Application of SystemC/SystemC-AMS for the Specification of a Complex Wired Telecommunication System

Karsten Einwich
Fraunhofer IIS/EAS Dresden

Abstract
Recently a first version of a SystemC-AMS library is public available. This library is an extension of SystemC which allows modeling of analogue mixed signal behavior in addition to the digital SystemC modeling features. This paper presents the application of this extension library for the development of a specification and the system design of a complex chip-set for a classical voice codec – the interface between the old analogue telephone and the digital world. Using this application some modeling examples and methodologies will be discussed.

Introduction

The new circuit technologies allow the integration of high complex **A**nalogue **M**ixed **S**ignal (AMS) systems. However the development and the production fix costs are also increased significantly. This again increases the pressure to expanding the application areas of those high integrated chip-sets. Due analogue parts can not be shrunk like the digital part, there is a continuous attempt to shift analogue functionality to digital. As the consequence the interconnection between analogue and digital is increased rapidly and the AMS devices will become flexible programmable to cover a wide range of applications.
The growing complexity of digital integrated systems was one of the driving forces to develop SystemC as an executable specification and modeling language. SystemC allows the specification of digital hardware systems and the accompanying software. But the pure digital approach is not sufficient for a number of telecommunication and automotive problems. New developments extend SystemC to analogue and mixed-signal systems. These extensions aim on a SystemC-AMS that can be used for the specification of digital, analogue, and mixed-signal systems including software components on a high level of abstraction [1,2]. The definition of SystemC-AMS is under work and a first prototype is public available [4].
The paper presents the application of this prototype for developing a specification and a "golden" reference model for the implementation of a voice codec chip-set used in interfaces between the digital world and the analogue POTS (plain old telephone services). This analogue mixed signal system is characterized by a very high number of features and standards which are supported. Additional the interactions between the analogue environment, the analogue parts on the circuits and the digital hard- and software are very tight.
The features and requirements for those mixed-signal systems are usually not given in a way which is directly translatable into the chip-set specification. Furthermore the specification has to be developed under consideration of the different application scenarios. The chip-set is always part of a larger system. A challenge for specification development is the only partially availability of the "larger system" know how at the circuit provider and the variety of "larger systems" which must be supported. For the example system this "larger system" can vary from a classical central office up to a PC which supports internet telephony. Thus the specification must be evolved in iterations with the application scenarios and thus in discussions with the (potential) customer. An other problem for the specification and system design of complex mixed signal systems are the features and requirements which are mostly defined as properties of the same analogue components. For the example system the most have to do with the two wired phone line – finally with the voltage and current there. Thus requirements can be contradictory or looking forward to the implementation very expensive (area, power consumption, …). In such cases a solution can be the involvement of the

application scenarios. Very often slight changes in the definition of the requirements and features can make a specification efficient feasible. This additional degree of freedom will be more and more needed to manage the number of requirements from different sources and although stay in the cost, area and power consumption limits.
To achieve first silicon right it will become more and more essential to verify all (or as far as possible) application scenarios. In the implementation phase it is usually not any longer possible to verify such complete scenarios. The simulation effort allows mostly only the verification of subsystems with a limited and reduced set of stimuli. To assure although a proper working overall system a “golden” reference model will be more and more required.

Brief Introduction to the SystemC-AMS – Implementation Concept

SystemC-AMS is an extension of SystemC [5]. The goal is to enlarge the SystemC abilities and philosophy to analogue mixed signal specification and system design [6]. Thus concepts of SystemC like the generic Model of Computation (MoC) and a layered architecture were expanded to modeling of analogue behavior and the analogue-digital interactions. SystemC-AMS uses analogue to SystemC object oriented features like inheritance (derivation) and C++ specific features like templates.
The generic Model of Computation of SystemC is based on the communication and synchronization of processes. The basic language elements therefore processes, events, sensitivity and channels/interfaces. Based on this elements a couple of MoC’s like the VHDL/Verilog like discrete event or dataflow MoC are built.
However the generic MoC of SystemC allows only a limited modeling of analog behavior (in [2] are some examples for analogue modeling in SystemC presented). The main lacks are the missing ability to setup equation systems and the absence of analogue (equation) solver. Additional analogue solvers in general use there own time axis – usually a floating point time instead the integer time for digital (SystemC) simulators. The run of an analogue curve is continuous (may be assumed as linear between calculation time points) instead of the time discrete changes of digital waves. The different time axis and value run interpretation requires a special synchronization of the analogue and digital MoC’s.
Based on this consideration the layered concept in Figure 1was developed.

