Real-Valued Verilog Models for Analog Circuits

Prof. Chien-Nan Liu
Institute of Electronics
National Chiao-Tung Univ.

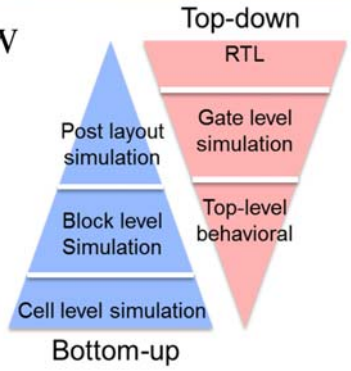
Tel: (03)5712121 ext:31211
E-mail: jimmyliu@nctu.edu.tw
http://mseda.ee.nctu.edu.tw/jimmyliu

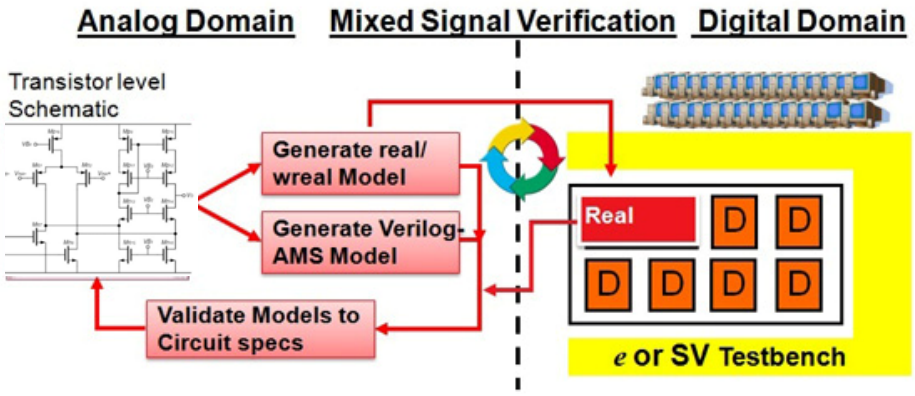
Outline

- Introduction
- Real-Valued Behavioral Models in Verilog
- Portable Models for Verilog/Verilog-A
- Test Results
- Conclusions

Difficulty in Mixed-Signal Verification

- Opposite circuit design/verification flow
  - Analog: Bottom-up; Digital: Top-down
- Use behavioral model to replace the analog part
  - Only way for system-level verification

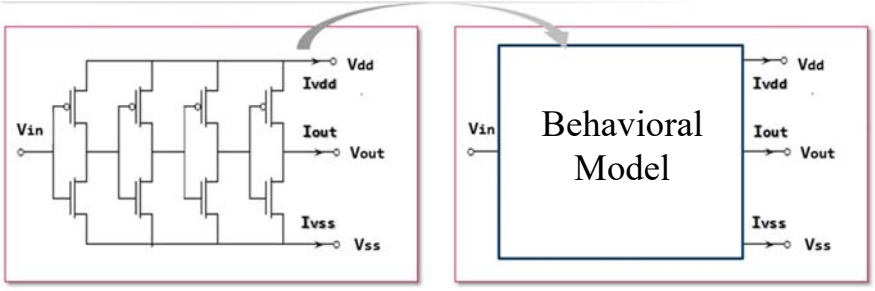




Ref: Sathish Bala, "Smart Devices and How They Affect Your Mixed-Signal SOC Verification", Cadence Mixed-Signal Design Blogs, Feb. 2013.

Behavioral Model

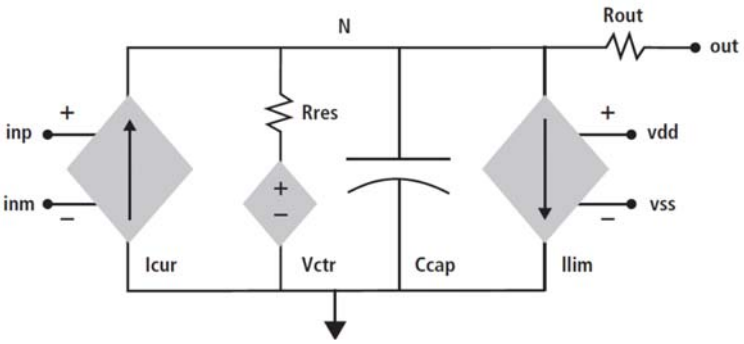
- A simplified circuit block that model of the same input-output functionality
  - Capture just the essential behaviors
- Adjustable equation-based verification
  - Composed of circuit equations
  - Modified by different size, process variation…
- Tradeoff between **accuracy** and **speed**



Modeling Strategy is Important

- Different abstraction will result in different runtime, even the same language (ex: Verilog-A) is used !!

Equivalent Circuit Model




```
I(N,vss) <+ -Gm*V(inp,inm);
I(N,vss) <+ (V(N,vss)-V(vdd,vss)/2)/Rres;
I(N,vss) <+ ddt(Ccap*V(N,vss));
I(N,vss) <+ `fclip(V(N,vdd),1,40m)
      - `fclip(V(vss,N),1,40m);
I(N,out) <+ V(N,out) / Rac;
```

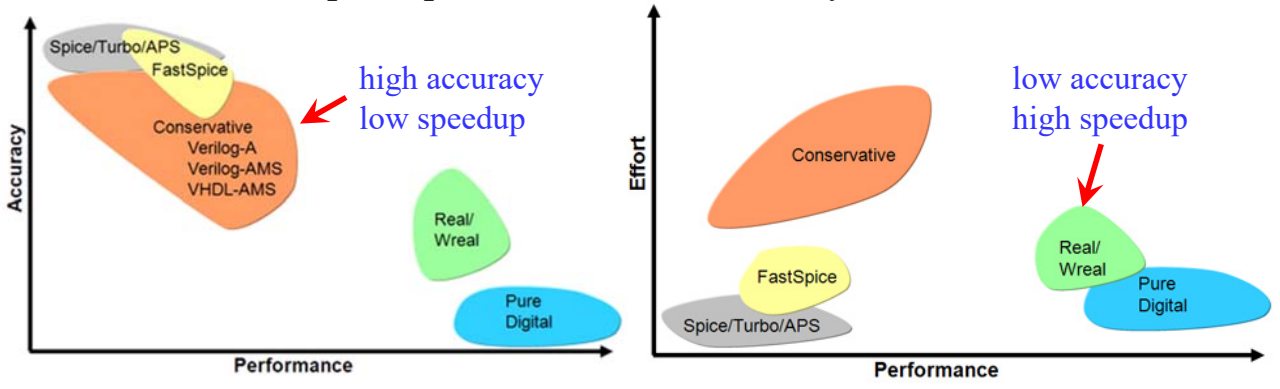
Fully Behavioral Model

```
module opamp (vout, vin_p, vin_n );
 inout vin_p , vin_n;
 output vout;
 electrical vin_p, vin_n, vout;

 analog begin

   I(vin_p) <+ 0.0;
   I(vin_n) <+ 0.0;
   V(vout) <+ V(vin_p, vin_n ) *Av;
 end
endmodule
```

Fast!

Why Using Verilog Model ?

