SAVESOC - SAFETY AWARE VIRTUAL PROTOTYPE GENERATION AND EVALUATION OF A SYSTEM ON CHIP

Ralph Weissnegger
Markus Pistauer
CISC Semiconductor GmbH
Klagenfurt, Austria
{r.weissnegger, m.pistauer}@cisc.at

Martin Schachner
Christian Kreiner
Kay Römer
Christian Steger
Institute for Technical Informatics
Graz University of Technology (TU Graz), Austria
{schachner, christian.kreiner, roemer, steger}@tugraz.at

ABSTRACT

The electrification of today's vehicles and the high amount of new assistance features imply more and more complex systems. The sensing and controlling of these systems is the work of the highly distributed and connected electronic control units. To keep pace with the fast growing automotive market, reusability of components and features is today the key to reduce costs and time-to-market. Especially when systems are safety-critical and demand reliability, new methods and tools are thus essential to support the reusability aspect in the development process. A model-based approach, in conjunction, moreover helps to communicate between different stakeholders, provides different views and serves as a central storage of information. Through applying reliability analysis and simulation-based verification methods on our hardware model and furthermore automatic generation of a first virtual prototype, we are able to reduce the tools involved, thus resulting in correctness, completeness and consistency of the entire system.

Keywords: UML, functional safety, ISO26262, cyber-physical system, virtual prototyping.

1 INTRODUCTION

In the world of today, the amount of embedded electrical/electronic (E/E) systems in various domains is highly increasing to a very great extent. When we think about the complexity of the past few years, it is apparent that new applications have emerged in which systems are not only interacting with each other but also have impact on the physical world, the so-called cyber-physical systems. Depending on their application, they must fulfill different requirements ranging from timing constraints, performance behavior, low power consumption, thermal or even working capability under different environmental conditions. The point here is, we live in a world where cyber-physical systems are ubiquitous, they have impact on our daily lives and the malfunction of these systems can lead to severe damage or injury to people. We must thus assure the dependability of these systems.

This is even more obvious when we turn to the automotive domain. It can be observed that there is a shift towards fully E/E systems resulting from the trend to electric vehicles.The sensing and controlling is the work of the highly distributed electronic control units (ECU) and it is no surprise, that through all these new features in cars, more than 100 of these microcontrollers (Charette 2009) are currently integrated in a modern car. This situation has also an impact on the amount of software in cars today, which can total 150 million lines of code (Eitdigital 2016). The industry is facing new problems through the emergence of many new (assistance-) features that are also influencing each other. In turn, this raises the complexity level in the design, development and verification of complex systems and imposes an enormous effort for engineers in developing their applications. In terms of safety, these systems must fulfill standards such as the ISO26262 (functional safety for road vehicles), (ISO 26262 2011). Since this standard is now treated as state of the art in court, OEMs and their suppliers are required to develop and test their systems towards the recommended measures and methods.

When we discuss the design of a system, a single modeling language immediately springs mind, the Unified Modeling Language (UML), (Group 2015). Having the roots in the software domain, UML paved the way and established a model-based thinking in various engineering domains, far across the borders of conventional software design. Since UML comes with several extensions such as MARTE (OMG 2016), SysML (Omg 2015) or EAST-ADL (EAST-ADL 2017), furthermore engineers from different domains can use the full potential of an object-oriented approach. MARTE was introduced to overcome the enormous complexity issues in the design of real-time and embedded systems. It provides capabilities to model hardware, software as also as system design.

Since a state of the art car of today exists not only in one single version, but rather in several hundreds of variants all with different features, each of these must be exhaustively tested to fulfill the standards. Millions of test kilometers must be driven to ensure the reliability of a car and it is neither economic nor safe to test them in a real environment (Maurer, Gerdes, Lenz, and Winner 2015). Simulation plays an ever increasing and important role in the verification of the modern car because of its advantage in easily varying the virtual environment and to representing the car in different variations, not least from an economical perspective. Simulations can be done in early development phases, where the detailed implementation of a function is still undecided and furthermore, on platform specific models where the hardware and software are explicitly defined (virtual prototype). Applied verification methods and tests can be monitored, reproduced and rerun every time. Another advantage of simulation is that it cannot only be run day and night, but also massively in parallel. A specification and simulation language, which shares the same philosophy as the UML/MARTE approach, is SystemC (Accellera Systems Initiative 2017). Like UML, it shares the MDA (Model Driven Architecture) approach, starting from a computational independent system design, down to hardware and software design.

The ISO26262 recommends different verification methodologies used for the hardware platform use. These includes design walk through, FTA/ FMEA, hardware architectural metrics evaluation but more importantly, hardware prototyping and simulation for higher ASIL levels. These simulations, including virtual prototyping, can then be used for further hardware verification methods such as fault injection test, which is currently the key for testing the reliability of the hardware. Several research institutes are now working on executing fault injection, also on higher abstraction level such as transaction level modeling (TLM). This has the advantage that this method can be applied on faster simulation models without losing information from the more detailed models (RTL, register transfer level). Virtual prototyping has the benefit that embedded software can be tested much earlier, before a first real hardware prototype is available. Changes on a virtual hardware design are much faster than changes on the real platform, which takes weeks or months of redesign and production, which in turn has impact on time to market. With intensive simulation, corner cases but also long term reliability errors can be encountered, which also prevents costly product recalls. Environmental impacts on the virtual prototype can be simulated and reproduced, where real testbeds are not capable of this kind of verification. The drawback, a complex virtual prototype (VP) is not developed overnight. It takes a lot of effort, experienced designers and engineers to build a so called digital twin of the actual hardware. Our proposal is thus to reuse models for virtual hardware prototyping from open libraries such as Open Virtual Platform (OVP), (OVP 2017).

