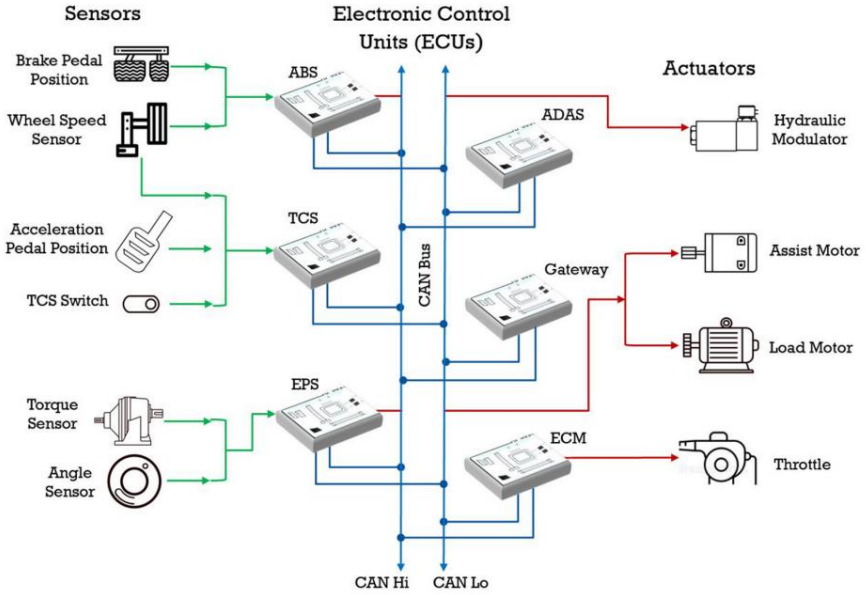
SystemC AMS Virtual Prototyping of RISC-V-Based Systems with Extra-Functional Awareness

