Hierarchical Design of Cyber-Physical Systems

Keywords: Cyber Physical Systems, Analog/Mixed Signal Design, Virtual Prototyping

Abstract: Cyber-physical systems are based upon analog / digital hardware and software components. The splitting into functionalities and interaction between analog and digital parts should be considered as early as possible in the design phase, relying on formal verification or simulation. While many papers pretend to propose a modeling environment supporting them, only a few of them really address the different Models of Computation of these systems because they strongly differ. The paper explains how to generate a virtual prototype of the analog and mixed-signal parts of CPS directly from a SysML model featuring whole parts of CPS, thus reconciling near-circuit precision with more abstract analog and digital models.

1 Introduction

Cyber-physical systems span three domains: analog, digital and physical. Most often, off-the-shelf analog and digital components can be used for their design. Yet, when special requirements have to be met (such as: low-power, very small size, specific application, etc.) such off-the-shelf components may be too costly or not available, thus advocating for from-scratch designs, sometimes both for analog and digital parts.

For such a custom design, the splitting of functionalities between analog and digital parts, and their interaction, is of prime importance, and should therefore be done as early as possible. In early design phases, simulation or formal verification helps taking decisions. But, in the case of mixed signal design, i.e., designs with analog and digital parts, the Models of Computation (MoC) of these two aspects strongly differ. Moreover, they are commonly designed at different abstraction levels, depending on the design patterns already available. Last, because of the significant semantic difference between MoCs, similar models or unique tools can not be used to study the same system.

Thus, an interesting contribution is to offer a multi-level virtual prototyping and simulation environment that can be executed from SysML models, featuring all necessary elements to capture analog and digital aspects. Abstraction levels could range from a near-circuit precision to more abstract analog and digital models (e.g., transactional models).

The paper explains how to use SysML diagrams to efficiently capture digital and analog parts of CPS –as well as the interactions between these parts– and how to generate a virtual prototype of the analog and mixed-signal parts directly from these SysML models. Just like many SysML tools, our tool already offers verification capabilities for the digital parts, thus this part is not addressed in the paper.

Related work is discussed in Section 2, then basic concepts are introduced in Section 3. Section 4 describes our contribution (modeling and simulation). Section 5 shows a complex case study, a scalable analog-to-digital converter.

2 Related Work

The contribution of the paper is at the intersection between two research domains: model-based design for cyber-physical systems 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] and Gaspard2 [Gamatié et al., 2011] have been employed to model cyber-physical systems [Selic and Gérard, 2013], but, with few exceptions [Taha et al., 2010,Li et al., 2018], they do not support refinement until a low level of abstraction nor provide full-system simulation.

Into-CPS [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 a discrete-event model, approximations of continuous-time behavior are subsequently replaced by couplings to continuous-time models.

TTool [Apvrille, 2011], an open-source modeling and verification framework, provides to some extend analog/mixed signal modeling and virtual prototyping [Genius et al., 2019]. The determination of the schedule and causality between analog and digital parts of the system is based on timed Synchronous Data Flow (SDF) [Lee and Messerschmitt, 1987].

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

2.2 Analog/Mixed signal hardware design

The following tools target analog/mixed signal or multi-domain design and virtual prototyping.

Ptolemy II [Ptolemy.org, 2014] is based upon a data-flow model. It addresses digital/analog systems by defining several sub domains.

Metro II [Davare et al., 2007] is based on hierarchical high level models. A common simulation kernel handles the entire execution, using *Adapters* for data synchronization between components belonging to different MoC, leaving the implementation of synchronization mechanisms to the designer.

Modelica [Fritzson and Engelson, 1998], an object-oriented modeling language for describing and simulating cyber-physical systems, comes without predefined time synchronization. The simulation engine manipulates objects containing sets of equations in a symbolic way in order to determine the order of execution.

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.

Many approaches are based on SystemC [IEEE, 2011], with or without alteration of the simulation kernel, initially targeted only toward discrete systems simulation.

3 SystemC-AMS modeling

SystemC AMS extensions [Accellera Systems Initiative, 2010] is an extension of SystemC with AMS (Analog/Mixed Signal) and RF (Radio Frequency) features [Vachoux et al., 2003]; several Models of Computation are predefined. A proof-of-concept implementation [Einwich, 2016] evolved into an industrial tool that also handles validation of hardware against software and generates Simulink or C Code.

Digital components are described by a Discrete Event (DE) MoC, while analog components follow the Timed Data Flow (TDF) MoC, based on the timeless Synchronous Data Flow semantics [Lee and Messerschmitt, 1987]. The most low-level MoC is called Electrical Linear Network (ELN). It relies on equations to capture the behavior of electrical circuits in a simplified way.

3.1 Discrete Event

A Discrete-Event (DE) simulation abstracts a system as a discrete sequence of events in time, where each event signals a change of state, in contrast to continuous simulation in which the system state changes continuously over time. System C AMS DE modules have input and output ports, and contain SystemC code.

3.2 Timed Data Flow

A Timed Data Flow (TDF) *module* samples continuous functions at discrete intervals. Such a module is described with an attribute representing the time step and a processing function, a mathematical function depending on the module inputs and/or internal states.

TDF modules have the following attributes:

1. Module time step (**Tm**) denotes the period during which the module is activated, which is the case if enough samples are available at its input ports.
2. Rate (**R**). Each module reads or writes a fixed number of data samples each time it is activated, annotated to the port as port rate.
3. Port time step (**Tp**) denotes the time interval between two operations (read or write).
4. Delay (**D**). A delay can be assigned to a port and will make the port handle a fixed number of samples at each activation, and read or write them in the following activation of the port.

