Chapter 8

Cell Characterization

```
/* General Syntax of a Technology Library */
library (nameoflibrary) {
... /* Library level simple and complex attributes */
... /* Library level group statements */
... /* Default attributes */
... /* Scaling Factors for delay calculation */

/* Cell definitions */

cell (cell_name) {
   ... /* cell level simple attributes */

   /* pin groups within the cell */
   pin(pin_name) {
        ... /* pin level simple attributes */

        /* timing group within the pin level */
        timing(){
           ... /* timing level simple attributes */
                } /* end of timing */

        ... /* additional timing groups */

        } /* end of pin */

    ... /* more pin descriptions  */

} /* end of cell */

... /* more cells */

} /* end of library */
```

Figure 8.1: General format of a **Liberty** (**.lib**) file

```
library(foo) {

  delay_model : table_lookup;
  in_place_swap_mode : match_footprint;

  /* unit attributes */
  time_unit : "1ns";
  voltage_unit : "1V";
  current_unit : "1uA";
  pulling_resistance_unit : "1kohm";
  leakage_power_unit : "1nW";
  capacitive_load_unit (1,pf);

  slew_upper_threshold_pct_rise : 80;
  slew_lower_threshold_pct_rise : 20;
  slew_upper_threshold_pct_fall : 80;
  slew_lower_threshold_pct_fall : 20;
  input_threshold_pct_rise : 30;
  input_threshold_pct_fall : 70;
  output_threshold_pct_rise : 70;
  output_threshold_pct_fall : 30;
  nom_process : 1;
  nom_voltage : 5;
  nom_temperature : 25;
  operating_conditions ( typical ) {
     process : 1;
     voltage : 5;
     temperature : 25;
  }
  default_operating_conditions : typical;

  lu_table_template(delay_template_5x5) {
    variable_1 : input_net_transition;
    variable_2 : total_output_net_capacitance;
    index_1 ("1000.0, 1001.0, 1002.0, 1003.0, 1004.0");
    index_2 ("1000.0, 1001.0, 1002.0, 1003.0, 1004.0");
  }
  power_lut_template(energy_template_5x5) {
    variable_1 : input_transition_time;
    variable_2 : total_output_net_capacitance;
    index_1 ("1000.0, 1001.0, 1002.0, 1003.0, 1004.0");
    index_2 ("1000.0, 1001.0, 1002.0, 1003.0, 1004.0");
  }
```

Figure 8.2: Example technology header for a **Liberty** (**.lib**) file

```
/* Because the cell area is in units of square microns, all the       *
 * distance units will be assumed to be in microns or square microns. */

/* fudge = correction factor, routing, placement, etc. */
   fudge = 1.0;

/* cap = fudge * cap per micron                    *
 * I assume cap is in capacitance units per micron *
 * (remember that our capacitance unit is 1.0pf)   */
   cap = fudge * 0.000030; /* .03ff/micron for avg metal     */
   res = fudge * 0.00008   /* 80 m-ohm/square, in kohm units */

/* length_top = the length of one side of a square die (in our case, *
 * a 4 TCU die of 2500u on a side of core area) length_10k = the     *
 * length of one side of a block containing 10k gates (I'll assume   *
 * this is a core of a single TCU which is 900u on a side)           */
   length_10k = 900;
   length_top = 2500.0;

/* sqrt(5000/10000) = .71        *
 * sqrt(2000/10000) = .45 etc    */
   length_5k  = length_10k * 0.71;
   length_2k  = length_10k * 0.45;
   length_1k  = length_10k * 0.32;
   length_500 = length_10k * 0.22;
```

Figure 8.3: Define some variables in the **.lib** file for wire load calculations

```
      wire_load("top") {
       resistance : res ;
       capacitance : cap ;
       area : 1 ; /* i.e. 1 sq micron */
       slope : length_top * .5 ;
       fanout_length(1,2500); /* length          */
       fanout_length(2,3750); /* length * 1.5    */
       fanout_length(3,5000); /* length * 2      */
       fanout_length(4,5625); /* length * 2.5    */
       fanout_length(5,6250); /* length * 2.5    */
       fanout_length(6,6875); /* length * 2.75   */
       fanout_length(7,7500); /* length * 3      */
       }

       wire_load("10k") {
       resistance : res ;
       capacitance : cap ;
       area : 1 ;
       slope : length_10k * .5 ;
       fanout_length(1,900);  /* length          */
       fanout_length(2,1350); /* length * 1.5    */
       fanout_length(3,1800); /* length * 2      */
       fanout_length(4,2025); /* length * 2.5    */
       fanout_length(5,2250); /* length * 2.5    */
       fanout_length(6,2475); /* length * 2.75   */
       fanout_length(7,2700); /* length * 3      */
       }

       wire_load("5k") {
       resistance : res ;
       capacitance : cap ;
       area : 1 ;
       slope : length_5k * .5 ;
       fanout_length(1,639);  /* length          */
       fanout_length(2,959);  /* length * 1.5    */
       fanout_length(3,1278); /* length * 2      */
       fanout_length(4,1439); /* length * 2.5    */
       fanout_length(5,1598); /* length * 2.5    */
       fanout_length(6,1757); /* length * 2.75   */
       fanout_length(7,1917); /* length * 3      */
       }

/* define how the wire loads are selected based on total circuit area */
   wire_load_selection (foo) {
   wire_load_from_area (   0,  3000000,  "5k");
   wire_load_from_area (3000000, 7000000, "10k");
   }

   default_wire_load_mode : enclosed ;
   default_wire_load : "top" ;
   default_wire_load_selection : "foo" ;
/* end of wire_load calculation */
```

