Comparative Application of Real-Time Verification Methods to an Automotive Architecture

Steffen Kollman, Victor Pollex, Kilian Kempf, Frank Slomka, Matthias Traub, Torsten Bone, Jurgen Becker

Comparative Application of Real-Time Verification Methods to an Automotive Architecture

Steffen Kollmann, Victor Pollex, Kilian Kempf and Frank Slomka
Ulm University
firstname.lastname@uni-ulm.de

Matthias Traub and Torsten Bone
Daimler AG
firstname.lastname@daimler.com

Jürgen Becker
Karlsruhe Institute of Technology
becker@kit.edu

Abstract

Designing embedded systems is a challenge. This applies especially to distributed automotive architectures. The high connectivity of different control units forms heterogeneous system architectures that have to handle the many different applications involved in providing the systems' services. This has a direct impact on model based design techniques which must be able to verify that different design constraints are satisfied. One issue is the real-time analysis for checking whether the different applications meet their real-time deadlines or not. In the past 40 years, many techniques have been developed to find an answer to this question. In this paper, different approaches for the verification of real-time behavior of an automotive architecture are compared. Two tools that are available on the market and two from universities, including a new prototype, have been chosen. In order to compare these tools in an appropriate way we have applied them to an automotive industry state of the art architecture model.

1. Introduction

Embedded systems are the most widespread computer systems in the world. Hidden from the users' view, they perform steady operations while the users rely on the correctness of the system. Cars, airplanes, and mobile phones are examples for this unnoticed relationship between computers and users. The confidence in the systems' function is based on highly dependable design processes present in the industry. Due to the increasing system complexity, new design processes and methodologies are necessary to maintain the standards of reliable systems. The complexity is mainly a result of the applications being distributed over the whole system, for example a layout caused by sensor fusion or the necessity of separating actors and sensors.

This work is supported in part by the Carl Zeiss Foundation and the German Research Foundation.

Many technical constraints have to be considered during the design process of these systems. Beyond that, the automotive industry has economical constraints. As a result, the number of applications which have to be integrated on a single electronic control unit (ECU) is constantly increasing.

Besides timing issues, additional constraints like power consumption, space requirements, weight, etc., have to be considered. This results in a multi-criteria optimization problem, where many different possible solutions have to be assessed during the design process in order to find an optimal solution. Such a design space exploration needs fast verification algorithms to check the requirements.

One challenging aspect in the design of automotive architectures involves the communication behavior and network topologies. Increasing data exchange between the ECUs results in increasing requirements concerning the networks. Key questions in this context include busload, communication relations between the ECUs, and the resulting timing behavior of a network branch or the entire network [24]. Different approaches are available for the evaluation of these questions. To our knowledge, no comparative case studies considering design challenges of the automotive area exist.

Perathoner et al. have published a comparison of different approaches in [18], using a set of five benchmark problems for their analysis. These benchmarks have been specifically constructed to analyze the behavior of different methodologies under certain conditions and reveal their particular characteristics. While this is appropriate for the analysis of specific properties, it hardly allows conclusions to be drawn about their performance when applied to real-world architectures and systems.

A more complex exemplary architecture has been provided by Künzli et al. in [15], which was used to show the benefit of a combination of two different methods on a distributed real-time system. The example is still specifically designed and does not necessarily match the industry's present systems.

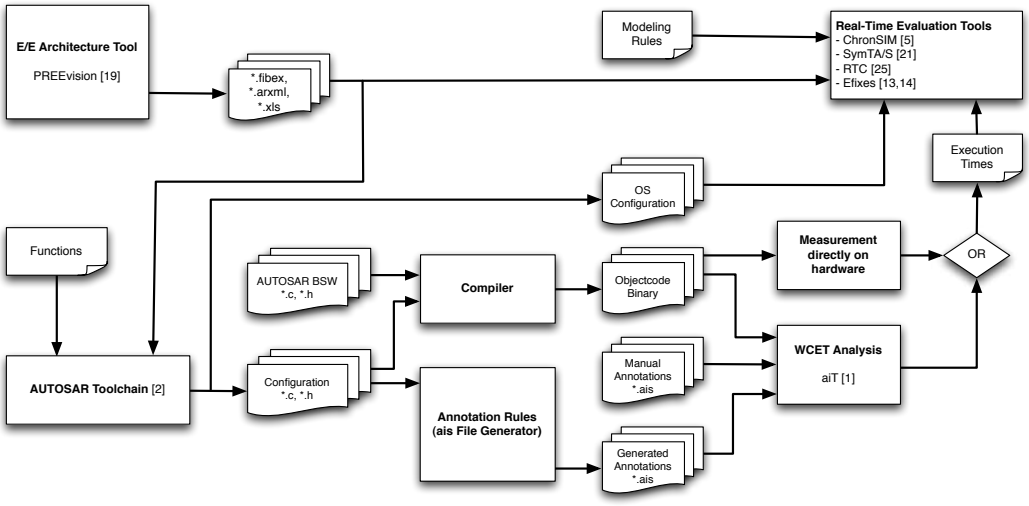

Figure 1: Design flow, including timing evaluation [23]

Electric/Electronic architectures (E/E architectures) from the automotive domain are the prime examples for the problems discussed above. Contemporary automotive E/E architectures are highly networked systems with many real-time constraints. Networked cars may have more than 70 ECUs connected by different bus systems like CAN (controller area network) and FlexRay. These different types of busses and ECUs designed by different suppliers are commonly used in the same networking architecture. Such a heterogeneous architecture has to be integrated by the original equipment manufacturer (OEM), which requires an appropriate design process including the consideration of real-time constraints of a single system and the whole network architecture.

In this paper, different approaches for the analysis of the real-time behavior of automotive E/E architectures as introduced above are compared. This should help to get an impression of the developed techniques and their ability to cover the challenges of the automotive area. For this comparison, a part of a larger automotive network architecture from Daimler has been considered and different real-time evaluation methods have been applied. On this basis it will be shown that it is not sufficient to use only one technique and that it is very important to have all information about the timing behavior of the systems available for a sophisticated analysis.

