2021 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE OCTOBER 26-27, 2021

A comparison of methodologies to simulate mixed-signal IC

Exploiting both virtual and real HW prototypes

Simone Fontanesi [1], Karsten Einwich [2], Paul Ehrlich [3], Gaetano Formato [4], Andrea Possemato [5]

[1,4,5] Infineon Technologies Austria AG, Villach, Austria ({*Simone.Fontanesi, Gaetano.Formato, Andrea.Possemato} @infineon.com*)

[2,3] COSEDA Technologies GmbH, Dresden, Germany ({*Karsten.Einwich, Paul.Ehrlich*}*@coseda-tech.com*)

***Abstract*—Providing the best product quality and doing it with aggressive time to market is fundamental in today´s market. Virtual and hardware prototypes support such approach in multiple ways. They enable an early understanding of the system functionality and the assessment of usefulness, completeness and consistency of requirements. In this way a virtual prototype supports much better the discussion with the customer compared to any paper document, and this improves the final customer satisfaction significantly. Later the model can be refined and used as an executable reference for the implementation phases and helps in this way to guarantee that the final product fulfills the defined system requirements. The challenges for a virtual prototype at system level are the simulation performance and the effort for creating and validating the model. At system level a virtual prototype is used to validate complete application scenarios with thousands of configurations for numerous use cases. This paper will discuss the pro and cons of different approaches and methodologies to meet these challenges on the example of magnetic sensor projects.**

Keywords—Virtual Prototype, SystemC, SystemC AMS, HiL

I. Introduction

If a behavioral model is developed at the beginning of the development process, it is possible to assess the clarity and completeness of the customer requirements and start with the product definition. In this stage a high-level model is developed by the application and concept engineers (E.g., SystemC model), and the findings from the simulations are used to iterate the discussion with the customer. The model can be in the form of a virtual or a hardware prototype, or both. A possible approach is to convert the virtual SystemC prototype into HDL via High Level Synthesis [1], in order to generate a hardware prototype. The model is then refined as the development continues and the architecture of each block is defined in a more and more detailed way. The model hence enables the verification of the hardware requirements for each block, even testing it in a real system in case of a hardware prototype. The design and post-silicon verification phase can then benefit of a fully verified model, to be used as a golden reference for the design activities and for the post-silicon verification measurements. Finally, the model can be used to support the customers during design-in activities. Here it is possible to stimulate it with inputs derived from real application scenarios and use the obtained outputs as inputs for the ECU algorithm.

Out of this brief introduction, we can derive the main use cases for the model: concept verification, design validation, HW prototyping and customer support. In all these cases, given the high number of system level use cases and sensor configurations, to have a good coverage the simulation time may become critical, since thousands of simulations need to be run for each regression. This becomes even more relevant when the real HDL is simulated together with the SystemC/SystemC AMS concept model. For these reasons, Infineon Sense and Control Magnetic Sensors is continuously looking for new methods and tools that can speed up the simulation time, without compromising the quality of the results and the easiness of use of the methods.

