DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING

CHITTAGONG UNIVERSITY OF ENGINEERING AND TECHNOLOGY

CHATTOGRAM – 4349, BANGLADESH

Experiment No. 3

Design and Performance Analysis of an Inverter Using Cadence Virtuoso

PRECAUTIONS:

- Students must carefully read the lab manual before coming to lab to avoid any inconveniences.
- Students must carry a flash drive to transfer files and lab manuals.
- Use of mobile phone in lab is strictly prohibited and punishable offence.
- Experiment files must be uploaded to GitHub including home tasks.

OBJECTIVES:

- To familiarize with Cadence tools and library creation.
- To familiarize with inverter design and implementation.
- To familiarize with transient analysis and performance measurements.

THEORY:

Custom IC Design Flow:

Figure 1 shows the basic design flow of a custom IC design, together with the Cadence tools required in each step.

First, a schematic view of the circuit is created using Cadence Virtuoso Schematic Editor L. Then, the circuit is simulated using Cadence Analog Design Environment (ADE L). Different simulators can be employed; some sold with the Cadence software (e.g., Spectre) some from other vendors (e.g., HSPICE) if they are installed and licensed.

Once circuit specifications are fulfilled in simulation, the circuit layout is created using Virtuoso Layout Editor L. The resulting layout must verify some geometric rules dependent on the technology (design rules). For enforcing it, a Design Rule Check (DRC) is performed. Optionally, some electrical errors (e.g., shorts) can also be detected using an Electrical Rule Check (ERC). Then, the layout should be compared to the circuit schematic to ensure that the intended functionality is implemented. This can be done with a Layout Versus Schematic (LVS) check. All these verification tools are included in the Assura software in Cadence.

Finally, a netlist including all layout parasitics should be extracted using Quantus QRC tool, and a final simulation of this netlist should be made. This is called a post-Layout simulation, and is performed with the same Cadence simulation tools.

Once verified the layout functionality, the final layout is converted to a certain standard file format (GDSII, CIF, etc.) depending on the foundry using the Cadence conversion tools.

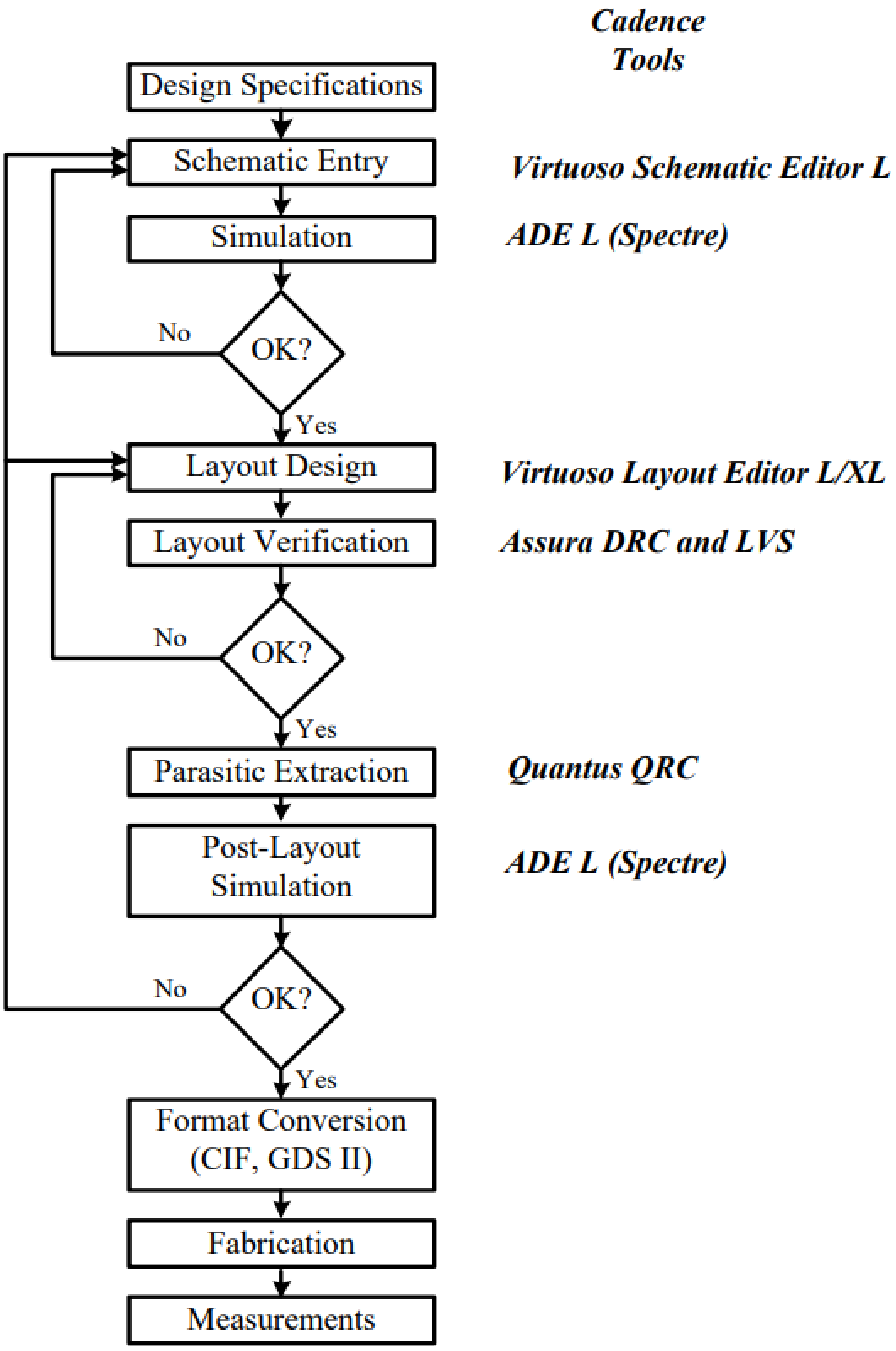


Figure 1: Flowchart of Custom IC design.

PDK:

Cadence is an Electronic Design Automation (EDA) environment that allows integrating in a single framework different applications and tools (both proprietary and from other vendors), allowing to support all the stages of IC design and verification from a single environment. These tools are completely general, supporting different fabrication technologies. When a particular technology is selected, a set of configuration and technology-related files are employed for customizing the Cadence environment. This set of files is commonly referred as a process design kit.

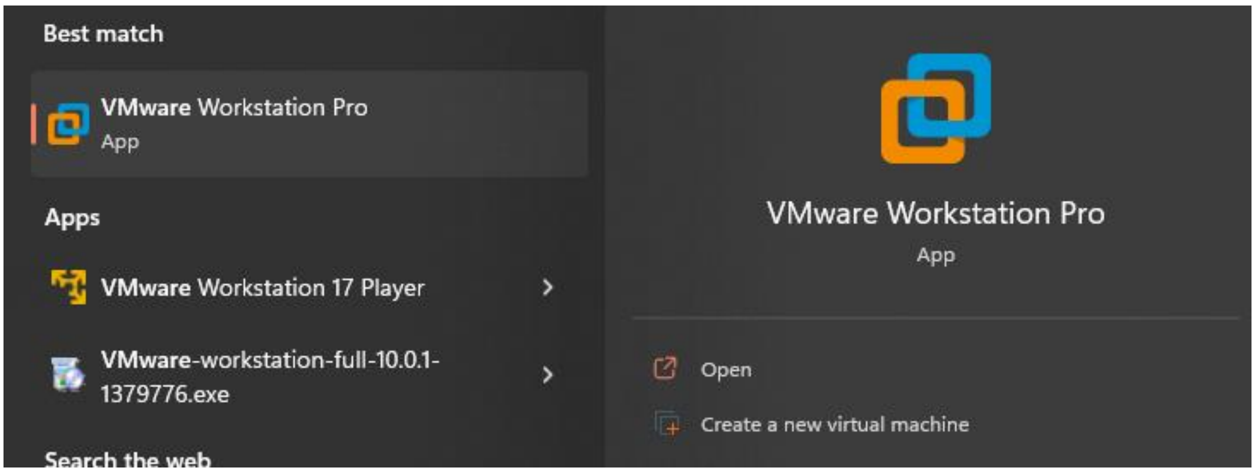
All VLSI designs start with a Process Design Kit known briefly as PDK. A PDK contains the process technology and needed information to do device-level design in the Cadence Design Framework II (DFII) environment.

Throughout the labs we will use a generic, foundry independent 90nm CMOS mixed-signal process kit developed by Cadence. We will call it generic PDK 90 nm briefly as gpdk090. A PDK contains all the necessary design and technology data to successfully design and simulate a VLSI chip on a particular foundry. The foundry provides the necessary technological data, design rules, and the device models. Also, PDK contains schematic symbols with all necessary views, as well as device extraction rules for Layout versus Schematic (LVS) check. It also provides parasitic extraction rules.

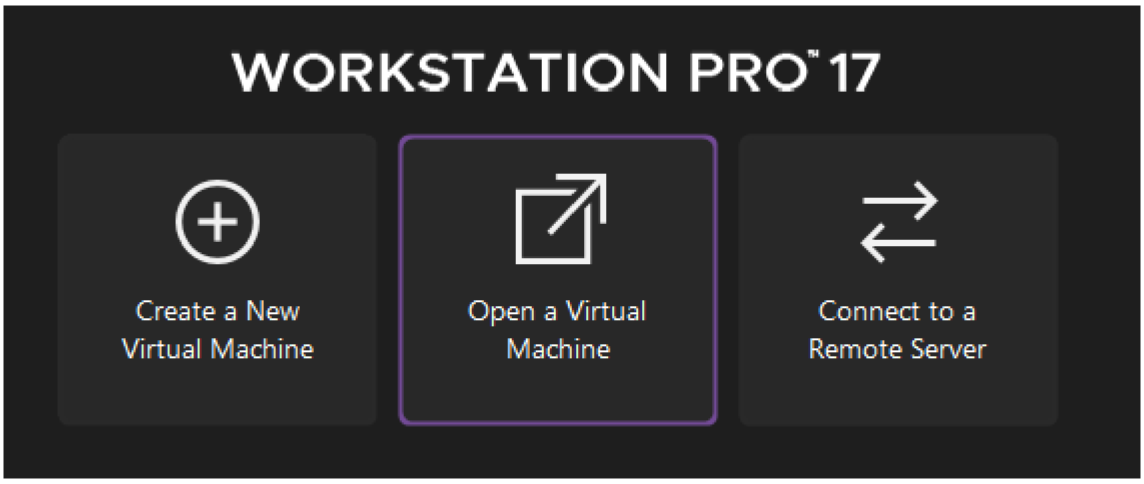
DESIGN PROCESS:

Opening Cadence:

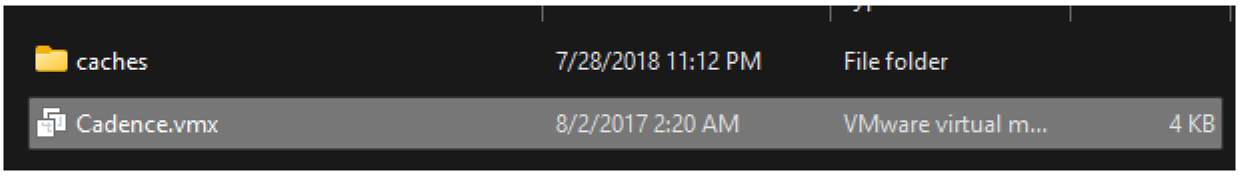
1. Start **VMware Workstation Pro** from the start menu.


Best match
VMware Workstation Pro
App
Apps
VMware Workstation 17 Player
VMware-workstation-full-10.0.1-1379776.exe
Search the web
VMware Workstation Pro
App
Open
Create a new virtual machine

2. Click **Open a Virtual Machine.**

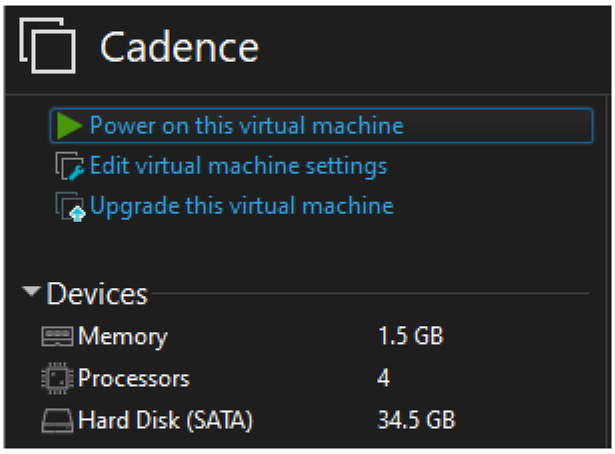


3. Locate the **Cadence.vmx** file from your storage media.



caches	7/28/2018 11:12 PM	File folder	
Cadence.vmx	8/2/2017 2:20 AM	VMware virtual m...	4 KB

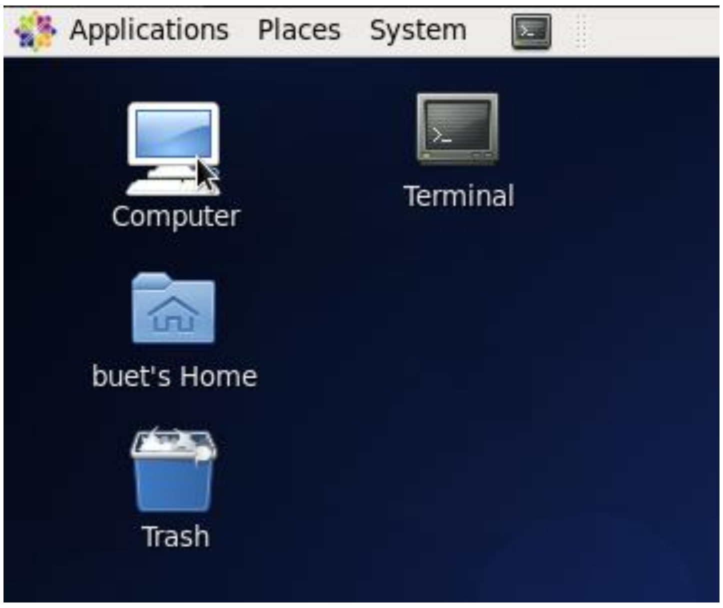
4. Change configuration as necessary then start the VM. Click **“I Copied It”** if asked.


