Model to Hardware: System-level Modeling for Wearable Devices

Daniela Genius and Roselyne Chotin
Sorbonne Université, LIP6, CNRS UMR 7606, Paris, France

Keywords: Embedded Systems, Analog/Mixed Signal Design, Virtual Prototyping.

Abstract: Wearable devices which capture and analyze physiological data in a non-invasive and not restraining manner are increasingly popular, but usually employ specialized hardware. Software is often limited, specific, and expensive. This paper shows how system-level modeling, virtual prototyping and electronic design can interact to help designing low-cost wearable bio-medical devices. We propose a design method which allows validation against actual electronic/mechanical prototypes and present a first case study, a daytime sleepiness detector.

1 INTRODUCTION

Cyber-physical systems (CPS) integrate sensors, computation and control into physical objects, connecting them among each other and to the external world. The medical domain is a typical application area of such systems (Dey et al., 2018). The spectrum of devices is very large, ranging from patches for capturing biomedical signals in home hospitalization (Silva and Tavakoli, 2020) to hospital beds for intensive care.

Similar from the technical point of view, but much less safety critical and subject to lower requirements for certification, a large number of products are proposed in sports and for leisure activities, where the collection of bio-physical data is used to monitor performance and well-being of the human body.

Wearable devices capture and analyze physiological data in a non-invasive and not restraining manner. Smart watches and bracelets are increasingly popular, but there are also shorts and stockings equipped with vibration sensors, clothing with temperature and rain sensors, smart glasses etc. Industrial wearables are almost always based on specialized hardware focused on a (set of) specific functionalities such as heart rate, running speed, temperature etc. From an electronics point of view, they contain ASICs (Application specific integrated circuits) which are produced at low cost in large numbers, and cannot be modified, which raises an obstacle to the exploration of new designs.

In early experimentation, during the process of designing wearable devices for new purposes, designers wish to check out combinations of several sensors, explore the best (cheapest, most energy efficient, most secure) way of communicating between devices and to the external world (data base, physician, ...) without investing in –often non-reusable– material. Also, the software is suject to frequent changes and the designer should ideally be able to test it on an as generic as possible platform, using standard processor cores.

Thus, the idea here is to validate simulations generated from system-level models against actual material implementations. Using an adequate abstraction level, we propose a system-level "construction kit" for designing wearable devices, based on combinations of models for sensors, actuators, communication and control. We extend an existing SysML-based modeling tool, which already contains analog design features. From the models, we automatically generate virtual prototypes for both the digital and analog part, interface them in order to perform full-system co-simulation. We finally validate simulation results against measurements taken from the physical prototypes, built from electronic and textile parts.

We discuss related work in Section 2, basic concepts are introduced in Section 3. Section 4 describes the design space: we evaluate abstractions of sensors wireless networks, and control. Section 5 shows a validation for a wearable bio-monitoring device destined for non-invasive daytime sleepiness detection.

2 RELATED WORK

In this paper, we bring together experience from two domains: SysML-based virtual prototyping and analog/mixed signal hardware design.

2.1 Model-based Design for Cyber-physical Systems

UML/SysML based modeling techniques such as MARTE (Demathieu et al., 2008) have been em-

Genius, D. and Chotin, R.
Model to Hardware: System-level Modeling for Wearable Devices.
DOI: 10.5220/0011005400003119
In *Proceedings of the 10th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2022)*, pages 387-394
ISBN: 978-989-758-550-0; ISSN: 2184-4348

ployed to model cyber-physical systems (Selic and Gérard, 2013).

(Fitzgerald et al., 2013) uses model-based formal methods by integrating discrete-event models of controllers with continuous-time models of their environments. Starting from an initial discrete-event model, approximations of continuous-time behavior are subsequently replaced by couplings to continuous-time models.

Modelica (Fritzson and Engelson, 1998) is an object-oriented modeling language for component-oriented systems containing e.g. mechanical, electrical, electronic and hydraulic components. Yet, since time synchronization is not predefined, the simulation engine must manipulate objects in a symbolic way in order to determine an execution order between components of different MoCs.

Linking simulations with different Models of Computation can be done by using e.g. the Functional Mock-up Interface (Blochwitz et al., 2011), closely related to the Modelica tools.

TTool (Apvrille, 2011), an open-source modeling and verification framework offering extensive formal verification features, has recently been extended with features which allow analog/mixed signal modeling and co-simulation (Genius et al., 2019). The extension is based on timed Synchronous Data Flow (SDF).

Another extension to the SDF formalism, called Polygraph, which includes frequency constraints and adjustable communication rates and ensures synchronization, is shown in (Dubrulle et al., 2019).

