chronVAL/chronSIM: A Tool Suite for Timing Verification of Auto-motive Applications

Saoussen Anssi, Karsten Albers, Matthias Dörfel, Sébastien Gérard

chronVAL/chronSIM: A Tool Suite for Timing Verification of Automotive Applications

Saoussen Anssi[1], Karsten Albers[2], Matthias Dörfel[2,] Sébastien Gérard[3]
[1]Continental Automotive France SAS, PowerTrain E IPP
1 Avenue Paul Ourliac - BP 83649, 31036 France
saoussen.ansi@continental-corporation.com
[2]Inchron GmbH, August-Bebel-Str. 88, 14482 Potsdam, Germany
{karsten.albers, matthias.doerfel}@inchron.com
[3]CEA LIST, Laboratory of model driven engineering for embedded systems,
Point Courrier 94, Gif-sur-Yvette, F-91191 France
sebastien.gerard@cea.fr

Abstract: Automotive software systems become more and more complex presenting tougher safety requirements and tighter timing constraints. On the other hand, today, timing verification of automotive systems is considered late in the development process, mainly at the implementation stage. Consequently, this leads generally to late detection of design mistakes involving extra costs. In this paper, we present a tool suite dedicated for timing analysis of automotive systems, chronSIM/chronVAL. We propose to evaluate the capabilities of this tool suite against the timing analysis needs of automotive applications.

Keywords: Automotive applications, scheduling analysis, scheduling analysis tools requirements & capabilities, chronSIM/chronVAL.

1. INTRODUCTION

Today, embedded automotive systems often involve hard real-time constraints intended to ensure full system correctness [1]. Power train and chassis applications, for example, include complex (multi-variable) control laws, with different sampling periods, for use in conveying real-time information to distributed devices. One hard real-time constraint controlled in power train applications is ignition timing, which varies with engine position. The latter is defined by a sporadic event characterizing the flywheel zero position. End-to-end response times must also be bounded, since a too long control loop response time may not only degrade performance, but also cause vehicle instability. These constraints have to be met in every possible situation.

Automotive software development costs are sharply impacted by wrong design choices made in the early stages of development but often detected after implementation. Most timing-related failures are detected very late in the development process, during implementation or in the system integration phase. Timing verification is usually addressed by means of measuring and testing rather than through formal and systematic analysis. For this reason, innovative and complex functionalities are not implemented in a cost-efficient way[1]. The needs for techniques and tools that permit early and accurate timing verification for automotive systems are thus obvious. Such techniques and tools would enable early prediction of system timing behavior and allow potential weak points in design to be corrected as early as possible.

To perform timing verification for automotive systems, approaches based on both analytical and simulation techniques are good candidates. In this context, we aim in this paper to verify to what extent a particular tool suite for timing analysis: chronSIM/chronVAL allows performing accurate timing analysis for automotive systems. This tools suite is based on both analytical verification (chronVAL) and simulation of timing behaviour (chronSIM).

The paper is organized as follows: In section 2, related work is presented as a general overview about analytical techniques, mainly scheduling analysis techniques and tools, developed in the context of real time verification. Section 3 is dedicated to the characterization of various aspects of automotive applications and inventorying their scheduling needs. This inventory serves for determining the requirements that scheduling analysis tools should meet. Section 4 gives a presentation of the studied tool suite. Section 5 highlights the capabilities and limitations of the tool suite with regard to the determined requirements. The conclusion summarizes the study and examines perspectives for use of the INCHRON tool suite within the automotive software development process.

[1] These statements are based on the study of current automotive software development practices and particularly in the case of Continental

2. RELATED WORK

During last decades many techniques and approaches have been developed to enable timing verification for real time systems. In this context analytical techniques, such as schedulability analysis are good candidates for analyzing non-functional properties at early design stages.

The first exact schedulability test for the preemptive monoprocessor scheduling of a set of periodic tasks, each with its deadline equal to its period, was introduced by Lehoczky et al. [2]. The test determines whether a set of tasks is schedulable using the rate monotonic algorithm of Liu and Layland [3]. The response time of each task is calculated, and checked against its deadline. Later, other tests were developed relaxing a number of assumptions: Audsley et al. [4] developed a feasibility test for sets of tasks in which deadlines can be less than periods, and which are scheduled using the deadline monotonic algorithm [5]. Lehoczky [6] provided a feasibility test for periodic tasks with arbitrary deadlines. For distributed systems a number of tests have also been developed, e.g. [7], [8] and [9]. To take into account dependencies between tasks, Tindell proposed in [8] a test for fixed priorities in which offsets among release times of dependent tasks can be taken into account. The test has been later extended to distributed systems by Palencia and González [10].