Cadence
Power on this virtual machine
Edit virtual machine settings
Upgrade this virtual machine
Devices

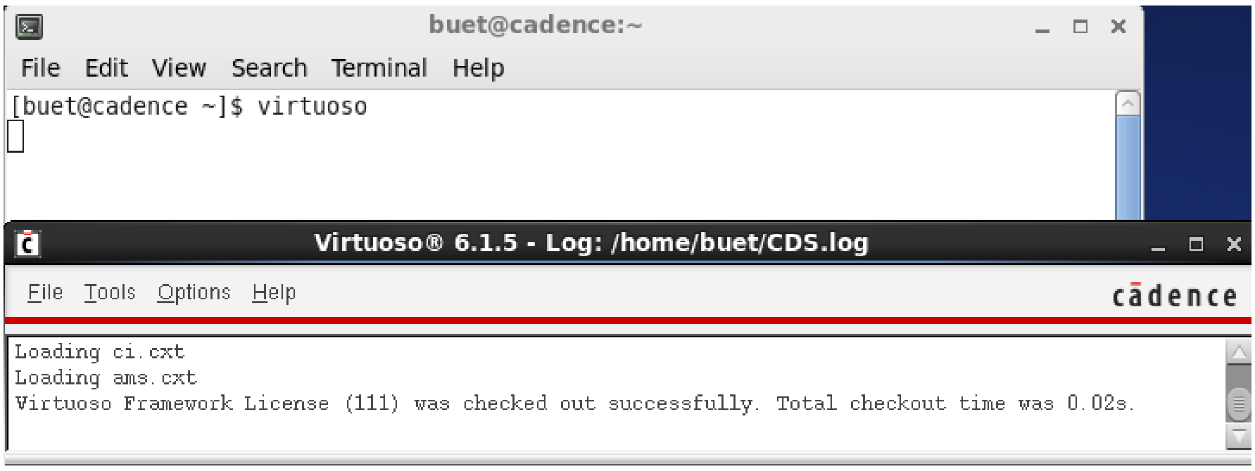
Memory	1.5 GB
Processors	4
Hard Disk (SATA)	34.5 GB

Starting Virtuoso:

1. Start the **“Terminal”** from the home screen.



2. Type in **“virtuoso”** and press enter. The **Command Interpreter Window (CIW)** will appear.



```
[buet@cadence ~]$ virtuoso
```

```
Loading ci.cxt
Loading ams.cxt
Virtuoso Framework License (111) was checked out successfully. Total checkout time was 0.02s.
```

Creating and Attaching Library:

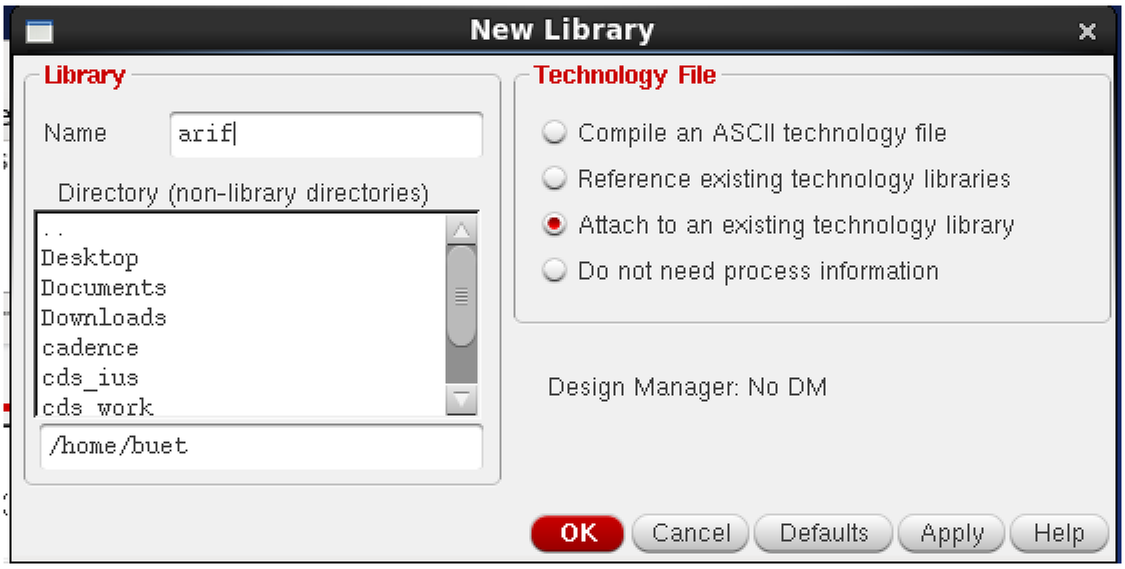
All the entities in Cadence are managed using libraries, and each library contains cells. Each cell contains different design views (the structure is similar –and physically corresponds - to a directory (library) containing subdirectories (cells), each one containing files (views). Thus, for instance, a certain circuit (e.g., an inverter) can be stored in a library, and such library can contain the different logic blocks (basic gates, flip-flops, registers, etc) stored as cells. Each block (cell) contains different views (schematic, layout, symbol, etc.).

There are usually three types of libraries:

- A set of common Cadence libraries that come with the Cadence software containing basic components, such as voltage and current sources, R, L, C, etc. (e.g., analogLib).
- Libraries that come with a certain design kit (e.g., gpdk090) and that are related to a certain technology (e.g., transistors with a certain model attached, etc.).
- User libraries; where the user stores its designs. These designs employ components from the Cadence/design kit libraries.

It is recommended that you use a library to store related cell views; e.g., use a library to hold all the cell views for a single project (that can involve a complete chip design). In our example, we are going to create a new library for our design and attach it to desired technology library.

1. In the **CIW**, execute **File > New > Library**.
2. The **‘New Library’** form appears. In the name field of the New Library type **mylib** or any name of your choice.



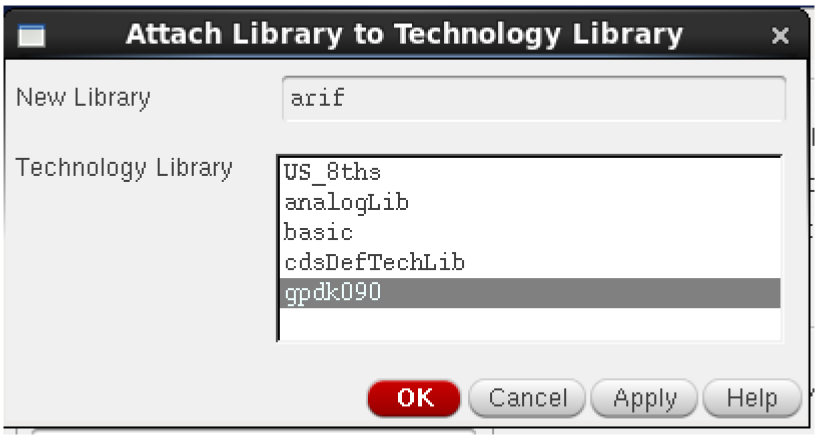
Name: arif

/home/buet

○ Compile an ASCII technology file
○ Reference existing technology libraries
● Attach to an existing technology library
○ Do not need process information

Design Manager: No DM

3. Select **Attach to an existing technology library** and click **OK**. **'Attach Library to Technology Library'** window will appear.

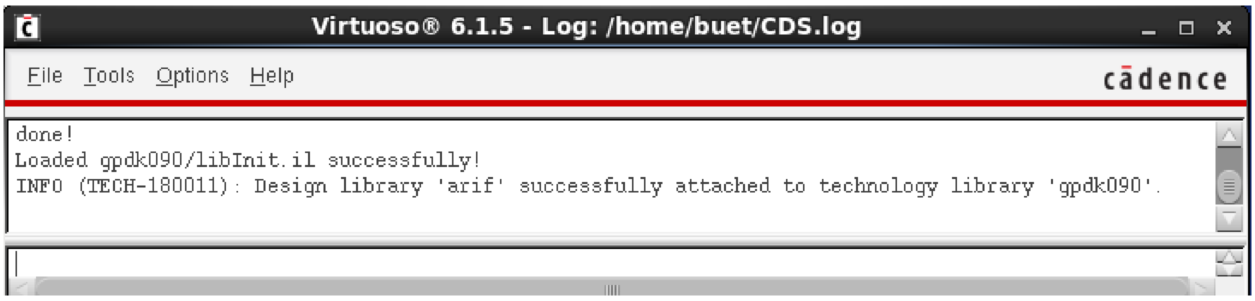


New Library: arif

Technology Library: gpdk090

4. Select **gpdk090** technology library and click **OK**. This will be the technology chosen for your design (that you will employ eventually for fabrication). Now all the designs made in this library are technology-dependent (e.g., the schematic MOS symbols have by default the model for this technology, the available layout layers correspond to this technology, etc.).

5. The following will appear on the **CIW.**

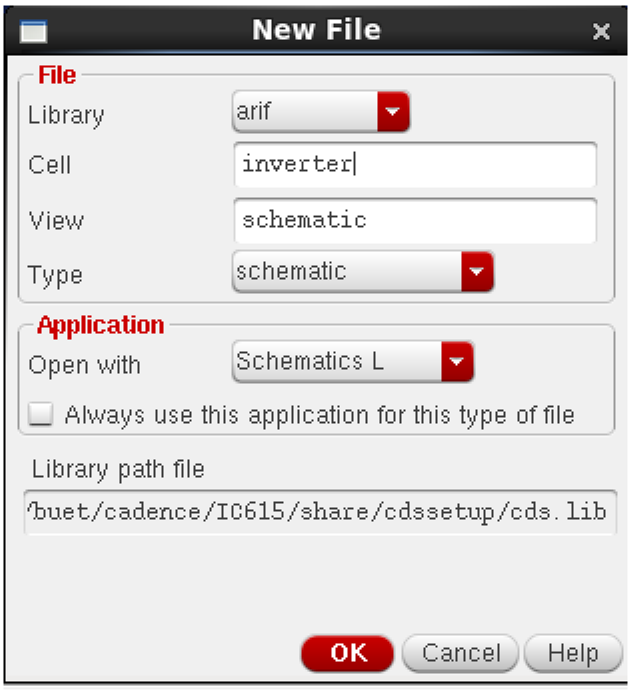


```
done!
Loaded gpdk090/libInit.il successfully!
INFO (TECH-180011): Design library 'arif' successfully attached to technology library 'gpdk090'.
```

Schematic Entry:

In this exercise, you will learn how to enter simple schematic and run a simulation to perform timing simulation of an inverter designed using gpdk090 technology.

1. In the **CIW**, execute **File** > **New** > **Cellview**. Set up the **'New File'** form as follows: **Library**: mylib, **Cell**: inverter, **View**: schematic, **Type**: schematic, **Open with**: Schematics L; then click **OK**.



New File

File

Library: arif

Cell: inverter

View: schematic

Type: schematic

Application

Open with: Schematics L

□ Always use this application for this type of file

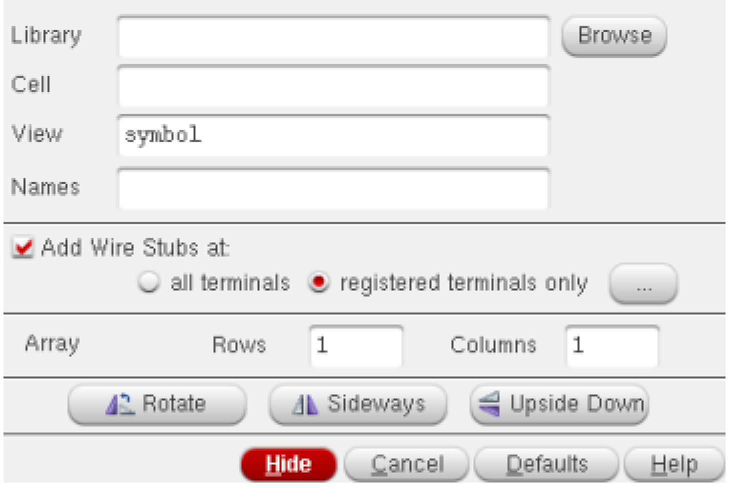
Library path file

/buet/cadence/IC615/share/cdssetup/cds.lib

2. This blank window shall now appear.

Instance Creation:

1. To create an instance, you can execute **Create > Instance** in Virtuoso schematic editor window or simply use shortcut key **“i”**. The following window will appear:



Library:

Cell:

View: symbol

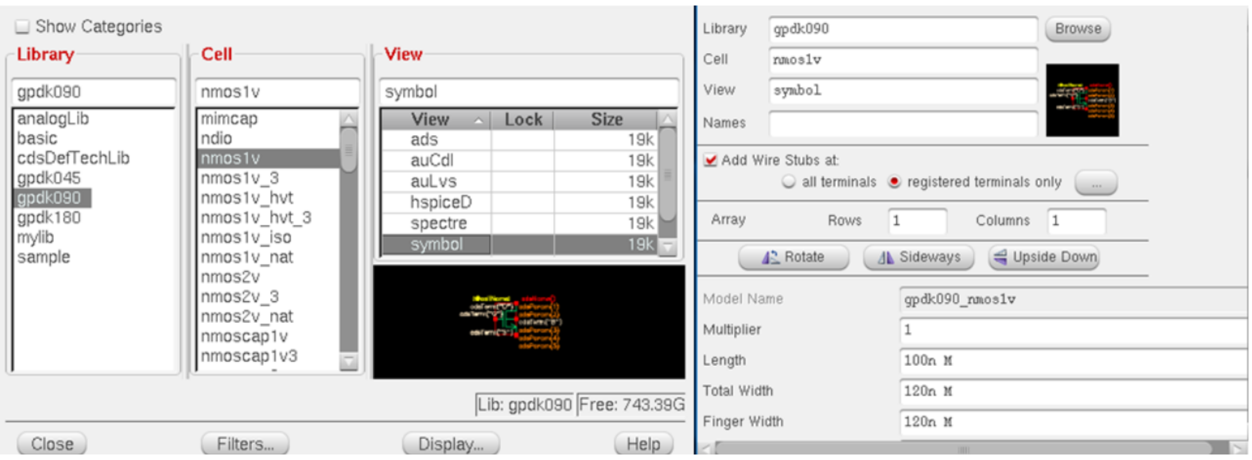
Names:

☑ Add Wire Stubs at:

○ all terminals ● registered terminals only

Array Rows: 1 Columns: 1

2. Click **Browse** to select a library component. Another window will show up. Choose **Library**: gpdk090, **Cell**: nmos1v, **View**: symbol. (Note that while you are doing this, the **‘Add Instance’** form is getting updated as well).



