

Resource-Aware Control - Model-Based Co-Engineering of Control Algorithms and Real-Time Systems

Steffen Lampke and Simon Schliecker
Symtvision GmbH

Dirk Ziegenbein and Arne Hamann
Robert Bosch GmbH

ABSTRACT

The underlying theories of both control engineering and real-time systems engineering assume idealized system abstractions that mutually neglect central aspects of the other discipline. Control engineering theory, on the one hand, usually assumes jitter free sampling and constant input-output latencies disregarding complex real-world timing effects. Real-time engineering theory, on the other hand, uses abstract performance models that neglect the functional behavior, and derives worst-case situations that have little expressiveness for control functionalities in physically dominated automotive systems. As a consequence, there is a lot of potential for a systematic co-engineering between both disciplines, increasing design efficiency and confidence.

We have taken a standard control-engineering tool, Simulink, and combined it with state-of-the-art real-time system design and analysis tools, SymTA/S and TraceAnalyzer from Symtvision. We define a timing-aware control design workflow that enables the control engineer to consider actual target timing during controller layout and provides the system integrator with additional properties to exploit the design freedom during scheduling design and target integration. The workflow is enabled through a Simulink and SymTA/S toolbox which is applied to a realistic (active suspension) application. This shows that the approach allows uncovering integration problems long before the actual implementation is complete or the software is integrated - leaving the opportunity for efficient and goal-oriented planning of countermeasures.

CITATION: Lampke, S., Schliecker, S., Ziegenbein, D., and Hamann, A., "Resource-Aware Control - Model-Based Co-Engineering of Control Algorithms and Real-Time Systems," *SAE Int. J. Passeng. Cars – Electron. Electr. Syst.* 8(1):2015, doi:10.4271/2015-01-0168.

INTRODUCTION

The underlying theories of both control engineering and real-time systems engineering, assume idealized system abstractions that mutually neglect central aspects of the other discipline.

The control engineer typically focuses on designing an efficient and most of all stable control application, mainly considering plant dynamics and discretization issues. To that end, he relies on control engineering theory, which usually assumes jitter free sampling and constant input-output latencies disregarding complex real-world timing effects.

The real-time engineer (or system integrator), on the other hand, is concerned with integrating many different applications onto the same target platform. Thereby, he is not concerned with the functional behavior of the applications but rather with their implementations as communicating runnables, where there is typically a one-to-one correspondence between a Simulink subsystem and a runnable. Therefore, he reasons about mapping the runnables to tasks and cores, finding a suitable scheduling strategy and setting its parameters (such as priorities and execution orders). The basis for these

considerations are abstract performance models that neglect the functional behavior reasoning only about worst-case situations that have little expressiveness for control applications in physically dominated automotive systems.

As a consequence, the resulting system design may suffer either from over-dimensioning of the resources (and thus e.g. higher than needed costs per unit) or unfulfilled functional requirements, and thus potentially unstable applications. As both problems will surface only relatively late in the development process (during actual target integration), addressing them is today costly and may easily endanger project deadlines.

Establishing an early exchange of requirements and assumptions between the two disciplines thus has the potential to significantly increase design efficiency and confidence. To achieve this, the following is required:

- the information available in the scheduling and execution platform design should be made available to the control engineers as early as possible

- the control engineer should take into account the platform constraints and real-world timing constraints as early as possible
- the control engineer should be able to explore different implementation alternatives to increase the design freedom during the integration of the functionality on a target platform.

In the following, we describe our approach that enables such model-based co-engineering between control and real-time systems design. We have taken a standard control-engineering tool, Simulink, and combined it with state-of-the-art real-time system design and analysis tools, SymTA/S and TraceAnalyzer from Syntavision.

The strength of SymTA/S is that it offers a very detailed performance model covering all real-world timing effects in complex automotive systems, including, complex interrupt patterns, task chaining across cores, blocking effects due to cooperative scheduling, etc. Consequently, SymTA/S can be used to generate timing descriptions for complex control applications that consist of several communicating runnables that are distributed over several tasks and cores.