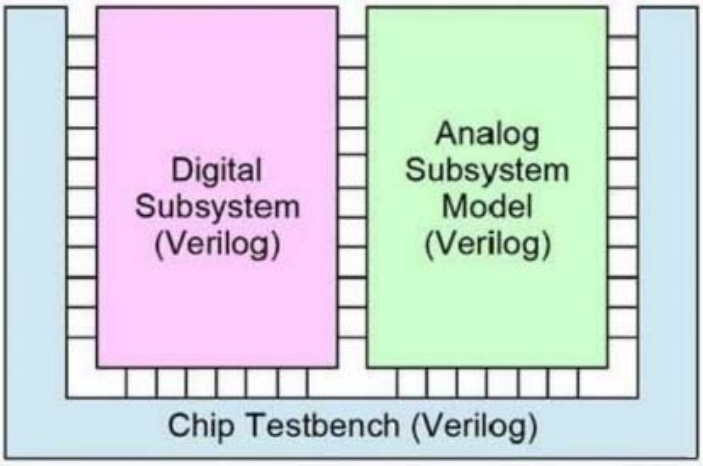
- Verilog-A does not fit well in a pure digital event-driven validation framework
  - Although accuracy is good with Verilog-A models, the gained speedup is limited
- Real-valued Verilog model provides another solution
  - Further speedup with sacrificed accuracy



Ref: G. Richard, "Analog/Mixed-Signal Behavioral Modeling – When to Use What", Cadence Mixed-Signal Design Blogs, Feb. 2011.

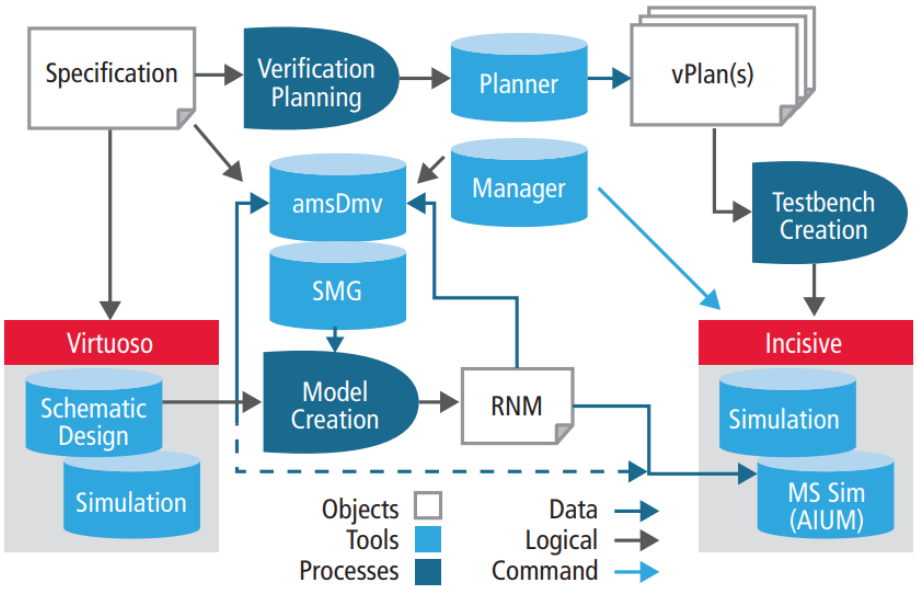
Adv. of Real-Valued Verilog Model

- Provide pin-accurate behavioral model for system-level validation in the same environment
  - Assertion-based verification (ABV) and metric-driven verification can also be applied in mixed-signal simulation
- Portable between digital and analog design environments



Adv. of Real-Valued Verilog Model

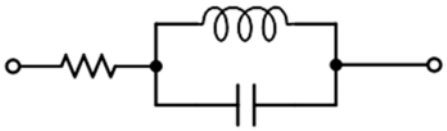
- Full-chip verification can be run with digital solvers
- Rapid simulation even for large SoCs
  - If accuracy loss is acceptable



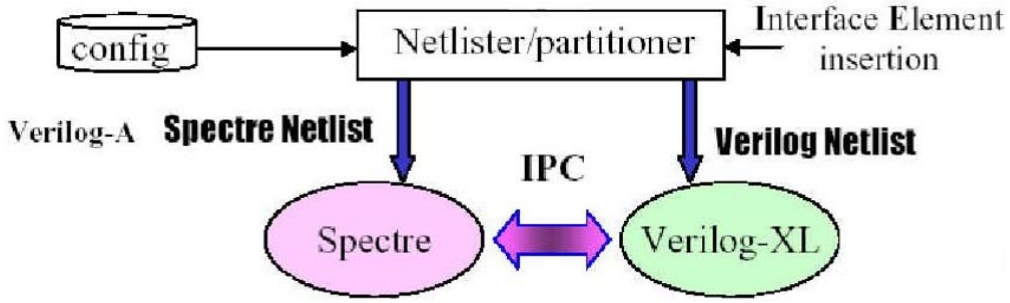
Ref: S. Balasubramanian, P. Hardee, "Solutions for Mixed-Signal SoC Verification Using Real Number Models," Cadence White Papers, 2013.

Adv. of Real-Valued Verilog Model

- Still able to model parasitic coupling
  - Although several approximations are required with Verilog



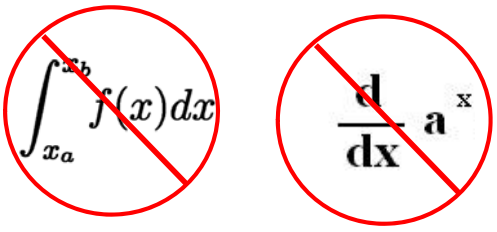
- No need for iterative analog SPICE engine
  - Eliminate the communication between different engines

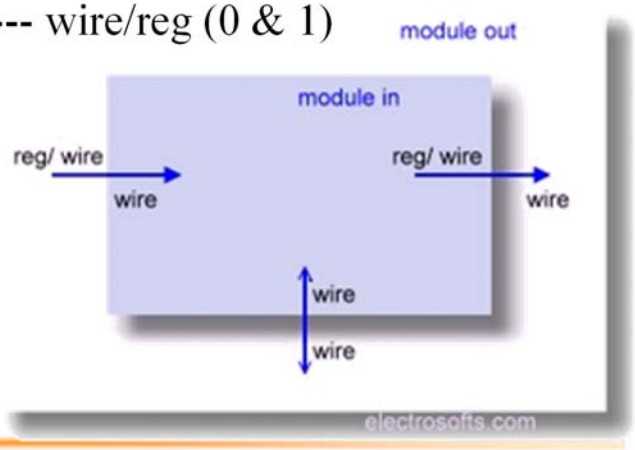


Ref: CIC training manual

Difficulty to Model Analog Behaviors

- Operators are limited in digital languages
  - Timing parameters: rise/fall time? slew rate?
  - Math operators: differential? integral? Laplace transform??
  - A per-timestep format is often required for approximation
- Interface between modules require special handling
  - Only support digital ports --- wire/reg (0 & 1)
  - PWL or RNM waveforms



