Deterministic Co-Simulation of Open RISC-V-Based Cyber-Physical Systems: A Regenerative Suspension Case Study

Deterministic Co-Simulation of Open RISC-V-Based Cyber-Physical Systems: A Regenerative Suspension Case Study

Ruicong Ni
Polytechnic University of Turin
Torino, Italy
ruicong.ni@polito.it

Sara Vinco
Polytechnic University of Turin
Torino, Italy
sara.vinco@polito.it

Abstract

Active suspension systems improve ride comfort but traditionally rely on energy-intensive actuators. Recent advances in regenerative technologies motivate the co-design of control strategies and energy-aware actuation mechanisms within tightly coupled cyber-physical architectures. This paper presents an open and deterministic HW/SW co-simulation framework for RISC-V-based cyber-physical systems. The framework integrates open-source heterogeneous modeling and execution environments, including Modelica for physical dynamics, IEEE-standard SystemC AMS for power modeling, the open-source GVSoC RISC-V instruction-set simulator for firmware execution, and the Lingua Franca coordination language for logical-time orchestration. By enforcing globally synchronized logical-time semantics across subsystems, the approach enables reproducible evaluation of mechanical, electrical, and RISC-V software components within a unified open hardware research infrastructure. A regenerative suspension case study demonstrates how co-simulation supports energy-aware control design, safety supervision, and early validation of software on open platforms.

CCS Concepts

• **Computer systems organization → Embedded and cyber-physical systems**; • **Computing methodologies → Simulation tools**; • **Hardware → Simulation and emulation**.

Keywords

Open Hardware, RISC-V, Cyber-Physical Co-Design, Embedded Software Validation, Multi-Domain Simulation

ACM Reference Format:
Ruicong Ni and Sara Vinco. 2026. Deterministic Co-Simulation of Open RISC-V-Based Cyber-Physical Systems: A Regenerative Suspension Case Study. In *Proceedings of the 23rd ACM International Conference on Computing Frontiers (CF Companion '26), May 19–21, 2026, Catania, Italy.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3801488.3806234

1 Introduction

Automotive systems are undergoing rapid transformation, driven by the dual pressures of electrification and software-defined architectures [13]. In this scenario, the industry is exploring open instruction-set architectures such as RISC-V and open software ecosystems to foster transparency, customizability, and long-term maintainability of embedded compute platforms [6]. In the automotive domain, such open hardware ecosystems require reproducible evaluation infrastructures capable of integrating firmware execution with realistic physical environments. This increasing complexity requires reproducible and deterministic validation infrastructures capable of integrating firmware, physical models, and power subsystems within a unified environment for the evaluation of modern embedded platforms [5, 11, 16].

CF Companion '26, Catania, Italy

ACM ISBN 979-8-4007-2569-2/26/05
https://doi.org/10.1145/3801488.3806234

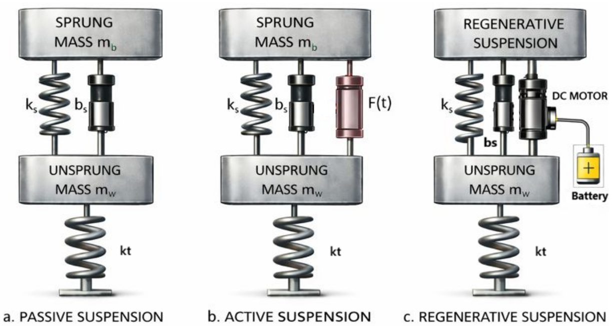

Figure 1: Quarter-car model used to simulate passive (a), active (b) or regenerative (c) suspension system.

A representative automotive subsystem requiring such validation strategies is the regenerative suspension system, a complex subsystem responsible for managing the vertical interaction between the vehicle body and the road to attenuate road-induced disturbances while preserving tire-road contact. This inherently spans multiple engineering domains that are tightly coupled in closed loop: mechanical dynamics of the suspension determine the relative motion between the vehicle body and the wheel; an electromechanical actuator generates suspension force whenever necessary, and may convert mechanical energy into electrical energy, to be stored in the vehicle battery; control algorithms executed as embedded firmware regulate the operating mode of the actuator and the energy flow in order to balance ride comfort, road holding, and energy efficiency [9]. Given that all such aspects continuously influence each other, the behavior of the system can only be accurately evaluated by simultaneously considering multiple domains within a unified simulation framework.

