Hierarchical Design of Cyber-Physical Systems

Daniela Genius[1] and Ludovic Apvrille[2]

[1]*Sorbonne Université, LIP6, CNRS UMR 7606, Paris, France*

[2]*LTCI, Télécom, Institut Polytechnique de Paris, Sophia-Antipolis, France*

Keywords: Cyber Physical Systems, Analog/Mixed Signal Design, SystemC AMS, 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 combined SystemC/SystemC AMS 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 (CPS) span three domains: analog, digital and physical, using most often off-the-shelf analog and digital components 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 unavailable, advocating for from-scratch designs, sometimes both for analog and digital parts.

For such custom designs, 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: they are commonly designed at different abstraction levels, depending on the design patterns already available. Last, due to the significant semantic difference between MoCs, similar models or unique tools can not be used to study the same system.

TTool (Apvrille,), an open-source modeling and verification framework, provides to some extend analog/mixed signal modeling and virtual prototyping. It offers 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. Analog / mixed-signal parts of the virtual prototype are generated in the SystemC AMS language (Vachoux et al., 2003).

Our previous contributions (Genius et al., 2019) already highlighted how to jointly express in SysML analog and digital parts, and how our tool can support this. Based on these, the present paper explains how to efficiently capture analog parts of CPS that require more detailed modeling in a third MoC – as well as the interactions between the three MoCs – and how to generate a SystemC AMS virtual prototype directly from these new SysML models. Abstraction levels thus now range from a near-circuit precision up to more abstract analog and digital models (e.g., transactional models). Just like many SysML tools, TTool already offers verification capabilities for the digital parts, thus this part is not addressed in the paper.

Related work is discussed in Section 2,basic concepts of SystemC AMS modelling 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.

UML/SysML based modeling techniques have been employed to model cyber-physical systems (Selic and Gérard, 2013). With few exceptions, however they do not support refinement until a low level of abstraction.

Ptolemy, perhaps the most well known and initially based on a data-flow model, evolved into

Genius, D. and Apvrille, L.
Hierarchical Design of Cyber-Physical Systems.
DOI: 10.5220/0011654400003402
In *Proceedings of the 11th International Conference on Model-Based Software and Systems Engineering (MODELSWARD 2023)*, pages 117-124
ISBN: 978-989-758-633-0; ISSN: 2184-4348

Ptolemy II (Ptolemy.org, 2014), which proposes many different MoCs including continuous time, data flow and DE and studies their heterogeneous combinations; MoCs can be combined hierarchically.

Metro II (Davare et al., 2007) is based on hierarchical high level models. A common simulation kernel handles the entire execution, 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.

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.

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.

The SICYPHOS framework (Wawrzik et al., 2015) proposes SysML for the overall model of the system structure and component interfaces between domains, which are then translated into domain-specific languages like SystemC AMS or Modelica.

3 SystemC-AMS MODELING

SystemC AMS (Accellera Systems Initiative, 2010) is an extension of SystemC with AMS (Analog/Mixed Signal) and RF (Radio Frequency) features (Vachoux et al., 2003); several MoC are predefined. The industrial framework COSIDE (Einwich, 2022) 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 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. SystemC AMS DE modules have input and output ports, and contain SystemC code.

3.2 Timed Data Flow

A 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 read or write samples only in the following activation of the port.

At each time step, a TDF module reads a fixed number of samples from its input ports, executes the processing function, and 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.

3.3 Electrical Linear Networks

The 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; 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.

