SECTION OF ELECTRICAL ENGINEERING

EPFL STI – SEL
ELG
Station nº 11
CH-1015 Lausanne

Téléphone : +4121 693 13 46
Fax :
E-mail : alexandre.levisse@epfl.ch
Site web : https://sti.epfl.ch/fr/sel/

EE429 2025/2026: Full custom labs 3 : Layout

SEL September 2025

PRACTICAL LABORATORY SESSION No. 3
Physical Design (Layout) and Verification

1. OBJECTIVES

The first part of this laboratory session will introduce the ***Layout Suite XL Editor***. You will learn the basics of the IC layout techniques on a simple example of the NAND logical gate. This tutorial will take you through all the necessary steps. At the end of the first phase, you will learn how to perform all the necessary layout verification steps. In the second phase, you will use the acquired knowledge to autonomously design the layout for the 4-to-1 MUX schematic that we designed in the first session.

First, you will design your own schematic of a 2-input NAND gate at the transistor-level. Then you will perform a simple transient test to make sure the circuit is functional. Finally, after the design has been validated, you will create the mask layout of this component and check the design rules for it. As the last step, you will extract the layout parasitic and compare the simulation results with the schematic.

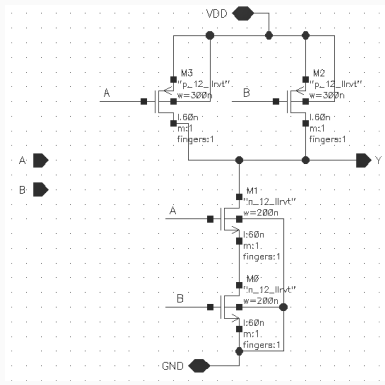
In the final stage, you will apply your knowledge to the 4-to-1 multiplexer and perform the full design flow as you have practiced on the NAND gate.

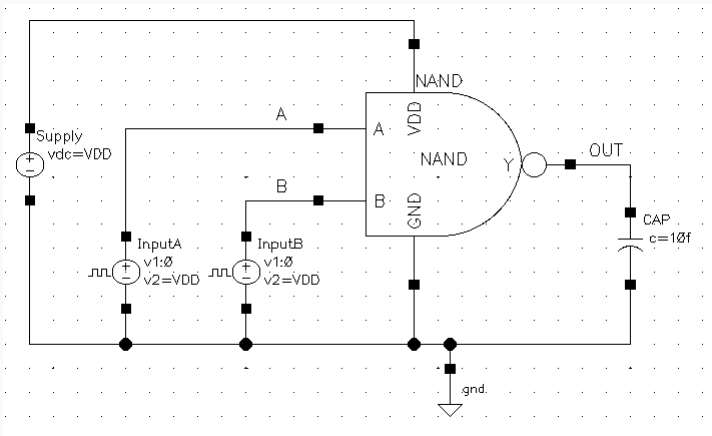
1.1. PREREQUISITES

- ✓ Start the Virtuoso Design Environment ***under your existing project directory*** – **`EE429_FULLCUSTOM`** (as described in laboratory session no.1, section 3.2).
- ✓ If the Library Manager does not appear automatically, activate it by selecting ***Tools→ Library Manager,*** from the **CIW** window.
- ✓ If a window appears asking for the available licences, simply click **Yes** and proceed.
- ✓ Use Library Manager and make sure that when creating cell views, the view type is set to **Schematic**. The default view name should be **schematic**. Also note that some windows may appear below other windows.

2. CREATING THE 2-INPUT NAND GATE SCHEMATIC

- ✓
- ✓ Use the ***Library Manager*** and in the **EDATP** library, create a new schematic called **NAND.**
- ✓ Draw the **NAND** schematic as shown in the figure. Use the minimum gate length, and the widths equal to ***Wn = 200nm*** and ***Wp = 300nm.*** Name the transistors as shown.
- ✓ ***Check and Save*** the schematic and create a symbol for the **NAND**.
- ✓ Again use the ***Library Manager*** and in the **EDATP** library, create another schematic called **NAND_TB.**
- ✓ Draw the **NAND** testbench as shown in the figure below.





- ✓ Create a meaningful transient analysis test and test the functionality of the circuit. Use ***VDD = 1.2V***, minimum ***Period = 2ns***, and the load capacitor ***C = 10fF***.

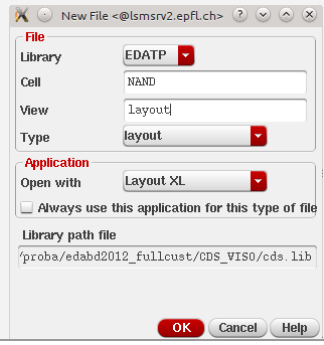
3. CREATING AND VERIFYING THE LAYOUT

3.1. THE LAYOUT EDITOR

As mentioned in the laboratory session no.1, there are different tools used for graphical editing. The tool used as a **layout editor** is called ***Virtuoso Layout Editor***. The goal of this session will be to draw a layout for the **NAND** gate.

- ✓ Create a new cell view in your **EDATP** library. Select **EDATP** library and **NAND** cell and choose ***File→ New→ CellView***.
- ✓ Select **layout** as the view name, and Layout XL as the desired ***Application*** as indicated in the figure.

  *Notice the **Type** field. It should be set to **layout** automatically as soon as you type layout as the desired **View**. If a license window pops-up, click **YES**.*



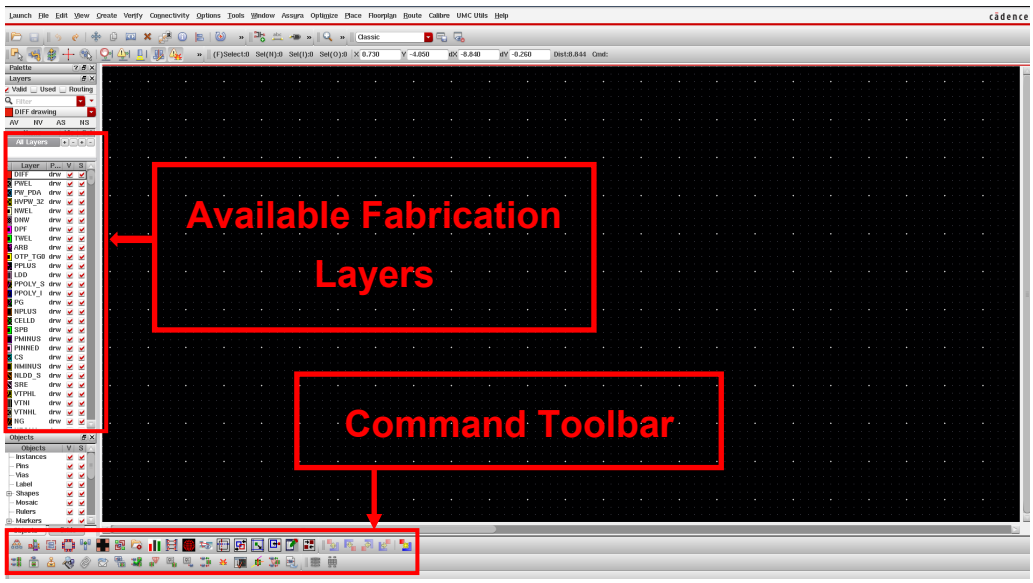


Figure 1 - Virtuoso Layout Editor

The Virtuoso Layout Editor window will appear. All available fabrication layers are listed in the layer sub-window (see Figure 1). Many useful commands can be found in the command toolbar below the drawing surface. Depending on the type of the workspace, different sub-windows and design assistants may be available. In Figure 1, we removed all the additional features that are not needed for this stage of the project.

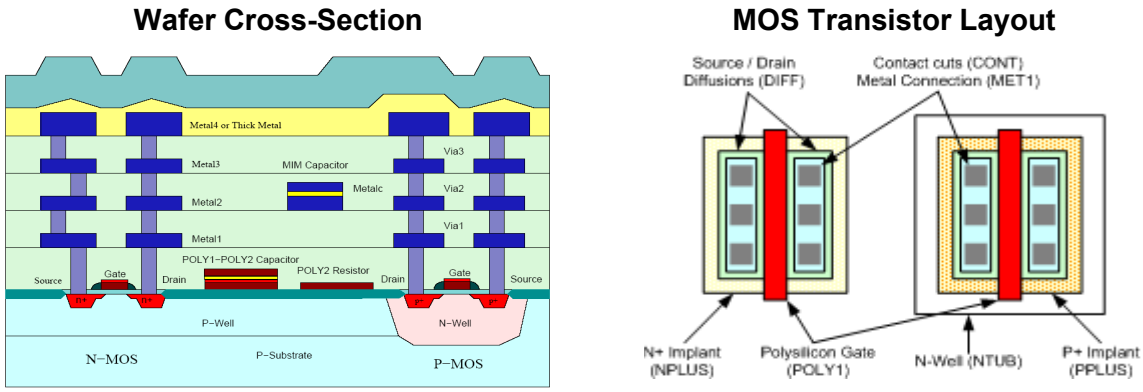


Figure 2 – Cross-section and layout of a MOS transistor.

