Laboratory 3: Basic Boolean Circuits

Fall 2022

Contents

Background Information

The objective of this lab is to create a basic CMOS logic circuit in the ON 0.5μ process, to simulate the circuit, and to observe how cascading logic circuit create large propagation delays. Students will be expected to begin familiarizing themselves with Virtuoso IC simulation and layout tools. A two-transistor CMOS inverter device will be used as an example circuit.

In future labs, a significant amount of time will be spent working on circuit layouts. Some of the file creation and manipulation commands that will be required in this lab were covered in previous experiments.

Checkpoints

The checkpoints for this lab are as follows:

1. Basic Inverter Testbench Results
2. Inverter Transfer Characteristic
3. Cascaded Inverter Testbench Results
4. Rectangles W/ Measurements
5. Extracted View

As with all future labs, these checkpoints must be shown to a lab TA before the submission deadline for the report for this experiment. You should include these checkpoints in your lab report.

Part 1: Simulation of a CMOS Inverter

In this section, focus will be on the creation of an inverter and on the simulation of this inverter in Spectre (ADE L). The inverter schematic will be created in your "lablib" library.

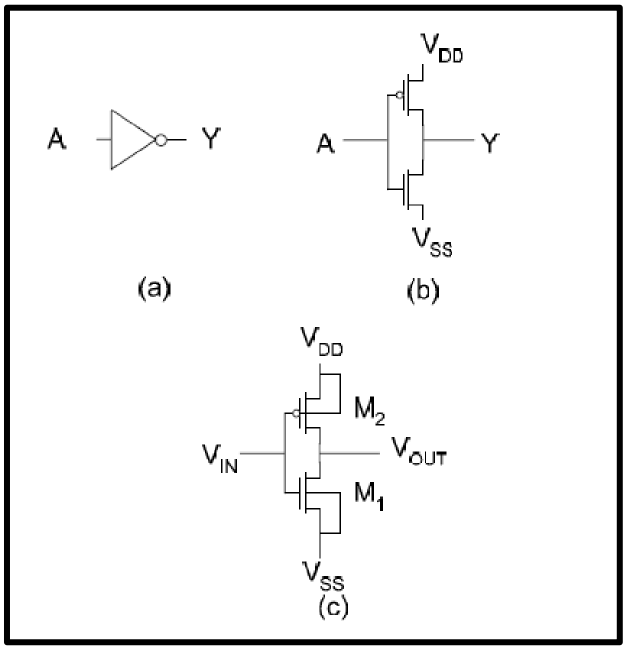


FIGURE 1 DIGITAL INVERTER. (A) GATE REPRESENTATION (B) SIMPLE TRANSISTOR IMPLEMENTATION (C) TRANSISTOR IMPLEMENTATION SHOWING BULK CONNECTIONS

Fig. 1a shows the gate level representation of an inverter. This provides no detail about the underlying circuit used to realize the gate. Fig, 1b shows the transistors utilized in the inverter to implement it but does not show how the MOSFETs' bulk terminals are connected. The representation in Fig. 1 (c) shows all connections of the CMOS inverter. In the later view, the input and output variables are labeled as voltage variables rather than as Boolean variables.

Creation of a Schematic

Create an **Inverter** cell using the circuit of Fig. 1c (do not connect it to any voltage sources!). You may instantiate the NMOS and PMOS transistors from the **NCSU_Analog_Parts** library by looking for the "nmos4" and "pmos4" cells. You are encouraged to check see if you have likely instantiated the right component by looking at how many terminals the transistors have when placed. If they have four terminals, you have likely selected the correct transistors. If the devices have three terminals, have not instantiated the right transistor.