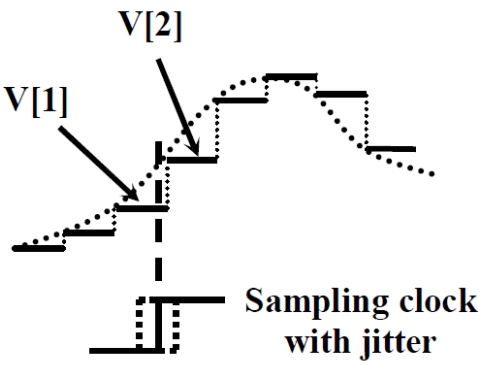


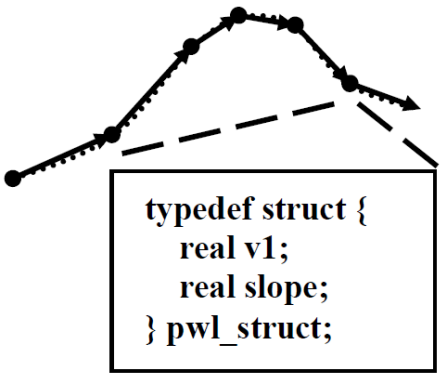
Outline

- Introduction
- Real-Valued Behavioral Models in Verilog
- Portable Models for Verilog/Verilog-A
- Test Results
- Conclusions

Represent Continuous-Time Signals

- Two popular approaches to represent analog signals in a digital simulator
  - Piecewise-constant (PWC)
  - Piecewise-linear (PWL)

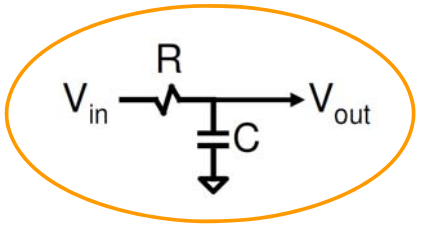




Ref: S. Liao and M. Horowitz, "A Verilog piecewise-linear analog behavior model for mixed-signal validation," IEEE TCAS-I, vol. 61, no. 8, pp. 2229–2235, Aug. 2014.

Model Comparison

- Real-valued models
  - Difference equations
  - $\Delta V_{out} = \Delta t \frac{I_r}{C}$
  - $I_r = (V_{in} - V_{out})/R$
- Accurate for small steps
- Much faster simulations
- Ex: RC filter



- Verilog-A models (slow)
  - Differential equations
  - $dV_{out} = dt \frac{I_r}{C}$
  - $I_r = (V_{in} - V_{out})/R$
  - Simulator chooses Δt

```
input Vin;     // input voltage
real Ir;       // resistor current
real Vout;     // output voltage
real lsttim;   // last time model was evaluat

initial begin
  Vout = 0.0;
  lsttim = 0.0;
end

Ir = (Vin - Vout) / R;
Vout = Vout + Ir * ($abstime - lsttim) / C;
lsttim = $realtime;
```

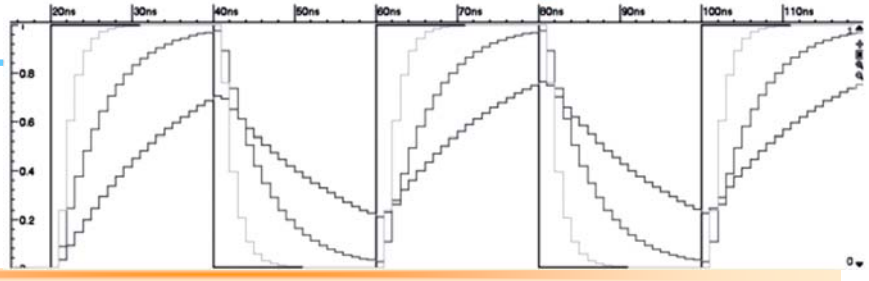
Δt is user specified

Ref: Bill Ellersick, "Real Portable Models for System/Verilog/A/AMS", SNUG 2010

Another Way: Discrete Transfer Function

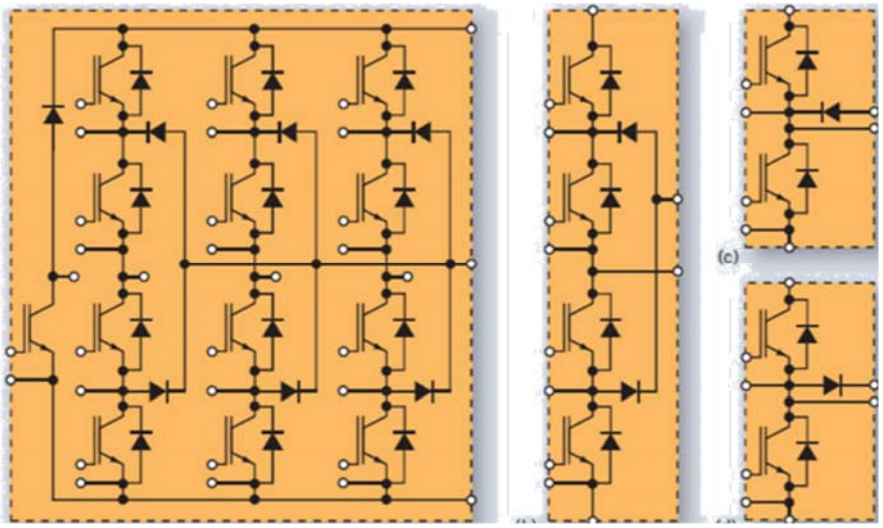
- In Laplace notation, the transfer of a RC low-pass filter is
  H(S) = K / (τS + 1), K = filter gain, τ = time constant
- Using Bilinear transform, its z-transform function is
  $H(z) = K * (n_0 + n_1 z^{-1}) / (d_0 + d_1 z^{-1})$, where $n_0 = n_1 = 1$
  $d_0 = 1 + 2\tau / Ts$, $d_1 = 1 - 2\tau / Ts$ (Ts = sampling rate)
- So, **$OUT_{new} = K*(IN_{new}*n_0 + IN_{old}*n_1 - OUT_{old}*d_1) / d_0$**

```
always #(Ts) begin   // sample input every Ts
   OUTval = K * (IN +INold - OUTval*d1) / d0;
   INold = IN;
end
```



Proper Circuit Partition

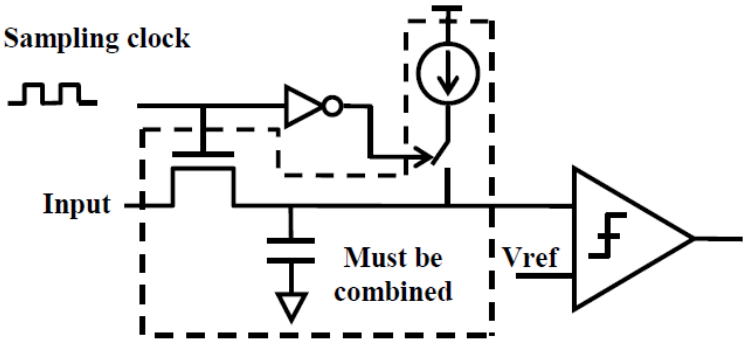
- Partitioning large circuits into unidirectional blocks
  - Easy to derive closed-form equations of each block for quick evaluation