2.2 Multi-domain Modeling

The following tools target analog/mixed signal or multi-domain design and co-simulation. Models of Computation (MoCs) are an important notion, as they basically differ depending on whether modeling digital or analog behavior is intended. Thus, the question of synchronization between two or more parts of a simulation poses itself.

Ptolemy II (Ptolemy.org, 2014) is based upon the data-flow model. It addresses digital/analog systems by defining several sub domains, however leaving time synchronization to the designer.

Metro II (Davare et al., 2007) is based on hierarchical high level models. So-called *Adapters* are used for data synchronization between components belonging to different MoC, yet the model designer has to implement time synchronization; a common simulation kernel handles the entire execution.

Modelica (Fritzson and Engelson, 1998) is an object-oriented modeling language for cyber-physical systems. Time synchronization is not predefined and the simulation engine must manipulate objects containing sets of equations in a symbolic way in order to determine an execution order between components of different MoCs.

Linking simulations with different Models of Computation can also be done by using the Functional Mock-up Interface (Blochwitz et al., 2011), which is closely related to the Modelica tools.

3 BASICS OF SystemC-AMS MODELING

TTool-AMS (Genius et al., 2018), which we use in our work, is an extension of TTool which in particular allows the generation of virtual prototype from a SysML-like representation. It generates a SystemC specification for the digital part together with software and operating system for full-system simulation, as well as a SystemC-AMS specification from the analog part of the virtual prototype, and runs both parts together in a co-simulation.

SystemC (IEEE, 2011) is a collection of C++ classes, while SystemC-AMS (Vachoux et al., 2003) is an extension based on SystemC (IEEE, 2011), providing analog and mixed-signal (AMS) features. A proof-of-concept simulator has been developed (Einwich, 2016).

In SystemC-AMS, digital components are described with a Discrete Event (DE) MoC, while analog components are described with the Timed Data Flow (TDF) MoC, based on the timeless Synchronous Data Flow semantics (Lee and Messerschmitt, 1987).

A TDF *module* has input and output ports and a *processing* function, which describes the module's functionality. TDF *clusters* are composed of several TDF modules which are connected by *signals*. TDF is the most abstract MoC proposed in SystemC-AMS to describe analog components, as continuous functions are sampled at regular intervals.

4 METHOD

Figure 1 shows the design flow and validation. Starting from a functional model of the application, after partitioning, software (left hand side) and hardware (right hand side) are designed in the SysML-like representation proposed in (Pedroza et al., 2011) for software; hardware is also described in a SysML-like style (Genius and Apvrille, 2016).

A so-called *deployment diagram*, shown in the center of the figure, describes the mapping of soft-

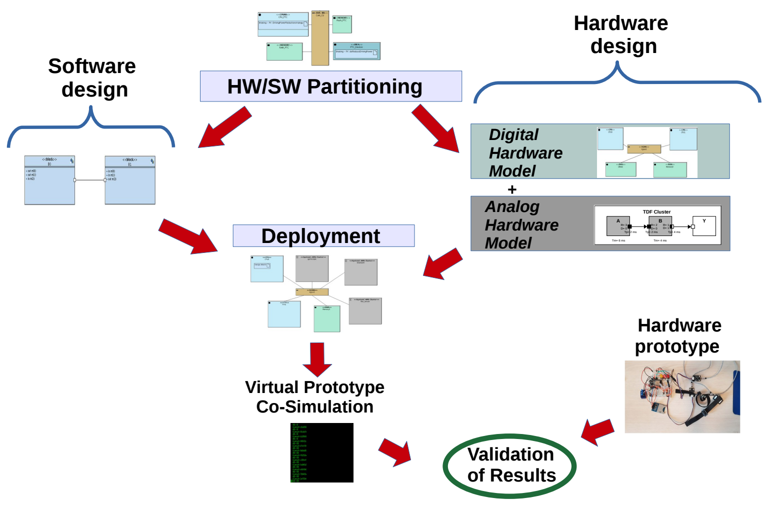


Figure 1: Integration in the Design Flow.

ware onto the hardware. From this diagram, the tool generates a platform containing a top cell instantiating hardware components from the SoCLib library (SoCLib consortium, 2003), a lightweight operating system and software in the form of Posix threads.

The analog part is generated in the form of SystemC-AMS clusters (Accellera Systems Initiative, 2010) connected to the digital platform by specific interfaces. This reflects the idea that analog parts are essentially sensors, whose task it is to supply information to the digital part, charged with control and running the software. Both parts are co-simulated, respecting causality between MoCs (Porto et al., 2021), as shown on the lower center of the figure. The lower right of the figure shows the actual hardware prototype implementation which is used to validate the simulations on SystemC/SystemC-AMS level.

