

# Ahsanullah University of Science and Technology Department of Electrical and Electronic Engineering

# LABORATORY MANUAL FOR ELECTRICAL AND ELECTRONIC SESSIONAL COURSES

Student Name : Student ID :

> Course No. : EEE 4134 Course Title : VLSI I Lab.

For the students of Department of Electrical and Electronic Engineering 4<sup>th</sup> Year, 1<sup>st</sup> Semester

# **Contents**

Lab	Title	Page
<u>No.</u> 0	Introductory Lab: Logging into Cadence Server, Tool Setup, Cell Library Creation, Introduction to Custom IC Design flow	1
1	Introduction to Virtuoso Schematic Editor, Creating Inverter schematic, Performing transient simulation of Inverter schematic, Power and delay measurement of designed inverter for different process corners	8
2	DC sweep, Parametric sweep and Symbol creation of inverter	29
3	Layout of an Inverter using Virtuoso L	40
4	DRC, LVS, RCX and Post-layout simulation of an inverter	60
5	Schematic Driven Layout of a 2-input NAND gate using Virtuoso Layout Suite Editor XL	70
6	Introduction to Hierarchical Design (2-input AND gate using 2-input NAND gate and an inverter)	81
7	Introduction to Verilog HDL and Quartus II	85
8	Combinational Logic circuit design in Verilog HDL using Quartus II	94
9	RTL synthesis and Sequential Logic Circuit design in Verilog HDL using Quartus II	97
	References and Further Readings	100

### EEE 4134 VLSI I Laboratory Lab 0 (Introductory Lab) Logging into Cadence Server, Tool Setup, Cell Library Creation, Introduction to Custom IC Design flow

#### **Objectives:**

- To login, start a shell tool and start the Cadence Virtuoso software
- To learn about PDK and add the PDK library to the Library Manager
- To create a working library and to get familiar with technology
- To be familiar with Custom IC Design flow

#### Logging in, starting a shell tool, and starting the Cadence Tool Suite

**1.** Find Desktop shortcut icon for **XLaunch**. Double-click on it. Click **Next**, **Next**, **Next**, **Finish** (*in that order*) in the windows that pop-up one after another.



After it starts, you will see the **Xming** icon at the bottom right corner of your Desktop screen.



2. Find icon for Putty. Double click on it to open it. 'Putty Configuration' window will pop-up.



**3.** Select **VLSI\_LAB** under '**Saved Sessions**' category. Click **Load**. The window will look like the following one:



Click Open. A Security Alert window may pop-up. Click Yes.

4. Now you will see a Terminal window which prompts you for login.



**5.** Log in to your workstation using user ID and password. Your user name and your password will be *your student ID*. When you are typing your password, the command window will not display the characters you type in, so make sure you are typing the right password. After logging in to your account, Terminal window should look like the following:



6. Type **csh** and press 'Enter' key.

Then type **source cshrc\_q** and press '**Enter**' key.

The following message will be displayed in the Terminal window:

#### Welcome to Cadence tools Suite

That means you can use Cadence tools now.

7. Go to your working directory by typing: cd cic

#### 8. Type virtuoso&

A sample command prompt screen is shown below:



**9.** Virtuoso® **Command Interpreter Window** (**CIW**) appears at the bottom of the screen. From the CIW menus, all Cadence main tools, online help and options can be accessed. In the window area, all kind of messages (info, errors, warnings, etc) generated by the different Cadence tools appear. You can also introduce commands.

〔 Virtuoso ◎ 6.1.6 - Log: /home/fall16/120105001/CDS.log	_		×
<u>F</u> ile <u>T</u> ools <u>O</u> ptions <u>H</u> elp		cād	lence
Loading ci.cxt Loading ams.cxt Virtuoso Framework License (111) was checked out successfully. Total checkout time w	as O	. 05s.	
Immouse L: M:			R:
1 >			

Another window 'What's new in IC6.1.6 Overview' appears too. Execute *File* →*Close and Do Not Show Again* and this window will not appear the next time you open Virtuoso.

NOTE: You have to perform these steps above in every class where Cadence tools will be used. The following figure summarizes the steps:



#### **Custom IC Design Flow**

The following figure 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.

#### Learning fundamentals of PDK and adding PDK library to the Library Manager

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.

#### Creating a library and attaching technology to 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.

🗙 New Library	- 🗆 X
Library Name mylib Directory (non-library directories)	Technology File Compile an ASCII technology file Reference existing technology libraries Attach to an existing technology library Do not need process information
/home/fall16/120105001/cic	Design Manager No design manager setup found
	Cancel <u>D</u> efaults <u>Apply</u> <u>H</u> elp

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

X Attach Library to Te	chnology Library	_		$\times$
New Library	mylib			
Technology Library	analogLib avTech basic cdsDefTechLib gpdk045 gpdk090 <u>OK</u> Cancel	Appl	y)(	

**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. In the CIW, the following message will appear:

Loaded gpdk180/libInit.il successfully! Created library "mylib" as "/home/fall16/120105001/cic/mylib" INFO (TECH-180011): Design library 'mylib' successfully attached to technology library 'gpdk090'.

×

### EEE 4134 VLSI I Laboratory Lab 1

### Introduction to Virtuoso Schematic Editor, Creating Inverter schematic, Performing transient simulation of Inverter schematic, Power and delay measurement of designed inverter for different process corners

#### **Objectives:**

- To learn how to draw schematic of basic logic gates in Cadence Virtuoso
- To learn how to perform transient simulation of logic gates
- To learn about process corners and their effects on delay and power dissipation
- To learn how to measure power dissipation and propagation delay of logic gates

#### Schematic Entry: Creating a Schematic cell view

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 **Command Interpreter Window** (**CIW**), execute *File* →*New* →*Cellview*. Set up the 'New **File**' form as follows:

**Library:** mylib, **Cell:** inverter, **View:** schematic, **Type:** schematic, **Application: Open with:** Schematics L



2. Click OK when done. The following window may appear. Click Yes/Always.



A blank schematic window for the inverter design appears.

🙀 Virtuoso® Schematic Editor L Editing: mylib inverter schematic						_			
Launch <u>F</u> ile <u>E</u> dit <u>V</u> iew <u>C</u> i	reate Chec	: <u>k</u> O <u>p</u> tions	<u>W</u> indow <u>H</u>	<u>H</u> elp					cādence
	0 🔟	× ①	T⁄ 🖻	5 ¢ 12	▼ TÎ T		ې 🔣 🔍	114	<u> </u>
	Basic						<b>Q</b> Search	-	
Navigator ? 🗗 🗙					N. N.	-77   <del>-</del> 77	- ocach		• • • • •
🔽 Defeuilt									
🎗 Search 🔤 🔻									
Name 🗠 📄									
		· · ·		 . <b>.</b>		• • • • •	· · · · · · ·		 •
									• • • • •
Property Editor ? 🗗 🗙									
Froperty Eultor									
		· · ·				• • • • •			 
mouse L: schSingleSelectPt()				M: deNe	wΛ			B <sup>,</sup> schHil	MousePopUp(
(2) >				m. acre	V			n. von m	Cmd: Sel: 0
									Cilia, Del, U

#### Schematic Entry: Adding an Instance to Schematic

Next, we will create simple schematic of an inverter consisting of an NMOS and a PMOS.

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:

X Add In	istance — 🗆	$\times$
Library	Browse	)
Cell		
View	symbol	
Names		
🗹 Add Wi	re Stubs at: Q all terminals	)
Array	Rows 1 Columns 1	
	🖹 Rotate 🔰 🕢 Sideways ) 🧲 Upside Down	
	Hide <u>C</u> ancel <u>D</u> efaults <u>H</u>	lelp

**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).



**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.

nmos	1v -	NMØ		Ì		45
		"gpdkØS	30_	ΠΠ	osl	Y''
		w:12Øn				
	╘>	I:100n				
•	•	m;1				-

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*  $\rightarrow$  *Properties*  $\rightarrow$  *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"*.

X Library Browser - Add Ins	tance	- 0	× 🕺 Library Browser - Add Ir	nstance	- 🗆 X			
Show Categories			Show Categories					
Library	Cell	View	Library	Cell	View			
analogLib	vdd	symbol	analogLib	gnd	symbol			
analogLib avTech	vccd	View A Lock Size	analogLib 19k avTech	diffstbprobe	View A Lock Size			
basic	vccsp		13k basic	dummy	symbol_xform 13k			
cdsDefTechLib gpdk045	vcres vcvs		cdsDefTechLib gpdk045	fourier fourier2ch				
gpdk090	vcvsp		gpdk090	fracpole				
gpdk180 mylib	vdc vdd		gpdk180 mylib	gnd gnda				
sample	vdd_inherit		sample	gndd				
	vdda vddd			iam ibis buffer				
	vee			ibit				
	veea 📼			idc 🔤	$\sim$			
	Lib: analogLib Free: 701.55G							
Close	Filters	Display He	lp Close	Filters	Display Help			

Next, instantiate power nets (cell vdd and gnd from analogLib library).

5. Execute *Create*  $\rightarrow$  *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**.

X Add Pin		-	$\Box$ ×	🗙 Add Pin -	
Pin Names	in			Pin Names out	
Direction	input 🔽 Bus Expansion	🖲 off 🔾	on	Direction 💿 off	🔾 on
Usage	schematic 🔽 Placement	🖲 single	🔾 multiple	Usage schematic 🔽 Placement 💽 sing	le 🔾 multiple
Signal Type	signal 🧧			Signal Type 🛛 signal 🔽	
Attach Net Expres	sion: 🧕 No 🔾 Yes			Attach Net Expression: 💿 No 🔾 Yes	
Property Name				Property Name	
Default Net Name				Default Net Name	
Font Height	0.0625 Font Style	stick	•	Font Height 0.0625 Font Style stick	-
🛛 🐴 Rotate	👍 Sideways 🛛 🚭 Upside Down	Show Se	ensitivity >>	🚯 🗛 Rotate 🛛 🗥 Sideways 🚭 Upside Down Show	Sensitivity >>
	<u>H</u> ide <u>C</u> ancel	<u>D</u> efaults	<u>H</u> elp	Hide Cancel Defa	ults <u>H</u> elp

6. Use  $Add \rightarrow Wire$  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*  $\rightarrow$ *Save*.

The final schematic looks like the following one:



# 7. Click Check and Save.

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

INF0 (SCH-1170): Extracting "inverter schematic" INF0 (SCH-1426): Schematic check completed with no errors. INF0 (SCH-1181): "mylib inverter schematic" saved.

#### **Netlist Creation and Simulation using Spectre**

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



**1.** In the Schematic editor window, execute *Launch*  $\rightarrow$  *ADE L*. The following window may appear. Click **Always**.



**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		(77)
📖 🔲 <click add="" file="" here="" model="" to=""></click>			

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

🛛 🗕 🗹 /process/gpdk090/libs.oa22/gpdk090/../../models/spectre/gpdk090.scs 🔍 🚬 🛛 🛛 🕬

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

X Choose Moo	del File		×
Look in:	🚞 /tech/process/gpdk090/models/spectre 🧧 😋 😒 🍋	🖻 🔝 🗉	
Comput 120105	<ul> <li>gpdk090.pcf</li> <li>gpdk090_bipolar.scs</li> <li>gpdk090_capacitor.scs</li> <li>gpdk090_diode.scs</li> <li>gpdk090_mos_scs</li> <li>gpdk090_mos_age.scs</li> <li>gpdk090_moscap.scs</li> <li>gpdk090_resistor.scs</li> <li>gpdk090_resistor.scs</li> <li>res_polywo.va</li> <li>res_va.va</li> <li>resd_va.va</li> <li>rnoise_va.va</li> </ul>		-
File <u>n</u> ame:		<u>O</u> pen	
Files of type:	x	Cancel	

#### 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**.



5. Now execute *Setup*  $\rightarrow$  *Stimuli* to assign signals to pins of the inverter.



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).



**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).

X Setup Analog	Stimuli		_		×
Stimulus Type	🖲 Inputs 🧲	Global Sourc	es		<u> </u>
ON in /gnd!	Voltage pul	.se "Voltage	1"=0 "	Voltage	2"=
1					
Enabled ⊻	Function	pulse 🔽	Туре	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			
	<u>0</u> K <u>C</u>	ancel <u>A</u> pp	oly ) ( c	hange)(	<u>H</u> elp

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.

X Choosing	🗙 Choosing Analyses ADE L (1) 🗆 🗙							
Analysis	🖲 tran	0	dc	0	ac	0	noise	
	🔾 xf	0	sens	0	dcmatch	0	acmatch	
	🔾 stb	0	pz	0	sp	0	envlp	
	🔾 pss	0	pac	0	pstb	0	pnoise	
	🔾 pxf	0	psp	0	qpss	0	qpac	
	🔾 qpnoise	0	qp×f	0	qpsp	0	hb	
	🔾 hbac	0	hbnoise	0	hbsp			
	Т	ran	sient Ana	lysi	s			
Stop Time	100n							
Accuracy I	Defaults (errp	res	et)					
Consei	rvative 🗹 m	ode	erate 📃 I	ibe	ral			
Transier	Transient Noise							
Dynamic Parameter								
Enabled 🕑	Enabled 🖌 Options							
	OK <u>C</u> ancel Defaults Apply <u>H</u> elp							

**9.** Select the output to be plotted by executing *Outputs*  $\rightarrow$ *To be plotted*  $\rightarrow$ *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 colours being assigned to these pins.



