SystemC-AMS concepts for Mixed-Signal System Design

Karsten Einwich
Fraunhofer IIS/EAS Dresden

Fraunhofer Institut Integrierte Schaltungen

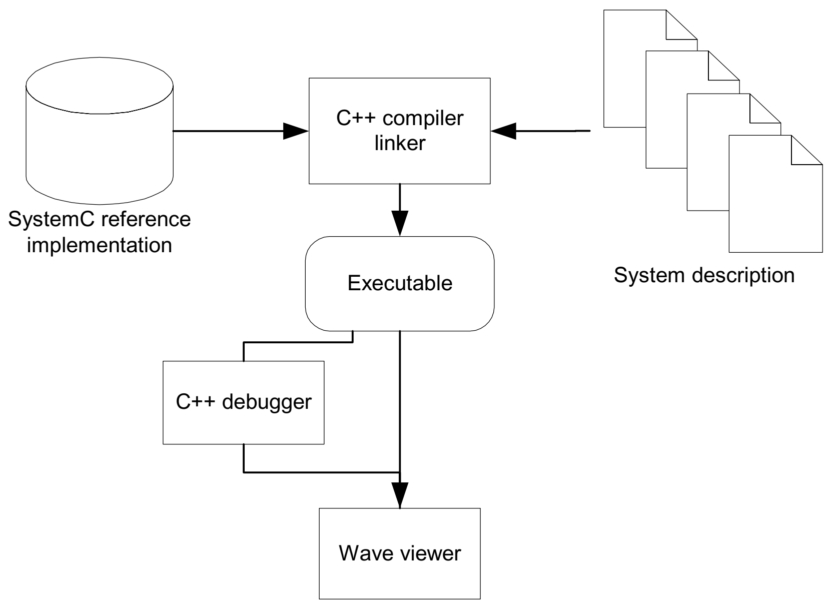
Outline

- Short introduction to SystemC
- Motivation for Analog and Mixed-Signal Extensions
- Digital versus analog simulation
- Requirements for Analog and Mixed Signal extensions
- Layered approach
- Examples
- Conclusion

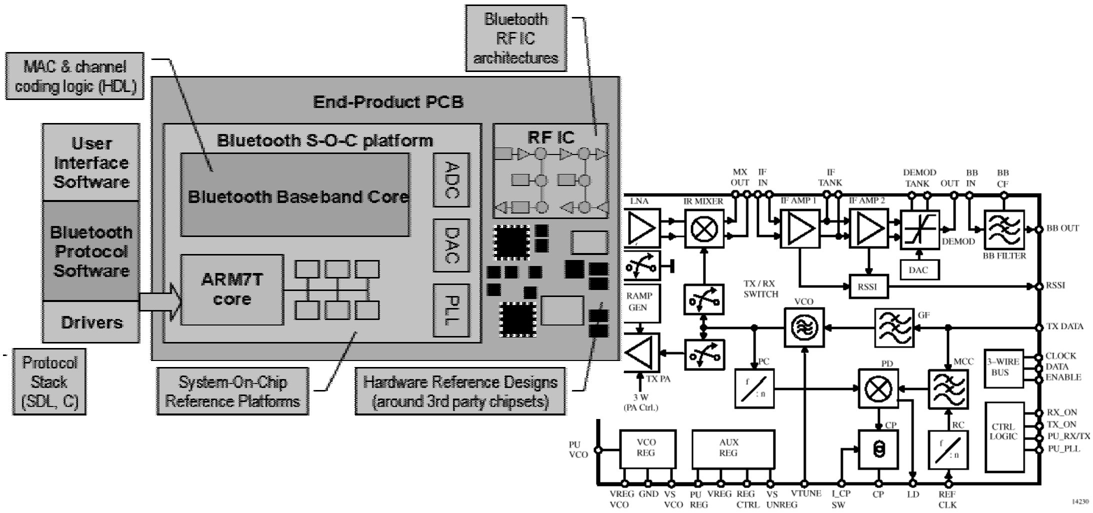
Introduction - SystemC is...

