Technische Universität Berlin

Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen

Carsten Gremzow
Nico Moser (Hrsg.)

Programmkomitee

Wir danken den folgenden Personen für die Begutachtung und Auswahl der Beiträge.

Bernd Becker, *Universität Freiburg*
Claudia Blank, *OneSpin Solutions, München*
Rolf Drechsler, *Universität Bremen*
Rolf Ernst, *Technische Universität Braunschweig*
Hans Eveking, *Technische Universität Darmstadt*
Martin Freibothe, *OneSpin Solutions, München*
Carsten Gremzow, *Technische Universität Berlin*
Christoph Grimm, *Technische Universität Wien*
Wolfram Hardt, *Technische Universität Chemnitz*
Christian Haubelt, *Universität Erlangen*
Lars Hedrich, *Universität Frankfurt/Main*
Ulrich Heinkel, *Technische Universität Chemnitz*
Jörg Henkel, *Universität Karlsruhe*
Sorin A. Huss, *Technische Universität Darmstadt*
Christoph Jäschke, *IBM Böblingen*
Thomas Kropf, *Bosch, Leonberg*
Wolfgang Kunz, *Technische Universität Kaiserslautern*
Gunther Lehmann, *Infineon Technologies, München*
Paul Molitor, *Universität Halle*
Wolfgang Müller, *Universität Paderborn*
Klaus D. Müller-Glaser, *Universität Karlsruhe*
Peter Oehler, *Continental Automotive Systems, Frankfurt/Main*
Michael Payer, *Infineon Technologies, München*
Achim Rettberg, *OFFIS, Oldenburg*
Jürgen Ruf, *IBM, Böblingen*
Klaus Schneider, *Technische Universität Kaiserslautern*
Christoph Scholl, *Universität Freiburg*
Jens Schönherr, *Signalion, Dresden*
Martin Speitel, *Fraunhofer IIS, Erlangen*
Dominik Stoffel, *Technische Universität Kaiserslautern*
Jürgen Teich, *Universität Erlangen*
Dietmar Tutsch, *Universität Wuppertal*
Reinhold Vahrmann, *Atmel, Heilbronn*
Klaus Waldschmidt, *Universität Frankfurt/Main*
Klaus Winkelmann, *OneSpin Solutions, München*
Reimund Wittmann, *IP GEN Rechte, Stuttgart*

Inhaltsverzeichnis

Eingeladene Vorträge

Session: Verifikation 1

Session: Verifikation 2

Session: Verifikation 3

Session: Entwurfsmethodik / Entwurfswiederverwendung

Session: Entwurfsmethodik / Co-Design

Session: Simulation und Funktionaler Test

Session: Spezifikation und Modellierung

Session: Funktionaler Test / Entwurfsmethodik (Analog)

Entwurf von Aufbau- und Verbindungstechniken für die Elektromagnetische Zuverlässigkeit von Mikrosystemen unter Verwendung des M3-Ansatzes

Herbert Reichl*,†, Ivan Ndip*

*Fraunhofer-Institut für Zuverlässigkeit und Mikrointegration, IZM
Gustav-Meyer-Allee 25,
13355 Berlin
herbert.reichl@izm.fraunhofer.de

†Technische Universität Berlin
Straße des 17. Juni 135,
10623 Berlin

Um den stetig wachsenden Bedarf an kleineren, preiswerteren, multifunktionalen und leistungsfähigeren mikroelektronischen Produkten zu erfüllen, werden neben leistungsfähigeren Integrierten Schaltkreisen (Chips) auch neue Aufbau- und Verbindungstechniken (AVT), z.B. die System-in-Package (SiP) Technologie, sowie neuartige Entwurfsmethoden benötigt. Die SiP Technologie ermöglicht die Integration von Chips unterschiedlicher Funktionalitäten (z.B. HF, High-Speed digital, Sensorik) in einem kompakten Modul, wodurch gleichzeitig Platz und Entwicklungskosten reduziert werden können. Der Entwurf von Signalpfaden in diesen miniaturisierten SiP-Modulen stellt aber eine große Herausforderung dar. Die parasitären Effekte auf Package- und Board-Ebene, die anfgrund der dichten Anordnung der Bauelemente und Leitungen entstehen, sowie die Diskontinuitäten anf den Signalpfaden führen zu Elektromagnetischen Zuverlässigkeitsproblemen (EMZ-Probleme; engl. *Electromagnetic Reliability (EMR) Problems*) wie beispielsweise mangelhafter Signal-/Power-Integrität und EMV-Problemen. Diese Probleme verstärken sich mit steigenden Taktfrequenzen und können dazu führen, dass das Gerät nach der Entwicklung nicht (einwandfrei) funktioniert. Da es immer sehr schwierig und vor allem teuer ist, solche Probleme und ihre Ursachen nach dem Aufbau des Systems zu identifizieren und zu lösen, ist es unbedingt erforderlich, Entwurfsregeln am Anfang der Designphase einzusetzen, die die parasitären Effekte aller Komponenten entlang des vollständigen Signalpfades bei Mikrowellen-Frequenzen berücksichtigen.