ADE L (1) - mylib inverter schematic	_		$\times$
Launch Session Setup <u>A</u> nalyses <u>V</u> ariables <u>O</u> utputs <u>S</u> imulation <u>R</u> esults <u>T</u> ools <u>H</u> elp		cāden	ı c e
🚰 🧽   🧊 🖅   👌 🎾 🖆 🗹 🗁			
Design Variables Arguments		? # X	AC   OC   CTrans
Name Value 1 tran 🗹 0 100n moderate			아.
			×
Outputs           Outputs          Name/Signal/Expr         Value   Plot   Save	Save C	Contions	0
1 in 🗹 🗐 a	allv allv		W
> Plot after simulation: Auto ▼ Plotting mode: E	leplace		
2(3) Environment   Status: Ready   T=27   C	Simula	ator: spec	tre:

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

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  $\rightarrow$  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  $\rightarrow$  Load State.

Save State Option	🖲 Directory 🔾 Cellview			
Directory Options				
State Save Directory	<pre>~/.artist_states</pre>		Br	owse
Save As	inverter			
Existing States			_	
Cellview Options				
Cellview Options	mylib			
	mylib inverter • Browse	3		
Library		2		

**12.** Now run the simulation by executing *Simulation*  $\rightarrow$ *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:



#### Definition of rise time, fall time and propagation delay

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.

#### Measuring propagation delay using Waveform calculator

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:



**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.



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

Special Fu	nctions		Q					
<u> </u>			-					_
PN	compare	dBm	dutyCycle	flip	gainMargin	iinteg	loadpull	peakTo
a2d	compression	delay	evmQAM	fourEval	getAsciiWave	inl	Ishift	period
abs_jitter	compressionVRI	deriv	evmQpsk	freq	groupDelay	integ	normalQQ	phasel
average	convolve	dft	eyeAperture	freq_jitter	ňarmonic (	intersect	overshoot	phasel
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			_	5×
Special Functions	<u> </u>			
delay				
Signal1	VT("/in")			
Signal2				
Threshold Value 1	2.5	Threshold Value 2	2.5	
Edge Number 1	1	Edge Number 2	1	U
Edge Type 1	either	Edge Type 2	either	
Periodicity 1	1	Periodicity 2	1	<b>T</b>
				OK Apply Defaults Close Help
Function Panel S	itack			

Fill u	ip the	e rest	of	the	form	as	follows:
--------	--------	--------	----	-----	------	----	----------

Function Panel					e s
Special Functions	<b>_</b> Q				
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		
				OK Apply Defaults C	lose <u>H</u> elp
Function Panel 🗍 🛛 S	Stack				



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

#### 6. Click on Evaluate the buffer icon.

The propagation delay (in seconds) will be displayed in the window.

	vsosopotmpvnspvswrrisoptvarvn2zpypg	
Off 🔾 Family 🔾 Wave	🗹 Clip   🍢 🐗 Append 🔽 Rectangular 🔽 🍪 目	
Key         Image: Constraint of the second sec	-12	
1 2 3 - 0 ± . +	🔁 Pop 📴 🗈 🛛 🕷 🕅 🏧 🚰 🥠 🦿	

#### Measuring rise time and fall time using Waveform calculator

**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 type**s 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 🔽 🚽
	·		OK Apply Defaults Close Help

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

🔍 🔍 Vt	O VI	O Vdc	⊖ vs	⊖ os	🔾 ор	O ot	$\bigcirc$ mp	⊖ vn	🔾 sp
🛛 🔾 it	🔾 if	🔾 idc	🔾 is		🔾 opt	🔾 var		🔾 vn2	🔾 zp
Off (	🔵 Family	/ 🔾 Wave	🛛 🗹 Cli	p   🏹	쇠 🗛	pend		Rectangu	lar 🔽
Key	Ð	× 709.9E	-12						
7 8	9 7								
4 5	6 ×								
1 2	3 -		ED.	Pop   Pop		I Baa	Maaa exp	Pr expr	fn   🛆
0 ±	. +	. ( 1 씨		Pop insert			N=   _		/≜   >>

#### **Power Measurement using Waveform Calculator**

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*  $\rightarrow$  *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 🛄 Ivlpub 🛄 Ivl 🗹 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*  $\rightarrow$  *Result Browser* in ADE L window. 'Result Browser' window will appear to 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.

⊖ vt	⊖ vf	🔾 vdc	OVS ○OS	○ op ○ ot	🔾 mp
🔾 it	🔾 if	🔾 idc	🔾 is	○ op ○ ot   ○ opt ○ va	r
			🗹 Clip   🖏		

**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:



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

Function Panel						
Special Functions 🧧 🔍						
PN a2d abs_jitter <mark>average</mark> bandwidth clip compare compression compressionVRI	dutyCycle evmQAM evmQpsk eyeAperture eyeDiagram fallTime flip fourEval freq	iinteg inl integ intersect ipn ipnVRI loadpull Ishift normalQQ	psd psdbb pstddev pzbode pzfilter riseTime rms rmsNoise rms_jitter	tangent thd unityGainFreq value waveVsWave xmax xmin xval ymax		

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

Key	Ð×	averagergetData(":pwr" ?result "tran")
7 ( 8 )	9 ( / )	

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

Ke	у		Ð×	1.838E-6
7	8	9		
4	5	6	×	

#### **# Exercises**

- **1.** Explain the nature of power consumption curve.
- **2.** Perform SS, FF, SF and FS process corner simulations and compare the power consumption and propagation delays.

Shortcut key	Tasks performed
W	Add a wire
i	Add an instance
р	Add a pin
1	Add label to a wire
e	Display options
q	Select an object and press q to open 'Edit
	Object Property' dialogue box
[	Zoom out
]	Zoom in
с	Сору
m	Move
u	Undo
Shift+u	Redo
f	Fit the entire schematic in the window

### Appendix: Cadence Virtuoso® Schematic Editor L Shortcuts

# EEE 4134 VLSI I Laboratory Lab 2

#### DC sweep, Parametric sweep and Symbol creation of inverter

#### **Objectives:**

- To learn how to perform DC sweep and parametric simulation in ADE L
- To learn how to create symbol view from schematic view

#### DC Simulation and Parametric Analysis in ADE L

**1.** To open the schematic of inverter, execute *File*  $\rightarrow Open$  in **CIW**. In the '**Open File**' window, select the inverter schematic from the list. Click **OK**.

X Open	File		_		$\times$
- File		Calla			
Library	mylib	Cells invert	er		
Cell	inverter	110010	01		
View	schematic 🔽				
Туре	schematic Browse				
Applicat Open wit					
Open for	● edit ⊖ read	,			
Library pa	th file /home/fall16/120105001/cic/c	ds.lib			
		<u>о</u> к	<u>C</u> an	cel	<u>H</u> elp

**2.** Schematic editor window will open. Execute *File*  $\rightarrow$  *Save a copy*. In the following window, change the name of the cell to *inverter2*. Click **OK**. Close the schematic editor window and open the *inverter2* cell from **CIW**.

X Save a Copy	—		×
Library Name	mylib		
Cell Name	inverter2		
View	schematic		
	<u>OK</u> Can	cel	<u>H</u> elp

**3.** This time we will perform both DC and parametric simulation at the same time on inverter schematic. We will obtain the transfer characteristic curve (TCC) of inverter from DC simulation

and by varying the width of the PMOS transistor; we will observe its effect on transfer characteristics.

Select the PMOS transistor in the schematic editor window, click '**q**' and '**Edit object properties**' window will open. Place *w* under '**Total Width**' and press **tab** on keyboard. The '**Finger Width**' field will be automatically changed as follows:

CDF Parameter	Value	Display
Model Name	gpdk090_pmos1v	
Multiplier	1	
Length	100n M	
Total Width	w M	
Finger Width	iPar("w") / iPar("fingers") M	

**4.** Place symbol of an instance **vdc** from **analogLib** to the schematic. In the '**DC voltage**' field, type **vin** and press **tab** on keyboard. Connect the voltage source between '**in**' and '**gnd!**'.

X Add Ir	istance	_	
Library	analogLib		Browse
Cell	vdc		
View			
Names			
🗹 Add Wi	ire Stubs at: Q all terminals 🧕	registered terminals	only
Array	Rows 1	Columns	1
	🖹 Rotate 🛛 🚺	Sideways 🛛 🚄 Ups	ide Down
Noise file	name		
Number of	f noise/freq pairs	0	
DC voltag	e	vin V	

5. Now place another instance of vdc on the schematic, in the 'DC voltage' field, put 1.2 and connect it between vdd! and gnd!

The final schematic should look like the following:



You can remove the 'in' pin and 'vdd' symbol. Click Check and Save.

6. Execute Launch  $\rightarrow$  ADE L and setup Model Library to gpdk090\_mos.scs and section to TT\_s1v similar to the way you did in Lab 1.

7. Execute *Variables* → *Edit* and 'Editing Design Variables' window will open.

X Editing Des	ign Variables ADE L (3)		-		$\times$
	Selected Variable	Design Variables			
Name Value (Expr)		Name		Value	
Add Dele					
Cellview Varia		Apply & Rup S	Simulation		lein
	<u> </u>	) Apply & Run S	Simulation		<u>l</u> elp

**8.** Select '**Copy From**'. You will see '**w**' and '**vin**' appear in the '**Design Variables**' window. Click on '**w**' and in '**Value (Expr)**' field, put a default value of **480n**. Click **Apply**. Similarly click on '**vin**' and in '**Value (Expr)**' field, put a default value of **0.6**.

X Editing Desi	gn Variables ADE L (5)		_		$\times$
	Selected Variable	Design Variables			
blaura		Name	000	Value	
Name	vin	1 vin	600m		
Value (Expr)	0.6	2 W	480n		
Add Dele Next Cle					
Cellview Variat	oles Copy From Copy To	< <u> </u>			
	OK <u>C</u> ancel <u>A</u> pply	Apply & Run S	imulatio	n <u>E</u>	<u>H</u> elp

Then click **OK**.

9. Execute *Analyses* → *Choose* and in the 'Choosing Analyses' form, select dc. Click on 'save dc operating point'.

10. Under 'Sweep Variable', select 'Design Variable' and click on 'Select design variable'.



Select 'vin' and click OK.

11. Under 'sweep range', select 'start-stop' and put a start value of 0 and a stop value of 1.2. Select 'Sweep type' to be 'linear' and 'Step size' to be 0.01. Click OK.

X Choosing	Analyses A	DE L (5)	_		$\times$
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	e 🔾 hbsp		
		DC Analy	sis		
Save DC O	perating Poir	nt 🗹			
Hysteresis S	Sweep				
Sweep Val Temper Design Compor Model F	ature Variable nent Parame		ble Name 🔽 Select Desig		
Sweep Rar Start-St Center-	top g	Start 0	Stop	1.2	
Sweep Typ Linear	00	● Step S ○ Numb	Size er of Steps	0.01	
Add Specifi	c Points 📃				
Enabled 🗹	ОК	Cancel	Defaults	Options Apply <u>E</u>	
			Delauro		Toub

**12.** Execute *Outputs*  $\rightarrow$  *To be plotted* and select 'out' pin on the schematic. ADE L window will now look like the following:

ADE L (5) - mylib inverter2 schematic —	$\Box$ ×
Launch Session Setup Analyses Variables Outputs Simulation Results Tools Help <b>C</b>	ādence
Image: Second state       Image: Second state<	
> Select on Schematic Outputs to Be Plot Plot after simulation: Auto Plotting mode: Replace	
	tor: spectre

Select Netlist and run to simulate a single TCC for the given default PMOS width of 480nm.

#### **13.** Execute **Tools→ Parametric Analysis.** 'Parametric Analysis' window will open.

🔄 Parametric Analysis - spectre(4): mylib inverter2 schematic	—		$\times$
<u>F</u> ile <u>A</u> nalysis <u>H</u> elp		cāde	nce
III Ready			
🖻 🔚 🏪 🐗 🗙 💿 🕓 📝 🎹 🗕 Run Mode: Sweeps & Ranges 🔽	0		
Variable         Value         Sweep?         Range Type         Start Point         End Point         Step           Add Variable         Image: Complex C	Mode	Step V	'alue
21 Delete selected rows			

**14.** Click on 'Add Variable'. From the drop down menu, select 'w'. Put From: 240n and To: 720n Select Step Mode: *linear steps* from the drop down menu and put Step Size: 120n

1201	7201. Select Step Mode. <i>linear steps</i> from the drop down menu and put Step Size. 1201.								
	Variable	Value	Sweep?	Range Type	From	То	Step Mode	Step Size	
W		480n		From/To	240n	720n	Linear Steps	120n	
						_			
<b><i>C</i>11 1</b>					. 💿				

Click on '**Run selected sweeps**' icon to start simulation.
$\times$ 



The simulated waveforms will be displayed.

These transfer characteristics can be further explored to find Noise margin, and inversion voltage for different widths of PMOS.

**15.** This parametric analysis option can be saved for later use by clicking on the save icon. In the following window, give a name to the configuration file and click **Save**.