The transistors that you place in the schematic will, by default, have a width of $1.5\mu m$ and a length of $600nm$. Change the PMOS (by opening its properties) transistor to have a width which is three times greater than that of the NMOS transistor. When doing this, modify the "Width" field, not the "Width (grid units)" field. Note that, when entering numbers in the Cadence design and simulation environment, you can use engineering notation such as k (10^3), M (10^6), m (10^{-3}), u (10^{-6}), n (10^{-9}), etc. If using engineering notation, **the suffix should be written right next to the number without spaces or unit**. For example, $1nm$ can be written as "1n" or "1e-9", but **not** as "1 n", "1nm", or "1 nm."

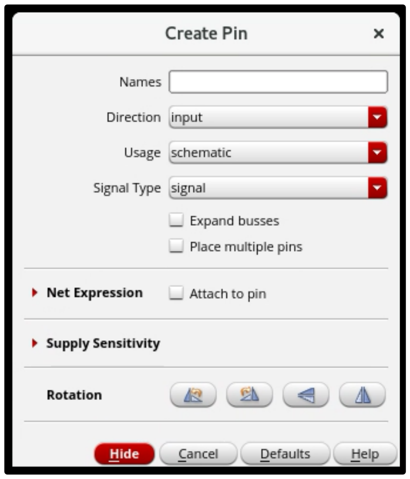
	W	L
M_1	1.5μ	$600n$
M_2	4.5μ	$600n$

Table 1: Device Sizes for Inverter

Now that you have instantiated the NMOS and PMOS transistors for your inverter, go ahead and connect the terminals as shown in Fig. 1c.

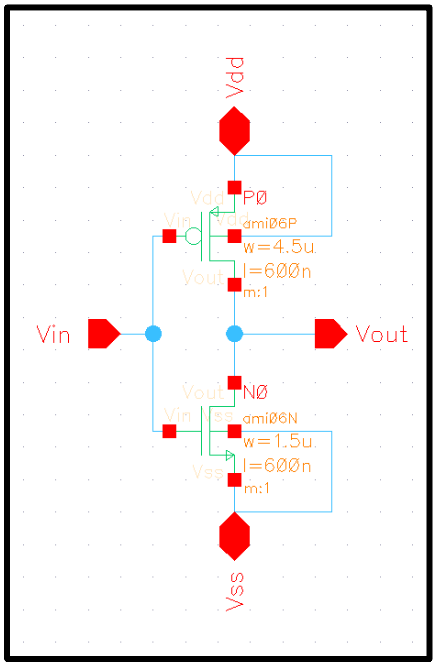
We have not yet learned enough about NMOS and PMOS devices to understand why we are making the PMOS three times wider than the NMOS, but for now, it may be entertaining to consider this question: **What moves faster, a hole or an electron?**

The schematic shown in Figure 1c includes four labeled wires: VDD, VSS, VIN, and VOUT. Our goal is to design this inverter so that we can use it in future labs and circuit designs, so we do not want to hard-set these wires to voltage sources. Instead, we will designate these wires as inputs and outputs so that other designs can interface with this inverter. To do this, we must create pins for each wire. Pins can be of different types depending upon the intended use of the pin. To add pins, go to **Create --> pin** (or hit "P" on your keyboard). This will bring up a "Create Pin" screen which will allow you to define pin names and IO types.


Create Pin

Names	
Direction	input
Usage	schematic
Signal Type	signal
	☐ Expand busses
	☐ Place multiple pins
Net Expression	☐ Attach to pin
Supply Sensitivity	
Rotation	

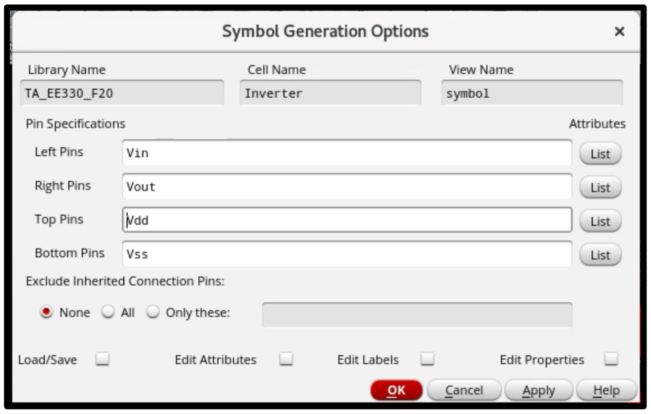
Create pins for each wire. Let the "VIN" pin be an input pin, the "VOUT" pin be an output pin, and the VDD and VSS pins be "inputOutput" pins. Note that, **if you have labeled the wires, the pin names must match the wire labels.** When you have done this, you should have a schematic which looks similar to this:



Once complete, check and save your schematic.