In diesem Vortrag wird ein geschlossener Entwurfsansatz, der **M3-Ansatz** für einen optimalen, zuverlässigen und kostengünstigen Entwurf von elektrischen Verbindungen, elektronischen Packages, Leiterplatten und integrierten Komponenten präsentiert und illustriert. Das Hauptziel des M3-Ansatzes ist es die Limitierungen der herkömmlichen „Trial-and-Error"-Methoden zu überwinden. Um dieses Ziel zu erreichen sind drei Schritte notwendig; 1) Entwicklung und/oder Anwendung von zuverlässigen **Methodologien** für die effiziente und akkurate Modellierung von AVT-Strukturen. Abhängig von der Komplexität der zu entwerfenden Strukturen und dem gewünschten Frequenzbereich (oder der Bandbreite), werden entweder statische, quasi-statische oder Voll-Wellen Modellierungsmethoden eingesetzt. Basierend auf den extrahierten Ergebnissen werden parametrisierte Modelle entwickelt.
2) Alle so hergeleiteten **Modelle** werden experimentell validiert. Sie werden benutzt um den Einfluss der Entwurfsparameter (Geometrie- und Materialparameter) sowie der unmittelbaren Umgebung der Packaging-Strukturen auf die elektrischen Eigenschaften des (Sub-) Systems zu

New Methods for System-level Verification using SystemC-AMS Extensions: Application to an Automotive ECU

Monica Rafaila, Christian Decker; Infineon AG – Automotive Power, Am Campeon 1-12, 85579 Neubiberg/Germany
Christoph Grimm; Institute of Computer Technology/Embedded Systems, Vienna University of Technology, Gusshausstr. 27-29/384, A1040 Vienna/Austria
Karsten Einwich, Thomas Markwirth; Fraunhofer Institute for Integrated Circuits, Design Automation Division, Zeunerstr. 38, 01069 Dresden/Germany
Georg Pelz; Infineon AG – Automotive Power, Am Campeon 1-12, 85579 Neubiberg/Germany

Abstract - This paper describes the difficulties faced in model-based verification of automotive ECUs (Electronic Control Units). Upcoming standards and new tools are applied to approach new solutions. The system heterogeneity is dealt with by abstracting and creating discrete-time representations, thus handling verification tasks of the complete system in its operating environment. A case study reflects the benefits, in terms of performance and flexibility, enabled by the description language, and achieved by a corresponding abstraction.

I. INTRODUCTION

Automotive ECUs need extensive verification. This is a fact now, more than ever, as the validation must be done for complete systems, in the form of SoCs (Systems-on-Chip) or SiPs (Systems-in-a-Package), integrating what used to be delivered at a component-level.

Model usage can accomplish in-depth verification tasks when hardware is not available, thus offers a cost-effective and systematic solution to deploy key phases of the development flow (e.g. concept validation, functional verification, architectural exploration, application demonstration, virtual platform software development). High-level models are needed in early phases, for concept definition and proof, but also along the way to tape-out, by reflecting the functionality of the complete system, within its real context of application, which would be otherwise hard to reproduce and almost impossible to explore. Thus, models ought to be created and maintained, refined and reused.

Moreover, when it comes to heterogeneous systems, aspects such as abstraction methodology, simulation performance, model exchange and reuse, within the development groups, and with the customer, are necessary, and often compromised. Thus, fast and flexible means to model and simulate need to be adopted and improved.