Figure 8.4: Use the wire load variables to compute wire load models based on wire RC

```
 /* -------------- *
  * Design : INVX1 *
  * -------------- */
cell (INVX1) {
  cell_footprint : inv;
  area : 129.6;
  cell_leakage_power : 0.0310651;
  pin(A)  {
    direction : input;
    capacitance : 0.0159685;
    rise_capacitance : 0.0159573;
    fall_capacitance : 0.0159685; }
  pin(Y)  {
    direction : output;
    capacitance : 0;
    rise_capacitance : 0;
    fall_capacitance : 0;
    max_capacitance : 0.394734;
    function : "(!A)";
    timing() {
      related_pin : "A";
      timing_sense : negative_unate;
      cell_rise(delay_template_5x5) {
        index_1 ("0.06, 0.18, 0.42, 0.6, 1.2");
        index_2 ("0.025, 0.05, 0.1, 0.3, 0.6");
        values ( \
          "0.147955, 0.218038, 0.359898, 0.922746, 1.76604", \
          "0.224384, 0.292903, 0.430394, 0.991288, 1.83116", \
          "0.365378, 0.448722, 0.584275, 1.13597, 1.97017", \
          "0.462096, 0.551586, 0.70164, 1.24437, 2.08131", \
          "0.756459, 0.874246, 1.05713, 1.62898, 2.44989"); }
      rise_transition(delay_template_5x5) {
        index_1 ("0.06, 0.18, 0.42, 0.6, 1.2");
        index_2 ("0.025, 0.05, 0.1, 0.3, 0.6");
        values ( ... ); }
      cell_fall(delay_template_5x5) {
        index_1 ("0.06, 0.18, 0.42, 0.6, 1.2");
        index_2 ("0.025, 0.05, 0.1, 0.3, 0.6");
        values ( ... ); }
      fall_transition(delay_template_5x5) {
        index_1 ("0.06, 0.18, 0.42, 0.6, 1.2");
        index_2 ("0.025, 0.05, 0.1, 0.3, 0.6");
        values ( ... ); }
    } /* end timing */
    internal_power() {
      related_pin : "A";
      rise_power(energy_template_5x5) {
        index_1 ("0.06, 0.18, 0.42, 0.6, 1.2");
        index_2 ("0.025, 0.05, 0.1, 0.3, 0.6");
        values ( ... ); }
      fall_power(energy_template_5x5) {
        index_1 ("0.06, 0.18, 0.42, 0.6, 1.2");
        index_2 ("0.025, 0.05, 0.1, 0.3, 0.6");
        values ( ... ); }
    } /* end internal_power */
  }   /* end Pin Y */
}     /* end INVX1 */
```

Figure 8.5: Example **Liberty** description of an inverter (with most look-up table data not included)

```
-----------------------------------------------------
     Operator   Description
-----------------------------------------------------
'       invert previous expression
!       invert following expression
^       logical XOR
*       logical AND
&       logical AND
space   logical AND
+       logical OR
|       logical OR
1       signal tied to logic 1
0       signal tied to logic 0
-----------------------------------------------------
```

Figure 8.6: Function description syntax for **Liberty** files (EQN format)

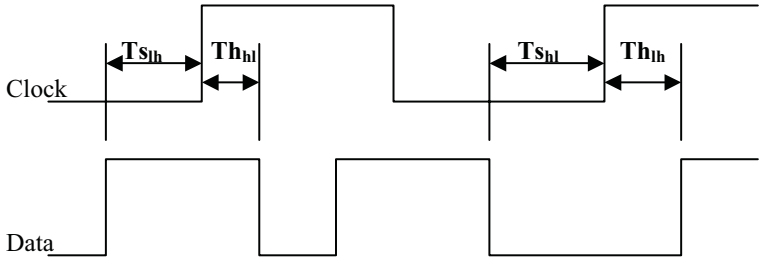

Figure 8.7: Setup and hold timing relative to a rising-edge clock or low-level gate

```
/* positive edge triggered JK flip flop */
ff(IQ,IQN) {
clocked_on : "CLK" ;
next_state : "(J K IQ') + (J K') + (J' K' IQ)" ;
clear : "CLR'" ;
preset : "SET'" ;
clear_preset_var1 : X ;
clear_preset_var2 : X ;}

/* positive edge triggered D flip flop */
ff (IQ, IQN) {
next_state : "D * CLR'" ;
clocked_on : "CLK" ;}
```