In order to reach our aim, a toolkit for fast design of lightweight biomedical devices, required four contributions that extend the existing toolkit.

The first new contribution is an abstraction of sensors: after analyzing common features of usual analog modules, a library of SysML modules emerges, containing parameterizable models not only of sensors but also of filters, Digital/Analog and Analog/Digital converters (DAC and ADC, respectively), and other typical analog and mixed-signal modules.

The second contribution is the modeling of several means of wireless communication in TTool-AMS.

The third contribution is the integration of (limited) software, destined to run on a microcontroller but tested beforehand on a general purpose processor platform, to obtain an as-realistic-as-possible full-system co-simulation.

The final contribution is the validation against the actual electronic prototype, consisting of microcontrollers, eventually microprocessors and FPGA (Field Programmable Gate Arrays) for the digital part, network components (WIFI, Bluetooth etc.), and sensors.

The advantage of using SystemC-AMS in the virtual prototype is that its interaction with the DE MoC is well understood (Porto et al., 2021). Another advantage of using SystemC-AMS is that models can be successively refined: the *processing* function of a TDF module could thus initially only contain the right number and type of input and output ports, which are only read on the input side and written on the output side of the module, then adding a table describing the output of the sensor for the respective input, finally using mathematical functions and transformations (Laplace transfer functions are available in SystemC-AMS).

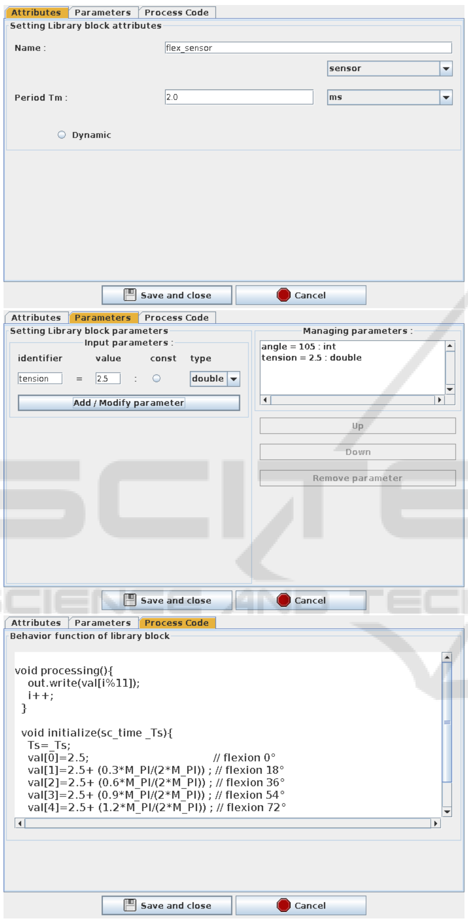
4.1 Sensor Abstraction

Our first aim is to find an abstract representation for sensors. Biomedical sensors are destined to capture signals emitted by the human body. Figure 3 shows a cluster containing a sensor –here a flex sensor– with ADC and clock.

All in all we determine the common characteristics of five sensor types. Apart from alimentation, they have one single output pin, providing analog data in most cases, digital via an ADC in some. Data output from the TDF cluster is a timed, synchronous flow of floating point (integer, bit vector) data corresponding to the data types available for TDF ports.

The flex sensor or *fleximeter* captures the level of bending by a variable resistor disposed throughout its length. *Accelerometers* and *gyroscopes* are very common in everyday life (joystick, smartphones, etc.)

and, as they name indicates, capture acceleration and orientation wrt. a referential. An electromyogram measures the electrical activity of nerves and muscle contraction. Non-invasive versions, the only ones of interest in our context, use electrodes glued to the skin and usually contain signal amplifiers. The pulse sensor, based on an infrared LED and a phototransistor measures variations of the blood flow. All sensors are modeled as look up tables in the *processing* function, as can be seen in the lowest part of Figure 2.