□ Show Categories

Library: gpdk090

analogLib
basic
cdsDefTechLib
gpdk045
gpdk090
gpdk180
mylib
sample

Cell: nmos1v

mimcap
ndio
nmos1v
nmos1v_3
nmos1v_hvt
nmos1v_hvt_3
nmos1v_iso
nmos1v_nat
nmos2v
nmos2v_3
nmos2v_nat
nmoscap1v
nmoscap1v3

View: symbol

View	Lock	Size
ads		19k
auCdl		19k
auLvs		19k
hspiceD		19k
spectre		19k
symbol		19k

Lib: gpdk090 Free: 743.39G

Library: gpdk090

Cell: nmos1v

View: symbol

Names:

☑ Add Wire Stubs at:

○ all terminals ● registered terminals only

Array Rows: 1 Columns: 1

Model Name: gpdk090_nmos1v

Multiplier: 1

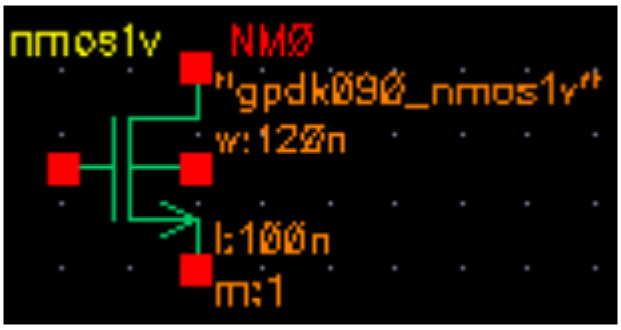
Length: 100n M

Total Width: 120n M

Finger Width: 120n M

3. Make sure that the view name field in the form is set to symbol. After you complete the form, move your cursor to the schematic window and click left button of mouse to place the

component. After entering the components, click Cancel in the Create Instance form or press Esc keeping your cursor in the schematic window.

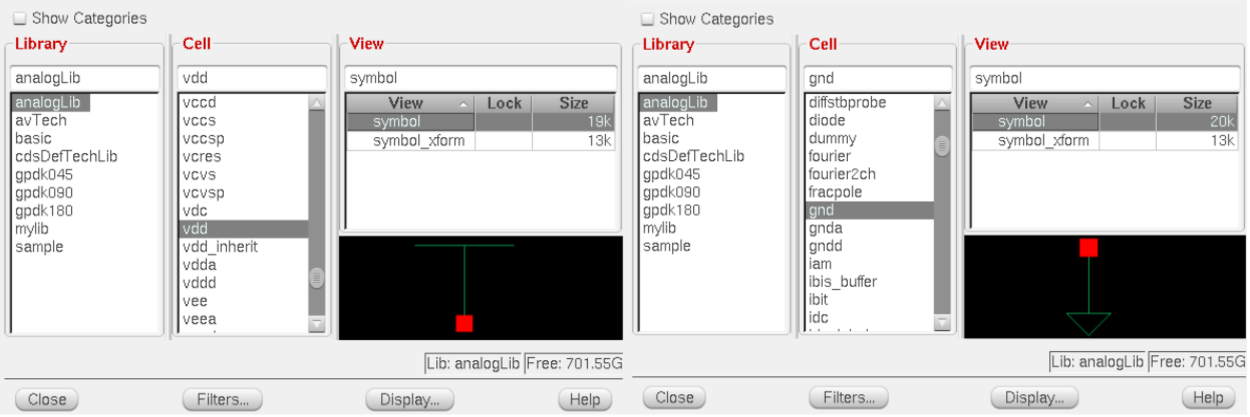


Similarly, add pmos1v cell. If you place component in the wrong location, press ‘m’ on keyboard, click once on the component to select it and move the mouse to move the component to your desired location.

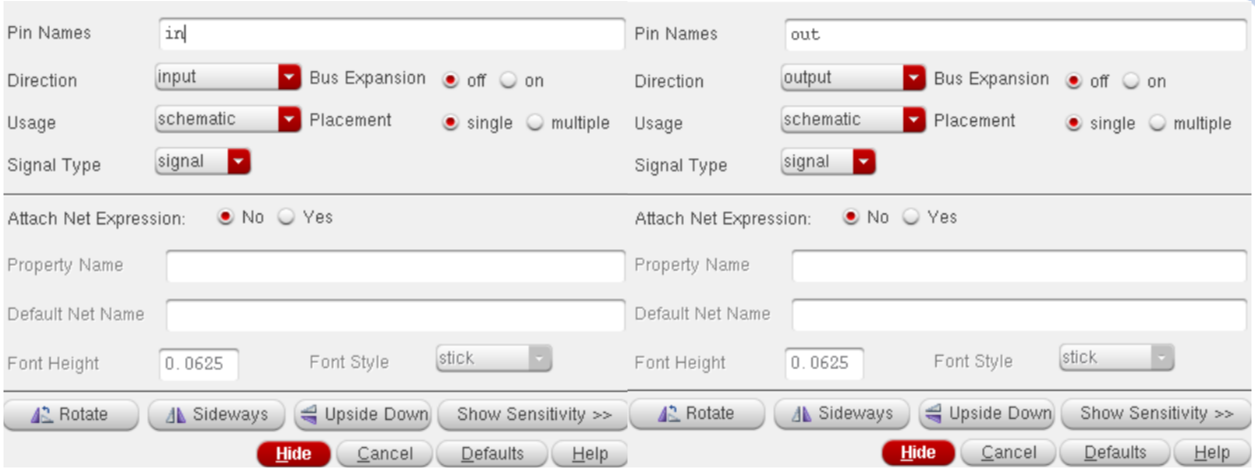
4. Now we can adjust the sizes of the transistors by editing instance properties. Left click on the NMOS to select the component. Then, press **“q”** to modify its properties, or in schematic editor window, execute **Edit** > **Properties** > **Object**. You will update the Library Name, Cell Name, and the property values given in the table below as you place each component. The inverter design contains the following cells from the following libraries.

Library Name	**Cell Name**	**Properties/Comment**
gpdk090	*nmos1v*	For NM0, Width=240n (this is 2x the minimum channel width)
gpdk090	*pmos1v*	For PM0, Width=480n
analogLib	*vdd*	
analogLib	*gnd*	

For example, while modifying the transistor width for NMOS, set **Total Width** to 240n, and then press **‘Tab’** key and the **Finger Width** will be set to the same value. Click **OK**. Repeat this for PMOS to set **Total Width** and **Finger Width** to 480n. To deselect any object, press keyboard command **“Ctrl+d”**. Next, instantiate power nets (cell **vdd** and **gnd** from **analogLib** library).



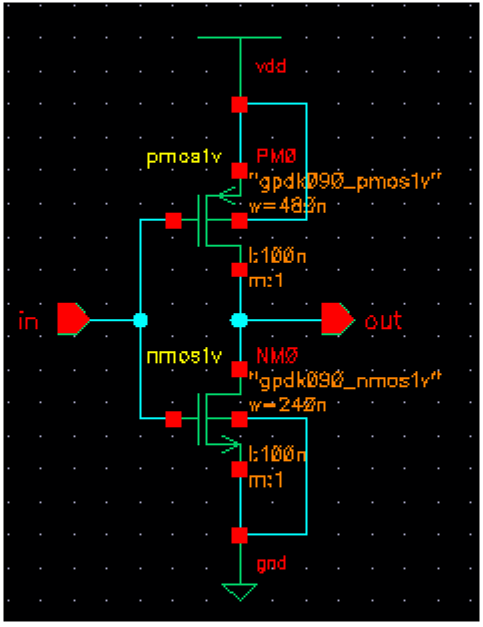
5. Execute **Create** > **Pin** or press **‘p’** on keyboard. **‘Add Pin’** form will appear. Enter the name of the pin and **Direction** of the pin. Add all the pins (**in, out**) to the schematic. For an inverter, gate input pin (e.g., in) is the input and output pin (e.g., out) at the common node between drains of NMOS and PMOS is output of the inverter. So, select **Direction** property as **input** for **in**, and **output** for **out**.



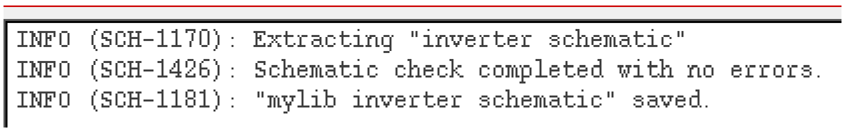
Pin Names: in
Direction: input   Bus Expansion ● off ○ on
Usage: schematic   Placement ● single ○ multiple
Signal Type: signal
Attach Net Expression: ● No ○ Yes
Property Name
Default Net Name
Font Height: 0.0625   Font Style: stick

Pin Names: out
Direction: output   Bus Expansion ● off ○ on
Usage: schematic   Placement ● single ○ multiple
Signal Type: signal
Attach Net Expression: ● No ○ Yes
Property Name
Default Net Name
Font Height: 0.0625   Font Style: stick

6. Use **Add** > **Wire** menu or simply press **‘w’** key while staying on the schematic editor to enter wiring mode / Esc to exit. Click and release left button of mouse to start wire connections and click again at another point to draw wire connection. It is a good practice to periodically save your work by clicking on **Check and Save** button (the checkmark button just below the Tools menu). You can also save your work from the drop-down menu **File** > **Save**. The final schematic looks like the following one:



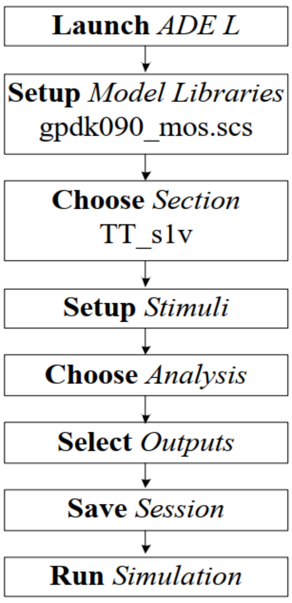
7. Click **Check and Save**.
8. Check CIW for errors or warnings. Some license warnings may be ignored. If there are no error or design warning, you should see the following message:



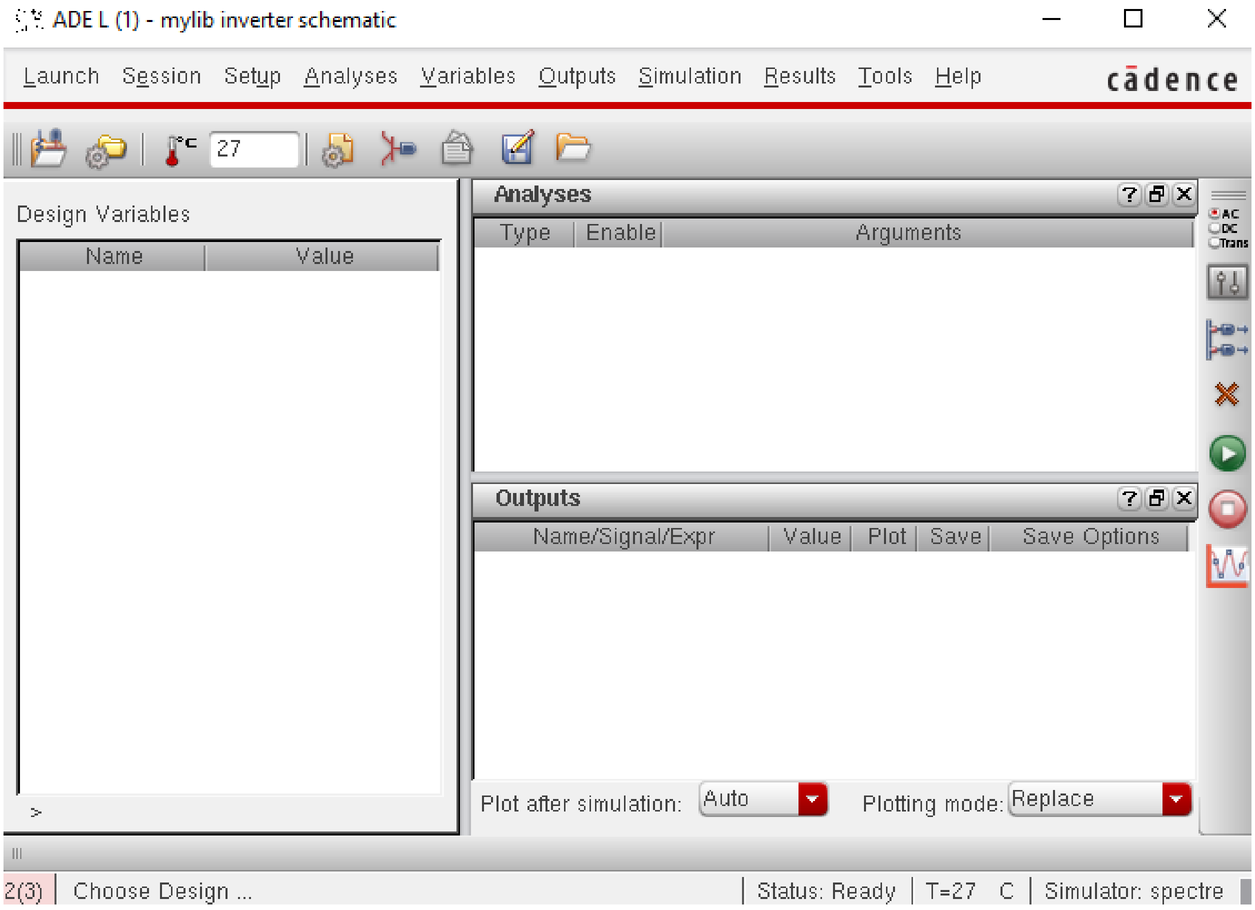
```
INFO (SCH-1170): Extracting "inverter schematic"
INFO (SCH-1426): Schematic check completed with no errors.
INFO (SCH-1181): "mylib inverter schematic" saved.
```

Netlist and Simulation:

The following flowchart shows the steps to be executed to simulate a design using ADE L:



1. In the Schematic editor window, execute **Launch** > **ADE L**. The **Analog Design Environment (ADE) L** window will arrive.