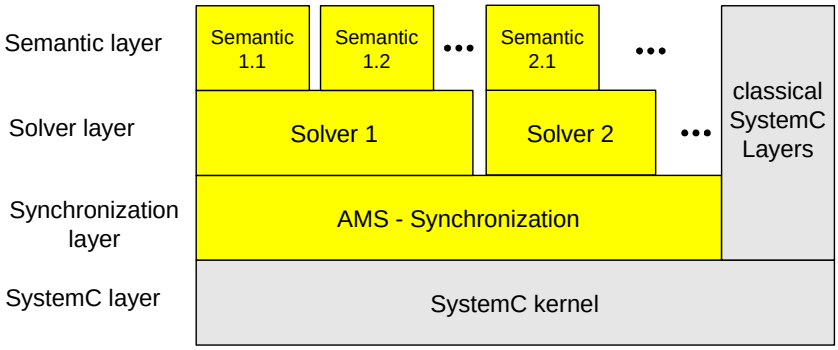


Figure 1 Layered Implementation Concept of SystemC-AMS

The semantic layer is responsible to setup the equation system(s) from a user (model) description. The user describes his primitive models (modules at the lowest hierarchy level which can not be further decomposed to sub modules) with derivations of sca_module, sca_port and sca_interfaces/sca_channel base classes, which again derived from the corresponding SystemC base classes (sc_module, sc_port, sc_interface/sc_channel). The used derivation of sca_module assures the assignment to a concrete semantic (for examples see next paragraph). As the result the semantic layer instantiates one or more solver. A solver is responsible to solve the equation system shipped by the semantic layer for certain time intervals given by the synchronization layer. For the synchronization layer a solver is

independently of his concrete implementation an abstract object which implements a unique synchronization interface. The synchronization layer is now responsible to activate (suspend/resume) the different solver and thus to synchronize the solver among each other and with the SystemC simulation kernel.
This concept allows a high degree of freedom for the algorithm which are realized inside a solver. Only the communication to other solvers and to the SystemC kernel is restricted by the abilities of the unique synchronization interface. Thus an always increasing time is assumed and the communication between solvers and the SystemC kernel must be directed.
The current implementation of the synchronization layer uses a synchronous dataflow [8,9] algorithm for solver activation. Thus the current implementation has for the communication of different solver instances additional the restrictions of a required one time step delay for loops of connected solvers, the communication of all connected solvers must take place to the same "global" time points and all solvers will be always activated at this time points and have to calculate (means have to produce at least one value at this time points) the corresponding time interval from the last time point to the current time.

SystemC-AMS for Signal processing dominated applications

One main idea of SystemC/SystemC-AMS is the possibility to built on top of a basic infrastructure application specific Model of Computation and methodologies. On this way modeling for a restricted application area and abstraction level can be simplified and the simulation performance can be increased by orders of magnitude. For digital systems transaction level modeling (TLM) is a good example for such a methodology. The pay off is a higher required knowledge about modeling, MoC's and simulation algorithm.
The first public available prototype of SystemC-AMS is optimized for the specification and the system design of signal processing dominated applications. Such systems are usually over sampled systems. Due they use sigma-delta converters for analogue-digital conversion the signal frequency is orders lower than the sampling frequency. They consisting of digital filters, control realized in hard- and software, analogue filter and analogue external components and an environment which can be very often modeled by macro models using linear elements like resistors and capacitors.
Due the sampled nature of such systems, synchronous dataflow is a very well suited MoC for modeling the signal path. At the considered level of abstraction (specification and overall system design) only the values at sample points are considered and a zero delay for communication between filter is assumed or a delay is modeled explicitly as multiple of the sample time. The synchronous dataflow MoC activates the modules in signal flow direction. On this way an activated module can read from his inports always the current results of his predecessors without delay. Synchronous dataflow can be simulated very efficiently, due the scheduling order does not change during simulation and thus can be calculated once before simulation start.
The dataflow MoC is also appropriate for non-conservative modeling of analogue subsystems (analogue modules which are connected by directed signals – thus there is no backlash). As far as there is no loop back such systems can be solved by consecutively solving the sub blocks in signal flow direction. With loop back a delay of one time step is required. Due the over sampling rates of the considered systems such a delay is mostly acceptable. The advantages of this methodology are:

- No overall analogue equation system has to be set up
- The overall system is solved explicitly without iterations
- There are no restrictions regarding the content of the sub blocks – there is no solvability problem for the overall system (except the discussed loop delay)
- The simulation performance is very high

Such non-conservative modules can embed in general nonlinear differential equations. However for this general case such a module has to use a nonlinear DAE solver. If we restrict the module behavior to linear dynamic and non-linear static, we are able to solve the dynamic part by a very simple (and such efficient) linear equation solver and the static non-linear part can be easily described and calculated by C++ - language elements (e.g. if – then – else). Experiences have shown, that for the considered abstraction level and systems this restriction can be mostly accepted. An other assumption which increases the simulation performance significantly is a constant sample time. This assumption simplifies the algorithm for the linear equation solver and the interaction between the analogue and the digital part.
Especially wired telecommunication systems like ADSL- or Voice codecs uses a couple of external devices like resistors, capacitors, transformers and they are embedded in an analogue environment which can be usually modeled by macro models also consisting of also of those devices. For such conservative analogue nets an equation system has to be setup. Therefore the discussed SystemC-AMS version provides a domain for describing linear electrical networks consisting of numerous basic elements like sources and the mentioned resistors and capacitors. For such a description the equation system is setup and embedded in a dataflow cluster by the semantic layer.
Recapitulating the used SystemC-AMS prototype supports:

- Synchronous multi rate (the number of consumed/produced sample can be different – required for modeling decimation/interpolation filter) dataflow
- Embedding of linear differential equations into dataflow modules using representations like linear transfer functions
- Linear electrical networks by providing basic elements like resistors and capacitors
- Interaction at constant time steps between the different domains (dataflow, linear network and discrete event)

```
SCA_SDF_MODULE(prefi_ac)
{
 sca_sdf_in<double>   in;     //signal inport
 sca_sdf_out<double> out;  //signal outport

 //control signal from de-SystemC
 ///(connected to sc_signal<bool>)
 sca_scsdf_in<bool>   fc_high;

 // parameter
 double fc0, fc1     //cut-off frequency
 double vmax;       //max. out value

 sca_ltf_nd  ltf_0, ltf_1; //filter equation inst.
 sca_vector<double>  A0, A1, B;
 sca_vector<double>   S;      //state vector

 void init() {
  //filter coeffs for transfer function
  const double r2pi=1.0/(2.0*M_PI);
  B(0)   = 1.0;
  A0(0) = 1.0;            A1(0)=1.0;
  A0(1) = r2pi/fc0;      A1(1)=r2pi/fc1;
 }
```

$$H(s) = \frac{1}{1 + \frac{1}{2\pi f_c} s}$$

```
void sig_proc() {
 double tmp;
//high or low cut-off freq. depending on the
//control signal (ltf is an example for linear
//dynamic equations)
if(fc_high.read())  tmp= ltf_1(B,A1,S,in.read());
else                tmp= ltf_0(B,A0,S,in.read());

//output voltage limitation (simple example for
//a static nonlinearity)
 if       (tmp >  vmax)       tmp =  vmax;
 else if  (tmp < -vmax)       tmp = -vmax;

 out.write(tmp); //assign output voltage to port
            }

 SCA_CTOR(prefi_ac) { //constructor
  // default parameter values
  fc0 = 1.0e3;             fc1=1.0e5;
  vmax  = 1.0; } };
```

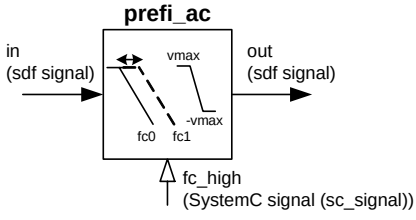


Figure 2 Example of the description of an analogue filter

Figure 2 shows an example of a synchronous dataflow module which embeds linear differential equations as ltf – numerator/denumerator representation, a control signal input for switching the cut-off frequency and a simple static nonlinearity for limiting the output value. This module models a for signal processing systems typical first order analogue filter for pre-filtering the input signal of an analogue-digital converter.
SCA_SDF_MODULE is a C++ - macro (according the SystemC SC_MODULE macro) which derives the current module (class) definition from a sca_sdf_module class (which again is derived from the sca_module base class). This sca_sdf_module class assigns the module to a synchronous dataflow semantic and provides the required language elements like an init – method which will be called once before simulation start and a sig_proc method which is called repeatedly during simulation. An access to the control port (fc_high) forces the synchronization of the SystemC-AMS and the SystemC time axis. For the current implementation the SystemC-AMS time axis is given by specifying the time distance between samples (calculation time points) once per dataflow cluster (connected dataflow modules).

```
SC_MODULE(prefi_externals)
{
  //synchronous dataflow inport
  sca_sdf_in<double>   kit

  //standard SystemC-discrete event inport
  //(to connect with sc_signal<bool>)
  sc_in<bool> fch;

  //electrical port
  sca_elec_port   pout;

  //component declarations
  sca_r          *r_it, *r_prefi, *r_prefi2;
  sca_c          *c_it;
  sca_sdf2i      *i_t;
  sca_rswitch    *sw_prefi;

  //internal nodes declaration
  sca_elec_node w_it, w_sw;

  //declaration of reference node
  sca_elec_ref    gnd;

  //module instantiations and binding
  //(connecting with nodes) in standard
  //SystemC constructor
```

```
  SC_CTOR(prefi_externals)
  {
    r_it = new sca_r("r_it");
        r_it->p(w_it);
        r_it->n(gnd);
        r_it->value=2.0e3;

    c_it = new sca_c("c_it");
        c_it->p(w_it);
        c_it->n(pout);
        c_it->value=100.0e-12;

    i_t = new sca_sdf2i("i_t");
        i_t->p(w_it);
        i_t->n(gnd);
        i_t->ctrl(kit);

                :
  }
};
```