In this work we present a novel design and development flow for a safety aware virtual prototype. This design flow is conform to the ISO26262 standard and meets all its requirements to produce a reliable product in the end. The whole system, from a first functional specification down to hardware design, is specified by standardized modeling languages. Before the virtual prototype is generated from the hardware specification, several reliability analysis methods are applied and executed on this specification. This allows an early evaluation of the hardware design regarding safety, before testing the prototype in simulation. Furthermore, we show the integration of the generated virtual prototype into the system design, to verify the interfaces and its functionality. The whole methodology is available through the developed design and verification framework named SHARC (Simulation and Verification of Hierarchical Embedded Microelectronic Systems), (CISC 2017).

2 RELATED WORK

The authors of (Macher, Stolz, Armengaud, and Kreiner 2015) aimed at achieving consistency of information between several tools involved in the development process, through to a single source of information principle. In the goal of achieving dependability (safety, security) in the development process between different teams and stakeholders, they decided against a document-centric approach and used the capabilities of UML and SysML for their design. This in turn improved consistency, correctness, and completeness of the entire system under development. The toolchain in this approach focused more on the system and software development and did not take hardware development into account. The authors also propose to update their profile in order to work efficiently on hardware development as such.

To overcome the issues with consistency within design and simulation models, the authors of (Sporer, Macher, Armengaud, and Kreiner 2015) proposed a model transformation framework between SysML and Matlab/Simulink. They support a consistent and traceable refinement from the early concept phase through to software implementation and this in a bidirectional manner. The authors also claim that a model-based design helps to enable different views for different stakeholders, different levels of abstraction, and central storage of information. Nevertheless, the author's focus was more on the software architecture generation from system design rather than on the requirements for hardware design.

Popular approaches (Adler, Domis, Höfig, Kemmann, Kuhn, Schwinn, and Trapp 2011) and (David, Idasiak, and Kratz 2009) have shown that UML as modeling language can be efficiently used with analysis and verification methods such as FMEA (failure mode and effect analysis), fault tree analysis (FTA), design walk through (Gvero, 2013), code-generation and many more. The drawback of UML, in terms of simulation to verify the system behavior is, that code-generation can only be done at a very late stage or even at the end of the design process, when all details are very well known. Later changes in design are costly and result in inconsistent models and furthermore reverse-engineering is an error prone and cumbersome task. The majority of components in new projects are reused and simply extended by the addition of new features to reduce costs and time-to market. The reuse of whole safety concepts, well-trusted designs and mechanisms is thus becoming more important to reduce the effort required for developing complex systems. This situation prompts the urgent demand for new techniques to simulate the behavior in early development-phases by reusing verified system components.

3 USE CASE

Throughout this paper we will demonstrate our methodologies on a relevant problem in today's automotive domain, a battery management system for Li-ion powered electrical vehicles. This industrial use case was provided by CISC Semiconductor GmbH, based in Austria and the United States. This use case will help to illustrate more fully the innovative capabilities and benefits of our approach. As more and more vehicles are now powered by Li-ion batteries, the challenge for engineers to ensure reliability and fault tolerance is also greatly increasing. It is crucial for ensuring safe operating conditions of a battery that monitoring systems such as the battery management systems (BMS) measure the voltage, temperature and current of the battery very precisely. This information must be forwarded to a vehicle-wide controller network to ensure a reliable and fully utilized system. Problems with overheating or even explosions have been frequent in the past. The main cause of these problems was an excessively high energy intake from regenerative braking or harsh environmental conditions. Management systems and mechanisms are thus essential to assure that persons are not put at risk and that no damage is caused.

To achieve a first executable specification of our use case, we used the methodologies provided in (Weissnegger, Schuß, Kreiner, Pistauer, Kay, and Steger 2016), to connect the UML/MARTE design models with reusable simulation models (model library) in SystemC (-TLM)/ SystemC-AMS. These functional models are early executable models on a high level of abstraction and can be refined, depending on their purpose, through to more detailed models. The provided model library contains analog and digital models to gain an early and fast evaluation of the functional specification on system level. With this approach moreover, we eliminate the tedious task of exporting our gathered data to other simulation tools such as Matlab/Simulink and leave the UML/MARTE design as a single source of information. Furthermore, we rely on standardized and open modeling and simulation languages and keep the costs for licenses low.

