SystemC Extensions for Mixed-Signal System Design

Karsten Einwich[1], Christoph Clauss[1] , Gerhard Noessing[2], Peter Schwarz[1], Herbert Zojer[2]

[1]Fraunhofer, IIS Design Automation Dept. Dresden

[2]Infineon MDCA Villach

Abstract

State-of-the-art systems especially for the telecommunication and internet market, consist of analog and digital parts. The challenge for future designs is the tight interconnection between all different domains (software, digital control, hardware, signal processing, ...). Unfortunately, currently available tools and methodologies for overall system specification and simulation focus on one or two domains only (like VHDL, Verilog, SystemC). The presented methodology allows an overall system specification and verification for mixed-signal systems in a homogeneous environment based on C++.

1 Introduction

By decreasing chip structures, the analog part of a mixed-signal device will become more and more dominant with respect to chip area, power consumption and silicon costs. Thus it is essential to shift analog functionality to digital hard- and software and to compensate the non-perfect properties of the analog part. Therefore the coupling between analog and digital hard- and software will become tighter and tighter. For the design of digital hardware, software and algorithms of such systems it is essential to include the analog components and the system environment into an overall simulation. Therefore simulation performance is very crucial.

Hardware description languages like VHDL-AMS or Verilog-AMS allow principally a description of a mixed-signal system. But the description style especially for higher abstraction levels is often not suitable. The performance of currently available mixed-signal simulators is orders of magnitude to low for an overall system verification of a complex SoC. Because of the time-consuming process of standardization and tool development „classic" hardware description languages cannot react fast to new requirements. The development and standardization of VHDL-AMS tooks more than 10 years.

Recently there have been a lot of activities for a wide usage of C++ as a hardware description language (SystemC [SYN00], SpecC [GAJ00], Cynlib [CYN99], OCAPI [VER99]). Unfortunately these activities cover only digital domains (hardware, software and communication). Analog domains currently are not considered.

This paper shows some possibilities to extend these existing digital approaches to signal processing and mixed-signal/analog domains. First results, are presented for a chip of a line driver for a ADSL system.

2 Domains, Abstraction, Model of Computation

A complex (electrical) system has to described w.r.t. different domains. The splitting into domains is subjective and depends strongly on the applied design methodologies. E.g. for a wired telecommunication system we can distinguish the following domains:

- Linear electrical network
- Analog filter
- Digital filter
- Signal processing algorithm
- Control algorithm
- Interface algorithm

The „creative" design process can start and stop at different abstraction levels. The starting point depends on the complexity and re-use aspects, the stopping is determined by the availability of

compiler and synthesis tools. By choosing an abstraction level, some properties of the model of computation (MoC) are assumed explicitly or implicitly, and by describing the system with a simulateable language the MoC is usually fixed.

Model of Computation	Field of application
Event driven	Digital/timing
Cycle based	Digital/RTL
Remote process call (RPC)	Software, Communication
Synchronous dataflow	System design digital filter/algorithm
Kahn process networks/dynamic dataflow	Algorithm design
Frequency domain	System design telecommunication
Linear DAE's	Mixed-Signal-System design
Signalflow	Mixed-Signal-System design
Nonlinear DAE's	Analog circuit design

Table 1: Main focus of MoC's

Principally different domains at different abstraction levels can be mapped on the same MoC. But a MoC is developed only for a few domains and abstraction levels. Using a MoC outside his main focus means usually a performance loss and/or an increasing modelling effort. Table 1 gives example for MoC's and his main focus and Table 2 a selection of tools/languages and his MoC's.

Tool/language	Model of Computation
VHDL/Verilog	Event driven
VHDL-AMS/Verilog-AMS	Event driven + nonlinear DAE's, frequency domain
COSSAP/SPW	synchronous/dynamic dataflow
Matlab/Simulink	Linear/nonlinear DAE's, signal flow, (cycle based), frequency domain calculations
Ptolemy	Datflow, signal flow, event driven
SDL	token driven
Programming languages	Sequential processes

Table 2: Tools/languages and the supported MoC

The consequence is that the different design groups are using different tools, e.g. the algorithm design group uses C/C++ based tools and Matlab, the system design group uses tools like Matlab, COSSAP and SPW, the digital design group uses VHDL/Verilog based tools and the analog design group SPICE or Mixed-Signal tools like Saber and AdvanceMS. Bringing these different views together is always a big issue. Such possibilities are tool couplings [EIN96] or the creation of an overall model, e.g with VHDL-AMS. The first method is difficult to handle and the second one needs a high additional modelling effort. Both ways are not optimal with respect to performance.