ADE L (1) - mylib inverter schematic
Launch Session Setup Analyses Variables Outputs Simulation Results Tools Help
cadence
AC
DC
Trans

°C 27

Design Variables

Name	Value

Analyses

Type	Enable	Arguments

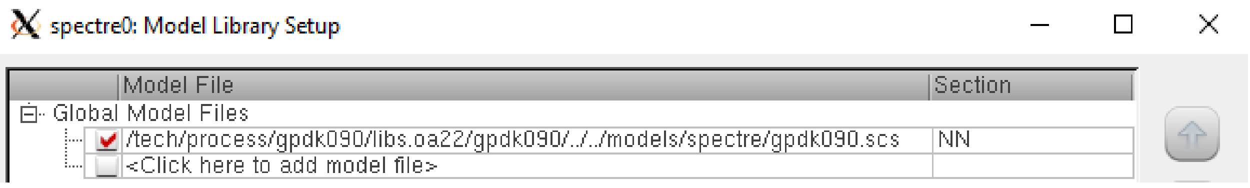
Outputs

Name/Signal/Expr	Value	Plot	Save	Save Options

Plot after simulation: Auto   Plotting mode: Replace

2(3) | Choose Design ... | Status: Ready | T=27 C | Simulator: spectre

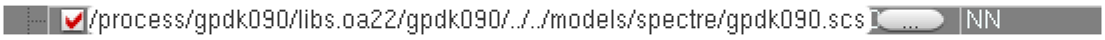
2. Set up the model libraries by executing **Setup > Model** Libraries. **'Model Library Setup'** Window will appear:



spectre0: Model Library Setup

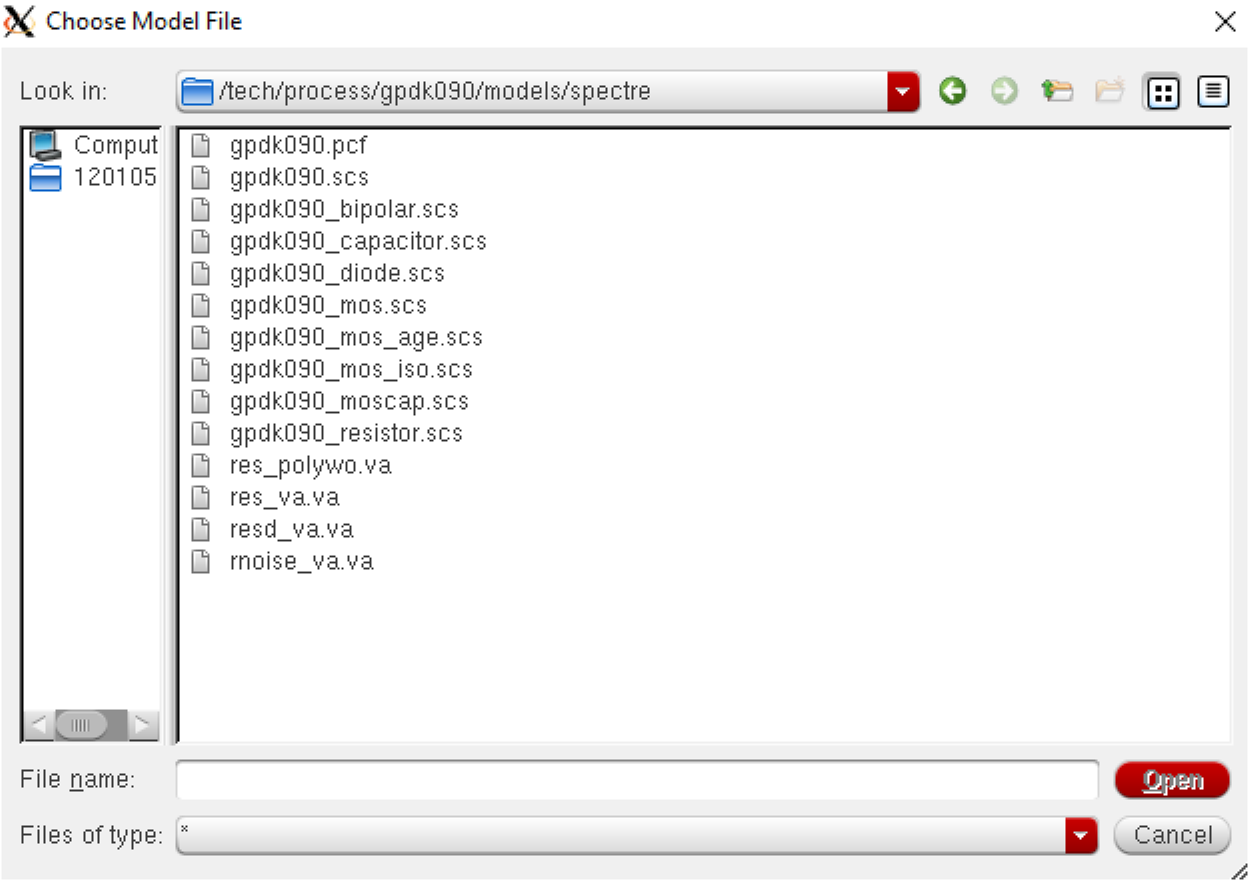
Model File	Section
Global Model Files	
☑ /tech/process/gpdk090/libs.oa22/gpdk090/../../models/spectre/gpdk090.scs	NN
□ <Click here to add model file>	

3. Click twice on the file name given under **Global Model Files**. An ash-colored button will appear.



☑ /process/gpdk090/libs.oa22/gpdk090/../../models/spectre/gpdk090.scs ... NN

Click on the button. **'Choose Model File'** window will appear.



Choose Model File

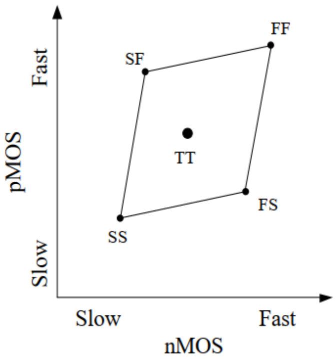
Look in: /tech/process/gpdk090/models/spectre

- gpdk090.pcf
- gpdk090.scs
- gpdk090_bipolar.scs
- gpdk090_capacitor.scs
- gpdk090_diode.scs
- gpdk090_mos.scs
- gpdk090_mos_age.scs
- gpdk090_mos_iso.scs
- gpdk090_moscap.scs
- gpdk090_resistor.scs
- res_polywo.va
- res_va.va
- resd_va.va
- rnoise_va.va

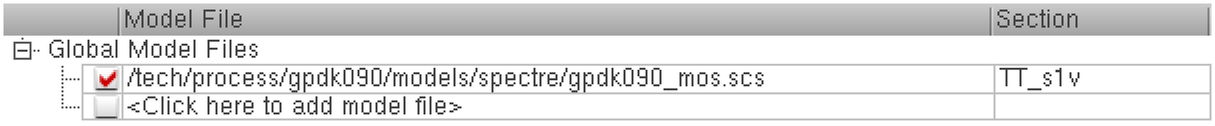
File name:

Files of type: *

4. Select **gpdk090_mos.scs** from the list. Click **Open**.
   In this model file, there are models to simulate various corners like fast-fast (FF), fast-slow (FS), typical-typical (TT) etc. These are called process corners, depending on the speed of MOS transistors (NMOS and PMOS). Refer to the following figure for the definition of process corners:

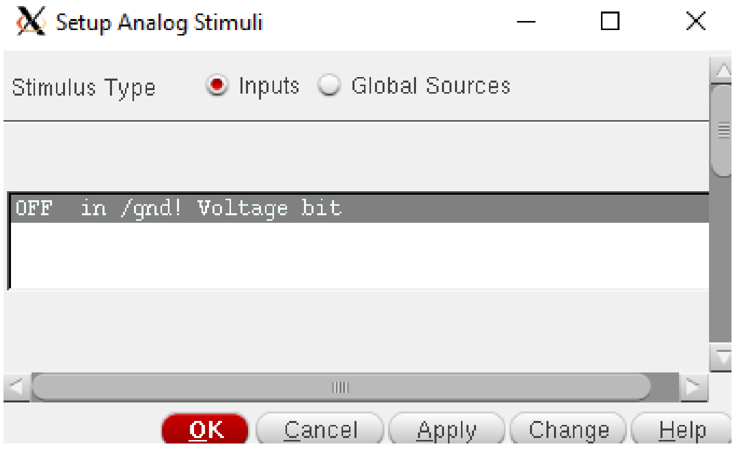


We will choose the section typical from the **Section** scroll bar and select the section **'TT_s1v'**. These will enable us to use the TT models of the 1.2 V MOS transistors. Only one **Global Model File** will be defined. Uncheck or delete any other model files that appear. Click **OK**.



Model File	Section
Global Model Files	
☑ /tech/process/gpdk090/models/spectre/gpdk090_mos.scs	TT_s1v
□ <Click here to add model file>	

5. Now execute **Setup** > **Stimuli** to assign signals to pins of the inverter.

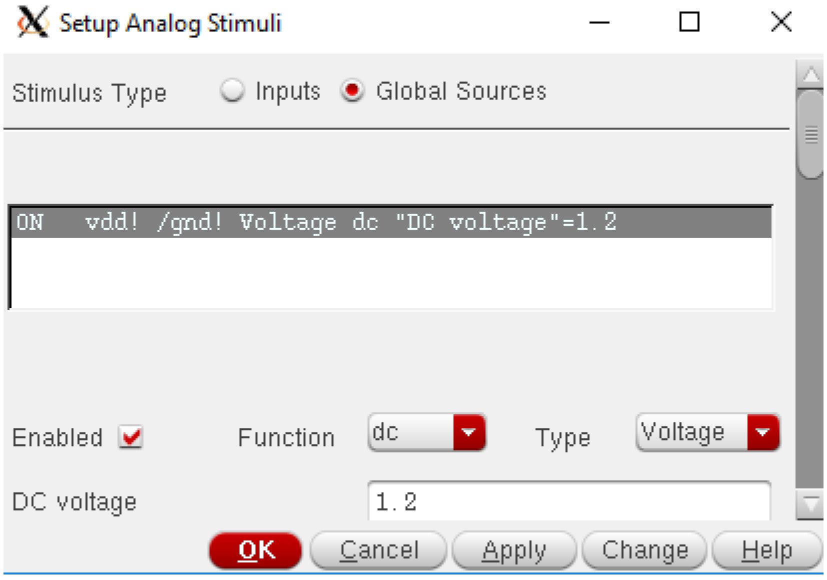


Setup Analog Stimuli

Stimulus Type ● Inputs ○ Global Sources

```
OFF  in /gnd! Voltage bit
```

6. In **'Setup Analog Stimuli'** window, select Global Sources. Now you can see global power net **vdd!**. Click on **Enabled**, Select **dc** under **Function** and **Voltage** under **Type**. Put a value of 1.2 on the **DC voltage** box. The filled-up form for **'vdd!'** will look like the one below. Click **Apply** (clicking **OK** will close the window and it will have to be reopened to setup inputs).



Setup Analog Stimuli

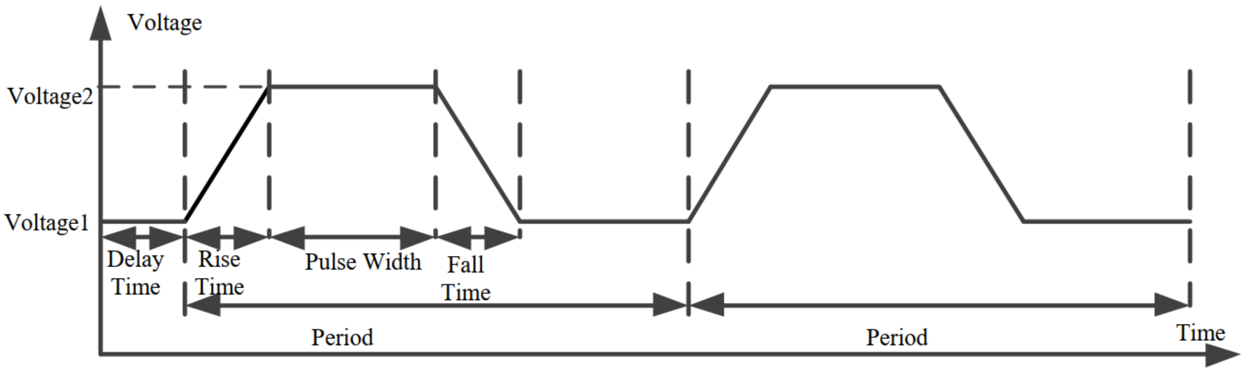
Stimulus Type ○ Inputs ● Global Sources

```
ON   vdd! /gnd! Voltage dc "DC voltage"=1.2
```

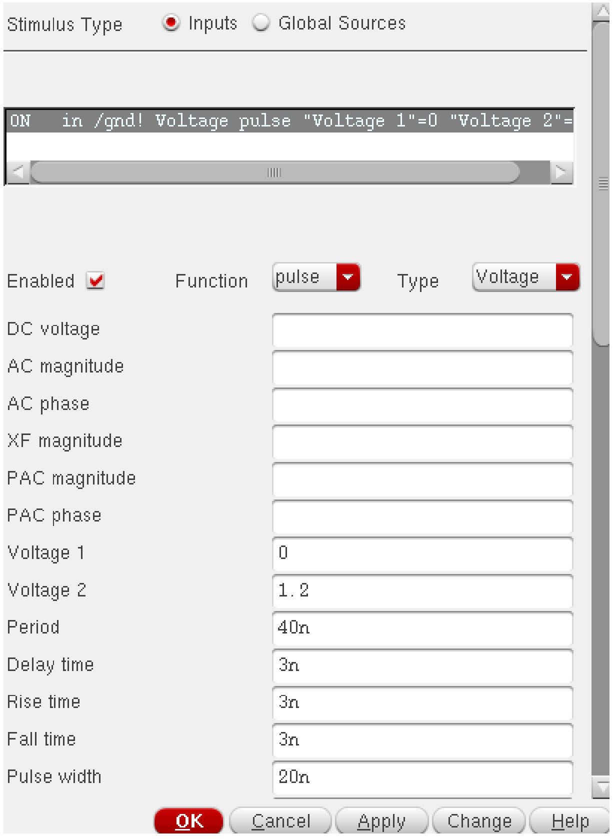
Enabled ☑ Function dc Type Voltage

DC voltage 1.2

7. Select **Inputs**. For input pin **'in'**, we have to set a pulse waveform. The following figure shows the definition of pulse parameters:



For setting signal to input pin **'in'**, select **Inputs** in **Setup Analog Stimuli** window. Click on **'Enabled'**, select **Function**: 'pulse', **Type**: 'Voltage'. Parameters for pulse source will be as follows: **Voltage1** = 0V, **Voltage2** = 1.2V, **Period** = 40n, **Delay** time = 3n, **Rise time** = 3n, **Fall time** = 3n, **Pulse width** = 20n. Click **Apply** and then click **OK**. (**Delay**, **Rise time** and **Fall time** can also be set at ps ranges for sharp transitions).