This paper is organized as follows: Section 2 provides an overview of the design flow aspired by industrial companies. After this, the networking architecture used for this case study is presented. Section 4 introduces the considered real-time evaluation tools. The results of this case study are discussed in section 5, conclusions are presented in section 6.

2. Design Flow

Due to increasing complexity of automotive E/E architectures, model based design techniques are a main branch of research in industrial companies. In this section, we characterize a possible design flow integrating timing evaluation tools into the development process.

As design entry for the network architecture, we use the E/E architecture concept tool PREEvision of the company aquintos [19]. This tool is a model based design approach based on a domain specific language which is used to design E/E architectures.

Based on this PREEvision model, exporting the necessary information of the system to the real-time evaluation tools described in section 4 is required. We have used the standard format FIBEX [7] for bus configurations and a simple comma separated value (csv) format for the remaining information of the architecture. This information is then imported into the considered real-time evaluations tools.

Unfortunately, not all information for the real-time analysis can be extracted directly from the PREEvision model, because necessary details about the software architecture are not provided. Performing a timing verification requires the knowledge of worst- and best-case execution times of messages and tasks in the system. The execution times of message transmissions can be directly derived from the bus speed and the message size. E. g. the transmission of a standard message with an eight byte payload on a 500 kBit/s CAN bus takes up to 240 µs. Since the execution of preemptive tasks on ECUs is not deterministic, the extraction of task execution times is more sophisticated.

For the extraction of the relevant task execution times, executable code has to be generated. Information on the network architecture and the functions are imported into the AUTOSAR toolchain [2]. This allows to take the operating system configuration into account, which is important, as the operating system has a direct impact on the execution times of the tasks. After the AUTOSAR software for each ECU has been configured, an executable binary

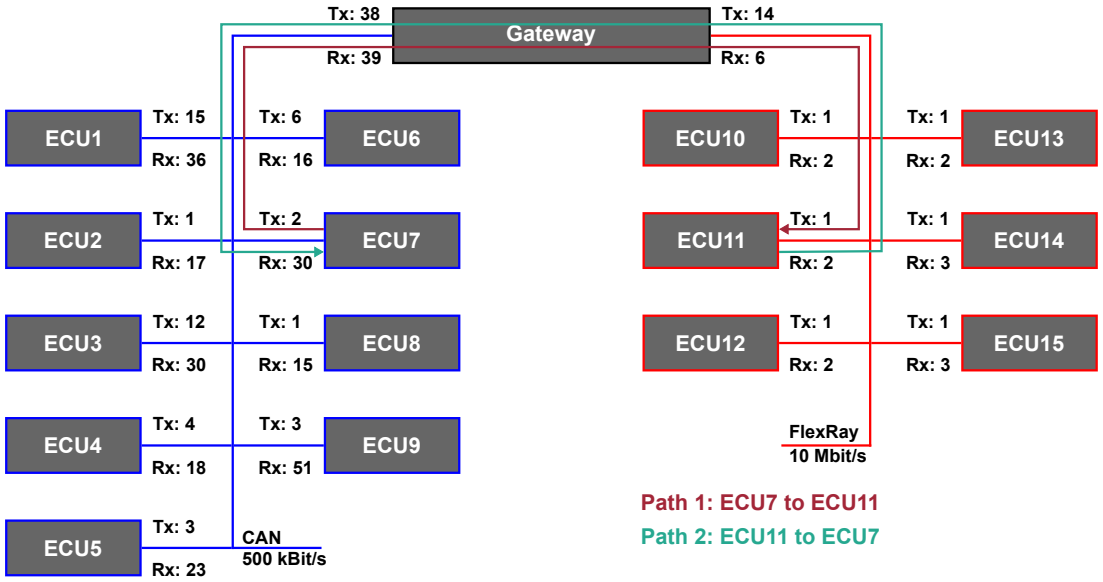

Figure 2: Network view of the E/E architecture

can be compiled. In the next step, the execution times of the functions and tasks can be measured or analyzed. This measurement can be done directly on a hardware prototype. Alternatively, it is possible to conduct a static code analysis as provided by the tool aiT of Absint [1].

For the refinement of the timing models, modeling rules are used. This means that the different approaches are not capable of modeling the system in the same level of detail. Some approaches consider more details than the others. These differences are discussed in section 4.

3. System Architecture

In this section, the networking part of an E/E architecture that we have used for the case study is presented. Since the whole architecture is too complex, only the parts relevant for the real-time analysis are described.

3.1. Networking Architecture

The networking architecture is shown in figure 2. It consists of 15 electronic control units (ECUs) and two busses connected by a gateway. The number of different messages transmitted (Tx) and received (Rx) by each ECU and gateway on the connected bus is annotated. E. g. the gateway transmits 38 messages and receives 39 messages over the CAN bus.

The gateway and ECU1 to ECU9 are connected by a 500 kBit/s CAN bus. In total, 85 different messages are sent over the CAN bus, generating an average bus load of 41.63 %. Each message has a unique identifier, where a lower identifier value denotes a higher priority. The transmission scheduling scheme of the CAN bus can be modeled with a fixed-priority non-preemptive scheduling policy.

In order to prevent peak loads on the bus, the startup times of the CAN messages have offsets. This leads to a more uniform bus load and thus to shorter response times of the messages, because fewer messages compete for simultaneous bus access.

Since most of the ECUs transmit more than one message, a scheduling policy for the transmission of the messages is necessary. Two main policies are used in such architectures. One of them is the fist-in first-out (FIFO) policy, which is used by a basic CAN controller. This policy can lead to a priority inversion, as illustrated:

Example 1 *Assume that ECU1 has a basic CAN controller that uses a FIFO policy and wants to transmit message 1 with identifier 0x2 and message 2 with identifier 0x100 in short succession. Message 2 is queued first and message 1 is queued after message 2. Therefore message 1 has to wait until message 2 has been sent although message 1 is the highest priority message. This means that message 1 can not only be delayed by higher priority messages from other ECUs but also by messages from them with an identifier in the range from 0x3 to 0xFF.*

The other policy mentioned above is a priority order policy, where the highest priority message in the queue of the controller competes for bus arbitration at any given time. This behavior leads to a more predictable behavior on the CAN bus. Therefore this behavior has been chosen as the default behavior of a CAN controller.

