Expanding RISC-V Horizons: Streamlining Heterogeneous Systems Evaluation with Open Source RISC-V AMS VP Framework

Sallar Ahmadi-Pour[1], Muhammad Hassan[1,2] and Rolf Drechsler[1,2]

[1]Institute of Computer Science, University of Bremen, 28359 Bremen, Germany
[2]Cyber-Physical Systems, DFKI GmbH, 28359 Bremen, Germany

Abstract

In this extended abstract, we present the RISC-V AMS VP framework, an evaluation platform that combines the open source RISC-V Virtual Prototype (VP) with SystemC AMS (Analog/Mixed Signal) for accurate environment modeling of heterogeneous systems. The RISC-V Instruction Set Architecture (ISA), with its modular and extensible design, has garnered significant popularity in academia and industry. The growing RISC-V ecosystem includes various tools, simulators, and processor implementations for supporting the development flow. However, the integration of SystemC AMS with RISC-V VPs has not been explored thus far. The proposed RISC-V AMS VP framework provides a unified view of software, hardware, and AMS environment models, enabling design and verification engineers to accurately interact with the system at different levels and perform powerful analyses. As a case study, we created a temperature control system that integrates a sensor, heater component, and control software running on a RISC-V platform. We also briefly discuss a fault-injection evaluation, considering different modeling layers. We provide the RISC-V AMS VP framework and the temperature control system case study as open source resources, laying the foundation for further research and education in this field.

Introduction

RISC-V, a popular open-source *Instruction Set Architecture* (ISA) [1, 2], is renowned for its modular and extensible design. It facilitates efficient, application-specific solutions, ideal for resource-constrained heterogeneous systems, in particular embedded devices in *Internet of Things* (IoT) and *Cyber-Physical Systems* (CPS). The RISC-V ecosystem offers tools, simulators [6, 7, 8], and processor implementations (open-source and commercial) to aid development. *Virtual Prototypes* (VPs), such as open-source RISC-V VP [3], enable early software development and testing by providing an executable hardware platform model, mainly created in SystemC TLM (Transaction-Level Modeling) [4]. VPs accurately capture hardware-software interactions, but additional environment models are needed for heterogeneous systems. SystemC AMS (Analog/Mixed Signal) [5], a modeling standard, hasn't been integrated with open VP solutions for RISC-V yet.

In this extended abstract, we propose the RISC-V AMS VP, a combination of open-source RISC-V VP and SystemC AMS, to create an evaluation platform for heterogeneous systems. Using a temperature control system as a case study, we illustrate the RISC-V AMS VP modeling principles and its application in building other heterogeneous systems. RISC-V AMS VP offers a unified view of software, hardware, and AMS environment models, allowing design and verification engineers to perform powerful analyses. We briefly discuss this through a fault-injection evaluation and provide the RISC-V AMS VP framework as open source [9].

RISC-V AMS VP Framework for Heterogenous Systems

The RISC-V AMS VP extends the open-source RISC-V VP [3] incorporating SystemC to simulate diverse embedded systems. Utilizing TLM-based software/hardware simulation and SystemC AMS for analog/mixed-signal and physical environment modeling, RISC-V AMS VP offers a unified solution for designing, simulating, and verifying heterogeneous systems tailored to RISC-V. RISC-V AMS VP serves as an executable reference model, enabling iterative refinement of communication protocols and environment models. Fig 1 illustrates the AMS modeling approach with RISC-V AMS VP, highlighting how a temperature sensor, for example, can progress from a basic mock-up to a more sophisticated model with real-world functionalities.

Case Study: Temperature Control System

In this section, we discuss the temperature control system case study created using the RISC-V AMS VP framework. Fig. 1. illustrates the system architecture view of the temperature control system. It offers a detailed view of the system and comprises of three main elements: 1) control software (top of Fig. 1), 2) the TLM-based VP representing the embedded system and executing control software (middle of Fig 1), and 3) the AMS-based sensor, heater, and environment models (bottom of Fig. 1). To develop the

VP, we extended the base RISC-V VP configuration with TLM peripherals that represent interactions with the heater and temperature sensor models.

