Evaluating the feasibility of a RISC-V core for real-time applications using a virtual prototype

Juan Santana[1], Gabriel Pachiana[1], Thomas Markwirth[1], Christoph Sohrmann[1],
Bernhard Fischer[2], Martin Matschnig[2]

[1] Fraunhofer IIS/EAS, Münchner Straße 16, 01187 Dresden, Germany
[2] Siemens AG, Siemensstraße 90, 1210 Vienna, Austria

***Abstract* - The replacement of a key component within industrial embedded systems usually requires huge verification and validation efforts. Especially the replacement of the MCU core architecture normally entails significant changes to the HW/SW co-design and co-verification process, possibly including the purchase of costly design and verification IP. Our intended use-case is a system redesign where an established MCU is replaced by a RISC-V core. Since the complete redesign process requires a significant effort, a feasibility evaluation study helps to elaborate the system requirements and to detect possible issues early in the replacement process. Once feasibility has been demonstrated, hardware (re-)design may start. In this paper we propose a HW/SW co-verification methodology to evaluate the feasibility of an MCU core replacement based on a virtual prototype, thereby saving time and cost for the redesign process. This methodology links the VP development process with the requirements management process to re-use the test cases.**

I. INTRODUCTION

Cyber-physical systems (CPS) are characterized by a tight interplay of complex hardware as well as multifaceted software in order to execute a specific physical task within its environment. The most prominent examples of such are industrial control systems where the software application uses sensor readings to control physical actuators. These are prone to frequent updates and functional improvements - single system components might be renewed to fulfill growing performance and functional demands. Unfortunately, it is very difficult to change, test or validate the proper behavior of new components in those highly integrated environments. Complex components are usually difficult to operate in isolation, and access/debug options are limited when components run within a system's context. Instead of replacing and debugging the components' behavior within the real-world system, it is highly recommended to test the component inside a virtual CPS well ahead of any physical implementation step.

The foundation of any computer architecture as presented in [1], is the computer's language called instructions. The computer vocabulary is called the instruction set. RISC-V, an open-source instructions set architecture (ISA) [2], has gained a lot of attention in the last years. It has been intensively developed, implemented and promoted in the commercial as well as in the educational field. Consequently, this led to a wide range of available tools including SW toolchains [3], simulation and prototyping tools, HW implementations and HW devices. Examples for these are the recent CPU board presented by SiFive [4], the (microcontroller unit) MCU from GigaDevice [5], simulators from OVP [6], open-source RTL implementations like PULP [7] and its successors, and others [8]. Many of these developments come into play for rapid system prototyping ahead of a proof of concept, where the application of virtual prototypes is advantageous due to their flexibility.

Industry-grade flows for the design, verification and implementation of a new system on chip (SoC) demand an exhaustive development process. In most cases, this process requires years to be accomplished and causes a significant amount of effort spent for the iterative refinement of the initial project idea. Nowadays, a strong verification ecosystem for the exists around the RISC-V core. Examples are the formal verification framework presented in [9] or the random instructions approach presented in [10].

Changes of the core in an MCU not just represents a HW change. It also involves significant changes in the software domain. Software toolchain adaptions are needed, applications have to be ported to the new HW (CPU) and the functionality of the application for the new platform has to be validated. Even if the proof of concept could be done in an FPGA it is still strongly hardware-dependent. The nature of the virtual prototype (VP) extends the so-called prototype testbench for the software development team. Besides supporting the HW model refinement in the VP, the SW development team is enabled to move forward with software adaptions and tests. With a VP the HW-dependency almost vanishes in the firsts stages of the project to proof the viability and the concept.

Virtual prototypes in model-based development (MBD) have been proven over time to be highly valuable and to provide support for SW developers to test software and elaborate test cases. Firmware (FW) design is becoming increasingly important and needs to be started very early in parallel to the HW design. Embedded software testing on the other side is difficult due to hardware limitations. The advantages of VPs include HW/SW Co-design supporting the SW developing tasks by adding more HW details to increase the modelled functionality. Increasing HW details using an adequate level of abstraction of the system, expands the ability of the system designers to explore HW architecture and its tradeoffs. The VP in this development flow can support full stack SW development. Going beyond pure instruction set simulator (ISS) and VPs, a digital twin integrates inter-domain simulation, by orchestrating sensors, actuators and other related parts of the system which are not purely in the electronic SoC domain. The goal is to prove the whole system within a real emulated application environment.

