ECE 559 Lab Tutorial 2

HSPICE, Nanosim, and NanoTime Simulation Introduction

Contents

1 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 use HSPICE and Nanosim to simulate your design and evaluate its performance by examining the simulation results.

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

- Simulate your schematic using HSPICE
- Examine the results of your HSPICE simulation
- Extract a netlist from your schematic
- Simulate your schematic using Nanosim
- Examine the results of your Nanosim simulation using CosmosScope

2 Cadence Analog Design Environment

The Analog Design Environment is the place 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.

2.1 Launch Analog Design Environment (ADE)

- 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.

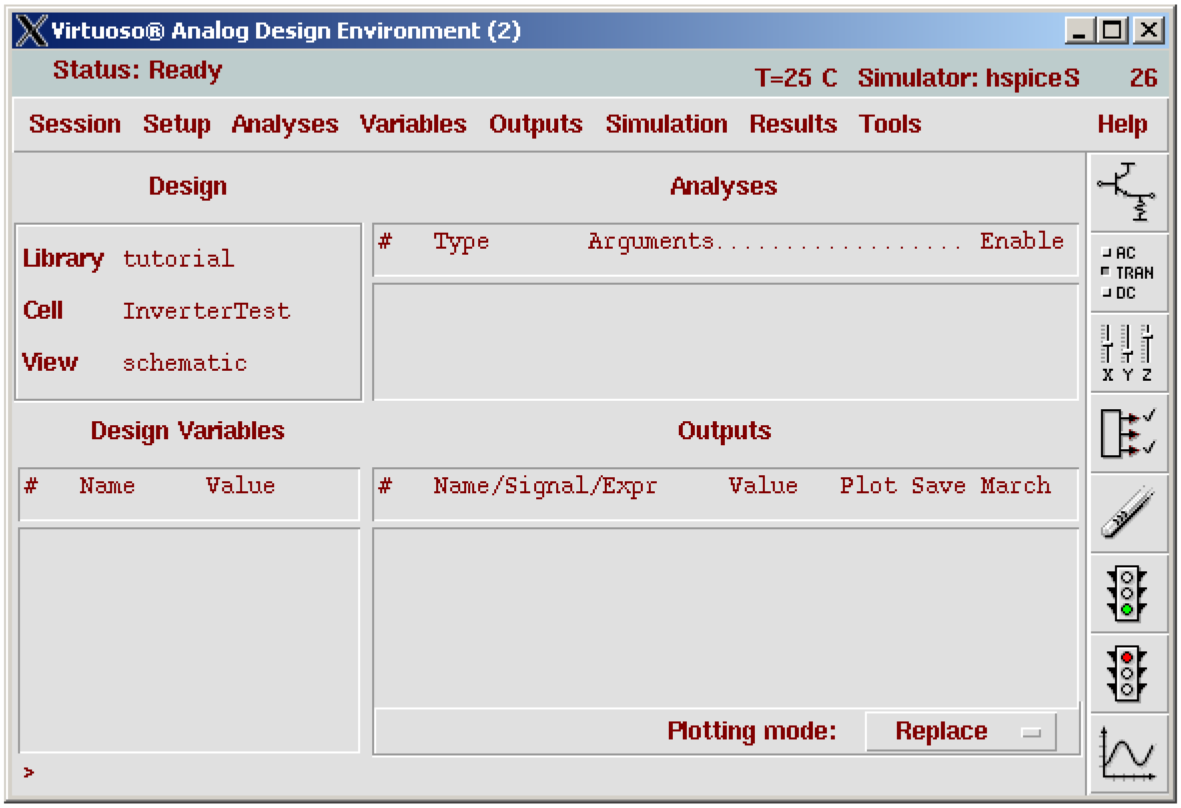
- In the schematic window, select **Tools -› Analog Environment** from the drop-down menu. A new window named *Virtuoso@ Analog Design Environment* appears.

- The correct design (Library, Cell, and View) to be simulated should be displayed in the figure below.

2.2 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.


Virtuoso® Analog Design Environment (2)

Status: Ready T=25 C Simulator: hspiceS 26

Session Setup Analyses Variables Outputs Simulation Results Tools Help

Design

Library tutorial

Cell InverterTest

View schematic

Analyses

```
#   Type        Arguments.................  Enable
```

Design Variables

```
#   Name     Value
```

Outputs

```
#   Name/Signal/Expr    Value   Plot Save March
```

Plotting mode: Replace

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

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

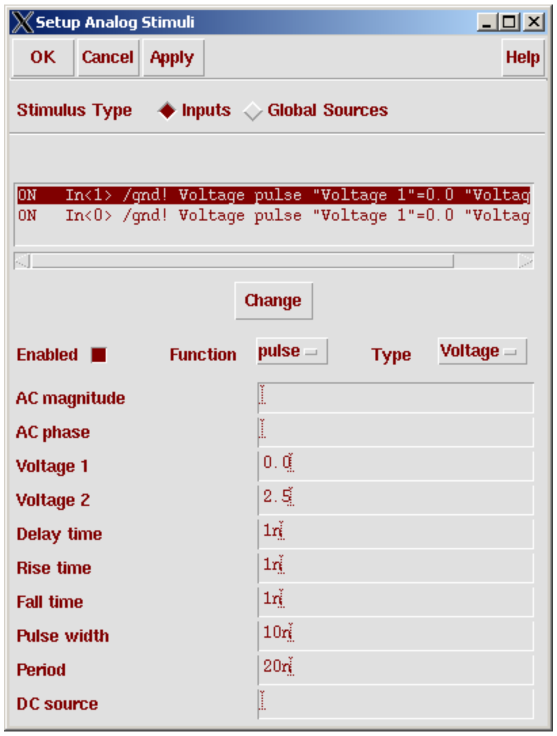
2.3 Stimuli Setup

- 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 `In<0>` to start entering the stimulus.

  - 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.



Stimulus Type [✓ Inputs] Global Sources

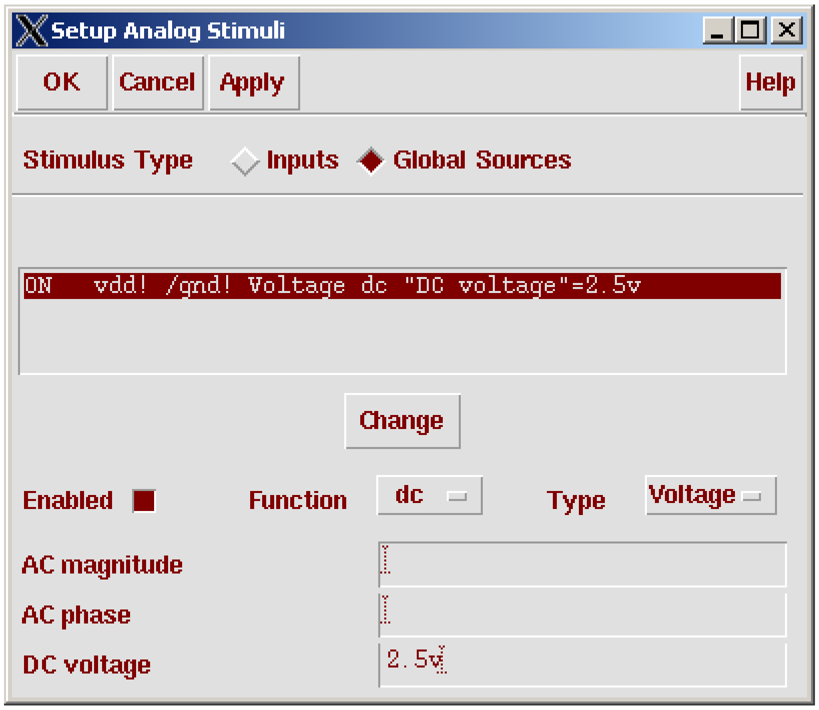
```
ON   In<1> /gnd! Voltage pulse "Voltage 1"=0.0 "Voltag
ON   In<0> /gnd! Voltage pulse "Voltage 1"=0.0 "Voltag
```

Enabled ■ Function pulse Type Voltage

AC magnitude	
AC phase	
Voltage 1	0.0
Voltage 2	2.5
Delay time	1n
Rise time	1n
Fall time	1n
Pulse width	10n
Period	20n
DC source	

- Repeat entering the stimulus for `In<1>.`

- Now, 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.



Stimulus Type [ ] Inputs [✓ Global Sources]

```
ON   vdd! /gnd! Voltage dc "DC voltage"=2.5v
```

Enabled ■ Function dc Type Voltage

AC magnitude

AC phase

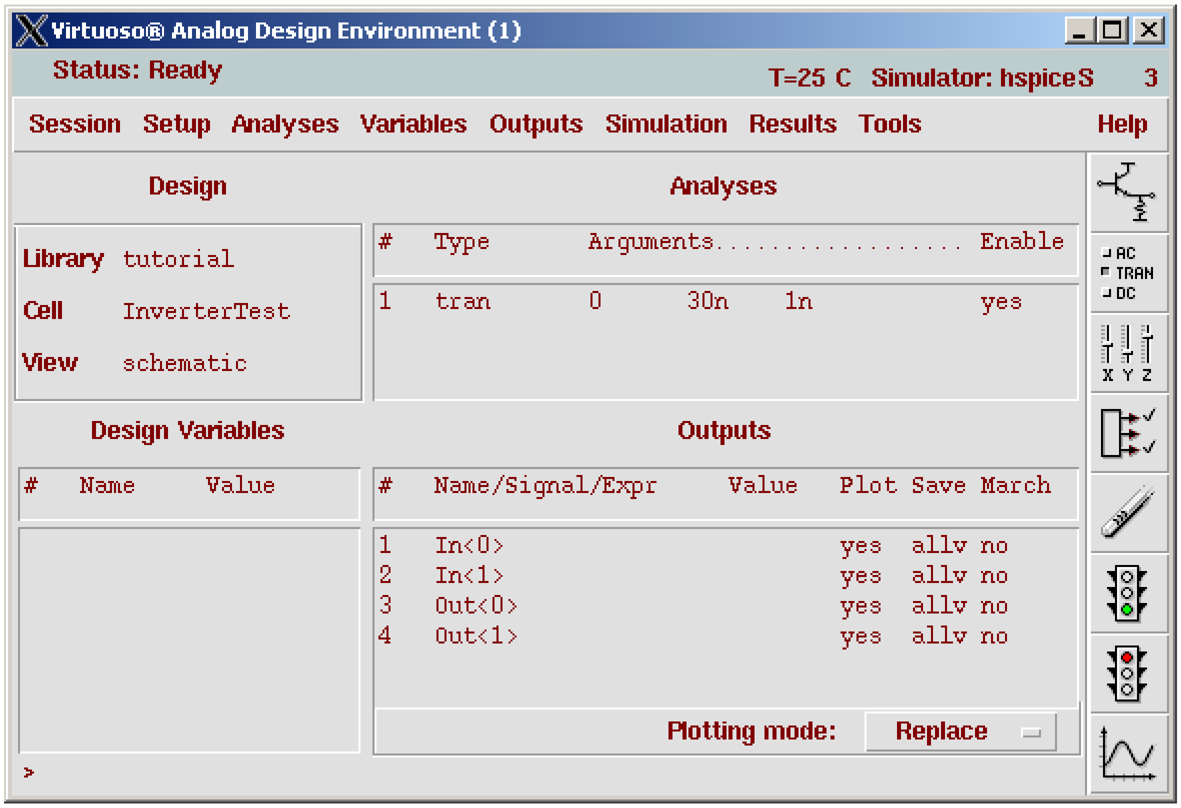
DC voltage 2.5v

2.4 *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>`.

- 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*.


Virtuoso® Analog Design Environment (1)
Session Setup Analyses Variables Outputs Simulation Results Tools Help
AC
TRAN
DC
X Y Z

Status: Ready   T=25 C   Simulator: hspiceS   3

Design

Library	tutorial
Cell	InverterTest
View	schematic

Analyses

#	Type	Arguments			Enable
1	tran	0	30n	1n	yes

Design Variables

#	Name	Value

Outputs

#	Name/Signal/Expr	Value	Plot	Save	March
1	In<0>		yes	allv	no
2	In<1>		yes	allv	no
3	Out<0>		yes	allv	no
4	Out<1>		yes	allv	no

Plotting mode: Replace

- 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.