X Save as Parametric File

Look in:	im/home/fall16/120105001/cic				- C O 🗠 e	
📃 Comput	Name	S	Size	Туре	Date Modified	
= 120105	🚞 .cadence			Folder	30 Mar3:35:43	
	mylib			Folder	5 Apr 24:30:52	
File <u>n</u> ame:	inverter2					<u>S</u> ave
Files of type:	Parametric Files (*.il)				<b></b>	Cancel

**16.** Also save the state of the ADE L window.

### # Exercises

- 1. Show the effect of changing NMOS width on the TCC of an inverter.
- 2. Perform a parametric analysis in transient simulation to show the effect of changing the PMOS width on the propagation delay. You are expected to obtain a similar graph as shown below:



Setup necessary settings for the delay function in the waveform calculator. Click OK and click on

# **'Evaluate the buffer and display the results in a table**' icon.

The results should be displayed in a table like below:

Virtuoso (R) Visua	alizatio	n —		×
<u>F</u> ile <u>E</u> dit <u>V</u> iew	<u>T</u> ools	<u>H</u> elp	cā	dence
📔 🔤 🛛 🦻	ぐ √T("/in	), ?value	e1 🗙	
W 1 240.0E-9 2 360.0E-9 3 480.0E-9 4 600.0E-9 5 720.0E-9	-292. -174. -87.9 -19.9	y(?le ni 1E-12 5E-12 1E-12 8E-12 8E-12	1)	

- 3. Some of the propagation delays are negative. Explain why?
- 4. Explain the change in propagation delay with the change in PMOS transistor width.
- 5. Obtain the output characteristic curve (I<sub>D</sub> vs V<sub>DS</sub>) and transfer characteristic curve (I<sub>D</sub> vs V<sub>GS</sub>) of NMOS and PMOS.

### **Symbol Creation**

In this section, you will create a symbol for your inverter design so that you can use this symbol view for the schematic in a hierarchical design. In addition, the symbol has attached properties (cdsParam) that facilitate the simulation and the design of the circuit.

**1.** Open the schematic of the cell *inverter* from library *mylib*.

2. In the schematic editor window for *inverter*, execute *Create* →*Cellview* →*From Cellview*. 'Cellview from cellview' window appears. Click OK.

	Cellview From Cellview	×
Library Name	mylib	Browse
Cell Name	inverter	
From View Name To View Name	schematic 🔽	1
Tool / Data Type	schematicSymbol	
Display Cellview	✓	
Edit Options		
	OK Cancel Defaults	Apply Help

In the 'Symbol Generation Options' window, you can choose the location of the pins.

-	Syı	nbol Gene	ration Optior	าร	×
Library Name		Cell Name		View Name	
mylib		inverter		symbol	
Pin Specificatio	ons				Attributes
Left Pins	in				List
Right Pins	out				List
Top Pins					List
Bottom Pins					List
Exclude Inherit	ed Connection Pin	s:			
🖲 None 🔾	All 🔾 Only these	9:			
Load/Save 📃	Edit Attrib	utes 📃	Edit Labels 🗌	Edit Prop	erties 📃
			ОК	Cancel App	ly Help

Click **OK** and the **Symbol Editor** window will open.

🖉 Virtuoso® Symbol Editor L Editing: mylib inverter symbol	_ = ×
Famicy Ele Est New Carety Object Audom Rudo	cādence
<mark>월 19 년 :   ∲ 8 월 × 0 17 년 5 € 1<sup>°</sup> 1    9, 9, 9, 9</mark>	
C + C + C C    Workspeer: Basic 📴 🗟    🗞 🕸 🗢 🕘 🔩 💿 🖊 💿 🖉 📋	
Haniyatar 7:5 x Y Defant	
$ \frac{1}{2} \frac{1}{1} 1$	
[@instanceName]	
Pryerty Editor (7.5%) in [@partName] out	
	ousePopUp() Cmd: Set: 0

3. Click **Delete** icon in the symbol window, delete the outer red rectangle and green rectangle.

		[@instanceName]
in.	[@partName] out	<b>.</b>

**4.** Execute *Create* →*Shape*→*polygon*, and draw a shape similar to triangle. After creating the triangle, press **Esc** key.

**5.** Execute *Create*  $\rightarrow$ *Shape*  $\rightarrow$ *Circle* to make a circle at the end of the triangle. You can move the pin names according to the location.

6. Execute *Create* →*Selection Box*. In the 'Add Selection Box' form, click 'Automatic'.

X Add Selection Box	_		$\times$					
Draw the rectangle to d or Click below to create			n box					
Automatic								
<u>H</u> ide	<u> </u>	cel )	<u>H</u> elp					

A new red selection box is automatically added.

7. After creating symbol, click on the save icon in the symbol editor window to save the symbol. In the symbol editor window, execute *File*  $\rightarrow$ *Check and Save*. Then close the symbol editor window.



# EEE 4134 VLSI I Laboratory Lab 3 Layout of an Inverter using Virtuoso L

#### **Objectives:**

- To create a layout view of the basic inverter circuit from scratch in Virtuoso Layout Editor
- To design the layout keeping basic design rules in mind
- To design cell layout of a constant height for use in hierarchical design

## Introduction to Layout, DRC and LVS

Layout is representation of a circuit in terms of planar geometric shapes (e.g. rectangles, polygons) showing the patterns of metal, polysilicon, oxide, or diffusion layers that make up the components (resistors, inductors, capacitors, transistors) of the integrated circuit.

When using a standard process (e.g. 45nm, 90nm or 180nm process available in our lab), the behaviour of the final integrated circuit depends significantly on the positions and interconnections of the geometric shapes due to parasitic resistances and capacitances contributed by them. While designing a layout, designer must keep in mind performance (e.g. power-delay product) and size (area occupied by the chip) criterion.

While designing digital circuits, one usually follows an ASIC design flow, where, the height of standard cells that are used is the same throughout the cell library, but their widths must vary according to their logical functions and drive strengths. The following figure shows a generalized standard cell height concept:



Although we will follow a full-custom IC design flow, we will maintain same cell height throughout our cell library.

The generated layout must pass a series of checks in a process known as physical verification. The most common checks in this verification process are:

- Design Rule Checking (DRC)
- Layout Versus Schematic (LVS) checking
- Parasitic extraction and post-layout simulation

#### **Design Rule Check (DRC):**

Design Rule Checking (DRC) is the process that determines whether the designed layout of a circuit satisfies a rules specified by the process being used.

Design Rules are a series of rules (e.g. area, width, overlap, enclosure, extension, spacing) provided by semiconductor manufacturers which are specific to a particular semiconductor manufacturing process. Design rules specify certain geometric and connectivity restrictions to ensure that the process can fabricate the device properly.

#### Layout versus Schematic (LVS) Check:

The Layout Versus Schematic (LVS) is the verification step to determine whether a particular integrated circuit layout corresponds to the original schematic or circuit diagram of the design. A successful Design rule check (DRC) ensures that the layout conforms to the rules designed/required for faultless fabrication. However, it does not guarantee if it really represents the circuit we desire to fabricate. This is why an LVS check is used.

### Layout design using Virtuoso Layout Suite L Editor

**1.** Invoke Virtuoso Layout Suite L Editor from the CIW by executing  $File \rightarrow New \rightarrow Cellview$ .

The 'New File' form appears. Fill it in as shown in the figure below:

Cell: inverter, View: layout. Click OK.

	New File	×
_ File		
Library	mylib	
Cell	inverter	
View	layout	
Туре	layout 🧧	
- Application		
Open with	Layout L	
🔲 Always use thi	s application for this type of file	
Library path file		
/buet/cadence/I	C615/share/cdssetup/cds.1	ib
	OK Cancel He	lp

2. Click 'Always' if the following window appears before Layout window appears.



The following window of Virtuoso Layout Suite L Editor will appear.

Launch <u>F</u> ile <u>E</u> dit <u>V</u> iew <u>C</u> re	eate V	er <u>i</u> fy	Co <u>n</u> n	ectiv	ity g	<u>O</u> ptio	ins	Tool	s <u>W</u>	<u>(</u> indo	w /	Ass <u>u</u> ra	Opti	mize	<u>H</u> el	р							сā	d e	nco
	~ 1			- <b>B</b> 1	0		1.2	0		0				1000						_					
	Ľ.		×	R				•)	»	Q	»	26	»	Clas	SSIC					1	•				
Pr 🔫 🏶 🕂 🗞 I 🧐	Դ 👍	1	J.	»	(F)	)Sele	ect:0	Sel	(N):0	Se	l(l):0	Sel(	D):0	X -13	3.595	0	Y 7	7.285	0	d	IX 🗌	_	_		
Palette ? 🗗 🗙			-					-														<b>*</b>			
Layers 🗗 🗙																									
Valid 🔄 Used 🔄 Routing																									
🖌 Filter 🔤 🔻																									
Nwell drawing																									
AV NV AS NS													•												
Name Vis Sel																									
All Layers + - + -																									
📔 Layer   Pu V   S 🔼																									
Nwell drw 🗹 🗹 📄																									
Oxide drw 🗹 🗹																									
Oxide_thk drw 🗹 🗹 Poly drw 🗹 🗹																									
Pimp drw 🗹 🗹																									
Nhvt drw 🗹 🔽																									
Nimp drw 🗹 🗹																									
Phvt drw 🗹 🗹																									
Nzvt drw 🗹 🗹																									
SiProt drw 🗹 🗹 Cont drw 🗹 🗹 🚽																									
Objects 🗗 🗙																									
Objects   V   S   🛆																									
Instances 🗹 🗹 🗐																									
Pins 🗹 🗹 🚽 🚽																									
Objects Grids																									
mouse L: mouseSingleSelectPt(	)							M	l: del	Jour	)										R.	IVHil	Mous	e Po	nHr

On the left side of the window, you will find a panel called 'Layers'. This panel is divided in three main categories which are: layer color, layer name and layer purpose. The details are described in the table below:

Color	Matches the color in the Editing window. Each layer has its own color and pattern. Each layer has two colors associated with it: a fill color and an outline color. These colors can be changed to fit your taste by editing the technology file.
Name	The type of layer (Nwell, Oxide, Poly, Metal1, etc)
Purpose	In gpdk090 the only purpose classifications are: drw = drawing, slot = slot Drawing is used in layout, slot is used to create a hole for metal stress relief

Verify that the layers display corresponds to the gpdk090 layers shown in the GPDK 90 nm Mixed Signal Process Specification manual (*gpdk090\_DRM.pdf*).

**3.** Before starting to design layout, you need to set the layout display configuration. Execute the following in the Virtuoso Layout Editor: *Options*  $\rightarrow$ *Display* or press 'e' on keyboard. Configure the form as shown in the figure below: You have to set the following parameters only:

Minor spacing	0.01
Major spacing	0.1
X snap spacing	0.005
Y snap spacing	0.005
Display Levels: Stop	10

