Laboratory 4: Layout, DRC, and LVS
Fall 2022

Table of Contents

Background Information

In the previous lab, you created a "layout" for a p-channel transistor. The process involved laying out the geometric features on several "layers" that comprised the transistor. After doing the layout, you created an extracted view of your design and that was it – no additional actions were performed. In reality, there are many checks that need to be performed on a design before it can be fabricated; so many, in fact, that it's not worth listing here and certainly not worth checking by hand. For this reason, many checks are implemented with automated tools which go through and look for possible mistakes or rule violations in your design. In this experiment, two of these tools will be investigated.

Checkpoints

The checkpoints for this lab are as follows:

1. Inverter DRC (from Part 1)
2. Inverter LVS (from Part 2)
3. Inverter DRC and LVS (from Part 3)
4. NAND/NOR Testbench Results

These checkpoints must be shown to a lab TA before you submit your report. These checkpoints should be included in your lab report.

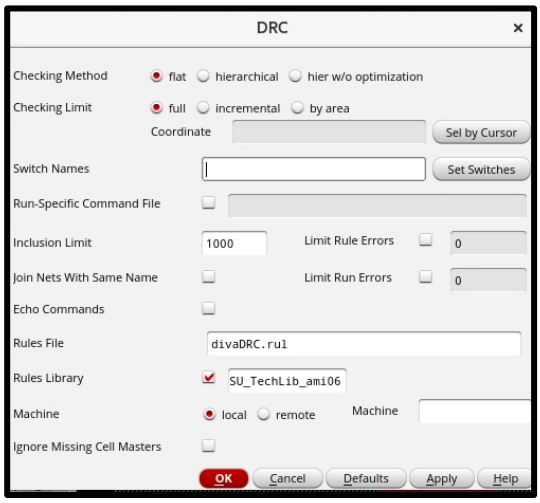
Part 1: Creating a Layout

Open the layout view of the inverter that you created in Part 3 of Lab 2. For this first layout, we **will not concern ourselves with size**, but in later layouts making the transistors the proper size will be important.

Last week we created a p-channel transistor. This week we will use that transistor and add an n-channel transistor to the design to create an inverter. Before continuing however, we need to go through some of the checks. The first such check is DRC (Design Rule Check). This tool checks your layout to make sure the different trace sizes, shapes and positioning of your layout is compatible with the manufacturing process. Normally we want to run DRC early and often in the process so that we do not have to make a lot of fixes at the end. It is sometimes hard to see where Cadence finds an error, so run the DRC check frequently so you know where you need to make changes. It must be emphasized, however, that the DRC tool checks only for design rule errors and not circuit errors.

Running DRC

To run a DRC, in the layout window go to **Verify → DRC → {OK** or **Apply}.** If there is a design rule violation, the DRC tool will identify what it is and where it is at. For example, in the process we are working with, if the smallest width of a poly strip is 0.2µm, and one was drawn at 0.1µm, then the Design Rule Check will print out an error and create a yellow symbol on the layout. This symbol will not go away until you fix the error **and run DRC again**. If the DRC does not find any errors, the circuit should not fail when fabricated because of spacing related concerns. To make sure what the output of the DRC check is, look at your CIW window.




```
DRC started.......Thu Sep  3 19:19:45 2020
    completed ....Thu Sep  3 19:19:45 2020
    CPU TIME = 00:00:00  TOTAL TIME = 00:00:00
*********   Summary of rule violations for cell "Inverter layout"   *********
   Total errors found: 0
```

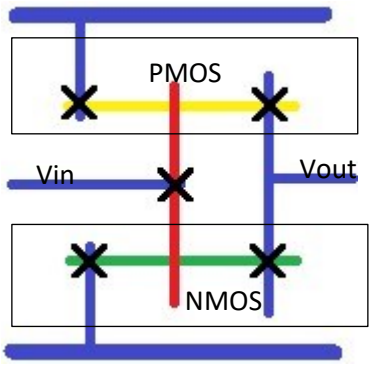
The DRC check does not guarantee that your layout is correct outside of meeting the physical requirements of your design process**. It does not check if your layout matches the performance requirements or even your schematic**. The file that contains the design rules in the ISU installation of the Cadence toolset is divaDRC.rul.