Another approach for real-time analysis is the real-time calculus. It was developed by Thiele et al. ([17], [18], [19]) and is based on the network calculus ([22], [23]). There interval-based curves are defined with a set of basic functions which are combined to the complex functions required for the analysis. The properties are modeled by curves and the method provides an algebra based on these curves.

The stimulation of the tasks and functions are modeled with upper and lower incoming event curves and the resources available for the tasks are given by available upper and lower capacity curves. The real-time calculus provides the calculation for the outgoing event curves and the remaining capacity curves taking the incoming events curves, the available capacity curves and the scheduling into account.

The advantages of this methods are that the curves allow the accurate modeling of complex stimulations, more accurate than only with period and jitter, and that it is based on a powerful algebra. Disadvantages are long calculation times.

The development of scheduling analysis tools lies at the very core of the schedulability analysis issue. While the number of such tools is constantly increasing, they also vary widely in terms of analysis capabilities and supported features. MAST [11] and Cheddar [12], two open source tools are based on classical feasibility tests allowing schedulability analysis of fixed-priority and EDF-based monoprocessor and distributed systems. Cheddar gives also the possibility for the user to specify new schedulers and task models that cannot be described by classical approaches. However, Cheddar focuses only on tasks and does not support a function-level characterization as MAST does. Rapid-RMA [13] is a commercial tool that is based on classical rate monotonic and deadline monotonic algorithms assuming, hence, tasks to be independent.

In [14] the authors show that, in despite the numerous schedulability tests that has been developed during last decades (and implemented in the aforementioned tools), a gap still exist between the assumptions considered in these schedulability tests and the automotive task model. Hence, to be able to perform accurate timing analysis for automotive systems, one needs to extend these tests to take into account automotive task model specificities. However, as shown in [14] and [15] many of these extensions seem to be intractable and inefficient. Thus, for analytical timing verification, new techniques should be used.

In addition, for complex real time systems, analytical techniques, in spite of their high verification quality, have restricted capacity and leads to pessimistic results. In contrast, simulation techniques are able to handle large and complex designs (but with lower quality). Hence to perform accurate timing verification, a better solution would be to apply the different verification approaches where they perform best. For instance, calculate analytically the safe worst-case response times of module components, but use performance simulation for obtaining average times at system level.

In this context, the tool suite chronVAL/chronSIM has been developed to enable performing timing verification based both on analysis (chronVAL) and simulation (chronSIM). In the remaining of this paper, we suggest to evaluate the capabilities of this tool suite to enable accurate timing verification for automotive applications.

3. AUTOMOTIVE NEEDS AND TOOLS REQUIREMENTS

This section characterizes the architecture of automotive applications. Such characterization suffices for the purpose of the present paper, which is to identify the timing analysis needs of automotive systems and hence the requirements that should be met by timing verification tools. It serves, finally, to provide an informal review of capabilities provided by the studied tool suite. For a better understandability, we will assign

an identifier to each requirement that we denote REQx where x is the requirement number.
Today's automotive systems have evolved constantly and now offer ever more challenging features that can be summed up as follows:

Limited hardware resources: Today, CPU load, has become day-to-day issue and is the very basis for the design of automotive systems. For these reasons, scheduling analysis is required to determine, or at least estimate, the processor performance needed for a given design. Hence, *Timing verification tools should have techniques to determine the processor utilization [REQ1].*

Timing Constraints: In addition to limited hardware resources, automotive applications must deal with many kinds of timing constraints. These may concern task or function deadlines or maximum jitters on task activation instants. Automotive tasks may have hard deadlines (e.g. for safety functions) or soft deadlines (for body comfort functions). In addition, the end-to-end delay after data is read by a sensor and the output generated from it and passed to an actuator (known as "data age") is crucial to control model stability. Scheduling analysis is hence needed to verify if those constraints are met or not. To enable this verification, scheduling analysis tools have to meet certain requirements that we summarize as follows:
When describing the system under analysis

- *Timing verification tools should allow specifying task or function deadlines [REQ2]*
- *Timing verification tools should allow specifying jitters related to the function or task activation instants [REQ3]*
- *Timing verification tools should allow specifying end-to-end timing constraints [REQ4]*

When analysing the system

- *Timing verification tools should have techniques to verify if a deadline is respected [REQ5]*
- *Timing verification tools should have techniques to verify if end-to-end constraints are respected [REQ6]*