Display Controls         ✓ Open to Stop Level       Nets         ✓ Axes       Access Edges         Instance Origins       Instance Pins         EIP Surround       Array Icons         ✓ Pin Names       ✓ Label Origins         Dot Pins       Use True BBox         ✓ Net Expressions       Cross Cursor         Stretch Handles       Row Name         ✓ Via Shapes       Row Name         ✓ Dynamic Hilight       ✓ True Color Drag         Ø Dragged Object Ghost       Transparent Group         Traversed instance BBox       Selection Hint         Maximum Number of Drag Figures       500         Scroll Percent       25         Instance Drawing Mode       BBox         Path Display       Borders and Centerlines         Stow Name Of       Instance Instance Instance         Full       Start         Ø Full       Start         Ø Border       Join         Source       Stop         Ø Cellview       Library         Save To       Load From	🗙 Display Options			_		×
<ul> <li>Axes</li> <li>Access Edges</li> <li>Instance Origins</li> <li>Instance Pins</li> <li>EIP Surround</li> <li>Array Icons</li> <li>Pin Names</li> <li>Label Origins</li> <li>Dot Pins</li> <li>Use True BBox</li> <li>X Snap Spacing</li> <li>0.05</li> <li>Y Snap Spacing</li> <li>0.06</li> <li>0.06</li> <li>0.07</li> <li>0.08</li> &lt;</ul>	Display Controls		- Grid Controls -			
Instance Origins       Instance Pins         ✓ EIP Surround       Array Icons         ✓ Pin Names       ✓ Label Origins         ○ Dot Pins       Use True BBox         ✓ Net Expressions       Cross Cursor         Stretch Handles       Row Name         Via Shapes       Row Site         ✓ Dynamic Hilight       ✓ True Color Drag         ✓ Dragged Object Ghost       Transparent Group         Traversed instance BBox       Selection Hint         Maximum Number of Drag Figures       S00         Scroll Percent       25         Instance Drawing Mode       BBox         ● Full       Start         ● Full       Start         ● Source       Stop         ● Full       Start         ● Source       Stop         ● Cellview       Library         Save To       Load From	🗹 Open to Stop Level	🔲 Nets	Type 🧕 none	e 🔾 di	ots 🔾 lin	es
<ul> <li>✓ EIP Surround</li> <li>Array Icons</li> <li>✓ Minor Spacing</li> <li>0.01</li> <li>Major Spacing</li> <li>0.1</li> <li>Major Spacing</li> <li>0.1</li> <li>Major Spacing</li> <li>0.1</li> <li>Major Spacing</li> <li>0.05</li> <li>Y Snap Spacing</li> <li>0.005</li> <li>0.005</li></ul>	🗹 Axes	Access Edges	Dim Major Dots			
	-	_	Minor Spacing	ſ	0.01	_
□ Dot Pins       □ Lise True BBox         □ Net Expressions       □ Cross Cursor         □ Stretch Handles       □ Row Name         □ Via Shapes       □ Row Site         □ Dynamic Hilight       □ True Color Drag         □ Dragged Object Ghost       □ Transparent Group         □ Traversed instance BBox       □ Selection Hint         Maximum Number of Drag Figures       500         Scroll Percent       25         Instance Drawing Mode       □ Dont check validity         Path Display       □ Dont check validity         Show Name Of ○ instance ○ master ○ both       □ Inming         Array Display       □ Display Levels         ○ Full       Start         ○ Border       10         ○ Source       Stop         10       □ Code From		, ,				_
✓ Net Expressions       Cross Cursor         Y Snap Spacing       0.005         Stretch Handles       Row Name         Via Shapes       Row Site         ✓ Dynamic Hilight       ✓ True Color Drag         ✓ Dragged Object Ghost       Transparent Group         Traversed instance BBox       Selection Hint         Maximum Drag Level       32         Maximum Number of Drag Figures       500         Scroll Percent       25         Instance Drawing Mode       BBox         Path Display       Borders and Centerlines         Stet LPP Visibility       Do not check validity         Show Name Of       instance         Full       Start         © Full       Start         © Source       Stop         Stop       10]         O Cellview       Library         Tech Library       File         */. cdsenv       Browse		-				_
Stretch Handles       Row Name         Via Shapes       Row Site         ✓ Dynamic Hilight       True Color Drag         ✓ Dragged Object Ghost       Transparent Group         Traversed instance BBox       Selection Hint         Maximum Drag Level       32         Maximum Number of Drag Figures       500         Scroll Percent       25         Instance Drawing Mode       BBox         Path Display       Borders and Centerlines         Stet LPP Visibility       Do not check validity         Show Name Of       instance         Full       Start         Source       Stop         Source       10          Order       10          Save To       Load From	🔲 Dot Pins	Use True BBox	X Snap Spacin	g	0.005	
<ul> <li>Via Shapes</li> <li>Row Site</li> <li>Dynamic Hilight</li> <li>True Color Drag</li> <li>Dragged Object Ghost</li> <li>Transparent Group</li> <li>Traversed instance BBox</li> <li>Selection Hint</li> <li>Maximum Drag Level 32</li> <li>Maximum Number of Drag Figures 500</li> <li>Scroll Percent 25</li> <li>Instance Drawing Mode</li> <li>BBox</li> <li>Path Display</li> <li>Borders and Centerlines</li> <li>Stow Name Of  <ul> <li>Instance Or instance</li> <li>master  <ul> <li>both</li> <li>Array Display</li> <li>Display Levels</li> <li>Full</li> <li>Start 0</li> <li>Border</li> <li>Source</li> <li>Stop 10</li> </ul> </li> <li>Cellview  <ul> <li>Library</li> <li>Tech Library</li> <li>File</li> </ul> </li> <li>Filter</li> <li>Size 6</li> <li>Style empty</li> <li>Size 6</li> <li>Style empty</li> <li>Snap Modes</li> <li>Create orthogonal</li> <li>Create orthogonal</li> <li>Browse</li> </ul> </li> </ul>	✓ Net Expressions	Cross Cursor	Y Snap Spacin	g	0.005	
✓ Dynamic Hilight       ✓ True Color Drag         ✓ Dragged Object Ghost       Transparent Group         Traversed instance BBox       Selection Hint         Maximum Drag Level       32         Maximum Number of Drag Figures       500         Scroll Percent       25         Instance Drawing Mode       Box         Path Display       Borders and Centerlines         Show Name Of       instance         ● Full       Start         ● Full       Start         ● Border       Stop         Source       Stop         Store       Stop         Oliming       Store         Ore       O         Display Levels       Source         Stop       10         Ore       Stop         Oliming       Stop         Oliming       Source         Stop       10         Stop       10         Browse       Save To         Load From       Delete From	Stretch Handles	🔲 Row Name				
✓ Dragged Object Ghost       Transparent Group         ✓ Traversed instance BBox       Selection Hint         Maximum Drag Level       32         Maximum Number of Drag Figures       500         Scroll Percent       25         Instance Drawing Mode       BBox         Path Display       Borders and Centerlines         Set LPP Visibility       Do not check validity         ✓ Show Name Of       instance         ● Full       Stat         ● Full       Stat         ● Source       Stop         10]       Stat         ● Cellview       Library         Cellview       Library         Tech Library       File         ✓/. cdsenv       Browse	🔲 Via Shapes	Row Site	Filter			
✓ Dragged Object Ghost ☐ Transparent Group ☐ Traversed instance BBox ☐ Selection Hint Maximum Drag Level 32 Maximum Number of Drag Figures 500 Scroll Percent 25 Instance Drawing Mode BBox ♥ Path Display Borders and Centerlines ♥ Set LPP Visibility Do not check validity ♥ Show Name Of ○ instance ● master ○ both Array Display Levels ● Full Start 0 ○ Source Stop 10] ● Cellview ○ Library ○ Tech Library ○ File ♥/. cdsenv Browse Save To Load From Delete From	🗹 Dynamic Hilight	🗹 True Color Drag	Size 6	Style	empty	
Maximum Drag Level 32 Maximum Number of Drag Figures 500 Scroll Percent 25 Instance Drawing Mode BBox Path Display Borders and Centerlines Set LPP Visibility Do not check validity Show Name Of instance master both Array Display Display Levels Full Start 0 Border Source Stop 10 Cellview Library Tech Library File Save To Load From Delete From Create orthogonal Create orthogonal Dimming Enable Dimming Scope none Dim Grid Lines Automatic Dimming Source Stop 10 Source Stop 10 Dim Intensity: Source Stop 10 Dim Selected Object Content True Color Selection only Browse	🗹 Dragged Object Ghost	🔲 Transparent Group		<i></i>		
Maximum Number of Drag Figures       500         Scroll Percent       25         Instance Drawing Mode       BBox         Path Display       Borders and Centerlines         Set LPP Visibility       Do not check validity         Show Name Of       instance         Instance       master         both       Array Display         Display       Display Levels         Full       Start         Source       10         Source       10         O       Cellview         Library       Tech Library         Full       V. cdsenv         Border       10         Source       10         Cellview       Library         Tech Library       File         V. cdsenv       Browse	🔲 Traversed instance BBox	Selection Hint	- Snap Modes			
Scroll Percent 25   Instance Drawing Mode BBox   Path Display Borders and Centerlines   Set LPP Visibility Do not check validity   Show Name Of instance   Marray Display Display Levels   Full Start   Border 10   Source Stop   Stop 10   Cellview Library Tech Library File File File File File Path Display Display Levels Dim Intensity: 50 File Source Source Stop Tech Library File File File File Path Display Display Display Levels Source Source Stop Tech Library File File File File Path Display Display Display Display Display Library Display Display Display Library Display D	Maximum Drag Level 32		Create	orthog	onal 🔽	
Instance Drawing Mode BBox Path Display Borders and Centerlines Set LPP Visibility Do not check validity Show Name Of instance master both Array Display Display Levels Full Start Border Source Stop 10 Cellview Library Tech Library File Save To Load From Delete From Dimming Enable Dimming Scope Dim Grid Lines Automatic Dimming Source Dim Grid Lines Source Dim Intensity: Source Source Save To Load From Delete From Dimming Scope Dim Grid Lines Scope Dim Grid Lines Scope Dim Grid Lines Scope Dim Grid Lines Source Source Source Save To Load From Delete From Source S	Maximum Number of Drag F	igures 500	Edit	orthog	onal 🔽	
Instance Drawing Mode BBox  Path Display Borders and Centerlines  Set LPP Visibility Do not check validity  Show Name Of  instance  master  both Array Display Display Levels Full Border Source Stop 10 Cellview Library Tech Library File /.cdsenv Browse Browse	Scroll Percent 25					
Path Display       Borders and Cententines         Set LPP Visibility       Do not check validity         Show Name Of       instance         Show Name Of       instance         Image: Show Name Of       instance         Show Name Of       instance         Image: Show Name Of       Image: Show Name Of         Image: Show Name Of       Image: Show Name Of </td <td>Instance Drawing Mode</td> <td>Box</td> <td>-</td> <td>4</td> <td></td> <td></td>	Instance Drawing Mode	Box	-	4		
Set LPP Visibility Do not check validity   Show Name Of instance   Show Name Of instance   Array Display Display Levels   Full Start   Border 50   Source Stop   10 True Color Selection only   Save To Load From Dim Grid Lines Automatic Dimming Automatic Dimming Full Start Source Stop Source Stop Tech Library File File File Provide From Dim Grid Lines Automatic Dimming Cellview Cellview	Path Display Borders a	nd Centerlines 🔽	Scope	none		
Show Name Of () instance () master () both   Array Display   O Full   O Full   Start   O   Border   Source   Stop   10   O Cellview   Library   Tech Library   File   V. cdsenv   Browse		eck validity 🔽	Dim Grid Lines			
Array Display Display Levels Full Start 0 Border Source Stop 10 Cellview Library Tech Library File //.cdsenv Browse Save To Load From Delete From			Automatic Dimm	ning	<b>~</b>	
• Full Start   • Border Start   • Source Stop   10 Image: Stop   • Cellview Library   • Cellview Library   • Cellview Library   • Cellview Cellview   • Cellview Eibrary   • Cellview File   • Cellview • Cellview			Dim Intensity:			
Border     Source     Stop     10     Cellview     Library     Tech Library     File     Save To     Load From     Delete From				50		
Source Stop 10 True Color Selection only     Cellview Clibrary Tech Library File //.cdsenv     Save To Load From Delete From	otait	0		۳ ۱۰:۰۰۰۰		-
Cellview O Library O Tech Library O File V. cdsenv     Save To Load From Delete From	Ston	10				_
Save To Load From Delete From	O Source		True Color Sele	ction o	nly	
	● Cellview ◯ Library ◯ T	Fech Library 🔾 File 🛛 🗸 🤆	dsenv		Bro	wse
	Save To	Load From	Delete From			
<u> </u>			el ) <u>D</u> efaults			

**4.** Now we are going to build the layout of the inverter. An inverter has an NMOS and a PMOS transistor. First we will build an NMOS transistor.

Layout of NMOS inverter consists of oxide, Nimp, Cont and Poly layers. Study the rules of these layers and calculate the minimum size of the Poly, Cont, Oxide and Nimp layer to create an NMOS transistor.

Contact size	$0.12 \ \mu\text{m} \times 0.12 \ \mu\text{m}$ (Fixed)
Poly width (Minimum)	0.1 $\mu$ m (Fixed MOS gate length)
Contact to poly spacing (Minimum)	0.1 μm
Contact to oxide enclosure (Minimum)	0.06 μm
Poly/Nimp extending from oxide (Minimum)	0.18 $\mu$ m (gate side enclosure)
Nimpenclosing oxide (Minimum)	$0.14\mu$ m (enclosure other than gate sides)
Minimum Metal 1 width	0.12 μm
Maximum Metal 1 width	12.0 μm
Minimum Metal 1 to Contact enclosure	0.06 $\mu$ m (on at least two opposite sides)

The rules related to the NMOS transistor can be summarised as follows:

The following figure illustrates some of the design rules mentioned above:



Now study the PMOS transistor structure in the GPDK 90 nm Mixed Signal Process Spec. The PMOS transistor consists of Oxide, Poly, Pimp, Cont and Nwell layer. Study the rules of these layers and calculate the minimum size of Poly, Cont, Oxide, Pimp and Nwell layer to create a PMOS transistor. The rules related to PMOS are same as NMOS except the there is an additional layer, the Nwell, whose rules are as follows:

Minimum Nwell width	0.6 µm
Minimum Nwell spacing to Nwell (same potential)	0.6 µm
Minimum Nwell spacing to Nwell (different potential)	1.2 μm
Minimum Nwell spacing to N+ active area	0.3 μm
Minimum Nwell spacing to P+ active area	0.3 μm
Minimum Nwell enclosure to P+ active area	0.12 μm
Minimum Nwell enclosure to N+ active area	0.12 μm
Minimum N+ Active Area to P+ Active Area Spacing	0.16µm

Now we start building the NMOS and PMOS transistor layout. Look at the LSW and find the current drawing layer.

**5.** Click on the following icon in **Virtuoso Layout Suite L Editor** window so that it notifies you anytime you make a violation of any design rule. When clicked, it will show '**DRD Notify ON**'. DRD stands for Design Rule Driven.



6. Select 'Cont (drw)' (contact) layer from the 'Layers' panel and draw a rectangle of 'Cont (drw)' layer using *Create*  $\rightarrow$ *Shape*  $\rightarrow$ *Rectangle* or simple pressing 'r'. Press 'Esc' to stop 'create rectangle' tool. In gpdk090 technology, Cont layers must be of dimension 0.12  $\mu$ m x 0.12  $\mu$ m. So, if your rectangle is not of that dimension, click on the rectangle, press 'q'. In the following window, check if the criterion has been met and change 'Width/Height' if required.