If your layout has errors, there will be a white marker on the problematic area. Correct any errors which are found. If it is not clear what the issue is, you can clarify it by going to **Verify → Markers → Explain** and click on any of the white error markers. Over time you will learn what white markers mean: for example, if you see that a rectangle has a white cross inside it; it means it is not properly shaped. If you see white markings in the area between two rectangles, it means that they are too close.

Completing the inverter: From Stick Diagram to Physical Layer

One way to look at the physical layout of systems at a rather high level is with a stick diagram representation. A diagram of an inverter is on the right with:

- Blue Lines – Metal 1
- Yellow Lines – P-diffusion, which is an N-well, P-select, and P-active
- Green Lines – N-diffusion, made of N-select and N-active
- Red Lines – Polysilicon
- Black X – Connection (Via)

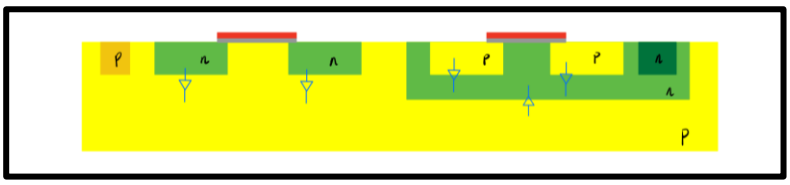


Finish the inverter by adding an n-channel transistor and the necessary connections to make your design look like the stick diagram. You already have the p-channel device so you will need to add the n-channel transistor as well as a Metal 1 line on top for Vdd and one on the bottom for Vss. You will also need to actually connect the drains and sources of the n-channel and p-channel devices with vias and Metal 1, but we will do that in the next few steps.

Bulk Connections

In lecture and design, we generally look at n-channel and p-channel devices as if they are three-terminal components, being made of only a gate, drain, and source. While these are typically the only terminals that we care about, this is not consistent with what MOSFETs actually look like, physically. In reality, MOSFETs have four terminals, with the fourth being the "bulk." In Virtuoso, the bulk connection for the nmos4 or pmos4 is the terminal located between the drain and source.

The bulk connection has several purposes in the operation of a MOSFET, including having an effect on the device's threshold voltage, but we'll ignore those purposes for this lab. What is important to realize right now is that, when you manufacture a MOSFET, you create a number of PN Junctions throughout the body of the device. Recall that a PN Junction forms a diode, as shown in the image below:



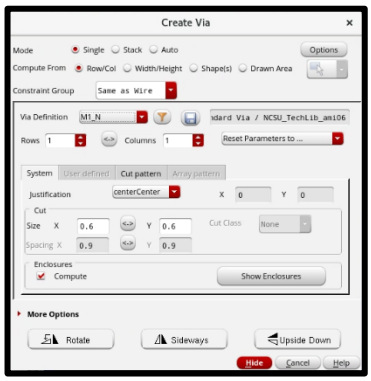
While these parasitic diodes can be useful in some cases (in fact, many cases), they can also be harmful. Consider if one of these diodes somehow becomes forward biased. Because the resistance that they see in series with them is minimal, if a single diode becomes forward biased, it can begin conducing large amounts of current – ultimately, enough current to destroy the MOS device that it is inside of. For this reason, it is critical that these parasitic diodes are kept reverse biased. This is one of the purposes of the MOSFET's bulk connection. In the image above, the extra p-doped region to the left of the n-channel device and the n-doped region to the right of the p-channel device form the bulk connections. In an n-channel device, the bulk connects to the p-substrate, and so it is desirable for this bulk to be connected to the lowest voltage present in a circuit (that is, V_{SS}). Because the n-channel bulk connects to the p-substrate, and there is only one p-substrate in an IC, it is only technically necessary to have one NMOS bulk connection in the circuit layout. In a p-channel device, the bulk connects to the n-well, and so it is often desirable for this bulk to be connected to the highest voltage present in the circuit (V_{DD}). Because the p-channel bulk connects to an n-well surrounding the p-channel device, it is necessary for every p-channel device in the layout to have its own bulk connection if there the n-wells for each p-channel device are electrically isolated from each other. It is possible to put several p-channel devices in a single n-well and in this case a single bulk connection for several p-channel devices is possible.

