

Exploration and Optimization of Gated Automotive Networks using Scheduling Analysis

Kai Richter and Marek Jersak
Symtavision GmbH

Copyright © 2008 SAE International

ABSTRACT

Today, gated networks with several buses are becoming standard in automotive E/E-systems but are evolving differently among the various vehicle manufactures, with different topologies, combinations of bus protocols, and speeds. Making the right architecture decisions requires systematic evaluation of the many alternatives during early design stages. However, there are many trade-offs in terms of performance, cost, extensibility, etc.. In this context, scheduling analysis is a powerful tool. It clarifies performance, end-to-end timing, and dynamic behavior. This enables evaluation of networking alternatives, foresight of bottlenecks, and provides guidance in the design process. In the paper, the application of scheduling analysis in automotive network exploration and optimization will be demonstrated. Specific emphasize will be put on end-to-end timing, migration from CAN to FlexRay, black-box integration and early-stage assumptions, extensibility, and trade-offs.

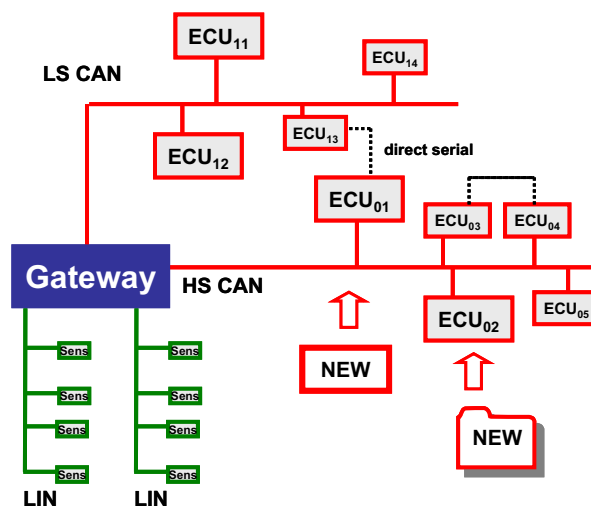


Figure 1 Typical Gated Network of Today

INTRODUCTION

Car manufacturers increasingly differentiate themselves from one another through a variety of novel electronic functions. New safety functions in the chassis domain, new environment-saving features in the engine domain, multi-media functions, body electronics, and driver assistance are only few examples that might make the difference for one or the other customer.

Each of these new functions has its own performance requirements in term of processing power, communication bandwidth or memory storage. As a consequence, more and more ECUs and buses are added in today's cars to enable innovation. The step-by-step evolution has lead to heterogeneous architectures with multiple bus speeds and protocols (CAN, Lin ...) as well as a variety of ECU types from different suppliers. A typical architecture of an upper mid-class car today is depicted in Figure 1. Premium-segment cars contain up to one hundred ECUs and more than 10 buses.

One big challenge for car manufacturers is answering the question if a new ECU or function (incl. the communication) can be reliably added without overloading the entire system, and (if yes) where and how to deploy it, and (if no) what else to do. Timing and performance play a key role but are difficult to predict because the dynamic interactions in such heterogeneous architectures result in a variety of non-functional performance dependencies due to scheduling, arbitration, blocking, buffering etc, especially when considered in end-to-end timing chains. The scheduling in particular leads to hard-to-find timing problems, incl. transient overload, buffer under- and over-flows, and missed deadlines; as well on buses as on ECUs and the central gateway(s). Therefore, a key question is: How can such timing and performance problems be reliably detected and eliminated, as early as possible in the design-process?