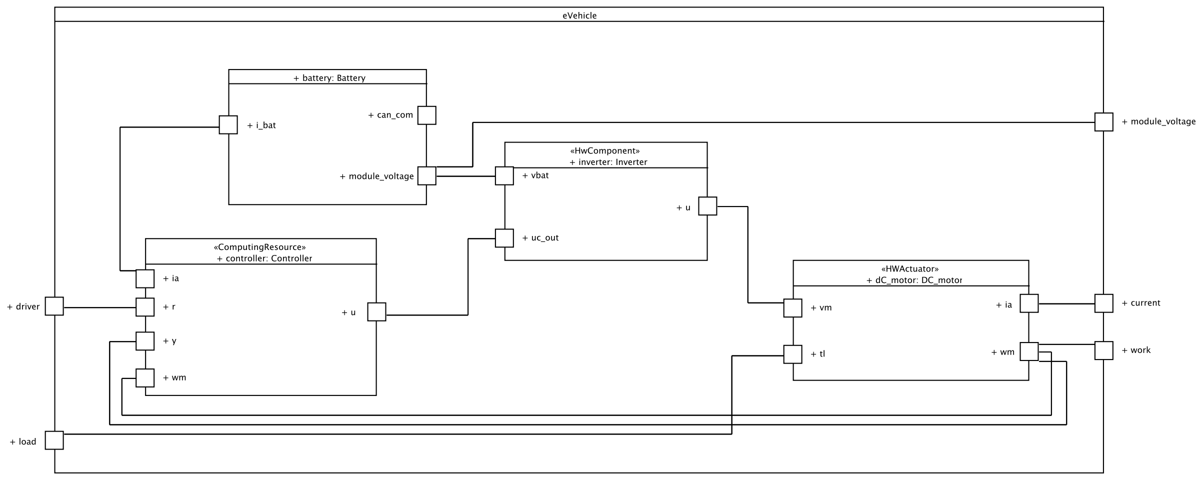

Figure 1: Overall system level description of the electric vehicle use case with UML/MARTE.

The overall system of the eVehicle is depicted in Figure 1. For reasons of simplification, we only consider the major components of the electric vehicle, for the analysis of the battery and the BMS. This includes the battery pack in Li-ion technology, the BMS which measures voltage and temperature of the battery, an inverter ECU, a controller and the electric motor model. Two main factors influence the behavior of the eVehicle. The driver provides the desired speed (rounds per minutes) and on the other hand the load on the motor shaft. These stimuli can be set according to standardized maneuvers such as the New European Drive Cycle (NEDC), or the newer standards known as the worldwide harmonized light-duty vehicles test procedure (WLTP), which will be introduced in 2017.

4 SPECIFICATION OF THE HARDWARE PLATFORM

From the gathered information of our functional and executable specification we now delve deeper into the hardware design. Since MARTE and SystemC share the same philosophy to move from system to hardware and software development, we are now able to specify, through refinement, the internal architecture of our BMS platform. As the major goal of this work is to build a ISO26262 safety aware platform, this is an important step in the development process and takes a lot of effort into account (Kreiner 2015), since many different methods and measures have to be applied to the hardware platform to guarantee the final result of a reliable and safe product.

Since the OVP virtual prototype consists more or less of a set of SystemC -(TLM) files including a TCL script, one goal of this work is to include the whole generation of the virtual prototype into a seamless design flow for safety critical systems. Furthermore, the design and configuration of the VP on a graphical modeling standard, in this case UML/MARTE. This approach brings the advantage of being able to evaluate and analyze the configuration of the hardware design before the actual prototype is generated from UML models. OVP itself does not support verification regarding safety, nor is OVP till now embedded in a seamless design flow.

As mentioned in the previous chapter, UML/MARTE provides several levels of detail for the specification of the hardware platform. The hardware resource model (HRM) package provides several models to describe subsystems such as HwProcessor, HwBus, HwDevice or HwMemory in a logical and physical way. Although, UML/MARTE provide a rich set of different stereotypes to describe the hardware platform, it does not provide the level of detail for the hardware configuration as expected by the OVP methodology. Several mandatory properties such as the BusInterface or Memory mapped or the VLNV (vendor, library, name, version) principle are by now not supported in the MARTE standard.

To overcome these issues, we rely on the specification of the IP-XACT standard, which helps us to define our platform with a sufficiently high degree of detail for the generation of the virtual prototype. Both, IP-XACT and OVP rely on the VLNV principle for structuring the models. We thus built on approaches such as (André, Mallet, Khan, and de Simone 2008) to extend the MARTE standard by properties in IP-XACT for hardware description.