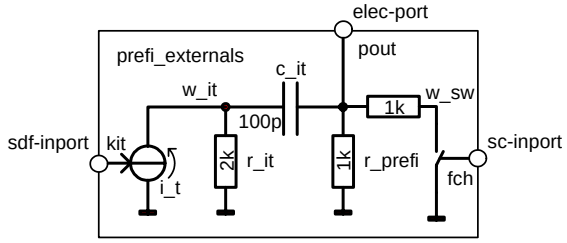


Figure 3 Hierarchical description of a conservative linear net

Figure 3 shows an example of a hierarchical description of a conservative linear electrical network using predefined elements like resistors and capacitors. Hierarchical descriptions which are using SystemC-AMS sub-modules do not differ from hierarchical descriptions of standard SystemC. Thus they use the module declaration macro SC_MODULE and the constructor macro SC_CTOR. All SystemC-AMS ports and channels (like sca_sdf_signal and sca_elec_node) can be used in the context of hierarchical SystemC descriptions.
The provided conservative linear electrical elements are implemented as modules derived from a sca_elec_module class. This sca_elec_module class, which is again derived from sca_module, attaches the module to a linear electrical semantic. Additional the sca_elec_module class provides an interface for describing the module contributions to the equation system, which will be setup by the linear electric semantic. This equation system

setup will be done during elaboration. As the result the linear electric instance of the semantic layer instantiates a linear differential equation solver. Like mentioned before a solver in the SystemC-AMS sense must implement the unique synchronization interfaces. Thereby the connections to other solvers and to the SystemC simulation kernel will be represented by the ports to other domains (in the example of Figure 3 by the sca_sdf_in - and the sc_in – ports).

Overview of the Example System

The example system is a voice codec which represents the interface between the analogue subscriber (POTS – Plain Old Telephone Service) and the digital world. Such circuit systems are used in numerous different telephone applications and have to fulfill a lot of standards due the different applications and the diversity of country specific telephony regularities.
The basic functionality of such system is described by the acronym BORSCHT (**B**attery **O**vervoltage **R**inging **S**ignaling **C**oding **H**ybrid **T**esting). Thus such systems have to realize the following main features:

- Feeding of the subscriber with a supply voltage (traditionally coming from **B**atteries)
- Protect against **O**ver voltages (e.g. due lighting)
- Let the subscriber to **R**ing
- Support a couple of special **S**ignaling functions like message waiting (the blinking lamp in hotels)
- Perform the analogue-digital and digital-analogue conversion and **C**oding/decoding to/from the digital transmission (e.g. PCM)
- Match the impedance of the analogue phone line and split the transmission directions of the two wire line (**H**ybrid) – the classical terminating set functionality
- Integrate a lot of **T**est and measurement features like the measurement of the resistance.

Besides this classical features a lot of new requirements came up due new application areas like internet telephony, intelligent network terminations (e.g. connection of an analogue phone to a ISDN net) and the combination with ADSL (asynchrony data subscriber line) data transmission systems. Figure 4 gives a raw functional overview of a two channel version of such a system.

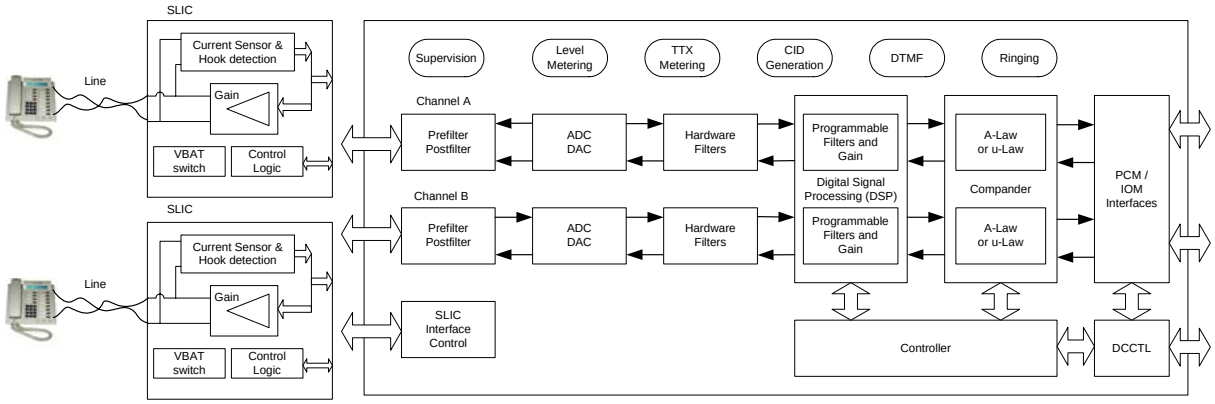


Figure 4 Functional overview of a two channel version of the example system

Today such systems are realized by a lot of analogue and digital signal processing, sophisticated control algorithm, processors for running application specific tasks and different digital interfaces. Additionally a high voltage (up to 150V) driver (SLIC – Subscriber Line Interface Circuit) and some analogue external elements are required.
The design challenges for the systems starting at the specification phase. There are a lot of inputs for the specification like the country specific regularities, application specific standards and requirements from different customers. A main problem consists in the fuzziness of the requirements which can not be directly translated to a specification which is usable for the

circuit design. A lot of feature are defined in a way like “The chip-set must be able to measure the ringer capacity with a certain accuracy.”. However such a measurement can be performed only in conjunction with the application system and not by the chip-set oneself. In such cases the chip-set specification must be developed - in the example case a feasible sequence has to be defined which performs the measurement. This has may be done in a tight cooperation with the customer.
The next challenge is the partitioning of the algorithm by re-using experiences from former designs and iterating with the specification (if possible to make the specification more feasible) and the implementation (to assure the viability, area, power consumption, …). In this phase the development of the embedded software is started.
Now it must be assured, that the performance defined in the specification will be achieved and all defined features (usually in the order of hundreds) are implemented and working in the expected way. Thereby the functionality of the embedded software must be tested.
Due the complexity an executable specification as reference for the implementation of the digital and analogue components is more and more required.