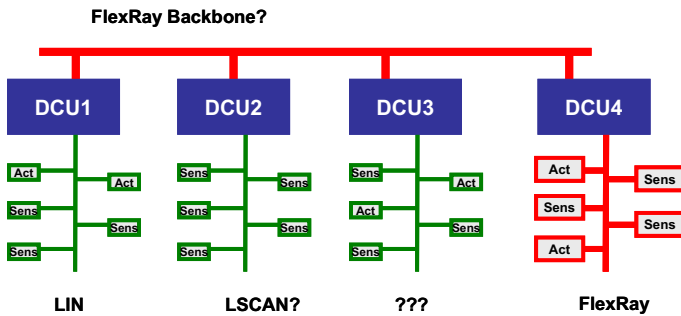


Figure 2 One Possible Future: Regular Domain Control Architectures

FlexRay is often proposed as one way to reduce (or even eliminate) timing problems due to its deterministic, predictable frame timing. However, frames are only containers for the signals that realize the actual communication. Signals exhibit a much more complex timing. Specifically the interactions of ECU COM layer software (between the application software and the lower-level protocol drivers) can be quite complex.

Furthermore, current trends and recent developments in the automotive industry indicate that a full switch to FlexRay will not be taken over-night. Quite to the contrary, FlexRay will also be deployed step-by-step. As a result, we can expect as well synchronous as asynchronous ECUs attached to one FlexRay bus in the future, increasing the heterogeneity further.

Other visions are already around to define completely new architectures. One idea is to replace the large number of ECUs realizing individual vehicle functions by a small number of powerful domain control units (DCUs). The DCUs are connected through a backbone network and have local links to a range of very small and cheap sensor and actuator ECUs (see Figure 2). Such a step may simplify bus-communication by keeping more communication local (inside the DCUs). However, in this process, the integration problem will shift to more complex software-component integration. Additionally, new challenges will come with such high-performance DCUs which will require multi-core chips that behave like just another distributed system. Therefore, timing and performance will remain important to analyze for any architecture, no matter how it looks like.

Key timing-related design questions include:

- Is any bus or network or ECU or gateway (temporarily) overloaded?
- Which frames can get lost, and how often?
- Can more ECUs (or sensors / actuators) and functions (and how many) be connected without overloading the ECU and/or bus?

- How to integrate legacy ECUs and software most effectively?
- How about diagnosis and ECU flashing?

Answering these questions requires understanding the intricate effects that individual decisions might have on the overall performance and timing. Furthermore, it requires a systematic procedure including appropriate supply-chain communications in terms of data sheets and requirements specifications, and finally tool-support.

In this paper, we demonstrate the application of scheduling analysis tools in architecture exploration and optimization based on two typical real-world problems. In the first case study, an OEM wants to add a new ECU to an existing CAN bus-based topology and want to check upfront if (and to what extent) this influences the timing and reliability of the existing systems. In the second case study, an OEM wants to evaluate (upfront) how to best replace an existing CAN bus by a FlexRay bus. In both case studies, we demonstrate how to predict the impact of the decisions up-front, i.e. as an explorative preparation before the decision is actually taken. For each case study, we highlight the original rationale for the architectural changes, the key questions related to timing and scheduling, and the solutions and decisions enabled through scheduling analysis.

But before we go into the details of the case studies, we will briefly review the general possibilities to escape the complexity dilemma.

TACKLING THE COMPLEXITY OF SCHEDULING

One common argument for the growing popularity of FlexRay is its determinism of frame timing (in the static segment). The underlying time-triggered mechanisms in principle enable a correct-by-construction approach that is so simple that it can be intuitively understood and controlled. This simplicity, however, can only be achieved when the entire system is a) based on time-triggered mechanisms and b) globally synchronized. While certain safety-critical domains may be realized in this way, such a solution is too costly and inflexible for most vehicle functions. Hence, there is general agreement that future cars will contain a mix of synchronous and asynchronous domains. Therefore, timing will remain a complex issue and will require profound analysis.

A variety of sophisticated methods are available from real-time scheduling theory (landmark contributions are found in [1, 2, 3, 4, 5]). From these basic analysis technologies, two different types of tools have been developed.