Attributes | Parameters | Process Code

Setting Library block attributes

Name : flex_sensor

sensor

Period Tm : 2.0 ms

Dynamic

Save and close | Cancel

Attributes | Parameters | Process Code

Setting Library block parameters

Input parameters :

identifier		value		const	type
tension	=	2.5	:		double

Add / Modify parameter

Managing parameters :

```
angle = 105 : int
tension = 2.5 : double
```

Up | Down | Remove parameter

Save and close | Cancel

Attributes | Parameters | Process Code

Behavior function of library block

```
void processing(){
  out.write(val[i%11]);
  i++;
 }

 void initialize(sc_time _Ts){
  Ts=_Ts;
  val[0]=2.5;                              // flexion 0°
  val[1]=2.5+ (0.3*M_PI/(2*M_PI)) ; // flexion 18°
  val[2]=2.5+ (0.6*M_PI/(2*M_PI)) ; // flexion 36°
  val[3]=2.5+ (0.9*M_PI/(2*M_PI)) ; // flexion 54°
  val[4]=2.5+ (1.2*M_PI/(2*M_PI)) ; // flexion 72°
```

Save and close | Cancel

Figure 2: Abstraction of flex sensor as TDF block: (top) parameters, (middle) attributes, (bottom) processing function.

4.2 Wireless Network Abstraction

Currently, we provide models of WIFI for mid/short range (body to server) and bluetooth for very short range (close to the body) communication. Details are out of the scope of this paper, there are fourteen TDF blocks forming a quite detailed, yet still abstracted model. The Wifi module contains nearly twice as many (twenty-six) TDF blocks.

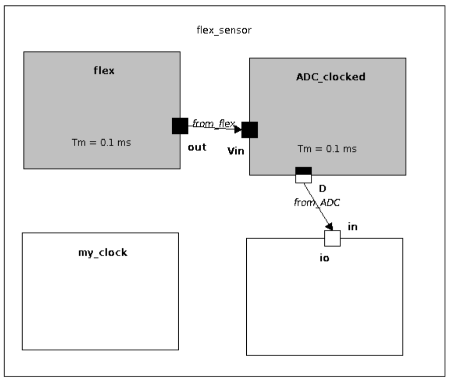


Figure 3: Model of a generic sensor.

4.3 Code Generation and Co-simulation

Software is abstracted by a SysML-like representation of communicating blocks with an underlying semantics of communicating Finite State Machines (Pedroza et al., 2011). Blocks can contain additional *entry code* to capture behavior that cannot be expressed in that representation, in particular the treatment of (sampled) floating point values typically arriving from the analog domain. A virtual cycle-bit-accurate prototype for full-system simulation is generated automatically (top cell, operating system and deployed software).

Analog clusters are instantiated together with the digital components in a common top cell. This top cell is generated by TTool-AMS, from an extended Deployment Diagram (Genius et al., 2019).

During co-simulation, the SystemC event-based simulation kernel controls the AMS simulation. The difficulty here is synchronization; errors are usually detected late, often only during simulation, causing an emergency stop. Recent work proposes a method to ensure correct synchronization of the DE and TDF MoCs of SystemC-AMS which is directly applicable at System Design level (Porto et al., 2021). Synchronization is ensured by detecting causality problems at converter ports and proposing adequate modification to the concerned TDF modules. General Purpose I/O (GPIO) serve interfaces between the analog clusters and the central interconnect of the digital platform.

4.4 Validation

We validate the resulting traces of the co-simulation against an actual implementation using electronic and mechanical parts, which is typically accomplished by master's students with off-the-shelf electronics components (Celik, 2021; Broux et al., 2019) and hand-crafted or off-the shelf mechanical/textile components. The sensors, whose parameters are identical to

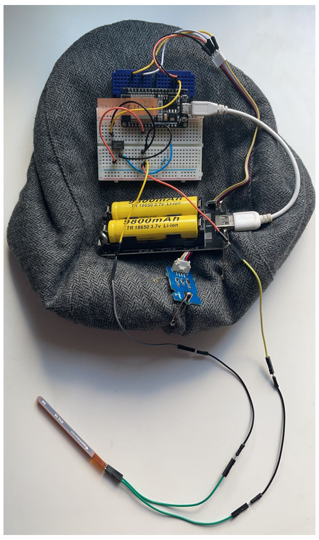


Figure 4: Wearable daytime sleepiness detector (prototype).

those used in the model, are connected to one or several microcontrollers, which communicate with each other and with a server running a small data base.

Figure 4 shows a prototype resulting from a recent study, made of textile parts, in which electronics and mechanical devices, wireless transmitter/receiver and power supply can be hidden (Celik, 2021).

5 CASE STUDY: WEARABLE DAYTIME SLEEPINESS DETECTOR

Daytime sleepiness, possible cause of traffic accidents when occurring through driving, is caused by problems during nighttime sleep (snoring, ...). It is very current and also concerns a population that, even if often older and more overweight, is still active in professional life.

In the domain of sleep disorder detection, usual polysomnography devices are most often very expensive and require overnight or several days' hospitalization (Douglas et al., 1992). On the initiative of the ICAN (cardiometabolics and nutrition) institute, part of the Pitié-Salpêtrière University Hospital complex, the issue was raised to develop a pre-diagnostic device, to be worn during at least two or three days, collected data then analyzed by physicians off-line.

