EE 330 Laboratory 5 From Boolean Equation to Silicon

Contents	
Background Information:	2
Checkpoints	3
Part 1: 3 - Input NAND or NOR Gate	3
Part 2: Exchanging gates	3
Part 3: Boolean Function	4
Design Constraints	5
Part 4: Debugging	5
Appendix	7
Generating From Source	7

Background Information:

The objective of this experiment is to experience the design flow of a digital system from the system description through layout where more than one designer is a key contributor to the design. This will be approached by using a simple Boolean system and two designers.

In this experiment, one designer will create a 3-input NAND gate and a second designer will create a 3-input NOR gate. Both should create an inverter. By sharing the gate you create with a partner, you will have all three gates. As part of the prelab for this experiment you will select a Boolean function that can be implemented (without Boolean simplification) with 3-input NAND gates, 3-input NOR gates, and Inverters. You will then implement the Boolean function you selected in silicon (the layout) given area and pin constraints. You can re-use an inverter designed previously or design an inverter for use in this experiment.

Of particular importance when sharing these gates is that they are compatible with the overall floorplan you have for your layout. Horizontal rails for power (VDD) and ground are widely used in the layout of digital circuits with the logic blocks themselves placed between these rails. At a minimum, the floorplan of both designers should have the same distance between the VDD and ground rails.



of a simple circuit. In this experiment, we will take the system description of a system in the form of a simple Boolean expression and convert it to a layout for fabrication through MOSIS. Before you start the design, consider your layout technique, specifically the floorplan. This will allow you to save considerable time in later parts of the experiment.

When considering the floorplan you will be using to implement the Boolean function in Part 2.3, you will find it useful to **layout your gates so that it is easy to interconnect them** in the layout of the Boolean function. For example, in addition to having a **VDD rail (BUS) on the top** of your layout and a **VSS (Ground) rail on the bottom**, it might be convenient to place all **inputs on the left** and the **output to the right** in your cells. With this approach, layout of the Boolean function could involve running signal paths horizontally in the layout. If the layout becomes to long horizontally, pairs of VDD and VSS rails (still equally spaced) can be stacked vertically.



By having you and your partner agree on the floorplan, most importantly the spacing between VDD and VSS busses, the interconnection of the NAND gates, NOR gates, and inverters will be simplified in Part 3 of this experiment.

Checkpoints

The checkpoints for this lab are as follows:

- 1. NAND/NOR DRC and LVS from Part 1
- 2. Exchanged NAND/NOR gate
- 3. Boolean Function Gate-Level Schematic
- 4. Boolean Function schematic TB
- 5. Boolean Function DRC and LVS
- 6. Debugged Schematic, ADE, and working DRC and LVS

Remember, all checkpoints must be shown to a lab TA before the report is submitted. You should include these checkpoints in your lab report.

Part 1: 3 - Input NAND or NOR Gate

As discussed in the prelab, create an Inverter along with a NAND gate or a NOR gate as agreed upon between you and your partner. **Use pcells** (the instances from Lab 3) for the transistors when creating the layout. In a previous experiment you created a schematic and test bench, so following the same approach, create a **layout** of the gates you designed. Be sure your layout passes both DRC and LVS.

When creating the gates for this experiment, use minimum allowable widths and lengths for the NMOS and PMOS devices. Though not of emphasis in this experiment, sizing of devices that are larger than minimum are often preferred in many applications. It will be easiest to achieve such sizing requirements if pcells with multiplicity and fingers are used.

Part 2: Exchanging gates

Exchange gates using the following steps. Though these steps are described for exchanging the cell named "inverter", they can be used for exchanging other cells as well. You will need to

```
# Navigate to your lablib folder:
cd ~/ee330/lablib
# Modify the permissions of the files you want to transfer
chmod 777 -R ~/ee330/lablib
# Create an archive of the cell
tar cvf ~/inverter.tar inverter
```

"pack" the **inverter** cell in the **lablib** library in a single file. Close Virtuoso, open a new terminal window, and from command line run the following:

The file **inverter.tar** in the home directory now contains all the views required to use the **inverter** cell along with read, write and execute (rwx) access for others to use **inverter** and all subdirectories.