Sara Vinco
Politecnico di Torino, IT

Politecnico
di Torino

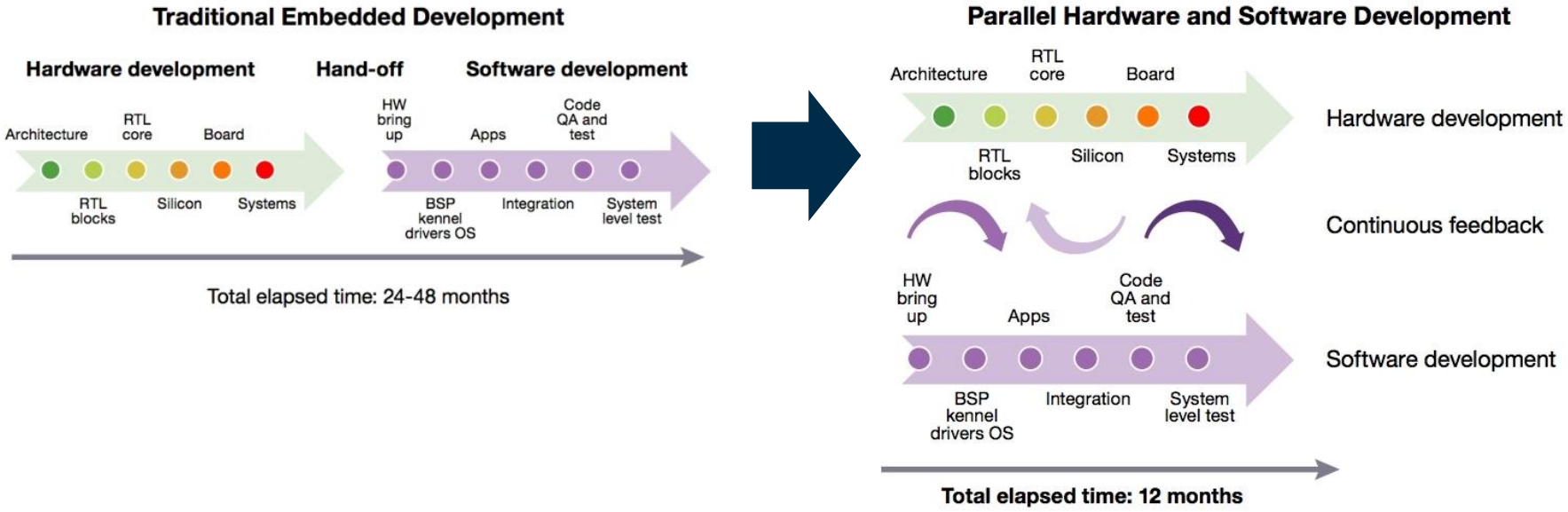
Virtual Prototyping

- Requirement for design of modern systems
  - Ability to do **system-level exploration** to comprehend the interactions involved among processor(s), sensors, peripherals, communication, etc.
  - Monitor the **impact** of design choices
  - Early in the system design: optimal design choices without **requiring a cost-prohibitive design overhaul**



M. R. Kabir, B. B. Yedla Ravi and S. Ray, "A Virtual Prototyping Platform for Exploration of Vehicular Electronics," in IEEE Internet of Things Journal, 2023

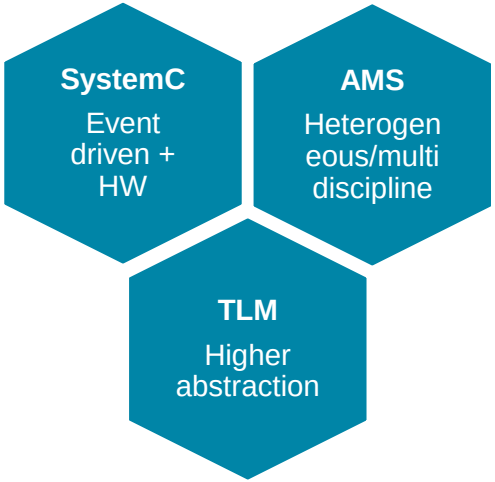
Virtual Prototyping



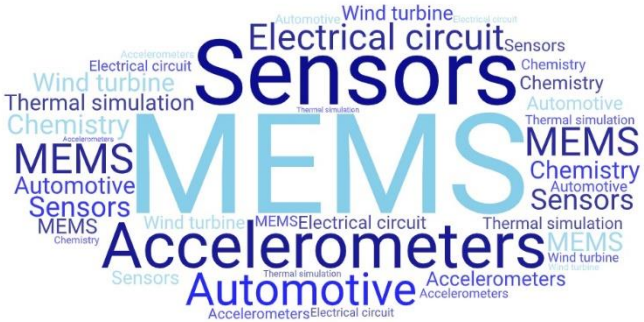
Synopsys, Virtual Prototyping,
https://www.synopsys.com/verification/virtual-prototyping.html

SystemC AMS and Virtual Prototyping

- **SystemC-based environment** long used for VP
  - Support for multiple aspects of a system
  - SystemC AMS especially useful when high degree of heterogeneity or multi-discipline modeling is required



- **Can provide added value to the RISC-V ecosystem!**



SystemC AMS and Virtual Prototyping

- Main goals:
  - Allow SW definition in the **ISS of the target RISC-V platform**
    - Same configuration as on the target SW stack and architecture
  - Model heterogeneous concerns of the system
    - Peripherals, communication, etc.
    - **Can not monitoring functionality in isolation**
      - Designers need to ensure correct behaviour of: power consumption, thermal behavior, etc…
      - Environment-awareness for applications like drones
      - RISC-V VP solutions do not model surrounding environment or extra-functional concerns