AUTOMATIC SCHEDULE SYNTHESIS

Automatic synthesis tools translate well-specified, formalized sets of requirements into optimized network configurations and schedules. Mentor's Volcano Network Architect (VNA) [6] is probably the most widely used synthesis framework. VNA automatically creates optimized communication for CAN networks based on Deadline-Monotonic Scheduling [1]. The synthesis algorithms are based on [7] that has recently been refuted [8].

The synthesis approach has several potential drawbacks: firstly, the necessary requirements are often not specified precisely enough to enable efficient automatic synthesis. Secondly, it is not easy for a synthesis algorithm to efficiently account for implementation constraints. Designs are typically evolutions of previous designs where legacy configurations and integration of re-used components have to be considered, and where many detailed architecture decisions have already been taken, e.g. dedicated basic software (drivers), certain communication types, or specific buffering mechanisms. Finally, synthesis also provides little feed-back to the designer on to the actual timing of the resulting system. Hence, users have limited means to qualify the synthesis results by their own metrics.

SCHEDULING ANALYSIS AND EXPLORATION

Scheduling analysis and exploration are based on similar algorithms as synthesis techniques but with a different philosophy.

The primary motivation of scheduling analysis is to evaluate the performance and timing of a possible system configuration, *not* to generate one. This allows analysis to be applied flexibly at different design stages, from the earliest stages (for optimization and exploration) until late (for verification). In particular, the possibility to quickly perform various types of "what-if" analyses lets architects explore and evaluate design alternatives, modifications or extensions before the design decisions are taken. I.e., scheduling analysis allows systematic timing testing at the earliest possible level.

It is also easy to consider implementation constraints such as legacy components: their properties are simply taken into account as part of analysis. By the same token, it is relatively straightforward to integrate scheduling analysis into different design-environments: for example, Symtvision's scheduling analysis tool suite SymTA/S [9, 10] can exchange required input parameters and results via standard formats such as FIBEX or AUTOSAR, and can be easily adapted to consider proprietary formats.

Scheduling analysis finds and evaluates critical timing situations automatically without user interaction. SymTA/S additionally includes powerful visualization

techniques, such as end-to-end Gantt-charts, to clarify to the designer the cause of a timing problem and give hints at possible remedies.

With SymTA/S, we have extended and adopted the most promising theoretical contributions to meet industrial requirements and combined them into a flexible and extensible system-level analysis tool suite. For example, SymTA/S is able to consider information about dynamic send patterns, interface queues, and register communication with sampling to further improve the accuracy and expressiveness of analysis results. In addition to those direct network timing influences, SymTA/S is also capable of considering ECU influences, most notably frame offsets and task-dependencies. These will also play a key role in our experiments later in this paper.

We have further embedded these analyses into design framework with automatic, customizable exploration and optimization possibilities based on "what-if" analysis [11, 12]. "What-if" analysis can evaluate effects of possible implementation decisions and thus enables the application of the technology already in earlier stages. Exploration as implemented in SymTA/S presents to the designer trade-offs between different design alternatives with respect to a set of properties that the designer has chosen to optimize. The main goal is to maximize designer understanding of design implications. It remains the designer's responsibility to draw the appropriate conclusions.

TIMING & PERFORMANCE ANALYSIS DURING THE COMMUNICATION DESIGN PROCESS

Figure 3 illustrates typical design steps that influence the final timing and thus performance of the system:

- Topology design: here, the number and location of ECUs and their physical connections are decided, largely determined by requirements such as power supply, wiring possibilities, available space, and access to sensors/actuators, etc. A coarse-grain function mapping is also performed. This allows estimating bus loads, and determining whether the envisioned protocols and bus speeds are capable of transmitting that load. Timing is not considered in detail at this stage.
- Detailed function mapping: the mapping of functions and software components to the available ECUs is detailed; this also details which signals will travel over which physical communication link. The combined impact on timing of these steps is significant but becomes apparent only later in the design. The output is a "Mapped System".