



HAL
open science

A Tool for Investigating Cyber-Physical Systems via SystemC AMS Virtual Prototypes Derived from SysML Models

Daniela Genius, Ludovic Apvrille

► **To cite this version:**

Daniela Genius, Ludovic Apvrille. A Tool for Investigating Cyber-Physical Systems via SystemC AMS Virtual Prototypes Derived from SysML Models. DVCon Europe, Accellera, Nov 2023, Munich (Germany), Germany. pp.1-6. hal-04499955

HAL Id: hal-04499955

<https://hal.sorbonne-universite.fr/hal-04499955v1>

Submitted on 11 Mar 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Tool for Investigating Cyber-Physical Systems via SystemC AMS Virtual Prototypes Derived from SysML Models

Daniela Genius *Sorbonne Université, UMR 7606, LIP6, Paris, France*

Ludovic Apvrille *LTCI, Télécom Paris, IP Paris, Sophia-Antipolis, France*

Abstract

This contribution introduces an open-source software tool, called TTool-AMS, designed to streamline the design process of cyber-physical systems. It achieves this by enabling the direct generation of SystemC AMS virtual prototypes from Systems Modeling Language (SysML) models. This tool acknowledges the diversity of Models of Computation (MoC) in the design process, accommodating three types of SystemC AMS MoC and their respective conversion interfaces. It aims to simplify early-stage integration and enhance exploration of the interactions between analog and digital components in cyber-physical systems.

1. Introduction

For custom design of Analog/Mixed Signal (AMS) systems, the splitting of functionality between analog and digital parts, as well as the study of their interaction, is of prime importance, and should therefore be done as early as possible in the design phase, relying on simulation or formal verification. As the Models of Computation (MoC) of these two aspects strongly differ, they are commonly designed at different abstraction levels. Better tool integration would be achieved by multi-level modeling and simulation of analog and digital aspects, ranging from near-circuit precision to more abstract analog and digital models.

In early experimentation, one wishes to determine the best (cheapest, most energy efficient, most secure) implementation without investing in non-reusable material. Moreover, software is subject to frequent changes and the designer should ideally be able to test it on a virtual prototype.

Our primary contribution is thus to offer a SysML-based tool featuring multi-level modeling capabilities, and able to generate a SystemC AMS virtual prototype of the analog and mixed-signal parts directly from SysML models. The present paper focuses on collabo-

ration of different part of our tool; a larger case study, originating from mixed-signal circuit design, is presented in [15].

By basing our tool on an existing one primarily intended for generation and verification of purely digital virtual prototypes, we benefit from its extensive verification capabilities provided for embedded software, using the approach to formal verification of safety and security properties approach described in [23]. As such, our tool provides robust facilities for modeling, verifying, and simulating embedded software on a virtual prototype of mixed systems. Simulation under SystemC AMS and optionally of a mixed SystemC/SystemC AMS virtual prototype [16] then give indications about performance.

Section 2 discusses related work. Section 3 overviews our contributions. Section 4 introduces our tool extension, Section 5 presents a larger case study featuring three different MoCs. Section 6 concludes.

2. Related Work

This section converges methodologies from two fields: SystemC-based and model-based design.

2.1. Analog/Mixed signal hardware design based on SystemC

Many AMS modeling approaches are based on SystemC [18], with or without alteration of SystemC's simulation kernel. SystemC initially targeted only discrete systems. Frameworks based on SystemC include HetSC [17], HetMoC [28] and ForSyDe [21]. Their main disadvantage is that instantiation of elements and synchronization control is totally left to designers.