To send this file to another person, simply send it as an attachment to an email message, or a shared object in Google Drive. The recipient should then download it into their home directory. The following steps describe how the recipient can then use of **inverter** in their design. To keep the imported designs separate, create a new library called **exchange** (or some other name) and attach it to the AMI06 process. Then run the following:

```
# Move the inverter.tar from home to exchange:
mv ~/inverter.tar ~/ee330/exchange
# Move to the exchange directory
cd ~/ee330/exchange
# Extract the design
tar xvf inverter.tar
```

The inverter cell will now appear in the **exchange** library. Instantiate (create an instance of **inverter** in your schematic using the "i" hot key) the symbol and the **layout** views from this cell to use **inverter** in your own design.

Part 3: Boolean Function

Implement the Boolean function from the prelab by going through the complete design flow (schematic, symbol, test bench, layout, LVS). In this implementation, adhere to the constraints given below. Assign an appropriate name for the cell you create that implements the Boolean function. In this implementation, use the Inverter you created in the previous lab, your NAND or NOR gate from Part 1, and your partners NAND or NOR gate from their Part 1.

By re-using cells that were created, considerable time and effort should be saved since you should not need to place any individual transistors when implementing the Boolean function. The test bench should verify proper output values for all 8 input combinations of A, B, C.

Do not **simplify the Boolean expression** you have chosen. If you follow the floorplanning suggestion of placing VDD on the top, VSS on the bottom, inputs on the left, and outputs on the right, the layout of your logic circuit comprising the Inverter, NAND, and NOR gates, your layout may have gate placements that look something like that shown in Fig. 1 (depicted for 2-input rather than 3-input gates). Of course, the actual layout will be with transistors, not symbols for the gates.



Figure 1: Cell input, output, signal flow, and power directions

Design Constraints

- You may only use the NAND, NOR and inverters you previously created.
- The Boolean function you selected should not be simplified
- The completed cell should fit inside a square of dimensions 40µm×40µm. (Use a ruler)
- The inputs (A, B, C,) for the complete cell must be available at the left boundary of the cell and enter the cell with M1.
- The output Y of the completed cell must be available at the right boundary of the cell and emerge with M1.
- Use a 3µ wide M1 bus at the top of the complete cell for VDD and a 3µ wide M1 bus at the bottom of the complete cell for VSS. Use these rails to supply power to your gates. You may use different widths for the power busses internal to your cell or internal to individual gates.
- You may not use M3 or higher level metals anywhere.

Part 4: Debugging

Cadence Virtuoso has a number of powerful debugging tools available, whether that be detailed output logs or ways of obtaining information from different parts of a circuit. In the final portion of this lab, you will get practice with these tools and with debugging in Virtuoso. To begin, download the "DebugMe" and "DebugMe_TB" compressed files from the course webpage and add them to the "exchange" library that you created earlier in this lab. This will create two new cell views, aptly named "DebugMe" and "DebugMe_TB" which together contain the schematic, symbol, layout, testbench, and ADE state for the following Boolean function:

$$F = \overline{A} + B$$

A number of errors have been purposefully made in the provided files. The only files which do not have errors in them are the Boolean function schematic (the "schematic" view located in the "DebugMe" cell) and the Boolean function symbol (the "symbol" view located in the "DebugMe" cell). Do not change these cell views. Your task in the final part of this lab is to use the debugging tools available to you to correct the existing errors. For your checkpoint, you must show a testbench which proves the circuit works correctly as well as valid DRC and LVS results. You may not create a new testbench or ADE instance; use the provided items only (that is, do not create any new views). To open the pre-made ADE instance, click on the "spectre_state1", pictured below.

V	iew			
	7			•
٦	spectre_state1			•
11	View	Lock	Size	
	schematic		29	k
	spectre_state1		131	

In your lab report, in addition to showing your checkpoints, you must show screenshots of your layout, of your ADE setup screen, and of the testbench circuit. You must also state what changes you made in each.