Triggering paradigms: Automotive applications often involve event-triggered and time-triggered tasks. Event-triggered means that tasks are executed or messages are transmitted by the occurrence of significant events. Time-triggered means that tasks are executed or messages transmitted at predetermined points in time. In automotive, the arrival pattern of an event may be periodic, sporadic or singular (arrives only once). Periodic tasks may involve timing recurrence (e.g. 10ms time base), angle recurrence (e.g. each 50 degree crank based on the crankshaft position) or external activation (e .g. can message). By angle recurrence, we mean the activation of some tasks that depend on the crankshaft and camshaft position. (*The camshaft is the element of the engine that allows the opening and the closure of intake and exhaust valves. The crankshaft is the part of the engine that translates reciprocating linear piston motion into rotation*). For a good analysis, it is thus necessary that analysis tools account for this diversity in triggering of automotive systems. We formalize this capability by the following requirements:

- *Timing verification tools should allow specifying periodic, sporadic and singular events/tasks [REQ7]*
- *For periodic events/tasks, timing verification tools should allow specifying angular recurrences [REQ8]*

Distributed architecture: In conventional automotive system design, externally supplied software is integrated by the car manufacturer into ECUs (Electronic Control Units). Its functionality is then distributed over many ECUs into a network that may even use multiple protocols. Most used protocols are CAN, LIN and FlexRay [16]. For such distributed functions, it is important to guarantee end-to-end response times. In addition, in such complex architectures, optimization of network resource consumption and message scheduling requires knowledge of the impact of network properties such as network overheads and driver overheads, and of different communication protocols. Consequently, scheduling analysis tools have to satisfy the following requirements:

- *Timing verification tools should allow easy description of distributed systems with multiple ECUs and communication buses [REQ9]*
- *Timing verification tools should have techniques to analyze multiprocessor systems [REQ10]*
- *Timing verification tools should have techniques for CAN, LIN and FlexRay [REQ11]*
- *Timing verification tools should allow taking into account processors overheads (basically context switch overhead) and network overhead (network driver overheads) [REQ12]*

Task concurrency and dependency: In automotive systems, tasks may be dependent. This dependency results basically from task chaining which means that a task is activated by the termination of its predecessor. Concerning the concurrency issue, in automotive design, although tasks are concurrent, different tasks may have the same priority level. As automotive applications are based on OSEK, these tasks are scheduled using the FIFO algorithm (First In First out) as a

second scheduling protocol. Moreover, automotive tasks are of three kinds: preemptive tasks, cooperative tasks and interrupts. Cooperative tasks may be interrupted by higher priority cooperative tasks only in predefined points called schedule points. The non-preemptible sections of a cooperative task are used to ensure data consistency in case of shared data. To enable

- *Timing verification tools should allow using FIFO as second scheduling algorithm for tasks having the same priority level [REQ14]*
- *Timing verification tools should allow specifying pre-emptive, cooperative tasks and interrupts [REQ15]*

an accurate scheduling analysis, analysis tools have to support the description and analysis of such a task model and hence:

- *Timing verification tools should allow describing task dependency resulting from task chaining [REQ13]*
- *Timing verification tools should allow specifying and analyzing tasks with static and variable offset [REQ16].*

Table1 Timing verification tools requirements

Requirement	Description
REQ1	Timing verification tools should have techniques to determine the processor utilization
REQ2	Timing verification tools should allow specifying task or function deadlines
REQ3	Timing verification tools should allow specifying jitters related to the function or task activation instants
REQ4	Timing verification tools should allow specifying end-to-end timing constraints
REQ5	Timing verification tools should have techniques to verify if a deadline is respected
REQ6	Timing verification tools should have techniques to verify if end-to-end constraints are respected
REQ7	Timing verification tools should allow specifying periodic, sporadic and singular events/tasks
REQ8	For periodic events/tasks, Timing verification tools should allow specifying angular recurrences (engine-synchronous tasks)
REQ9	Timing verification tools should allow easy description of distributed systems with multiple ECUs and communication buses
REQ10	Timing verification tools should have techniques to analyze multiprocessor systems
REQ11	Timing verification tools should have techniques for CAN, LIN and FlexRay
REQ12	Timing verification tools should allow taking into account processors overheads (basically context switch overhead) and network overhead (network driver overheads)
REQ13	Timing verification tools should allow describing task dependency resulting from task chaining

Requirement	Description
REQ14	Timing verification tools should allow using FIFO as second scheduling algorithm for tasks having the same priority level
REQ15	Timing verification tools should allow specifying preemptive, cooperative tasks and interrupts
REQ16	Timing verification tools should allow specifying and analyzing tasks with static and variable offsets

4. CHRONSIM/CHRONVAL PRESENTATION

The chronSIM/chronVAL tool suite allows a detailed investigation of the timing issues of complex embedded systems by means of model-based real-time simulation (chronSIM) and analysis (chronVAL). It supports many stages of the development process starting with abstract models requiring only some information up to complete implemented C-code.