Both SystemC-H [22] and SystemC-A [27] extend the SystemC simulation kernel. The former allows only one hierarchical level in models; execution is based on a master-slave relation, while a modified discrete event simulation kernel initializes and simulates the processes

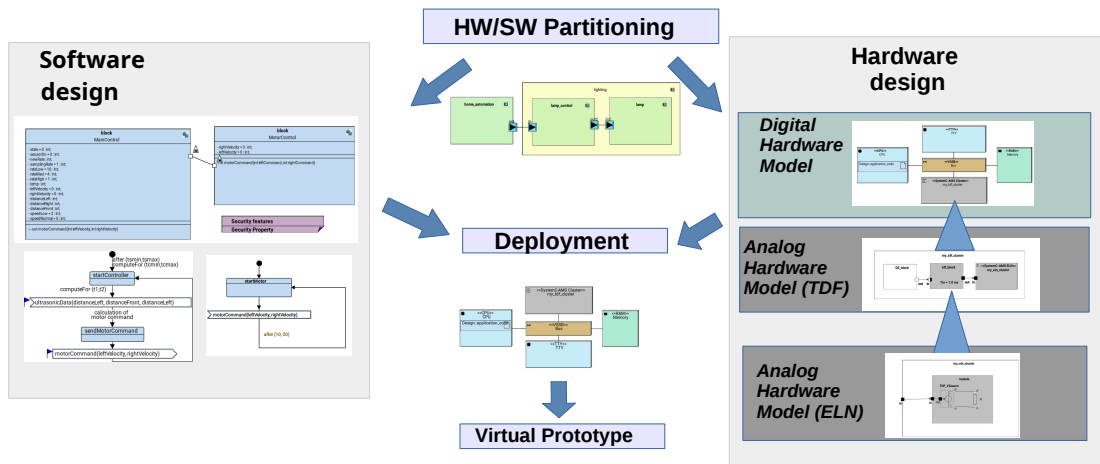


Figure 1: Method overview from [15]. Software design is on the left, and hardware design on the right

described by means of different MoC-specific kernels. Synchronization mechanisms among components defined with different MoC are not available. SystemC-A's scheduler calls the analog kernel phases (iteration and verification) before the SystemC scheduling. The independent analog simulation kernel is able to synchronize with the DE simulation kernel.

SystemC AMS extensions [1] is a standard describing an extension of SystemC with AMS and RF features [26], defining several Models of Computation. In the scope of the project BeyondDreams [5], a proof-of-concept simulator for analog-digital systems has been developed [12], and is distributed by COSEDA.

Within the context of the H-Inception project [24], significant advancements have enabled automatic pre-simulation checks for causality issues [2] from SystemC AMS models. Extending this progress, we can now identify and address causality problems directly from SysML models [8].

2.2. Model-based design for AMS systems

Embedded systems can be designed using different Models of Computation, and with different notions of concurrency and time [19].

A well-known tool for designing analog/mixed signal systems is *Ptolemy II* [20], based upon a dataflow model. Heterogeneous systems are addressed by defining several sub domains, instantiating elements controlling the time synchronization between domains. Yet, time synchronization is left to designer.

Metro II [9] uses *Adaptors* for data synchronization

between components belonging to different MoCs, but again designers are responsible for time synchronization. A common simulation kernel handles all process execution, in which the MoCs are not well separated.

Modelica [14] is an object-oriented modeling language for component-oriented systems containing, e.g., mechanical, electrical, electronic and hydraulic components. Time synchronization is not predefined, the simulation engine must thus manipulate objects in a symbolic way in order to determine an execution order. Linking simulation with different MoCs can be done by using the Functional Mockup Interface [6].

The Discrete Event System Specification (DEVS [7]) is a flexible, hierarchical framework capable of handling both discrete events and continuous systems, including those defined by differential equations. Despite its broad implementation across various platforms, from Petri Net-based to Python-based, DEVS operates on a globally uniform timeline.

Into-CPS [13] leverages model-based formal methods to combine discrete-event controller models with their continuous-time environmental counterparts. It begins with a discrete-event model, which is progressively enhanced by substituting approximated continuous-time behaviors with connections to actual continuous-time models.

