Verilog-A/MS for RF Simulation

Marek Mierzwinski, Patrick O'Halloran, Boris Troyanovsky, and David Sharrit

Tiburon Design Automation
Santa Rosa, CA

MOS-AK /GSA Workshop
December, 2010 San Francisco

Outline

- Introduction/Motivation
- Verilog-A/MS for RF Simulation Implementation issues
  - Use issues
  - Implementation issues
  - Barriers to adoption in RF
- Verilog-AMS for RF simulation
- Conclusions/Future Directions

Introduction

- **Why custom modeling?**
  - **High frequency processes vary and the standard models usually can't keep up**
  - **RF modeling doesn't have the installed base for any one model type so users/foundries are forced to use simulator's interfaces to develop their own models**

Introduction

- **Verilog-A is a natural language for analog model development**
  - **Succinct**
    - derivatives, loads all handled by compiler
    - simple parameter support
    - powerful modelcard language
  - **Availability**
    - Verilog-A now supported by all major commercial vendors
- **Active and progressing standard**

Verilog-A: Example

- **Geometric resistor**

```
`include "disciplines.vams"
`include "constants.vams"

module resistor(p,n);
inout p,n;
electrical p,n;

parameter real R=0.0 from [0:inf); // resistance ohm
parameter real L=0.0 from [0:inf); // length m
parameter real W=0.0 from [0:inf); // width m
parameter real TEMP=0.0 from [0:inf); // resistor temperature C
parameter real TC1=0.0; // first order temperature coeff.  ohm-C^1
parameter real TC2=0.0; // second order temperature coeff. ohm-C^2
parameter real RSH=1000.0; // sheet resistance     ohm/[]
parameter real DEFW=10u from [0:inf); //   default width    m
parameter real NARROW=0.0 from [0:inf); // narrowing due to side etching    m
parameter real TNOM=27.0 from (-`P_CELSIUS0:inf); // Temp @ model extracted C

real Width, Tdev, DeltaT, Rscaled, Reff;

analog begin

    if ($param_given(R))
        Rscaled = R;
    else begin // If R is not specified, calculate it from geometry
        if (!$param_given(L) || !$param_given(RSH))
            Rscaled = 1k;
        else begin
            if ($param_given(W))
                Width=W;
            else
                Width=DEFW;
            Rscaled = RSH * (L - NARROW) / (Width - NARROW);
        end
    end

    // Calculate thermal offset, if necessary:
    if ($param_given(TEMP)) begin
        Tdev = TEMP + `P_CELSIUS0;
        DeltaT = TEMP - TNOM;
    end
    else begin
        Tdev = $temperature;
        DeltaT = Tdev - (TNOM + `P_CELSIUS0);
    end
    Reff = Rscaled * (1 + TC1 * DeltaT + TC2 * DeltaT * DeltaT);

    if (Reff > 0.0)
        I(p,n) <+ V(p,n)/Reff +
              white_noise(4*`P_K*Tdev/Reff, "thermal");
    else
        V(p,n) <+ 0.0;
end
endmodule
```

