MEbots: Robotic Simulation using SystemC

Sara Vinco
Politecnico di Torino, IT

Politecnico di Torino

- These slides have been presented at the SystemC Evolution Day in June 2025

- https://systemc.org/events/scef202506/

Home Overview Events Resources Playground Get Involved Search GitHub

SystemC Evolution Fika - 26 June 2025

Workshop on the Evolution of SystemC Standards

The SystemC Evolution Fika is a series of online workshops to discuss the latest SystemC developments and applications. We refer to these workshops as *fikas*, to honor the fika tradition of sharing a coffee, slowing down a bit, and talking about things that we care about.

EVOLUTION FIKA

Event information

Date: **26 June 2025**
Time: **16:00 - 18:00 CEST**
Location: Online, Virtual Workshop.

Registration

Registration is free of charge. Register here.

NOTE: After registration you will receive an email including meeting details to attend the online event.

Organization Team

- Martin Barnasconi, NXP
- Mark Burton, Qualcomm
- Jerome Cornet, STMicroelectronics
- Peter de Jager, Intel
- Nils Luetke-Steinhorst, Cadence

Program

Tentative program, subject to change

Time (CEST)	Title	Presenter(s)
16:00 - 16:10	**Welcome & Introduction**	Mark Burton, SystemC Fika Chair
16:10 - 16:40	**DVCon Europe 2025 SystemC Modeling Challenge** Abstract	Christoph Grimm, Rheinland-Pfälzische Technische Universität (RPTU) Germany Moritz Herzog, Rheinland-Pfälzische Technische Universität (RPTU) Germany
16:40 - 17:10	**MEbots: Robotic Simulation using SystemC** Abstract	Sara Vinco, Politecnico di Torino Italy
17:10 - 17:30	**SystemC CCI Summer of Code** Abstract	Antonios Salios, RWTH Aachen University Lukas Jünger, Machineware GmbH Germany
17:30 - 18:00	**A SystemC-UVM testbench for a student lab exercise** Abstract	Thilo Vörtler, COSEDA Technologies Jens Schönherr, Hochschule für Technik und Wirtschaft Dresden Germany
18:00 - 18:10	**Q&A and Closing**	Mark Burton, SystemC Fika Chair

SystemC scope

2002

What does SystemC actually provide?

- SystemC 1.0 provided RTL and behavioral HDL modeling capabilities. HW is modeled using zero-delay semantics for combinational logic. Signals are modeled using 01XZ, “C” data types and complex data types can also be used within signals.
- SystemC 1.0 includes good support for fixed point modeling.
- SystemC 1.1 beta and 1.2 beta provided some limited communication refinement capabilities.
- SystemC 2.0 has more general system level modeling capabilities with *channels, interfaces*, and *events*.

2014

What is SystemC Used For?

- Behavioral Modeling and Reference Models
- Virtual Platforms (aka Software Virtual Prototypes)
  - Architectural exploration, performance modeling
  - Software development
  - Reference model for functional verification
- High-Level Synthesis (C/C++)

Why SystemC AMS extensions?

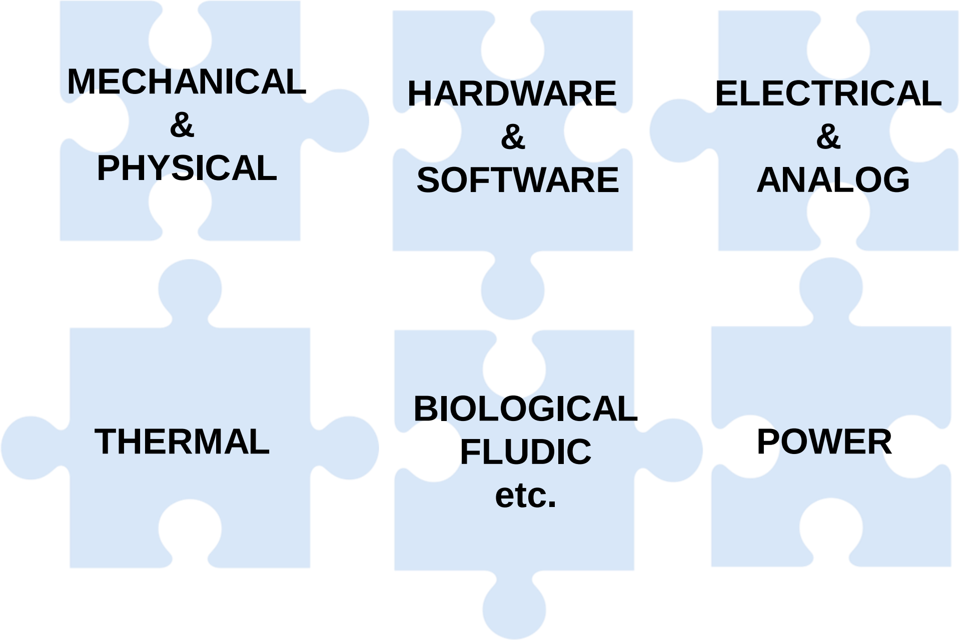
- **Unified and standardized modeling language to design and verify embedded mixed-signal architectures**
  - *Abstract AMS model descriptions* supporting a design refinement methodology, from functional/algorithm down to implementation views
  - Enabling *tool-independent exchange* and reuse of AMS models and building blocks
  - System-level language for analog *and* digital signal processing
- **Facilitate the creation of mixed-signal virtual prototypes**

2014

The context changed

- Evolution from embedded devices to complex, multi-physics, intelligent cyber physical systems

MUST TAME THE COMPLEXITY & HETEROGENEITY!