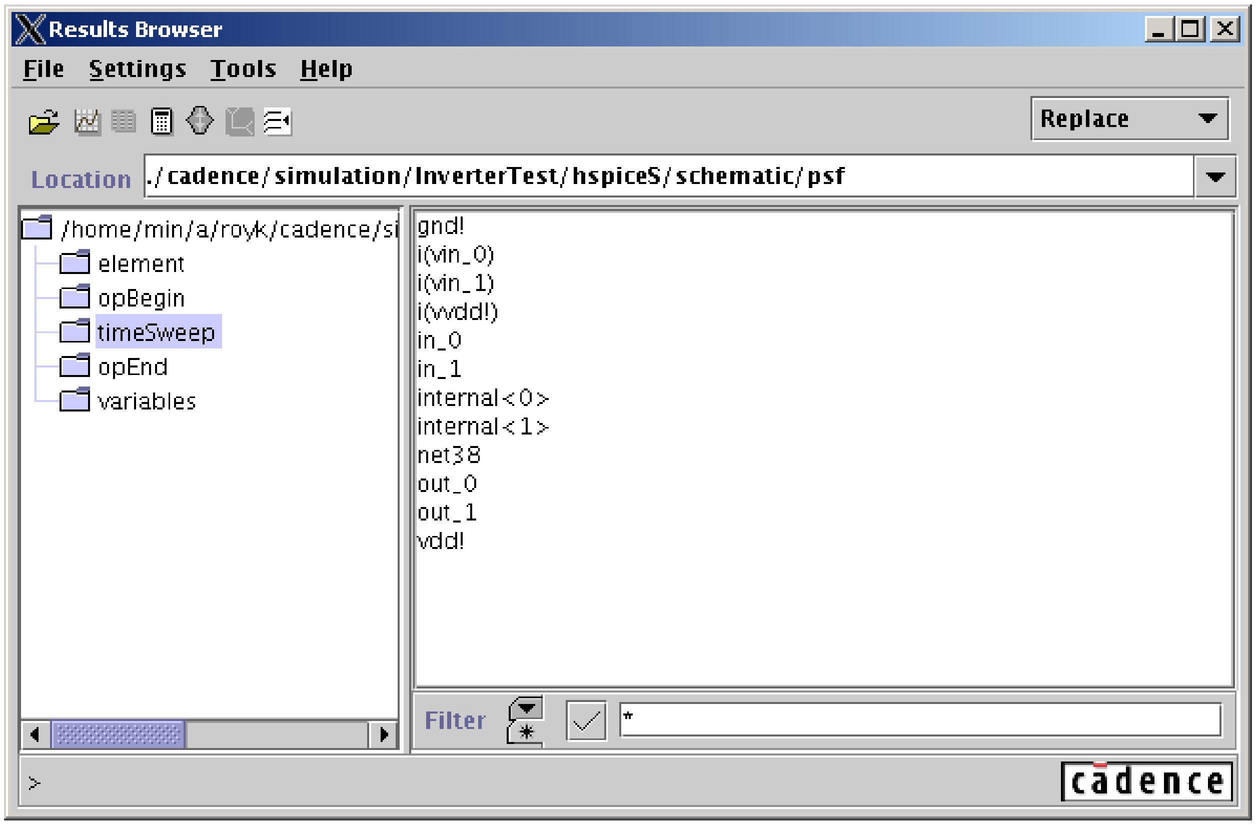
- To delete a node from all three sets, highlight the node in the simulation window and choose **Outputs -› Setup** or click the *Delete* icon on the right.

- 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.

2.5 Viewing Simulation Results

- 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 corresponding to the schematic (eg : *./simulation/InverterTest/hspiceS/schematic/psf*). Double click on **timesweep** in the left subwindow. A list of nodes appears in the right subwindow.

- Since we would like to display the transient waveform, **right click** on *in_1* and select the *New Win* option. The waveform can also be seen by double clicking on the signal.



```
./cadence/simulation/InverterTest/hspiceS/schematic/psf
```

```
/home/min/a/royk/cadence/si
  element
  opBegin
  timeSweep
  opEnd
  variables
```

```
gnd!
i(vin_0)
i(vin_1)
i(vvdd!)
in_0
in_1
internal<0>
internal<1>
net38
out_0
out_1
vdd!
```

- The waveform of the *In_1* signal is displayed in the *Waveform Window* that appears. To display more signals in the same waveform window **right click** on the signal and select the *Append* option. To view the waveform in a new subwindow select the *New Subwin* option.

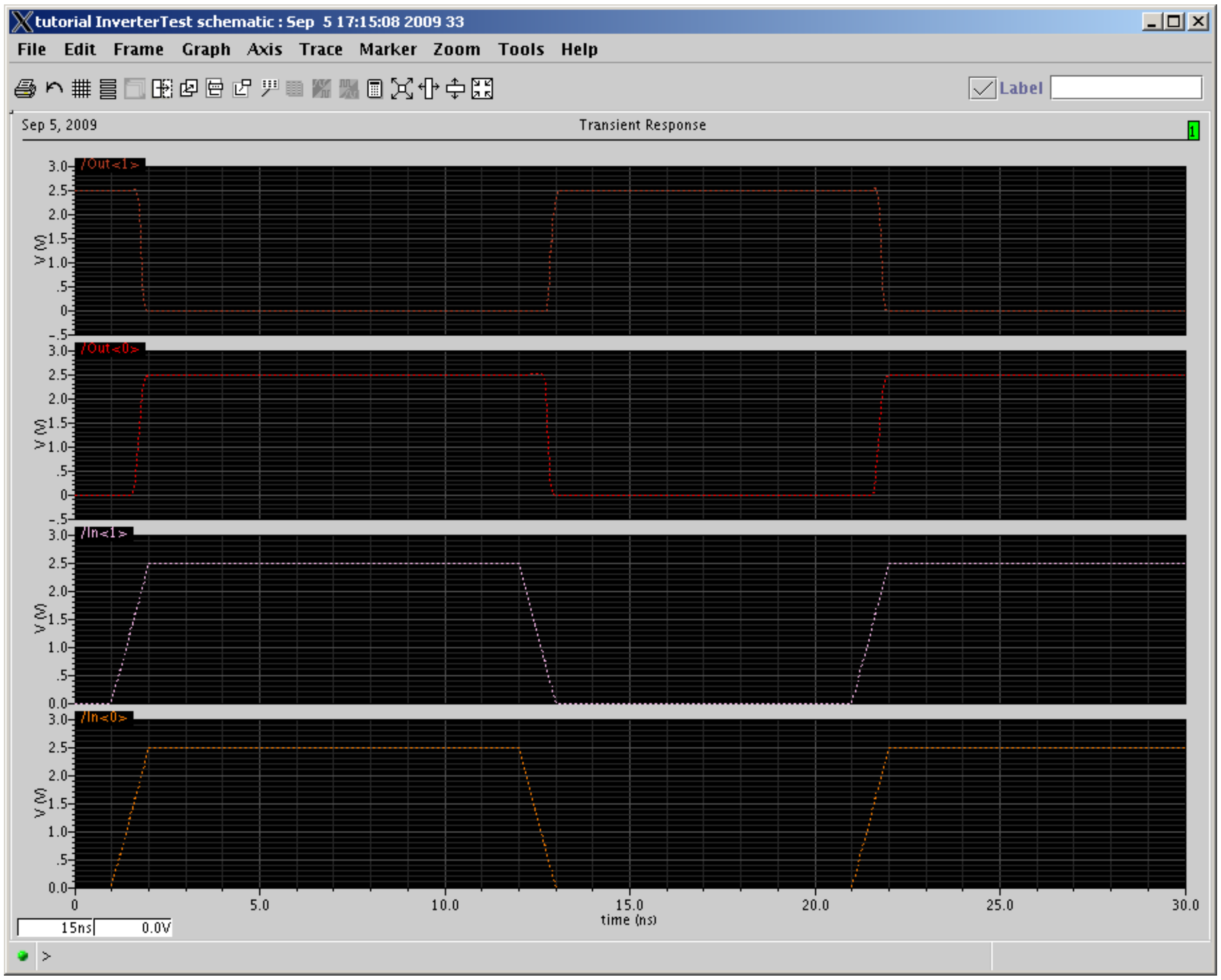
- To display the waveform of the *Out* signal, return to the Results Browser and do the above steps for the output signals. The waveform will be plotted in the *Waveform Window*.

- 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 *Strip Chart Mode* icon on the top. 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.

2.6 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**.


tutorial InverterTest schematic : Sep 5 17:15:08 2009 33
File Edit Frame Graph Axis Trace Marker Zoom Tools Help
Label
Sep 5, 2009
Transient Response
/Out<1>
/Out<0>
/In<1>
/In<0>
V (V)
time (ns)
15ns
0.0V

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

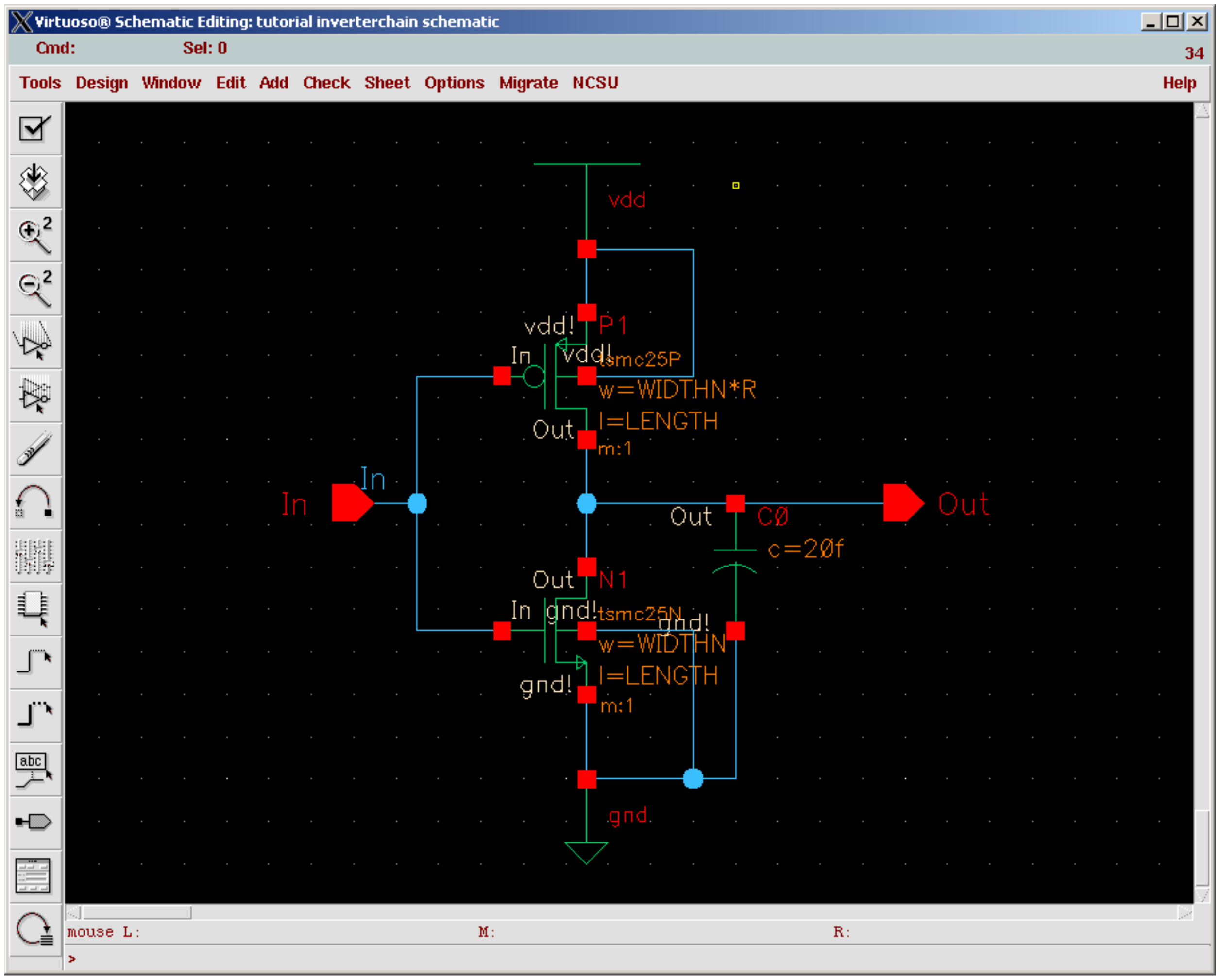
- *State Name* specifies all the saved states for the cell and simulator combination that you specified. Choose the *What to Load* options you need and click **OK**.

3 Variable and Parametric Analysis

3.1 *Create a cellview*

Create a cellview as in the figure with following parameters. Note that the sizes of the transistors are specified using variables.

- PMOS width : WIDTHN*R
- PMOS length : LENGTH
- NMOS width : WIDTHN
- NMOS length : LENGTH



3.2 *Launch Analog Design Environment (ADE)*

- Launch Analog Design Environment **Tools -› Analog Environment**.
- In ADE, **Setup -› Simulator/Directory/Host**: select *hspiceS* for the *Simulator*.

- In ADE, **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. Click on `In` to start entering the stimulus.

  - 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, 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.

- 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.