At each time step, a TDF module thus first reads a fixed number of samples from its input ports, then executes the processing function, and finally writes a fixed number of samples to its output ports.

Schedulability denotes the correct static execution order of TDF modules in a cluster containing several modules. A cluster is *schedulable* if the module time step is consistent with the rate and time step of any port within a module. Le T_M denote the module time step, T_{pi} and T_{po} respectively denote the input and output port time steps, R_{pi} and R_{po} respectively denote the input and output port rates:

$$T_M = T_{pi} \times R_{pi} = T_{po} \times R_{po}$$

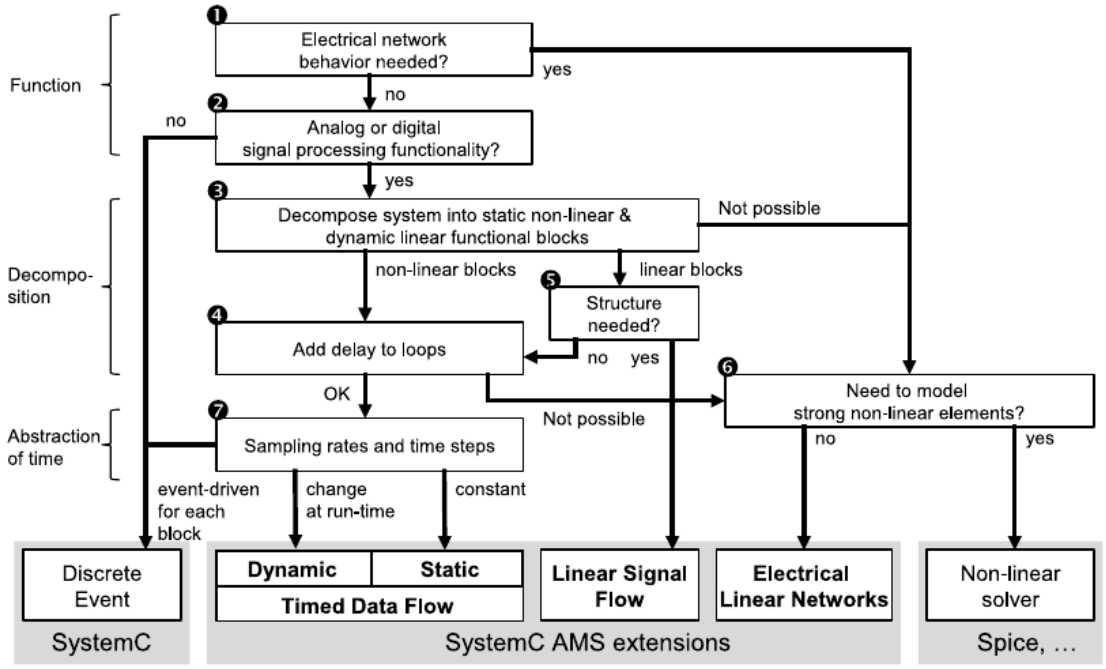

Figure 1: Partitioning of analog behavior into different models of computation [Accellera Systems Initiative, 2010]

Before the static schedule of a cluster can be computed, *time steps* and *sampling rates* that are not indicated in the model need to be calculated by upstream and downstream propagation, as explained in [Accellera Systems Initiative, 2010].

3.3 Electrical Linear Networks

The Electrical Linear Networks (ELN) model of computation introduces the use of electrical primitives and their interconnections to model conservative, continuous-time behavior. The ELN modeling style allows the instantiation of electrical primitives, connected by electrical nodes. The mathematical relations between the primitives are defined at each node in the network, where both the potential (voltage) and flow (current) quantities are used according to Kirchhoff's laws. The electrical network is represented by a set of differential algebraic equations that are taken into account at simulation.

SystemC AMS extensions offers a limited set of primitive modules: voltage or current sources, linear lumped elements (resistors, capacitors, inductors), transmission lines, ideal transformers and amplifiers, linear gyrators, and ideal switches. Unlike for TDF models, there is no possibility to implement user-defined electrical primitives.

An ELN module gives a detailed representation of an electrical circuit. Yet, non-linear behavior cannot be represented; as a consequence, nonlinear elements such as diodes and transistors must be approximated with the existing linear components.

3.4 Simulating the MoC

Converter *ports* are required to connect DE components to TDF components, and reciprocally. Converter *modules* can connect ELN components to TDF or DE modules. When connecting such components, the timing and consistency issues between their different MoC, in particular between TDF and DE, are delicate to handle [Cortés Porto et al., 2021, Andrade et al., 2015].

For ELN modules, a time step can be directly assigned to modules or propagated using the mechanism of the time step within an ELN equation system. In case an ELN model is connected to a TDF model, the time steps from the connected TDF ports are propagated to the ELN model.

Figure 1, published by [Accellera Systems Initiative, 2010], shows the partitioning of analog behavior into different models of computation as recommended by the SystemC AMS working group. For the modeling of digital hardware/software systems (e.g., microcontrollers and processors), interconnect and communication protocols, SystemC (DE) modeling ap-