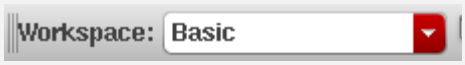
Integrated circuits are manufactured in multiple fabrication steps and are composed of a large number of planar layers. Multiple steps can be required to fabricate a single layer of a specific material (aluminum, copper, silicon di-oxide, polycrystalline silicon, etc.). Integrated circuit layout is composed of the geometric shapes that correspond to the actual patterns of these materials in all the different layers. Based on the layout information, precise masks are generated and provided to the fabrication tools. Layout therefore often contains additional information necessary for the proper fabrication such as dummy or dummy blockage metal layers, different doping layers, salicidation layers etc.

For layout drawing, similar as for schematic composition, multiple useful keyboard shortcuts exist. Short list of the most important **keyboard shortcuts** is given in the following **table:**

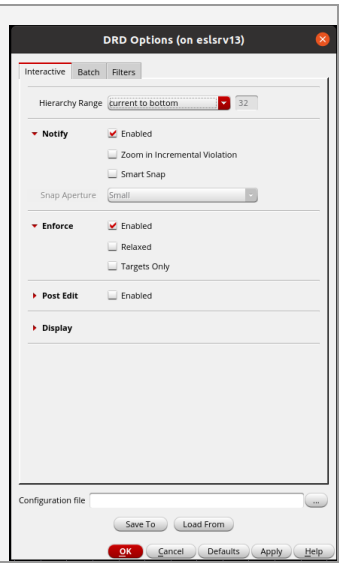
Shortcut	Menu point	Description
u	Edit ▷ Undo	Undo, most important command. Undo levels are limited !
r	Create ▷ Rectangle	Rectangle the basic building block of the layout
p	Create ▷ Path	Lets the user draw a minimum width path between two points and even change layers while doing so.
i	Create ▷ Instance	Instantiates another design in the current design
o	Create ▷ Contact	Adds a special instance from the technology library to connect two layers.
CTRL+p	Create ▷ Pin	Adds a 'logical' layer specifying an electrical contact
c	Edit ▷ Copy	Copy selected instances
m	Edit ▷ Move	Move selected instances
s	Edit ▷ Stretch	Modify the shape of objects
F3	N/A	Shows additional properties for the selected command.
F4	N/A	Toggles partial and full select. When selection method is partial, edges of rectangles become selectable.
f	Window ▷ Fit All	Fits the entire design into the layout window
z	Window ▷ Zoom ▷ In	Lets the user select a rectangular area for zooming in
k	Window ▷ Create Ruler	Creates a ruler to measure distances
q	Create ▷ Properties	Change and view properties of objects and shapes
e	Options ▷ Display	Changes settings of the display
CTRL+f	N/A	Shows only the current level of hierarchy. The instances in the lower level are shown using a red instance box.
SHIFT+f	N/A	Show all hierarchy layers. All drawing layers regardless of hierarchy are shown.

Table 1: Keyboard Shortcuts (Note that in the latest versions of Virtuoso (6.1.6 and higher), the ***p*** shortcut creates wire instead of path)

- ✓ Now, we will draw the layout of our NAND gate.
- ✓ Set the workspace in the ***Layout Editor*** to ***Basic*** so that you can see the ***Navigator***



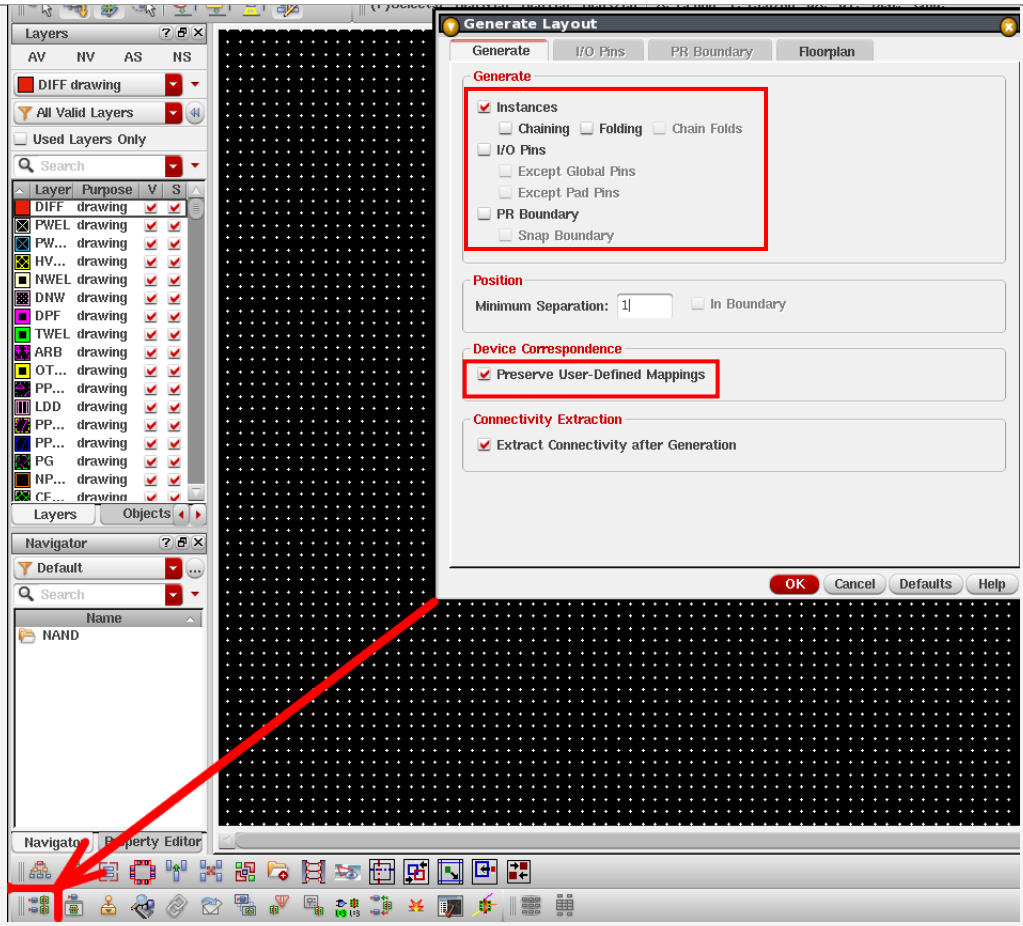
- ✓ Before we start drawing the layout, it is useful to set the **D**esign **R**ule **D**riven (**DRD**) drawing. This will allow us to check the design rules in real time and easily draw the correct design.
- ✓ In the ***Layout Editor*** window, choose ***Options→ DRD Edit...***
- ✓ Set the ***DRD Mode*** (enable ***Enforce*** and ***Notify***) and the ***Hierarchy Depth*** to ***Current & Below,*** as in the figure below.



3.2. LAYOUT TUTORIAL

Step 1 - Generate the Layout Instances:

- ✓ We are finally ready to start drawing the layout. First, we will place all the devices from the schematic in our layout (schematic-driven layout).
- ✓ From the layout editor ***Command Toolbar***, choose ***Generate All From Source*** (see figure on the next page), and set it up to generate only the instances and to preserve the device correspondence (mapping between the schematic and layout). Do not forget to uncheck I/O Pins and PR Boundary options.



