Lab 1 Part 1: Schematic Design and Simulation

Report Due Date: February 17, 2020 at 11:59 pm

Real-Time Evaluation Date: February 14, 2020

- You will create a schematic and symbol for a CMOS inverter in Cadence Virtuoso.
- You will perform a transient simulation and DC analysis for your inverter.
- **Note: Please read *all* instructions carefully! If you miss one step, then nothing may work and it will be nearly impossible to figure out what is going on.**

---

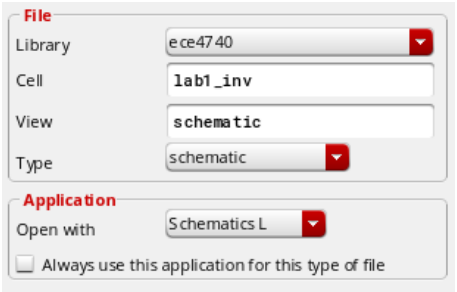
1 Creating an Inverter Schematic

The technology we will use in ECE 4740 is the Generic Process Design Kit 90nm ("GPDK090") CMOS process. This process is called a 90nm technology (gate pitch), but the minimum transistor length is 100nm (minor gate length expansion, which is common in more recent technologies). In this part, you will create a schematic for a CMOS inverter.

Note: Anytime Cadence asks you about checking out a license, select "Always".

1. We will start creating the schematic for our CMOS inverter. Start by selecting "File"→"New"→"Cellview" from the Cadence Command Interpreter Window (CIW).

2. A window will show up. Set the following values:
   Library Name: `ece4740`
   Cell Name: `lab1_inv`
   View: `schematic`
   Type: `schematic`
   Open with: `Schematics L`
   Click "OK".

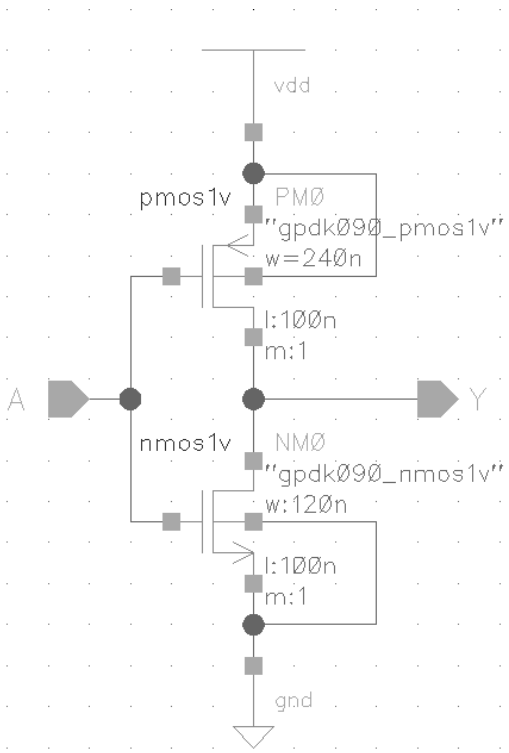


3. A new window with the title "Virtuoso Schematic Editor L" should appear. Now, it is time to create our own circuit. To do so, create a new instance by right-clicking on the Design Canvas (the black region of the window). Then select the "Add Instance..." option. Alternatively, you can simply press "**i**" on your keyboard, which stands for "instance."

4. A new window will pop-up, asking for a few values. Notice the first two fields labeled "Library" and "Cell". Library indicates the technology library that contains the component of which you wish to make an instance. Cell indicates the name of the component that you want to instantiate. Note that to add an instance of a cell to a schematic, there needs to be a symbol of that specific cell. Let us add an `nmos` transistor from the `gpdk090` technology, by clicking the "Browse" button next to "Library". In the resulting window select the following:
   Library: `gpdk090`
   Cell: `nmos1v`
   View: `symbol`
   Click "Close".

   You will notice that there is now a transistor symbol attached to your mouse cursor when you hover over the Design Canvas in the Schematic Editor window. If you left-click once, then you will place a single instance of this transistor on your schematic. Press "**Esc**" on your keyboard to stop placing instances of this cell.
   To choose the cell in the "Add Instance" window, you could have also just filled in the appropriate fields without having to browse for the cell. You can adjust the parameters (such as gate width and length) of your `nmos` device by changing them in the "Add Instance" window under the corresponding parameter fields. In this case, keep the default values of 100nm for the length and 120nm for the total width. If you later need to modify an instance's parameters, then click on the instance you wish to modify and press "**q**" standing for "qualities."

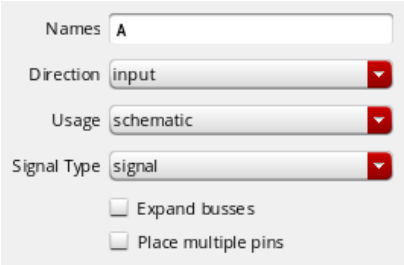
5. To make an inverter, we also need to add the components `pmos`, `vdd`, and `gnd` as shown in the figure to the right. Use the same method as before to add these components. The `pmos` transistors can be found in the `gpdk090` library as `pmos1V`. For this device, keep the default value of 100nm for the length, but change the total width to 240nm, so the inverter has a $W_p : W_n = 2 : 1$ ratio for more balanced rise and fall times at the output. The components `vdd` and `gnd` are in the `analogLib` library. We will add the pins and wires in the next steps.



   The names `vdd` and `gnd` are most often used as the power rails in CMOS. However, legacy names are summarized in the following table:

	BJT, TTL	FET, CMOS
Positive supply voltage or nominally highest voltage	V_{cc}, V_+, V_{s+}	V_{dd}, V_+
Negative supply voltage or nominally lowest voltage	V_{cc}, V_-, V_{s-}	V_{ss}, GND, V_-

6. Create input and output pins with right-click on the Design Canvas→"Add Pin...", or by simply pressing "**p**". Now create an input pin by filling in the following:
   Names: `A`
   Direction: `input`
   Then click back on the Design Canvas of the Schematic Editor window. You should see an arrow attached to your mouse cursor, much like you saw when adding an `nmos` transistor. Place this pin accordingly.



7. Use the same method to add an output pin. Make sure to set the "Direction" field to "output". We do not need to create pins for `vdd` and `gnd` because they are globally defined (hence, the net names `vdd!` and `gnd!`).

8. Next, we need to wire all components as shown in the figure. Press "**w**" to start drawing a wire. Click once on the starting point and draw the wire, then click again on the stopping point. You can also use "**l**" to create labels to connect wires via naming; a must for cleaning up otherwise messy schematics. Keep in mind that you can undo or redo your last action by pressing "**u**" and "**Shift+u**" on your keyboard, respectively.

9. The transistors in the `gpdk090` library are four terminal devices (Source, Drain, Gate, and Bulk). For this lab, we will tie the bulk connection of `nmos` to ground and the bulk of `pmos` to `vdd`. This approach minimizes leakage to the substrate by avoiding forward biases at the Source/Drain (S/D) contacts.

10. Click the "Check and Save" button (whose icon is shown to the right) on the top left area of your toolbar. You should click this often to make sure you do not lose your work.

Student Activity 1: Checking your schematic

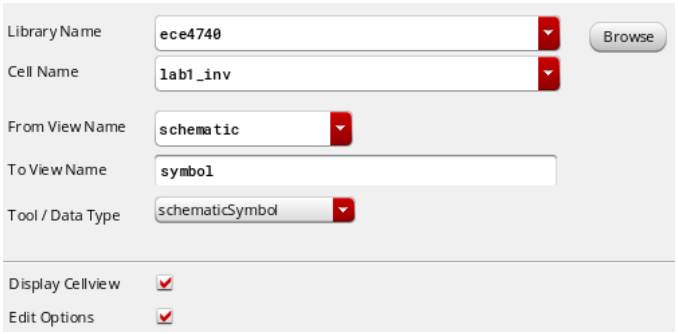
After clicking the "Check and Save" button, look into the CIW. Draw here your favorite animal if you can see a message saying "INFO (SCH-1426): Schematic check completed with no errors":

If you cannot see this message, then you must have seen a dialog box telling you about warnings/errors related to your design. In that case, check the CIW, as well as the blinking yellow boxes on the Design Canvas, to find out where the warnings/errors are. Fix all the warnings/errors, and check and save your design until you see the above mentioned message in the CIW.

2 Symbol Generation

Now that you have created a CMOS inverter schematic, let us create a symbol for it so that it can be instantiated within other schematics, just like the `nmos` or `pmos` symbols in the `gpdk090` library.

1. Select "Create"→"Cellview"→"From Cellview". Now set the following parameters:
   Library Name: `ece4740`
   Cell Name: `lab1_inv`
   From View Name: `schematic`
   To View Name: `symbol`
   Click "OK".

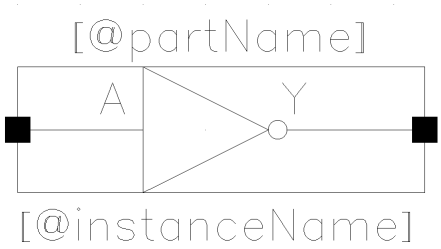


Student Activity 2: Verifying the pins

A new window, titled "Symbol Generation Options" will show up. Look at the "Pin specifications" section. Do you recognize the names of the pins? How many pins are there? _____

If you do not have exactly two pins, then you made a mistake in Steps 6 and 7 of Section 1, "Creating an Inverter Schematic". Should that be the case, click "Cancel", go back and carefully repeat the given procedure until solving this error. If you have exactly two pins, then click "OK".

2. Use the drawing tools to create a symbol for the inverter cell. You *could* just use the automatically created box symbol, but that is just plain embarrassing for an inverter (and you will lose points). So let us create a new symbol. Start by deleting the green rectangle. Then, draw a triangle with the line tool, add a circle, and move the pins into the right place, as shown to the right. Move the selection box (the red rectangle) over your inverter when you are done. Make sure to check and save your symbol.



Student Activity 3: Sanity-checking your inverter

From the CIW, open the Library Manager ("Tools"→"Library Manager"). Select the library `ece4740` and the cell `lab1_inv`. Write here the two views for this cell:
1) _____ 2) _____

Click on each view. You should see a thumbnail preview of the selected view on the lower right corner of the Library Manager window. If not, then go back to Section 1, "Creating an Inverter Schematic," to fix the schematic view, or to the beginning of this section, "Symbol Generation," to fix the symbol view.

