Using Timing paths to validate end-to-end requirements with methods of schedulability simulation and analysis

Karsten Albers, Matthias Dörfel

INCHRON GmbH, Potsdam
{karsten.albers;matthias.doerfel}@inchron.com

Abstract. Requirements on end-to-end response times are essential for the correct behavior of complex cyber-physical systems. To ensure such requirements not only the start and end points like sensors and actuators are important but all functions and systems in between. For such requirements an adequate model is necessary to describe the flow of information involved in the requirements and allow to trace its change through all phases of the system design. The concept of timing paths and its application in method of schedulability simulation and analysis can be such a model.

Keywords: Timing path; scheduling simulation; schedulability analysis; worst-case response time

1 Introduction

The timing behavior and the definition of timing requirements are essential parts of state-of-the-art embedded systems like driver assistant systems in cars. To ensure an adequate reactivity of a system is not only important for the safety and a correct cooperation of the different systems but also for the good user impression.

In many applications the functionality is provided by complex cyber-physical systems consisting of functions distributed on different physical systems which are communicating using networks that could consists of different bus systems like CAN. FlexRay, Ethernet. For example a driver-assistant system “Lane-Keeping” will require sensors like a camera to get the vision of the environment, ECUs to preprocess and process this information with functions for pattern recognition and with functions to analyze the situation, ECUs to calculate an adequate reaction and actuators for performing the reaction. The responsiveness of the complete system will be distinct by the time between a critical situation in real-life and the reaction of the car on this critical situation. So all these mentioned functions and systems are part of this responsiveness and requirements on the responsiveness have to consider the whole path between the camera and the actuators.

For the development of cyber-physical systems it is beneficial to describe the flow of information behind the required responsiveness in an early stage and to consider its impact in every stage of development. The flow of information forms a path through

the distributed interacting systems with a set of different steps. Each step is connected to a certain function, part of a function or communication message which can themselves be bound to certain processors or bus systems. But not only the set of steps but also how the information flows between the steps is important for the path.

To ensure the timing behavior requirements the flow of information is necessary. For example an essential requirements is the reaction time from the occurrence of a certain situation, for example leaving the lane, to the time when the corresponding reaction occurs. Another example is to ensure that every single information, for example every camera picture is processed completely.

It is not sufficient to verify such requirements only in a late stage of development by testing the implemented system. Design decisions which a due early in the development depend on those timing requirements. They can cause the need for more ECUs or ECUs with a better performance or a different scheduling policy.

In this paper we will present a concept and the tool support for timing paths which enable the description and tracing of end-to-end timing behavior during the whole design process of embedded systems and supports the evaluation of those systems with the method of timing simulation and timing analysis.

The obvious way to deal with requirements like end-to-end worst-case response times is the distribution of the response time on the single parts. This is a simple concept which unfortunately can result in a significant overestimation and to tide limits for the local worst-case response times due to dependencies. Section 2 gives a closer insight into this problem.

An advanced model is the concept of event chains proposed in the timing extension (AUTOSAR Release 4.1) of the AUTOSAR standard. Here the timing path is described with a chain of pairs of request-response events. The response event corresponds to the following request event in the chain and so on, this allows to consider step-by-step the whole chain. The limitation of this model includes situations with overlapping instances of functions and backlog of data in queues. It is not always sufficient to consider only when a certain kind of data is requested, but also which concrete instance of data is used. Instances can get lost or a follow-up calculation can use the same instance multiple time due to missing updates of the data. Additionally the timing extensions, as a modelling standard, do not provide the tool support and the integration in the requirement engineering.

In (Kramer & Münzenberger, 2009) some preliminary remarks on the concept event chains can be found. The timing paths succeeds these concepts with a better foundation, a hierarchical concept and extended tool support.

2 Distribution of response times

A simple possibility for the refinement and verification of end-to-end response times is to decompose the required response time of the timing path and assign each single step its own share of the response time. When it is than possible to proof for each step that its specific share of the worst-case response time is always achieved, the end-to-

end response time, which is the sum of these shares, will also be reached. The concept of distributing the response time has several advantages:

- The concept is simple and easy to implement and comprehensible
- The single functions can be designed and analyzed independently of each other
- A contract-based design approach for requirement engineering is possible in an easy way. The response time of a contract for the overall system is provided by guaranteed response time coming from contracts of the single functions

Unfortunately, in most real cases, the distribution of allowed response times will lead to a significant overestimation. Consider, just as an example, a simple system with a timing path having, as steps, three consecutive tasks, which share the same processor. Let there be some high priority load on the processor which occurs sporadically.

