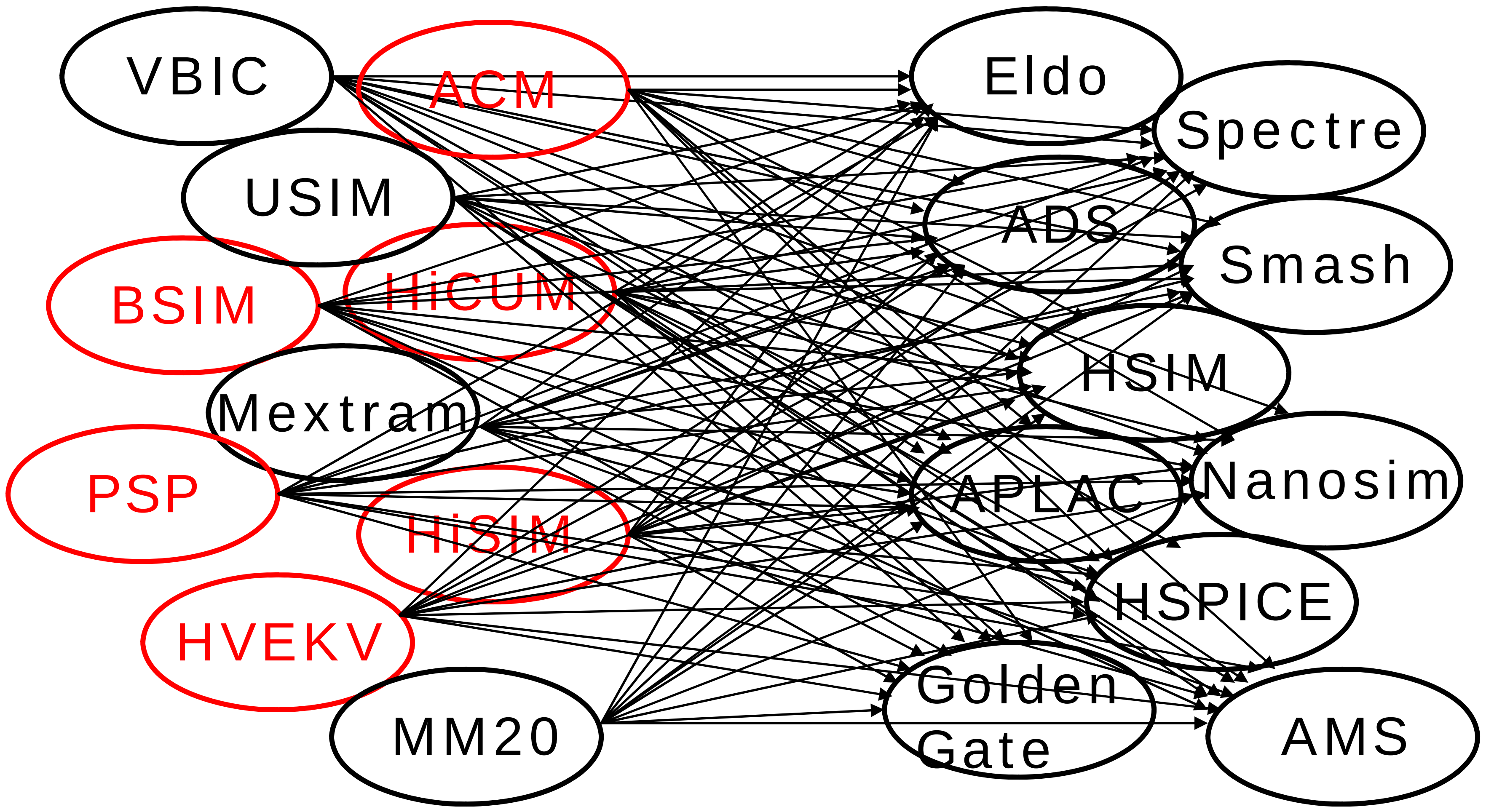
Lecture 9. Introduction to Analog Behavioral Description Language