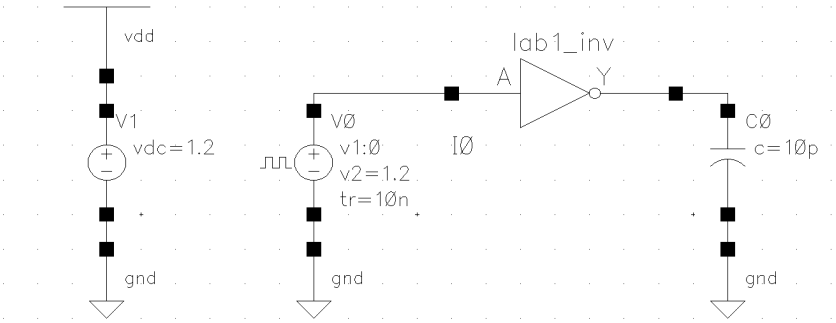
***Remark**: In Cadence Virtuoso, a circuit module receives the name of "Cell". The same cell can be represented using different abstraction layers, which are called "Views".*

3 Transient Simulation

1. Now let us execute a transient simulation. To do this, we will first create a schematic in which to run the simulation. Create a new schematic named `lab1_sim` as shown in the following figure. The inverter you made before is located in the `ece4740` library. All the other components that you need are in the `analogLib` library: Use `vdc` with a DC voltage of 1.2V, and a `cap` with a capacitance of 10pF. For the input square wave, use the `vpulse` cell with the following parameters:

   Voltage 1: `0 V` Period: `2u s` Rise time: `10n s` Pulse width: `1u s`
   Voltage 2: `1.2 V` Delay time: `1u s` Fall time: `10n s`

   Make sure to check and save your schematic, and fix the warnings/errors if any should arise.



2. Now that we have the simulation schematic, let us setup the simulation. Launch the Analog Design Environment (ADE) with "Launch"→"ADE L".

3. In the new window, select "Setup"→"Simulator/Directory/Host..." and set the following fields:
   Simulator: `spectre`
   Project Directory: `~/Cadence/simulation`
   Host Mode: `local`
   Click "OK". This will set Cadence to use the Spectre simulator and will create a folder within your Cadence directory for storing simulation files. For this simple simulation, not many simulation files will be generated, so it may not yet be a problem, but when you simulate much larger projects, this may fill up your disk quota. To solve this, you can redirect your project directory to the temporal directory on the server at `/tmp/YourNetID/simulation`. Since this is your temporal directory, it will be cleared once you log off, but you should not need to access your simulation files anyways. If you need the results again, then just re-run the simulation.

4. Select "Setup"→"Temperature" and set it to 25 degrees Celsius.

Student Activity 4: Setting up the simulation environment

Select "Setup"→"Environment". In the new window, set the specified fields as indicated in the following checklist. Check each box after filling in the corresponding field.

☐ Switch View List: `spectre cmos_sch cmos.sch av_extracted schematic veriloga`
☐ Stop View List: `spectre`
☐ Check "Automatic output log"
☐ Check "Run with 64 bit binary"

Click "OK".
***Important**: If you ever experience problems while doing simulations in Cadence Virtuoso, then use this checklist as your first troubleshooting step!*

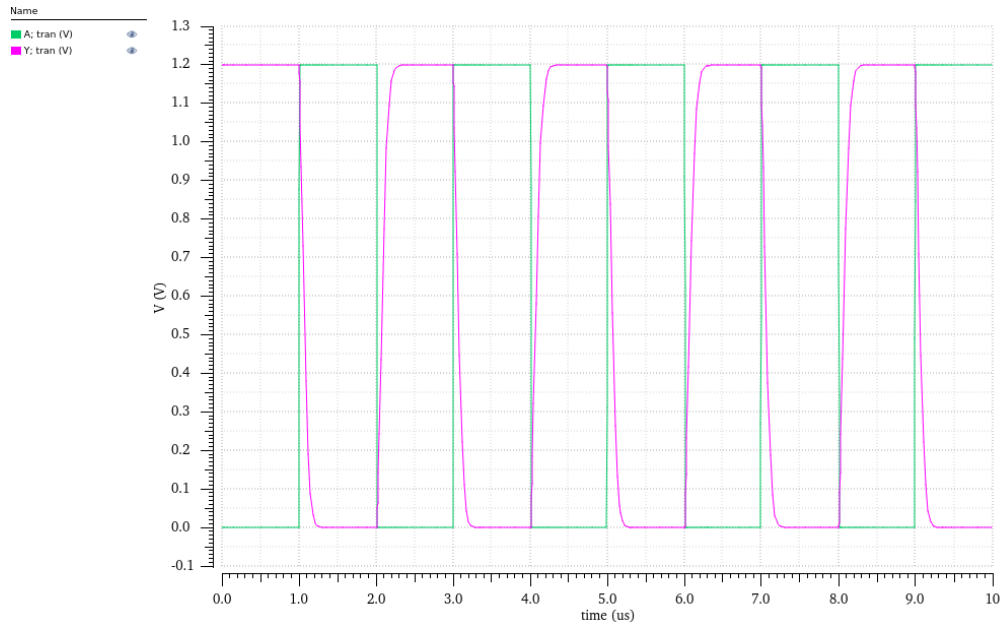
5. Save your simulation state with "Session"→"Save State". In future simulations, you will have to first load this state and then modify it as necessary (You can save to either "Directory" or "Cellview").

6. Now let us set up the transient analysis. Select "Analyses"→"Choose..." from the ADE window or click the "Choose analysis" button from the right toolbar (shown to the right).

7. Select `tran` in the "Analysis" section. Enter a stop time of 10u for 10μs. Check `moderate` under accuracy defaults. Make sure "Enabled" is checked at the bottom and click "OK".

8. Select "Simulation"→"Netlist and Run" or click the icon with the green stoplight in the right-hand-side toolbar. This will create a SPICE netlist of your design and run the selected simulations. Make sure there were no errors by looking through the CIW and verifying that everything was successful.

9. To plot your results, go to "Results"→"Direct Plot"→"Main Form". A small window will appear. In that window, select "Voltage" under "Function". Now, without closing the window, select a `net` (or a wire) on your schematic. This will plot the voltage on that node across time. Verify that the input is the square pulse that we wanted by clicking on that node. Then, use the same method to see the output voltage across time. This will overlay both plots on the same axes. Your plot should look something like the one below. The amplitude of the signal should be 1.2 V if $V_{DD} = 1.2$ V.



If you want to plot each signal on a different axis, then you can click the button showed here to the right, which you can find on the top of the plot window.

Student Activity 5: Measuring the propagation delays

You can place markers on your plots with "Marker"→"Create Marker..." (or by pressing "**a**" and then "**b**" for a difference marker) in the plot window. Use this to find the time difference between when the input signal crosses half of the supply voltage value, and when the output voltage equals half of the supply voltage, for both rising and falling output signals. These are respectively called the low-to-high and high-to-low propagation delays, commonly denoted as t_{pLH} and t_{pHL}. Write down your results:

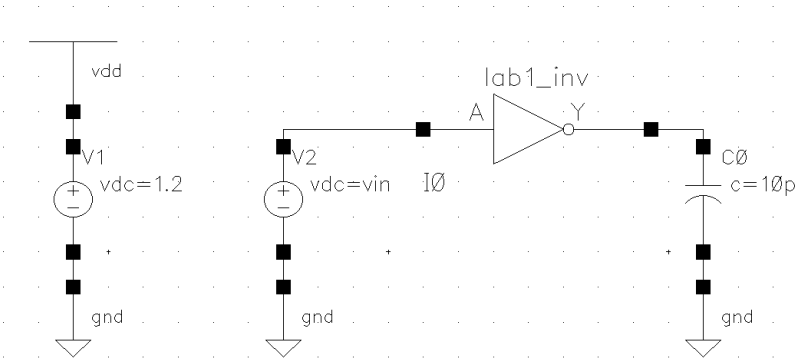
t_{pLH}: _____ t_{pHL}: _____

10. Finally, save a copy of the plot by selecting "File"→"Save Image", and then name the file as you wish. Include the file type in your filename (e.g., *.jpg, *.png). Also make sure to change the background color to white using the option located on the lower right section of the window. Alternatively, you can make a nice screenshot of your plot that clearly shows the axes and waveform. In this case, you can make the background white by going to "Edit"→"Properties", or pressing "**q**". ***Important: Do not forget to label all your plots properly (axes, units, and names of the plotted signals); this also applies to all your homework sets.***

11. Close all the ADE windows.

4 DC Analysis: VTC Curve

1. This time we will apply a DC analysis to the inverter, to obtain its Voltage Transfer Characteristic (VTC curve). Let us start by replacing the `vpulse` cell from the `lab1_sim` schematic with a `vdc` cell. Write `vin` in the DC voltage field of the `vdc` instance. Your schematic should now look like this:



   Check and save your schematic.

2. Launch the ADE ("Launch"→"ADE L"). If you saved the simulation state on step 5 of Section 3, "Transient Simulation", then you can load it using "Session"→"Load State...". If not, then follow steps 3, 4, and Student Activity 4 from Section 3 to correctly configure the simulator.

3. In the ADE window, right-click on the white area of the "Design Variables" section. Then, click the "Copy From Cellview" option. The variable `vin` that we used for the new `vdc` cell should show up. If not, then go back and review what you did in step 1 of this section.

4. In the "Design Variables" section, set a value of 0 to `vin` by double clicking on it and writing the appropriate value in the emerging window. The click "OK". *If you do not do this, then you will get an error message later.*

5. Right-click on the white area of the "Outputs" section, and select "Edit". Then click the "From Design" button, go to the Design Canvas in the Schematic Editor window, and click on the net connected to the inverter's output. Finally, click "OK". With this you have configured the voltage at the inverter's output as a simulation's output.