This work proposes an open and deterministic multi-platform co-simulation framework for automotive cyber-physical systems, built by integrating the GVSoC RISC-V Instruction Set Simulator (ISS) [4] with open-source modeling and coordination languages. The framework adopts the most suitable language for each domain while enforcing deterministic coordination for co-simulation and data exchange. The framework is applied to a regenerative suspension case study, thus enabling reproducible comparison of energy-aware control strategies, while relying exclusively on open solutions.

2 Multi-Domain Model of the Regenerative Suspension System

Vehicle suspension systems are commonly modeled using the quarter-car representation shown in Fig. 1. Passive suspensions employ fixed mechanical parameters and therefore provide limited adaptability to varying road conditions (Fig. 1a). To improve performance, passive solutions have been progressively extended toward *active* suspensions, in which control software regulates an actuation force $F(t)$ to enable real-time adjustment of suspension dynamics, typically within the dominant frequency range of 10-15 Hz [18] (Fig. 1b). An active suspension becomes *regenerative* when it additionally harvests energy from road-induced vibrations (Fig. 1c).

Figure 2 represents the main components of a regenerative suspension system, where colors represent the different domains (labeled with the corresponding implementation language or framework) and arrows represent dependencies.

The backbone of simulation is a Lingua Franca (LF) system [14], which coordinates data exchange and enforces the global simulation order. Modelica simulates a full-car dynamic model, that generates the vehicle sensor data [15]. The sensor signals are forwarded to the SystemC AMS subsystem [1] and then passed (via shared memory) to the GVSoC subsystem [4], which hosts the software controller. The computed control signals are returned to SystemC AMS, where the regeneration circuit, vehicle power and battery model are simulated to evaluate actual force generation and energy consumption. The resulting actuator force is finally fed back to the Modelica subsystem, closing the control loop and completing one simulation step of the active suspension system.

2.1 Mechanical and physical subsystem

The mechanical and physical subsystem (orange) consists of two main components. The *full-car mechanical model* is obtained by assembling four quarter car models (Figure 1) on a full vehicle body frame [3]. The model is described as dynamic equations, modeling forces, suspension displacement, and overall pitch and roll dynamics. The suspension model requires input information about road characteristics. This is generated via a *road profile generator* compliant with standard ISO 8608:2016, that specifies a uniform method of reporting measured vertical road profile data [2].

Purely mechanical and physical models of suspension systems can be effectively and easily implemented in Modelica [12], an open-source object oriented language designed for modeling CPSs as differential, algebraic and discrete equations. Modelica supports causal connection of reusable components governed by mathematical equations to facilitate modeling from first principles [15].

2.2 Power subsystem

The power subsystem (blue) includes three main actors. The first component is *power demand caused by vehicle movement*. A simplified model is adopted from [5], that derives EV power consumption as a function of road slope, vehicle speed and acceleration over time. Such power is provided by a *battery*, that can be modeled in a fast yet accurate way by using an electrical equivalent model, where electrical primitives represent battery dynamics [20].

Finally, the last and most relevant model is the *regenerative circuit*, whose full description is available at [19]. Based on the operating mode classification proposed in [21] and on the energy flows, a mode variable γ is used to identify three working conditions of the regenerative suspensions. When $\gamma \leq 0$, the suspension system is in *drive mode* and energy is used to produce a negative damping force. When $\gamma \geq 1$, the system is in *brake mode*: the actuator provides a large dissipative damping force beyond the regenerative capability and mechanical energy is dissipated through the electrical resistance. When $0 < \gamma < 1$, the system is in the *regeneration mode*, and the controller charges the battery.

It is evident that all power components rely on either equations or circuit simulations. This makes the power subsystem suited for SystemC AMS (blue). SystemC AMS [1] is an IEEE standard that extends SystemC support for complex HW/SW systems with analog and mixed-signal (AMS) modeling capabilities, enabling the simulation of continuous-time and multi-domain physical behaviors [8]. SystemC AMS has been widely used to model power subsystems and electric vehicles, thanks to its flexibility, simulation speed and to the native support for electrical linear components [5].

