EE456 Lab Tutorial 2

Cadence HSPICE Simulation Introduction

1.0 Introduction

The purpose of the second lab tutorial is to help you in simulating your inverter design that you designed in the first lab tutorial. You will create a hierarchical design, i.e., use the inverter symbol that you generated in another schematic in this design. You will also use HSPICE to simulate your design and evaluate its performance by examining the simulation results.

Upon completion of this tutorial, you should be able to:

- Do a hierarchical design using cells that you have designed
- Use buses and tap wires from buses.
- Simulate your schematic using HSPICE
- Examine the results of your HSPICE simulation

1.1 Getting started

- Copy the `hspice.include` file in the `/package/cae/cadence/cells/tsmc025` directory to your home directory.

- Use any text editor to view the contents of the file. The file includes various process corners that will be used for simulation. To select a particular process corner, uncomment (remove the `*`) the .LIB line corresponding to the supply voltage and process corner you wish to use. For this tutorial we will be using the 2.5V *typical model* process corner.

- A process corner is a simulation environment that represents some set of operating conditions that the designer is concerned about.

2.0 Create a New Schematic to be Simulated

- Under the same library (tutorial), create a new cellview named *InverterTest*.

- Use the *Add Instance* command to place your inverter symbol in the cellview. It should be found under the *tutorial* library.

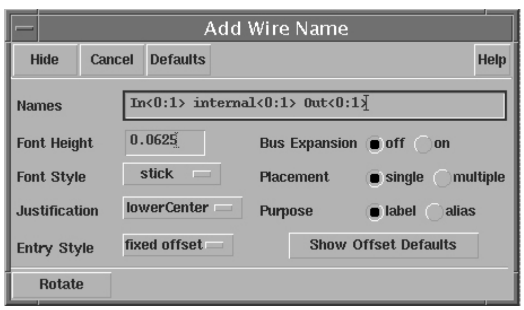
- Use the *Add Instance* command to place two capacitors in the cellview from the *analogLib* library and give them a value of `10fF (when you type in the value in the property window, just type 10f).`

- For this tutorial, we will create two bit wide buses for input, internal nodes and output. To add a bus, click the *Wire (wide)* icon. Draw the bus as you would draw any normal (narrow) wire.

- After creating the bus, the bus must be named. The same command to add wire names for narrow wires is used, however the syntax for naming the bus is slightly different.

- Bus names is in the form `Name<a:b>`, where `a` and `b` denotes the range of bits of the bus. Several two bit buses that will be created in this tutorial include `In<0:1>`, `internal<0:1>` and `Out<0:1>`.

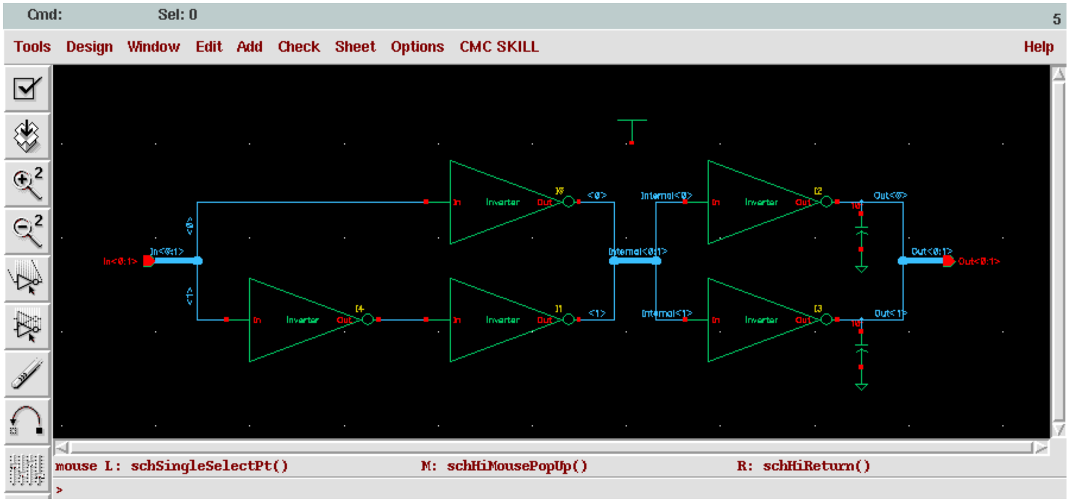
- Similar to the previous tutorial, it is possible to add several names to different buses at one time. To do this, make sure the *Bus Expansion* button is off and simply type the names of the buses separated by a space in the *Names* field. The names will be added in order when you click on the various nets. As an exercise, create and name the buses similar to the ones in the final schematic.