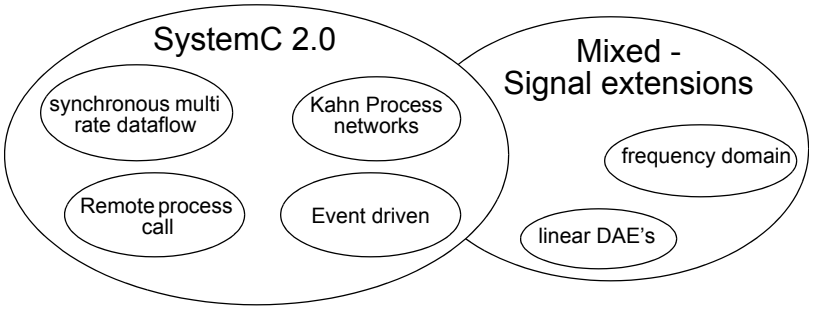

Fig. 1 Examples for SytemC MoC's and proposed extensions

Through its powerful generic model of computation, SystemC 2.0 supports a wide range of MoC's for discrete time systems (Figure 1 shows a selection). But MoC's for analog modeling cannot be constructed.

Driven by our mixed-signal applications we are developing an extension of SystemC for linear DAE's and frequency domain simulation. The linear DAE-solvers will be integrated into the synchronous dataflow MoC.

3 Mixed-Signal System example

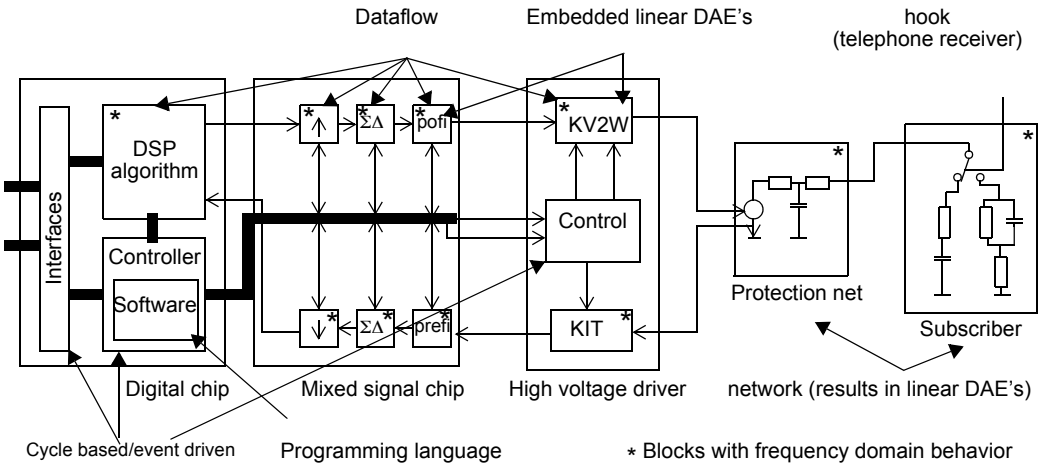

Fig. 2 Use of different MoC's in a mixed signal design

Figure 2 shows a extremely simplified system view on a so-called Subscriber Line Interface and Codec Filter (SLICOFI) system [ZOJ00]. 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 parallel to voice (ADSL).
In 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 synchronous dataflow (SDF) MoC 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 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.

4 Mixed-Signal SystemC extensions

Dataflow

Within the generic core language of SystemC 2.0 a dataflow MoC can be constructed. But we expect for synchronous dataflow (SDF) a low performance compared to a specialized scheduler.
For the described application the SDF performance will dominate the overall performance. Thus we introduce a specialized scheduler for SDF-clusters (independent graph of SDF-blocks). Additionally we include some properties which makes the synchronization with time driven domains more efficient and easier. Furthermore we perform a pre-simulation dead-lock detection and localization (so far as possible) which includes also dead-locks in conjunction with a time driven domain (e.g. event driven). Nevertheless we will become compatible with the SystemC dataflow modeling style which is still under discussion. In this case our scheduler can be used for SDF - clusters in conjunction with the scheduler of the SystemC reference implementation.

