Behavioral Modeling using Verilog-A

Dr. Vishal Saxena

Electrical and Computer Engineering Department
Boise State University, Boise, ID

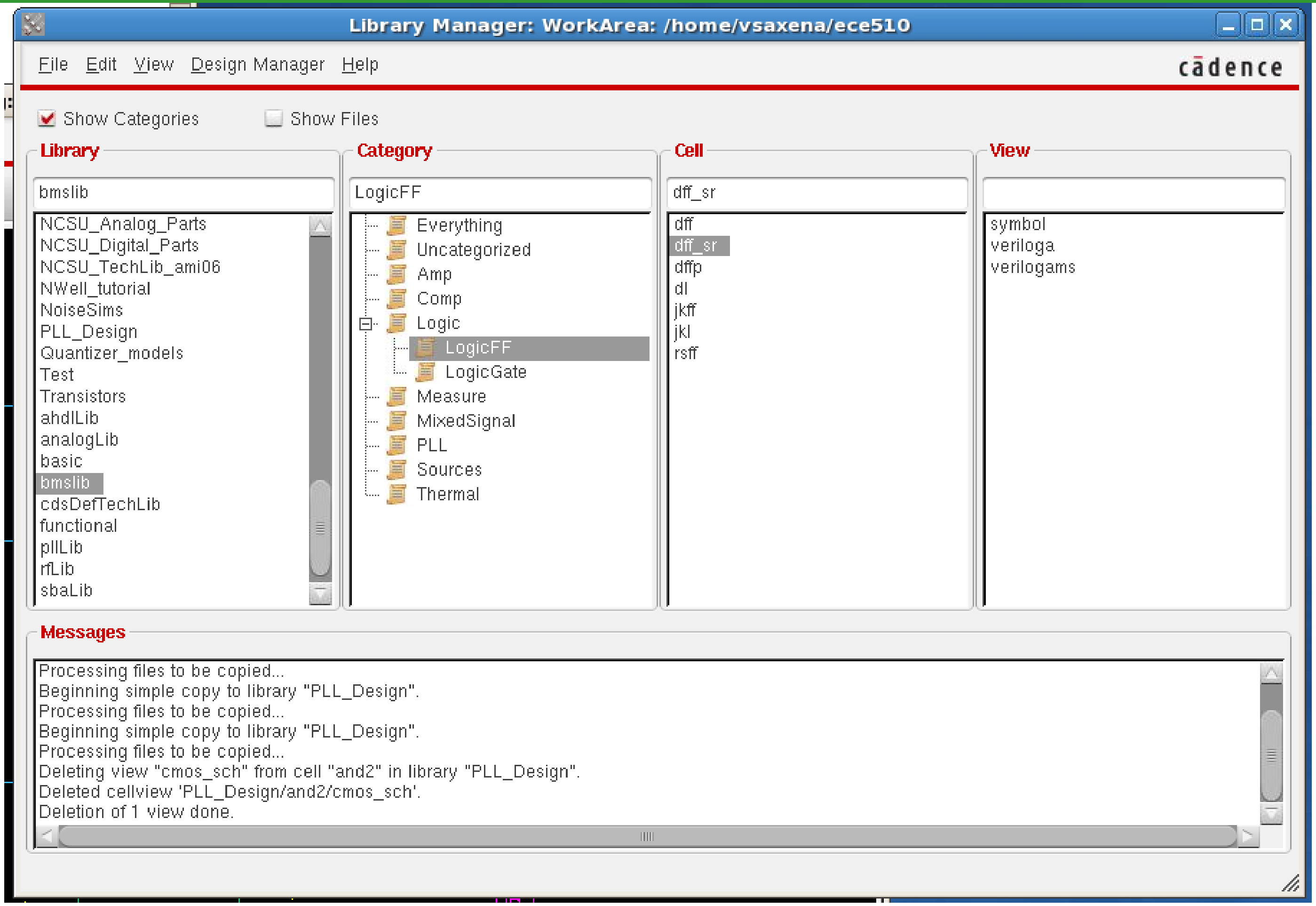
Verilog-A

- 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 ahdlLib libs have *verilogams* views along with *veriloga*
- Don't worry about it for now….

