Automatic Abstraction of Multi-Discipline Analog Models for Efficient Functional Simulation

Enrico Fraccaroli, Michele Lora and Franco Fummi
Department of Computer Science, University of Verona, Italy, Email: `name.surname@univr.it`

***Abstract*—Multi-discipline components introduce problems when inserted within virtual platforms of Smart Systems for functional validation. This paper lists the most common emerging problems and it proposes a set of solutions to them. It presents a set of techniques, unified in an automatic abstraction methodology, useful to achieve fast analog mixed-signal simulation even when different physical disciplines and modeling styles are combined into a single analog model. The paper makes use of a complex case study.It deals with multiple-discipline descriptions, non-electrical conservative models, non-linear equation systems, and mixed time/frequency domain models. The original component behavior has been modeled in Verilog-AMS by using electrical, mechanical and kinematic equations. Then, it has been abstracted and integrated within a virtual platform of a mixed-signal smart system for efficient functional simulation.**

I. Introduction

Smart devices require to integrate many different technologies in order to perform sensing, actuation, computation and communication. For this reason, they are usually designed by integrating heterogeneous components, modeled by using paradigms from different design domains [1]. Digital HW coexists with embedded SW to compose the computational platform of the device. At the same time, many analog and physical devices are integrated to perform sensing, actuation and communication. Models of such components are based on different physical disciplines (*e.g.*, electrical, kinematic, magnetic, thermal, etc.) and they usually require discipline-specific simulators to study their behavior.

In order to analyze the functionality of the entire system, all its parts need to be simulated together. However, this implies the need of integrating many simulators for the different disciplines involved, other than for the digital HW and SW parts of the system. Virtual platforms [2] have proved to be powerful solutions to perform such a task. However, they mostly focus on digital components and prototyping of analog HW does not allow integration within the simulation environments [3]–[5] or do not deal with multiple disciplines [6]. Thus, multi-discipline analog devices are simulated by employing multiple ad-hoc simulators, integrated through co-simulation interfaces introducing synchronization overhead.

This paper tries to overcome such issues by employing automatic abstraction of multi-discipline analog models. These models are automatically translated into a unifying language and the result can be then integrated within a functional description. Figure 1 gives a pictorial representation of such a flow. The guiding idea is to preserve only the input/output values characterizing the behavior of the multi-discipline analog device, while abstracting away as much internal details as possible. As such, the methodology provides an abstraction flow for analog models that allows easy integration within system-level executable models of heterogeneous smart systems.

II. Multidiscipline analog modeling

Verilog-AMS and VHDL-AMS are extensions of traditional Hardware Description Languages (HDL) to model analog and mixed-signal behaviors. They both allow to model multi-discipline systems and they differ only for syntactic details [7]. This work focuses on Verilog-AMS models, but it is **easily applicable to VHDL-AMS as well**.

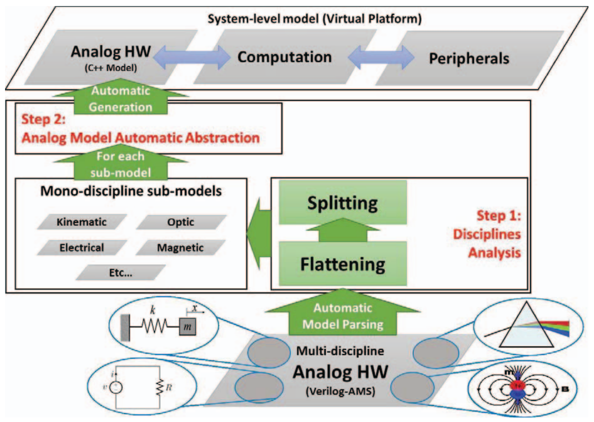

Figure 1: Application scenario of the proposed methodology.