The **most common thing for new students to forget** when designing the layout is to create bulk connections. As stated in the previous paragraph, **each p-active region requires a bulk connection**. For the p-channel device, a via called an "NTap" is used to form the bulk connection. The n-active regions also need a bulk, so for these we will use the "M1_P" via. Unlike the NTap, most designs we will be making will only need one M1_P connection for the entire design.

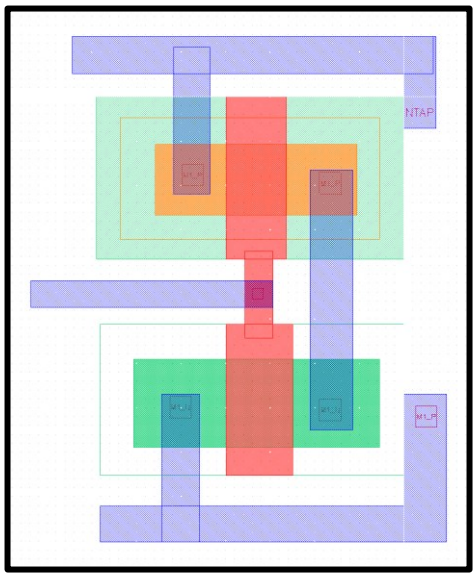
For reasons that will be discussed later, however, it is common to have more than one connection to an n-well or a p-bulk. This is necessary to prevent an undesired phenomenon that can cause the circuit to fail during normal operation that is termed “latchup”. At this point, we will not worry about latchup issues.

After setting the NMOS bulk, we now need to address the Drain and Source connections to both the NMOS and PMOS. These connections need to be made to the P-active and N-active regions. Use an M1_P and M1_N for P active and N active respectively, and then use Metal 1 rectangles to route the connection where you need it.

Now the only connection left is for the gates of the MOS devices, which are already made of POLY. For routing purposes, you can use “M1_Poly” to go from the Poly layer to Metal 1.



Your layout should now look something like this:



Pins

The final step to complete your layout will be to make inputs and outputs for your circuit. This is done so that when you use your component's layout in other designs, you will be able to run a check to make sure everything is connected to the right ports.

To create a pin, go to **Create → Pin**. Make sure the pins are consistent with the schematic: names *and* I/O types. **Pin names** are **case sensitive** and **cannot be changed** without deleting and remaking them. Any other variables of a pin, such as its layer or I/O type can be changed in its properties later. Check the **Display Terminal Name (now called Create Label)** and **Physical Only** boxes. When you know where the pin goes, make sure to have the same layer selected in the Layers toolbox.