- A definition of **C++ language constructs** for the description of complex digital systems on different abstraction levels, using different Models of Computation (MoC)
- Definition of classes for modeling:
  - discrete signals
  - discrete, concurrent processes
  - generic communication channels
- SystemC – models can be simulated using a reference implementation of the C++ class library

SystemC Use Flow



Analog and Mixed Signal System



Design of Analog and Mixed-Signal Systems

- Design embraces multiple disciplines:
  - Software, Digital Hardware, RF, Power electronics, Mechanics, etc.
- Design needs many specialists:
  - System developer, Hardware designer, Programmer, Mechanical engineers, etc.
- Disciplines are strongly linked due to integration
  - Exchange of models, co-simulation

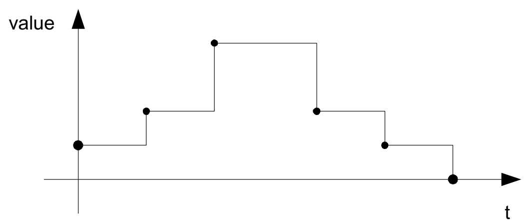
Problems ...

- ❑ Each specialist uses his preferred languages/tools:
  - Many different models exist and are often not consistent
- ❑ Verification of the system by mixed-signal mixed-domain simulation
  - Simulator coupling is often unpredictable, difficult, slow.
- ❑ Overall system simulation would need years and more.
- ❑ Mixed-Level-Simulation often impossible
  - model interfaces are modified within design flow: from equations to transactions to physical signals

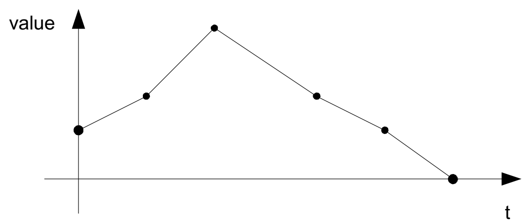
Fraunhofer Institut Integrierte Schaltungen

Analog and Discrete Signals

„Discrete" Signals:
- time discret
- discrete event
  - piecewise constant



„Analog" Signals:
- time continuous
- usually piecewise linear



Analog and Discrete Signals

- „Analog“ = behavior is „analog“ to (differential) equations
- Examples
  - Coil (differential equation): $$\frac{dI}{dt} = U / L$$
  - Diode (algebraic equation): $$I_D(t) = I_S(\exp\left(\frac{U_D(t)}{U_T}\right) - 1)$$

MEDEA+ Fraunhofer Institut Integrierte Schaltungen Infineon technologies ANASTASIA+

Discrete Event versus Analog Simulation

- Discrete Event Simulation (SystemC 2.0)
- “Analog” Simulation

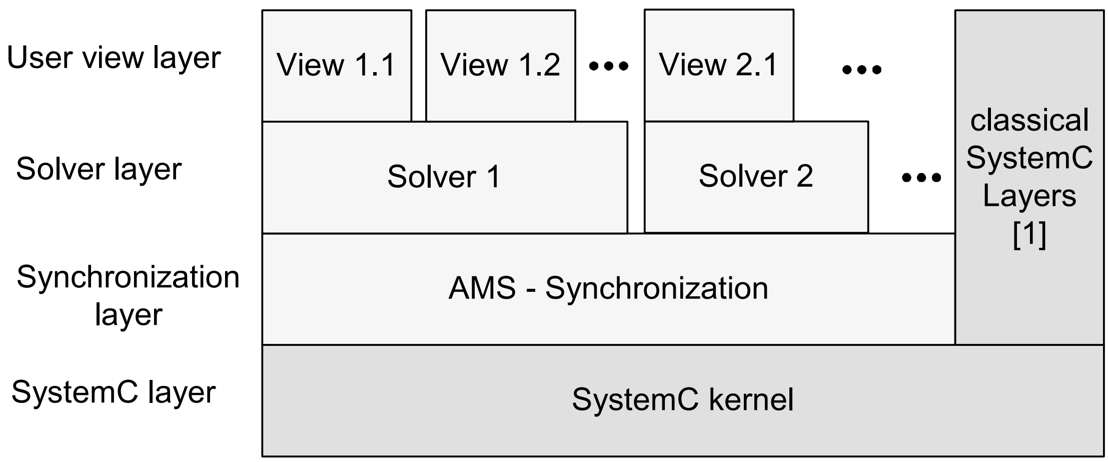
- Based on communication of processes
- Solve set of differential and algebraic equations