- To draw or tap wires to and from the buses, draw wires from the buses and name them correspondingly to their bits. To name the individual bit lines, type in the bus name (e.g. `In<0:1>` or simply `<0:1>`) and turn on *Bus Expansion* in the *Add Wire Name* window. The *Bus Expansion* button is used to extract individual bit names from the bus. When you start placing the names on the wires, you will notice that the first name will be `In<0>` (or `<0>`), the next will be `In<1>` (or `<1>`), and so on.

- Setting the *Placement* button to multiple allows you to place an array of names, rather than one at a time.

- Complete, check and save the schematic so that it looks like the schematic shown below. Don't forget to add the pins `In<0:1>` and `Out<0:1>`. Note that a Vdd instance is added (from the *analogLib* library) without any connection to the schematic. This is required to specify a global Vdd source for simulation.



3.0 Hierarchy Editing

- Suppose that you wanted to explore the schematic of the inverter. You can traverse a design hierarchy to view or edit your inverter design by selecting the inverter, then select **Design -> Hierarchy -> Descend Edit / Read**. Select *schematic* for the *View Name*. Click **OK** in the new window that appears.

- Now the cellview should change to the schematic view of your inverter, showing the transistors and the connections. If there are more levels to descend, you can continue to do so from here.

- Note that if you make and save any changes to the schematic in the lower levels, the whole design (top level) will be changed, including other designs that use this symbol.

- To return up one level or to the top view, use **Design -> Hierarchy -> Return / Return To Top**.

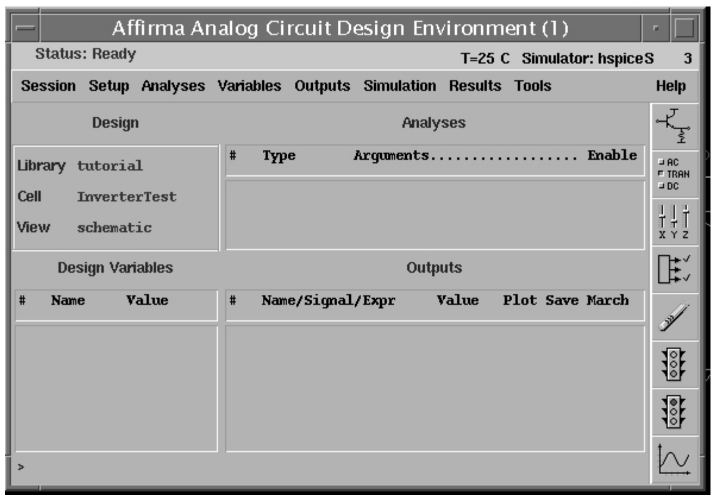
4.0 Affirma Analog Circuit Design Environment

The Affirma Analog Circuit Design Environment is where all the simulations and the netlist extractions of your inverter design can be done. Here you can set up the type of simulation to be run with the appropriate input signals, run the simulation and display the results in a graphical viewer.

4.1 Launch Affirma Analog Circuit Design Environment

- If you have not opened your *InverterTest* schematic cellview, open it by selecting **File -› Open** from the drop down menu in the CIW. Select the *tutorial* library and make sure that you select the schematic view.

- In the schematic window, select **Tools -› Analog Environment** from the drop down menu A new window named *Affirma Analog Circuit Design Environment* appears.



- The correct design (Library, Cell, View) to be simulated should be displayed in the window under the *Design* box.

4.2 HSPICE Simulation Setup

- To setup a HSPICE simulation, select **Setup -› Simulator/Directory/Host** from the drop down menu. A new window named *Choosing Simulator/Directory/Host* appears. Select *hspiceS* for the *Simulator*.

- The *Project Directory* field specifies where the simulation files and results are to be stored. If you need to store results in another directory or path, you can change it here. Note that results from subsequent simulation runs will overwrite the previous results unless they are to be stored in a different location. For this tutorial, just use the default path. Click **OK** to finish.

- The schematic window will refresh and the schematic should reappear in a few seconds. Now in the upper right corner of the *Affirma Analog Circuit Design Environment* window, the simulator should be *hspiceS*.

- To include the HSPICE model file that you have copied into your home directory earlier, select **Setup -› Environment** and type `~/hspice.include` in the *Include File* field. Make sure that *hspice* is selected for *Include/Stimulus File Syntax*. Click **OK** to finish.

- To setup the type of analysis to be performed (transient, DC, AC etc). Select **Analyses -› Choose** or click the *Choose Analyses* icon on the right. A new window appears.