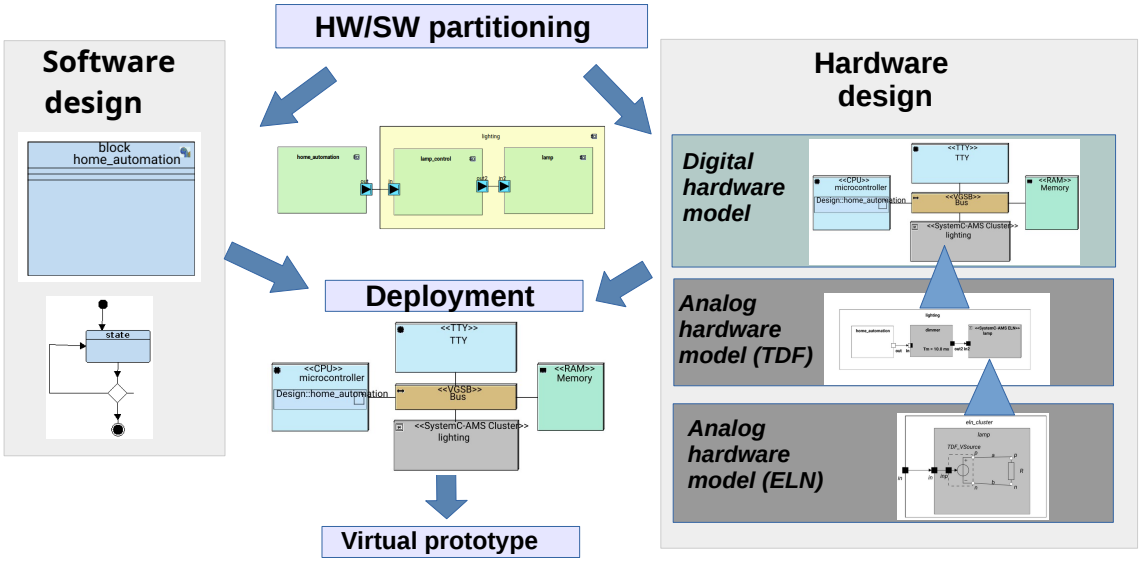

Figure 2: Method with software design

proaches are most suitable. Signal processing functionality should be modeled by LSF. If sampling rates and time steps need to be described, either statically or dynamically, the TDF MoC is chosen. If nonlinear elements such as diodes and transistors have to be precisely modeled, it is preferable to resort to Spice [Quarles et al., 2003]. In combination with SystemC, SystemC AMS covers all of the above MoC except Spice. Our tool currently treats neither nonlinear model generation nor Linear Signal Flow (LSF).

4 Hierarchical modeling of analog hardware components

In the following, we highlight our new contribution. The Timed Data Flow (TDF) model of computation (higher abstraction level) is often insufficient to deal with highly specialized custom circuits [Accellera Systems Initiative, 2010]. In particular, precise interactions with the environment are expected to be studied as soon as possible, i.e., before the actual design is complete: this can be done with TDF descriptions only. We present in the following a top-down, hierarchical manner, using a customized SysML meta-model and generating code that can be used in a SystemC/SystemC AMS simulation environment.

[Cortés Porto et al., 2021, Andrade et al., 2015] already proposed a simulation environment for SystemC-AMS integrating TDF and DE MoC: the simulation of DE components there controls the TDF simulation. Inspired by this simulation hierarchy, we propose a three-level modeling –between which a designer can navigate back and forth– using three kinds of diagrams representing analog/mixed signal hardware, where the DE simulator controls the TDF simulator, which in turn controls the ELN simulation.

4.1 Method overview

Figure 2 displays the overall design method which we suggest for systems with digital and analog parts. The top of the figure focuses on the hardware/-software partitioning step: a functional representation is mapped into a hardware platform, like in [Apvrille, 2011]. This mapping concerns both functions (mapped to e.g. processors or hardware accelerators) and communications (mapped to buses, bridges, memories, ...). Once the functionality has been partitioned into software tasks (represented on the left) and hardware, the deployment diagram (top right) represents all of the selected hardware.

The "Hardware design" part is the main contribution of this paper. The top part of the right part of Figure 2 captures an analog/mixed signal cluster as a grey box in the bottom of the "Digital Hardware Model". The other nodes correspond to the digital parts of the Virtual Prototype. The middle part ("Analog hardware model") zooms into this grey box (it can be opened with a double-click). It shows the SysML representation of the TDF model of this cluster. The three modules of this level capture, from left to right, an output to the digital domain, a TDF block and an abstract representation of an ELN module. Last, the lower hierarchical level ("Analog hardware model") is destined for detailing ELN modules.

Once software and (digital and analog) hardware have been designed, a virtual prototype can be generated. This prototype is built from the free SystemC library [SocLib consortium, 2003], and from analog hardware components described in SystemC AMS, some of these components being detailed in ELN.

4.2 Modeling DE-TDF-ELN modules

Figure 3 displays SysML blocks used to describe a small home automation/lighting system, composed of a light bulb supplied with a voltage controlled by a dimmer, which in turn is controlled by the software running on the —digital–microcontroller of a home automation system. Figures 4 to 6 show the digital, TDF and ELN hardware views, respectively.

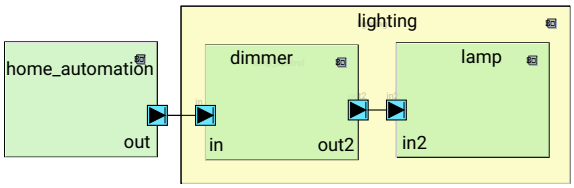

Figure 3: Functional model of the lighting system

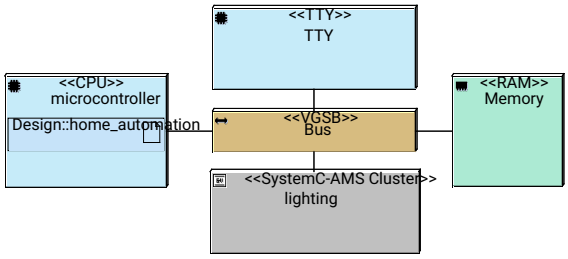

Figure 4: Deployment Diagram

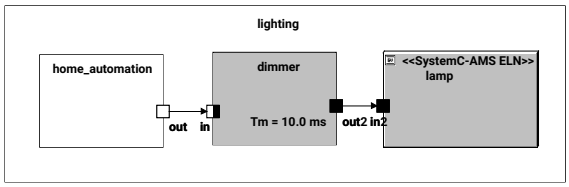

Figure 5: Representation combining three different MoC

Digital hardware, with possibly software running on it, is represented in a UML Deployment Diagram (see Figure 4: A microcontroller (CPU) and its software application are shown in the light blue box on the left (named CPU and *application_code*, respectively). The platform also features a bus, a RAM memory and a TTY for monitoring and debugging. TDF clusters are represented in the deployment diagram as grey boxes; in Figure 4, the TDF controller is shown as a grey box on the bottom. The DE block

