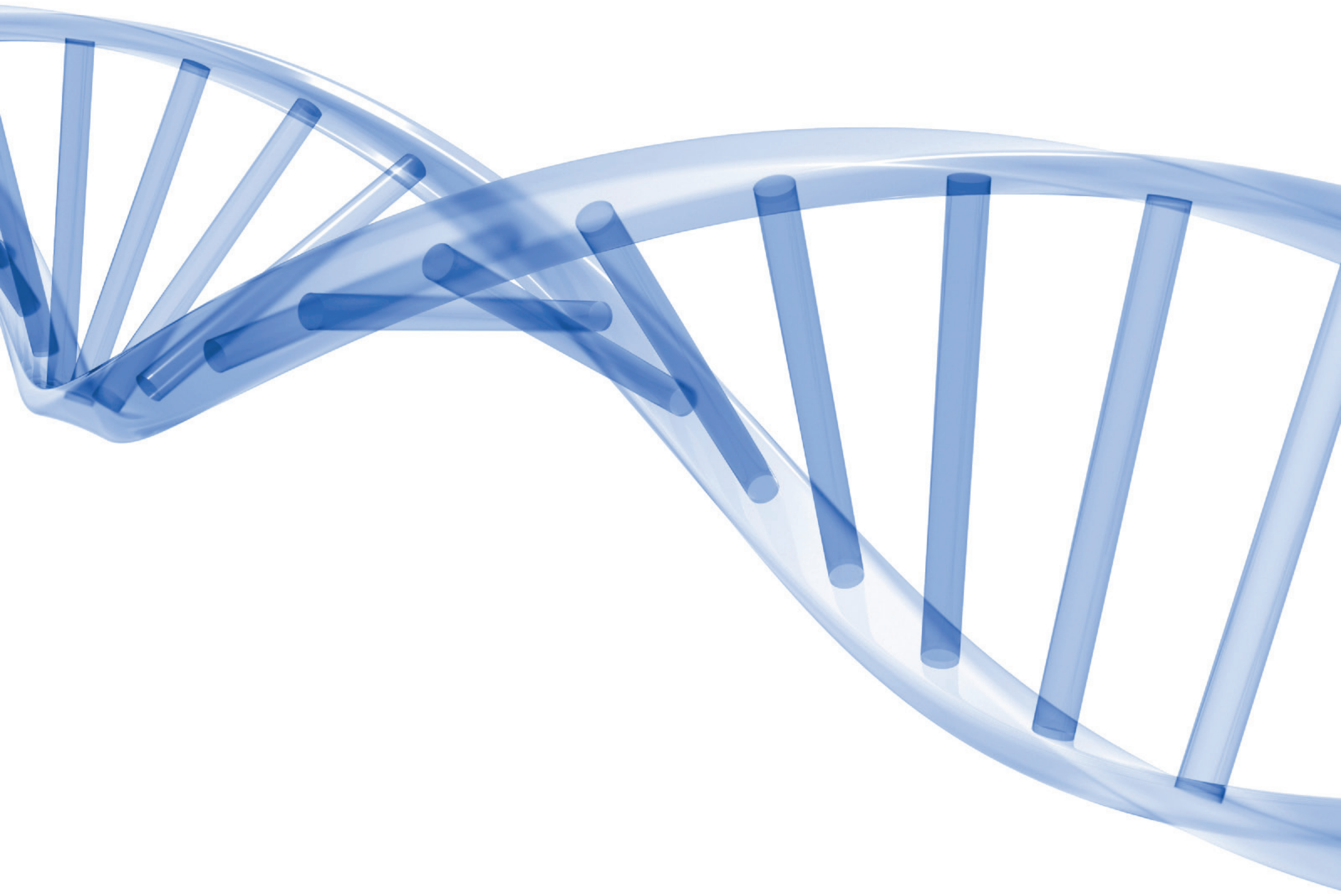


Timing-Architects Embedded Systems GmbH

Timing Simulation of Multi-Core based AUTOSAR Models

Andreas Sailer

www.timing-architects.com

Introduction

With growing success, AUTOSAR has been established as the de facto industry standard for the development of automotive E/E architectures. Furthermore, the use of timing simulation for the evaluation of a system's behavior, especially with the introduction of multi-core systems, gets increasingly popular. Consequently, the question arises: Can all information, needed for a conclusive timing analysis, be obtained from an AUTOSAR model or are there crucial pieces missing?