Avoid Time Integration

- Ex: Single-slope ADC
  - Combining analog blocks to form 2 unidirectional modules
  - Tracking + ramping
  - Closed form equations:

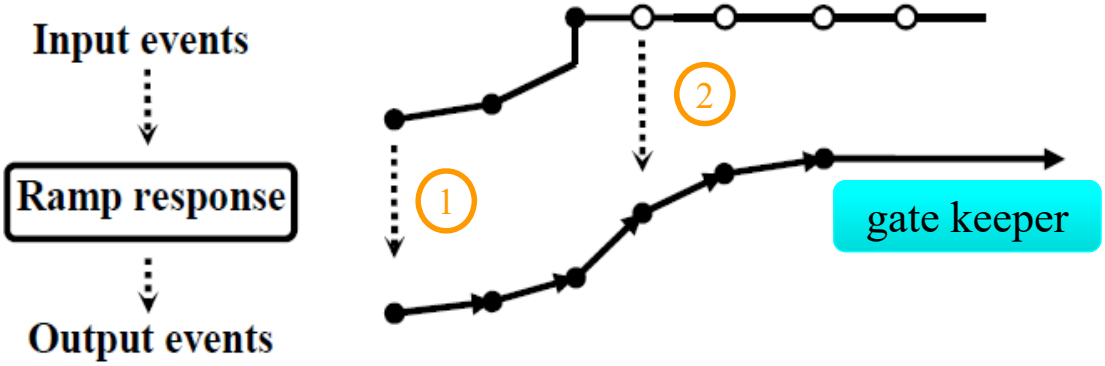
$$V_{out} = \beta t + \beta RC\left(e^{-\frac{t}{RC}} - 1\right) + \alpha\left(1 - e^{-\frac{t}{RC}}\right) + V_0 e^{-\frac{t}{RC}}$$



Ref: S. Liao and M. Horowitz, "A Verilog piecewise-linear analog behavior model for mixed-signal validation," IEEE TCAS-I, vol. 61, no. 8, pp. 2229–2235, Aug. 2014.

Reshape Model Output

- Ramp response appears very often in digital circuits
  - Not reasonable for analog models
- A gate keeper is often required to "smooth" the sharp output signals

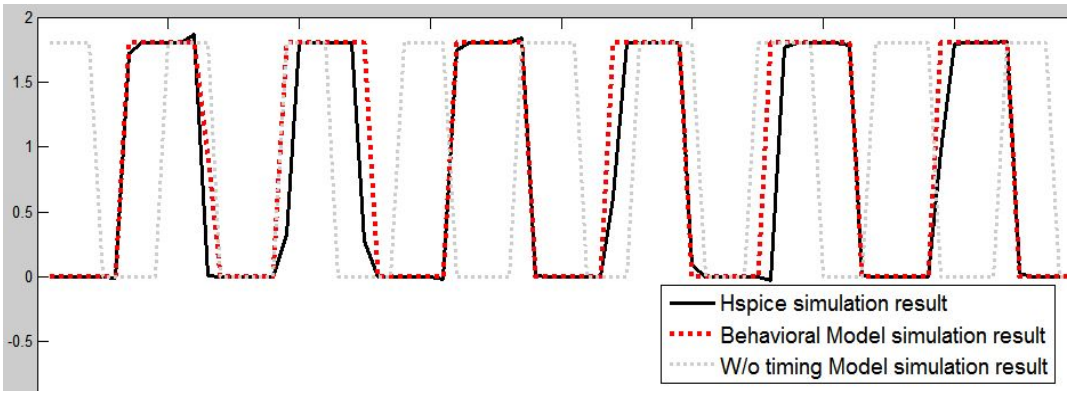


Ref: S. Liao and M. Horowitz, "A Verilog piecewise-linear analog behavior model for mixed-signal validation," IEEE TCAS-I, vol. 61, no. 8, pp. 2229–2235, Aug. 2014.

Timing Models

- Digital simulation often use zero-delay mode for functional validation
  - However, timing information (rise/fall/delay) is important to analog circuits
- Calibration is required to fix the timing parameters

	Hspice	w/o timing model	With timing model
Lock frequency	800MHz	1000MHz	793MHz



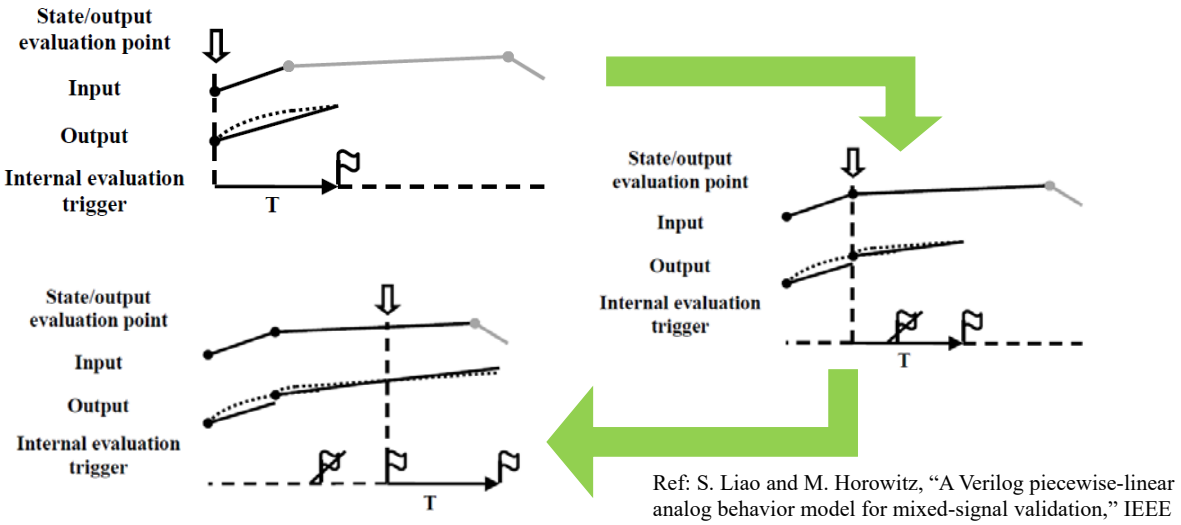
Ref: J.-Y. Chen, S.-W. Wang, C.-H. Lin, C.-N. Liu, Y.-J. Lin, M.-J. Lee, Y.-L. Lo, S.-Y. Kao, "Automatic Behavioral Model Generator for Mixed-Signal Circuits Based on Structure Recognition and Auto-Calibration", in Proc. Int'l SOC Design Conf., Nov. 2015

Interface Modeling

- According to the input/output types, different approaches are adopted to model the interface
  - Digital in & out
    - → use standard Verilog input/output format
  - Analog in & out
    - → model input/output signals as PWL form or
    - → use real-value connection approaches
  - Digital ⟷ Analog
    - → use PWL waveform or direct connection
- Be careful about the **multi-driven** problem

PWL Waveforms as Input/Output

- Determine the system output according to the transfer function for every input
- Set the evaluation period T based on the required accuracy
  - Calculate the output value for every T seconds



Ref: S. Liao and M. Horowitz, "A Verilog piecewise-linear analog behavior model for mixed-signal validation," IEEE TCAS-I, vol. 61, no. 8, pp. 2229–2235, Aug. 2014.