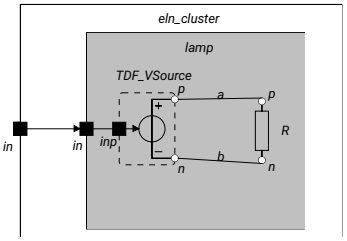

Figure 6: TDF cluster encapsulating an ELN diagram

shown in the left of Figure 5 represents the interface to the microcontroller.

By selecting such a TDF cluster (here: *lighting*), the user opens a panel like the one shown in Figure 5. The left part of this Figure, *home_automation*, represents the interface to the digital hardware, for example a micro controller or general purpose platform running application code. This block is connected to a TDF block (in the middle) which samples the input on the converter port at a given frequency (indicated by $T\_m = 10.0ms$ in the TDF block *dimmer*). Causality issues between the TDF and the DE MoC are explained in [Cortés Porto et al., 2021].

The right hand side of the Figure shows the encapsulation of a ELN cluster (*lamp*) into a TDF cluster (*lighting*). Input and output are handled via TDF ports, for which the sampling frequency of 10 ms is imposed.

The main idea is that an ELN module –like for instance *lamp* in Figure 6– is represented in the TDF panel, featuring the appropriate TDF ports. However, the precise handling of inputs and outputs by ELN components is hidden at this abstraction level: it will be supplied later. At this level of abstraction, a *processing function* reading from and writing to TDF port guarantees that schedulability of the entire TDF cluster can be determined.

By selecting the ELN cluster block, the user opens the corresponding ELN panel (Figure 6). In this toy example, the left hand module has a TDF input port connected to a TDF-to-ELN converter module, a TDF controlled voltage source TDF_VSource. This module is in turn connected via its positive (p) and negative (n) *terminals* to an ELN resistor.

Currently, we support 20 elements out of the 29 defined in the SystemC AMS standard, not counting ports, connectors and terminals. In particular, nullors, gyrators, and the TDF and DE controlled versions of variable resistors, capacitors and inductors were not yet required for our models, but could easily be added if necessary.

Figure 7 shows the toolbar of the new ELN panel featuring all graphical operators which are supported. These ELN elements are also listed in Table 1.

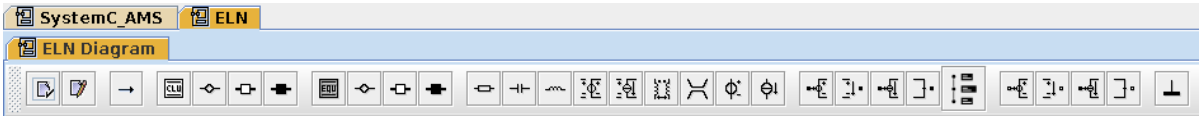

Figure 7: ELN Panel Toolbar

element	symbol
Cluster and module terminal	
TDF cluster and module port	
DE cluster and module port	
Resistor	
Capacitor	
Inductor	
Voltage controlled voltage source	
Voltage controlled current source	
Ideal transformer	
Transmission line	
Independent voltage source	
Independent current source	
Voltage source driven by TDF input signal	
Voltage converted to TDF output signal	
Current source driven by TDF input signal	
Current source converted to a TDF output signal	
Switch driven by a TDF input signal	
Switch driven by a DE input signal	
Voltage source driven by DE input signal	
Voltage converted to DE output signal	
Current source driven by DE input signal	
Current source converted to DE output signal	
Reference node (ground)	

Table 1: ELN modules currently available in the panel

4.3 Virtual prototype generation

For a virtual prototype containing three different Models of Computation, it is particularly important that interactions between the MoC are handled as early as possible in the design process. [Cortés Porto et al., 2021] has already shown how to efficiently generate TDF and DE parts of the prototype as well as their interaction, including validation of causality on the TDF/DEconverter ports. In our tool, this can be done as follows:

1. Select one of the TDF clusters, including the ones with ELN clusters.
2. Activate the "Validation" button. This checks for coherency of the time steps within the TDF cluster as well as for respect of temporal causality between TDF and DE models as described in [Cortés Porto et al., 2021].
3. Activate "Code generation" button. This generates, for all ELN clusters, SystemC AMS TDF module templates with the appropriate input and output ports, leaving their processing functions empty.
4. Select one of the ELN clusters within the TDF cluster.
5. Activate the "Code generation" button. This generates, for the selected ELN cluster representation, SystemC AMS code for the ELN module itself and updates the template for the surrounding TDF block with the instantiation of the internal ELN blocks and the signals connected to the internal ports.

In our tool, the design choice was made that ELN clusters are always modeled inside TDF blocks. The algorithm given in [Cortés Porto et al., 2021] propagates the time steps and checks schedulability and causality on the abstraction level level where interaction between TDF/DE blocks is analyzed. An ELN cluster thus can never be simulated alone, it requires a TDF block that forces its time step. Thus, an ELN module has to be encapsulated within a TDF module.