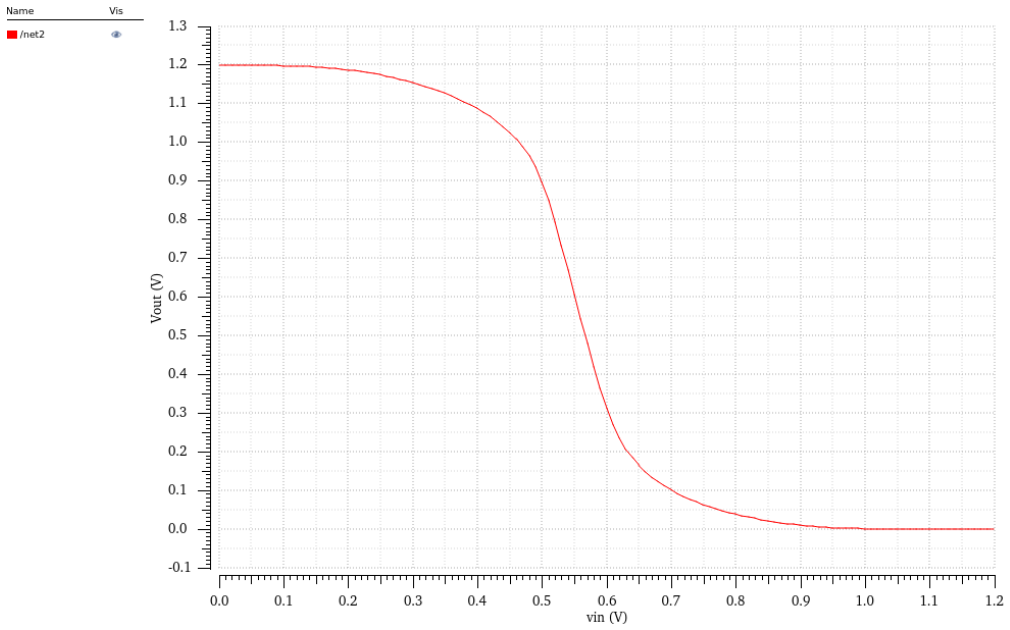
6. Select "Analyses"→"Choose..." from the ADE window or click the "Choose analysis" button.

7. In the emerging window, choose `dc` in the "Analysis" section. In the "Sweep Variables" section, check "Design Variable". Click the "Select Design Variable" button that just appeared, and then select `vin`, and click "OK".

8. In the "Sweep Range" section, write 0 for the "Start" field and 1.2 for the "Stop" field. Then, in the "Sweep Type" section, choose the "Linear" option, with a step size of 0.01. With this, we have configured the simulator to perform a DC analysis varying the DC input voltage from a value of 0V to 1.2V, following increments of 0.01V.

9. Run the simulation with "Simulation"→"Netlist and Run" or by clicking the green stoplight icon. Verify no errors occurred by looking through the CIW.

10. If the simulation was successful, then a plot like the one below will appear automatically. This plot is showing the inverter's output voltage as a function of its input voltage, i.e., the inverter's VTC curve. ***Remark**: By adding nodes to the "Outputs" section of the ADE window, the corresponding plot will be automatically generated after executing the simulation, regardless of the analysis type (transient, DC, etc.). This is a more efficient approach to generating plots than using "Results"→"Direct Plot"→"Main Form".*



11. Save a copy of the plot. Note that the x-axis is missing its corresponding units. To add them, double-click on the x-axis, untick the "Default" checkbox of the "Name" section, and modify the name field to include the units. You can do something similar to rename the y-axis if you need to.

Student Activity 6: Logic levels

Based only on the VTC curve obtained for the inverter, where would you define the logic levels for this technology? As a reminder, the logic levels are the minimum/maximum input/output voltages were a signal will be considered a logic 0 or logic 1.

V_{OH}: _____ V_{IH}: _____ V_{OL}: _____ V_{IL}: _____

With these logic levels, what would be the noise margins?

NM_H: _____ NM_L: _____

Last updated: February 2, 2020 by Thomas Tapen

Lab 1 Part 2: Layout Design, Verification, and Simulation

Report Due Date: February 17, 2020 at 11:59 pm

Real-Time Evaluation Date: February 14, 2020

- You will learn how to create a layout of the inverter from Part 1.
- You will check the design rules and verify the layout of your inverter.
- You will perform a parasitic extraction over your inverter layout and use it for simulation.
- **Note: Please read *all* instructions carefully! If you miss one step, then nothing may work and it will be nearly impossible to figure out what is going on.**

---

1 Overview: What is Layout?

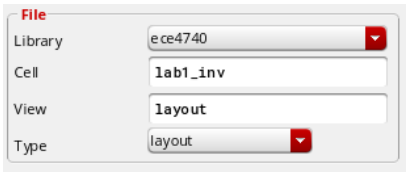
The schematic view that you created in Lab 1 Part 1 is good for conceptualizing the circuit. However, the circuit must now be put in silicon, and this is what layout is about. The layout gives you a top-down view of the various components on your chip. Since it is difficult to view more than one layer in a top-down view, different colors are used to identify the various layers. To see which color corresponds to each layer, refer to the Layers Panel (which we will explain in more detail later).

The main idea of layout is for you to place the components and then complete the necessary connections between them. The former part is referred to as "placement", while the latter part is known as "routing". Routing is done by drawing rectangles (which represent wires) in the metal layers to create the connections that are specified in the schematic. To connect different metal layers, for example to make a `Metal1` layer touch a `Metal2` layer, a so-called "via" must be generated. A via is nothing more than a plug that forms a connection between two metals on different layers. Once the layout is finished, one needs to verify (i) that it meets certain design rules, and (ii) that the layout implements the desired circuit correctly. After completing the verification successfully, one can perform a simulation using information extracted from the layout to get a better approximation of how will the fabricated circuit behave in real life.

2 The Layout Editor

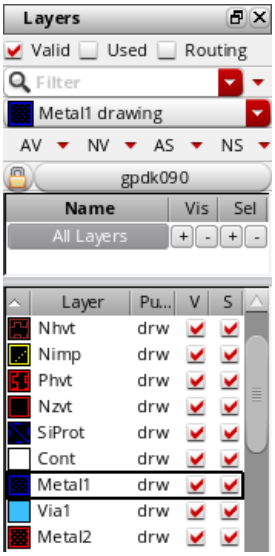
1. Open the schematic view of your inverter. In the Schematic Editor window, select "Launch"→"Layout XL". A window will show up. In the "Layout" section, select "Create New", and in the "Configuration" section, select "Automatic". Click "OK".

2. A new window will show. Make sure that `layout` is written in the "View" field. Click "OK". You should see the "Virtuoso Layout Suite XL" window open. On the very top of this Layout Editor window, the title bar should say "Virtuoso Layout Suite XL Editing: ece4740 lab1_inv layout". This means that you are editing the layout view of `lab1_inv` cell from `ece4740` library.



2.1 The Layers Panel

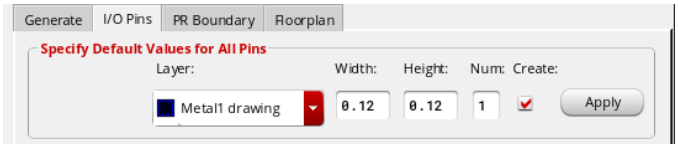
The layers panel is shown to the right. This is your command toolbar, and is very important in facilitating and completing your layout task successfully. While it looks quite complicated at first, its function is very simple. The top box, currently set to `Metal1` drawing, shows the active layer. This is the layer that gets drawn when you create a rectangle. The other buttons to note are the four buttons that say "AV", "NV", "AS", and "NS". "AV" means "All Visible", and will allow all layers to be seen. "NV" means "None Visible", which will hide all layers except the currently selected one. After clicking "NV", layers can be turned on individually by ticking their checkbox on the "V" column. "AS" and "NS" are similar and mean "All Selectable" and "None Selectable" respectively, and allows or disallows selection of particular layers, respectively. Strategically hiding and showing layers during routing will make your task much easier to finish.



2.2 Generating Layout from Schematic

1. In the Layout Editor window, select "Connectivity"→"Generate"→"All From Source...".

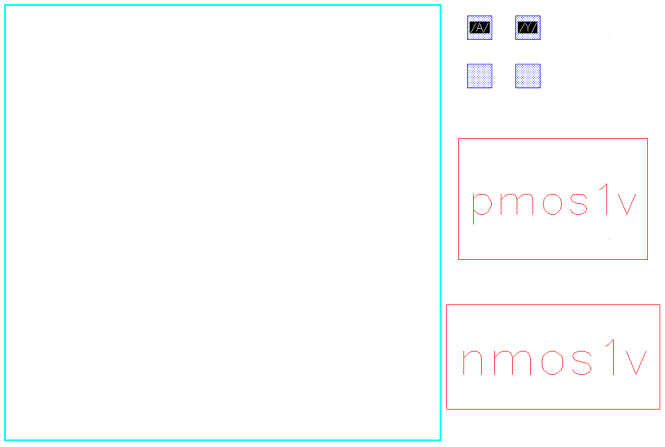
2. This will open a new window. Go to the "I/O Pins" tab. In the "Specify Default Values for All Pins" section, make sure that the "Layer" field has the `Metal1 drawing` layer selected. Click "Apply".



   By doing this, you have indicated Cadence Virtuoso to create the pins for this layout on the `Metal1` layer; note that the I/O pins are the metal regions of the layout to which external wires can connect. ***Remark**: While there are several `Metal1` layers, we will only be using the `drawing` one. This is also true for all other layers that we will use in the layout.*

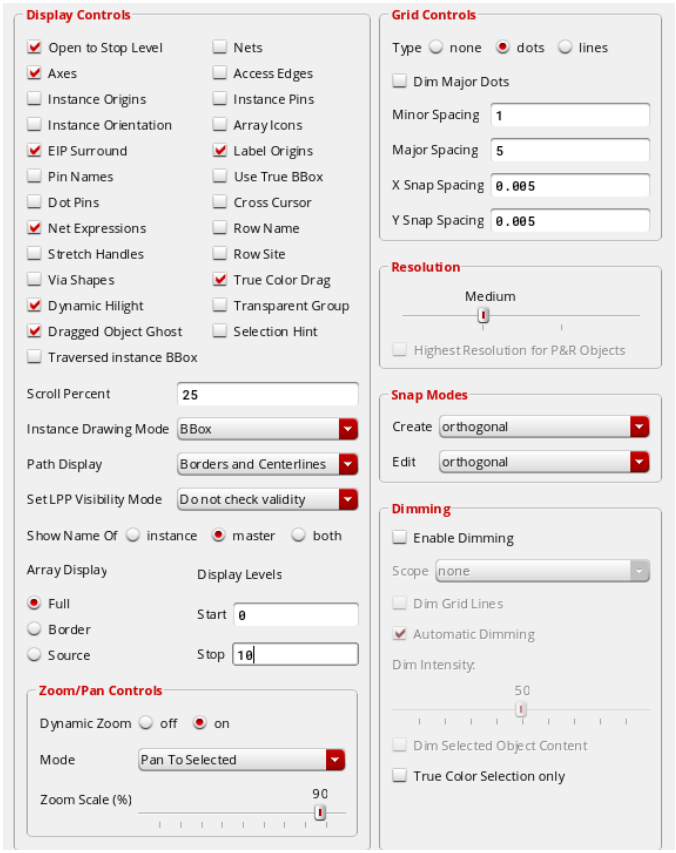
3. Leave all the other fields in this window to their default values. Click "OK".

