High Level Performance Simulation of a quadruple CAN Gateway

Florian Bogenberger, Ulf Warschat

ABSTRACT

This paper shows methodologies used to develop a high level simulation model of a quadruple CAN gateway. The project was done in tight cooperation between Audi and Motorola. The gateway model was integrated and simulated in the complete automotive network architecture together with models of all ECUs. Different simulation scenarios were run in order to measure CPU load, CAN bus load, message latencies in the gateway, memory requirements, etc. The modeling approach is described and illustrated with practical examples of the project. It is shown which results can be achieved, what there quality is and how they can contribute to the system development process. This paper shows and correlates practical experience of virtual prototyping with expectations frequently outlined in theoretical proposals.

1. Introduction

Many features of a today's car are controlled by distributed systems with components communicating via different busses. One of the most commonly used is the CAN protocol. For different applications in the car different networks have been growing over time. With increasing features in the car there is a growing need for gateways connecting these major networks. There is a significant dependence of the overall system on a stable and reliable gateway implementation. This paper describes a study done in close cooperation between Audi and Motorola on a quadruple CAN gateway.

2. The Application

The target application is a central automotive gateway which links the following networks:

- Body
- Powertrain
- Infotainment
- Dashboard
- Diagnostic

The following section describes some characteristics and differences between the networks.

2.1. Characteristics of the Networks

The body network contains things like door modules, seat control, sliding roof, etc. This network may connect many ECUs (Electronic Control Units) and a lot of the information they exchange is event-based. The data traffic is medium, so a data rate up to 125kBit (low speed CAN [6]) is usually sufficient. As the data traffic depends a lot on the activity of the passengers, the actual communication load can vary in wide boundaries above 50%.

In contrast to that the powertrain communication is quite stable. Due to real-time requirements in this area there is large amount of state-based communication with messages sent repetitively in short periods. These periods are calculated to serve the fastest possible reaction time. As a dominating number of messages is transmitted all the time, the data rate remains relatively constant.

So overload situations are prevented which could delay important messages. Due to that approach powertrain usually uses high-speed CAN[7].

Again a completely different area is infotainment. This area is evolving quickly and will certainly undergo big changes in the future. It includes things like CD and radio control, telematics, navigation, etc. Data transmitted in this network comprise event-based control information, but also graphic data, video and audio data. So at least in high-end cars large amounts of data will be transmitted, certainly exceeding the capabilities of CAN. Many carmakers are in favor of the MOST (Media Oriented Systems Transport) bus [5], an optical bus with a data rate up to 22.5 MBit/s. Nevertheless this network is not completely self contained and requires a connection to other networks. For example navigation needs wheel speed data from the powertrain network. More and more data go from infotainment into other networks, e.g. for applications like predictive navigation or adaptive light control. This is where the gateway comes into the game. Moreover for low-cost cars a MOST solution might be too expensive, so CAN could be used to carry control information.

The remaining two connection points, dashboard and diagnostic, are not necessarily multi-node networks but more or less point-to-point connections to the gateway. However both require communicate with all other networks. The dashboard displays all kind of system information, so it is primarily a data sink.

Diagnostic has traditionally been done via K-line, but with the new ISO15765 standard diagnostic will be done via CAN. Every ECU built in the car must be somehow reachable, so the gateway is an ideal entry point. Please note that diagnosis must work during normal operation of the car, so additional data traffic comes to the normal communication. Obviously this constitutes a worst-case condition for the gateway load.

2.2. Possible Architectures

Even though the networks surrounding the gateway were clear at the beginning of the project, there were some options on how to connect them. A star topology is a obvious solution, but it is not necessarily the best one. The following options were discussed at the beginning of the project:

- Is a separate connection required for the dashboard, or can it be plugged to the body, infotainment or powertrain network?
- If the dashboard requires a separate communication line, what is the best protocol - CAN, SCI, SPI, IIC or a customized parallel connection (all of which can be used for a point-to-point connection)?
- Is a dedicated connection required for diagnostic or can it be plugged to any of the existing networks?