In Verilog-AMS, systems behavior is described by means of a set of relations between physical values on analog nodes. Each node is associated with two quantities: *potential* and *flow*. Nodes and quantities of the system are defined in terms of *natures* and *disciplines* to model diverse domains [8]. A discipline is the type associated to an analog node (*e.g.*, electric, magnetic, kinematic, *etc.*). A discipline is associated to at most two natures. Each nature is a collection of attributes shared by the signals using it. A discipline specifying both *flow* and *potential* natures is said to be *conservative*, otherwise it is said to be a *signal-flow*. Table I summarizes the set of disciplines defined by the Verilog-AMS standard. The standard also allows to define any custom nature or discipline.

The system behavior is described by means of *contribution statements* (denoted with <+) that relates flow and potential of nodes by using differential and algebraic equations.

Verilog-AMS execution semantic mixes discrete-event and SPICE-based tools, making simulation very accurate but poorly effective [1]. A first attempt to integrate analog components into virtual platform of smart systems has been proposed in [6]. It exploits an automatic abstraction procedure similar to the one used by this work, however it supports only simple devices expressed by using only the electrical discipline and linear equations. While guided by the same philosophy this work aims at supporting more complex models, based on heterogeneous physical disciplines and modeling styles.

III. Problem Statement and Objectives

The explanation of the methodology is paired with its application to a complex case study: the model of a micro-mirror model used in a pico-projector [9]. The design is kindly furnished by an industrial partner, in the context of a European

Table I: Disciplines defined by the Verilog-AMS standard [8].

Name	Potential	Flow
logic	–	–
electrical	Voltage	Current
voltage	Voltage	–
current	–	Current
magnetic	Magneto_Motive_Force	Flux
thermal	Temperature	Power
kinematic	Position	Force
kinematic_v	Velocity	Force
rotational	Angle	Angular_Force
rotational_omega	Angular_Velocity	Angular_Force

Project[1]. Figure 2 depicts the analog system components moving the micro-mirror and their characteristics:

- The *Digital board* implements the control logic for the mirror, and it generates the clock signal for the *digital-to-analog* (DAC) and *analog-to-digital converters* (ADC).
- Two *DAC* convert the received signals into voltage values.
- Each DAC generates a voltage value that is used as input for a series of two *Operational Amplifiers* (OpAmp) parametrized to specify a certain threshold.
- The *Vertical Mirror* models the dynamic of one of the two projector mirrors. It uses the *electrical* discipline as interface, while internally it uses the standard *rotational_velocity* and a used-defined *rotational_acceleration* which relates angular force and acceleration. It is non-linear: it employs polynomial functions described with the *logic* discipline.
- The *Transimpedance amplifier* (TIA) converts the current flow generated by the mirror into a voltage value. It is piecewise-linear: it sets a threshold to the amplification specified by using the *logic* discipline.
- The *ADC* converts the voltage generated by the TIA into a digital value for the board.

The model has been implemented in Verilog-AMS and it is intended to be used to prototype the SW removing the *ripple effect* affecting the pico-projector. Simulation of the component requires a co-simulation environment made by a discrete-event paired with a SPICE-based simulator. Such a simulation framework suffers of synchronization overhead caused by the interfaces necessary to make simulators co-exist.

A virtual platform able to emulate the entire system will require to introduce even more co-simulation interfaces, thus introducing further overhead. The alternative approach is to exploits automatic abstraction in order to reconcile the multidisciplinary platform into a unique model. This allows to remove the overhead for synchronization between different simulators, and further speed-up the simulation.

[1]SMAC (SMArt system Co-design), FP7-ICT-2011-7-288827

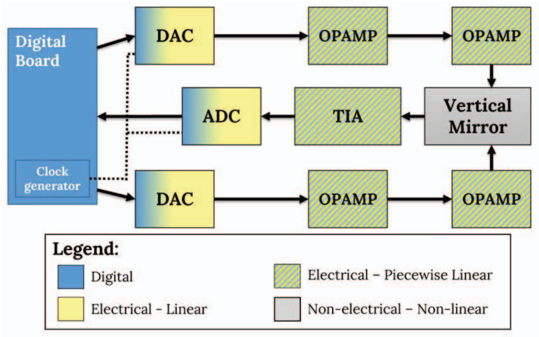

Figure 2: Structure of the Vertical Mirror system.