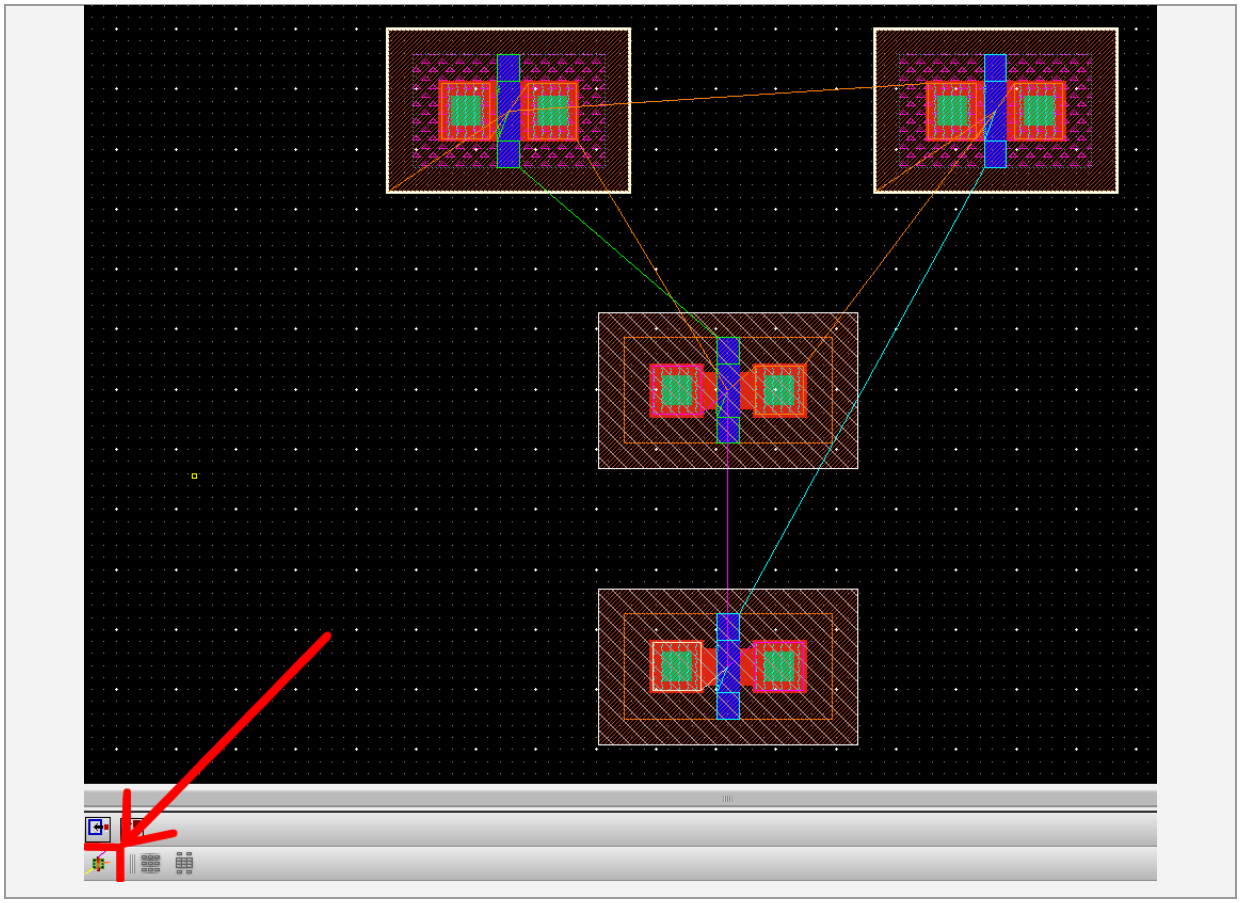
✓ Click OK.

Step 2 - Displaying the Net Connectivity:

✓ All the instances existing in the schematic will now be placed in the layout, as in the figure below (transistor sizes in your design will larger than this screenshot).

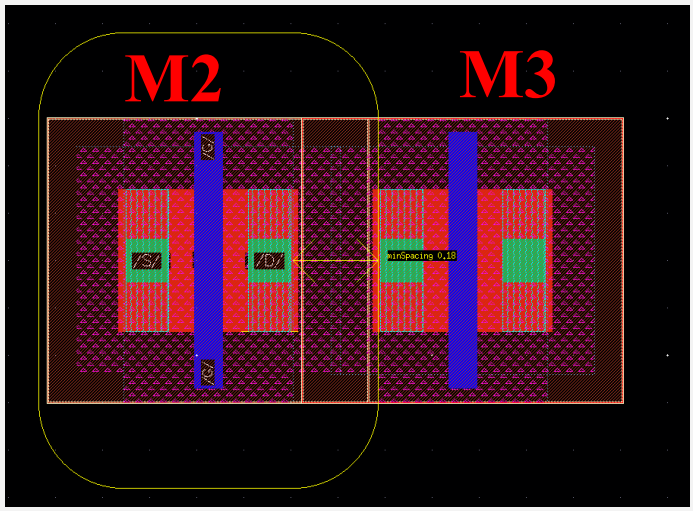
✓ You can now show the incomplete nets (nodes to be connected) by selecting all the transistors (hold the left mouse button to make the selection box), and clicking on the ***Show/Hide selected incomplete nets*** from the ***Command Toolbar*** (see figure on the next page).

✓ The incomplete nets will be depicted. You can use this feature from time to time, to check if your connections are correct. It can be quite useful. You can turn it off by using the same command button.

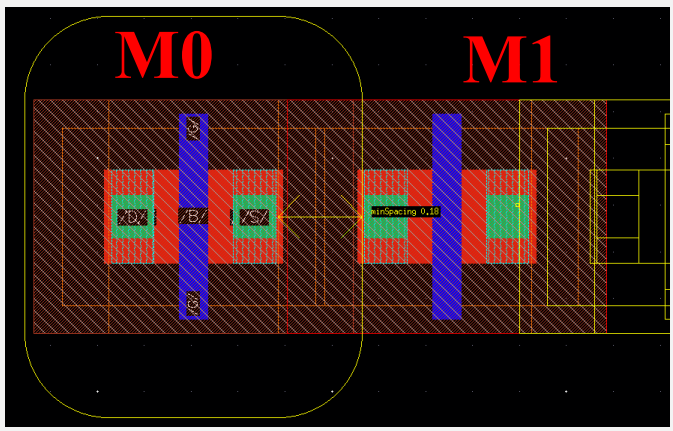


Step 3 - Moving and Aligning the Transistors:

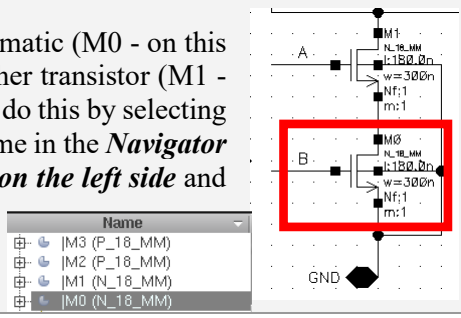
- ✓ Use ***m*** and select the device to be able to move and align the transistors.
- ✓ While you are moving the transistor, hold the right mouse button to make the zooming box. In the layout editor you will be able to zoom while drawing shapes or moving and editing instances. This can be quite useful.
- ✓ Put the PMOS transistors as close as possible to each other.



- ✓ Repeat the same for the NMOS transistors.

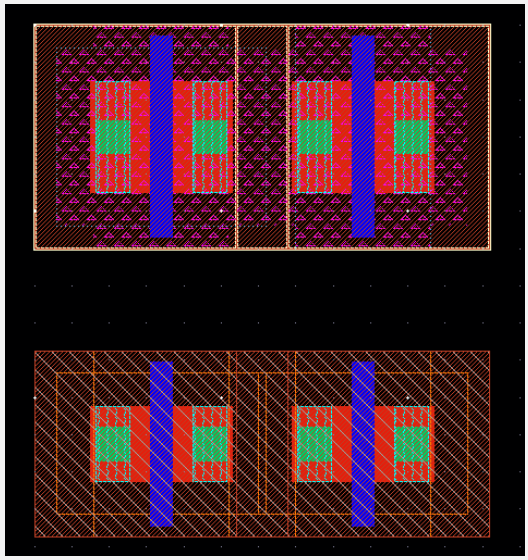


- ✓ You will see that the tool will make difficult to move transistors freely, or even not allow you to push the transistors closer together.
- ✓ Make sure that the bottom NMOS transistor in the schematic (M0 - on this figure), is on ***the left side in the layout***, and that the other transistor (M1 - on the figure) is on ***the right side in the layout***. You can do this by selecting the transistor in the layout and checking the instance name in the ***Navigator*** sub-window. Moreover, for PMOS part, M2 should be ***on the left side*** and M3 should be ***on the right side***.



Step 4 - Controlling DRD Enforce:

✓ So far, your layout should look as follows:



✓ In order to be able to put transistors where we want and connect them, we need to turn off the DRD Enforce:

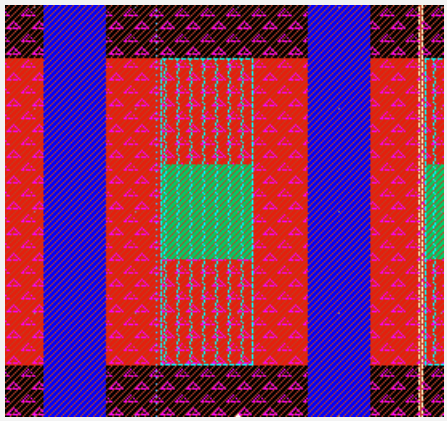
✓ DRD Enforce is now off, and we can move transistors more freely.

✓ We can turn the DRD Enforce back on if needed by using the same command button.