2.3. Event-based versus State-based

Depending on their content messages can be classified in state-based and event-based. A state-based message says “the current state is abc”. The message contents does not change as long as the state remains the same. An event-based message says “now the state changed to xyz”. Obviously the later can only be transmitted when the state actually changes, whereas the first one can be repeated over and over. Actually this is a major difference, where both approaches have advantages and disadvantages. Event-based messages allow to detect the time of the state change more accurately and the data traffic is kept low unless there are many state changes. State-based messages are safer in case messages are lost, but the data rate is usually quite high because it must fulfill the shortest reaction time requirement. On the other hand the data traffic is constant, so that reaction times can be guaranteed independent on the system’s state.

Of course a mixture of both types is possible. Whereas state-based are preferred in the powertrain environment, event-based is dominating the body environment.

2.4. Message Types

Apart from the classification mentioned above the following message types have to be supported:

- cyclic messages: This is the simplest type - the message is sent periodically with a fixed period. Please note that even when a state change happens in the middle of the cycle period, no new message is transmitted.
- immediate messages: Data of this type is so important that it is not possible to wait for completion of the message cycle - instead it is transmitted immediately. Consequently it has to be processed as fast as possible by the gateway
- BAF messages: Data that is transmitted only under certain conditions, e.g. if somebody opens a window. If that condition is fulfilled, the message is sent repetitively with a fixed transmission period. After the condition has become false, it is repeated for several times and then transmission is stopped.
- TP (Transport Protocol) channel messages: The CAN protocol restricts the number of data bytes in one message to a maximum of eight. For data blocks exceeding this limit the transportation layer generates a bundle of CAN messages each containing up to eight data bytes. Messages of such a bundle are transmitted right following each other with a a short cycle period. Another characteristic is that the transmitter and receiver node negotiate an appropriate CAN identifier, which leads to a large number of valid CAN identifiers.
- diagnostic messages: Transmitted only in diagnostic mode - please note that these messages can additionally be either cyclic or TP channel

2.5. Message Operations

The major task for the gateway is to forward information from one network to another one. The simplest operation is to route messages without further processing them. In most cases, though, this is not sufficient:

- all messages received by the gateway have to be checked for a valid CAN identifier. Many invalid messages are already filtered out by the CAN controller's filters (see 4.1), but an additional software check is performed for every message passing the filters.
- the importance of messages is different in different networks - to accomodate this the gateway can assign new CAN identifiers
- periodic transmission of state-messages - this must be done in a way that (temporal) CAN bus overloading is prevented (see 4.2)
- for state-messages adjust different transmission periods in different networks, e.g. messages transmitted with a 7ms period in powertrain could have 100ms period in body.
- check timeout conditions for messages to be transmitted - either set a data bit indicating this condition or restrain the message. This is important for state-based messages repetitively transmitted by the gateway to indicate that the updating source message was not received in time and therefore data may be out-of-date.
- compose new messages from several source messages - this includes moving and assembling bit- and byte fields. Furthermore a composed message might be locked (prevented from transmission) until all fields have been updated with new data
- dynamically allocate and manage buffers for TP channel messages. As the gateway does not have substantial memory to buffer large quantities of data, the goal is always to forward messages as fast as possible. However sometimes the target network is temporarily overload. Especially TP channel messages are likely to suffer from significant throughput problems in such a situation, so they must be buffered. The large number of possible TP channels do not allow to reserve fixed buffers upfront, therefore an efficient, dynamic memory management is required.

- dynamic routing for TP channel messages - in contrast to other messages these do not have an a priori target network - instead it must be deducted from the transport protocol

Please note that this is only an excerpt of the most important operations that dominate the gateways performance.

2.6. Gateway Requirements

The gateway has to perform all operations mentioned above in a timely manner. Especially immediate message must be processed with highest priority - ideally they pass the gateway in less than a millisecond.

Moreover no messages may be lost (e.g. due to temporarily blocked interrupts). Please note, however, that this is inevitable if the target network is overloaded for too long because the gateway has a limited buffer capacity. Higher levels of the transport protocol can compensate in this case.