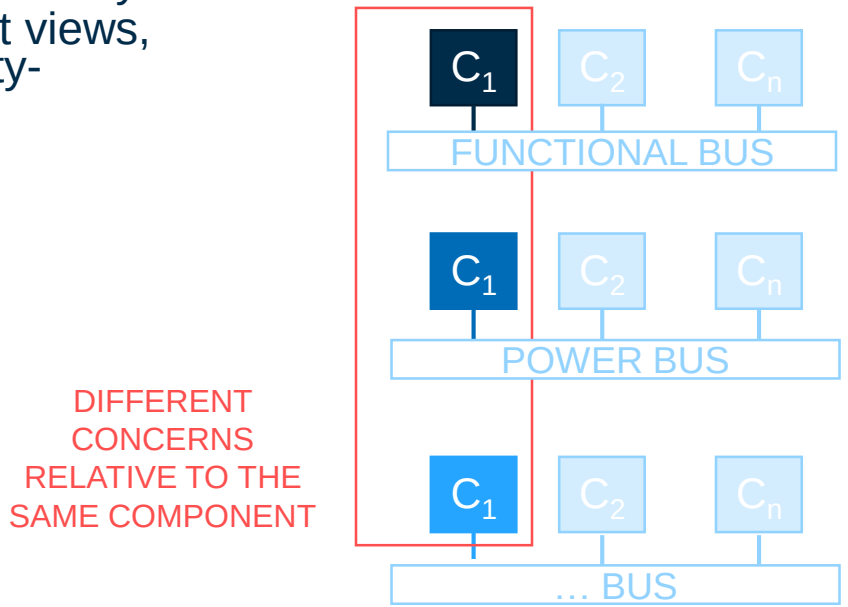
Simulation Architecture Requirements

- Ensure extensibility and modularity
  - Support multiple components and multiple concerns of a system
  - Allow integration of non-SystemC-based simulators

- Define a clever infrastructure that can:
  - Accommodate all concerns in an easy way
  - Clearly define the information flows
  - Cope with potentially different time scales

Simulation Architecture Structure

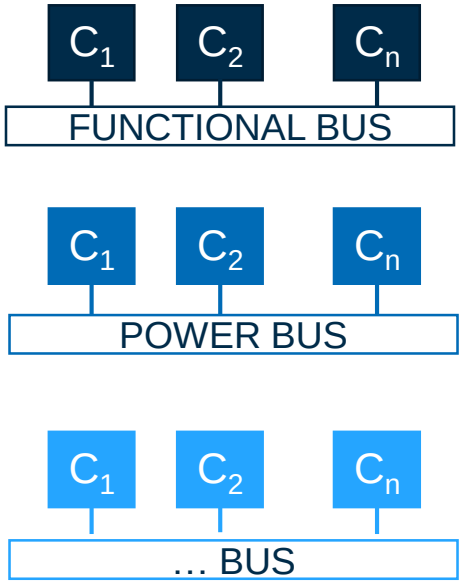
- **Multi-layer:** structured hierarchically
  - Each concerns corresponds to a simulation layer
  - Each component has/may have different views, i.e., it is implemented as a set of property-specific models
  - Allow integration with external tools



Simulation Architecture Structure

- **Multi-layer:** structured hierarchically
  - Each concerns corresponds to a simulation layer
  - Each component has/may have different views, i.e., it is implemented as a set of property-specific models
  - Allow integration with external tools

- **Bus-centric:** each property is simulated by adopting a specific “virtual bus”
  - Conveys and elaborates property-specific information
  - All layers have a common structure, easing synchro, information exchange and scalability
  - Export the standard architecture of functionality to extra-functional aspects



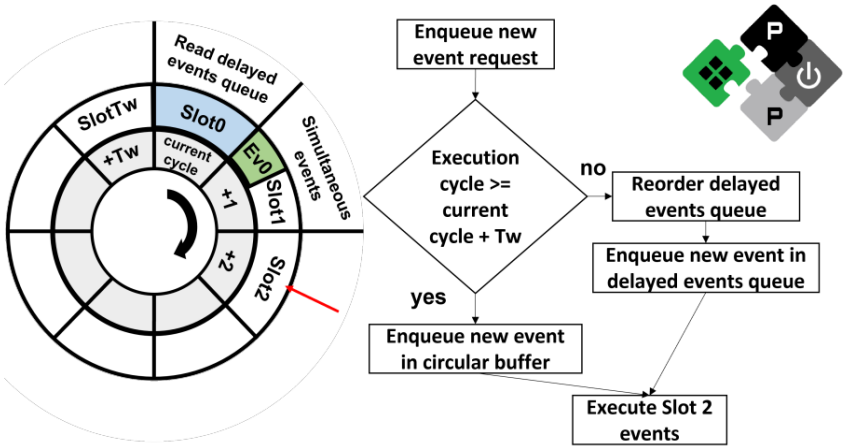
Integration with a RISC-V ISS

- Connect the functional model of the core to a **RISC-V ISS**
  - Avoid implementation of a custom RISC-V simulator
  - Allow same SW stack as the target implementation