à	Edit Rectangl	e Properties			-		$\times$
	< Attribute	> Apply Com Connectivity Parameter	mon Property	ROD			
	Layer	Cont drw -	Layer Filter				
	Left	-9.92	Bottom	1.935			
	Right	-9.8	Тор	2.055			
	Width	0.12	Height	0.12			
				OK Cancel	Ар	ply H	lelp

Contact to poly spacing must be 0.1  $\mu$ m in this technology and the channel length of NMOS/PMOS in our design is 0.1  $\mu$ m. So, we need a minimum space of 0.3  $\mu$ m between the contacts at source and drain.

7. Press 'k' to invoke the 'ruler' tool. Use it to measure lengths whenever needed. To copy, press'c'. After placing two contacts, the layout looks like this:



8. Now, contact to oxide spacing is minimum 0.06  $\mu$ m. So, draw a rectangle of 'Oxide (drw)' layer so that it covers both the contacts and extends from each side by 0.06  $\mu$ m.



While drawing this, you will see Design rule violations when they are committed.

9. After drawing oxide layer, the layout should look like this:



10. Now we will draw 'Nimp (drw)' layer, which must extend from the oxide layer by a minimum of 0.14  $\mu$ m. First, draw a rectangle and then extend it to meet design rules. Use stretch tool by pressing 's'. Layout will look like the following:



11. Now, copy it and create another copy of all these layers by selecting all and pressing 'c'.

12. Click on the 'Nimp (drw)' layer of the copy in the upper portion of the layout and press 'q' to edit properties. From 'Edit Rectangle Properties' window, select 'Pimp (drw)' layer under 'Layer' option. Click OK.

🔆 Edit Rectangle	e Properties			-		×
< Attribute	> Apply Com Connectivity Parameter	mon Property	ROD			
Layer	Pimp drw 🗸	Layer Filter				
Left	-10.12	Bottom	3.6			
Right	-9.18	Тор	4.12			
Width	0.94	Height	0.52			
			OK Cancel	Ap	ply) (F	lelp

**13.** With more metal layers available in today's silicon processes, using the routing approach, such as first metal traverse vertically and second metal traverse horizontally, would be advantageous in standar cell physical design. Using this method, the second layer (e.g. Metal2) can be used for power and ground routing over internal standard cell transistors. In standard cell layout, it is preferable to use firt conducting layer, such as Metal1, as much as possible to make internal connections of NMOS and PMOS transitors within the cell. If there is a nedd to use other conducting layers, such as, Metal2, use of such layers must be kept to a minimum. It is desired to use first routing (e.g. Metal1) layer for standard cell ports.

Our cells will have a height of 5  $\mu$ m. Place the two parts (NMOS and PMOS) 2  $\mu$ m apart, and create a ruler so that the cell height can be checked whenever needed and the separation between the NMOS and PMOS can be maintained properly. Now, the layout will look like the following:



14. Next, draw a 'Poly (drw)' path by selecting 'Poly' layer from the 'Layers' panel and pressing 'p' to invoke 'create path' tool. This layer must be of  $0.1 \,\mu m$  in width and in between the two contacts, extending from the oxide layer by  $0.18 \,\mu m$  (at least, on both sides). After placing the 'poly' gate, the layout will look like the following one:



**15.** Now that you know most of the shortcuts and layers, draw contact for body terminals for NMOS and PMOS. These portions should consist of Cont, Oxide and Nimp (for body of PMOS) or Pimp (for body of NMOS). Check DRD notifications for design rule violations. The following figure shows a Psubstrate and an Nwell contact. The measurement dimensions are shown only on the left one, as they are same for both contacts.



16. Connect the Drain regions of the NMOS and PMOS. Also connect the source of both MOS's to respective body terminals using 'Metal1 (drw)' layer. Connect the drains of the MOS's using 'Metal1 (drw)' layer.

**17.** PMOS should be in '**Nwell (drw**)'. So draw an '**Nwell (drw**)' rectangle surrounding both the PMOS and the body contact for PMOS. The layout will look like the following:



18. Now, we have to place pins. The gate is in 'poly (drw)' layer. Let's bring it to 'Metal1(drw)' layer by extending the 'Poly (drw)' layer, creating a contact between 'Poly' and 'Metal1' layer by pressing 'o' to 'create via' and selecting 'M1\_POv' under 'via definition' and placing it on layout.

N		_	
🗙 Create Via	-		×
Mode 💿 Single 🔾 Stack 🔾 Auto			
Options Compute From Shape(s)			
Net Name			
Create as ROD Object Name via0			
- ,			
Via Definition M1_POV 🔽 🍸 Standard Via	/ mdk00	0	
	/ gpuxos	,0	
Save Via Variant			
System User defined Cut pattern Array	/ pattern		
Reset Parameters to			
Justification centerCenter			
Justification CenterCenter X 0	Y	0	
	Length	D. 12	
widin 0.12	Length	0.12	
Rows 1 🗧 Row Spacing 0.14	*		
Columns 1 😝 🖉 Column Spacing 0.14	\$		
C Enclosures			51.
Compute Show	v Enclosu	res	
A Rotate A Sideways		ide Down	
<u>H</u> id	e <u>C</u> a	ncel	<u>H</u> elp

**19.** Also draw a '**Metal1** (**drw**)' rectangle on the via, because the default **Metal1** rectangle area is less than the required minimum.





For in, vdd! and gnd! select 'input' as 'I/O type' and for out select 'output' as 'I/O type'. Now, draw rectangles on the Poly-Metal1 via for 'in' pin, PMOS source-to-body 'Metal1' connection for 'vdd!' pin and NMOS source-to-body connection for 'gnd!' pin. For 'out' pin, draw the rectangle on the Metal1 layer connecting the two drains of MOS's.

X Create Pin		-		×
Mode	🖲 Manual 🔾 Auto			
Connectivity	🖲 Strong 🔾 Weak			
Terminal Names	in vdd! gnd!	Physica	al Only	
🔲 Keep First Name	X Pitch 0 Y Pitch	0		
📃 Create Label	Options			
🔲 Create as ROD C	Dbject			
Name	rectO			
Pin Shape	🖲 rectangle 🔾 dot 🔾 polygon 🔾	circle		
I/О Туре	🖲 input 🔾 output 🔾 inputOutput	🔾 switch	1	
	🔾 jumper 🔾 unused 🔾 tristate			
Snap Mode	orthogonal 🧧			
Access Direction	🗹 top 🗹 bottom 🗹 left ⊻ right			
	🗹 any 📃 none			
		Com		la la
	<u>H</u> ide	<u>C</u> anc		lelp

You may add label to pins.

**21.** Finally, add **Metal2** paths of **0.5**  $\mu$  width for power rails and connect them to power nets in **Metal1** by using **Metal1 to Metal2** via by invoking 'create via'.

The final layout will look like the following:



## **Creating Body ties**

Now you know what body tie is. We will now make two instances for body ties one for **Psub** and one for **Nwell**.

**1.** Execute *File*  $\rightarrow$ *New*  $\rightarrow$ *Cellview* and fill in the New File form as follows. Click **OK**.



2. Draw psubstrate contact in the same way as you have made it in Lab3.



**3.** Save it and make nwell contact similarly (name it **M1\_NWELL**), just change the **Pimp** layer to **Nimp** and everything else is the same. Save these two for later use.

Shortcut Key	Tasks performed
f	Fit display to window
r	Draw rectangle
q	Edit property of an object
р	Makes a min width path of the layer
	selected in LSW
Ctrl+a	Select all
Ctrl+d	Deselect all
с	Сору
m	Move
S	Stretch side of a rectangle
k	Invoke ruler tool
Shift+k	Delete all rulers
i	Add an instance
u	Undo
Shift+u	Redo
e	Display options
0	Add via between layers
1	Create a label

# Appendix A (Shortcut keys for Cadence Virtuoso ® Layout Editor L)

# Appendix B (gpdk090 Design Rules Guide (Abridged Version for VLSI-I Lab))

# **Terminology Definitions**

**Spacing -** distance from the outside of the edge of a shape to the outside of the edge of another shape.



**Enclosure** - distance from the inside of the edge of a shape to the outside of the edge of another shape.



**Overlap** - distance from the inside of the edge of a shape to the inside of the edgeof another shape.



Butting - outside of the edge of a shape touching the outside of the edge of anothershape.



Description (For metals k=2 to 6, for vias k=1 to 6)	Value (µm/µm <sup>2</sup> ) wherever
	applicable
Minimum oxide (active) area	0.06
Minimum 1.2V N/P Channel gate length	0.1
Minimum poly interconnect width	0.1
Minimum gate/poly interconnect space	0.12
Minimum N/P-channel gate extension beyond active area	0.18
Minimum poly interconnect to related/unrelated active area space	0.1
Minimum poly interconnect area	0.1
Bent gate is not allowed	
Minimum N+/P+ implant width	0.24
Minimum N+/P+ implant space	0.24
Minimum N+/P+ implant to active area enclosure	0.14
Minimum N+/P+ implant to gate side enclosure	0.18
Minimum N+ to P+ active area (inside Nwell) spacing	0.16
Minimum N+/P+ implant area	0.15
N+ implant is not allowed over P+ implant	
Minimum P+ to N+ active area (outside Nwell) spacing	0.16
Maximum and minimum Contact width/length	0.12
Minimum Contact to Contact spacing	0.14
Minimum Contact on Active Area to gate spacing	0.1
Minimum gate Contact to Active Area spacing	0.12
Minimum Active Area to Contact enclosure	0.06
Minimum Poly to Contact enclosure	0.04
Minimum Poly to Contact enclosure on at least two opposite sides (end of line)	0.06
Contact on gate is not allowed, Contact must be covered by Metal1 and active area/poly	
Minimum Metal 1 width	0.12
Maximum Metal 1 width	12
Minimum Metal 1 to Metal 1 spacing	0.12
Minimum Metal 1 to Contact enclosure	0

# Appendix C (Some Most Commonly Violated Design Rules for gpdk090 technology)

Minimum Metal 1 to Contact enclosure on two opposite sides of the Contact	0.06
Minimum Metal1 area	0.07
Minimum Metal k width	0.14
Maximum Metal k width	12
Minimum Metal k enclosure of Via k-1	0.005
Minimum Metal k enclosure of Via k-1on at least two opposite sides	0.06
Minimum Metal k area	0.08
Minimum and maximum Via k width	0.14
Minimum Via k to Via k spacing	0.15
Minimum Metal k to Via k enclosure	0.005
Minimum Metal k to Via k enclosure on at least two opposite sides of Via k	0.06
Minimum of four Via k with spacing $\leq 0.30 \mu m$ or nine Via k with spacing $\leq 0.60 \mu m$ are required when connecting Metal k and Metal k+1 when one of the Metals has a width $> 1.0 \mu m$ at the connection point	
Minimum Nwell width	0.6
Minimum Nwell spacing to Nwell (same potential)	0.6
Minimum Nwell spacing to Nwell (different potential)	1.2
Minimum Nwell spacing to N+/P+ Active Area	0.3
Minimum Nwell enclosure of N+/P+ Active Area	0.12

# **EEE 4134 VLSI I Laboratory** Lab 4

# DRC, LVS, RCX and Post-layout simulation of an inverter

#### **Objectives:**

- To perform Design rules check (DRC), Layout vs. Schematic check (LVS) of inverter • layout
- To extract parasitic resistance and capacitance from layout of designed inverter •
- To perform transient simulation of extracted view
- To create layout views for body ties of NMOS and PMOS for further use •

### DRC Rules check by Cadence's ASSURA

1. Now we would like to check the DRC rules by ASSURA. Execute Assura *>Technology*. In the

following window, type the path of 'Assura Technolgy File' as shown. Click OK. X Assura Technology Lib Select

X Assura Technology Lib S	elect	_		$\times$
Assura Technology File	/tech/process/gpdk090/assura_tech.lib		View)	Edit
	<u>O</u> K <u>C</u> ancel <u>A</u>	pply Defa	ults	<u>H</u> elp

2. Execute Assura  $\rightarrow$ Run DRC. A DRC window appears as shown below. Fill the form as indicated in the picture.

Give a Run name. Select 'gpdk090' under 'Technology'. Then click OK.

#### EEE 4134 VLSI I Sessional

Run Assura DRC ×				
Layout Design Source DFI Compare two layouts Generate LvL Compare Rules Library mylib Cell inverter View layout Browse Save Extracted View View Name drc_extracted Area To Be Checked Full				
Run Name     drc1     Run Directory     .        Run Location     local				
View Rules Files       Technology       gpdk090       Rule Set       default         Rules File       /buet/cadence/gpdk090_v4. 6/assura/drc. rul       View       Reload         Switch Names       Set Switches       Set Switches         RSF Include       View       View         Variable       Value       Default       Description				
View avParameters       Modify avParameters       8 avParameters are set.         View Additional Functions       Image: Construction of the set				
Enable limitDrcCheck				

**3.** A DRC completed window appears as shown below, after completion of DRC run:



Click Yes.

