An Open VHDL-AMS Simulation Framework

Thomas Schneider

Darmstadt University of Technology
Institute of Microelectronic Systems

Infineon Technologies
Corporate Development

TECHNISCHE UNIVERSITÄT DARMSTADT

Infineon technologies

Overview

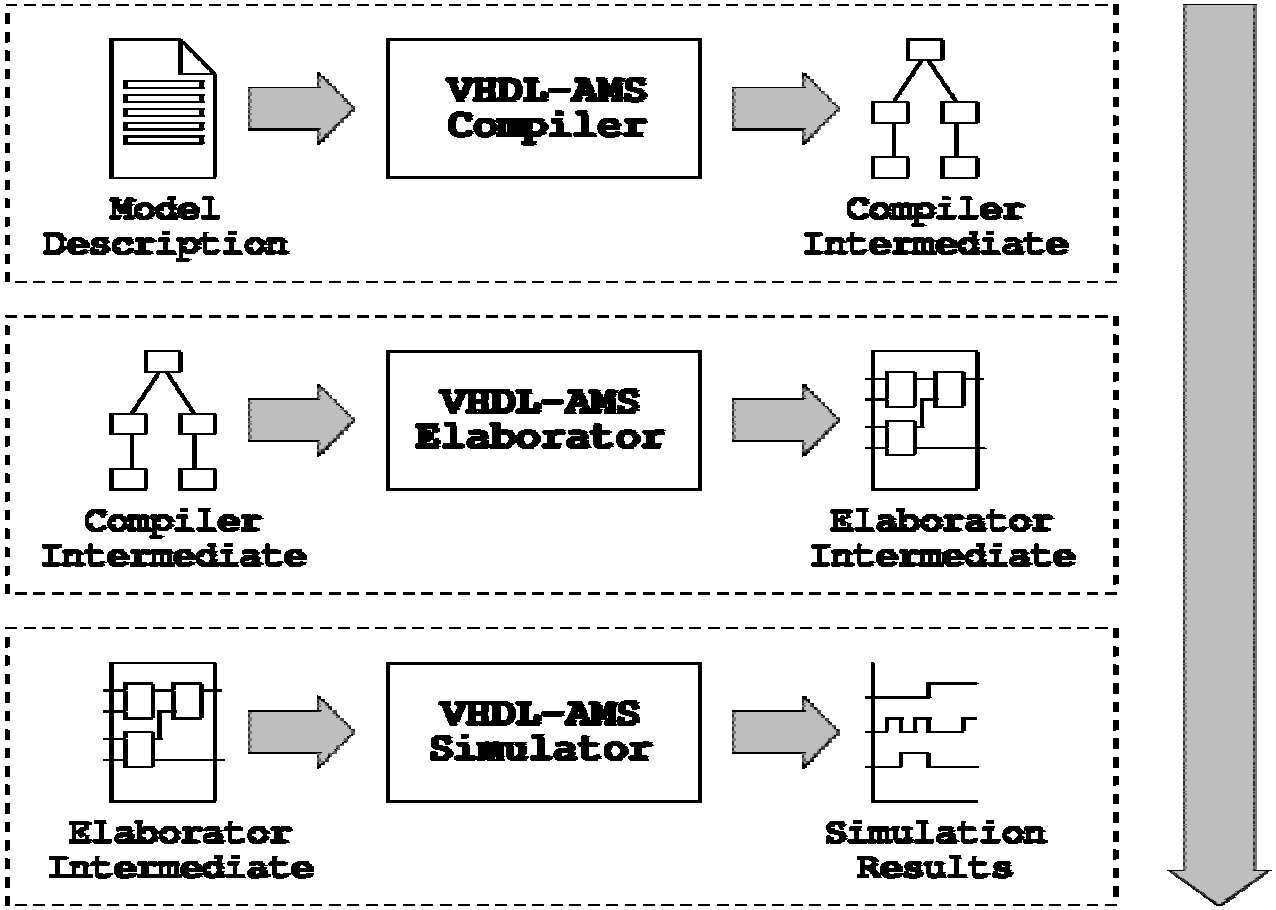
- Motivation
- Design Flow
- Framework Components
  - Compiler
  - Elaborator
  - Simulator
- Example
- Outlook

Motivation

- Infineon design-flow support
  - Closing design-flow gaps
  - Extending existing tools (e.g. in-house circuit simulator TITAN)

- Research platform
  - mixed-signal synchronization algorithms
  - solvability of analog descriptions in mixed-signal designs
  - mixed-language simulation