SystemC needs algorithm for solving differential and algebraic equation systems and methods for a equation system set up

Requirements

- Different and partial oppositional requirements
- A lot of very efficient however high specialized existing solutions
- A generic and extendable approach necessary
- The approach must be simple and efficient feasible
- The generic concept of SystemC has to be extended for AMS-Systems

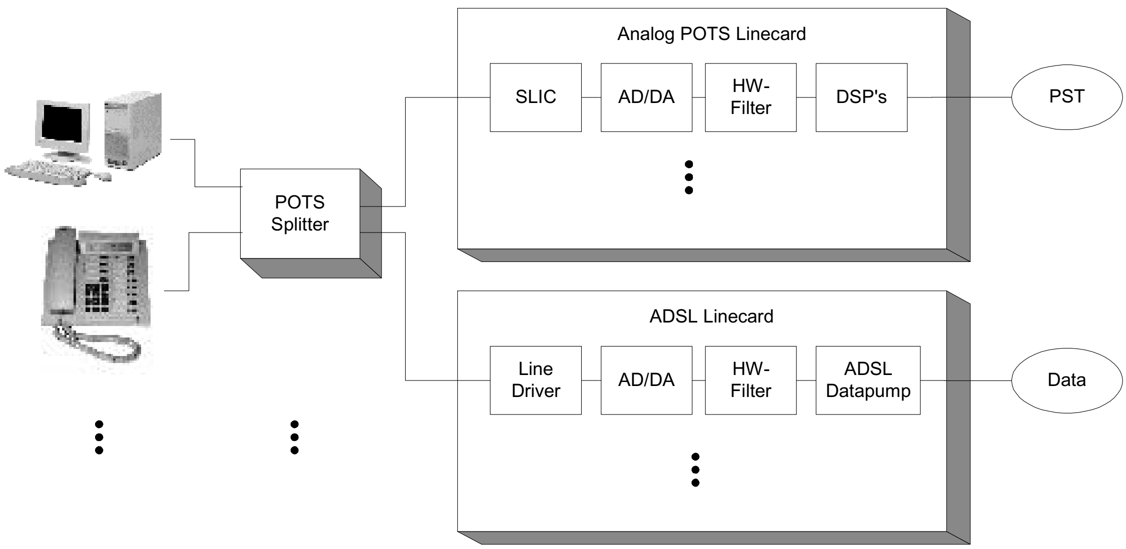
Layered Approach



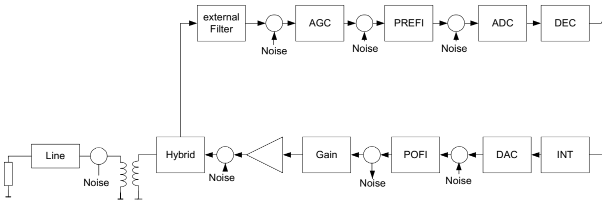
Application domains

- ❑ **Signal processing dominated application**
- ❑ RF- and wireless communication applications
- ❑ Automotive applications

Wired Telecommunication



System Simulation for Signal processing

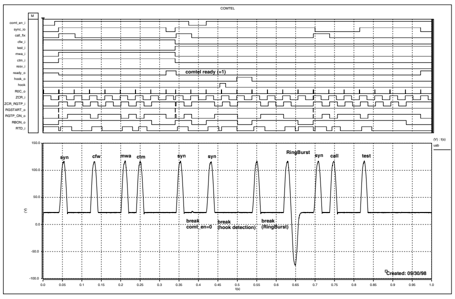


- ❑ Frequency analysis
- ❑ Small signal noise analysis
- ❑ Estimations
- ❑ Specification / design goal definition
- ❑ Level calculations