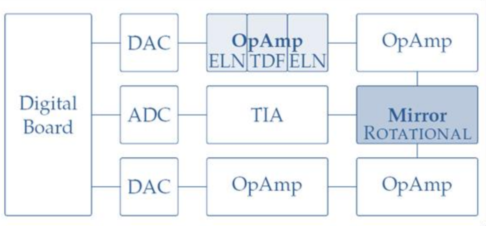


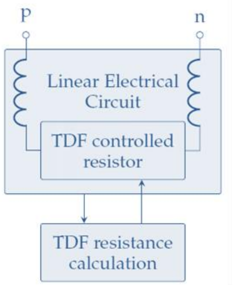
Requires cross-domain simulation!

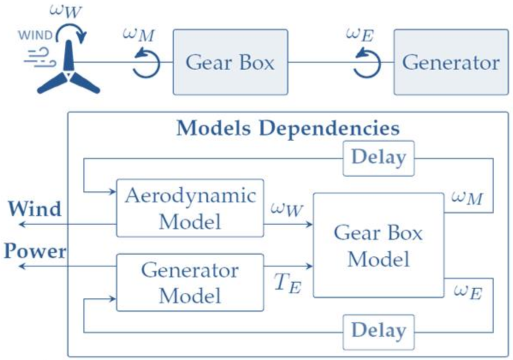
- Virtual prototyping to allow **early design space exploration**
  - Model of non-functional effects to improve reliability, power efficiency, thermal robustness
  - Allow validation of cross-domain interactions
- **Unified modeling** to reduce toolchain complexity and accelerate prototyping
  - Enhance reliability and anticipate bug/fault management
  - Time synchronization for timing estimation

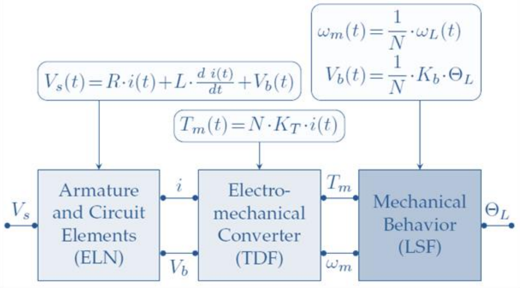
SystemC and heterogeneity

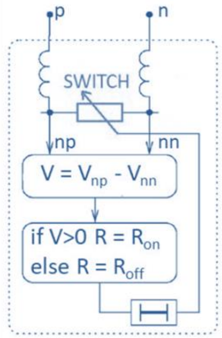
- Over time, SystemC and its extenstions proved to cover heterogeneous domains effectively…











Modified from E. Fraccaroli and S. Vinco, Modeling Cyber-Physical Production Systems With SystemC-AMS, IEEE Transactions on Computers, 2023

SystemC and heterogeneity

- Over time, SystemC and its extenstions proved to cover heterogeneous domains effectively…
- … with some limitations
  - Solver-based issues that affect dynamic systems (limited physics based simulation, no cyclic dependencies, etc.)
  - Lack of user interaction and of environment modeling
  - Heterogeneity management lies in the expertise of the designer

```
SC_MODULE{
  SC_METHOD(policy);
  sensitive << clk;
}
```

```
SC_MODULE{
  c = new sca_eln::sca_c("cap");
  r = new sca_eln::sca_r("res");
}
```

```
SCA_TDF_MODULE{
  if(in.read()) > 0
    out.write(f(in.read()));
}
```

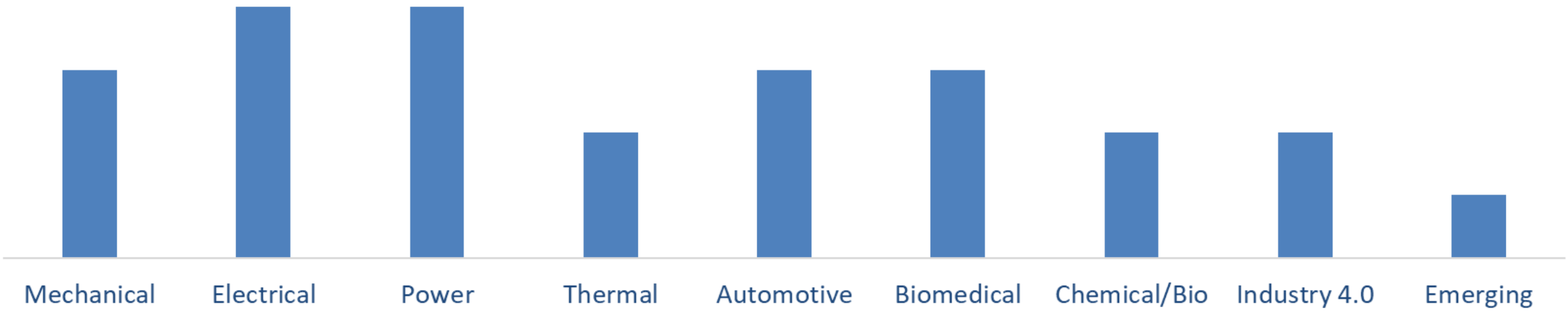
```
SC_MODULE{
  g = new sca_lsf::sca_gain("gain");
  d = new sca_lsf::sca_dot("dot");
}
```

```
SC_MODULE{
  payload->set_command( cmd ); payload->set_address( i );
  socket.b_transport(payload, delay);
}
```

SYSTEMC™

SystemC and heterogeneity

- The impact of SystemC-AMS went far beyond the initial expectations
  - Applications to non-standard contexts
  - Simultaneous presence of multiple domains in a single simulation run



Robots as an *ideal* case study

- Naturally embody the core characteristics of cyber-physical systems
  - Inherent multi-domain nature