Jaeha Kim
Mixed-Signal IC and System Group (MICS)
Seoul National University
jaeha@ieee.org

Overview

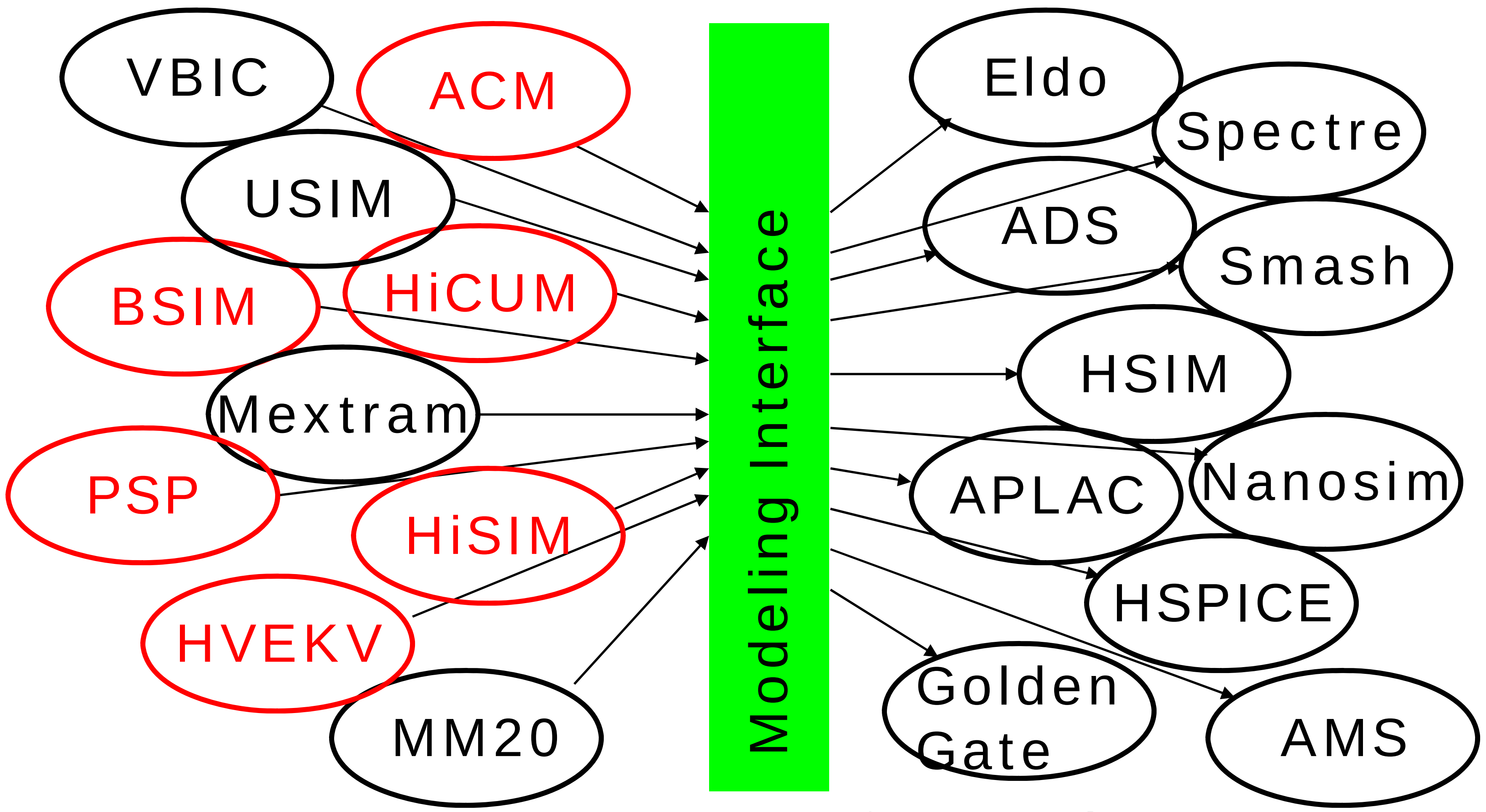
- Readings
  - Verilog-A Langauge Reference Manual: http://www.vhdl.org/verilog-ams/htmlpages/public-docs/lrm/VerilogA/verilog-a-lrm-1-0.pdf
  - Designer's guide website (http://www.designers-guide.org)
- Background
  - To describe ideal behaviors of analog circuits, we used to build artificial circuits with ideal sources/RLC, switches, and dependent elements. Analog behavioral description language lets you do this procedurally. While the original intent of Verilog-A is to ease writing compact models, I find it most useful while writing testbenches.