You will now see an initial pin and transistor placement on the Layout Editor, which should look similar to the image on the right. The small blue squares are the `A` (input), `Y` (output), `vdd!`, and `gnd!` pins created in `Metal1` as previously specified. The aqua colored rectangle is the Place-and-Route (PR) bounding box, which should enclose the cell's layout and gives an estimate of the final layout's size. The PR box may need to be re-sized to accommodate all components.



Currently, in the layout you can only see outlines of the layout cells for the `nmos` and `pmos`.

4. To see all the layers, go to the Layout Editor window and select "Options"→"Display..." (or hit the "**e**" key). Then, in the "Display Levels" section, assign a value of 10 to the "Stop" field. Or you can press "**Shift+f**" and "**Ctrl+f**" to show and hide, respectively, layers from lower hierarchy levels.



5. The "X/Y Snap Spacing" (in the "Grid Controls" section) defines how fine is the increment you can do when sizing your layout shapes. Set both these values to 0.005.

6. Click "OK".

2.3 Turn Off Gravity

Before proceeding, be sure to turn off the "gravity" option. This is a feature that allows your cursor to snap to certain critical points on the layout, and we recommend not using it. You can press "**g**" to toggle gravity on and off (the CIW will show a message that gravity has been turned on or off). Alternatively, in the Layout Editor window, go to "Options"→"Editor" (or press "**Shift+e**") and, on the "Gravity Controls" section, uncheck "Gravity On". This will allow you to move your mouse freely and create a metal rectangle anywhere that you want.

2.4 Tip to Avoid Design Rule Check (DRC) Violations

In the Layout Editor window, select "Options"→"DRD Edit...". Then, set "DRD Mode" to "Notify" by checking the corresponding box. Also, set the "Hierarchy Depth" to "Current & Below", so you get

notification of infractions on all sub-levels of the hierarchy. Finally, in the "Interactive Display" section, check the "Halos", "Rule Text", "Arrows" and "Violation Edges" boxes.

2.5 Zooming In/Out

As you continue working with Virtuoso Layout Editor, you will often find it useful to zoom parts of your layout. You can use the mouse wheel, or simply remember these useful hotkeys:

"**z**"	Zoom-in the selected area (hit "**z**", left-click and define the zoom area)
"**Ctrl+z**"	Zoom-in by 2x
"**Shift+z**"	Zoom-out by 2x
"**f**"	Zoom fit

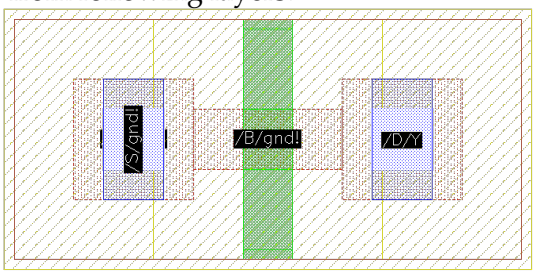
2.6 Transistor Layers

In the Layout Editor window, you should now be able to see multiple layers in the transistor instances as shown below. If you cannot see these layers, then press "**Shift+f**" to display them. All the colors in the transistors represent the multiple layers that are involved in their fabrication process. We will now detail such layers under the following format:

- *`Layer name` (Visual appearance): Layer description*.

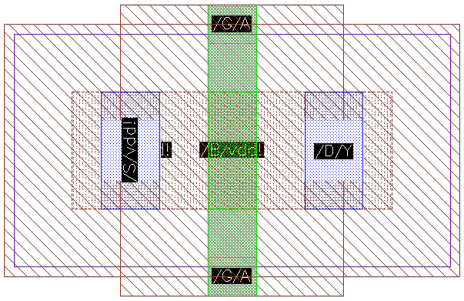
The `nmos` transistor (zoom-in if you need a closer look) is made from following layers:

- `Nimp` (yellow diagonal stripes): n+ implant
- `PWdummy` (unfilled red rectangle): p-well indicator
- `Oxide` (red stripes): oxide
- `Poly` (green): polysilicon
- `Cont` (white-filled squares): source/drain contacts
- `Metal1` (blue): first layer of metal



Similarly, the `pmos` is composed by the following layers:

- `Pimp` (red diagonal stripes): p+ implant
- `Nwell` (purple rectangle dot-filled): n-well indicator
- `Oxide` (red stripes): oxide
- `Poly` (green): polysilicon
- `Cont` (white-filled squares): source/drain contacts
- `Metal1` (blue): first layer of metal



3 Standard Cell Design

We will now start doing our layout. In this class, we will follow a *standard cell approach*, which means all our lowest-hierarchy-level cells will have the same total height, n-well height, and power rails' width. Following this approach will simplify your work when cascading cells, as the power rails and n-wells will align without problem. We will first create the power rails, and then the n-well:

1. Select the `Metal1` layer (remember to always use the drawing `drw` layers). On the Design Canvas (the black region of the Layout Editor window), press "**r**" on your keyboard to create a metal rectangle. Click once on the Design Canvas to start creating a rectangle of `Metal1`; this will define one corner of such rectangle. Then drag your mouse to a new position on the Design Canvas, and click again to

define the opposite corner of the `Metal1` rectangle. Do not worry about the exact position and size of the rectangle; we will fix that next.

2. Click on the newly created rectangle and press "**q**" on your keyboard. Fill the following fields:
   Left: 0 Right: 2 Bottom: -0.08 Top: 0.08
   Click "OK". Note that all distances you see in the Layout Editor have μm as units.
   With this, you have created the ground rail, which will connect your cell to `gnd!`. This rail should be the bottom-most metal in your layout; no other metal or component should appear below it.

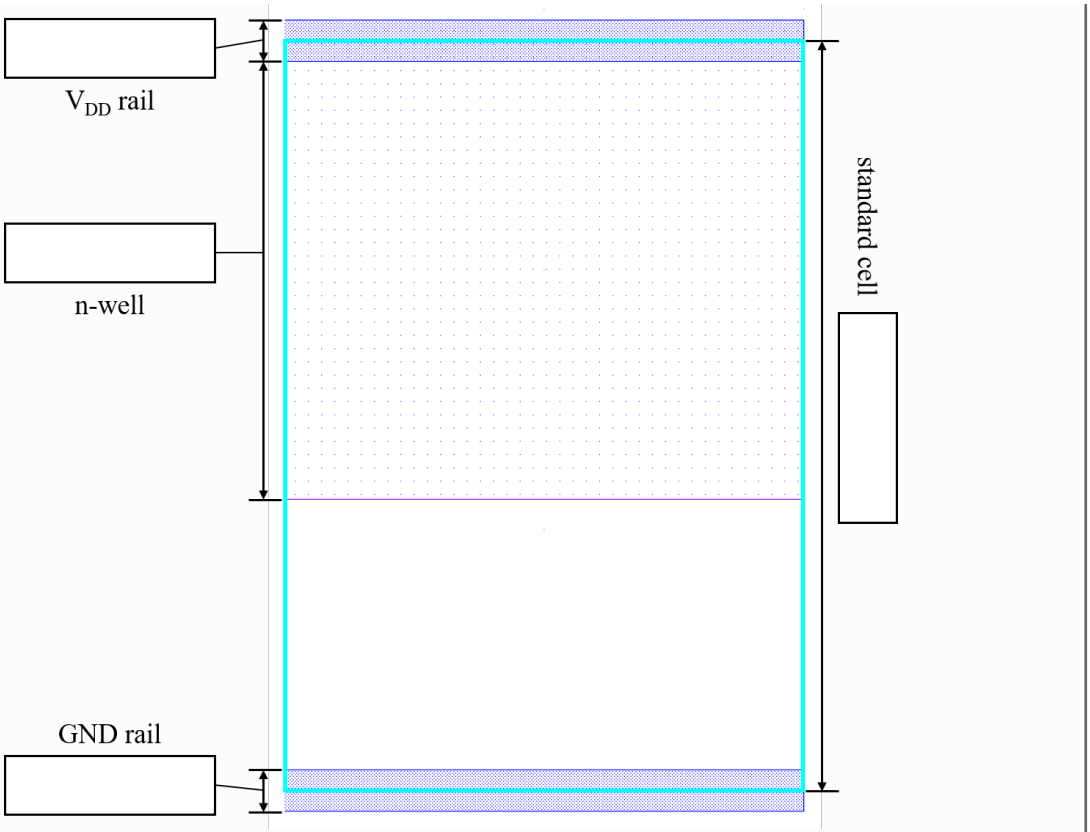
3. Repeat the previous procedure for a new `Metal1` rectangle, but now with these coordinates:
   Left: 0 Right: 2 Bottom: 2.8 Top: 2.96
   You have just created the `vdd` rail. No other metal or component should appear above this rail.
   The next step is to create the n-well. Note that we are working in an n-well process: The wafer is a p-type substrate, i.e. you can consider all the black area in the background as a p-substrate. The `nmos` transistors lie in a p-well, so we can simply use the wafer substrate as this p-well. The `pmos` transistors lie in an n-well so we need to create an n-well for them to lie in:

4. Select the `Nwell` layer. As before, create a rectangle of this layer, with the following coordinates:
   Left: 0 Right: 2 Bottom: 1.12 Top: 2.96

5. Resize the PR boundary to enclose your layout, from half of the ground rail to half of the `vdd` rail (see the figure below for reference). To do so, you can press "**s**" (for "stretch") on your keyboard, hover your mouse over the edge of the PR boundary that you want to extend/shorten until it gets highlighted, and click it. Then move your mouse to stretch the figure as desired and click to apply the changes. Before pressing "**s**" you should make sure no object is selected (including the object that you want to stretch); you can do this by clicking on an empty space of the Design Canvas. In general, it is easier to select what you want when there are fewer things around it. This is where hiding layers (as mentioned before) you are not working with comes in useful.

Student Activity 1: Measuring the standard cell

Right now, your layout should look exactly like the one below. With the help of the ruler tool on the Layout Editor (press "**k**" on your keyboard; click and drag to measure distances), annotate the dimensions (including units) on the following figure. Once you are done, press "**Shift+k**" to clear all your ruler markings.



***Remark**: For all the ECE4740 labs, you must use the dimensions you just measured for the **power rails width**, **n-well height**, **and standard cell height**. Note that the n-well also needs to entirely cover the `vdd` rail. If you ever need your standard cell to be bigger, then you are only allowed to modify its width, but none of the dimensions you just annotated!*

4 Placement

Now, we will place our transistors before doing any kind of connection.

1. Press "**m**" and click on the `nmos` to move the transistor. Note that you can only move it either horizontally or vertically. If you want to move it in any direction, then you can click, on the upper bar, the button shown on the upper right until it becomes the one on the lower right. Put the `nmos` transistor somewhere inside the PR box, but outside the n-well. Click again to stop moving the transistor.

