Analog and Mixed Signal Modelling with SystemC-AMS

Alain Vachoux, EPFL Lausanne
Christoph Grimm, University Frankfurt
Karsten Einwich Fraunhofer, IIS/EAS Dresden

Abstract

SystemC will become more and more important for the design of digital circuits from the specification down to the RT-Level. Complex systems often contain analog components. This paper introduces concepts for the extension of the SystemC methodology for the specification and design of analog and mixed signal systems. The concepts will be illustrated on a telecommunication system including digital hard- and software, analog filter and an analog environment.

1. Motivation

SystemC supports a wide range of Models of Computation (MoC) and is very well suited for the design and refinement of HW/SW-systems from functional down to register transfer level. However, for a broad range of applications the digital parts and algorithms interact with analog parts and the continuous-time environment. Due to the complexity of these interactions and the importance of the analog parts in the global system's behavior, it is essential to include the analog parts in the design process of an **A**nalog and **M**ixed **S**ignal system.

Simulation performance is therefore very crucial - especially for the analog parts that usually require more detailed models than the digital parts. Thus, different and specialized analog simulators must be introduced to permit the use of the most efficient simulator for the considered application and level of abstraction. In this paper, we describe a design methodology based on analog and mixed-signal extensions of SystemC, .a.k.a. SystemC-AMS. We also illustrate the methodology with a signal processing dominated application example.

2. SystemC Overview

SystemC is a design language for discrete-time (digital) systems. This language is an application of the object oriented programming language C++ for system design, so SystemC descriptions can be compiled, executed and debugged using standard C++ tools. In comparison to languages like VHDL or Verilog, SystemC supports an arbitrary number of Models of Computation (MoC) which allows an efficient development of executable specifications at high abstraction levels and an order of magnitude faster simulation for abstract models.

The SystemC 2.0 [1] methodology combines features of existing HDL's, object oriented techniques and new methodologies for the design and refinement of digital hardware and software systems. This methodology is strongly inspired by the communication model introduced by Gajski [6]. In this methodology, *modules*, which consists of other modules or algorithms (sequential assignments) implemented in methods, communicates via *channels*. A set of methods for communication is specified in an *interface*. These methods are implemented in a channel. Modules can call methods of a channel, and events in a channel can activate methods in a module connected to the channel. This concept is generic enough to describe systems using various models of computation, including static and dynamic multirate dataflow networks, Kahn process networks, communicating sequential processes, and discrete events (the MoC of Verilog and VHDL). We call such systems *discrete systems*. Predefined description styles allow, e.g. at RT-level, a description similar to what can be done in VHDL or Verilog. However due to the generic concept those models can be easily connected to models described in other styles (like dataflow or abstract software models) or can be re-used in another context.

Numerous tools supporting SystemC from the specification level down to register transfer level (RTL) are now available on the market and SystemC RTL descriptions can be synthesized. Thus a seamless design flow from the early design stages to the circuit level is becoming available.

3. SystemC-AMS Methodology

SystemC is used for system-level design tasks, such as the modeling and refinement of hardware/software – systems. In such a context, analog models are used most notably for the following tasks:

Executable Specification: Analog models are often used as an executable specification of signal processing functions. Currently, interactive tools with a graphical interface such as Matlab/Simulink are used for such tasks.

Behavioral Modeling: In the design of analog systems, there is always a large "bottom-up" part. Behavioral modelling, or macromodelling, of analog netlists allows us the simulation of analog circuits in a reasonable time.

Co-Simulation with Environment: On system level, the analog (continuous-time) environment is co-simulated with the embedded system. This allows a rough validation of an executable specification. Furthermore, many designs can be only validated with such a co-simulation.

In difference to digital systems, analog systems often combine different physical domains and are very application-specific. Therefore, concrete applications must be considered.

In telecommunication and multimedia applications, the modeling of signal processing functions is dominant. These systems are mostly oversampled using constant time steps. The system is modeled by a block diagram with directed signal flow. The blocks are described by linear transfer functions and static non-linear functions. Often, such a system level description is combined with linear networks used for (macro)modeling the system environment. For RF applications, the ability to simulate the baseband behaviour is required.

In the automotive domain, analog and mixed-signal systems are often non-linear systems, and usually embrace multiple physical domains (electrical, mechanical, fluidic, etc.). In difference to telecommunication and multimedia applications, control systems in the automotive domain often exhibit very different time constants ("stiff systems"). Nevertheless, executable specifications and executable prototypes are also modeled as block diagrams with directed signal flow.