Such a device requires a certain, if limited, battery autonomy. It should also be easy to use (ergonomic user interface, out of the scope of this paper), lightweight and ideally insconspicious (minimization) and, if wireless data transmission is chosen, satisfy security requirements (out of the scope of the current paper but supported by TTool).

A wearable daytime sleepiness detection equipment stemming from several master student's projects (Broux et al., 2019; Celik, 2021) serves as our case study. It was decided beforehand that, in order to be inconspicious, the device consists of two parts hidden in the clothing: a headgear and a bracelet.

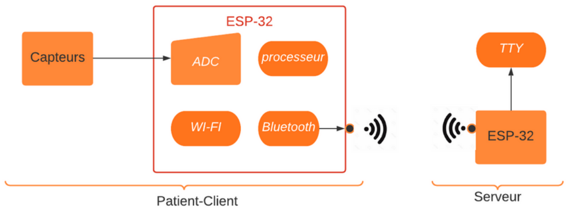


Figure 5: Overview of the client-server architecture.

We wish to evaluate combinations of different sensors. The following sensors have been employed in our experimentation:

- A flex sensor FS7954 to monitor the bending angle of the neck. It measures 7.4 cm and requires 0.5 Watts and 3.3 V thus 150 mA.
- A gyroscope/accelerometer combination Grove 101020584 which detects sudden movements (head falling brusquely forward at onset of sleep). The tension is 3.3V for the aaccelerometer which consumes 150μA and the gyroscope consumes 5 mA. This sensor has an integrated ADC.
- An EMG-Sensor SEN0240 (electromyograph) to monitor muscle tension supplied by 3.3V and minimum 20mA.
- A pulse sensor KY039HS used to determine the heart rate. Its received signal varies in function of the blood flow present in the finger, i.e. the contraction and decontraction of the heart. Each amplitude variation between two extreme values corresponds to a heart beat.

Only the last two are exclusively used in biomedical sensing. With exception of the last one, all sensors are located in the upper part of the device, in proximity of the neck. The generic sensor discussed in 4 and shown in Figure 3 can be used as part of this device, parameterized to represent each of the five sensors. The combination of sensors allows to detect indications for daytime sleepiness: for example whether a movement of the head is involuntary, for example when accompanied by a slowdown of the heart rate.

Three ESP-32 micro-controllers implement control and communication. Each micro-controller is connected to two sensors (flex sensor and gyroscope/accelerometer in the neck area, EMG and pulse sensor on arm/hand).

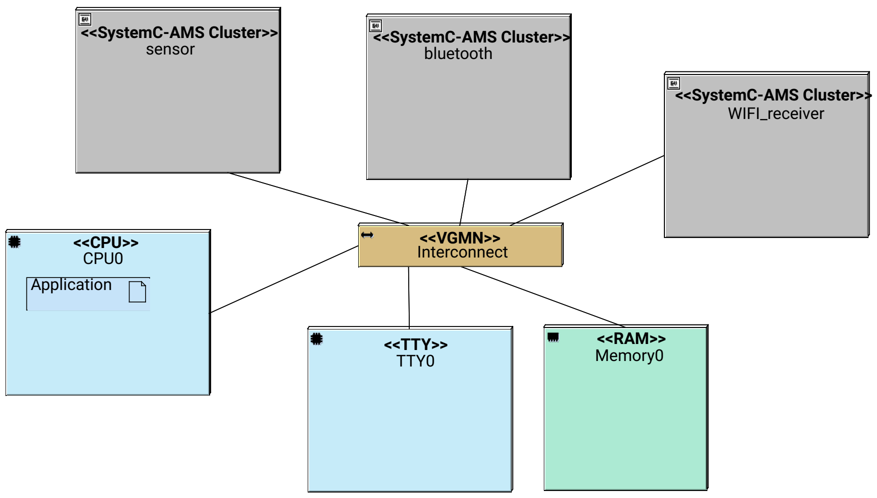


Figure 6: TTool Deployment Diagram for the wearable daytime sleepiness detector.

Bluetooth is used for the communication between the different parts of the device fixed to the body (short distance), finally WIFI for communicating with the server (mid/short distance). We represent WIFI and Bluetooth, ensuring the communication, modeled as TDF clusters.

The server and software (among others a simple database SQLite3) are simulated by a SoCLib monoprocessor platform. Figure 5 gives an overview: each sensor pair worn by the patient is equipped with bluetooth and WIFI and an optinal ADC. Figure 6 shows the deployment of software (*Application* task) on the processor, and the three analog clusters attached.