The automotive industry is facing increasing demands for more sophisticated functions, e.g. comfort features, increasing safety requirements or the "internet of things" in smart electronic devices. To handle this demand, more than 100 million lines of code are mapped to many ECUs (Electronic Control Units), which can number up to about 100 units for high-end cars [1]. This number of ECU leads to an ever increasing complexity resulting in greater engineering effort, additional weight and restricted flexibility for updates and modifications. With the ongoing shift towards multi-core systems, the automotive industry has found a solution not only for counteracting this development, but also for leveraging additional processing power for new functionality. However, most existing applications cannot benefit from the new multi-core architecture without significant modifications due to the concurrent nature of the new processor technology. Therefore, multi-core has a huge influence on the complexity that has to be handled during development. Each additional core for example, increases the number of possible task allocations exponentially. At the same time, task allocation has direct effects on real-time properties, efficiency- and data-communication overhead. For that reason, over the last years, timing simulation evolved into an inherent part in the development of software systems for multi-core architectures [2].

What is a Timing Simulation?

As the name implies, a timing simulation forecasts the dynamic behavior of a software system by simulating its scheduling behavior. The simulation is based on a model, which is basically an abstract description of the system under investigation. The necessary information on the system which is provided by this model covers in general the following parts:

› Software
  Represents an abstract description of the application software. This means primarily the tasks in the system, their properties like priority or pre-emptability, and all its runnable entities, including its execution condition. Runnables themselves are described by properties like the execution time consumed, the communication APIs called as well as the data signals and resources accessed.

› Hardware
  Represents an abstract description of the hardware, especially of its internal procedures. In this description the modeled level of detail can vary by the number of processing cores and its processing performance up to detailed memory topology and behavior descriptions including memory modules, caches, bus networks or crossbars.

› Operating System
  Represents an abstract description of the kernel. In general this includes the specification of the schedulers including their scheduling algorithm as well as the management of resources like semaphores, event-notification and certain protocols like priority ceiling.

› Runtime Environment (RTE)
  Represents an abstract description of the communication backbone between different cores and to network interfaces as well as hardware abstraction functions.

› Environmental Stimulation
  The Environmental Stimulation represents an abstract description of external inputs, interacting with the system.

One of the big advantages, which were conducive to the popularity of timing simulation is, that the level of abstraction, represented by the model, is not fixed but can be varied and adapted to the information at hand. This enables the evaluation of a system's behavior at all stages of the development process. In an early development stage, for example, a timing simulation can help to estimate the future processing demand of the software system. Already, this vague information on the system under development can be used to make design decisions, e.g. regarding the required hardware or if already available hardware platforms satisfy resource needs. With an increased level of detail, available at later steps in the development, the simulation model can be refined step by step. Hereby the behavior of the real system can be evaluated more precisely with each piece of information.

„Timing simulation gives insight in the system's behavior and helps to evaluate the system at all stages of the development process."

About AUTOSAR

AUTOSAR (AUTomotive Open System ARchitecture) [3] is standardized software architecture for the automotive industry. One of its main goals is to enable the integration of software from different suppliers in order to increase the functional reuse. Therefore modular software architecture was defined, where software functions can be implemented independently from the underlying hardware. In AUTOSAR this decoupling is even point of view it doesn‘t matter whether its communication partner resides on the same ECU or somewhere on the other end of the vehicle as long as no timing constraints are violated. The same holds for more detailed multi-core mapping decisions. To achieve this, the individual software functions are encapsulated in so called components. Each component however can communicate with each other