- For this tutorial, we will perform a transient analysis that starts from 0ns to 30ns in 1ns steps (Type in '30n' instead of '30ns'). Select *tran* for *Analysis* and enter the corresponding values. Click **OK** when done.

- Now in the *Affirma Analog Circuit Design Environment* window, the *Analyses* field should now display the type of analysis to be performed with the corresponding arguments.

4.3 Stimuli Setup

- To setup stimulus or input signals, select **Setup -› Stimulus -› Edit Analog**. In the new window that appears, click on ***graphical*** and then click OK.

- A new window named *Setup Analog Stimuli* appears. Click on *Inputs* for *Stimulus Type* and select an input signal from the list below. In this tutorial, there are two inputs `In<1>` and `In<0>`. Click on either one to start entering the stimulus.

- For this tutorial we will set up an input waveform for both inputs with the following parameters:

  Function: Pulse
  Type: Voltage
  Voltage 1: 0.0
  Voltage 2: 2.5
  Delay time: 1n
  Rise time: 1n
  Fall time: 1n
  Pulse Width: 10n
  Period: 20n

  After you have entered all the parameters, check on the '**enabled'** option to turn on the input signal. Click on the **Change** or **Apply** button to set the values. Do not click **OK** yet.

- Now that you have defined your input waveform, you need to specify the voltages of the global sources (Vdd). Click on *Global Sources* and a new list should appear.

Select *dc* for *Function*, *Voltage* for *Type* and enter `2.5v` for *DC voltage*. Check on the '**enabled'** option to turn on the voltage source and click **Change** or **Apply** to set the values. Click **OK** to finish setting up stimuli.

4.4 Simulation Output Data

- The data obtained from the simulation can be saved, plotted or marched. The saved data will be written to disk in the *Project Directory* specified above. The plotted set of data can be plotted in a Waveform window. The marched set of data will be plotted in the Marching Waveform window during simulation.

- To add nodes and terminals to the saved, plotted, or marched set, select **Outputs -› To Be … -› Select On Schematic**. In the schematic window, choose one or more nodes or terminals. When selecting, click on the square pin symbols to choose currents and click on wires to choose voltages. For this tutorial, select the input and output wires (bus). Note that if you select a bus, three nodes will appear e.g. `In<0:1>`, `In<0>` and `In<1>`. For HSPICE simulations, it is best to delete the `In<0:1>` node from the list because HSPICE does not support bus names.

- Although it is not recommended, it is possible to save all node voltages and terminal currents of the simulation. To do so, select **Outputs -› Save all**. In the window that appears, check the box next to *Select all node voltages* and click **OK**. Note that this may create unnecessary large data files, especially for a large design. To undo the save all function, choose **Outputs -› Save all** and uncheck the box next to *Select all node voltages*.

- To delete a node or terminal from the saved, plotted, or marched set, select **Outputs -› Setup**. In the window that appears, double-click on the node or terminal in the *Table Of Outputs* list box and check the appropriate *Will Be* boxes. Click **Change** to update the changes. Click **OK** when done.

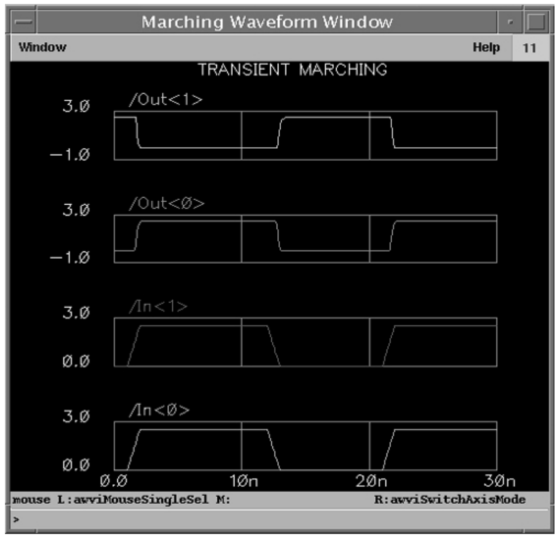
- To delete a node from all three sets, highlight the node in the simulation window and choose **Outputs -› Setup** or click the *Delete* icon on the right.

- To run the simulation, select **Simulation -› Run** or click the *Run Simulation* icon on the right. The CIW will display the status and the results of the simulation. Errors or warnings will be displayed here if found.