Linear equation solver principle

For a lot of applications in system level design, a linear equation solver is sufficient for the simulation of dynamic analog blocks. These blocks can contain transfer functions, state-space equations or linear electrical networks. In the linear case the resulting system of equation (1) can be solved without iteration. In connection with a sampled system, these blocks can be solved using a constant step-size h normally. If the backward Euler formula is applied the equations are transformed into a linear

system of equations (2) with constant matrices. It can be solved using a very fast Gaussian

$$A\dot{x} + Bx + q(t) = 0 \qquad \text{(eq. 1)}$$

$$\left(\frac{1}{h}A + B\right)x = \frac{1}{h}Ax_{last} - q \qquad \text{(eq. 2)}$$

elimination algorithm, because the factorization phase is carried out only once in the initialization phase [DER86]. If the assumption of a constant step-size cannot be fulfilled or the error will become to high a more complex (and thus slower) algorithm will be used.

Solving the equation of RLC-networks

To calculate the node voltages and (if necessary) some branch currents of a linear circuit the system of equations (1) have to be set up. Using the modified nodal analysis method (MNA) [HRB75] the matrices A, B, and the vector q can be simply constructed by utilizing the typical contributions („stamps“) of each linear circuit element (R, L, C, G, ...). Within the C++ language this is realized by a method matrix_entries, which is called in the model construction phase before the simulation is started. As an example, the contributions of a resistor R are described in (fig.3) (a resistor delivers no contribution to matrix A and vector q, node0 and node1 are the node numbers).
The solution of the system of equation delivers the nodal voltages only. Thus each element has a method for an optional current calculation. In this way, models of linear elements like R, L, C, controlled sources, linear transformers and transmission lines (by scattering parameters [VOL95]) are implemented.

```
class R : elec_elements {
    unsigned long node0, node1;
  public:
    elec_port a;
    elec_port b;

    R(elec_wire& aw,  elec_wire& bw, double val):
                                  elec_elements(Res)
                        {  a.bind(aw); b.bind(bw); value=val; };

    void matrix_entries(fhg_matrix_sq<double>&  A,
                        fhg_matrix_sq<double>&  B,
                        fnct_vector&            q )
        { //matrix entries
          B(node0,node0) +=  1.0/value;
          B(node0,node1) += -1.0/value;
          B(node1,node0) += -1.0/value;
          B(node1,node1) +=  1.0/value;
        }
      //method for current calculation
      double i_t(double* x, double dt)
            { return((x[node0]-x[node1])/value); }
};
```

Fig. 3 Implementation of a resistor

Frequency domain simulation

The linear DAE's can be solved in the frequency domain also. For dataflow and event driven blocks with frequency domain behavior an additional frequency domain implementation is necessary. For simulation a linear equation system has to be constructed and solved for each frequency point.

5 MoC synchronization

Synchronous dataflow and linear networks

Each independent network will be included during simulation set-up as a dataflow primitive block. The in- and out-ports of this block derived from the connections between dataflow blocks and the analog network (e.g. a dataflow input to a voltage source or a node voltage which is used as input for a dataflow block). If this dataflow block called from the scheduler, a time interval is calculated which is equal to the time distance of two samples. Usually we perform one analog time step. This is mostly sufficient for system level simulation, especially in systems with single bit Sigma-Delta-converters with high oversampling rates. Nevertheless we are able to oversample the analog network, whereby we can hold or interpolate the input signals.

SystemC (event driven) and synchronous dataflow

For synchronization SDF and SystemC, the sample time must be specified at least at one point in a SDF-cluster. This is done by setting a port attribute. Thus the SystemC to SDF synchronization library (hidden in port classes) can activate the SDF-cluster and respectively the SDF cluster can resume if data needed. Thus the SDF-cluster samples the signals from SystemC and schedules events (if the sample is not equal to the previous sample) to SystemC signals.