Compiler directives

Module and port definitions

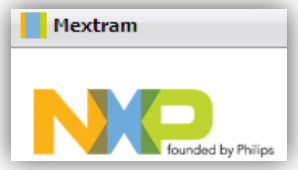
Parameter definitions

Analog behavior

- **Spice c-code would be ~10k lines**

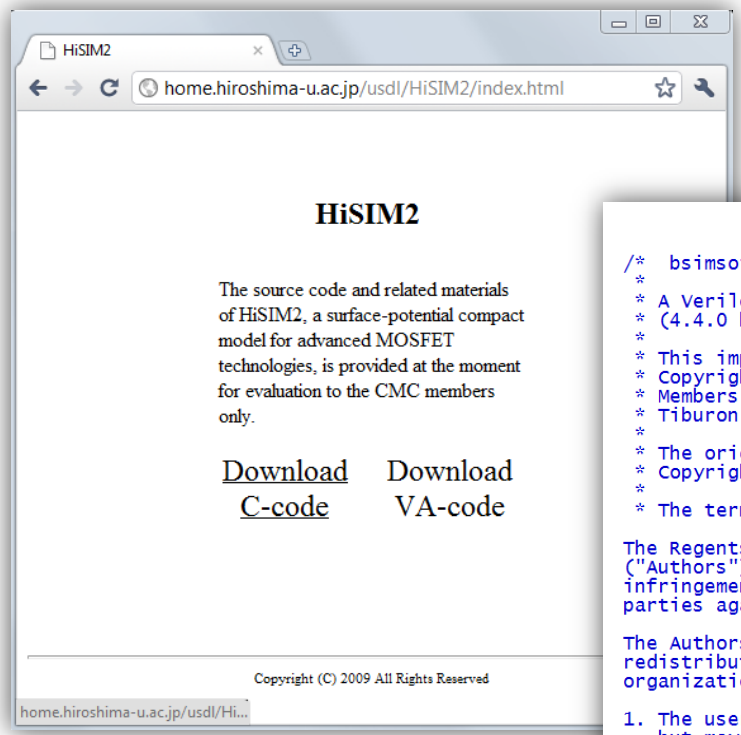
Current Implementations

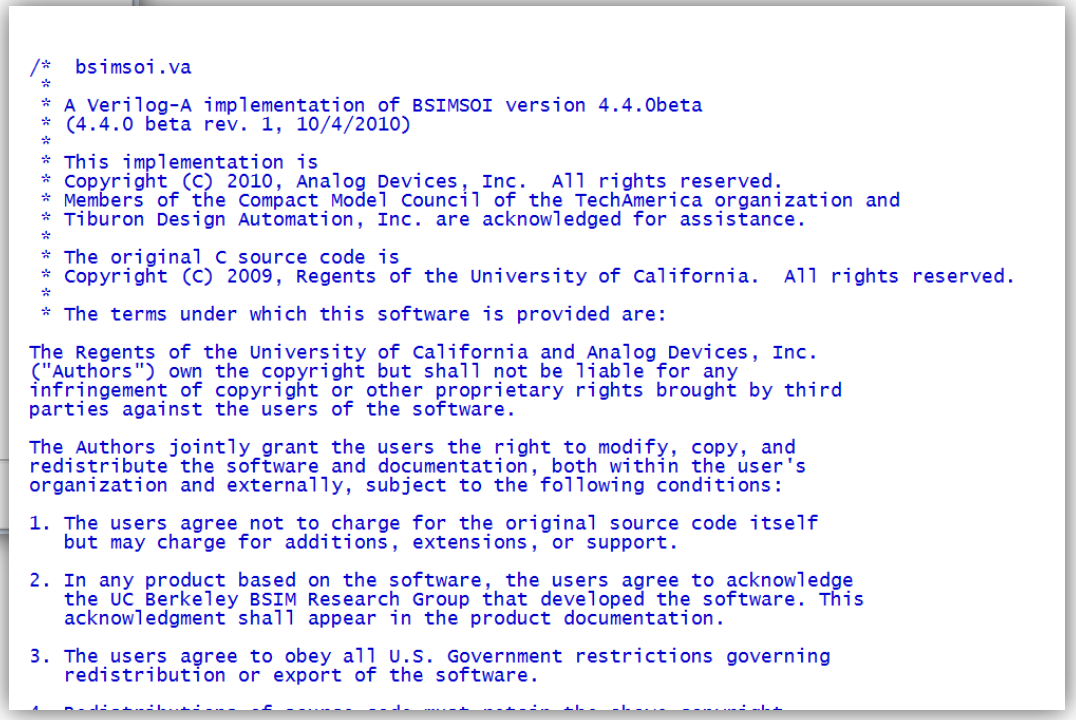
- **Verilog-A is the development language of choice for new compact models**