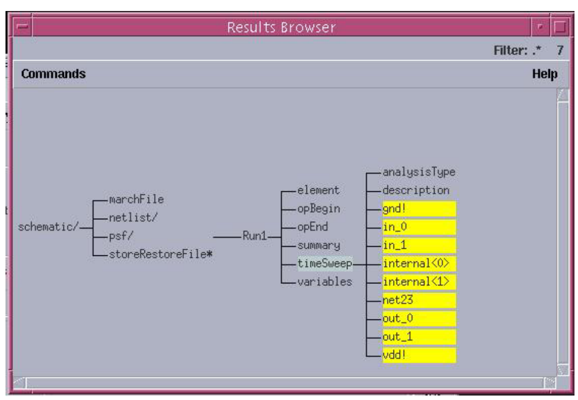
- When running HSPICE simulations, you might see the warning saying that it is unable to open nch.m or pch.m. This means that Cadence is trying to look for the n and p transistor model files in the default location and couldn't find them. This is fine as long as you have specified the *hspice.include* file, which includes the path of the transistor model files. The simulation should still run without problems.

4.5 Viewing Simulation Results

- During a simulation, the marched results will be shown progressively in a *Marching Waveform* window that appears automatically.



- After a successful simulation, the plotted results can be displayed by selecting **Tools -› Results Browser**. A new window appears. Make sure the specified project directory is correct (corresponds to the directory path specified at simulation setup). Click **OK** to continue.

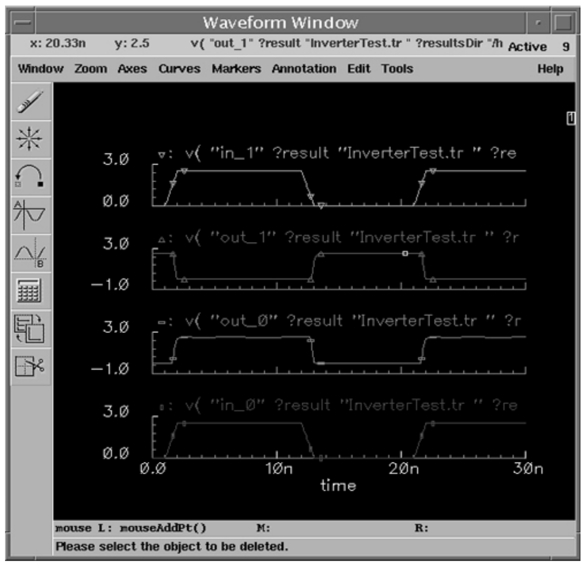


- Since we would like to display the transient waveform, **right click** on */psf* -› *Run1* -› *timeSweep->in_0* .

- The waveform of the *In_0* signal is displayed in the *Waveform Window* that appears.

- To display the waveform of the *Out* signal, return to the Results Browser and right click on */psf* -› *Run1->timeSweep->out_**. The waveform will be plotted in the *Waveform Window*.

- The two waveforms are displayed on the same axis and should be inverted with respect to each other. To split them into separate axis, click on the *Switch Axis Mode* icon on the left. Clicking on the icon again combines the waveforms into the same axis.



- Note that not all of the functions under **Results** and **Tools** work with the HSPICE interface.

4.6 Saving and loading States

- Saving states allow you to save and restore all or part of the simulation setup so that lot of steps can be saved when setting up the simulation (for a similar simulation run). To do this, select **Session -› Save State**.

- In the new window that appears, choose a name for the simulation setup to be saved, and select what items to save. Click **OK** to save.

- Close **Affirma Analog Circuit Design Environment** and restart it by selecting **Tools -› Analog Environment** menu in the schematic window. To retrieve and restore previously saved setup, **select the correct simulator(hspiceS) first by menu Setup -› Simulator/Directory/Host**, then select **Session -› Load State**. A new window appears. Choose the desired value in the *Library*, *Cell* and *Simulator* fields.

- *State Name* specifies all the saved states for the cell and simulator combination that you specified. Choose the *What to Load* options you need and click **OK**.

APPENDIX. Other Useful CAD Tools

A.1.0 Netlist Extraction & running HSPICE in stand alone mode

A netlist file describes the system or circuit to be simulated. The file specifies the components or instances in the circuit, how is the circuit connected together, what type of simulation is to be run, what inputs are supplied and what outputs are to be obtained.

- To extract a HSPICE format netlist, the procedures are identical to the ones required to setup a HSPICE simulation, except that instead of running the simulation, a netlist is extracted. This is because although it is transparent to the user, a netlist is generated before the HSPICE simulation is run under the *Affirma Analog Circuit Design Environment*.