The gateway must transmit a large number of messages with fixed but different periods. Special care must be taken not to cause temporary overload situations by trying to transmit a bunch of messages at the same time (see 4.2).

3. Simulating the System

3.1. General

The described application was modeled as Virtual Prototype with an event-driven simulator. For simulating the CAN protocol we used a methodology presented at ICC'98 for accurate simulation on message level [2]. ECU's were modeled as abstract message generators, which transmit messages with real CAN identifiers and number of data bytes in realistic repetiton rate. They do receive messages and obey the CAN protocol, but do not process the data they receive. Parameters control jitter and pseudo random processes (e.g. to trigger BAF messages). The gateway hard- and software was implemented as mixture of graphic models (using the simulators predefined blocks) and dedicated C++ modules that were linked into the simulator.

The ECU models constitute kind of a responsive environment for the gateway model. Please note that it would not have been possible to record and playback CAN messages of existing systems as messages coming from the gateway compete with those coming from ECUs.

3.2. Simulation Parameters

Many characteristics of the simulation can be controlled by global parameters in order to emulate best- and worst-case conditions for the system. The most important parameters are:

BAF_probability: Likelyness that a BAF functions becomes active in one message cycle. In the simulation every cycle a random generator is triggered giving a 'yes' or 'no' to activate a BAF-message.

jitter_factor: Controls how accurate the message cycle is simulated. In reality the cycle time of periodic messages is never exactly identical with the specified value but varies continuously. This phenomena is controlled by this factor and immitated using a random function.

immediate_factor: For each node that transmits immediate messages transmission events are triggered with a random period.

diagnostic_mode: disable or enable diagnostic messages

clock_frequency: frequency of the CPU clock

transportation_msg_period: Controls the time span between two consecutive TP channel messages.

global_seed: seed for all random functions in the simulator - this is important to guarantee for 100% reproducibility of simulations.

Playing with these parameter almost every system condition can be immitated.

4. Additional Tools

4.1. CAN Filter Optimization

Most CAN controllers provide filters, that can be programmed to match a certain set of identifiers. Whenever the controller receives a CAN message it checks its identifier. In case it does not match the message is discarded. This prevents the CPU from wasting time to sort out these messages. For the gateway application there are messages in each network that are not forwarded to any other network. There is not need for the gateway to process these, so they should be filtered out.

Ideally filters can be programmed in a way that only messages to be processed by the gateway pass. How filtering works in most CAN controllers is that for each identifier bit it can be defined if it should recessive, dominant or any value (don't care). Having a limited number of these filters, it is quite difficult to program them in the ideal way. Practically this optimzation problem is quite difficult to be solved manually.

Therefore Motorola has developed a tool that performs this task. It takes two lists of CAN identifiers, those that should pass the fil and those that should not, and generates an optimized filter set for it. Originally this tool was developed only for the gateway project, but due its general *usability* it is now available as standalone tool for general filter optimization for the MSCAN controller.

Please note a FullCAN controller is not necessarily a way around the filter optimization problem. FullCAN controllers provide a set of buffers, each of which is reserved for a specific CAN identifier. Nevertheless for the gateway application the number of CAN identifiers is quite big, which requires a lot of memory in the controller.

4.2. A Priori Transmission Schedule

The gateway has to transmit a lot of state-based messages in a periodical manner. The duration of these periods ranges from a few milliseconds to seconds. In ECU's this is realized by programming hardware timers that trigger interrupt service routines with the appropriate period. The ad-hoc approach would be to use the same method for the gateway. A high frequency timer interrupt would be triggered periodically which would then check each message and transmit it as soon as its period has been reached *(abgelaufen)*. A drawback of this approach is that it can happen that many messages are transmitted at the same time, which would then lead to a peak load of the CAN bus. These peak load situations can significantly impact the system's timing behavior, therefore they should be avoided.