Using Behavioral Cells

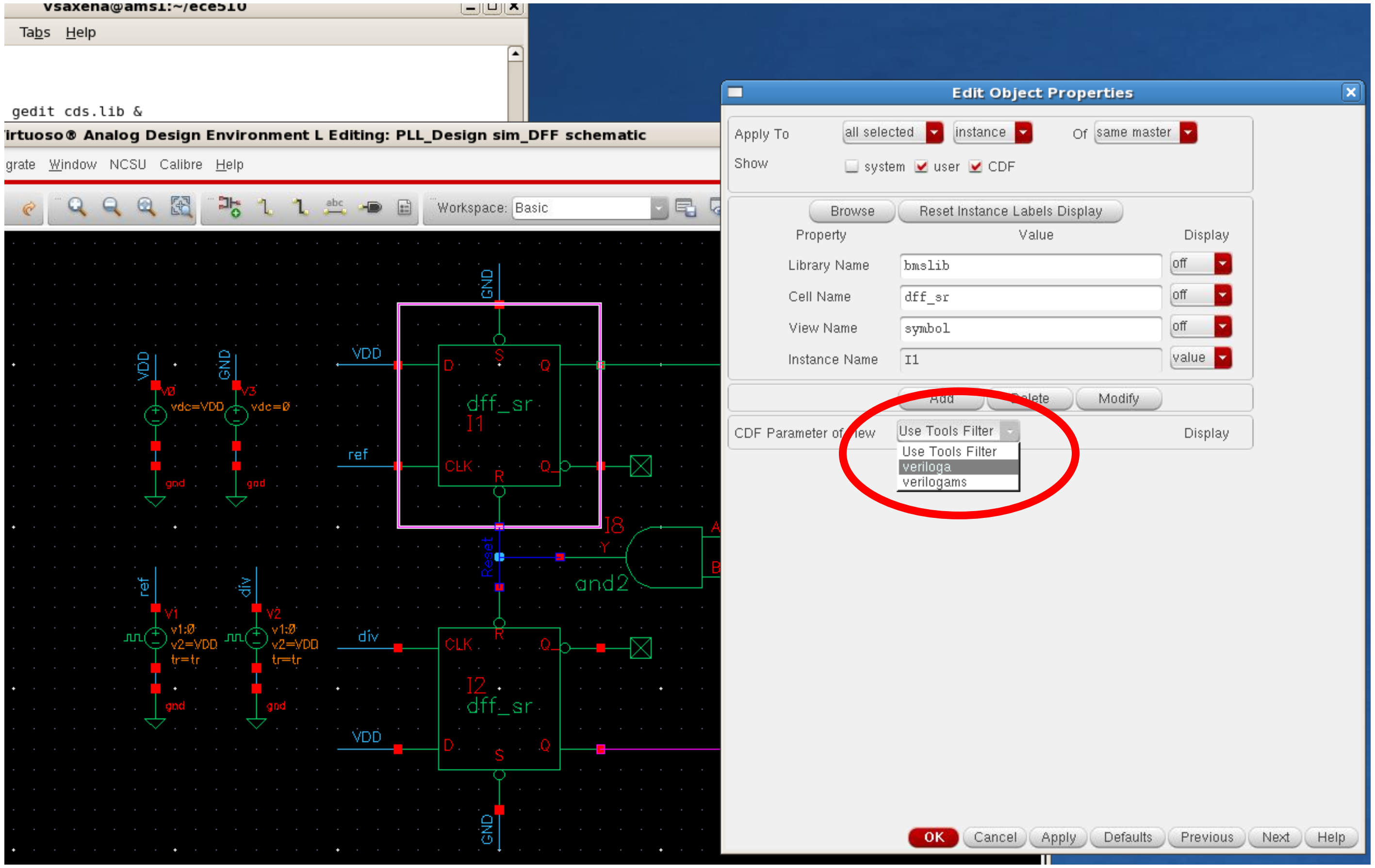

Library Manager: WorkArea: /home/vsaxena/ece510
File Edit View Design Manager Help
cadence
Show Categories
Show Files
Library
Category
Cell
View
Messages

