Spectre Tutorial

- Spectre will be used for transistor level simulation
- To use the SUNs, remote login into either hoth0, hoth1, or hoth2 and do “swsetup cadence-ncsu”.
- To use the LINUX boxes
  - Login into PC Linux cluster: ssh talus.ece.msstate.edu
  - Type ‘swsetup sungrid’, then ‘qlogin’, will start an interactive job on the processor with the lowest load.
  - To place on path,do: ‘swsetup cadence-ncsu-new’
- Use online help or PDF docs at ~reese/cadence_docs
- Documentation
  - Spectre User Manual – most helpful for first time users
  - Spectre Reference Manual – details on all available statements
  - Verilog-A Reference Manual – details on Verilog-A language
  - Spectre HDL – details Spectre HDL, a proprietary HDL. Has been largely replaced by standard languages such as Verilog-A. However, Analog library model detailed in Verilog-A reference manual is written in Spectre HDL

More Docs

- Appendix D of Verilog-A reference gives a pre-defined library of Analog Component library
- Source code for these models are at:

/opt/ecad/cadence/v5.0/ic_50/tools/dfII/samples/artist/ahdlLib/

For model details, look at source code for a model under:

model_name/ahdl/ahdl.def

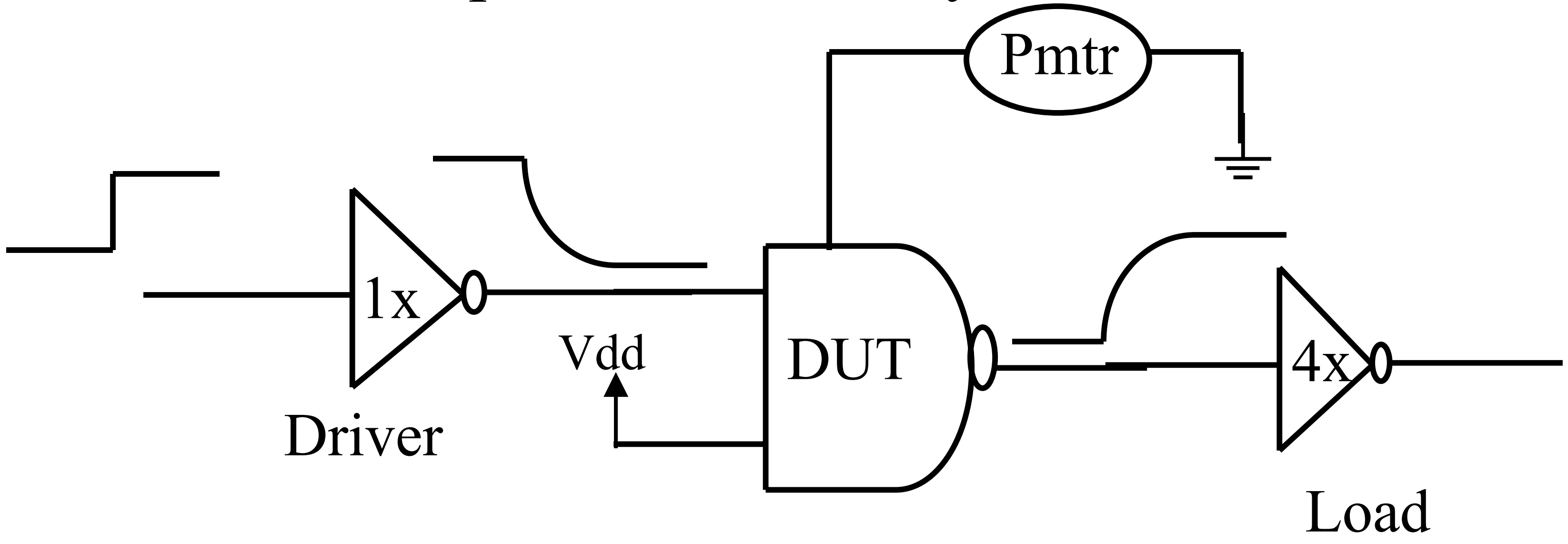
(e.g., delta_probe/ahdl/ahdl.def)

Note: These models are written in Spectre HDL.

Spectre

- Spectre accepts either SPICE or Spectre syntax
  - Spectre syntax less restrictive than SPICE (I.e., in Spectre syntax, element names do not have to start with a particular letter)
  - In my files,will mix SPICE and Spectre syntax freely just because I am used to SPICE
- The 'awd' program is used to view waveforms
  - A powerful waveform viewer, but can take many button clicks to produce a value from a waveform. It is better to use HDL models for signal measurement purposes.
- Verilog-A is an extension of Verilog that supports analog concepts such as Voltage, Current
  - Verilog-A can be called like sub-circuits from Spectre files
  - I will use Verilog-A for things like measurements because it is easier, faster than popping up a waveform viewer. Will use waveform viewer for debugging.
  - Predefined models in Affirma Analog Library are written in SpectreHDL

Example: Power, Delay Measurement



Measure average power through DUT

Measure TPLH, TPHL of DUT.