Real Value Connection (1/2)

- Directly assign the real value form module A to module B --> skip the port connection
- Format : **Module_name . Real_value_name**

Module b

```
module inv(in,out);
input in;
output reg out;
real Vin,Vout;
parameter vdd=1.8,gnd=0.0;
always@(in)
begin
    out=~in;
    if(Vin>(0.5*vdd))
    begin
        Vout=gnd;
    end
    else
    begin
        Vout=vdd;
    end
end
endmodule
```

Top Module

```
inv a(.in(in),.out(net));
inv b(.in(net),.out(out));
always @*
b.Vin = a.Vout;
endmodule
```

Module a

```
module inv(in,out);
input in;
output reg out;
real Vin,Vout;
parameter vdd=1.8,gnd=0.0;
always@(in)
begin
    out=~in;
    if(Vin>(0.5*vdd))
    begin
        Vout=gnd;
    end
    else
    begin
        Vout=vdd;
    end
end
endmodule
```


Real Value Connection (2/2)

- Real values can be passed through ports in Verilog-AMS and latest version of Verilog → **wreal** type
  - ***wreal*** = wire + real value
  - Supported in Verilog-AMS and SystemVerilog 2012
- Used as normal wire variables → only for I/O ports
  - ***real*** type for internal variables

Passing real values for those input ports

Internal variables use **reg** and **real** types

```
module sdm_rnm (Ain, clk_1mhz, reset_n, Dout);

  input  Ain, clk_1mhz, reset_n;
  output Dout;

  wreal Ain, dlay[2:1];
  reg   Dout;
  real sdm_sign_val;
  real sdm_sum1, real sdm_sum2;
```

Real-Valued Verilog Model (1/2)

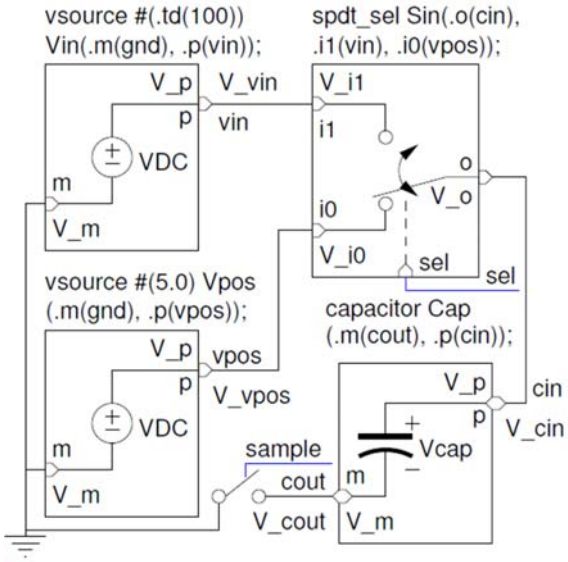
- An example of continuous voltage/current values

```
`timescale 1us/100ps
module vin_pos_switch_tb();
// top-level signals and variables
wire gnd = 1'b0;
wire vin, vpos, cin, cout;
reg sample=1'b0, sel=1'b0;
real V_vin, V_vpos, V_cin, V_cout;
// sub-module instantiations
vsource #(.td(100)) Vin (.m(gnd), .p(vin));
vsource #(5.0) Vpos (.m(gnd), .p(vpos));
spdt_sel Sin(.o(cin),.i1(vin),.i0(vpos),.sel(sel));
capacitor Cap (.m(cout), .p(cin));
// sample-switch modeling
assign cout = (sample) ? gnd : 1'bz;
always @* V_cout = (sample) ? 0.0 : Cap.V_m;
    always @* Cap.V_m = V_cout;
    // connectivity for variables
always @* V_vin = Vin.V_p;
    always @* Sin.V_i1 = V_vin;
always @* V_vpos = Vpos.V_p;
    always @* Sin.V_i0 = V_vpos;
always @* V_cin = Sin.V_o;
    always @* Cap.V_p = V_cin;
```

Declaration

Circuit connect

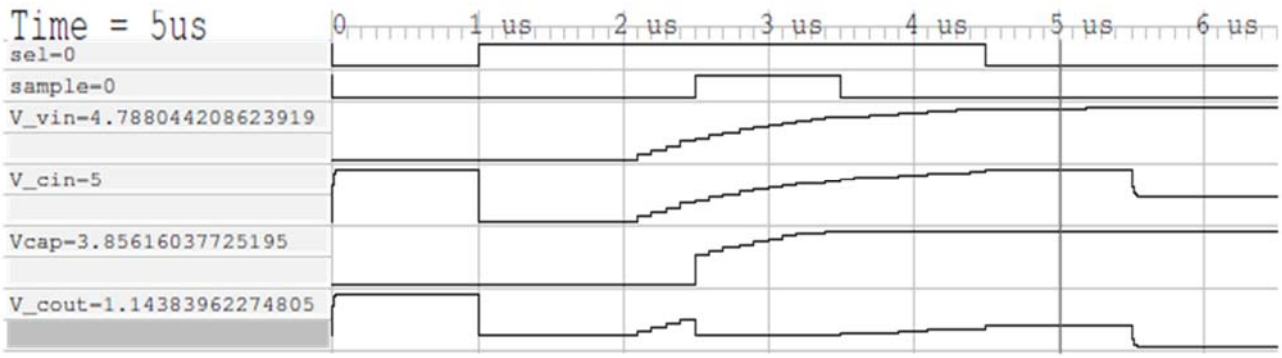
Assign real type variable



Ref: C.Wenger, "Method of Modeling Analog Circuits in Verilog for Mixed-signal Design Simulations," IEEE European Conf. on Circuit Theory and Design, 2013.

Real-Valued Verilog Model (2/2)

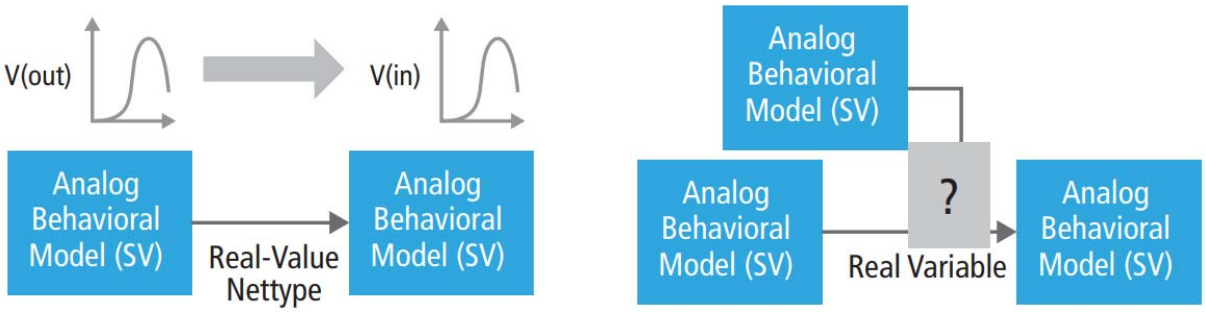
- Output waveform of continuous voltage/current



Ref: C.Wenger, "Method of Modeling Analog Circuits in Verilog for Mixed-signal Design Simulations," IEEE European Conf. on Circuit Theory and Design, 2013.