Figure 8.8: ff descriptions for a JK and a D flip flop

```
/* D-latch with active high gate (clock) signal, active low clear */
latch(IQ, IQN) {
    enable : "GATE" ;
    data_in : "D" ;
    clear : "CLR'" ;}

/* set-reset (SR) latch with active-low set and reset */
latch(IQ, IQN) {
    clear : "S'" ;
    preset : "R'" ;
    clear_preset_var1 : L ;
    clear_preset_var2 : L ;}
```

Figure 8.9: **latch** descriptions for a D-latch and SR-latch

```
statetable ("J          K       CLK     CLR",   "IQ" ) {
     table: "-          -       -       L   :   -       : L, \
             -          -       ~F      H   :   -       : N, \
             L          L       F       H   :   L/H     : L/H, \
             H          L       F       H   :   -       : H, \
             L          H       F       H   :   -       : L, \
             H          H       F       H   :   L/H     : H/L" ;
       }
```

Figure 8.10: **statetable** description of a JK flip-flop

```
/* positive edge triggered D flip-flop with active low reset  */
cell(dff) {
    area : 972;
    cell_footprint : "dff";
    ff("IQ", "IQN") {
        next_state : " D ";
        clocked_on : " G ";
        clear : " CLR' ";
    }
    pin(D)    {
      direction : input;
      capacitance : 0.0225;

      timing() { /* hold time constraint for a rising transition on G */
        timing_type : hold_rising;
                rise_constraint(scalar) { values("-0.298"); }
                fall_constraint(scalar) { values("-0.298"); }
        related_pin : "G";
      }
      timing() { /* setup time constraint for a rising transition on G */
        timing_type : setup_rising;
                rise_constraint(scalar) { values("0.018"); }
                fall_constraint(scalar) { values("0.018"); }
        related_pin : "G";
      }
    } /* end of pin D */
    pin ( CLK )    {
      direction : input;
      capacitance : 0.0585;
      clock : true;
    } /* end of pin CLK */
    pin ( CLR )    {
      direction : input;
      capacitance : 0.0135;
    } /* end of pin CLR */
    pin ( Q )    {
      direction : output;
      function  : "IQ";

      timing () { /* propagation delay from rising edge of CLK to Q */
        timing_type : rising_edge;
                 cell_rise(lu5x5) { values( "..." );}
                 rise_transition(lu5x5) { values( "..." );}
                 cell_fall(lu5x5) { values( "..." );}
                 fall_transition(lu5x5) { values( "..." );}
        related_pin : "CLK";
      } /* end of Q timing related to CLK */

      timing () { /* propagation delay from falling edge of clear to Q=0 */
        timing_type : clear;
        timing_sense : positive_unate;
                 cell_fall(lu5x5) { values( "..." );}
                 fall_transition(lu5x5) { values( "..." );}
        related_pin : "CLR";
      } /* end of Q timing related to CLR */
    } /* end of pin Q */
  } /* end of cell dff */
```

Figure 8.11: Example D-type flip-flop description in **.lib** format

```
/* Enabled inverter or tristate inverter  */
  cell(eninv) {
    area : 324;
    cell_footprint : "eninv";
    pin(A) {
      direction : input;
      capacitance : 0.027;
    } /* end of pin A */
    pin(En) {
      direction : input;
      capacitance : 0.0135;
    } /*end of pin En */
    pin(Y) {
      direction : output;
      function : "A'";
      three_state : "En'";
      timing ()      {
        timing_sense : negative_unate;
        related_pin : "A";
         cell_rise(lu5x5) { values( " ... " );}
         rise_transition(lu5x5) { values( " ... " );}
         cell_fall(lu5x5) { values( " ... " );}
         fall_transition(lu5x5) { values( " ... " );}
         } /* end of enabled timing */
      timing() {
        timing_sense : positive_unate;
        timing_type : three_state_enable;
        related_pin : "En";
        cell_rise(delay_template_5x5) { values( " ... " );}
        rise_transition(delay_template_5x5) { values( " ... " );}
        cell_fall(delay_template_5x5) { values( " ... " );}
        fall_transition(delay_template_5x5) { values( " ... " );}
        } /* end of enable timing */
      timing() {
        timing_sense : negative_unate;
        timing_type : three_state_disable;
        related_pin : "En";
        cell_rise(delay_template_5x1) { values( " ... " );}
        rise_transition(delay_template_5x1) { values( " ... " );}
        cell_fall(delay_template_5x1) { values( " ... " );}
        fall_transition(delay_template_5x1) { values( " ... " );}
        } /* end of disable timing */
    } /* end of pin Y */
  } /* end of eninv */
```