The second bus used in the architecture is a FlexRay bus, which connects the gateway and ECU10 to ECU15. This bus has a data rate of 10 Mbit/s with a cycle time of 5 ms. Only the static configuration part of the bus protocol, which is implemented as the deterministic Time Division Multiple Access (TDMA) policy, is considered.

The gateway connects the CAN bus to the FlexRay bus by translating CAN message frames into FlexRay message frames and vice versa. It is not modeled in detail and the time spent on the protocol translation of the

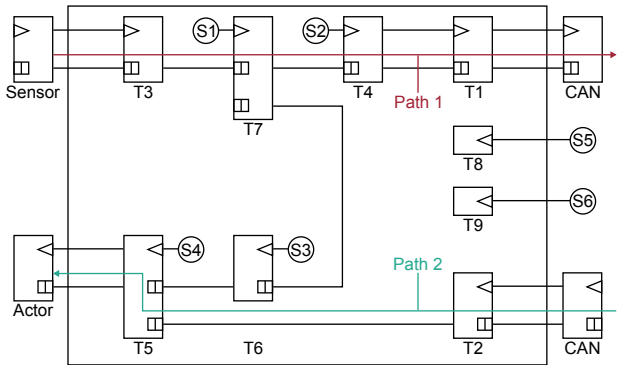

Figure 3: Software architecture of ECU7

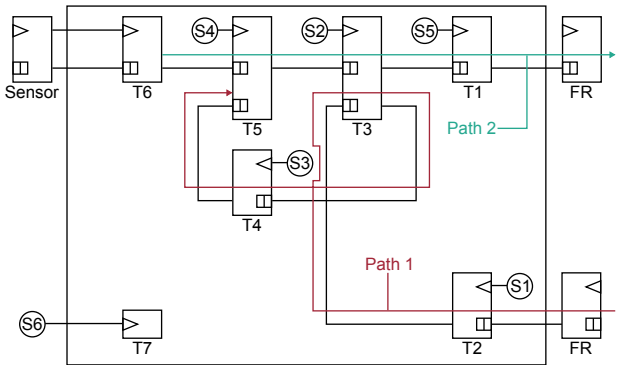

Figure 4: Software architecture of ECU11

data is assumed to be 1 ms regardless of translation direction and data amount. The delay caused by the transition from the event triggered CAN bus to the time triggered FlexRay bus is considered when the end-to-end delay is determined.

To consider path latencies in the analysis, the data exchange of ECU7 and ECU10 is modeled in detail. They are described in the following sections. Both use an OSEK-OS operating system [17].

3.2. Software Architecture of ECU7

ECU7 consists of three interrupt service routines (ISRs), T1 to T3, and six tasks, T4 to T9, as shown in figure 3. The corresponding parameters of the tasks are listed in table 1. The data input of a task or ISR is represented by two rectangles and the trigger by a triangle.

Each task can be an ISR or is either preemptable or not. All ISRs have the same and also the highest priority. They are scheduled according to a FIFO policy, all the other tasks according to their respective priority. This can be regarded as a hierarchical scheduling on the ECU.

The ISRs T1 and T2 are triggered asynchronously. T1 is triggered after T4 has finished its execution and data is ready to be sent over the CAN bus. T2 is triggered when a CAN message is received by this ECU. The other tasks that are triggered asynchronously are T8 and T9. They are triggered every 15 ms and every 50 ms, respectively.

Name	Type	Priority	WCET [µs]	EVENT [ms]	offset [ms]
T1	ISR_CAT2	255	20	-	-
T2	ISR_CAT2	255	20	-	-
T3	ISR_CAT2	255	10	20 (Sync)	0
T4	Non-Preemptive	25	80	20 (Sync)	1
T5	Non-Preemptive	24	200	20 (Sync)	10
T6	Non-Preemptive	23	500	20 (Sync)	5
T7	Non-Preemptive	22	100	20 (Sync)	0.5
T8	Preemptive	21	50	15	-
T9	Preemptive	20	150	50	-

Table 1: Parameters of the tasks on ECU7

The remaining tasks T3 to T7 are triggered synchronously every 20 ms, where the triggering events are delayed by time offsets. This eliminates blocking times and reduces response times.

Sensor data is propagated from the receiving ISR T3 via task T7, which is activated 0.5 ms after T3, and via task T6 (activation offset 5 ms) and T5 (activation offset 10 ms) to the actuator.

A second data path starts at T7 via the task T4 to the ISR T1 sending a message to the CAN bus. The message is propagated to ECU11 on the FlexRay side. At ISR T2, ECU7 receives a second sensor value originating from ECU11, which is then transmitted to T5.

3.3. Software Architecture of ECU11

The software architecture, the data paths, and the corresponding parameters of ECU11 are shown in figure 4 and table 2. The control unit consists of five tasks, T3 to T7, and two ISRs, T1 and T2. As on ECU7, the operating system is OSEK, so the same scheduling policies apply to this unit.

The two ISRs represent the communication with the FlexRay bus. Since the bus uses a time-triggered protocol, the ISRs are activated by a strictly periodic stimulation that is synchronized to the FlexRay cycle.

Beside the main application, another small application is running on the ECU and is implemented by the task T7.

Name	Type	Priority	WCET [µs]	EVENT [ms]	offset [ms]
T1	ISR_CAT2	255	80	20 (Sync)	1.5
T2	ISR_CAT2	255	70	5 (Sync)	0
T3	Non-Preemptive	25	120	20 (Sync)	1
T4	Non-Preemptive	24	10	20 (Sync)	1.5
T5	Non-Preemptive	23	180	20 (Sync)	0.5
T6	Non-Preemptive	22	200	20 (Sync)	0
T7	Preemptive	21	100	15	-

Table 2: Parameters of the tasks on ECU11

This task is triggered every 15 ms, asynchronously to the other tasks.