Symbol Creation

To use our inverter in other designs, we need to create a symbol for our design. While in the schematic editor, in the menu bar, click on Create, **Cellview → From Cellview**. Verify that the location information in the "Library Name" and "Cell Name" fields are correct and then click on OK. A symbol generation window will open. Make sure your VIN pin is in the "Left Pins" section, your VOUT pin is in the "Right Pins" section, your VDD pin is in the "Top Pins" section, and your VSS pin is in the "Bottom Pins" section, as seen below.


Symbol Generation Options

Library Name	Cell Name	View Name
TA_EE330_F20	Inverter	symbol

Pin Specifications

		Attributes
Left Pins	Vin	List
Right Pins	Vout	List
Top Pins	Vdd	List
Bottom Pins	Vss	List

Exclude Inherited Connection Pins: [✓ None] All Only these:

Load/Save ☐ Edit Attributes ☐ Edit Labels ☐ Edit Properties ☐

Figure 2: Symbol Generation Window

Once this has been done, click OK. A new window will appear with a basic rectangular symbol in it. Delete the red and green boxes, leaving only the pins, and use the editing pallet available in the toolbar to create a symbol which looks like an inverter. You may find the line and circle tools particularly useful. Note that the only items in this editor which have electrical properties are the red squares created by the symbol generator, which are the inverter pins that you will connect to later in this lab. Green lines and shapes are purely graphical and have no electrical connectivity. You can move the red pins around so that they fit nicely in your inverter symbol.

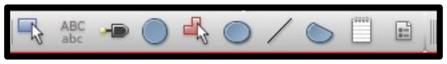
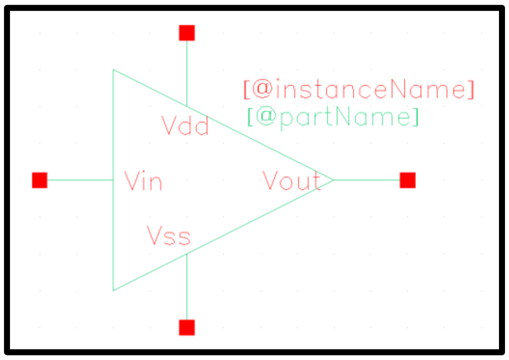


Figure 3: Symbol Editing Pallet

It may be tempting to skip this step because it's not electrically important, however, you are encouraged to not do this. In future labs, you will use this inverter heavily, and will quickly realize that it is much easier to analyze a circuit when you have proper symbols which describe the component's function. Creating descriptive symbols is an important step in the design process.

When you are complete, you may have a symbol that looks something like the image below. Don't forget to check and save before continuing.



Inverter transient response simulation

Now that we have a symbol for our inverter, let us set up the test bench. Create a new schematic cellview and call it **inverter_TB** to test the design. A recommended test bench is shown below.