TAs will provide limited help on this portion of the lab, although if you believe something may have gone seriously wrong or you are really stuck, you may ask and see what help will be provided. To aid you in your debugging efforts, the testbench simulation results that you should expect to receive are provided below. To receive full credit, your results should match the provided plot.



Appendix

Generating From Source

Sometimes, when doing layout, it is more convenient to generate components rather than instantiating them. Luckily, Cadence is powerful software, and allows us to do this. However, use of these tools may not be intuitive or user friendly. So, this guide in the appendix is meant to explain how to generate components in a layout and reduce the amount of work we do in cell placement and routing.

To generate our layout, first we will need a fully functional schematic; you'll probably want to have something that you know works, has been verified by a testbench, and only contains components that have layouts or standard cells. Now, launch Layout XL within Cadence like we would for any other layout. From here, navigate to the bottom left of the window, and press the "Generate All From Source" button, as seen below. This feature is also available under the "Connectivity" dropdown menu, under the "Generate" tab, as the "All From Source" button.

Launch File Edit View Create V	rify Connectivity Options Tools <u>W</u> indow Ass <u>u</u> ra Floorpl <u>a</u> n <u>P</u> lace <u>R</u> oute <u>H</u> elp	cādence
🖻 🛱 🖬 🕥 🦿 🚔	🗅 💷 🗶 💒 🛈 🔚 🕹 » 🏞 🚈 🛥 » 🔍 » Classic 🔤 🖏 🖓	off
IR, 🧠 🌮 🕂 🍕 🖓 🕂	門 🛄 💷 🗛 💊 📕 (F)Select:0 Sel(N):0 Sel(I):0 Sel(O):0 X −6.150 Y 25.500 dX	dY Dist: 3
Palette ? 🗗 🗙 💼		
Layers 🗗 🗙		
Valid 🔲 Used 🛄 Routing		
Filter 🔍 🗸	· · · · · · · · · · · · · · · · · · ·	
pwell drawing		
AV V NV AS V NS V		
NCSU Techlib ami06		
Name Vis Sel		
All Lavers		
Pu V S	· · · · · · · · · · · · · · · · · · ·	
pwell drw 🗹 🗹		
active dow 🖌 🖌		
mactive drw V		
👿 pactive drw 🗹 🔽 🗐		dd4/dd _
nselect drw 🗹 🗹 🔸		· · ·
🗖 pselect drw ⊻ ⊻		
poly drw 🗹 🗹 🔸		
elec drw 🗹 🗹 🖉		
metall drw 👱 🗹		
metal3 drw V		
cc drw 🖌 🖌		38/V8
🔤 via drw 🗹 🗹		
🧱 via2 drw ⊻ ⊻		
🚻 glass drw 🗹 🗹		
highres drw ⊻ ⊻		
🗙 noarc arw 👱 🗹		
noipe drw 🖉 🗹		
Objects d' ×		
Objects V S		
Instances 🗹 🗹 🔄		
Objects Grids		• • • •
là 🕂 🕆 🖾 🖾 🙀	: 🔁 📊 🗮 🔜 📨 🖽 🖼 💽 🖃 📝 📰	
🛤 🍺 🕹 羲 🖉 🗄	i 📽 🖉 🖫 🖳 🇊 🐱 🇊 🌾 🍫 🎘 🖳 🕅	
PreSel: Inst Name(I4) CellName(IN	/ERTER)	
mouse L: showClickInfo()	M: leHiClearRuler()	R: _lxHiMousePopUp()

Pressing the above button should open a menu. If you already have objects placed in your layout, it will ask if you want to proceed. If you don't care about what you've already placed, go ahead and hit "yes". If not, make a backup copy before you proceed.