- If a successful simulation run has been done, the netlist is already created and can be readily viewed by selecting **Simulation -› Netlist -› Display Final**.

- If the HSPICE simulation has not been set up or run for the inverter schematic, use the *Affirma Analog Circuit Design Environment* to setup the HSPICE simulation as done in the previous section, but do not run the simulation. If you have saved states, you could use them here to simplify the set up process.

- Select **Simulation -› Netlist -› Create Final**. The CIW will display the status and the results of the netlist extraction process. Errors or warnings will be displayed here if found.

- If the netlist extraction is successful, a new window displaying the extracted netlist appears. The netlist itself is named `hspiceFinal` and is stored in the *project directory* specified in the *Affirma Analog Circuit Design Environment* (~/simulation/InverterTest/hspiceS/schematic/netlist). You can use any text editor to view or edit the netlist.

- If you are familiar with HSPICE/SPICE, you should be able to recognize all of the statements and elements in the netlist.

  The SPICE manual is available at:

  `/package/cae/hspice/2000.2/docs/2000.2.html/hspice.html`

- Before running HSPICE in command line, open the input file (`hspiceFinal`) with a text editor and do the following first.

  - Find the lines that include following and delete it.

```
.OPTION INGOLD=2 ARTIST=2 PSF=2
+         PROBE=0
```

- Add following line .

```
.OPTIONS POST
```

(If you don't like to modify the netlist after generation, you can choose *Simulation - Options – Analog* menu in Affirma Environment window and set POST option to '1' before the generation of the final netlist. Then, you can use the hspiceFinal in your HSPICE simulation without modification. Note that it's only for the case of stand-alone HSPICE simulation)

- Now that you have obtained a netlist, you can run HSPICE by typing the following command at the UNIX prompt,

  ```
  hspice inputfilename
  ```

  where inputfile is the name of your input file. The output will be displayed on the screen. You can also store the output of HSPICE in an output file, as follows,

  ```
  hspice inputfile > outputfile
  ```

  the output file will be a text file and you can use any text editor to view it. It may be useful to check the output file when your HSPICE simulation is not successful.

NOTE The following table lists the basic statements and elements of an HSPICE input file. You can run more

Statement or element	Definition
title	***first line is the input file titl (considered as comment)***
* or $	designates comments to describe the circuit
.OPTIONS	sets conditions for simulation
.DC/ .AC/ .TRAN	analysis statement
.PRINT/.PLOT/.GRAPH/.PROBE	statements to set print, plot and graph variables
.IC/.NODESET	set initial state; can also be put in subcircuits
source (I or V)	set input stimuli
netlist	circuit description
.LIB	library
.INCLUDE	general include files
.MODEL	model description
.PARAM	parameter definition
.ALTER	sequence for in-line case analysis
.END	required statement to terminate the simulation
.MEASURE	calculate time, power and etc.

EXAMPLE Here is an example of how a HSPICE netlist would look like (Try to understand each statement. *.measure*, *.param*, *.alter* are especially important because it would be very useful in later simulation of your own circuit)