Grab zip archive ‘spectre_example.zip’ from WWW page

Spectre Example Files

- BSIM3V3 Model files from http://mosis.com
  - ami06.m for AMI 0.6u process
  - tsmc025.m for TSMC 0.25 process
  - tsmc018.m for TSMC 0.18 process
  - Transistor model names are ‘N’, ‘P’.
- Parameters lmin, wmin have been added to files:
  - Lmin – minimum channel length
  - Wmin – minimum gate width
  - Specify L,W parameters for transistors in terms of these parameters and your designs can be tested with different processes.

Spectre Example Files (cont)

- ‘def.m’ defines ‘N_def’, ‘P_def’ models which calculate default values for AS, AD, PS, PD based upon L, W parameters
  - Handy if just testing pre-layout designs (extracted transistors will always have these values).
  - Default values for AS,AD, PS, PD in BSIM3V3 model is zero – this is bad – always use non-zero geometry values.

‘def.m’ file

```
subckt N_def (d g s b)
  parameters L=lmin W=wmin
M0 (d g s b) N w=W l=L as=(W*wmin) ad=(W*wmin) \
ps=(2*W+2*wmin) pd=(2*W+2*wmin)  region=sat
ends N_def

subckt P_def (d g s b)
  parameters L=lmin W=wmin
M0 (d g s b) P w=W l=L as=(W*wmin) ad=(W*wmin) \
ps=(2*W+2*wmin) pd=(2*W+2*wmin)  region=sat
ends P_def
```

pmeas.va, *delta_probe.def*

- *pmeas.va* is a Verilog-A model that implements a power supply that reports average power usage
  - Included by *power_dly.sp* which is the top level Spice file
- *delta_probe.def* is a Spectre HDL model that implements a probe for measuring delay between two events
  - Included by *power_dly.sp* which is the top level Spice file
  - Provided in the sample Analog HDL model library in the Cadence distribution
  - A very flexible model, look at the source code for more documentation or Appendix D in the Verilog-A language reference guide. Very similar in capability to the HSPICE '.measure' statement if you are used to that.

‘power_dly.sp’ File

```
simulator lang=spectre
global 0 gnd vdd

//pwr=subckts saves power for subcircuts
opts options pwr=subckts save=allpub
aliasGnd (gnd 0 ) vsource type=dc dc=0

//include "ami06.m"

//include "tsmc025.m"

include "tsmc018.m"

include "def.m"
```

Global nodes, can be used in subcircuits without passing them on parameter list.

Save power info for all subcircuits,

Vsupply for ground.

Include files for different technologies

Defines ‘N_def’, ‘P_def’ default devices based on ‘N’, ‘P’ models from included tech file.

Parameter Definitions

```
parameters vdd_core=3.3 vss_core=0.0
parameters tr=.1n tf=.1n
parameters cload=15f
parameters pdelta=0.05n
parameters clkper=4n
parameters  pwrstop=10*clkper
```

Should use parameter definitions for constant values – makes it easier to experiment with different values.

Subcircuit Definitions

```
// lmin, wmin defined in model file
subckt NAND2X1 A B Y vddc gndc
    M3 (Y A net29 gndc) N_def w=2*wmin l=lmin
    M4 (net29 B gndc gndc) N_def w=2*wmin l=lmin
    M2 (Y B vddc vddc) P_def w=2*wmin l=lmin
    M1 (Y A vddc vddc) P_def w=2*wmin l=lmin
ends NAND2X1

subckt INVX1 A Y vddc gndc
    M2 (Y A gndc gndc) N_def w=wmin l=lmin
    M1 (Y A vddc vddc) P_def w=2*wmin l=lmin
ends INVX1

subckt INVX4 A Y vddc gndc
    M2 (Y A gndc gndc) N_def w=wmin*4 l=lmin
    M1 (Y A vddc vddc) P_def w=2*wmin*4 l=lmin
ends INVX4
```

Using lmin, wmin so can experiment with different technologies.

Using N_def, P_def so that default values for AS, AD, PS, PD are calculated from W,L.

Passing in nodes for Vdd, Gnd so that power supplies can be kept separate for power tracking.

Sources, Circuit Instantiation

Default power supply

```
v_vdd (vdd gnd) vsource type=dc dc=vdd_core

vpulse1 (a gnd) vsource type=pulse val0=vss_core
val1=vdd_core period=clkper rise=tr fall=tf width=clkper/2
```

Clock source for input node.

```
Driver (a a_out vdd gnd) INVX1

Dut (a_out vdd nand2_out vdd_dut gnd) NAND2X1

Load (nand2_out nand2_out1 vdd gnd) INVX4
```

Instantiate *driver*, *dut*, and *load* cells. Note that *dut* has a separate power supply (defined later in file).

Power, Delay Measurement

Verilog-A model for power supply model

```
ahdl_include "pmeas.va"
Pmtr (vdd_dut) pmeas vsrc=vdd_core deltatime=pdelta period=clkper
```

Instantiate power supply,will report avg pwr used.

```
// delay measurement
ahdl_include "delta_probe.def"
dut_tplh (a_out 0 nand2_out 0) delta_probe start_val=0.7*vdd_core \
start_mode=fall stop_val=0.3*vdd_core stop_mode=rise
dut_tphl (a_out 0 nand2_out 0) delta_probe start_val=0.3*vdd_core \
start_mode=rise stop_val=0.7*vdd_core stop_mode=fall
```

'delta_probe' model used to measure delay between two events. Instantiate two probes to measure tplh, tphl of DUT (input 'a_out' to output 'nand2_out').