Device Model Writer's Nightmare



from McAndrew, BMAS 2003

The Solution



from McAndrew, BMAS 2003

Verilog-A: Analog Behavioral Modeling

- The original intent: replace C compact device models
- Versatile and simulator-independent
- No need to explicitly compute the derivatives

```
module mos(d,g,s,b);
inout d,g,s,b;
analog
  I(d,s) <+
      k*(V(g,s)-Vth)^alpha;
endmodule
```

```
module mos(d,g,s,b);
inout d,g,s,b;
analog
  I(d,s) <+
      $table_model(
      “mos_IV.dat”, V(g,s));
endmodule
```

Verilog-A: Jump Start

- Verilog-A borrows the syntax from Verilog
  - Intuitive and easy to read and learn
- Based on through and across variables
  - Set up is for KCL and KVL
  - Understand the "contribution" operator

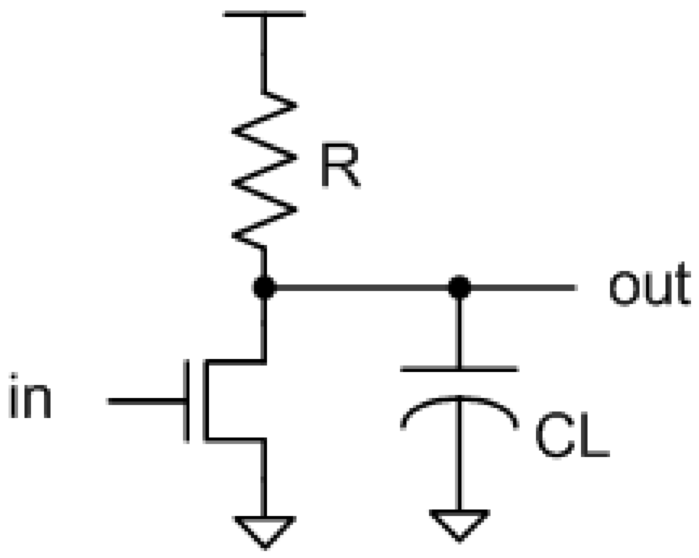
```
I(di,si) <+ Ids;
V(d ,di) <+ I(b_rd)*rd;
```

- Dynamic flows are done via `ddt()`

```
I(t,b) <+ ddt(C * V(t,b));
```

Verilog-A: First Example

```
module cg_amp(in,out,vdd);
inout in, out, vdd;
electrical in, out, vdd;
parameter real Cg = 100f;
parameter real CL = 1p;
parameter real k = 100u;
Parameter real alpha = 1.25;
analog begin
  I(in) <+ Cg*ddt(V(in));
  I(out) <+ k*(V(in)-Vth)^alpha;
  I(out) <+ CL*ddt(V(out));
  I(vdd,out) <+ V(vdd,out)/R;
end
endmodule
```



Input/Output Ports

- Conservative systems (**`electrical`**)
  - Two values are associated with every node: potential (voltage) and flow (current)
  - The system obeys the KVL and KCL: their values need to be solved by an ODE
  - In general, signals flow bidirectional