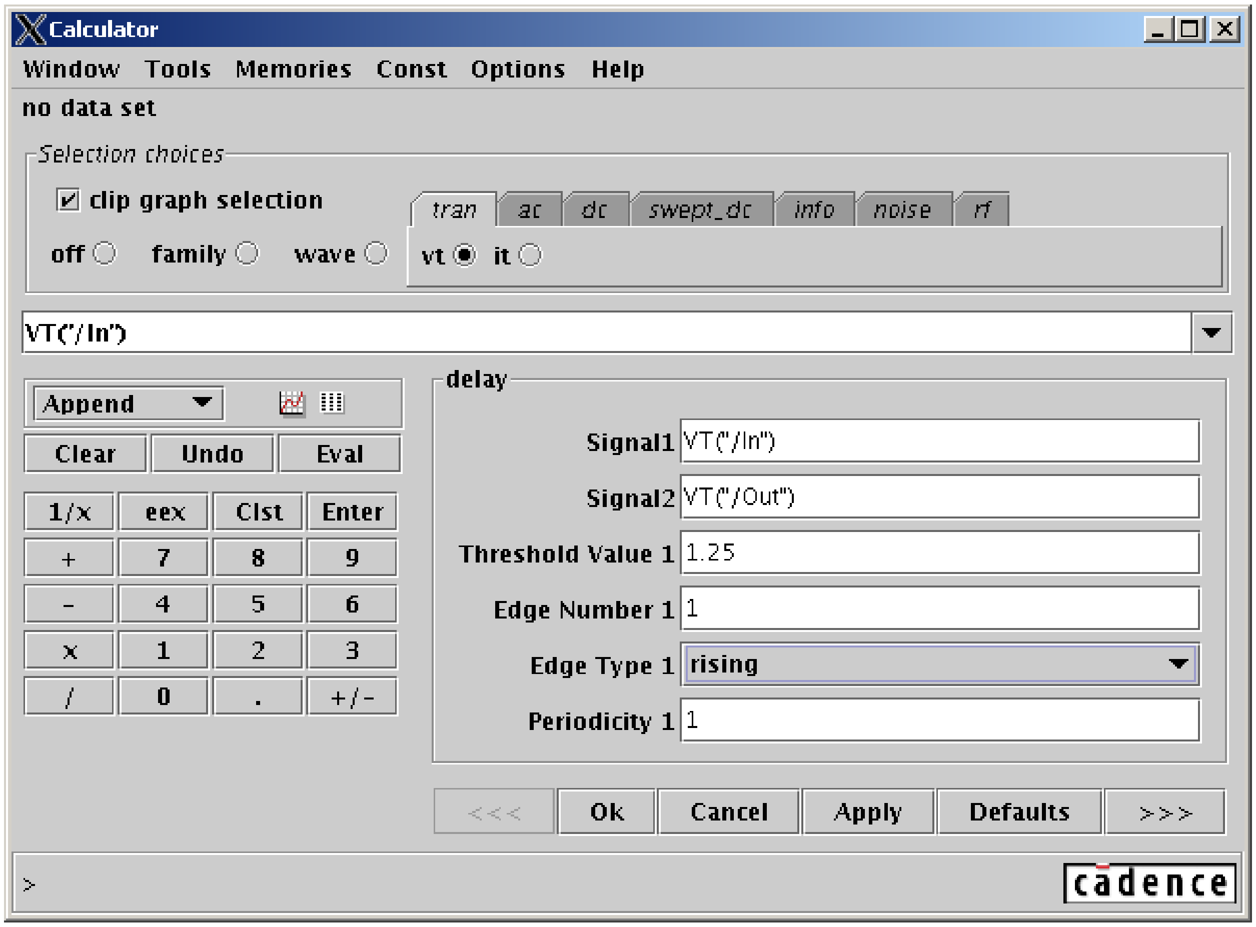
- Now in the *Analog Circuit Design Environment* window, the *Analyses* field should now display the type of analysis to be performed with the corresponding arguments.

- 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.

3.3 Setting up Additional Output for Delay Measurements

- In ADE, **Outputs -› Setup**
- In *Setting Outputs* window, Click **Calculator Open**
- Click *vt* button in *Calculator,*
- Click *out* wire in schematic window,
- Click *in* wire in schematic window,

- Select **delay** from the functions in the sub-window on the lower right corner. The window changes into the figure shown. Please make sure that the **Signal1** is corresponding to *VT("/In")* and **Signal 2** is corresponding to *VT("/Out")*. Change the other values as shown. Then click on the ">>>" button.



VT("/In")

delay

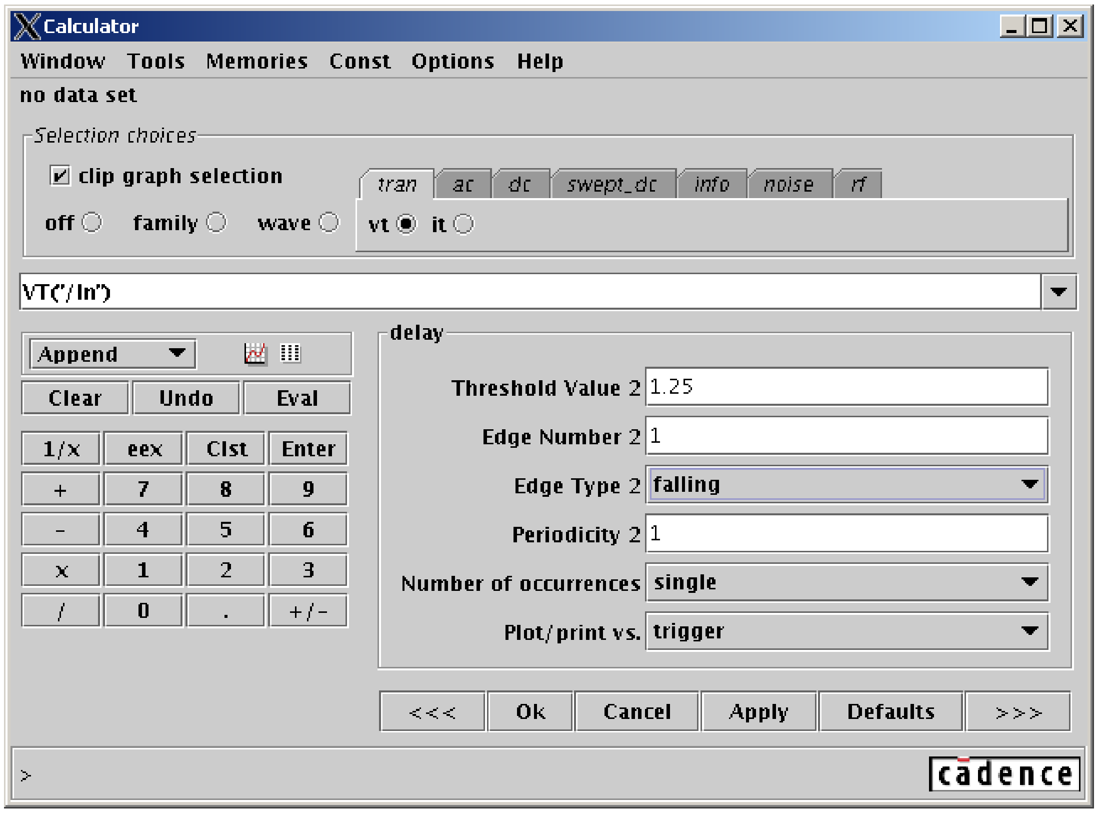
Signal1	VT("/In")
Signal2	VT("/Out")
Threshold Value 1	1.25
Edge Number 1	1
Edge Type 1	rising
Periodicity 1	1

- This will bring the next parameters to be entered as in the next figure. Complete the values as shown. Click **OK** and you will see the following screen with the formula depicted. What we have done here is generated an expression to measure the delay from the first rising edge of the input to the first falling edge of the output. The threshold voltage levels have been given as 1.25 V for both the input and the output signals (as filled in *Threshold Value 1* and *Threshold Value 2* boxes).

- In *Setting Outputs* window, Click **Get Expression**

- Name this output as *delay1* and click **Add**

- Similarly, add one more output *delay2* which measures the delay from the 1st falling edge of the input to the 1st rising edge of the output.


Calculator
Window Tools Memories Const Options Help
tran ac dc swept_dc info noise rf
Append
Clear Undo Eval
1/x eex Clst Enter
+ 7 8 9
- 4 5 6
x 1 2 3
/ 0 . +/-
<<< Ok Cancel Apply Defaults >>>
cadence

no data set

Selection choices

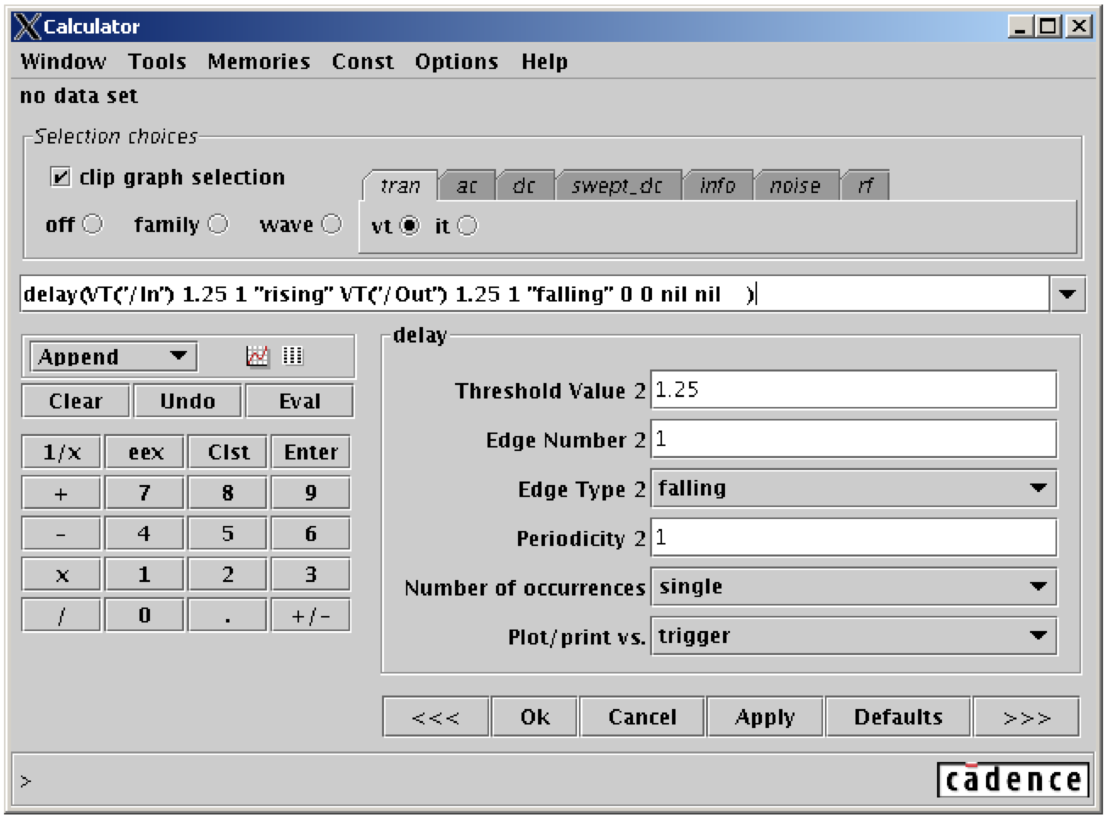
[✓] clip graph selection

off ○ family ○ wave ○ [✓ vt] it ○

```
VT("/In")
```

delay

Threshold Value 2	1.25
Edge Number 2	1
Edge Type 2	falling
Periodicity 2	1
Number of occurrences	single
Plot/print vs.	trigger


Calculator
Window Tools Memories Const Options Help
tran ac dc swept_dc info noise rf
Append
Clear Undo Eval
1/x eex Clst Enter
+ 7 8 9
- 4 5 6
x 1 2 3
/ 0 . +/-
<<< Ok Cancel Apply Defaults >>>
cadence

no data set

Selection choices

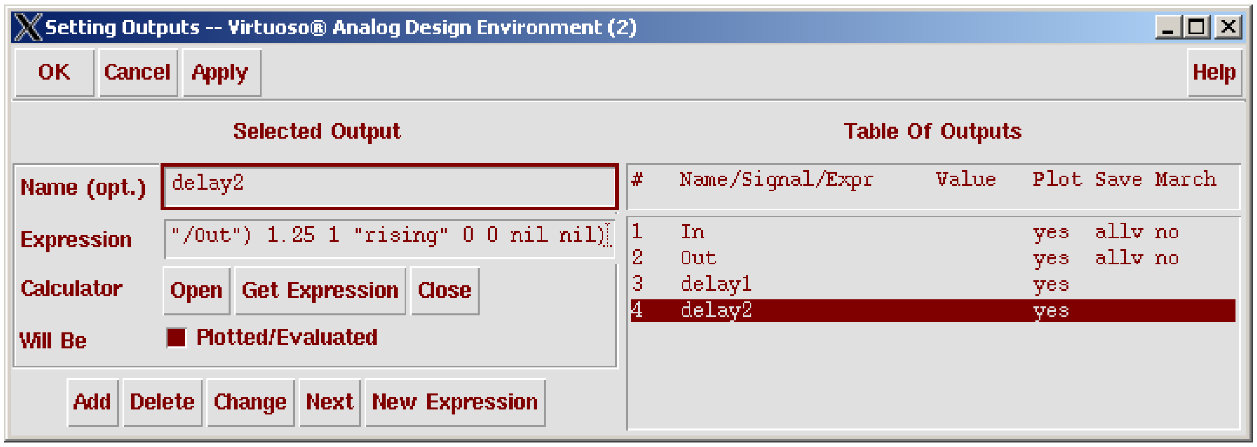
[✓] clip graph selection

off ○ family ○ wave ○ [✓ vt] it ○

```
delay(VT("/In") 1.25 1 "rising" VT("/Out") 1.25 1 "falling" 0 0 nil nil    )
```

delay

Threshold Value 2	1.25
Edge Number 2	1
Edge Type 2	falling
Periodicity 2	1
Number of occurrences	single
Plot/print vs.	trigger