Finally, UML/SysML based modeling techniques like MARTE [11] have been used in AMS modeling. However, these techniques usually lack the capability to refine to lower abstraction levels, with a notable exception being TTool [4, 3], a free and open-source modeling framework, which allows for a degree

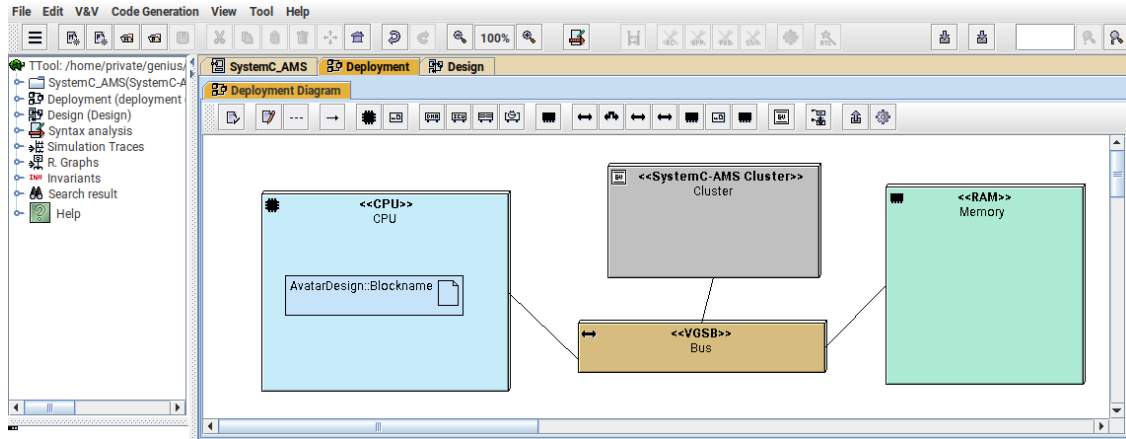


Figure 2: Deployment diagram with analog module

of analog/mixed signal modeling and virtual prototyping [16], including SystemC-based virtual prototyping, while also enabling formal verification.

3. Contribution

Our tool equally accommodates three of the four SystemC AMS MoC: DE, TDF, and ELN. We'll first outline our methodology, then briefly review these recognized MoCs, and finally focus on their representation.

3.1. Overview of our SysML-based method

The top of Figure 1 outlines the hardware/software partitioning step that follows functional modeling. This process splits functions into software tasks and hardware components. The former are depicted using SysML blocks and state machine diagrams (as shown in the left part of the figure), while the latter are represented by UML deployment diagrams and SysML internal block diagrams (featured in the right part of the figure). A virtual prototype is subsequently generated from both digital and analog models. SystemC forms the basis for describing discrete hardware components, utilizing SoCLib [25], while analog components are articulated in SystemC AMS with some details provided in Electrical Linear Network (ELN) the most low-level MoC which relies on equations to capture the behavior of electrical circuits in a simplified way.

The interest in integrating ELN components is more thoroughly discussed in [15]. Specifically, there are instances where standard components fall short due to unique demands, whether they pertain to power consumption, compactness, specific applications, budget constraints, and the like. In such cases, custom designs or modified existing solutions become imperative.

Supporting ELN representation involves three challenges:

1. Representing ELN with SysML elements
2. Revisiting scheduling and causality validation [8]
3. Extending the code generation Algorithm given in [15]

Let us now focus on the right part of Figure 1. At the top right, a Deployment Diagram features the analog cluster as a grey box. Digital hardware executing software is represented in blue.

The middle right picture is a zoom into the analog cluster (a double click in the tool) represented with a SysML internal block diagram. The white block on the left communicates with the outer (digital) world.

The lowest right part of the Figure features the SysML representation of electrical circuits. Our modeling framework can represent resistors, current and voltage sources and sinks, as well as means to connect them to the more abstract TDF and DE components.

3.2. SysML representation of the three MoC

As said above, the main diagram in TTool describing the hardware from which the virtual prototype is the deployment diagram. In Figure 2, an analog cluster is represented as a grey box. Software allocated to processors/microcontrollers is presented in a block diagram and state machines (for the behavior). Their safety is verified with TTool's internal model checker and their security with ProVerif directly from TTool [10].

Figure 3 shows a so-called SystemC AMS panel, featuring the three kinds of modules (DE, TDF, ELN), the TDF and DE ports by which they are connected.