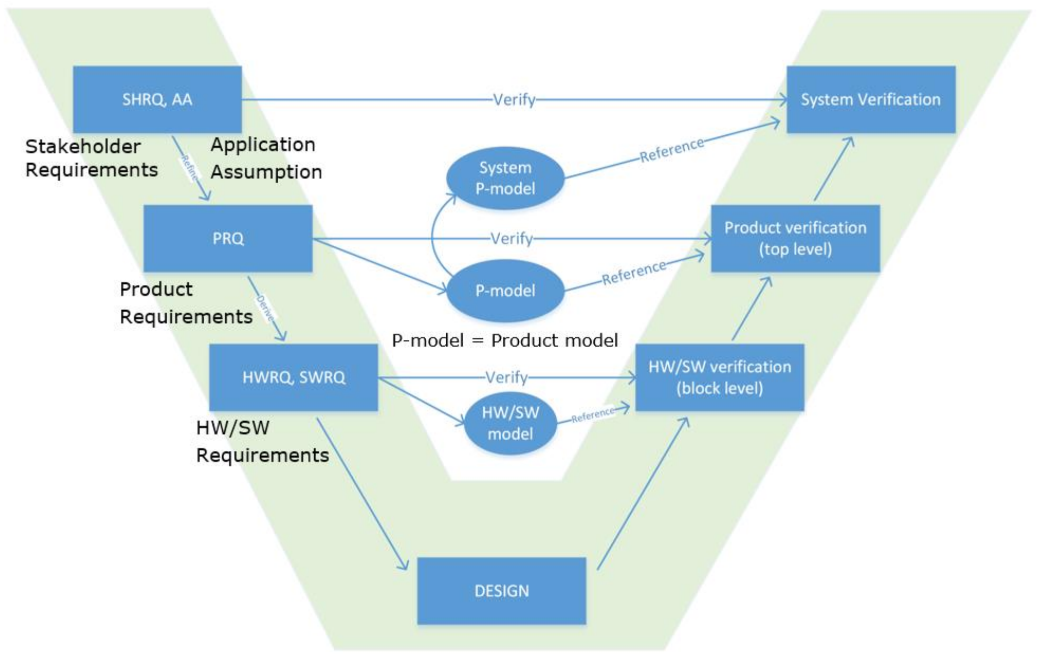

Figure 1 Product model use cases during the development cycle

In this paper, we compare different methods for the simulation of mixed-signal IC models, highlighting the pros and cons of each method. In particular, four approaches are compared, two purely based on a SW environment and two that exploit a FPGA and SOC to accelerate the simulations. Results from previous work are integrated with new findings from a collaboration with COSEDA Technologies, so we provide to the reader an overview of the overall experience we have cumulated in the recent years. The different approaches are evaluated both in a qualitative and quantitative way. Finally, the areas that still have room for improvement are shown and an outlook for future activities is given at the end of the paper.

II. State of the Art

Since many years Infineon Sense and Control develops SystemC/SystemC AMS based concept models that are controlled by a Matlab based regression suite for their sensor products. These models are used as executable specification, for discussions with customer, as reference for the implementation phase and for embedded software development.

The abstract modelling capabilities of SystemC/SystemC AMS allow to create virtual prototypes for mixed signal systems that run fast enough to cover the whole application scenarios / use cases within a virtual environment. However, due to the fact that the configurability and application use cases of Infineon’s sensor products are continuously increasing, thousands of scenarios have to be verified. For the digital core of the device, a bit-level/cycle accuracy is required because sensors are usually used in a very timing sensitive context. Although the accuracy of the analog SystemC AMS modelled components is sufficient to run the application scenarios, all this makes the simulation performance a critical factor.

Modelling the digital core on bit and cycle accurate level has been turned out to be a high effort especially because all design changes have to be back propagated to the SystemC model. Thus, an approach where the derived RTL code can be co-simulated was preferred.

In order to meet the challenges of the required simulation performance for the many simulation runs, together with the combined analog/digital validation and the required high accuracy of the digital model, four different approaches have been investigated.

The first approach uses a commercial VHDL simulator to simulate the SystemC-AMS model in conjunction with the VHDL model. For the second approach the analog part of the model is transformed into digital components, which are synthesized together with the digital core on a FPGA. The third uses a FPGA with an embedded ARM processor where the digital core is synthesized in the FPGA and the analog portion of the model is simulated in the ARM processor. Similar to the first approach, the last approach simulates the SystemC/SystemC AMS model in conjunction with a VHDL simulator but is linked into and controlled by the SystemC/SystemC AMS model and thus allows an unlimited parallelization of simulation runs.

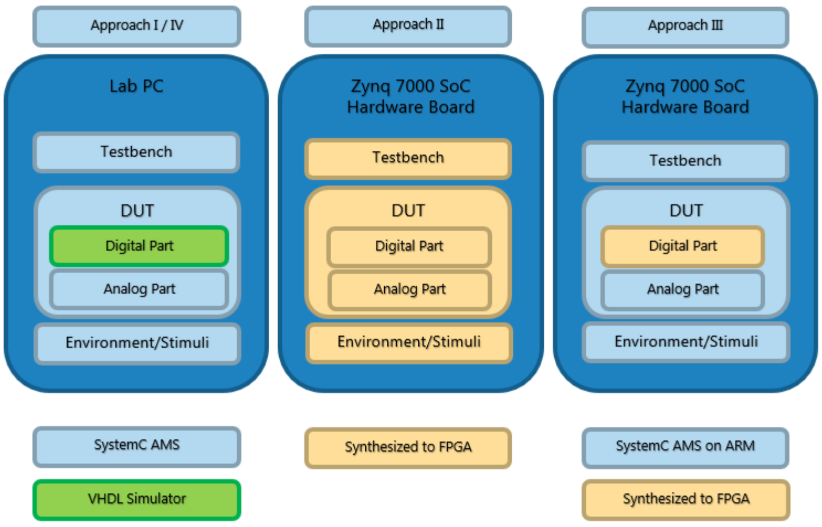

Figure 2 Approach comparison I/IV, II and III

The four different methodologies have been applied to Infineon next generation TMR transmission speed sensor [4]. The sensor comprises a magnetic sensing element (TMR), analog signal path and digital core for signal processing. A high-level block diagram of the sensor is shown in the figure below.