And coming soon…

- **HiSIM and BSIMSOI models**





```
/*  bsimsoi.va
 *
 * A Verilog-A implementation of BSIMSOI version 4.4.0beta
 * (4.4.0 beta rev. 1, 10/4/2010)
 *
 * This implementation is
 * Copyright (C) 2010, Analog Devices, Inc.  All rights reserved.
 * Members of the Compact Model Council of the TechAmerica organization and
 * Tiburon Design Automation, Inc. are acknowledged for assistance.
 *
 * The original C source code is
 * Copyright (C) 2009, Regents of the University of California.  All rights reserved.
 *
 * The terms under which this software is provided are:

The Regents of the University of California and Analog Devices, Inc.
("Authors") own the copyright but shall not be liable for any
infringement of copyright or other proprietary rights brought by third
parties against the users of the software.

The Authors jointly grant the users the right to modify, copy, and
redistribute the software and documentation, both within the user's
organization and externally, subject to the following conditions:

1. The users agree not to charge for the original source code itself
   but may charge for additions, extensions, or support.

2. In any product based on the software, the users agree to acknowledge
   the UC Berkeley BSIM Research Group that developed the software. This
   acknowledgment shall appear in the product documentation.

3. The users agree to obey all U.S. Government restrictions governing
   redistribution or export of the software.
```

Overcoming the Barriers

- **Model Developer's Environment**
  - **Easy to code**
  - **Not always as easy to debug**
- **End user**
  - **Verilog-A models are behind the scenes, either in the simulator or PDK**
  - **Convergence issues related to numerical precision**
    - MOS-AK 2008
- **Time domain – centric**
  - **Not always what RF models are best described in**

Model Development Debugging

- **Basic**
  - **$strobe – outputs every converged iteration**
  - **$debug – outputs every call to module**
  - **Use macros to disable in general use**

```
`ifdef DEBUG
```

- **Compiler flags for runtime**
  - **Too expensive for production code**
  - **Very useful during development phase**
- **Compile time diagnostics**

Compiler Diagnostics

```
$ vacomp -Xinfo angelov_gan.va

module 'angelov_gan_va', no memory states
CML analog block

=== Summary information for module 'angelov_gan_va':

Branch information:
  <unnamed>(bi, gsi) : Current Branch (implicit)
  <unnamed>(bi, si) : Switch Branch
  <unnamed>(di, d) : Switch Branch
  <unnamed>(di, rf) : Current Branch (implicit)
  <unnamed>(di, si) : Current Branch (implicit)
  <unnamed>(g, di) : Current Branch (implicit)
  <unnamed>(g, gi) : Switch Branch
  <unnamed>(gdi, di) : Current Branch (implicit)
  <unnamed>(gi, di) : Current Branch (implicit)
  <unnamed>(gi, gdi) : Switch Branch
  <unnamed>(gi, gsi) : Switch Branch
  <unnamed>(gii, si) : Current Branch (implicit)
  <unnamed>(gsi, gii) : Switch Branch
  <unnamed>(gsi, si) : Current Branch (implici
  <unnamed>(rf, si) : Switch Branch
  <unnamed>(si, s) : Switch Branch
  <unnamed>(t, <gnd>) : Current Branch (implicit)
  <unnamed>(xt1, <gnd>) : Current Branch (implicit)
  <unnamed>(xt1, xt2) : Voltage Branch
  <unnamed>(xt2, <gnd>) : Current Branch (implicit)

Branch ddt operators:
  [ line 526, col 15 ]
  [ line 537, col 41 ]
  [ line 541, col 24 ]
  [ line 546, col 43 ]
  [ line 550, col 25 ]
  [ line 558, col 16 ]
  [ line 499, col 26 ]
  [ line 525, col 31 ]

Potential memory states:
  [ none ]

=== End of summary information for module 'angelov_gan_va':
```

```
if (Rs > 0.0) begin
    I(si,s) <+ V(si,s) / Rs;
    I(si,s) <+ white_noise(4.0 * `P_K * T / (Rs), "Rs");
end
else
    V(si,s) <+ 0.0;
```

```
I(xt2)      <+ V(xt2);
```

End User Performance

- No theoretical reason for Verilog-A to be inferior in performance to built-ins
- Not as critical in HB since model evaluation is less important than for transient analyses
- Model coding can have a big influence
  - execution speed
  - memory use
  - Convergence/numerical stability
  - MOS-AK 2009

Analog/RF Models

- **Special considerations in Analog/RF modeling:**
  - **Support for special RF analyses (e.g., Harmonic Balance, Shooting, Envelope)**
  - **Support for noise analysis is important**
    - Correlation effects
    - Colored noise
  - **Frequency-dependent characteristics**

