Integrating SystemC-AMS power modeling with a RISC-V ISS for virtual prototyping of battery-operated embedded devices

Integrating SystemC-AMS Power Modeling with a RISC-V ISS for Virtual Prototyping of Battery-operated Embedded Devices

Mohamed Amine Hamdi[1], Giovanni Pollo[1], Matteo Risso[1], Germain Haugou[2], Alessio Burrello[1], Enrico Macii[1], Massimo Poncino[1], Sara Vinco[1] and Daniele Jahier Pagliari[1]
[1]Politecnico di Torino, Turin, Italy - first_name.first_surname@polito.it;
[2]GreenWaves Technologies SAS, Grenoble, France - germain.haugou@greenwaves-technologies.com

ABSTRACT

RISC-V cores have gained a lot of popularity over the last few years. However, being quite a recent and novel technology, there is still a gap in the availability of comprehensive simulation frameworks for RISC-V that cover both the functional and extra-functional aspects. This gap hinders progress in the field, as fast yet accurate system-level simulation is crucial for Design Space Exploration (DSE).

This work presents an open-source framework designed to tackle this challenge, integrating functional RISC-V simulation (achieved with GVSoC) with SystemC-AMS (used to model extra-functional aspects, in detail power storage and distribution). The combination of GVSoC and SystemC-AMS in a single simulation framework allows to perform a DSE that is dependent on the mutual impact between functional and extra-functional aspects. In our experiments, we validate the framework's effectiveness by creating a virtual prototype of a compact, battery-powered embedded system.

CCS CONCEPTS

• **Hardware → Power estimation and optimization**; **Simulation and emulation**.

KEYWORDS

SystemC, RISC-V, Virtual Prototyping, Power, Design Space Exploration

ACM Reference Format:
Mohamed Amine Hamdi[1], Giovanni Pollo[1], Matteo Risso[1], Germain Haugou[2], Alessio Burrello[1], Enrico Macii[1], Massimo Poncino[1], Sara Vinco[1] and Daniele Jahier Pagliari[1]. 2024. Integrating SystemC-AMS Power Modeling with a RISC-V ISS for Virtual Prototyping of Battery-operated Embedded Devices. In *Proceedings of Proceedings of the 21st ACM International Conference on Computing Frontiers Workshops and Special Sessions (CF '24 Companion).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3637543.3652873

1 INTRODUCTION

RISC-V has become a pivotal player in the semiconductor industry by offering a customizable and free alternative to the conventional, proprietary ISAs [16]. Its widespread adoption in various sectors, from embedded systems to high-performance computing, highlights its potential to transform the design and implementation of computing systems.

The interest of the research community led to the development of many frameworks and implementations supporting the functional simulation and customization of RISC-V cores [14]. The goal of such frameworks is to allow the development of virtual prototypes, enabling designers and engineers to model, analyze, and optimize systems based on RISC-V before constructing physical prototypes [8, 9]. This drastically cuts down on development time and costs, allows for the early identification of design issues, and supports the exploration of various architectural configurations to achieve specific performance, power, and area objectives.

However, all such solutions focus on functionality and timing-related aspects, thus allowing only a partial DSE. This is a severe limitation: the high degree of heterogeneity, the technology challenges, and the tight coupling with the environment of modern systems require indeed to *also consider extra-functional metrics* such as power consumption and thermal behavior to ensure correct operations [20]. It is thus necessary to *monitor the evolution over time of functionality not in isolation, but rather together with extra-functional properties.* As highlighted in [4], RISC-V simulators provide at most an analysis of the total energy consumption, with no integrated simulation of functionality and power dynamics.

Outside of the RISC-V community, SystemC and its Analog Mixed Signal (AMS) extension [1] emerged as a widespread solution for the creation of virtual platforms [12]. SystemC-AMS facilitates high-level modeling and simulation of mixed-signal systems, supporting a wide range of components. Additionally, it can effectively cover extra-functional behaviors, ranging from mechanical systems to power dynamics [7, 20]. Thus, SystemC-AMS seems a viable solution for the construction of power-aware virtual platforms.

This paper proposes an open-source framework that integrates SystemC-AMS with GVSoC [5], a RISC-V Instruction Set Simulator (ISS), to build an open-source virtual platform that is aware of both functionality and extra-functional aspects, with a focus on power. The goal is to allow the simulation of the power flows in the system, not only in terms of power demand of the RISC-V core but also considering, for instance, the battery subsystem that feeds the functional parts. With experiments on a simulation setup that models a simple battery-operated embedded device for audio processing, we show that our framework enables detailed and extensive DSE, with an acceptable simulation time overhead with respect to vanilla GVSoC[1].