Figure 8.12: Example tristate inverter **.lib** description

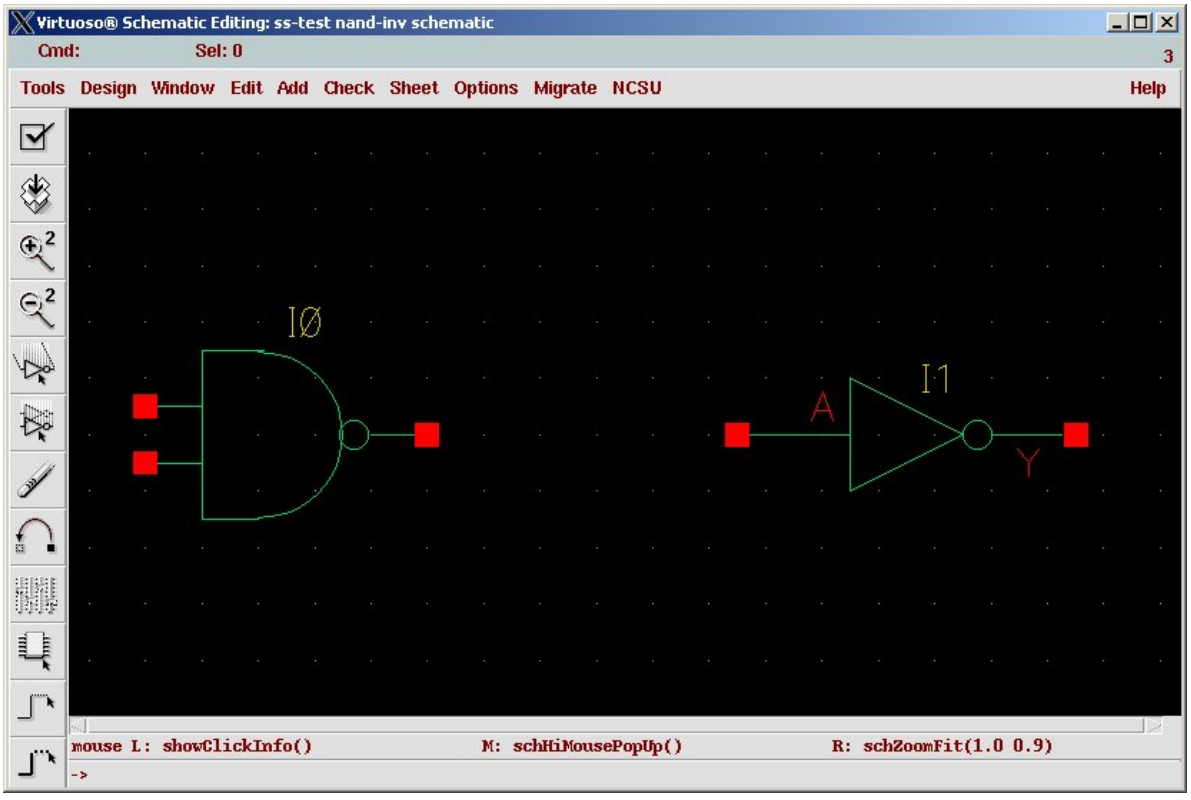

Figure 8.13: Schematic with one instance of **inv** and **nand2**

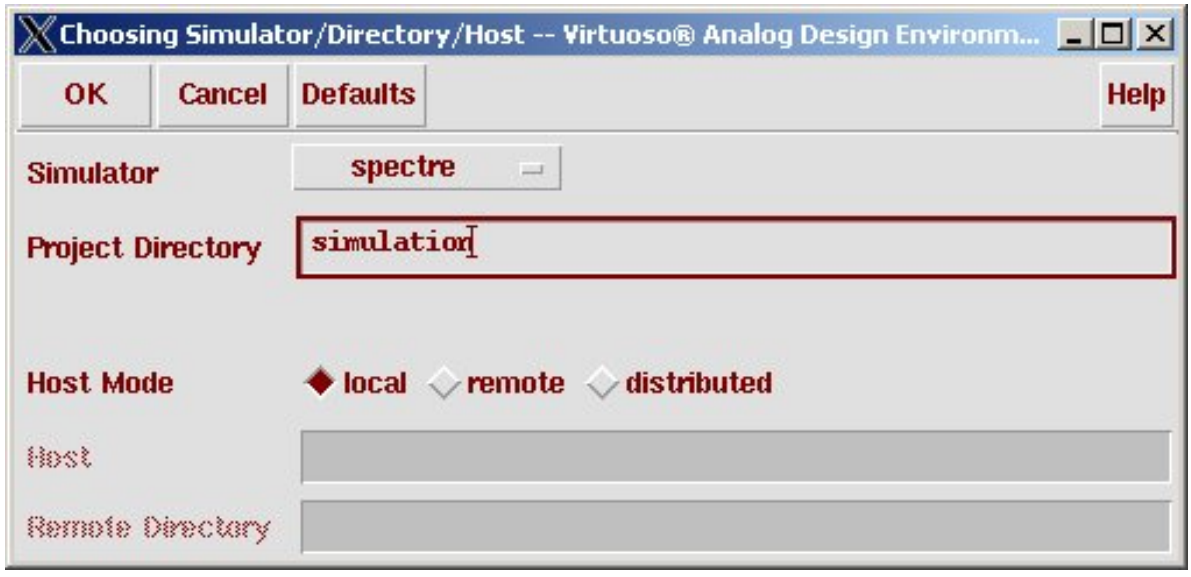

Figure 8.14: Choosing *Spectre* as the simulator

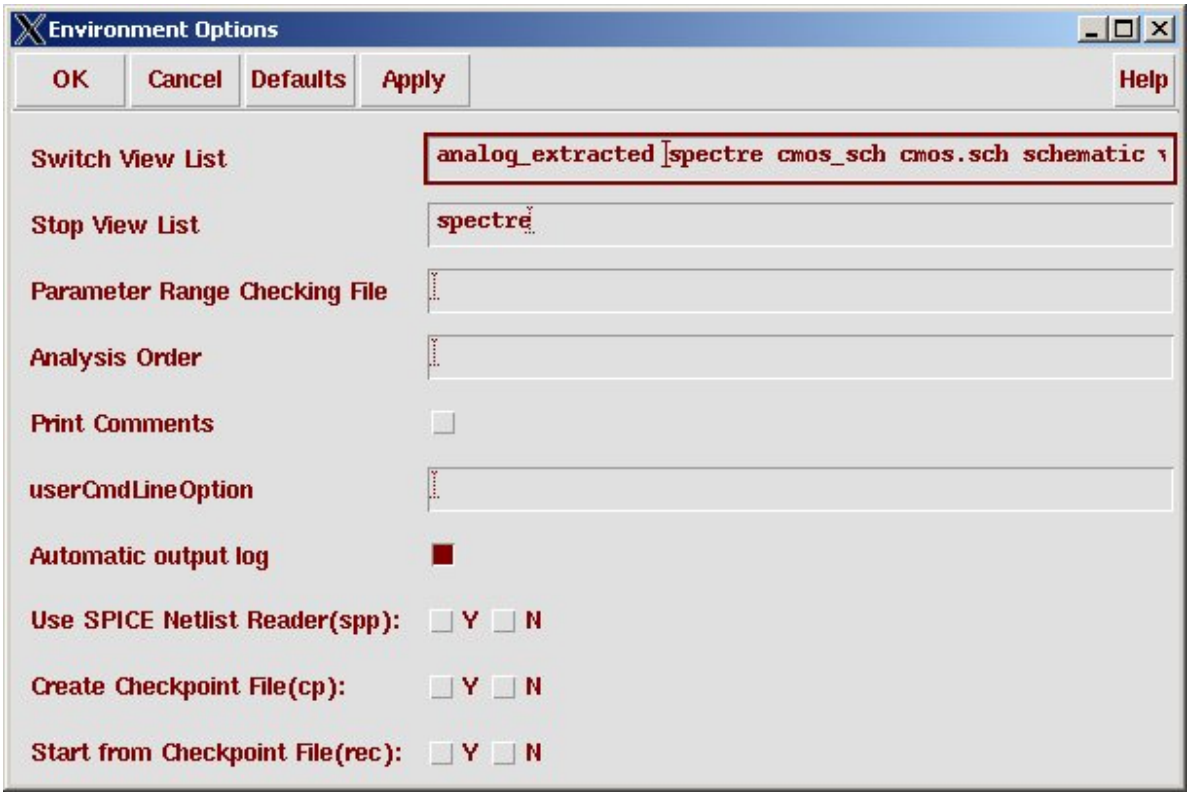

Figure 8.15: Adding **analog_extracted** to the **Switch View List**

```
// Generated for: Spectre
// Generated on: Aug 31 18:05:16 2006
// Design library name: tutorial
// Design cell name: signalstorm
// Design view name: schematic
simulator lang=spectre
global 0 vdd!

// Library name: tutorial
// Cell name: inv
// View name: analog_extracted
subckt inv A Y
    \+1 (Y A vdd! vdd!) ami06P w=6e-06 l=6e-07 as=9e-12 ad=9e-12 ps=9e-06 \
        pd=9e-06 m=1 region=sat
    \+0 (Y A 0 0) ami06N w=3e-06 l=6e-07 as=4.5e-12 ad=4.5e-12 ps=6e-06 \
        pd=6e-06 m=1 region=sat
ends inv
// End of subcircuit definition.

// Library name: tutorial
// Cell name: nand2
// View name: analog_extracted
subckt nand2 A B Y
    \+3 (vdd! B Y vdd!) ami06P w=6e-06 l=6e-07 as=5.4e-12 ad=9e-12 \