Design Flow

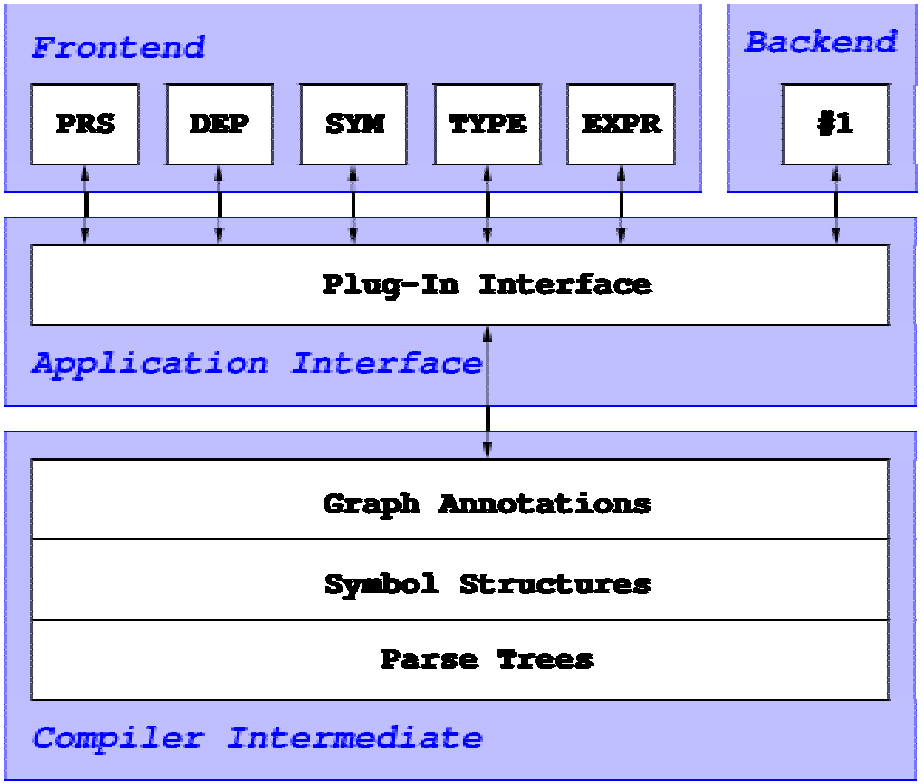


TECHNISCHE
UNIVERSITÄT
DARMSTADT

Infineon technologies

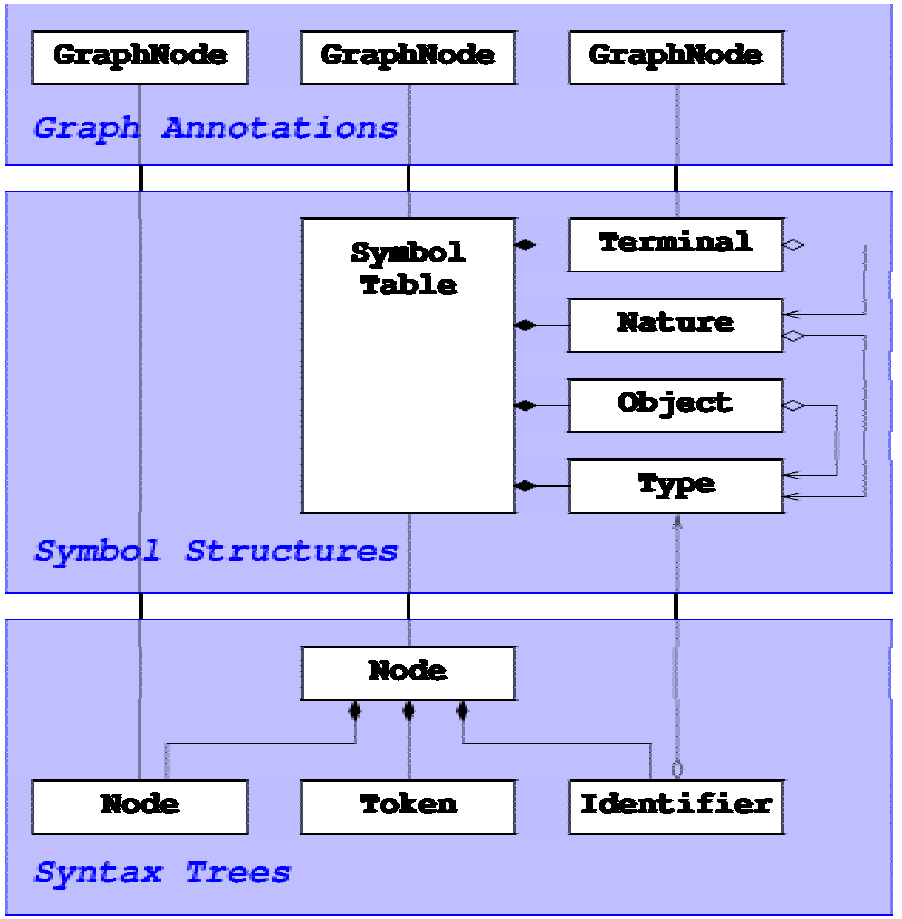
Architecture

- Frontend / backend
- Five-pass frontend
  - Parser
  - Depend. Generator
  - Symbol Generator
  - Type Checker
  - Expression Evaluator
- Open application interface
- Design intermediate