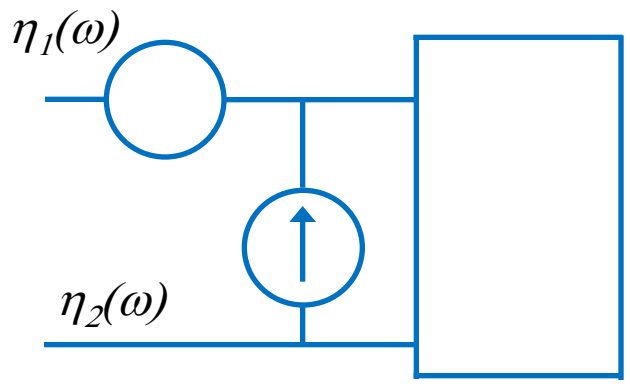
Convergence

- RF simulators more susceptible to convergence problems
- Models must be charge conserving
  - Written in terms of charge, not capacitance
- Models must behave well for large pin voltages
  - Including derivatives (even though these are generated by the compiler)
- Verilog-A allows users to easily create non-physical models

Noise Analysis

- **Noise analysis is an area of key importance for RF simulation**
- **Verilog-A has comprehensive support for**
  - **Basic noise**
  - **correlated noise sources**
  - **colored noise sources**
    - rational polynomial filters
    - flicker noise ($1/f^a$)
    - table-based noise sources

Noise Analysis (cont.)

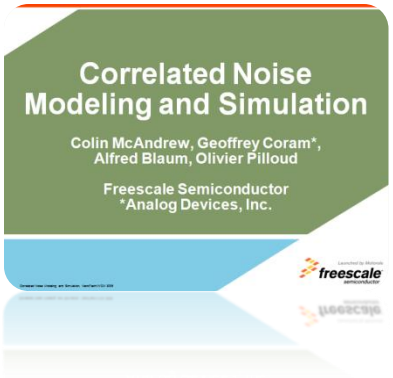


Correlation

```
A = white_noise(K);
B = white_noise(P1-K);
C = white_noise(P2-K);
η1 = A+B;
η2 = A+C;
```

```
x = white_noise(...);
Kj = ddt(K); // For imaginary
correlation coefficient
Xc = laplace_pz(noise_pwr,
{Poles}, {Zeros});
```

See Correlated Noise Modeling and Simulation by McAndrew et al.



Modeling Burst Noise

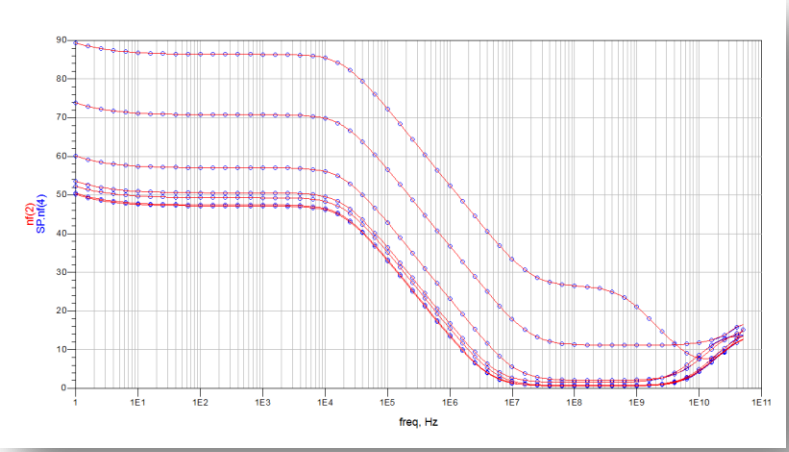
Implementing burst noise of the form can be done using the non-standard extension, `$realfreq`:

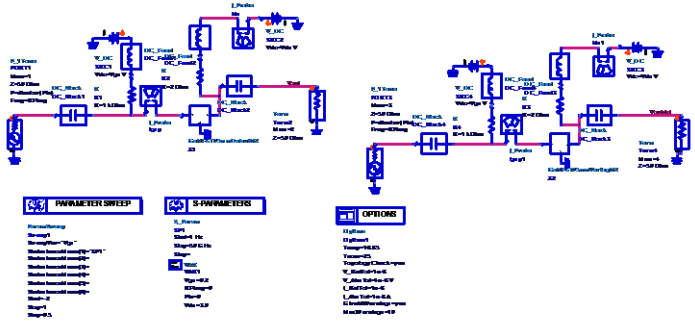
$$\langle i^2 \rangle = \frac{Kb \times Idc^{Ab}}{1 + \left(\frac{f}{Fb}\right)^2}$$

```
I(di,si) <+ white_noise(NoisePwr * Klf, "Burst") * sqrt(1/(1+pow($realfreq/Fgr,2)));
```

Or it can be done using standard Verilog-A functions by coloring the noise via a Laplace transform.

```
I(di,si) <+ laplace_nd(white_noise(Klf*NoisePwr, "burst"), {1}, {1,0,-1/(`M_TWO_PI*Fgr*`M_TWO_PI*Fgr) });
```

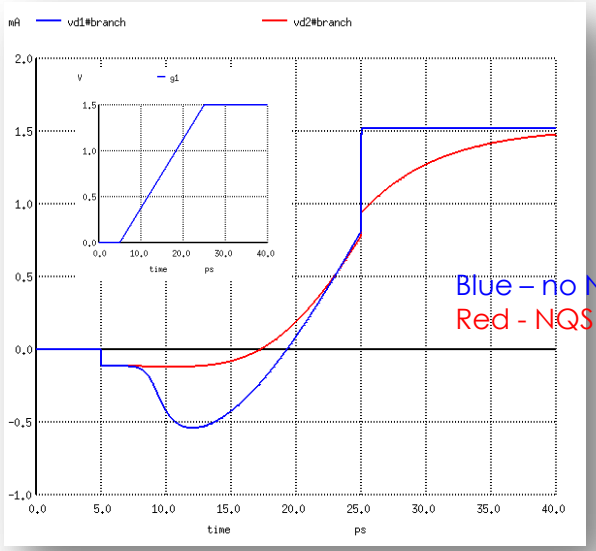




Compare built-in (c-code) version of Angelov to Verilog-A

Modeling NQS Effects in RF

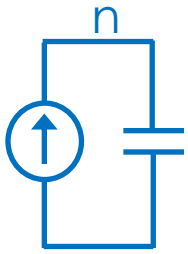
- At high frequencies with smaller devices, non-quasi-static effects become important
- Non-quasi-static effects are typically modeled with a delay on the charge
  - But this explicitly uses *time*



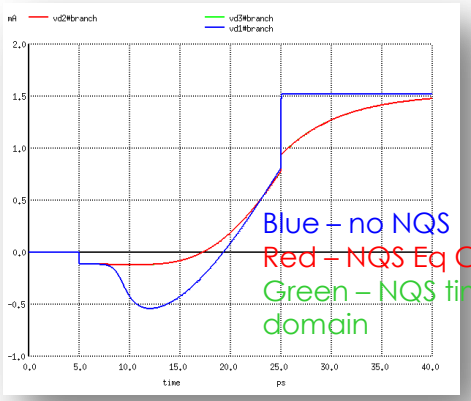
$$q(t_i) = \frac{q(t_{i-1}) + \frac{\Delta t}{\tau} Q(t_i)}{1 + \frac{\Delta t}{\tau}}$$

Modeling NQS Effects in RF

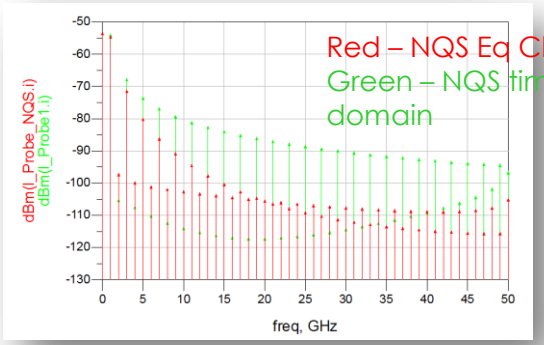
- Instead, model using an additional node with current source and capacitor




```
I(n) <+ (Qb_nqs - Qb) / taub + ddt(V(n));
Qb_nqs = V(n);
```