Application of SystemC/SystemC-AMS for Modeling the Example System

As mentioned before the system consists of large signal processing parts, control, interfaces and analogue driver. The digital signal processing consist of:

- Approx. 100 programmable filter
- Numerous multiplexer for opening/closing loops
- Tone generators
- A nonlinear characteristic for generating the subscriber feeding voltage
- Integrators and registers for measurement functions
- A lot of different thresholds for measurements and detecting certain states like an on- or off-hook (telephone receiver) of the subscriber.

This functionality is modeled by synchronous multi rate dataflow modules. Due the much higher performance of the static dataflow scheduling, the SystemC-AMS dataflow modeling domain was used also for this digital modules instead of the SystemC dataflow Model of Computation with “sc_fifo”. The required accuracy is called “bittrue”, which means, that the digital numbers at the sample time points are the same like latter on the circuit implementation by using the same digital numbers as input. This “bittrue” tests are performed for all digital paths from the analogue-digital to the digital-analogue converter. To achieve this accuracy SystemC data types like “sc_int” were used to describe the filter algorithm.
The sigma delta converter also modeled as synchronous dataflow modules. They are modeled with C-Code and taking into account only a few effects like the saturation of the integrators.
The analog filter, multiplexer and analogue gain control blocks are also modeled with dataflow modules using embedded linear differential equations (linear transfer functions) and static nonlinearities described by C++ (like Figure 2). All performance relevant analogue pols, zeros and nonlinearities are modeled.
The driver circuit is implemented by a kind of generic model, which is able to represent different types of the circuit. This circuit is modeled by a combination of synchronous dataflow models and linear network elements. The model implements all operating modes (according the type 6-15 different modes), voltage/current limitations, behavior during mode changes (especially loading of the middle voltage and pols and zeros.
Figure 5 shows the principal procedure for modeling the driver circuit. An intention is to use always the most appropriate (regarding modeling effort, required accuracy and simulation performance) MoC. Thus the control functionality is modeled in an event-driven SystemC module due the rare and not periodical changes of the control signals. The signal path is modeled with synchronous dataflow modules due he has to be calculated very often and periodically. For this abstraction level the transformation from the input voltage (v2w) to the

output voltage (between tip and ring) and reverse the output current to the voltage at “it” can be considered as reactionless. However the external networks for modeling components like over voltage protection, the line and the subscriber will become so complex that they can’t be efficiently modeled with a reactionless dataflow MoC. Thus the driver outports modeled by electrical ports.
An other intention is to reduce the calculation effort for modules which has to be activated very frequently. Thus in the simple example of Figure 5 the control module calculates parameter for the signal path according to the selected mode. On this way the operations which has to be performed in the very frequently called dataflow modules can be reduced significantly.