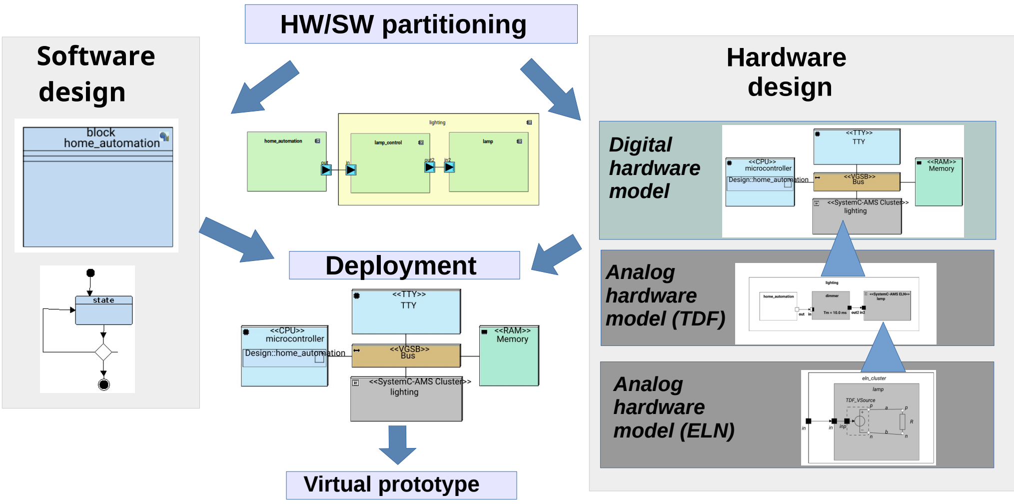

Figure 1: Overview of the hierarchical Method with software design.

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.

4 HIERARCHICAL MODELING OF ANALOG HARDWARE COMPONENTS

In the following, we highlight our new contribution: the integration of the ELN MoC. The 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, 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 to be used in a SystemC/SystemC AMS simulation environment.

In simulation environments 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 1 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 onto a hardware platform, like in (Apvrille,). 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 on the right of Figure 1 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 (TDF)") 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 (ELN)") 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 a free SystemC library, 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 2 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 3 to 5 show the digital, TDF and ELN hardware views, respectively.

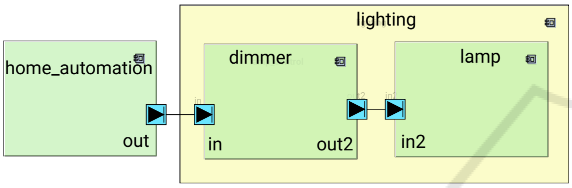

Figure 2: Functional model of the lighting system.

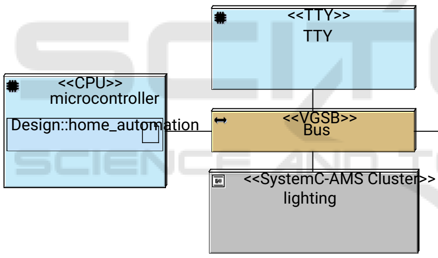

Figure 3: Deployment Diagram.

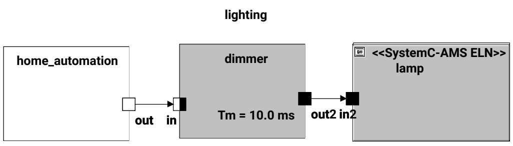

Figure 4: Representation combining three different MoC.

Digital hardware, with possibly software running on it, is represented in a UML Deployment Diagram (see Figure 3: 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 3, the TDF controller is shown as a grey box on the bottom. The DE block shown in the left of Figure 4 represents the interface to the microcontroller.

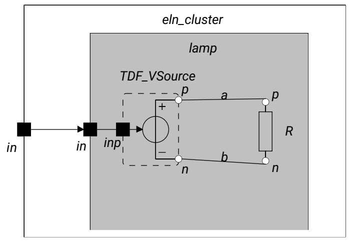

Figure 5: TDF cluster encapsulating an ELN diagram.

By selecting such a TDF cluster (here: *lighting*), the user opens a panel like the one shown in Figure 4. 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 5– 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.

By selecting the ELN cluster block, the user opens the corresponding ELN panel (Figure 5). 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.

Figure 6 shows the toolbar of TTool's new ELN panel featuring all graphical operators which are supported: voltage or current sources, linear lumped elements (resistors, capacitors, inductors), transmission lines, ideal transformers and amplifiers, and ideal switches, each with either TDF or DE ports.

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

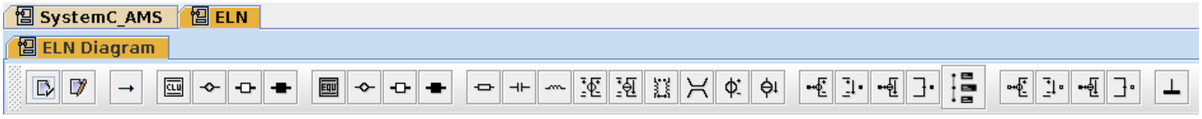

Figure 6: ELN Panel Toolbar.

et al., 2021) has already shown how to efficiently generate TDF and DE parts of the prototype as well as their interaction. In TTool, 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.
3. Activate "Code generation" button. This generates, for all ELN clusters, SystemC AMS TDF module templates with the appropriate input and output ports, leaving processing functions empty.
4. Select one of the ELN clusters in 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. Our algorithm (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.

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.