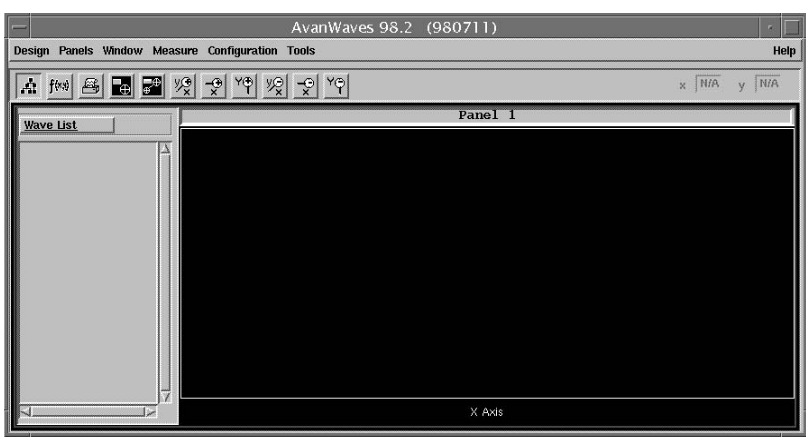
```
* INVERTER

*-------------------------------------
* Include model file
*-------------------------------------

.include ~/hspice.include

*------------------------------------------------
* Param's
* Now several parameters are declared
* You can use these parameters later in this file
*-------------------------------------------------
.param wn=300n wp=600n
.param l=240n
.param vref=1.25

*-------------------------------------------
* NETLIST
* The wedth and length of the tr's are not
* numbers but the parameters
*-------------------------------------------

M1 OUT IN VDD! VDD!  PCH  L="l" W="wp"
+AD=503.999999287158E-15 AS=521.999977964871E-15
PD=2.27999998969608E-6
+PS=2.34000003729307E-6 NRD=+1.66666661E+00 NRS=+1.66666661E+00
M=1.0
M3 OUT IN 0 0  NCH  L="l" W="wn"
+AD=459.000025487821E-15 AS=477.000004165534E-15
PD=2.58000000030734E-6
+PS=2.64000004790432E-6 NRD=+3.33333322E+00 NRS=+3.33333322E+00
M=1.0
CL OUT 0 40f

*-------------------------------------
* OPTIONS
*-------------------------------------
.OPTIONS POST
.PRINT TRAN V(IN) V(OUT)
.TRAN  1.00000E-09 3.00000E-08 START=      0.
.TEMP  25.0000

*-------------------------------------
* INPUT
*-------------------------------------
.global vdd!
vvdd! vdd! 0 DC=2.5
vIN IN 0 pulse 0.0 2.5 1n 1n 1n 5n 10n
```

```
*------------------------------------
* MEASURE
*------------------------------------
.measure tplh trig v(in) val=vref fall=2
+ targ v(out) val=vref rise==2
.measure a_power avg power from=0ns to=60ns

*----------------------------------------------
* ALTER.
* HSPICE runs several tims according to
* the number of .alter statement.
* In this example, HSPICE runs 5 times
* ( 1 for the initial value & 4 for the .alter's )
* The measurement results for each simulations
* are stored in .mt* files.
*-----------------------------------------------
.alter
.param wn=300n wp=700n

.alter
.param wn=300n wp=800n

.alter
.param wn=600n wp=900n

.alter
.param wn=600n wp=1200n

.end
```

A.2.0 Awaves

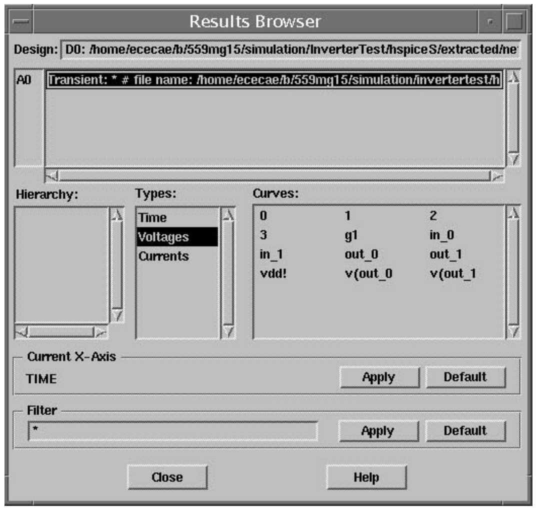
Awaves is a graphical waveform viewer. In previous section, you ran HSPICE in command line and the simulation results can be seen by using Awaves.

- To run Awaves, enter `awaves` in a UNIX window. When the *AvanWaves* window appears. Select **Design -› Open** from the drop down menu.



- When the *Open Design* box appears, select the .tr0 file. If you cannot see any file, select **Filter** menu and check **All Files** check box.

- The *Results Browser* will appear. Select the types (voltages or currents) and waveforms you want to see.



A.3.0 Nanosim

Nanosim is a transistor-level power simulator and analyzer for CMOS and BiCMOS circuit designs. As more components are integrated into smaller silicon chips, power consumption becomes a major concern. Detailed current information about each circuit component is critically important to the designer. Designs must be optimized for minimum power consumption.

An advantage using Nanosim over HSPICE is that Nanosim can run transistor-level simulations (with HSPICE-like accuracy) with a run time 10 to 1000 times faster than HSPICE. This would be very helpful especially when the design complexity increases as these simulations may take hours.

Note that Nanosim is not set up to run under the *Affirma Analog Circuit Design Environment*. Therefore it is necessary to extract a netlist from your schematic under the *Affirma Analog Circuit Design Environment* and execute Nanosim from UNIX (refer to section 4.7). The stimuli and output set up in the *Affirma Analog Circuit Design Environment* will work in Nanosim, however, it might be easier and more efficient to use the Nanosim config file commands and input vector files to set up the inputs and outputs for designs that deal with buses.

A.3.1 Nanosim Simulation

Nanosim accepts several formats of netlists, including HSPICE, however, the netlist filename will need to have a `.sp` suffix for Nanosim to recognize it as an HSPICE netlist.