- **GVSoC:**
  - Event-driven simulator for modeling RISC-V PULP platforms
  - Including cores, HW units, and peripherals
  - Event driven and almost cycle-accurate (10% accuracy)
  - Facilitates VP and design space exploration (DSE)
    - Realistic workload simulations and I/O interactions
    - Limited simulation time

Integration with a RISC-V ISS

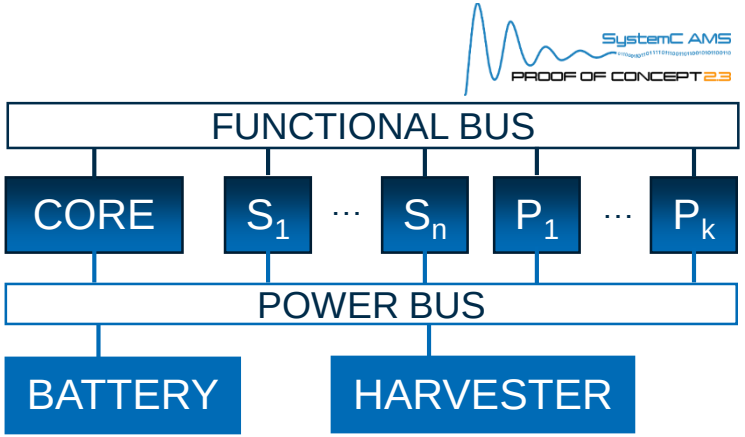
- Relatively easy to integrate in SystemC AMS:
  - **Event-driven**
    - Simulates a clock source with a queue of associated events
    - Circular buffer, allowing cycle-by-cycle execution, with simultaneous events processed sequentially
  - C-based implementation (+ Python automation)
  - API to interact with its simulation
    - Enqueue events, get power statistics, MMIO, etc.



N. Bruschi, G. Haugou, G. Tagliavini, F. Conti, L. Benini and D. Rossi, "GVSoC: A Highly Configurable, Fast and Accurate Full-Platform Simulator for RISC-V based IoT Processors," IEEE 39th International Conference on Computer Design (ICCD), 2021

