saber's modeling language. The physical layer includes the CAN transceiver, connected to the voltage regulator and the appropriate termination circuits, consisting of stabilization circuit, choke coil and additional capacitors. The oscillator is modeled as a digital pulse source that samples the CAN controller.

The first test bench example illustrates how the synchronization of the CAN nodes works. ECU1 sends a series of 15 bits to the bus. The bit stream contains a series of falling edges (recessive to dominant) in order to show the synchronization activities of the ECUs. All ECUs are supposed to send an acknowledgement bit (dominant state) after receiving a series of

15 bits. The controllers are configured as shown in Table 1.

Parameter	Value
Frequency	9 MHz
TSEG1	5 tq
TSEG2	3 tq
BRP	2
SJW	3 tq
Sample mode	Single

Table 1

The entire bit time is divided into nine time quanta. In conjunction with an oscillator frequency of 9 MHz this results in a nominal bit time of TBit=2µs and transmission baud rate of 500 kB. Figure 10 shows the simulated differential bus voltage of the CAN architecture measured at ECU5. ECU1, acting as transmitter, sends the bit stream “001010101010101” to the bus and the other ECUs

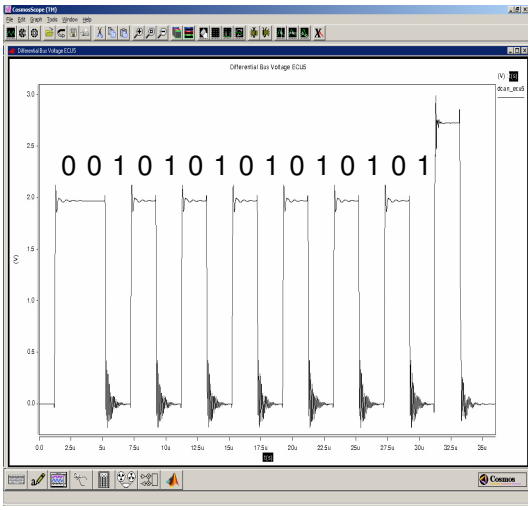
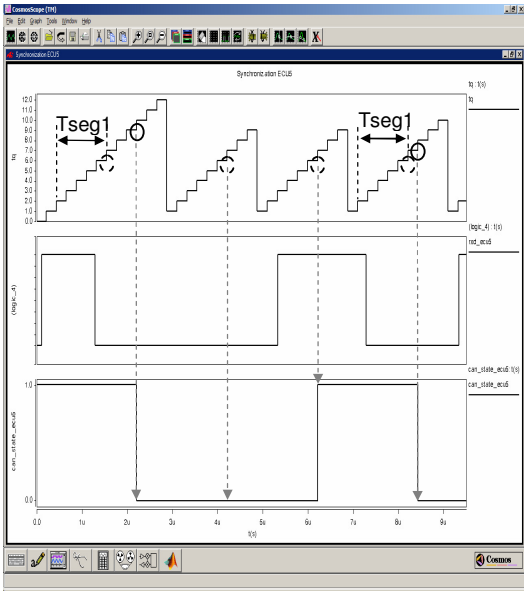


Figure 10 Differential voltage

acknowledge this bit stream by sending a dominant bit which causes an increase of the bus voltage. The simulation results also allow the designer to analyze and verify the bit timing of the ECUs. Figure 11 illustrates the bit timing of ECU5. The upper signal “tq” indicates the time quantum of the bit time where the CAN controller is currently working. The middle



Default SP

Shifted SP

Figure 11 Sampling & timing

signal is the receive signal “rxd” which is measured at the transceiver’s digital receive pin connected to the CAN controller. It shows the bus status being detected by the receive network of the bus driver. The lower signal “can_state_ecu5” reflects the received bit sampled by the CAN controller at its receive pin connected to the transceiver output “rxd”. These signals allow the designer to track the bit that is sampled on the bus and follow the synchronization of the protocol engine during a message frame. According to Figure 19, the controller detects a falling edge in TSeg1 (tq=5) which means it has to resynchronize as the falling edge has been detected outside of the SyncSeg. As the edge falls into TSeg1, TSeg1 is extended and the sample point moves automatically to the right on the time axis and the entire bit time is extended as well.