CF '24 Companion, May 7–9, 2024, Ischia, Italy

[1]The source code of our simulator is available at https://github.com/eml-eda/messy

2 VIRTUAL PROTOTYPING ARCHITECTURE

2.1 Power Simulation with SystemC-AMS

SystemC-AMS is an extension of SystemC that allows modeling the analog-mixed signal part of an embedded system. It covers a wide range of domains thanks to three different modeling styles. Timed Data Flow (TDF) relies on static scheduling and is used to represent discrete-time systems. Linear Signal Flow (LSF) adopts linear algebraic equations to model control system modeling. Electrical Linear Networks (ELN) offers a collection of standard linear electrical components, like resistors and capacitors. When building the simulatable implementation of a component, one can determine the suitable SystemC-AMS flavor by considering the desired (i) level of abstraction, i.e., discrete time or continuous time, (ii) level of detail of the description (e.g., based on equations or transfer functions), and (iii) adherence to conservation laws.

SystemC-AMS simulation is event-driven, where a centralized scheduler controls the execution of processes based on events, i.e., synchronizations, time notifications, or signal value changes.

The suitability of SystemC-AMS for the simulation of power systems has been investigated in the literature, ranging from small battery-powered systems [19] to electric vehicles [2]. Power components may be described at a functional level (e.g., as an efficiency equation in case of DC-DC converters) or as a circuit model (e.g., in case of a battery). This implies a corresponding choice of SystemC-AMS constructs, i.e., TDF in the former case and ELN in the latter.

In SystemC-AMS simulations, the power flow is simulated and managed by a *power bus*, that extends the physical Charge Transfer Interconnect bus (CTI) with the simulation of the power flow, i.e., the aggregation of load demands to determine the amount of current required from the energy sources and storage devices [7]. All components are connected (possibly via DC-DC converters) to the power bus, which has a role very similar to a functional bus in managing the overall power flow (rather than information flow).

2.2 Proposed architecture and requirements

The coupling of functional and extra-functional simulation with SystemC-AMS has been first proposed in [20]. The proposed architecture exploits a *bus-centric paradigm* so that the simulation features one bus for each modeled aspect (in this case, functionality and power). Each system component may be connected to each bus by implementing one model per bus: e.g., a core will have a functional implementation that describes instruction processing connected to the functional bus and handles the timing behavior of the system, and a power model that estimates the corresponding power demand, exported to the power bus. Figure 1 shows the overall architecture, where most system components are connected both to both buses.

The mutual interdependency between the functional and the power models is achieved through an *information exchange between the models of the same component*. In case of the core, the functional model may share its current state (e.g., the instruction being processed, active-idle-sleep state, etc) with the power model, which can estimate the corresponding voltage level and current demand.

This architecture benefits from the modularity of SystemC-AMS, and allows to define the *requirement that a RISC-V functional ISS must possess* to be compatible with our framework:

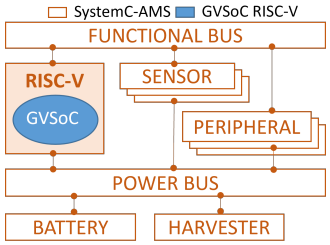

Figure 1: Bus-centric architecture adopted to simulate functional and power aspects of a RISC-V system.

- To ease the integration with SystemC-AMS, the simulator implementation must be C/C++-based;
- The simulator must model instruction-level timing. This is fundamental to allow a precise estimation of power over time, accounting for the impact of the processor's execution;
- The simulator must be embeddable as a software component within a SystemC module and must expose APIs to export state information or direct power estimates.

2.3 The GVSoC RISC-V ISS

GVSoC [5] is an open-source C++ event-driven simulation platform for RISC-V cores of the Parallel Ultra-Low Power (PULP) family [15], supporting the modeling of low-power CPUs and the definition of complex full-platforms, including multicore, multi-memory levels (i.e., on- and off-chip), complex I/O peripherals, and accelerators [5]. GVSoC supports virtual prototyping and DSE through early-stage performance evaluation based, e.g., on hardware counters and timing models, and it is *almost cycle-accurate* (with a maximum error margin for cycle accuracy of 10%).

GVSoC simulation is event-driven. A circular buffer contains every event generated in the system, enqueued based on the corresponding latency. At any time, the simulator identifies the next event, processes the corresponding actions, and updates the queue. When not used as a stand-alone component, GVSoC exposes the `step_until()` API primitive, which runs the simulation until a specified time and returns the timestamp of the next event in the queue.