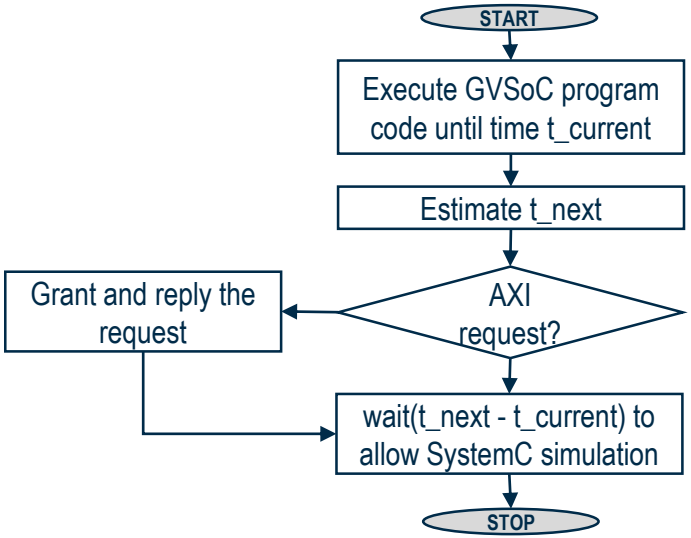
Simulation Setup

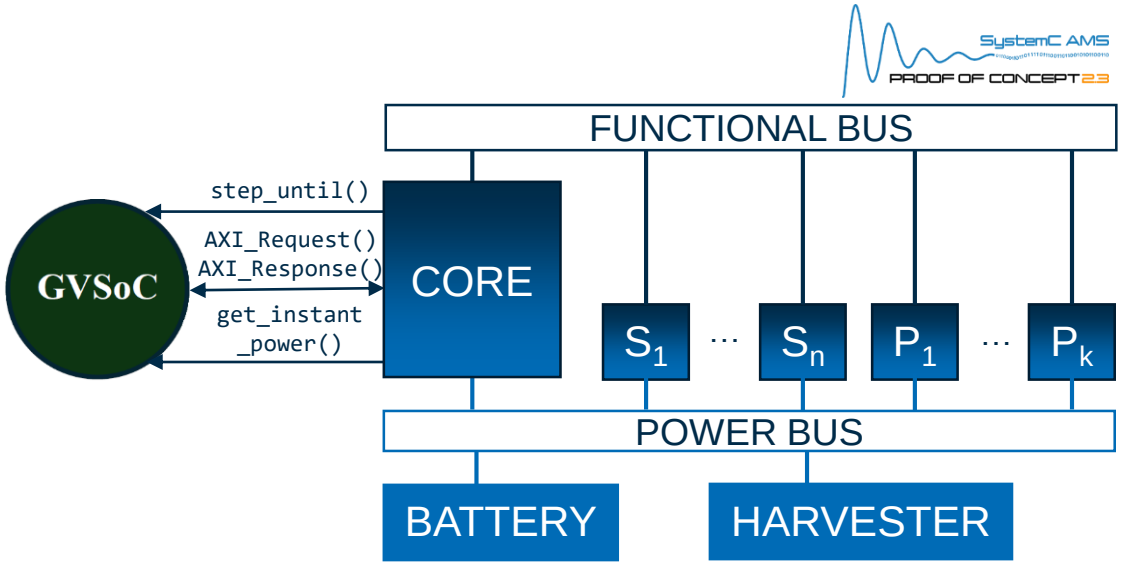
- **Functionality:**
  - RISC-V core + other components (digital, analog, AMS)
  - Full support for functionality modeling
- **Power:**
  - Power models of functional components (e.g., power consumption given device state)
  - Power-specific components like batteries
    - Easy to implement in SystemC AMS



Integration with GVSoC

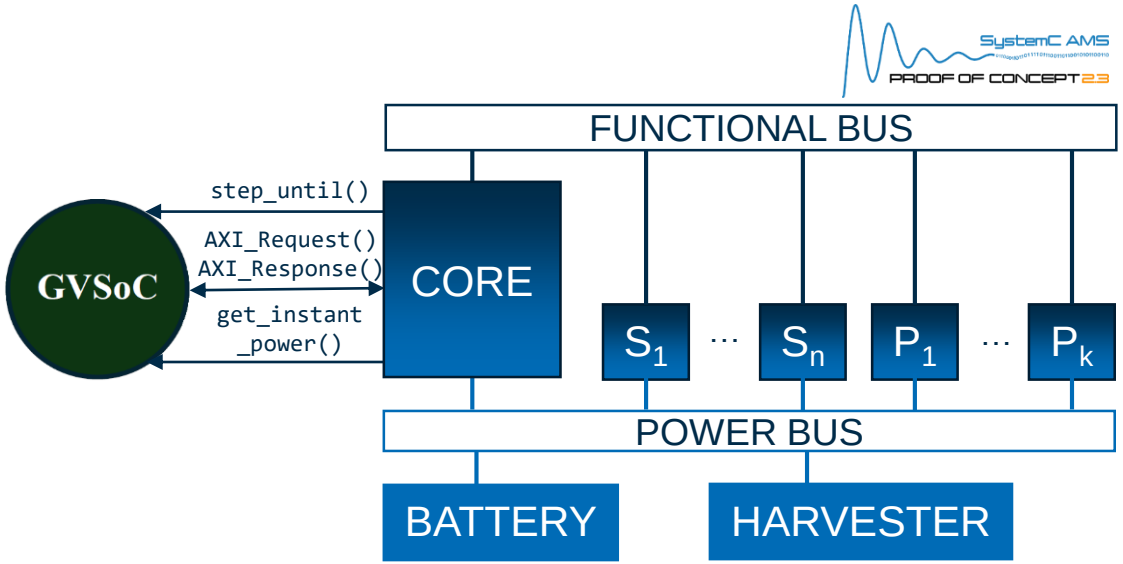
- **Functional integration:**
  - SystemC AMS as master of GVSoC simulation