**4. 'Error layer Window'** (ELW) appear as shown below with *INVX1* layout window which shows the errors.

K Error Layer Window -		
File View Error-Visibility Show Error by Help	cādence	2
[2] VIA1.X.1: Metal1 must connect to Metal2 with >= 2 Via1 spaced < 0.30 um 0 SignOff 0	Fixed	)
Rules Filter:		F
[2] VIA1.X.1: Metal1 must connect to Metal2 with >= 2 Via1 spaced < 0.30 2 INVX1 layout mylib		_
[2] NIMP.E.4: Nimp to gate end enclosure must be >= 0.18 um		
[2] PIMP.E.4: Pimp to gate end enclosure must be >= 0.18 um		

**5.** To correct the errors, find the error location by clicking on the error and then clicking on the right arrow key on ELW. To hide all the errors, click on '**NV**' (no Layers visible) button.

The particular error in the following figure is due to a lower than 0.18  $\mu$ m enclosure of gate by Pimp (drw) layer. Stretch the Pimp layer to correct it. Similarly, do this for all errors in your layout. No one can list all the errors one may commit, try correcting one after another and be familiar with them by solving them through practice.



After correcting all the errors, run the DRC again.

**6.** If your design is error free, you should get the following message.



# LVS check by Cadence's ASSURA

### **1.** Execute *Assura* →*Run LVS*. Select gpdk090 and give a **Run name**.

	Run Assura LVS ×		
Schematic Design Sc	ource DFII 🔽 Use Existing Netlist 🗌 Netlisting Options Use Verilog Top Cell		
Library mylib	Cell inverter View schematic Browse		
Layout Design Sourc	e DFII 🔽 Use Existing Extracted Netlist		
Library mylib	Cell inverter View Layout Browse		
Run Name 1vs2	Run Directory .		
Run Location	local 🔽		
View Rules Files	🗹 Technology gpdk090 🔽 Rule Set default 🔽		
Extract Rules	<pre>pt/cadence/gpdk090_v4.6/assura/extract.rul View Reload</pre>		
Compare Rules	<pre>iome/buet/cadence/gpdk090_v4.6/assura/compare.rul View</pre>		
Switch Names	Set Switches		
Binding File(s)	/home/buet/cadence/gpdk090_v4.6/assura/bind.rul View		
RSF Include	<pre>&gt;/buet/cadence/gpdk090_v4.6/assura/LVSinclude.rsf View</pre>		
Variable None	Value Default Description		
View avParameters	Modify avParameters 7 avParameters are set.		
View avCompareRules 🗌 Modify avCompareRules 1 avCompare rule is set.			
View Additional Fund	ctions 🗌 No additional functions are set.		
ОК	Cancel Apply Defaults Load State Save State View RSF Help		

#### Click OK.

**2.** After completion of lvs run, you will get a result window. If you have done everything right, it will say that Schematic and Layout match.

Suppose that, you have done a mistake, your schematic says the PMOS width is 480n where layout says it is 240n. What will happen in LVS report? Let's see.

\_

×

X Run: "lvs1" has completed SUCCESSFULLY!

autoPinSwap() results for schematic network. \_\_\_\_\_ ====File: lvs1.cfr \_\_\_\_\_ The LVS run "lvs1" has completed successfully. Compare problems were detected in 1 cells. 1 cells had parameters mismatches. 0 cells matched No Extraction Problems were detected. You currently have an open run (project). Press "OK" to close this run and enter the LVS Debug Environment. Press "Cancel" to leave this run open and close this Dialog box. LVS Run "lvs1" is located in /home/fall16/120105001/cic/ Next Warn. Next Error Prev. Error Prev. Warn.

Note that, LVS report says that 1 cell had parameter mismatch. That means one cell had two different dimensions in schematic and layout. Click **OK**. 'LVS **Debug**' window will appear:

C LVS Debug - lvs1	-		$\times$
File View Options Tools <u>H</u> elp		cāde	nce
Cell List (sch    lay) 💿 Extract 💿 Compare Summary (sch	lay)		
INVX1{mylib}    INVX1{my			
Open Schematic Cell Open Layout Cell Open	Tool		

**3.** Select INVX1 (mylib). Notice that summary window shows the list of errors.

Cell List (sch    lay)	🔾 Extract 🌾	🥑 Compare	Summary (sch	lay)
INVX1(mylib)		INVX1 (my	Rewires Nets Devices Pins Parameters	0 0 0 0    0 1
$\leq$	II			

4. Only parameter mismatch has occurred. Click on Parameters and Open Tool.

Summary (sch	lay)
Rewires	0
Nets	0
Devices	0
Pins	0    0
Parameters	1
Open	Tool

#### 5. 'Parameters mismatch tool' will open.

C Parameters Mismatch Tool		– 🗆 X
File Options View Tools <u>H</u> elp		cādence
Sch  Lay Cells [INVX1{mylib} Message		INVX1{mylib}
Schematic Info (/)	Layout Info (/)	
Name PMO pmoslv	Name avD7_1 pmos1v	
Probe Zoom Remove	Probe Zo	om Remove

6. Select PM0 pmos1v. Now in the Message box, you can see the mismatch error.



**7.** Change the width of PMOS to the value of width in layout and run LVS again. This time you will get an LVS match.

X Ru	n: "lvs1"	—		×
	Run: "lvs1" from /home/fall16/120105001/cic/			
	Schematic and Layout Match. You currently have an open run (project).			
	Do you want to close current project and view th	ne result	is of new	v run?
	Summary of LVS Issues			
	Extraction Information:			
Ő	O cells have 0 mal-formed device problems O cells have 0 label short problems O cells have 0 label open problems			
	Comparison Information:			
	0 cells have 0 Net mismatches 0 cells have 0 Device mismatches 0 cells have 0 Pin mismatches 0 cells have 0 Parameter mismatches			
	ELW Information:			
	Total DRC violations: 0			
	Yes No Help			

# Parasitic Extraction by Quantus QRC

(In this section, the inverter cell name is given as INVX1, which is inverter in your case)

**1.** Execute *Assura*  $\rightarrow$ *Open Run*. Select the final error free lvs run name in the run name field (it automatically loads the last lvs run). Click **OK**.

X Open Run			—		$\times$
Run Directory	II.				
Run Name	Ivs1 🔽	LVS Debug Environment LVS Error Report	ERO	C Brows	er
			<u>C</u> anc	el 🤶	<u>H</u> elp

2. Execute Assura →Run Quantus QRC. Select Extracted View in the output field under Setup tab.

Quantus QRC (Assura) Parasitic Ext	action Run Form		-	]
Setup Extraction Filt	ering Netlisting Run Details	Substrate		
Technology gpdk090 - p2lvsSet NONE - Setup Dir /tech/proc	RuleSet default UseMultRuleSets ess/qpdk090/assura/rcx			
Include Command File			View Ed	it)
Rule Command File Include			View Ed	it)
Tech Cmd File			View Ed	it)
_			View Ed	it)
_ /				
Output Extracted View	Lib mylib Cell INVX1	View av_	extracted	£
Enable CellView Check	⊻			
Parasitic Res Component	presistor	Prop Id	r	
Parasitic Cap Component	pcapacitor	Prop Id	С	
Parasitic Ind Component	pinductor	Prop Id	1	
Parasitic M Component	pmind	Prop Id	k	
Inductance L1 Prop Id	ind1 Inductance	L2 Prop Id	ind2	
Parasitic CCVS Component	ccvs Hgain Prop Id hgain	Vref Prop Id	vref	
Parasitic VS Component	vsource	Prod Id	vr	
Call Procedure				
Substrate Extract	Substrate Profile	N	IONE 🔽	
Add LVS MOS Diffusion Res	Extract MOS Diffu	sion Res		
& norary precedry: spectry	a directory for writing focal in	JIALIES CLEAC	eu	-
uring the hierarchical ext	raction of an extracted view.			- j

**3.** Go to **Extraction** tab. Select **RC** as **Extraction type** and put the name of your reference node (**gnd!** in the given case) and click **OK**.

Setup Extraction Filtering Netlisting	Run Details Substrate
Extraction Type RC	Name Space 🛛 Layout Names 🔽 🦯
Max fracture length infinite microns 🔽	Temperature 25.0 C Edit
Cap Coupling Mode Decoupled	Ref Node gr.d.!

**4.** After completion of the run, you will get the following message:



5. Execute *File* →*Open* and fill in the form as follows. Click **OK**.

X Open F	ïle		_		×			
File Library Cell View Type	mylib  mylib  av_extracted  layout Browse	Cells INVX1 inverter inverter: inverter:	_					
Open with	s use this application for this type of file							
Open for	● edit ⊖ read							
Library pat	Library path file /home/fall16/120105001/cic/cds.lib							
			<u>C</u> ano	el	<u>H</u> elp			

Extracted view will open.



6. Now Launch ADE L from the **av\_extracted** view and setup everything other than outputs to be plotted in the same way as you have done in Lab 1. After setting everything else, execute *Outputs*  $\rightarrow$  *To be plotted*  $\rightarrow$  *Select on design*. Go to **av\_extracted** view. Click on the **in** pin location on that view. The following window will appear. Select pin name **in** and click **OK**. Do the same for **out** pin.

ò	Select nets/terminals			—		$\times$	
							Δ
	/in						Ē
						1	7
<		1111				$\geq$	
			<u>о</u> к	<u>C</u> an	icel	<u>H</u> elp	

Then run the simulation and observe the output waveforms. Then measure power and delay using waveform calculator.

## # Exercise

1. Analyse the effect of parasitic RC elements on the power consumption and propagation delay of an inverter.

# EEE 4134 VLSI I Laboratory Lab 5

# Schematic Driven Layout of a 2-input NAND gate using Virtuoso Layout Suite Editor XL

#### **Objectives:**

- To be familiar with schematic-driven layout with the example of a 2-input NAND gate.
- To perform Schematic Level Verification, Layout Design, DRC and LVS check and perform post-layout simulation from extracted view

# **Creating Layout using Virtuoso Layout Editor XL**

**1.** Virtuoso Layout Editor XL is a schematic-driven layout generation tool. To learn schematic driven layout we will create the schematic view of a 2-input NAND gate cell which we named *NAND2X1*.

#### 2. Instantiate the following cells to your schematic.

Library Name	Cell Name	Properties/Comment
gpdk090	nmos1v	For NM0 and NM1, Width $= 240n$
gpdk090	pmos1v	For PM0 and PM1, Width = 480n
analogLib	vdd	
analogLib	gnd	

Use your experience from Lab1 to draw the schematic diagram of the nand gate.


**3.** Launch **ADE** L and simulate the design to verify its functionality. Setup **Model library**, **Analysis** type and **Outputs to be plotted** as you have done in Lab1.

While setting inputs for signals A and B, you have to use different periods and delays for the two signals, so that you can observe all four cases (00, 01, 10, 11) of input signals. Also make sure  $0 \rightarrow 1$  and/or  $1 \rightarrow 0$  transitions for both input signals do not occur at the same time. The following figure shows a sample of two signals meeting these criteria:



A sample waveform window would look like the following after simulation:



Check the functionality of the schematic (whether it acts like a nand gate).

**4.** Then create a symbol in the same way you have made symbol of inverter in lab2. Make it look like a nand gate.



5. In schematic editor window, execute: *Launch →Layout XL*. The following window will appear:

X Startup Option	—		$\times$
Physical Implemen	ntation S	itartup O	ptions—
Create New	Ope	en Existii	ng
Configuration Create New Automatic	Ope	en Existii	ng
<u>o</u> k	<u>C</u> ar		Help

6. Click OK. 'New File' window for layout will appear. Click OK.

X New File		-		×
- File				
Library	mylib			
Cell	NAND2X1			
View	layout			
Туре	layout			
Application				
Open with	Layout XL	-		
🔲 Always use th	is application	for this	type of	file
Library path file				
/home/fall16/1	.20105001/c:	ic/cds	.lib	
	<u>0</u> K	Can	cel )	<u>H</u> elp

'Virtuoso Layout Editor XL' window will appear.