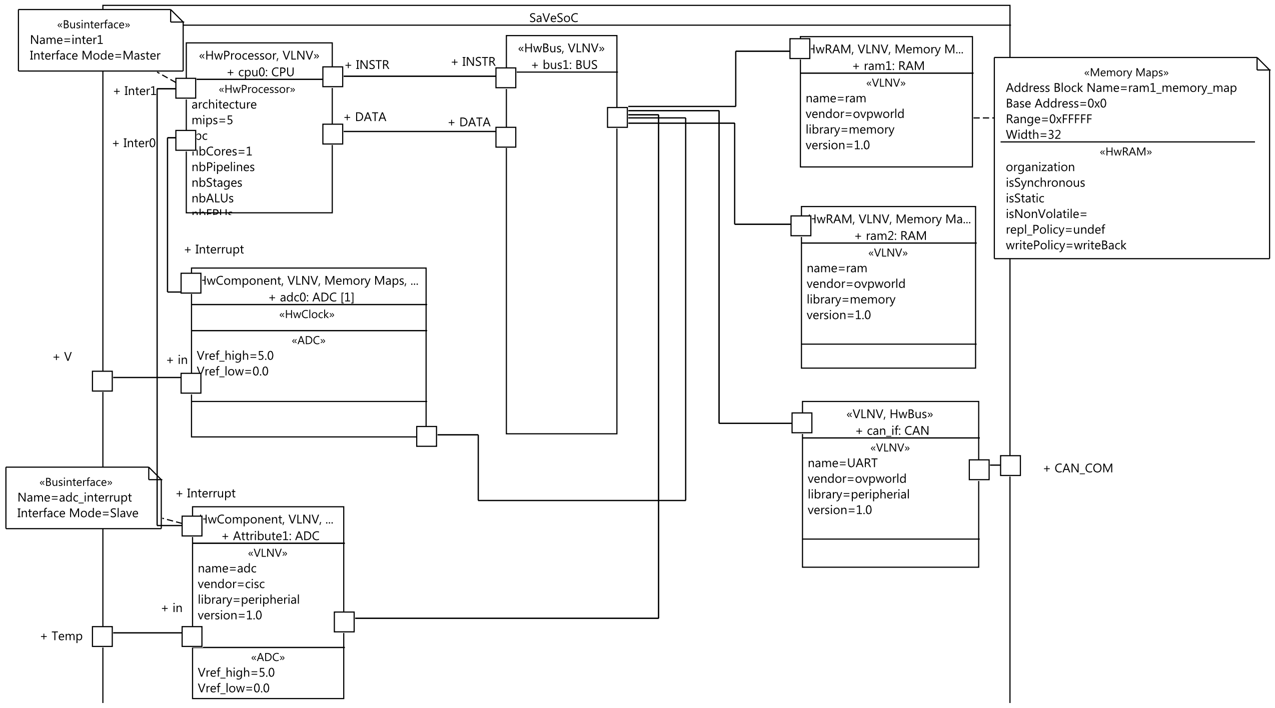

Figure 2: Hardware platform of SaVeSoC including IP-XACT extensions for MARTE.

Figure 2 depicts the composed structural architecture model of the platform consisting of a processor, bus, memory, CAN and two ADCs. For simplification we only show the major components of the design of the battery management system. The designer can easily compose his system by using the standard models from the MARTE library such as HwProcessor for the CPU, HwI_O or HwComponent for peripherals, HwRam for memory or HwBus for the internal bus or CAN bus. Depending on the nature of the component, the designer is able to extend the component with specific hardware properties in the IP-XACT standard such as MemoryMaps and BusInterface. The properties from the IP-XACT standard are additionally shown in the description of the MARTE model. Further extensions for IP-XACT such as VLNV can be added to the hardware components as well.

5 GENERATION AND EVALUATION OF THE VIRTUAL PROTOTYPE

5.1 Evaluation of the Hardware Configuration

Based on the specification of our hardware platform in Fig. 2 we are able to perform different evaluation methods on our hardware design. As mentioned in the previous chapters, there are several methods which can be used to evaluate the reliability and safety of our hardware architecture, one of these is the hardware architectural metrics evaluation. The hardware architectural metrics are handled in Part 5 (ISO 26262 2011) of the ISO26262, product development at the hardware level. This evaluates the hardware architecture of the item against the requirements for fault handling. This part also includes guidance on avoiding systematic and random hardware failures by means of appropriate safety mechanisms. Each safety-related hardware element is analyzed regarding single point (SPFM), residual and multiple point faults (LFM). It also describes the effectiveness of the hardware architecture in coping with random hardware failures (PMHF). Each hardware part is to be protected by means of safety-mechanisms. The diagnostic coverage gives evidence of the effectiveness of these mechanisms. Whether the item (system or array of systems according to ISO26262) passes or fails a given ASIL is also a result of the hardware architectural metrics evaluation. In order to achieve a specific ASIL, the values from Table 1 must be met (FIT- failure in time).

Table 1: Architectural Metrics - evaluates whether the hardware achieves a certain ASIL, according to ISO26262.

	ASIL B	ASIL C	ASIL D
SPFM	$\geq 90\%$	$\geq 97\%$	$\geq 99\%$
LFM	$\geq 60\%$	$\geq 80\%$	$\geq 90\%$
PMHF	$< 10^{-7}h^{-1}$	$< 10^{-7}h^{-1}$	$< 10^{-8}h^{-1}$

To perform this evaluation on our hardware specification, we built on approaches such as (Weissnegger, Pistauer, Kreiner, Römer, and Steger 2015), (Weissnegger, Pistauer, Kreiner, Kay, and Steger 2015) and (Das and Taylor 2016). These approaches use the capabilities of UML/MARTE and IP-XACT to perform several quantified methods to evaluate the reliability of the hardware platform. Furthermore, they defined an extension in IP-XACT for fault models, which can be reused for different hardware platform configurations. These reuse strategies are commonly used in industry and are known as clone and own. The existing and reused safety models can give important feedback through an early safety assessment for a modified platform or even a new product. Changes and adaption on the overall platform must be taken into account, of course when reusing safety artifacts during development. Since the main contribution of this paper is the generation and integration of the virtual prototype into a seamless design and development flow, we do not go into detail on the hardware evaluation. A more detailed information is given in (Weissnegger, Pistauer, Kreiner, Römer, and Steger 2015).