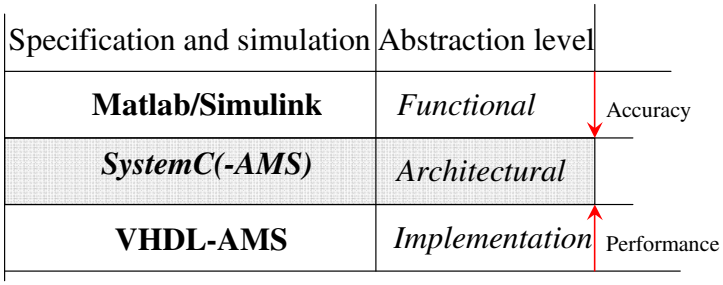

Figure 1. Specification languages and design abstractions

Figure 1 gives an overview of current approaches for modeling AMS (Analog/Mixed-Signal) systems. Specification and simulation of complete heterogeneous systems are currently covered only at pure-functional level by tools like Matlab/Simulink. However, such tools do not cover architectural level details. On the other hand, implementation-focused solutions

like VHDL-AMS [1] do not address the high complexity issues which occur in system-level modeling.

The above mentioned reasons motivate the use of SystemC [2], enhanced by its TLM library [3], as description means which enables abstraction, and extended by SystemC-AMS [4], [5], offering the possibility to describe heterogeneous systems. Adopting open source tools gives the opportunity to have interoperable models, not to mention more perspectives to integrate them in a flow which suits best the needs of the respective field of application and level of abstraction desired.

The extension has been continually improved [6], [7], and proven its applicability in system-level modeling and verification [8], [9]. As a consequence, first industry design flows are adopting it [10], [11], thus provide motivations to approach it for domain-specific problems. Automotive ECUs, facing critical requirements in all development phases, demand more, as the existent approaches still lack the methodology for abstract, flexible and fast models, not to mention verification practices that must come along.

II. PRELIMINARIES

A. *Modeling formalisms*

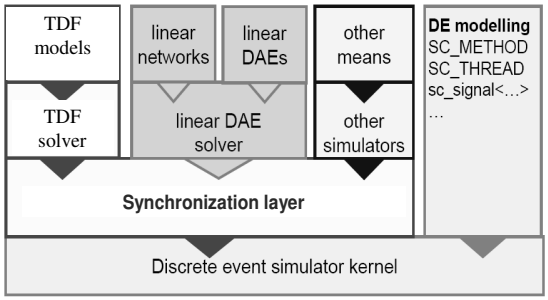

Figure 2. SystemC/SystemC-AMS layered architecture

Figure 2 presents a layered architecture of the language standard and its extensions. The basic unit in SystemC is the module, which is defined by its interface (ports), and its behaviour (processes). Modules are connected through channels, which implement the interfaces specified by the ports bound to them. Processes describe the functionality of the module, and allow expressing concurrency in the system. Processes access external channel interfaces through the module's ports. Processes can be sensitive to events, either by means of static sensitivity, i.e. it cannot change during simulation, or dynamic sensitivity. The scheduler must determine the order of execution of processes within the design, based on the event sensitivity of the processes and the event notifications which occur. Similar to VHDL and Verilog, the SystemC scheduler supports the notion of delta cycle, which is comprised of separate evaluate and update phases. The scheduler executes, in the current delta cycle, the processes sensitive to the events being notified in the previous delta cycle of the simulation, except for immediate notifications, which cause the execution in the same delta cycle. Multiple delta cycles may occur at a particular simulated time, and lead to increased simulation time [12].

The layered structure of the standard enables describing various models of computation (MoC) [13]. In the TDF (Timed Data Flow) MoC, similar to the SDF (Synchronous Dataflow) formalism [14], a module will be activated if a pre-specified number of samples (data tokens) are available at the inport(s). During activation, the module behaviour is executed, the sample(s) from the inport(s) are read, and a pre-specified number of samples is written to the outport(s). The numbers of read/write samples is constant, so the scheduler can determine the firing sequence before simulation starts, thus gaining performance in simulation.

The SystemC TLM standard [3] establishes a fundamental set of general interfaces, thus enables high levels of abstraction in modeling communication, therefore simulation efficiency.

Choosing specific models of computation and communication to describe a system is a modeling decision, which will inevitably impact the ease of implementation, and, in a later phase, the performance of simulation, consequently, of the verification process.

B. System abstract modeling – general considerations

With respect to modeling, SystemC provides features such as: modularity, hierarchy, refinement, scalability, while, from the verification point of view, it enables high simulation performance, ease of test-bench setup and reuse, flexibility in parameter and stimuli control, use of efficient checkers and monitors along each test-case. Previous studies state the need to adopt such a modeling framework [15], and several works have proven efficient in making use of abstraction capabilities available with SystemC, [16], [17].