- Signal-flow systems (**`voltage, current`**)
  - Only voltage or current is associated with the node
  - Unidirectional signal flow: their values are determined by simple assignments

Contribution Operator “<+”

- Only valid within the **`analog`** block
  - You cannot assign both flow and potential to the same branch

```
analog begin
  I(n1,n2) <+ expression;
  V(n3,n4) <+ expression;
end
```

- “<+” accumulates the contributions on the branch within the simulation cycle
  - Differers from ‘=‘ in that the contributions are accumulated
  - Differs from ‘+=‘ in that the value is reset every time step

But Verilog-A Can Also Do:

- Run-time measurement
- Assertion checks
- Stimuli generation
- Testbenches
- Solve and optimization

Verilog-A: Run-time Measurement

- Recording the transition times of a clock
  - No need to store .tr0; can auto-start/stop

```
module snork (clk);
input clk;
parameter real threshold = Vdd/2;
integer file;

analog begin
   @(initial_step) file = $fopen(“clk_transition”);
   @(cross(V(clk)-threshold, +1, 1e-12))
       $fdisplay(file, “%1.4g”, $abstime);
   @(final_step) $fclose(file);
end

endmodule
```

C S

Verilog-A: Assertion Checks

- Check circuit behavior in run-time
  - Report when violation occurs
  - e.g., gate oxide breakdown checks, PN forward-bias checks, saturation checks, …
- Checker can be instantiated as subckt
  - Embed checkers in your circuit
  - Then each of its multiple instances will have its own checkers automatically

Example: Oxide Breakdown Check

```
.SUBCKT nmos_ox d g s b W=W1 L=L1
M1 d g s b nmos W=W1 L=L1
X1 d g s b ox_checker Vmax=2V
.ENDS
```

```
module ox_checker(d,g,s,b);
inout d,g,s,b;
parameter real Vmax=inf;
analog
  if (V(g,s) > Vmax or V(g,d) > Vmax)
      $display("transistor is toast at %1.4g sec",
            $abstime);
endmodule
```

Verilog-A: Stimuli Generation

- Create stimuli not readily available
- e.g., clock waveform with AC jitter

```
module clock_w_jitter (clk, jitter);
input jitter;
output clk;
parameter real vos, vsw, tcyc, tedge;
analog begin
  V(sine) <+ cos(2*`M_PI*$abstime/tcyc + V(jitter));
  V(clk) <+ vos + vsw*tanh(tcyc/tedge/`M*PI*V(sine));
end
endmodule
```

Jitter Source (Spectre Format)

- AC jitter source:

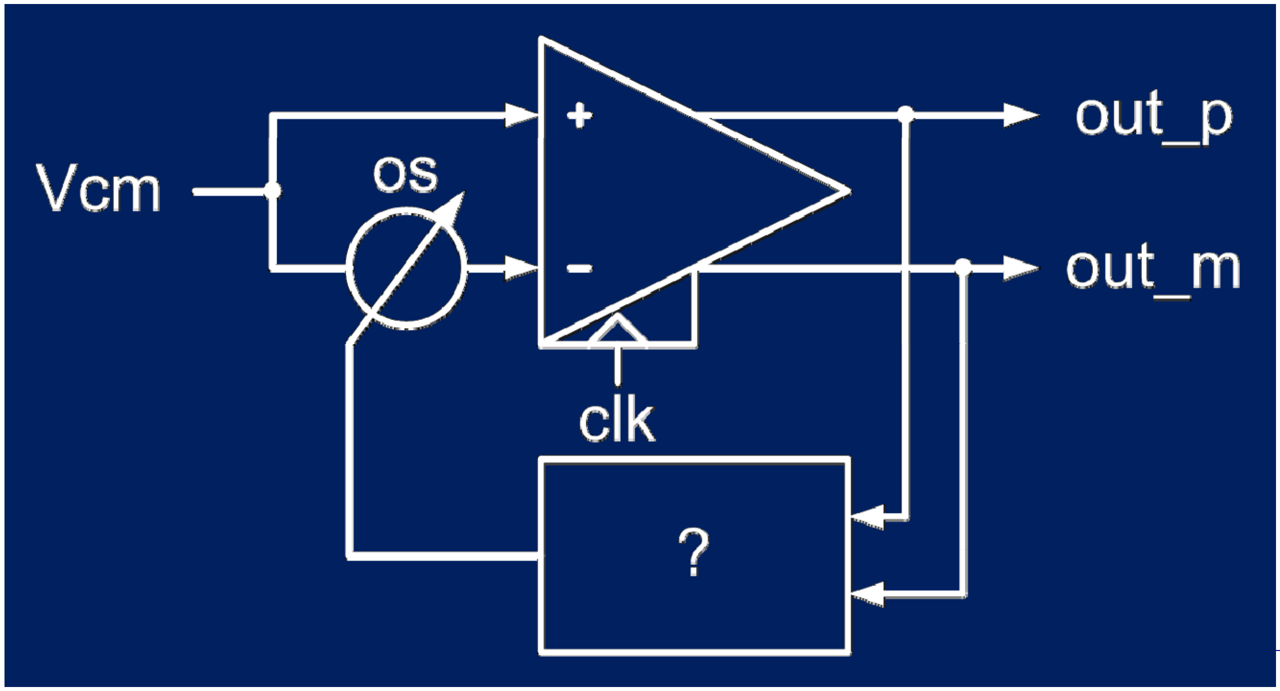
```
Vjtr   (jitter gnd) vsource pacmag=1
Clksrc (clk jitter) clk_w_jitter
+ vos=vdd/2 vsw=vdd/2 tedge=100p tcyc=1n
```

- White jitter source:

```
Vjtr   (jitter gnd) bsource v=0
+ isnoisy=yes white_noise(var_j)
Clksrc (clk jitter) clk_w_jitter
+ vos=vdd/2 vsw=vdd/2 tedge=100p tcyc=1n
```

Verilog-A: Testbenches

- Measure input offset of a comparator
  - Takes input voltage sweep with transient sim
- Imagine a module that finds the input offset through feedback