Intermediate

- Concrete syntax trees
- Symbol tables
- Object, nature and type instances
  - Nature links
  - Type links
  - Attribute links
- Inter-linked syntactic & semantic information
- Open graph interface

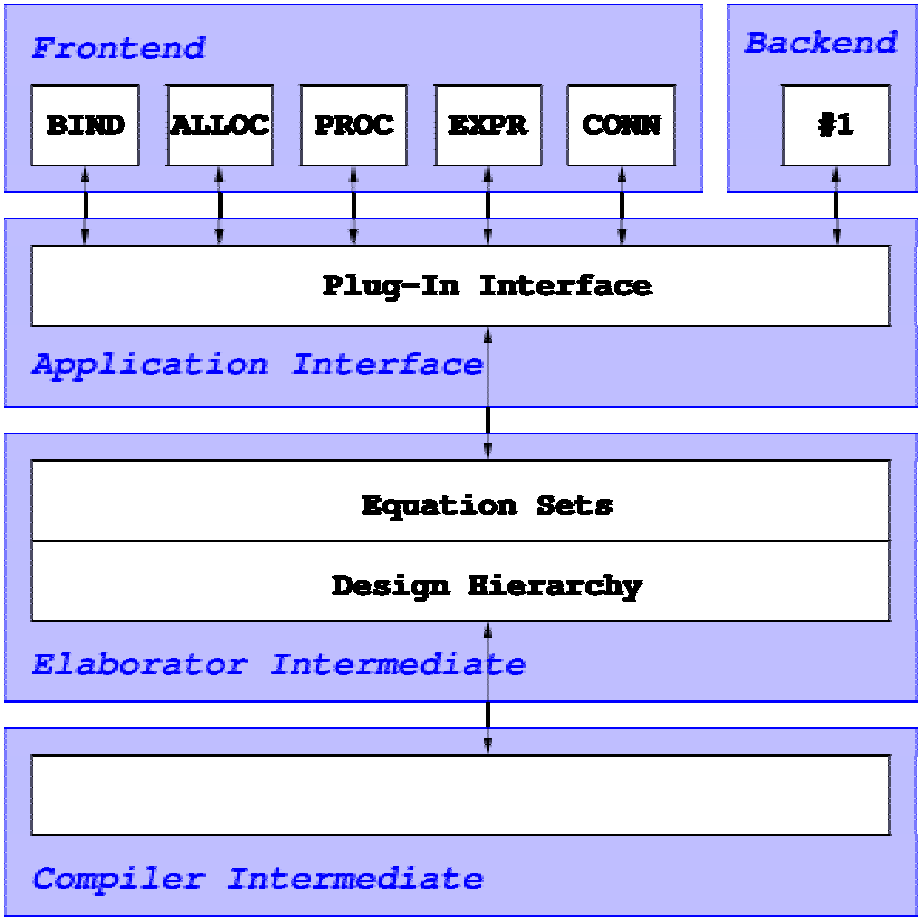


TECHNISCHE UNIVERSITÄT DARMSTADT

Infineon technologies

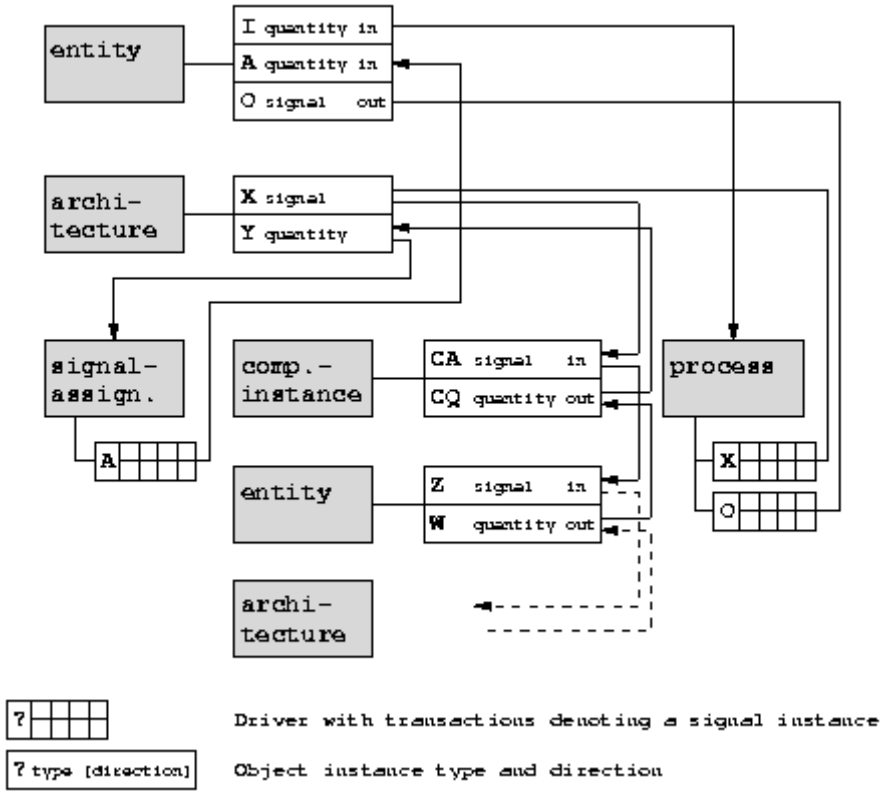
Architecture

- Frontend / backend
- Five-pass frontend
  - Binding Evaluator
  - Object Allocation
  - Process Evaluator
  - Expression Generator
  - Inter-connector
- Plug-In application interface



Intermediate (1)

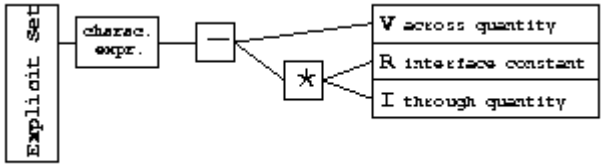
- Design hierarchy
- Activation records
- Object inter-connection across design units
  - Terminal connection
  - Quantity flow
  - Signal flow
- Process evaluation
  - Signal drivers
  - Source and sink information
- Calculation of initial expressions



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Intermediate (2)