Transient Analysis

```
typ_tran1 tran step=0.01n stop=pwrstop

Per2 alter param=clkper value=2n

typ_tran2 tran step=0.01n stop=pwrstop

Per3 alter param=clkper value=8n

typ_tran3 tran step=0.01n stop=pwrstop
```

First transient analysis.

Change clock period to 2ns, run again.

Change clock period to 8 ns, run again.

pmeas.va

```
//VerilogA

`include "constants.h"
`include "discipline.h"


module pmeas (psrc);

output psrc;

electrical psrc;

parameter real vsrc=5.0;

parameter real deltatime=0.05e-9;

parameter real period=1.0e-9;
```

Standard include files

Functions as a power supply, so single connection is an output node.

Voltage for supply output

Defines how often output current is sampled for power measurement.

Toggle frequency of DUT so can compute equivalent capacitance value.

pmeas.va (cont.)

```
integer numsteps;
real avg_pwr,real cap, i_sum ;

analog begin
 @(initial_step)   begin
    i_sum = 0.0;
    avg_pwr = 0.0;
    numsteps = 0;
  end

 @(timer(0, deltatime))
  begin
    i_sum = i_sum + I(psrc);
    numsteps = numsteps + 1;
  end;

 @(final_step) begin
     avg_pwr = (i_sum*vsrc)/numsteps;
     cap = avg_pwr*period/(vsrc*vsrc);
     $display("%M: Avg pwr = %g, Capacitance: %g\n",avg_pwr, cap);

 end

 V(psrc)  <+ vsrc;
end
```

Internal variables.

Triggered once at time=0, initialize variables.

Triggered every deltatime, keep output current sum, and # of steps.

Triggered at simulation end, compute/print average power, capacitance.

Assign output voltage.

Running Spectre

```
%  swsetup ncsu-cadence
%  spectre power_dly.sp
```

When running Spectre, will get several warnings about switching between Spice input mode and Spectre input mode – this is ok.

Will also get a warning about ‘Only one connection found to node 0’. Node ‘0’ is the default node name for ground – we use the ‘gnd’ power supply for this, so this warning can be ignored.

Spectre Output

Not all Spectre output is shown, only part of it.

```
****************************************************
Transient Analysis `typ_tran1': time = (0 s -> 40 ns)
****************************************************
..........9...........8...........7...........6...........5........
...4...........3...........2...........1...........0

Pmtr: Avg pwr = -6.71038e-05, Capacitance: -2.46479e-14

Number of accepted tran steps = 1582.
Initial condition solution time = 10 ms.
Intrinsic tran analysis time = 1.19 s.

Module instance: dut_tphl
    Start argument = 2.18142721893e-09 s.
    Stop argument = 2.23273276011e-09 s.
    Delta = 5.13055411827e-11 s.
Module instance: dut_tplh
    Start argument = 7.69079045269e-11 s.
    Stop argument = 1.40853531626e-10 s.
    Delta = 6.3945627099e-11 s.
Total time required for tran analysis `typ_tran1' was 1.2 s.
```

Output from ‘pmeas’ model.

Output from ‘delta_probe’ instances.

Waveform Display

Output data is placed in *power_dly.raw/* directory.

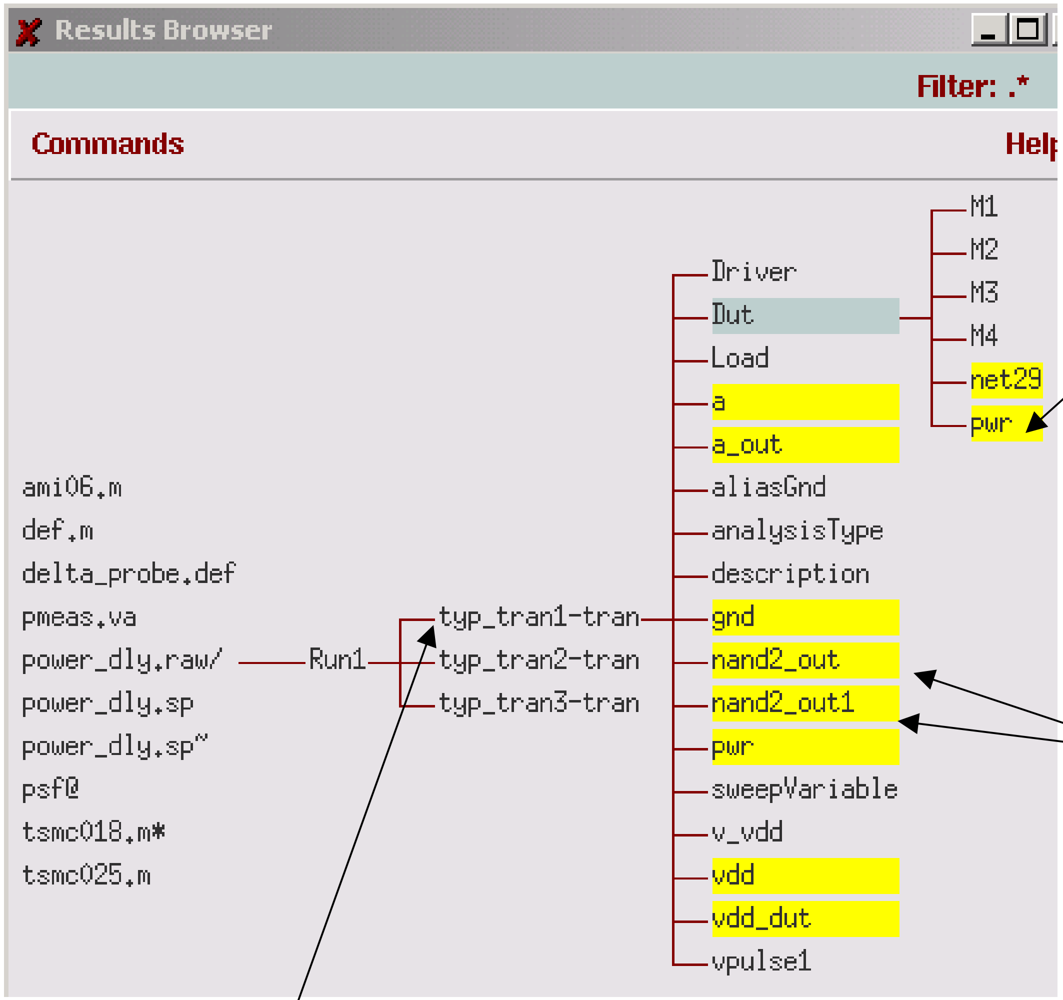
The waveform display tool is called ‘awd’.