The main application consists of the tasks T3 to T6. Incoming data, for instance from ECU7, is received by task T2. The data is then propagated via T3 and T4 to T5. Just as on ECU7, the tasks are triggered synchronously and they are delayed to avoid preemption by other tasks.

A second data path starts at task T6. From the output of T6, the data is processed by T5 in conjunction with data received from ECU7 and then delivered via T3 to T1, the transmit task. The message is then sent to the gateway and back to ECU7.

3.4. Data Paths of Interest

Two aspects are of special interest for the evaluation of the timing tools: The worst-case delays that can occur to messages on the CAN bus and the end-to-end delay of the two paths shown in figures 2, 3, and 4.

The first path starts at the input of T3 of ECU7 and runs through T7, T4, and T1 to the CAN bus. The data is sent as CAN message 47. At the gateway, it is translated into a FlexRay frame and sent to ECU11. T2 of ECU11 receives the data and the path continues via T3 and T4 to the input of T5 where the path ends.

The second path starts at the input of T5 of ECU11 and runs through T3 and T1 to the FlexRay bus. The data is received by the gateway and translated into CAN message 38. It is received by ECU7 (T2) and propagated through T5 to the input of the actuator where the path ends.

4. Evaluation of Real-Time Behavior

After discussing the architecture we will now introduce the approaches used in this case study. In principle, only two main concepts can be applied in early design states of such systems. One is a simulation and the other is an analytical approach, where both have their advantages and disadvantages. With the help of a simulation tool it is possible to gain a detailed insight into the system but it is very difficult to get guaranteed results regarding the response times. As in every simulation approach, the main problem is the coverage of the cases. It cannot be guaranteed that the worst case has been found in a simulation run. We used only one simulation tool for this case study, and that is the software chronSIM by the company Inchron [5].

Analytical approaches solve the coverage problem mentioned above. They construct a worst case for the analyzed system and calculate a bound for that case. The limitation of such approaches is the difficulty to consider the different system contexts, like offsets between the task stimulations or mutual exclusion of tasks. For tight response times, it is important to consider all relevant contexts. Many approaches have been developed, differing in the ability to model the system, especially in the consideration of system contexts. Due to significant differences between the analysis methods, three approaches have been chosen for this case study.

The first one is the real-time calculus [25] developed at the ETH Zurich. This approach is not able to consider the contexts necessary to analyze the system, so we used it as the one that describes the case where tasks' stimulations are independent.

The second approach we used is SymTA/S [20] by the company Symtavision [21]. It is based on the busy window approach and is able to consider the contexts in the system and is therefore assumed to be accurate.

The last tool we have considered is a prototypical development of Ulm University and is called Efixes. It differs from the SymTA/S approach in that it uses a far more general technique to describe stimulations [14] in the system and is able to model dependencies in a more general way. This is a technique that is approximative as well as scalable. All the mentioned approaches are introduced in the next four sections.

4.1. ChronSIM

Simulation is often used in different design phases. The advantage of such a technique is the possibility for the designer to gain a very deep insight into the system. Based on simulation traces, the interaction of the different components of the architecture can be observed, for example the impact of combined scheduling strategies on each other. With the help of a simulation it is possible to observe and explain side effects caused by the components.

In this case study chronSIM version 2.0.0 (Build 07/05/2010) has been used. With this software, C or C++ code that is directly executable on the real ECUs can be used to describe the behavior of the system. The architecture and the interconnection of the different ECUs are modeled by the simulator, which then compiles a binary program that is executed on the host computer and generates the desired simulation traces.

In this case study we have used the generic ECU model provided by the software and annotated the previously determined worst-case execution times.

Due to the typical coverage problems of simulation approaches, it can not be guaranteed to find the worst case. This is a general simulation weakness and also applies to measurements. We expect the response times always to be lower than those calculated by the analytical methods.

4.2. Real-Time Calculus

The real-time calculus is based on the network calculus [3]. It has been extended by Chakraborty et al. [4] and Wandeler [25] to be applicable to distributed real-time systems.

The basic idea is to have a modular approach in which every task can be analyzed independently. One type of necessary parameters are the arrival curves. They are used to model the rate at which the task is triggered. Additionally, the service curves are needed, which implicitly model the scheduling policy the task is subjected to. By using convolution and deconvolution operations in the min-plus and max-plus algebra, the outgoing arrival curves and the

remaining service curves are computed. They can be used by other tasks as their respective arrival or service curves.

For the analysis, the freely available RTC Toolbox [26] has been used. The CAN bus in the architecture can be modeled as a resource that uses a non-preemptive fixed priority scheduling policy, therefore the method found in [9] was implemented.

It is expected that the analysis with the real-time calculus will give overly pessimistic results for the worst-case response times. This is due to the approach currently not being able to consider the offset relations between messages or tasks.

4.3. SymTA/S

For the case study we have used version 2.2 of SymTA/S based on the Eclipse framework.

The method this tool uses to analyze systems is based on a busy window approach that has been introduced by Lehoczky [16] to analyze tasks with arbitrary deadlines. It was extended by Tindell and Clark [22] to a holistic analysis for distributed real-time systems. An improvement of this analysis was achieved by the introduction of the minimum distance by Richter [20]. The messages that are sent over the CAN bus have known temporal offsets to each other. By applying [10] and [11], this knowledge can be used in the analysis. The idea of the approach is to group the correlated tasks into transactions. In a transaction, each task has a relative offset to a previous event. To find the worst case, all possible combinations of the tasks have to be considered, which can lead to longer runtimes in large systems.

SymTA/S uses the techniques of all of the presented work above. We expect that this approach delivers the tightest bounds for the worst-case response times.

4.4. Efixes

As last analytical tool we have used Efixes from Ulm University. The tool is based on the Eclipse framework and prototypically implements a schedulability analysis for distributed systems.