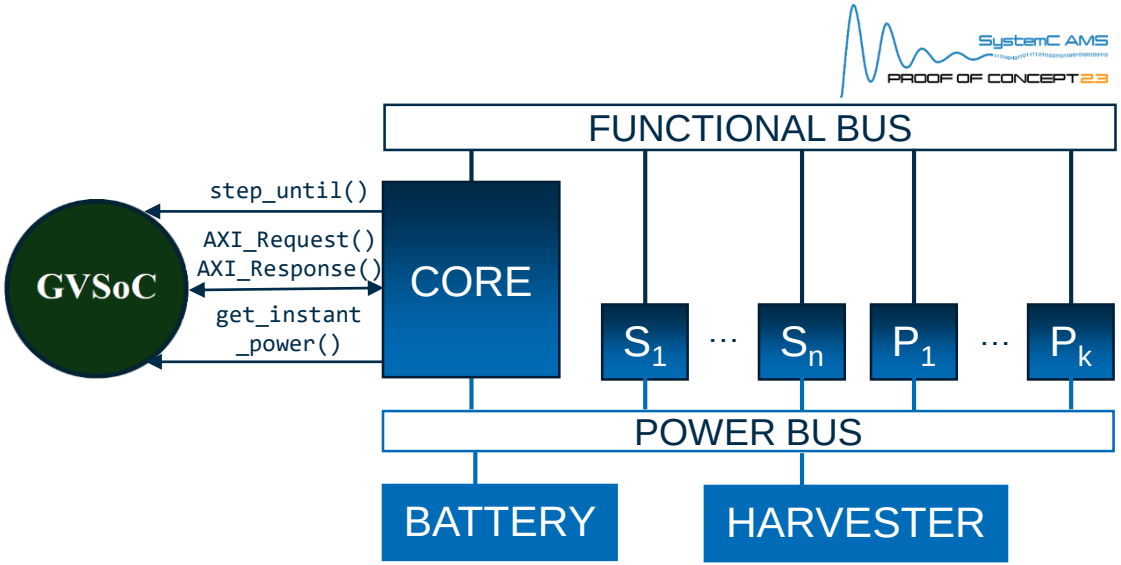
Integration with GVSoC

- **Power integration:**
  - Extract power consumption models from GVSoC with `get_instant_power()` primitive
  - Bring to power bus to take into account core consumption



System Modeling

- Still allows to benefit from the **flexibility of SystemC AMS** for modeling the other aspects of the system



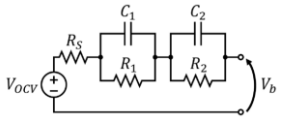
$$T = \int_{R_0}^{.97R}\int_{0}^{2\pi} 0.5N\rho cu\left(a\alpha u_{pl} - c_d u_{pr}\right)d\psi dr/(2\pi)$$

$$Q = \int_{R_0}^{R}\int_{0}^{2\pi} 0.5N\rho cur\left(a\alpha u_{pr} + c_d u_{pl}\right)d\psi dr/(2\pi).$$

ROTOR EQUATIONS (LSF)

$$u_{pl}(r,\psi) = \omega r + v_x sin(\psi)$$
$$u_{pr}(r) = v_i + v_z$$
$$u(r,\psi) = \sqrt{u_{pl}^2 + u_{pr}^2}$$
$$\varphi(r,\psi) = tan^{-1}\left(u_{pr}/u_{pl}\right).$$

AERODYNAMIC EQUATIONS (TDF)