Virtuoso® Layout	Suite XL	Editing:	mylib NA	ND2X1	layout								_		]	×
<u>L</u> aunch <u>F</u> ile <u>E</u> dit	<u>V</u> iew	<u>C</u> reate	Ver <u>i</u> fy	Co <u>n</u> n	ectivity	<u>O</u> ption	s <u>T</u> ool	s <u>W</u>	indow	Ass	<u>u</u> ra	Opti <u>m</u> i:	ze	»cā	der	nce
1 🗁 🗔 🗌 🥱	¢   •	<b>₽</b> 0		× į	R (		1	»			»	Q	»    (	Classic		<b>v</b> »
IR 🧠 👙 🕂	- 🛞	🙅	🏰 🗕		4	»	(F)Sele	ect:0	Sel(N	4):0 S	Sel(I):I	) Seli	(0):0			>>
Palette	? 🗗 🗙															
Layers	Ð×															
🖌 Valid 📄 Used 📄	Routing	- I - I														
<b>Q</b> Filter	-															
Nwell drawing																
AV NV AS	NS											•				
Name Vi	s Sel	T I														
All Layers 💽 🕂	-+-	j 🖂														
∧  Layer  Pu  V	'  S [_	1.1														
📕 Nwell 🛛 🖌 🖌																
🎬 Oxide 🛛 drw 👿																
🔳 Oxide_thk drw 👿																
🎆 Poly 🛛 drw 🖳 📉 Pimp drw 🕞	: -															
Objects	Ð×											•				
Objects V																
Instances 🛛 🖳		1														
Pins 💽 Vias 💽																
Objects Grid		-														
I 🗛 🌵 🗄 🕻			22 Po	<b>ili</b>	Ħ			<b>F</b> b		<b>-</b>	7	<b>→</b>		-	4	×
131 着 🕹 🍕		2		<b>W</b>			¥	V	<u></u>	34	2					

7. Execute *Connectivity* -> *Generate All From Source*. The following pop-up window will appear:

0 - u - u - t - 1						
Generate	I/O Pins	PR Boundary	Floorpla	n		
Generate —						
🕑 Instance:	3					
📃 Chair	n 🗌 Fold 📃	Chain Folds				
🕑 I/O Pins						
📃 Exce	pt Global Pins					
📃 Exce	pt Pad Pins					
🕑 PR Boun	· ·					
📃 Snap	Boundary					
Minimum Se		12 🗹 In Bour	ndary			
Minimum Se Device Com	espondence		ndary			
Minimum Se Device Com			ndary			
Device Com	<b>espondence</b> User-Defined		ndary			
Minimum Se Device Corr Preserve Connectivity	<b>espondence</b> User-Defined	Bindings	ndary			
Minimum Se Device Corr Preserve Connectivity	espondence – User-Defined / Extraction –	Bindings	ndary			
Minimum Se Device Corr Preserve Connectivity	espondence – User-Defined / Extraction –	Bindings	ndary			
Minimum Se Device Corr Preserve Connectivity	espondence – User-Defined / Extraction –	Bindings	ndary			
Minimum Se Device Corr Preserve Connectivity	espondence – User-Defined / Extraction –	Bindings	ndary			
Minimum Se Device Corr Preserve Connectivity	espondence – User-Defined / Extraction –	Bindings	ndary			

**8.** Go to **I/O pins** tab. The dialog box shows that all I/O pins are in **Metal1** layer (**Metal1 drw**). Also put a tick mark on **Create label as Label**. Click **Options**. Set **Height** to 0.1.

Generate Layou	ıt				-	-		$\times$
Generate	I/O Pins	PR Bou	ndary F	Floorplan				
Specify Defa	ult Values fo	or All Pins –						_
	Layer:		Width:	Height: N	Num: Cre	ate:		
	Metal	L1 drw	- 0.12	0.12	1 🗹		Apply	
Specify Pins	to be Gener	ated						5
Select:		N	umber Of Mat	ches: O	Add	1 New	Pin	
Term Name	Net Name	Layer		Width	Height	Num	Create	
"A"	"A"		"drawing")		0.12	1	t	
"B" "V"	"B" "Y"		"drawing")		0.12	1	t	
"gnd!"	"gnd!"		"drawing") "drawing")	0.12 0.12	0.12 0.12	1 1	t t	
"vdd!"	"vdd!"		"drawing")	0.12	0.12	1	t t	
Pin Label	Meta	dl1 drw	• 0.12	0.12	1		Jpdate	
Min Lawer	abel As: 🖲 l	abel	Options					
	01	Fext Display						
	X	Set Pin Lab	<u>_</u>		cel C	)efault:	<u>с н</u>	elp
	Heig	gnt	0.1					
	Fon	t	stick	2				

Click OK, and OK.

9. The initial pin and transistor placement in layout will look like the following:



**10.** Execute *Options* →*Display* or Press 'e' on keyboard to open '**Display Options**'. Fill it in as shown:

X Display Options			_		$\times$
Display Controls		- Grid Controls -			
🗹 Open to Stop Level	🔲 Nets	Type 🔾 non	e 🥑 do	ots 🔾 lin	es
🗹 Axes	Access Edges	Dim Major Dots			
🔲 Instance Origins	🔲 Instance Pins	· ·	_	_	_
🗹 EIP Surround	🔲 Array Icons	Minor Spacing		0.01	_
🔲 Pin Names	🗹 Label Origins	Major Spacing	1	0.1	
🔲 Dot Pins	🔲 Use True BBox	X Snap Spacin	ig 🗍	0.005	
🗹 Net Expressions	Cross Cursor	Y Snap Spacin	ig 🗍	0.005	
Stretch Handles	🔲 Row Name				
🔲 Via Shapes	Row Site	Filter			
🗹 Dynamic Hilight	🗹 True Color Drag	Size 6	Style	empty	
🗹 Dragged Object Ghost	🔲 Transparent Group		otyte		
🔲 Traversed instance BBox	Selection Hint	- Snap Modes -			
Maximum Drag Level 32		Create	orthog	onal 🔽	
Maximum Number of Drag Fig	gures 500	Edit	orthog	onal 🔽	
Scroll Percent 25					
Instance Drawing Mode	Box 🔽	<b>Dimming</b> Enable Dimmin	a		
			none	-	
Path Display Borders an	nd Centerlines 🔽	Scope	lione		
Set LPP Visibility Do not che	eck validity 🧧	Dim Grid Lines			
Show Name Of O instance	🖲 master 🔾 both	Automatic Dimr	ning	<b>~</b>	
		Dim Intensity:			
Array Display Display			50		
Full     Start	0	I I I			_
Border     Source Stop	10	Dim Selected C			
Source Stop		True Color Sele	ection or	nly	
● Cellview ◯ Library ◯ T	ech Library 🔾 File 🛛 🤟	cdsenv		Bro	wse
Save To		Delete From			
Save Io	Load From	Delete From			
	OK Canc	el ) <u>D</u> efaults	) <u>A</u> p	ply ) ( <u>F</u>	<u>H</u> elp

**11.** The transistors and pins are shown inside a bounding box, which is an estimate of the optimum size of the final layout. Automatic router will use the bounding box to constrain all routing to occur within the box. The bounding box may need to be re-sized to accommodate all components. An important concept to keep in mind during resizing is that standard cells typically have fixed height (so that power/ground rails line up correctly for routing purposes).

#### Delete the PR Boundary for now.

**Virtuoso Layout Editor XL (VXL)** and gpdk090 allow us to create stacked transistors with shared source/drain areas. Zoom in to two transistors at the bottom (to zoom in, type "**z**" and draw a box around the transistors). Click on the transistor on the right and type "**m**" to move the object. As you start dragging the object to the left, fly-lines indicating connectivity will appear as shown below:



**12.** When the source/drain areas are overlapped, left-click to fix the position. You should see a transistor stack with shared source/drain areas like this (depending on how far you move, you may need to move left/right a bit):



This is a nice NMOS stack for the NAND gate. As you can see, the source/drain contacts have disappeared. Back to the big picture, zoom to fit (press "**f**").

Let's do the same exercise for the PMOS transistors. The PMOS transistors in nand gate do have shared drain contacts because they work in parallel. Connectivity information is extracted from schematic by VXL. The pull-up network looks like the following:



13. Now, connect different layers using path tool (press '**p**' on keyboard), and fill areas by drawing rectangles where necessary (press '**r**' on keyboard). To connect one layer to another (e.g. **Poly to Metal1** or **Metal1 to Metal2**), create via by pressing '**o**' on keyboard and selecting proper 'Via **Definition**'.

🗙 Create Via		-	$\times$
Mode    Single  Stack    Options  Compute From Shap  Net Name			
Create as ROD Object	ame viaO		
Via Definition M1_POv	Standard Via /	gpdk090	

**14.** Instantiate **M1\_PSUB** and **M1\_NWELL** cells (that you have created earlier) by pressing 'i' on keyboard and selecting the layout view from library browser.



**15.** Wire up the layout. When you do so, you may encounter multiple options for certain pins. For example, when you select the PMOS to connect its source to VDD, there are multiple Metal1 wires in the PMOS. The desired path will be highlighted and you'll see the fly-line. Continue until you finish routing all the signals. Move **vdd!** and **gnd!** pins to the power rails. As you are moving the pins around, notice the fly-lines that indicate the connections.



A practice that you can follow while wiring is to use **Metal1** for all vertical wiring and **Metal2** for all horizontal wiring inside the cell.

Also make the cell height 5  $\mu$ m.

**16.** Your final layout will look something like the following:



**17.** Perform DRC, LVS and QRC for NAND2X1 as you have done in Lab 4. Generate **av\_extracted** view and simulate the circuit from that view to verify the functionality.

# # Exercises

1. Analyse the difference between stacked and unstacked transistors in post-layout simulation and explain why we should stack transistors with common drain/source terminals.

# EEE 4134 VLSI I Laboratory Lab 6 Introduction to Hierarchical Design (2-input AND gate using 2-input NAND gate and an inverter)

#### **Objectives:**

- To be familiar with concept of hierarchical design
- To perform Schematic Level Verification, Layout Design, DRC and LVS check
- To perform post-layout simulation of top level design

# **Introduction to Hierarchical Design**

By this time, you should have completed layout of *INVX1* and *NAND2X1*. Now, you will learn how to perform hierarchical design. (cell *INVX1* in this section is **inverter** in your case)

**1.** Create a new cellview of type schematic named *AND2X1*.

**2.** Instantiate *NAND2X1* and *INVX1* symbol from your library *mylib* in that schematic. Your final schematic should look something like the following:



**3.** Now make a symbol of *AND2X1*.



**4.** Now create a new cellview named *AND2X1\_test*, instantiate *AND2X1* and *vdd* in that cell and the final schematic should look like the following:



**5.** Now, launch **ADE L**, setup **Stimuli**, **Model library**, **Analysis** type and **Outputs to be plotted** in the same way as you have done in Lab 5. Run the simulation.



6. If functional verification is okay, then execute Launch  $\rightarrow$  Layout XL from the schematic of AND2X1.

**7.** Follow procedure of Lab 5 to generate instances and set display options. You will get something like the following:



8. Connect them as required and the final layout should look like the following (probably better!):



**9.** Perform DRC, LVS and QRC as you have done in Lab 4 and Lab 5.

# EEE 4134 VLSI I Laboratory Lab 7 Introduction to Verilog HDL and Quartus II

**Objectives:** 

- To get familiar with Quartus II
- To get introduced to Hardware Description Language (HDL)
- To understand behavioral and structural Verilog descriptions

# Verilog HDL

A hardware description language (HDL) is similar to a typical computer programming language except that an HDL is used to describe hardware rather than a program to be executed on a computer. Two HDLs are IEEE standards: Verilog HDL and VHDL (Very High Speed Integrated Circuit Hardware Description Language).

# **Creating a New Project**

**1.** Run Quartus II by clicking on the shortcut icon on desktop named *Quartus II 9.0sp1 Web Edition*. A getting started window may open up. You may put a tick on '**Don't show this screen again**', if you do not want this to appear again.

**2.** Each logic circuit or sub-circuit being designed with Quartus II software is called a *project*. The software works on one project at a time and keeps all the information for that project in s single directory (folder) in the file system. To begin a new logic circuit design, the first step is to create a directory to hold its files. Create a folder in  $D:\$ ,  $E:\$  or  $F:\$  drive named after your student ID. Execute File  $\rightarrow$ New Project Wizard.

**3.** In the following window, change the working directory of the project to your directory (e.g. D:\120105001) and give a name to the project as shown. Note that **the project must have a name**, which is usually the same as the top-level design entity that will be included in the project. Click *Next*.

	me, rop cerei em	ity [page 1 of	5]	
What is the working directory for t	his project?			
D:\Verilog				
What is the name of this project?				
mux_2to1				
What is the name of the top-level exactly match the entity name in th		project? This na	me is case sensi	itive and mus
mux_2to1				
Use Existing Project Settings	1			
	1			

**4.** We can specify any existing file to be included in the project, if we want, in the next window. But we will skip it for now. Click *Next* on the next three windows that open up. Then in the last window, click *Finish*.

# Creating a new Verilog HDL file under a project (Design Entry), Compilation and Simulation

1. Execute *File* →*New*. A window will open up. Select *Verilog HDL File* and click *OK*.

**2.** The following Text Editor Window will open up and you can write your Verilog code in here and **save it using the same name as the project**.

🕸 Veril	og1.v			
Werling         Werling	1			
	<u>ح</u>			•

3. Write your Verilog code. This example shows Verilog code for a 2to1 MUX.



The syntax of Verilog code is sometimes difficult for a designer to remember. To help with the issue, the Text Editor provides a collection of Verilog templates. The templates provides examples of various types of Verilog statements, such as a module declaration, an always block, and assignment statements. It is worthwhile to browse through the templates by selecting *Edit*  $\rightarrow$ *Insert Template*  $\rightarrow$ *Verilog HDL* to become familiar with this resource.

4. Click on the purple play button to start compilation of your Verilog code.



**5.** After successful compilation, you will get the following message. Ignore the Warnings for now. Click *OK*.



If the compiler does not report zero errors, then there is at least one mistake in Verilog code. In this case, a message corresponding to each error found will be displayed in the **Messages** window. Double-clicking on an error message will highlight the statement which is affected by the error, in the Verilog code in the Text Editor window. The user can obtain more information about a specific error by selecting the error and pressing the F1 function key. Correct the error and recompile the design.