Library	Category	Cell	View
bmslib	LogicFF	dff_sr	
NCSU_Analog_Parts	Everything	dff	symbol
NCSU_Digital_Parts	Uncategorized	dff_sr	veriloga
NCSU_TechLib_ami06	Amp	dffp	verilogams
NWell_tutorial	Comp	dl	
NoiseSims	Logic	jkff	
PLL_Design	LogicFF	jkl	
Quantizer_models	LogicGate	rsff	
Test	Measure		
Transistors	MixedSignal		
ahdlLib	PLL		
analogLib	Sources		
basic	Thermal		
bmslib			
cdsDefTechLib			
functional			
pllLib			
rfLib			
sbaLib			

```
Processing files to be copied...
Beginning simple copy to library "PLL_Design".
Processing files to be copied...
Beginning simple copy to library "PLL_Design".
Processing files to be copied...
Deleting view "cmos_sch" from cell "and2" in library "PLL_Design".
Deleted cellview 'PLL_Design/and2/cmos_sch'.
Deletion of 1 view done.
```

- *bmslib* → *dff_sr cell* for a DFF with reset

Setting cell parameters



```
gedit cds.lib &
```

Apply To: all selected, instance; Of: same master

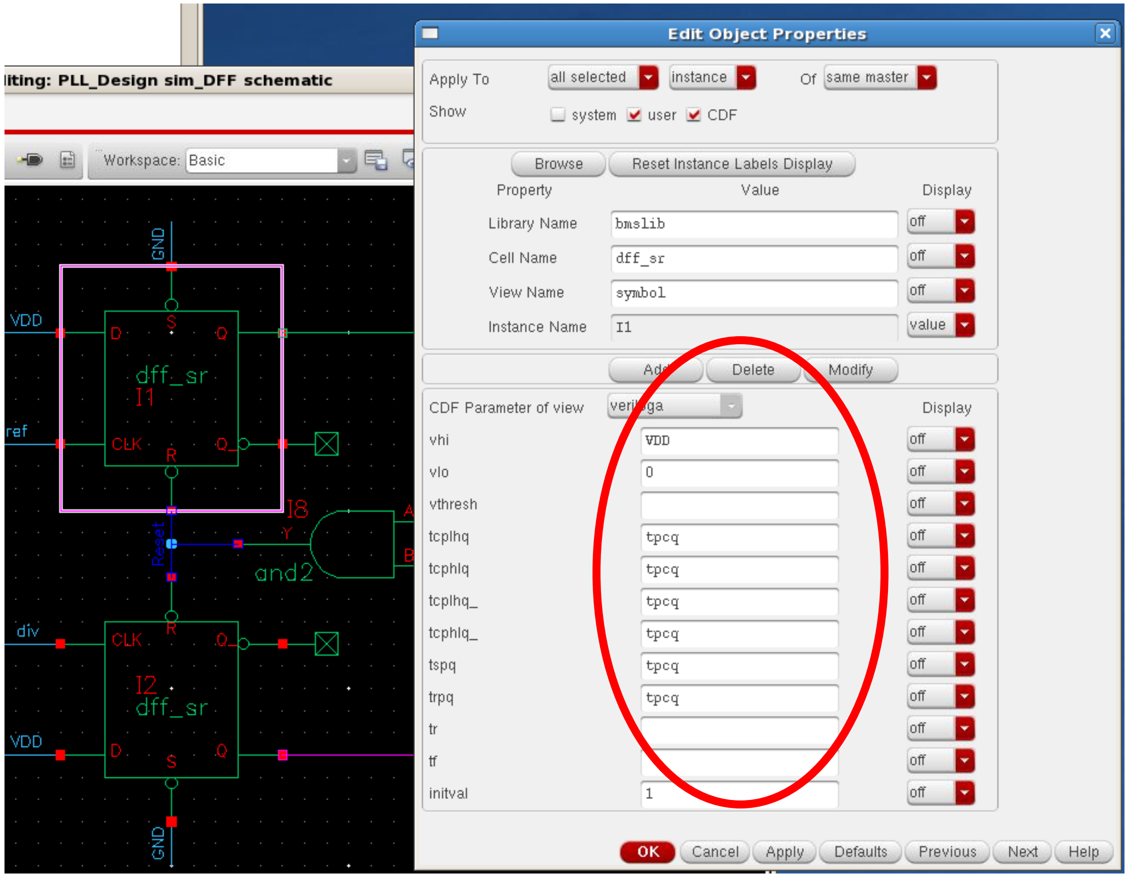
Show: ☐ system ☑ user ☑ CDF

Property	Value	Display
Library Name	bmslib	off
Cell Name	dff_sr	off
View Name	symbol	off
Instance Name	I1	value