- Characteristic expressions
- Expression tree structure
- Stored in sets
  - Implicit expressions of structural set
  - Explicit expressions of explicit set
  - Dynamic expressions of different augmentation sets and break set
  - Contribution expression of terminal declarations



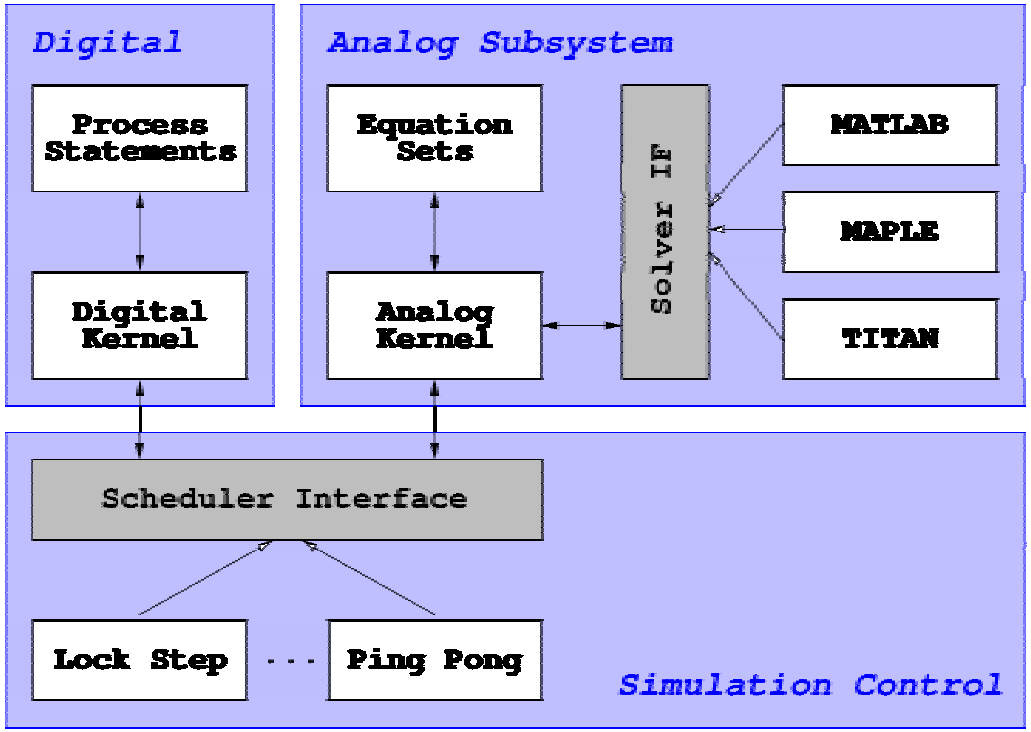
$$V == I \times R$$

	Structural Set
(1):	0 = O_REFERENCE_16305-ANALOG_IN_REFERENCE_13520
(2):	0 = S_19067-O_REFERENCE_16305
(3):	0 = I_REFERENCE_21862-ANALOG_IN_REFERENCE_13520
(4):	0 = O_REFERENCE_21865-ANALOG_OUT_REFERENCE_13523
(5):	0 = PR_REFERENCE_27609-I_REFERENCE_21862
(6):	0 = MR_REFERENCE_27612-O_REFERENCE_21865
(7):	0 = VR_30386-PR_REFERENCE_27609+MR_REFERENCE_27612
(8):	0 = PC_REFERENCE_33182-O_REFERENCE_21865
(9):	0 = MC_REFERENCE_33185
(10):	0 = VC_33190-PC_REFERENCE_33182+MC_REFERENCE_33185
(11):	0 = INPUT_REFERENCE_36049-ANALOG_OUT_REFERENCE_13523
(12):	0 = I_19068+IR_30387
(13):	0 = -IR_30387+IC_33191