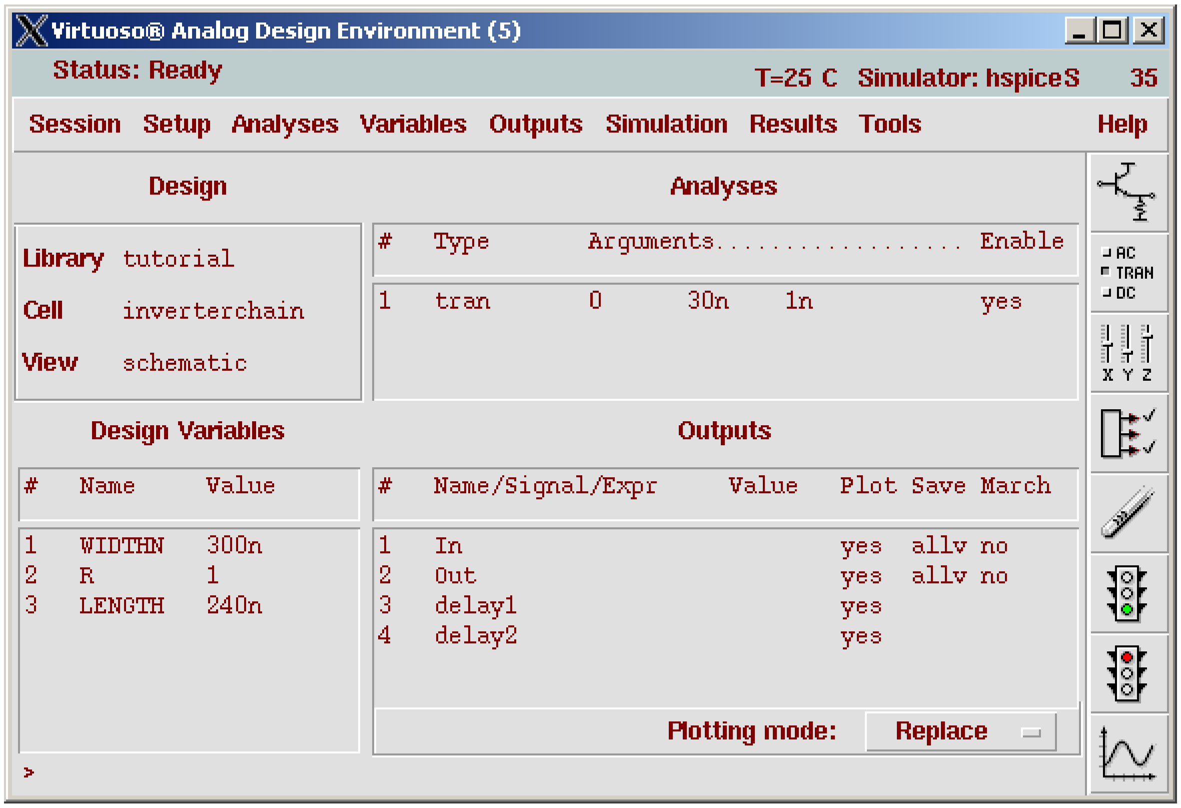
Name (opt.)	delay2
Expression	"/Out") 1.25 1 "rising" 0 0 nil nil)
Will Be	■ Plotted/Evaluated

#	Name/Signal/Expr	Value	Plot	Save	March
1	In		yes	allv	no
2	Out		yes	allv	no
3	delay1		yes		
4	delay2		yes		

- In *Setting Outputs* window, Click **OK.**

3.4 Variable setup

- In ADE, **Variables -› Copy from Cellview**
- Variables will appear in ADE window.
- Double click the variables to set the values as the next figure.



Status: Ready T=25 C Simulator: hspiceS 35

Design

Library	tutorial
Cell	inverterchain
View	schematic

Analyses

#	Type	Arguments			Enable
1	tran	0	30n	1n	yes

Design Variables

#	Name	Value
1	WIDTHN	300n
2	R	1
3	LENGTH	240n

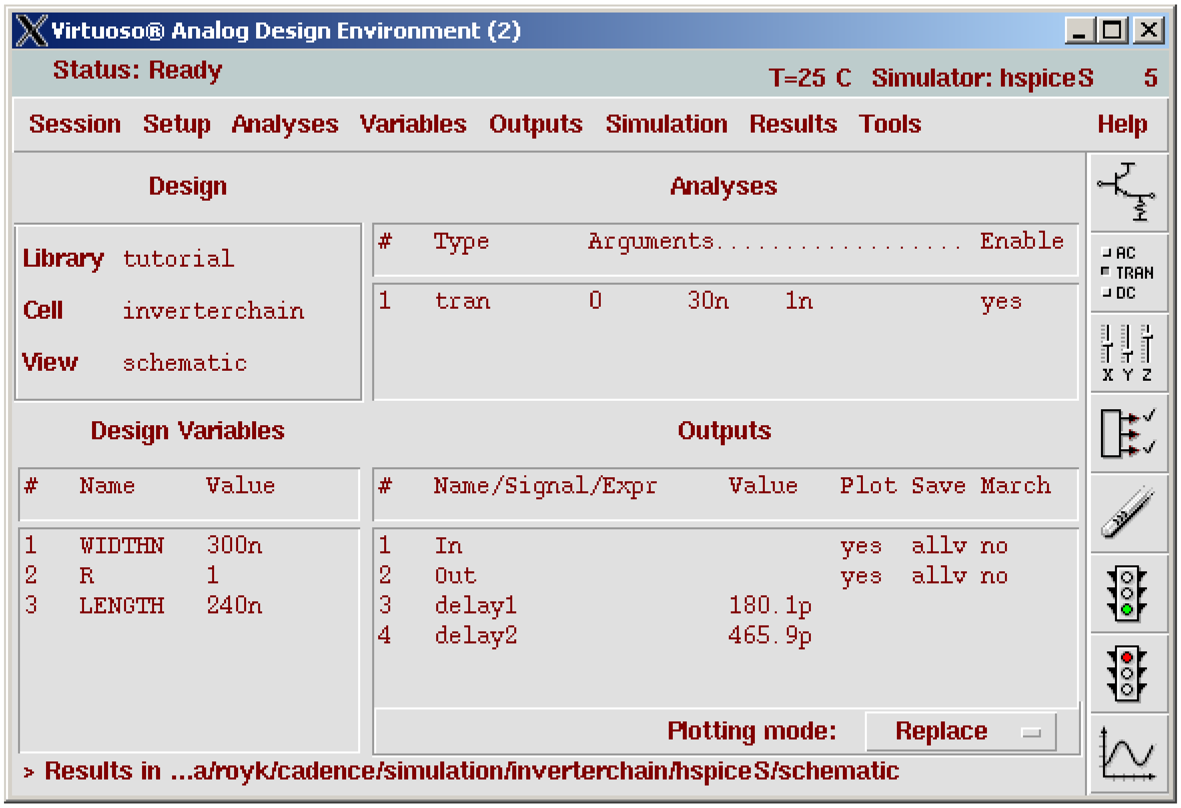
Outputs

#	Name/Signal/Expr	Value	Plot	Save	March
1	In		yes	allv	no
2	Out		yes	allv	no
3	delay1		yes		
4	delay2		yes		

Plotting mode: Replace

3.5 *Running Simulation*

- Run the simulation and note that there are delay measurement results beside the output names delay1 and delay2.


Virtuoso® Analog Design Environment (2)
Status: Ready
T=25 C Simulator: hspiceS 5
Session Setup Analyses Variables Outputs Simulation Results Tools Help
Design
Analyses
AC
TRAN
DC
X Y Z
Design Variables
Outputs
Plotting mode: Replace

Library	tutorial
Cell	inverterchain
View	schematic

#	Type	Arguments			Enable
1	tran	0	30n	1n	yes

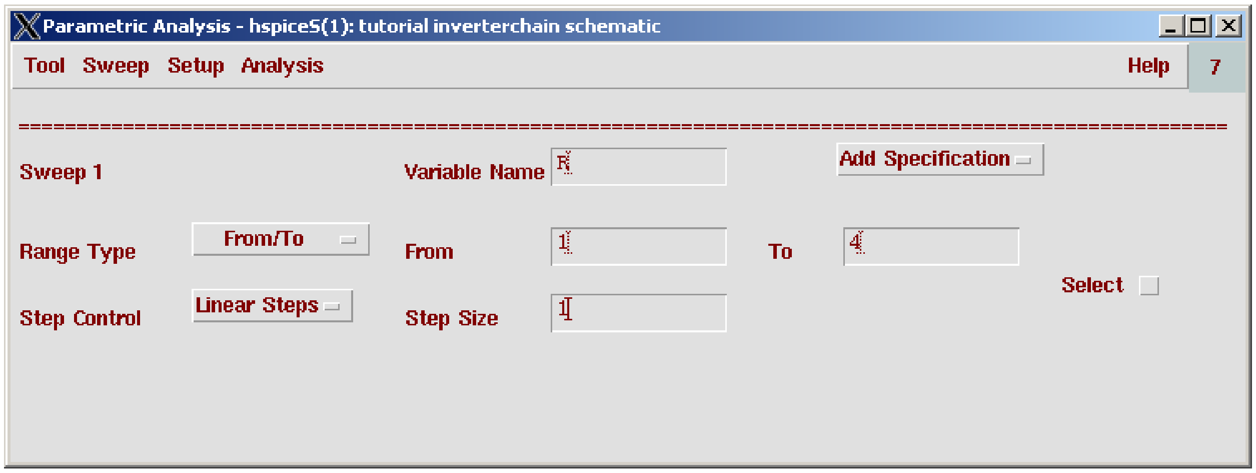
#	Name	Value
1	WIDTHN	300n
2	R	1
3	LENGTH	240n