Modeling a complete system, in the application context, must consider different perspectives:

- functional, as defined by the application;
- architectural;
- non-functional (thermal effects, power supply issues, uncertainties: tolerances, not-defined properties, parameters to be calibrated);

For each functional block, abstraction decisions must be taken. In-depth analysis is necessary, to identify its role in the application (importance, frequency of behaviour activation), its nature (digital/analog) in function and in interface, and system functional composition (interactions to other blocks). Applying the formalisms detailed above, each functional block is modeled either as a pure, DE SystemC module, or as a TDF SystemC-AMS module. These are enhanced by converter ports, when communication between them is necessary, and by TLM interfaces, where communication abstraction is possible.

A choice for DE MoC is made for system parts with an event-driven role in the system, for which time-accuracy of the outputs, in response to event notifications, is essential. Another aspect under consideration is that these blocks, during normal operation, as defined by their role in the application, are not periodically stimulated, nor frequently. Still, they react instantly to the events, considered crucial under the application context. The underlying framework of SystemC scheduler demands for as few event notifications as possible during each delta cycle, as they are costly in terms of simulation time. So only those events will be notified which may trigger behaviour changes with impact on the module's interface. Moreover, this effect must be implemented only when the module interacts with others, sensitive to these changes. That is, behaviour and interface modeling must not be done clock-accurate, for digital hardware, nor with a big granularity to capture analog effects, for AMS hardware modeling, but abstractly, making use of event notifications that make a difference in the overall system behaviour.

There are components which require a frequent evaluation of the behaviour for periodic update of the outputs, necessary in a given application. These should be implemented using the TDF modeling formalism, thus periodically sampling the inputs, and writing to the output ports. It is important to note here that the sampling period of these modules is a refinement criterion, with respect to time-accuracy. It is also a numeric parameter, and, together with other component parameters, is modeled as a simulation-time configurable variable.

Usage of the TLM library is another key element of the approach, when it comes to abstracting details of communication between blocks. This is because: 1) a clock-accurate and bit or pin-accurate interface behaviour is not necessary in abstract descriptions, and not desirable when it comes to simulation performance; and 2) this interface must be scalable, in terms of granularity of information exchange between the blocks.

These considerations are independent of the structure of the modeled components, so are implementation un-aware. Still, certain structural characteristics can be modeled, but only as reflected in their impact on the module's interface (thermal behaviour, dependency on supply inputs). Interface and function modeling must be separated, but coherence has to exist, as the input changes will dictate the behaviour evaluation, which in turn affects the outputs. Also, interactions between components must be considered before choosing a particular implementation, as making too much use of the synchronization between TDF and DE modules will compromise the advantages of the modeling decisions.

C. Simulation-based verification – general considerations

In what follows, system properties refer to characteristics of the complete system, meaningful under the application context. These are defined under specific sets of stimuli, and influenced by parameters, i.e. characteristics of system components.

- The verification efforts are focused in several directions:
- Functional validation: define and prove the system-level concept.
- Sensitivity analysis: determine the impact of components parameters on system properties.
- Parameter tuning: extract/adjust/confirm parameter values, for target system properties.
- Corner case analysis: explore the stimuli and the parameter spaces, to find worst cases.
- Statistical analysis: find worst-cases by Monte Carlo system simulations/Monte Carlo corner system simulations.
- Determine impact of level of model accuracy on fidelity of results: configure accuracy-related parameters and switch between levels of model refinement, to compare results, and analyze differences.

The verification performed must be adapted to the modeling abstractions, as they have significant consequences. Event-driven monitoring functions are dedicated for supervising properties at the interface of the system under study. By simple means, such as dynamic sensitivity and timestamps on notified events, these properties are validated and/or measured:

- Expected event sequences
- Signals and modules states sampled at event notification times
- Delays between these times

For measurements of properties, comparators against thresholds and maximum value detectors are used. These were implemented by value-dependent converters from signals TDF signals to DE signals, thus notifying events to trigger processes concluding about properties. Moreover, the specific values can be configured, and become simulation-time variables.

To extract system properties, characteristic to several parameter sets, multiple simulation runs are performed, on a script-basis implementation.