2.3 Software subsystem

Active suspension control requires the development of control firmware, to estimate the necessary negative damping force, control the γ operating mode, and manage the power flows to achieve ride comfort improvement and safety management (yellow). The development of such software cannot be done in isolation, and it rather benefits from the development in an instruction set simulator (ISS), to execute with a specific software stack and allow at the same time exploration and validation of performance, stability, and response to various commands. For this simulation framework, the choice fell on the ISS GVSoC, an open-source C++ event-driven simulation platform for RISC-V cores of the Parallel Ultra-Low Power (PULP) family [4]. GVSoC supports full-platform modeling, including multi-core architectures, multi-level memory hierarchies, I/O peripherals, and hardware accelerators.

Many different control strategies can be used. In this work, we focus on three alternatives representing different trade-offs between computational complexity, timing sensitivity, and closed-loop stability under discrete-time implementation constraints [17]:

- *PID Control* regulates a selected performance variable (e.g., body acceleration) by combining proportional, integral, and derivative actions on the tracking error. Each action works on a different dimension, e.g., oscillation attenuation;
- *Adaptive PID Control* adjusts its gains dynamically in response to variations in operating conditions (e.g., vehicle speed) to balance competing objectives, e.g., ride comfort and road holding;
- *Skyhook Control* reduces body oscillations by generating a damping force proportional to the vertical velocity of the sprung mass.

Each control strategy offers different trade-offs in terms of complexity, ride comfort, safety, and energy efficiency. Evaluating these trade-offs requires a system-level model capable of capturing the interactions between control algorithms, mechanical dynamics, and electrical power flows within a unified simulation environment.

2.4 Simulation backbone

Mixing continuous-time dynamics (Modelica), discrete-time firmware (GVSoC), and mixed-signal power modeling (SystemC AMS) is extremely challenging. While GVSoC is embedded in SystemC AMS

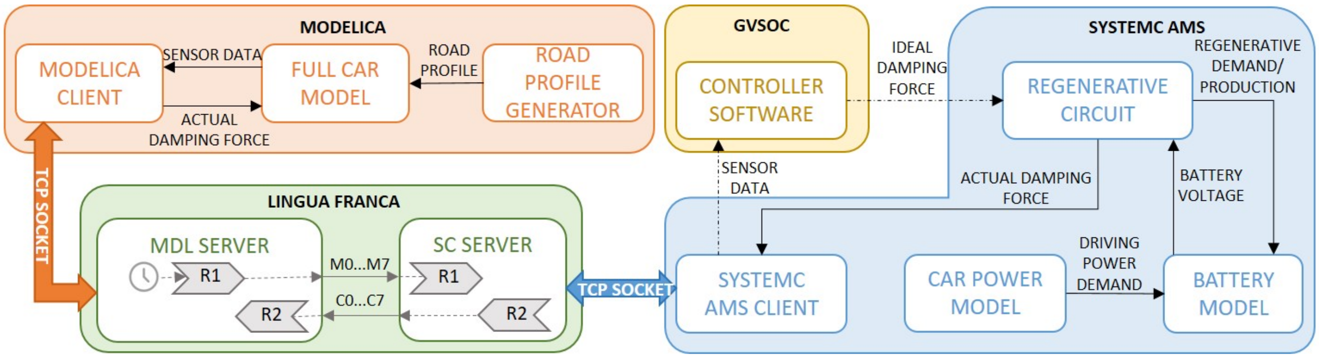

Figure 2: Regenerative suspension simulation framework: colors highlight domains involved and corresponding implementation languages and frameworks, and arrows visualize data and synchronization dependencies.

simulations via the MESSY project [10], integration with Modelica is still an open problem. For this reason, we adopted as simulation backbone LinguaFranca (LF), an open-source polyglot coordination language built to enrich mainstream target programming languages (e.g., C, Python) with deterministic reactive concurrency and the ability to specify timed behavior [14]. In LF, Reactors encapsulate state, timed ports, and reactions, and are executed according to deterministic logical-time semantics. This model ensures that event ordering is fully specified and reproducible across simulation runs.