Although concrete requirements from the above mentioned application domains must be considered, SystemC-AMS must also be flexible and generic in order to support modeling, simulation and design in new application domains [4][5]. Furthermore, the requirements in the discussed application domains may seem partially contradictory. Rather simple block diagrams connected with directed signal flow and some external components seem to be sufficient for many applications on a high level of abstraction. However, some requirements can only be fulfilled by solutions which are specific for each application and level of abstraction considered. For example, a designer might want to use a dedicated circuit simulator, such as SPICE, for electrical parts and a dedicated simulator for mechanical systems for the precise simulation of mechanical components.

This situation can be handled in an *open and layered approach* such as the one offered by SystemC-AMS. These extensions to SystemC are used for the simulation of executable specifications, implemented as behavioral models, and the environment as far as possible. Furthermore, a synchronization mechanism permits the easy integration of additional simulators or solvers. These additional solvers can be specific for - maybe new - applications and levels of abstractions that are not covered by the "built-in" solvers (Figure 1).

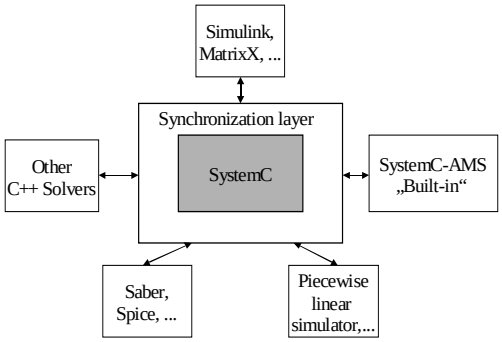


Figure: 1 Illustrates the discussed scenario.

SystemC-AMS must define clear interfaces between different layers, on which users or programmers can describe their models or add features to the simulator, depending on their application.

On top of the existing standard SystemC kernel, a *synchronization layer* provides methods to synchronize:

- Analog solvers and the discrete kernel of SystemC.
- Different-continuous-time solvers/MoCs.

For coupling analog solvers and the discrete kernel, the following approach is used. Before the first delta cycle of a time step is executed, all analog simulators are executed, reading the "old" discrete values and producing updated output signals, which are then used by the digital processes. Although this synchronization scheme is simple, it fits the requirements of system level better than more complex but much slower schemes found for example in VHDL-AMS.

The synchronization between analog solvers is handled via an abstract interface, which is implemented in the synchronization layer. The synchronization layer determines the points in time, at which analog and digital simulation are synchronized, and the order in which the analog solvers are executed. For the first prototype the synchronization is realized by static dataflow. Again, this synchronization is simple and fast. Note that one can also realize this interface by another synchronization scheme, since all communication and synchronization between analog solvers must only use the abstract interface methods.

On top of the synchronization layer, different analog solvers (*solver layer*) compute the behavior of analog blocks. Analog solvers are accessed from the upper layer via an interface, for example differential or algebraic equations in a machine-readable form, such as in a nodal matrix representation. A first prototype implements a solver for linear differential equations. Together with static nonlinear functions (which require no solver), this is sufficient for system level simulations, and results in a faster simulation than a nonlinear solver.

The *view layer* provides convenient user interfaces for the analog solvers. The view layer converts user views, for example netlists that are mapped to a set of differential or algebraic equations in the solver layer. The interaction of solver- and view layer can be compared to a model/view architecture, where one object provides an abstract data structure, and other objects provide graphical views of the data structure.

4. Application Example

As an illustration of modelling with SystemC-AMS, a signal processing dominated application is used. Such kind of application usually includes analog, digital filter, some analog linear elements, digital control algorithm and is characterized by over sampling. Thus on system level several description styles has to be combined.

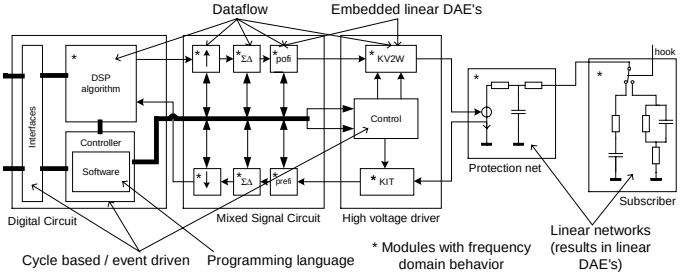


Figure: 2 Simplified block diagram of the application example

Figure 2 shows a strong simplified system view of a so-called Subscriber Line Interface and Codec Filter (SLICOFI) system

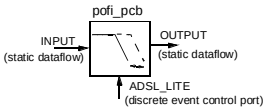
[7]. Such systems establish the connection between the analog subscriber line and the digital (e.g. PCM) transmission network. This system is realized by different IC's (chip-set) and external (analog) components. The chip-set includes a high-voltage line driver, analog filters, A/D- and D/A-converters, digital hardware filters, DSP-algorithms, control algorithms and interface algorithms. The new design challenge for such systems is data transmission in parallel with voice (ADSL).