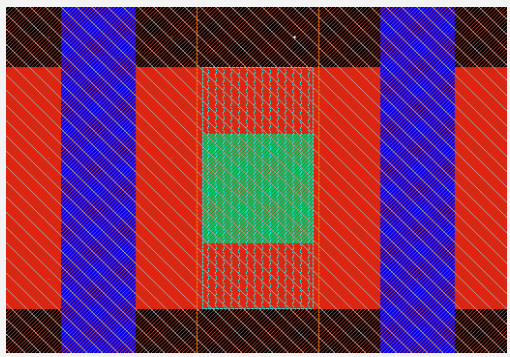
> *When drawing shapes, it can sometimes be very useful to turn the DRD Enforce off. Remember at this point how to do that.*

Step 5 - Connecting the Transistors:

✓ Use the move and zoom functions to connect the future drains of your PMOS transistors. To make sure that the contacts are matched (aligned) perfectly (see figure).



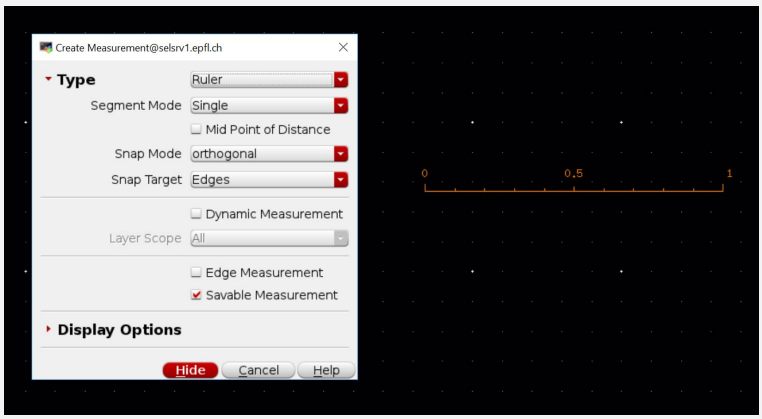
✓ Repeat the same for NMOS transistors to connect the drain of ***M0*** to the source of ***M1***:



As an alternative, you can use ***Quick Align*** (shortcut ***a***) for alignment of objects. Let us assume that you have two objects, like with the transistors in our case, not aligned. If you press ***a***, you are prompted to choose lines, mid-points or edges, depending on with respect to what you want to align. First, left-click the edge, line or mid-point of the object you want to align with other object (reference point), then left-click the other objects edge, line or mid-point (target point). The two shapes will be aligned. In addition, you can choose whether you want to align an object or a copy of it by pressing ***F3*** after pressing ***a***. You can try it with different shapes during or after the tutorial. It is a useful function to be aware of because ***Quick Align*** is easy to understand and can save time if used efficiently.

Step 6 - Using the Rulers:

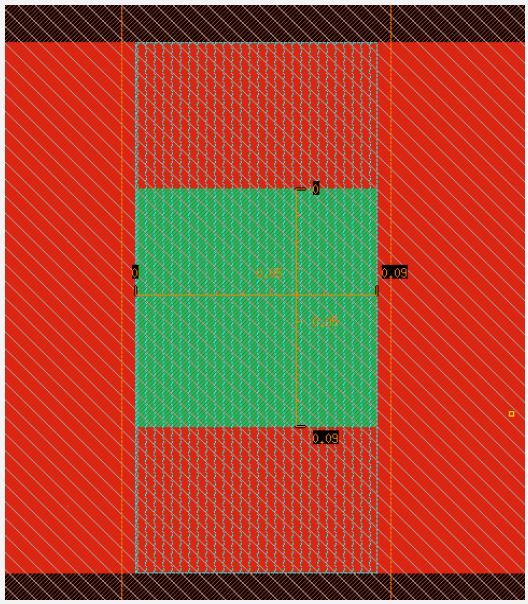
✓ Press ***k*** to draw the ruler. To draw, select starting and ending points of the ruler by left-click after pressing ***k***. To access more options about ruler, you can open ***Create Measurement*** dialog by pressing ***F3*** after pressing ***k***. In some cases, depending on Virtuoso settings, ***Create Measurement*** dialog may pop-up right after clicking ***k***, without pressing ***F3***.



✓ You can always remove the existing rulers by pressing ***Shift+k***.

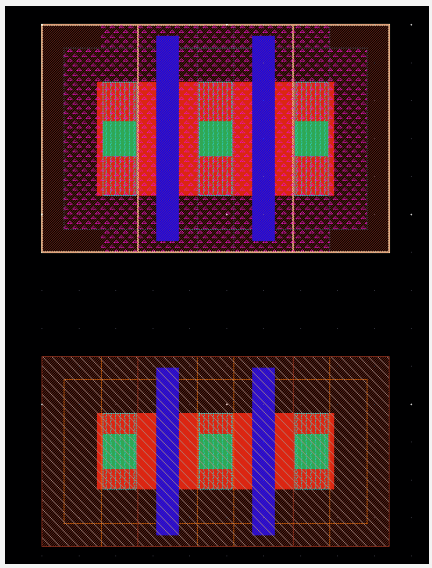
✓ Use the rulers to check the size of the **via** in the drain-source transistor connections to make sure they are matched (light green layer in the middle connection - see the figure). If the transistor drain and source (both PMOS and NMOS), are aligned perfectly, the size of the via should be exactly *0.09um x 0.09um* (see figure below).

*Note that if you consider a larger transistor, the tool will automatically create more vias for its drain and source. If you select the instance of your transistor PCELL and type **q**, you will be able to explore the transistor parameters and modify them. Just make sure that your layout and schematic match!*



Step 7 - Drawing the Poly-Silicon Wire:

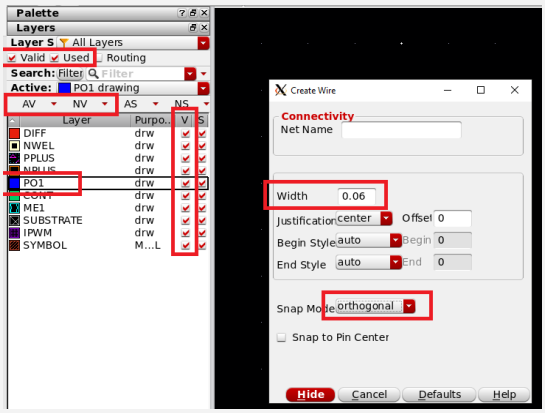
✓ So far, your layout should approximately look like in the following figure:



✓ To draw the poly-silicon wire, in the ***Layers*** sub-window, select the poly-silicon drawing layer (***PO1 drawing*** - see figure). Press ***p*** and use the left mouse button to start and end the wire. You can access ***Create Wire*** window if you press ***F3*** after pressing ***p***, just like the case with rulers (Step 6).

Note that you can tick the "used" button to only show layers that are currently used in your layout. Be careful check that this box is unticked before using a new layer! otherwise, you'll not be able to find it.

*Under these checkboxes, you will find the buttons **AV**, **NV**, **AS** and **NS**. By default, **AV** will make all the layers visible, **NV** will remove the visibility for all the layers except the one selected, **AS** will make all the layers selectable while **NS** will make all the layers non-selectable. Just keep your mouse on the button and wait for the popup window, to see their purpose. Use the **AV** and **NV** options to rapidly see some specific layers. You can manually add and remove visibility for some layers with the column of **V** tickboxes next to the layers.*

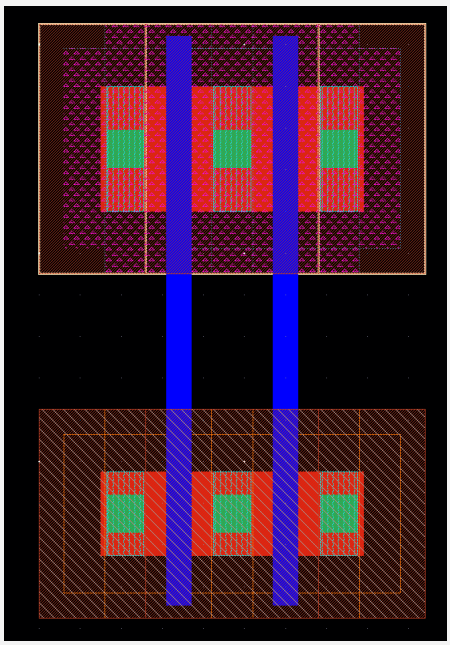


Note that you can make the wire to break under the 90 degree angle if you use the left mouse button multiple times. Try it out! You can also change the Snap Mode to diagonal to draw 45 degree lines and width of the wire (See above figure).