The analysis uses the event stream model by Gresser [8]. The complete methodology is described in [12] and [14]. The difference to the SymTA/S approach is the ability of the event stream model to describe any kind of stimulation. So in order to describe burst behavior it is not necessary to change the model and therefore the analysis methodology. A second and far more important difference of this concept was introduced in [13], where a general model for the description of dependencies between several event streams is proposed. The idea is to describe an arbitrary dependency between any stimulations, by which a limitation of the cumulated number of events is given. As an advantage of the concept, each limitation is computed separately from the analysis. This technique makes it possible to describe any kind of dependency, like offsets or mutual exclusion.

To get a general idea of how Efixes works, the stimulations can be considered as vertices in a graph and the correlation between them as edges. For each maximal clique in the graph of size 2 or bigger a set S_i is built which contains all stimulations that are represented by the nodes of the clique. Note that these cliques are usually given as input and are not searched for. To consider every possibility which is described by the correlations, the power set of such a previously mentioned set $\mathcal{P}(S_i)$ is taken. For each element of such a power set $M \in \mathcal{P}(S_i)$ a limitation for the number of events that can occur by the combined stimulations in M is computed.

This very modular concept permits to consider correlations in a scalable way by choosing only a subset of the power set which should be considered in the analysis. This is done in the CAN analysis where for each ECU a set S_i is built containing all stimulations of the messages that are sent by the ECU. Out of all elements of the corresponding power set $\mathcal{P}(S_i)$ only those were chosen that contain exactly the stimulations of the k-highest priority CAN messages that are sent by the ECU, where $2 \leq k \leq n$ and n is the number of messages sent by the ECU.

We expect that the response times are not as tight as those of the SymTA/S approach, but we expect them to be computed faster. The speed of such an approach is very important for a design space exploration.

5. Results

The tools presented in the last section have been used to analyze the networking architecture introduced in section 3. The aim of each approach is to find the worst-case response times of the tasks and messages. Furthermore, we have considered two end-to-end-path latencies introduced in section 3.

While the verification tools cover the worst case, the simulation usually does not, so it is not intuitively known how long the simulation has to run. To find an answer, we calculated the hyper-period of the system, which is two seconds, and ran the simulation 50 times as long (100 seconds were simulated).

First of all, we are interested in the worst-case response times of the messages that are caused by the busses. The response times of the FlexRay bus is not really surprising as it uses a TDMA policy and the messages are not longer than the slot length. The worst-case response time is equal for all messages, in this case 5 ms or a multiple of 5 ms, depending on the repetition factor. Therefore, we do not consider the results concerning the FlexRay bus in detail.

The worst-case response times introduced by the CAN bus are far more interesting, as the transmission offsets between the messages have a great impact on them. In figure 5, the absolute worst-case response time of each CAN message is shown for all the approaches covered in this case study. The messages have been ordered by descending priority. The deadline of each CAN message has been set to 70 % of its period.

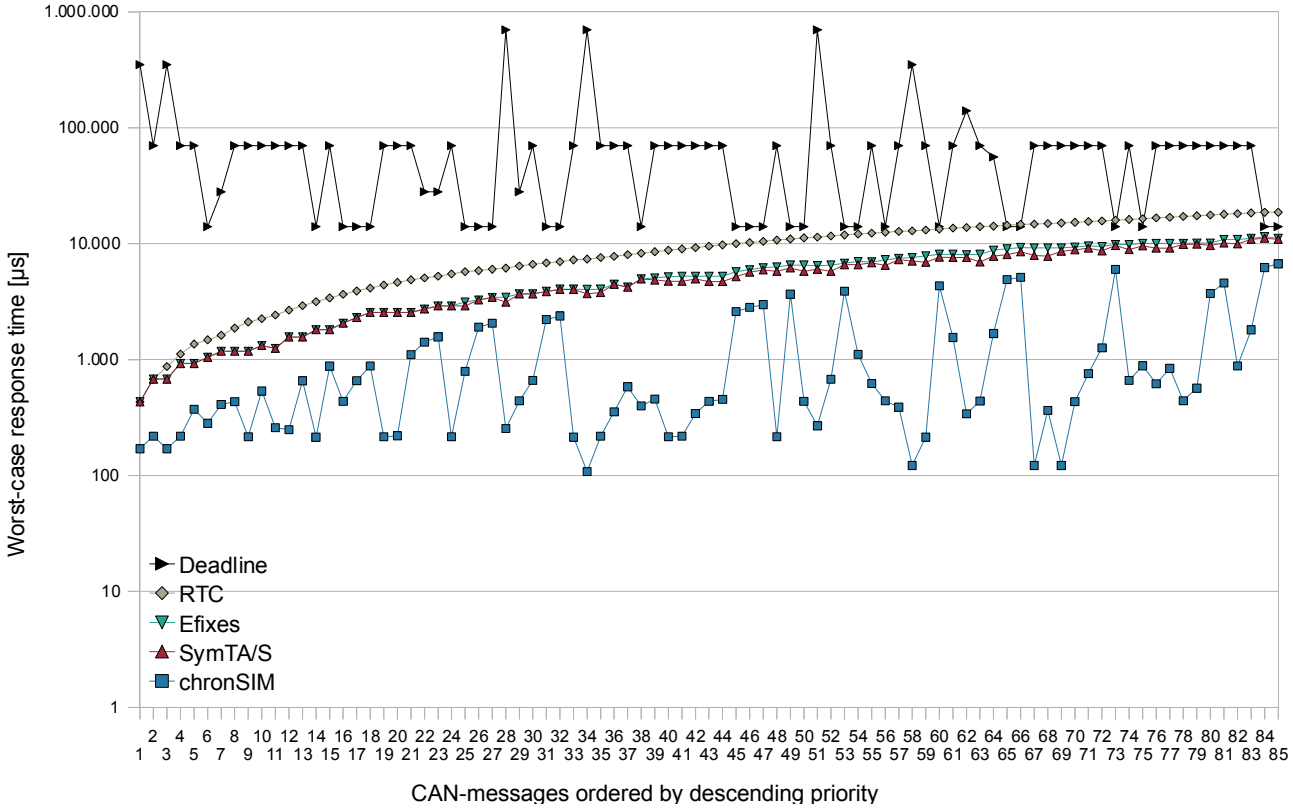

Figure 5: Worst-case response times of the approaches (absolute)

As expected, the RTC toolbox is generally overly pessimistic due to not considering the offset dependencies between the messages. It can be observed that the worst-case response time of a lower priority message is always greater than that of the next higher priority, because without considering contexts, the interference of the messages is always maximal. Generally, the worst-case response time of a message must be at least the same as that of the next higher priority plus the worst-case execution time of the message itself.