A virtual prototype represents a support tool for software development in early stages of the software development life cycle. The author in [11] shows the application of a CPS model-based HW/SW co-design approach to support HW microarchitecture exploration, control software design and software verification. It was shown that the same software developed for the VP was runnable unchanged in the physical system. That shows the advantages of early VP development for system validation and software development. The authors in [12] present the extension of a RISC-V VP towards the creation of a fast and accurate performance VP using timing annotations, which allows timing estimations to be closer to the real platform execution. This is achieved by an analysis of three main processing elements (PE) in the core: the pipeline, the branch predictor and the cache access. Derived from that analysis, latency estimations are created for each case and then added to the untimed VP. The authors in [13] back-annotate timings and add delays to the processing elements of the system. Those delays are extracted using estimation algorithms based on number of cycles for operation schedule, delays due to cache misses and branch misprediction penalties.

II. EXAMPLE USE-CASE

The system in Figure 1 contains a *Motor Controller System* composed by an electronic unit with a microcontroller (µC) that implements a field-oriented control (FOC) algorithm, which interacts with the plant - in this case a permanent magnet synchronous motor (PMSM). From the µC unit, Pulse Width Modulation (PWM) signals are used as a stimulus interface and registers mapped to the encoded analog inputs (represented by the motor shaft position and currents in the motor coils) are used to monitor the motor. Tight time interaction between the control software with the peripherals and external system components imposes real-time constraints on this system. The objective of the example use-case is to evaluate the feasibility of replacing an existing legacy MCU by a new RISC-V core. Due to a hierarchy of timing requirements from the PMSM system, from the application down to the embedded system, the control algorithms must run within a cycle time in the magnitude of microseconds.

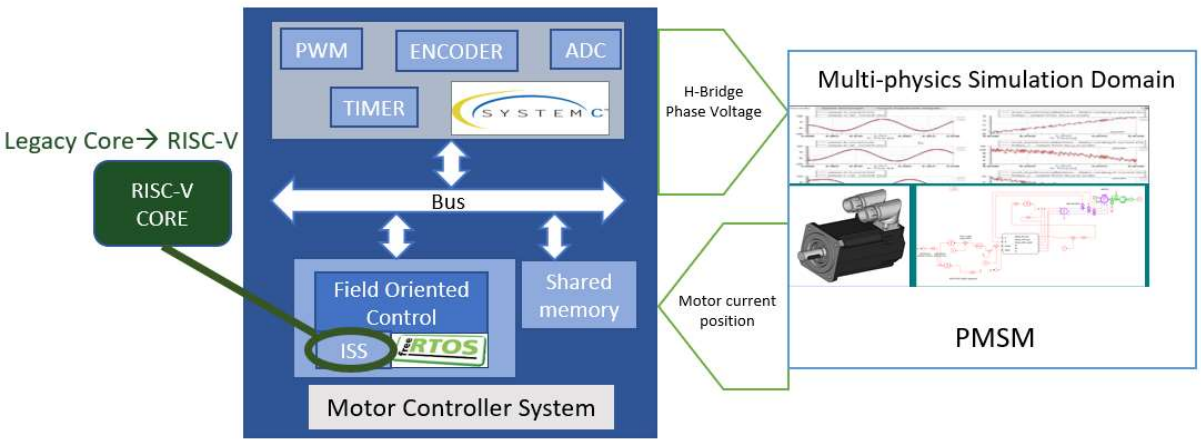

Figure 1 Virtual Prototype system-level hardware architecture

III. PROPOSED METHODOLOGY

In order to study the feasibility of the HW change, we are proposing the following workflow steps and apply them to the use-case described in the previous section:

1. Mapping of the available requirements to the new scenario in order to create an untimed Virtual Prototype that is based on the legacy system specifications,
2. Building an untimed Virtual Prototype for the new system reusing existing model IP blocks,
3. Porting the application software to the new target MCU core architecture,
4. Extracting the timing characterization of the existing system in the RTL domain to extend the VP capabilities by back-annotation of the timing behavior to the system model,
5. Verify timing constrains by comparing the reference timing from the existing RTL simulations vs. the time-annotated VP.

With the aim to simplify the timing annotation of the implemented VP, the analysis of the system starts on a very high level (Figure 2). There we look at the interaction of the Unit Under Test (UUT) and the motor to get metrics related to the behavior of the motor controller by using the PWMs as output and motor stimuli, and the encoder and analog signals as inputs and feedback for the controller. The specification on this abstraction level helps to define the timing metrics at the moment to go one level below into the microcontroller.