	Explicit Set
(1):	0 = 10.0-S_19067
(2):	0 = 100.0*IR_30387-VR_30386
(3):	0 = 1.0/1.0E-6*IC_INTEG_33192-VC_33190
(4):	0 = INPUT_REFERENCE_36049-CHECK_39666

Architecture

- Dual-kernel architecture
- Adaptable scheduler interface
- Open analog solver interface

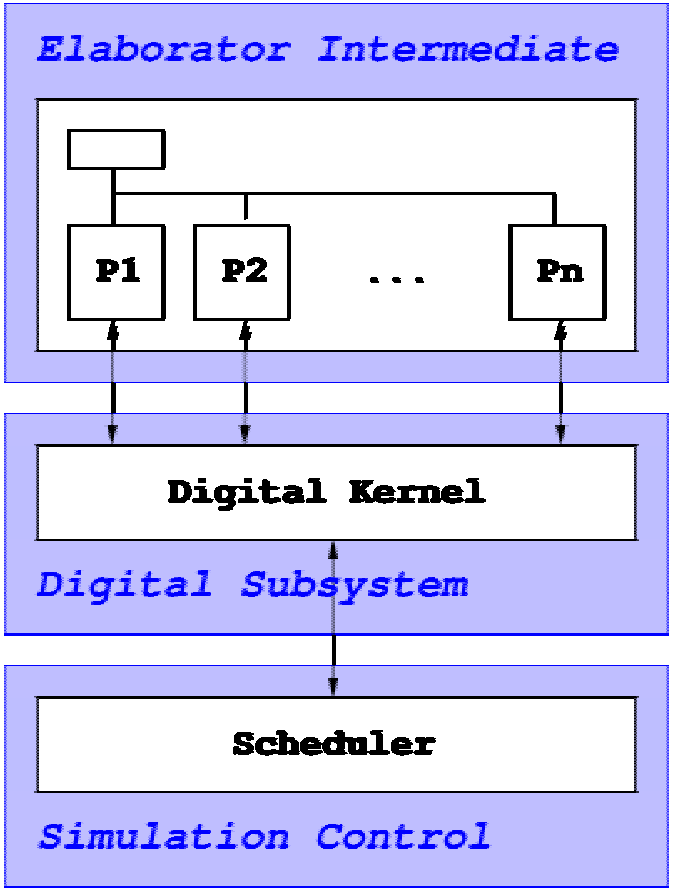


TECHNISCHE UNIVERSITÄT DARMSTADT

Infineon technologies

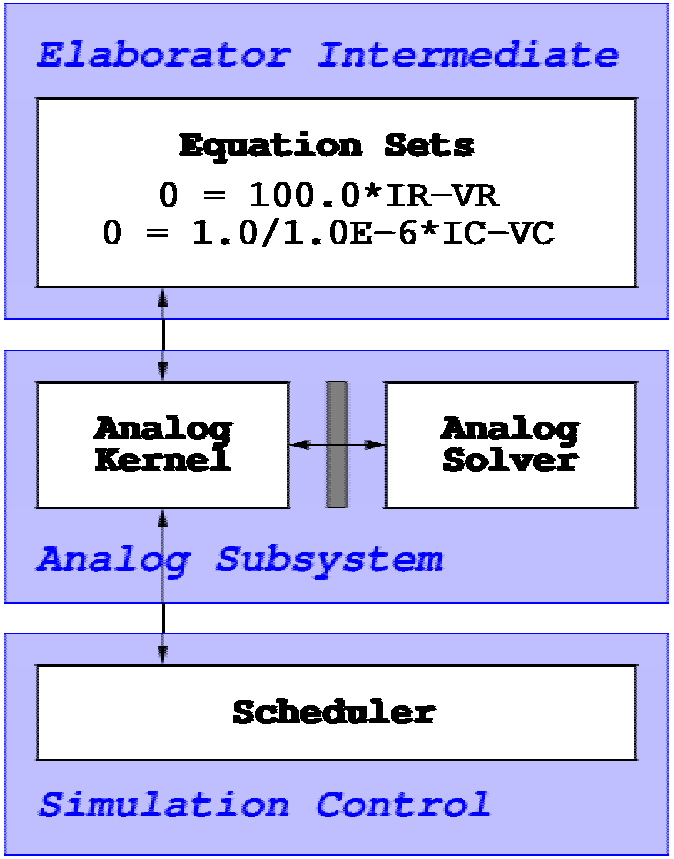
Digital Kernel

- Simulation cycles invoked by scheduler
  - delta
  - non-delta
- Interprets VHDL process statements
- Adds transactions to the signal drivers
- Processes break statements
- Updates break set, break flag