To start waveform viewer, do:

```
 % awd  -dataDir power_dly.raw
```

This will open several windows. Chapter 2 of the Spectre User Guide has a good introduction to ‘awd’ usage.

AWD Results Browser



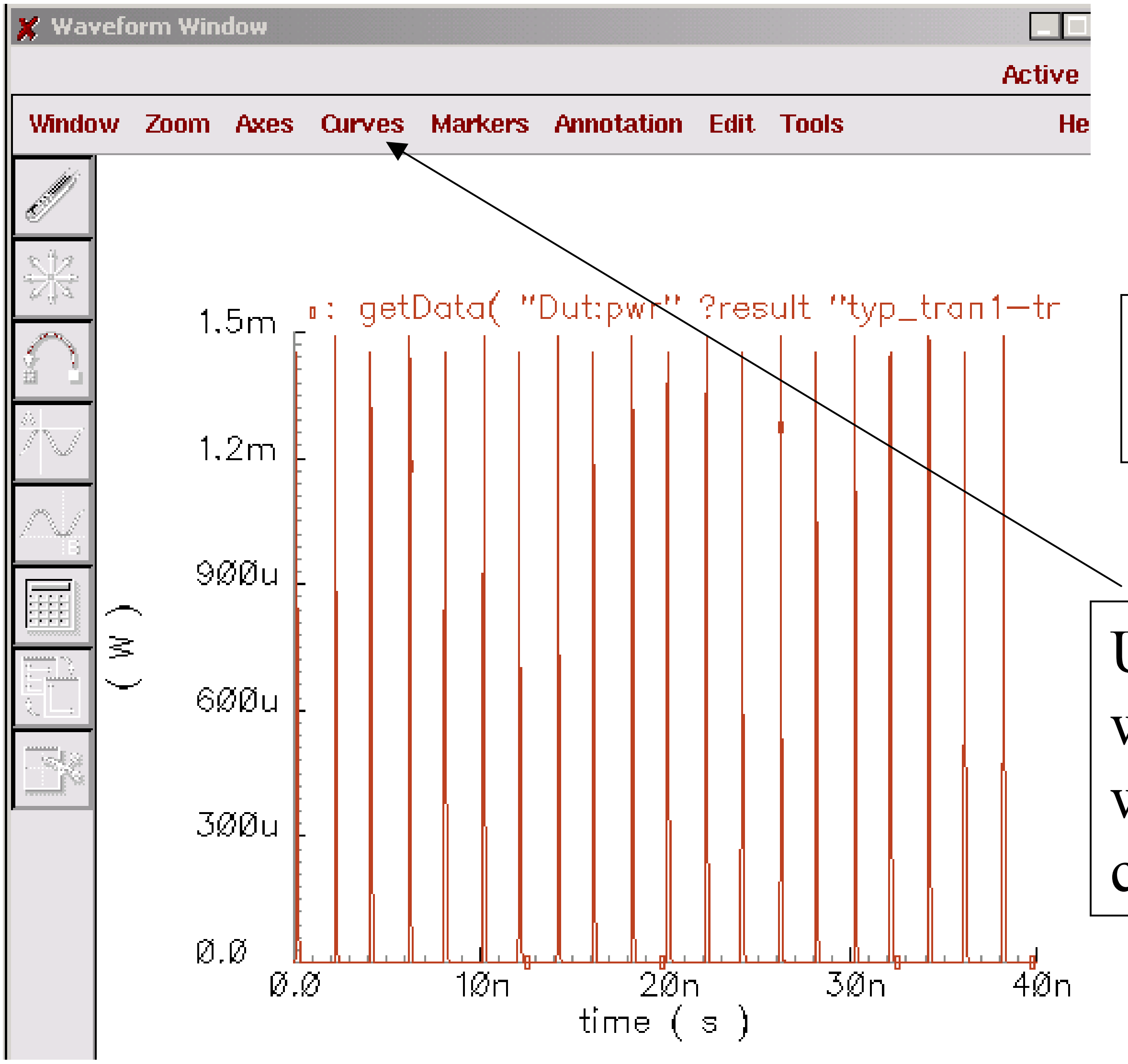
```
ami06.m
def.m
delta_probe.def
pmeas.va
power_dly.raw/ ──Run1──┬ typ_tran1-tran──┬ Driver ──
                       ├ typ_tran2-tran  ├ Dut ─────┬ M1
                       └ typ_tran3-tran  ├ Load     ├ M2
power_dly.sp                             ├ a        ├ M3
power_dly.sp~                            ├ a_out    ├ M4
psf@                                     ├ aliasGnd ├ net29
tsmc018.m*                               ├ analysisType └ pwr
tsmc025.m                                ├ description
                                         ├ gnd
                                         ├ nand2_out
                                         ├ nand2_out1
                                         ├ pwr
                                         ├ sweepVariable
                                         ├ v_vdd
                                         ├ vdd
                                         ├ vdd_dut
                                         └ vpulse1
```