CDF Parameter of View: Use Tools Filter (options: Use Tools Filter, [✓ veriloga], verilogams) — Display

- Select: *CDF parameter of View → veriloga*
- Connect the **S**et pin to GND to disable it, **R**eset is asserted when high.

Setting cell parameters



Edit Object Properties

Apply To: all selected, instance, Of: same master

Show: ☐ system ✓ user ✓ CDF

Browse | Reset Instance Labels Display

Property	Value	Display
Library Name	bmslib	off
Cell Name	dff_sr	off
View Name	symbol	off
Instance Name	I1	value

Add | Delete | Modify

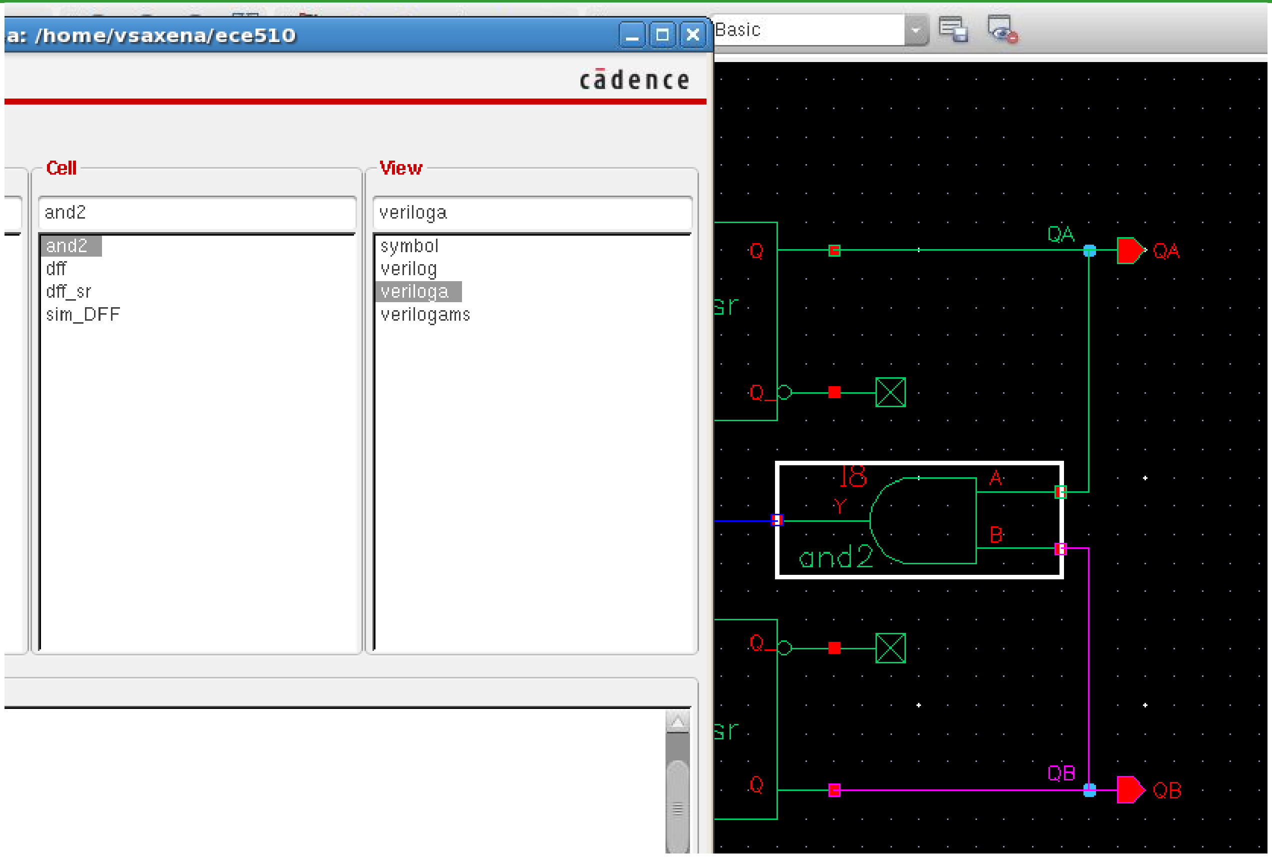
CDF Parameter of view: veriloga

		Display
vhi	VDD	off
vlo	0	off
vthresh		off
tcplhq	tpcq	off
tcphlq	tpcq	off
tcplhq_	tpcq	off
tcphlq_	tpcq	off
tspq	tpcq	off
trpq	tpcq	off