Preliminary Investigations

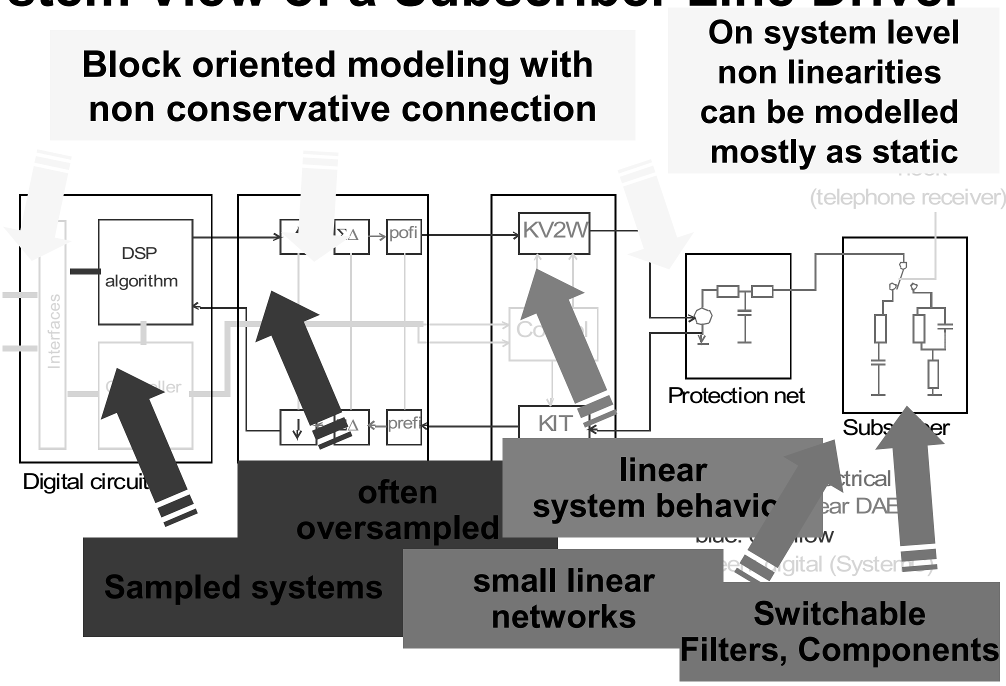
↓

Detailed Overall Model

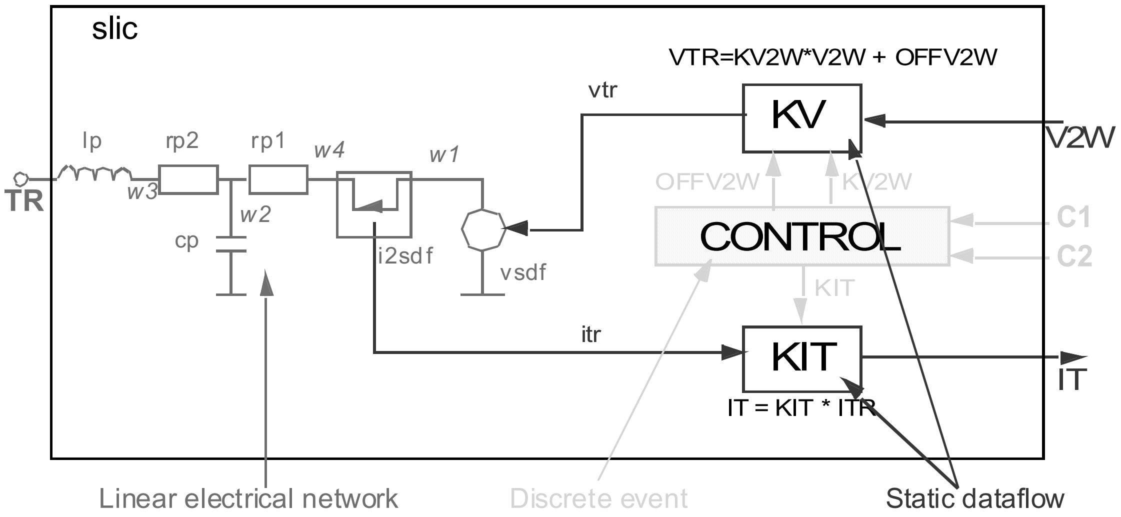


- ❑ Interconnection between analog and digital-HW/SW
- ❑ Bittrue digital filter
- ❑ Settling behavior
- ❑ Not neglect able second order effects
- ❑ Netlist verification