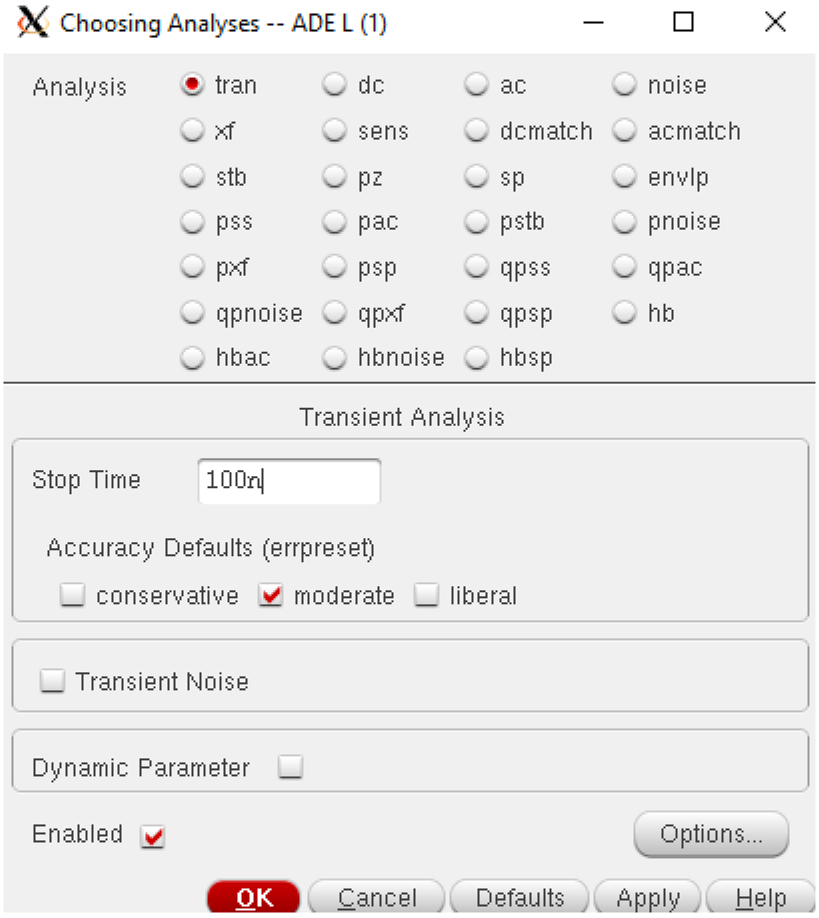
Stimulus Type ● Inputs ○ Global Sources

```
ON   in /gnd! Voltage pulse "Voltage 1"=0 "Voltage 2"=
```

Enabled ☑ Function pulse Type Voltage

DC voltage	
AC magnitude	
AC phase	
XF magnitude	
PAC magnitude	
PAC phase	
Voltage 1	0
Voltage 2	1.2
Period	40n
Delay time	3n
Rise time	3n
Fall time	3n
Pulse width	20n

8. Now choose the analysis to be done from **Analyses** > **Choose**. Select transient (**tran**) analysis to be done. Provide a reasonable value for ‘stop time’ to observe few periods of signals. (e.g., **Analysis**: tran, **Stop Time**: 100n, **Accuracy Defaults**: moderate). Click **OK**.



Choosing Analyses -- ADE L (1)

Analysis ● tran ○ dc ○ ac ○ noise ○ xf ○ sens ○ dcmatch ○ acmatch ○ stb ○ pz ○ sp ○ envlp ○ pss ○ pac ○ pstb ○ pnoise ○ pxf ○ psp ○ qpss ○ qpac ○ qpnoise ○ qpxf ○ qpsp ○ hb ○ hbac ○ hbnoise ○ hbsp

Transient Analysis

Stop Time 100n

Accuracy Defaults (errpreset)

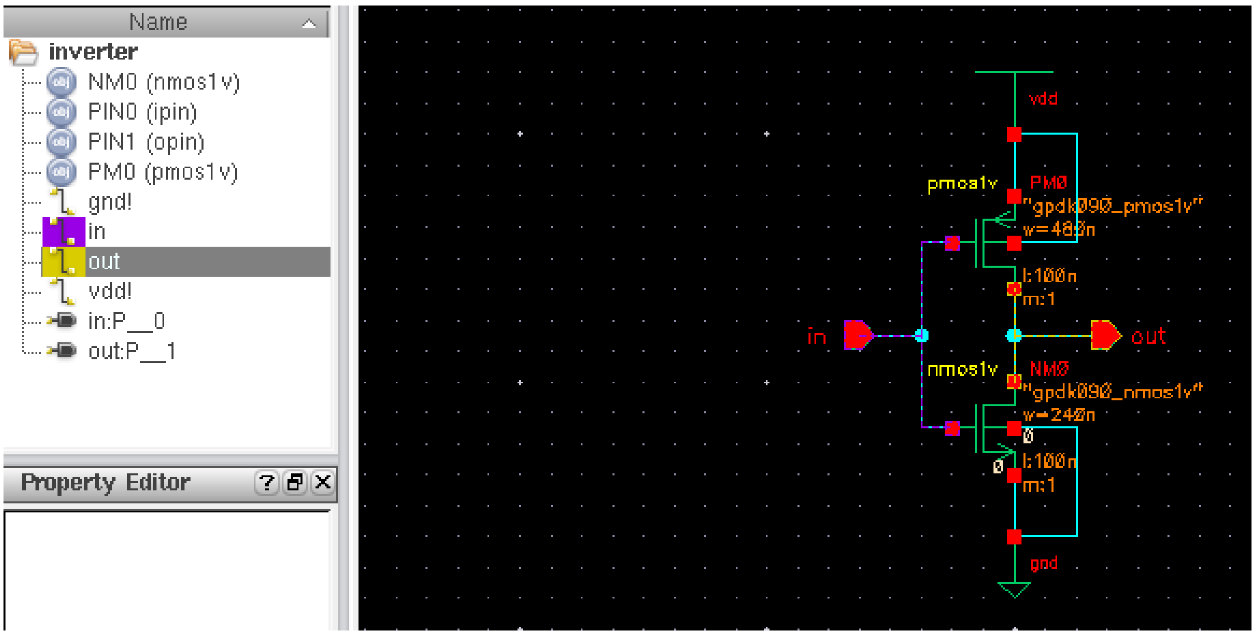
□ conservative ☑ moderate □ liberal

□ Transient Noise

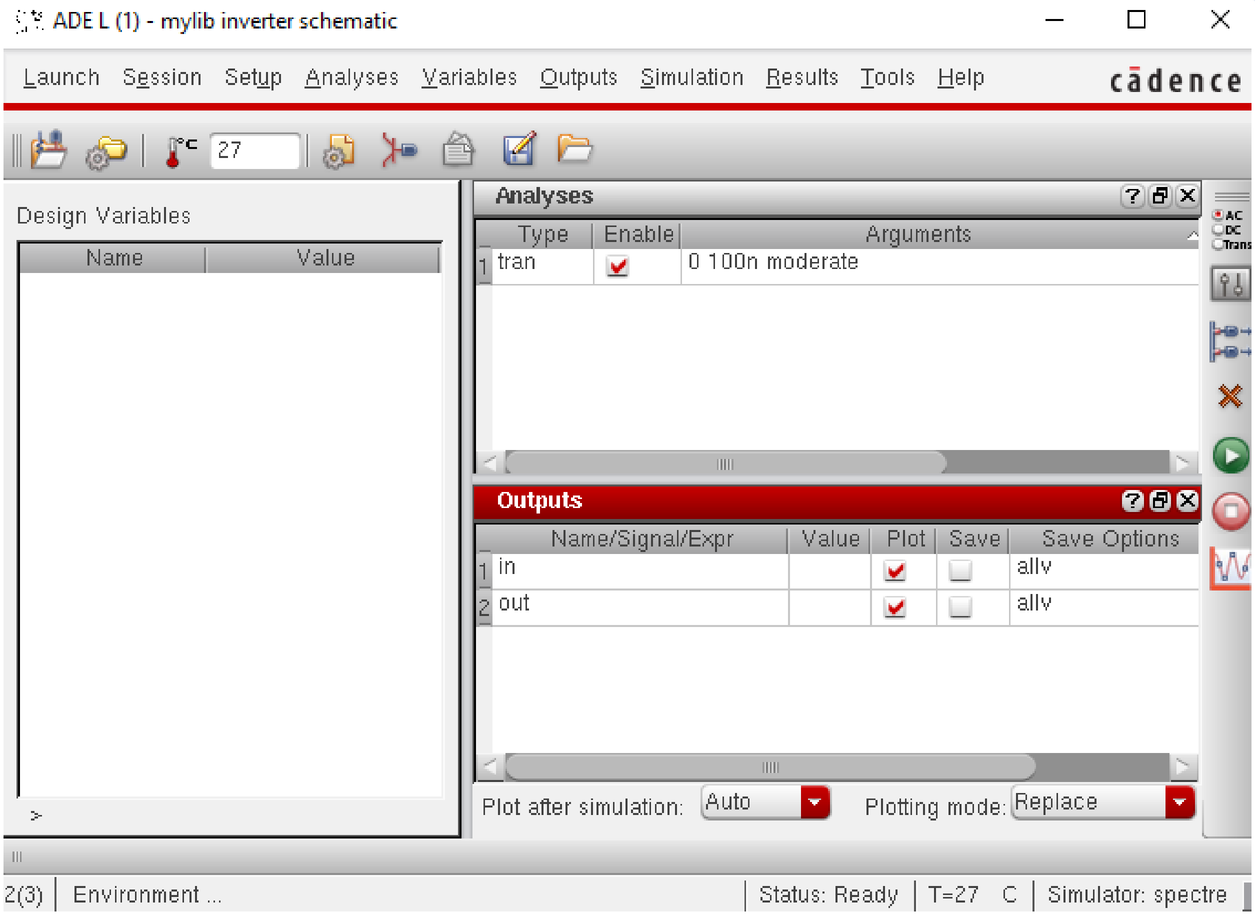
Dynamic Parameter □

Enabled ☑

9. Select the output to be plotted by executing **Outputs** > **To be plotted** > **Select on Design** in the ADE window. Schematic editor window will pop up, select ‘out’ and ‘in’ by clicking on the pins/terminals or selecting from the list on the left-hand side as shown in the figure below. When you select them, you will see colors being assigned to these pins.



10. Your Analog Design Environment window should now look like the following:



ADE L (1) - mylib inverter schematic

Launch Session Setup Analyses Variables Outputs Simulation Results Tools Help
cādence
27
Design Variables
Name
Value
Analyses
Outputs

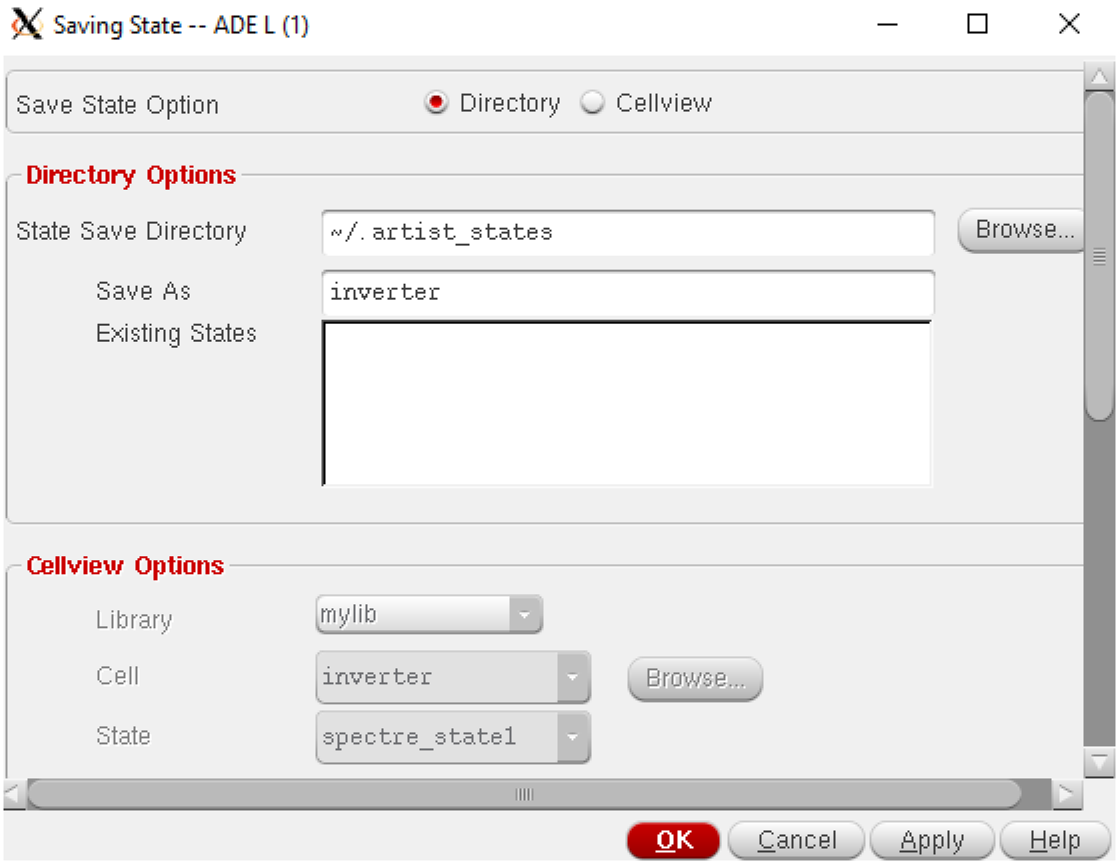
	Type	Enable	Arguments
1	tran	☑	0 100n moderate

	Name/Signal/Expr	Value	Plot	Save	Save Options
1	in		☑	□	allv
2	out		☑	□	allv

Plot after simulation: Auto   Plotting mode: Replace

2(3) | Environment ... | Status: Ready | T=27 C | Simulator: spectre

11. Before closing the Virtuoso Analog Design Environment window, it is a good idea to save design settings in a state file, so we can load it up next time. To do this, execute **Session** > **Save State** and save state name in the 'Save As' field as 'inverter'. Next time you run Cadence, you can simply load the simulation settings from this file by executing **Session** > **Load State**.