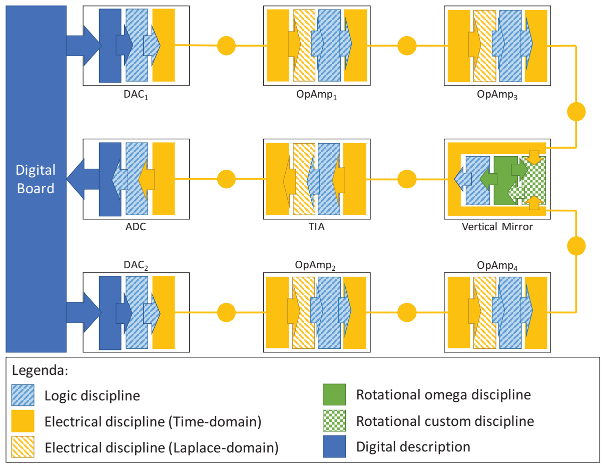
Digital Board
DAC1
OpAmp1
OpAmp3
ADC
TIA
Vertical Mirror
DAC2
OpAmp2
OpAmp4
Legenda:
Logic discipline
Electrical discipline (Time-domain)
Electrical discipline (Laplace-domain)
Rotational omega discipline
Rotational custom discipline
Digital description

Figure 3: Disciplines analysis for the pico-projector case study.

IV. Multi-discipline models abstraction

System-level functional simulation is usually employed to evaluate the behavior of an entire system as a whole. As mentioned in Section I, the idea is to preserve only the input/output relations and, as such, it is necessary to identify what can be considered as input and output in an analog model. The **inputs** are those signals used to interface the device to the other components (*e.g.*, DAC), together with the set of continuous values coming from the environment sensing. The **otputs** are those signals that interface the device with the platform (*e.g.*, ADC), and the physical signals representing actuation to the environment. Identifying them is not trivial and tightly related to the purpose of the model.

As shown in Figure 1, the abstraction flow starts from a multi-discipline Verilog-AMS description to generate a functionally equivalent C++ executable model.

A. Discipline analysis

The Verilog-AMS model is parsed, mapped into a XML-based representation and provided to this step that identifies the disciplines used in the system. The *flattening* process analyzes the boundaries between sub-models of a hierarchical model and merges them into a single non-hierarchical one. It comprises three steps: *1)* Each input/output port of every sub-model is replaced by an intermediate node with the same discipline of the original ports. *2)* Each variable defined inside a sub-model is imported into the flattened description by assigning to it a fresh name. *3)* The sub-model behavior is copied in the flattened model. The *Splitting* process analyzes the flattened description and splits it in sub-models made of parts of the system belonging to a single discipline.

Consider Opamp$_1$ and Opamp$_2$ in Figure 3. It is necessary to consider the composition of their electrical networks as a unique system of equations in order to preserve the correct behavior of the system, since the two net are connected by means of a conservative node. The issues just discussed are highlighted by applying the flattening step and solved by applying the splitting procedure. Figure 3 depicts the result of applying the flattening and splitting processes to the case study. The behavior of each sub-model has been partitioned based on the discipline and the modeling style.

B. Analog model automatic abstraction

This step is inspired by the automatic state-of-the-art abstraction methodologies for linear electrical circuits mentioned in Section II. The abstraction step comprises four sub-steps:

- *Acquisition*: all the equations specified by the sub-model are collected and used to infer the structure of the system. Section IV-C extends this step to deal with functions expressed in the Laplace domain.
- *Enrichment*: every discipline is carrying its own set of implicit relations. The information gathered in the previous step is used to infer the set of implicit relations among the physical quantities. Section IV-D will show how this step applies to non-electrical disciplines.
- *Assemble*: for each output, an abstract syntax tree is built to represent the relations connecting it to the inputs.
- *Solve*: algebraic loops introduced by previous steps are broken using a symbolic solver, that also performs arithmetic and algebraic simplifications.