Listing 1 shows the transformation for code generation and scheduling, leaving out the DE part and using the scheduling algorithm CALCULATESCHEDULE of (Cortés Porto et al., 2021) for TDF clusters.

Listing 1: Code generation and scheduling algorithm.

procedure GENERATESYSTEMCAMSCODE
  ▷ T time step, $\mathscr{B}$ block, ▷ $\mathscr{C}$ cluster, $\mathscr{M}$ module
    for each TDF cluster $\mathscr{C}_{\mathrm{TDF}}$ **do**
        generate cluster code
        for all TDF blocks $\mathscr{B}_{\mathrm{TDF}}$ in $\mathscr{C}_{\mathrm{TDF}}$ **do**
            CALCULATESCHEDULE ($\mathscr{C}_{\mathrm{TDF}}$)
            if $\mathscr{B}_{\mathrm{TDF}}$ simple TDF block **then**
                generate TDF block code
            else
                ▷ $\mathscr{B}_{\mathrm{TDF}}$ contains ELN cluster $\mathscr{C}_{\mathrm{ELN}}$
                for all $\mathscr{M}_{\mathrm{ELN}} \in \mathscr{C}_{\mathrm{ELN}}$ **do**
                    set $T_{\mathscr{M}_{\mathrm{ELN}}}$ from $\mathscr{B}_{\mathrm{TDF}}$
                    determine T_{pi}, T_{po} for $\mathscr{B}_{\mathrm{TDF}}$
                end for
                calculate $T_{\mathscr{C}_{\mathrm{ELN}}}$
                CALCULATESCHEDULE ($\mathscr{C}_{\mathrm{TDF}}$)
                if $\mathscr{C}_{\mathrm{TDF}}$ schedulable **then**
                    generate encapsulation code
                    for all $\mathscr{M}_{\mathrm{ELN}} \in \mathscr{C}_{\mathrm{ELN}}$ **do**
                        generate ELN code
                    end for
                end if
            end if
        end for
    end for
end procedure

4.4 Virtual Prototype

Our simulation environment applies the following hypotheses:

- 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.
- In the presence of DE modules, the DE simulator controls the entire simulation via the converter ports, respecting temporal causality.
- 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.

In the simulation, the TDF cluster is analyzed, then the equations of the ELN cluster solved.

5 CASE STUDY: SCALABLE SAR ADC

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. ADC should furthermore support multiple applications, so their design is required to be easily reconfigurable. 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, spanning from 3 to 12-bit.

We consider a SAR ADC designed by our electrical circuits team in a recent common project (Louërat and Porte, 2022). One interesting challenge was to obtain a system-level model of the ADC circuit, which was not yet available at the beginning, in order to evaluate the interplay of digital (Ctl logic) and analog (Comparator and DAC) circuits. In particular, the number of bits of precision ultimately required by the system was not yet known, the corresponding system-level model thus had to be easily parameterizable. In the scope of this project, we could thus evaluate the new extensions our SysML-based modeling tool on a rather complex use case.

Figure 7 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. 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.

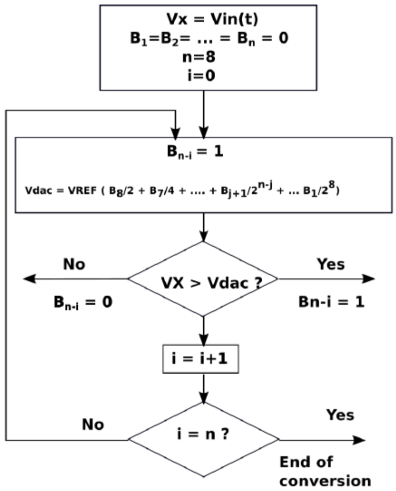

Figure 7: Conversion algorithm (with permission from (Louërat and Porte, 2022)).

5.1 Models

Figure 8 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. 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, ... 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.

5.1.1 Digital Hardware Model (DE)

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

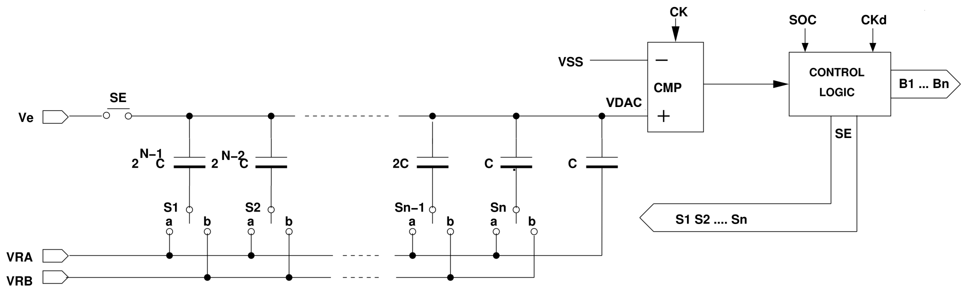

Figure 8: Analog-Digital converter: electronic design (simplified, with permission from (Louërat and Porte, 2022)).

shown in the algorithm of Figure 7.