- NQS model now works in time and frequency domain



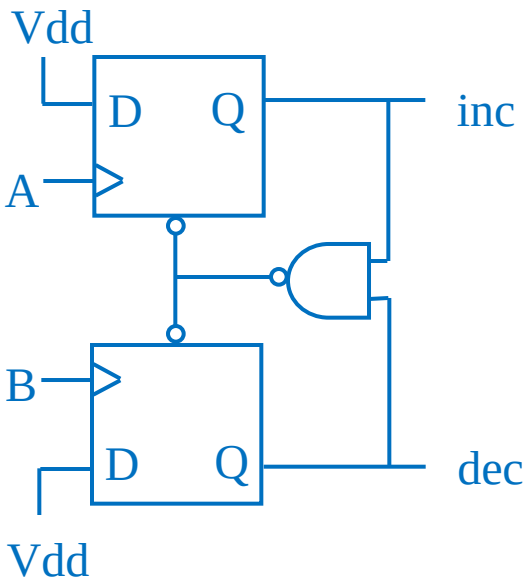
RF Restrictions

- **Explicit use of time `$abstime`**
- **Analog Operators**
  - **Allowed:**
    - Differentiation **`ddt()`**, **`ddx()`**
    - Delay **`absdelay()`**
    - Laplace **`laplace()`**
    - Integration **`idt()`** *without* initial conditions
  - **Others are:**
    - Not safe for RF analysis
    - Not (typically) useful for compact modeling
    - Avoid any analysis() dependent code

Memory States and Events

- Also known as hidden states
- models may be written to retain values from one timepoint to the next
  - If used in assignment before it is assigned, it will have the value of the previous iteration
- Variables are initialized to zero on first call to module
- Compact models should not use them
  - could cause unexpected behavior

Behavioral Modeling for RF



Phase-Frequency Detector

- Convenient to model with memory states
  - great for transient
  - not suitable for (e.g.) harmonic balance, PSS
- Solution 1: use physical understanding
- Solution 2: use simplified gates/transistors

RF Analysis and Verilog-A

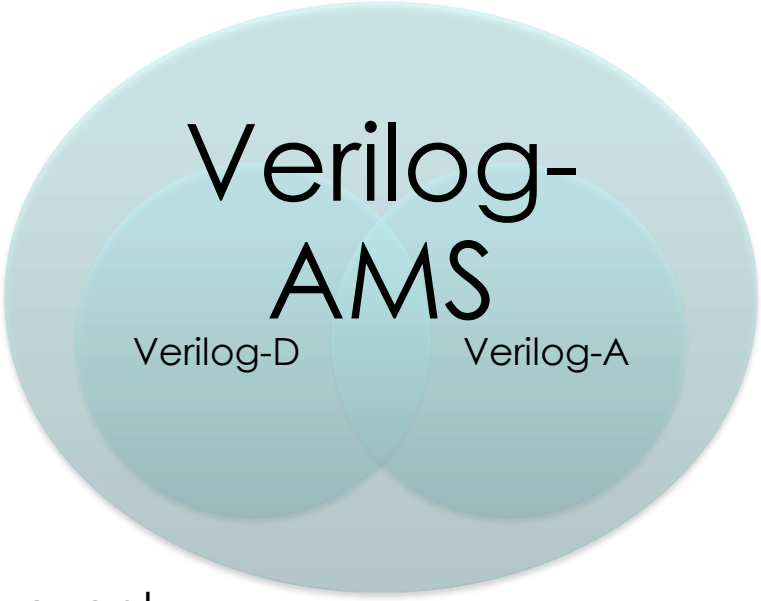
- **Verilog-A is capable of supporting a full range of modeling capabilities from the behavioral level to the transistor level**
  - **RF analysis engines (HB, ENV, PSS) are unable to deal with some behavioral constructs**

Remaining Barriers

- RF models are often better described in frequency domain
  - Frequency-domain characteristics are restricted to be “physical” (e.g., causal)
  - Responses shaped by rational polynomial filters
- Support for standard file formats
  - S-parameters
  - CITIfile
- End users tend to design with one flow
  - Portability between simulators is less important