Multi-driven Problem of RVM

- In legacy Verilog code
  - No bidirectional real number ports
  - Only one driver allowed on each port connection
  - No direct support for real valued signals
- In latest verision SystemVerilog
  - Support Nettype and User-defined types (UDT)
  - Similar to structure in C → pack multiple variables



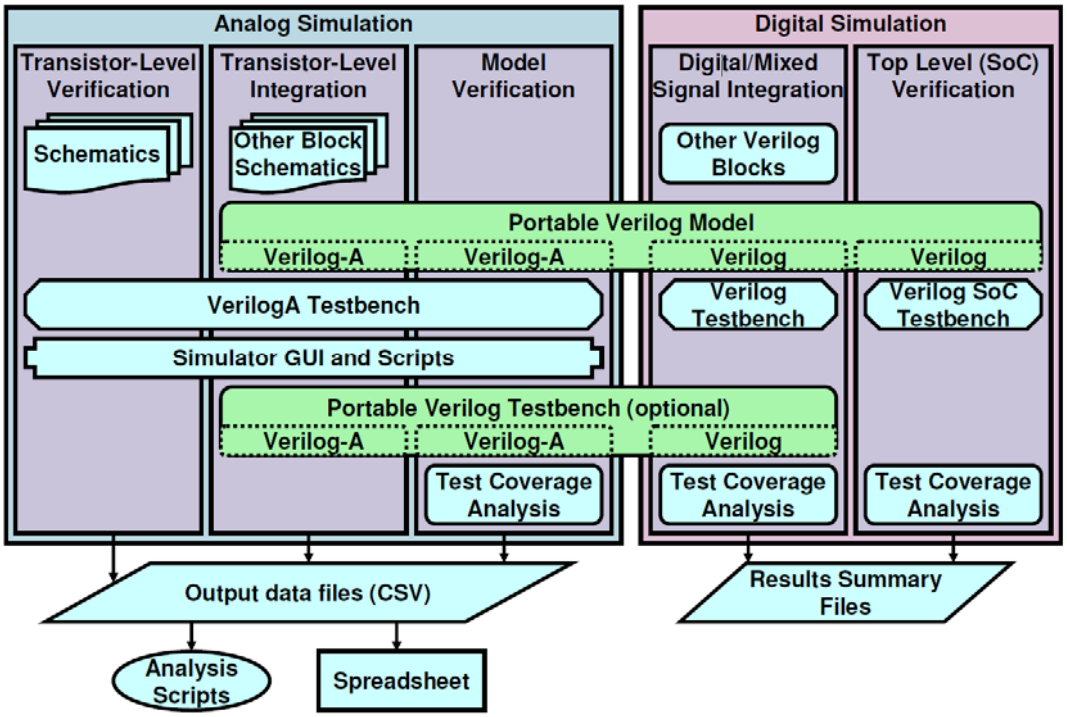
Ref: S. Balasubramanian, P. Hardee, "Solutions for Mixed-Signal SoC Verification Using Real Number Models," Cadence White Papers, 2013.

Outline

- Introduction
- Real-Valued Behavioral Models in Verilog
- Portable Models for Verilog/Verilog-A
- Test Results
- Conclusions

Portable Methodology Flow

- Create a common code that is portable to Verilog/A



Ref: Bill Ellersick, "Real Portable Models for System/Verilog/A/AMS", SNUG 2010.

Portable Methodology

- Use discrete-time features common to Verilog/A
  - Common code to manipulate reals and integers
  - Use powerful `define to write portable models

Define Macro

Verilog

```
`define AnalogInput wire real
`define AnalogOutput real
`define LogicInput wire
`define LogicOutput wire
```

Verilog-A

```
`define AnalogInput electrical
`define AnalogOutput electrical
`define LogicInput electrical
`define LogicOutput electrical
```

Verilog/A code

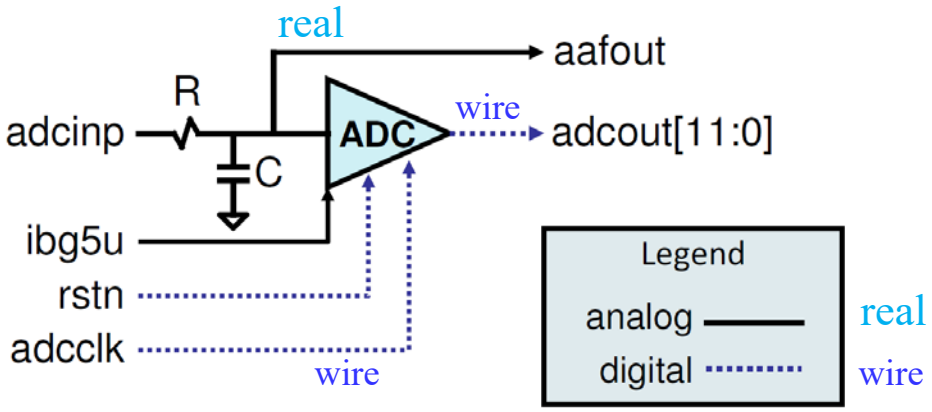
```
`LogicInput [11:0] adcout;
`AnalogInput aafout;
`AnalogOutput ibg5u;
`AnalogOutput adcinp;
`LogicOutput adcclk;
`LogicOutput rstn;
 wire [11:0] adcout;
 real aafout;
 wire real ibg5u;
 wire real adcinp;
 wire adcclk;
 wire rstn;
```

```
electrical [11:0] adcout;
electrical aafout;
electrical ibg5u;
electrical adcinp;
electrical adcclk;
electrical rstn;
```

Ref: Bill Ellersick, "Real Portable Models for System/Verilog/A/AMS", SNUG 2010.

Verilog v.s. Verilog-A

- Both Verilog and Verilog-A support:
  - Couple real or binary values in and out of module ports
  - Specify when to update values
  - Equations to update real and integer values
- Example : ADC with RC filter



Ref: Bill Ellersick, "Real Portable Models for System/Verilog/A/AMS", SNUG 2010.