5.2 Generation of the Virtual Prototype

One of the outlined goals in this work is to develop and test embedded software within the development phases, on a realistic hardware prototype of the target system, before the actual platform is available. Embedded software is often written in a desktop environment, on a general purpose operating system of the host system. This approach often differs often significantly from the target platform and parts of the written software need to be adjusted. One way to deal with this problem is the usage of an instruction set simulator (ISS) and hardware visualization. Due to a vendor variety of components within SoC design, the simulation can be very difficult. Hardware emulators are very popular for this issue, but require the detailed RTL description of the developed system, which is a contradiction to the outlined goal of a homogeneous development environment. OVP makes it possible to create virtual platform models, with SystemC TLM 2.0 support. OVP models can be executed much faster than their counterparts developed in RTL, since their level of abstraction is higher but still appropriate for modeling purposes.

Relying on the ideas of model driven engineering the virtual hardware prototypes is modeled in a graphical way. After a comprehensive hardware safety analysis, the developed system design is moreover generated and integrated into the existing modeling framework. Therefore a methodology was defined, which converts the according UML platform description into a proprietary TCL file, containing the necessary information to generate source code with OVPs iGen tool. For the graphical platform description we rely on the capabilities of UML/MARTE. Additional properties, which exceeded MARTEs capabilities got defined by additional IP-XACT stereotypes. Both TCL and IP-XACT are relying on the VLNV principle. Compiled to a shared object the virtual platform can be simulated along with other components from the provided library. This approach guarantees a seamless integration of platforms into the existing simulation framework and enable the co-simulation of a virtual hardware platform together with a model of the physical environment in which it is embedded. The OVP iGen converter takes the information from the TCL script and generates a full SystemC TLM2.0 platform using the modular components of the OVP library. The resulting virtual prototype can then be used for further hardware simulations, such as fault-injections on TLM level.

5.3 Integration and Verification of the Virtual Prototype

Since our whole methodology (from functional specification through to hardware and software design) relies on the same modeling language and furthermore the same hardware description language respectively system-modeling language, we are easily able to reapply our generated safety aware virtual prototype into the functional specification of the system design. After generation, the hardware description of the SaVeSoC platform with the whole interface specification is added to the UML model library and can be reapplied to the system design including the generated simulation files in SystemC. This saves time in terms of integration effort, when testing the virtual prototype on the functionality of the whole system, as recommended in ISO26262. System-level testbenches can also be reused for the verification of the entire system including the integrated VP. The whole process is depicted in Figure 3. By adding a smaller V-model to the traditional approach, we are closing the technological and organizational gap between system design and hardware development which exists in today's tool flows. Since our design and simulation languages in use, share a seamless development flow, no information transfer is needed between those design levels. Changing requirements in the specification can also be easily and efficient tested for the virtual prototype.

Figure 4 depicts the resulting system level description including the battery and new defined BMS hardware. The battery component is no longer a black box where the functionality is described in SystemC. It is now a white box, where the detailed hardware of the battery component is specified. It consists of the SaVeSoC element, which is the hardware platform of the BMS including an application for plausibility checks. It measures the voltage and temperature of the batterypack over two ADC and computes the state of charge

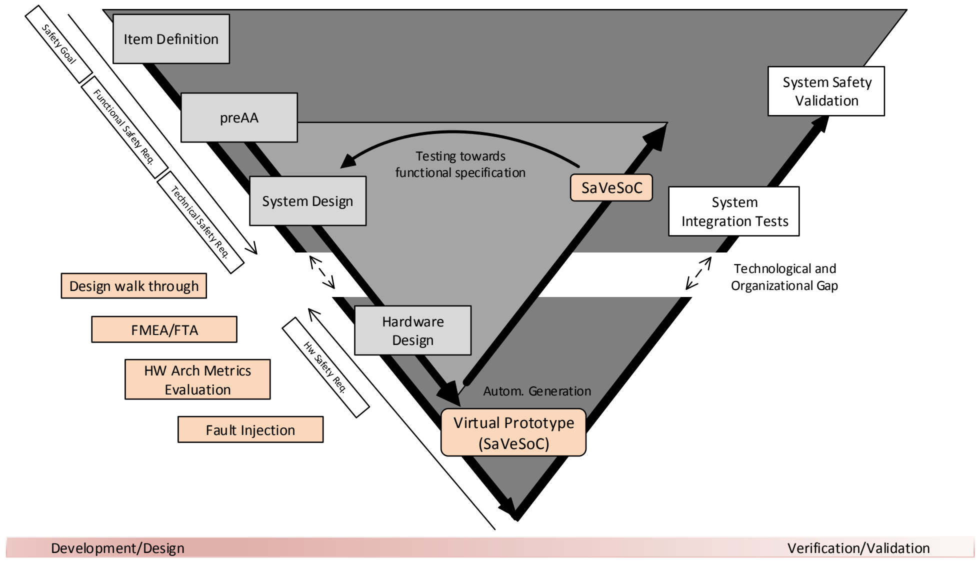

Figure 3: Process of SaVeSoC integration into functional specification.