        ps=1.8e-06 pd=9e-06 m=1 region=sat
    \+2 (Y A vdd! vdd!) ami06P w=6e-06 l=6e-07 as=9e-12 ad=5.4e-12 \
        ps=9e-06 pd=1.8e-06 m=1 region=sat
    \+1 (Y B _6 0) ami06N w=6e-06 l=6e-07 as=2.7e-12 ad=9e-12 ps=9e-07 \
        pd=9e-06 m=1 region=sat
    \+0 (_6 A 0 0) ami06N w=6e-06 l=6e-07 as=9e-12 ad=2.7e-12 ps=9e-06 \
        pd=9e-07 m=1 region=sat
ends nand2
// End of subcircuit definition.

// Library name: tutorial
// Cell name: signalstorm
// View name: schematic
I1 (net1 net2) inv
I0 (net5 net4 net3) nand2
simulatorOptions options reltol=1e-3 vabstol=1e-6 iabstol=1e-12 temp=27 \
    tnom=27 scalem=1.0 scale=1.0 gmin=1e-12 rforce=1 maxnotes=5 maxwarns=5 \
    digits=5 cols=80 pivrel=1e-3 ckptclock=1800 \
    sensfile="../psf/sens.output" checklimitdest=psf
modelParameter info what=models where=rawfile
element info what=inst where=rawfile
outputParameter info what=output where=rawfile
designParamVals info what=parameters where=rawfile
primitives info what=primitives where=rawfile
subckts info what=subckts  where=rawfile
saveOptions options save=allpub
```