Power waveform selected subcircuit (Dut). Must have *pwr=subckts* option in Spice file for this to be generated.

Waveforms for internal nodes. Right click to display them.

Results from 1st tran analysis, left click to expand.

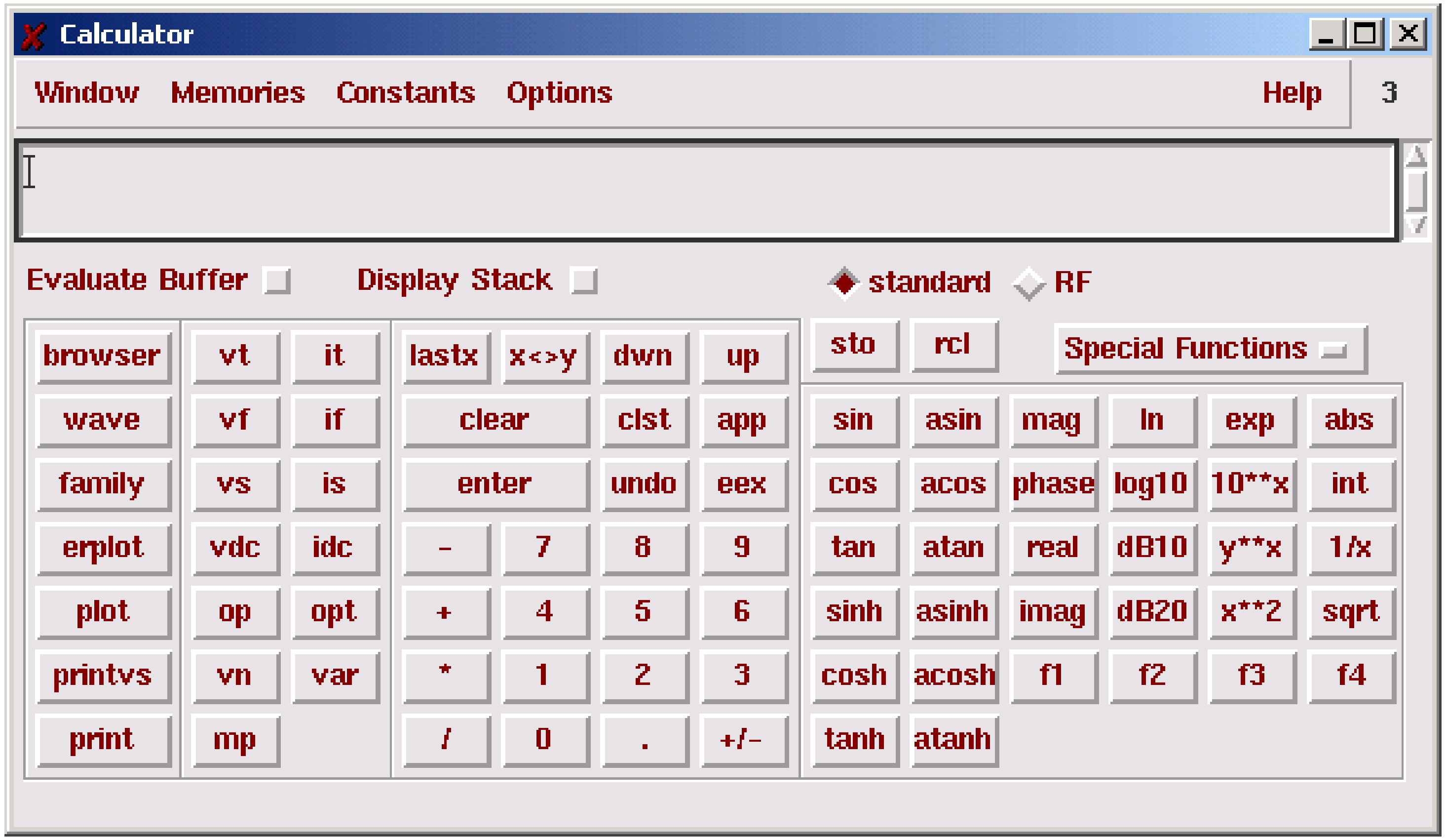
AWD Waveform Display



Power Waveform for DUT subcircuit.

