Homogeneous Modeling and Simulation of Cyber-Physical Energy Systems using HDLs

Sara Vinco

Politecnico di Torino

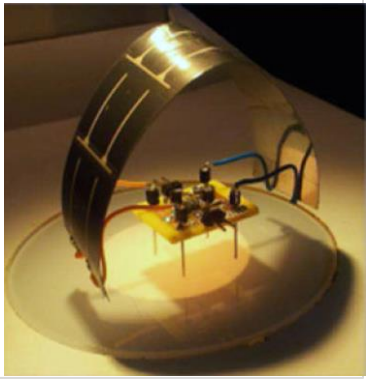
Electronic Design Automation Group

http://eda.polito.it

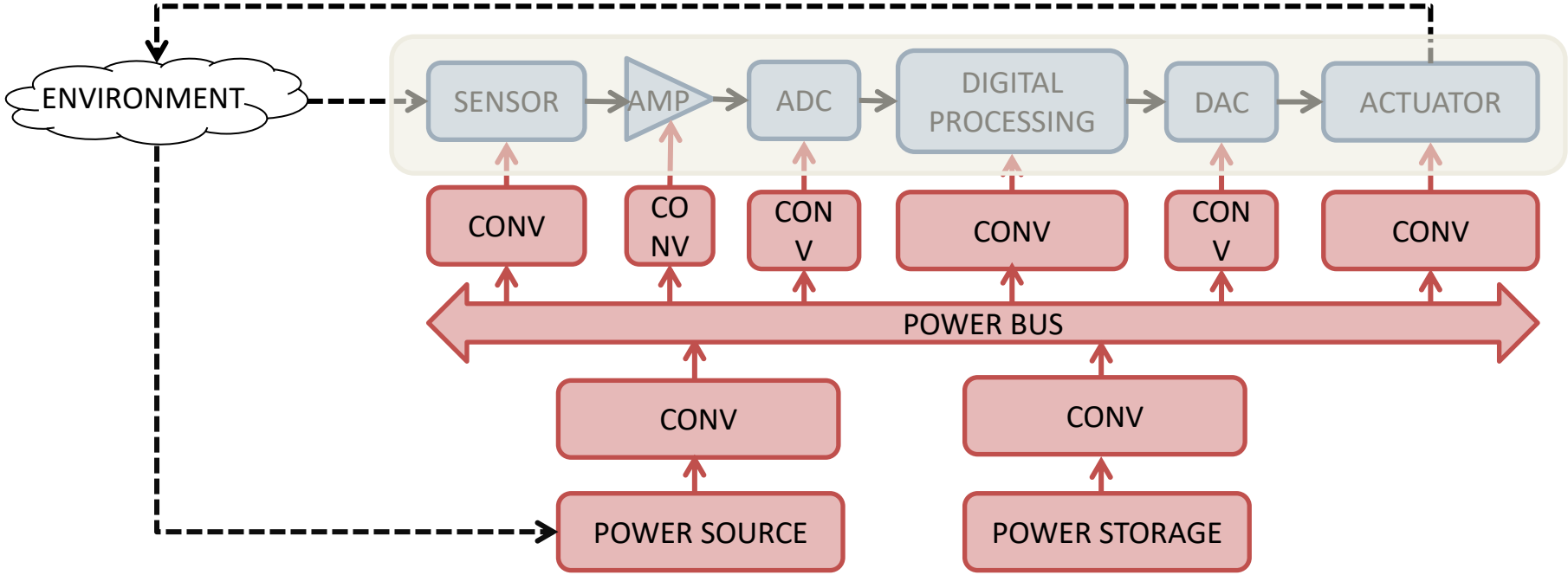
Why CP Energy Systems?