Saving State -- ADE L (1)

Save State Option ● Directory ○ Cellview

Directory Options

State Save Directory: ~/.artist_states

Save As: inverter

Existing States:

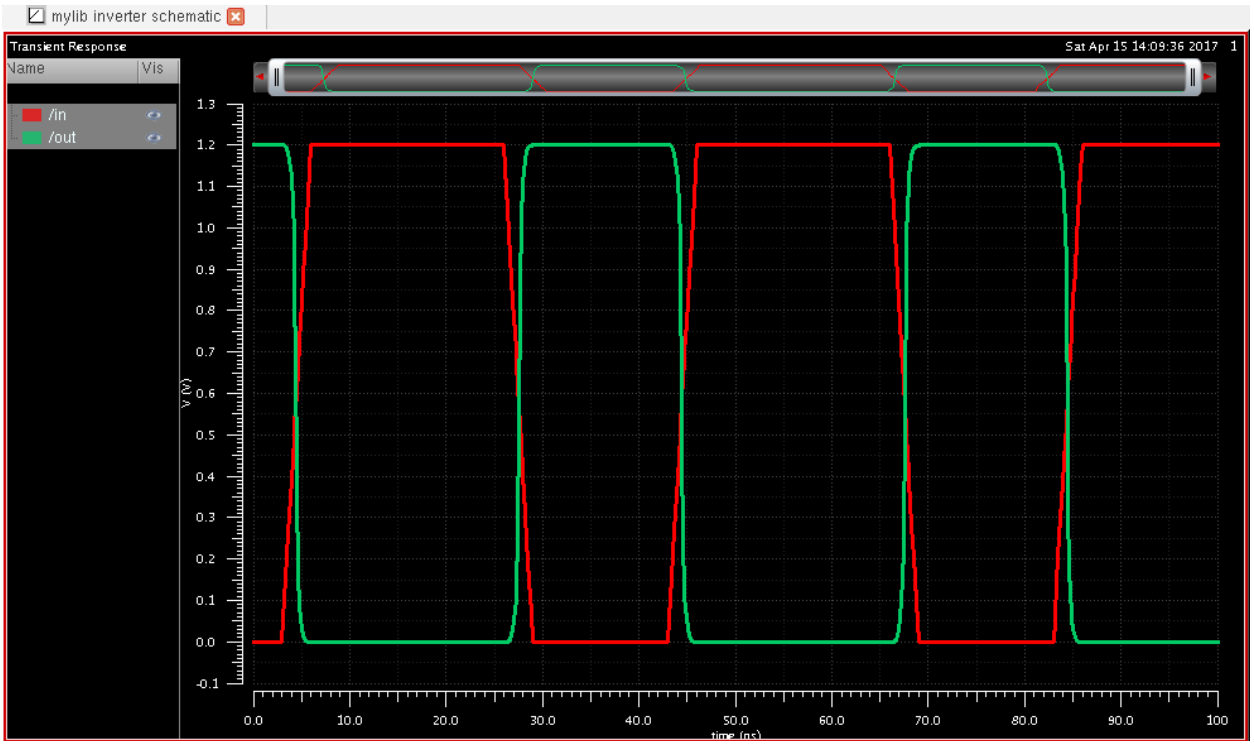
Cellview Options

Library: mylib

Cell: inverter

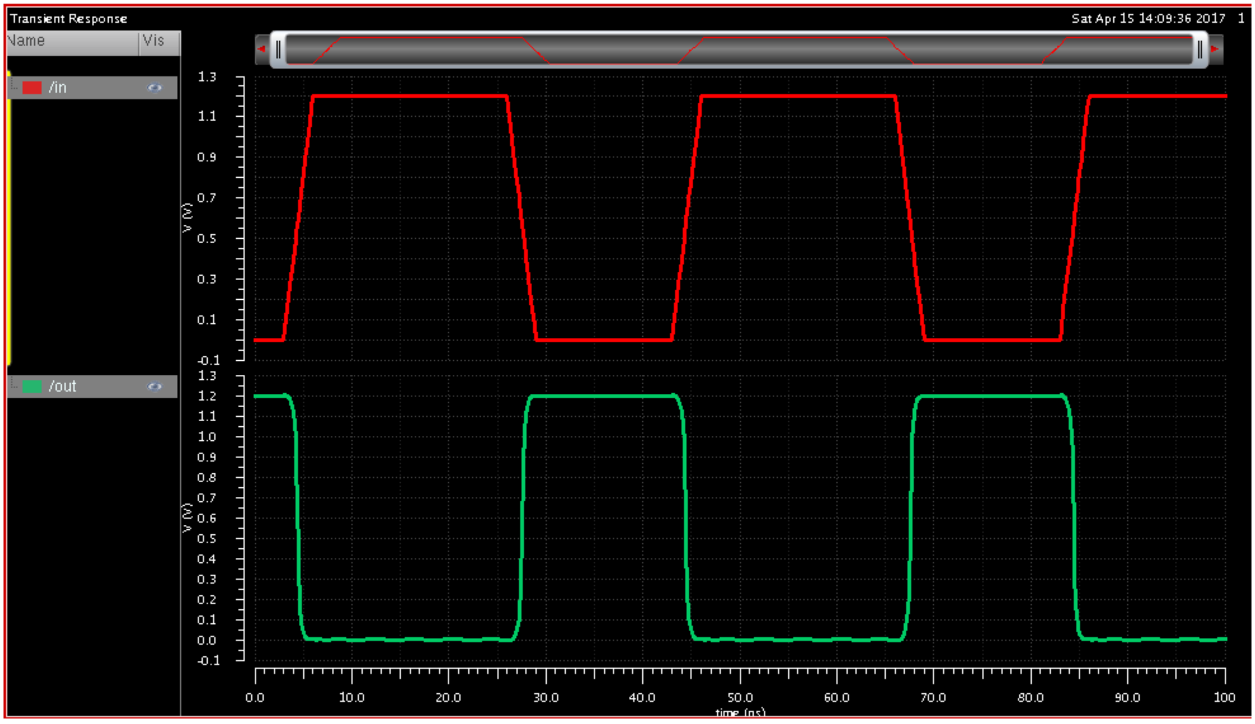
State: spectre_state1

12. Now run the simulation by executing **Simulation** > **Netlist and Run** in the ADE window. The simulation will run and the output will appear in Virtuoso Visualization & Analysis XL window as shown below.



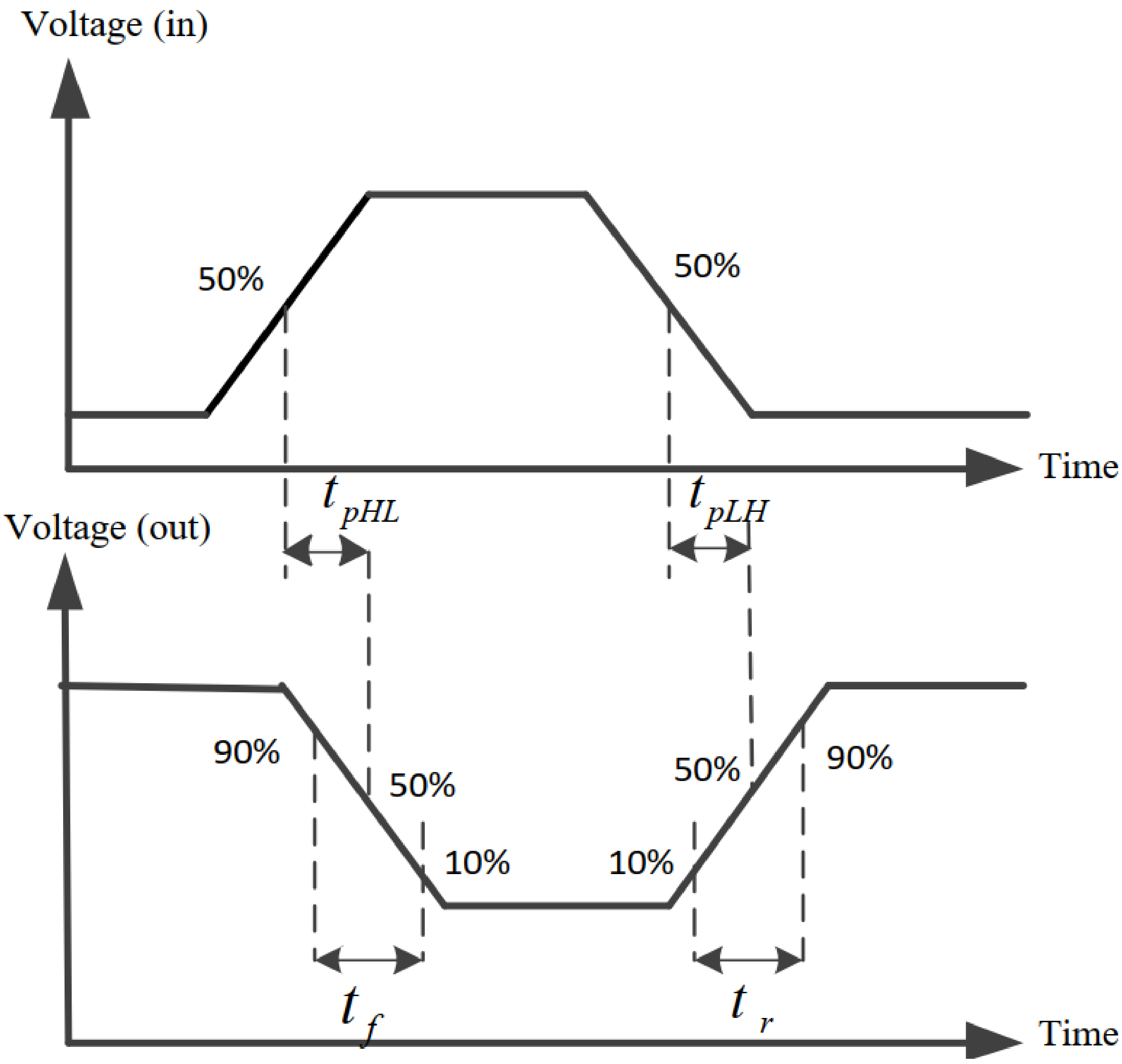
13. Finally, we are going to separate the plots into two sub-graphs. Click on the following icon for splitting graphs.

The final plot should look like the one shown below:



Definitions:

Three main timing parameters are associated with CMOS devices – rise time, fall time, and propagation delay. Most often, in discussion with regard to these parameters, the system response of an inverter is used. The following figure defines rise time, fall time and propagation delay of a gate with the example of an inverter:



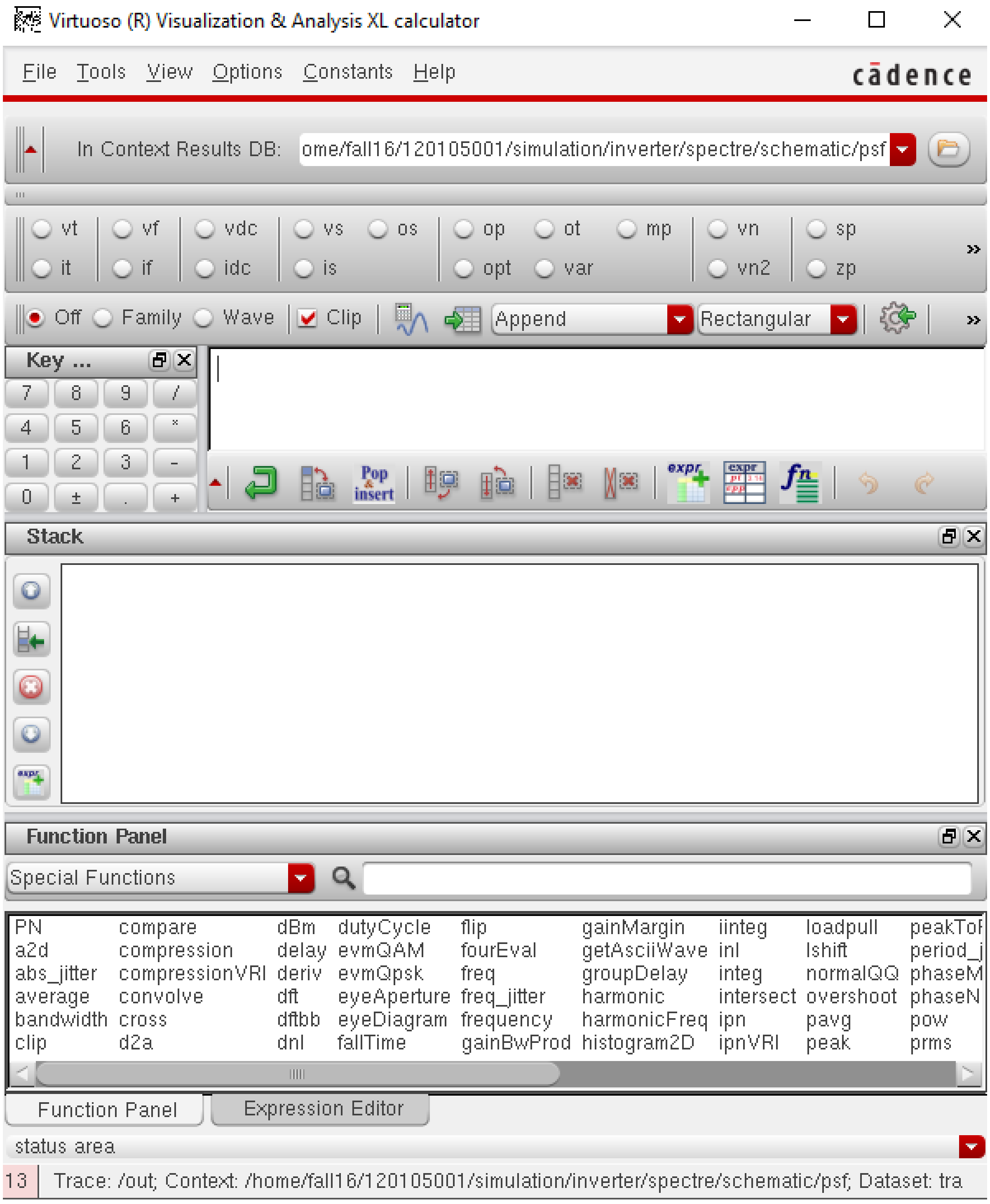
Referring to the above figure, rise time is the time that it takes to charge the output capacitive load. Fall time is the time it takes for the output capacitive load to discharge. The rise and fall time are usually measured from 10% to 90% and from 90% to 10% of the steady state value of a waveform, respectively.

Propagation delay is the time difference between approximately 50% of the input transition and approximately 50% of the output transition.

Measurement:

Waveform calculator can be used to perform many different measurements and transformations on the waveforms displayed in the waveform window. This includes – computing the average of a waveform (e.g., power) over the entire length of the simulation or in a given period of time, finding the propagation delay of between input and output signals, or addition / subtraction / multiplication / division of waveforms, etc.