Analog Kernel

- Invoked by scheduler
- Calculation of time steps in $[T_c, T_n]$
- ASP calculation using the analog solver
- Update of quantity values at each ASP
- Calculation of Q'above events



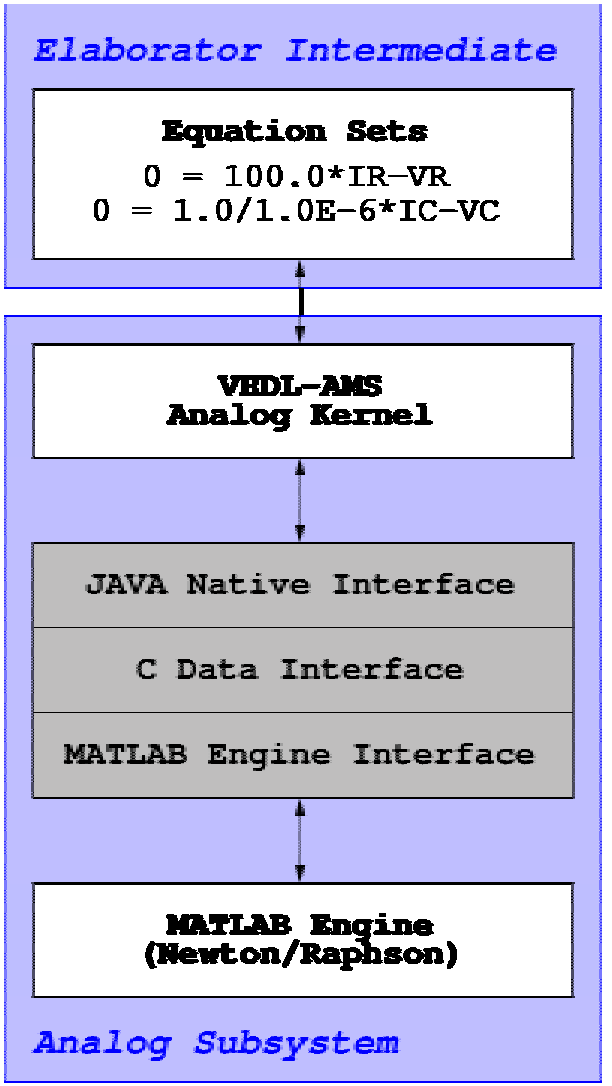
Analog Solver Interface

- Adaptation of different analog solvers
- Export mechanism for unknowns of the
  - structural set
  - explicit set
  - current augmentation set
- Export mechanism for equations and characteristic expressions (internal representation)
- Update mechanism for quantity values of the calculated ASP

```
public interface analog_solver
{
   public exportUnknowns()
   throws UnknownException;

   public exportEquations()
   throws EquationException;

   public getASP()
   throws ASPException;

   public updateUnknowns()
   throws UpdateException;
}
```


Prototype using MATLAB

- C-Library implementation for coupling the MATLAB-engine to the analog kernel
- Java-Native Interface (JNI)
- MATLAB engine library
- MATLAB programming language for solving algorithm implementation



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Infineon technologies

Prototype using MATLAB (2)

- Two implementations of a Newton-Raphson algorithm
- Variant 1 uses the MATLAB symbolic toolbox
  - simple implementation
  - weak performance
- Variant 2 uses a numerical approach
  - higher implementation effort
  - about 50% faster

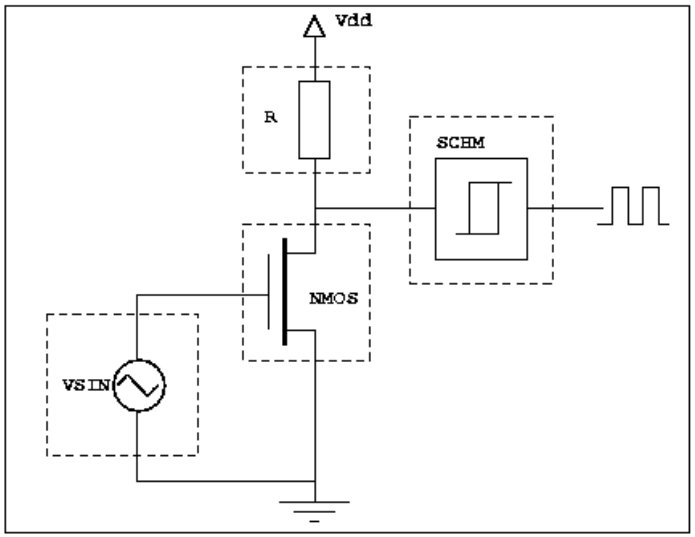
```
%--------------------------------------
% NEWTON-RAPHSON                SYMBOLIC
% File:    nrs.m
% Author:  J. Mades              19.03.00
%--------------------------------------
function [yfin]= NRS(F,x,x0,t,ta,tol)
   %--------------------------------------
   % Value of functions at x0
   %--------------------------------------
   F1=subs(F,x,x0);

   %--------------------------------------
   % inverse Jacobi-Matrix at x0
   %--------------------------------------
   J =jacobian(F,x);
   J1=subs(J,x,x0);
   J2=inv(J1);

   %--------------------------------------
   % Difference
   %--------------------------------------
   K  =J2*F1;
   x11=x0-K;
   x1 =subs(x11,t,ta);

   %--------------------------------------
   % Value of functions at x0, ta
   %--------------------------------------
   Fx2=subs(F,x,x1);
   Fx =subs(Fx2,t,ta)

   %--------------------------------------
   % Tolerance check and recursion
   %--------------------------------------
   for i = 1:length(Fx)
   ....
   ....
```