6 Examples for Mixed-Signal Descriptions

Figure 4 shows the C++ description of a pure electrical network, Figure 5 is the same network as hierachical module whereby vbslic is controlled by a dataflow inport and i2v converts a current to a dataflow outport. Figure 6 shows a dataflow primitive with an embedded DAE-system (second order lowpass). Besides this primitive has a SystemC event driven inport which controls the cut-off frequency. The structural descriptions (Figure 4 and Figure 5) can be generated by a netlister.

```
                                :
elec_wire w1, w2, w3, w4, w5, tr;        //nodes
elec_gnd gnd;                            //reference node

   double Rp1=60.0, Rp2=40.0;            //parameter
   double Cp=1e-12, Lp=1e-3;

     V vbslic(w1,gnd,2.0);
     R rp1 (w4,w2,Rp1);
     R rp2 (w2,w3,Rp2);
     C cp  (w2,gnd,Cp);
     L lp   (w3,tr,Lp);
     CCVS i2v (w5,gnd,w1,w4,1.0); //Current Controlled Voltage Source
```

```
// signal tracing, Matlab, WSF and CSDF
// currently supported
trace tr1(MATLAB,"tr1.dat");

tr1.add(&w1);   //node voltage
tr1.add(&lp);   //current through lp
                :
```

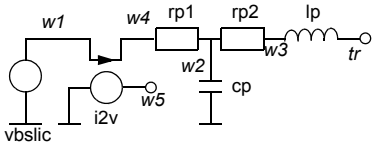

Fig. 4 C++ description of a linear electrical network

```
ELSDF_MODULE(prot_net)
{
 sdf_inport<double> slic_out;
 sdf_outport<double> i_tr;

 elec_port tr;

 double Rp1, Rp2, Cp, Lp; //parameter

 Vsdf      *vbslic; C2SDF *i2v
 R *rp1, *rp2;  C *cp; L *lp;
```

```
elec_wire w1, w2, w3, w4;
elec_gnd  gnd;

ELSDF_CTOR(prot_net)  {
     vbslic=new Vsdf(w1,gnd,slic_out); //voltage source with sdf-inport
     rp1   =new R(w4,w2,Rp1);
     rp2   =new R(w2,w3,Rp2);
     cp    =new C(w2,gnd,Cp);
     lp    =new L(w3,tr,Lp);
     i2v   =new C2SDF(w1,w4,i_tr,1.0);   //conversion current to sdf
} };
```

Fig. 5 Hierarchical netlist with SDF-in- and out-ports and a electrical port

```
SDF_MODULE(pofi_pcb)
{
 sdf_inport<double> INPUT;              //dataflow inport
 sc2sdf_inport<bool> ADSL_LITE;         //SystemC inport
 sdf_outport<double> OUTPUT;            //dataflow outport

 double FG0, FG1, K, h;                 //parameters

 LTF_ID ltf_id0, ltf_id1;
 fhg_vector<double> A0,A1, B0,B1, S;

 void attributes() {            //port attributes for synchronization
      ADSL_LITE.h=h;            //h sample time
 }

 void init() {
      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 sig_proc() {
     if(ADSL_LITE)
          OUTPUT=LTF(A1,B1,S,ltf_id1,INPUT);
     else
          OUTPUT=LTF(A0,B0,S,ltf_id0,INPUT);
}

 SDF_CTOR(pofi_pcb){}
};
```

$$H(s) = \frac{K}{1 + \frac{1,41}{(2\pi FG)^2}s^2 + \frac{1}{2\pi FG}s}$$

Fig. 6 Primitive with SDF-in- and out-ports and SystemC control in-port

7 First results

Figure 7 shows the top level netlist of an ADSL line driver front-end. This front-end is modeled by aprox. 50 linear network elements (R, L, C, controlled sources). Figure 8 shows the simulation results compared to a Saber AC and transient simulation. The AC simulation is used as reference. For the Saber and C++ transient simulations we used a multi tone signal with logarithmically spaced frequencies. By post-processing with Matlab (FFT, ...) the transfer function was calculated and compared with the AC-simulation. The samplerate was fixed to 17Mhz by the ADSL-chip set. For 40 ms real time Saber tooks with default parameters 2670 sec. and the C++ description 122 sec. We checked that the time for applying the input-signal and tracing the output signals is negligible.