Modeling in chronSIM:

The simulator chronSIM takes the stimuli of tasks and interrupts, the distribution and the execution times of tasks and ISRs together with the selected scheduling policies into account to determine the concrete and detailed scheduling and timing execution of the tasks and runnables.

The modeling can solely be done model-based or in combination with c-functions. The model can therefore have a adjustable degree of abstraction.

In the user interface the model elements are represented by a hierarchical tree. The modeling starts with adding the resources, schedulers, tasks, interrupts and runnables (if applicable). The activation between tasks and interrupts can be set up by adding connections between them. Each resource has an initial scheduler for which a scheduling strategy, like fixed-priority, TDMA or OSEK can be selected. It is possible to add sub-schedulers below these first level schedulers. So for example the first level scheduler can be set to TDMA and a fixed priority scheduler for one slot of the TDMA cycle can be added as second level scheduler.

The external stimulation of the tasks and interrupts is defined in a separate widget. There are several kinds of patterns like the periodic pattern or the burst pattern. In the periodic pattern the stimulus is defined by parameters like a repetition time, a possible limit on the number of events, a relative jitter value, an initial offset and so on. For the simulation not only the size of the jitter value is required but also its type of random distribution.

It is possible to add C-code to the simulation model. Tasks can have dedicated C-functions to model the behavior of the task in more details. During the simulation the c-code is executed on the host computer but the timing behavior is considered by the simulation as if it would run on the target platform. In the C-code special macro commands are integrated to model and track the timing behavior. Estimated execution time are often sufficient to reproduce errors and identify their root causes as well as other effects due to timing issues.

Execution times: To model the execution time there are several options. First they can be modeled in the GUI on task and runnable level with best-case/worst-case execution time and optionally a selectable random distribution. Secondly, for functions for which c-code is provided, the estimator chronEST can be used to calculate execution times. The calculation is performed on a basic block level taking the hardware architecture, the compiler and the operating system into account.

```
TASK(P1, int i, int j)
{
    DELAY(10, unit_us);
    if (i == j)
        DELAY(1, unit_ms);
    else
        DELAY(i*10, unit_us);
    TerminateTask();
}
```

Fig 1: Definitions for execution time macro

Finally the third option is to use c-code with a special macro modeling the execution time (Fig. 1). The macro takes a value, a unit and, optionally, the kind of random distribution. It models the execution time taken into account by the simulation in this C-function. A function may contain several execution time macros simultaneously, the call of the macro can depend on conditions and the value can be calculated using variables.

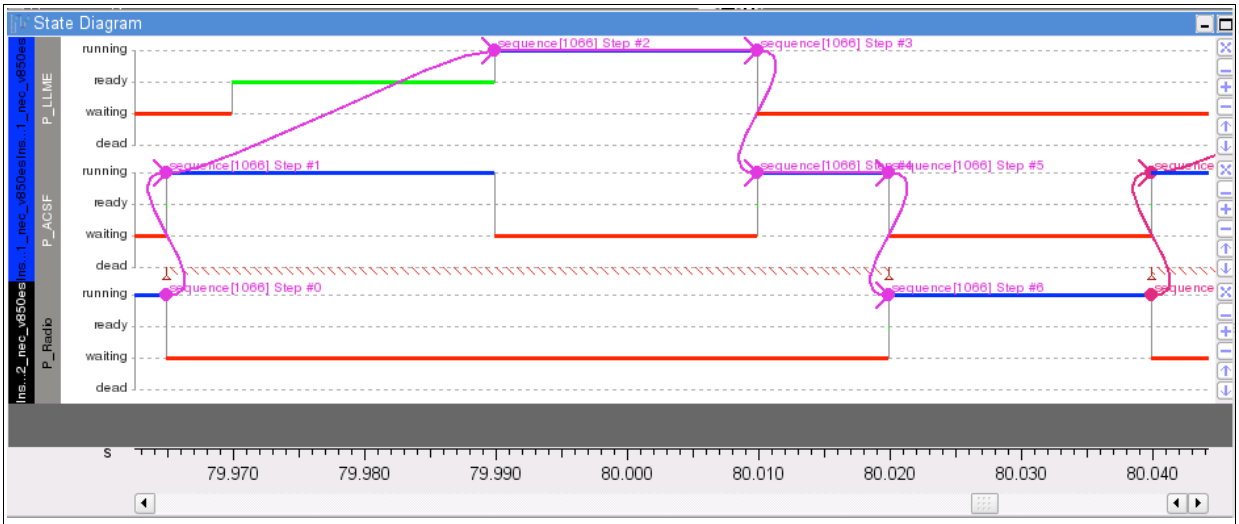

Fig. 2: State diagram for chronSIM

All these different modeling options can be mixed in the same simulation model. For example it is possible to model most of a system in a simple GUI-based way and provide only for one or two critical functions a more detailed concrete implementation in C. Or to start in an early development phase with a simple GUI-based model, and step-by-step substitute parts of the model with more detailed modeling or c-code. Also for the c-code itself the user can start with simple functions containing only the execution-time macro and substitute them later with more detailed functions up to the real used implementation code.