Scheduler Interface

- Homogeneous data-structure for analog and digital kernel
- Simulation cycle is divided into single simulation steps
- Sequential or parallel execution of digital and analog kernel
- Implementation of different synchronization algorithms through class inheritance

```
class lockstep extends scheduler
{
  ...
  void executeCycle()
  {
    aKernel.execute();
    setCurrentTime(getNextTime());
    updateImplicitSignals();
    updateExplicitSignals();
    dKernel.executeSProcesses();
    dKernel.executeRNPProcesses();
    calculateNextTime();

    if(!isDeltaCycle())
      dKernel.executePPProcesses();
  }
  ...
}
```

Example




```
architecture struct of example is
  terminal drain, gate, vdd: electrical;
  signal clk : bit := '0';
begin

   NMOS: entity work.mosfet(nmos)
         generic map(tc => 1.0, vth => 0.7)
         port map(drain, gate, ground);
   R   : entity work.res(behave)
         generic map(100.0)
         port map( vdd, drain);
   VSIN: entity work.vsin(behave)
         generic map (offset => 1.0,
                      T      => 1.0e-3,
                      mag    => 0.1)
         port map(v1 => gate, v2 =>ground);
   SCHM: entity work.schmitt(behave)
         generic map (vl => 3.5,
                      vh => 7.5)
         port map (refTerm => drain,
                   s       => clk);
   VDD1: entity work.vdd(behave)
         generic map (v => 10.0)
         port map(drain, ground);

end struct;
```

Example

MOSFET

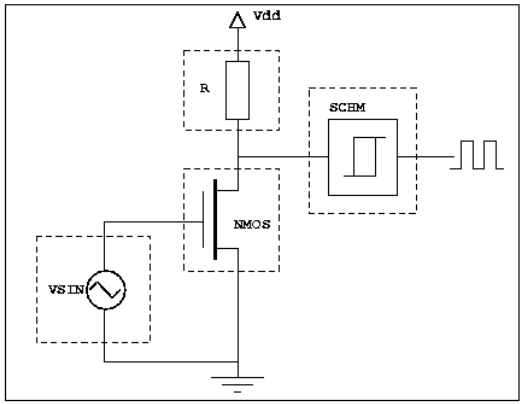
```
entity mosfet is
    generic ( tc  : real := 1.0;    -- transconductance (A/V**2)
              vth : real := 0.7);   -- threshold voltage
    port (terminal d, g, s : electrical);
end entity mosfet;

architecture nmos of mosfet is
    quantity vgs across g to s;
    quantity vds across ids through d to s;
    quantity vgd across g to d;
begin
    if vds >= 0.0 use                       -- forward
        if (vgs - vth) <= 0.0 use           -- cut off
            ids == 0.0;
        elsif (vgs - vth) <= vds use        -- saturation
            ids == 0.5 * tc *(vgs - vth)**2
        else                                -- triode
            ids ==  tc * vds *(vgs - vth - 0.5*vds)
        end use;
    else                                    -- reverse
        if (vgd - vth) <= 0.0 use           -- cut off
          ...
        end use;
    end use;
end architecture nmos;
```


Infineon technologies

Example

VSIN

```
entity vsin is
  generic ( offset  : real := 0.0;
            T       : real := 1.0;
            mag     : real := 1.0);
  port (terminal v1, v2: electrical);
end entity vsin;

architecture behave of vsin is
  quantity V across I through v1 to v2;
begin  -- behave
  V == offset + mag * sin(2.0*MATH_PI/T * now);
end behave;
```



VDD

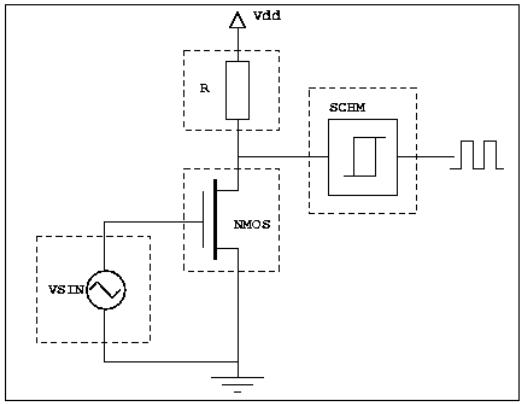
```
entity vdd is
  generic ( v  : real := 0.0);
  port (terminal v1, v2: electrical);
end entity vdd;

architecture behave of vdd is
  quantity Vout across I through v1 to v2;
begin  -- behave
  Vout == v;
end behave;
```