To save the trouble of renaming the `hspiceFinal` file created by the netlister every time, create a soft link by using the following command:

```
ln -s hspiceFinal nanosim.sp
```

- To run Nanosim, the command is `nanosim`, but you should find out more from the online documentation first to find out what command line arguments are needed.

- The **online documentation** for `nanosim` can be viewed by entering `sold` in any UNIX window. As well as the basic explanation about Nanosim simulation, you can also find information about how to provide stimuli in the section 'Nanosim Library' -> 'Circuit Simulation and Analysis Tools Reference Guide'.

- The Nanosim command line used for this tutoral would be:

  `nanosim -n` *`nanosim.sp`* `-c` *`nanosim.cfg`* `-z` *`tt`*

  The `-n` command line option is the only required option to specify a netlist file.

  The `-c` option is used to specify a configuration file. See the next section for further details on how to create the configuration file.

  The `-z` option is to tell Nanosim to create technology files automatically which start with the name (tt) given. The reason for the name `tt` used is because it corresponds to the name of the *typical* process corner selected in the `hspice.include` file. For subsequent runs, the `-z` option can be replaced by a `-p` option to save time and to re-use the technology files generated by `-z`.

A.3.2 Nanosim Configuration

- The purpose of the configuration file is to add additional Nanosim commands to control the simulation. One of the more relevant uses is to specify the types of output that is to be reported or saved to a file (`nanosim.out`). Without specifying any output commands, no output will be reported or saved. Other uses include the option to perform power analysis of the circuit and to control the accuracy and speed of the simulation.

- To create a configuration file, use any text editor to create a text file with any name. The commands are line oriented and can be continued on the next line with the use of the a space and a back slash (\) at the end of a line. Comments are preceded by a semicolon (`;`).

- The contents of the configuration file will depend on the type of simulation and types of results that one would like to obtain. A typical configuration file would contain the following commands:

```
print_node_v IN* OUT_0 OUT_1
print_node_logic IN* OUT_0 OUT_1
report_block_powr total track_power=1 *
```

Node names can be replaced by the wildkey "*" to specify all nodes are to be printed or reported, however, this will cause lots of data to be generated unnecessarily.

The **print_node_v** command prints the voltage waveform information for the given nodes to the *.out* file.

The **print_node_logic** command tracks the logic values of the specified nodes and records their changes in the *.out* file.

The **report_block_powr** command creates a summary report of the power consumption of a specified circuit block.

The **set_sim_mode** *rc* | *ud* | *pwl* controls the speed and accuracy of the simulation. RC and UD modes use switch-level simulation techniques and produce faster average power estimation with reasonable accuracy, as compared to the default PWL mode.

Please refer to the online documentation for more details of the above commands.

A.3.3 *EXAMPLE* Advanced nanosim simulation example using input vector file.

This example shows how to use a vector file in your simulation. Note that there is additional command line option *'-n nanosim.cmd'*. In this example, the simulation is using an input vector file (*nanosim.vec*) which contains input values for input ports A and B. For more information, refer to the online manual **Nanosim Library -> Circuit Simulation and Analysis Tools Reference Guide, chapter 5** (command is 'sold').

- **Command**

nanosim -nspice xor.sp -n nanosim.cmd -c nanosim.cfg -o output -t 500 -z tt

- **netlist**

xor.sp

```
This is the netlist file
* CAUTION! Now this file is using upper case for VDD
* to match with final.sp
.global VDD!

* Actual circuit is not shown here

.include final.sp

.include ~/hspice.include
.temp 25.0
VVDD! VDD! 0 DC=2.5
.end
```

- **.cmd file example**

nanosim.cmd

```
(is=nsvt) (en=nanosim.vec) (ot= A,B);
```

- **.cfg file example**

nanosim.cfg

```
print_node_v A* B* O*
print_node_logic A* B* O*
report_block_powr total track_power=1 *
```

- **.vec file example**

nanosim.vec

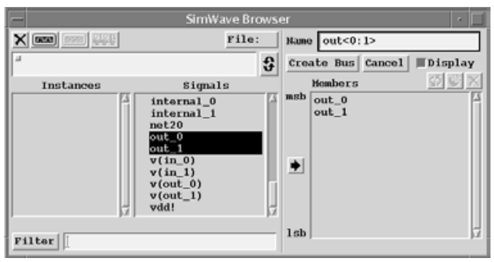
```
;input vector - this is comment line
    radix 11
    io ii
    period 5.0
    fall 0.2
    rise 0.2
    ;A B
    1 0
    1 0
    0 1
    . .
    . .
```