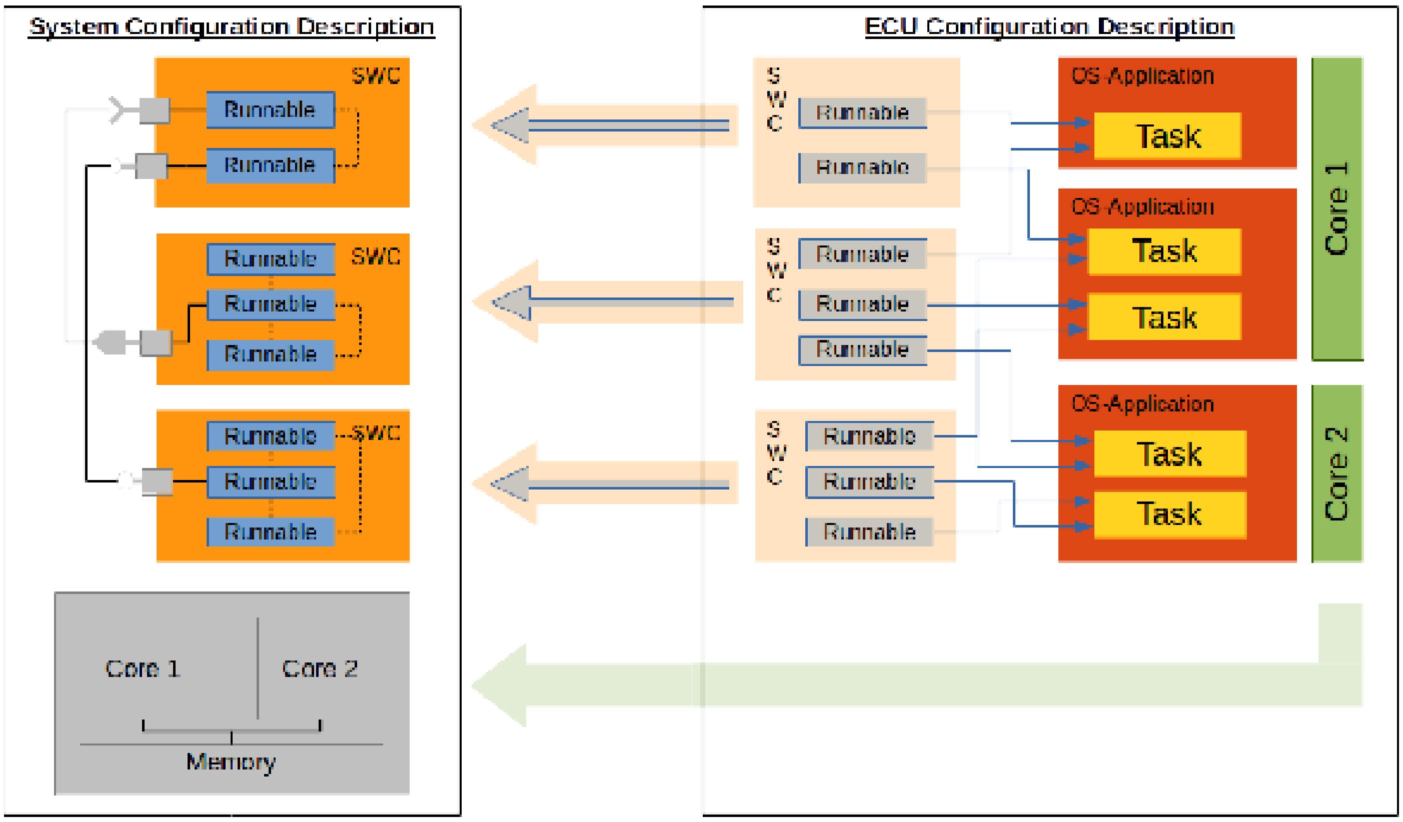


Figure 1: Responsibilities of AUTOSAR Configuration Descriptions

getting to a point where the developer doesn‘t have to be concerned with the final mapping of the software to the hardware e.g. while implementing the communication with other software functions or the environment. This means that from the software function or the operating system only through standardized interfaces. To make this interoperability possible, the

„AUTOSAR consolidates necessary information in the System and ECU Configuration Description“

software architecture has to be configured by providing information on each software component. These required pieces of information are collected , as depicted in Figure 1, within the following AUTOSAR models:

› **System Configuration Description / ECU extract of System Configuration**
  Contains an abstract description of the software components, ECUs and system constraints in the whole system or a specific ECU respectively.

› **ECU Configuration Description**
  Contains concrete values for all configuration parameters of the basic software including operating system and runtime environment for a single ECU.