The latest version of GVSoC includes power models that account for both dynamic and leakage power of the different frequency domains and power islands within the modeled PULP chip. Power information can be retrieved by invoking the `get_instant_power()` API primitive, which outputs the overall consumption of the system, aggregating all the contributions due to the set of events that occurred in the latest simulated timestamp.

In this work, we select GVSoC to demonstrate our SystemC integration, as it respects all requirements described at the end of Sec. 2.2. However, we remark that, in principle, our architecture is orthogonal to the specific RISC-V ISS.

3 GVSOC-SYSTEMC-AMS INTEGRATION

Fig. 2 shows the detailed implementation of the general simulation architecture discussed above. At the heart of the system is the SystemC core unit, which orchestrates the simulation and facilitates the interaction between SystemC and GVSoC. The simulation is

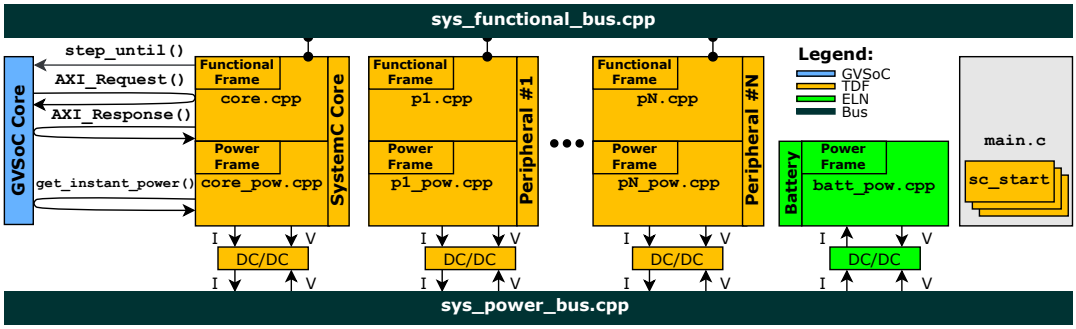

Figure 2: Implementation of the integration between SystemC-AMS and GVSoC.

organized as follows. At the beginning, the `sc_main` function (implemented by the `main.c` file) instantiates the GVSoC ISS and all SystemC-AMS components. Both GVSoC and SystemC initialize their event queues by starting SW execution (in the former case) and executing all components exactly once (in the latter case).

The subsequent simulation proceeds by alternating the execution of GVSoC and SystemC-AMS. Precisely:

1. The functional model of the core (`core.cpp`) invokes the `step_until()` API function of GVSoC, which executes the SoC functionality and returns the timestamp t of the following event in the GVSoC queue;
2. `core.cpp` then executes a SystemC-AMS `wait()` until t, to allow the execution of other SystemC-AMS components and the temporal alignment of the two simulators.
3. These two steps are repeated until the simulation is completed.

Whenever the SW executing in GVSoC needs to communicate with a bus-connected functional component instantiated in SystemC-AMS (e.g., a peripheral), it propagates the necessary information (i.e., address, control signals, data) to the functional model of the core through an `AXI_Request()`, which is intercepted by a callback in `core.cpp`. The functional model of the core then forwards this information to the SystemC functional bus, where it can be intercepted by the appropriate receiver (e.g., the peripheral's functional model). Lastly, `core.cpp` also collects from the functional bus any response received from peripherals. Such information is propagated to GVSoC with an `AXI_Response()` call, thus closing the cycle.

Functional execution triggers the execution of the corresponding power models. In case of the core (i.e., `core_pow.cpp`), this corresponds to an invocation of the `get_instant_power()` primitive of GVSoC. The power model of other peripherals is implemented directly in SystemC-AMS (i.e., `*_pow.cpp`). For instance, a peripheral might be modeled as a simple Power State Machine (PSM) sensitive to its operating mode (e.g., active-idle-sleep).

This dual-instance approach allows for a synchronized and detailed simulation of both the functional interactions and power consumption dynamics within the system.

4 EXPERIMENTAL RESULTS

We consider an audio processing task executed on a battery-operated device to evaluate the simulation methodology. We selected this use case since audio DSP is common in devices such as smart wearables, for which power simulation and optimization are critical. Furthermore, the PULP devices simulated by GVSoC have ISA extensions and architectural features specifically tailored for DSP applications [15]. More specifically, we mimic an Adaptive Filtering workload, commonly found in applications such as noise cancellation, echo suppression, and dynamic equalization [3]. To this end, we implement a simple software application in GVSoC that repeatedly stores input samples from a microphone into a buffer, processes them with a FIR filter with 40 taps, and then waits for the next buffer to be available before repeating the whole process.