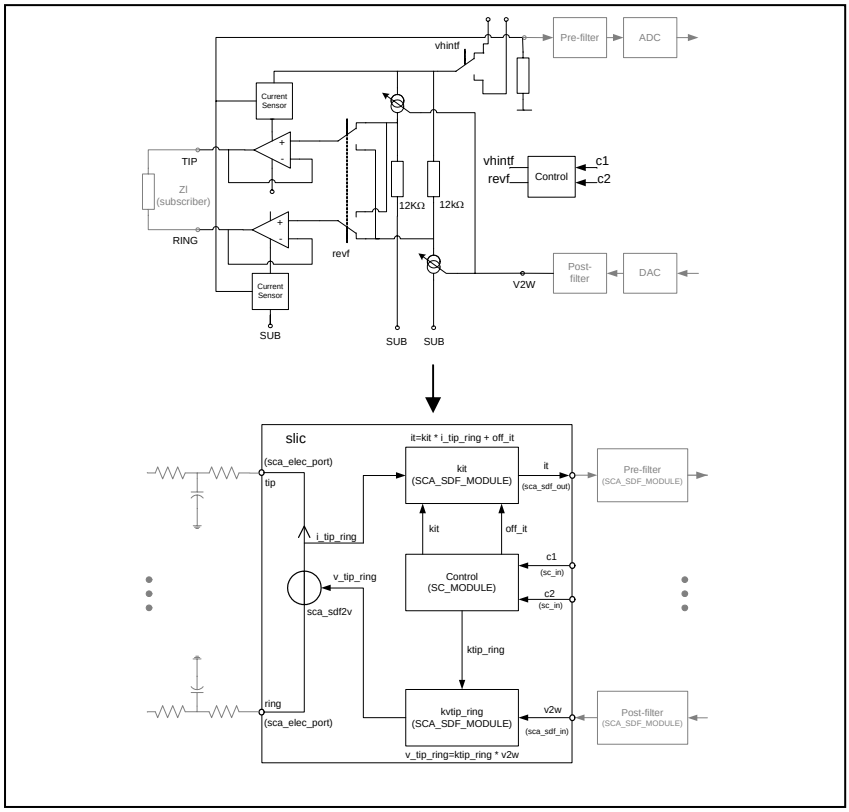


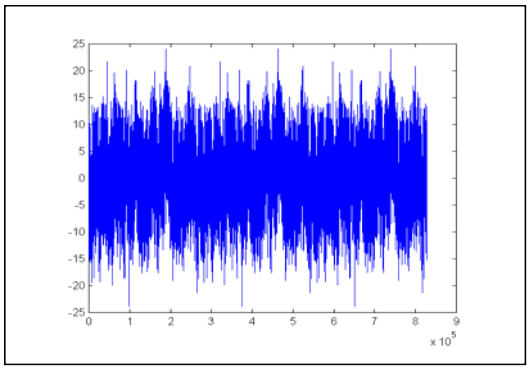
Figure 5 Principal procedure for modeling the driver circuit (SLIC)

The external components like over voltage protection and the environment like the subscriber modeled by macro models using linear analogue elements (similiar the example of Figure 3).
A very important issue is the control. A lot is required for configuration of the system, e.g. to program it for a certain application or to fulfill special country regularities. Other parts are required to detect certain states or to implement features like the ringer capacity measurement mentioned before. Some hundreds of control signals going to and coming from all parts of the system. Thus there are control signals going to the driver circuit for driver-mode switching, and coming from this circuit like the result of an analogue comparator which allows the detection of the subscriber state in a power down mode by a sophisticated software algorithm. Other signals going into the analogue filter part switching gains or cut-off frequencies e.g. for increasing the settling performance after mode changes or reducing the noise for other operating states. Signals to the digital signal processing switching signal paths, filter modes, coefficient sets, tone generators and so on. Signals from this part detecting reached thresholds and time outs. The most of this signals will be read or generated by a software control algorithm. The controller is modeled by an event driven SystemC module embedding the original C-code, which will be latter run on the implemented controller. Therefore a simple C++ shell is required which main tasks are the mapping of the C-code symbols to the control

signals and managing the activation of the algorithm to the required time points. This simple integration was possible due all used data types and operations of the target processor could be mapped to C – types and operations of the simulation platforms (PC or Sun). On this way we were able to compile the control software with the same C-compiler like the model. Besides a high simulation performance this leads to good debug feasibilities due the same debugger like for the model can be used to debug the control software and special debug code like the "classical" printf can be added to the control code.
Nearly all digital hard- and software filter are programmable by several hundred filter coefficients. Connecting to all this filters a bus or signal for coefficient reading will be cumbersome and slow down the simulation performance. Thus we decided to use a global C++ - variable for modeling the RAM. Thus the RAM can be accessed by all programmable filter and by the interfaces or alternatively by a file reader for loading the coefficients directly instead using the digital interfaces. The disadvantage of using global variables is the with the simulation time not synchronized access. Thus a certain fuzziness and non causality regarding the time points when a coefficient is read or written can occur. However in the special case of digital filter programming this is acceptable, due they are used for configuration only, the behavior during programming is not specified and usually not critical, programming the coefficient on the real circuit via the interfaces leads also to a certain fuzziness which can't be modeled efficiently at the considered abstraction level.

Usage of the Model

The model was implemented to develop the specification, to design feasible algorithm, perform the partition between analogue/digital and hard-/software, develop the embedded software and to generate an executable specification and a "golden" reference for the implementation. MATLAB was used to perform frequency domain estimations and partially for post-processing of SystemC-AMS simulation results tasks which require a high interactivity and moderate computational power. A time domain Matlab implementation of the required detailed level ("bittrue", electrical nets, HW/SW-interaction) will be cumbersome and due the script-language which has to be partially interpreted at run time the required performance will not be achieved.
To be able to perform all with the goals involved simulations a performance in the order of one hour per second real time is required. Due the a/d-converter sample with 32MHz this corresponds to a order of 10^7 simulation steps per hour for the fast modules and external nets. Additionally a lot of different simulation scenarios must be handled. Like mentioned before the system has hundreds of different features and must support a lot of standards and regularities. To achieve a high probability of a first silicon right design several hundred simulation scenarios are required.
The scenarios can be split into scenarios for analyzing the performance and for scenarios for testing features and checking the control algorithm and there interaction with the digital and analogue components.
Performance analysis covers the simulation of characteristics like the synthesized impedance, echo cancellation and signal to noise ratio. This simulation is performed for a few important configurations and the known "corner cases". Therefore usually a multi-tone source stimulates the system and a FFT sink calculates the frequency domain behavior from the time domain results. Such scenarios usually took approximately 100ms real time for voice transmission and several seconds for the dc-regulation characteristics. The results will be compared with MATLAB frequency domain estimations. These estimations are done by consecutively called MATLAB functions which implement the frequency dependent complex transfer functions of the sub blocks. Figure 6 shows the result of such a performance analysis for a decimation path.