Figure 8.16: dut.scs file as generated by *Analog Environment*

```
simulator lang = spectre

// Library name: tutorial
// Cell name: inv
// View name: analog_extracted
subckt inv A Y vdd gnd
    \+1 (Y A vdd vdd) ami06P w=6e-06 l=6e-07 as=9e-12 ad=9e-12 ps=9e-06 \
        pd=9e-06 m=1 region=sat
    \+0 (Y A gnd gnd) ami06N w=3e-06 l=6e-07 as=4.5e-12 ad=4.5e-12 ps=6e-06 \
        pd=6e-06 m=1 region=sat
ends inv
// End of subcircuit definition.

// Library name: tutorial
// Cell name: nand2
// View name: analog_extracted
subckt nand2 A B Y vdd gnd
    \+3 (vdd B Y vdd) ami06P w=6e-06 l=6e-07 as=5.4e-12 ad=9e-12 \
        ps=1.8e-06 pd=9e-06 m=1 region=sat
    \+2 (Y A vdd vdd) ami06P w=6e-06 l=6e-07 as=9e-12 ad=5.4e-12 \
        ps=9e-06 pd=1.8e-06 m=1 region=sat
    \+1 (Y B _6 gnd) ami06N w=6e-06 l=6e-07 as=2.7e-12 ad=9e-12 ps=9e-07 \
        pd=9e-06 m=1 region=sat
    \+0 (_6 A gnd gnd) ami06N w=6e-06 l=6e-07 as=9e-12 ad=2.7e-12 ps=9e-06 \
        pd=9e-07 m=1 region=sat
ends nand2
// End of subcircuit definition.
```

Figure 8.17: **dut.scs** after required *ELC* modifications

```
db_open foo

set_var EC_SPICE_SIMPLIFY true
set_var EC_HALF_WIDTH_HOLD_FLAG true
set_var EC_SIM_NAME ''spectre''
set_var EC_SIM_TYPE ''spectre''
set_var EC_SPICE_SUPPLY1_NAMES ''vdd''
set_var EC_SPICE_SUPPLY0_NAMES ''gnd''

db_prepare -force
db_gate
db_close
exit
```

Figure 8.18: *Encounter Library Characterizer* **step1** script

```
==============================
      DESIGN : INV
==============================
DESIGN ( INV );
//      =================
//       PORT DEFINITION
//      =================
        INPUT A ( A );
        OUTPUT Y ( Y );
        SUPPLY0 GND ( GND );
        SUPPLY1 VDD ( VDD );
//      ===========
//       INSTANCES
//      ===========
        NOT ( Y, A );
END_OF_DESIGN;

==============================
      DESIGN : NAND2
==============================
DESIGN ( NAND2 );
//      =================
//       PORT DEFINITION
//      =================
        INPUT A ( A );
        INPUT B ( B );
        OUTPUT Y ( Y );
        SUPPLY0 GND ( GND );
        SUPPLY1 VDD ( VDD );
//      ===========
//       INSTANCES
//      ===========
        NAND ( Y, A, B );
END_OF_DESIGN;
```

Figure 8.19: *Encounter Library Characterizer* **step1** output (**db_gate** portion)

```
db_open foo

# Remove the next 3 lines to use the ipsd/ipsc
# deamons for load balancing on multiple machines
set_var EC_SIM_USE_LSF 1
set_var EC_SIM_LSF_CMD ``''
set_var EC_SIM_LSF_PARALLEL 10

set_var EC_SIM_NAME ``spectre''
set_var EC_SIM_TYPE ``spectre''
set_var EC_SPICE_SUPPLY1_NAMES ``vdd''
set_var EC_SPICE_SUPPLY0_NAMES ``gnd''
set_var EC_HALF_WIDTH_HOLD_FLAG true

db_spice -s spectre  -p typical -keep_log
db_close
exit
```