Monte Carlo simulations are performed using the Statistical package available with VHDL-AMS, as SAE standard J2748, adapted to SystemC/SystemC-AMS, as described in [18]. Various distributions can model uncertain or to-be-defined elements of the system, to perform in-depth sensitivity analysis, or architectural exploration.

An important aspect is that when running complete system simulations, the simulated time must correspond to real application scenarios, up to 7 orders of magnitude bigger than certain time-constants of system components. This can lead to long simulation times, if using higher accuracy then necessary, especially when running sets of simulations for corner-case tests or parameter range exploration. The trade-off between level of model accuracy (determining time-accuracy of components outputs) and simulation efficiency can be directly controlled, in order to asses the impact on the fidelity of results. This is possible, because time-accuracy parameters (e.g.: period of sampling the inputs of TDF modules, period of execution in supervising blocks) are controlled at simulation time, so sets of simulations can be validated against results with more accurate models.

III. THE CASE STUDY

A. Functional and architectural description of the system

The verified system is an ECU designed for window lift applications, which must be validated in this context. It represents a heterogeneous system, covering multiple domains (mechanics, analog electronics, digital electronics, software, thermal).

As represented in Figure 3, the system to be modeled consists of the ECU and the Electro-Mechanical subsystem driven by it.

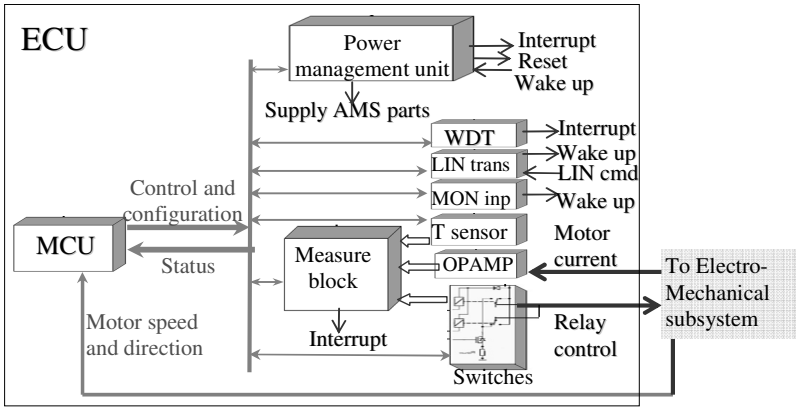

Figure 3. Functional blocks of the ECU

The MCU (microcontroller) subsystem includes an 8-bit state-of-the-art microcontroller, compatible to the standard 8051 core, and several peripherals. The AMS functional blocks are: power management unit, measurement block, switches (two low-side switches, one high-side switch), LIN transceiver, high-voltage monitoring inputs (MON inp), watchdog timer (WDT), temperature sensor, current-sense operational amplifier (OPAMP).

The power management unit contains several blocks, and is responsible mainly for internal and external power supply and supervision, controlling the power modes for the entire system. Low power modes are available, from which wake-up is possible via the LIN Transceiver or monitoring inputs. The watchdog timer supervises periodic trigger inputs from the MCU.

For the application of window lifting, the Electro-Mechanical subsystem has a standard structure: relays, DC motor, gear box, gear rack. A double Hall sensor is necessary to provide the MCU with the speed and direction of the DC motor, in the form of electric signals.

Within this application context, the low-side switches are dedicated to relays control and the high-side switch to an external LED control. The LIN transceiver acts as interface between the MCU and the LIN master, providing the commands to control the mechanical load. The current-sense amplifier generates the amplified value of the DC motor current at the measurement block interface, while the temperature sensor offers the temperature information. The measurement block includes a measurement interface, consisting of 8-bit ADCs, and a control and post-processing unit, which evaluates whether error conditions occur (over-temperature, over-voltage, over-current), and passes this information to the MCU.

B. The model of the system

The DE modeling formalism was applied for several parts: low-side switches, high-side switch, LIN transceiver, monitoring inputs, watchdog timer, power control unit (the power management unit control block). The triggering events mark behaviour changes for the whole system, in the application context (e.g. switch input, wake-up input, reset power mode entry).

System components modeled as TDF modules are: temperature sensor, current-sense amplifier, components of the power management unit (voltage regulators, bias current generator). The periodic evaluation of inputs and update of outputs is necessary because there

are no specific events making output changes, but frequent updates are needed to supply several blocks with accurate values. This process must not be done more frequently than the supplied blocks are capable to react.