In this example the phase error is equal to 4 t_q. Due to the fact that the synchronization jump width limits the maximum synchronization step to 3 t_q, the CAN controller cannot completely compensate the phase error and a phase error of 1 t_q remains. The next two bits do not contain a falling edge, meaning that the CAN controller has no opportunity during this time to synchronize the CAN

node. The next synchronization happens when the fourth bit is being sent. During this time the falling edge is detected in t_q=2 causing a phase error equal to 1 t_q. In this case the controller can compensate the phase error completely and the CAN node is completely resynchronized. For this scenario the correct bit stream has been sampled by the CAN controller, however the frequency tolerances of the oscillators have not been taken into account. The following test bench applies the same CAN network architecture with two modifications. The oscillators of ECU1 and ECU5 are modeled with a tolerance of 2% that might be caused by manufacturing variations or temperature dependency. The oscillator of ECU1 is assigned a frequency of 9.18 MHz and the oscillator of ECU2 has a frequency of 8.82 MHz. Another situation that must be considered occurs when a series of five dominant and five recessive bits (0000011111) is being sent. This specific bit stream does not contain a falling edge that allows the controller to resynchronize. A series of five equal bits in a row is the maximum number as the controller automatically inserts a stuff bit after five identical bits. This bit has the inverted value of the previous bit in order to ensure that the controller has the chance to resynchronize during a message frame. Figure 12 shows

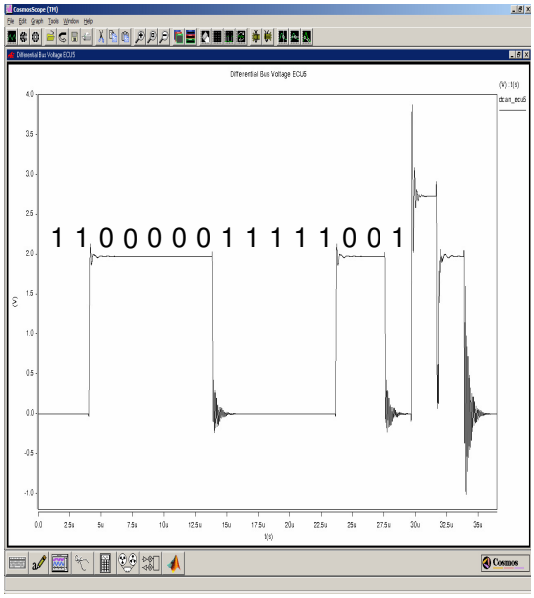


Figure 12 Acknowledgement

the result of this scenario. The transmitter sends the bit stream “110000011111001”. The differential bus voltage of ECU5 shows an additional dominant status of the bus after the acknowledgement bit. This is not allowed as the bits prior and after the acknowledgment must be recessive. This indicates there is something going wrong in the system which seems to be related to the timing of the ECUs. Taking a deeper look into the timing of ECU5, shown in Figure 13, confirms this suspicion. The first falling edge of the bit stream sent by ECU1 is detected in the controllers SyncSeg of ECU5. This means that no