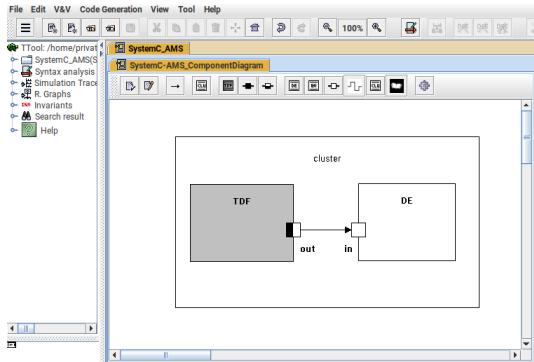


Figure 3: The AMS Panel

The screenshot shows a dialog box titled 'Setting converter input port attributes'. It contains several input fields and dropdown menus:

- Name: out
- Period Tp: 10 ms
- Nbits: 8
- Rate: 1
- Delay: 0
- Type: double
- Origin: Output

 At the bottom, there are two buttons: 'Save and close' and 'Cancel'.

Figure 4: Configuration of the converter port

3.2.1. Discrete Event. In Discrete-Event (DE) modeling, a system is seen as a sequence of discrete events, with each event indicating a state change. This contrasts with continuous simulation where state changes occur continuously over time. DE modules embed SystemC code and are simulated via discrete event simulation. The SystemC AMS panel in Figure 3, on the right side, illustrates that we depict them as white SysML blocks, using white squares for their ports. This visual representation aligns with that used in the SystemC AMS User's Guide [1].

3.2.2. 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 with a processing function. This mathematical function logically depends on the module inputs and/or internal states. At each time step, the module first reads a fixed number of samples from its input ports, then executes its processing function, and finally writes a fixed number of samples to its output ports. TDF modules are grouped into clusters. To ensure consistency between modules, time step and rate are calculated and propagated. Finally, these modules are simulated by the dataflow solver.

On its left side, Figure 3 displays a TDF block with

a converter port. The interface for entering the port's parameters is shown in Figure 4. Similar windows are used to parameterize the different modules and ports, including the definition of the processing function of TDF modules.

3.2.3. Electrical Linear Networks. Electrical Linear Networks are an abstraction of electrical circuits since non-linear behaviors are not represented. As a consequence, nonlinear elements such as diodes and transistors must be approximated with linear components. An equation system has to be solved in order to obtain tension and current at so-called *terminals*. Primitive modules are connected by their terminals via so-called *nodes*, but they can also be connected to TDF or DE modules via converter modules featuring TDF or DE ports, respectively.

3.3. Simulation

Converter *ports* are required to connect DE to TDF. Converter *modules* connect TLM to TDF and DE modules, respectively. Timing and causality issues between the different MoC, in particular between TDF and DE, are delicate to handle (see [8, 2]). In short, the DE simulation controls the TDF simulation, which in turn controls the ELN simulation by imposing the time step. To avoid temporal causality problems, an algorithm given in [8] shows how to propagate the time step to the ELN converter modules, before code can be generated from the SysML models. By inserting adequate delays before code generation, we avoid most of the causality errors raised by the SystemC AMS simulator.

4. Using the Tool

The SystemC AMS panel also contains a tool bar to select blocks, ports and connectors, and to generate code (rightmost button in the toolbar).

ELN circuits are represented using distinct views, each catering to a specific subset of modules. Table 1 enumerates the usual graphical operators employed for capturing ELN circuits. Out of the 29 elements defined in the SystemC AMS standard, we have implemented 20, exclusive of ports, connectors, and terminals.

We significantly extend the generation approach of [15] by introducing ELN blocks featuring DE as well as TDF ports, while the previous contribution was limited to the encapsulation of ELN in TDF blocks, thus allowing only the use of TDF ports only. Consequently, the algorithm given in [15] was modified to consider our new ELN modules. Actually, this leads to a simplified code generation process (Figure 5). In comparison to a
























| 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 TDF input signal |  |
| switch driven by 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

first code generation algorithm presented in [15], we derive a more straightforward and more general algorithm (see Figure 1).

On the operational side, too, the process has also been simplified.

1. Select one of the TDF clusters and open its panel.
2. Activate the "Validation" button. This propagates time steps within the TDF cluster as well as assures respect of causality between TDF and DE models.
3. Activate the "Code generation" button. This generates SystemC AMS code for all TDF clusters, DE and ELN modules. It also generates the top cell.