For the gateway application a special tool was developed, that generates static transmission schedule in a way that transmission points are equally distributed. Moreover the schedule is optimized to avoid accumulations of high or low priority messages. The result is a well predictable, constant CAN bus load through the gateway messages. Even though

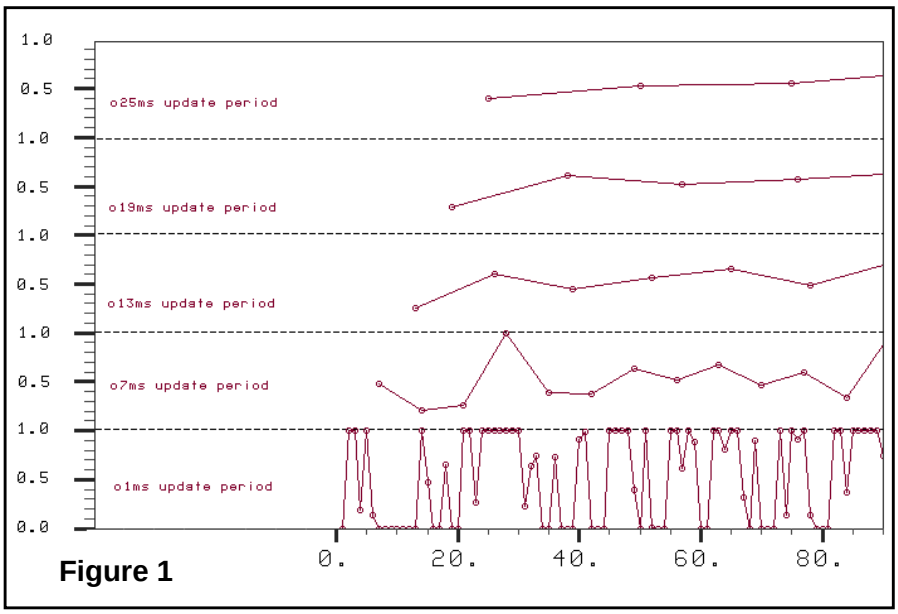

Figure 1

there is a certain interferance with messages coming from ECUs a constant load from the gateway helps to minimize peak load situations.

5. Measurements & Results

This section gives a brief overview on the type of results achieved by simulation. Please note that example diagrams given here do not necessarily represent a typical situation. Instead diagrams are taken from a random scenario which can be best-case, worst-case or even completely unrealistic. The intention of this section is to demonstrate the type and quality of those results.

5.1. CAN Bus Load

The goal was to measure the CAN bus load continuously in order to identify peak loads and their duration. To get an index of the load the following formula was used:

$$load(t, \Delta t) = \frac{\sum_{t-\Delta t \le TransmStart < t} MessageDuration}{\Delta t}$$

Obviously this depends on the time span Δt during which the load is measured (see figure 1). If this is smaller than the duration of a normal CAN message, the load-curve will jump between 1 and 0. For our measurements we used Δt=10ms. One message takes about 1ms at a datarate of 125kBit, so a load of 1 means that 10 consecutive message follow each other. Figure 2 shows an example for a CAN bus load diagram.