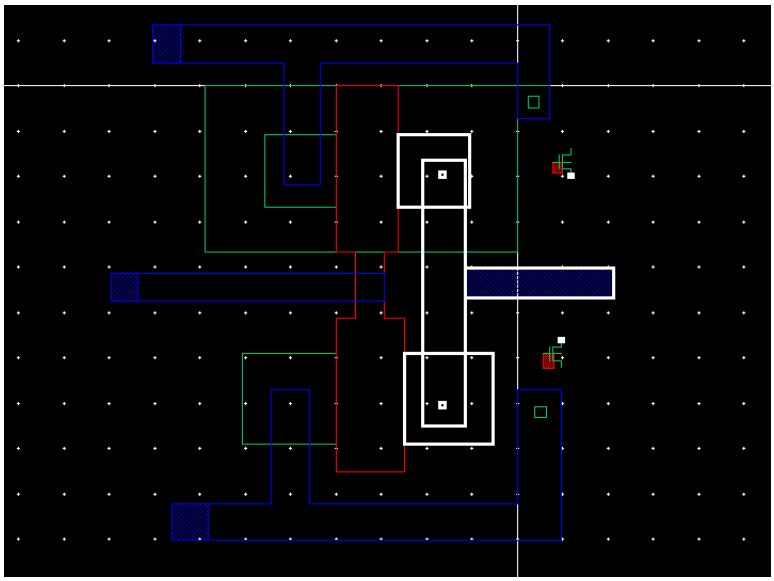
Create the pins for your Input, Output, and V_{SS} and V_{DD}. Once complete, run a DRC on your circuit. The CIW output showing a DRC with zero errors is your first checkpoint for this lab.

Part 2: Extracted View

Once you believe the design is complete run DRC one last time and extract the layout. This means that the computer will generate a schematic from the layout. To create this schematic, go to **Verify → Extract → OK**. The extraction rules are in the *divaEXT.rul* file.

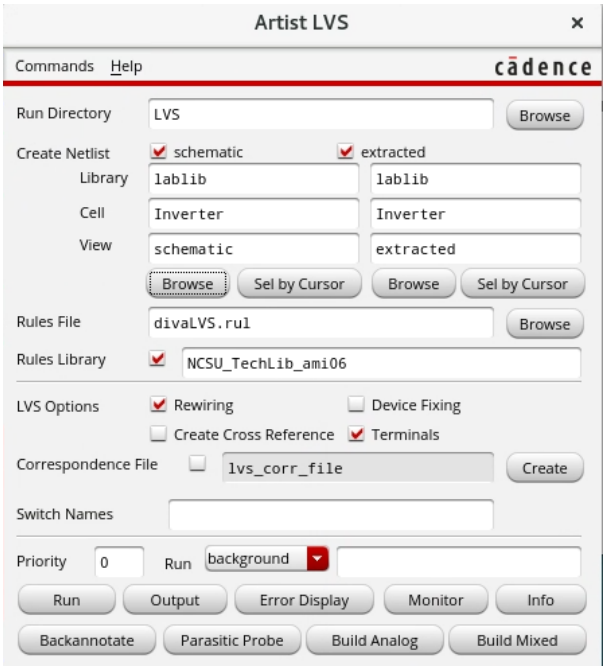
With the layout now extracted open the extracted view. The different outlined rectangles show where nets are, while a solid color shows where a pin is. The color tells you what they are made of, the blue is Metal 1, the red is Poly, etc. Later we will use this to see what the computer interprets as resistors, capacitors, diodes, etc. Move the pmos4 box and the nmos4 box so they are not over the design and press **Shift+F**.

Shift+F is used to show more detail in these boxes, they should now look like Transistors with a width and length. By clicking on any box it will change the outline to be white and show what parts are connected. It also shows what point on each component that connection correlates to. As can be seen on in the figure the metal on the right is connected to the Drain of both the NMOS and PMOS, as we want for an inverter (Image in Black to show the highlight in white).



Layout vs. Schematic (LVS)

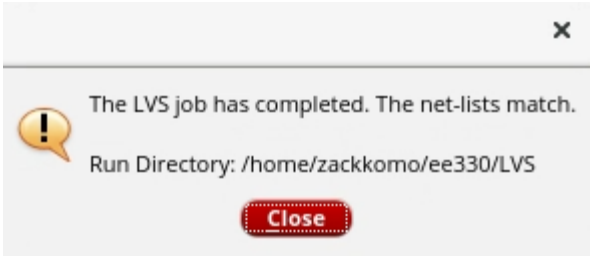
When you are satisfied the design is correct it is time to run another check on it. This time, we will run the LVS tool on your layout to verify that the circuit you have laid out agrees with your original schematic. Diva is the name of the LVS tool we will be working with in the Cadence toolset. To run LVS in the Cadence environment, go to **Verify → LVS**. Make sure you are comparing the right schematic and extracted views by hitting **Browse**. Hit **Run** when you are ready to run the LVS.