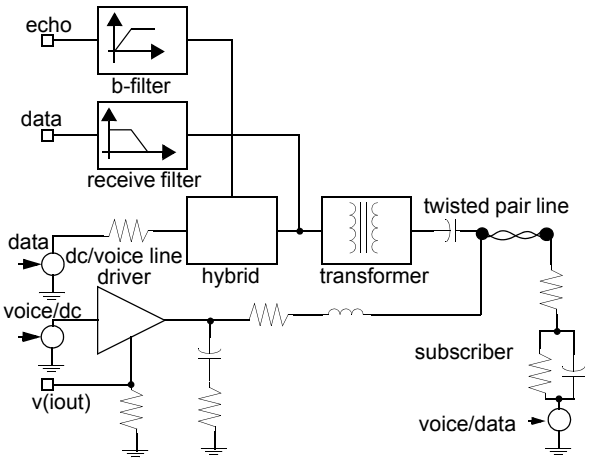

Fig. 7 ADSL-Line driver front end network

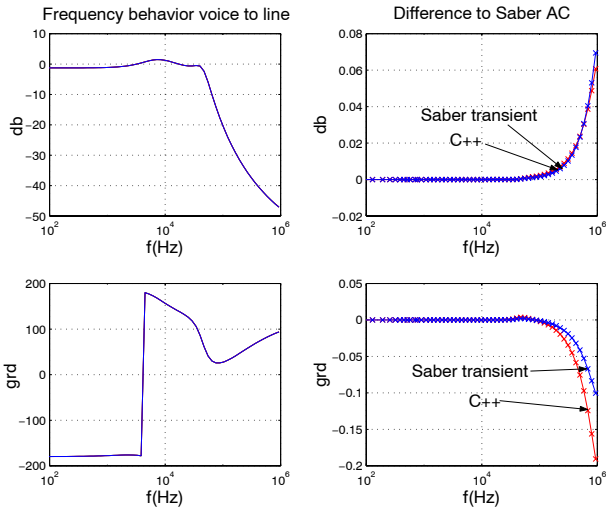

Fig. 8 Transient simulations compared in frequency domain

8 References

[BCP89] Brennau, K.E., Campbell, S.L., Petzold, L.R.: „Numerical Solution of initial value problems in Differential-Algebric Equations“, North-Holland, New York, 1989

[CYN99] CynApps, „Systems Modeling with Cynlib“, www.cynapps.com, 1999

[DER86] I.S.Duff, A.M.Erisman, J.K.Reid, „Direct Methods for Sparce Matrices“, CLARENDON PRESS, Oxford, 1986

[EIN96] K. Einwich.; Schwarz, P.; Trappe, P.; Zojer, H., „Simulatorkopplung für den Entwurf komplexer Schaltkreise der Nachrichtentechnik“, 7. ITG-Fachtagung "Mikroelektronik für die Informationstechnik", Chemnitz, 18./19. März 1996, 139-144

[GAJ00] D. D. Gajski, J. Zhu, „SpecC Specification language and Methodology“, Kluwer Academic Publishers, Boston 2000

[HRB75] Ho, C.W, Ruehli, A.E., Brennau, P.A., „The Modified Nodal Approach to Network Analysis“ IEEE Transa. CAS-22 (1975) June, 504-509

[MAY00] A. Mayer, „SystemC Applications“ First European SystemC Users Group Meeting, Munich, January 2000

[NAG75] Nagel, L.W., „SPICE2: A Computer program to simulate semiconductor circuits“ Berkeley University of California, 1975

[RUF00] J. Ruf, D.W. Hoffmann, T. Kropf, W. Rosenstiel, „Simulation Based Validation of FLTL Formulas in Executable System Descriptions“, FDL2000, Tuebingen, 7.9.2000

[SCH98] Schwarz, P.; Clauß, C.; Einwich, K.; Knöchel, U.; Matz, K., „Hybride Simulation nachrichtentechnischer Systeme“, 12. ASIM-Workshop "Systemsimulation", Zuerich, 15.-18.9.1998

[SPE00] M.Speitel, B.Niemann, R.Büttner, „The Application of SystemC in an Industrial Project“, FDL2000, Tuebingen 4.-8.9.2000

[SYN00] Synopsys Inc., CoWare Inc., Frontier Design Inc. „SystemC User's Guide Version 1.1“, www.systemc.org, 2000

[VAN00] G.Vandersteen et.al., „A methodology for efficient high-level dataflow simulation of mixed-signal front-ends of digital telecom transceivers“, DAC2000, Los Angeles

[VER99] S. Vernalde, P.Schaumont, I.Bolsens, „An Object Oriented Programming Approach for Hardware Design“, IEEE Computer Society Workshop on VLSI'99, Orlando, April 1999

[VOL95] I. Voll, J. Haase, „Rekursives Faltungsmodell für ein allgemeines Netzwerksimulationsprogramm“, 40. Intern. Wiss. Kolloqium der TH Ilmenau, 7.-9.Sept. 1995, Band 3, 269-274

[ZOJ00] 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