Code has also to be generated for:

- The top cell, containing the simulation entry point, TDF and DE block instantiation, code for starting and stopping the simulation and optional code for tracing.
- The ELN cluster encapsulation module. This is a TDF module instantiating the ELN modules, their connections among each other and to the TDF modules.
- The ELN module itself.

Algorithm 1 shows the transformation for code generation and scheduling, leaving out the DE part and

Listing 1 Code generation and scheduling algorithm

```
1:  procedure GENERATESYSTEMCAMSCODE
                                  ▷ T time step, 𝔅 block,
                                  ▷ ℭ cluster, 𝔐 module
2:      for each TDF cluster ℭ_TDF do
3:          generate cluster code
4:          for all TDF blocks 𝔅_TDF in ℭ_TDF do
5:              CALCULATESCHEDULE (ℭ_TDF)
6:              if 𝔅_TDF simple TDF block then
7:                  generate TDF block code
8:              else
                    ▷ 𝔅_TDF contains ELN cluster ℭ_ELN
9:                  for all 𝔐_ELN ∈ ℭ_ELN do
10:                     set T_𝔐ELN from 𝔅_TDF
11:                     determine T_pi, T_po for 𝔅_TDF
12:                 end for
13:                 calculate T_ℭELN
14:                 CALCULATESCHEDULE (ℭ_TDF)
15:                 if ℭ_TDF schedulable then
16:                     generate encapsulation code
17:                     for all 𝔐_ELN ∈ ℭ_ELN do
18:                         generate ELN code
19:                     end for
20:                 end if
21:
22:             end if
23:         end for
24:     end for
25: end procedure
```

using the scheduling algorithm CALCULATESCHEDULE of [Cortés Porto et al., 2021] for TDF clusters.

Figure 8 shows the generated SystemC AMS code for the ELN cluster encapsulation and ELN module of the introductory example, respectively. The top cell (not shown here) includes the code for the three parts: DE block, TDF block and ELN representation. The TDF ports have to be connected to the sub modules of the ELN cluster. As a consequence, in the ELN cluster code (top of Figure 8), we find port-to-port binding: port "in" of the module is bound to the cluster port named "in". The ELN module *lamp* (bottom of Figure 8) contains a voltage source controlled by the TDF module *dimmer* via "inp" port connected to the cluster port "in", a ground element, and two nodes a and b connecting them.

4.4 Virtual Prototype

Our simulation environment applies the following hypotheses, sometimes conforming to or contradicting previous approaches:

- The behavior of ELN modules can be described with mathematical equations: these equation systems are solved numerically by the simulation engine at appropriate time steps. Also, for ELN modules connected to a TDF module, the time step from the connected TDF port(s) is propagated to the ELN module. Consistency between locally defined ELN module time steps and propagated time steps is checked by SystemC AMS. Otherwise, the time points for the solution of the ELN equation system or communication with the connected TDF model cannot be defined properly.
- In the presence of DE modules, the DE simulator controls the entire simulation via the converter ports, respecting temporal causality [Cortés Porto et al., 2021, Andrade et al., 2015].
- TDF modules impose their timestep on the ELN modules, as described in [Accellera Systems Initiative, 2010].
- Even if possible according to [Accellera Systems Initiative, 2010], direct assignment of a timestep to an ELN module is currently not allowed.

```
#include <systemc-ams>
#include "lamp.h"

class eln_cluster : public sc_core::sc_module {

    public:
     lamp lamp_0;
    sca_tdf::sca_in<double> in;

     SC_CTOR(eln_cluster) :
        lamp_0("lamp_0"),
        in("in"){
    lamp_0.in(in);
};

SC_MODULE(lamp){
        sca_tdf::sca_in<double> in;
        sca_eln::sca_r R;
        sca_eln::sca_tdf_vsource TDF_VSource;

        SC_CTOR(lamp)
        : in("in"), R("R", 1.0)
        , TDF_VSource("TDF_VSource", 1.0)
        , b("b"), a("a"){
                R.p(a);
                R.n(b);
                TDF_VSource.p(a);
                TDF_VSource.n(b);
                TDF_VSource.inp(in);
        }
    private:
        sca_eln::sca_node b;
        sca_eln::sca_node a;
};
```

Figure 8: Generated encapsulation and ELN code

```
Info: SystemC-AMS:
        3 SystemC-AMS modules instantiated
        2 SystemC-AMS views created
        2 SystemC-AMS synchronization objects/solvers instantiated

Info: SystemC-AMS:
        1 dataflow clusters instantiated
          cluster 0:
                2 dataflow modules/solver, contains e.g. module: dimmer_0
                2 elements in schedule list,
                10 ms cluster period,
                ratio to lowest:  1              e.g. module: dimmer_0
                ratio to highest: 1 sample time  e.g. module: dimmer_0
                0 connections to SystemC de, 1 connections from SystemC de

Info: SystemC-AMS:
        ELN solver instance: sca_linear_solver_0 (cluster 0)
                has 3 equations for 2 modules (e.g. lamp_0.R),
                1 inputs and 0 outputs to other (TDF) SystemC-AMS domains,
                0 inputs and 0 outputs to SystemC de.
                10 ms initial time step

Info: SystemC-AMS:
        ELN solver instance: sca_linear_solver_0 (cluster 0)
                has calculated 10000 time steps the equation system was 1 times re-initialized
                the following max. 10 modules requested the most re-initializations: lamp_0.R  1
```

Figure 9: SystemC AMS simulation

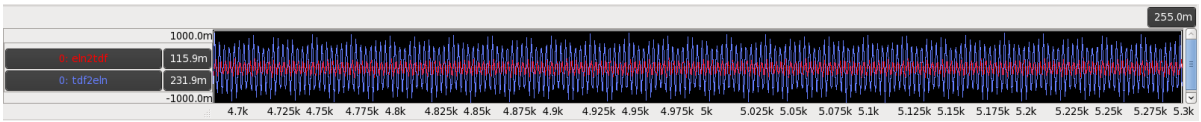

Figure 10: Waveform results: exchanges between ELN and TDF module

Figure 9 shows the SystemC AMS simulation for the introductory example. Three modules, one DE, TDF and ELN each, are instantiated. Two views correspond to the TDF and the ELN MoC. After this, the dataflow (TDF) cluster is analyzed, then the equations of the ELN cluster solved and finally the simulation is run for 10000 simulation steps.