6. Now, execute  $File \rightarrow New$  to create a vector waveform file which is required for simulating inputs and outputs. Select *Vector Waveform File* from the list and click *OK*.

7. The following window will open up.

alue at 5.08 ns	10.0 ns	20.0 ns 15.075 ns	30,0 ns
5.08 ns		15.075 ns	
1			
	) < III.	• < m	

8. Double-click on the white space under 'Name|Value at 15.08 ns'. Or Right-click on that space and select *Insert* →*Insert Node or Bus*.

9. In the 'Insert Node or Bus' window, click Node Finder.

Name:	J		ОК
Туре:	INPUT	•	Cancel
Value type:	9-Level	•	Node Finder
Radix:	ASCII	•	
Bus width:	1		
Start index:	0		

10. In the 'Node Finder' window, Click List. Make sure 'Pins:all' is selected under 'Filter'.

**11.** Now the window will look like the following one. Click on the '>>' Button.

de Finder	I line	in the second se		_Σ
amed: 🛛 *	Filter: Pins: all	Customize.	List	OK
ook in: Imux_2to11		💽 🔽 Include sube	ntities Stop	Cancel
odes Found:		Selected Nodes:		
Name	Assignments	Name	Assignments :	
►A	Unassigned I	-15 - 10 - 10 - 10 - 10 - 10 - 10 - 10 -		
▶B	Unassigned I			
⊳s	Unassigned I			
€Y	Unassigned (			
	>	1		
	>>			
	<			
	<<			
m	•	۲ III		
m		1	•	

**12.** Now, it should appear like the following. Click *OK*.

		and the second sec		
lamed: 🛛 🛛	▼ Filter: Pins: all	Customize	List 👝	OK
ook in: <mark>[mux_2to1]</mark>		👻 🔽 Include subentitie	s Stop	Cancel
lodes Found:		Selected Nodes:		5
Name	Assignments	Name	Assignments	
►A	Unassigned I	imux_2to1 A	Unassigned I	
в	Unassigned I	Imux_2to1 B	Unassigned I	
₽S	Unassigned I	Imux_2to1 S	Unassigned I	
ĭ⊇Y	Unassigned (	Imux_2to1[Y	Unassigned (	
	>>> >>> <	Ī		
<		< III.	•	

**13.** In the following window, click *OK*.

Name:	**Multiple Items**		OK
Туре:	**Multiple Items**	•	Cancel
Value type:	9-Level	-	Node Finder.
Radix:	ASCII	•	
Bus width:	1		
Start index:	0		

**14.** The vector waveform file will now look like the following:

1	Master	Time Bar:	15.075	5 ns	Pointer:	0 ps	Interval:	-15.08 ns	Start
A			Value at	0 ps		10.0 ns		20.	0 ns
£€				15.075				75 ns	
	<b>10</b> >0	A	A 0				T		
	1	В	A O						
的线	<b>₽</b> 2	S	A O	2		8			
₩ 🕅	<b>@</b> 3	Y	AX	×***	*********	*********	*******	*********	******
-0 L						1			

Now, you can clearly see the inputs and outputs.

15. Select an input and from the left palette, click on the 'Overwrite clock' icon.



16. In the 'Clock' window, set parameters of the clock.

Start time: 🛛	j	ps	•
End time: [	1.0	us	-
Base wavefo	m on		
C Clock set	tings:		
l			
Time period	od:		
Period:	10.0	ns	•
Offset:	0.0	ns	-
Duty cyc	le (%): 50	<u>.</u>	

**17.** Now, after setting all the input clocks, **Vector Waveform File** will look like the following. Note that, the output Y is displayed as having an unknown value at that time, which is indicated by a hashed pattern; its value will be determined during simulation.



#### 18. Save the Vector Waveform File. It must have the same name as the Verilog file.

**19.** A designed circuit can be simulated in two ways. The simplest way is to assume that there is no delay in propagation of signals through the circuit. This is called *functional* simulation. A more complex way is to take all propagation delays into account, which leads to *timing* simulation. Typically, functional simulation is used to verify the functional correctness of a circuit as it is being designed. This takes much less time, because the simulation can be performed simply by using the logic expressions that define the circuit.

Click on the blue play button and you should observe the simulated waveforms.

200 B 200 B 200	Waveforms								
liation m	ode: Timing								
Master	Time Bar:	15.075 ns	• Point	ter:	Interva	Ŀ	Start:	End	
		Value at	0 ps	10.0 ns	20.0 ns	30.0 ns	40.0 ns	50.0 ns	60.0 n:
	Name	15.08 ns		15.07	75 ns				
0	A	A1							
1	В	A 1							
2	S	A O				_			
<b>@</b> 3	Y	A 1							
			1						

Note there is a delay between input and outputs which simulates the real effect of gate delays. If you are interested in only functional analysis rather than timing analysis, go through some more steps.

#### **20.** Execute *Processing* →*Simulator Tool*.

21. In the following window, select *Functional* as *Simulation Mode* and click on *Generate Functional Simulation Netlist*.

🚯 Simulator Too	bl			
Simulation mode:	Functional	•	Generate Fun	ctional Simulation Netlis
Simulation input:	mux_2to1.vwf			Add Multiple Files

22. After generating functional simulation netlist, the following window will appear. Click OK.



**23.** Click on the blue play button and you should observe the following message.



Quartus II	
i	Simulator was successful
	ОК

**24.** Now observe the simulation waveforms from *Processing*  $\rightarrow$ *Simulation Report*, and note that there is no time delay between inputs and output and the function of this code is indeed that of a 2to1 MUX.

Master	Time Bar:	15.075 ns	Pointer:	1.53 ns	Interval:	-13.55 ns	Start	End:	
	Name	Value at 15.08 ns	0 ps 10	0.0 ns 2 15.075 ns	20.0 ns	30.0 ns	40.0 ns	50.0 ns	60.0
0	A	A 1		- 1-					
<b>i</b> ≥1	В	A 1							
<b>i</b> ▶2	S	AO							
	Y	A1			1				

# **QUESTIONS:**

1. Which ones of the following identifier names are incorrect according to Verilog syntax?

2to1MUX, MUX\$2to1, mux\_2to1, reg, reg4bit, 130105\_counter

2. What are the differences between 'structural' and 'behavioral' Verilog codes? Show an example of each for a particular logic circuit.

# EEE 4134 VLSI I Laboratory Lab 8

# Combinational Logic circuit design in Verilog HDL using Quartus II

**Objectives:** 

- To design combinational circuits with Verilog codes, and verify the codes through simulation
- To learn about hierarchical design using Verilog HDL

```
Verilog codes for combinational logic circuits
```

```
Half-Adder
module half adder(A,B,Cout,S);
input A,B;
output S, Cout;
assign S = A^B;
assign Cout = A&B;
endmodule
2-to-1 MUX
module mux21 fn(I0,I1,S,f);
input I0,I1,S;
output reg f;
always@(I0,I1,S)
begin
if (S == 0)
f = I0;
else
f = I1;
end
endmodule
Priority Encoder
module priority(W, Y, z);
input [3:0]W;
output reg [1:0]Y;
output reg Z;
always @(W)
begin
z = 1;
casex (W)
4' b1xxx: Y = 3;
4'b01xx: Y = 2;
4'b001x: Y = 1;
4'b0001: Y = 0;
default: begin
z = 0;
Y = 2'bx;
end
endcase
end
endmodule
```

#### **Full Adder using Half-Adders**

```
module FA_001(A,B,C,Cout,S);
input A,B,C;
output Cout,S;
wire C1,C2,S1;
HA_001 f1(A, B, C1, S1);
HA_001 f2(S1, C, C2, S);
assign Cout = C1|C2;
endmodule
module HA_001(a,b,c,s);
input a,b;
output c,s;
assign c = a&b;
assign s = a^b;
endmodule
2-to-4 Decoder
```

```
module dec2to4(W, En, Y);
input [1:0]W;
input En;
output reg [0:3]Y;
integer k;
always@(W, En)
for(k = 0; k < = 3; k = k+1)
if ((W == k) && (En == 1))
Y[k] = 1;
else
Y[k] = 0;
endmodule
```

## # Exercises

- **1.** Explain the differences between 'concurrent' and 'procedural' statements? Show an example of each.
- **2.** Examine whether the following two code segments will yield different outputs. Explain your reasoning.



3. Read the following Verilog code and find out what it does. State your reasoning.



- 4. Write Verilog codes for
  - a) BCD Adder
  - b) 4-to-1 MUX using 2-to-1 MUX
  - c) 8-to-1 MUX using 4-to-1 MUX, 8-to-1 MUX using 2-to-1 MUX
  - d) 16-to-1 MUX using 8-to-1 MUX, 16-to-1 MUX using 4-to-1 MUX
  - e) 4-bit comparator
  - f) 4-bit Full-Adder using 1-bit Full-Adder modules
  - g) 4-bit adder/subtractor
  - h) 4-bit Arithmetic logic unit (ALU) with 8 functions

# EEE 4134 VLSI I Laboratory Lab 9 RTL synthesis and Sequential Logic Circuit design in Verilog HDL using Ouartus II

#### **Objectives:**

- To get familiar with RTL Synthesis in Quartus II
- To design sequential logic circuits in Verilog HDL, and verify the codes through simulation

# **RTL Synthesis in Quartus II**

Logic synthesis is a process by which an abstract form of desired circuit behavior), is turned into a design implementation in terms of logic gates, by a synthesis tool. Common examples of this process include synthesis of HDLs, including VHDL and Verilog. Some synthesis tools generate bit-streams for programmable logic devices such as PALs or FPGAs, while others target the creation of ASICs. Logic synthesis is one aspect of electronic design automation.

In digital circuit design, **register-transfer level** (**RTL**) is a design abstraction which models a synchronous digital circuit in terms of the flow of digital signals (data) between hardware registers, and the logical operations performed on those signals. Register-transfer-level abstraction is used in hardware description languages (HDLs) to create high-level representations of a circuit, from which lower-level representations and ultimately actual wiring can be derived.

**1.** Write Verilog code for a logic circuit, compile it and verify its functionality through simulation.

Suppose, you have completed all these steps for Verilog code of a half-adder.

```
1 ■module halfadder (A,B,Cout,S);
2 input A,B;
3 output S, Cout;
4 assign S = A^B;
5 assign Cout = A&B;
6 endmodule
7
```

Then you can proceed to obtain RTL view for this logic circuit.

2. Execute *Tools* →*Netlist Viewers* →*RTL Viewer*. A window will appear and you can see the RTL view.



As you can see, this is the logic circuit for a half-adder.

#### Verilog codes for sequential logic circuits

#### Latch

module latch01(D,clk,Q);
input D,clk;
output reg Q;
always@(D,clk)
if (clk)
Q=D;
endmodule
Flip-flop

```
module flipflop(D,Clock,Q);
input D,Clock;
output reg Q;
always@(posedge Clock)
Q<=D;
endmodule
```

#### **Shift Register**

```
module shift3(w,Clock,Q);
input w,Clock;
output reg [1:3]Q;
always@(posedge Clock)
begin
Q[3]<=w;
Q[2]<=Q[3];
Q[1]<=Q[2];
end
```

#### endmodule

#### Counter

```
module count4(Clock,Resetn,E,Q);
input Clock,Resetn,E;
output reg [3:0]Q;
always@(posedge Clock, negedge Resetn)
if(Resetn==0)
Q<=0;
else if (E)
Q<=Q+1;
endmodule</pre>
```

## # Exercises

1. Evaluate out the function of following logic circuit and write a Verilog code to obtain the same logic circuit in '**RTL Viewer**'.



- 2. Write Verilog codes for a 4-bit up/down counter, which counts up if select pin=0 and counts down if select pin=1.
- **3.** Write Verilog code for a universal shift register which can shift left/right and can load data both in parallel mode and serial mode.
- 4. Write Verilog code for a J-K flip-flop using D flip-flop and a T flip-flop using J-K flip-flop.
- 5. Synthesize already written Verilog codes to obtain RTL diagrams.
- 6. Write the Verilog code for a logic circuit which counts to 3 if selection input (composed of 2 bits)  $S_1S_0 = 01$ , counts to 7 if  $S_1S_0 = 10$ , counts to 15 if  $S_1S_0 = 11$ . If  $S_1S_0 = 00$ , then it halts counting. Use positive edge-triggered counter.
- 7. Explain the differences between 'synchronous' and 'asynchronous' resettable flip-flops?
- 8. Find 10 mistakes in the following Verilog code which implements a 5-bit shift register:

9. Explain the differences between 'blocking' and 'non-blocking' assignments with examples.

# **References and Further Readings**

 CMOS VLSI Design: A Circuits and Systems Perspective (4th Edition) by Neil Weste, David Harris
 Fundamentals of Digital Logic with Verilog Design (3rd Edition) by Stephen Brown, Zvonko Vranesic
 Physical Design Essentials: An ASIC Design Implementation Perspective By Khosrow Golshan
 Digital VLSI Chip Design with Cadence and Synopsys CAD tools by Erik Brunvand
 Fully custom design tutorials of EEE Department, Hong Kong University
 VLSI I and II Lab Manuals of EEE Department, Bangladesh University of Engineering and

Technology (BUET) 7. Custom IC Design Manual provided by University Support Team, Cadence® Design Systems, Bangalore, India