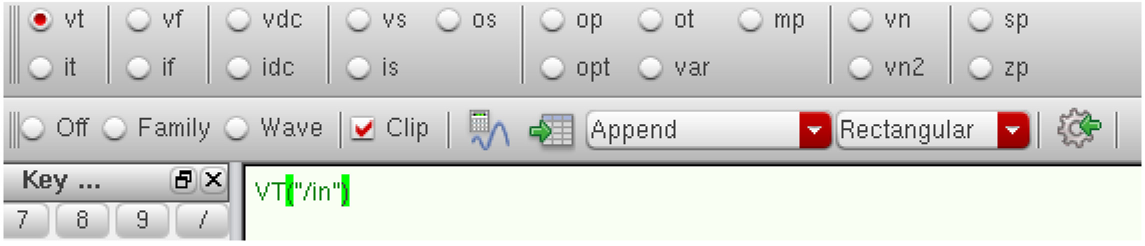
1. Execute **Tools** > **Calculator** in Virtuoso Visualization & Analysis XL window. **‘Virtuoso Visualization & Analysis XL calculator’** window will pop-up:


Virtuoso (R) Visualization & Analysis XL calculator
File Tools View Options Constants Help
cādence
In Context Results DB:
vt vf vdc vs os op ot mp vn sp
it if idc is opt var vn2 zp
Off Family Wave Clip Append Rectangular
Key ...
7 8 9 /
4 5 6 *
1 2 3 -
0 ± . +
Pop & insert
Stack
Function Panel
Special Functions
PN compare dBm dutyCycle flip gainMargin iinteg loadpull peakToF
a2d compression delay evmQAM fourEval getAsciiWave inl lshift period_j
abs_jitter compressionVRI deriv evmQpsk freq groupDelay integ normalQQ phaseM
average convolve dft eyeAperture freq_jitter harmonic intersect overshoot phaseN
bandwidth cross dftbb eyeDiagram frequency harmonicFreq ipn pavg pow
clip d2a dnl fallTime gainBwProd histogram2D ipnVRI peak prms
Function Panel
Expression Editor
status area

In Context Results DB: ome/fall16/120105001/simulation/inverter/spectre/schematic/psf

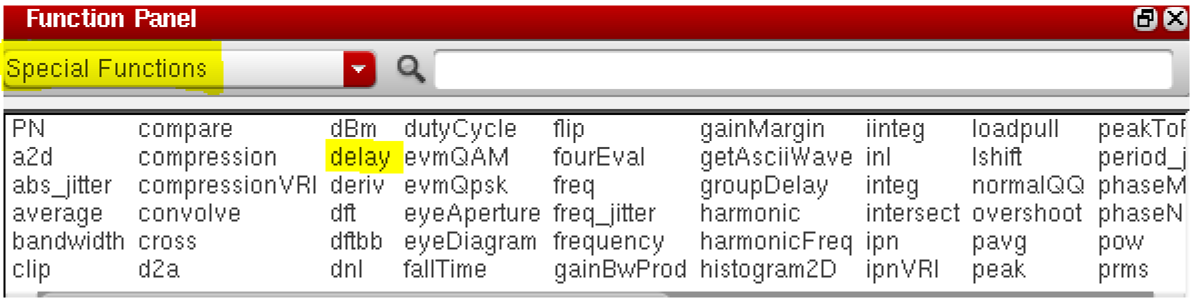
13 Trace: /out; Context: /home/fall16/120105001/simulation/inverter/spectre/schematic/psf; Dataset: tra

2. Select **'vt'**. Go to Schematic editor window and click on input node 'in'. An expression (e.g., **VT("/in")**) will appear. Copy the expression.

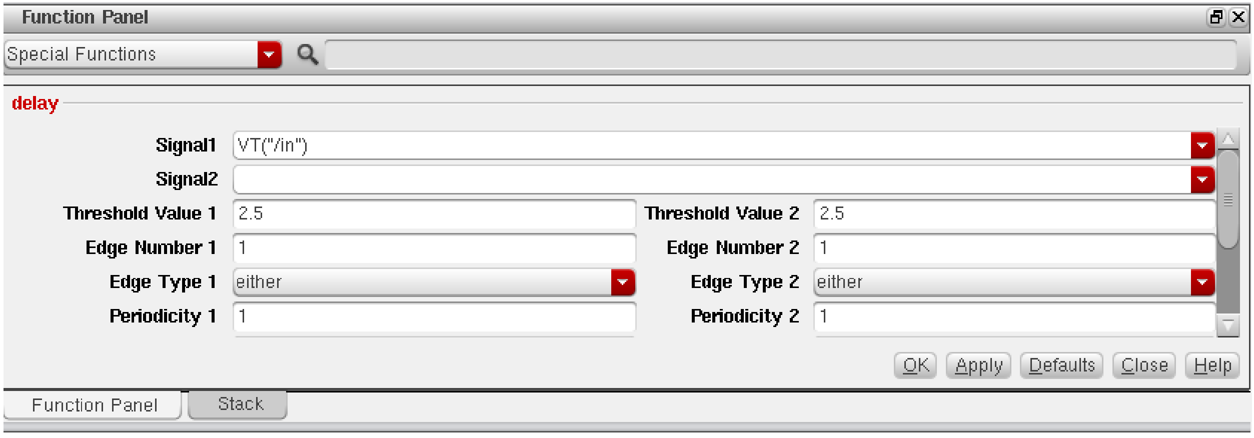

vt vf vdc vs os op ot mp vn sp
it if idc is opt var vn2 zp
Off Family Wave Clip Append Rectangular
Key ...
7 8 9 /

VT("/in")

3. In the **Function Panel**, select **'Special functions'** and select **'delay'**.


Function Panel
Special Functions
PN compare dBm dutyCycle flip gainMargin iinteg loadpull peakToF
a2d compression delay evmQAM fourEval getAsciiWave inl lshift period_j
abs_jitter compressionVRI deriv evmQpsk freq groupDelay integ normalQQ phaseM
average convolve dft eyeAperture freq_jitter harmonic intersect overshoot phaseN
bandwidth cross dftbb eyeDiagram frequency harmonicFreq ipn pavg pow
clip d2a dnl fallTime gainBwProd histogram2D ipnVRI peak prms

4. The following window will appear. Put the expression previously obtained in the field **'Signal1'**. Do the same for output signal **'out'** to fill in the field **'Signal2'**.



Function Panel

Special Functions

delay

Signal1: VT("/in")

Signal2:

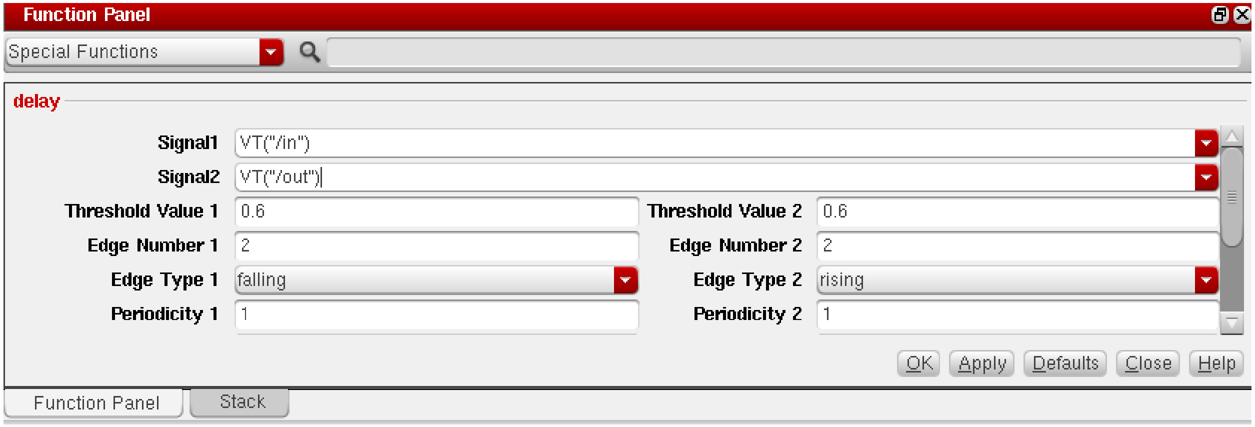
Threshold Value 1: 2.5 | Threshold Value 2: 2.5

Edge Number 1: 1 | Edge Number 2: 1

Edge Type 1: either | Edge Type 2: either

Periodicity 1: 1 | Periodicity 2: 1

Fill up the rest as follows:



Function Panel

Special Functions

delay

Signal1: VT("/in")

Signal2: VT("/out")

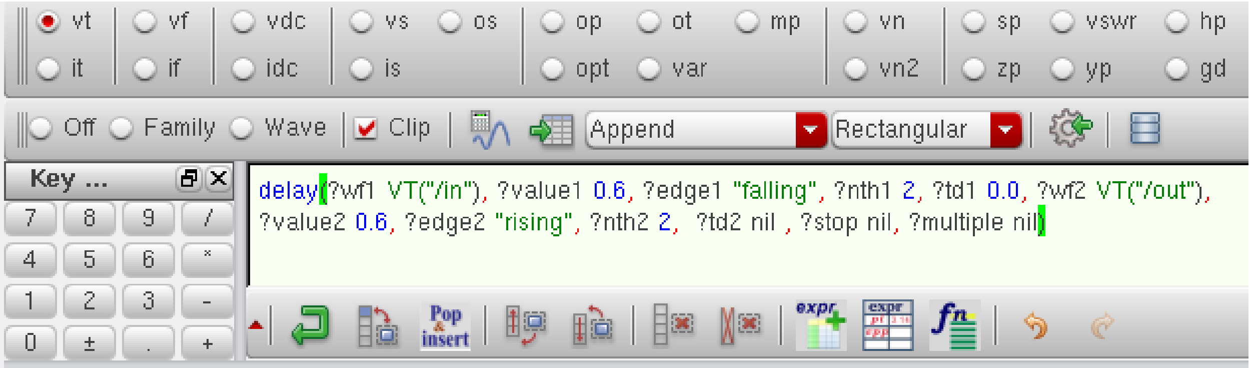
Threshold Value 1: 0.6 | Threshold Value 2: 0.6

Edge Number 1: 2 | Edge Number 2: 2

Edge Type 1: falling | Edge Type 2: rising

Periodicity 1: 1 | Periodicity 2: 1

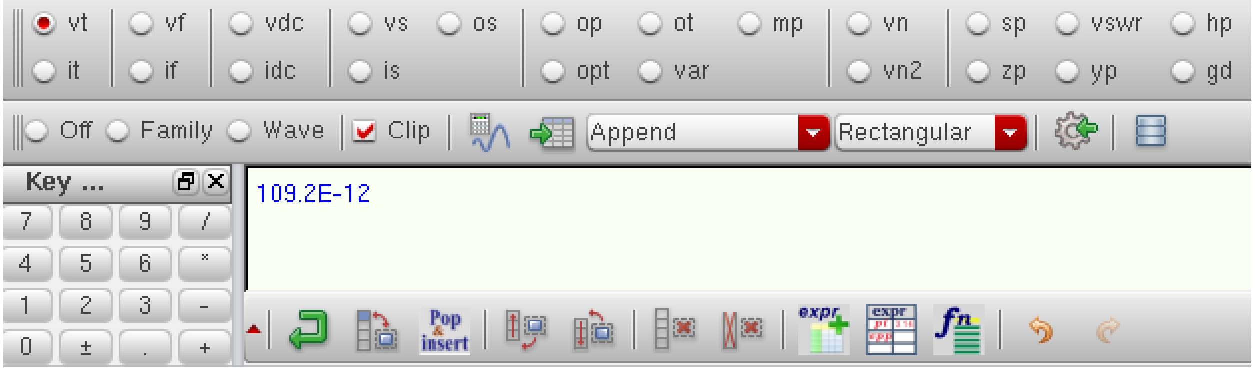
5. Click **OK**. The following expression should appear:



● vt ○ vf ○ vdc ○ vs ○ os ○ op ○ ot ○ mp ○ vn ○ sp ○ vswr ○ hp
○ it ○ if ○ idc ○ is ○ opt ○ var ○ vn2 ○ zp ○ yp ○ gd
○ Off ○ Family ○ Wave ☑ Clip
Append
Rectangular
Key ...
7 8 9 /
4 5 6 *
1 2 3 -
0 ± . +
Pop & insert
expr
fn

```
delay(?wf1 VT("/in"), ?value1 0.6, ?edge1 "falling", ?nth1 2, ?td1 0.0, ?wf2 VT("/out"),
?value2 0.6, ?edge2 "rising", ?nth2 2,  ?td2 nil , ?stop nil, ?multiple nil)
```

6. Click on Evaluate the buffer icon.
The propagation delay (in seconds) will be displayed in the window.



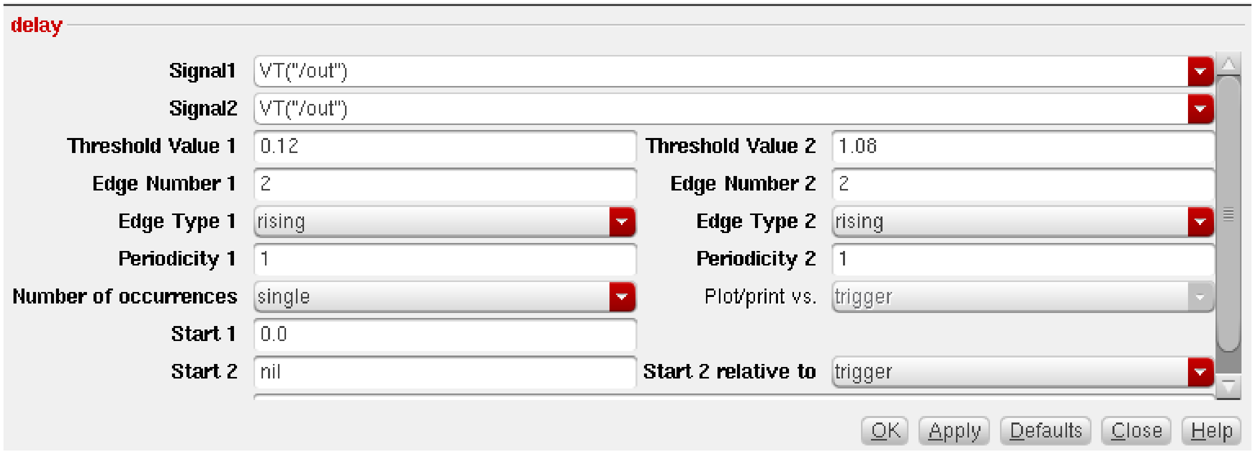
● vt ○ vf ○ vdc ○ vs ○ os ○ op ○ ot ○ mp ○ vn ○ sp ○ vswr ○ hp
○ it ○ if ○ idc ○ is ○ opt ○ var ○ vn2 ○ zp ○ yp ○ gd
○ Off ○ Family ○ Wave ☑ Clip
Append
Rectangular
Key ...
7 8 9 /
4 5 6 *
1 2 3 -
0 ± . +
Pop & insert
expr
fn

```
109.2E-12
```

Waveform Calculation:

1. Open the 'delay' function window under **Waveform calculator** in the same way that you followed for propagation delay measurement. This time both **Signal1** and **Signal2** will be **VT("/out").**

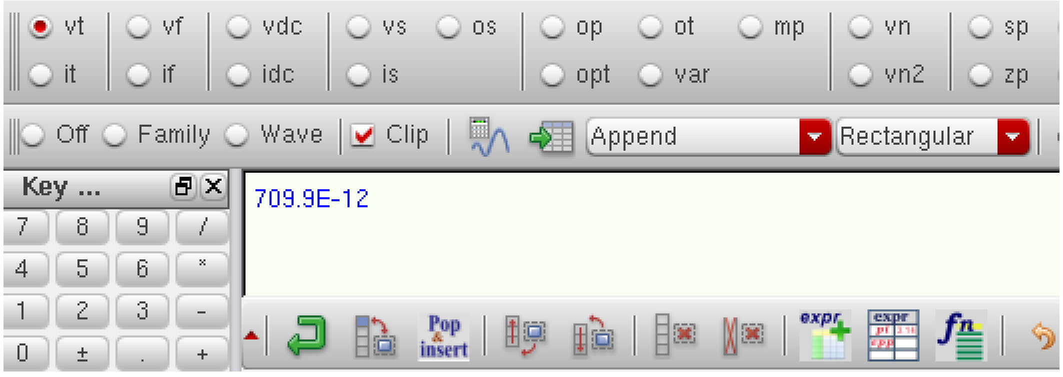
2. **Threshold** value 1 and 2 should be **0.12** (10% of 1.2 V supply) and **1.08** (90% of 1.2 V supply) respectively for 10% to 90% rise time calculation. These values should be swapped for fall time calculation.
3. For rise time/fall time calculation, both the **Edge numbers** must be the same.
4. The **Edge types** should be rising for rise time calculation and falling for fall time calculation. Example: Rise time calculation of rising edge 2 for an inverter:



delay

Signal1: VT("/out")
Signal2: VT("/out")
Threshold Value 1: 0.12 | Threshold Value 2: 1.08
Edge Number 1: 2 | Edge Number 2: 2
Edge Type 1: rising | Edge Type 2: rising
Periodicity 1: 1 | Periodicity 2: 1
Number of occurrences: single | Plot/print vs.: trigger
Start 1: 0.0
Start 2: nil | Start 2 relative to: trigger

5. Click **OK** after filling in the form as shown above. The following expression should appear:



● vt ○ vf ○ vdc ○ vs ○ os ○ op ○ ot ○ mp ○ vn ○ sp
○ it ○ if ○ idc ○ is ○ opt ○ var ○ vn2 ○ zp

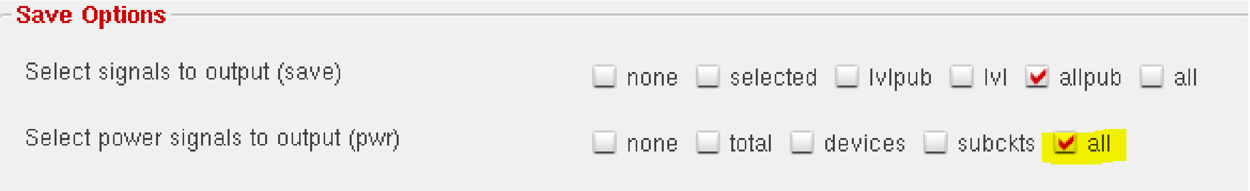
○ Off ○ Family ○ Wave ☑ Clip | Append | Rectangular

709.9E-12

Power Measurement:

In this tutorial, we will compute the average power consumed in a circuit for the duration of transient simulation window.