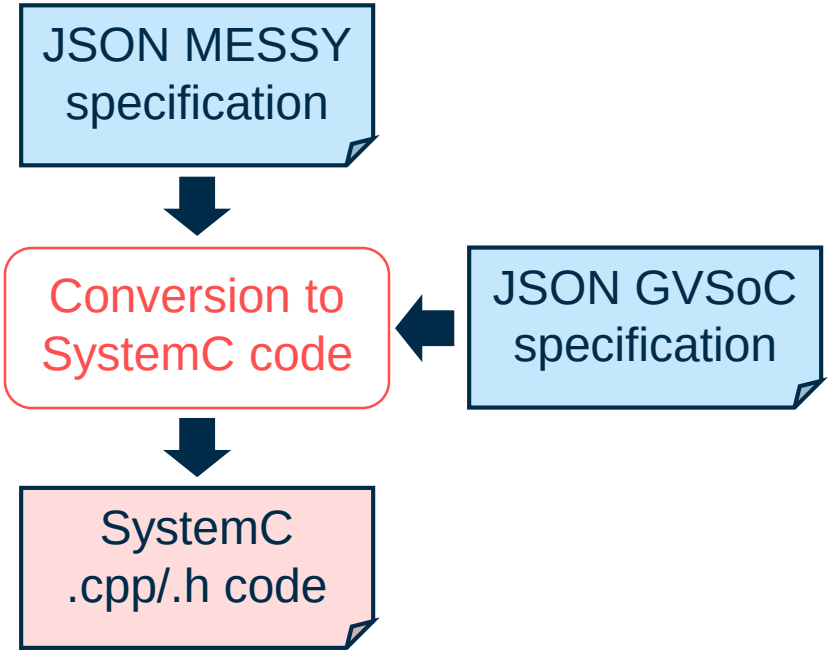
BATTERY LINEAR CIRCUIT (ELN)

Code Generation Automation

- Currently automated in the MESSY tool:
  - *MESSY: Multi-layer Extra-functional Simulator in SYstemc*
  - Repository: https://github.com/eml-eda/messy
  - Documentation: https://eml-eda.github.io/messy/

- System specifications in JSON
  - Automated generation of the corresponding SystemC AMS code:
    - Modules instantiation + connection
    - GVSoC instantiation
    - Support for automatic generation of simple power models (e.g., based on lookup tables or power state machines)
  - Ease setup of simulation infrastructure

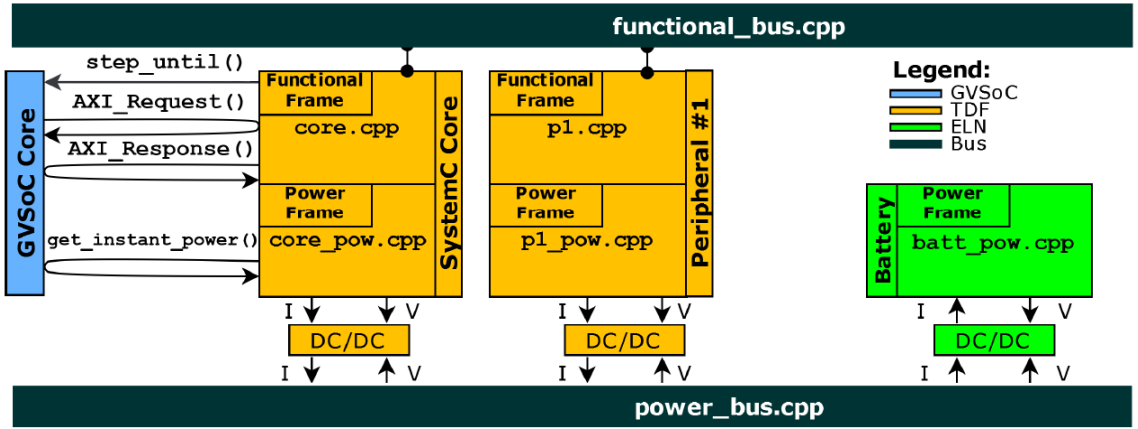
Code Generation Automation




```
{
   "resolution":"SC_MS",
   "bus":{"vref":3.3},
   "core":{
      "vref":1.8,
      "iss":"gvsoc",
      "config":{
         "path":"./gvsoc_deps/gvsoc_config.json",
         "axi_path":"/chip/axi_proxy"
      },},
      "converter":{
         "input_variable":"current", "lut":{ … } // LUT specs
      } },
   "peripherals":{
      "sensors" : {
         "mic_click": {
            "power": true,
             "vref": 3.3,
            "register_memory" : 256,
            "states":{ … } } // PSM specs
         }
      },
      …
}
```