The *Automatic Generation* step composes the model resulting from the abstraction of mono-discipline sub-models, managing non-linearities (Section IV-E). Finally, the model resulting from the composition is translated in C++.

C. Dealing with the frequency domain

Linear time-invariant systems may be expressed by using their transfer function-based representation in the Laplace domain. If an equation is expressed in the Laplace domain, its inverse transform is symbolically computed to retrieve the corresponding time-domain representation. This allows to move resolution overhead from simulation- to generation-time, and it is done by transforming the transfer function into an equivalent time-domain equation. As such, all the equations composing the mono-discipline sub-model are expressed in the time-domain and they can be solved all together.

A Laplace domain representation is used to model the OpAmp and TIA components inside the pico-projector and thus, they undergo the inverse transformation. Listing 1 shows the Verilog-AMS code of the electrical sub-model spread among the DAC_1 and the OpAmp_1 components. In details, Listing 1a shows the original code using a Laplace function to describe the behavior of the value `A_opamp_1`. While, Listing 1b shows the Verilog-AMS code describing the model specified in the time-domain after the inverse transformation.

D. Generalized approach to conservative disciplines

This step takes care of retrieving the set of equations implied by the application of energy conservation laws. The generalized Kirchhoff's Potential Law (KPL) and Kirchhoff's Flow Law (KFL) are considered as in the Verilog-AMS standard [8]. Two new sets of equations are introduced that assure that the sum of all flows entering and exiting each node and the sum of the branches potentials around each loop are zero by using respectively the KFL and KPL. The generated equations are used to enrich the set of equations used to select those linking inputs and outputs of the sub-model. The vertical mirror of the pico-projector make use of four disciplines: logic, electrical, rotational_omega and the used-define rotational_alpha. The mono-discipline sub-models identified by the previous step are analyzed separately in order to infer the implicit relations necessary for compliance with the energy conservation laws.

```
V(out_dac_1)  <+ value_dac_1;
I(in_opamp_1) <+    (V(in_opamp_1) / rin_opamp_1) +
                 ddt(V(in_opamp_1) * cin_opamp_1);
A_opamp_1 = laplace_nd(V(in_opamp_1) * C1, 1, {1, C2});
```

(a)

```
V(out_dac_1)  <+ value_dac_1;
I(in_opamp_1) <+    (V(in_opamp_1) / rin_opamp_1) +
                 ddt(V(in_opamp_1) * cin_opamp_1);
A_opamp_1 = C1 * V(in_opamp_1) - C2 * ddt(A_opamp_1);
```

(b)

Listing 1: Manipulation of an electrical sub-model of the pico-projector, to remove the Laplace-domain specification.

E. Dealing with non-linear behavior

Verilog-AMS provides different ways to introduce non-linearities inside an analog model: *1)* Logic discipline can be used to model a piecewise-linear behavior by using control flow statements. *2)* Contribution statements on conservative nodes may express non-linear functions on their right-hand side. *3)* Definition of any kind of mathematical analog function. This work addresses these three issues separately.

Verilog-AMS standard forbids to use analog access functions inside a logic block [8], as consequence real variables are introduced to interface the Logic discipline with conservative or signal-flow ones. These variables are the only ones used within partitions described by using the logic discipline, thus making these partitions a sequence of procedural statements. This work exploits such a feature. It reproduces non-linear behaviors introduced by the logic discipline by translating the procedural logic block into an equivalent C++ function.

Consider the portion of the OpAmp component shown in Listing 2a. Variables `A` and `C` are used to interface logic and electrical discipline. A conditional statement on `A` introduces a non-linear behavior. Listing 2b shows the corresponding C++ implementation. The function `logic_block_1` implements the behavior specified using the logic discipline in the Verilog-AMS code, while the `opamp` function implements the behavior specified by the fragment of code shown in Listing 2a. Note that this kind of behavior is often used to interface different disciplines. Thus, the application of the presented strategy must be applied when aggregating the abstracted mono-disciplinary sub-models after the abstraction step.