During system design the external components and the environment, consisting of the subscriber and the subscriber-line, will be considered as linear analog networks. The high-voltage driver is modeled by unidirectional blocks (without feedback from the previous block). Thus, the static dataflow (SDF) Model of Computation can be used for block scheduling. Dynamics (e.g. poles and zeros of the amplifier) are modeled by using embedded linear DAE's. Nonlinearities may be included as static or as an black box model identified by measurements or circuit simulation. The analog filter (e.g. pre- and post-filter), D/A- and A/D-converter are modeled similarly. For the digital filter and DSP-algorithm dataflow blocks are used also. The control algorithm software is embedded in an event-driven digital model using a bus functional model. The interfaces are described at RT-Level.

Usually the design of such a system starts in the frequency domain as the frequency response. Small-signal signal to noise ratio estimations are also very important.

In the following, SystemC-AMS models for some blocks using a prototype implementation of the extensions are given. The **sca_** prefix and boldface words are used to denote AMS extensions. This prototype implementation is optimized for signal processing dominated applications and uses the static dataflow model of computation with constant time steps for synchronization. Non-linear static behavior, linear dynamic equations can be modeled using representations like transfer functions or state space equations. Additionally, a description of a conservative linear network consisting of pre-defined elements like resistors, capacitors and controlled sources can be described. Such a network is embedded in a dataflow block. Due to the high over sampling rates of the considered systems and the constant time steps, very simple and thus fast integration and synchronization algorithms can be used. For each dataflow block an optional frequency domain implementation can be added. For the network elements this implementations are predefined. This way the system can be modeled and simulated both in the frequency domain and the time domain.

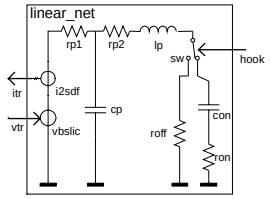
Figure 3 shows the description of a analog post-filter whose cutoff frequency can be selected by a control signal which is generated in the discrete event (classical SystemC) domain. A frequency domain implementation for the filter block is also given.




```
SCA_MODULE(pofi_pcb),sca_sdf_synchronization,
                     sca_ac_domain_view,
                     sca_behavioral_view
{
 sca_sdf_in<double>  INPUT;     //dataflow inport
 sca_de2sdf_in<bool> ADSL_LITE; //de to df inport
 sca_sdf_out<double> OUTPUT;    //dataflow outp.
 double FG0, FG1, K, h;         //parameters
 SCA_DAE_ID ltf_id0, ltf_id1;
 sca_vector<double> A0,A1, B0,B1, S;

 void sca_init() // initialization
{
  double wpre0;           double wpre1;
  wpre0=2.0*M_PI*FG0;     wpre1=2.0*M_PI*FG1;
  A0(0)=1.0;              A1(0)=1.0;
  A0(1)=1.41/wpre0;       A1(1)=1.41/wpre1;
  A0(2)=1.0/wpre0/wpre0; A1(2)=1.0/wpre1/wpre1;
  B0(0)=K;                B1(0)=K;
}

  void  sca_sig_proc() //time domain
  {
    if(ADSL_LITE)
      OUTPUT=LTF(A1,B1,S,ltf_id1,INPUT);
    else
       OUTPUT=LTF(A0,B0,S,ltf_id0,INPUT);
  }

  void sca_ac_domain() //frequency domain
  {
   if(ADSL_LITE)
      SCA_AC(OUTPUT)=
              LTF(A1,B1,S,ltf_id1,SCA_AC(INPUT));
   else
      SCA_AC(OUTPUT)=
              LTF(A0,B0,S,ltf_id0,SCA_AC(INPUT));
  }
  SCA_CTOR(pofi_pcb){}
};
```

Figure: 3 Description of an analog switched post-filter

Figure 4 shows the description of a linear network which can be switched by a control signal. An input voltage is provided by a dataflow block and a current is generated to a dataflow outport.