The LF subsystem (green) acts as an event scheduler and data transmitter. It consists of two reactors: a `MdlServer` reactor to communicate with Modelica, and a `ScServer` to communicate with SystemC AMS. Each reactor contains two reactions, one dedicated to receiving data from its associated simulator (via a TCP socket) and the other to transmitting data to the opposite subsystem. The activation sequence of LF reactions is listed below:

(1) **MdlServer - R1:** the reaction is activated by a timer at fixed timestep; sensor data are received from the Modelica subsystem via TCP and forwarded to the `ScServer` reactor;
(2) **ScServer - R1:** the received values are reconstructed into a delimited string and transmitted to the SystemC AMS subsystem via TCP;
(3) **ScServer - R2:** a control string received from the SystemC AMS subsystem via TCP is parsed into five control signals (i.e., the computed actuator forces for each quarter-car model, plus battery status), that are sent to `MdlServer`;
(4) **MdlServer – Reaction 2:** the five control signals are reassembled into a string and sent to Modelica via TCP.

2.5 Coordination strategy

Each global co-simulation step corresponds to a logical timestamp of fixed size Δt, used to configure both Modelica and the SystemC AMS subsystem, to ensure that fresh data is available with the same frequency. The communication via TCP sockets allows to synchronize in closed loop the two simulations. Data exchange and coordination is enforced by LF, as depicted in Figure 3. Subsystems are not allowed to advance independently. They rather progress only when triggered by the LF runtime at a well-defined logical time, with each reaction being activated within the same logical timestamp (at subsequent microsteps). This enforces a lockstep execution policy in which all participating simulators observe the same

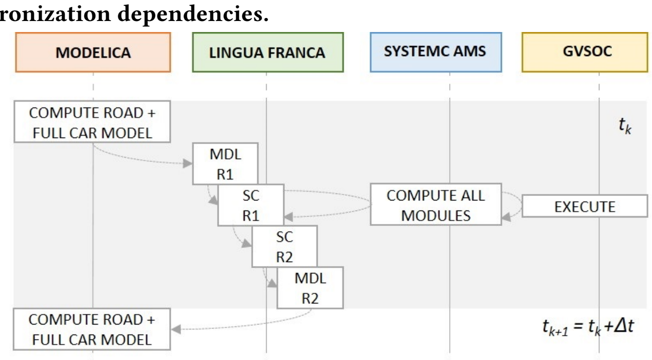

Figure 3: Coordination strategy

temporal progression, and thus avoiding variations of logical-time causality. Because reactions are executed according to a deterministic partial order, the framework eliminates race conditions and unintended concurrency between subsystems. The deterministic coordination ensures that the same firmware execution traces and physical behaviors are obtained across repeated simulations.

3 Simulation

This section presents full-car simulations conducted at 72 km/h on a 100 m ISO 8608 Class C road profile (i.e., moderately rough road surface). Vehicle parameters are configured to reproduce a Tesla Model 3, both in terms of power consumption [5] and battery (CATL 161 Ah $LiFePO_4$ battery pack with 108 cells in series[7]). The timestep is fixed at $1ms$, sufficient to capture the dominant suspension dynamics while ensuring stable discrete-time implementation and avoiding excessive overheads. The code is available at https://github.com/nrc1231/Mesa-RAS-Study.

Figure 4 presents an excerpt of simulation over a time window, including vertical acceleration, battery State of Charge (SoC) evolution, and total power consumption of the four suspension units (positive for consumption, negative for regeneration). It is worth noting that repeated runs with identical inputs produced identical traces, confirming reproducibility, and that all traces are generated in a single simulation run.