and stage of health. The interfaces of the battery model remained the same, since no changes have been made to the interface description. Since we are relying on the same simulation engine, the virtual prototype can now be easily tested on a higher level environment, which also speeds up the overall simulation time. A challenge when integrating the VP to the functional specification was the communication interface between the functional simulation environment of SystemC-AMS and the OVP platform, because these platforms are mainly used to process data measured from embedded sensors. We thus implemented communication channels which guarantee data exchanges between different SystemC dialects. The authors of (Lonardi and Pravadelli 2015) rely on the idea that the simulation engine is executed in a single process, with the SystemC and OVP simulator running in different software threads. The authors mainly focused on the capability to co-simulate SystemC RTL models, with either QEMU or OVP. To avoid overheads from the use of sockets, they established the communication channel via a shared memory and synchronization mechanism. Furthermore they developed a SystemC bridge to enable the connections to the external hardware simulator. Since the original component library is not meant to be cycle accurate, the main focus was set to establish the communication between the existing SystemC-AMS components and the TLM2.0 models provided in the OVP. The paper (Damm, Grimm, Haas, Herrholz, and Nebel 2008) describes the main properties of both sides and how synchronization is performed internally. SystemC AMS provides so-called converter ports to establish a connection between timed data flow (TDF) modules and an ordinary SystemC signal. In the event of an access to such a port, the AMS kernel triggers an interrupt, which causes a context switch to the SystemC/OVP simulator. The crucial part of the implementation was therefore the conversion from SystemC-AMS linear signal flow (LSF) to TLM and how to handle the data stream of an arbitrary LSF module to the ADC peripheral of the OVP platform.

Figure 5 shows the used components for converting the initial LSF signal to a TLM signal. The sca_lsf::sca_-tdf_sink is a component of the SystemC AMS library, used to sample arbitrary input data and convert it to TDF. The self-defined adc_module processes the TDF signal so that it is written to the Adin port of the adc0 within its processing() procedure. The adc0 is part of the OVP library, and was adapted slightly to meet our needs. The port of the ADC is implemented with a call back function, which triggers the conversion of the ADC. Afterwards it can be read with the implemented driver, executed on the CPU. Since the OVP module requires the usage of a certail tlm_signal_port as TLM target socket for the communication, we adapted and advanced the approach of (Damm, Grimm, Haas, Herrholz, and Nebel 2008) to meet our needs.

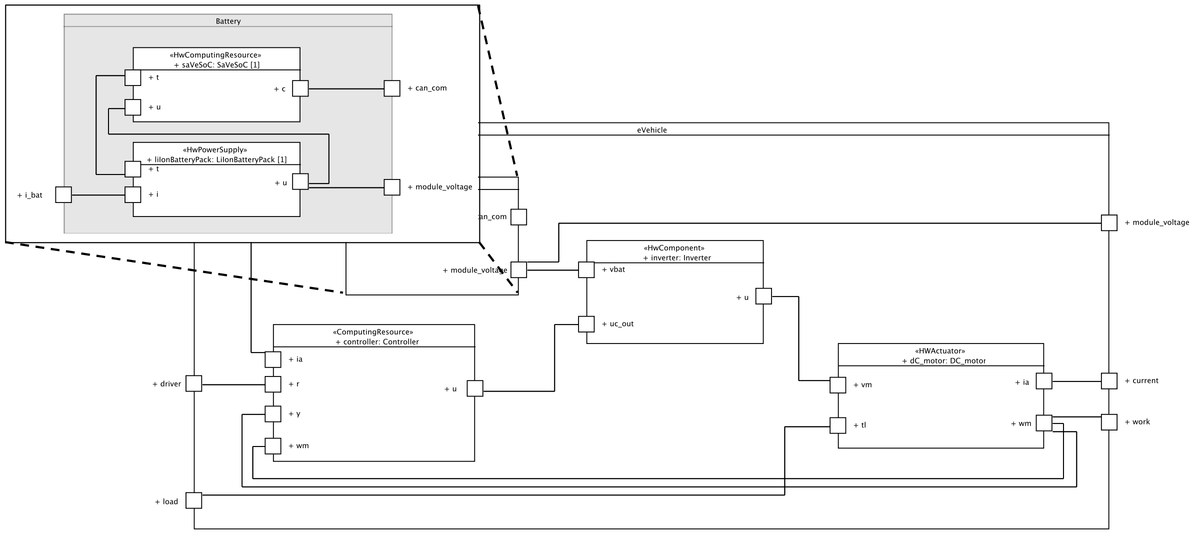

Figure 4: Virtual Prototype integration into the functional specification of the system design to verify the functionality.

Secondly, we omitted the suggested internal FIFO regarding data loss. This can be reasoned by reference to the constant sampling rate of the AMS simulation. A fixed size FIFO would nevertheless lead to data loss if the processor does not execute sufficient instructions per second.

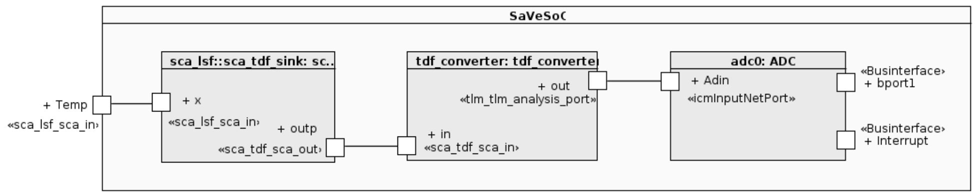

Figure 5: Communication channel for the interaction between SystemC AMS and OVP TLM2.0.