- Embedded systems are inherently **energy-autonomous**
  - Equipped with devices that store power/harvest power
  - This implies significant challenges for simulation and validation
    - What is the «functionality» of a battery?
    - Can not be incorporated in standard functional simulation (e.g., RTL)

- Requires **«native» power simulation**
  - Power as quantity explicitly tracked in simulation



Power Perspective of CPS



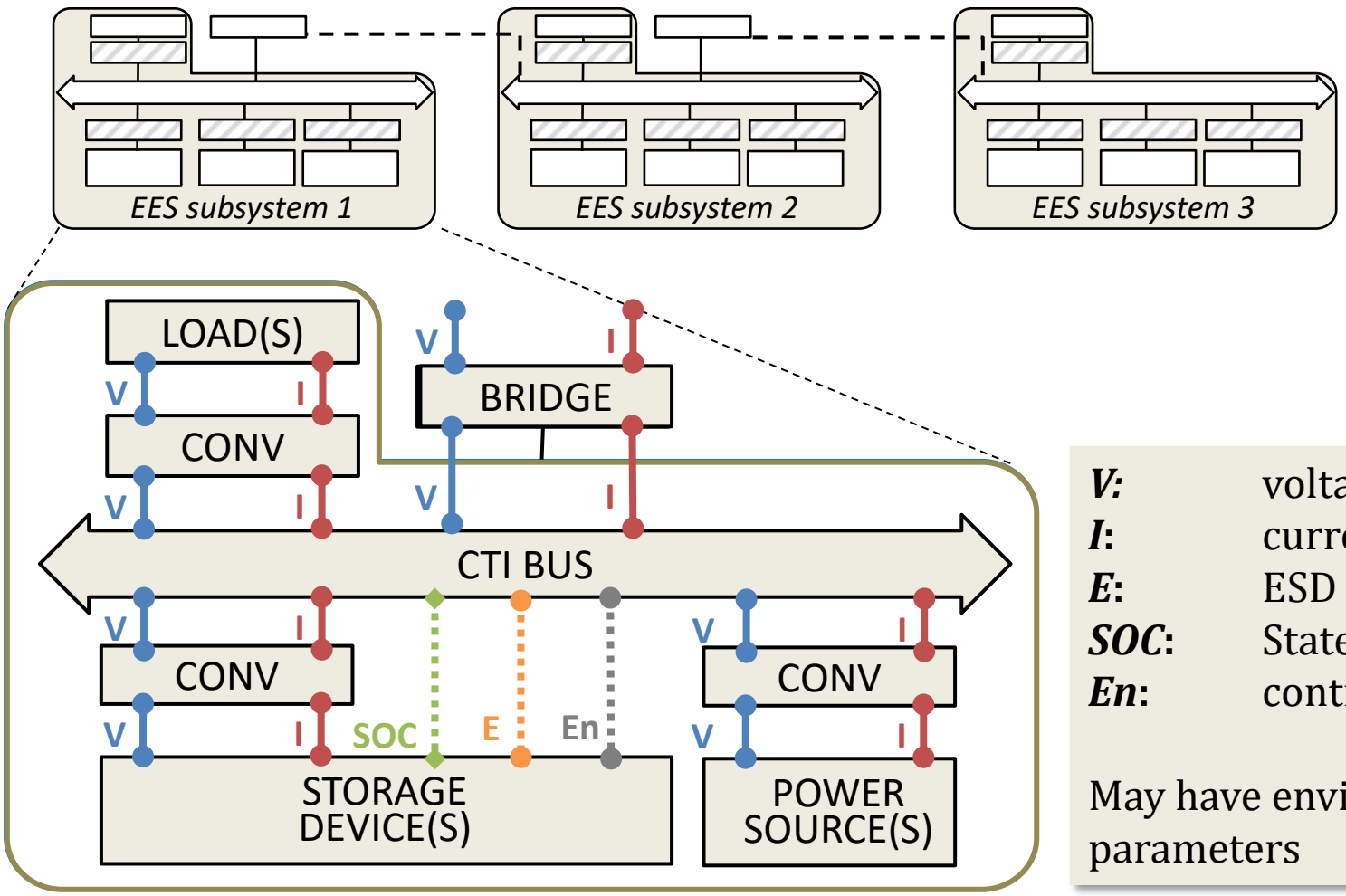
- Functional blocks ≈ their power consumption
  - **Energy systems** as systems where electrical energy is generated, stored, consumed and distributed

Modeling and Simulation of ES

- ES design is strongly manual and dependent on expertise of designers
  - Wish to enhance design automation and optimization

- Design methodology based on formalization and standardization
  1. **Formalization of the architecture**
     - Precise role and interface for each class of components
  2. Adoption of **functional languages** for homogeneous modeling and simulation
  3. Automated **model construction**
     - Models of actual devices are fitted to the defined interface