Example

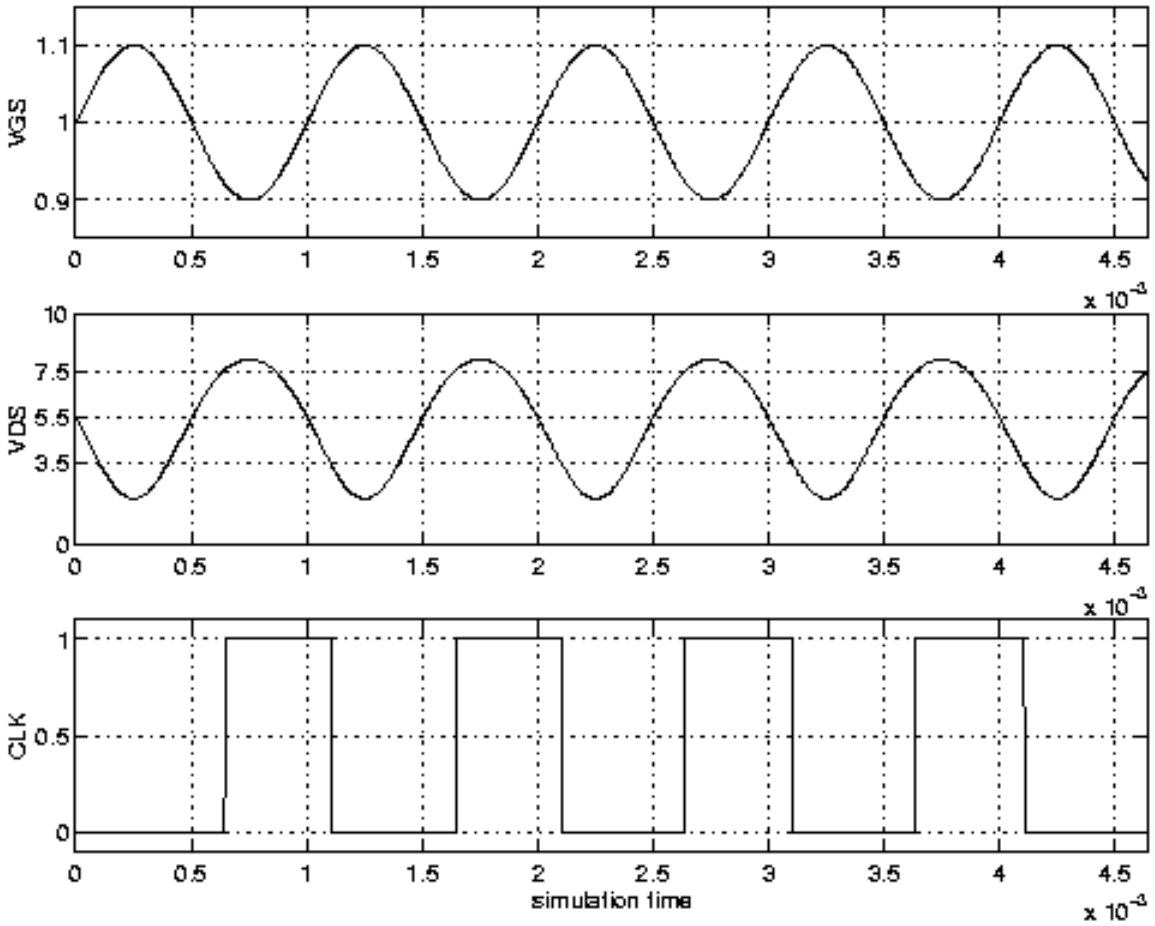
SCHMITT

```
entity schmitt is
   generic ( constant vl, vh : real :=0.0 );
   port    ( terminal refTerm: electrical;
             signal s : out bit:='0');
end entity;
```

```
architecture behave of schmitt is
  quantity ref : real;
begin
  ref == refTerm'reference;
  P: process
     begin
       if(ref'above(vh)) then
          s <= '1';
       else
         if(not(ref'above(vl))) then
            s <= '0';
         end if;
       end if;
       wait on ref'above(vh),ref'above(vl);
     end process;
end behave;
```



Simulation Results



Thomas Schneider

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Infineon technologies

Outlook

Summary

- Open VHDL-AMS design environment
- Entirely implemented in JAVA
- Research platform for mixed-signal and WEB-based design

Outlook

- Development of different synchronisation algorithms
- Solvability checks of hierarchical analog descriptions
- Integration of our in-house circuit simulator TITAN