The TLM set of interfaces is used to abstract the communication between the MCU and the AMS components, i.e. the register set used to control them and to get status information. This is because there is no need for clock-accurate and register-accurate interface implementation, and because TLM interfaces allow further refinement of the exchanged information, if required. This was implemented using TLM1.0 library, on the master-slave principle, as recommended in [19]. The access is always initiated by the MCU, i.e. the master. Consequently, several AMS modules were enhanced with a slave interface, providing/changing the register data referred to by read/write accesses from the master side.

For the measurement block, the data to be measured, offered by TDF modules, must be periodically converted to digital, then passed to the control and post-processing unit. So TDF MoC is appropriate, while TLM communication to the measurement control unit abstracts the bit-accurate information exchange. Figure 4 summarizes some of the modeling decisions.

	AMS components			Pure digital components	
Function	TDF	DE	TDF TLM	DE	DE TLM
Interface	TDF	DE TLM	TDF TLM	DE TLM	DE TLM
Examples	-Voltage regulator -Bias current generator	-Switches -LIN transceiver -MON inputs	-Measurement Interface -Temperature sensor	-Power control unit -WDT	- Microcontroller

Figure 4. MoCs for system functional blocks

The MCU subsystem model introduces new abstraction elements. As its behaviour must reflect the function of the application, firmware and driver layers, it was logically separated, as in Figure 5. The subsystem consists of a scheduler module, application modules, and a HAL (Hardware Abstraction Layer). The scheduler, based on a priority scheme, runs the applications, as functions contained in the dedicated modules, by means of interface method calls. The HAL provides an interface to the AMS subsystem, by TLM read/write accesses.

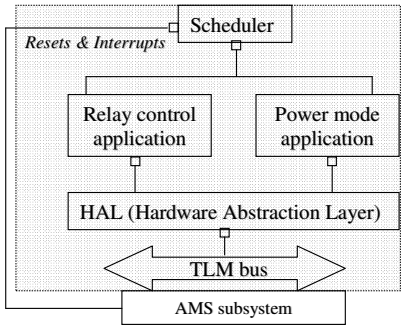

Figure 5. MCU subsystem abstract model

For the MCU subsystem, a cycle-accurate SystemC model, of the MCU core and peripherals is also available. Such an accurate model offers the opportunity to validate and optimize the behaviour of the abstract model, by comparison of simulation results, and truly embed application software. The disadvantage is a reduced simulation performance. For compatibility with the abstract models of the AMS parts, and for simulation efficiency, the

accurate model was enhanced with a TLM interface, similar to the HAL used by the abstract model. In this way, simulations can run faster and results more clearly compared.

To evaluate such an ECU system-level model in its operating environment, the Electro-Mechanical parts were modeled in detail. Their reduced complexity allows this, in terms of modeling effort and simulation performance, as observed in the results, and shifts the focus on the impact of abstraction decisions on simulation results.

C. Results

The functional test-cases verified on the system model are related to the system's main functional features:

- relays control, with window position track in the MCU, by Hall sensor input sense and interpretation, and with anti-pinch effect, by obstacle sense and control of relays.
- protection features, of shutting down in case of over-temperature, over-voltage or over-current conditions.
- general power modes management, with low power mode entry in case of inactivity, and wake-up sequence in case of LIN bus wake, or event at monitoring inputs.

Simulation performance, defined as the ratio (time to simulate/simulated time), was summarized in Table 1.

With abstract MCU											*With cycle-accurate MCU model*
Accuracy parameter	100	90	80	70	60	50	40	30	20	10	10
Simulation performance	4.3	5	5.6	6	6.6	7.6	9.3	11	15.3	30.3	1178.6

Table 1. Simulation performance

The accuracy parameter equals the TDF sample period, in microseconds, and determines other time-accuracy related parameters of the system (e.g. measurement block execution period). The performance strongly depends on the configured accuracy, and on the choice for the MCU representation. The abstract MCU model can be switched with the cycle-accurate one, and the algorithm executed in the application modules becomes embedded software. The performance is also influenced by the test-case's functional coverage, more precisely on the amount of event notifications, relative to the simulated time. These events were considered, in modeling, as rare and important. Simulation of TDF modules is statically scheduled, so their impact on performance is smaller compared to an equivalent DE implementation. This impact increases as the sample period gets smaller, as can be noticed in Table 1.