There are plenty of tools available to us in the generation menu. There are two tabs that we will want to use every time we generate a layout: the I/O Pins tab and the PR Boundary tab. The I/O pins tab will allow you to specify the size, quantity, and layer that each pin will be placed on. The default configuration is usually good enough, and pins can be edited after they are generated. However, if you already have a plan in mind, this is where you can change pins before they are made. The PR Boundary pin will allow you to specify a boundary region for your layout; this is useful if you have a certain space constraint you must meet with your layout. You can specify shapes as well as a general shape and area utilization percentage.

		Generate Layout ×	Generate Layout
Generate VO P	Pins PR Boundary	Floorplan	Generate VO Pins PR Boundary Floorplan
-Default Value	es for Electrical Pins- Layer:	Width: Height: Num: Create:	Shape Origin (x y): (0.0 0.0) Polygon Points List (k1 y1)(x2 y2))
Specify Pins t	o be Generated	Number Of Matches: 0 Add New Pin	Area Estimation
Term Name A B c out vdd vss Name:	Net Name a b c out vdd vas Layer: metal1 drawing	Layer Mdth Height Num Create ('metal1' 'drawin 0.9 0.9 1 t ('metal1' 'drawin 0.9 0.9 1 t	Utilization (%) 25 Aspect Ratio (W/H) 1 Estimator: PR Boundary Based Register
Pin Label	bel As Label Text Dis	Options	

Once your generation configuration is completed, press the "OK" button to generate your layout. This will place pins and standard cells in your layout as well as creating a physConfig cell view. If you only see your pins in your layout and not cells, the physConfig cell view is what you will want to change. This cell view contains all the information that the Generate from Source tool uses to place cell layout information within your layout. For example, if you wanted to generate a PMOS transistor, you would select the library that contains the process you want to use for Physical Library, and then specify PMOS under Physical Cell. This also works if you've already created a layout for a cell; select your lab library under Physical Library, and pick the cell you're interested in under Physical Cell. Once you've successfully selected everything, save the file and rerun the Generate from Source tool. You should see the cells you are trying to if they did not generate previously.

Application Physication Usery residences Physication Very residences Physication Physication Physication Physication <th></th> <th></th> <th></th> <th>Cell</th>				Cell
thorpy exits 7,416 Cit Wh 3 We therease Our untransit There is the set of	Physical library list: ee330_TAlib	Physical		igical
Cit Mono Verre: schemalic Verre: underseene Verre: schemalic Verre: underseene Verre: schemalic Verre: underseene Verre: underseene Verre: underseene Verre: underseene Verre: underseene Verre: underseene Verre: underseene Verre: Underseene Verre: Underseene Verre: Verre: Underseene Verre: Underseene Verre: Verre: Verre: Underseene Verre: Underseene Verre: Verre	Logical switch view list: physConfig schematic symbol	Library: <unknown></unknown>	Allb	Library: ee330 '
Your stands Your seakcoso Open Impaid log length of the log length	Physical stop view list: Javout abstract	Cell: <unknown></unknown>		Cell: NAND
		View: <unknown></unknown>	tic	View: scheme
Contraction of the set of th	Physical stop library lisc			0
Tators Constraint Over Sealing View Tables Physical Cole Physica	Constraint view list: constraint	CANEL O		open
None Methods Methods Methods Propulation Physical data MMDD physical detronast symbol NCU [Test is under ress tayout data tayout data MMDD physical detronast symbol NCU [Test is under ress tayout data tayout data MMDD physical detronast symbol NCU [Test is under ress tayout data tayout data MMDD physical detronast symbol NCU [Test is under ress tayout data tayout data MMDD physical detronast symbol NCU [Test is under ress tayout data tayout data Ministrational detronast symbol NCU [Test is under ress tayout data tayout data P1 physical detronast symbol NCU [Test is under ress tayout data tayout data P2 physical getermatic symbol NCU [Test is under ress tayout data tayout data Reserver rest NCU [Test is under ress tayout data tayout data				stances Cells
Mode physic/afg_schematics yneld McDU TestsB_amidi mms upwai datarat Mps/Sofig_schematics yneld McDU TestsB_amidi pms upwai upwai datarat	Physical Library Physical Cell Physical View Inherited Stop List	View To Use	Inherited View List	Name
Burnerson Long	layout abstract		physConfig schematic symbol	E NAND_3
Image: State	NCSU_TechLib_ami06 mmos layout layout abstract		physConfig schematic symbol	🥥 N0
Burnelson Terminel March Street Agendance March Stre	NCSU_TechLib_ami06 nmos layout layout abstract		physConfig schematic symbol	- 🥥 N1
Processor Processor Barrier Marce Stress Barrier Interest Processor Transact Transact	NCSU_TechLib_ami06_nmos layout layout abstract		physConfig schematic symbol	- Q N2
Process and sources (mission in the source of the sou	NCSU_TechLib_amlob_pmos layout layout abstract		physiconfig schematic symbol	- 90 P0
Ruore Termane	NCSU_TechLib_amilo6_pmos layout layout abstract		physicontig schematic symbol	
Norden mannan Typenstera Terminak				
			meters Terminals	ributes neration Para