System View of a Subscriber Line Driver



Example for System Description



Top Level netlist

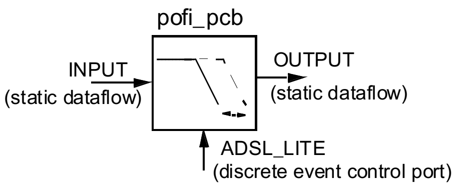
```
SC_MODULE(slic) {
  sca_sdf_in<double>  V2W;    //dataflow outport
  sca_sdf_out<double> IT;     //dataflow inport

  //discrete event (control) inports
  sc_in<three_level> C1, C2;

  sca_elecport tr;            //electrical port

  //discrete event signals
   sc_signal<double> kv2w_s, off_s, kit_s;
  //static dataflow signals
   sca_sdf_signal<double> itr, vtr;

   sca_wire  w1, w2, w3, w4; //electrical nodes
   sca_gnd   gnd;            //reference node

   //discrete event primitive
   slic_control *control;
   kit          *kit1;    //dataflow primitives
   kv2w         *kv2w1;
   //electrical primitives
   Vsdf         *vbslic;
   R            *rp1, *rp2;
   C            *cp;
   L            *lp;
   I2SDF        *i2sdf;
```

```
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);

    vbslic =new Vsdf("vbslic",w1,gnd,vtr);
    rp1    =new R("rp1",w4,w2,60.0);
    rp2    =new R("rp2",w2,w3,40.0);
    cp     =new C("cp",w2,gnd,1e-12);
    lp     =new L("lp",w3,TR,1e-3);
    i2sdf  =new I2SDF("i2sdf",w1,w4,itr);
                                   }  };
```

Dataflow Block with Discrete event inport

```
SCA_SDF_MODULE(pofi_pcb)
{
 sca_sdf_in<double>   INPUT;  //dataflow inport
 sca_de2sdf_in<bool>  ADSL_LITE; //de - 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()
{
  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()
{
    if(ADSL_LITE)
        OUTPUT=sca_ltf(A1,B1,S,ltf_id1,INPUT);
    else
        OUTPUT= sca_ltf(A0,B0,S,ltf_id0,INPUT);
    }

    SCA_CTOR(pofi_pcb){}
};
```



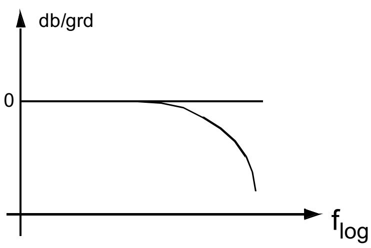
$$H(s) = \frac{K}{1 + \frac{1,41}{(2\pi FG)^2}s^2 + \frac{1}{2\pi FG}s}$$

MEDEA+ Fraunhofer Institut Integrierte Schaltungen Infineon technologies ANASTASIA+

Frequency Domain Implementation

```
SCA_SDF_MODULE(delay)
{
 sca_sdf_in <double>  inp;
 sca_sdf_out<double>  outp;

 unsigned long  delays;  //parameter
 double         init_val;

 void sca_attributes() {//attribute setting
                         outp.delay(delay);
                        }


 void sca_init() {
   //initialization for time domain
   for(long i=0;i<delays;i++)
    outp[i]=init_val;
                 }

void sca_sig_proc() {
                    //time domain implementation
                    outp=inp;
                     }
```

```
void ac_domain()
 {
  complex<double> j(0,1);
  double delay_time=inp.get_Tsec()* delays;

   SCA_AC(outp)=SCA_AC(inp)*
          exp(j*2.0*M_PI*SCA_FREQ*delay_time);
  }

   SCA_CTOR(delay)
   {
    //registers frequency domain implementation
    SCA_AC_DOMAIN(ac_domain);
   }
};
```



Conclusions

- SystemC can be extended for Analog and Mixed Signal design
- SystemC-AMS should cover the design phases starting from specification level and ending before circuit level
- However connection to lower levels should be established
- A realization concept based on a layred approach has been introduced
- For requirement definition different application domains has been identified