6 CONCLUSION

In this paper we presented a seamless design and verification process for safety-critical systems. A standardized modeling language based on UML was used to represent the design flow, from functional specification down to hardware and software. This model-based approach eases the communication between different stakeholders involved in the development process and serves as a single-source of information. Through tight integration of recommended safety analysis methods such as FTA, FMEA, hardware architectural metrics and simulation-based verification, we achieved consistency, correctness and completeness throughout the development process. The hardware architecture was evaluated by extensions to a well-known hardware description in the industry, IP-XACT. Existing and reusable hardware description was used for system design and integration. Our tool-aided method helped to speed up the evaluation process, and to reduce costs through reusability. The evaluated hardware description was then used to automatically generate a safety aware hardware virtual prototype, which was used to test correctness regarding the functional specification. This closes the technological and organizational gap in today's toolchain of safety-critical system development. Furthermore, this early virtual prototype can be used for fault-injection tests, as recommended by the functional safety standard. In addition our approach was developed as a plugin for the Eclipse, with the result that every Papyrus UML editor can be used for safety aware development of cyber-physical systems, simply by adding our plugin. This tool is named SHARC (Simulation and verification of HierARChical embedded microelectronic systems) is to be published for download and is also used for educational purposes. Further work will include the automatic generation of TLM fault-injection tests for the generated virtual prototype.

ACKNOWLEDGMENTS

A part of the work has been performed in the project eRamp (Grant Agreement N621270), co-funded by grants from Austria, Germany, Slovakia and the ENIAC Joint Undertaking. Some parts have also been experimented in the OpenES CATRENE Project: CA703 - 2013 research program supported by the FFG (Austrian Research Promotion Agency), project-number 843380 in tight cooperation with CISC Semiconductor.

REFERENCES

Accellera Systems Initiative 2017. “SystemC”. http://www.accellera.org/. Accessed February, 2017.

Adler, R., D. Domis, K. Höfig, S. Kemmann, T. Kuhn, J. P. Schwinn, and M. Trapp. 2011. “Integration of component fault trees into the UML”. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* vol. 6627 LNCS, pp. 312–327.

André, C., F. Mallet, A. M. Khan, and R. de Simone. 2008. “Modeling SPIRIT IP-XACT with UML MARTE”. *Proc. DATE Workshop on Modeling and Analysis of Real-Time and Embedded Systems with the MARTE UML profile*.

Charette, Robert N. 2009. “This Car Runs on Code - IEEE Spectrum”. http://spectrum.ieee.org/transportation/systems/this-car-runs-on-code.

CISC 2017. “CISC Semiconductor GmbH”. https://www.cisc.at/. Accessed February, 2017.

Damm, M., C. Grimm, J. Haas, A. Herrholz, and W. Nebel. 2008, sep. “Connecting SystemC-AMS models with OSCI TLM 2.0 models using temporal decoupling”. In *2008 Forum on Specification, Verification and Design Languages*, pp. 25–30.

Das, N., and W. Taylor. 2016. “Quantified fault tree techniques for calculating hardware fault metrics according to ISO 26262”. In *2016 IEEE Symposium on Product Compliance Engineering (ISPCE)*, pp. 1–8.

David, P., V. Idasiak, and F. Kratz. 2009. “Towards a better interaction between design and dependability analysis: FMEA derived from UML/SysML models”. *Safety, Reliability and Risk Analysis: Theory, Methods and Applications - Proceedings of the Joint ESREL and SRA-Europe Conference* vol. 3 (August 2015), pp. 2259–2266.

EAST-ADL 2017. “EAST-ADL Association”. http://www.east-adl.info/.

Eitdigital 2016. “FORD GT - Lines of code”. https://www.eitdigital.eu/news-events/blog/article/guess-what-requires-150-million-lines-of-code/.

Group, O. M. 2015. “OMG Unified Modeling Language TM (OMG UML), Superstructure v.2.5”. *InformatikSpektrum* vol. 21 (May), pp. 758.

ISO 26262 2011. “Road vehicles-functional safety-Part 5: Product development at the hardware level”.

Kreiner, C. 2015. “Trident Architectural Views: A Pattern for Dependable Systems Design”. In *Proceedings of the 20th European Conference on Pattern Languages of Programs*, EuroPLoP ’15, pp. 18:1—-18:9. New York, NY, USA, ACM.

Lonardi, A., and G. Pravadelli. 2015. *On the Co-simulation of SystemC with QEMU and OVP Virtual Platforms*, pp. 110–128. Cham, Springer International Publishing.

Macher, G., M. Stolz, E. Armengaud, and C. Kreiner. 2015. “Filling the gap between automotive systems, safety, and software engineering”. *e & i Elektrotechnik und Informationstechnik* vol. 132 (3), pp. 142–148.

Maurer, M., J. C. Gerdes, B. Lenz, and H. Winner. 2015. *Autonomes Fahren: Technische, rechtliche und gesellschaftliche Aspekte*. Springer Open. Springer Berlin Heidelberg.

Omg 2015. “OMG Systems Modeling Language (OMG SysML ^{TM}) v.1.4”. *Source* (June), pp. 260.

OMG 2016. “UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems”. Technical report, Object Management Group.

OVP 2017. “Open Virtual Platform”. http://www.ovpworld.org.

Sporer, H., G. Macher, E. Armengaud, and C. Kreiner. 2015. “Incorporation of Model-Based System and Software Development Environments”. *Proceedings - 41st Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2015*, pp. 177–180.