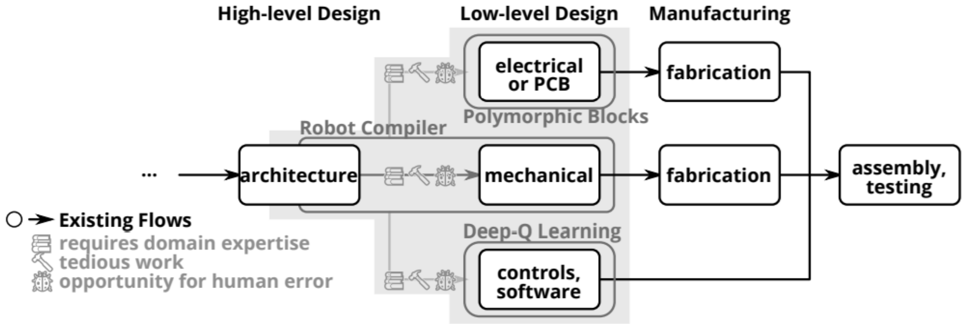
Subsystem	Domain
Actuators & motors	Mechanical & Electrical
Sensors	Analog + Digital + Mechanical
Control loops	Digital with real-time constraints
Power system	Electrical + Thermal
Networked digital systems (CAN, I2C, etc.)	Communication
Environment interaction	Thermal, mechanical, fluidic, etc.

Robots as an *ideal* case study

- Naturally embody the core characteristics of cyber-physical systems
  - Inherent multi-domain nature
  - Highly coupled hardware and software co-design
  - Easily observable functional + extra-functional properties
    - Their failure modes (e.g., unstable flight, dropped payloads, missed trajectories) are directly tied to system design errors
  - Relevance in many modern applications

Robots as a *challenging* case study

- Design flow still compartimented for different domains
  - Separate tools and flows
- Lack of a holistic model-based perspective
  - E.g., focus on mechanical behavior, with no support for power simulation
  - Require integration between different tools and simulation infrastructures



Modified from A. Chattoraj, R. Lin, D. Martinez and A. Mehta, A Cross-domain, Computational Robot Design Flow from Morphology to Fabrication, IEEE CASE, 2024

Three main steps

1. Managing the heterogeneity through layering
2. Integrating an Instruction Set Simulator for software support
3. Integrating a robotic simulator for sensing and visualization

- References:
  - S. Vinco et al., A Layered Methodology for the Simulation of Extra-Functional Properties in Smart Systems, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2017
  - M.A. Hamdi et al., *Integrating SystemC-AMS Power Modeling with a RISC-V ISS for Virtual Prototyping of Battery-operated Embedded Devices*, CF, 2024 (plus github repository)
  - G. Pollo et al., *MEbots: Integrating a RISC-V Virtual Platform with a Robotic Simulator for Energy-aware Design*, ISPLED 2025 (available on arxiv)

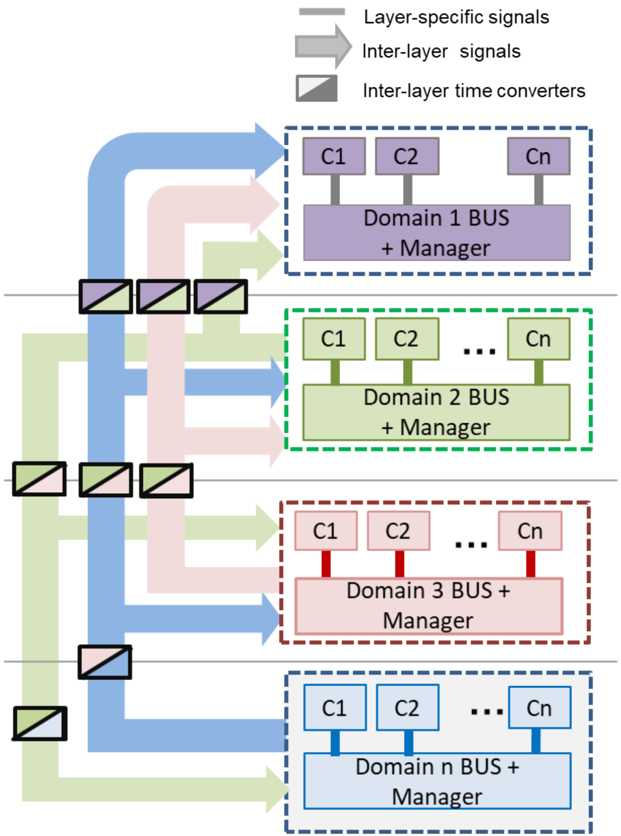
Managing heterogeneity through layering

- **Define a clever simulation structure**
  - Accommodate all domains in an easy way
  - Clearly define the information flows and the mutual exchange between domains
  - Cope with potentially different time scales
  - Ensure extensibility and modularity

- **Exploit the flexibility of SystemC**
  - One simulation kernels supporting multiple levels of abstraction
  - Native converters (e.g., between TDF and ELN or LSF)