Use Curves →Edit if want to delete waveforms, change color, etc.

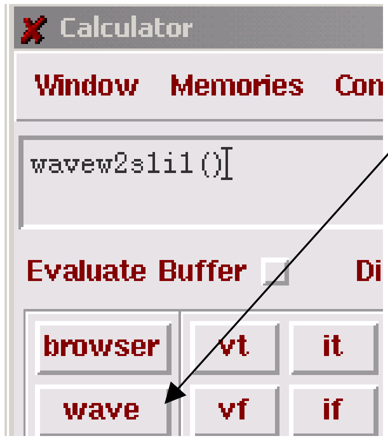
AWD Calculator Window



Evaluate Buffer ☐ Display Stack ☐ [✓ standard] RF

Used to perform calculations on waveforms, see Spectre User manual for more details.

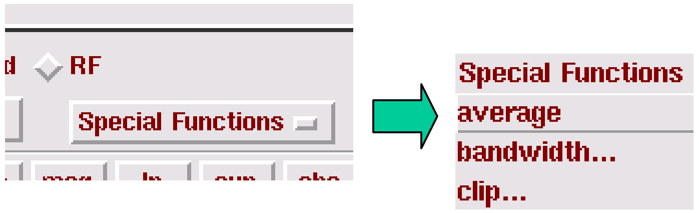
Calculating Average Power

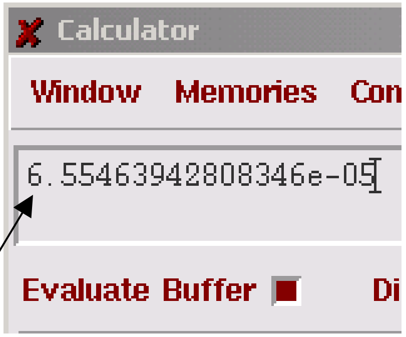


```
wavew2sli1()
```

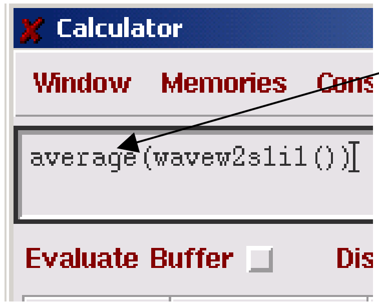
1. Left click on 'wave' , then left click on waveform in waveform display. Should see ''wavew2sli()' appear in calculator window.

2. Use 'Special Functions' button on calculator to select 'average'.





```
6.55463942808346e-05
```



```
average(wavew2sli1())
```

3. Window now shows 'average' function applied to waveform.

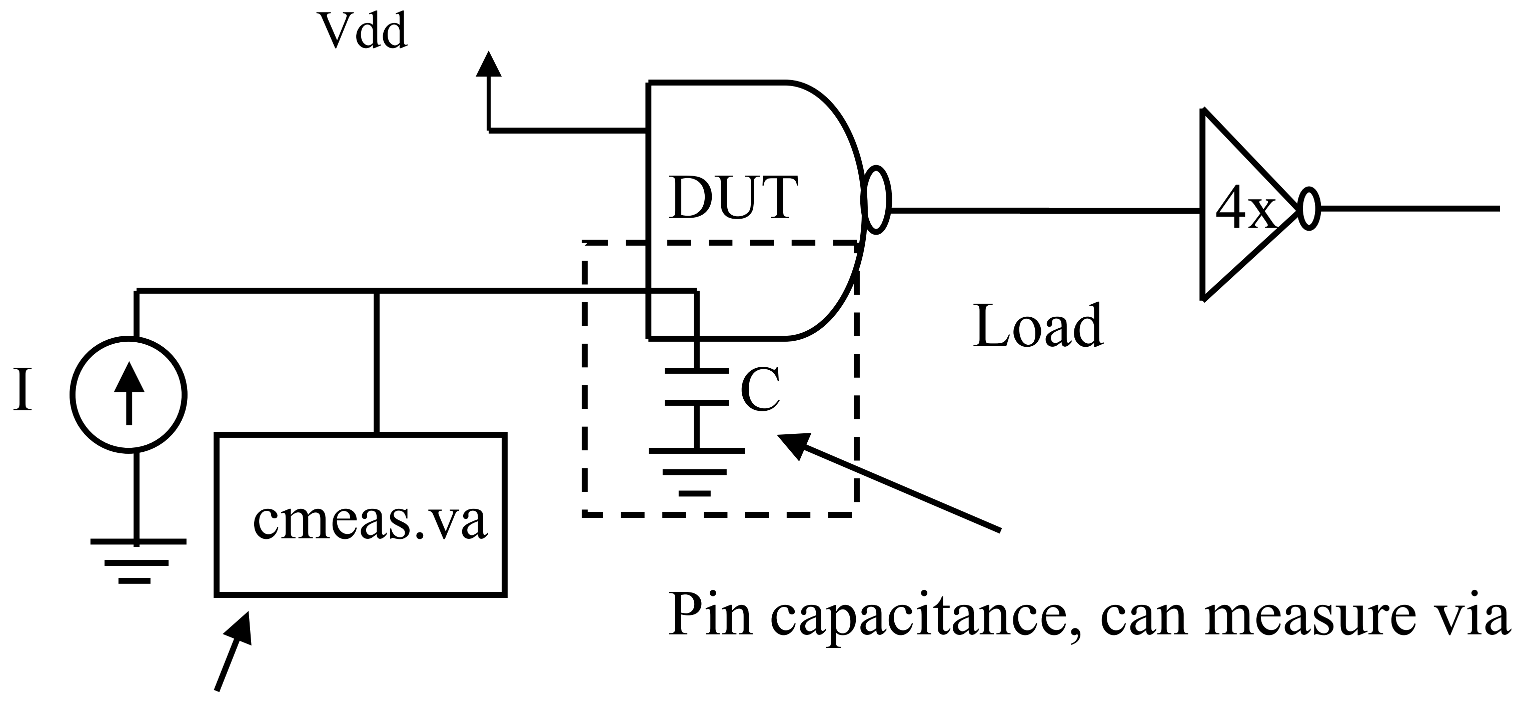
4. Use 'evaluate buffer' option perform calculation. Results will not be exact match of 'pmeas.va' but will be close.