Finally, Figure 10 shows the waveforms of the signals exchanged between the ELN and TDF module.

5 Case Study: Scalable SAR ADC

5.1 Analog-to-digital converters

As stated in the introduction, cyber-physical systems span three domains (analog, digital and physical). The digital and analog domains are interconnected with digital-to-analog (DAC) and analog-to digital (ADC) converters. These converters are expected to be of small size and designed with high energy efficiency in mind.

Successive Approximation Register (SAR) ADCs provide good power efficiency for medium-resolution applications. For instance, as stated by [Shen et al., 2017], Wireless local area networks (WLAN) require resolutions above 10 bit and sampling rates of about 40 MS/s, digital TV receivers up to 80 MS/s sampling rate, while Bluetooth applications require resolution and sampling rates of around 12-bit and 11 MS/s, respectively. ADC should furthermore support multiple applications, so their design is required to be easily reconfigurable.

[Gylling and Olsson, 2015] overview several possible designs for SAR-ADC circuits. The basic idea of SAR ADCs is to approximate the actual voltage successively by several iterations, corresponding to the number of bits which are fed back to a DAC. The most essential parameter of this circuit is its bit precision: from 3 to 12-bit precision.

5.2 Overview of the use case

We consider a SAR ADC designed in a recent project [Louërat and Porte, 2022], shown in Figure

11. This SAR ADC mostly consists of analog components, with one digital component used for controlling the device. The implementation was performed with the *Oceane* toolchain [Porte, 2008]. In this project, one interesting challenge was to couple the implementation process with system-level modeling approach. Indeed, a very detailed model of the ADC circuit was not yet available at the beginning. For instance, the number of bits of precision ultimately required by the system was not yet known, thus the corresponding system-level model had to be easily parameterizable. Also, a TDF overview representation of the mixed analog-digital circuit was necessary to evaluate the interplay of digital (Ctl logic) and analog (Comparator and DAC) circuits. In the scope of this project, we could thus evaluate our SysML-based modeling tool, including the newly introduced panel for circuit-level design explained in the contribution section.

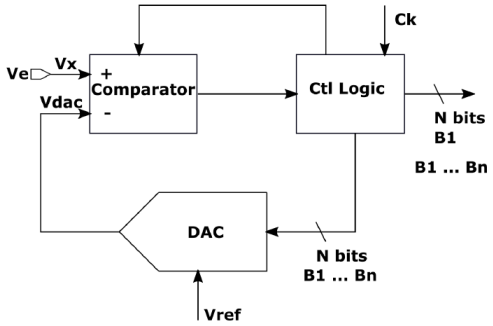

Figure 11: SAR Analog-Digital converter design overview [Louërat and Porte, 2022]

Figure 12 shows the main algorithm of the ADC [Louërat and Porte, 2022]. Basically, an incoming voltage V_x is to be determined iteratively. An initial voltage value is set to V_in and all bits are set to 0. Then, the most significant bit (MSB) B_n is set to 1. At a given iteration i, V_x is compared to a generated voltage V_dac. A bit is set to 1 if the voltage is higher, set to 0 if the voltage is lower, starting with MSB B_n and progressing down to B_1. This algorithm was implemented in the VHDL hardware description language [IEEE, 1987] for an FPGA-based test (Field-programmable Gate Array).

These bits B_1 to B_n are used to control a digital analog converter (DAC), which produces the more precise voltage for the next iteration. For example, if we assume $V_{ref} = 2V$ and a 3-bit resolution then $V_x = 2.6V$. In the first iteration, V_x is compared to $V_{dac} = V_{ref}$ and as it is higher, thus bits are 100 and $V_{ref}/2$ is added to V_{dac}. In the second iteration, V_x is compared to $V_{dac} = 3V$ and as it is lower, the control bit remains unchanged and $V_ref/4$ is added to V_{dac}

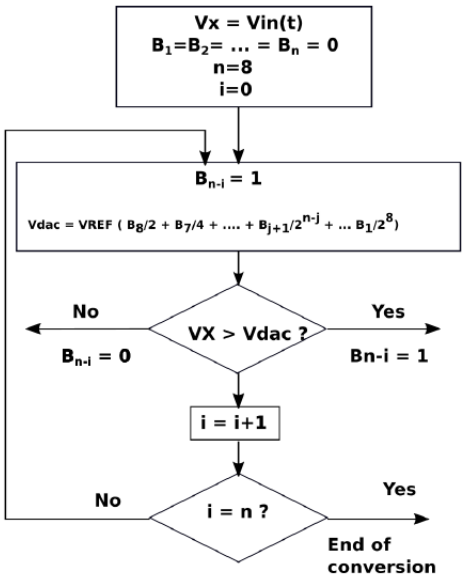

Figure 12: Conversion algorithm [Louërat and Porte, 2022]

and it becomes 2.5V. In the final iteration, $V_x > V_{dac}$, thus the third control bit is set to 1. We obtain an approximation of 2.5V.