Note that with the use of the execution time macros only those parts of the code are relevant which can affect the timing behavior. This is code modifying variables required for the calculation of the execution time in the macros and code influencing the further path of the program, the number of loop iterations or the call of functions. In many cases the concrete calculation of execution time can be substituted by selecting a random distribution.

In addition to the execution-time macro, macros are available for other timing-related activity like the generation and receiving of events, sending and receiving communication messages like for CAN or FlexRay-bus systems, tracking event chains and so on.

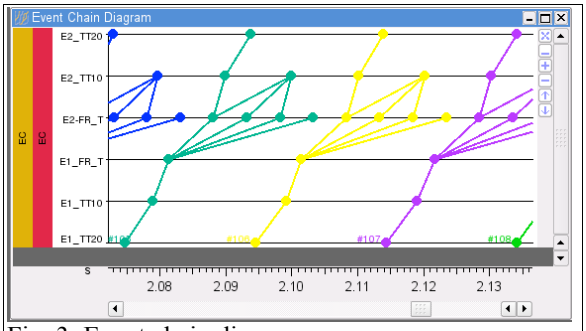

Fig. 3: Event chain diagram

The simulation allows to model different clocks for the different resources in the system. So the stimulus and the execution time does not necessarily depend on a global ideal time base but on a local time base. The time bases can be correlated to each other, especially there can be an offset, drift, etc. between different clocks. For example the real FlexRay bus has its own time base which is derived from the different time bases of the ECUs connected to the bus. Such systems can be modeled accurately and it allows finding effects originating from not completely synchronized time bases.

Simulation results:

During the simulation timing relevant information is stored in a trace file. This is the base for a lot of useful diagrams depicting the details of the simulation runs.

The state diagram (Fig. 2) shows for each task at any time its actual state (running, waiting, idle, preempted, …). So the user can observe for each concrete point of time which task is running and which other tasks are available for execution. The effects of the scheduling policy and of the selected priorities are visible. It is easy to investigate the effects of changes in the priorities and scheduling policies or of mapping tasks to other resources.

The load diagram gives not only the overall load of the system but also load of a gliding window over time based on a selectable range and granularity. It visualizes whether the resource utilization is distributed evenly or whether there are areas with high (or full) utilization on the one hand and areas with low utiliza-

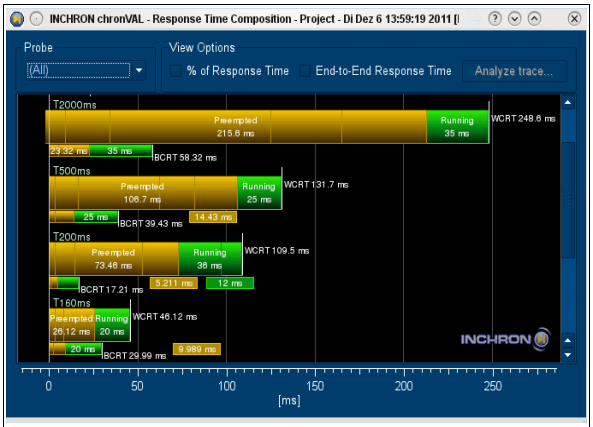

Fig. 4: Response time composition for chronVAL

tion (or idle time) on the other hand. This allows the user to systematic shift tasks in the areas with low utilization to reduce response times.

Other available diagrams are the stack diagram, the call nesting diagram, event chain diagram and others.

For a quick overview histograms give statistical information on the probability for different values of an indicator based on the concrete simulation trace. Indicators can be for example the response time or the jitter of a task or of an event, the distance between two consecutive events of the same or of different kinds and other more.

The results for event chains are visualized in a specialized event chain diagram. Each event chain shows the different steps of the processing and propagation of a specified data over time and through the system. The diagram shows the multiple usages of the same data or the loss of data (Fig. 3).

The event chains can also be visualized directly in the state diagram.

With a histogram all kind of data for event chains can be collected. Examples for histogram values are the response time between every first and last step of an event chain or of the time between the same steps of two event chains of the same type.

chronVAL:

The chronVAL analysis is mainly based on the real-time calculus with the necessary supplements for a better support of realistic systems. These are for example analysis methods for cooperative scheduling, support of offset analysis and so on.