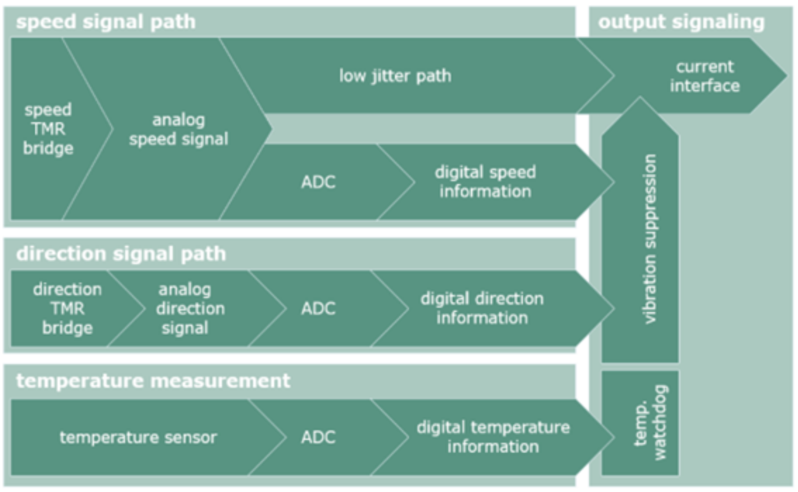

Figure 3 High-level block diagram of the sensor

III. Approaches

A. *Using a commercial VHDL/SystemC simulator*

Commercial simulators, like those provided by Cadence, allow the simulation of designs described in different languages. For example, the Cadence Incisive simulator supports SystemC as well as VHDL or Verilog. Due to the fact that SystemC AMS is based on SystemC language, SystemC AMS designs can also be simulated [5]. Thus, the analog part of the SystemC/SystemC AMS model can be re-used and co-simulated with the digital VHDL based core.

The advantage of this approach is that the RTL code is simulated with a high-performance design simulator, which also provides very good debug capabilities.

However, the required simulation speed using this approach can be achieved only by parallelization. Therefore, the simulation startup time for each simulation run takes a significant amount of time and the license models, provided by commercial simulator providers, made the usage of this approach very cost-intensive or slow.

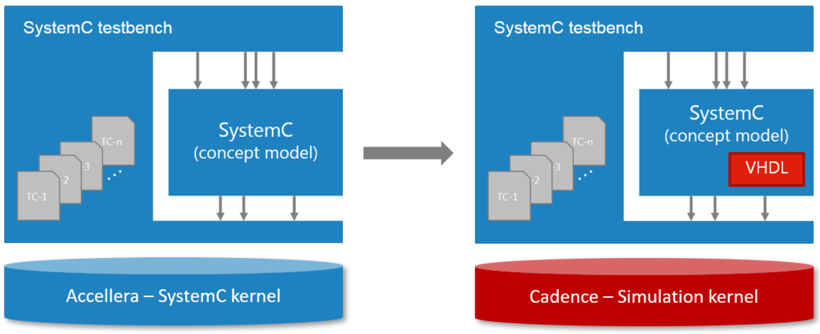

Figure 4 Simulator coupling – model partitioning

B. *Synthesize the digital and analog components to an FPGA*

On an abstract level, analog components can be split into non-linear static and linear dynamic parts. A static nonlinearity can be represented by a table model and linear dynamic functions by filter, e.g., characterizable by step responses. Hence, when written by using a synthesizable code, analog models can be synthesized on a FPGA. Since the digital core is synthesizable, it is possible to synthesize the whole design to an FPGA and accelerate in this way the simulation.

A methodology for synthesizing analog components for FPGA acceleration, by automatically converting analog model written in Python into synthesizable SystemVerilog, has been developed to ease this process [3].

The great advantage of this approach is that a very fast simulation for a single run can be achieved, as everything is happening in real time in the FPGA.

The main disadvantage is the overhead spent in the communication between the FPGA and the PC, which becomes relevant for particular use cases and for a large number of regressions. This problem could be circumvented if the testbench would also become synthesizable, but this costs time and effort. Furthermore, the FPGA hardware limits the possibility of parallel runs.