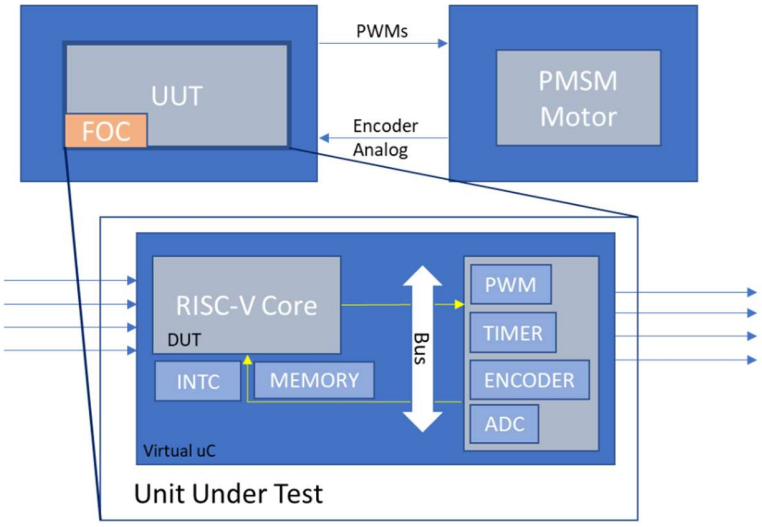

Figure 2. System architecture with Unit Under Test (UUT) which internally contains the RISC-V core (DUT)

The basic functionality of the UUT is:

- Receiving basic user actions to control the motor (e.g., start, stop, accelerate and brake),
- Send PWM stimulus to the motor according to the requested action,
- Analog-to-digital conversion of the signals between motor and UUT and storage in registers,
- Reporting of the motor status by the UUT depending on the received information.

The UUT also has to be aware of time in order to process and output the PWM signals. This timing reference is also an output of the UUT in order to properly verify the response time requirements. Given the high level of detail (cycle-accurate) of the prior system, system level verification for this particular challenge has to rely on verification results and measurements of it. These will be the reference metrics for the new system, obtained by response time measurements of the UUT in the different test case scenarios and regression runs. We are particularly interested in measuring the response time from interrupts/actions relevant for PWM signals, the motor behaviour and the system reporting the motor state.

For this purpose, the UVM-SystemC [14] testbench for the verification is applied to the legacy UUT as shown in Figure 3. It runs each of the test cases implemented according to the System Test Case Specification document (SysTCS), following a common strategy which consist of the following steps/tasks:

- Configure the motor parameters using constrained randomization,
- Take the motor to the desired state,
- Send a sequence of actions separated by a (constrained) random time,
- Monitor the desired response times and send them to the scoreboard for further analysis.

After implementing and running the complete verification suite, we can get a profile of the response times for each test scenario.

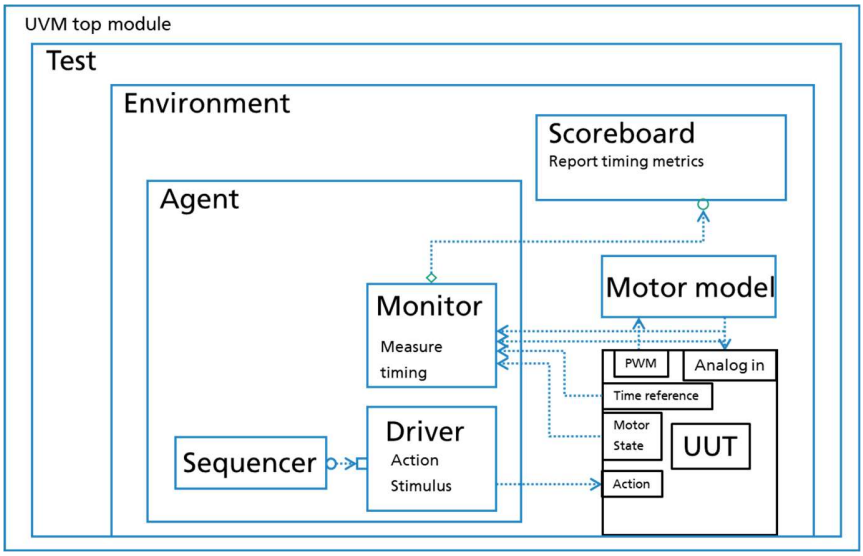

Figure 3. UVM testbench for the new system

The proposed workflow for the new UUT reuses several artifacts from the previous design project, especially the System Requirements Specification (SysRS) document as well as the System Test Case Specification (SysTCS). For the latter, an addition of the timing metrics (timing data) from the legacy UUT has to be done. The same degree of reusability is achieved in the UVM-SystemC verification infrastructure (except for the new checking capabilities), as well as for the continuous integration setup.