tr		off
tf		off
initval	1	off

OK | Cancel | Apply | Defaults | Previous | Next | Help

- Set desired model parameters such as voltages and delays
- Preferably use variables controlled from the ADE-L window (e.g. t_{pcq} here)

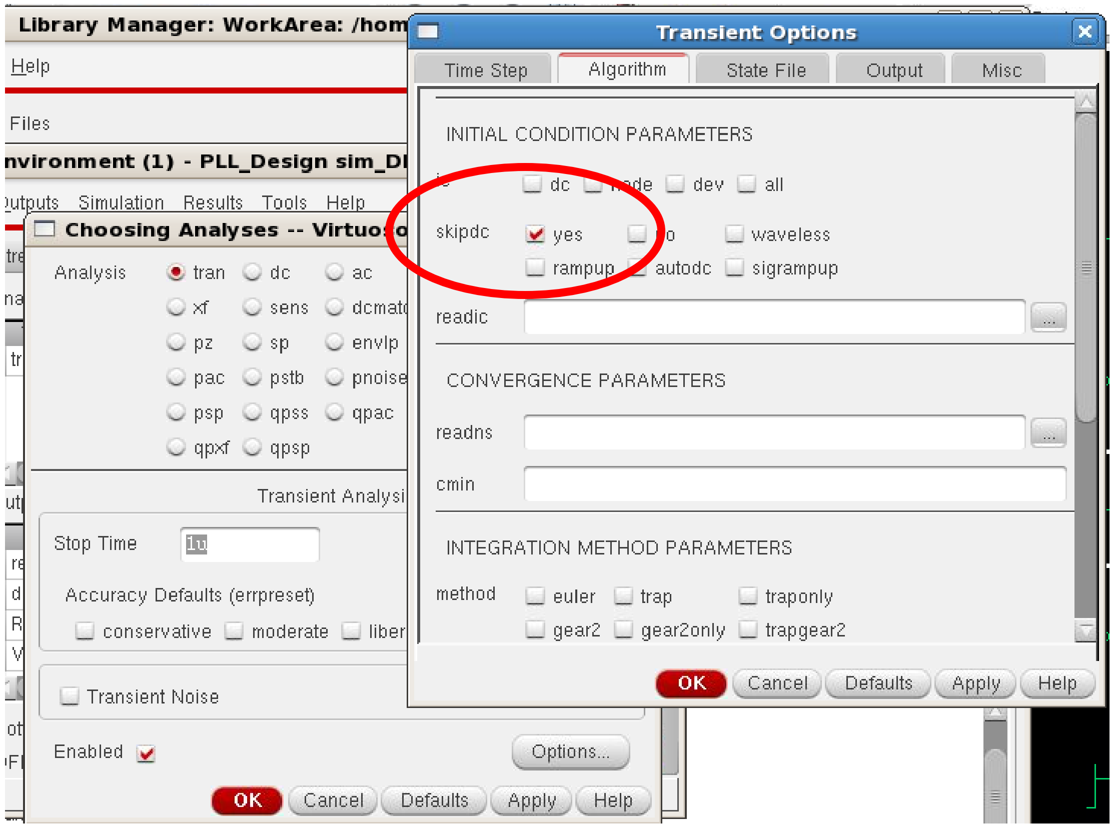
Logic Cells



- 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


Transient Options

Time Step | Algorithm | State File | Output | Misc

INITIAL CONDITION PARAMETERS

skipdc [✓ yes] no waveless rampup autodc sigrampup

readic

CONVERGENCE PARAMETERS

readns

cmin

INTEGRATION METHOD PARAMETERS

method euler trap traponly gear2 gear2only trapgear2

Choosing Analyses -- Virtuoso

Analysis [✓ tran] dc ac xf sens dcmatch pz sp envlp pac pstb pnoise psp qpss qpac qpxf qpsp