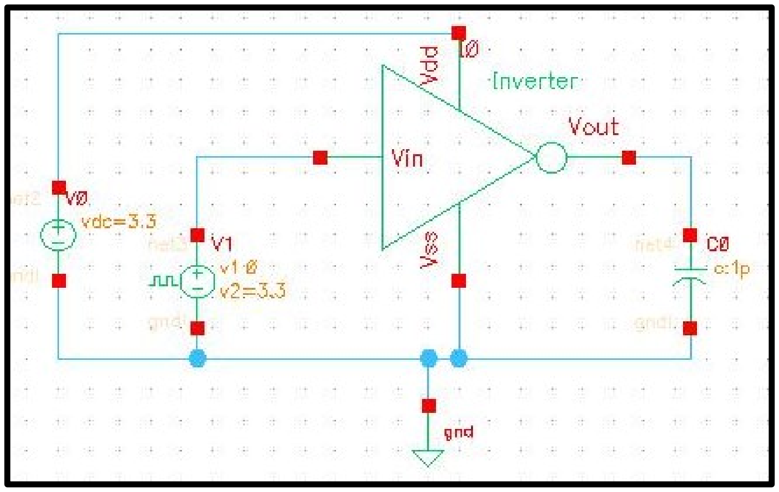


Figure 2: Inverter simulation test bench

Use the voltage sources from **analogLib**. For the pulse input source, you can use vpulse. You will need to apply an input square wave of frequency 1MHz with magnitude of 3.3V and rise and fall times of 1 picosecond each. To instantiate your own inverter, use the instantiation procedure as you would for any other component from analogLib but choose the inverter cell instead from your own library (lablib). Set the dc power supply voltage to 3.3V and the load capacitor to 1pF. **Check and Save** your work and correct all errors or warnings.

Open the **ADE Assembler** and set up the transient analysis to run long enough for 5 complete input cycles. Select the input and output voltages for plotting and start the simulation. Did you create labels for those nodes of interest (recall how we used the labels vIn, vMid, and vOut in lab1)? Debugging and evaluation of results become easier with labels.

Inverter transfer characteristics simulation

Modify the source to have a variable DC value (vdc) and run a DC sweep to obtain the dc transfer characteristics (V_{OUT} vs V_{IN}) of the inverter for inputs voltages between 0V and 3.3V. *At what value of V_{IN} is the input and output equal? What is the minimum value of VIN that can be applied and still keep the output near Vdd? What is the maximum value of V_{IN} that can be applied and still keep the output near 0V?*

Inverter driving a load

Return to transient analysis and increase the load to 10pF. What do you notice at the output? **What if you have a 100pF load? Or a 1uF load? Explain what you think happens.** You can set the load capacitance as a variable and sweep it with given values (10p 100p 1u).

Revert back to a 10pF load and increase the width of the transistors of the inverter by a factor of ten, then simulate (check and save before simulating). Revert the widths and increase the lengths of the transistors by a factor of ten, run the new simulation. ***What happened, and why did it occur?***

Return your inverter back to its original state when finished.

Part 2: Cascaded Inverters

We will now cascade two inverters and analyze how the signal propagates from one stage to the other. Create an **inverter_cascade_TB** cell so it looks like the following figure:

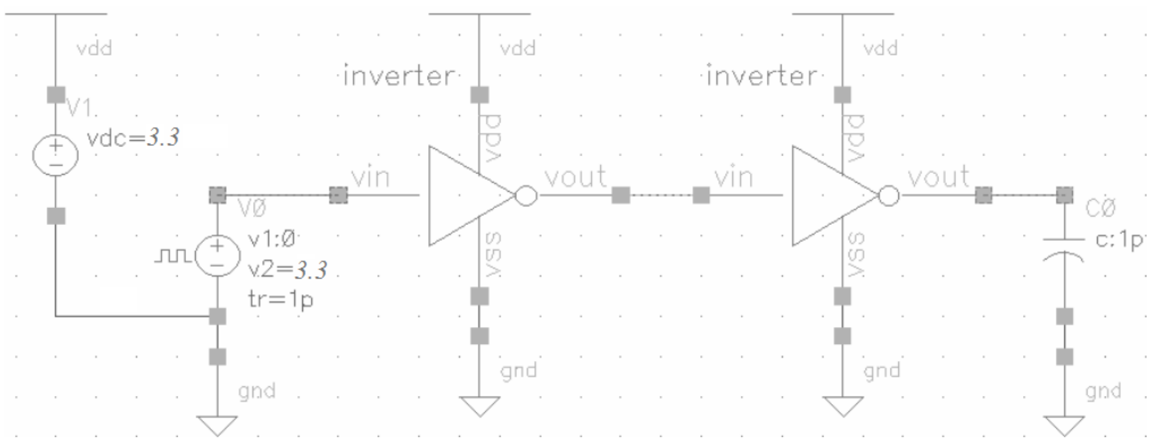


Figure 3: Cascaded inverter simulation

The "antenna" on top of the vdc source and the power supply pin of the inverters is a visual aid and a method of connecting multiple power supply nets together. The antenna shaped cell is called "vdd" and is available from analogLib.

Run a simulation and plot the three voltages of the circuit: input of inverter 1, output of inverter 1, and output of inverter 2. Label these nodes before running the simulation.

Compare the output of inverter 2 with the input of inverter 1. Do they look the same? Do you see any loss of signal integrity? Is there any noticeable delay?

Part 3: Introduction to Layout

You will soon learn in lecture that integrated circuits are created layer-by-layer by repeating the same basic processing steps over and over again, slowly adding and removing from areas of the circuit on each layer. This allows you to reliably create a single three-dimensional object which is composed of regions of different materials and different doping concentrations. For example, the figure below shows a simplified image of what an NMOS looks like.

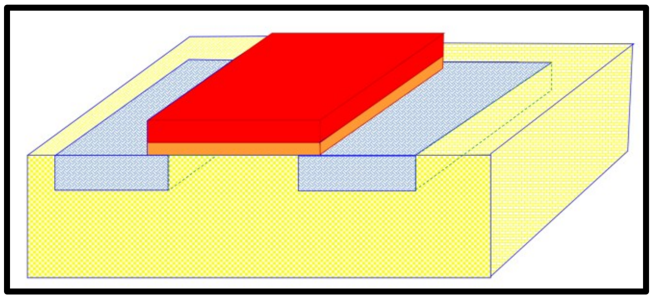


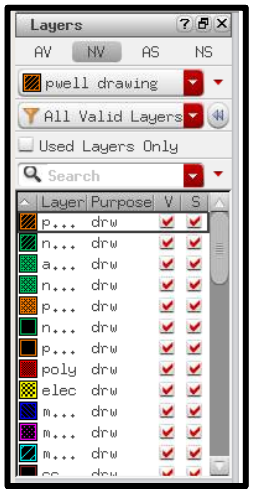
Figure 4: NMOS Cross Section

Just like how the fabrication process builds ICs layer-by-layer, designers' layout their circuits layer-by-layer, but not by looking at a cross-sectional view. Instead, designers view the IC from a 2D top view. The reason for this is simple: because designers cannot change the vertical dimensions of the layers added, or which layers are on top of others, there's no reason for them to see the excess information presented by the third dimension. If you don't fully follow this, that's fine; it'll make more sense once you've covered IC fabrication and once you've had some practice.

In this section of the lab, you will begin becoming comfortable with this way of looking at ICs. You will do so by creating your very own PMOS device, featuring its own gate, drain, source, and bulk connections.

Creating a Layout

To begin, create a "layout" view for the inverter cell you created earlier. This can be done in the schematic view by selecting **Launch → Layout XL** or from the Library Manager through **File → New → Cell View → Layout** in the Inverter folder.



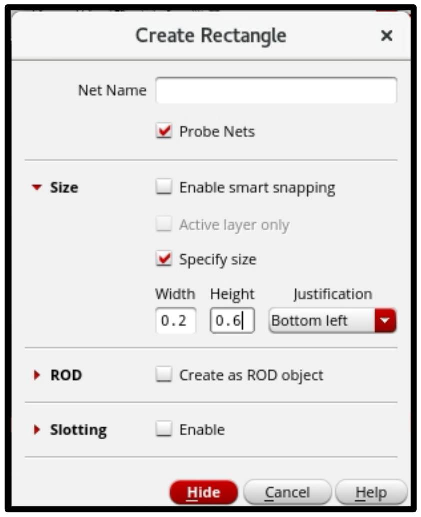
Layer	Purpose	V	S
p...	drw	✓	✓
n...	drw	✓	✓
a...	drw	✓	✓
n...	drw	✓	✓
p...	drw	✓	✓
n...	drw	✓	✓
p...	drw	✓	✓
poly	drw	✓	✓
elec	drw	✓	✓
m...	drw	✓	✓
m...	drw	✓	✓
m...	drw	✓	✓
cc	drw	✓	✓