General Notes on Simulation

- Use waveform display for debugging, use probes, models for measuring values
  - Much faster, repeatable
- Question your results – .ie, if you get a power in 10’s of watts or delay in microseconds, something is probably wrong.
- In reports, don’t use more than 3 significant digits. Providing an answer like 67.0332459 is meaningless.
- Do not wait until the last minute – most simulation assignments will take multiple tries.

Pin Capacitance Measurement

To calculate net loading, need to perform pin capacitance characterization.

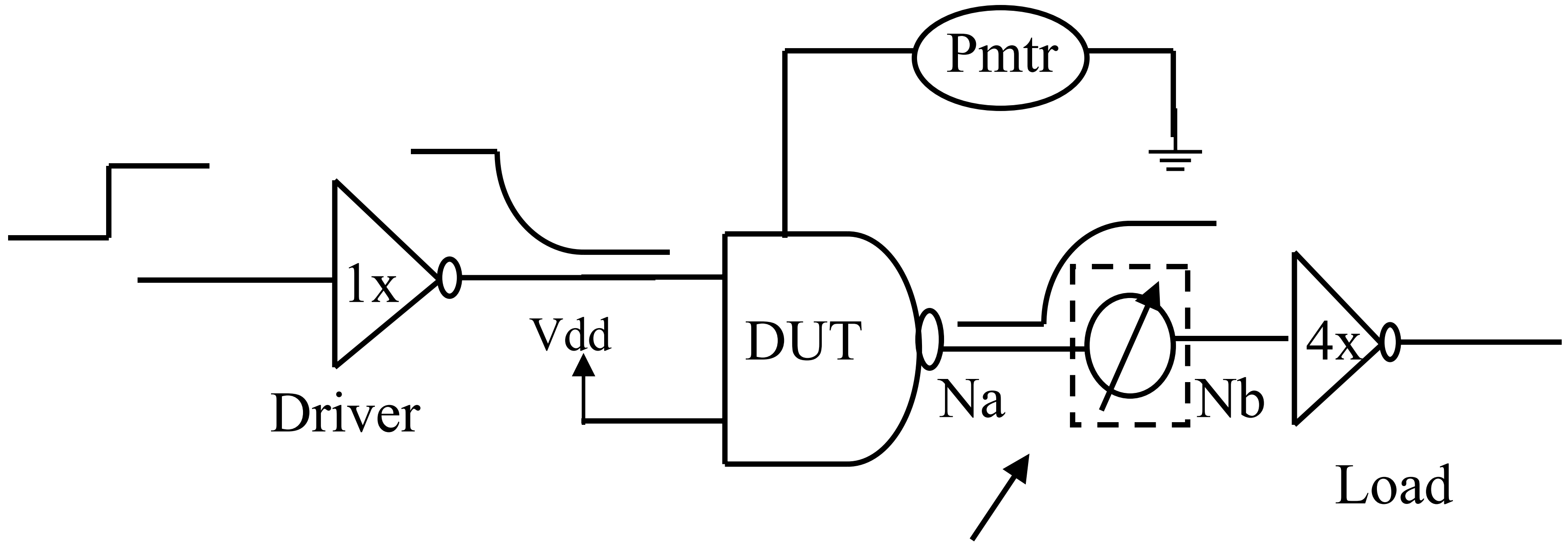


Pin capacitance, can measure via:

$C = I/(dV/dt) = (I * dt)/dV$

Will Verilog-A model to measure capacitance

Current Measurement

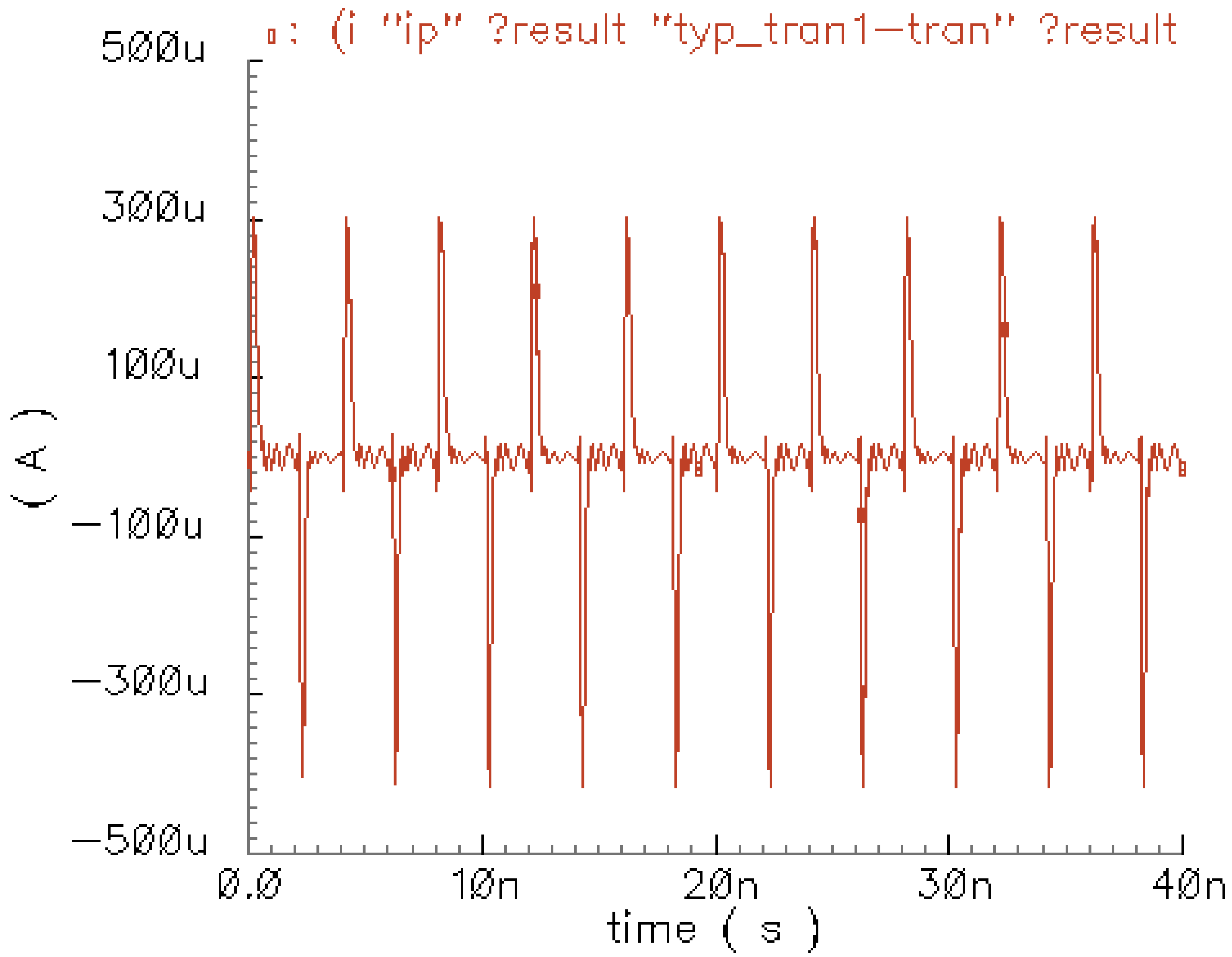


Insert a current probe on some gate output to get a feel for what IDSAT is for a technology. Format of a current probe is:

```
ip (na  nb) iprobe
```

The probe name ‘ip’ will show as value that can be plotted in the waveform window.

Current Waveform



max current about 0.3 mA, use this value for constant current source in capacitance measurements.

cmeas.va

```
module cmeas (din);
input din;
electrical din;

parameter real isrc=1.0e-4;
parameter real deltatime=0.05e-9;
parameter real start_time = 2.0e-9;
parameter real stop_time = 7.0e-9;

analog begin
 @(initial_step)   begin
   flag = 0;
  end

 @(timer(0, deltatime))
  begin
        if ( ($realtime > stop_time) && (flag == 1)) begin
           delta_v = V(din) - start_v;
           cap_val = isrc*(stop_time-start_time)/(delta_v);
           $display("Cap= %g\n",cap_val);
           flag = 2;
         end
        if ( ($realtime > start_time) && !flag) begin
           start_v = V(din);
           flag = 1;
         end;
  end;
end
```

dV between start/stop times

Capacitance calculation

Voltage at start time

Capacitance Measurement Results

tsmc 0.25u (cap in fF)

	Pin A	Pin B
INVX1	4.7	
INVX4	20	
NAND2X	6.5	6.5

Current source in cap_meas.sp

```
i1 (0 a) isource dc=0 type=pulse delay=1n val0=0 val1=i_src
rise=0.0001n  fall=0.01n width=100n


Dut (a nand2_out vdd gnd) INVX4

ahdl_include "cmeas.va"

cmeter (a) cmeas isrc=i_src v_vdd=vdd_core

Load (nand2_out nand2_out1 vdd gnd) INVX4
```