#	Name/Signal/Expr	Value	Plot	Save	March
1	In		yes	allv	no
2	Out		yes	allv	no
3	delay1	180.1p			
4	delay2	465.9p			

> Results in ...a/royk/cadence/simulation/inverterchain/hspiceS/schematic

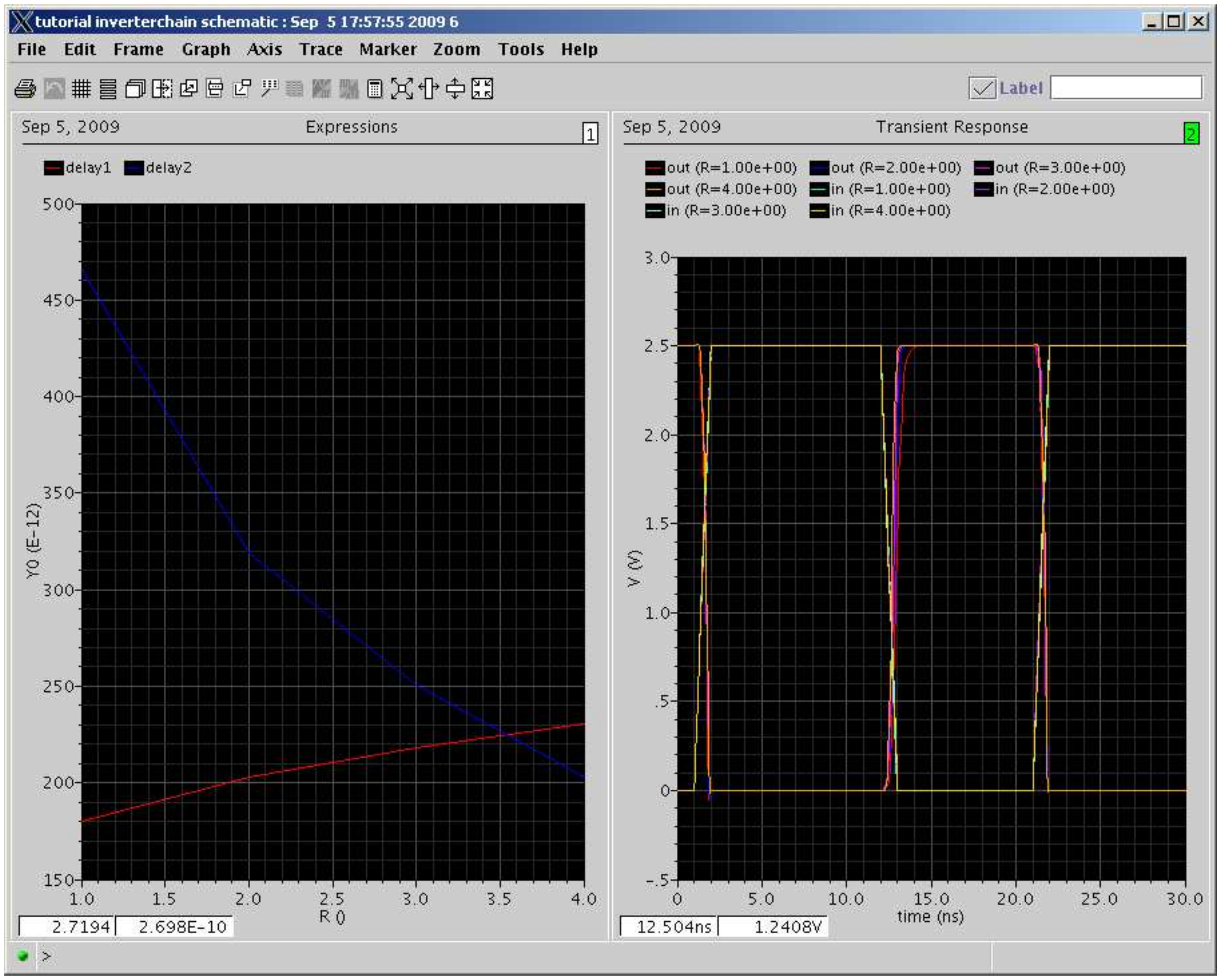
3.6 *Parametric Analysis*

- In ADE, **Tools -› Parametric Analysis**
- Complete the sweep variables as in the following figure.


Parametric Analysis - hspiceS(1): tutorial inverterchain schematic
Tool Sweep Setup Analysis
Help 7
Add Specification
Select

Sweep 1		Variable Name	R		
Range Type	From/To	From	1	To	4
Step Control	Linear Steps	Step Size	1		

- In *Parametric Analysis* window, **Analysis -› Start**
- *Waveform window* will appear with two plots. The right side shows the waveforms of *In* and *Out* pins. The left side plot represents *delay1* and *delay2* values for different *R*.



4 Netlist Extraction & Running HSPICE from Command Line

4.1 *Extracting Netlist*

A netlist file describes the system or circuit to be simulated. The file specifies the components or instances in the circuit, how the circuit is 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 in *Analog Design Environment*, 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 *Analog 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 *Analog 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 *Analog Design Environment* (**~/cadence/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.

4.2 Online Documentation

- To access the online documentation, type `grid hspice -help` in a terminal window.

- In the online documentation, you can get the *pdf* copy of different manuals by clicking the *pdf* link at the bottom of the right frame.

4.3 Running HSPICE

- 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: `.OPTION POST`

  .option post instructs HSPICE to write an output file ending in .tr0 containing the simulation waveforms.

[If you don't like to edit the netlist file after generation, you can choose *Simulation - Options – Analog* menu in *Analog Design 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 command line HSPICE simulation]

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

  ```
  grid hspice inputfile
  ```

  where `inputfile` is the name of your netlist (default name is `hspiceFinal`). The output will be displayed on the screen. You can also store the output of HSPICE in an output file, as follows,

  ```
  grid 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.

Statement or element	**Definition**
title	***first line is the input file title (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.

4.4 HSPICE Netlist Example

Here is an example that shows 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.

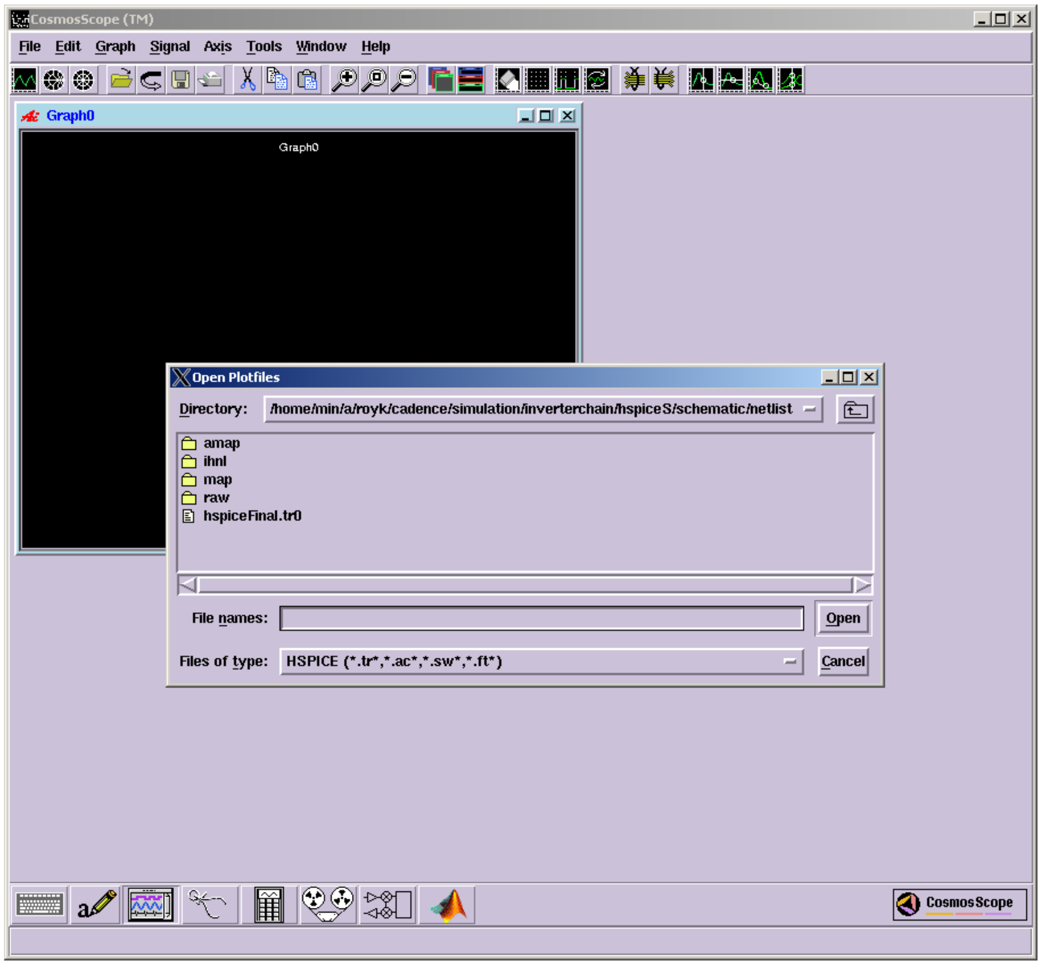
```
* INVERTER

*------------------------------------
* Include model/library file
*------------------------------------
.lib "$CDK_DIR/models/hspice/public/publicModel/tsmc25N" NMOS
.lib "$CDK_DIR/models/hspice/public/publicModel/tsmc25P" PMOS

*-----------------------------------------------
* Parameters
* Several parameters are declared
* These parameters are used later in this file
*------------------------------------------------
.param wn=300n
.param wp=700n
.param l=240n
.param vref=1.25

*-------------------------------------------
* NETLIST
* The width and length of the tr's are not
* numbers but the parameters
*-------------------------------------------
C0 OUT 0  20E-15 M=1.0
MP1 OUT IN VDD! VDD!  TSMC25P  L='l' W='wp'
+AD=+3.00000000E-13 AS=+3.00000000E-13 PD=+2.30000000E-06 PS=+2.30000000E-
06
+M=1
MN1 OUT IN 0 0  TSMC25N  L='l' W='wn'
+AD=+3.00000000E-13 AS=+3.00000000E-13 PD=+2.30000000E-06 PS=+2.30000000E-
06
+M=1

*------------------------------------
* 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=1
+ targ v(out) val=vref rise=1

.measure a_power avg power from=0ns to=30ns
*----------------------------------------------
* ALTER
* HSPICE runs several times 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 simulation
* will be stored in .mt0, .mt1, 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
```

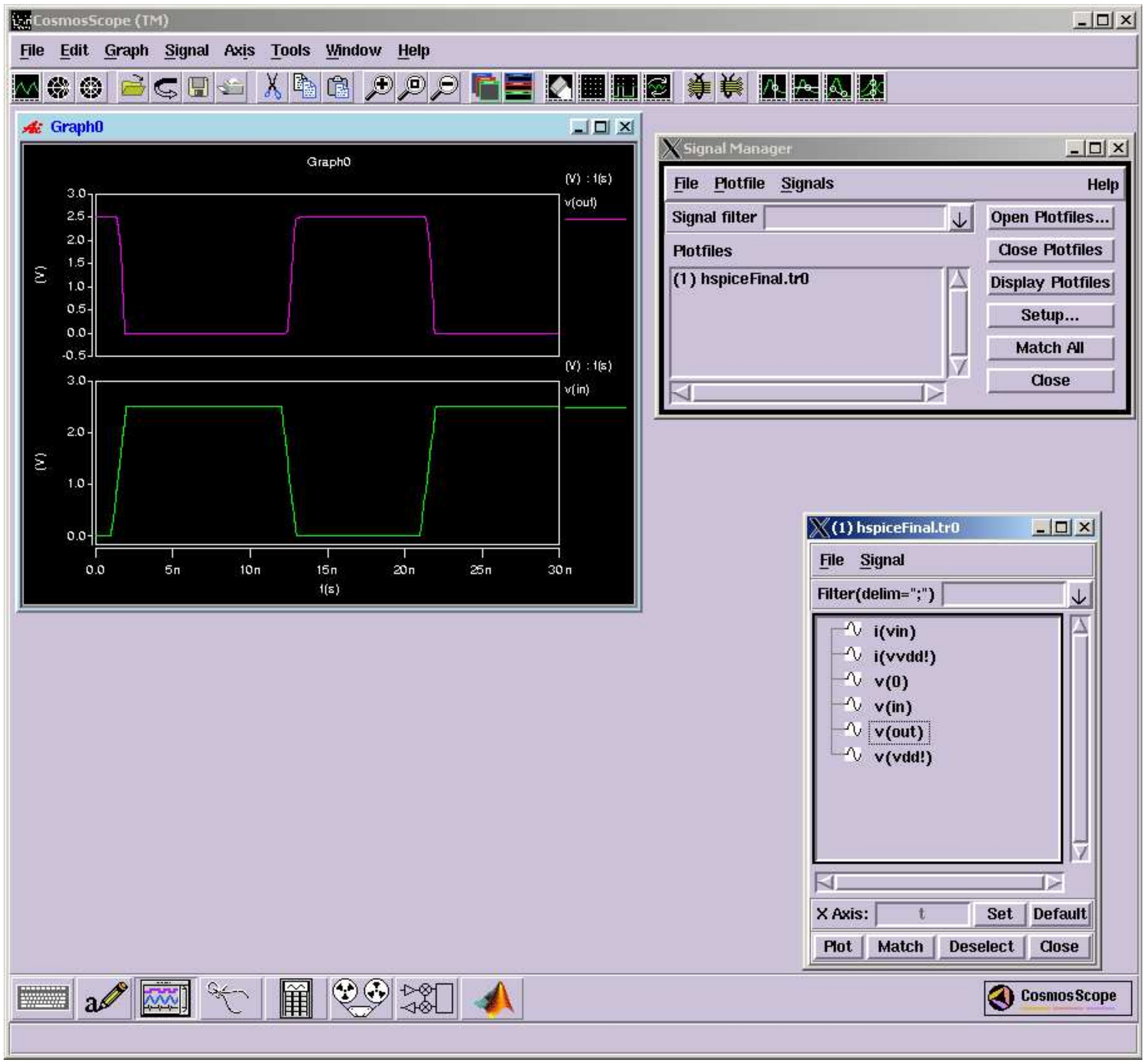
4.5 CosmosScope

CosmosScope and Awaves are graphical waveform viewer. In previous section, you ran HSPICE in command line and the simulation results can be seen by using either *CosmosScope* or *Awaves*. However *Awaves* is no longer supported on linux (ecegrid is a linux cluster). Hence we will not be using *Awaves* in this course.

- To run *CosmosScope*, enter `grid scope` in a terminal window command prompt. After the *CosmosScope* window is loaded, select **File -› Open -› Plotfiles** from the drop down menu.


CosmosScope (TM)
File Edit Graph Signal Axis Tools Window Help
Graph0
Graph0
Open Plotfiles
Directory:
/home/min/a/royk/cadence/simulation/inverterchain/hspiceS/schematic/netlist
amap
ihnl
map
raw
hspiceFinal.tr0
File names:
Open
Files of type:
HSPICE (*.tr*,*.ac*,*.sw*,*.ft*)
Cancel
CosmosScope

- When the *Open Plotfiles* box appears, select *HSPICE(*.tr*,*.ac*,*.sw*,*.ft*)* in the *Files of type* drop box and enter the filename by typing or clicking the file in the browser window (`hspiceFinal.tr0`). Click **OPEN** when done. Two windows will appear - *Signal manager* and *Plot file window*. From Plot file window, select all the signal names and click PLOT. A new waveform window will appear showing the simulation results.



5 Nanosim

Nanosim is a transistor-level circuit simulator that can run 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. This is the reason Nanosim comes under the category *Fast-SPICE*. There is another similar kind of tool from Synopsys that is *HSIM*.

Note that Nanosim is not set up to run under the *Analog Design Environment*. Therefore it is necessary to extract a netlist from your schematic under the *Analog Design Environment* and execute Nanosim from UNIX terminal command prompt. The stimuli and output set up in the *Analog Design Environment* will work in Nanosim, however, it might be easier and more efficient to use the Nanosim configuration file and input vector files to set up the inputs and outputs for designs.

5.1 Online Documentation

The online documentation for `nanosim` can be viewed by entering `grid nssold` at any UNIX command window. You will find there *nanosim user guide* as well as *nanosim command reference guide*.

5.2 Netlist

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

We will use the `hspiceFinal` file from *InverterTest* example. Since you need to modify the file `hspiceFinal` file a bit, copy the file to another file `nanosim.sp`.