1. ES Modeling



V:	voltage
I:	current
E:	ESD capacity
SOC:	State of Charge
En:	control signal

May have environmental parameters

2. Modeling and simulation languages

- Interfaces modeled with IP-XACT
  - XML format for describing interfaces
  - Extensions for AMS available
  - Descriptions:
    - Component
      - Interface as ports
    - Design
      - Connection between components

```
<component>
 <vendor> polito </vendor>
 <library> es_lib </library>
 <name> battery </name>
 … <ports>
     <port>
       <name> V </name>
       <vendorExtension>
         <value unit = «volt»
                 prefix = «»> 3.3
         </value>
         <typeName> voltage </typeName>
       </vendorExtension>
     </port>
  …
  <vendorExtension>
    <typeName>storage device
</typeName>
  </vendorExtension>
</component>
```

2. Modeling and simulation languages

- Interfaces constitute a skeleton of the framework
- Need a simulatable language for the implementation
  - Enhance reuse
  - Different levels of abstraction
    - Functional models (state machines, functions,..)
    - Circuit-equivalent models

- SystemC-AMS
  - Several different abstraction levels to cover a wide variety of domains

- Timed Data-Flow (TDF)
  - Data sampled in time
  - Static scheduling
- Electrical Linear Network (ELN)
  - Conservative
  - Predefined linear network primitives
  - AD solver

2. Modeling and simulation languages

- SystemC-AMS
  - **Code generation** from IP-XACT descriptions
    - Instantiate one module per component
    - IP-XACT ports converted to TDF ports
    - Design connections to build hierarchy of modules

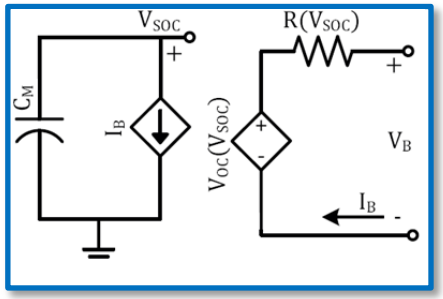
```
SC_MODULE (battery)
{
  sca_tdf::sca_in<double> I;
  sca_tdf::sca_in<bool> En;
  sca_tdf::sca_out<double> V,SOC,E;
};

sc_main(){
  battery * b = new battery("b");
  converter * c = new converter("c");
  sc_signal double current, voltage, …;
  b->I(current);
  b->V(voltage);
  c->I(current);
  c->V(voltage);
  …
}
```

