Behavioral Modeling using Verilog-A

Dr. Vishal Saxena

Electrical and Computer Engineering Department
Boise State University, Boise, ID
Verilog-A

- VerilogA is the standard behavioral modeling language in Cadence Spectre environment
- Allows to simulate complex systems without transistor-level implementation
- Some of the functionality is similar to Matlab Simulink but more circuit oriented
- Can interchange VerilogA, Transistor-level and parasitic extracted circuit views for system-level simulation using the Hierarchy editor
  - Powerful method for complex design verification
- Language construct is similar to digital Verilog RTL, but not quite the same
  - Easy to pick up, but mastery comes with experience
  - Can be used to model novel devices not covered by bsim
Verilog-AMS

- Verilog-AMS is an extension of Verilog-A to include digital Verilog co-simulation functionality
- Works with the **ams** simulator instead of spectre
- Need to clearly define interfaces between analog and digital circuits
- bmslib and ahdLib libs have *verilogams* views along with *veriloga*
- Don’t worry about it for now….
Using Behavioral Cells

- `bmslib → dff_sr cell` for a DFF with reset
Select: **CDF parameter of View → veriloga**

Connect the **Set** pin to GND to disable it, **Reset** is asserted when high.
Setting cell parameters

- Set desired model parameters such as voltages and delays
- Preferably use variables controlled from the ADE-L window (e.g. $t_{pq}$ here)
- Make a local copy of the `bmslib → and2 cell`
- Delete the `cmos_sch` view as it interferes with the simulation
- Could also use cells from the `ahdlLib` library
Convergence Hints

- Since verilog-A models are idealized models they can cause convergence problems.
- In a transient sim use the `skipdc` option if DC operating point convergence is not achieved by the simulator.
Convergence Hints contd.

- Use initial conditions to help with convergence
  - ADE L → Simulation → Convergence Aids → Initial Condition

- Can relax tolerances in the simulator options
  - ADE L → Simulation → Options → Analog

- Use common-sense when using idealized elements and models…
  - Turn on Spectre debug mode to help fix the problem
  - Look into the convergence related help in the Spectre references (listed later)
Dff Code Synopsis

module dff (Co, Q, CLK, D);
output Qo;
electrical Qe;
output Qe;
electrical Qe;
input CLK;
electrical CLK;
input D;
electrical D;

// INSTANCE PARAMETERS:
parameter real vhi = 3; // voltage [v] for logic high
parameter real vlo = 0 from (-inf:vhi); // voltage [v] for logic low
parameter real vthresh = 0.5*(vhi+vlo); // switch voltage [v]
parameter real tophq = 5n from (0:1n);
// prop delay from clock to 0, low to high
parameter real tophl = 5n from (0:1n);
// prop delay from clock to 0, high to low
parameter real tophlq = 5n from (0:1n);
// prop delay from clock to 0, low to high
parameter real tophlq = 5n from (0:1n);
// prop delay from clock to 0, high to low
parameter real tr = tophq from (0:2*tophq); // rise time [s] for gate output
parameter real tf = tophlq from (0:2*tophlq); // fall time [s] for gate output
parameter integer initval = -1 from [-1:1]; // initial value for 0/(V/2)

// LOCAL VARIABLES
real tdelay;
real tdelay;
integer q;
integer qnew;
integer qnew;
integer q;
integer q;
integer q;
integer q;

analog function real tdelay;
inout outn, out, toph, tph, trise, tfall;
real toph, tph, trise, tfall;
inout outn, out;
case (1)
(outn > out) : begin // high to low on 0
Dff Code Synopsis contd.

```verilog
default function
endfunction

analog begin
  (initial_step) begin
    if (vthrese > vhi || vthrese < vlo) begin
      display
        ("$M: Inconsistent input threshold specification w/logic family \n");
    end
    case (initial)
      1: begin q = 1; _q = 0; end
      0: begin q = 0; _q = 1; end
      -1: begin q = (V(D) > vthrese); _q = (V(D) <= vthrese); end
    default begin q = (V(D) > vthrese); _q = (V(D) <= vthrese); end
    endcase
    xflag = 1; // initial time, input data is allowed
  end

  (cross($'(CLK) - vthrese, +1)) xflag = 1;
  if ($'clock) begin
    xflag = 0;
    d = V(D) > vthrese;
    qnow = q;
    q_now = _q;
  end
  if (d) begin
    q = 1;
    _q = 0;
    tdelq = tdelay(qnow, q, tcplhq, tcplhq, tr, tf);
    tdelq_ = tdelay(q_now, _q, tcphtq, tcphtq, tr, tf);
  end
  if (!d) begin
    q = 0;
    _q = 1;
    tdelq = tdelay(qnow, q, tcplhq, tcphtq, tr, tf);
    tdelq_ = tdelay(q_now, _q, tcphtq, tcphtq, tr, tf);
  end
  V(Co) <+ transition( q ? vhi : vlo, tdelq, tr, tf);
  V(Q_) <+ transition( _q? vhi : vlo, tdelq_, tr, tf);
endmodule
```

---

**Main analog block defining the model functionality**

**Initial step**

**Behavior definitions**

**Output transitions**
How to get started using Verilog-A modeling

- Start with the available behavioral blocks with Spectre
- Don’t create a fresh model from scratch unless you really need it
  - Modify the existing ones
- Don’t get bogged down with the code complexity of these professionally coded models
  - Your custom behavioral codes can be really simple
  - Once you start using verilogA, it will get easier…..
- Great skill to have for an analog designer!
  - All circuit design these days is at system level
References and Online Resources

- Spectre reference libraries with behavioral cells
  - bmslib and ahdllib

- Must read: Cadence Whitepaper, "Creating Analog Behavioral Models"

- Designers Guide Community Site
  - http://www.designers-guide.org/

- Books

- AMS CAD Wiki
Happy Circuit Modeling with VerilogA!
3. Cadence Whitepaper: Creating Analog Behavioral Models
5. Virtuoso Spectre Designer Reference [Online]
7. Information on linking Matlab and Spectre in Linux environment. [Online]