```
cp hspiceFinal nanosim.sp
```

5.3 Configuration File

- 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 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. Create a file with those commands in it and save the file as *nanosim.cfg*.

```
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 that 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_eou** *[sim=1|2|3|4|5|6|7] [model=1|2|3|3.5|4|5|6|7] [net=1|2|3|4|5|6|7]* controls the speed and accuracy of the simulation. Lower settings correspond to less accuracy-more performance and higher settings correspond to more accuracy-less performance. **sim_sim_eou** *sim=2 model=2 net=2* is equivalent to a default NanoSim simulation.. Please refer to the online documentation for more details of the above commands.

5.4 Running Nanosim

- To run Nanosim, the command is `nanosim.`
- The Nanosim command line used for this tutorial would be:

  ```
  grid nanosim -n nanosim.sp -c nanosim.cfg -out fsdb -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 `-out` option is used to specify the output file format. Later, you can see the simulation waveform by opening *nanosim.fsdb* 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 library files used. For the subsequent simulations, the `-z` option can be replaced by a `-p` option to save time and to re-use the technology files generated by `-z`. You will find two files created `tt_tech_tsmc25n` and `tt_tech_tsmc25p`. The corresponding part of the command using `-p` option would be `-p tt_tech_tsmc25n tt_tech_tsmc25p`. Note that you can remove the mention of library files from `nanosim.sp`.

5.5 *Using Input Vector File*

This example shows how to use a vector file in your simulation. Create a file named `nanosim.vec` as below.

```
type nsvt
signal IN_0 IN_1

period 20
fall 1
rise 1

radix 1 1

0 0
0 1
1 0
0 1
1 0
1 1
```

- The `type nsvt` command provides a method for reading in a file containing input stimuli and expected data values to be applied at fixed intervals. There are other types are well like `type vec` that provides a method for reading in a file containing input stimuli and expected data values that are organized in a tabular format, i.e., you can specify the time at which some input vector is applied.

- `signal` specifies the list of input/output signal names.

- `period` specifies the fixed time interval applied between two consecutive vectors in the nsvt file.

- `fall` and `rise` specify the *rise* and *fall* times in nanoseconds, respectively.

- `radix` specify the number of bits (the radix) associated with each input signal.

5.5.1 SPICE Netlist

The SPICE netlist file should look like as below. It was extracted from `InverterTest` design schematic. The input stimulus inside the netlist file should be removed or commented while doing nanosim simulation with vector file.

```
.GLOBAL VDD!

C2 OUT_0 0  10E-15 M=1.0
C1 OUT_1 0  10E-15 M=1.0
XI1 IN_0 INTERNAL_0 INVERTERP_1
XI2 INTERNAL_0 OUT_0 INVERTERP_1
XI3 IN_1 NET38 INVERTERP_1
XI4 NET38 INTERNAL_1 INVERTERP_1
XI5 INTERNAL_1 OUT_1 INVERTERP_1

.SUBCKT INVERTERP_1 IN OUT
MP0 OUT IN VDD! VDD!  TSMC25P  L=(3E-07) W=(9E-07)
AD=+6.75000000E-13 +AS=+6.75000000E-13 PD=+3.30000000E-06
PS=+3.30000000E-06 M=1
MN0 OUT IN 0 0  TSMC25N  L=(3E-07) W=(4.5E-07)
AD=+3.37500000E-13 +AS=+3.37500000E-13 PD=+2.40000000E-06
PS=+2.40000000E-06 M=1
.ENDS INVERTERP_1

.lib "$CDK_DIR/models/hspice/public/publicModel/tsmc25N" NMOS
.lib "$CDK_DIR/models/hspice/public/publicModel/tsmc25P" PMOS

*.TRAN  1.00000E-09 5.00000E-08 START=  0.0000
.TEMP    25.0000
*.OP
*.save
*.OPTIONS POST
*vIN_1 IN_1 0 pulse 0 2.5v 1n 1n 1n 10n 20n
*vIN_0 IN_0 0 pulse 0 2.5v 1n 1n 1n 10n 20n
vvdd! vdd! 0 DC=2.5v
.END
```

5.5.2 Command

```
grid nanosim -n nanosim.sp -c nanosim.cfg -nvec nanosim.vec -out
fsdb -z tt
```

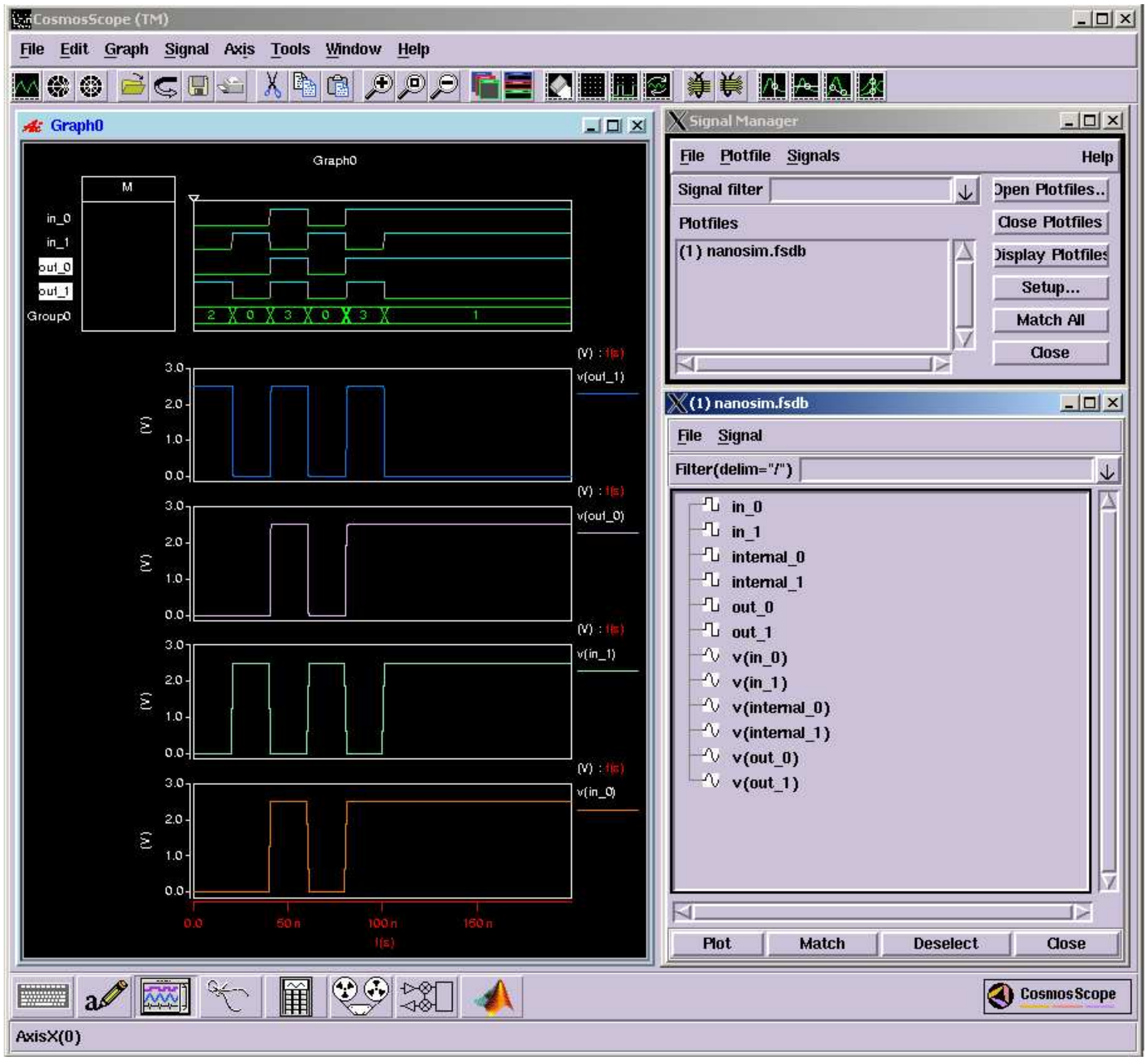
5.6 CosmosScope

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

- To run *CosmosScope*, enter `grid scope` in a terminal window command prompt. After the *CosmosScope* window is loaded, select **File -› Open -› Plotfiles** from the drop down menu. Select *FSDB* for *Files of type* and enter the filename by typing or clicking the file in the browser window (`nanosim.fsdb`). Click **OPEN** when done. Two windows will appear - *Signal manager* and *Plot file window*. From *Plot file window,* select all the signal names and click PLOT. A new waveform window will appear showing the simulation results.

- It is also possible to display signals as a bus. In the waveform window, select `out_0` and `out_1` (these are logic values) by clicking the names and click *create bus* icon. Note that only signals that are logic values can be displayed as buses.

- You can change the signal attributes by double-clicking the signal names.



5.7 Delay Measurement

Create a new configuration file with the following content.

```
print_meas_cmt
print_meas_path
meas_n2n_delay IN_0 - OUT_0 -
meas_n2n_delay IN_1 - OUT_1 -
```

The **print_meas_cmt** command prints BDC (Block Delay Calculation) commands to the SDF (Standard Delay Format) file. The name of the corresponding file would be `nanosim.sdf`.

The **print_meas_path** command prints paths found by BDC. The name of the corresponding file would be `nanosim.path`.

The **meas_n2n_delay** command measures node-to-node delay. The syntax is as follows.

```
meas_n2n_delay [print_neg=0|1] <source_node_name(s)> <edge>
<sink_node_name> <edge>

<edge> ::= (rise | fall | -) ['-' denotes both rise and fall]
```

5.7.1 Command

```
grid nanosim -n nanosim.sp -c nanosim.cfg -nvec nanosim.vec -out
out -z tt
```

5.7.2 Running Simulation

You will get the following warning messages after running the above command.

```
nanosim: WARNING: BDC : No path found between rising edge of in_0 and falling edge of out_0
nanosim: WARNING: BDC : No path found between falling edge of in_0 and rising edge of out_0
nanosim: WARNING: BDC : No path found between rising edge of in_1 and rising edge of out_1
nanosim: WARNING: BDC : No path found between falling edge of in_1 and falling edge of out_1
```

Explain why these warning messages are coming.

5.7.3 Viewing SDF and PATH Files

- In the generated *nanosim.sdf* file, you will find the following lines.