3. Automatic model generation

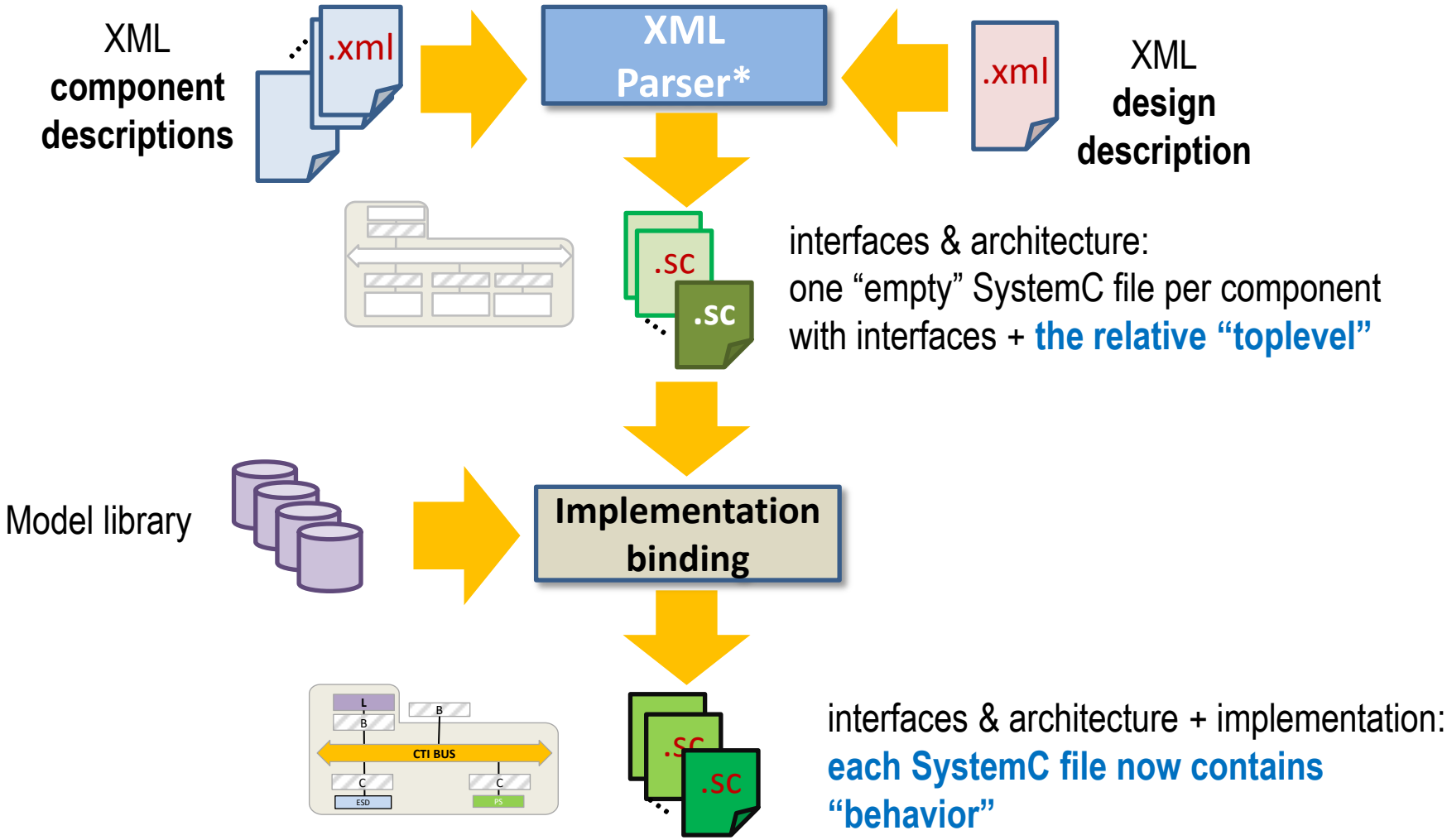
- **Plug and play models** for the various components…
  - Follow the standardized interfaces
  - Model construction
    - Rather than populating a chosen template with data
      - Obtained with measurements or publicly available
    - …the **data available** determines the degree of accuracy of the model

Peukert's equation :