This behavior does not apply when offsets between the tasks are taken into consideration. Due to the time-shifted activation of the messages it is possible for a message to have a lower worst-case response time than one of a higher priority. With messages 66 and 67, for example, this behavior can be observed, because SymTA/S and Efixes consider the offsets.

The results of SymTA/S and Efixes are close together, where SymTA/S gives either equal or better results than Efixes. This is due to Efixes using an approximation when considering the dependencies between the messages. The differences between SymTA/S and Efixes can be seen better in figure 6, where the response-time of a message is given as the ratio of its period. The approximation level can be easily chosen by the Efixes tool as described in section 4.4. For this case study the Efixes model leads to good results concerning the runtime and the exactness of the response times. But this can vary from application to application. In other scenarios the difference, concerning the response times, to the SymTA/S approach can be greater depending on the chosen approximation level.

The simulation approach of chronSIM results in the lowest values of the five curves. Out of the simulated 100 seconds, the maximum response time of each message has been extracted from the simulation trace. It can be seen that the response times diverge widely, so it can be assumed that the simulation has to run far longer to converge at the worst case for all messages.

As expected, the worst-case response times encountered in the simulation with chronSim are always lower than the values given by the analytical approaches. The variability in the response times is due to the general coverage problem of simulations, where it is unclear if the worst case has been simulated. It might be assumed that a long-term run of the simulation will lead to a smoothing of the curve.

The question resulting from the computed values is how realistic the results are. As the analysis usually tends to overestimate the worst case and the simulation underestimates it, the true value must lie between both results. Take message 23 for an example. The lowest value computed by an analysis tool is 2898 ms and the highest value of the simulation is 1570 ms. The difference is 1328 ms, which is 54 % of the worst case of the analysis. In contrast, message 33 has a difference of 3818 ms. The simulation value is in this case 5.3 % of the worst case. It is

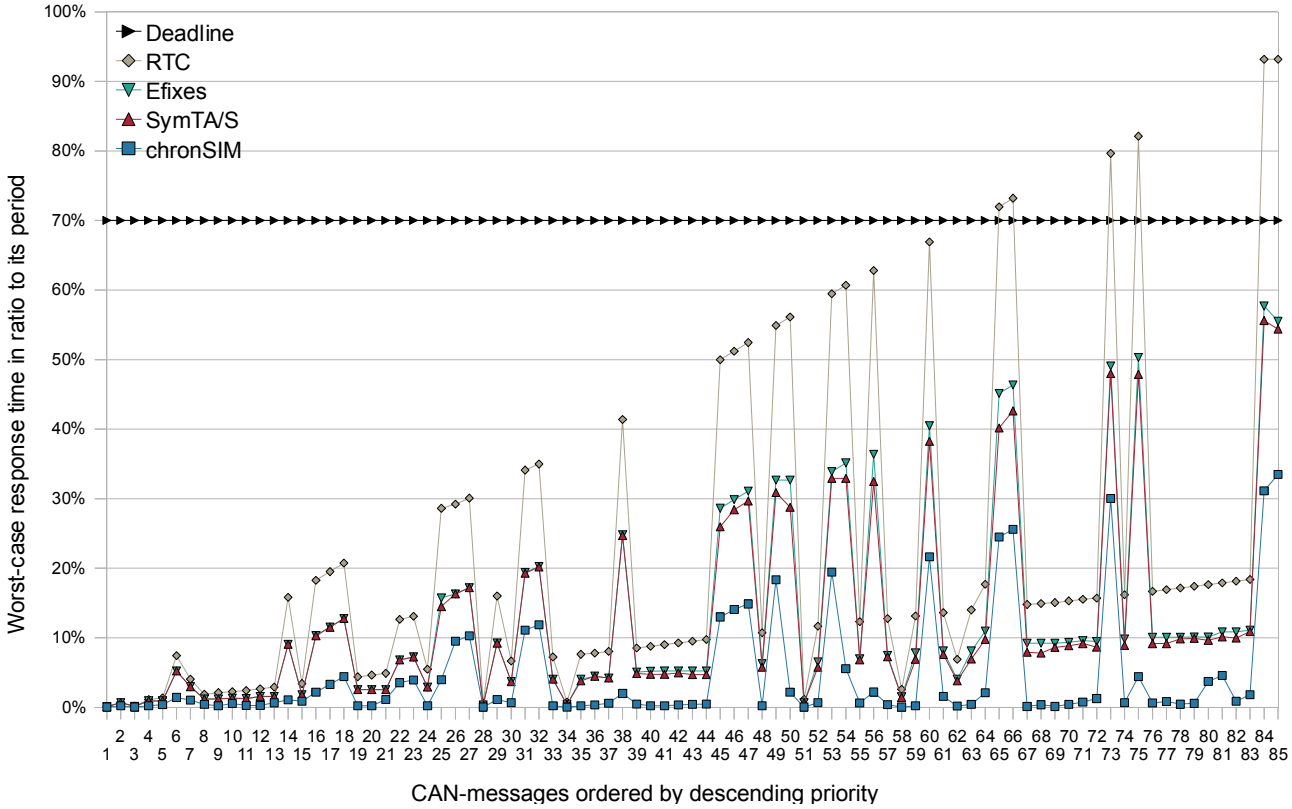

Figure 6: Worst-case response times of the approaches (normalized)

not clear if the simulation has not found the worst case or the analysis has been overestimating, but small distances indicate a good quality of the results.

After having discussed the response times of the CAN messages, we will now consider the path latency of the paths described in section 3. The results of both paths are depicted in figure 7. In order to determine the path latencies, we have used two approaches. The simulation tool provides a concept of event chains, which allows to extract the path latency directly from the trace file. To compare the different verification tools, we have used the approach of SymTA/S, presented in [6], where a possibility to determine the path latency is described.