5.1.2 Analog Hardware Model (TDF)

Figure 9 gives an overview of the overall SysML based representation of the TDF design. On this level, the entire digital part running the software is represented on the lower right within one DE block, whose only role is to provide the start of conversion (SoC) signal on its output port. The *control_logic* block 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 *in_bits* signal contains the *n* bits controlling the switches in the DAC; 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 8), a floating point value.

5.1.3 Analog Hardware Model (ELN)

Double clicking on the *comparator_and_DAC* block opens the most detailed view (Figure 10). The comparator_and_DAC block actuelly contains the two analog ADC blocks *Comparator (CMP)* and *DAC* (representing the entire left part of Figure 8). These blocks are described in SystemC AMS ELN.

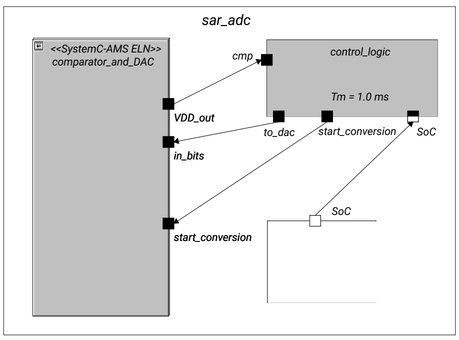

Figure 9: TDF model view of the SAR-ADC in the tool.

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) .

The lower part of the Figure represents the DAC. The voltage *Ve* is generated by the independent voltage source (upper left). Two rows of three TDF controlled switches 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. Of the four capacitors, three are controlled by two switches each. C0, not controlled by a switch, 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 is connected to the start_conversion port.

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. TTool already 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.

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.

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.

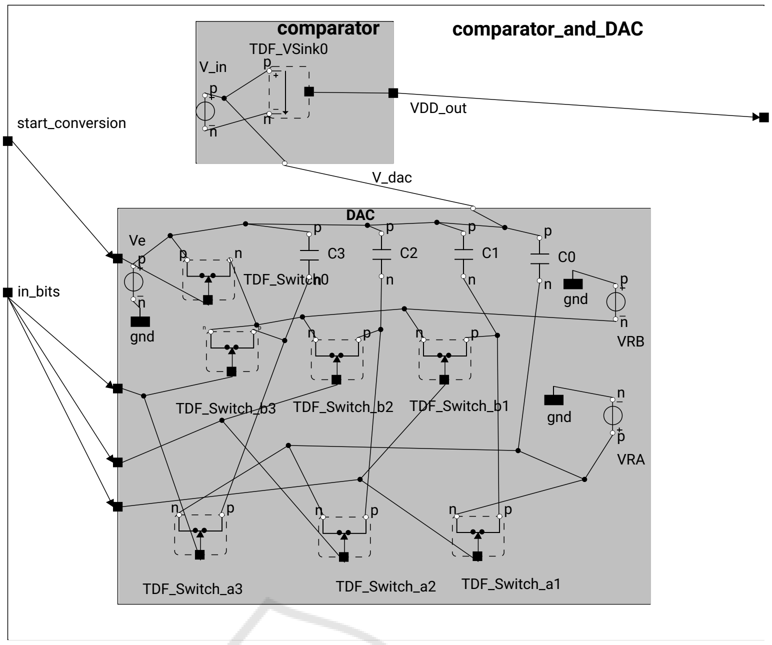

Figure 10: Overview of the ELN Model of the comparator and DAC module in TTool.

REFERENCES

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

Andrade, L. et al. (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. *TTool, an open-source toolkit for the modeling and verification of embedded systems, https://ttool.telecom-paris.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. et al. (2007). A next-generation design framework for platform-based design. In *DVCon*, volume 152.

Einwich, K. (2022). *Coside, https://www.coseda-tech.com*.

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

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

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

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.

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

Wawrzik, F., Chipman, W., Molina, J. M., and Grimm, C. (2015). Modeling and simulation of cyber-physical systems with sicyphos. In *2015 10th Int. Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)*.