$$LT = C/I^k$$



Overall ES design flow

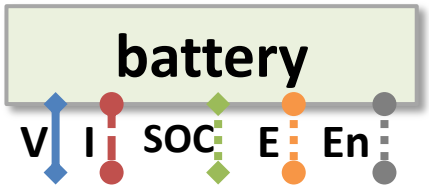


3. Automatic model generation

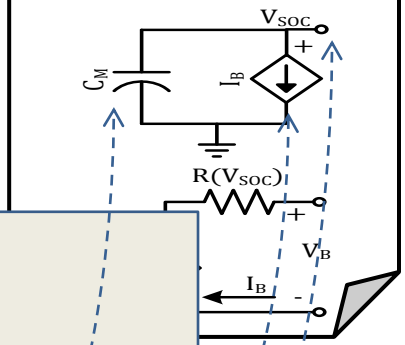
PEUKERT'S LAW

$$C_m = I^n t$$

$$SOC = 1 - \frac{I\,t}{C_m}$$



CIRCUIT MODEL



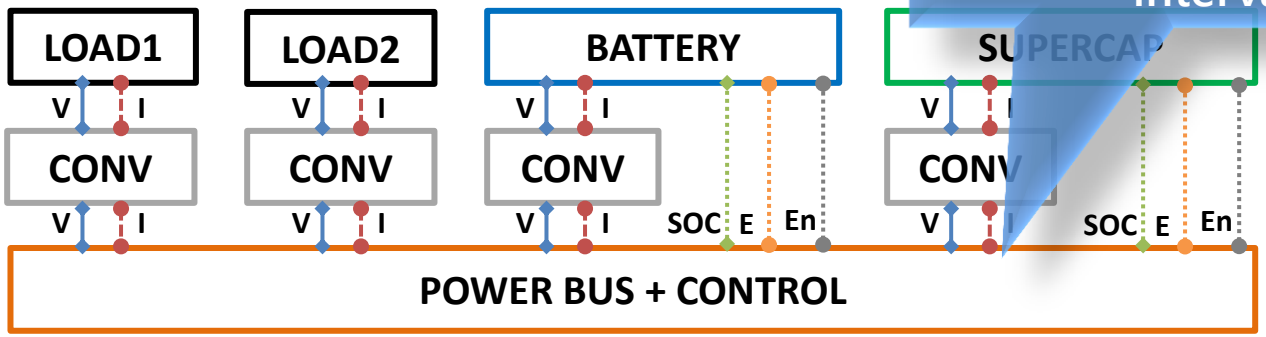

```
SC_MODULE (battery)
{
 sca_tdf::sca_in<double> I;
 sca_tdf::sca_in<bool> En;
 sca_tdf::sca_out<double> V,SOC,E;
 double Vnom, Cm, I0, n;
 ...
};

esd_battery:run(){
 double input, C, soc;

 input = I.read();
 C = Cm * pow((I0/input), n-1);
 soc = 1 - (input/Cm);
 E.write(C);
 SOC.write(soc);
 V.write(Vnom);
}
```

```
SC_MODULE (battery)
{
 sca_tdf::sca_in<double> I;
 sca_tdf::sca_in<bool> En;
 sca_tdf::sca_out<double> V, SOC, E;
 sca_eln::sca_tdf::sca_isource* Ib;
 sca_eln::sca_tdf::sca_vsink* Vsoc;
 sca_eln::sca_c* Cm;
 sca_eln::sca_node n1;
 sca_eln::sca_node_ref gnd;
 ...
 SC_CTOR (battery_lt)
 {
  Ib = new sca_isource("Ib");
  Ib->inp(in);  Ib->p(n1); Ib->n(gnd);
   ...
   ...
 }
};
```

Case study

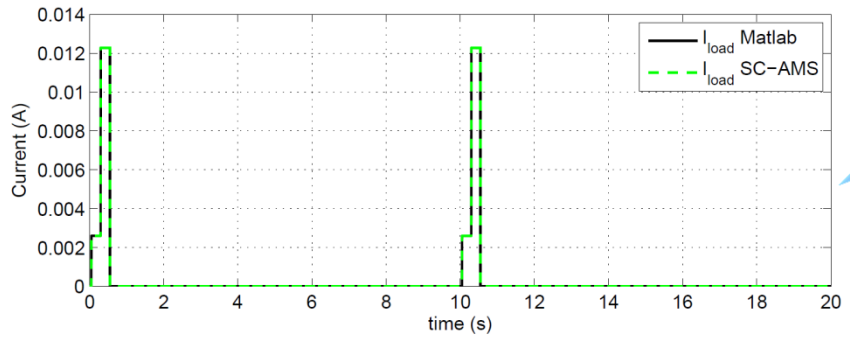
- Loads
  - A STM8L processor and a Wifi Transceiver active every 10s
  - Power profile extracted from functional simulation
- Battery:
  - Lithium thin film battery by ST (C=700Ah, V=3.9V)
- Supercapacitor
  - NessCap supercapacitor with 10 F capacitance
- CTI voltage = 3V (fixed)
- Converter
  - Efficiency function of V_{in}, V_{out}, I_{out}

Charge allocation policy: supercap for high load demand intervals, battery for low load demand intervals