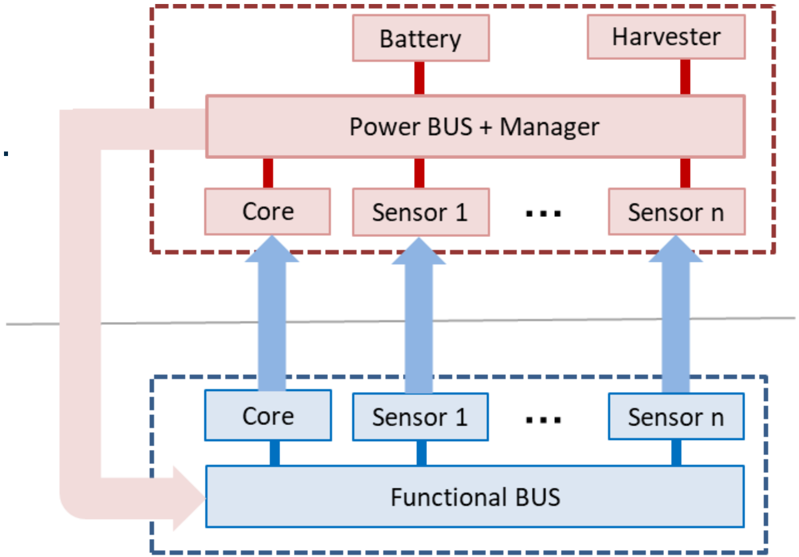
Managing heterogeneity through layering

- **Bus-centric**: each domain is simulated by adopting a specific “virtual bus”
  - Conveys and elaborates domain-specific information
  - Export the standard architecture of functionality
- **Multi-layer**: structured hierarchically
  - Each component may have different models, one per domain/bus
- **Mutual-impact of domains** preserved:
  - Communication between different models of a component



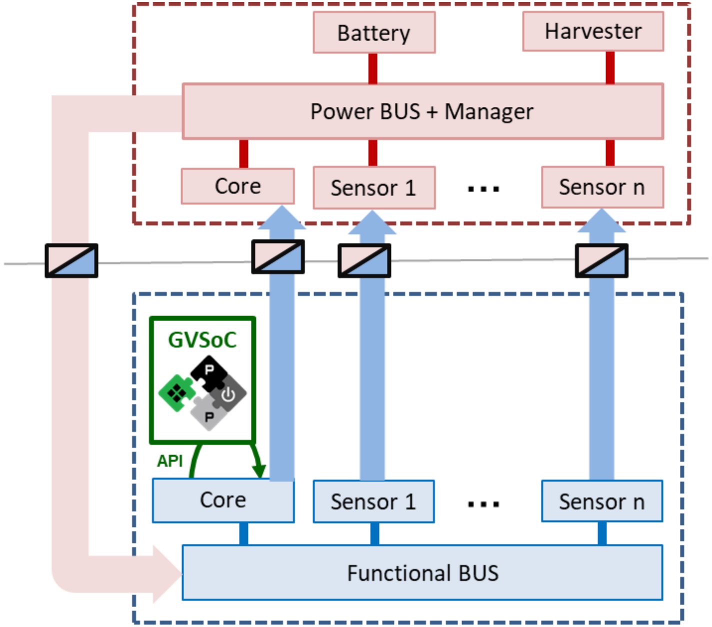
Managing heterogeneity through layering

- Example:
  - Functional layer:
    - Instruction processing, peripheral simulation, etc.
    - Influenced by power (e.g., stop due to discharged battery)
  - Power layer:
    - Focus on power distribution through bus with policy
    - Functional components as consumers
    - Battery and harvester as providers
  - Intrinsecally models the inter-dependency between different aspects of the system



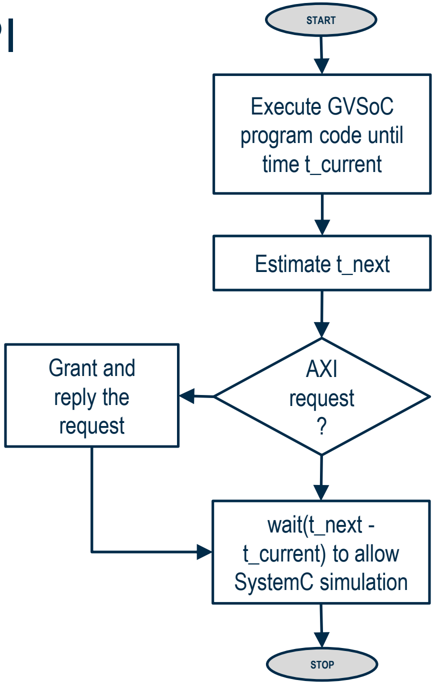
Integrating an Instruction Set Simulator

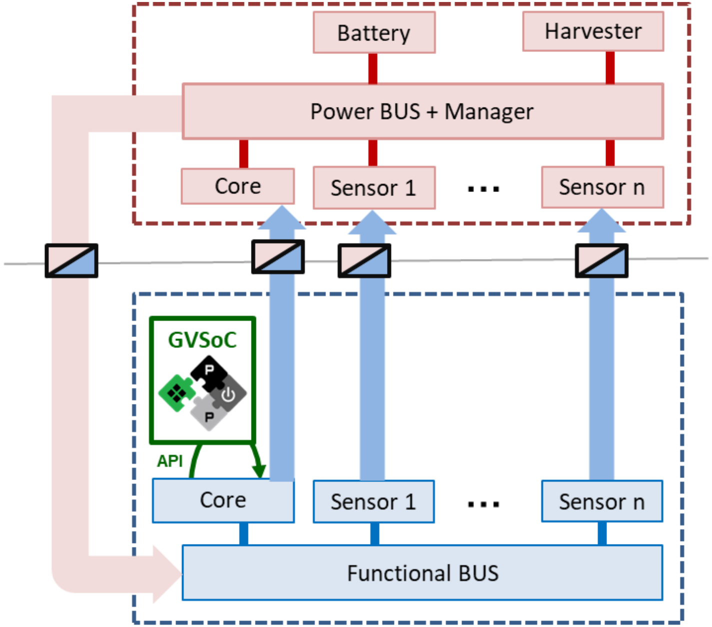
- … to avoid reinventing the wheel
  - SystemC is C++ based, so integration of C++-based tools is quite straightforward
  - Integrate GVSoC RISC-V ISS as slave
    - Open-source event-driven simulation platform
    - Almost cycle-accurate = good timing estimation
  - Based on API to:
    - Activate and synchronize event queue
    - Gather power estimations