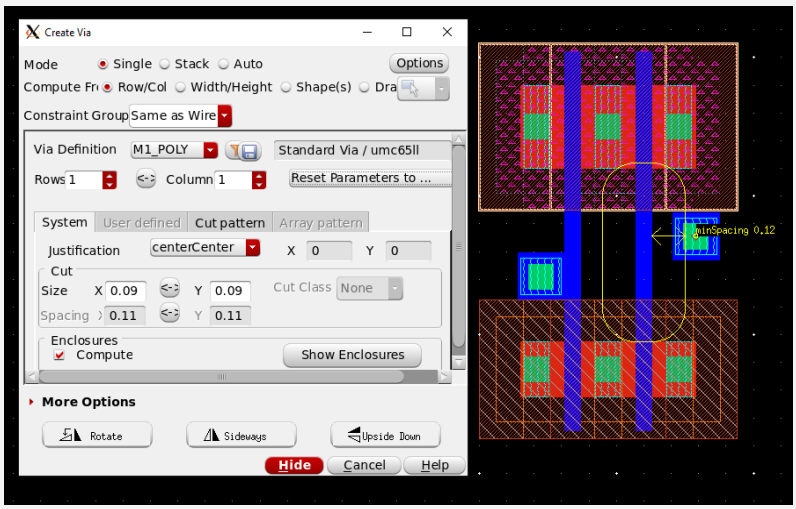
It is important to know that with the advances of integrated technologies, constraints on drawing are strongly increasing. In sub-30nm technologies PO layers can only have one orientation (i.e., no 90 degrees angles). 45 degrees angles in metals are usually prohibited for sub-90nm technologies. In this lab, you are using a 65nm technology node, keep your angles at 90 degrees.

Step 8 - Drawing Wires and Creating Metal 1 to Poly-Silicon Via:

- ✓ Note that while drawing wires, you can use the zoom buttons  (or the right click selection box) to zoom in and out. You may need to turn the DRD Enforce off to be able to draw freely (see *Step 4*). Turning the wheel of the mouse can also be used to zoom in and out.
- ✓ So far, your layout should approximately look like in the following figure:



- ✓ To draw the poly to metal 1 contact, press ***o*** and select **M1_POLY** as the ***Via Definition***. Press ***Hide***.

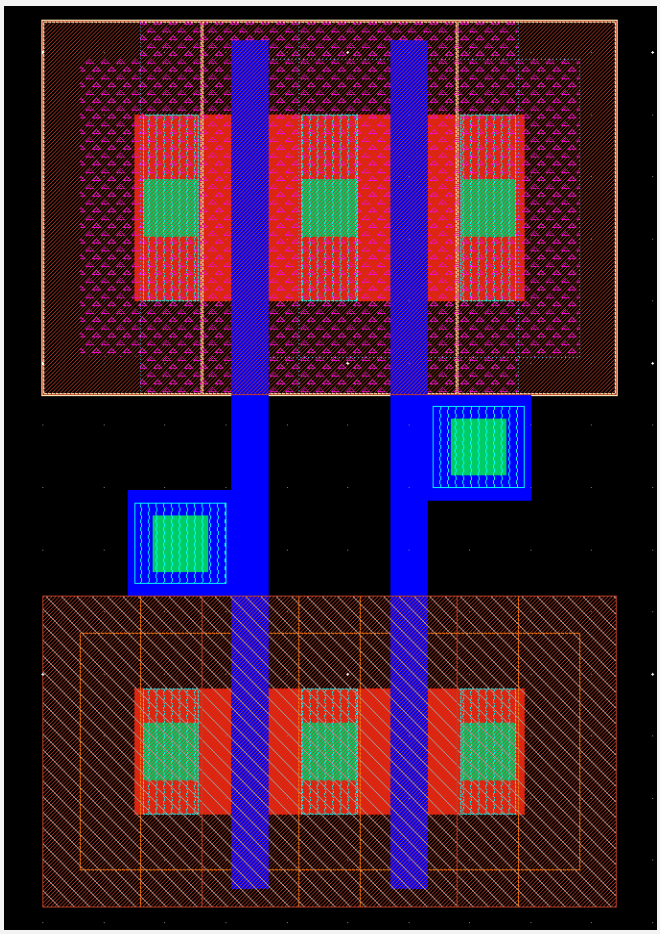


The two following tips require you to check carefully the Design Rule Manual (DRM) document:

- ***Optional note for optimizing you design 1:*** *you can click on "reset parameters to…": and select "minimum rules" to have a via definition that corresponds to the minimum rules. This via will not pass DRC checks so be careful when using this option.*
- ***Optional note for optimizing you design 2:*** *you can draw a via yourself if you know which layers are used, using the key* ***R****.*

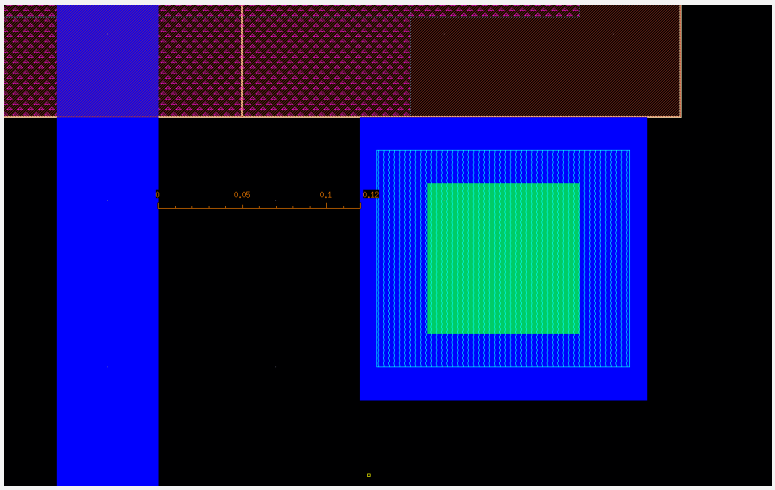
Step 9 - Placing the Poly-Silicon to Metal 1 Contact (Via):

- ✓ Place the contacts trying not to break the design rules:
- ✓ So far, your layout should approximately look like in the following figure:

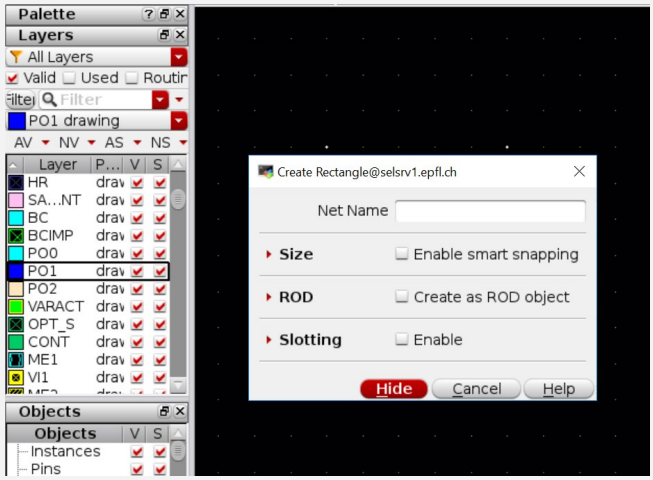


Step 10 - Drawing Rectangle:

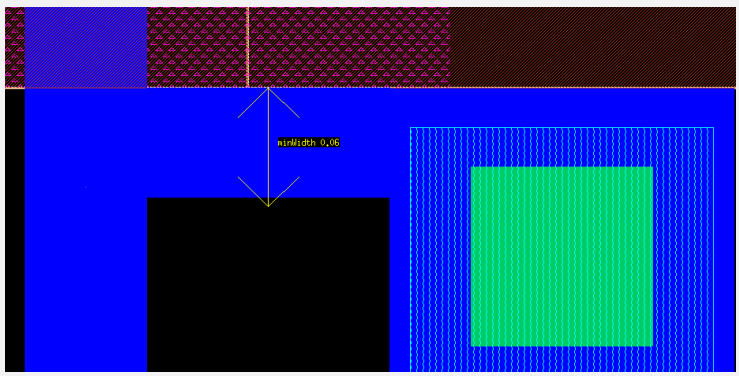
✓ The current organization of you layout is good already, but you could want to have your gate contact further away from the polysilicon gate. Select the contact and press **m**. Bring the contact 120nm away from the vertical PO1 line (you can use the ruler– press **k** to use it).



✓ Make sure the ***PO1 drawing*** layer is still selected and press ***r*** to enter the rectangle mode. ***Create Rectangle*** option can be accessed with ***F3***.