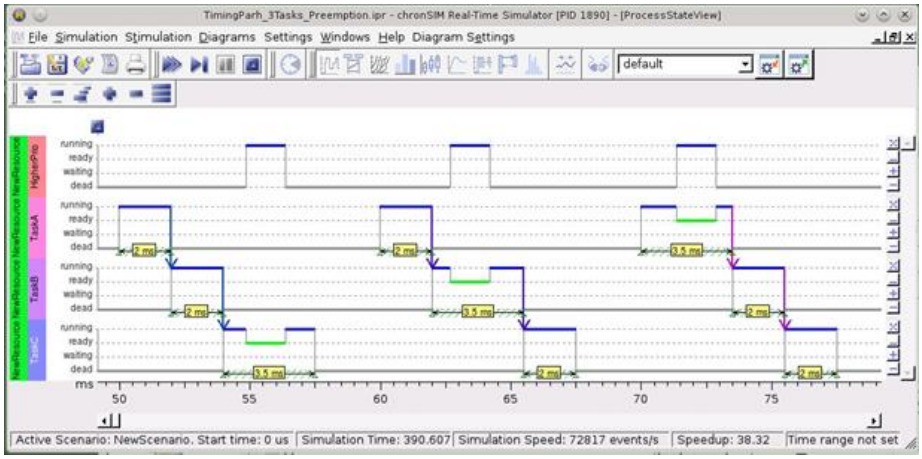

Fig. 1. Timing Path: Problem of Disturbing the Response Time

Fig. 1 depicts a simulation by the tool *chronSIM* of the system showing three (consecutive) occurrences of the timing path. In each instance another task will be preempted and will therefore get a large response-time. When distributing the end-to-end response time to the single tasks a share large enough to cover its worst-case response time has to be assigned to each task. But in many systems, like in the example, the higher priority load occurs in bursts with a distance between the bursts. When this distance is large enough, not all of the steps can be delayed by this higher priority load in the same instance of the timing paths, as in the example where always only one of the tasks is delayed and gets their worst-case response time. Therefore the worst-case response times for all three task cannot occur in the same instance of the timing path. But the sum of the worst-case response times will include three times the higher priority load while it can only occur once in each instance of the timing paths. So distributing the response time to the single steps will require an overestimation by two times the higher priority load in this small and simple example. There are also other dependencies leading to overestimations for the distribution of response times.

So better models and methods for describing and analyzing the end-to-end timing behavior are required.

3 Timing Requirements

Timing requirements can be affecting single systems or a set of communicating systems or functions. The correct overall behavior depends on the correct interaction of the single systems. Requirements can not only be worst-case or best-case response times. Other example for timing requirements affecting the end-to-end behavior are requirements on the frequency of successful evaluation or requirements which ensures the data consistency during the processing. They could also limit or forbid the loss of data instances during the processing.

An example requirement for an adaptive light assistant system can be:

If a camera detects an oncoming car, the upper beam will be reduced to lower beam within 0.5 seconds.

It would be not sufficient to transfer this requirement to a 500ms worst-case response-time requirement for the corresponding timing path. There will be a frequency in which pictures of the camera are taken and processed, this frequency needs to be taken into account for the 500ms reaction time. The time available for the worst-case response time of the timing path is therefore in fact less than 500ms, reduced by the maximum distance between two camera pictures to ensure the correct reaction time. This is based on the implicit assuming that the approaching car appears exactly after one picture was captured. If for example 250ms response time is enough for the timing path, the frequency for the pictures can be also 250ms and the overall deadline is still save. In this case only one picture would be processed at each point of time. If the timing path instead requires more time for processing the frequency for the pictures has to increase to keep the limit. Assuming 400ms response time requires to capture a picture every 100ms. This will also require to process 2.5 times more pictures (5 pictures concurrently) which would require more resources. Hence, choosing the right frequency early in the design process can substantially save resource. A lower frequency allows less processing capacity; still keeping all deadlines. In conclusion, it is essential to investigate the timing behavior of the planned system as early as possible.

4 The concept of timing path

A concept to describe the information flow and processing in a distributed cyber-physical system are the timing paths. A timing path consists of a sequence of steps, which are traversed in their specific order during the information processing. These steps can involve functions or interactions and are part of the functional network. In the underlying model the dependencies between consecutive steps can be given by a processor–consumer or an input–output relationship and the path can cover the complete way between the sensor and the actuators.