The complete flow is shown in Figure 4, which follows a requirements-driven flow described in [15]. In the requirements engineering domain (Req. Eng. Domain) both SysRS and SysTCS documents are described using a requirements engineering tool (RE Tool) that allows to create "<verifies>" relations between SysTCS and SysRS items. Each test case contains the expected response time metrics which are used to verify and report the test status (pass or fail). This information is included in each UVM test case implementation (represented as ① in Figure 4). Also, the test suite can be triggered from a continuous integration (CI) tool which reads the status or each test run (represented as ② in Figure 4). Finally, the pass/fail results are sent back to the RE tool (represented as ③ in Figure 4). When a test of a regression fails, it is marked as such in the RE tool and a defect/issue/bug must be created and linked to the System Test Case. This allows to have a complete and traceable status of the project inside the same RE tool.

Inside the microcontroller, the transactions between peripherals and the DUT are observed to abstract the timing behavior at this level. It is also possible to add timing annotations to the metrics from an available HW implementation of a RISC-V core to complement and increase the accuracy of the estimation.

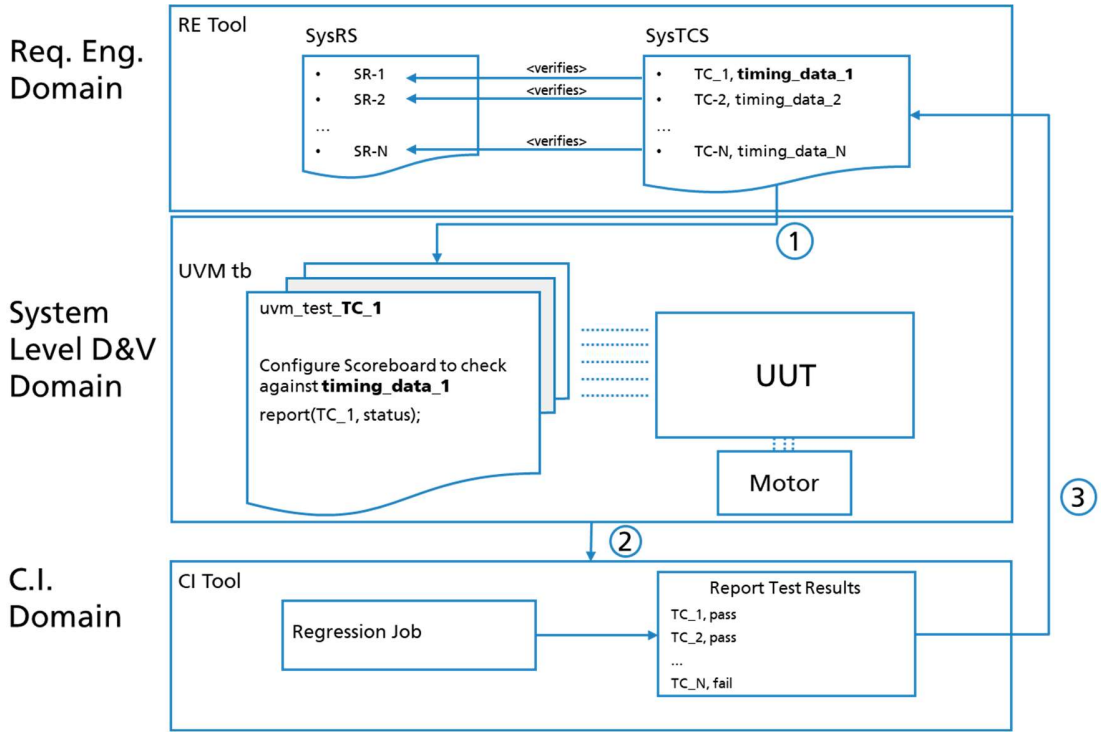

Figure 4. System Level Verification flow for the new UUT.

To support the characterization process for obtaining the timing behavior, we rely on results obtained from the classic functional verification of the legacy system. In this context, the aim of functional verification is to (a) carry out classic functional verification of the legacy system, and (b) get detailed performance metrics to be used in the architectural exploration of a new development of the systems using other sub-components.