Using a Simulink toolbox that we developed, it is possible to combine these timing descriptions with the functional model of the control application. This enables true co-engineering between control and real-time systems design. For instance, already on model level the control engineer can analyze the functional effects of real-world timing on his control application, and e.g. optimize the design w.r.t. robustness. Alternatively, the system integrator is enabled to quickly explore the space of possible OS configurations, e.g. optimizing the runnable order in a task to minimize the activation jitter for a control loop of a highly dynamic physical process.

The paper is structured as follows: The next chapter introduces the use cases that the solution aims to address and the main intended workflow. We then discuss related work. The following chapter briefly introduces the established tools that are the basis for our approach and explains the technical realization of our approach. We have applied the new approach to a set of case studies with different complexity, one of them - a suspension controller previously introduced in related work - will be discussed in this paper. Finally, we conclude.

USE CASES AND BASIC WORKFLOW

To focus our efforts, we have identified a set of representative use cases that should be achieved with our toolbox.

- UC1: The control engineer is enabled to check the quality of his control designs in realistic execution scenarios, considering real-world timing effects such as execution and scheduling delays.
- UC2: The control engineer receives better tools to derive explicit timing requirements that are needed to guarantee the desired control quality.
- UC3: The control engineer is enabled to explore suitable and optimized alternative control design concepts and their robustness under realistic execution conditions.

- UC4: The system integrator can work with explicit timing requirements and implementation options to exploit the design freedom and optimize the OS configuration (choose runnable cycle times, systematically decide about runnable orders, etc.)
- UC5: The system integrator can rely on explicit timing requirements and implementation options to allow an easy migration of control applications to different target platforms (including multi-core).

To implement these use cases, we have created a Simulink real-time toolbox that represents a flexible link between the control modeling (in Simulink) and the scheduling design (in SymTA/S). The goal is to expand the current design workflow without major modification of established practices in either engineering domain. For instance, the starting point is that the control engineer develops a control function according to his or her established design process. How this has been achieved is explained in the chapter “Implementation”.

As a reference, we have defined the “basic timing aware control design workflow” illustrated in in [Figure 1](#). It consists of the following steps:

1. When the initial control design has reached a certain maturity, the control engineer extracts the set of subcomponents in his model and sends this list to the system integrator.
2. The system integrator takes the existing performance model of the target hardware/software platform as a basis (if exists) and adds the new subcomponents as runnables with according cycle times.
3. The performance model is then analyzed, and the system integrator extracts the timing descriptions for the different runnables and their communication dependencies.
4. These timing descriptions are provided to the control engineer, who annotates her model with these descriptions using our timing toolbox explained below.
5. He then simulates again. As the execution time and data ages are now considered, the control engineer can now realistically assess the control quality.

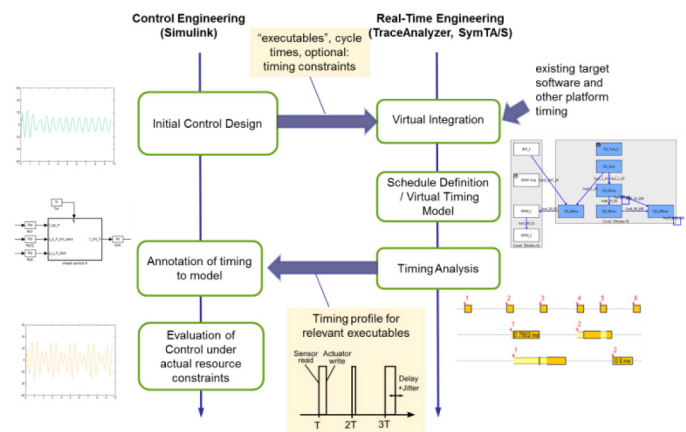


Figure 1. Basic Timing Aware Control Design Workflow

The establishment of this workflow alone provides a large efficiency increase as the design iterations between control designer and scheduling architect will happen much earlier. In typical current workflows, their first interaction happens rather late in the rapid control prototyping (RCP) phase which already requires fully functional ECUs and testbenches. Due to this front-loading, the proposed workflow enables several options for optimizations and iterations that further improve the design quality.