Assuming you've done everything correctly, you should see every cell and pin you need for your layout. In addition, every terminal within your layout should now be labeled as to what net it corresponds to within your layout. The Generate from Source tool will also pull in the netlist from the corresponding schematic, giving the connections that will need to be made within the layout. This tool will prove valuable when it comes time to run the LVS tool; if you already know what connections need to be made, it is less likely that you will miss one and fail LVS check. Now that you have all of your cells and pins generated, you can place and route to your heart's content.

						٧	/irtu	oso®	Layo	out S	uite	XL Ec	litin	ıg: ee	330_	.TAlii	NOR.	_3cop	y layou	t*				-		×
Launch Fil	e <u>E</u> dit	Vie	w g	reate	Verifi	/ Co	onnect	tivity	<u>O</u> ptic	ons	ools	Wind	ow	Ass <u>u</u> r	a Flo	orplan	Place	<u>R</u> oute	Help					cā	i d e n	ce
		9	¢	1	• 0	k		2	R (B Q	٩	»	л <mark>у</mark>	abc	-	» Q	»	Classic			- 5	•	Off		
P. 4	∦ -	+		0	4		-	4		»	(F)S	elect:0	Sel	(N):0	Sel(I):	0 Sel(D):0 X	-28.050	Y -3	4.350	dX		dY		Dist	t 23
Palette		(? 8	×																						
Layers	_		5	×																						
Valid 🗌 I	Jsed 📃	Rou	uting																							
Filter Q Filt				-									ļ													
well dra	wing																									
AV - NV	-Δς	-	NS																							
	Tachli		106	-									İ	•												
Name	recita	Mie	100	<u> </u>																						
Name		VIS	Se																							
All Laye	15	±	JEI	-																						
 Layer 	Pu	۷	S .																							
💹 pwell	drw	~	⊻																							
nwell	drw	~	<u> </u>										Ļ													
active	drw	~	_																							
nactive	dow	ž	- Čl	=																						
nselect	drw	Ĵ	Ĵ.																							
pselect	drw	7	2																							
poly	drw	~	2										0													
🗱 elec	drw	~	~																							
metal1	drw	•	≤																							
🗱 metal2	drw	•	≤																							
🗾 metal3	drw	~	~													9										
CC CC	drw	~	<u> </u>																							
via via	drw	~	_																							
alass	drw	ž	Ľ.																							
highres	drw	5	ő.																							
nodrc	drw	7	÷.																							
nolpe	drw	-	2																							
🐹 pad	drw	~																								
Objects	_	_	8	×																						
Objects	-	N	6																							
Instance		-		2									п	mos		mos	nme	22								
instance		~	2	-																						
Objects G	rias		0.0			•		M	•	•	æ						•	•	•	•	•	•	•		•	
			"du			0	11	Ħ		20	1	6	Ŀ			1										
	۵ 4	3	Ø			2		9:	9:		¥	Ţ	¢	•	.39	2	M									
mouse L: sh	owClick	Info()									M:	IxHiF	ReInitE	esign	0							R:	_IxHiMo	usePopl	Up()