Model Transformation

In order to manage the collection of all those pieces of information required for a system configuration , AUTOSAR also provides a description of development steps that have to be executed during system development. Consequently, models also represent a fundamental part in the AUTOSAR methodology [4]. This rises the question of whether or not all the information needed for a conclusive analysis can be transformed from an AUTOSAR model to a simulation model? In the following, we want to respond to this question by giving an overview of essential simulation model elements that can be taken from an AUTOSAR model. An overview of the intended can be seen in Figure 2, where a timing simulation fits seamlessly within the AUTOSAR Methodology. Although the latest available release of AUTOSAR is 4.1, this information is based on AUTOSAR 4.0 Rev 3. The reason for this is, that it is more common in the industry then the latest release, which was released as late as March 2013. Compared to previous releases (AUTOSAR releases 3.x) however, it introduced two significant innovations: the support of multi-core architectures as well as the possibility to define timing constraints. The available information is grouped based on the work product it can be found in. That way it is also possible to make a statement about the point in time this information is available in the development process.

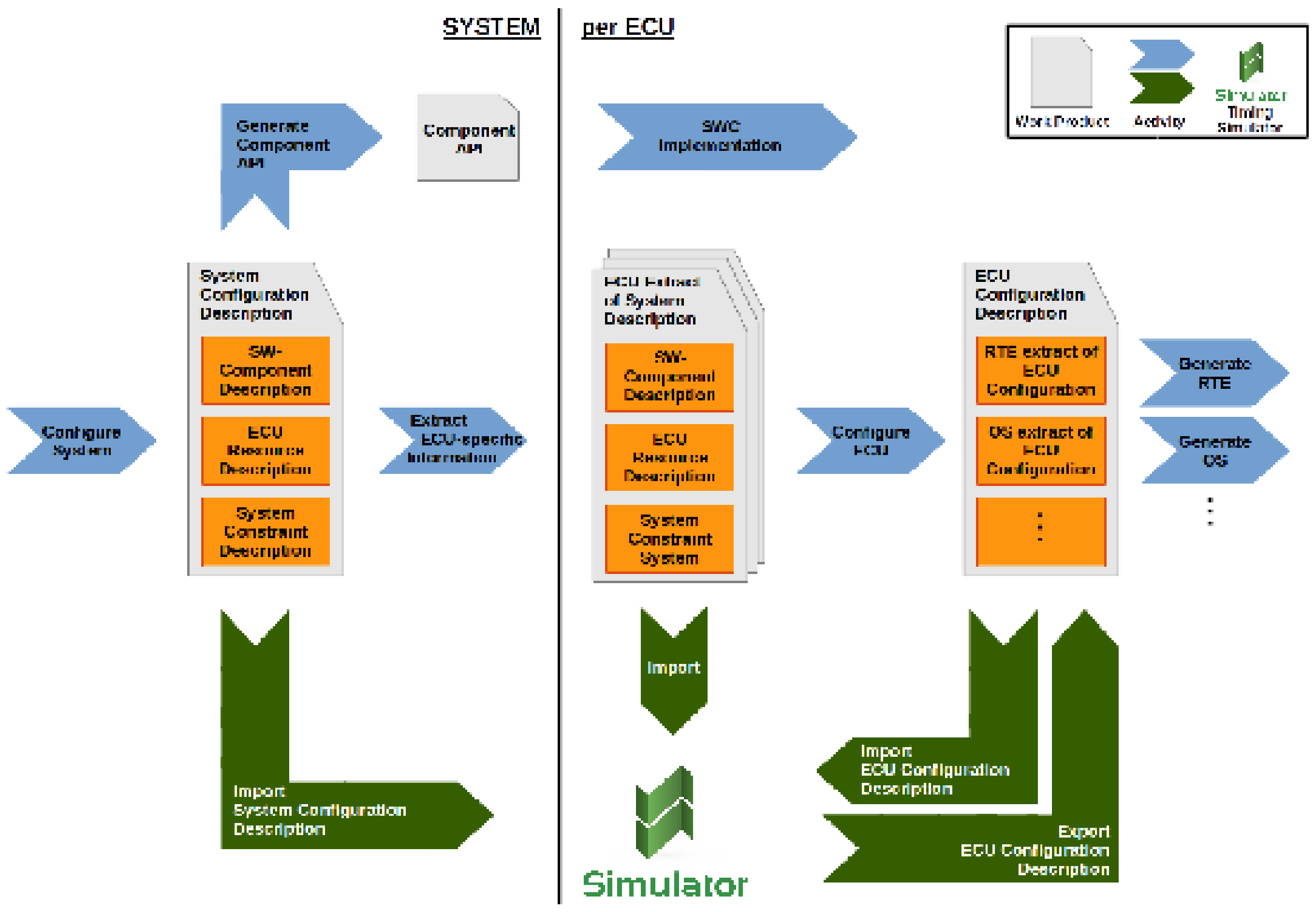

SYSTEM
per ECU
Work Product
Activity
Simulator Timing Simulator
Generate Component API
Component API
SWC Implementation
System Configuration Description
SW-Component Description
ECU Resource Description
System Constraint Description
Configure System
Extract ECU-specific Information
ECU Extract of System Description
SW-Component Description
ECU Resource Description
System Constraint System
Configure ECU
ECU Configuration Description
RTE extract of ECU Configuration
OS extract of ECU Configuration
Generate RTE
Generate OS
Import
Import System Configuration Description
Import ECU Configuration Description
Export ECU Configuration Description
Simulator

Figure 2: Use Cases for Timing Simulation within the AUTOSAR Methodology

"AUTOSAR 4.0 introduced two significant innovations: support of multi-core architectures and timing constraints"

System Configuration Description

› Runnables

In AUTOSAR a Runnable Entity represents a sequence of instructions that can be executed and scheduled independently. Thus, it is comparable to a representation of a function. Each software component consists of a set of Runnable Entities. Due to the fact that those have to be mapped to a particular ECU, the necessary Runnable Entities can be extracted from a given System Configuration Description. Furthermore, for each Runnable Entity the data accesses can be determined. AUTOSAR knows two different communication paradigms: sender-receiver and client-server communication. Those paradigms along with their approaches, implicit and explicit in case of sender-receiver communication and synchronous and asynchronous for client-server calls, can be transformed into corresponding read/write data accesses and function calls. In order to protect those data accesses within a Runnable Entity, AUTOSAR provides the constraint of an Exclusive Area. This concept can then be transformed into a resource protection, e.g. by using semaphores. Information about the dynamic behavior can also be stored within a System Configuration Description. It is, for example, possible to state the execution time of a specific Runnable Entity implementation. This can be done for different levels of quality. Starting with a rough estimation of the execution time up to the precision of an analyzed execution time, where an analytic method was used to find guaranteed boundaries.