RELATED WORK

Of course, we are not the first to recognize the need of integrating real-time scheduling approaches with functional behavioral models. One example is the work on the TrueTime toolbox [1] that enhances Simulink with scheduling simulation for simple scheduling policies. This enables function developers to assess the functional behavior of control applications under temporal effects that are due to real-time scheduling (latency and jitter) already on model level. One drawback of the TrueTime approach is that the supported basic scheduling policies do not cover realistic timing effects in automotive systems that are due to complex interrupt patterns, task chaining (across cores), mixed preemptive and cooperative task scheduling, etc. Also, the TrueTime approach requires modeling the concrete platform configuration directly in Simulink. In other words, each explored platform alternative requires adapting the functional model, a task a control engineer shall not be bothered with. In our approach, generic timing blocks are introduced such that the functional Simulink model stays agnostic to platform and scheduling changes. By this means, a strict separation of concerns between control engineering, focusing solely on controller design, and real-time engineering, focusing on platform configuration and scheduling, is achieved.

Another approach in this line of research is implemented in the JitterBug toolbox [2]. Here, timing effects are described using probabilistic models that can elegantly be integrated into control engineering system theory to prove stability and control performance. The drawback is that probabilistic models are rarely adequate to describe the timing behavior in automotive systems. The main reasons are unknown dependencies due to digital hardware mechanisms such as caching and pipelining which are impossible to capture.

Related to our approach is the work on cyber-physical system design contracts [3]. Also here, the vision of establishing a co-engineering approach between control engineering and real-time systems design is pursued. A CPS contract models the latency and jitter that the system integrator guarantees for a specific control task. The control engineer can take these guarantees into account during controller design to choose an adequate control strategy. In case he fails to find a stable controller, the authors propose an iterative negotiation process between control engineer and system integrator.

Also SymTA/S has previously been used to derive the actual delays (in particular worst-case delays) of the control applications running on target hardware [4]. There, annotating these delays into the Simulink model has been suggested but not worked out as provided

by the present approach. The topic is also visited in [7] which discusses the conceptual gap in the perception of timing requirements as defined by control designers and those defined and inspected by the system integrator. The introduction of delay elements during control design is proposed to bridge this gap. The following approach goes one step further and allows linking the controller simulation in Simulink to the actual timing of the real or virtual target platform.

IMPLEMENTATION

The main tools that our approach relies on are Simulink (for the control design and analysis) and SymTA/S (for the design and analysis of the platform scheduling and timing).

Simulink

Simulink is a graphical interface for the numerical solvers of MATLAB specialized in modeling differential equations of dynamic systems. For control engineering this is used to simulate the behavior of continuous or discrete controllers together with the plant representation under the influence of test functions. Digital controllers in Simulink are modeled as time discrete systems with zero execution time and hence zero sensor to actuator delay as well as a strictly constant sampling period. All of this is not true in real scheduled systems. However, this corresponds to the z-transform in which such systems can be studied well theoretically. The only modeled timing parameter is the sampling period respectively in the real-time terminology the cycle time.

SymTA/S

SymTA/S is a performance analysis tool for embedded real-time systems. Based on models of the software and scheduling architecture it provides worst-case and statistical timing analysis for single ECUs (uni- and multicore) as well as complete networks encompassing functions distributed over several ECUs connected via a shared bus (such as CAN, FlexRay), gateways or switched networks (such as Ethernet).

Typical results of the timing analysis are CPU load, task response times, and end-to-end latency. The base for deriving these metrics is a) the static scheduling architecture (such as the tasks' priorities, periods, offsets and the runnables' cycle times) and b) their dynamic properties (such as the pattern of their activating events and their net execution times). While the static design information is typically available in a structured fashion (e.g. an AUTOSAR system configuration file) and can be directly imported, processes for gathering dynamic design data are less established. Here SymTA/S allows working with time budgets, measurements, or values derived from formal analysis. The simplest way to measure the net execution times of runnables for example is by logging time stamps at each context switch. Such trace data can then be analysed for best- and worst-case timing and fed into the model.

The timing metrics now have to be matched to the control applications' requirements. This can be done via the proposed co-design approach.