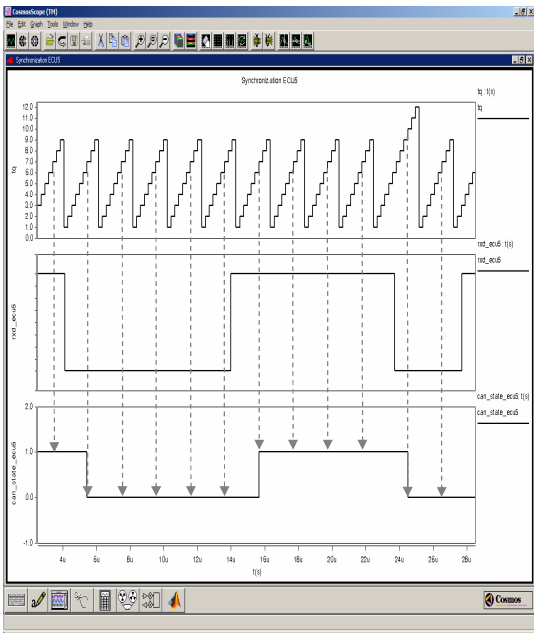
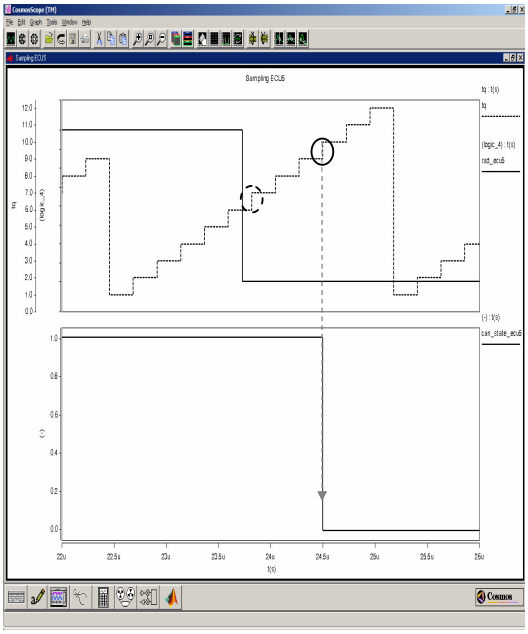


Figure 13 Timing

synchronization activities are required as the receiver is in phase with the transmitter. During the next ten bits no synchronization happens as there is no further falling edge which can be applied by the controller in order to synchronize the appropriate node. The controller correctly samples five dominant bits after the falling edge. When the bus status changes from dominant to recessive, the controller detects four recessive bits and then a dominant bit. Even though the transmitter has sent five recessive bits, the receiver detects only four of them. The reason for this is the situation occurring after the detection of the fourth recessive bit, since the falling edge after the fifth

dominant bit occurs prior to the sample point for this bit. This means that the controller drifted out of the synchronized mode as no_further synchronization was possible, which is required to compensate the oscillator tolerances and the asynchronous behavior of the oscillators. Figure 14 shows that the falling edge after the fifth recessive bit is detected in the sixth time quantum right before the sample point. The result is that the controller samples a dominant bit instead of a recessive bit, resulting in a bit error. Due to the fact that the falling edge is detected in t_q=6, the controller extends TSeg1 and shifts the sample point to the right on the



Default SP

Shifted SP

Figure 14 Sampling error

time axis about the maximum synchronization jump width. This shift also explains the additional dominant bit after the acknowledgement, as the bit timing of ECU5 contains an additional phase error of approximately one complete bit time. This example clearly shows that detailed analysis is possible when simulating the physical layer of a communication system like CAN. Discovering a problem is only the first step during the verification of these communication architectures. In order to solve the problem properly a deeper look into the system is required.

CONCLUSION

This paper demonstrates simulation of the physical layer of a Volkswagen CAN bus network using the Saber simulation environment. It emphasizes the importance of simulation of this type of system at an early stage in the design process in order to reduce the number of prototypes. Simulation allows the analysis of different vehicle network architectures without having the hardware or the real vehicle network. Required changes are detected before the network architecture will be delivered to manufacturing and the simulation covers all tolerance limits and random variants. Beside early problem detection, a deeper understanding of the appropriate architecture is guaranteed and it is easier to analyze targeted network variants. This makes the network developer more flexible and allows faster addressing of application-specific problems. In addition to the examples shown in this article, further applications are important candidates for future investigation. The analysis of heterogeneous network architectures, e.g. a combination of high speed CAN and low speed CAN is important, in order to simulate latencies of message frames. The methodologies outlined in this paper can be used for other in-vehicle network technologies like LIN and FlexRay and will offer similar benefits.

Carsten Schanze
Volkswagen AG
38436 Wolfsburg (Germany)
+ 49 (0) 5361 9-44218
carsten.schanze@volkswagen.de
http://www.volkswagen.de

Thorsten Gerke
Synopsys Inc.
Karl Hammer Schmidt Straße 34
85609 Aschheim/Dornach (Germany)
+49 (0) 89 99320-227
thorsten.gerke@synopsys.com
http://www.synopsys.com/saber