Figure 8.20: *Encounter Library Characterizer* **step2** script

```
elc>  db_spice -s spectre  -p typical -keep_log


   DESIGN        PROCESS       #ID          STATUS     IPDB
-------------+-------------+----------+--------------+-----------
INV            typical       D0000          SIMULATE     foo
INV            typical       D0001          SIMULATE     foo
============|=============|=============|==========|==============
INV            typical       2           2            foo
NAND2          typical       D0000          SIMULATE     foo
NAND2          typical       D0001          SIMULATE     foo
NAND2          typical       D0002          SIMULATE     foo
NAND2          typical       D0003          SIMULATE     foo
NAND2          typical       D0004          SIMULATE     foo
NAND2          typical       D0005          SIMULATE     foo
NAND2          typical       D0006          SIMULATE     foo
NAND2          typical       D0007          SIMULATE     foo
============|=============|=============|==========|==============
NAND2          typical       8           8            foo
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
               Simulation Summary
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
-------------+-------------+----------+--------------+-----------
-------------+-------------+----------+-------------+-----------+------------
   DESIGN    |   PROCESS   |   #ID    |   STAGE     |  STATUS   |
   IPDB
-------------+-------------+----------+-------------+-----------+------------
INV            typical       D0000     VERIFICATE   PASS        foo
INV            typical       D0001     VERIFICATE   PASS        foo
NAND2          typical       D0000     VERIFICATE   PASS        foo
NAND2          typical       D0001     VERIFICATE   PASS        foo
NAND2          typical       D0002     VERIFICATE   PASS        foo
NAND2          typical       D0003     VERIFICATE   PASS        foo
NAND2          typical       D0004     VERIFICATE   PASS        foo
NAND2          typical       D0005     VERIFICATE   PASS        foo
NAND2          typical       D0006     VERIFICATE   PASS        foo
NAND2          typical       D0007     VERIFICATE   PASS        foo
-------------+-------------+----------+-------------+----------

-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
 - Total Simulation : 10
 - Total Passed     : 10(100.00%)
 - Total Failed     : 0(0.00%)
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
elc>  db_close

Database : foo is closed
elc>  exit
```

Figure 8.21: *Encounter Library Characterizer* **step2** output (**db_spice** portion)

```
db_open foo
db_output -r foo.alf.rep -alf foo.alf -p typical
db_verilog -r foo.v
db_close
exit
```

Figure 8.22: *Encounter Library Characterizer* **step3** script

```
elc>  db_open foo

Database : foo is now opened
elc>  db_output -r foo.alf.rep -alf foo.alf -p typical

INV             typical   2008-11-02 18:18:14 (2008-11-03 01:18:14 GMT)
2 (100%)
NAND2           typical   2008-11-02 18:18:14 (2008-11-03 01:18:14 GMT)
8 (100%)
elc>  db_verilog -r foo.v

Reading : foo.ipdb/INV.design

==============================
      DESIGN : INV
==============================
Reading : foo.ipdb/NAND2.design

==============================
      DESIGN : NAND2
==============================
elc>  db_close

Database : foo is closed
elc>  exit
```

Figure 8.23: Example of running **step3** on this **foo** library

```
cell INV* {
footprint inv ;
area 129.6 ;
};

cell NAND2* {
footprint nand2 ;
area 194.4 ;
};

cell NOR2* {
footprint nor2 ;
area 194.4 ;
};
```