C. *ARM processor with FPGA*

FPGA manufacturers like Xilinx provide System on a Chips that embed real processor and a FPGA, like the Zynq device from Xilinx (Zedboard) which features a powerful dual core A9 with an AXI interfaces towards the FPGA situated on the same SOC. Since SystemC/SystemC-AMS is based on C++ language, the libraries included in the model code can be compiled for these embedded processors is such a way that the SystemC/SystemC AMS models can run on those CPU’s [2].

The simulation control is done via a Lab PC which talks via secure shell over Ethernet with the ZedBoard. This is possible by running a real time patched Linux on the dual core A9, which comes with full Ethernet stack and block device support such as a SD card. Over this connection the Lab PC transfers the executable and then the ARM processor takes over the regression control by re-running the simulation for each scenario, gathering result logs and traces onto the SD card. Therefore, the regression runs independently from the Lab PC.

To utilize the ARM in addition to the FPGA of approach II, both have to exchange data and synchronize on time. Thereby COSIDE was used to realize this coupling: it provides a timing accurate coupling based on a hardware clock within the FPGA, which controls the timing of the SystemC simulation and scales down the FPGA clock in case the ARM processor cannot keep up with the execution of the compiled SystemC code. In addition, this approach allows to connect real hardware like analog test chips to the FPGA IO.

In this way the analog components and the testbench can both run on the embedded ARM processor and the digital core can be synthesized on the FPGA (Figure 5). Due to the tight connection of the processor and the FPGA in those circuits, tight loops between the software model and the FPGA are possible.

The advantage of this approach is that the analog model and the testbench can be directly re-used and different simulation runs can be launched directly from the embedded processor, which makes the start-up time negligible. The disadvantages are the limited performance of the embedded processor A9 running with 666MHz and an arm-v7 inferior CPU architecture comparing it to full blown x86 Server setups and are in addition limited capability for parallelization because of shared resources like the FPGA and Interfaces.

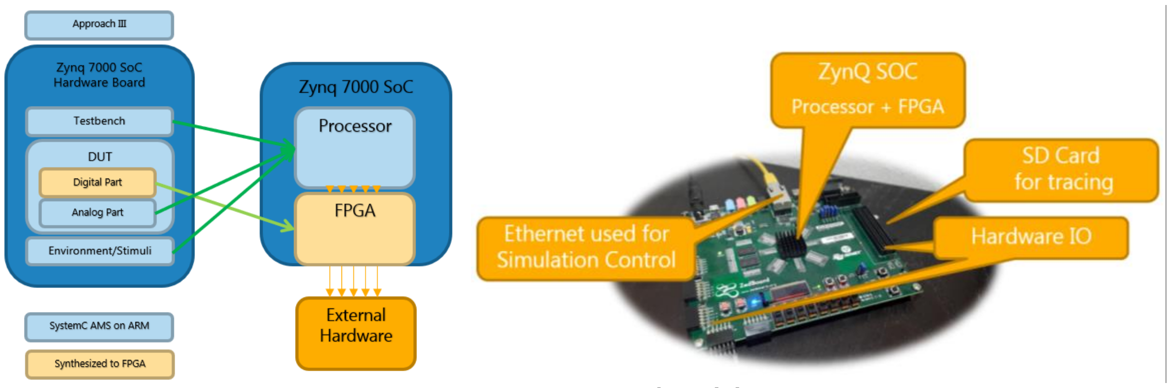

Figure 5 Overview ARM core with coupled FPGA

D. *Linking a VHDL/Verilog simulator into a SystemC/SystemC AMS model*

Xilinx FPGA design suite provides a VHDL/Verilog simulator which can be run as slave, controlled by a SystemC (C/C++). In this case a dynamic library (shared object (so) or dynamic link library (DLL)) can be loaded and controlled via a C-interface. This allows to simulate the VHDL RTL digital core with a pure VHDL simulator while the testbench and the analog components are in SystemC/SystemC AMS.

Since the library is linked into the SystemC simulator, the communication and synchronization overheads are negligible. Additionally, this simulator requires no run-time license and this means that an arbitrary number of simulations can be spawned in parallel without exploding costs.

This approach has several advantages, described hereafter. The first one is the capability to do an arbitrary number of parallel runs (and thus increase the performance). Another one is that the original code of the analog models, the digital RTL core and the testbench can be re-used. Finally, there is nearly no overhead at the simulation start-up, so the simulation performance for a single run is acceptable.