Background of the analysis:

The real-time calculus was developed by Thiele et al. ([17], [18], [19]) and is based on the network calculus ([22], [23]). The real-time calculus is an approach for compositional real-time analysis based on the concepts of min-plus and max-plus algebra. It splits the whole distributed system into processing components having each an incoming upper and lower arrival and upper and lower capacity curve. The curves give for each time interval length the maximum respectively minimum value which can occur in any time interval of this length in any possible schedule. The theory provides the equations necessary to calculate the outgoing upper and lower arrival curves and the remaining capacity curves out of these incoming curves.

A processing component can for example be a task. The incoming arrival curve is than the stimulus of the task. In chronVAL it is denoted as incoming event spectrum. The upper incoming event spectrum represents for all time interval length the maximum number of activations of the task occurring within each time interval of a certain length. The lower incoming event spectrum represents the minimum number of activations for each time interval length.

The upper available capacity curve or capacity spectrum represents for each interval length the maximum computation time available for this task. For example in a simple fixed-priority scheduling, the available capacity is the available capacity for the resource of the task reduced by the sum of computation time required by all tasks with a higher priority than the task in question. The capacity spectra are not only calculated for tasks but also for scheduler and resources. The lower remaining capacity of a resource gives for each interval length the minimum amount of computation time left in any interval of this length. For very large intervals the fraction between the required computation time and the available computation time for the resource leads directly to the maximum utilization of the resource.

Approximation: One extension of the real-time calculus for chronVAL is an approximative representation ([20], [21]) for the curves or spectra which enables a fast and efficient analysis. The degree of exactness for the approximation is selectable and a trade-off between the run-time of the analysis and the degree of exactness is possible. The basic concept behind the approximation is to consider the number of stimuli for a task exactly for intervals with only a few stimulations for the task and to approximate it for intervals with many stimulations. The approximation is done separately for each task so that for a certain interval a task occurring very often will be approximated whereas a task occurring more seldom will still be considered exactly.

The approximation limits the number of elements required to represent a spectrum. It is directly related to the number of exactly considered activations for the tasks. The effort required for the operations of the real-time calculus directly depends on the number of elements required for the representation of the spectra. Therefore the approximation reduces the effort and allows using the analysis for realistic systems.

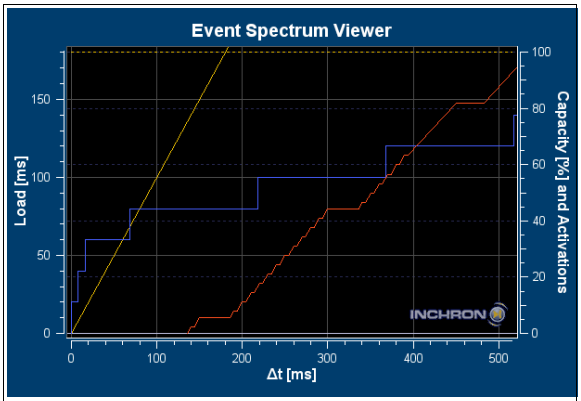

Fig. 5: Event spectrum viewer for chronVAL

Modeling for chronVAL:
For the analysis an initial definition of tasks, their distribution on resources, their stimuli and their execution times are necessary.
The stimuli for the analysis are the same as for the simulation.
The modeling for the analysis is nearly the same as the models for the simulation, one exception is c-code which is at the moment only supported by simulation. The random distributions of the jitter or the execution time is not relevant for the analysis and therefore not used. The analysis always has to consider the worst-case (which can for the jitter also be the middle value).

Analysis results: chronVAL calculates for example worst-case and best-case response times, their composition and various event spectra giving detailed information on the stimuli and capacities within the system.
The best- and worst-case response time for tasks and runnables are not only presented with their specific value but also with their composition (Fig. 4). So it is visualized which part the response time the task is running, which part it is preempted and by which other task and so on. Additionally a trace leading to the worst-case response time can be visualized in a simulation state diagram.
The event spectra (fig 5) provides more detailed information generated by the analysis. For each task, runnable, scheduler and resource the maximum and minimum available and the remaining capacity and, if applicable, maximum and minimum incoming and outgoing stimulation is calculated.
An event spectrum is a mathematical function which describes the specific value over time intervals. This definition is identical to the definition of curves in the network / real-time calculus. So the spectrum for the maximum stimulus of a task gives for each interval-length the maximum amount of events which can activate this task within any interval of this length.
In the same way the available capacity spectrum depicts for each interval-length the maximum resp. minimum capacity available for the task or resource within any interval of this length within any concrete possible schedule. So with the event spectra all worst-case situations which can happen within any possible schedule of the system are concentrated in one function and are visible at first glance.