"Runnable Entities define the internal behavior: data accesses, resource protection and time consumption"

› Runnable Execution Sequence

Although Runnable Entities are bundled in software components, no information about their execution order is provided. However, with the help of Execution Order Constraints, which were introduced with the Timing Extensions in AUTOSAR Release 4.0, mandatory sequences of Runnable Entities can be defined as early as in the System Configuration Description [5].

› Timing Constraints

With AUTOSAR Release 4.0, the Timing Extensions found their way into the standard. Their purpose is to provide a consolidated and consistent way to represent relevant timing dependencies and constraints. This enables the analysis of a system's timing behavior and the validation of the analysis results against timing constraints throughout the development process. Depending on the availability of necessary information in each development phase, the so called views, different timing constraints are available to describe the timing behavior of an AUTOSAR system. This can for example be the maximum repetition time between the starts of a Runnable Entity or the minimum distance between subsequent data accesses within a given time interval. Additionally, it is not only possible to specify particular occurrences of events but also sequence relationships between those during the runtime of a system by using Event Chains. That way it is possible to specify the timing behavior of arbitrary scenarios within the whole system. A popular application is for example the specification of data-flow from input over multiple Runnable Entities to output.

"AUTOSAR Timing Extensions enable a comprehensive analysis and validation of the system's timing behavior."

› Hardware

In AUTOSAR it is also possible to provide a description of the relevant aspects of the actual hardware. In a hierarchical manner, the individual hardware elements of an ECU and their interconnections can be described. For that, a set of categories for hardware elements are predefined, which then provide dedicated attributes. That way it is for example possible to determine the number of processing units within a micro-controller.

ECU Configuration Description

› Tasks

As with the Runnable Entities, the ECU Configuration Description contains a separate entry for each task. The parameters of this entry include information on its priority, the maximum number of queued activation requests as well as the preemption of that task. The Task-To-Core Allocation is in AUTOSAR specified by the OS-Applications. With an

OS-Application a block of software including tasks, interrupts are bound into a cohesive functional unit. The special characteristic of such an OS-Application however is that it is mapped to a specific core as a whole. That way, the Task-To-Core Allocation for each task within that OS-Application is specified.