The user interface for co-simulation is currently very basic, essentially textual output on a terminal. Traces are stored in a text file, and compared to measurements taken from the electronic prototype by a Python script. Figure 7 shows some results for one of the sensors, a flex sensor: output tension in mV (Y axis) in function of time in seconds (X axis): at 150 seconds, a brusque movement is detected.

Figure 8 is a screenshot of the co-simulation: the upper part shows the instantiation of the top cell components of the SystemC part of the platform (processor with cache, TTY, memory segments, interconnect, three GPIO interfaces), the lower part the SystemC-AMS simulation (three clusters of different complexity and number of modules: WIFI, by far the most complex with more than 20 TDF modules, an example sensor, and bluetooth).

The current prototype of the sleepiness detection device, already shown earlier in Figure 4, is integrated in a cap and a bracelet. An alternative version, less targeted towards mid-age male patients, would integrate the sensors monitoring the neck (flex sensor, gyroscope/accelerometer) into a scarf. The most expensive, and most heavy, part, are the batteries, which must maintain alimentation during two or three days: we require 2 batteries providing 9.600mAh. Two possibilities are foreseen for data transfer to the server: either by SIM card or by WIFI, and fed into the database. The cost of the prototype is less than 100$.

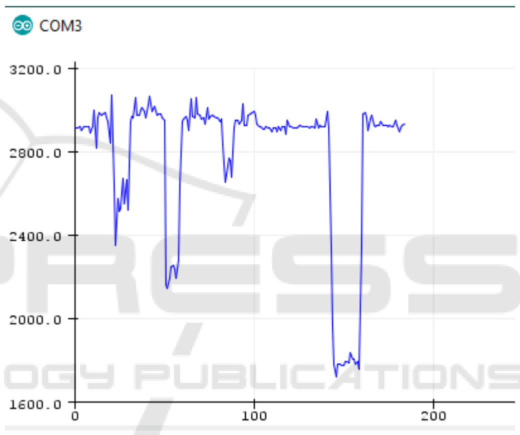
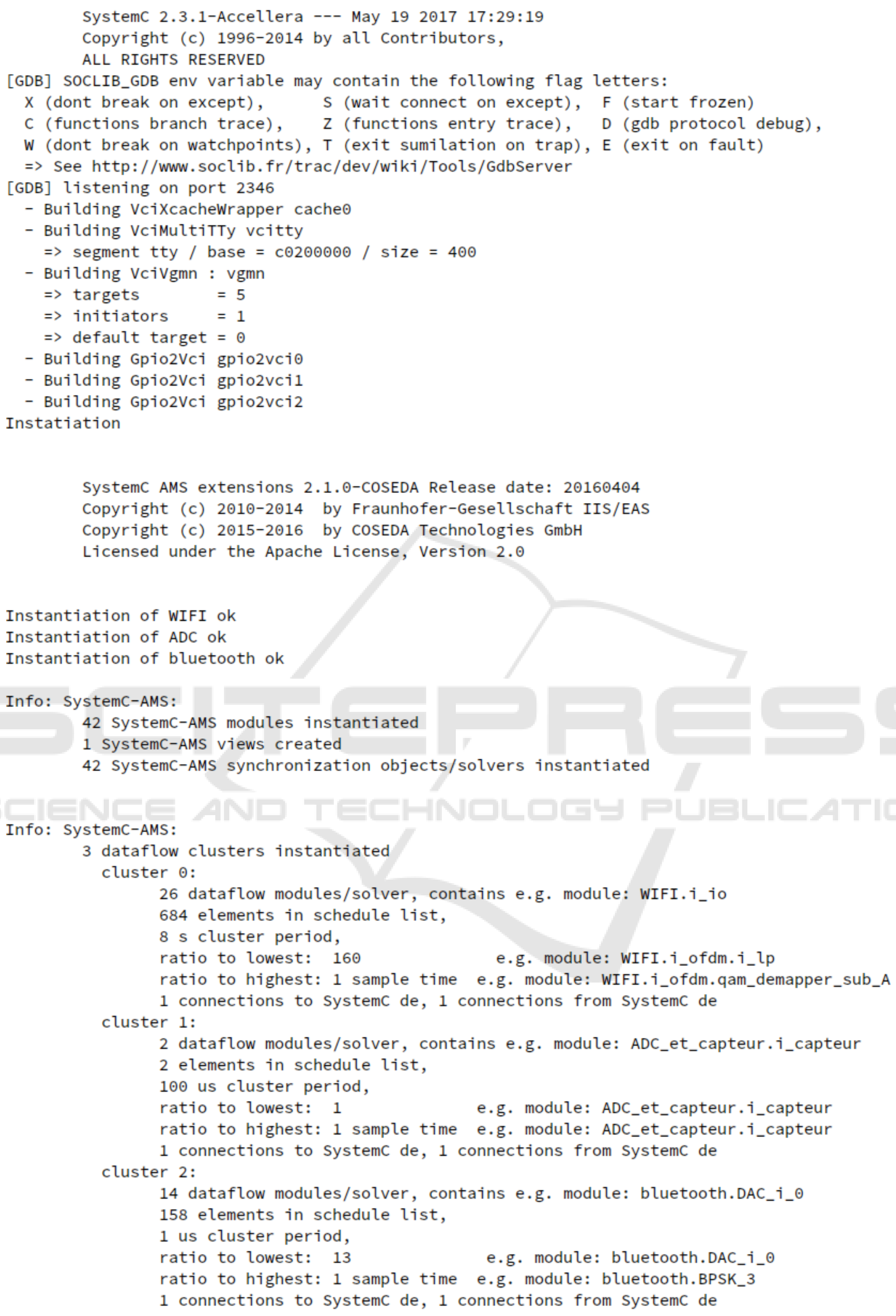