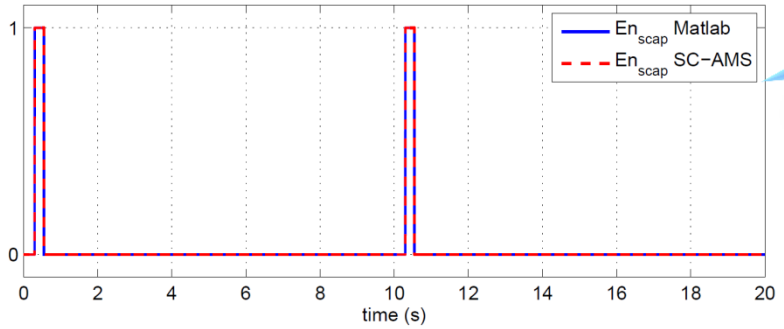
Case study

- Accuracy and speed-up



Average error <1% w.r.t. Simulink

28x speedup w.r.t. Simulink

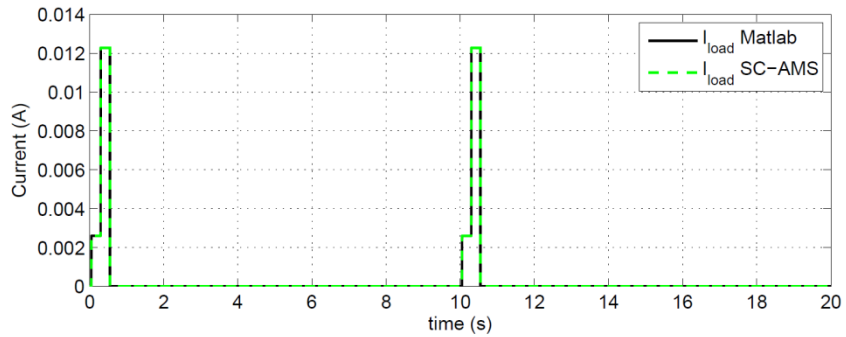


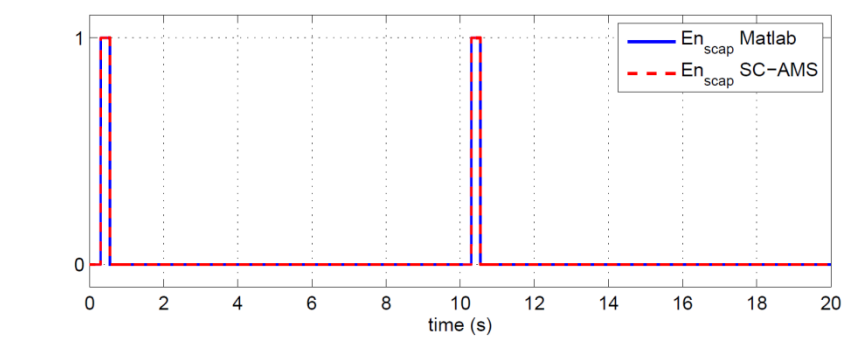
Supercap active on high load demand intervals