As the main computing device, we use GVSoC's model for GAP9 [18], a PULP system comprising a cluster of 9 RISC-V cores to speed-up compute-intensive DSP workloads. We configure the SystemC-AMS power bus to deliver a regulated 3.3V supply to all components. The rest of the simulation setup includes: i) a Mic Click microphone sensor from Mikroe [10], whose power consumption, ranging in 120-160μA, is modeled as a state-sensitive PSM; ii) a Panasonic CG-425A 32mAh@3.8V battery [11], simulated using the circuit-equivalent model of [13], whose parameters are populated from datasheet information; iii) two DC-DC converters. While the microphone directly expects a 3.3V supply, the battery is connected to the power bus through a RT8097A converter [6], and GAP9 is supplied at 1.8V by a ST1PS03 [17]. For both DC/DCs, we use LUT-based efficiency models that depend on their current output, which is extracted from datasheet curves.

Fig.3-A shows the complete depletion of the battery throughout a simulation. The battery can survive almost **25 hours** for this workload, showcasing the low-power nature of GAP9. The state of charge decreases almost linearly due to the low currents drawn by the sensor and computing subsystem. However, "zooming in" to look more in detail at the cumulated state-of-charge difference (ΔSoC) in 1s intervals (Fig. 3-B), the non-linearity of the discharge behavior can be noticed. This is due to the variation of the battery's internal resistance embedded in our circuit-equivalent model.

Fig. 3-C displays the zoomed-in load power profile for a portion of an iteration in the simulated application. The plot clearly shows the phases that compose each iteration: i) the initial low and constant power consumption when GAP9 waits for new sensor data; ii) a region of higher consumption linked to the activation of GAP9's cluster, which also includes a set of power spikes, in correspondence to the start of inner processing loops in the cluster cores (the whole execution of the FIR is *tiled* in smaller sections); iii) the final wait before the start of the following iteration.

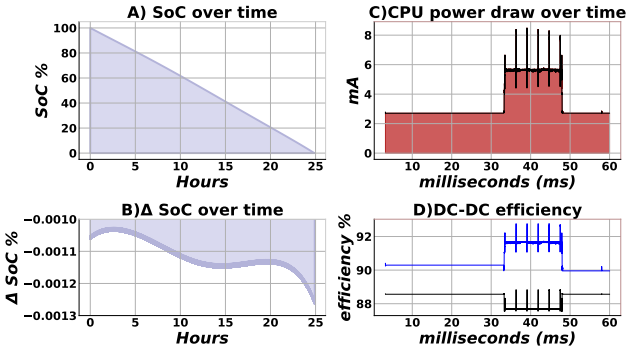

Figure 3: Simulation of an audio filtering application.

Fig. 3-D shows the efficiency of the DC/DC converters for the battery (blue curve) and core (black curve) over the same time period. This graph shows that the DC/DC selected for GAP has a lower efficiency overall, which, however, as desired, is higher during long idle phases and decreases during the short higher-power computation phases.

4.1 Design Space Exploration

In this section, we show an example of how our framework can be used for a DSE that considers both functional and power aspects. Namely, starting from the experiment of Fig. 3, which we denote as configuration **A**, we first reduce the processing workload by decreasing the filter taps to 20 (**B**). This mimics a quality-of-service reduction to increase the battery lifetime. Then, based on the results of Fig. 3, we try replacing the GAP9 DC/DC converter with the potentially more efficient RT8097A (**C**). Lastly, we combine both modifications (**D**). To stress the power differences, we reduce the interval between two filter executions from 1s, as in Fig. 3, to 50ms.

The results of this exploration are shown in Table 1, which reports the average power drawn from the battery (Battery P.), the average GAP9 converter efficiency (DC/DC Eff.), and the ΔSoC over 1 hour of simulation. Fig 4 shows the estimated battery lifetime of the four configurations, normalized to the one of configuration **A**.

As shown, reducing the GAP9 workload (**B**) leads to an estimated 12% increase in battery life, mainly due to lower power demand and secondarily to a slightly higher DC/DC efficiency (which is higher when GAP9 is not computing, as shown in Fig. 3). Conversely, replacing the DC/DC only prolongs the battery life by 4%, but without impacting quality-of-service (**C**). Lastly, combining the two optimizations (**D**) leads to a 16% lifespan improvement.

Overall, the results of Fig. 3-4 and Table 1 showcase the type of fine-grained power analysis enabled by the proposed simulation framework. These insights are obtained at a limited cost in terms of simulation time overhead with respect to vanilla GVSoC, i.e., approximately 7.3% when GVSoC's power model is active (for a purely functional simulation, disabling GVSoC's power model, the overhead would grow to approximately 50%).