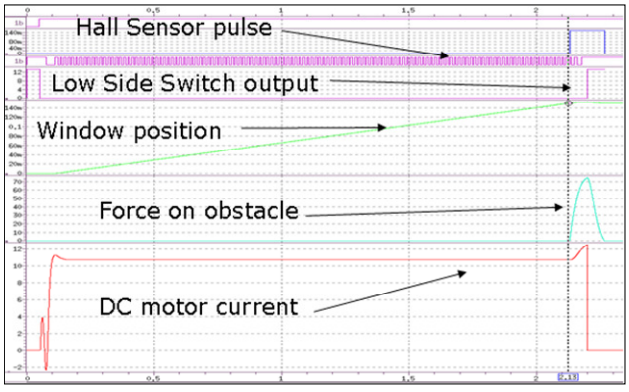

Figure 6. Sample simulated scenario

Figure 6 shows a sample simulated scenario, a LIN command to close the window, followed by internal event sequence, leading to low-side switch output change to control the relays.

These will drive the DC motor, whose current is also represented in Figure 6. An obstacle presence determines an increase in the force exercised by the window, and in the DC motor current. The system reaction is visible, by switching off the low-side switch after over-current measurement.

The simulation results, for the same sample scenario, but with different values for the low-side switch delay parameter are compared, for different choices of time-accuracy parameters. Figure 7 presents the distribution of the maximum value of force pressing the obstacle, which is a system property. The accuracy choice must be made as to have minimal impact on property values, which is much smaller than the impact of component parameter variation, which is under study. As can be seen in Figure 7, the time-accuracy choice is appropriate, and validated also against simulation results run with the cycle-accurate MCU model.

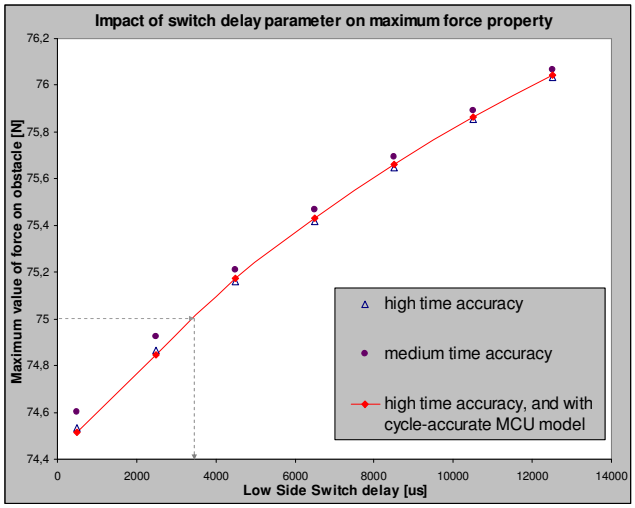

Figure 7. Impact of switch delay parameter on value of force property, with different time-accuracy choices

The performance can be observed in Table 1, so the time to simulate the complete system, in complete application scenarios, is not an issue, as long as correlated choices, for time-accuracy, model refinement, and test-case under focus are taken.

Sets of parameters are randomized to determine the impact on the same property under study. A simple 2-parameter set variation can be observed in Figure 8. The parameters (OPAMP gain and shunt resistance used for measurement of DC motor current) influence the maximum force applied on an inserted obstacle.

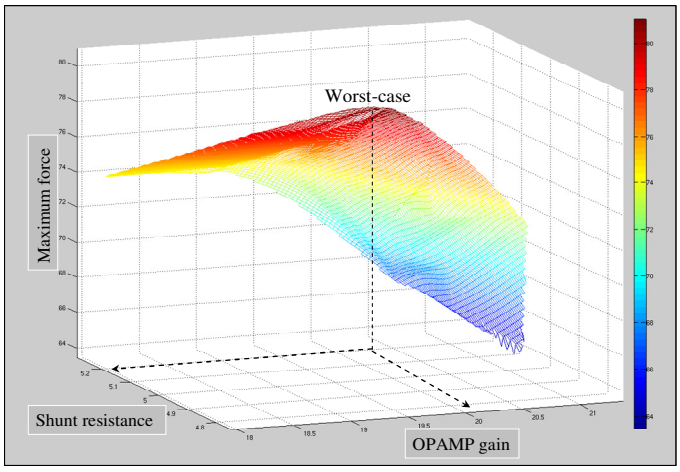

Figure 8. Impact of parameters on property

Parameters are configured as statistically distributed variables of the simulation process, within their specified range. Thus, pass/fail and/or worst-case conditions can be defined and monitored during simulation, and reported for post-processing once sets of simulations are run.