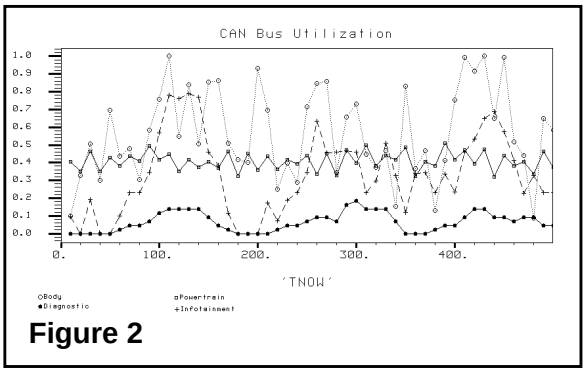

Figure 2

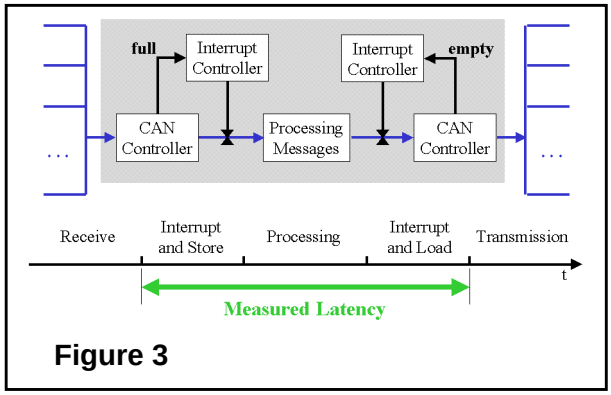

Figure 3

5.2. Filter Efficiency

Under “filter efficiency” we understand the ratio of suppressed messages to number of all unwanted messages (which are not processed by the gateway) - ideally this is 100%. The filter efficiency obviously depends on the number of messages in a network and their CAN identifiers. Due to the large number of ECUs in the Body network, the filter efficiency is lowest. Our results range in the area between 60-70% suppressed messages of all unwanted messages. For all other networks we achieved a suppression ratio of 90-100%.

5.3. Message Latency

On of the most important questions was how long the CPU takes to process high priority messages. Figure 3 is showing what the measurement includes. Please note that the time spent in the CAN controller is not included in the measurement, because this is not affected by the CPU but only by the CAN bus load. Figure 4 shows an example result. This diagram shows that the average message latency is around 230µs, and maximal values range at about 1.6ms. We did not expect variations of this size, and actually investigations showed, that the reason is not necessarily an overloaded CPU. Instead this is a feedback effect from a temporarily highly loaded target network. In that case it is likely that CAN messages cannot be transmitted by the gateway (loosing arbitration), while the followup message has already been received from the source network. The CPU then continuously tries to process the received message, finds that its predecessor has not yet been transmitted on the target network. As

these are high priority messages which should not be handled after normal messages the CPU gets into kind of a polling mode - which in turn exceeds the CPU load. The increased CPU load then affects as well messages with other target networks. This example shows the interaction between gateway and the networks and how it can affect the communication.

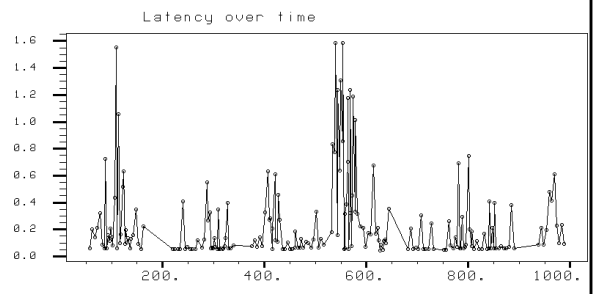

Figure 4

6. Summary

The given paper describes a application with a quadruple CAN gateway, which was simulated in a realistic environment of ECUs. The project was done in close cooperation between Audi and Motorola. The simulation approach is outlined, together with a description of global parameters used to test the model in different scenarios. Examples of measurements and results are given.

7. References

[1] Uwe Kiencke, Dirk John, Sandra Schneider, “Performance analysis of a distributed automotive real-time system”, ICC’97 Proceedings, page 07-02

[2] Florian Bogenberger, Carsten Mielenz, “Accurate Message Level CAN Simulation”, ICC’98 Proceedings, page 05-08

[3] Jan Krellner, A. J. Pohlmeyer, “Virtual Prototyping of a fault tolerant CAN physical layer transceiver”, ICC’98 Proceedings, page 02-02

[4] Simonot-Lion, Y.Q. Song, J. Raymond, “Validating real-time applications distributed over CAN: an interoperability verification”, ICC’97 Proceedings, page 07-09

[5] The Hansen Report on Automotive Electronics, Vol. 11, No. 5, June 1998, page 1

[6] ISO Standard: Low speed controller area network (CAN), ISO/DIS 11519-1

[7] ISO Standard: Road vehicles - Interchange of digital information - Controller are network (CAN) for high-speed communication, ISO/DIS 11898

Florian Bogenberger
Motorola GmbH
Schatzbogen 7
81829 Munich, Germany
Phone: +49-89-92103-421
Fax: +49-89-92103-820
Email: Florian.Bogenberger@motorola.com

Ulf Warschat
Audi AG
D-85045 Ingolstadt
Phone: +49-841-89-89414
Fax: +49-841-89-90483
Email: ulf.warschat@audi.de