To facilitate a multi-criteria comparison of the control strategies, Fig. 5 reports the normalized performance of the considered control strategies with respect to the passive suspension (baseline value equal to 1). The radar plot axes represent ISO-weighted comfort, RMS comfort, tire-load safety, suspension deflection safety, and

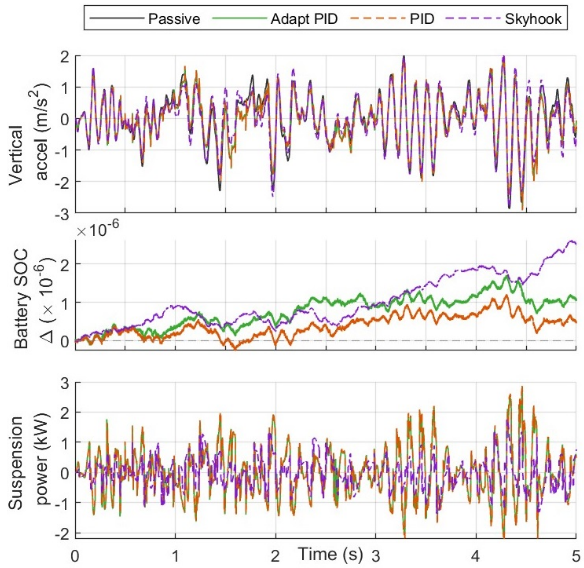

Figure 4: Simulation for the proposed control algorithms.

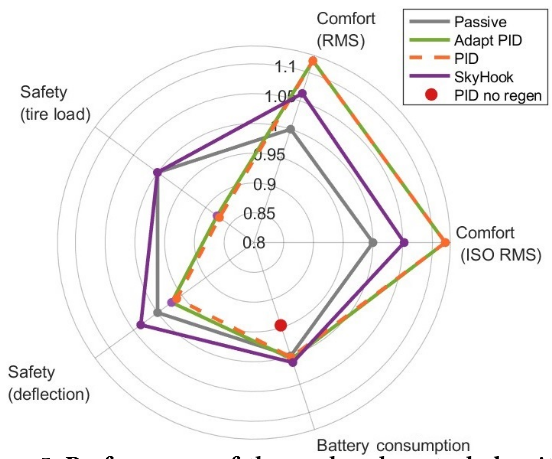

Figure 5: Performance of the explored control algorithms w.r.t. a purely passive suspension system.

battery consumption. This demonstrates how the proposed co-simulation enables unified evaluation of heterogeneous components and control strategies.

In terms of ride comfort, both PID and Adaptive PID achieve the largest improvements, with a reduction of 12.17% in ISO-weighted body acceleration compared with the passive case. The Adaptive PID exhibits slightly more balanced behavior, maintaining similar comfort gains (12.14%) while mitigating the degradation observed in tire dynamic load and suspension deflection. The Skyhook controller provides a smaller improvement in ride comfort (about 6%), but demonstrates a more favorable trade-off in safety-related metrics, preserving tire load and suspension deflection closer to the passive reference. From an energy perspective, PID and Adaptive PID provide modest battery savings (approximately 0.25% and 0.52%, respectively), while Skyhook achieves the highest energy benefit (about 1.15%) among the regenerative strategies. Conversely, the red marker in the radar chart represents a PID-controlled active suspension without regeneration, which incurs an energy penalty of approximately 5.6% relative to the passive configuration.

4 Conclusion

The proposed framework demonstrates how open hardware simulation platforms can be integrated within deterministic multi-domain co-simulation environments, with a regenerative suspension system serving as a representative case study. This capability is particularly relevant for the validation of embedded firmware on emerging open architectures such as RISC-V, as it enables reliable cross-platform co-simulation and the integration of diverse system functionalities.

Acknowledgments

This work has received funding from the European Chips Joint Undertaking under Framework Partnership Agreement No. 101139789 (HAL4SDV).