1. To do this, make sure that before running simulation you select the Outputs > Save All option in ADE L window. 'Save Options' window will appear. Under 'Select power signals to output (pwr)' option, put a tick mark in all option. Click OK.

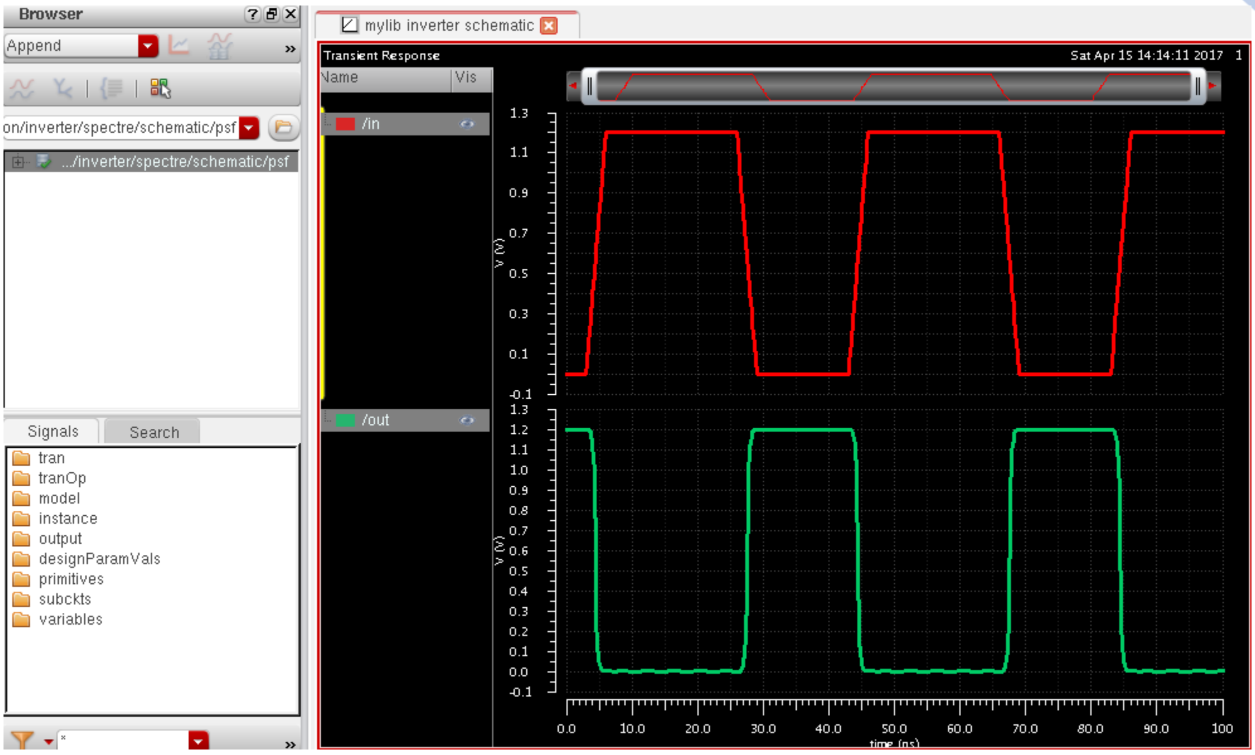


Save Options

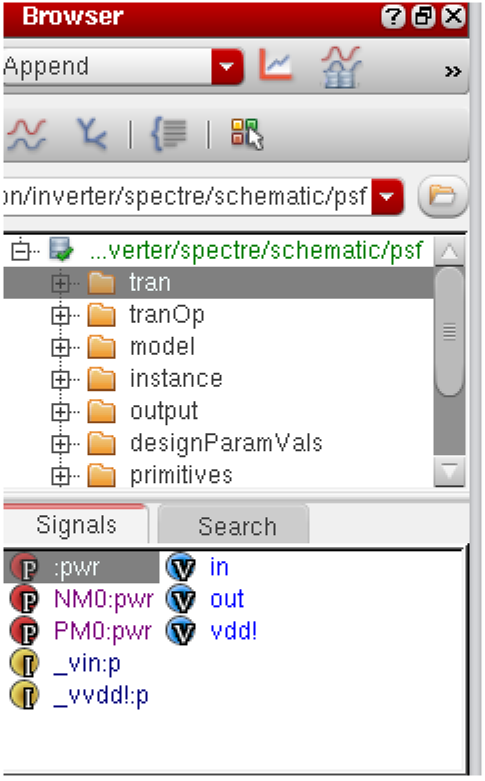
Select signals to output (save) □ none □ selected □ lvlpub □ lvl ☑ allpub □ all

Select power signals to output (pwr) □ none □ total □ devices □ subckts ☑ all

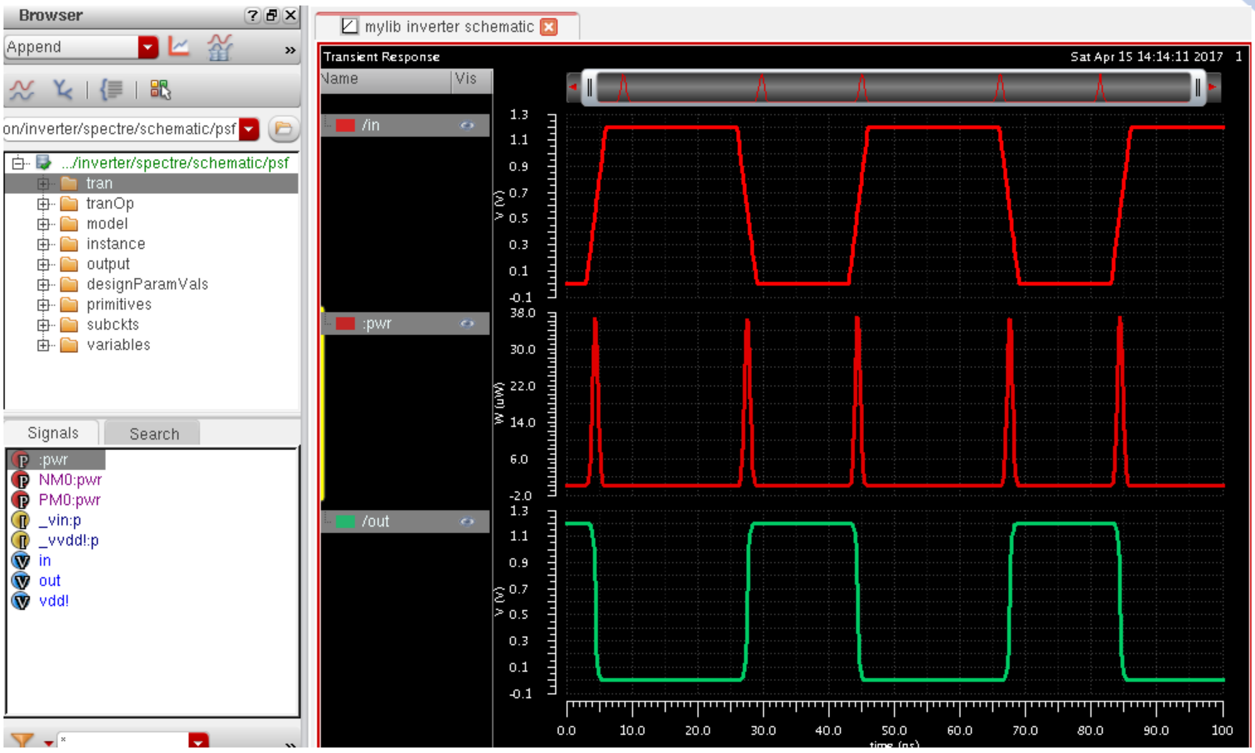
2. Then simulate the circuit as usual, by executing Simulation > Netlist and Run. Execute Tools > Result Browser in ADE L window. 'Result Browser' window will appear on the left side in 'Virtuoso Analysis and Visualization XL' window.



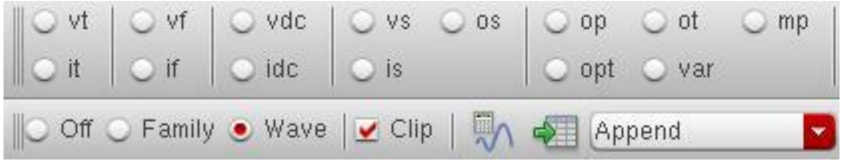
3. Double-click on tran. From the signals list, double-click on :pwr



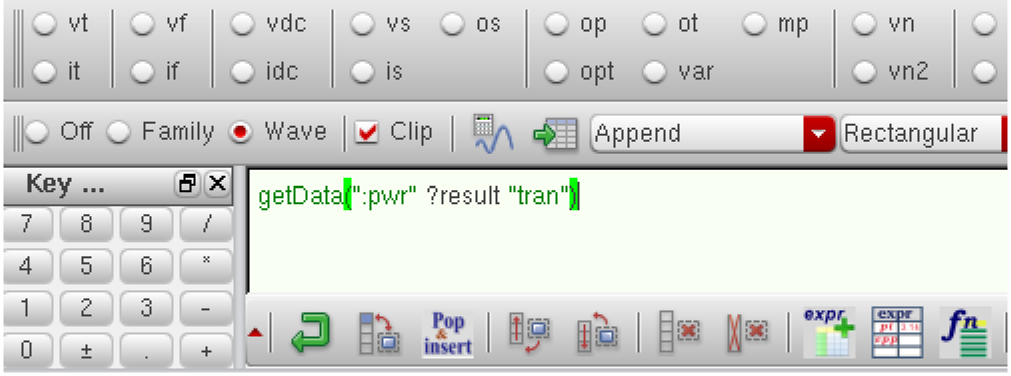
4. The waveform display window will show the ":pwr" (the instantaneous power consumed by the whole circuit) along with 'in' and 'out' signals.



5. Now, open Waveform calculator window. The calculator window appears. Make sure the "Wave" and "Clip" options are selected.

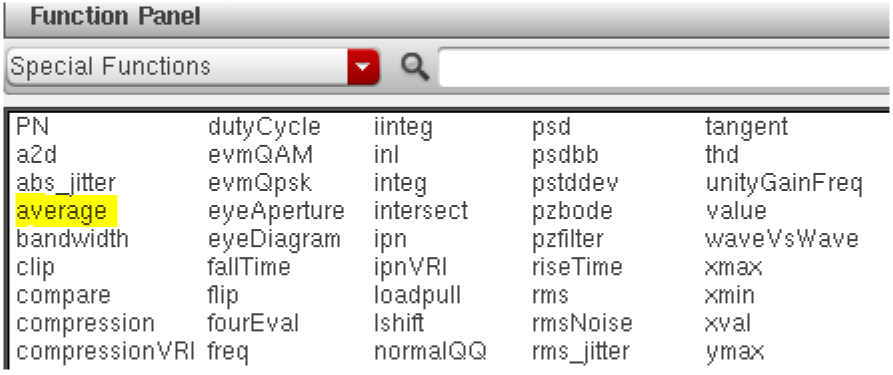


6. Now switch back to the waveform window and left click the mouse once on the power waveform. Then switch back to the calculator window. The buffer window should be filled in as follows:

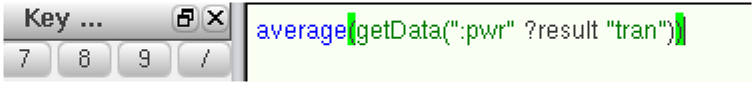


```
getData(":pwr" ?result "tran")
```

7. Now select 'average' from 'Special Functions' Menu.

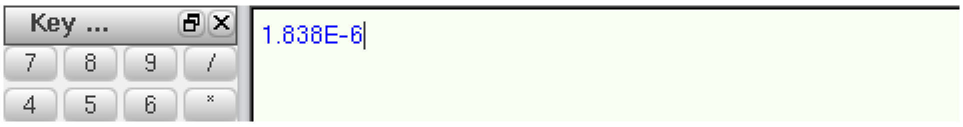


8. The buffer will now look like the following one:



```
average(getData(":pwr" ?result "tran"))
```

9. Click on Evaluate the buffer icon and the average power dissipation in that time window will be displayed (about 1.838 μW in this example).



```
1.838E-6
```

Rubrics:

Criteria	Below Average (1)	Average (2)	Good (3)
Formatting	Report is not properly formatted, missing objectives, discussions, and references.	Report is somewhat formatted but missing proper references.	Report is properly formatted.
Design	Design steps are somewhat followed, and results have errors.	Design steps are properly followed, and the results are flawless.	
Writing	Writing is poor and not informative. It does not address the design decisions and contains high plagiarism.	Writing is average and original. Design decisions are somewhat explored.	Writing is excellent with every design decision adequately explored.
Diagram	Diagrams are of bad quality and unreadable.	Diagrams are clear and of high quality.	