If your extracted view matches your schematic, you will get a pop-up confirming your net-lists matches and the CIW window will output:

```
Run Directory: /home/zackkomo/ee330/LVS
LVS job is now started...
The LVS job has completed. The net-lists match.

Run Directory: /home/zackkomo/ee330/LVS
```



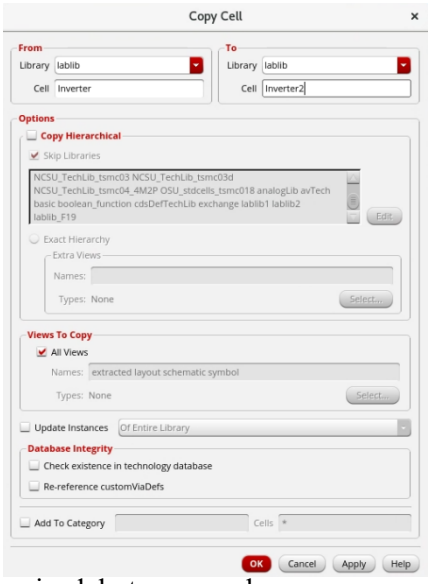
In the case where your extracted view fails LVS, you will need to go back to the layout, make changes, run DRC and extract again before re-running LVS. If you do not save your schematic or layout before running the LVS you will get a fail message, so make sure you always save your work. To get a more detailed view of what LVS found, you should choose the “Output” button, next to “Run”. Go over the file “si.out” that opens and study its different sections. This is the only text you will have for debugging. If your layout is acceptable and if you scroll down a bit you should see “The net-lists match”. If not, then you will see what connections LVS found in your schematic but not in your layout. The “the net-lists match” notification for your inverter is the second checkpoint for this lab.

How to locate a net on the extracted view:

Sometimes it is very hard to find a net in your extracted view. Probing the design makes finding nets a much easier job. Probing can be initiated by the command **Verify -> Probe**. In the Probing form, click on **Add Net**. Then go to the CIW, and type "X" (X is the name of the net that you wish to locate) at the command prompt, including the double quotes, and press Enter. Back to the extracted cell-view window, you should see a mask layer being highlighted, which has the given net name. You can also leftclick on a net and look at the CIW to know its name. The latter method is faster and more suitable for smaller designs, while the first one is better for larger designs.

Part 3: Using P cells

Now that you are used to using the Layout tools, we are going to remake the inverter more efficiently. Instead of creating the PMOS and NMOS cells by hand, we are going to use Instances of standard cells that automatically create minimum sized P active and N active regions for use. This is much faster since they are pre-made and you can resize them to the ratio you need.



Start by going to the library manager and select the Inverter schematic. Right click on it and go to Copy, under "To" name the Cell "Inverter2". Create a layout for this new Inverter2 schematic **(File → New → Cell View → Layout)**. It will prompt you to overwrite the old one.