Integrating an Instruction Set Simulator

- Communication protocol
  - SystemC uses GVSoC API
    - Advance SoC simulation in time `step_until()`
    - Gather power estimation `get_instant_power()`
  - Handle software I/O requests to external devices
    - Forward them through the bus to corresponding SystemC modules



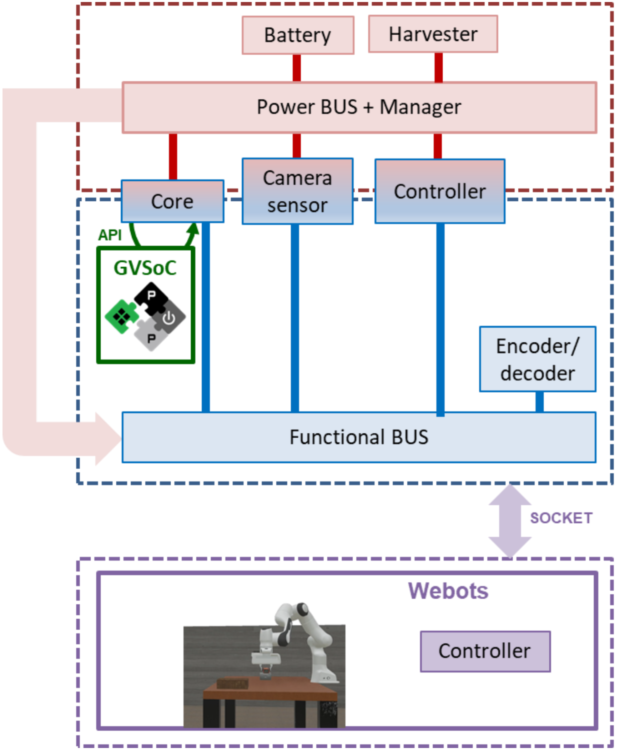


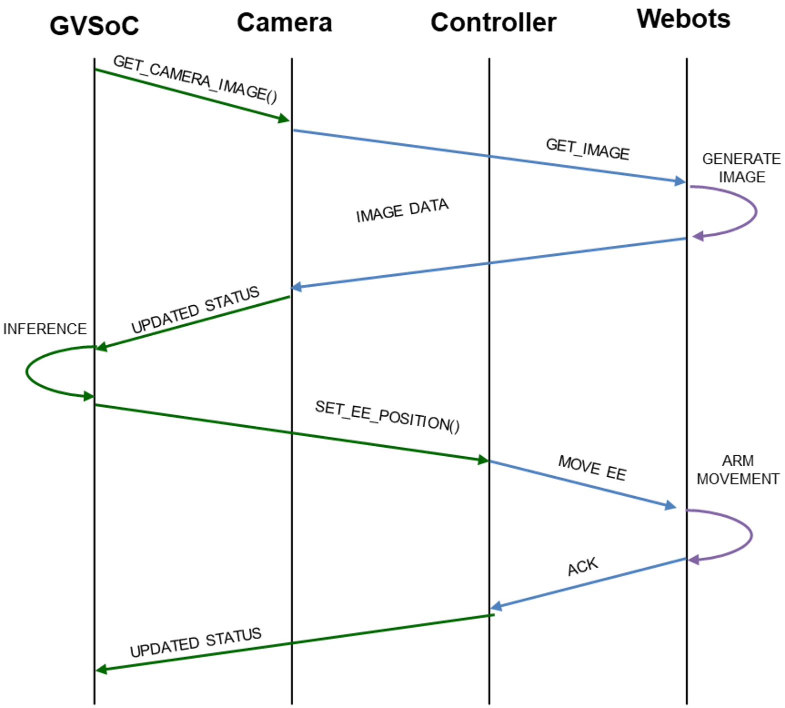
Integrating a robotic simulator for sensing and visualization

- Missing parts
  - User interaction (e.g., to intercept keypressed)
  - Visualization of robot movement
  - Physical and environmental aspects such as wind
- Not trivial to be modeled in SystemC…

- **Integration with Webots**
  - Open source and multi-platform desktop application used to simulate robots
  - Easily complete robotics simulations using a library including robots, sensors, actuators, objects and materials
  - Each robot is assigned a “controller” file to control the machine within the virtual world
  - Allow communication through Unix sockets to connect with external applications

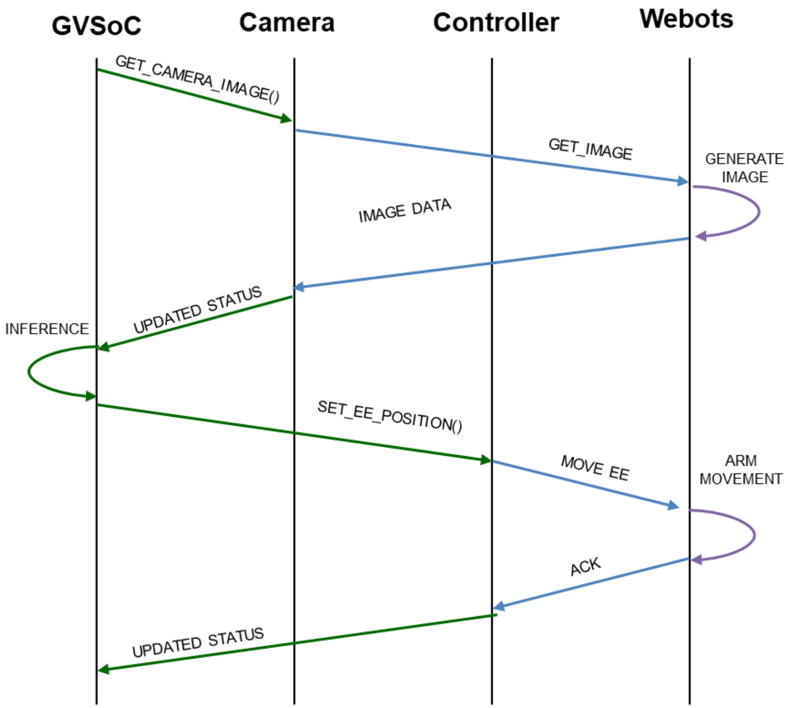
Integrating a robotic simulator for sensing and visualization