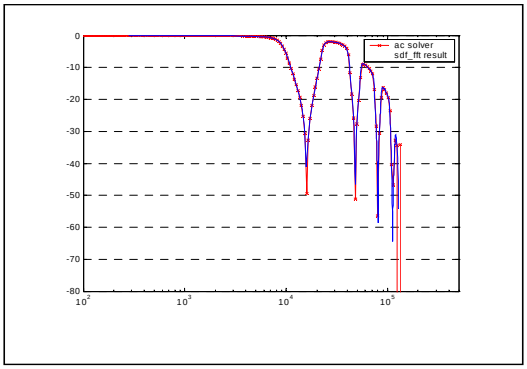


Figure 6 Time domain result (voltage over calculation points with 1/32MHz time distance) and FFT post-processing compared with a MATLAB frequency domain (ac) estimation (dB over frequency in Hz) for a decimation path (receive path)

An other class of simulations aims to check the settling behavior. This simulations go through the hole characteristic by changing the subscriber impedance. On this way the feeding voltage accuracy, the settling behavior and algorithm with speed up the settling by pre-loading filters or changes cut-off frequencies will be verified for a couple of programmed characteristics. Due the dc-regulation has time constants in the order of seconds such simulation can took several 10 seconds real time per configuration, whereby the ac-regulation loop with the 32MHz a/d-converter has a significant influence to the behavior – thus we have to deal with time constant differences in the order of 10^8 (for 10 seconds this corresponds to 320 million calculation points of the fastest modules).

A third very large class of stimuli verifies certain features like the hook-detection and the mentioned before ringer capacity measurement. These scenarios are characterized by a strong interaction of all system parts Figure 7 shows the simulation result of the ringer capacity measurement. A subscriber in the on-hook state can be considered as a resistor (the ringer) in serial to a capacity (for dc-decoupling). If a voltage ramp applied to the line, the resulting current is proportional to this capacity after a settling time. The measurement configuration has to be programmed and the sequence has been started by sending several commands via one of the digital interfaces. The sequence control has to be done by the controller software. The ramp generation and average calculation of the measured current is done in software dsp-algorithm. The digital and analogue hardware filter has to be controlled in a way, that the settling behavior is reduced and the required signal paths are switched to perform the measurement. All this activities have to be synchronized with the “analogue” line state (the arrows in Figure 7 illustrating some dependencies) and the interface commands. Besides the basic functionality like the hook-detection must sill work during the measurement.

A challenge was the handling of the amount (several hundred) of stimuli. Therefore a generic testbench concept was developed. Using this concept only one toplevel schematic is used for all overall system simulation scenarios. To this toplevel different stimuli can be applied described with derivations of a stimuli base class. In this class the user can provide different stimuli specific methods like for dut (device under test)-configuration, inclusion of additional modules like sources or for post-processing, a stimuli sequence, trace and simulation control commands. Thus we separated model from stimuli development.

During the design process the feature were added sequentially or in parallel by different people. How mentioned before the implementation of different feature can require modifications of the same components or the definition can be contradictory. This requires a repeatedly execution of all previously implemented stimuli to figure out such problems. For this task a kind of assertion based simulation for analogue mixed signal systems were desirable. Currently for the most stimuli a manual analysis (visual with a Mixed-Signal Waveviewer or MATLAB) is required.

The effort for stimuli development was probably higher than the effort for developing the model. The manual analysis of the simulation results and there comprehension was also very time consuming.