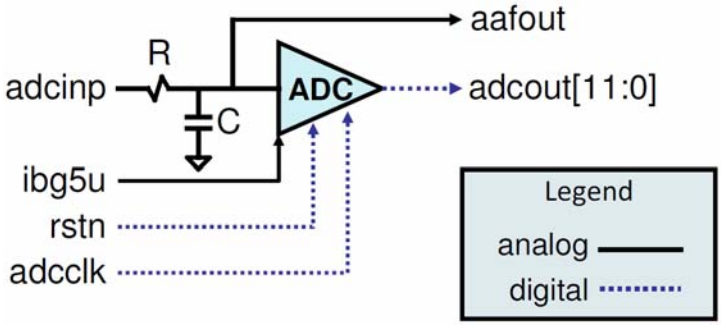
adcX (Input/Output Declarations)

Verilog

```
wire [11:0] adcout;
real aafout;
wire real ibg5u;
wire real adcinp;
wire adcclk;
wire rstn;
```

Verilog-A

```
electrical [11:0] adcout;
electrical aafout;
electrical ibg5u;
electrical adcinp;
electrical adcclk;
electrical rstn;
```



Ref: Bill Ellersick, "Real Portable Models for System/Verilog/A/AMS", SNUG 2010.

adcX (RC Filter)

Verilog

```
initial #1e-10 forever begin      // evaluate ~10 times highest freq
  ires = (adcinp - aafout_v) / Raaf;      // resistor current
              // capacitor difference equation C = V + I * deltaT / C
  aafout_v = aafout_v + ires * ($realtime-lsttim) / Caaf;
  lsttim = $realtime;                              // save time
#1e-10; end
```

Verilog-A

```
@(timer(1e-10,1e-10)) begin       // evaluate ~10 times highest freq
  ires = (V(adcinp) - aafout_v) / Raaf;      // resistor current
             // capacitor difference equation C = V + I * deltaT / C
  aafout_v = aafout_v + ires * ($realtime-lsttim) / Caaf;
  lsttim = $realtime;                              // save time
end
```


adcX (Excerpts)

Verilog

```
always @(posedge adcclk or negedge rstn) begin
  if(!(rstn)) begin
    for(i=11; i>0; i=i-1) int[i] = 0;
    $display("adcX reset at %10.4g\n",$realtime);
  end else begin
    refv0 = 0.5 * ibg5u/5e-6;
    refv = refv0;                  // start with ful
    vadc = adcinp-Voff+refv0;      // sample input, s
    for(i=11; i>0; i=i-1) begin
      if(vadc > refv) begin
        adcout_int[i] = 1;         // outp
        vadc = vadc - refv;        // and
      end else begin
        adcout_int[i] = 0;         // else
      end
      refv = refv / 2.0;           // halv
    end
  end
end
```

Verilog-A

```
@(cross(V(adcclk)-(vdd_v+vss_v)/2.0, 1)
    or cross(V(rstn)-(vdd_v+vss_v)/2.0, -1)) begin
  if(!(V(rstn)>(vdd_v+vss_v)/2.0) ? 1 : 0)) begin
    generate i(11,0) adcout_int[i] = 0;
    $display("adcX reset at %10.4g\n",$realtime);
  end else begin
    refv0 = 0.5 * I(ibg5u)/5e-6;
    refv = refv0;                  // start with full
    vadc = V(adcinp)-Voff+refv0;   // sample input, su
    generate i(11,0) begin
      if(vadc > refv) begin
        adcout_int[i] = 1;         // output logic hig
        vadc = vadc - refv;        // and subtract ref
      end else begin
        adcout_int[i] = 0;         // else output logi
      end
      refv = refv / 2.0;           // halve refv for n
    end
  end
end
```

adcX(Assignments)

Verilog

```
assign adcout[0] = adcout_int[0];
assign adcout[1] = adcout_int[1];
assign adcout[2] = adcout_int[2];
assign adcout[3] = adcout_int[3];
assign adcout[4] = adcout_int[4];
assign adcout[5] = adcout_int[5];
assign adcout[6] = adcout_int[6];
assign adcout[7] = adcout_int[7];
assign adcout[8] = adcout_int[8];
assign adcout[9] = adcout_int[9];
assign adcout[10] = adcout_int[10];
assign adcout[11] = adcout_int[11];
always @( aafout_v) aafout =  aafo
```

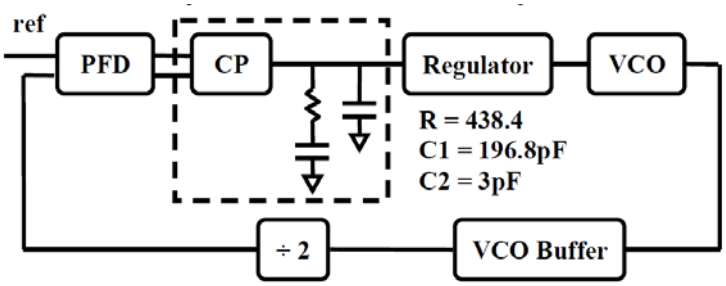
Verilog-A

```
adcout[0] <+ transition((adcout_int[0] ? vdd_v : vss_v), 100e-12, 100e-12);
adcout[1] <+ transition((adcout_int[1] ? vdd_v : vss_v), 100e-12, 100e-12);
adcout[2] <+ transition((adcout_int[2] ? vdd_v : vss_v), 100e-12, 100e-12);
adcout[3] <+ transition((adcout_int[3] ? vdd_v : vss_v), 100e-12, 100e-12);
adcout[4] <+ transition((adcout_int[4] ? vdd_v : vss_v), 100e-12, 100e-12);
adcout[5] <+ transition((adcout_int[5] ? vdd_v : vss_v), 100e-12, 100e-12);
adcout[6] <+ transition((adcout_int[6] ? vdd_v : vss_v), 100e-12, 100e-12);
adcout[7] <+ transition((adcout_int[7] ? vdd_v : vss_v), 100e-12, 100e-12);
adcout[8] <+ transition((adcout_int[8] ? vdd_v : vss_v), 100e-12, 100e-12);
adcout[9] <+ transition((adcout_int[9] ? vdd_v : vss_v), 100e-12, 100e-12);
adcout[10] <+ transition((adcout_int[10] ? vdd_v : vss_v), 100e-12, 100e-12);
adcout[11] <+ transition((adcout_int[11] ? vdd_v : vss_v), 100e-12, 100e-12);
V(aafout) <+  aafout_v;
V(ibg5u) <+ 1.1;                // voltage termination at current input
```

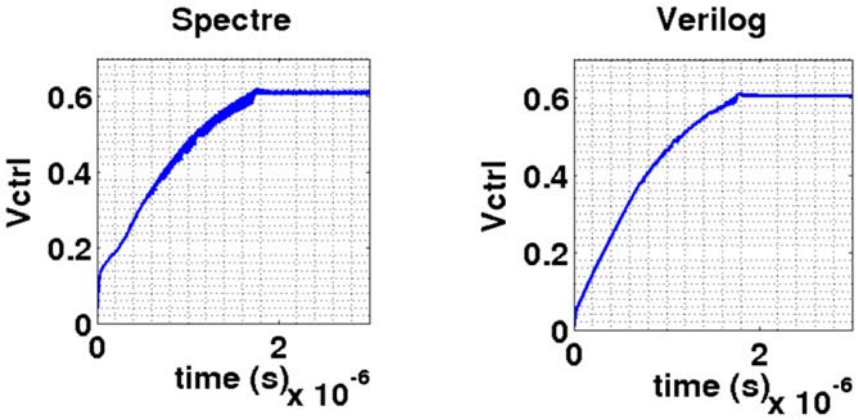

Outline

- Introduction
- Real-Valued Behavioral Models in Verilog
- Portable Models for Verilog/Verilog-A
- Test Results
- Conclusions