2. Do the same to place the `pmos` inside the n-well.

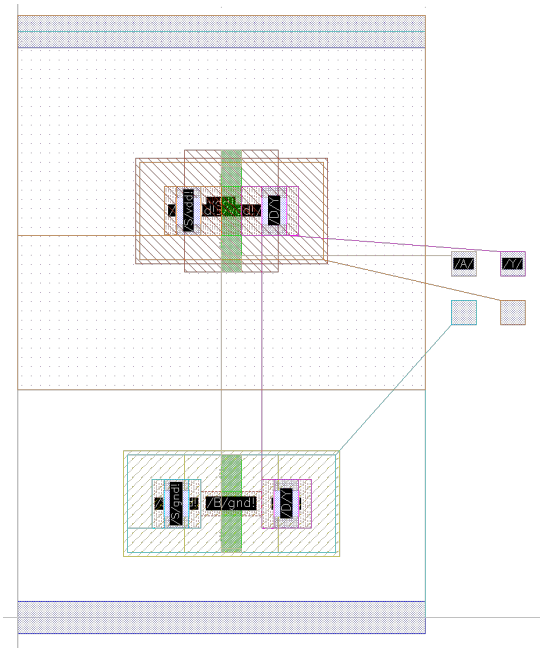
5 Routing

We can now begin routing metal to make all the necessary connections. Wires in the layout are done by drawing rectangles in the different metal layers. For the specific process we are using, there are 9 metal layers, named `Metal1` through `Metal9`. To ensure that they can be properly fabricated, the rectangles that you draw must follow certain design rules, so they cannot be too narrow, be too close to each other, be too big nor too small, among others. For the exact rules, check the `gpdk090` rules manual. As you learned before when creating the power rails, you can create a rectangle by pressing "**r**" on your keyboard, then clicking once to define one of the rectangle's corners, and then a second time to define the opposite corner.

Alternatively, you can create a path by tapping "**p**". Click once every time you want to make a turn in your path, and double click to end the path. Between these two approaches, we recommend the rectangle one.

As you know, both `nmos` and `pmos` are symmetrical devices (with respect to drain and source); this means that either terminal can function as source or drain. Then, it is natural to think that it does not matter how you connect your transistors together, but this is not the case. Since your layout is based on a schematic, their connectivity on the layout has to match that of the schematic. Otherwise, your design will not pass LVS (Layout vs. Schematic) later on.

To make it easier to see what nets should be connected together, you may want to visualize the unconnected nets. You can do this by selecting "Connectivity"→"Incomplete Nets"→"Show/Hide All...". You should now see different color lines connecting different nets in your layout, indicating they should be connected.



Now, connect the following elements:

1. Gates of the transistors together using the `Poly` layer
2. Drains of the transistors together using the `Metal1` layer, to create the inverter's output
3. Source of the `nmos` to the ground rail using `Metal1`
4. Source of the `pmos` to the `vdd` rail using `Metal1`

Remember that you can use the stretch tool (by pressing "**s**" on your keyboard) if needed, and that you can undo/redo your last action with "**u**"/"**Shift+u**".

5.1 Connecting Pins

Pins are the metal regions in our cell to which external wires can connect. Our inverter needs pins for its power supply (`vdd!` and `gnd!`), input (`A`), and output (`Y`). Since the pins are the only areas to which external wires can connect, we recommend making them as big as the metal that connects to the pin (if this is not clear right now, then do not worry; we will illustrate this next). Doing so will simplify routing when using this cell as a component of a bigger design.

1. Move the `gnd!` pin to be on top of the ground rail. Stretch the `gnd!` pin so it has the same dimensions as the ground rail.

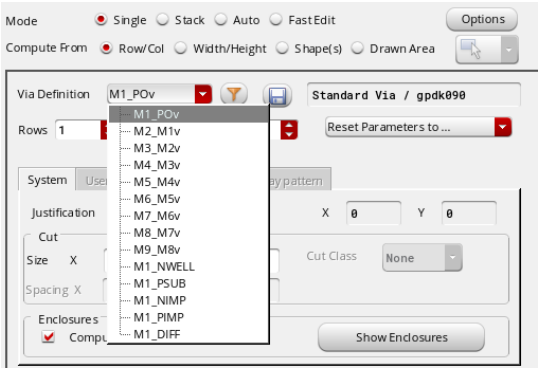
2. Repeat the previous step, but this time with the `vdd!` pin and the `vdd` rail.

3. Move the `Y` output pin to be on top of the `Metal1` wire that you created to connect the transistors' drains. Stretch the `Y` pin so it has the same dimensions than the `Metal1` rectangle below it.

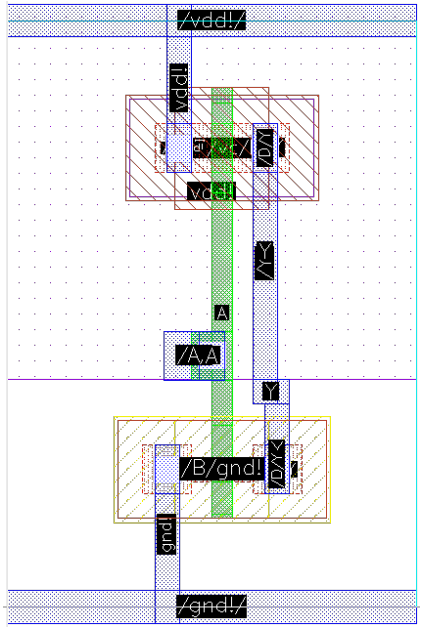
4. The `A` input pin is in the `Metal1` layer, but so far, the node `A` in our layout only exists in the `Poly` layer. As these two elements are in different layers, we need a via to connect them, which we will do next.

5.2 Vias

1. Select "Create"→"Via...", or press the "**o**" key. This will open the "Create Via" window.



2. In the "Via Definition" drop-down menu, select the "M1_POv" option, which will create a via between `Metal1` and `Poly`.
   As you can see in this drop-down menu, there are vias to connect any two adjacent metal layers, plus other options we will soon use. If you want to connect layers that are not adjacent, then select "Stack" in the "Mode" section. Do not change any dimension of a via as it might lead to a design rule violation.
3. Move your mouse over the Design Canvas, and click to add an instance of the via to your layout. Add it on top of the `Poly` strip that interconnects the transistors' gates. If it was needed, then you could keep on clicking to add more vias, but in this case, one is enough. Once you are done, press "**Esc**" on your keyboard.
   This white part is the actual via, a "tunnel" between metal layers. You can see that the via also contains green and blue; these are actual rectangles of `Poly` and `Metal1` layers. Note that, since you are putting the via on top of the `Poly` strip, an electrical connection already exists between them.
4. Move the `A` input pin next to the via and stretch it so it touches the `Metal1` on the via and it has the minimum area required by the design rules.
   Your layout should now look like the one to the right.



5.3 A Note about Routing

As you go up in metal layers, they get thicker and less resistive, therefore they can propagate a signal faster. However, the vias that connect metal layers have high resistance and hence, can be slow. The slowdown of the vias may not make up for the speed-up of higher metal layers. A general rule of thumb is to route local signals in low metal layers and global signals in higher metal layers.

Another useful rule of thumb is to route all even metal layers horizontally only, and all odd metal layers vertically, or vice-versa. This is to minimize the amount of metal on one layer that runs on top (or below) metal from an adjacent layer. This "overlap" between metal strips of different layers creates parasitic capacitances that can impact negatively your design's performance.

5.4 Well Taps

Now we must connect our substrates to `gnd` and `vdd`. This is basically connecting the bulk of our `nmos` and `pmos` to `gnd` and `vdd`, respectively. The substrate taps are a via that you can select from the "Create Via"

menu just like other vias.

1. Press "**o**" and select `M1_NWELL` to add an n-well tap. Place it outside the PR box.

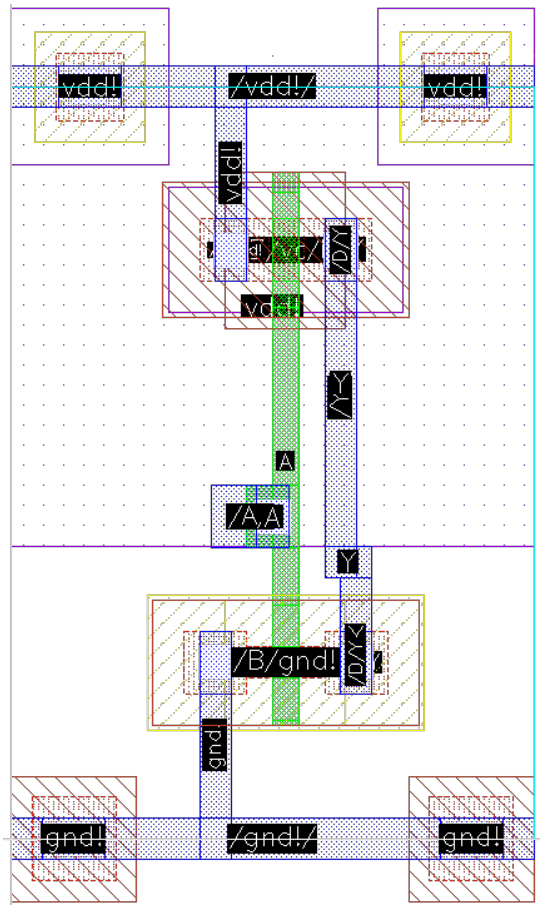
2. Select the n-well tap and press "**q**". In the "Enclosures" section, on the `Metal1` column, change both the "Top" and "Bottom" entries to 0.02. Do not change the values on the "Oxide" column. Click "OK".

3. Move the n-well tap to be on top of the `vdd` rail, on its left extreme. Be careful to align the `Metal1` rectangle on the n-well tap exactly with the rail.

4. Press "**c**" and click on the n-well tap to copy it. Move the copy to the right extreme of the `vdd` rail.

5. Repeat the four previous steps, but this time using `M1_PSUB` to add a p-substrate tap on top of the ground rail.

A general rule of thumb is to add many substrate taps when possible. Your layout should now look like the one to the right.



6 DRC (Design Rule Check)

Now that we have finished drawing our layout, we must check that it meets all the design rules. The design rules are provided to increase the chances of correct fabrication of your circuit in the presence of inevitable process variations, such as misalignment of masks and variations across wafers, just to mention a few. The tool that does this check is called DRC. DRC checks the layout to ensure that it conforms to the design rules for the technology. For instance, the metal lines must be separated by a certain distance to accommodate process variations and limitations. It is a good idea to run DRC often as you go, to ensure that each small change of your design conforms to all of the rules.