5.3 Models

Figure 13 shows the detailed hardware implementation proposed by [Louërat and Porte, 2022]: a non-differential ADC with implicit sampling using capacitor top plates. On the upper center, we find a comparator (CMP) which compares zero/ground voltage (VSS) to the voltage generated in each cycle by the DAC (VDAC). Shown on the lower left hand side, the DAC produces this voltage from $i+1$ capacitors which are either activated (switch closed) when the control bit S_i is 1, deactivated when it is 0. B_i has the same value as S_i but is destined for digital output as a bit vector. Thus, during n iterations, the incoming voltage is approximated with n bit precision. The additional capacitor on the right sets the starting capacity, the others then yield 2^0,2^1, etc ... 2^{N-1} times that capacity.

The implementation of this design is a challenging test case for our tool extension, because (i) the digital control circuit and the analog comparator and DAC are combined on a single chip and (ii) the complexity of low level modeling is high.

5.3.1 Digital hardware model

System-level design is restricted to the external digital control in our study to the generation of a Start-of-

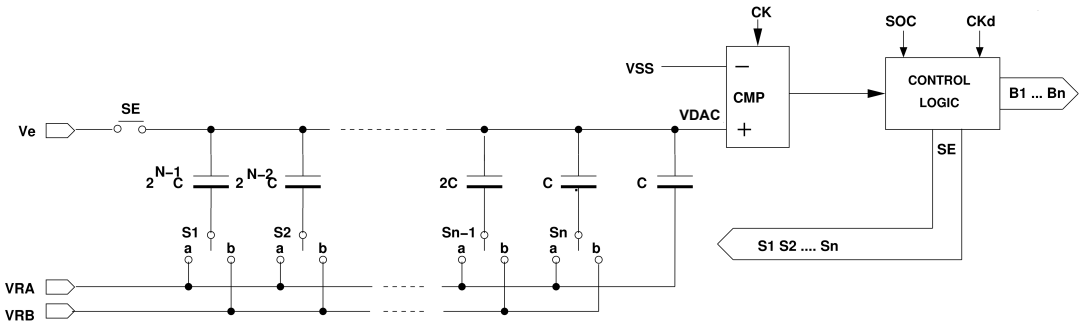

Figure 13: Analog-Digital converter: electronic design (simplified) from [Louërat and Porte, 2022]

Conversion (SoC) signal generated by software.

In current experimentation, code on the digital platform is essentially limited to giving start/stop signals, to I/O and debug functionality. The deployment on a momo processor platform with one analog component ressembles the one of the introductory example shown in Figure 4.

The sampling algorithm is implemented in the Control Logic component. Destined to be implemented in hardware, it was translated to SystemC from the VHDL digital hardware description language and precisely reflects the functionality shown in the algorithm of Figure 12.

5.3.2 Analog hardware model (TDF)

Figure 15 gives an overview of the overall SysML based representation of the SystemC AMS TDF design. On this level, the entire digital part running the software is in fact represented on the lower right within one DE block, whose only role here is to provide the start of conversion (SoC) signal on its single output port as currently the tool allows only TDF blocks to contain ELN.

The control algorithm thus has been implemented in the *processing* function of the *control_logic* block (Figure 14). The *control_logic* block is considered mixed-signal; it features TDF and converter ports.

The comparator-and-DAC block has two TDF entry ports called *start_conversion* and *in_bits*. The *start_conversion* TDF signal is received from the *control_logic* block, the configurable *in_bits* signal contains the *n* bits controlling the switches in the DAC. As the number of control bits is configurable, we make use of the multiport mechanism: the arity of a TDF port can be configured.

The block also features an output port *VDD_out*, providing the voltage calculated after each iteration of the algorithm (output of CMP block in Figure 13), a floating point value.

5.3.3 Analog hardware model (ELN)

Double clicking on the *comparator_and_DAC* block opens the most detailed view (Figure 16). The comparator_and_DAC block actuelly contains the two analog ADC blocks *Comparator (CMP)* and *DAC* (representing the entire left part of Figure 13). These blocks are described in SystemC AMS ELN.

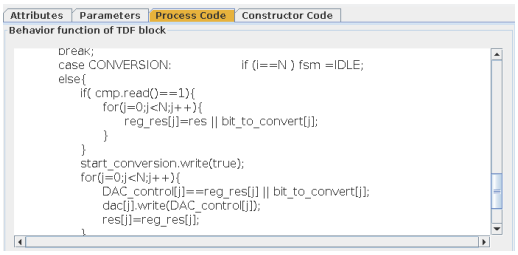

Figure 14: ADC control logic: processing function

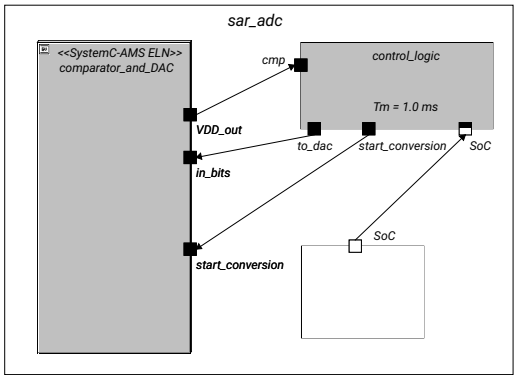

Figure 15: TDF model view of the SAR-ADC in the tool

The comparator on the top receives *V_dac*, on an ELN terminal (bottom of the block) from the DAC that produced it with the SAR method. It is compared to the voltage to be measured, *V_x*, modeled by an independent voltage source on the left of the comparator. At each iteration, the result of the successive aproximations is transmitted to the *control_logic* block by a TDF port *VDD_out* (a *double* value) .