Verilog Languages

Modeling of mixed-signal systems



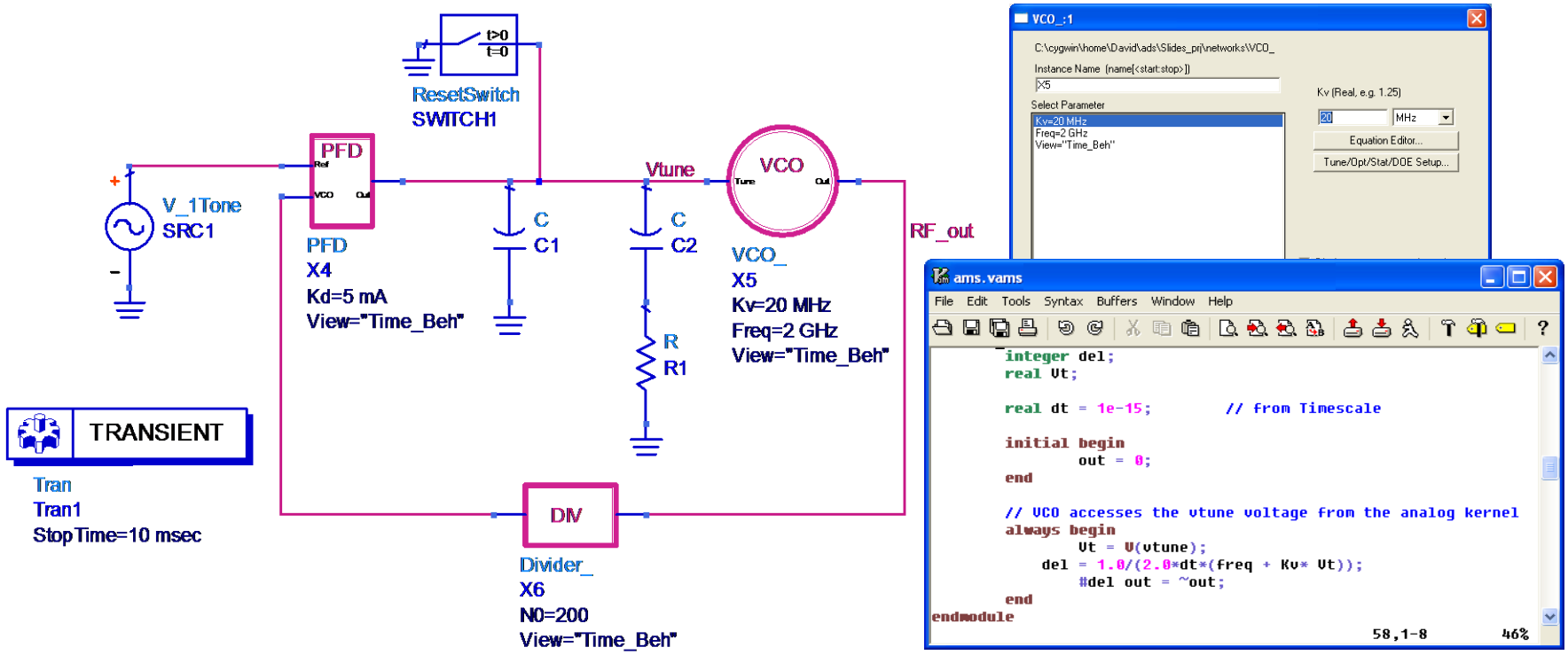
Modeling of discrete event (digital) systems

Modeling of continuous-time (analog) systems

Verilog-AMS Benefits

- Simulate entire system (A & D) in its native design format using realistic signals and with their optimum simulation algorithms
- Much faster simulations
- Industry standard language
- Able to use models, IP and test benches that are only available, or are more easily created, in Verilog or Verilog-AMS format
- Applies to both system level (behavioral/ procedural) as well as transistor/gate level) to fully support Tops-Down Design process