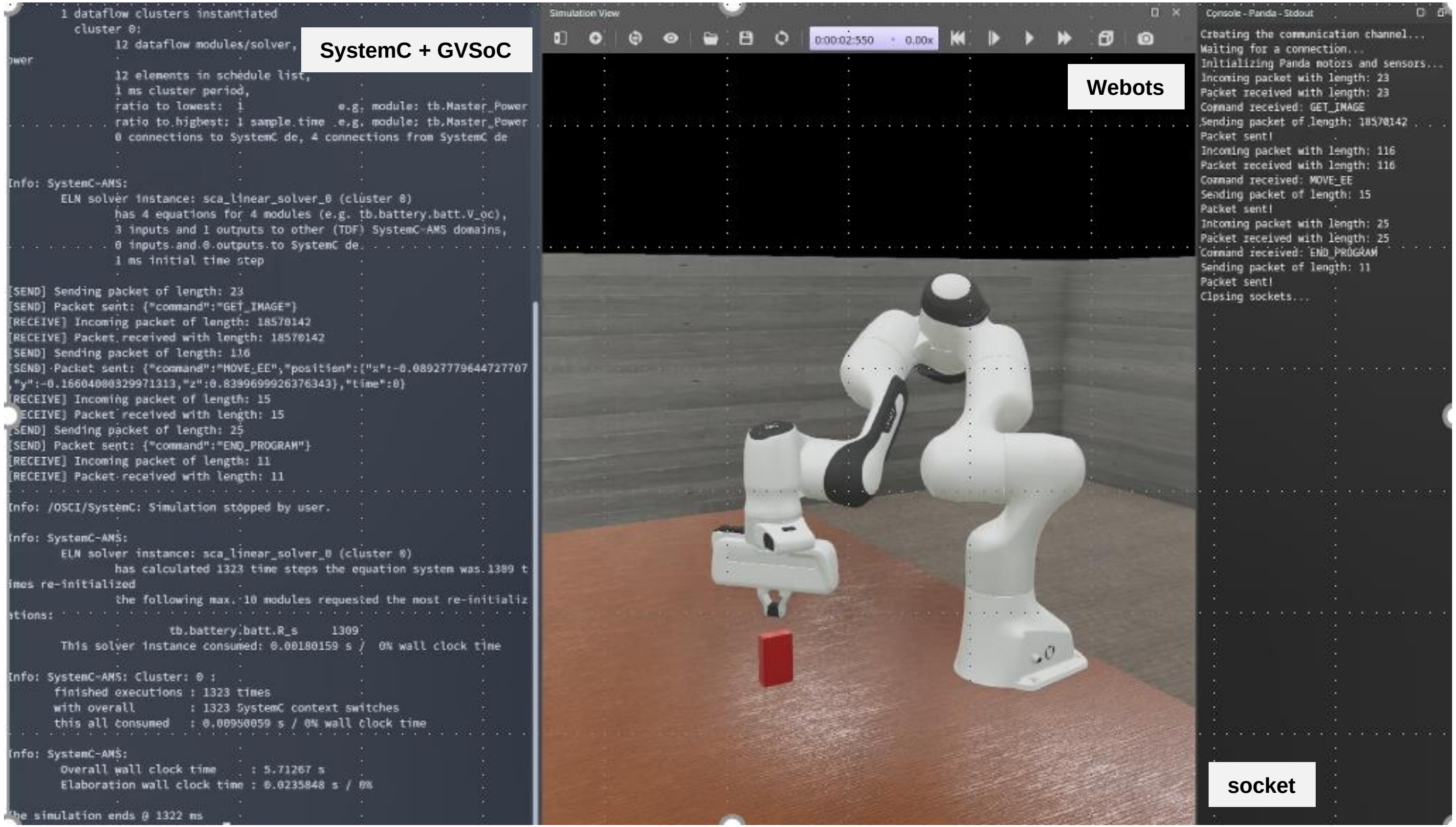


Integrating a robotic simulator for sensing and visualization

- **Socket-based synchronization** with SystemC as master
  1. SystemC activates GVSoC
  2. GVSoC requests an image from the Webots camera module
  3. The image is received and parsed through an object detection network to find a target block
  4. The inference returns the spatial coordinates of the target
  5. The generated coordinates are sent to Webots (and time is aligned to GVSoC time)
  6. The controller moves the robotic manipulator to pick up the block and returns an ack