5. Case Study: POTS

We modeled the Plain Old Telephone System (POTS) taken from [1]. This small to mid-size example exhibits all the features we wish to show: TDF modules

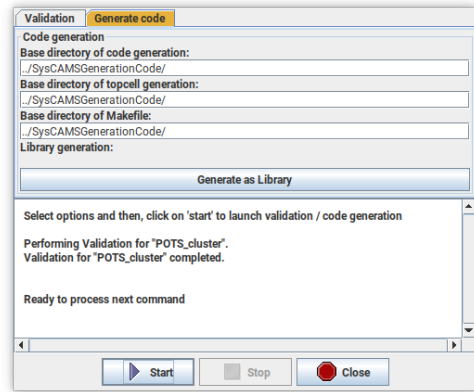


Figure 5: Validation and Code Generation

(*caller* and *subscriber*, a DE module (the phone *device*) as well as four ELN modules regrouping various ELN primitive elements. It features TDF and DE ports (note that *hook* is not a converter port but can be connected directly to the ELN module). The heart of the system consists in four circuits modeled in ELN with our tool, integrated into an environment built upon three TDF and one DE module.

Figure 6 shows an overview of the overall SysML based design created with our tool. It really looks like a SystemC AMS representation, therefore offering a smooth transition for designers used to SystemC AMS. The two modules on the left represent the phone device, the central module represents the phone circuit, and the two modules on the right capture the subscriber line.

Figure 7 displays the design of the ELN part, made of the four circuits. Again, a particular care was taken such that it resembles to the graphical SystemC AMS representations found in [1], while offering the capability of model-driven toolkits in terms of verification and code generation.

A larger case study, showing the high-level design of an AMS prototype of an analog-digital converter, is explained in [15]. Our tool helped colleagues from the analog design group—from whom the from-scratch analog design originated—to make their design parametrizable for high-level exploration.

6. Conclusion and Future Work

We have introduced a tool that seamlessly integrates SystemC AMS-based designs of analog and mixed-signal systems within a complementary SysML high-level modeling environment. This facilitates the creation of platforms that harmoniously integrate three SystemC AMS Models of Computation (MoCs) within

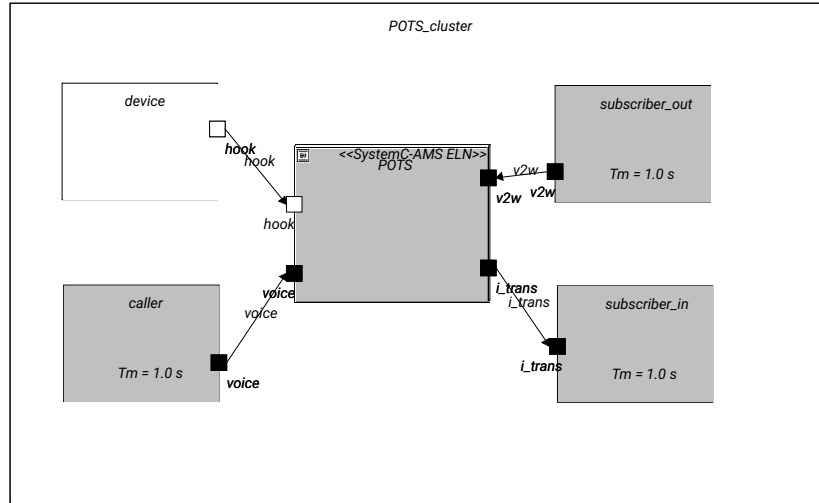


Figure 6: SystemC AMS diagram of POTS circuit and TDF Modules

Listing 1 Code generation and scheduling algorithm (simplified wrt. [15])

```

1: procedure GENERATECODE    ▷ T time step,  $\mathcal{B}$ 
   block,  $\mathcal{C}$  cluster,  $\mathcal{M}$  module
2:   for each TDF cluster  $\mathcal{C}_{\text{TDF}}$  do
3:     generate cluster code
4:     for all TDF blocks  $\mathcal{B}_{\text{TDF}}$  in  $\mathcal{C}_{\text{TDF}}$  do
5:       CALCULATESCHEDULE ( $\mathcal{C}_{\text{TDF}}$ )
6:       if  $\mathcal{B}_{\text{TDF}}$  TDF block then
7:         generate TDF block code
8:         for all  $\mathcal{M}_{\text{ELN}} \in \mathcal{C}_{\text{ELN}}$  do
9:           calculate  $T_{\text{CELN}}$ 
10:          CALCULATESCHEDULE ( $\mathcal{C}_{\text{TDF}}$ )
11:          if  $\mathcal{C}_{\text{TDF}}$  schedulable then
12:            generate ELN code
13:          end if
14:        end for
15:      end if
16:    end for
17:  end for
18: end procedure