1. To run DRC, go to "Verify"→"DRC". Make sure the "Rules File" field is set to `divaDRC.rul` and the "Rules Library" field is checked and set to `gpdk090` . Click "OK".

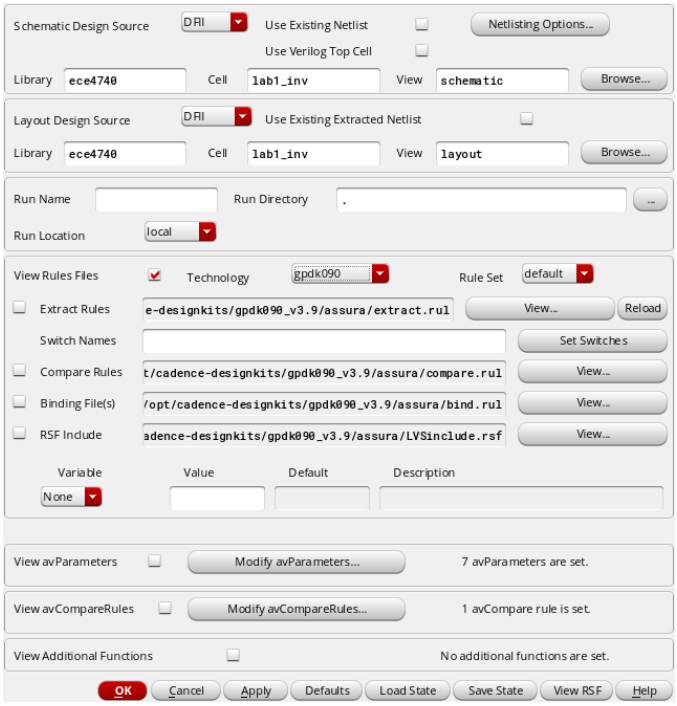
2. Check the CIW to see the results of DRC. Here you will see a printout of each DRC error. In the Layout Editor window, you will see flashing white lines indicating where an error exists.
   To easily examine each error, select "Verify"→"Markers"→"Find...". Tick the "Zoom To Markers" checkbox, and click "Next" to move to the next error marker. Sometimes the flashing white lines can be annoying! To delete the DRC markers, go to "Verify"→"Markers"→"Delete All...".

3. Fix all DRC errors. Run a final DRC to make sure your layout is free of DRC violations.

7 LVS (Layout versus Schematic)

LVS compares the circuit implemented on your layout against your schematic(s). When the design grows large, it is quite common to make silly mistakes such as shorting two lines which are not supposed to be connected, leaving floating nodes, among others. LVS helps you to find these problems by comparing the schematic netlist with the layout's extracted netlist to ensure they match.

1. We start by configuring the technology files. Go to "Assura"→"Technology...". Insert the following in the "Assura Technology File" field:
   `/opt/cadence-designkits/gpdk090_v3.9/assura_tech.lib` ; click "OK".
2. Now, we are ready to run LVS. Go to "Assura"→"Run LVS...". Make sure that the "Schematic Design Source" section is the schematic view of the `lab1_inv` cell (or whatever you named your cell), and that the "Layout Design Source" section is the layout view of the same cell.
3. Using the drop-down menu, set the "Technology" field to `gpdk090`, and the "Rule Set" field to `default`.
4. Check the "View Rules Files" box. Your Assura LVS configuration window should look like the one to the right.
5. Press "OK". A window will show up saying that Assura LVS is running.
6. Wait until another window shows up giving the results of the LVS run.



If your netlists match, then you will see a message saying "Schematic and Layout Match", indicating that you have completed layout successfully. Otherwise, you will see a summary of the LVS problems, in which case you need to correct the layout. In either scenario, you *must* click "Yes" in this new window. The "LVS Debug Environment" will open. Here you can find more information about the mismatches between the layout and the schematic. Use this to determine what needs to be fixed. After fixing your layout, close the "LVS Debug" window and run LVS again to verify that all problems have been solved.

8 Parasitic Extraction

As you will see (or have already seen) in lecture, the physical design of a layout introduces parasitic elements that are not accounted for in your schematic, but that have an impact on your circuit's performance. Examples of these parasitic elements are the wires' resistance, and the capacitance between different wires, just to name a few. In this sense, good layout designs become critical to attain circuits that perform as expected, without significant degradation. To have a better approximation of how will the circuit behave after being fabricated, a parasitic extraction is needed, which will create a netlist that includes parasitic resistors and capacitances. We will perform this extraction using a tool called "Quantus QRC".

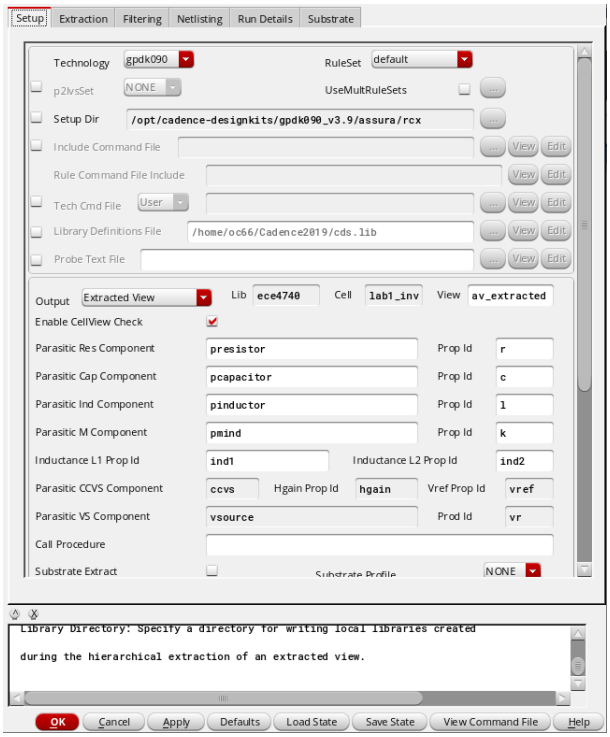
1. In the Layout Editor window, go to "Assura"→"Run Quantus QRC...".
2. In the "Setup" tab, set the "Technology" field to `gpdk090` and the "RuleSet" field to `default`.

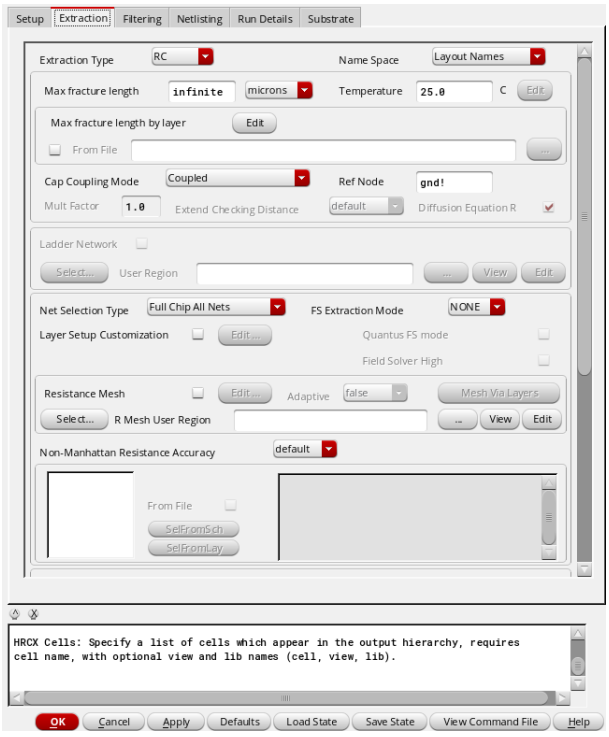
3. Still in the "Setup" tab, set "Output" field to "Extracted View". Make sure the "Lib" and "Cell" entries correspond to the cell you are working on, and enter `av_extracted` in the "View" field.

4. Verify that the "Setup" tab looks like the one shown below to the left.

5. Go to the "Extraction" tab. Set "Extraction Type" to `RC`, "Cap Coupling Mode" to `Coupled`, and "Ref Node" to `gnd!` (do not forget to include the exclamation mark). Verify that this tab looks like the one shown below to the right.

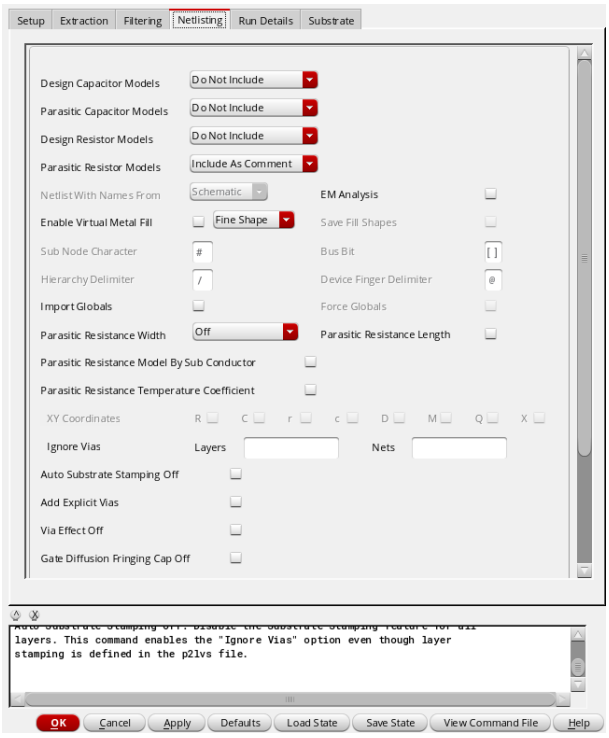
6. Go to the "Netlisting" tab. Set the fields "Design Capacitor Models", "Parasitic Capacitor Models", and "Design Resistor Models" to `Do Not Include`. Verify that this tab looks like the one shown on the next page. Click "OK".





A window will show up informing you that Quantus QRC is running. After waiting for a while, a dialog box will appear, telling you that Quantus QRC has completed successfully. Should you get an error, go back and review the configuration.

Now, go to the Library Manager (in the CIW, go to "Tools"→"Library Manager..."). Go to the `lab1_inv` cell, and you should see the extracted view that we just created, called `av_extracted`. Double-click on it to open it. This will open a window similar to the Layout Editor, except that it contains the elements that were identified during the parasitic extraction. Press "**Shift+f**" to see all the hierarchy levels. You will see the transistors that form your inverter, as well as the metal lines, but also, if you zoom in, then you will see capacitors and resistors that were obtained during the extraction.



9 Post-layout Simulation

Now that we have extracted the parasitics from our circuit, we can re-run the simulation done in section 3 ("Transient Simulation") of Lab 1 Part 1 to see their effect. Since this simulation includes the effects of the layout we have done, it receives the name of "post-layout simulation".