5. TOOL SUITE EVALUATION

This section presents the capabilities of the studied tools suite based on the requirements determined on section 3.

REQ1: chronVAL does not display a value showing the global utilization of the processor by the different tasks. However, through a graph called "event spectrum viewer", it is possible to visualize the variation of the available and the remaining processor capacity for each task. The utilization is the value of the event spectrum "minimum remaining capacity" for the infinite interval. The utilization value is also shown in the report. For chronSIM the load diagram shows the utilization for a sliding window over the simulation trace. The user can choose the size and the granularity of this window. This diagram shows whether the utilization is distributed evenly or whether there are large hotspots with high utilizations together with ranges of low utilization or idle time.

REQ2: To describe task deadlines, chronSIM/chronVAL allows assigning a timing requirement to a task. This requirement allows specifying a bound on the delay between the activation event of the task and its termination event.

REQ3: To describe the activation of a task, chronSIM and chronVAL uses the concept of "stimulation". A stimulation is an element that is connected to a task to describe its activation patterns such as its period, the minimum inter-arrival time, the offset, its activation jitter and the jitter of further occurrences, a possible limitation for the number of occurrences and so on. Important for the simulation is the kind of random distribution for the jitter values. There are additional stimulation patterns for other kind of activations like bursts.

REQ4: Specifying end-to-end timing constraints is also supported through the concept of requirement in chronSIM/chronVAL. To specify an end-to-end constraint on a flow of tasks, one can specify a requirement between the activation event of the first task and the termination event of the last task in the flow.

REQ5: To verify if a deadline is respected or not, chronVAL calculates the worst-case response time for each task and compares it with the deadline. For chronSIM the requirement is compared with every occurrence of the task and additionally to the overall result a statistic is provided on how often and in which extend the response time is missed together with links to the specific occurrences of the task in the state diagram.

REQ6*:* As for deadlines, chronSIM and chronVAL calculates end-to-end response times and compare them with end-to-end requirements.

REQ7: When describing task activation, chronSIM and chronVAL allows describing periodic and sporadic tasks either with or without jitters. Singular tasks can also be described by specifying no repetition for them.

REQ8: The graphical interface of chronVAL uses only a timing base, hence the activation of engine synchronous tasks cannot be specified e.g. in CRK degrees but in time units. So, the dependency of the period of these tasks on the engine speed cannot be described directly and the analysis may be done only for a fixed engine speed. It is possible to use induced stimulations to model such a scenario. The base is a periodic stimulation which generates one event for every single CRK degree as a base time. The induced stimulations can be configured to propagate only one out of x events and can get also the offset for this event. Therefore it can be ensured that the events propagated by different induced stimulations are separated. For chronSIM additionally the model of clocks can be used. A drift can be specified modeling increasing speed. Or a few lines of C-code together with the simulation macros allow the modeling of even complex scenarios.

REQ9: chronSIM and chronVAL allows modeling and analyzing distributed systems. When describing the system under analysis, the tool allows describing multiple ECUs and buses.

REQ10: chronSIM and chronVAL allows analyzing multiprocessor systems (distributed systems).

REQ11: The tool has dedicated analysis and simulation techniques for CAN and simulation techniques for FlexRay. The upcoming version covers CAN, FlexRay, LIN and additional switched Ethernet for both simulation and analysis.

REQ12: Context switch overheads and network overheads can be described in chronSIM and chronVAL separately for each resource.

REQ13: In chronVAL, task chaining can be described. In fact, each task has got a “connection” field. In this field, it is possible to describe an activation source for the task or to specify that this task is activated by another task.
Additionally the concept of event chains is available allowing to model data dependencies more accurately. For chronVAL it is possible to define event (data) chains for runnables of tasks. For chronSIM the steps of the event chains can be defined in C-Code by an event chain macro. An event chain diagram gives detailed information on multiple data usage, lost data or the jitter of the data chain.

REQ14 & REQ15 & REQ16: The Tool allows specifying offsets, interrupts as well as preemptive and cooperative tasks. Cooperation aspect is described through specifying non-preemptible sections and their execution times for each task.
chronVAL analysis is based on the “real time calculus” technique. The tool allows specifying the different aspects described in REQ14, REQ15 and REQ16 including offsets and tasks with the same priority, and these aspects are taken into account by the analysis.

6. CONCLUSION

In this paper, we discussed crucial capabilities that need to be available in tools for timing verification in automotive applications. We evaluated the extent to which a commercial tool suite, chronVAL/chronSIM[2], satisfies such requirements. This evaluation reveals that for an accurate timing verification for automotive systems, both analytical and simulation techniques are needed. Thus, a complete timing verification approach (such as the case for the evaluated tool suite) should combine the two techniques enabling hence to apply each one where it would perform best.[3]

[2] The development of the underlying technology of the INCHRON Tool-Suite was funded by financial means of the Ministry of Economics of the State Brandenburg and the European Union. The responsibility for the content is solely with the authors.