Figure 7: First results for the flex sensor.

6 FUTURE WORK

The current paper presents work in progress. Co-simulation results shown here stem from a device including a single sensor cluster, while the current electronic prototype already contains up to three clusters carrying five sensors. It is possible to refine measurements by adding more sensors. Nevertheless, the simulated results are close to those obtained with the actual implementation, which is clearly encouraging.

The students tested the device by wearing it themselves during several hours; due to longer-term legal



```
        SystemC 2.3.1-Accellera --- May 19 2017 17:29:19
        Copyright (c) 1996-2014 by all Contributors,
        ALL RIGHTS RESERVED
[GDB] SOCLIB_GDB env variable may contain the following flag letters:
  X (dont break on except),      S (wait connect on except),  F (start frozen)
  C (functions branch trace),    Z (functions entry trace),   D (gdb protocol debug),
  W (dont break on watchpoints), T (exit sumilation on trap), E (exit on fault)
  => See http://www.soclib.fr/trac/dev/wiki/Tools/GdbServer
[GDB] listening on port 2346
  - Building VciXcacheWrapper cache0
  - Building VciMultiTTy vcitty
    => segment tty / base = c0200000 / size = 400
  - Building VciVgmn : vgmn
    => targets        = 5
    => initiators     = 1
    => default target = 0
  - Building Gpio2Vci gpio2vci0
  - Building Gpio2Vci gpio2vci1
  - Building Gpio2Vci gpio2vci2
Instatiation


        SystemC AMS extensions 2.1.0-COSEDA Release date: 20160404
        Copyright (c) 2010-2014  by Fraunhofer-Gesellschaft IIS/EAS
        Copyright (c) 2015-2016  by COSEDA Technologies GmbH
        Licensed under the Apache License, Version 2.0


Instantiation of WIFI ok
Instantiation of ADC ok
Instantiation of bluetooth ok

Info: SystemC-AMS:
        42 SystemC-AMS modules instantiated
        1 SystemC-AMS views created
        42 SystemC-AMS synchronization objects/solvers instantiated

Info: SystemC-AMS:
        3 dataflow clusters instantiated
          cluster 0:
                26 dataflow modules/solver, contains e.g. module: WIFI.i_io
                684 elements in schedule list,
                8 s cluster period,
                ratio to lowest:  160              e.g. module: WIFI.i_ofdm.i_lp
                ratio to highest: 1 sample time  e.g. module: WIFI.i_ofdm.qam_demapper_sub_A
                1 connections to SystemC de, 1 connections from SystemC de
          cluster 1:
                2 dataflow modules/solver, contains e.g. module: ADC_et_capteur.i_capteur
                2 elements in schedule list,
                100 us cluster period,
                ratio to lowest:  1              e.g. module: ADC_et_capteur.i_capteur
                ratio to highest: 1 sample time  e.g. module: ADC_et_capteur.i_capteur
                1 connections to SystemC de, 1 connections from SystemC de
          cluster 2:
                14 dataflow modules/solver, contains e.g. module: bluetooth.DAC_i_0
                158 elements in schedule list,
                1 us cluster period,
                ratio to lowest:  13              e.g. module: bluetooth.DAC_i_0
                ratio to highest: 1 sample time  e.g. module: bluetooth.BPSK_3
                1 connections to SystemC de, 1 connections from SystemC de
```

Figure 8: Co-simulation.

questions and the sanitary situation, currently larger-scale tests were not yet possible.