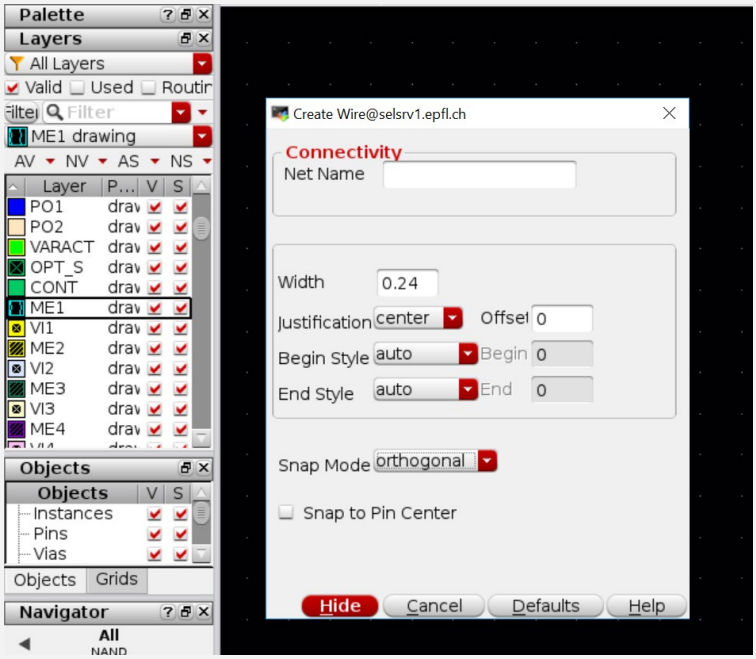


✓ Draw the rectangle by using the left mouse button. Make sure not to break the design rules: Here the minimum width for PO1 is 0.06um (60nm).



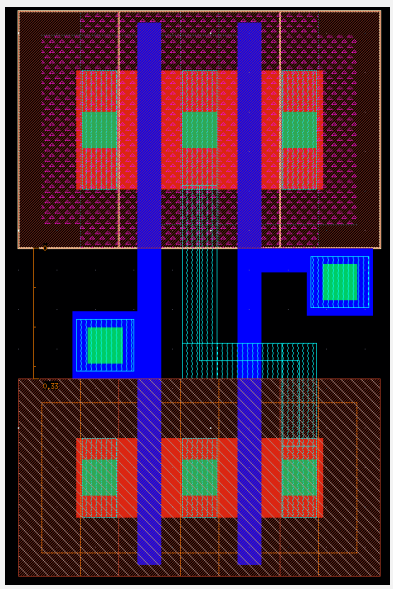
Step 11 - Creating Metal 1 Wire:

✓ In the ***Layers*** sub-window select ***ME1 drawing*** layer. Start drawing the ***ME1*** wire same as for the poly-silicon (you may need to turn the DRD Enforce off - see *Step 4*). To see the options (***Create Wire*** window, see figure), you can press ***F3***.



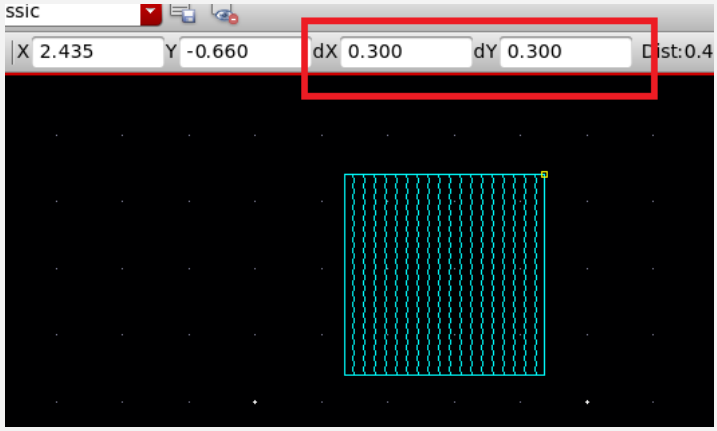
✓ So far, your layout should approximately look like in the following figure:

ℹ *To help you with the sizing, in this view, we use a 0.33um distance between the transistors symbol layers.*



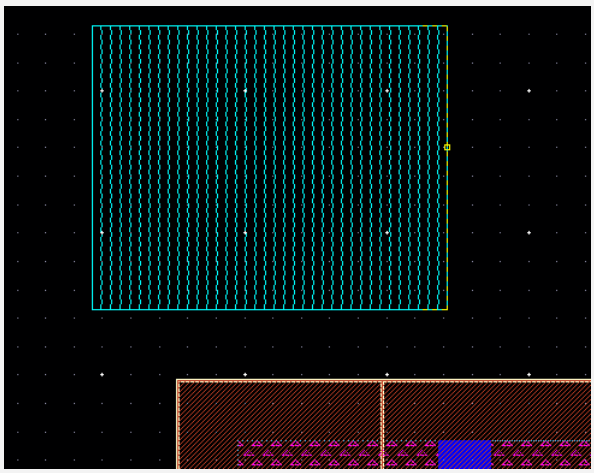
Step 12 - Stretching/Skewing the Shapes:

✓ In the ***Layers*** sub-window make sure ***ME1 drawing*** layer is selected. Press ***r*** and try to draw a rectangle of *0.3um x 0.3um* (see figure).



✓ Press F4 and after that put the cursor close to the rectangle edge. You will be able to select the edge. Select the edge and try to drag it left and right. Alternatively, you can use ***s*** for stretching.

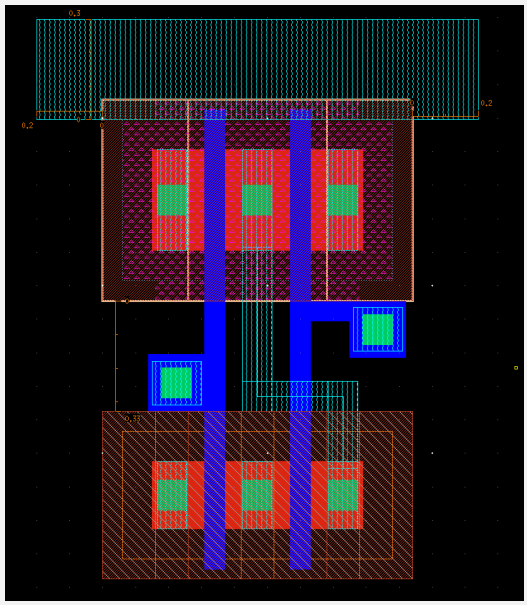
Note that F4 will toggle this button from  *to*  *. You can alternatively click on it.*



Step 13 - Creating VDD and GND Supply Lines:

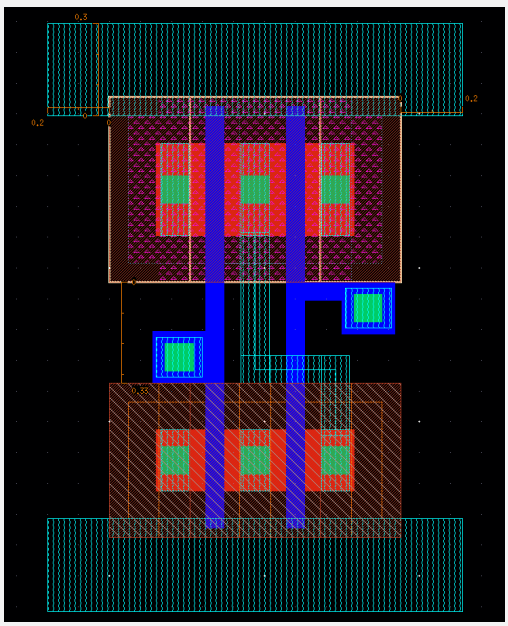
✓ Stretch the rectangle and place it as in the figure below:

You can place it at the minimum distance from the transistors drain/source metals (i.e. 90nm)



✓ Press F4 once again and you will be back in the normal selection mode. Click on the rectangle and press ***c***. Copy the rectangle (use left mouse button) and place it as in the figure below.

✓ Use the distances shown in the screenshot. Make the rails 300nm high, and 200nm larger than the transistors symbol layers.