[3]The work has been partly funded by the German Ministry for Education and Research (BMBF) under the funding ID

.

7. REFERENCES

[1] N. Navet, F. Simonot-Lion, editors: The Automotive Embedded Systems Handbook. Industrial Information Technology series, CRC Press / Taylor and Francis, ISBN 978-0849380266, December 2008.

[2] Lehoczky, J., L. Sha and Y. Ding, "The Rate Monotonic Scheduling Algorithm: Exact Characterisation and Average Case Behaviour," Proceedings of the Real-Time Systems Symposium (1989).

[3] Liu, C. L., J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," Journal of the ACM, 20, 1 (1973), 46-61

[4] Audsley, N., Burns, A., Richardson, M., Tindell, K., and Wellings, A. Applying new scheduling theory to static priority preemptive scheduling. Software Engineering Journal 8, 5 (1993), 285–292.

[5] Leung, J., and Whitehead, J. On the complexity of fixed-priority scheduling of periodic, real-time tasks. Performance Evaluation 2 (1982), 237–250.

[6] Lehoczky, J. "Fixed Priority Scheduling of Periodic Task Sets with Arbitrary Deadlines," Proceedings 11th IEEE Real-Time Systems Symposium (5-7 December 1990) pp.201-209.

[7] K. Tindell and J. Clark. Holistic schedulability analysis for distributed real-time systems. Microprocessing and Microprogramming - Euromicro Journal (Special Issue on Parallel Embedded Real-Time Systems), 40:117–134, 1994

[8] K. Tindell, "Adding Time-Offsets to Schedulability Analysis", Technical Report YCS 221, Dept. of Computer Science, University of York, England, January 1994.

[9] J.C. Palencia Gutiérrez and M. González Harbour, "Schedulability Analysis for Tasks with Static and Dynamic Offsets". Proceedings of the 18th. IEEE Real-Time Systems Symposium, Madrid, Spain, December 1998.

[10] J.C. Palencia Gutiérrez and M. González Harbour, "Schedulability Analysis for Tasks with Static and Dynamic Offsets". Proceedings of the 18th. IEEE Real-Time Systems Symposium, Madrid, Spain, December 1998.

[11] MAST website (http://mast.unica.es)

[12] Cheddar website (https://beru.univ-brest.fr)

[13] Rapid-RMA website (http://www.tripac.com/rapid-rma)

[14] Saoussen Anssi, Sébastien Gérard, Stefan Kuntz, François Terrier, "On the Gap between Schedulability Tests and Automotive Task Model", International Workshop on Analysis Tools and Methodologies for Embedded and Real time Systems, In conjunction with the ECRTS conference, Porto, Portugal, July 5th-8th, 2011

[15] P. Hladik, A. Deplanche, S. Faucou, and Y. Trinquet, Schedulability analysis of OSEKNVDX applications, in 15th International Conference on Real-Time and Network Systems, 2007.

[16] M. Traub, V. Lauer, J. Becker, M. Jersak, K. Richter and M. Kuhl: Using timing analysis for evaluating communication behaviour network topologies in an early design phase of automotive electric/electronic architectures. SAE World Congress, Detriot, MI, USA, April 2009.

[17] S. Chakraborty, S. Künzli, L. Thiele, Performance evaluation of network processor architectures: Combining simulation with analytical estimations, Computer networks, 41(5), pp. 641-665, 2003

[18] L. Thiele, S. Chakraborty, M. Naedele, Real-time calculus for scheduling hard real-time systems, Proceedings of the IEEE Conference on Circiuts and Systems, 2000.

[19] E. Wandler, Modular performance analysis and interface-based design for embedded real-time systems, Phd-thesis nr. 16819, ETH Zürich, 2006

[20] K. Albers: Approximative Real-Time Analysis, Phd. Thesis, University of Ulm, 2011

[21] K. Albers, F. Slomka: An event stream driven approximation for the analysis of real-time systems, in Proceedings of the 16th Euromicro Conference on Real-Time Systems, pp. 187-195, 2004

[22] L. Cruz, A calculus for network delays, in IEEE Transactions in Information Theory, vol. 37, pp. 114-141, 1991

[23] J.-Y. Le Boudec, P. Thiran, Network calculus: A theory of deterministic queuing systems for the internet, Lecture Notes in Computer Science, Springer, 2001

01IS10034 as part of the TIMMO-2-USE research project in the framework of the ITEA2, EUREKA cluster program 3674.