  ```
  (IOPATH in_0 out_0 (0.220:0.220:0.220) (0.280:0.280:0.280))
  (IOPATH in_1 out_1 (0.350:0.350:0.350) (0.260:0.260:0.260))
  ```

  The two triples following are for the *rising* and *falling* delays, respectively. The first number in each triple is the *minimum delay*, the middle number is the *nominal delay*, and the third number is the *maximum delay*. The delays measured by NanoSim use only the minimum and maximum numbers. We have all the nominal, minimum, and maximum delays same as we are using only one typical technology library.

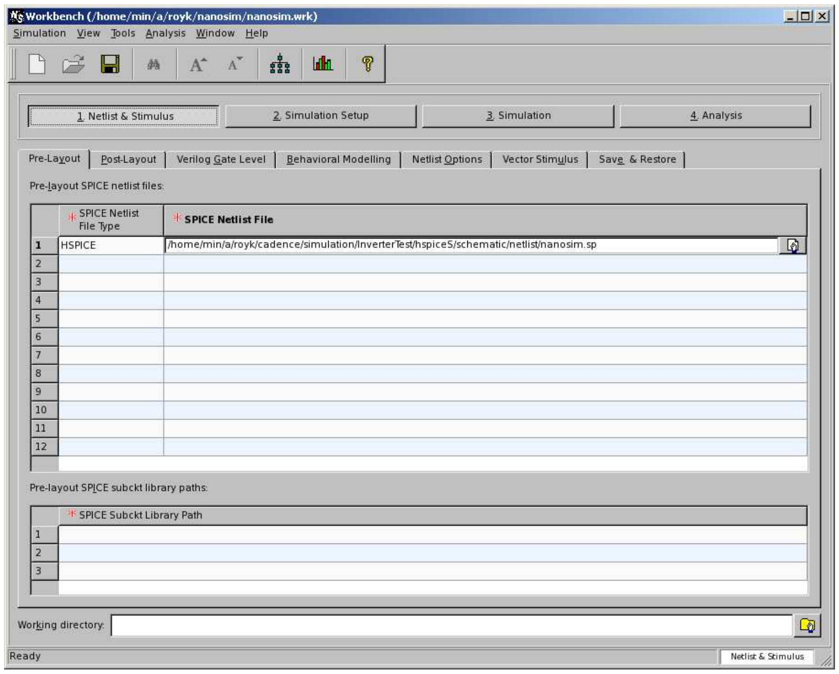
- View the *nanosim.path* file as well. It will tell you the complete signal propagation paths.
  Example: Path from in_1 to out_1 :

  ```
  Node in_1 is rising at time 20.500ns
  Node net38 is falling at time 20.590ns
  Node internal_1 is rising at time 20.680ns
  Node out_1 is falling at time 20.760ns
  ```

5.8 *Using Graphical User Interface*

Type `grid nanosimgui` from a command prompt. It should open the nanosim GUI where you can create a project work file.

- You can include a SPICE file at **Netlist & Stimulas → Pre-Layout** tab.



Pre-layout SPICE netlist files:

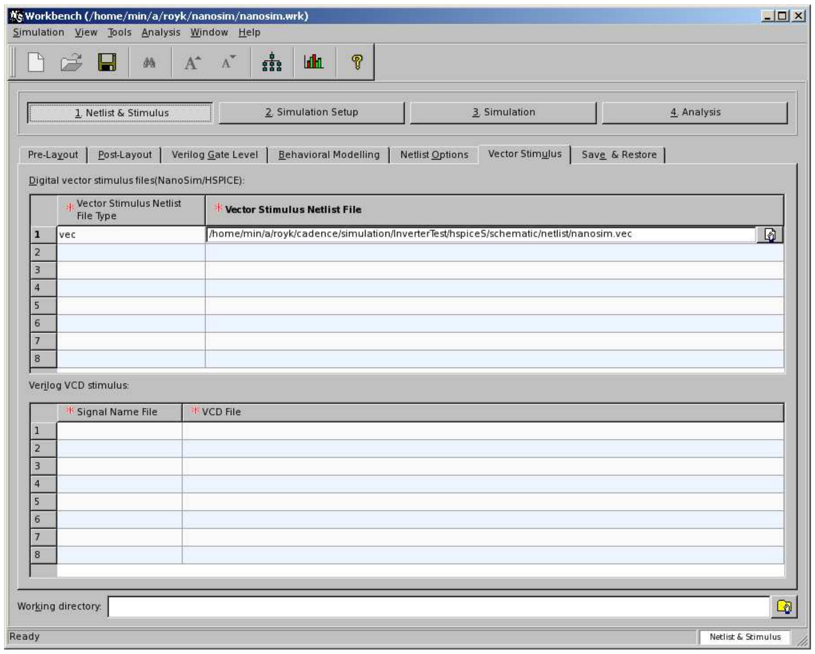
	SPICE Netlist File Type	SPICE Netlist File
1	HSPICE	/home/min/a/royk/cadence/simulation/InverterTest/hspiceS/schematic/netlist/nanosim.sp
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		

Pre-layout SPICE subckt library paths:

	SPICE Subckt Library Path
1	
2	
3	

Working directory:

- You can include a vector file at **Netlist & Stimulas → Vector Stimulas** tab.



Digital vector stimulus files(NanoSim/HSPICE):

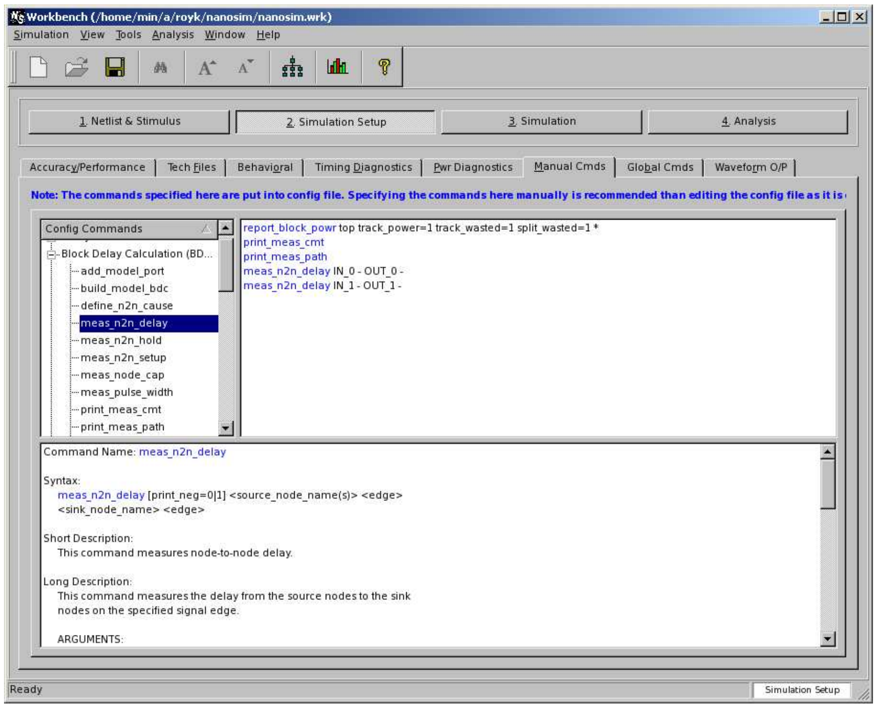
	Vector Stimulus Netlist File Type	Vector Stimulus Netlist File
1	vec	/home/min/a/royk/cadence/simulation/InverterTest/hspiceS/schematic/netlist/nanosim.vec
2		
3		
4		
5		
6		
7		
8		

Verilog VCD stimulus:

	Signal Name File	VCD File
1		
2		
3		
4		
5		
6		
7		
8		

Working directory:

- You can insert commands from **Simulation Setup → Manual Commands** tab. The commands will be automatically included in the nanosim configuration file for simulation.

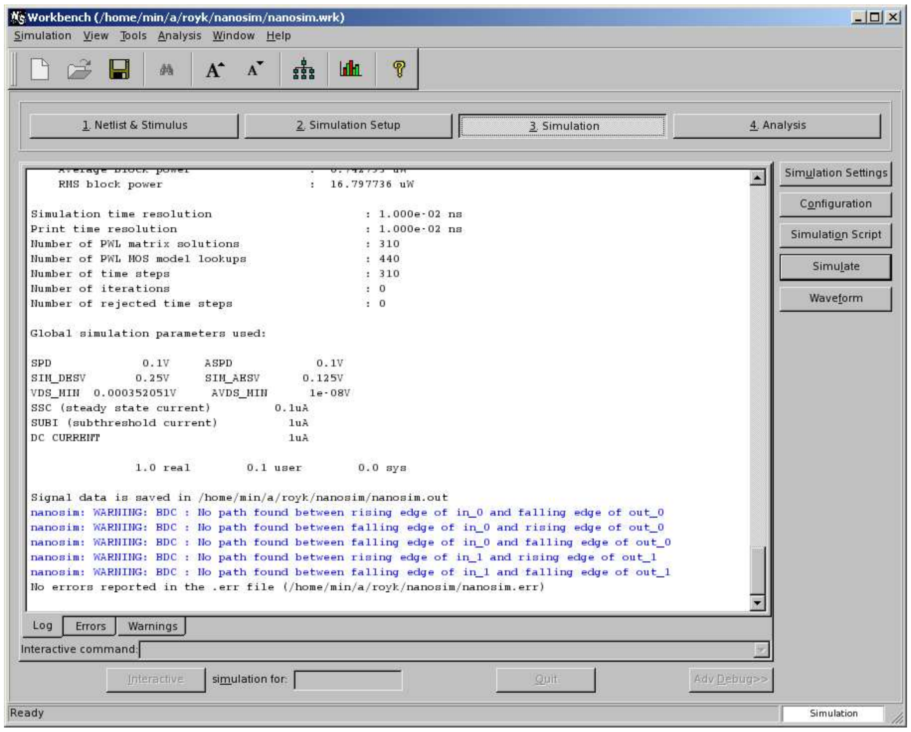


Note: The commands specified here are put into config file. Specifying the commands here manually is recommended than editing the config file as it is

```
report_block_powr top track_power=1 track_wasted=1 split_wasted=1 *
print_meas_cmt
print_meas_path
meas_n2n_delay IN_0 - OUT_0 -
meas_n2n_delay IN_1 - OUT_1 -
```

```
Command Name: meas_n2n_delay

Syntax:
   meas_n2n_delay [print_neg=0|1] <source_node_name(s)> <edge>
   <sink_node_name> <edge>

Short Description:
   This command measures node-to-node delay.

Long Description:
   This command measures the delay from the source nodes to the sink
   nodes on the specified signal edge.

   ARGUMENTS:
```

- You can simulate the specified SPICE file with the corresponding vector file and the configuration file by clicking **Simulation → Simulate** button.



```
    RMS block power                    :  16.797736 uW

Simulation time resolution                     : 1.000e-02 ns
Print time resolution                          : 1.000e-02 ns
Number of PWL matrix solutions                 : 310
Number of PWL MOS model lookups                : 440
Number of time steps                           : 310
Number of iterations                           : 0
Number of rejected time steps                  : 0

Global simulation parameters used:

SPD              0.1V     ASPD            0.1V
SIM_DESV        0.25V     SIM_AESV      0.125V
VDS_MIN  0.000352051V      AVDS_MIN      1e-08V
SSC (steady state current)         0.1uA
SUBI (subthreshold current)          1uA
DC CURRENT                           1uA

              1.0 real        0.1 user        0.0 sys

Signal data is saved in /home/min/a/royk/nanosim/nanosim.out
nanosim: WARNING: BDC : No path found between rising edge of in_0 and falling edge of out_0
nanosim: WARNING: BDC : No path found between falling edge of in_0 and rising edge of out_0
nanosim: WARNING: BDC : No path found between falling edge of in_0 and falling edge of out_0
nanosim: WARNING: BDC : No path found between rising edge of in_1 and rising edge of out_1
nanosim: WARNING: BDC : No path found between falling edge of in_1 and falling edge of out_1
No errors reported in the .err file (/home/min/a/royk/nanosim/nanosim.err)
```

6 NanoTime

The NanoTime is a full-chip, transistor-level critical path finder and static timing analysis solution 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. It is a replacement of the previous tool PathMill. NanoTime is the next-generation transistor-level static timing analysis solution that addresses the emerging challenges in signal integrity (SI) analysis associated with custom designs. NanoTime offers concurrent timing and SI analysis, accuracy within five percent of HSPICE. Its integration with *PrimeTime* (that is a gate-level critical path finder and static timing analysis tool) enables full-chip analysis of designs that includes both gate- and transistor-level blocks.

6.1 Online Documentation

The *user guide* for `nanotime` can be viewed by entering `grid ntdoc` at a UNIX command window.

6.2 SPICE Netlist

We will use the `hspiceFinal` file from *InverterTest* example. Since you need to modify the file `hspiceFinal` file a bit, copy the file to another file `nanotime.sp`.

```
cp hspiceFinal nanotime.sp
```

A couple of modifications need to be done in `nanotime.sp`.

- Enclose the design inside a sub-circuit as below.

  ```
  .SUBCKT InverterTest IN_0 IN_1 OUT_0 OUT_1
  ...
  .ENDS
  ```

  It's necessary because `nanotime` will work on one sub-circuit (however, it can go down hierarchically to another sub-circuit definition) with the specified input and output singals.

- You will find the following line for supply voltage definition in the netlist

  ```
  vvdd! vdd! 0 DC=2.5v
  ```

  Modify the line as follows

  ```
  vvdd! VDD! 0 DC=2.5v
  ```

  It is necessary as `nanotime` works in case-sensitive manner.

After the modification, the netlist should look like as below.

```
.GLOBAL VDD!