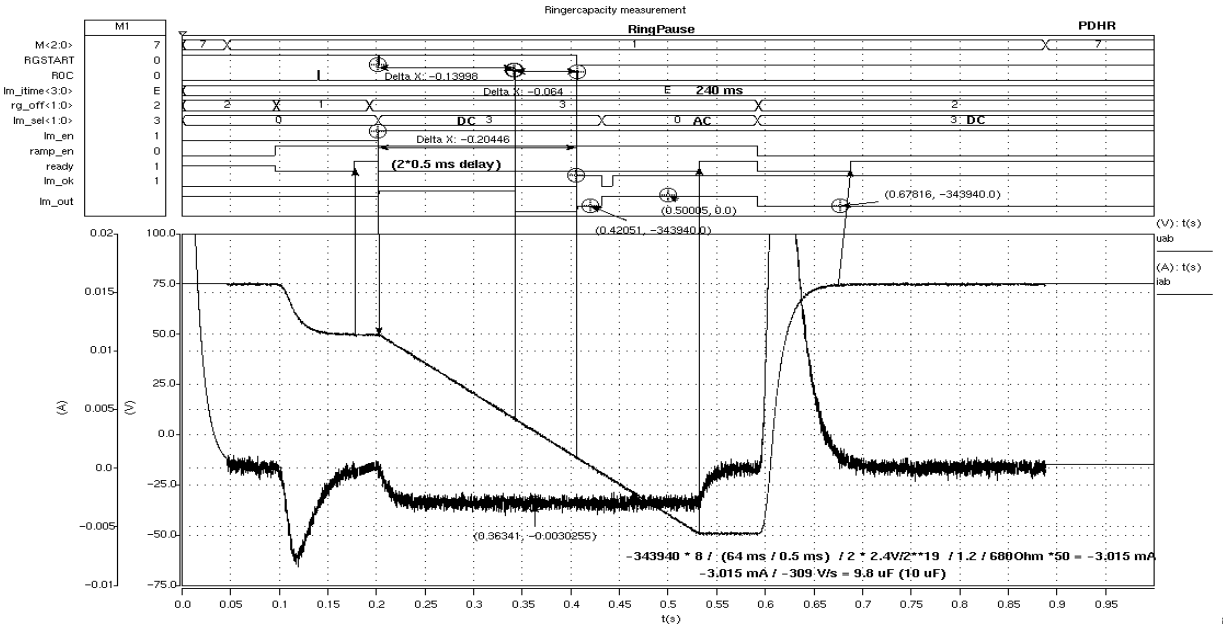


Figure 7 Example simulation of ringer capacity measurement

In parallel to the development of the exutable specification model the analogue and digital design (implementation) was started. For this implementation design phases the model was used as reference. The digital design used the model to generate stimuli and there corresponding expected values.
The described proceeding was one main factor to achieve a first silicon which were delivarable for productive use.

Future Work

The introduced SystemC-AMS extension library will be further developed and extended to other application areas. The most promising applications will come from the automotive industry and RF-wireless applications. The simulation and design problems which are deal with high functional complexity and the tight interaction of analogue, digital hard- and software will be quite similar. However automotive applications are often characterized by a still higher heterogeneity. Integrated automotive systems can include sensors and actors. Thus we have to deal additional with different physical domains and dynamic nonlinearities.
The modeling requirements for wireless applications are very similar to the one of the described system. Additional we have to handle the high carrier frequencies. Therefore established methods like baseband modeling must be supported.
The experiences with the example system have shown a shift of the main effort from the simulation to the development of the simulation scenarios and the analysis of the results. Effort has to be spent to simplify and unify those stimuli descriptions and to combine them with the established methods of digital systems (e.g. the SystemC verification library).
Desirable is a viable extension of the digital assertion based simulation techniques to analogue mixed signal systems. Such an extension will increase the efficiency during the development and decrease the probability of errors significantly. Additionally the re-use or a re-implementation of feature or supported standards will be simplified.
An other lack is the verification of the implemented analogue modules against the models of the executable specification. Currently there is no method with formalizes this verification.

Conclusion

The application of the SystemC-AMS prototype was motivated and discussed on a wired telecommunication system. It was shown, that the required simulation performance and accuracy was achieved for the considered abstraction levels. The modeling effort will not dominate, but for the usage of the enhanced SystemC/SystemC-AMS features more knowledge (C++, MoC and there interaction, …) is required compared to traditional HDL's like VHDL or Verilog. For the introduced example, simulation performance was no longer the limiting factor for an overall verification on specification and system level. Thus it was possible to increase the specification and system design reliability significantly. Due the number of simulation scenarios was highly increased, the bottleneck shifted to the development of those scenarios and the analysis of the simulation results.

Acknowledgement

Parts of the work were founded in the scope of the Medea ANASTASIA+ project. Special thanks to all colleagues from Infineon Design Center Austria GmbH which were mainly responsible for the applicability of the results. Thanks also to all colleagues of the Fraunhofer Institute IIS/EAS Dresden which contributed to the SystemC-AMS concepts and especially for implementing parts of the models. The work of the SystemC-AMS study group brought several ideas from proprietary solutions together and made thus the development of a more generic concept possible. Thus also thanks to all members of this group.

References

[1] A. Vachoux, C. Grimm and K. Einwich; "SystemC-AMS Requirements, Design Objectives and Rationale; DATE03 (pp. 388-393) Munich, March 3-7, 2003

[2] K.Einwich, P. Schwarz, C. Grimm, C. Meise; „SystemC-AMS: Rationales, State of the Art, and Examples" In : W. Müller, W. Rosenstiel and J. Ruf (Eds.) „SystemC Methodologies and Applications" (pp. 273-297); Kluwer Academic Publishers 2003

[3] 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.

[4] www.systemc-ams.org

[5] Synopsys Inc., CoWare Inc., Frontier Design Inc. „SystemC User's Guide Version 2.0", www.systemc.org, 2001

[6] 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

[7] C.Grimm, C.Meise, P.Oehler, K.Waldschmidt and W.Fey; „AnalogSL: A Library for modeling analog power drivers with C++", FDL01 Sept. 3-7 2001 Lyon, France

[8] E.A. Lee, D.G. Messerschmidt, Synchronous data flow, Proceedings of the IEEE, vol. 75, no. 9, 1987.

[9] H.D. Patel, S.K. Shukla, Towards A Heterogeneous Simulation Kernel for System Level Models: A SystemC Kernel for Synchronous Data Flow Models, Proc. of International Symposium of VLSI Systems, IEEE Computer Society Press, 2004.

[10] A.Vachoux, C.Grimm, K.Einwich, "Analog and Mixed-Signal Modeling with SystemC-AMS", "; ISCAS2003 (pp. III-914 - III-917), Bangkok May 2003

[11] R.Sommer, I.Rugen-Herzig, E.Hennig, U.Gatti, P.Malcovati, F.Maloberti, K.Einwich, C.Clauss, P.Schwarz, G.Noessing, "From System Specification to Layout: Seamless Top-Down Design Methods for Analog and Mixed-Signal Applications", DATE02, Paris 2002

[12] 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.