Currently the AMS part is restricted to TDF, which is a rather coarse abstraction of analog behavior. We plan to refine analog models by integrating ELN (Electrical Linear Network), another MoC available in SystemC-AMS.

Most crucial is the optimization of battery usage; power consumption estimation should be added to the tool. Finally, even if the method was developed in a specific domain, it might be generalized to other IoT devices equipped with multiple sensors.

REFERENCES

Accellera Systems Initiative (2010). *SystemC AMS extensions Users Guide, Version 1.0*.

Apvrille, L. (2011). *TTool, an open-source toolkit for the modeling and verification of embedded systems*.

Blochwitz, T. et al. (2011). The functional mockup interface for tool independent exchange of simulation models. In *8th Int. Modelica Conference, Dresden, Germany*, pages 105–114.

Broux, P.-E., Lavalade, P., Sylla, S., and Zhuang, K. (2019). Somnolence diurne sodi.

Celik, H. O. (2021). Minimization, deployment and evaluation with ai methods of a portable sleepyness detection device, master's thesis (French), Sorbonne Université.

Davare, A., Densmore, D., Meyerowitz, T., Pinto, A., Sangiovanni-Vincentelli, A., Yang, G., Zeng, H., and Zhu, Q. (2007). A next-generation design framework for platform-based design. In *DVCon*, volume 152.

Demathieu, S., Thomas, F., André, C., Gérard, S., and Terrier, F. (2008). First experiments using the uml profile for marte. pages 50–57. IEEE.

Dey, N., Ashour, A. S., Shi, F., Fong, S. J., and Tavares, J. M. R. (2018). Medical cyber-physical systems: A survey. *J. of medical syst.*, 42(4):74.

Douglas, N. J., Thomas, S., and Jan, M. A. (1992). Clinical value of polysomnography. *The Lancet*, 339(8789):347–350.

Dubrulle, P., Gaston, C., Kosmatov, N., Lapitre, A., and Louise, S. (2019). A data flow model with frequency arithmetic. In *International Conference on Fundamental Approaches to Software Engineering*, pages 369–385. Springer, Cham.

Einwich, K. (2016). *SystemC AMS PoC2.1 Library, COSEDA, Dresden*.

Fitzgerald, J. S., Larsen, P. G., Pierce, K. G., and Verhoef, M. H. G. (2013). A formal approach to collaborative modelling and co-simulation for embedded systems. *Mathematical Structures in Computer Science*, 23(4):726–750.

Fritzson, P. and Engelson, V. (1998). Modelica—a unified object-oriented language for system modeling and simulation. In *European Conference on Object-Oriented Programming*, pages 67–90. Springer.

Genius, D. and Apvrille, L. (2016). Virtual yet precise prototyping: An automotive case study. In *ERTSS'2016*, Toulouse.

Genius, D., Cortés Porto, R., Apvrille, L., and Pêcheux, F. (2019). A tool for high-level modeling of analog/mixed signal embedded systems. In *MODELSWARD*.

Genius, D., Cortés Porto, R., and Apvrille, L. (2018). *TTool AMS*.

IEEE (2011). *SystemC*. IEEE Standard 1666-2011.

Lee, E. A. and Messerschmitt, D. G. (1987). Synchronous data flow. *Proceedings of the IEEE*, 75(9):1235–1245.

Pedroza, G., Knorreck, D., and Apvrille, L. (2011). AVATAR: A SysML environment for the formal verification of safety and security properties. In *11th IEEE Conference on Distributed Systems and New Technologies*, Paris, France.

Porto, R. C., Genius, D., and Apvrille, L. (2021). Handling causality and schedulability when designing and prototyping cyber-physical systems. *Software and Systems Modeling*, pages 1–17.

Ptolemy.org, editor (2014). *System Design, Modeling, and Simulation using Ptolemy II*.

Selic, B. and Gérard, S. (2013). *Modeling and Analysis of Real-Time and Embedded Systems with UML and MARTE: Developing Cyber-Physical Systems*. Elsevier.

Silva, A. F. and Tavakoli, M. (2020). Domiciliary hospitalization through wearable biomonitoring patches: Recent advances, technical challenges, and the relation to covid-19. *Sensors*, 20(23):6835.

SocLib consortium (2003). *The SoCLib project: An Integrated System-on-Chip Modelling and Simulation Platform, www.soclib.fr*.

Vachoux, A., Grimm, C., and Einwich, K. (2003). Analog and mixed signal modelling with SystemC-AMS. In *ISCAS (3)*, pages 914–917. IEEE.