As mentioned above, the first path starts on ECU7 at the input of T3 and leads to the input of T5 on ECU11. Due to the architecture, many delays of the path are equal in all analytical tools. So only the tasks or messages where a difference exists were considered. One difference is caused by the delay of the CAN message as described in table 3. Since the RTC considers no correlation between the messages, the response time is the worst. The SymTA/S approach delivers the best results. The other tasks T1 and T4 on ECU7 and T4 on ECU11 have different response times for the same reasons.

The simulation approach extracts the path latency directly from the trace file. Figure 7 shows that the latencies of most approaches are close together. Only the RTC overestimates the latency significantly.

The second path examined is the one from the input of task T5 on ECU11 to the actor on ECU7. Again, only the most important differences between the analytical tools were considered, which are described in table 4. As in the first path, the worst-case response time of the CAN message and the response time of the tasks are responsible for the different results of the analytical tools. On this path there is a large gap between the simulation the analytical

Task	RTC	Efixes	SymTA/S
Message 47	10488 µs	6216 µs	5928 µs
T1 ECU7	50 µs	50 µs	40 µs
T4 ECU7	630 µs	120 µs	120 µs
T4 ECU11	480 µs	90 µs	90 µs

Table 3: Worst-case response times of tasks and messages on path 1 of the analytical tools

Task	RTC	Efixes	SymTA/S
Message 38	8274 µs	4962 µs	4944 µs
T1 ECU11	150 µs	150 µs	80 µs
T2 ECU7	50 µs	50 µs	40 µs
T5 ECU7	830 µs	240 µs	240 µs

Table 4: Worst-case response times of tasks and messages on path 2 of the analytical tools

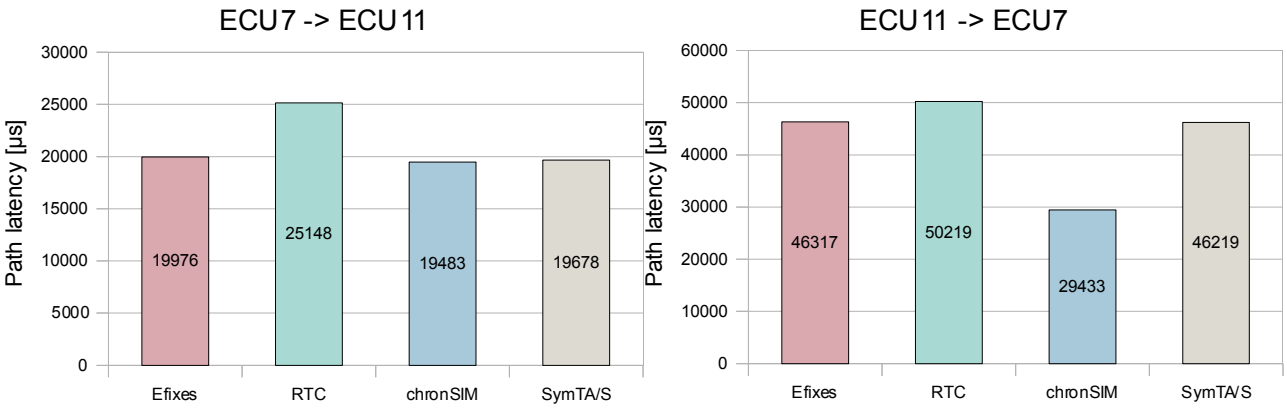

Figure 7: Path latency

approaches, as depicted in figure 7. The analytical methods as well as the simulation have an inherent problem. For the analytical approaches it is not clear if enough contexts have been considered during the analysis to compute tight results. For the simulation approach it is difficult to choose the right modeling parameters in order to construct the worst case. Therefore it is important to have both results (analytical and simulation) to get an impression of the quality of the results.

After considering the different results of the worst-case response times, we will now evaluate the runtime of the different approaches. In figure 8, the computation time needed for each approach is shown.

Both Efixes and RTC were run on a machine with an Intel 2.66 GHz processor, 4 GB of RAM. JavaVM 1.6.0_20 was used for Efixes and Matlab 7.8.0 was used for the RTC Toolbox.

The ChronSIM experiments were carried out with a license of ChronSIM 2.0.0 (Build 07/05/2010) inside a virtual machine running Windows XP on the same computer. The time shown in figure 8 was needed to simulate two seconds, which is the hyper-period of the architecture.

The SymTA/S experiments were carried out with a license of SymTA/S version 2.2 on an Intel Core Duo T7500 2.2 GHz, 3 GB of RAM.

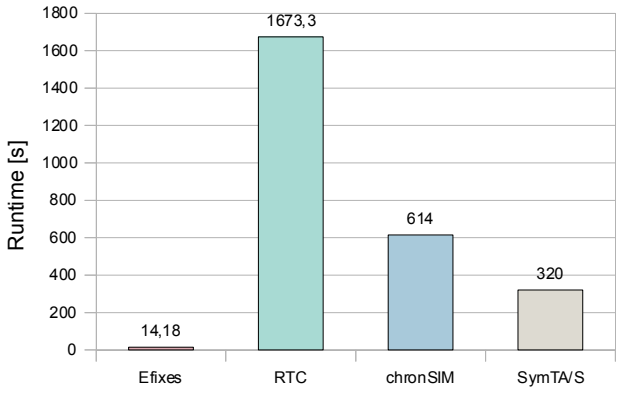

Figure 8: Runtime of the approaches

For the runtime comparison, it is not important to know the exact values in milliseconds, because the results differ so much from each other. The first surprising point we observed is that the RTC has by far the longest execution time, even though the RTC does not consider any system context and therefore should have been very fast. One explanation could be that each arrival and service curve is explicitly described for the hyper-period of two seconds. Applying the different convolution and deconvolution operations leads to an enormous runtime.

The SymTA/S approach considers all system contexts and is therefore not so fast. Such an analysis is appropriate for hard real-time verifications where the accuracy is more important than the runtime.