The **first thing to check** when you open a new layout for the first time is to make sure that the technology file is attached correctly. To do this, check the available layers (located on the left of the screen) and ensure that each layer is colored differently (see image on the left). If the boxes are not different colors, first close Virtuoso and double check that you ran the "virtuoso" command from inside the "ee330" folder. If you did, and there is still a problem, you should bring it up to your TA. You can select different layers by clicking them in this select bar. These layers represent the layers on a chip. The two basic ones we will be using today are Poly (red), which is what the gate of MOS devices are comprised of, and Metal 1 (blue), which is the metal layer closest to the substrate and what most of your traces will be made of. In addition to these two, we will be using vias, which are pathways between each layer which allow for layers to interconnect. In this lab, we'll be using vias to connect our Metal 1 layers to Poly.

The first thing to do in the layout view is get used to how it works. To do so, lets start by drawing a rectangle. Select **Create → Shape → Rectangle** (**shortcut key 'r'**). This will bring up a small box with options, which we will ignore for now.

Go to the middle window (the layout canvas) and **left click once**. Then **move the mouse**. **Click again** and the program will make a rectangle. Note that **Click and Drag will not create a rectangle,** you must click on each of the rectangle's opposing corners.

Next let's create a rectangle which is exactly 0.6 microns by 0.2 microns. To do this, press 'r', but this time **do not ignore** the pop-up box. Click on the "Size" drop down and check the "Specify size" box. Put in 0.6 and 0.2 into the Width and Height fields, as shown in the image to the right. In the layout canvas, you should now be able to place rectangles. Create three rectangles which are 0.6μ by 0.2μ, one of type Metal 1, one of type Metal 2, and one of type Poly.


Create Rectangle

Net Name

[✓] Probe Nets

Size
[ ] Enable smart snapping
[ ] Active layer only
[✓] Specify size

Width	Height	Justification
0.2	0.6	Bottom left

ROD
[ ] Create as ROD object

Slotting
[ ] Enable

You can use the ruler to check your dimensions (shortcut key "k"). If the screen becomes cluttered with rulers, go to **Tool → Clear All Rulers (shortcut key "Shift K")**. Make sure you experiment with the buttons to the left as well as the menu to get comfortable moving, copying, stretching, cropping or rotating a rectangle; the options are endless.

Show the three 1.2x0.6 rectangles with rulers next to them to your TA in the form of a checkpoint.

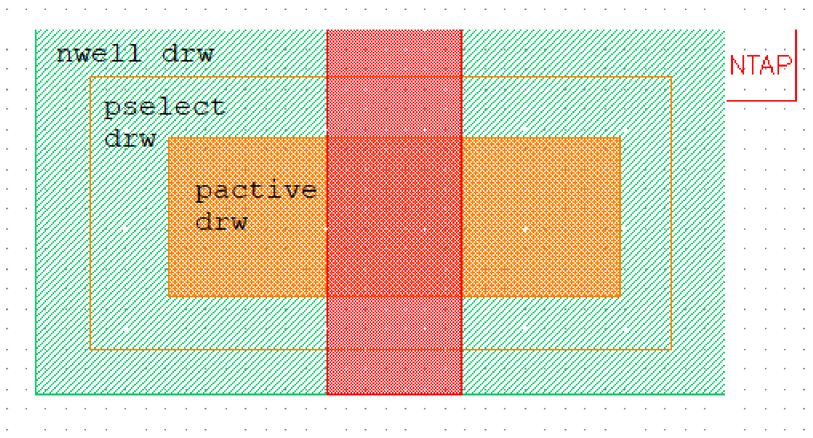
Creating a PMOS transistor

After getting a picture for your checkpoint of the part above, go ahead and delete the boxes that you made. We will now go through the process of creating a single transistor layout. A transistors physical layout is made of many parts:

- Well – a lightly doped region around the transistor that is doped the opposite of the active region. NMOS transistors need a P-well and PMOS transistors need an N-well. Due to the settings in Cadence we assume a P-well substrate, so we do not need to place a P-well layer for NMOS devices.
- Active – The Active region is the region that is heavily doped and is the portion of the main body of the transistor.
- Select –This is a region surrounding the Active region of a transistor, inside the well. It is required for manufacturing reasons.

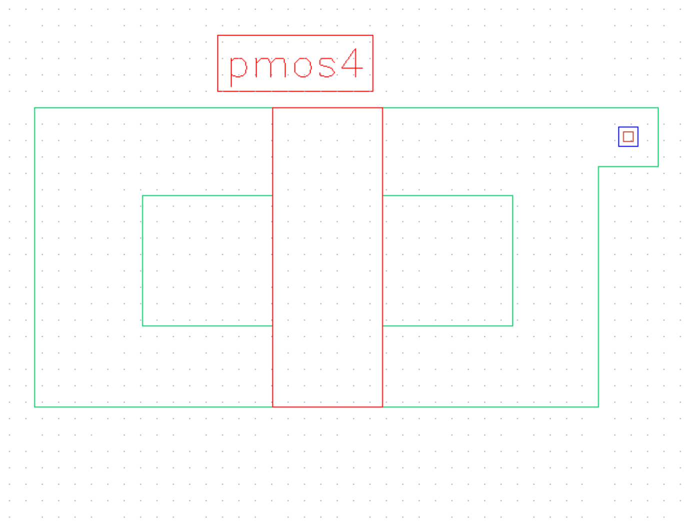
- Gate – This is a polysilicon layer that passes across the transistor touching both ends of the select. It determines what state the transistor is operating in.
- Source/Drain – Physically identical in our process the source and drain are the two regions the active area is divided into by the gate.
- Vias – Multiple vias are needed to electrically connect parts of the transistor. The vias we will use will be,
    - M1_P – used to connect metal to the source and drain for PMOS devices, as well as the bulk connection for NMOS devices. To attach the bulk of an NMOS you just have to place an M1_P connection from the metal that will connect to Vss to the general bulk.
    - M1_N – used to connect metal to the source and drain for NMOS devices.
    - NTAP – used to connect the bulk of PMOS devices. This is made up of an N-well and a via, attach it to the metal connected to the source side while keeping it outside the select region.

With the above information you should attempt to create a PMOS transistor. You should refer to the image shown below for help. Essentially, we are layering different rectangles at different layers and different materials to get the desired functionality. Follow the image and the extracted view should look the same (to create the NTAP go to create → via →NTAP from the dropdown):



In the next lab we will see how to add "pins" to this layout so that we can actually chain components together. For now, you should be able to see that the center red rectangle made out of Poly is the Gate of your PMOS. The pactive layer makes up your channel and so it is a whole rectangle going under your Poly rectangle. One side will act as the Drain and the other will act like the Source. The NTAP is the bulk of your PMOS and it connects to the N-Well.

Once you have completed it go to the top tool bar and select **verify → extract**. This will create a new cell view called "extracted". Open it and there should be a box that says pmos4 if you have built the device correctly. Press **Shift+F** to be able to see more information (Width, Length, etc), and **Control+F** to return to the box that says pmos4.



Tip: Familiarize yourself with the user interface before proceeding to the next lab. This will save a lot of time next week. Make sure to check the keyboard shortcuts on the next page.

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