```

a single topcell. A standout functionality of our tool is its capability to validate both schedulability and causality before generating SystemC AMS code.

In this paper, we have focused on the hardware design aspect, leaving the aforementioned software potential untouched. Case studies with a larger proportion of software are planned for future work. The strength of the underlying tool [3] lies in the verification of software running on embedded hardware.

We also plan to extend the tool to generate SystemC AMS Linear Signal Flow, thus enlarging the class of applications that can be treated. A remaining challenge lies in formally proving the consistency between TDF and ELN MoCs in converter modules. Currently, we rely on the SystemC AMS simulator for this purpose.

References

- [1] Accellera Systems Initiative. *SystemC AMS extensions Users Guide, Version 1.0*, March 2010.
- [2] L. Andrade, T. Maehne, A. Vachoux, C. Ben Aoun, F. Pêcheux, and M.-M. Louërat. 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*, Mar. 2015.
- [3] L. Apvrille. *TTool, an open-source toolkit for the modeling and verification of embedded systems*, <https://ttool.telecom-paris.fr>, (accessed 2023).
- [4] L. Apvrille, W. Muhammad, R. Ameur-Boulifa, S. Coudert, and R. Pacalet. A UML-based environment for system design space exploration. In *2006 13th IEEE International Conference on Electronics, Circuits and Systems*, pages 1272–1275. IEEE, 2006.
- [5] Beyond Dreams Consortium. *Beyond Dreams (Design Refinement of Embedded Analogue and Mixed-Signal Systems)*, 2008-2011. projects.eas.iis.fraunhofer.de/beyonddreams.
- [6] T. Blochwitz et al. The functional mockup interface for tool independent exchange of simulation models. In *8th Int. Modelica Conference, Dresden, Germany*, pages 105–114, 2011.