An example for a timing path coming from an adaptive light system, is depicted in figure 3. It covers the functionality of reducing the upper beam when a forthcoming

car is approaching. The timing path describes the complete way from the camera which captures the forthcoming traffic, over the processing of the captured picture, the decisions on the required reactions through to the actuator which reduces the light. The timing path has to cover steps like *image processing*, *object detection*, *object classification*, *decision on light reduction*, *processing of light reduction* and the messages between these steps.

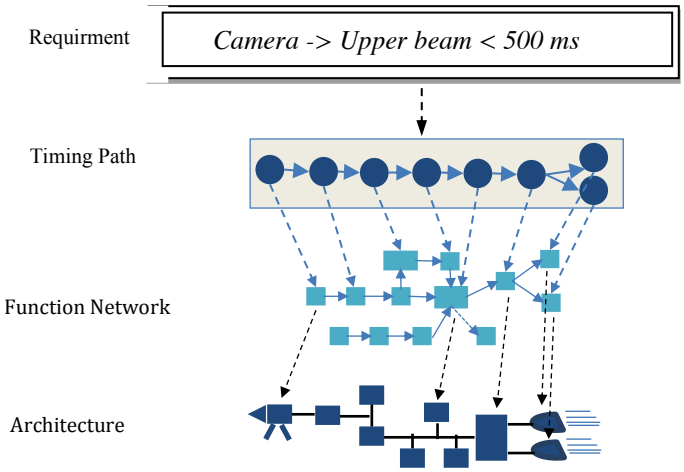

Fig. 2. The concept of timing path

A timing path is in principal defined within the combination of the functional design and particular timing requirements. In addition, a timing path can explicitly be documented as an analysis artifact. Timing requirements, like end-to-end response times, can be linked to the timing path, which depict the data processing that must adhere to the requirement.

The timing path separates the requirement and the functions designed to fulfill the requirement, allowing to refine and change the functions without changing the requirement. This separation supports the traceability of the requirements during the system development process. One function/message can appear several times in the same timing path. As the order of the steps within the timing path is fixed, several steps referencing the same function will not lead to infinite loops.

A timing path can have a hierarchical structure. It can be consist of consecutive connected sub-timing paths, which can also consists of several timing paths. The refinement of the functions can be reflected in the hierarchical timing path. When a function is refined in several sub-function the corresponding timing path is refined in sub-timing paths. The original timing path remains and requirements will still refer to the same timing path resulting in a hierarchical structure of timing paths and sub timing paths.

5 Tool support for timing paths

The concept of timing paths is implemented in the INCHRON tool suite[1] with the simulation tool chronSIM, the schedulability analysis tool chronVAL and the trace analysis chronVIEW, which therefore allows to investigate the timing behavior in all the different stages of system development. Timing plays a crucial role in the simulative validation of distributed cyber-physical systems. All embedded systems take input from their context and provide output to the context. Usually, this input-output behavior has some timing restrictions or the impact on the environment depends also on the concrete points in time when resulting values are available. In general a set of functions is executed on a target system. They compete with each other on the execution time available on the Electronic Control Unit (ECU) and on the communication time on the bus systems. Depending on the ECU and the functions, simple approaches like a fixed pre-calculated static schedule, simple scheduler using only some interrupts, up to powerful operating systems can be used to distribute the available execution times on the different functions. For a bus system with more than one sender some kind of scheduling is required to avoid collisions. The functionality of the systems is provided by a set of software functions, which communicate with each other and run on several heterogeneous ECUs with an interconnecting bus system.

Methods for simulating the timing behavior, as implemented in the tool chronSIM, use the available knowledge about the scheduling, execution times, and functional distribution to make predictions on the behavior of the later system. This includes the evaluation of timing requirements (see Fig 2). The simulation allows here to get from an overview, the detailed distribution of results up to the concrete behavior for single occurrences with all relevant influences. An early simulation indicates whether the timing requirements are reached by the considered system design, and, for example, which execution time budgets and scheduling constraints need to be fulfilled by the later implementation. The used models incorporate different abstraction levels. For some parts, only functions with estimated timing budgets exist, for other parts the same model includes the detailed behavior. The model can be abstract including, for example, interacting functions and information on the scheduling. It is possible to enrich and combine such a model with C-code describing functions and their communication. Messages defined in C-code are connected to an abstract model so that the simulation of the model can decide when a function is activated and executed (in the context of the simulation time). The code can also be used as a modeling language that provides the necessary flexibility and allows integrating different abstraction levels.