5 CONCLUSIONS

We have introduced a fully open-source framework to simulate functional and non-functional properties of RISC-V systems, with a specific focus on power consumption. On a realistic audio processing use case, we have shown how our framework can be leveraged for DSE and battery lifespan estimation. Our proposed architecture is fully modular, and can support much more complex configurations (e.g. higher n. of components) than the one shown here. Moreover, all models can be swapped with alternative implementations depending on the desired trade-off between accuracy and simulation speed, including functional and power models of the computing system and peripherals, and power models of the battery and DC/DC converters. Our future works include the extension of the framework to support other non-functional properties (e.g., thermal behaviour), and its application to more complex use cases.

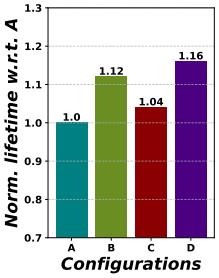

Figure 4: Battery life

Table 1: Configurations comparison

Conf.	Battery P. [mW]	DC/DC Eff. [%]	ΔSoC @1h [%]
A	1.67	88.3	5.2
B	1.49	88.4	4.6
C	1.60	91.6	5.0
D	1.43	91.3	4.4

ACKNOWLEDGMENTS

This work has received funding from the Key Digital Technologies Joint Undertaking (KDT-JU) under grant agreement No 101095947 and grant agreement No 101112274. The JU re-ceives support from the European Union's Horizon Europe research and innovation programme. This publication is part of the project PNRR-NGEU which has received funding from the MUR – DM 117/2023.

REFERENCES

[1] Accelera. 2024. *SystemC AMS.* https://systemc.org/overview/systemc-ams/
[2] Y. Chen et al. 2019. A SystemC-AMS Framework for the Design and Simulation of Energy Management in Electric Vehicles. *IEEE Access* (2019).
[3] A. B. Diggikar et al. 2012. Design and implementation of adaptive filtering algorithm for Noise Cancellation in speech signal on FPGA. In *ICCEET.*
[4] I. Elsadek et al. 2021. RISC-V Resource-Constrained Cores: A Survey and Energy Comparison. In *IEEE NEWCAS.*
[5] N. Bruschi et al. 2021. GVSoC: A Highly Configurable, Fast and Accurate Full-Platform Simulator for RISC-V based IoT Processors. In *IEEE ICCD.*
[6] Farnell. 2024. *RT8097A.* https://www.farnell.com/datasheets/2259301.pdf
[7] E. Fraccaroli and S. Vinco. 2023. Modeling Cyber-Physical Production Systems With SystemC-AMS. *IEEE Trans. Comput.* 72, 7 (2023), 2039–2051.
[8] V. Herdt et al. 2018. Extensible and Configurable RISC-V Based Virtual Prototype. In *IEEE FDL.*
[9] V. Herdt et al. 2020. RISC-V based virtual prototype: An extenlible and configurable platform for the system-level. *Journal of Systems Architecture* (2020).
[10] MikroElektronika. 2024. *Mic click.* http://www.mikroe.com/mic-click
[11] Panasonic. 2024. CG-425A/M3 Panasonic Battery | Mouser. https://www.mouser.it/ProductDetail/658-CG-425A-M3
[12] F. Pecheux et al. 2018. SystemC AMS Based Frameworks for Virtual Prototyping of Heterogeneous Systems. In *IEEE ISCAS.*
[13] Massimo Petricca et al. 2013. An automated framework for generating variable-accuracy battery models from datasheet information. In *IEEE ISLPED.*
[14] RISC-V International. 2024. *Emulators and Simulators.* https://wiki.riscv.org
[15] D. Rossi et al. 2014. Energy efficient parallel computing on the PULP platform with support for OpenMP. In *IEEEI.*
[16] S. Sharma. 2023. *RISC-V Architecture: A Comprehensive Guide to the Open-Source ISA.*
[17] STMicroelectronics. 2024. *ST1PS03.* https://www.st.com/en/power-management/st1ps03.html
[18] Greenwaves Technologies. 2024. *Gap9.* https://greenwaves-technologies.com
[19] S. Vinco et al. 2014. An open-source framework for formal specification and simulation of electrical energy systems. In *ACM/IEEE ISLPED.*
[20] Sara Vinco et al. 2017. A Layered Methodology for the Simulation of Extra-Functional Properties in Smart Systems. *IEEE TCAD* (2017).