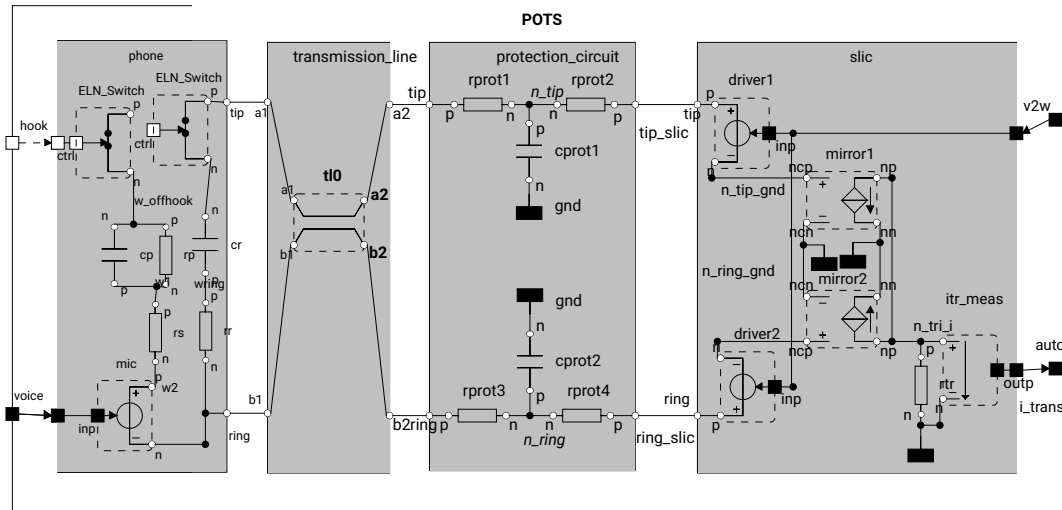


Figure 7: ELN diagram of the four POTS circuits

- [7] A. I. Concepcion and B. P. Zeigler. DEVS formalism: A framework for hierarchical model development. *IEEE Trans. on Software Engineering*, 14(2):228–241, 1988.
- [8] R. Cortés Porto, D. Genius, and L. Apvrille. Handling causality and schedulability when designing and prototyping cyber-physical systems. *Software and Systems Modeling*, pages 1–17, 2021.
- [9] A. Davare, D. Densmore, T. Meyerowitz, A. Pinto, A. Sangiovanni-Vincentelli, G. Yang, H. Zeng, and Q. Zhu. A next-generation design framework for platform-based design. In *DVCon*, 2007.
- [10] P. de Saqui-Sannes, L. Apvrille, and R. Vingerhoeds. Checking SysML Models Against Safety and Security Properties. *Journal of Aerospace Information Systems*, pages 1 – 13, Nov. 2021.
- [11] S. Demathieu, F. Thomas, C. André, S. Gérard, and F. Terrier. First experiments using the uml profile for marte. pages 50–57. IEEE, 2008.
- [12] K. Einwich. *SystemC AMS PoC2.1 Library*, COSEDA, Dresden, 2016.
- [13] J. S. Fitzgerald, P. G. Larsen, K. G. Pierce, and M. H. G. Verhoef. A formal approach to collaborative modelling and co-simulation for embedded systems. *Mathematical Structures in Computer Science*, 23(4):726–750, 2013.
- [14] P. Fritzson and V. Engelson. Modelica—a unified object-oriented language for system modeling and simulation. In *European Conf. on Object-Oriented Programming*, pages 67–90. Springer, 1998.
- [15] D. Genius and L. Apvrille. Hierarchical design of cyber-physical systems. In *Modelsward*, 2023.
- [16] D. Genius, R. Cortés Porto, L. Apvrille, and F. Pêcheux. A tool for high-level modeling of analog/mixed signal embedded systems. In *MODELWARD*, 2019.
- [17] F. Herrera and E. Villar. A framework for heterogeneous specification and design of electronic embedded systems in systemc. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 12(3):22, 2007.
- [18] IEEE. *SystemC*. IEEE Standard 1666-2011, 2011.
- [19] A. Jantsch. *Modeling Embedded Systems and SoC's: Concurrency and Time in Models of Computation*. Elsevier, 2003.
- [20] E. A. Lee. Disciplined heterogeneous modeling. In D. Petriu, N. Rouquette, and O. Haugen, editors, *Proceedings of the ACM/IEEE 13th International Conference on Model Driven Engineering, Languages, and Systems (MODELS)*, pages 273–287. LNCS 6395, Springer-Verlag, Oct. 2010.
- [21] S. H. A. Niaki, M. K. Jakobsen, T. Sulonen, and I. Sander. Formal heterogeneous system modeling with systemc. In *Specification and Design Languages (FDL)*, pages 160–167. IEEE, 2012.
- [22] H. D. Patel and S. K. Shukla. Towards a heterogeneous simulation kernel for system-level models: a systemc kernel for SDF models. *TCAD*, 24(8):1261–1271, 2005.
- [23] G. Pedroza, D. Knorreck, and L. Apvrille. 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, May 2011.
- [24] Project consortium. *Heterogeneous Inception*, 2012–2015. <https://www-soc.lip6.fr/trac/hinception>.
- [25] SocLib consortium. *The SoCLib project: An Integrated System-on-Chip Modelling and Simulation Platform*, 2003.
- [26] A. Vachoux, C. Grimm, and K. Einwich. Analog and mixed signal modelling with SystemC-AMS. In *ISCAS (3)*, pages 914–917. IEEE, 2003.
- [27] C. Zhao and T. J. Kazmierski. An extension to SystemC-A to support mixed-technology systems with distributed components. In *DATE*, pages 1–6. IEEE, 2011.
- [28] J. Zhu, I. Sander, and A. Jantsch. Hetmoc: Heterogeneous modelling in systemc. In *Specification & Design Languages (FDL 2010), 2010 Forum on*, pages 1–6. IET, 2010.