Transient Analysis

Stop Time 1u

Accuracy Defaults (errpreset) conservative moderate liberal

Transient Noise

Enabled [✓]

Library Manager: WorkArea: /hom
Help
Files
nvironment (1) - PLL_Design sim_DI
Outputs Simulation Results Tools Help
Options...
OK Cancel Defaults Apply Help

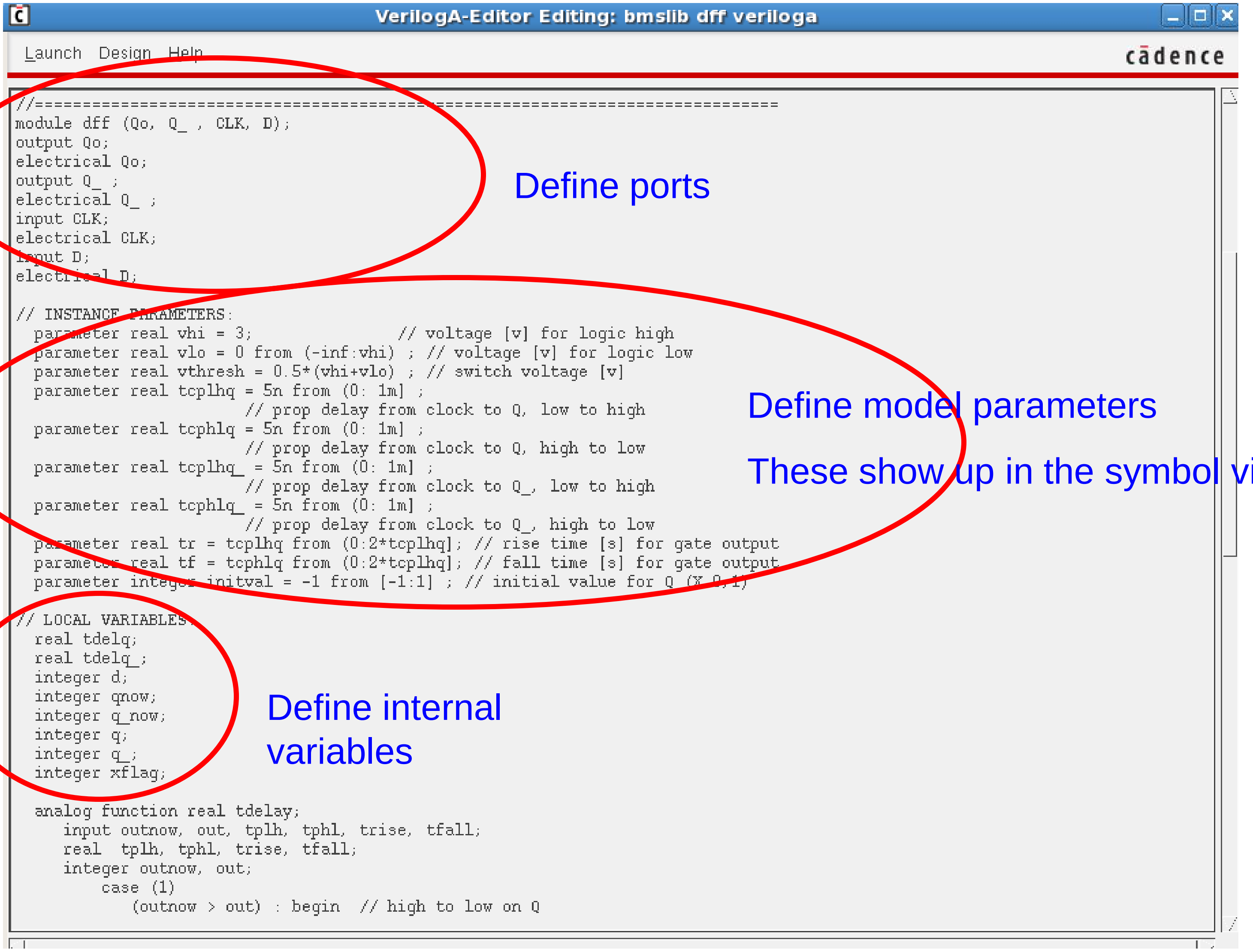
Convergence Hints contd.