Figure 8.24: Example **footprints.def** file

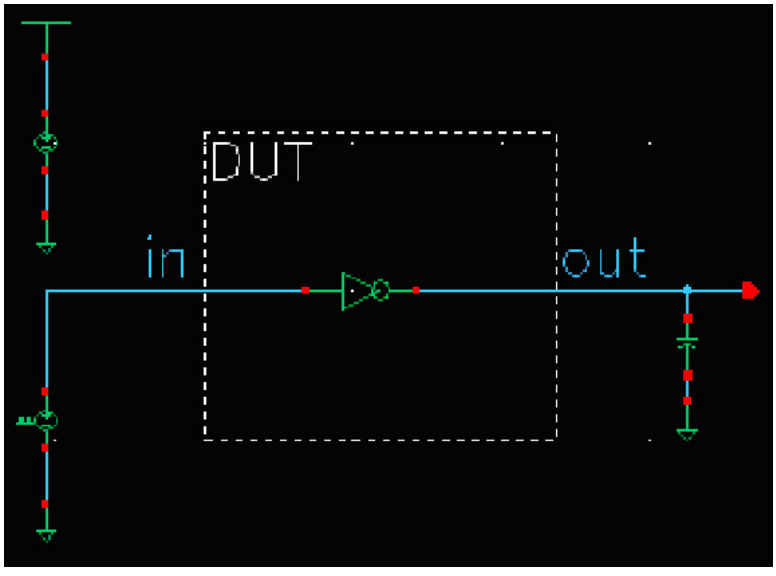

Figure 8.25: Testfixture for hand-simulation and characterization

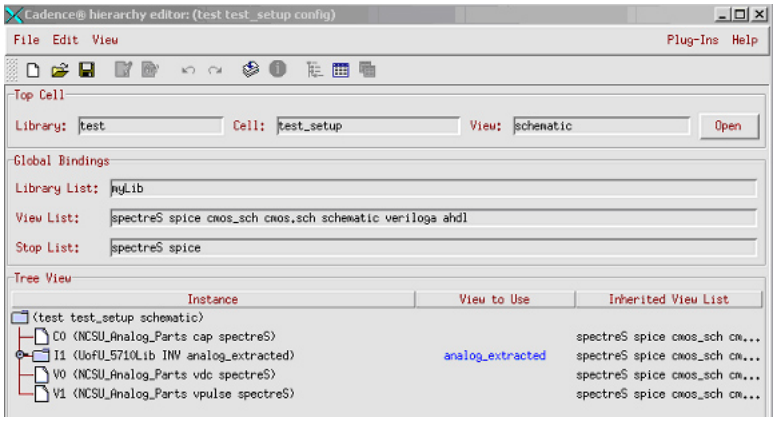

Figure 8.26: **config** view for hand-simulation and characterization

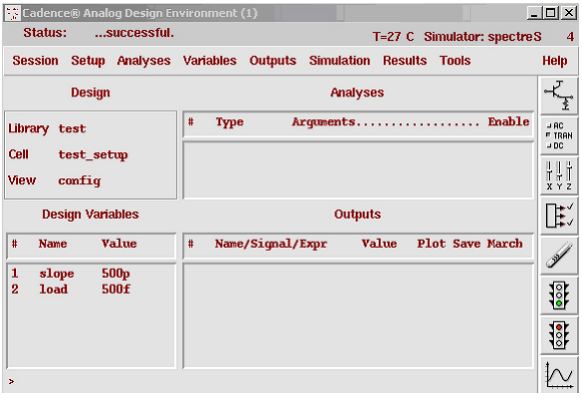

Figure 8.27: Setting variables in the **Analog Environment**

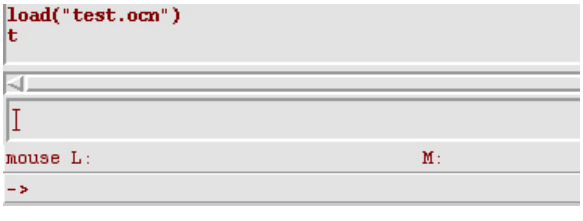

Figure 8.28: Loading the **test.ocn** script

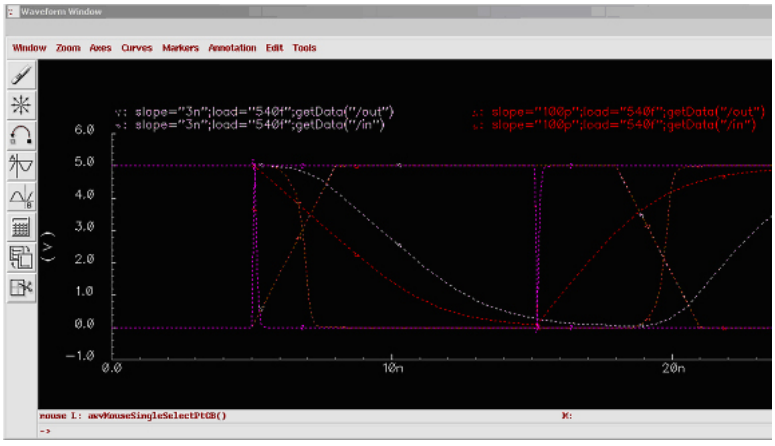

Figure 8.29: Plot output from a **test.ocn** simulation

```
 cell_rise(lu5x5) {
  values( \
 ".145649, .461407, .821977, 1.54147, 2.95759" ,\
 ".384865, .72295, 1.07269, 1.79433, 3.21885" ,\
 ".60943, 1.02507, 1.38113, 2.08612, 3.51231" ,\
 "1.0132, 1.55007, 1.98756, 2.70889, 4.09905" ,\
 "1.65337, 2.35774, 2.89704, 3.77152, 5.1905" );}

rise_transition(lu5x5) {
  values( \
 ".148306, .610152, 1.14291, 2.2072, 4.34568" ,\
 ".305532, .700173, 1.17722, 2.21651, 4.35869" ,\
 ".469617, .894771, 1.35714, 2.32296, 4.38333" ,\
 ".743882, 1.26289, 1.70498, 2.63657, 4.52904" ,\
 "1.20801, 1.84408, 2.32966, 3.29812, 5.05465" );}

cell_fall(lu5x5) {
  values( \
 ".185292, .602238, 1.07644, 2.02688, 3.91734" ,\
 ".432254, .852127, 1.3175, 2.2643, 4.14154" ,\
 ".683672, 1.15366, 1.61688, 2.5389, 4.4227" ,\
 "1.14297, 1.72898, 2.2261, 3.14691, 4.99237" ,\
 "1.89231, 2.63401, 3.23074, 4.2367, 6.03552" );}

fall_transition(lu5x5) {
  values( \
 ".167624, .72171, 1.355, 2.62284, 5.13434" ,\
 ".309461, .78558, 1.37218, 2.62191, 5.10869" ,\
 ".474777, .961437, 1.49416, 2.65833, 5.13505" ,\
 ".780225, 1.2901, 1.82791, 2.9399, 5.23852" ,\
 "1.24947, 1.86456, 2.44295, 3.52342, 5.67453" );}
```

Figure 8.30: Results of running **test.ocn** on an inverter and printing as **.lib** tables