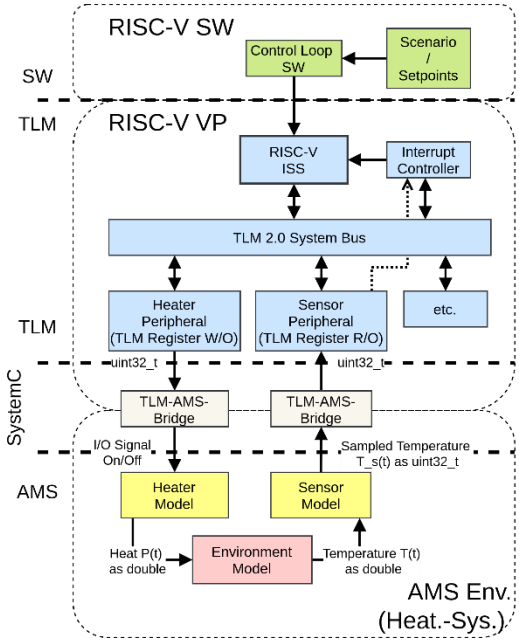

Figure 1 : Architecture view of the temperature control system as implemented using our RISC-V AMS VP framework

At the TLM level, these peripherals offer registers for the heater's state and the sensor's processed temperature, using C/C++ types (uint32_t). The software accesses these registers, which are connected to the AMS-based environment simulation. The C/C++ types are then converted and translated into their physical representations. When the heater is activated, the heat acts as an input for the AMS environment, modeled through differential equations. Through simulations steps the environment's time progresses, and the temperature of the environment changes. The sensed temperature is updated alongside the other modules and used by the software running on the VP.

We opted for a 1° C temperature resolution for the controller for practical reasons, as it is adequate for temperature control given that the modeled sensor (based on a Maxim Integrated DS18B20 Temperature Sensor) has an accuracy of 0.5° C. We use a hysteresis controller, which is popular in multiple embedded application. In our evaluation, we consider a scenario with two setpoints, one at 20° C and another at 10° C. Each setpoint is maintained for a specific number of time steps to represent a realistic temperature control scenario. Fig. 2 displays the system's temperature traced directly from the environment (black line) and the sensor trace as read by the control loop software (blue line). The temperature setpoint is shown as a red dashed-dotted line, with the hysteresis band indicated by purple dashed lines. The hysteresis controller generates the characteristic sawtooth shape on the actual temperature.

Evaluation using Fault-Injection

We extend this use-case with fault-injection evaluation results that consider different modeling layers, specifically: 1) the sensor TLM peripheral, 2) the *Instruction Set Simulator* (ISS), and 3) the control loop software. Covering a wide range of potential error scenarios, this provides a solid foundation for verification engineers to assess the robustness of heterogeneous systems more thoroughly.

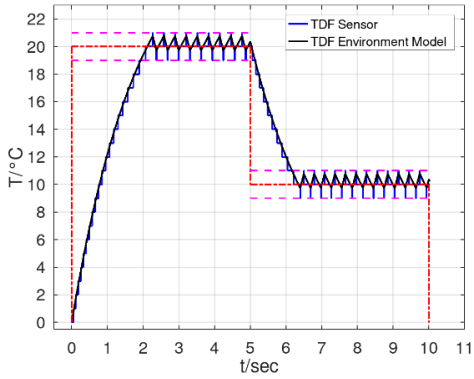

Figure 2: Heat control for two setpoints (20° C and 10° C)

Discussion and Future Work

To encourage further research and education, we offer the RISC-V AMS VP framework and temperature control system case study as open source [9]. For future work, we plan to explore cross-level modeling aspects, performance evaluation, and library of mixed-signal specific blocks. Additionally, we plan to integrate the RISC-V AMS VP with the concept of *Completeness-driven Development* (CDD).

Acknowledgement

This work was supported in part by the German Federal Ministry of Education and Research (BMBF) within the project ECXL under contract no. 01IW22002.

References

[1] Waterman, A., & Asanovic, K. (2019). The risc-v instruction set manual volume i: Unprivileged isa. *Document Version, 20191213*.

[2] Waterman, A., & Asanovic, K. (2019). The RISC-V instruction set manual, volume II: Privileged architecture. *RISC-V Foundation*.

[3] "RISC-V virtual prototype," https://github.com/agra-uni-bremen/riscv-vp

[4] *IEEE Standard SystemC Language Reference Manual*, IEEE Std. 1666, 2011

[5] *IEEE Standard for Standard SystemC(R) Analog/Mixed-Signal Extensions Language Reference Manual*, IEEE Std. 1666.1-2016, 2016

[6] "Spike RISC-V ISA simulator," https://github.com/riscv/riscv-isa-sim

[7] RISCV-QEMU," https://github.com/riscv/riscv-qemu, accessed: 2018-05- 13

[8] "gem5," https://github.com/gem5/gem5.

[9] https://github.com/agra-uni-bremen/riscv-vp/tree/ams-vp