The disadvantages are the limited capabilities of the Xilinx simulator and the fact that a lot of parallel computation power is required to achieve the overall performance goals.

RESULTS

In order to assess the different approaches, two simulations use cases have been selected: forward movement and rotational vibrations.

In the first use case, the target wheel rotates in one direction, while in the latter, the wheel moves back and forth many times, to emulate a mechanical rotational vibration. Various airgaps (the distance between the wheel and the sensor), speed, starting phases and vibration amplitudes have been selected, to have a total of 250 simulations for the first use case and 12500 simulations for the second use case. The simulation parameters are shown in details the following table.

Table I. Simulation parameters

Use case	Rotational frequency	Starting phase	Airgap	Vibration frequency	Vibration number	Vibration amplitude	Simulation number
Forward rotation	250, 500, 1000, 2000, 5000 Hz	0:36:330 degrees	0.5, 1.0, 2.0, 3.0, 4.0 mm	-	-	-	250
Rotational vibrations	-	0:7:349 degrees	0.5, 1.0, 2.0, 3.0, 4.0 mm	1000 Hz	10	± [10:1:59] degrees	12500

First of all, we run the simulations using the first approach, to have a benchmark. The performance is shown in the following table.

Table III. Simulation performance – Approach I (Benchmark)

Use case	Simulation number	Simulation Duration
Forward rotation	250	01h 00min
Rotational vibrations	12500	54h 34min

With the second approach, explained in details in [3], the single simulation time decreases dramatically, as everything (stimuli generator and evaluator, analog and digital part) is implemented within the FPGA. With one run the performance has been improved by factor ~ 12500x compared to the benchmark (approach I).

However, as explained in [3] the LAB PC and the FPGA had to communicate continuously in order to set the stimuli generator and other simulation parameters. If the entire regression setup could be implemented on the FPGA, this approach would show the best results. For comparisons the expected simulation time has been estimated based on the data shown in [3].

Table IIIII. Simulation performance – Approach II

Use case	Simulation number	Simulation Duration	Theoretical Achievable Simulation Duration
Forward rotation	250	~ 6min	~ 1min
Rotational vibrations	12500	~ 13h 30min	~ 18min

In comparison to approach II where the whole model was running on the FPGA, in approach III the testbench and analog parts are assigned to the ARM core. This was easily done by reusing both model parts with the help of a simple cross compile targeting the ARM architecture. The digital part did not provide any challenge to the synthesis into FPGA as the VHDL core was already synthesizable. Investigating the resulting workload of this use case we discovered that the testbench and analog parts contain a significant share of the total simulation load (including digital parts). Therefore, the FPGA had to wait for the synchronization with the ARM data and the effective simulation speed scaled down significantly.

As a result, the overall system was running by a factor of 14.000 slower than real time. This means a scenario featuring 5ms simulation time, took 70s on the ZedBoard hardware compared to approach I, where the pure simulation took approximately 5s and therefore achieved real time factor of 1000.

The difference of factor 14 between the two approaches seems reasonable, when considering the architecture and frequency difference (666MHz vs. 4GHz) of the systems as well as the synchronization overhead of the Zedboard based approach. For the third approach, only the smaller set of 250 testcases (forward rotation) was simulated.

Table IV. Simulation performance – Approach III

Use case	Simulation number	Simulation Duration
Forward rotation	250	~ 4h 50min

Thus, due to the fact that the digital part is relatively small and produces a not dominating simulation time load, this approach is not suitable for this application. This will be different, if the digital part would dominate the simulation performance.

Finally, with the fourth approach we could achieve a significant speed up compared to the benchmark (approach I), also considering that we are still running all the simulations in a virtual environment and no major adaptation to our model and our testbench was needed. In the following table the performance achieved on a 16-core machine is shown.

Table V. Simulation performance – Approach IV

Use case	Simulation number	Simulation Duration
Forward rotation	250	TODO*
Rotational vibrations	12500	59min*

A summary comparison of all four approaches is finally provided in the following table.

Table VI. Simulation performance – Comparison of the four approaches

Use case	Simulation number	Simulation Duration (Approach I – Benchmark)	Simulation Duration (Approach II)	Theoretical Achievable Simulation Duration (Approach II)	Simulation Duration (Approach III)	Simulation Duration (Approach IV)
Forward rotation	250	01h 00min	~ 6min	~ 1min	~ 4h 50min	TODO*
Rotational vibrations	12500	54h 34min	~ 13h 30min	~ 18min	N.A.	59min*

*For camera-ready paper additional results will be shown, including simulations for "forward rotation use case" and different number of cores (e.g. 8, 32…).