[1] http://www.inchron.com

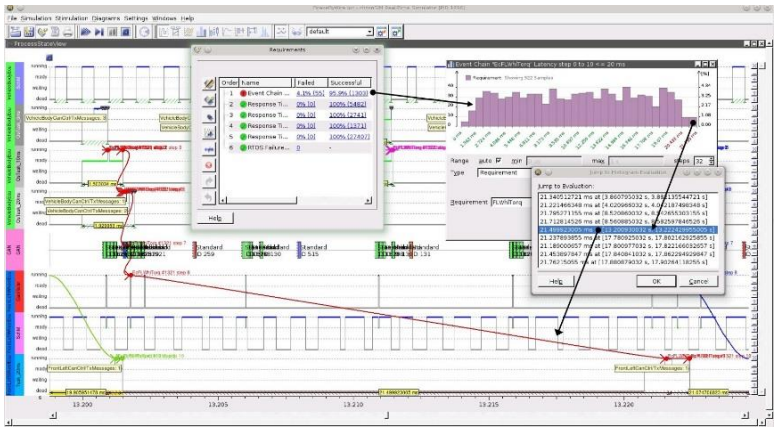

Fig. 3. Requirement evaluation with timing simulation

The early simulation or analysis of the system allows to consider the trade-off and to select the appropriate frequency and required response time.

The schedulability analysis tool chronVAL (Albers, 2011)works on an abstract model shared with the simulation and allows to calculate save upper and lower bounds on the timing behavior. Here also the complete distributed systems with their ECUs, interconnecting bus systems and the whole set of timing paths is taken into account. The analysis is based on the real-time calculus (Thiele, Chakraborty, & Naedele, 2000) but with several extensions to take particularities of relevant scheduling methods and dependencies like offset into account.

The concept of a timing paths enables a comprehensive and detailed analysis of the end-to-end system behavior. In early stages of the engineering-process it is of importance to consider information regarding the estimated response times, the data consistency, the jitter, and the robustness of the system-under-development. In particular, the evaluation of the requirements linked to the timing paths is essential to ensure a correct and safe system to emerge the development process.

Data consistency is another important aspect, which can be evaluated by analyzing the timing paths. During execution of a timing path data can get lost, can be processed multiple times, or can be delayed due to excessive queuing. The reasons can be based in the scheduling, in resulting jitter, in different activation rates, and in asynchronous communication. For example, in a timing path data can get lost when the data exchanged is realized by a shared variable; the value of the variable can be overwritten when the next instance of the same processing step is executed before the value of the variable is read in the following step of the timing path. These reasons can be a delay of the following processing step. Double processing of the same data can happen in the same situation if the variable is accessed twice before it was updated. Then the same calculation can occur twice and the same value is propagated in two executions of the timing path. The proposed method aims at detecting such data inconsistencies by analysis of the timing path. In particular, if there are multiple of such consistency problems within the same timing path only an appropriate simulation or analysis

method will clarify on which original data instances a result of the timing path is based.

6 Summary

The concept of timing paths, as introduced in this paper, allows to separate the end-to-end timing requirements from the network of functions and the system. This allows to trace the requirements through all different changes of the system. For real-life systems it is necessary to consider the timing path as a whole in the methods for analyzing the system. The distribution of response times to the single functions will not work as it will often lead to an overestimation. The introduced methods and tools for the simulation and analysis of real-time systems, chronSIM and chronVAL, considers the timing paths as a whole, take care of the concrete data instances within the timing path and therefore allow the adequate evaluation of timing requirements for the end-to-end behavior of embedded real-time systems.

The timing path allows a detailed and comprehensive investigation of the timing behavior and the fulfillment of the respective timing requirement by tools for simulations and analysis.

Acknowledgement

This research was partly funded by the German Federal Ministry of Education and Research (BMBF), under grant "SPES XTCore" (01IS12005I)

References

Albers, K. (2011). *Approximative Real-Time Analysis*. Dissertation, Ulm.

AUTOSAR Release 4.1. (n.d.). *Specification of Timing Extensions*. Retrieved from http://www.autosar.org/download/R4.1/AUTOSAR_SWS_OS.pdf

Kramer, T., & Münzenberger, R. (2009). Echtzeitverhalten simulieren und validieren - verstehen und absichern. *DESIGN & ELEKTRONIK Entwicklerforum*.

Kramer, T., & Münzenberger, R. (2009). New Functions, New Sensors, New Architectures – How to Cope with the Real-Time Requirements. *proceedings of Advanced Microsystems for Automotive Applications*.

Thiele, L., Chakraborty, S., & Naedele, M. (2000). Real-Time Calculus for Scheduling Hard Real-Time Systems. *IEEE Conference for Circuits and Systems*.