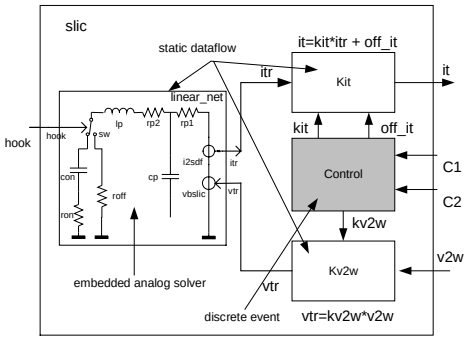



```
SCA_MODULE(linear_net),sca_sdf_synchronization,
                       sca_linear_netlist_view
{
 sca_sdf_in<double> vtr;  //data flow inport
 sc_in<bool>        hook; //discrete event inport
 sca_sdf_out<double> itr; //dataflow outport

 sca_node w1, w2, w3, w4, w5, w6, w7;
 sca_reference_node gnd;
```

```
sca_r      *rp1, *rp2, *roff, *ron;
sca_c      *cp, *con;
sca_l      *lp;
sca_sw_tog *sw;
sca_i2sdf  *i2sdf;

SCA_CTOR(linear_net) {
   vbslic =new sca_vsdf("vbslic",w1,gnd,vtr);
   rp1    =new sca_r("rp1",w4,w2,60.0);
   rp2    =new sca_r("rp2",w2,w3,40.0);
   cp     =new sca_c("cp",w2,gnd,1e-12);
   lp     =new sca_l("lp",w3,w4,1e-3);
   sw     =new sca_sw_tog("sw",w4,w5,w6,hook);
   roff   =new sca_r("roff",w5,gnd,600.0);
   ron    =new sca_r("ron",w5,w7,1e3);
   con    =new sca_c("con",w7,gnd,1e-6);
   i2sdf  =new sca_i2sdf("i2sdf",w1,w4,itr);
}};
```

Figure: 4 Description of a linear (conservative) net

Figure 5 gives an example for an hierarchical description which embraces multiple Models of Computations.




```
SC_MODULE(slic)
{
    sca_sdf_in<double>  V2W;   //dataflow outport
    sca_sdf_out<double> IT;    //dataflow inport
    //discrete event (control) inports
    sc_in<three_level> C1, C2;
    sc_in<bool>        hook;

    //discrete event signals
    sc_signal<double> kv2w_s, off_s, kit_s;
    //static dataflow signals
    sca_sdf_signal<double> itr, vtr;
    //discrete event primitive
    slic_control *control;
    kit          *kit1;      //dataflow primitive
    kv2w         *kv2w1;     //dataflow primitive
    //dataflow block with embedded linear net
    linear_net   *front_end;

SC_CTOR(slic)   //netlist
{
     control=new slic_control("control");
        control->c1(C1);
        control->c2(C2);
        control->KV2W(kv2w_s);
        control->OFF_DC(off_s);
        control->KIT(kit_s);
     kit1=new kit("kit1");
       kit1->inp(itr);
       kit1->outp(IT);
       kit1->gain_control(kit_s);
     kv2w1=new kv2w("kv2w1");
       kv2w1->inp(V2W);
       kv2w1->outp(vtr);
       kv2w1->gain_control(kv2w_s);
       kv2w1->off(off_s);
     front_end= new linear_net("front_end");
       front_end->vtr(V2W);
       front_end->itr(itr);
       front_end->hook(hook);
}};
```

Figure: 5 Description of a hierarchical block including different Modles of Computation

5. Conclusion

In this paper we presented the main aspects of a first version of the SystemC-AMS language and model examples using this language. The AMS extensions are first designed to efficiently support signal processing dominated applications without the need to use sophisticated continuous-time simulators. Also, the synchronization with the discrete-time kernel of SystemC is kept as simple as possible. Future versions of the SystemC-AMS language will include better support for conservative systems and systems with arbitrary nonlinear behaviors and a more generic synchronization layer.

6. REFERENCES

[1] An Introduction to System-Level Modeling in SystemC 2.0. Technical report of the Open SystemC Initiative, 2001. http://www.systemc.org/technical_papers.html

[2] Karsten Einwich, Christoph Clauss, Gerhard Noessing, Peter Schwarz, and Herbert Zojer: “SystemC Extensions for Mixed-Signal System Design”. Proc. Forum on Design Languages (FDL'01), Lyon, France, September 2001.

[3] Christoph Grimm, Peter Oehler, Christian Meise, Klaus Waldschmidt, and Wolfgang Fey. “AnalogSL: A Library for Modeling Analog Power Drivers with C++. In Proceedings of the Forum on Design Languages”, Lyon, France, September 2001.

[4] K. Einwich, Ch. Grimm, A. Vachoux, N. Martinez-Madrid, F. R. Moreno, Ch. Meise: “Analog Mixed Signal Extensions for SystemC”. White paper of the OSCI SystemC-AMS Working Group.

[5] G. Nössing, K. Einwich, C. Clauss, P. Schwarz: “SystemC and Mixed-Signal – Simulation Concepts”, in Proc. 4th European SystemC Users Group Meeting, Copenhagen, Denmark, October 2001.

[6] D. D. Gajski, J. Zhu, R. Dömer, A. Gerstlauer, S. Zhao: “SpecC Specification Language and Methodology”, Kluwer Academic Publisher 2000

[7] B. Zojer, R. Koban, J. Pichler, G. Paoli: “A Broadband High-Voltage SLIC for a Splitter- and Transformerless Combined ADSL-Lite/POTS Linecard”, IEEE J. Solid-State Circuits, vol. 35, pp 1976-1987, Dec. 2000.