IV. SUMMARY

All investigated approaches have advantages and disadvantages. The approach I – the usage of a commercial simulator – has the advantages that the VHDL code is simulated and debugged with the same commercial simulator used for the RTL design. This makes this approach easy applicable and thus acceptable for the designers. Regarding the disadvantages, the simulation performance for a single run is good, but in order to achieve the required good performance when a high number of configurations are tested, a parallelization would be required. The main problem here is that a parallel run of simulations is limited due to the high number of required licenses. Additionally, the startup time for a simulation run is high, which has a significant impact to the overall run time.

The second approach uses a FPGA to accelerate the simulation. Since an FPGA can only emulate digital circuits, the analog components must be converted into digital models. In general this conversion costs resources and it has limitations, but regarding the considered circuit it is acceptable. Unfortunately, in the used setup the communication overhead cancelled the performance advantage. Besides the communication overheads, the effort to convert the analog components into synthesizable digital components and the required FPGA board with limited parallelization are also seen as possible disadvantages. Nevertheless, this approach seems to have potential, but unfortunately it is not observable in this application.

The third approach also uses an FPGA, but compared to the second approach it utilizes also the embedded ARM processor of Zynq-7000 SoC. In this way it is possible to emulate the analog components as well as the testbench directly in the ARM processor. This solution solves the long simulation startup issue and allows a greater flexibility, thanks to the fact that the SystemC / SystemC AMS models could be re-used directly, without any conversions. Specifically in the examined application the analog part is dominant respect to the digital part, and this is also reflected on the simulation performances, where most of the resources are used for the analog part. This consequence is already visible using standard PC/server simulation but becomes more noticeable when an embedded ARM processor is used since it has significantly lower performance compared to the first one. This effect could not be compensated by the FPGA acceleration of the digital part. Hence, the current approach is interesting only in cases where digital part dominates.

For future works a combination of this approach with the approach II could be interesting, considering the double advantage of solving the long startup time issue of approach I and the analog performance issue of this approach, since a conversion is not necessary here. As for the second approach, the capability of parallelization is also limited, due to the fact that a SoC is used.

The fourth approach is a variation of approach I. It solves the issue of the limited parallelization capabilities of the VHDL simulator since it requires no run-time license and can be directly linked into the executable. With this approach an arbitrary parallelization has been possible thus the performance goals have been achieved. Another advantage of this approach is the flexibility since the SystemC / SystemC AMS model and the VHDL-RTL source code could directly be used.

Thus practically, the achievable performance for this use case depends only on the available computing resources. The remaining disadvantage of this approach for the examined application is that a significant CPU time is required.

In conclusion, considering that the application required thousands of simulation runs with a relatively short run time for each single run and a simulation load from the analog part bigger than the digital part, the parallelization has been the key. However, this study showed that, in case of different properties – e.g. the time for a single run increases and/or the digital part becomes more dominant – the other approaches may become interesting again.

References

[1] S. Fontanesi, G. Formato, A. Monterastelli, T. Arndt, "Fast and Furious - Quick Innovation from Idea to Real Prototype", Design and Verification Conference & Exhibition (DVCon Europe), October 2018, Munich.

[2] P. Ehrlich, T. Nguyen, T. Vörtler, "UVM-SystemC based hardware in the loop simulations for accelerated Co-Verification", Design and Verification Conference & Exhibition (DVCon Europe), October 2014, Munich.

[3] G. Rutsch, S. Fontanesi, S. G. Herbst, S. Tan Hee Yeng, A. Possemato, G. Formato, M. Horowitz, W. Ecker, "Boosting mixed-signal design productivity with FPGA-based methods throughout the chip design process", Design and Verification Conference & Exhibition (DVCon Europe), October 2020, Munich.

[4] S. Hainz, E. de la Torre, J. Güttinger, "New Magnetic Sensor Technology for Modern Speed Sensing Application of Rotating Shafts", AmE 2021

[5] Einwich, K., Adhikari, S., Barnasconi, M., Motel, V., 2016. Instrumenting SystemC AMS for Efficient Interactive Debugging and Tracing of Mixed-Signal Systems in Cadence Incisive. Presented at the CDNLive EMEA 2016, Munich.