1. Open the `lab1_sim` schematic. Verify that the inverter's input is a `vpulse` cell with the parameters given in section 3 of Lab 1 Part 1.
2. Launch ADE L and load the state used in section 3 of Lab 1 Part 1.
3. Set a transient analysis, just like it was done in section 3 of Lab 1 Part 1.
4. Run the simulation.

Student Activity 2: Measuring the propagation delays in the post-layout simulation

Plot the input and output voltages of the inverter. Then, use markers to find the low-to-high and high-to-low propagation delays. Your results should be different from the ones obtained in the "Student Activity 5" of Lab 1 Part 1. Save two copies of the plot: One including the markers used to find t_{pLH}, and another one with the markers used to measure t_{pHL}.

t_{pLH}: _____ t_{pHL}: _____

Note that, to execute the post-layout simulation, you followed exactly the same procedure as for the schematic simulation done in Lab 1 Part 1. So you might be asking yourself: How did Cadence Virtuoso know that it should use the extracted circuit netlist instead of the schematic one? To understand this, in the ADE L window, go to "Setup"→"Environment". Then, check the contents of the "Switch View List" field. This field tells the tool the preference order of the view to be used for simulation of each cell. Note that `av_extracted` appears before `schematic`, which means that the `av_extracted` view should be preferred

over the `schematic` view for simulation. For the simulation done in Lab 1 Part 1, our inverter cell still did not have an `av_extracted` view, so the simulation was based on the `schematic` one. Now, in Lab 1 Part 2, the inverter has an `av_extracted` view, so the netlist for simulation was taken from there. If you ever want to simulate your circuit again using the `schematic` view, then you can either change the "Switch View List" so `av_extracted` appears *after* `schematic`, or you can also remove `av_extracted`. You are encouraged to try this out to familiarize yourself with the procedure. However, keep in mind that all the results that you report for the labs in this class have to be from the post-layout simulation, unless otherwise noted.

Student Activity 3: VTC, revisited

Generate again the VTC curve for your inverter, but this time doing a post-layout simulation. Review section 4, "DC Analysis: VTC Curve", of Lab 1 Part 1 if you need help to setup the simulation. Save a copy of the plot. Is your VTC curve different from the one in Lab 1 Part 1? Explain.

Lab 1 Part 3: Inverter Sizing using Cadence Virtuoso

Report Due Date: February 17, 2020 at 11:59 pm

Real-Time Evaluation Date: February 14, 2020

- You will learn how to do a parametric analysis and use the "Calculator" Tool in Cadence Virtuoso.
- You will find the optimal size for an inverter using simulation results.
- **Note: Please read *all* instructions carefully! If you miss one step, then nothing may work and it will be nearly impossible to figure out what is going on.**

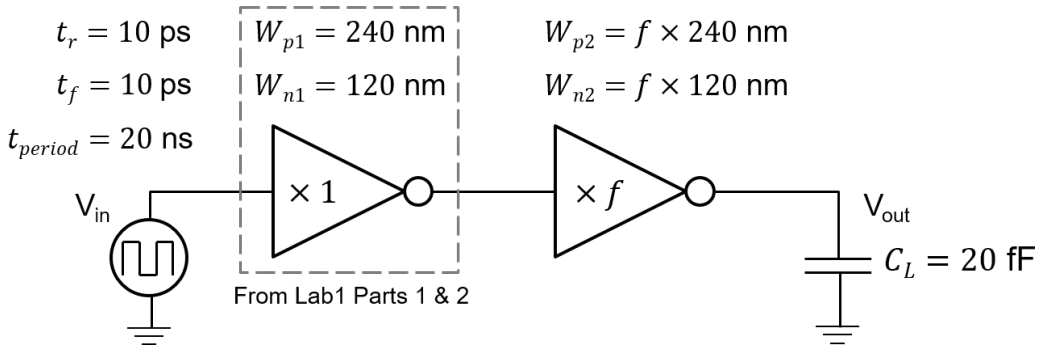
As you may well know, the accuracy of hand analysis for even relatively simple digital systems depends on the accuracy of our circuit models, which may often be quite involved. A good model must be reasonably accurate, yet tractable, in order to provide intuition into the operation of a circuit, without getting lost in the complicated physics and mathematics that clutter the design process. Once a first-pass design has been achieved through simplified models, then second and third order effects may be taken into account by simulation to optimize the design. In this Lab 1 Part 3, we will be using the simulation results to optimally size the second inverter on a two-stage inverter chain.

1 Inverter Sizing with Parametric Analysis

Consider the schematic shown below: The dimensions of the inverter on the left are fixed to $W_{\text{p1}} = 240$ nm, and $W_{\text{n1}} = 120$ nm. An inverter with these widths is said to have a "size" or "driving strength" of one, which is commonly represented with $\times 1$. The inverter on the right keeps the W_{p2}/W_{n2} ratio $2:1$, but its transistors are f times wider than the ones of the $\times 1$ inverter. We say that such inverter has a size of f, which is represented with $\times f$. The $\times f$ inverter is driving a fixed load C_L, and, in what follows, we would like to find the sizing factor f that minimizes the propagation delay between V_{in} and V_{out}.

1. Create a new cell with the schematic shown below. Include a `vdc` voltage supply that drives `vdd` to 1.2 V. For both inverters, use the `lab1_inv` cell. For the `vpulse` V_{in} signal generator, use:

Voltage 1:	`0 V`	Period:	`20n s`	Rise time:	`10p s`	Pulse width:	`10n s`
Voltage 2:	`1.2 V`	Delay time:	`10n s`	Fall time:	`10p s`		



2. Select the $\times f$ inverter, and press "**q**". The "Edit Object Properties" window will show up. Click the "Add" button. This will open the "Add property window". Fill in the "Name" field with `m`, and the "Value" field with `f`. Press "OK".

3. Look at the bottom of the "Edit Object Properties" window. You should see "m" as a "User Property" of the inverter, with a "local value" of "f". In the "Display" drop-down menu, select the `both` option. Press "OK". Now, on the Design Canvas, you should see "`m=f`" above the $\times f$ inverter.

   What did we just do? We want to vary the size of the $\times f$ inverter, but we do not want to create a different schematic for each inverter size we try. Instead, we will modify a parameter that every cell has: The "multiplicity" parameter, which is represented in Cadence Virtuoso with "m". Let us explain this parameter with an example: If "m" has a value of 2, then the circuit will be simulated as if two instances of the cell were connected in parallel. In the case of CMOS circuits, this would have the same effect as increasing the width of all transistors in the cell by a factor of $m = 2$.

   While the multiplicity parameter can be applied to any cell, it is not included in the default properties of the cells. This is why we had to define "m" as a "User Property" on step 2 above. Since we will be trying different size factors, we assigned this parameter a value of "f", which will work as a variable in the simulations we will perform next. On the other hand, what we did on step 3 was to make the "m" parameter visible on the schematic, so our schematic reflects accurately what will be simulated.

4. Launch ADE L and load the state used in Section 3 of Lab 1 Part 1. ***Remark**: From now on, every time you are asked to launch ADE L, you should also always load the state used in Section 3 of Lab 1 Part 1, or equivalently, repeat steps* 2, 3, 4 *and Student Activity* 4 *from that same section.*

5. In the ADE window, right-click the white area of the "Design Variables" section, and select "Edit".

6. A new window will show up. Fill in the "Name" field with `f`, and write 1 in the "Value (Expr)" field. Click the "Add" button, and then click "OK".

   With this, we have assigned a value of 1 to the `f` variable that controls the multiplicity of the $\times f$ inverter. We will now proceed to perform a "Parametric Analysis", in which we will vary the value of `f` to see its effect on the propagation delay from V_{in} to V_{out}.

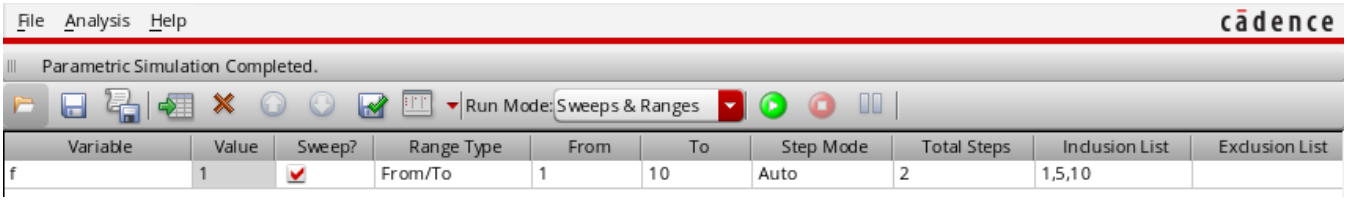
7. Add the input and output nodes (where V_{in} and V_{out} are, respectively) of the schematic to the "Outputs" section of the ADE L. If you do not remember how to do this, then look at step 5 from Section 4 of Lab 1 Part 1.

8. Select "Analyses"→"Choose..." from the ADE window, or click the "Choose analysis" button.

9. Choose `tran` under "Analysis". Enter `100n s` in "Stop Time", and check the `moderate` box under "Accuracy Defaults". Make sure that the "Enabled" box at the bottom is checked, and click "OK".

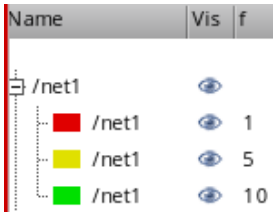
10. In the ADE window, go to "Tools"→"Parametric Analysis...". This will open a new window.

11. In the "Variable" field, use the drop-down menu to select the variable `f`. Verify that the "Range Type" field is set to `From/To`. In the "From" field, enter 1, and in the "To" field, enter 10. Leave the "Step Mode" in `Auto`, but enter 2 for "Total Steps". Then, in the "Inclusion List" field, enter `1,5,10`. Verify that your window looks now like the one below.



By doing this, we have configured the parametric analysis to run the simulation for the cases where the variable `f` (and hence, the multiplicity of the $\times f$ inverter) takes the values 1, 5, and 10. Note that one could also use linear and logarithmic steps to modify the variable's value, or specify the range in which the variable will vary by indicating the center and a span of the range.

12. In the Parametric Analysis window, start the simulation by going to "Analysis"→"Start Selected", or by pressing the green stoplight button on the top right region of the window. If you are pressing the green stoplight button, then make sure that you are pressing the one located in the Parametric Analysis window, and not the one in the ADE window.
13. Once the simulation has finished, a plot with different curves will appear. This plot contains V_{in} and V_{out} across time, for the different values that the variable `f` takes. To know which curve corresponds to what case, go to the left panel in the plot window, and click on the "+" sign next to each net. Now you can check what color corresponds to what node and variable value. You can also show/hide each plot by clicking on the eyes that appear in the "Vis" (for "visibility") column.