› Runnable Mapping

For every Runnable Entity in a software component, a separate entry within the ECU Configuration Description exists. There, the necessary parameters for a Runnable Entity are set. This part, for example, defines the mapping of Runnable Entity to Task. Another parameter states then the position within that task. In addition to that, another parameter can specify, whether a Schedule Point follows the execution of this Runnable Entity.

› Stimulation

Activation patterns for each Task are specified by the so called Schedule Tables. In order to get a notion of time, each Schedule Table is driven by a counter. Expiry Points define then the point on a Schedule Table at which the operating system activates Tasks and/or sets Events. In addition, a parameter states, whether the Schedule Table starts again from the beginning after the final Expiry Point was processed.

› Hardware

Information on the hardware is contained in the ECU Configuration Description just indirectly. For example the number of processing cores available for that ECU can be determined from a parameter, which however is optional.

Wrap-Up

Although this was just an overview, it has been shown that big parts of information, which are essential in order to derive a simulation model, already, are present within an AUTOSAR compliant development process. However, due to the initial intention of AUTOSAR to mainly support the static software integration, a few limitations to fully support timing simulation are present. Certain dynamic properties, reflecting e.g. detailed timing behavior are still missing which are described in the following:

› Dynamic Behavior of Runnable Entities

Signal accesses as well as their protection can be stated within AUTOSAR. The model however still lacks crucial timing details. There is, for example, no information on how often or under which condition a data signal is accessed by a Runnable Entity, just that it is accessed at all. The model also makes no statement about the chronological order of the actions within a Runnable Entity. Consider for example a model with a Runnable Entity, having a data read and data write accesses. On the one hand, it is not clear whether the data signal is read or written first. On the other hand, it is also not known if there are multiple read or write accesses to that data signal. The same problem occurs in cases, where Exclusive Areas are used. AUTOSAR specifies in that case only that Runnable Entities with the same Exclusive Area shall not run concurrently. But in cases, where the Exclusive Area does not protect the entire Runnable execution, there is no information in the model about when exactly that Exclusive Area is entered or left. For this topic, reasonable assumptions like read before write can help in a first approximation.

› Dynamic Behavior of Operating System and RTE

For Runnable Entities, execution times can be specified. However, no information about the OS and RTE execution time can be modeled within AUTOSAR. Although there are ways to state the resource consumption of application software, it would also be necessary to have information about the overhead produced by the kernel itself as well as the communication layer in order to perform an exact simulation. To address this topic, operating system vendor and simulation tool vendor have to exchange details on the runtime behavior of the operating system. The simulation tool vendor is then able to integrate this overhead information in the simulation and can provide more detailed analysis on the runtime behavior.

› Hardware Description

Although AUTOSAR provides a way to describe the relevant aspects of the actual hardware, this is done in a very generic way. The flexibility of individual, but non-standardized, attributes restricts the generic usage over different tool vendors. For example considering the predefined category processing unit which represents a micro-controller core. By default, AUTOSAR defines no attributes for this category. As a consequence, properties like the processing frequency of a core have to be defined by generic attributes. This however can lead to problems with different interpretations and naming of these attributes. As a consequence, this flexibility was gained at the expense of the lack of an automatic exchange between different tool vendors. For this topic, inspirations could be gained from other industry consortium, e.g. Multicore Association [6].