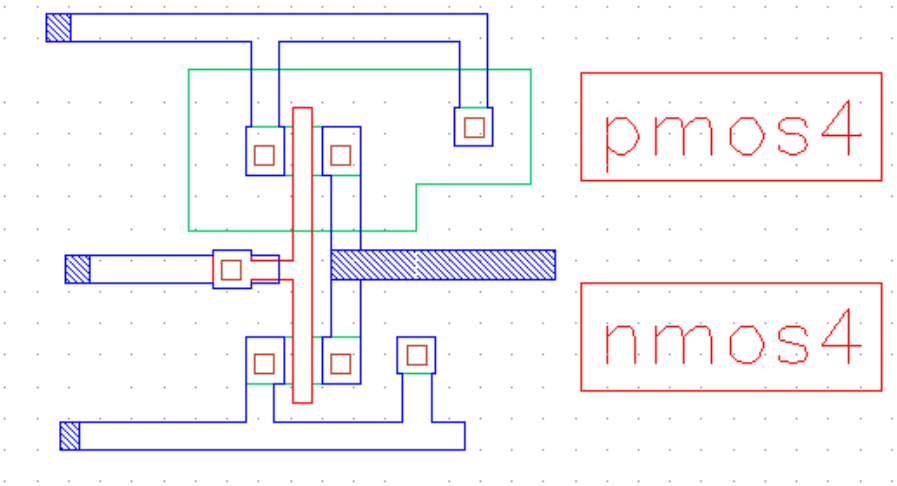
In the layout create a new **Instance (shortcut "I")** we want to automatically generate a PMOS layout, so select the NCSU_TechLib_ami06 library and the pmos Cell, the view should default to layout. This will appear as a red box that says PMOS, place this in the layout and press **Shift+F.**

You will now see something similar to the original PMOS you made, but likely much smaller. The instance is automatically minimally sized, but we can change many things by clicking on it and going to Properties **(shortcut "Q")**.

There are many important things that can be changed here, including the Width, Length, **Fingers**, and **Multiplicity**. For an Inverter we want these at their default values, but in the Parameter tab change the Multiplier to 3 to see the design change to include 3 gates and 4 areas designed to make connections with Metal 1. **This will be used to create NAND and NOR gates later**, for now return to Multiplicity 1 to continue making the inverter.

Next create another instance this time of NMOS and connect the two gates with a Path **(Shortcut "P")** of Poly, connect one side of each transistor with Metal 1, and add extensions out to be used to connect to Pins and Bulk Connections. There are many ways to design this to take up as little area as possible, but remember to always make it easy to connect to the inputs and outputs with different parts.

We advise you always put **Vdd on the top**, **Vss on the Bottom**, all of the **Inputs on the Left**, and all the **Outputs on the Right**. Remember to add the Bulks and Pins as well. When finished **run a DRC, extract, and run LVS**. Your correct DRC and LVS outputs are the third checkpoint for this lab.



Part 4: NAND or NOR

In the next lab, a logic circuit implementing an arbitrary Boolean function will be designed jointly by two students. The Boolean function will be realized with NAND and NOR logic gates. One student will be responsible for creating a three-input NAND gate and the other for creating a three-input NOR gate.

Find a partner and decide who will be responsible for each gate, then **create the schematic and test bench** for the gate you are responsible for. **Run the test bench** and verify your gate works as expected.

Looking Forward

Next week we will be creating a layout for your three-input NAND / NOR gate, exchanging gates, and creating the schematic, test bench, and layout for your Boolean function. It will be a long lab, so if you have time you may want to try to finish the layout for the NAND or NOR gate this week. Also, there is a **Pre-Lab** for next week, including creating a stick diagram for your gate, which will make creating the layout easier if you do it first. To create the gate layout, you will want to use pcells with **fingers** and **multiplicity,** discussed above.

Useful keyboard shortcuts in schematic view:

Action	Key
Add Instance	i
Add Pin	P
Wire	w
Undo	u
Redo	shift +u

Properties	q
Rotate	r
Copy	c
Check and Save	F8
Zoom to Fit	f
Move	m
Wire Name	L

Useful keyboard shortcuts in layout view:

Action	Key
Create rectangle	r
More detail in layout	shift + f
Less detail in layout	ctrl + f
Stretch rectangle	s
Zoom to Fit	f
create ruler	k
clear all rulers	shift + k
Undo	u
Redo	shift +u
Copy	c
Properties	q