.SUBCKT InverterTest IN_0 IN_1 OUT_0 OUT_1
C2 OUT_0 0  10E-15 M=1.0
C1 OUT_1 0  10E-15 M=1.0
XI1 IN_0 INTERNAL_0 INVERTERP_1
XI2 INTERNAL_0 OUT_0 INVERTERP_1
XI3 IN_1 NET38 INVERTERP_1
XI4 NET38 INTERNAL_1 INVERTERP_1
XI5 INTERNAL_1 OUT_1 INVERTERP_1
.ENDS

.SUBCKT INVERTERP_1 IN OUT
MP0 OUT IN VDD! VDD!  TSMC25P  L=(3E-07) W=(9E-07)
AD=+6.75000000E-13 +AS=+6.75000000E-13 PD=+3.30000000E-06
PS=+3.30000000E-06 M=1
MN0 OUT IN 0 0  TSMC25N  L=(3E-07) W=(4.5E-07)
AD=+3.37500000E-13 +AS=+3.37500000E-13 PD=+2.40000000E-06
PS=+2.40000000E-06 M=1
.ENDS INVERTERP_1

.lib "$CDK_DIR/models/hspice/public/publicModel/tsmc25N" NMOS
.lib "$CDK_DIR/models/hspice/public/publicModel/tsmc25P" PMOS

*.TRAN  1.00000E-09 5.00000E-08 START=  0.0000
.TEMP    25.0000
*.OP
*.save
*.OPTIONS POST
*vIN_1 IN_1 0 pulse 0 2.5v 1n 1n 1n 10n 20n
*vIN_0 IN_0 0 pulse 0 2.5v 1n 1n 1n 10n 20n
vvdd! VDD! 0 DC=2.5v     * make vdd! in caps VDD!
.END
```

6.3 Tech File

Create a file `tech.sp` with the contents as follows.

```
*nanosim tech="direct"
*nanosim tech="voltage 2.50"
*nanosim tech="vds 0 2.50 0.02"
*nanosim tech="vgs 0 2.50 0.02"
```

The SPICE model file might contain directives to specify how NanoTime will expand the model to make the lookup tables. The `direct` directive is an optional header to indicate that the following lines are directives for generating the transistor lookup tables.

- `*nanosim tech= "voltage VDD"` NanoTime uses the voltage information to calculate transistor capacitance values used in simulation. In the absence of a voltage directive, NanoTime uses the largest of the V_{dd} voltages specified in the netlist, if available. Otherwise, it uses the voltage specified by the `oc_global_voltage` variable (to be introduced in the next section).

- The `vds` and `vgs` directives specify the end voltage and the voltage step size for the sweeps used to generate the lookup tables for the drain-to-source, gate-to-source, and body-to-source voltage tables, respectively. You might want to specify a finer resolution to minimize interpolation error, or a larger end voltage to accommodate a larger voltage swing, at the cost of more simulation time.

6.4 Script/Configuration File

Create a file `nanotime.scr` with the contents as follows.

```
set search_path {.}
set library_path {.}
set link_path {*}

set oc_global_voltage 2.5

register_netlist -format spice {nanotime.sp tech.sp}

link_design InverterTest

report_port

set input_ports {IN_1 IN_0}
set_port_direction -input $input_ports
set output_ports {OUT_1 OUT_0}
set_port_direction -output $output_ports

report_port

report_design
report_net
report_cell

create_clock -name MCLK -period 10.0
report_clock


match_topology
report_transistor_direction
report_topology
check_topology


set_input_delay -clock MCLK -rise 1.0 {IN_1 IN_0}
set_input_delay -clock MCLK -fall 1.0 {IN_1 IN_0}
set_input_delay -clock MCLK -add_delay 1.0 {IN_1 IN_0}
report_port -input_delay

set_output_delay -clock MCLK 0.0 {OUT_1 OUT_0}

report_port -output_delay
check_design

trace_paths

report_paths -max -max_paths 10

exit
```

- NanoTime searches for each file in each directory listed in the `search_path` variable and uses the first one found.

- `library_path` contains a list of directories to locate undefined sub-circuit references, for compatibility with PathMill.

- `link_path` specifies the location of a list of libraries, design files, and library files used during linking.

- `oc_global_voltage` specifies the default power supply voltage.

- The `match_topology` command runs the NanoTime topology recognition algorithm. NanoTime examines the configuration of transistors in the design to identify and mark various circuit structures such as latches, inverters, and multiplexers. NanoTime needs this information so that it can perform timing checks at the appropriate structures in the design.

- `check_topology` checks for errors in topologies.

- `set_input_delay` specifies the data arrival time at an input relative to a clock.

- `set_output_delay` specifies the data required time at an output relative to a clock.

- The `check_design` command divides the design into units called channel connected blocks (CCBs). Each CCB is a set of transistors whose sources and drains are connected together in a network, in series or in parallel, or both. NanoTime checks the CCBs and reports any problems it has in interpreting their configuration.

- `trace_paths` traces timing paths in the current design and saves the worst-case paths into a database for reporting and other processing.

- `report_paths` reports the timing of paths found during path tracing. You can use `-path_type full` option to report the complete path.

- You can use wildcard character '*' to specify a group of signals, e.g., one can use `IN*` instead of `IN_0` and `IN_1`.

6.5 Command

```
grid nt_shell -f nanotime.scr
```

You can type `quit` to get out from the command prompt `nt_shell>`. Another way is to type `grid nt_shell` at the UNIX command prompt and then write `source nanotime.scr` at the `nt_shell command prompt`.

6.6 Results

The simulation result reports the delay from source nodes to sink nodes specified in script file. At the end you should get an output similar kind of as shown below. Note that the path delays are in `ns`.

```
             Path    Path
   Slack     Delay   Type    Startpoint           Endpoint
-------- -------- ----- -- --------------- -- --------------
   8.783     0.217   D-O f   IN_1            r  OUT_1
   8.810     0.190   D-O r   IN_1            f  OUT_1
   8.860     0.140   D-O f   IN_0            f  OUT_0
   8.860     0.140   D-O r   IN_0            r  OUT_0
```

6.7 Worst Case Input Vector

In the above example we have seen that NanoTime provides us paths (having a start-point and an end-point) with rising/falling transitions along the paths. However, to activate the path we need to set some logic values to the other inputs, not only the input from where the path originates.

For example, consider an OR gate. A rising or falling transition from one input will propagate to the output, only if the other inputs are set as *non-controlling* values, i.e., *logic 0* for an OR gate.

NanoTime can write down a SPICE file for simulating a timing path. The command is as follows.

```
write_spice -output maxpath.spc -header tech.sp
[get_timing_paths -max -max_paths 1]
```

In the generated `maxpath.spc` file, NanoTime writes down the SPICE netlist for a signal-transition path. You may extract an worst case primary input vector from the signal-transition path. If you choose to get more than one timing paths in the `get_timing_paths` command, NanoTime will generate multiple output files each for one timing path.

6.7.1 Script/Configuration File

```
set search_path {.}
set library_path {.}
set link_path {*}

set oc_global_voltage 2.5

register_netlist -format spice {OR3.sp tech.sp}
link_design OR3

report_port
set input_ports {IN*}
set_port_direction -input $input_ports
set output_ports {OUT*}
set_port_direction -output $output_ports
report_port

report_design
report_net
report_cell

create_clock -name MCLK -period 10.0
report_clock

match_topology

report_transistor_direction

report_topology

check_topology

set_input_delay -clock MCLK -rise 1.0 {IN*}
set_input_delay -clock MCLK -fall 1.0 {IN*}
set_input_delay -clock MCLK -add_delay 1.0 {IN*}
report_port -input_delay

set_output_delay -clock MCLK 0.0 {OUT*}
report_port -output_delay

check_design

trace_paths
report_paths -path_type full -max -max_paths 10
write_spice -output maxpath.spc -header tech.sp
[get_timing_paths -max -max_paths 1]
```

6.7.2 SPICE File

```
.GLOBAL VDD!

.SUBCKT OR3 IN_0 IN_1 IN_2 OUT_0
C2 OUT_0 0 10E-15 M=1.0
C1 OUT_1 0 10E-15 M=1.0
XI1 IN_0 IN_1 IN_2 INTERNAL_1 NOR3_1
XI5 INTERNAL_1 OUT_0 INVERTERP_1
.ENDS

.SUBCKT NOR3_1 IN1 IN2 IN3 OUT
MP0 OUT IN1 INT VDD! TSMC25P L=(3E-07) W=(9E-07)
AD=+6.75000000E-13 AS=+6.75000000E-13 PD=+3.30000000E-06
PS=+3.30000000E-06 M=1
MP1 INT IN2 INT2 VDD! TSMC25P L=(3E-07) W=(9E-07)
AD=+6.75000000E-13 AS=+6.75000000E-13 PD=+3.30000000E-06
PS=+3.30000000E-06 M=1
MP2 INT2 IN3 VDD! VDD! TSMC25P L=(3E-07) W=(9E-07)
AD=+6.75000000E-13 AS=+6.75000000E-13 PD=+3.30000000E-06
PS=+3.30000000E-06 M=1
MN0 OUT IN1 0 0 TSMC25N L=(3E-07) W=(4.5E-07)
AD=+3.37500000E-13 AS=+3.37500000E-13 PD=+2.40000000E-06
PS=+2.40000000E-06 M=1
MN1 OUT IN2 0 0 TSMC25N L=(3E-07) W=(4.5E-07)
AD=+3.37500000E-13 AS=+3.37500000E-13 PD=+2.40000000E-06
PS=+2.40000000E-06 M=1
MN2 OUT IN3 0 0 TSMC25N L=(3E-07) W=(4.5E-07)
AD=+3.37500000E-13 AS=+3.37500000E-13 PD=+2.40000000E-06
PS=+2.40000000E-06 M=1
.ENDS NOR3_1

.SUBCKT INVERTERP_1 IN OUT
MP0 OUT IN VDD! VDD! TSMC25P L=(3E-07) W=(9E-07)
AD=+6.75000000E-13 AS=+6.75000000E-13 PD=+3.30000000E-06
PS=+3.30000000E-06 M=1
MN0 OUT IN 0 0 TSMC25N L=(3E-07) W=(4.5E-07) AD=+3.37500000E-
13 AS=+3.37500000E-13 PD=+2.40000000E-06 PS=+2.40000000E-06
M=1
.ENDS INVERTERP_1

.lib "$CDK_DIR/models/hspice/public/publicModel/tsmc25N" NMOS
.lib "$CDK_DIR/models/hspice/public/publicModel/tsmc25P" PMOS

.TEMP 25.0000
vvdd! VDD! 0 DC=2.5v

.END
```

6.7.3 Generated SPICE File

Locate the following sub-circuit as given below in the generated SPICE file (`maxpath.spc`). You can easily get an idea about the generated SPICE file once you go through it. You see that NanoTime has reported the path IN_2 (falling) to OUT_0 (falling) as the worst case delay path and has set the other two inputs IN_0 and IN_1 as the *non-controlling* values, i.e., *logic 0* for an OR gate.

```
.SUBCKT Path6_Arc1 IN_2 INTERNAL_1

* IN_2 input conn_to_trigger
* INTERNAL_1 output conn_to_trigger
MNXI1_MN2 INTERNAL_1 IN_2 Path6_Arc1_0 Path6_Arc1_0 tsmc25n W=0.45u L=0.3u
AS=0.3375p AD=0.3375p PS=2.4u PD=2.4u
MPXI1_MP0 INTERNAL_1 IN_0 XI1_INT Path6_Arc1_XI1_VDD! tsmc25p W=0.9u L=0.3u AS=0.675p
AD=0.675p PS=3.3u PD=3.3u
MPXI1_MP1 XI1_INT IN_1 XI1_INT2 Path6_Arc1_XI1_VDD! tsmc25p W=0.9u L=0.3u AS=0.675p
AD=0.675p PS=3.3u PD=3.3u
MPXI1_MP2 XI1_INT2 IN_2 Path6_Arc1_XI1_VDD! Path6_Arc1_XI1_VDD! tsmc25p W=0.9u L=0.3u
AS=0.675p AD=0.675p PS=3.3u PD=3.3u
* Off transistors
MNXI1_MN0 INTERNAL_1 0 0 Path6_Arc1_0 tsmc25n W=0.45u L=0.3u AS=0.3375p AD=0.3375p
PS=2.4u PD=2.4u
MNXI1_MN1 INTERNAL_1 0 0 Path6_Arc1_0 tsmc25n W=0.45u L=0.3u AS=0.3375p AD=0.3375p
PS=2.4u PD=2.4u
V1 Path6_Arc1_0 0 DC=0
.IC V(INTERNAL_1)=0
V2 IN_0 0 DC=0
V3 IN_1 0 DC=0
.IC V(IN_2)=2.5
.IC V(XI1_INT)=0.028717
.IC V(XI1_INT2)=0.028717
V4 Path6_Arc1_XI1_VDD! 0 DC=2.5

.ENDS Path6_Arc1
```