Student Activity 1: Propagation delay for different sizing factors

Measure the propagation delay when both input and output signals rise, $t_{\text{pd,LH}}$, as well as the propagation delay when both signals fall, $t_{\text{pd,HL}}$, for the different values that `f` takes. Report your results on the table below. Does increasing `f` always decrease the propagation delay?
***Tip**: To measure the propagation delays precisely, start by placing a marker nearby the point of interest. Then, click on the marker, and press "**q**". In the "Position" section of the emerging window, use the drop-down menu to select "byYMode", and then write `600mV` in the right-most field. This will snap your marker to the point on the curve whose y-coordinate is closest to 600 mV. Note that here we are using 600 mV as $V_{DD} = 1.2$ V, and we measure propagation delays with respect to $V_{DD}/2$.*

`m=f`	1	5	10
$t_{\text{pd,LH}}$			
$t_{\text{pd,HL}}$			

As you can see, increasing the sizing of the $\times f$ inverter does not always decrease the propagation delay. At first, increasing `f` from 1 to 5 does help, as the $\times f$ inverter is able to deliver a greater current to quickly

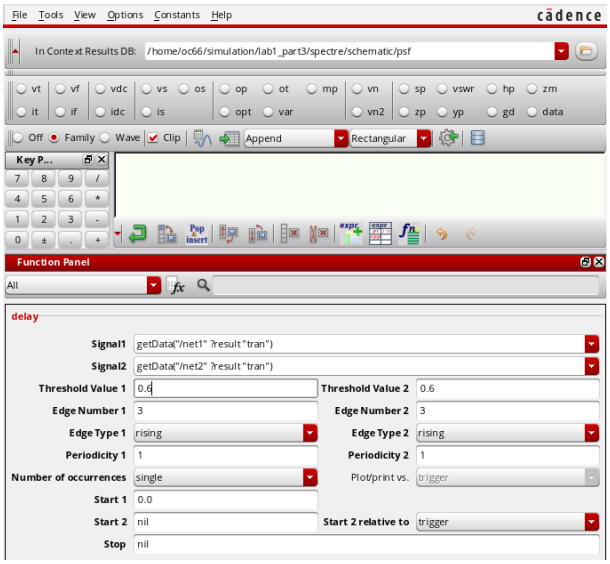
charge up C_L. However, as you increase `f`, its input capacitance also increases, so the $\times 1$ inverter takes a longer time to drive the $\times f$ one. As you just saw in the previous Acctivity, there is a moment where the $\times f$ inverter becomes such a great load for the $\times 1$ inverter that the total propagation delay increases.

2 Finding the Optimal Sizing Factor using the Calculator Tool

In the previous section, we saw how increasing the driving strength of the $\times f$ inverter does not always decrease the propagation delay. Then, there must be an optimal sizing factor `f` that minimizes such propagation delay. To find it, we should repeat the parametric analysis for many more values of `f`. In such case, measuring the propagation delays using markers will quickly become a highly tedious, long, and prone-to-error process. To avoid this, we will learn now how to use the "Calculator" tool to automate these measurements.

1. In the ADE window, go to "Tools"→"Calculator...". This will open the Calculator window.
2. Verify that on the top region of the Calculator window, the "Family" radio button is selected.
3. In the "Function Panel", select "All" from the drop-down menu, and then click on `delay`.
4. Click on the "Signal 1" field, clear it, and then click on the input net on the left panel of the plot window. Do the same with "Signal 2" and the output net.
5. Set the following parameters:
   (a) "Threshold 1" and "Threshold 2" to 0.6.
   (b) "Edge Type 1" and "Edge Type 2" to `rising`.
   (c) "Edge Number 1" and "Edge Number 2" to 3.

   Your window should look like the one to the right.



6. Press "OK". You will see an expression appear above the Function Panel.

   With steps 3 to 6, you have created a function that measures the delay between the moment in which the input voltage's third rising edge is equal to 0.6 V and the moment in which the output voltage's third rising edge is equal to 0.6 V. In other words, you are measuring $t_{\text{pd,LH}}$.
7. Repeat steps 3 to 5, but this time assign "Edge Type 1" and "Edge Type 2" to `falling`. This will measure $t_{\text{pd,HL}}$. Press "OK".
8. Note that the new delay expression overwrote the previous one. Add a "+" after the current expression, and then press the button "Pop & Insert" button (shown to the right). This will insert the previous expression that we created to measure $t_{\text{pd,LH}}$.

9. Enclose the current expression with parenthesis, and add "`/2`" at the end. Your expression window should have a format similar to: `(delay(...)+delay(...))/2`. By doing this, you have created a new expression that calculates the average between $t_{\text{pd,LH}}$ and $t_{\text{pd,HL}}$, which we will refer to as t_{pd}.
10. Add this expression to your simulation outputs by pressing the "Send buffer expression to ADE outputs" (shown to the right).

11. Go to the ADE window and verify that the expression appears in the "Outputs" section. Uncheck the "Plot" boxes for the input and output voltages, but keep the box for the expression checked.

12. Go to the Parametric Analysis window. Change "Step Mode" to `Linear Steps`, "Step Size" to `1`, and empty the "Inclusion List" field.
13. Start the simulation from the Parametric Analysis window.

After the parametric analysis finishes, a plot will show up, which graphs the propagation delay through the inverter chain, t_{pd}, versus the size of the $\times f$ inverter. Save a copy of this plot.

Student Activity 2: Determining the optimal inverter size

With the plot you just generated, find the **integer** `f` that gives you the shortest t_{pd}: _____

Lab 1 Part 4: Integrating Standard Cells and the Effects of Wiring

Report Due Date: February 17, 2020 at 11:59 pm

Real-Time Evaluation Date: February 14, 2020

- You will create the layout for the two-stage inverter chain you analyzed on Lab 1 Part 3.
- You will see how a long wire can have an impact on the propagation delay of your circuit.
- You have to complete a PowerPoint report.

For this lab, you have two tasks: (i) create the layout for the inverter chain that you sized in Lab 1 Part 3, and (ii) modify such layout to increase the length of the wire that interconnects them.

1 Layout of inverter chain

1. Create the layout for the two-stage inverter chain from Lab 1 Part 3, using the sizing factor f that you found on Student Activity 2 of Lab 1 Part 3.
2. Successfully perform DRC, LVS, and parasitic extraction on your layout.
3. Measure the area (given by the smallest rectangular bounding box) in μm^2 of your circuit.
4. Simulate the schematic from Lab 1 Part 3, this time using the extracted view of your layout. Measure $t_{\text{pd,LH}}$ and $t_{\text{pd,HL}}$; save the plots were you make these measurements.

2 Effect of wiring

1. Create a copy of the cell where you interconnect the two inverters.
2. Modify the layout so the two inverters are separated at least 8μm. Re-wire the connection between these two inverters.
3. Simulate the schematic from Lab 1 Part 3, this time using the extracted view of your modified layout. Measure $t_{\text{pd,LH}}$ and $t_{\text{pd,HL}}$; save the plots were you make these measurements.

3 Rules

1. Use $V_{DD} = 1.2$ V.
2. Adhere to the standard cell dimensions that you measured in Lab 1 Part 2.
3. Design the inverters in a hierarchical fashion (i.e., layout for every inverter).
4. Make sensible symbols for each cell in your hierarchy.

4 Hints

1. Modify the layout of the $\times 1$ inverter you made in Lab 1 Part 2 to reduce its width.
2. Use "fingers" for your transistors. This is a property of the transistors (you can find it in the "Object Properties" that shows up when pressing "**q**" after selecting a transistor) that you can modify from the schematic view. It allows you to create wider transistors by connecting several transistors in parallel (with shared diffusions) instead of literally making the transistor wider. If you do not use fingers, your transistors might exceed the standard cell's height. Do not forget go connect the different gates (i.e., the "fingers") together; you can do such connection in polysilicon.
3. A suggested cell hierarchy *for the first task*, "Layout of inverter chain", would look like this:
   (a) One cell for the $\times 1$ inverter
   (b) One cell for the $\times f$ inverter
   (c) One cell for the two previous inverters connected in series (for the second task, you will need another cell for the modified layout)
   (d) One cell for the testbench (i.e., where you drive the input of the $\times 1$ inverter, and add the capacitor at the output of the $\times f$ inverter); this cell should only contain the schematic view.
4. Route adjacent metal layers orthogonally (e.g., M1 vertically, M2 horizontally, M3 vertically).
5. If the well taps are complicating the interconnection of standard cells or increasing their area, you can remove them after performing DRC, LVS, and parasitic extraction. However, you can only remove the taps from the non-top cells of your hierarchy (in this case, the cells of the inverters alone). Should you remove the well taps, you need to add the well taps in the layout of the top cell of your hierarchy (in this case, the one in which both inverters are interconnected); otherwise, DRC and LVS will not pass. We will only verify that DRC and LVS are passing for the layout of the top cell.

5 Grading

Grading consists of a real-time evaluation by a TA and a report.

The real-time evaluation will take place in PHL 314 on February 14, 2020. 8 minute time slots will be scheduled by students individually.

An evaluation time will be scheduled with your TA. First, the TA will verify the delay you measured (with the standard cells close together), area, DRC, and LVS. Then the TA will ask you **individually** to perform one of the tasks below. The selection will be random and not volunteered. Possible tasks are:

- Rotate the layout of the entire buffer by 90° and then pass DRC and LVS

- Re-simulate the delay using a resistive load or a different VDD
- Perform a transient simulation with a sine wave input
- Add one of the inverters to the chain and re-simulate delay

You will receive an individual grade of 0-5 based on the evaluation:
5: finished task quickly without TA help
4: finished task with little TA help
3: finished task with significant TA help
2: some clues of how to perform the task
1: little clue of how to perform the task
0: absent without reason; does not know how to start

Report: Fill out the PowerPoint report template posted on Canvas. Make sure your schematics and plots are clear and legible. You can label Cadence screenshots or draw the schematics in a program of your choice. The objective is to create a concise report that can be quickly evaluated by a TA.

Turn in a zip file with your *schematics, symbols, layouts, and the report*. Name the zip file **Lab1-(NetId).zip**. The report is due on February 17, 2020 at 11:59 pm.

Important: *Do not cheat!* We are going to randomly try out 10 out of all the submitted designs and compare it to the reported numbers. In case you fudge on benchmark numbers, copy the schematic and layout from previous years' lab, or the design does not pass DRC or LVS, you will get ZERO points for the entire lab.

Last updated: February 2, 2020 by Thomas Tapen