Case study

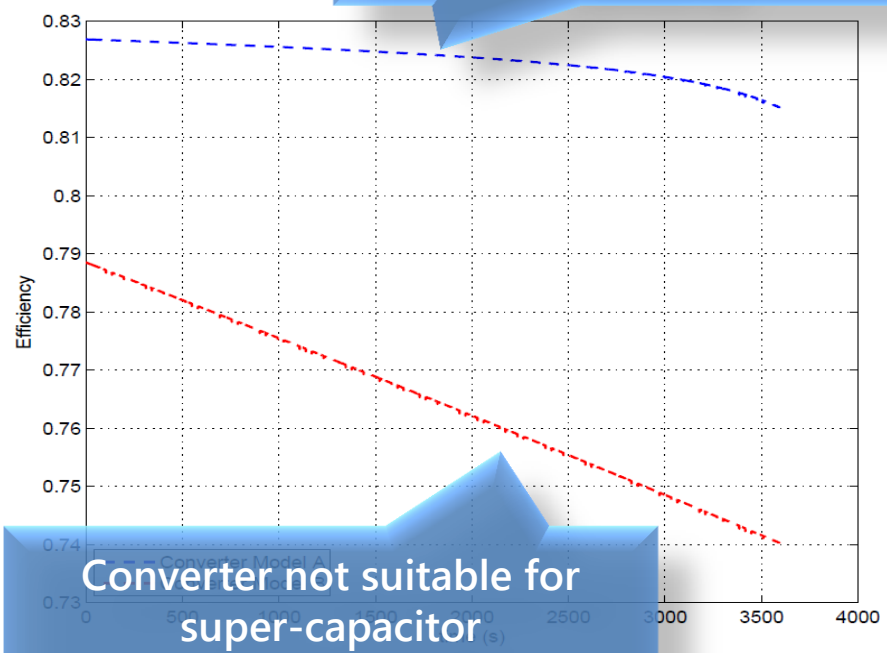
- Accuracy and speed-up

- Design space exploration





Design space exploration allows to find the correct converter



Converter not suitable for super-capacitor

Efficiency decreases too quickly

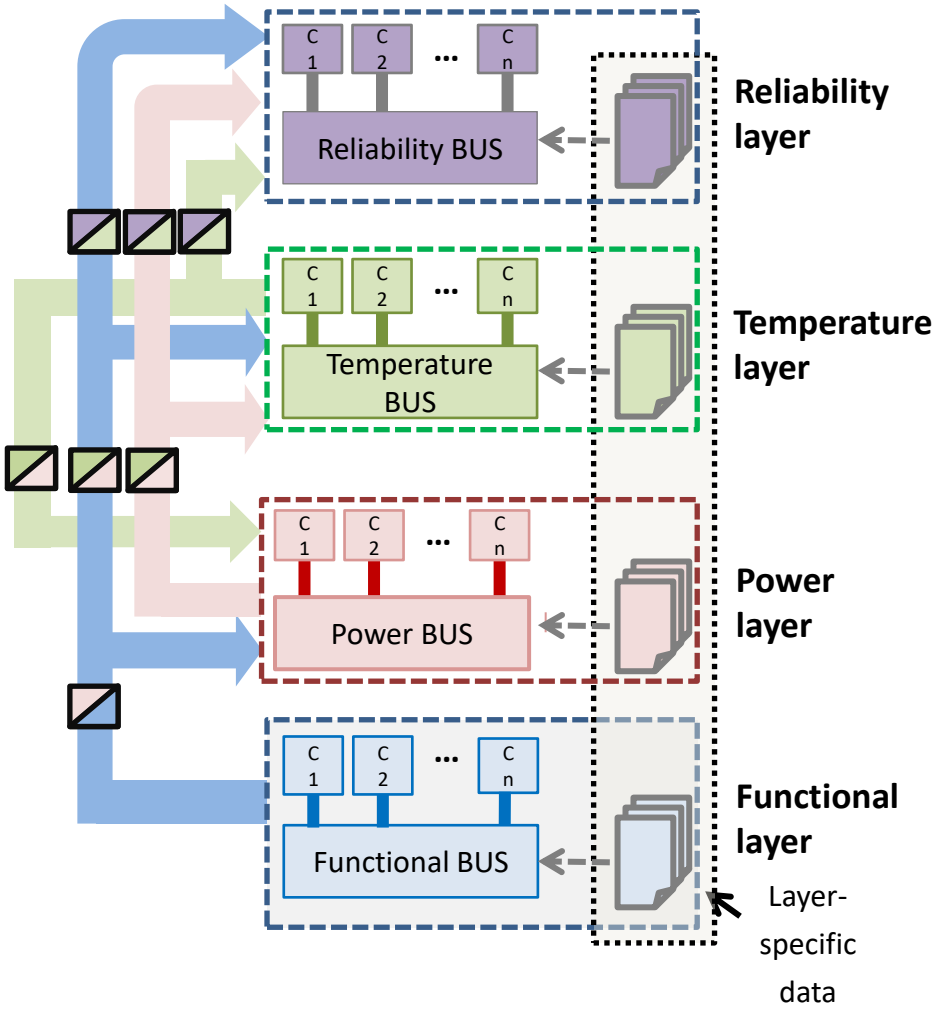
What's next: extra-functional properties

- Power is not the only relevant extra-functional dimension for embedded systems…
  - Other metrics must be considered to ensure their correct operations, including **temperature** and **reliability**

- Modeling in isolation is not a new problem
  - …but how to manage their complex interactions is still an open problem
  - Specific tools and simulators
  - Co-simulation frameworks
    - Miss **mutual effects**, e.g., between power and temperature