Example of Adoption of MESSY

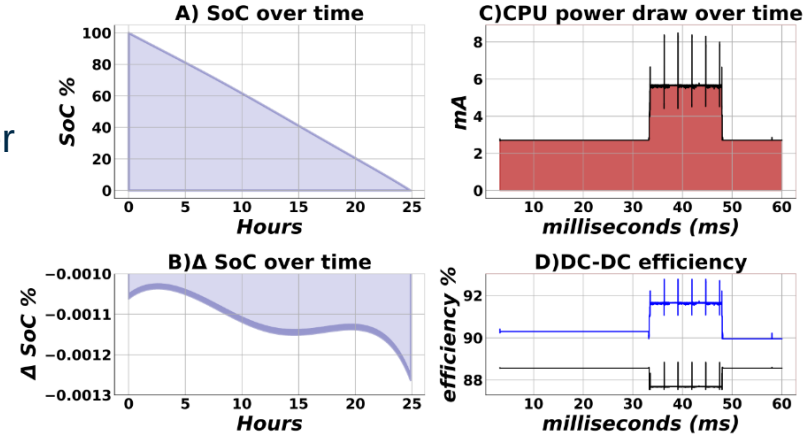
- **Adaptive Filtering Workload** (e.g., noise cancellation, echo suppression, and dynamic equalization)
  - GVSoC for running SW
  - Mic Click microphone (120-160μA)
  - Panasonic CG-425A32mAh @3.8V battery
  - DC/DC converters



M Amine Hamdi, G Pollo, M Risso, G Haugou, et al. Integrating SystemC-AMS Power Modeling with a RISC-V ISS for Virtual Prototyping of Battery-operated Embedded Devices. ACM Int Conf on Computing Frontiers: Workshops and Special Sessions. 2024

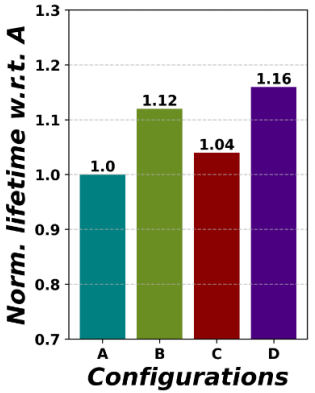
Example of Adoption of MESSY

- **Simulation until battery SoC = 0%**
  A. SoC decreases almost linearly (low currents demand)
  B. Can emphasize the non-linearities of the discharge behavior on zoom-ins
  C. GVSoC power demand:
     - Waits for new sensor data
     - Higher consumption: activation of GAP9's cluster; spikes for inner processing of FIR filter
     - Final wait before following iteration
  D. Corresponding DC/DC efficiency



Example of Adoption of MESSY

- Apply **design space exploration**
  A. Former experiment (40 filter taps)
  B. Reduce the processing workload by decreasing the filter taps to 20 to mimic a quality-of-service reduction to increase the battery lifetime
     → +12% battery life
  C. Replacing the GVSoC DC/DC converter with a potentially more efficient one
     → +4% battery life
  D. Combine both modifications (B+C)
     → +16% battery life
- **Minor HW/SW changes may improve lifetime and autonomy of the system**
- Example of fine-grained power-aware analysis enabled at a limited simulation cost



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

What’s Next

- Adoption of IP-XACT for specification of system architecture and connection
  - Stardard + easy to be integrated in code generation flows
- Adoption of COSIDE for extra-functional modeling
  - Support for wider range of model descriptions that better support non-functional domains
- Support for other ISSs
  - Including native SystemC-based ones and Qemu
  - Integration with HIL solutions

Thank you for your attention!

- … and of course any questions welcome!

- Contact info:
  - Sara Vinco, Politecnico di Torino, IT
    sara.vinco@polito.it

DUMAREY

COSEDA technologies