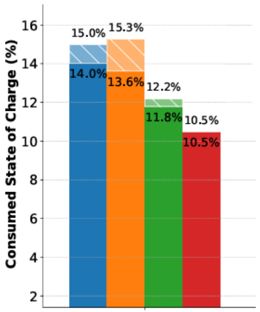
Pick and place robot arm application

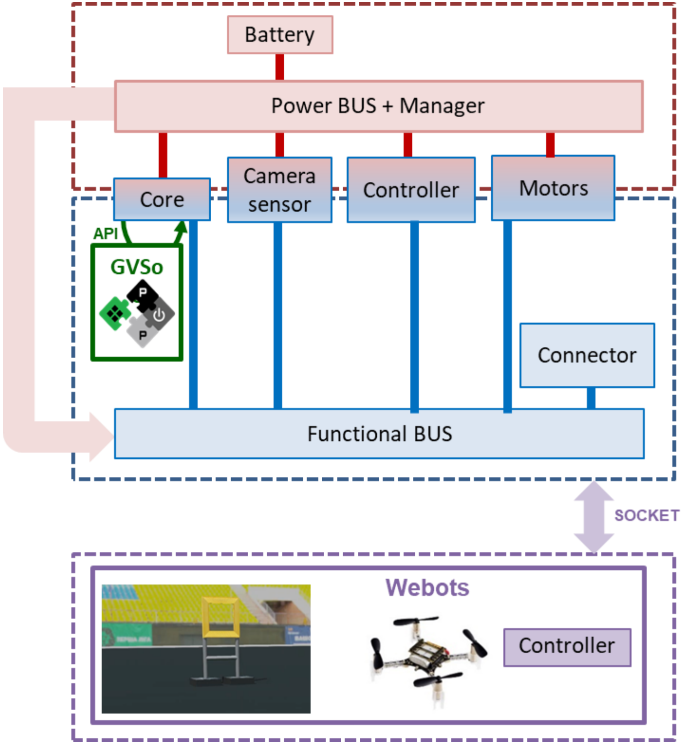


Drone gate traversal

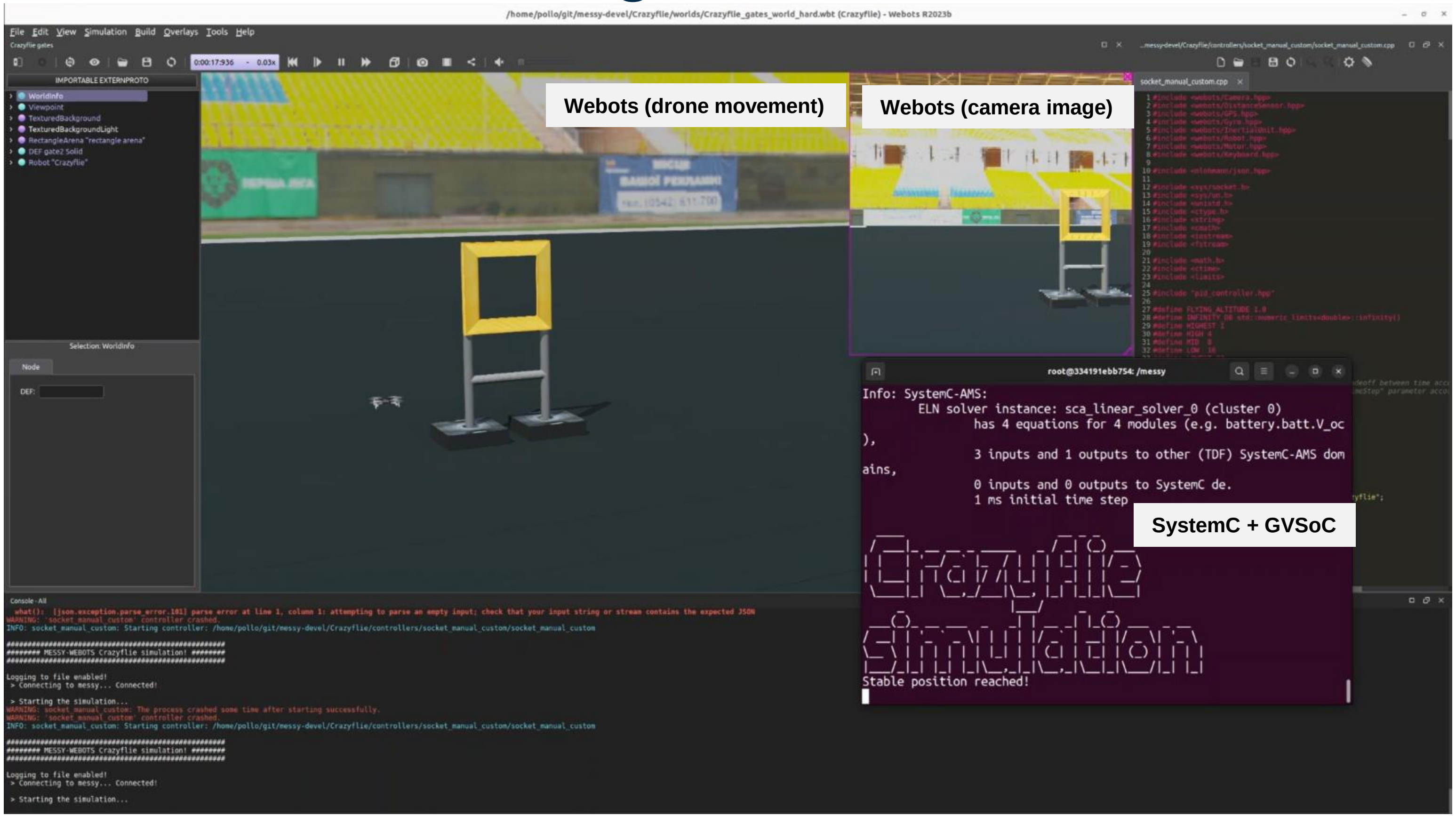
- Crazyflie nano-UAV
  - Gate traversal application
    - CNN processing camera images
  - Power exploration
    - Compare 4 different battery packs
      - +4% flight time with same payload capacity
      - +33% flight time with 29.5% increased battery weight