- 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 (Qo, Q_ , CLK, D);
output Qo;
electrical Qo;
output Q_ ;
electrical Q_ ;
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 tcplhq = 5n from (0: 1m] ;
                       // prop delay from clock to Q, low to high
  parameter real tcphlq = 5n from (0: 1m] ;
                       // prop delay from clock to Q, high to low
  parameter real tcplhq_ = 5n from (0: 1m] ;
                       // prop delay from clock to Q_, low to high
  parameter real tcphlq_ = 5n from (0: 1m] ;
                       // prop delay from clock to Q_, high to low
  parameter real tr = tcplhq from (0:2*tcplhq]; // rise time [s] for gate output
  parameter real tf = tcphlq from (0:2*tcphlq]; // fall time [s] for gate output
  parameter integer initval = -1 from [-1:1] ; // initial value for Q (X,0,1)

// LOCAL VARIABLES:
  real tdelq;
  real tdelq_;
  integer d;
  integer qnow;
  integer q_now;
  integer q;
  integer q_;
  integer xflag;

  analog function real tdelay;
     input outnow, out, tplh, tphl, trise, tfall;
     real  tplh, tphl, trise, tfall;
     integer outnow, out;
         case (1)
            (outnow > out) : begin  // high to low on Q
```

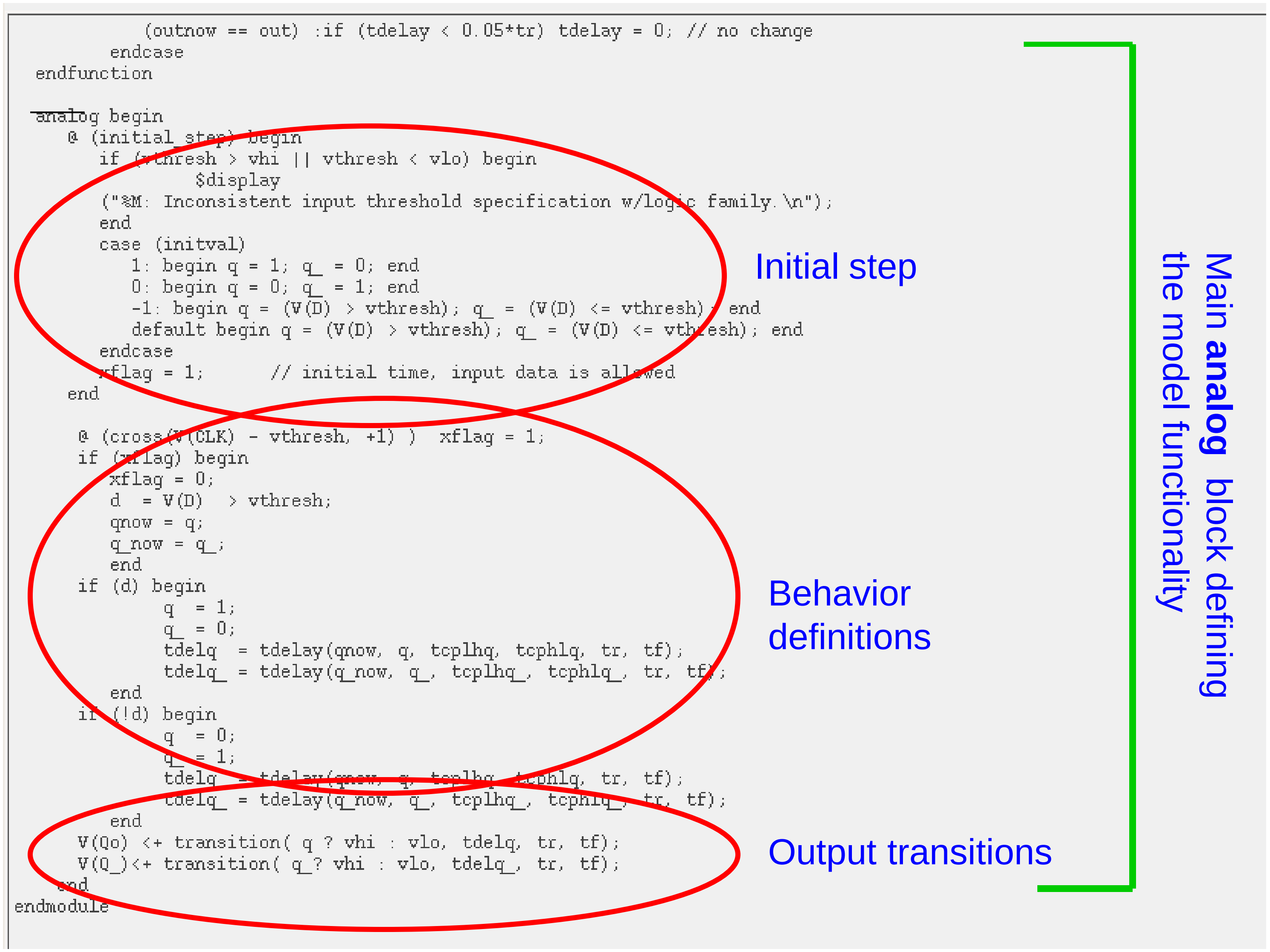
Define ports

Define model parameters

These show up in the symbol view

Define internal variables

Dff Code Synopsis contd.