AMS PLL Time Domain Design

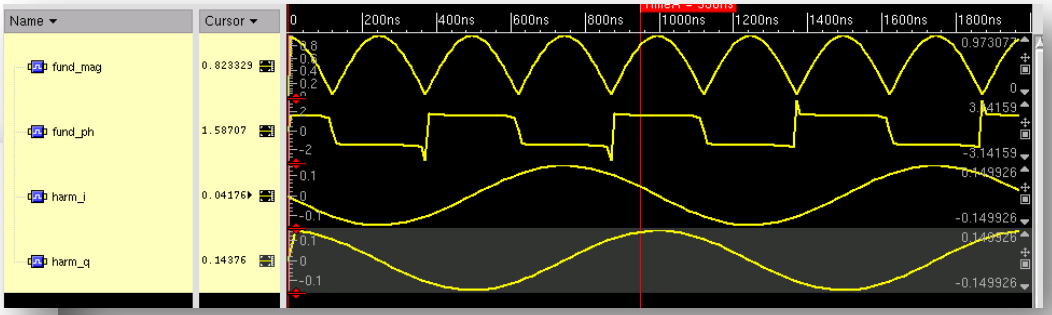


- Using PLL models in their behavioral, time domain view, Transient analyses can be used to design, optimize and verify lock time, spurious and jitter performance.

Envelope Demodulation

In envelope simulations, instantaneous envelope information is available to the analog simulator

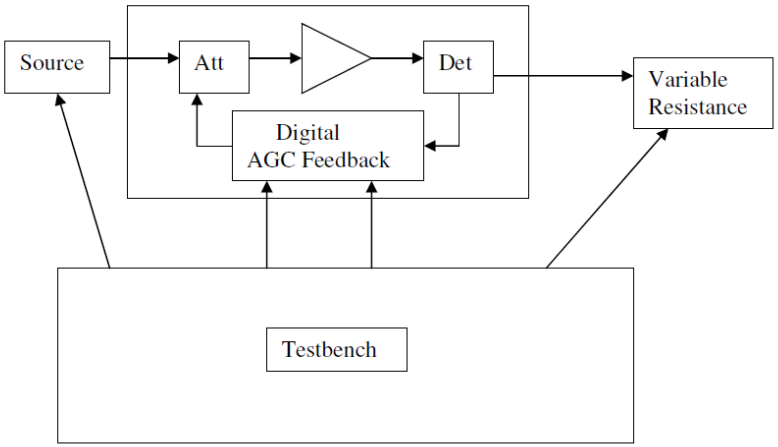
```
module  block(in);
inout in;
electrical in;
real fund_mag, fund_ph, harm_i, harm_q;

    always begin
        fund_mag=$demod_mag(V(in),1G);
        fund_ph =$demod_phase(V(in),1G);
        #10;
    end

    always begin
        harm_i  =$demod_I(V(in),3G);
        harm_q  =$demod_Q(V(in),3G);
        #20;
    end

    analog I(in) <+ V(in);

endmodule
```

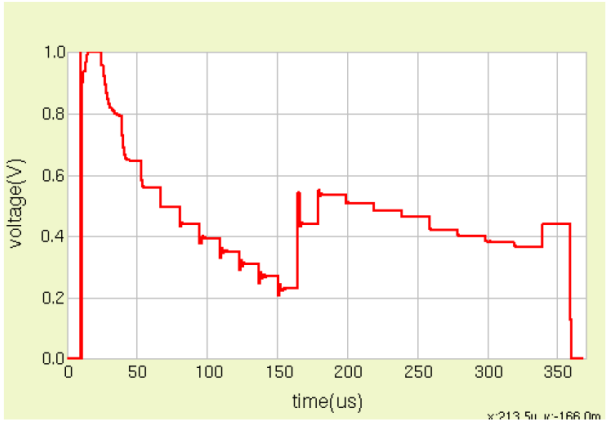


Extensions to AMS language allow direct demodulation of magnitude, phase, I, Q or envelope waveform in the digital domain.

Example: AGC circuit

- **Digital control is very common in RF circuits**





- **Besides actual elements, test bench can be implemented as a digital element to test circuit in appropriate modes during AMS simulation**

Summary

- Verilog-A is the standard for compact model development
- RF modeling has other requirements related to frequency dependences, but Verilog-A can usually accommodate them
- Verilog-AMS provides critical support for simulation of mixed signal circuits
- The language is being extended to for efficiently simulate modulated signals