IV. Proof of Concept

In order to implement and validate the methodology presented in the previous chapters, we have created a virtual platform for the use-case. The main components of this platform are the UUT, H Motor Driver Model and the PMSM motor model itself, as shown in Figure 5. This platform serves as the basis for the proposed UVM testbench.

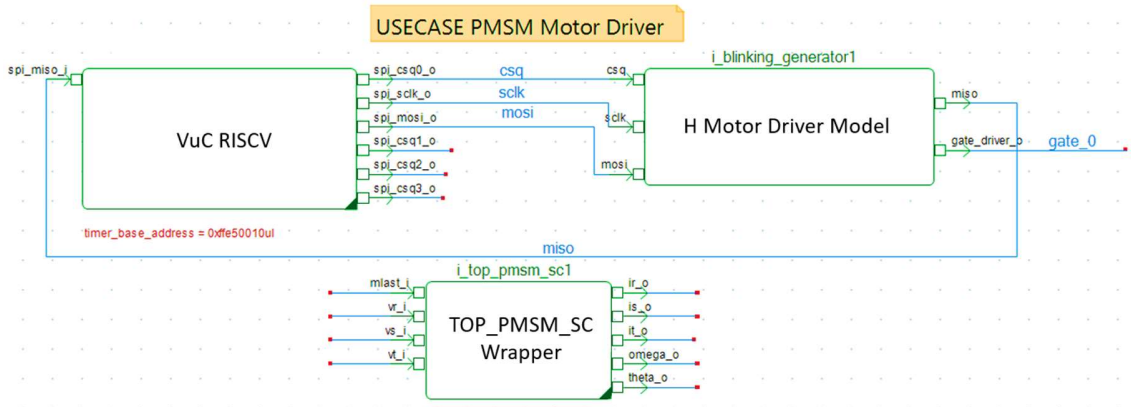

Figure 5 Simplified Virtual Motor Controller Platform

The models of the peripherals are re-used from the existing system. As presented in Figure 5, the block system is interconnected via SystemC/TLM (Transaction Level Modelling). The ISS wrapper is instantiating a RISC-V core. The previous system was implemented based on a QEMU ISS. For the new architecture, QEMU and also OVP (Open Virtual Platform) [6] ISS were integrated to validate the functionality of the VP. As basic requirement for this VP SoC, a timer-based interrupt is included. This supports the time base for the Real Time Operating System that runs the FW application. Creating this cycle-accurate time base affects the simulation runtime. It slows down the simulation execution but allows to measure the motor control behavior more precisely. It also supports the UVM framework for the time metrics to be acquire from the UUT. Using TLM the FW has access to the values of the motor model currents, voltages, position and speed.

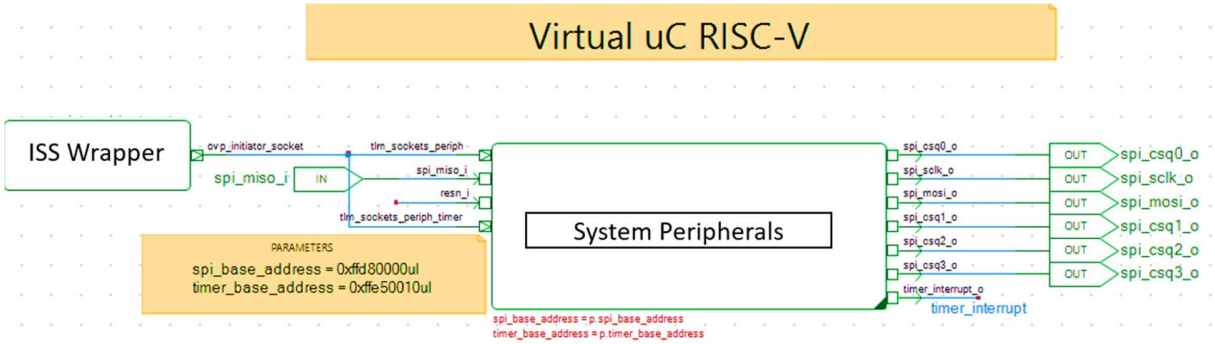

Figure 6 Virtual SoC Implementation

Porting the existing FOC application software to the RISC-V virtual hardware demands the adaption of some of the existing drivers for the VP. In functions where a RISC-V instructions subset is implemented by the ISS, algorithmic operations have to be split and/or rewritten to avoid variables overflow in the ISS. Compiling the FW is reduced to the point where the RISC-V GNU SW toolchain and the compiler options are matched to the target architecture supported by the ISS. Taking advantage of the SystemC nature (based on C++), the simulation runs based on the model executable and calls the firmware application as an executable argument. This executable is refined taking advantage of the framework previously described.