If a contribution statement defines a non-linear behavior, then it has to be considered as a signal-flow equation. Thus, it is simply parsed and inserted into the equations set built during the acquisition step. Finally, if the non-linear behavior is introduced by an analog function, this is reproduced by its equivalent C++ implementation. This recalls the strategy adopted for non-linear dynamics induced by interfacing logic and conservative disciplines. An example of this is the polynomial analog functions used to specify the behavior of the vertical mirror. The application of this strategy to the second order polynomial function written in Verilog-AMS shown in Listing 3a yields the C++ functions shown in Listing 3b.

V. Experimental Results

The approach has been implemented in a prototypical tool on top of HIFSuite [10] and applied to a set of Verilog-AMS

```
A = V(in) * gain;
if( A > threshold ) C = threshold;
else C = A;
I(out) <+ C;
```

(a)

```
void logic_block_1(double & A, double & C) {
    if ( A > threshold ) C = threshold;
    else C = A;
}
void opamp() {
    A = V_in * gain;
    logic_block_1(A, C);
    V_out = C;
}
```

(b)

Listing 2: OpAmp component of the pico-projector.

```
analog function real polyfit_2;
    input a,b,c,x; real  a,b,c,x;
    polyfit_2 = a + b * pow(x,1) + c * pow(x,2);
endfunction;
```

(a)

```
double polyfit_2(double a,b,c,x) {
    return a + b * pow(x,1) + c * pow(x,2);
}
```

(b)

Listing 3: Polynomial functions used by the vertical mirror.

Table II: Characteristics of the selected benchmarks.

	Used Disciplines									
Benchmark	(1)	(2)	(3)	(4)	(5)	(6)	Domain	LoC	# Vars.	# Eq.
Low Pass Filter		✓					Laplace	26	2	1
Magnetic Winding		✓		✓			Time	25	10	2
Motor		✓	✓				Time	41	12	2
Bouncing Ball	✓					✓	Time	64	5	5
Mass-Spring-Damper					✓		Time	98	8	4
Pico-Projector	✓	✓	✓			✓	Time& Laplace	546	41	46

analog models. Table II summarizes the main characteristics of the used benchmarks. Considered models are expressed by using the (1) Logic, (2) Electrical, (3) Rotational, (4) Magnetic, (5) Kinematic, and (6) User defined disciplines. For each model, it is reported the specification domain (*i.e.*, Laplace- or Time-domain), the number of lines of code (LoC) of the original description, the number of continuous variables and the number of equations. The first five benchmarks come from a repository of open-source models[2]. They are used to show the applicability of the methodology to different physical disciplines and modeling styles. The pico-projector is an example of realistic case of a complex device, used to show the methodology effectiveness.

Table III reports the time need to simulate 1 second with a timestep of 50ns obtained on the benchmarks. The achieved speed-up is reported. The benchmarks have been simulated both alone and integrated within a virtual platform for smart devices: the SMAC Open Source Test Case (OSTC) [9]. It includes a CPU and a memory to perform computation, both connected to a set of digital peripherals. The pico-projector is connected to the peripheral bus and acts according to the commands computed by the SW running on the CPU. The same testbenches have been applied to both the Verilog-AMS and the C++ implementations. A 64-bit Linux machine with 16 GB and six 3.50GHz CPU cores has been used for simulation. A commercial SPICE-based simulator has been used to simulate the original mixed-signal virtual platform.

The comparison between the simulations of the components without the platform highlights the impact of the abstraction. The open-source benchmarks achieve up-to three order of magnitude speed-up, while the pico-projector reaches a 80x speed-up. This difference is due to the structure of the models. The Open-source benchmarks have a more coarse partition of the system that allows a better simplifications during the sub-modules abstraction. The pico-projector partition, instead is really fine grained as depicted in Figure 3, leading to small portions more difficult to be optimized. These small parts must be reconnected to each other with a communication mechanisms which introduces overhead. The pico-projector also requires to solve polynomial and complex non-linear functions at run-time that cannot be removed if the original behavior of the abstracted component must be preserved.