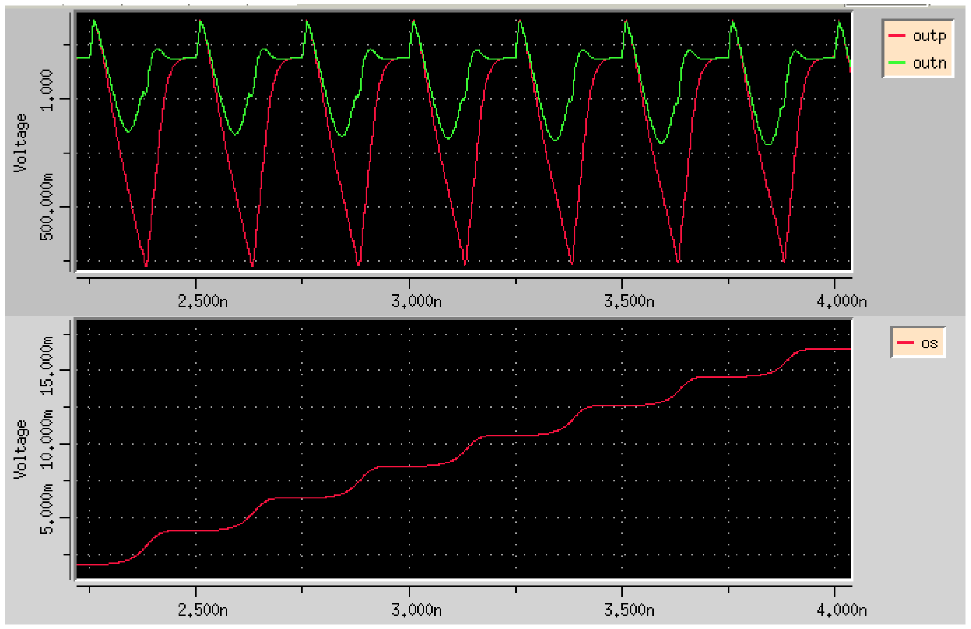
The Imaginary Module

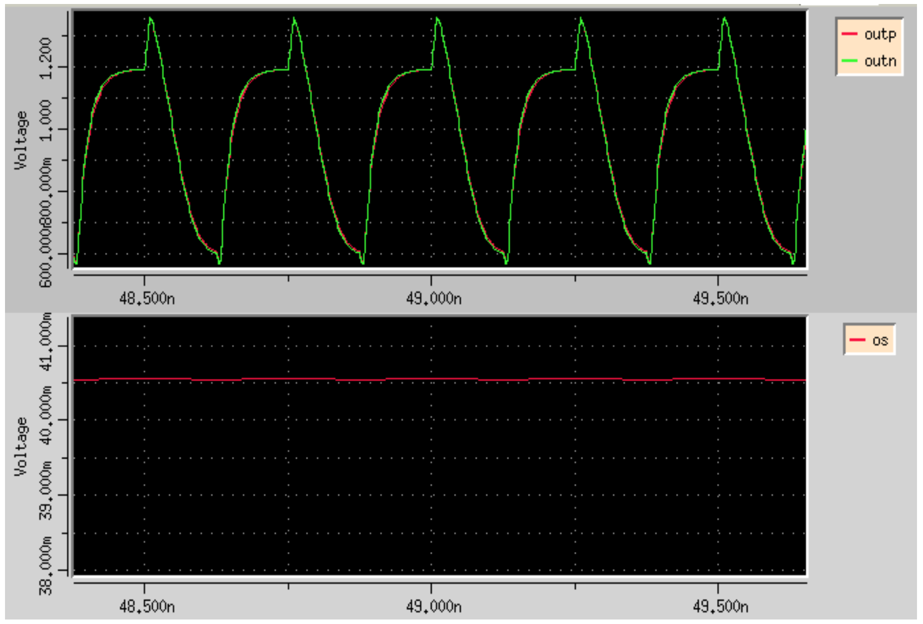
- Metastable: outp == outn at all times
- Adjust offset based on outp-outn

```
module meas_comp_offset(osp, osn, outp, outn);
input outp, outn;
output osp, osn;

analog begin
  V(osp,osn) <+ -0.1 * idt(V(outp,outn), 0.0);
end

endmodule
```

Simulation Response

- Single transient sim finds the offset





Before Settling **After Settling**

Verilog-A: Indirect Assignment

- **`V(x)`**: <condition>;

  Find V(x) that satisfies the <condition>

- Can describe differential equations

```
V(x): ddt(V(x)) = A*V(x) + b;
```

Verilog-A: Solver/Optimizer

- Solve f(x) = 0:

```
V(f) <+ V(x)^2 - 2*V(x) – 3;
V(x): V(f) == 0;
```

- Minimize f(x):

```
V(f) <+ V(x)^2 - 2*V(x) – 3;
V(x): ddx(V(f)) == 0;
```

Verilog-A: Caveats

- Resurgence of almost forgotten errors:

  "Time step too small"
  "DC convergence failure"

- Easy to write models that are numerically ill-conditioned
  - Bad news: simulator dependent

- Some rules:
  - Avoid discontinuities in values and derivatives
  - Avoid sharp transitions (or zero capacitance nodes)
    - Ideal system for SPICE is a low-order linear system

Using Verilog-A in SpectreRF

- Verilog-A modules must not have “hidden states”
  - Shooting method needs access to all the states in the circuit
  - States (memories) that affect the behavior, but not visible to SPICE engine as V or I
- For example, you can’t use “absdelay”
  - exp(-Td*s) has infinite number of states
  - Internal states are not accessible by SPICE
  - Instead, approximate it as a finite-bandwidth system