Efixes is the fastest approach but delivers not the tightest results, which makes this approach suitable for design space explorations as described in the introduction.

The simulation needs several hours to simulate 100 seconds while not even ensuring that the worst case has been found. This is not very appropriate to find worst-case behavior. Therefore it is better to use the simulation approach to examine side effects in the system. This cannot be done with analysis tools.

6. Conclusion

For this paper, a large case study of a real automotive networking architecture was conducted. This architecture represents the current complexity in the vehicle networking design processes of automotive companies. Due to the complexity of heterogeneous architectures, one of the big issues of today's systems is the timing verification of the real-time applications.

For the real-time evaluation, different available approaches have been evaluated, one tool for the simulation and three analytical tools. The latter differ in the way they consider different kinds of system contexts. The relevant contexts were the offsets between the different CAN message transmissions and those between the ECUs' tasks.

The real-time calculus was used to describe the case of independence between the tasks' stimulations. This

method delivers the worst results since it cannot consider the offset correlation between the tasks.

The second verification tool SymTA/S considers all the offset correlations. It delivers the best results of all tools. This results in a longer runtime caused by the huge number of possible combinations that have to be computed to determine the worst case.

The tool Efixes makes use of an approximation in order to achieve shorter runtimes. This tool only consideres a subset of the contexts. The result is a fast analysis with some bounds that are less accurate than those of the SymTA/S approach.

The conclusion of this case study is that simulation approaches and analytical approaches are both very important for the design of complex systems. The simulation helps to understand in detail what the system is doing. The analysis tools are important to obtain guaranteed bounds for the response times in the system.

Since the real response times of the system must be between the simulation and verification results, it is desirable to close the gap between them. Therefore the specific properties of a system have to be considered to get realistic bounds.

References

[1] AbsInt. `http://www.absint.com`.

[2] AUTOSAR. `http://www.autosar.org`.

[3] J.-Y. L. Boudec and P. Thiran. *Network calculus: a theory of deterministic queuing systems for the internet*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[4] S. Chakraborty, S. Künzli, and L. Thiele. A General Framework for Analysing System Properties in Platform-Based Embedded System Designs. In *DATE '03: Proceedings of the conference on Design, Automation and Test in Europe*, pages 1–6, Washington, DC, USA, 2003. IEEE Computer Society.

[5] chronSIM. `http://www.inchron.com`.

[6] N. Feiertag, K. Richter, J. Nordlander, and J. Jonsson. A compositional framework for end-to-end path delay calculation of automotive sytems under different path semantics. *Work. on Compositional Theory and Technology for Real-Time Embedded Systems CRTS, Barcelona (E)*, 2008.

[7] FIBEX. `http://www.asam.net`.

[8] K. Gresser. An event model for deadline verification of hard real-time systems. In *Proceedings of the 5th Euromicro Workshop on Real-Time Systems*, pages 118–123, 1993.

[9] W. Haid and L. Thiele. Complex Task Activation Schemes in System Level Performance Analysis. In *CODES+ISSS '07: Proceedings of the 5th IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis*, pages 173–178, New York, NY, USA, 2007. ACM.

[10] R. Henia and R. Ernst. Context-aware scheduling analysis of distributed systems with tree-shaped task-dependencies. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, pages 480–485, Washington, DC, USA, 2005. IEEE Computer Society.

[11] R. Henia and R. Ernst. Improved offset-analysis using multiple timing-references. In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 450–455, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association.

[12] S. Kollmann, K. Albers, and F. Slomka. Effects of simultaneous stimulation on the event stream densities of fixed-priority systems. In *Spects'08: Proceedings of the International Simulation Multi-Conference*. IEEE, June 2008.

[13] S. Kollmann, V. Pollex, K. Kempf, and F. Slomka. A scalable approach for the description of dependencies in hard real-time systems. In *proceedings of the 4th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*, 2010.

[14] S. Kollmann, V. Pollex, and F. Slomka. Holisitc real-time analysis with an expressive event model. In *proceedings of the 13th Workshop of Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen*, 2010.

[15] S. Künzli, A. Hamann, R. Ernst, and L. Thiele. Combined approach to system level performance analysis of embedded systems. In *Proceedings of the 5th IEEE/ACM international conference on Hardware/software codesign and system synthesis*, page 68. ACM, 2007.

[16] J. P. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *Proceedings of the 11th IEEE Real-Time Systems Symposium*, pages 201–209, December 1990.

[17] OSEK/VDX. `http://portal.osek-vdx.org`.

[18] S. Perathoner, E. Wandeler, L. Thiele, A. Hamann, S. Schliecker, R. Henia, R. Racu, R. Ernst, and M. González Harbour. Influence of different abstractions on the performance analysis of distributed hard real-time systems. *Design Automation for Embedded Systems*, 13(1):27–49, 2009.

[19] PREEvision. `http://www.aquintos.com`.

[20] K. Richter. *Compositional Scheduling Analysis Using Standard Event Models - The SymTA/S Approach*. PhD thesis, University of Braunschweig, 2005.

[21] SymTA/S. `http://www.symtavision.com`.

[22] K. Tindell and J. Clark. Holistic schedulability analysis for distributed hard real-time systems. *Microprocessing and Microprogramming*, 40:117–134, 1994.

[23] M. Traub. *Durchgängige Timing-Bewertung von Vernetzungsarchitekturen und Gateway-Systemen im Kraftfahrzeug*. PhD thesis, University of Karlsruhe, 2010.

[24] M. Traub, V. Lauer, J. Becker, M. Jersak, K. Richter, and M. Kuehl. Using timing analysis for evaluating communication behavior and network topologies in an early design phase of automotive electric/electronic architectures. In *SAE '09: Proceedings of the conference of the SAE*, 2009.

[25] E. Wandeler. *Modular Performance Analysis and Interface-Based Design for Embedded Real-Time Systems*. PhD thesis, ETH Zurich, September 2006.

[26] E. Wandeler and L. Thiele. Real-Time Calculus (RTC) Toolbox. `http://www.mpa.ethz.ch/Rtctoolbox`, 2006.