Weissnegger, R., M. Pistauer, C. Kreiner, R. Kay, and C. Steger. 2015. “A Novel Method for Fast Evaluation of Cyber-Physical Systems in Compliance with Functional Safety”. In *Euromicro Conference in Software Engineering and Advanced Applications - Proceedings of the Work in Progress Session*, pp. 24–25. Funchal, Madeira, Johannes Kepler University Linz.

Weissnegger, R., M. Pistauer, C. Kreiner, K. Römer, and C. Steger. 2015. “A novel method to speed-up the evaluation of cyber-physical systems (ISO 26262)”. In *2015 12th International Workshop on Intelligent Solutions in Embedded Systems (WISES)*, pp. 109–114. Ancona, Italy.

Weissnegger, R., M. Schuß, C. Kreiner, M. Pistauer, R. Kay, and C. Steger. 2016. “Bringing UML / MARTE to life : Simulation-based Verification of Safety-Critical Systems”. In *2016 Forum on Specification and Design Languages (FDL)*. Bremen, Germany.

AUTHOR BIOGRAPHIES

RALPH WEISSNEGGER received his Bachelor’s and Master’s degree in Telematics (Information and Computer Engineering) from Graz University of Technology, Austria, in 2013. Since 2014 he is with the Institute for Technical Informatics at Graz University of Technology where he is working towards his Ph.D in Electrical Engineering. His research interests include design and verification of HW/SW codesigns, especially safety-critical systems. His Ph.D is done in tight cooperation with CISC Semiconductor GmbH, r.weissnegger@cisc.at.

MARKUS PISTAUER (CEO, Member IEEE) holds a Master degree in Electrical and Electronic Engineering (1991) and a Ph.D. degree in Electronic and Control Engineering (1995), both from Graz University of Technology, Austria. From 1995 to 1999 he worked at Siemens AG (Semiconductor Division, now Infineon Technologies) and also as Professor at University of Applied Sciences, Carinthia. He has founded CISC Semiconductor in 1999 where he acts as CEO and in 2012 CISC Semiconductor Corp. in Mountain View, CA, USA. He is author and co-author of more than 70 papers published and holds several patents in the area of embedded systems.

MARTIN SCHACHNER is a Computer Science student at Graz University of Technology, Austria. In 2016 he graduated his bachelor with distinction and is currently working towards his master degree in the field of pervasive computing and computational intelligence. Since 2015 he is involved in European projects as a project assistant at the Institute for Technical Informatics (ITI). His work involves the research on new development methodologies for safety critical systems.

CHRISTIAN KREINER graduated and received the Ph.D. degree in electrical engineering from Graz University of Technology, in 1991 and 1999, respectively. From 1999 to 2007, he served as head of the R&D Department, Salomon Automation, Austria, focusing on software architecture, technologies, and processes for logistics software systems. He was in charge to establish a company-wide software product line development process and headed the product development team. There, he led and coordinated a long-term research program together with the Institute for Technical Informatics of Graz University of Technology. There, he currently leads the Industrial Informatics and Model-based Architectures group. His research interests include systems and software engineering, software technology, and process improvement.

KAY RÖMER is professor at and director of the Institute for Technical Informatics at Graz University of Technology, Austria. Before he held positions of Professor at the University of Lübeck in Germany, and senior researcher at ETH Zürich in Switzerland. Prof. Römer obtained his Doctorate in computer science from ETH Zürich in 2005 with a thesis on wireless sensor networks. His research interests encompass wireless networking, fundamental services, operating systems, programming models, dependability, and deployment methodology of networked embedded systems, in particular Internet of Things, Cyber-Physical Systems, and sensor networks.

CHRISTIAN STEGER received 1990 the Dipl.-Ing. degree and 1995 the Dr.techn. degree in Electrical Engineering, Graz University of Technology, Austria. Graduated from Export, International Management and Marketing course in June 1993 at Karl-Franzens-University of Graz. In January 2010 he completed the Entrepreneurship Development Program at MIT SLOAN SCHOOL OF MANAGEMENT in Boston. He is strategy board member of the Virtual Vehicle Competence Center (ViF, COMET K2) in Graz. From 1989 to 1991 Software Trainer and Consultant at SPC Computer Training Ges.m.b.H., Vienna. From 1990 to 1991 Research Engineer at the Institute for Technical Informatics, Graz University of Technology. Since 1992 Assistant Professor at the Institute for Technical Informatics, Graz University of Technology. From October 2010 to February 2011 he had a substitute professor at the University of Saarland (Chair "Reactice system"). He was project leader of the BMVIT (FIT-IT) funded projects POWER-CARD, LOWSOM, SIMBA, HyPerSec, DAVID and scientific leader in POWERHOUSE, CoCoon, META[:SEC:], OpenEs, and SmartLX/ASIDS (FFG Competence Headquarter Program). Currently he is involved in the European ECSEL project IoSense and eRamp. He heads the HW/SW codesign group at the Institute for Technical Informatics. His research interests include embedded systems, HW/SW codesign, HW/SW coverfication, SoC, power awareness, smart cards, and multi-DSPs. Christian Steger has supervised and co-supervised over 180 master theses and co-supervised 32 PhD students, and published more than 220 scientific papers as author and co-author. He is member of the IEEE and member of the OVE (Austrian Electrotechnical Association).