Components are simulated after being integrated in the virtual platform to show the positive impact of abstraction on functional simulation. The achieved speed-up ranges from 24x to 36x. It is worth noticing the impact of the state-of-the-art abstraction for digital systems coupled with the one obtained by applying the proposed methodology. The overhead introduced by the original digital components may be estimated in around 1500 seconds, while it decreases to 150 seconds for the C++ implementation. As such, abstraction of digital components provides around 10x speed-up while, the proposed automatic abstraction for analog models provides 80x speed-up in the worst case. Considering the increasing amount of sensors and actuators in nowadays systems, analog models abstraction may be crucial for future design flows.

Table III: Simulation times needed to simulate the benchmarks.

	Verilog-AMS (Original model)		C++ (Automatic abstraction)			
	Comp.	Platform	Component		Platform	
Benchmark	time (s)	time (s)	time (s)	speed-up (x)	time (s)	speed-up (x)
Low Pass Filter	3493.49	5268.20	2.07	1687.67	153.46	34.32
Magnetic Winding	3477.08	5197.33	1.89	1839.72	154.43	33.65
Motor	2281.46	3762.47	2.66	857.69	151.68	24.80
Bouncing Ball	2173.82	3535.51	0.85	2557.43	149.77	23.60
Mass-Spring-Damper	3653.10	5345.86	1.57	2326.81	148.71	35.94
Pico-Projector	9067.63	10156.27	112.43	80.65	276.62	36.71

Accuracy has been measured by using the Normalized Root Mean Square Error (NRMSE) on the outputs of the C++ simulations *w.r.t.* the ones generated simulating the original models. The NRMSE ranges from 10^{-5} to 10^{-7}.

VI. Conclusions

This paper proposed a set of solutions, unified within a single methodology, to automatically abstract multi-discipline analog models. This aims at achieving efficient functional simulation once the abstracted models are integrated into mixed-signal virtual platforms. The set of presented experiments showed the approach effectiveness, while an industrial case study highlights the applicability to complex smart systems.

VII. Acknowledgements

We would like to acknowledge Salvatore Rinaudo and Giuliana Gangemi of STMicroelectronics for the effective contribution, under the EU SMAC project, in developing the smart system platforms used in this work.

References

[1] F. Fummi *et al.*, "Moving from Co-Simulation to Simulation for Effective Smart Systems Design," in *Proc. of ACM/IEEE DATE 2014*, pp. 1–4.
[2] Imperas Software, "OVP - Open Virtual Platforms," www.ovpworld.org.
[3] F. Cenni *et al.*, "Behavioral modeling of a CMOS video sensor platform using SystemC AMS/TLM," in *Proc. of IEEE/ECSI FDL 2011*, pp. 1–6.
[4] C. B. Aoun *et al.*, "Pre-simulation elaboration of heterogeneous systems: The SystemC multi-disciplinary virtual prototyping approach," in *Proc. of IEEE SAMOS 2015*. IEEE, pp. 278–285.
[5] M. D. Alassir *et al.*, "Modeling I2C Communication Between SoCs with SystemC-AMS," in *Proc. of IEEE ISIE*, 2007, pp. 1412–1417.
[6] M. Lora *et al.*, "Virtual prototyping of smart systems through automatic abstraction and mixed-signal scheduling," in *Proc. of IEEE/ACM ASP-DAC 2017*, pp. 1–6.
[7] F. Pecheux *et al.*, "VHDL-AMS and Verilog-AMS as alternative hardware description languages for efficient modeling of multidiscipline systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 2, pp. 204–225, feb 2005.
[8] Accellera, "Verilog-AMS Language Reference Manual."
[9] I. Blanco *et al.*, "Smart system case studies," in *Smart Systems Integration and Simulation*. Springer, 2016, pp. 195–227.
[10] N. Bombieri, M. Ferrari, F. Fummi *et al.*, "HIFSuite: tools for HDL code conversion and manipulation," *EURASIP Journal on Embedded Systems*, pp. 1–20, 2010.