IV. CONCLUSION

The paper shows that abstraction in modeling with SystemC/SystemC-AMS can bring significant results in the verification process of complete heterogeneous systems. This was reflected on an automotive ECU, which faces several strict requirements, and complexity issues. The presented work opens the door for an improved verification environment, which will enable in-depth analysis on complete ECUs, together with their context of operation. Perspectives for future work include:

- more in-depth exploration of the stimuli and parameter spaces to validate that system properties are within admitted ranges, as long as parameters are specification-compliant.
- adaptive testing, to optimize worst-case analysis.
- software development for embedding in both models of the MCU subsystem.
- development of abstraction techniques for performing, still reliable, models.

REFERENCES

[1] IEEE 1076.1 (VHDL-AMS), http://www.vhdl.org/vhdl-ams.

[2] Open SystemC Initiative, http://www.systemc.org.

[3] SystemC TLM2.0 standard, http://www.systemc.org/downloads/standards/tlm20.

[4] SystemC-AMS, http://www.systemc-ams.org.

[5] C. Grimm, M. Barnasconi, A. Vachoux, K. Einwich: "An Introduction to Modeling Embedded Analog/Mixed-Signal Systems using SystemC-AMS Extensions", *Open SystemC Initiative*, 2008.

[6] A. Vachoux, C. Grimm, K. Einwich: "Extending SystemC to support mixed discrete-continuous system modeling and simulation", *Proceedings of IEEE International Symposium on Circuits and Systems*, vol. 5, p. 5166–5169, 2005.

[7] A. Vachoux, C. Grimm, K. Einwich: "Towards Analog and Mixed-Signal SOC Design with SystemC-AMS", *IEEE International Workshop on Electronic Design, Test and Applications*, 2004.

[8] M. Vasilevski, F. Pecheux, N. Beilleau, H. Aboushady, K. Einwich: "Modeling and Refining Heterogeneous Systems With SystemC-AMS: Application to WSN", *Design, Automation and Test in Europe*, 2008.

[9] M. Alassir, J. Denoulet, O. Romain, A. Suissa, P. Garda: "Modelling Field Bus Communications in Mixed-Signal Embedded Systems", *EURASIP Journal on Embedded Systems*, Article ID 134798, 2008.

[10] W. Scherr: "SystemC-AMS Modeling of Embedded Sensors for Automotive Applications", *Workshop on C/C++ Modeling for Mixed-Signals Embedded Systems*, http://www.eas.iis.fraunhofer.de/events/workshops/2007/cmemss/presentations.pdf, 2007.

[11] G. Noessing: "SystemC-AMS Modelling for Voice over IP Physical Interfaces", *Workshop on C/C++ Modeling for Mixed-Signals Embedded Systems*, http://www.eas.iis.fraunhofer.de/events/workshops/2007/cmemss/presentations.pdf, 2007.

[12] Functional Specification for SystemC 2.0, p. 19-23, 2002.

[13] T. Groetker, S. Liao, G. Martin, S. Swan: “System Design with SystemC”, Kluwer Academic Publishers, p. 41-48, 2002.

[14] H.D. Patel, S.K. Shukla: “Towards a Heterogeneous Simulation Kernel for System Level Models: A SystemC Kernel for Synchronous Data Flow Models”, *Proceedings of International Symposium of VLSI Systems*, 2004.

[15] A. Vachoux, C. Grimm, R. Kakerow, C. Meise: “Embedded Mixed-Signal Systems: New Challenges for Modeling and Simulation”, *Proceedings of IEEE International Symposium on Circuits and Systems*, 2006.

[16] M. Schnieringer, K. Brand: “SystemC: Key Modeling Concepts besides TLM to Boost your Simulation Performance”, http://www.design-reuse.com/articles/17877/systemc-tlm.html.

[17] S. M. Aziz, J. M. D. Tran: “Modeling for Performance: SystemC Model of a Communication Bus in a Distributed Network”, *International Conference on Information and Communication Technology*, 2007.

[18] T. Markwirth, J. Haase, K. Einwich: “Statistical Modeling with SystemC-AMS for Automotive Systems”, *Proceedings of Forum on Specification and Design Languages*, 2008.

[19] A. Rose, S. Swan, J. Pierce, J. Fernandez: “Transaction Level Modeling in SystemC”, http://www.systemc.org/downloads/standards/, 2005.