IV. Summary and Outlook

We have presented a methodology based on virtual prototypes that simplifies the hardware exploration process when running real-time application software. The virtual CPS helps to identify issues with the application software long before an intended redesign. Reuse of test cases through the concept exploration phase allows to continuously refine requirements and generate testbenches for a test framework. This could be reutilized in the next stage for RTL design verification as a golden reference and it could extend the coverage metrics with system level stimulus. Within a classic design and verification process, any change of this extent will potentially lead to many failed test cases of a regression and thus, other verification goals (e.g., structural coverage) will not be achieved. The presented methodology shows how to evaluate the viability of the new concept while still complying with the System Requirements Specification (SysRS). As a result, the obtained platform allows hardware architecture exploration and validation at early stages with joint software execution and close analysis between both domains (HW/SW interaction). That reflects performance metrics for the application software like timing, control loop frequency and interrupt handling to match the real-time ability of the system.

As a next step, a detailed evaluation of the introduced methodology will be conducted in the context of the EU-funded research project VALU3S. The presented methodology will be applied to the use-case presented above and the results will be reported and discussed in a subsequent publication.

Acknowledgment

This work has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876852 (VALU3S). The JU receives support from the European Union's Horizon 2020 research and innovation program and Austria, Czech Republic, Germany, Ireland, Italy, Portugal, Spain, Sweden, Turkey.

REFERENCES

[1] D. Harris and S. L. Harris, Digital Design and Computer Architecture, Morgan Kaufmann, 2008.

[2] K. Asanovic and A. Waterman, The RISC-V Instruction Set Manual Volume I: User-Level ISA Version 2.2, 2017.

[3] "https://github.com," [Online]. Available: https://github.com/riscv-collab/riscv-gnu-toolchain. [Accessed 21 10 2021].

[4] "www.sifive.com," SiFive, 29 10 2020. [Online]. Available: https://www.sifive.com/blog/the-heart-of-risc-v-development-is-unmatched. [Accessed 21 10 2021].

[5] "https://www.gigadevice.com," GigaDevice, [Online]. Available: https://www.gigadevice.com/microcontroller/gd32vf103c8t6/. [Accessed 21 10 2021].

[6] "https://www.ovpworld.org," Imperas, [Online]. Available: https://www.ovpworld.org/riscvOVPsimPlus/. [Accessed 21 10 2021].

[7] "https://pulp-platform.org/," [Online]. Available: https://pulp-platform.org/. [Accessed 21 10 2021].

[8] R.-V. International, "riscv.org," RISC-V International, [Online]. Available: https://riscv.org/exchange/cores-socs/. [Accessed 21 10 2021].

[9] C. Duran, H. Morales, C. Rojas , A. Ruospo, E. Sanchez and E. Roa, "Simulation and Formal: The Best of Both Domains," 2020.

[10] S. Davidmann, L. Moore, R. Ho, T. Liu, D. Letcher and A. Sutton, "Rolling the dice with random instructions is," in *DVConUS*, 2020.

[11] M. Ishikawa, D. McCune and G. Saikalis, "CPU Model-based Hardware/Software Co-design, Co-simulation and Analysis Technology for Real-time Embedded Control Systems," *RTAS,* 2007.

[12] V. Herdt, D. Große and R. Drechsler, "Fast and Accurate Performance Evaluation for RISC-V using Virtual prototypes".

[13] Y. Hwang, S. Abdi and D. Gajski, "Cycle-approximate Retargetable Performance Estimation at the Transaction Level".

[14] Accellera Systems Initiative, "UVM-SystemC Library 1.0-beta3," 08 07 2020. [Online]. Available: https://www.accellera.org/images/downloads/drafts-review/uvm-systemc-10-beta3tar.gz. [Accessed 21 10 2021].

[15] G. Pachiana , M. Grunwald, T. Markwirth and C. Sohrmann, "Automated traceability of requirements in the design and verification process of safety-critical mixed-signal systems," *DV Con US ,* 2020.

[16] T. Markwirth, R. Jancke and C. Sohrmann, "Dynamic fault injection into digital twins of safety-critical systems," *Design, Automation & Test in Europe Conference & Exhibition (DATE),* pp. 446-450, 2021.