```
            (outnow == out) :if (tdelay < 0.05*tr) tdelay = 0; // no change
         endcase
  endfunction

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

     @ (cross(V(CLK) - vthresh, +1) )  xflag = 1;
     if (xflag) begin
        xflag = 0;
        d  = V(D)  > vthresh;
        qnow = q;
        q_now = q_;
        end
     if (d) begin
             q  = 1;
             q_ = 0;
             tdelq  = tdelay(qnow, q, tcplhq, tcphlq, tr, tf);
             tdelq_ = tdelay(q_now, q_, tcplhq_, tcphlq_, tr, tf);
        end
     if (!d) begin
             q  = 0;
             q_ = 1;
             tdelq  = tdelay(qnow, q, tcplhq, tcphlq, tr, tf);
             tdelq_ = tdelay(q_now, q_, tcplhq_, tcphlq_, tr, tf);
        end
     V(Qo) <+ transition( q ? vhi : vlo, tdelq, tr, tf);
     V(Q_)<+ transition( q_? vhi : vlo, tdelq_, tr, tf);
    end
endmodule
```

Initial step

Behavior definitions

Output transitions

Main **analog** block defining the model functionality

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”
  - http://lumerink.com/courses/ECE614/Handouts/CDN_Creating_Analog_Behavioral_Models.pdf

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

- **Books**
  - *The Designer's Guide to Verilog-AMS* by Kenneth S. Kundert & Olaf Zinke, 2004.
  - *The Designer's Guide to SPICE and Spectre* by Kenneth S. Kundert, 1995.

- AMS CAD Wiki
  - http://lumerink.com/cadwiki/doku.php

Happy Circuit Modeling with VerilogA!

References

1. Ken Kundert, “The Designer’s Guide to Spice & Spectre,” Boston, Kluwer, 1995.

2. Ken Kundert, “The Designer's Guide to Verilog-AMS, 2004.

3. Cadence Whitepaper : Creating Analog Behavioral Models

4. Designers Guide Community. [Online] http://www.designers-guide.org/

5. Virtuoso Spectre DesignerReference [Online]
6. http://lumerink.com\courses\ECE614\Handouts\Spectre Designer Reference.pdf

7. Virtuoso Spectre Circuit Simulator RF Analysis User Guide [Online]
8. http://www.seas.gwu.edu/~vlsi/ece218/SPRING/reference/manual_cadence_spectreRF.pdf

9. Information on linking Matlab and Spectre in Linux environment. [Online] http://www.lumerink.com/courses/ECE697A/docs/Cadence+Spectre+Matlab+Toolbox.pdf

1. Verilog-AMS Language Reference Manual. [Online] http://www.eda.org/verilog-ams/htmlpages/lit.html