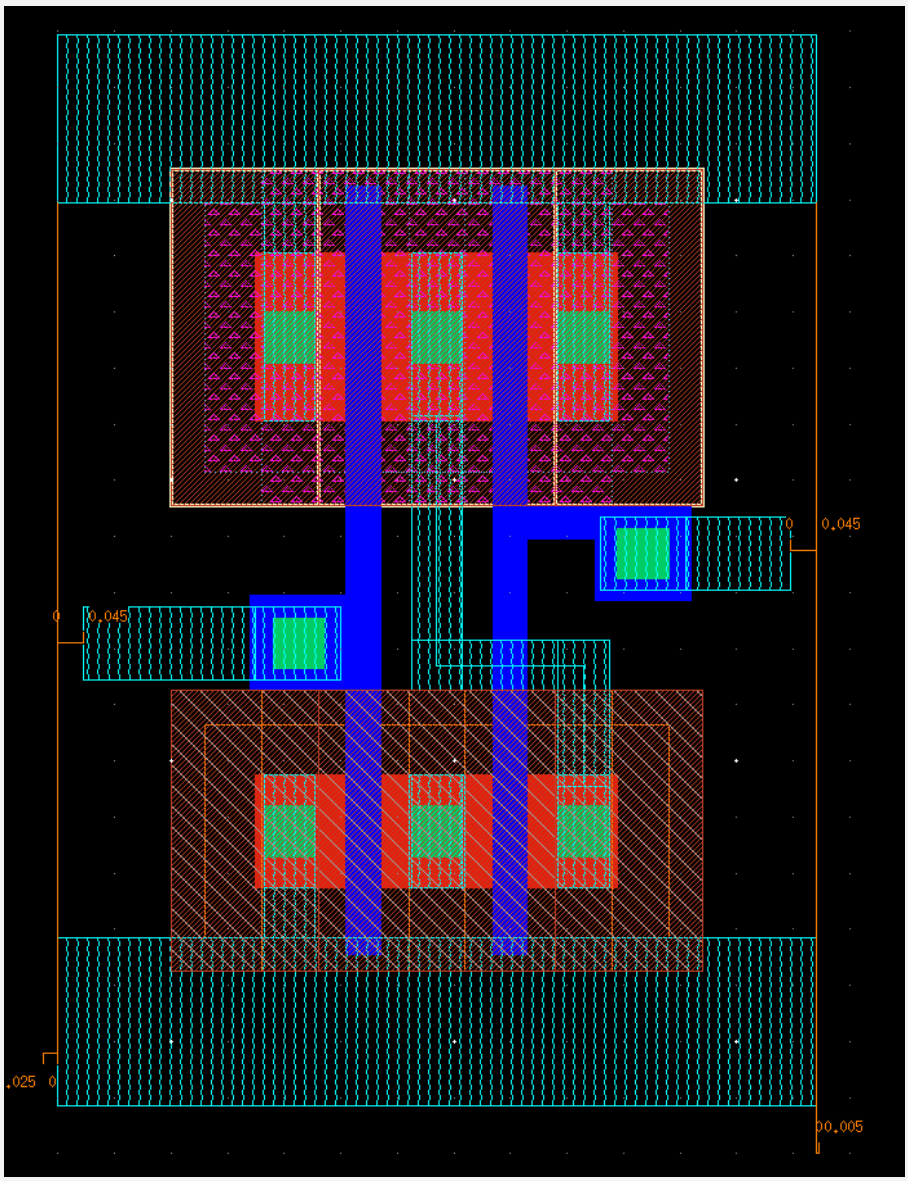


*Notice that as for the **move** function, when copying you need to press the left-mouse button to define the reference point and then press it once again to place the shape. Reference point can be anywhere on the drawing surface, but it is usually good to keep it close to the shape. Especially if you need to zoom later, reference point should be defined carefully.*

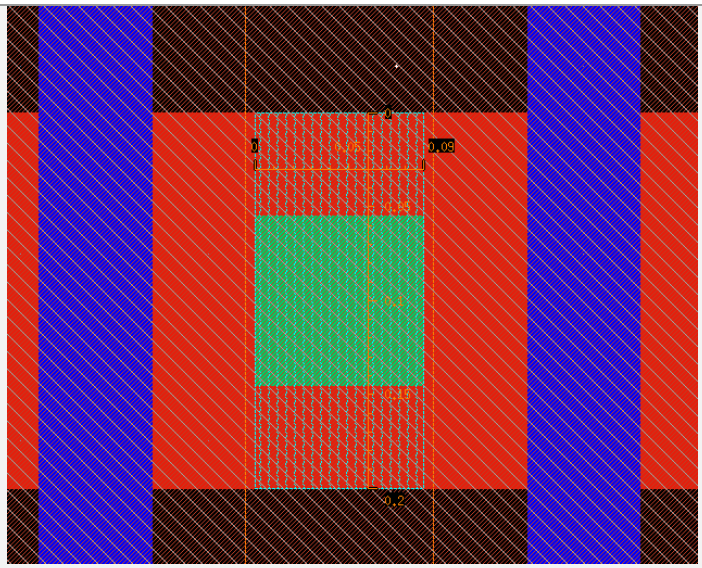
Step 14 - Finishing the Metal Connections:

✓ Complete the remaining metal connections as in the figure:

A good practice when designing logic cells is to leave enough space on the boundaries of the cells, to be able to abut them to an identical cell. For e.g., if ME1 minimum spacing is 90nm, leave 45nm before the cell boundary. Use the ruler for it.

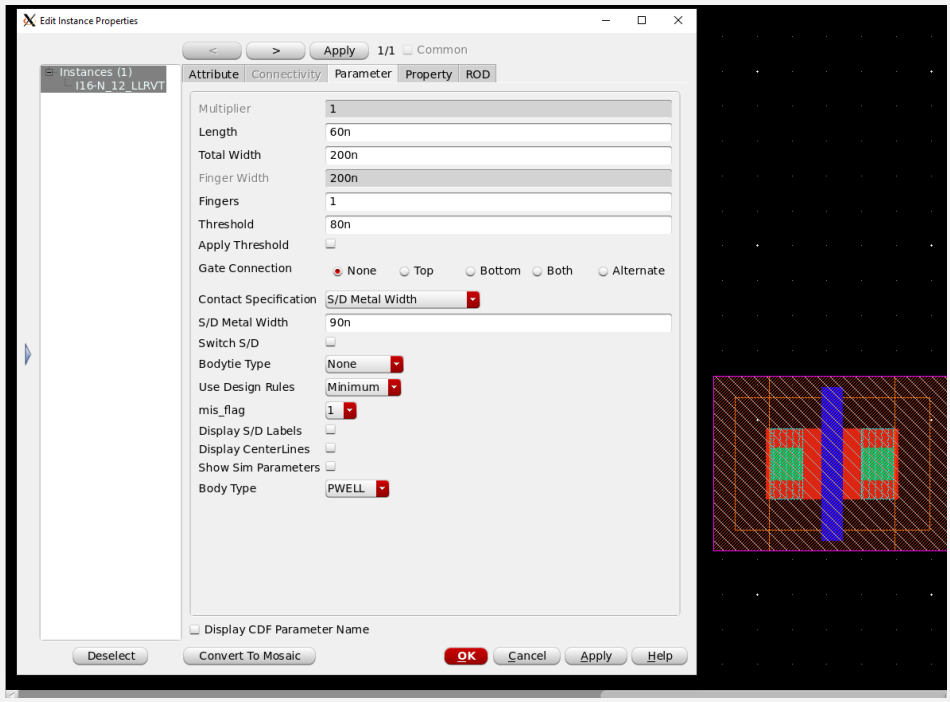


✓ This cell is good. As you will see later, during the Design Rule Check (DRC) phase, your design needs to respect some rules, such as minimum area, minimum distances etc. The minimum area for ME1 layer is 0.042squm. In between your NMOS, you currently have a piece of metal that you are not using. Measure it and calculate its area.



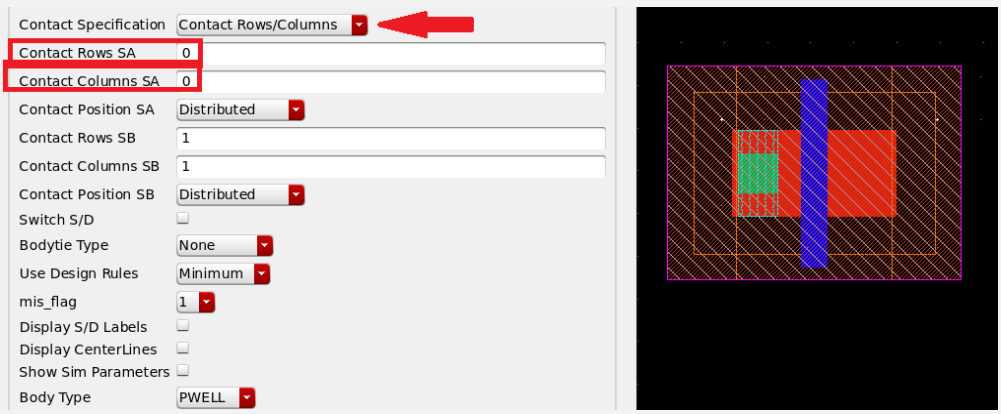
Step 15 - tune device parameters:

✓ Select one of the two transistors and copy it (with **c**) next to your design. Select the newly created transistor and press **q**. Select the Parameter panel. The following window should appear.

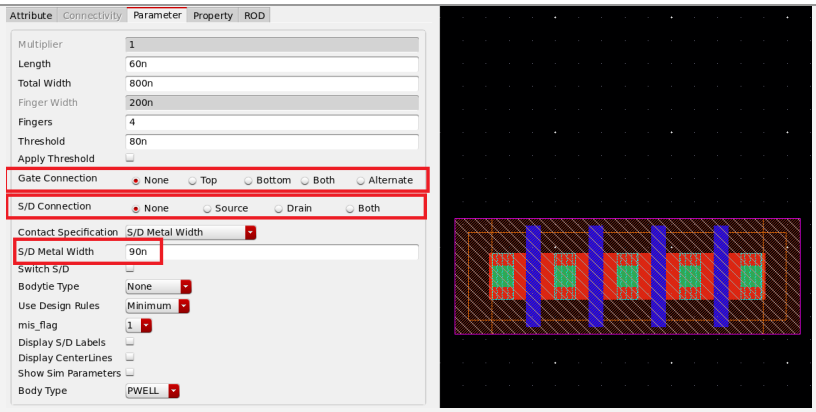


✓ Click on **contact specification** and select "**Contact rows/Columns**". This allows you to select the type of contacts you want in the source and drain of your transistor. Put **0** on "**Contact Rows SA**" and "**Contact Column SA**". Press **Apply**. This option disabled the metal contacts on one side of the transistor. Conversely SB-related options concern the other terminal.