BATTERY	CAPACITY	WEIGHT
Original	250mAh	7.1g
UFX	250mAh	5.8g
Cyclone	300mAh	8.1g
Lipol	350mAh	9.2g

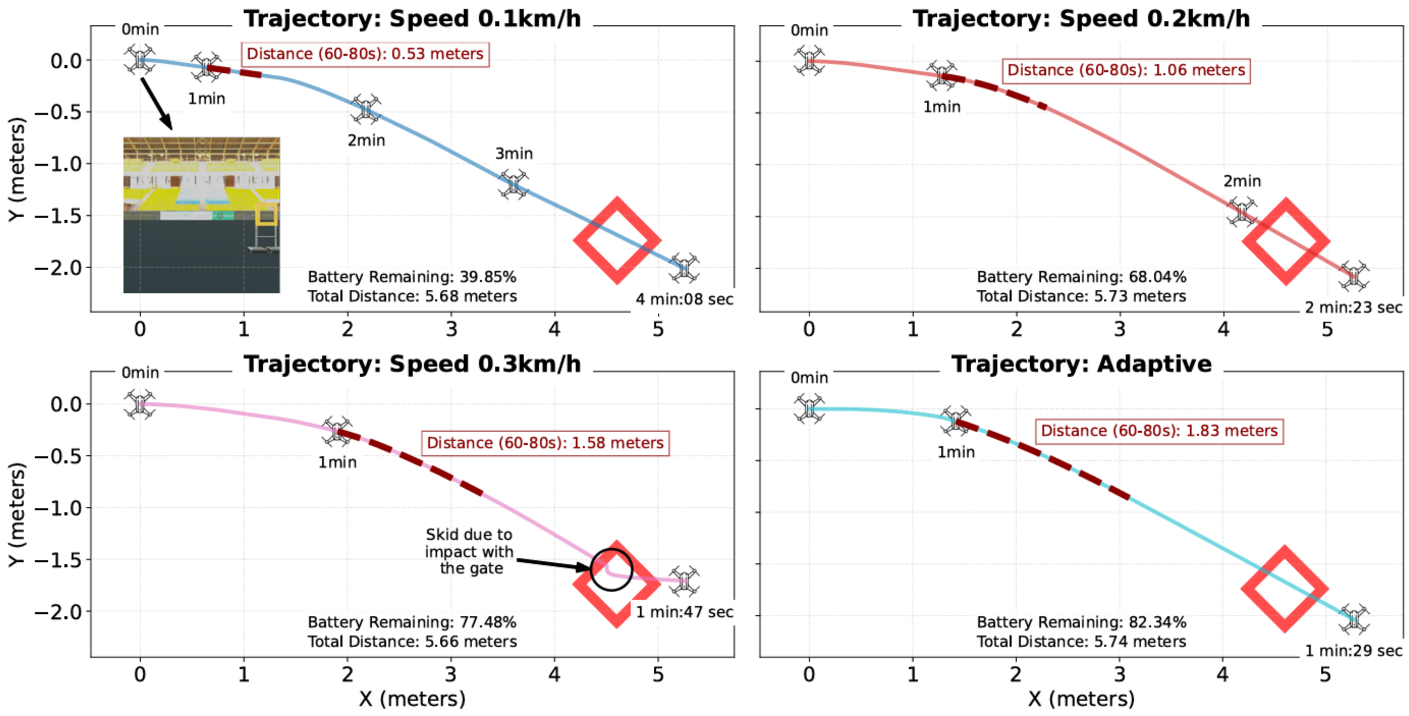




Drone gate traversal



Drone gate traversal



Conclusions

- SystemC environment at large extremely successful for multi-domain modeling
  - Margin for extending support
    - E.g., non linear behaviors, dynamic systems, power AC domain
  - Benefits from the integration with external simulators
    - Avoid reinventing ISS-level timing estimation, environment modeling, etc.
  - Each integration lies on custom solutions
    - Promising direction: define standardised interfaces to ease integration (e.g., FMI, Federated Simulation)

& Thank you!

References

- Stuart Swan, *An Introduction to System Design with SystemC*, UCI SOC DESC/MODELING, 2002
- John Aynsley, *Virtual Prototyping using SystemC and TLM-2.0*, DVCon, 2014
- K. Einwich, C. Grimm, M. Barnasconi and A. Vachoux, *Introduction to the SystemC AMS DRAFT standard*, IEEE SOCC, 2009
- E. Fraccaroli and S. Vinco, *Modeling Cyber-Physical Production Systems With SystemC-AMS*, IEEE Transactions on Computers, vol. 72, no. 7, pp. 2039-2051, 2023
- S. Vinco, Y. Chen, F. Fummi, E. Macii, M. Poncino, *A Layered Methodology for the Simulation of Extra-Functional Properties in Smart Systems*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 36, no. 10, pp. 1702-1715, 2017
- M. A. Hamdi, G. Giovanni, M. Risso, G. Haugou, A. Burrello, E. Macii, M. Poncino, S. Vinco, D. J. Pagliari , *Integrating SystemC-AMS Power Modeling with a RISC-V ISS for Virtual Prototyping of Battery-operated Embedded Devices*, Computing Frontiers, 2024
- G. Pollo, M. A. Hamdi, M. Risso, L. Ruotolo, P. Furbatto, M. Isoldi, Y. Chen, A. Burrello, E. Macii, M. Poncino, D. J. Pagliari, S. Vinco, *MEbots: Integrating a RISC-V Virtual Platform with a Robotic Simulator for Energy-aware Design*, ISLPED, 2025
- Icons from kayaheart, iconcheese, Magicon, from nounsproject

To hear more about our research: sara.vinco@polito.it https://github.com/eml-eda