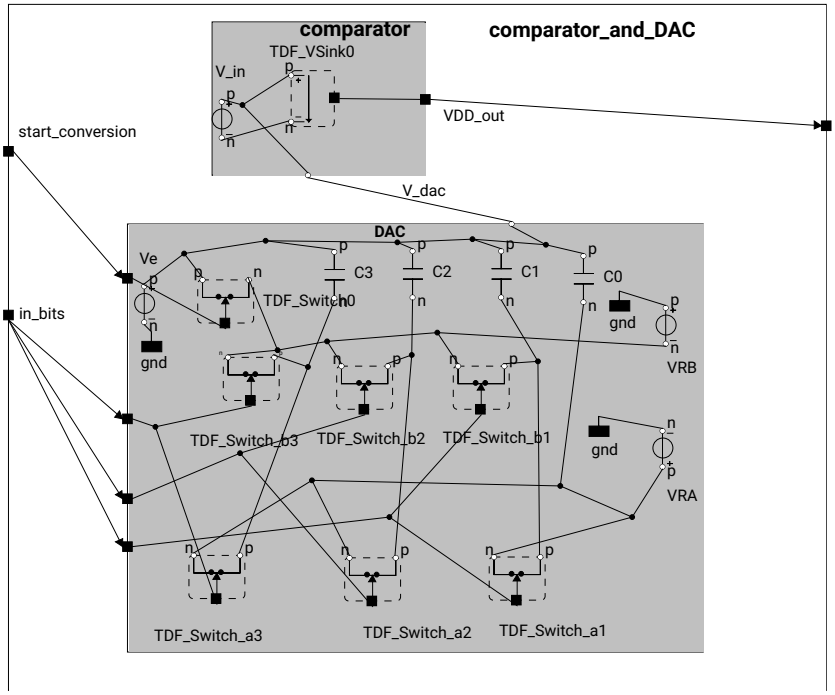

Figure 16: Overview of the ELN Model of the comparator and DAC module: TTool-AMS

The lower part of the Figure represents the DAC. The voltage *Ve* is generated by the independent voltage source on the upper left of the DAC model.

As in ELN modeling it is not foreseen to handle arrays of ports, the ports in the cluster port *in_bits* (on the lower left in the figure and in our example of arity 3) are split into three individual TDF module ports controlling two rows of three switches each.

Two rows of three TDF controlled Switches (TDF_Switch_a1, TDF_Switch_a2, TDF_Switch_a3, TDF_Switch_b1, TDF_Switch_b2, TDF_Switch_b3) take up the central part of the design and are connected, by their control ports, to one of three TDF in_bits signals, each representing one of the control bits for one switch of each row. There are four capacitors, three to be controlled by two switches each. C0 is not controlled by a switch and imposes the initial capacitor value which is then doubled, quadrupled etc. as described in the algorithm above, by activating more and more switches. A seventh switch *TDF_Switch*_0 is connected to the start_conversion port. Voltage sources Va and Vb on the right are set to 2V and 0V, respectively.

6 Conclusion and future work

SystemC AMS based hierarchical design of cyber-physical systems is now possible with our tool, with a real support for both digital and analog parts.

Currently, the consistency between ELN and TDF is checked by the SystemC AMS simulator. The hypotheses from section 4.4 can be used to validate schedulability and causality between TDF and ELN *before* simulation, at prototyping time, similar to what was shown in [Cortés Porto et al., 2021] for these properties between TDF and DE.

ELN diagrams can quickly become complex to be read: we are thus working on visual improvements, such as the use of colors and a better representation for line crossing.

Our tool also provides comfortable possibilities to model, verify and simulate embedded software on a virtual prototype: this aspect has been left out in the present paper in order to focus on the hardware design part.

REFERENCES

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

Andrade, L., Maehne, T., Vachoux, A., Ben Aoun, C., Pêcheux, F., and Louërat, M.-M. (2015). Pre-Simulation Formal Analysis of Synchronization Issues between Discrete Event and Timed Data

Flow Models of Computation. In *Design, Automation and Test in Europe, DATE Conference*.

Apvrille, L. (2011). *TTool, an open-source toolkit for the modeling and verification of embedded systems, http://ttool.telecom-paristech.fr/*.

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.

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

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.

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.

Gamatié, A., Beux, S. L., Piel, É., Atitallah, R. B., Etien, A., Marquet, P., and Dekeyser, J.-L. (2011). A model-driven design framework for massively parallel embedded systems. *ACM Trans. Embedded Comput. Syst*, 10(4):39.

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*.

Gylling, V. and Olsson, R. (2015). Implementation of a 200 msps 12-bit sar adc.

IEEE (1987). *IEEE Standard VHDL Language Reference Manual*.

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.

Li, L. W., Genius, D., and Apvrille, L. (2018). Formal and virtual multi-level design space exploration. In *MODELSWARD, Springer CCIS vol 880*, pages 47–71.

Louërat, M.-M. and Porte, J. (2022). scalable sar adc, technicat report, chips4makers.io.

Porte, J. (2008). Oceane: Software tool for analog design and education". *https://www-soc.lip6.fr/en/team-cian/softwares/oceane/*.

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

Quarles, T., Pederson, D., Newton, R., Sangiovanni-Vincentelli, A., and Wayne, C. (2003). Spice home page. *Go online to http://bwrc. eecs. berkeley. edu/Classes/IcBook/SPICE*.

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

Shen, Y., Zhu, Z., Liu, S., and Yang, Y. (2017). A reconfigurable 10-to-12-b 80-to-20-ms/s bandwidth scalable sar adc. *IEEE Transactions on Circuits and Systems I*, 65(1):51–60.

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

Taha, S., Radermacher, A., and Gérard, S. (2010). An entirely model-based framework for hardware design and simulation. In *DIPES/BICC*, volume 329 of *IFIP Advances in Information and Communication Technology*, pages 31–42. Springer.

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