A.4.0 SimWave

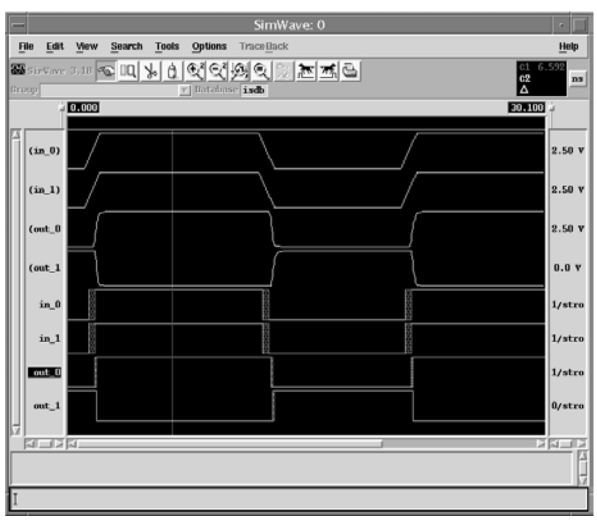
- SimWave is used to view the waveform results of a Nanosim simulation. It can display both digital and analog signals from the simulation.

- To run SimWave, enter `wd` in a terminal window command prompt. After the SimWave window is loaded, select **File -› Database -› Load** from the drop down menu. Select *epic* for *format* and enter the filename by typing or clicking the icon on the right to browse for the file (`nanosim.out`). Click **OK** when done.

- Select **Edit -› Add Signals**. A new window named *SimWave Browser* appears. Click on the signals that you want to be displayed in the list under *Signals*. Use the `ctrl` key to select multiple signals. For this tutorial, select `v(in_0)`, `v(in_1)`, `v(out_0)`, `v(out_1)`, `out_0` and `out_1`. When ready, click the *Display Signals* icon near the top. If you are not sure which icon is the right one, move the cursor over the icon and the name will be displayed.



- It is also possible to display signals as a bus. In the *SimWave Browser*, click on the *Create Bus* icon. Select `out_0` and `out_1` (these are logic values) and click on the right arrow. Note that only signals that are logic values can be displayed as buses. Give the bus a name `out<0:1>` (make sure that the order is correct – msb on the top and the lsb at the bottom) and click the **Create Bus** button. Now you should have several waveforms displayed in the *SimWave* window.



- To remove signals from the window, right click (hold) on the signal name and select delete, or first select the signal name, then select **Edit -> Delete**. To move signals around, use the "cut and paste" method.

- There are two cursors (vertical markers) C1 and C2 that can be used to measure the time between two events, to mark certain times, and to zoom to a specific time interval. The time of these two cursors as well as the difference between them is shown on the upper right corner. To set C1, click and drag the left mouse button. To set C2, use the right mouse button.

A.5.0 Pathmill

The PathMill timing analysis tool is a full-chip, transistor-level critical path finder and timing verifier for custom circuit designs. It employs static timing analysis techniques to enable you to detect and correct timing violations. It searches all possible paths in your design, calculates delay, and checks timing requirements at all the timing nodes.

In the following, the command line format, netlist example and configuration file examples are shown.

For more information,
- Type 'sold' in command window and look into Pathmill manual
- Tutorial section can be found in : Pathmill manual -> User Guide

A.5.1 Command

(Input : hspiceFinal, Config : pathmill.cfg, Result : output.out)

```
pathmill -nspice hspiceFinal -c pathmill.cfg -o output -z tt
```

A.5.2 netlist example

mult.sp

```
.
.
.
* CAUTION! From the HSPICE netlist, remove the lines for stimulus

* vA_1 A_1 0 pwl 0 0.0 8n 0.0 8.2n 2.5 16n 2.5
* vA_0 A_0 0 pwl 0 2.5 8n 2.5 8.2n 0.0 16n 0.0
.end
```

A.5.3 .cfg file example

pathmill.cfg

```
; The main purposes of this config file are:
; 1. To define source and sink nodes
; 2. To report critical delay paths
; 3. No timing verification
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Define clock nodes

set_vdd VDD!
set_voltage 2.5

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Define source nodes

source_node A_0 B_0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Define sink nodes

sink_node PRODUCT_15

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Eliminate false paths
; Recognize circuit topology
; search_structure mux

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Clock propagation
; set_high EN

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Delay calculation/Wire capacitance
pwl *
node_capacitance PRODUCT_15 50

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Report potential errors
;log_on mux
;print_unset_transistors

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Report delay paths
report_paths critical max min 20
print_spice_paths critical max 1
```