Conculsion

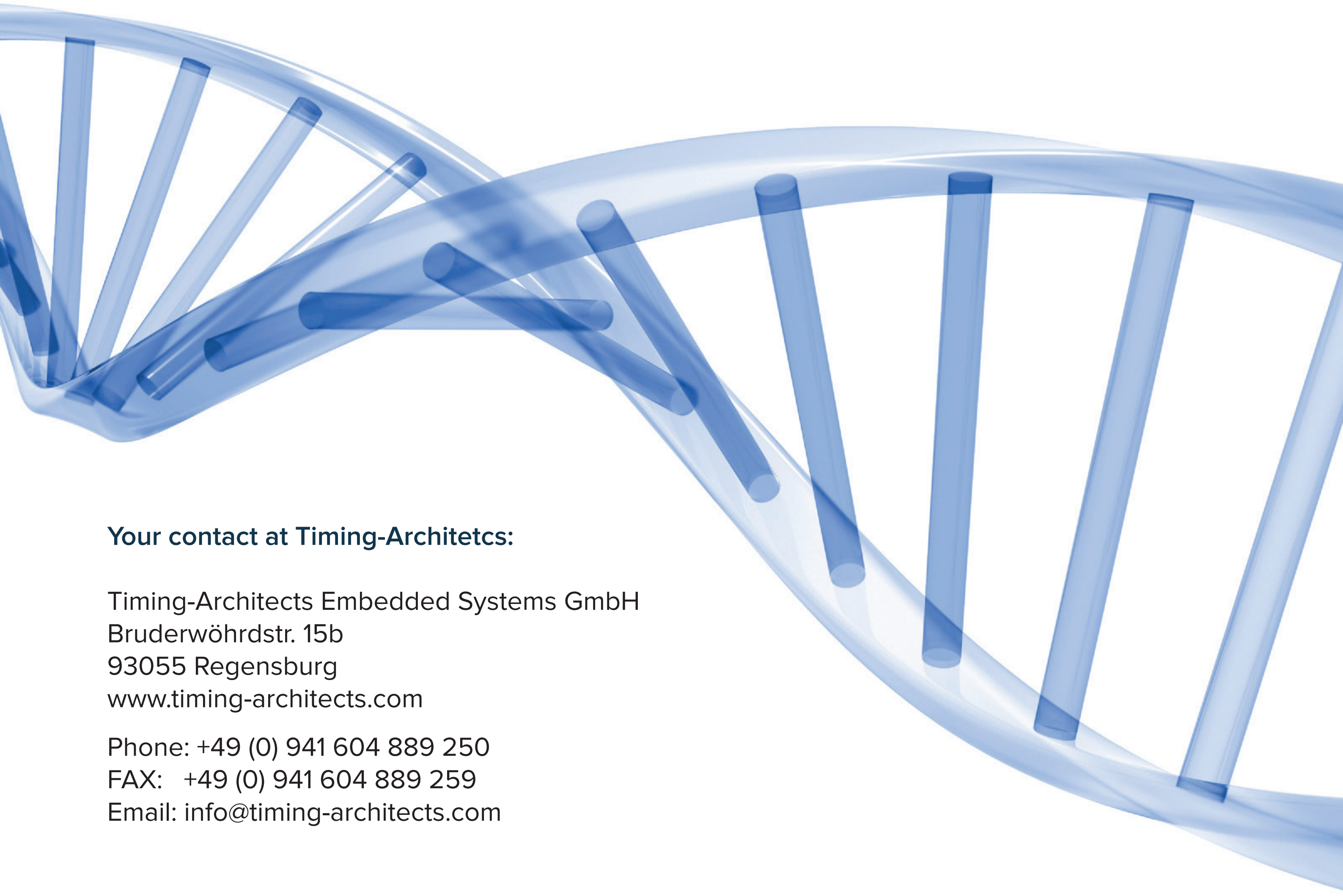
This article reflected the capabilities of AUTOSAR providing detailed timing information as well as deficits of the standard. As a conclusion, it can be stated that it is possible to apply a beneficial timing simulation based on a maintained AUTOSAR description. Open points can be handled in practical application by reasonable assumption and minor reviewed extensions (Annexes) between vendors. These extensions can be contributed in a next step to the AUTOSAR standard.

About the Autor

Andreas Sailer is a Ph.D. student at the University of Bamberg. He received a master‘s degree in Computer Engineering from the Ostbayerische Technische Hochschule (OTH) Regensburg, where he currently works as a research assistant for the ITEA2 research project AMALTHEA4public. Besides his activity there, he also functions as a technical consultant for Timing-Architects. His research focuses on topics regarding model-based development and timing simulation.

About Timing-Architects

Timing-Architects (TA) is an international operating high-tech company for development tools and methods, specialized in the optimization of software for embedded multi- and many-core real-time systems. For their innovative tool solution “TA Tool Suite” the company received multiple awards and is also recognized by experts as a leading player in research and development of multi-core embedded systems. With their tools supports project managers, architects, developers, integration and test engineers, to broaden the performance and increase the efficiency of their multi-core projects.



Your contact at Timing-Architetcs:

Timing-Architects Embedded Systems GmbH
Bruderwöhrdstr. 15b
93055 Regensburg
www.timing-architects.com

Phone: +49 (0) 941 604 889 250
FAX: +49 (0) 941 604 889 259
Email: info@timing-architects.com

Timing Architects