Phase Locked Loop (PLL) Model



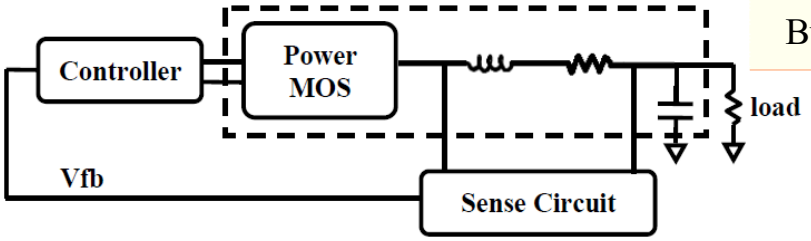
Circuit	Spectre	Verilog
PLL	33min 48s	4.4s



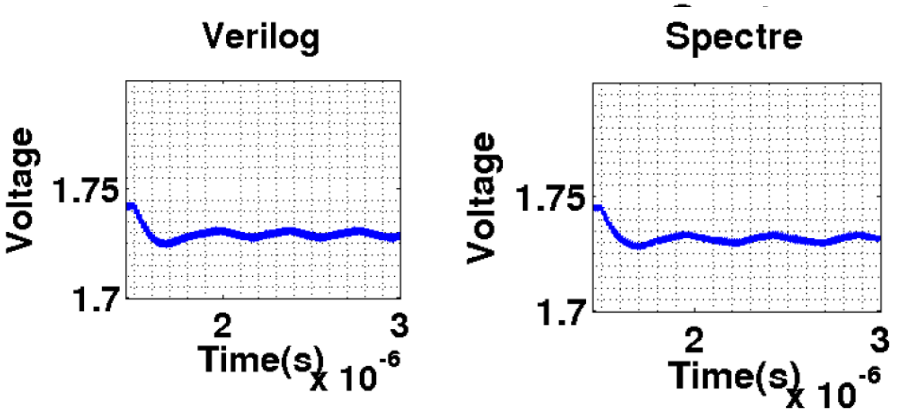
Ref: S. Liao and M. Horowitz, "A Verilog piecewise-linear analog behavior model for mixed-signal validation," IEEE TCAS-I, vol. 61, no. 8, pp. 2229–2235, Aug. 2014.

Buck Converter Model

Circuit	Spectre	Verilog
Buck	25h 15min	0.46s

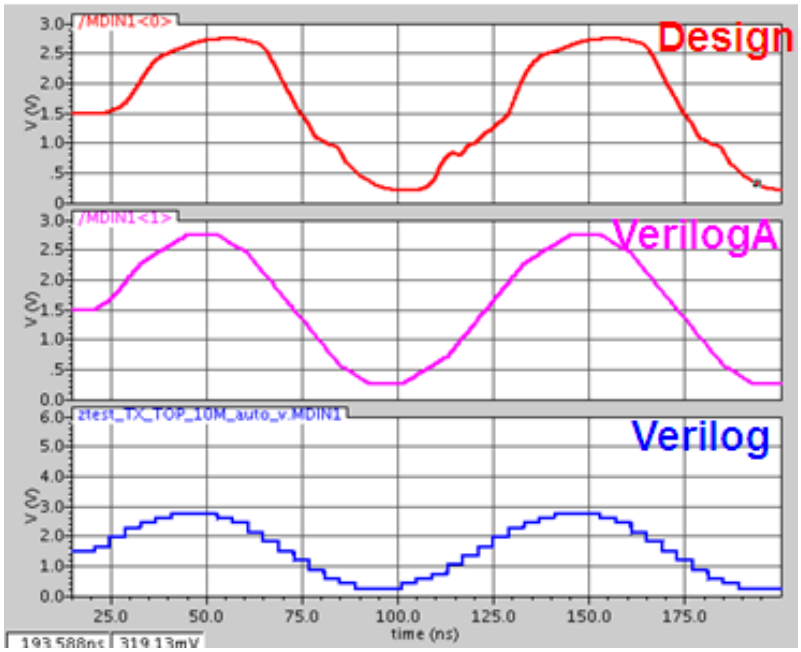


Rload change at 1.5us from 20 Ω to 10 Ω



Ref: S. Liao and M. Horowitz, "A Verilog piecewise-linear analog behavior model for mixed-signal validation," IEEE TCAS-I, vol. 61, no. 8, pp. 2229–2235, Aug. 2014.

D/A Converter Model



Spectre	7m 23.5s
Verilog-A	17.0s
Verilog	16.4s

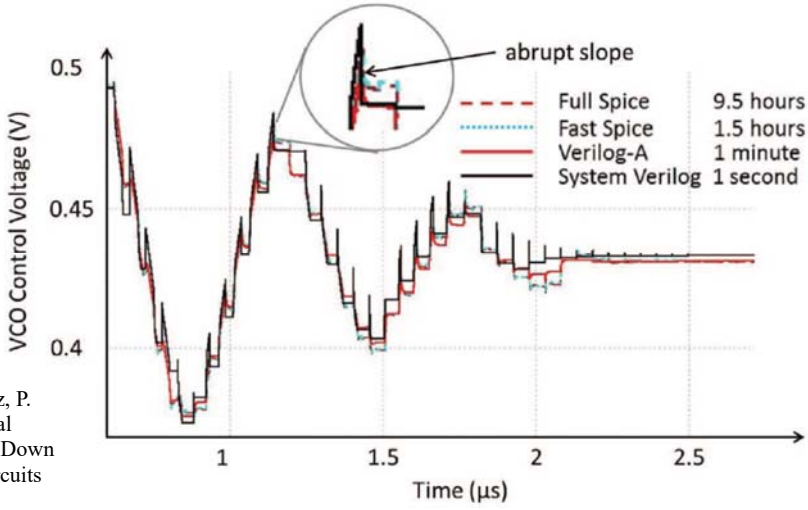
- Verilog-A waveform is more smooth
  → high accuracy
- Beh. models do provide great speedup

Ref: Y.-J. Lin, M.-J. Lee, Y.-L. Lo, S.-Y. Kao, "Automatic Mixed-Signal Behavioral Model Generation Environment", IEEE Int'l Symp. on VLSI Design, Automation, and Test, Apr. 2016. (**Invited Paper**)

PLL Model in Different Language

	Run Time (5us)	Correlation with Hspice
Full Spice	9.5 hours	1
Fast Spice	1.5 hours	99.9%
Verilog-A	1 minute	98.8%
System-Verilog	1 second	97.6%

- Model PLL circuits based on PWC (piece-wise constant) real numbers and lookup tables
- Super fast with good accuracy !!



Ref: A. Lotfy, S. F. S. Farooq, Q. S. Wang, A. Yaldiz, P. Mosalikanti, N. Kurd, "A System-Verilog Behavioral Model for PLLs for Pre-Silicon Validation and Top-Down Design Methodology," IEEE Custom Integrated Circuits Conf. (CICC), Sep. 2015.

Conclusions

- A good behavioral model considers both accuracy and simulation speed
- Verilog-A models are good to provide high accuracy with various operators
  - Speedup is limited due to extra communication between analog/digital simulators
- Verilog models provide another choice to further improve simulation speedup
  - Accuracy may be sacrificed due to language limitation
- Choose different models at different applications
  - Portable behavioral models are convenient to be switched