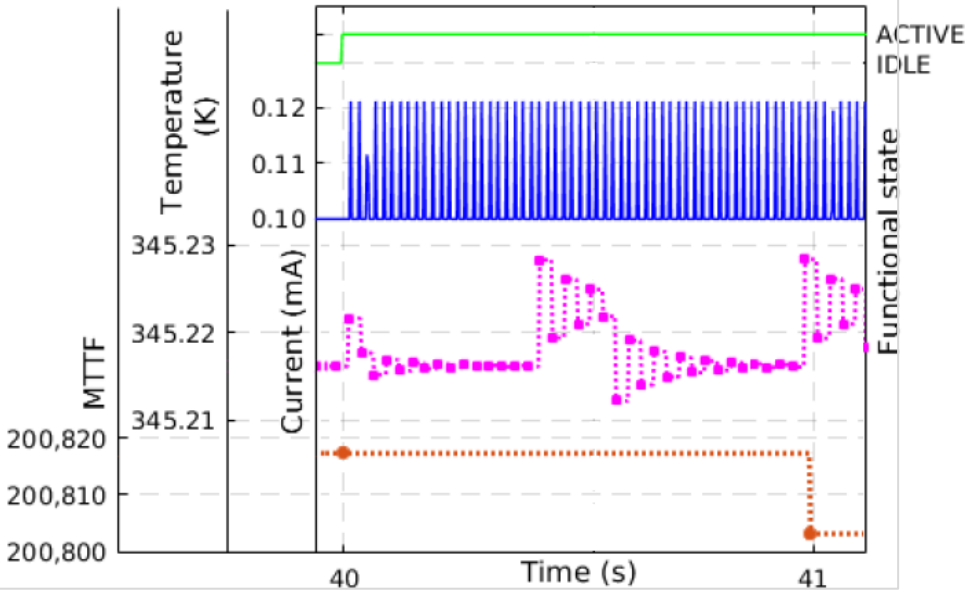
Extra-functional properties simulation

- Proposed solution:
  - **Multi-layer**
    - One layer per property
    - Models energy and information flows
  - **Bus-based template**
    - Property-specific bus conveys and elaborates property-specific information
    - Derive status of the overall system



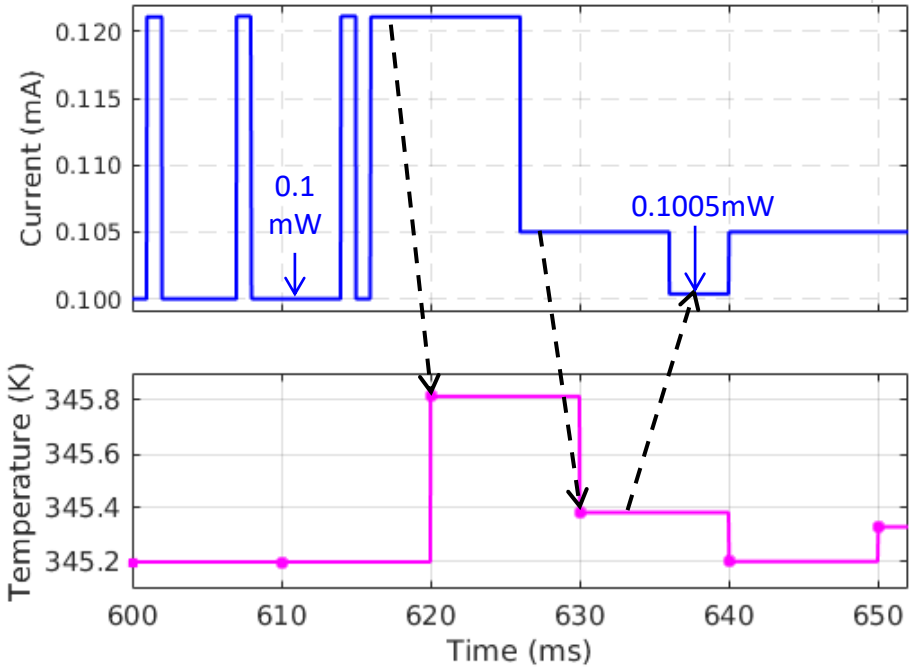
Extra-functional properties simulation

- Enhanced by the adoption of standard functional languages
- IP-XACT
  - Interface standardization
- SystemC-AMS
  - Simulate all properties **in a single simulator run**
    - Different abstraction levels
    - Different timesteps



Extra-functional properties simulation

- Enhanced by the adoption of standard functional languages
- IP-XACT
  - Interface standardization
- SystemC-AMS
  - Simulate all properties **in a single simulator run**
    - Different abstraction levels
    - Different timesteps
  - Reproduce the **mutual impact** of properties
    - Instantaneous reaction



Conclusions

- Presented a unified approach to simulate the flow of **power in embedded systems**
  - Based on functional languages: IP-XACT and SystemC-AMS
- Extended to the simulation of other **non-functional** properties
  - Allows to simulate and validate different aspects in a single simulator run
- Future directions
  - Modeling for non “functional” metrics
  - Automated design/optimization

Thank you!