References

[1] 2016. IEEE Standard for Standard SystemC Analog/Mixed-Signal Extensions Language Reference Manual. *IEEE Std 1666.1-2016* (2016), 1–236. doi:10.1109/IEEESTD.2016.7448795
[2] 2016. ISO 8608:2016 Mechanical vibration — Road surface profiles — Reporting of measured data.
[3] Ayman Aljarbouh, Muhammad Fayaz, Muhammad Shuaib Qureshi, and Younes Boujoudar. 2021. Hybrid Sliding Mode Control of Full-Car Semi-Active Suspension Systems. *Symmetry* 13, 12 (2021).
[4] Nazareno Bruschi, Germain Haugou, Giuseppe Tagliavini, Francesco Conti, Luca Benini, and Davide Rossi. 2021. GVSoC: A Highly Configurable, Fast and Accurate Full-Platform Simulator for RISC-V based IoT Processors. In *Proc. of IEEE ICCD*. 409–416.
[5] Yukai Chen, Donkyu Baek, Jaemin Kim, Santa Di Cataldo, Naehyuck Chang, Enrico Macii, Sara Vinco, and Massimo Poncino. 2019. A SystemC-AMS Framework for the Design and Simulation of Energy Management in Electric Vehicles. *IEEE Access* 7 (2019), 25779–25791.
[6] L. Cuomo, C. Scordino, A. Ottaviano, N. Wistoff, R. Balas, L. Benini, E. Guidieri, and I. Maria Savino. 2023. Towards a RISC-V Open Platform for Next-generation Automotive ECUs. In *Proc. of MECO*.
[7] Evlithium. [n. d.]. CATL 161 Ah $LiFePO_4$ Prismatic Cell Specification. https://www.evlithium.com/catl-battery-cell/catl-lifepo4-battery-cell-161ah.html. Accessed: 2026-02.
[8] Enrico Fraccaroli and Sara Vinco. 2023. Modeling Cyber-Physical Production Systems With SystemC-AMS. *IEEE Trans. Comput.* 72, 7 (2023).
[9] Changzhong Fu, Jiayu Lu, Wenqing Ge, Cao Tan, and Bo Li. 2022. A Review of Electromagnetic Energy Regenerative Suspension System & Key Technologies. *Computer Modeling in Engineering and Sciences* 135, 3 (2022), 1779–1824.
[10] Mohamed Amine Hamdi, Giovanni Pollo, Matteo Risso, Germain Haugou, Alessio Burrello, Enrico Macii, Massimo Poncino, Sara Vinco, and Daniele Jahier Pagliari. 2024. Integrating SystemC-AMS Power Modeling with a RISC-V ISS for Virtual Prototyping of Battery-operated Embedded Devices. In *Proc. of CF*.
[11] P. Hehenberger, B. Vogel-Heuser, D. Bradley, B. Eynard, T. Tomiyama, and S. Achiche. 2016. Design, modelling, simulation and integration of cyber physical systems: Methods and applications. *Computers in Industry* 82 (2016), 273–289.
[12] Y. Hou, L. Li, P. He, Y. Zhang, and L. Chen. 2011. Shock Absorber Modeling and Simulation Based on Modelica. *Modelica Conference* (2011), 843–846.
[13] S.J. Kwag and B. Kang. 2025. Key to the future of the automotive industry: SDV technology and reliability. *JMST Advances* 7 (2025), 99–104. Issue 1.
[14] Lingua Franca Project. 2025. Lingua Franca Documentation. https://www.lf-lang.org/docs/. Accessed: 2026-02-03.
[15] Modelica Association. 2026. Modelica — A Language for Modeling of Cyber-Physical Systems. https://modelica.org/. Accessed: 2026-02-04.
[16] R.G. Oliveira, I. Raghupatruni, A. Hamann, and A. Henkel. 2021. Virtual Verification of Cause-Effect Chains in Automotive Cyber-Physical Systems. In *Proc. of Internationales Stuttgarter Symposium*.
[17] Sergio M. Savaresi, Charles Poussot-Vassal, Cristiano Spelta, Oliver Sename, and Luc Dugard. 2011. *Semi-Active Suspension Control Design for Vehicles.* Elsevier.
[18] Marek Sibielak, Jarosław Konieczny, and Adam Smoter. 2024. Model and Control Synthesis for an Active Vehicle Suspension Strut Involving a Single-Rod Electrohydraulic Actuator. In *Proc. of ICCC*.
[19] Gianluca Sodano. 2021. *Self-powered Active Suspension.* Technical Report.
[20] Sara Vinco, Alessandro Sassone, Franco Fummi, Enrico Macii, and Massimo Poncino. 2014. An open-source framework for formal specification and simulation of electrical energy systems. In *Proc. of ISPLED*. 287–290.
[21] Shuai Yan and Weichao Sun. 2017. Self-powered suspension criterion and energy regeneration implementation scheme of motor-driven active suspension. *Mechanical Systems and Signal Processing* 94 (2017), 297–311.