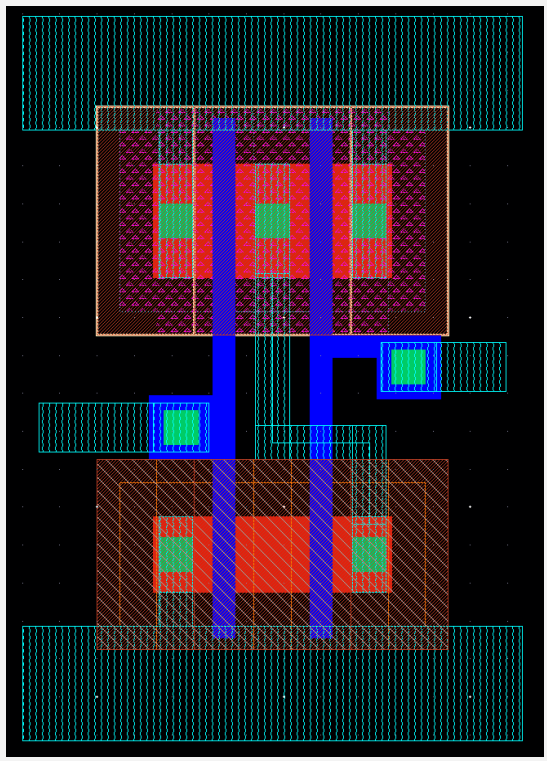
✓ Note that there is a minimum width accepted in the design kit for these automated options. If you need to do it for smaller transistors, you will need to flatten it. See later.



✓ Put back the terminal on this transistor, select back "S/D Metal Width", and make it multi-finger. Put the total width to 800nm and the finger parameter to 4. Press Apply, you should now have created the following transistor.

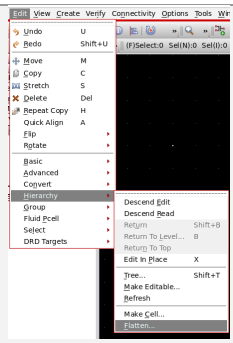


- ✓ Use the **Gate connection** option and try different configurations with "**Top**", "**Bottom**", "**Both**" and "**Alternate**"
- ✓ Do the same with the **S/D connection** option. What do you observe?
- ✓ Change the **S/D Metal wi**dth parameter to **150nm**. What do you observe? try to force it to **50nm**, what happens?
- ✓ Suppress the transistor you just created and come back to your design.
- ✓ Tune the **contact specification** parameters to remove the middle terminal in the pull-down network of your NAND gate as follows.



- ✓ *Note that almost all of the devices provided in PDKs have such kind of options. At the end of the day, all of these options are things that you could do by hand, i.e., drawing transistors using the* ***r*** *drawing tool.*
- ✓ *You could also modify one transistor PCELL by using the flatten option.* ***Be careful with this option! flattening is an irreversible action!***

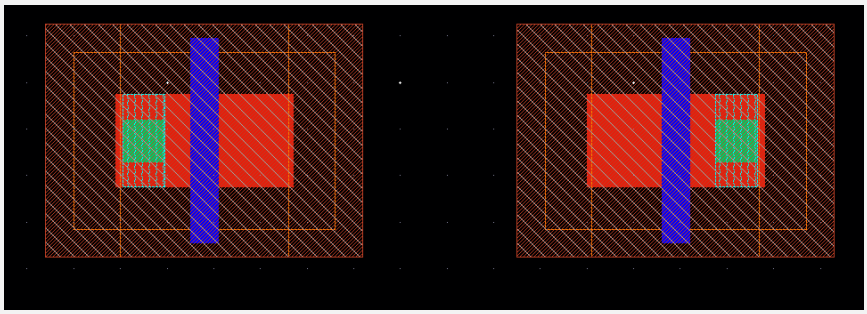
- ✓ **Optional Step:** Do as at the beginning of step 15, copy one of your transistors (with **c**) or instantiate a new one (with **i**). Select the transistor and select Edit>Hierarchy>Flatten.
- ✓ Select displayed levels, and preserve pin geometry. Click OK.

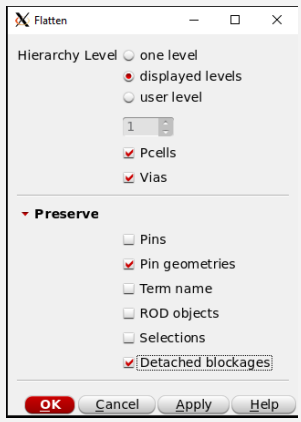


This Option removed all the hierarchy levels associated to this cell and transformed it in shapes that you could create with the **r** drawing tool.

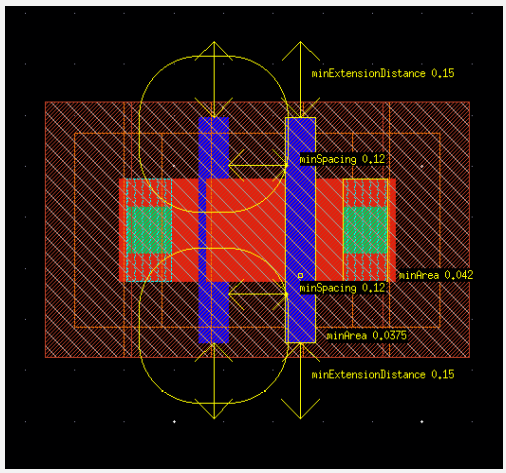
Important! *Flattening can be very useful for the Full-Custom design, since it provides the possibility to work in layer level. However,* ***you should be very careful when you use flattening****. First, if you are flattening the primitive cells like we did here, you need to be aware of all the layers required for the proper functioning of the transistor. Otherwise, if you make a mistake, unexpected errors and/or issues may occur. This is why, in practice, flattening is reserved for more experienced designers. Second,* ***flattening destroys the hierarchy of your design.*** *This can be a major issue for larger designs, since the loss of hierarchy makes the layout debugging practically impossible. Therefore, as a general rule:* ***you should never flatten the higher level cells.***

- ✓ Try to select layer by layer (left click) and delete (press ***delete***) the unnecessary layers exactly as in the figure below (if you make a mistake, press ***u*** - undo):



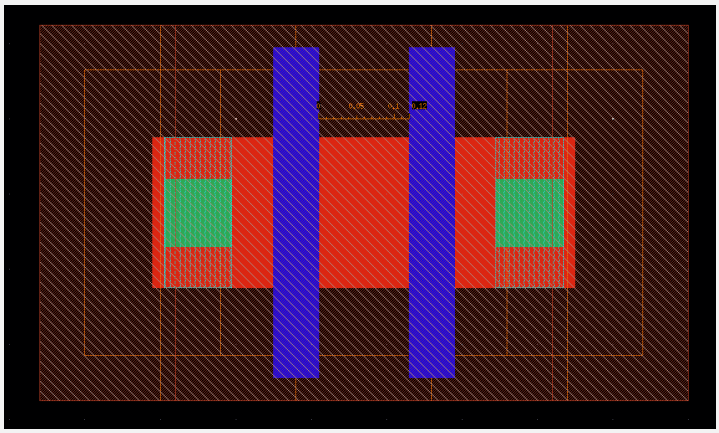


- ✓ Select one of the transistor and try to merge the two transistors. How close can you put them before the DRD notifier complains ?



- ✓ The minimum distance between the gates is *120nm*. Compare it to the previous implementation using the contact specification option. This approach seems to improve the density. Though,

your PO1 lines will not be aligned anymore. This will improve the density of your layout at the price of a bit more complexity in the routing.



Important comment : This industrial tool provides a lot of functions which can be extremely useful for the designer. Though, these are not always properly supported by the design kits.

As a reminder, the software is developed by a software company (Cadence), and the design kit is integrated inside the software by the foundry (here, UMC). The foundry does not always update the kit to the latest changes, and so, some features are sometimes not supported, partially supported, or simply bugged. **This is part of your engineer job, to know what is, and what is not supported. And… deal with it.**

The question is… how and where to find that. inside PDK_HOME/doc/ you will find a pdf called Release_Note_UMK65FDKLLC00000OA_B11_PB.pdf

From page 27, it lists the known issues and solutions to the problems of this design kit. Some of them are irrelevant for you, some are critical. Specifically :

MOS dog-bone does not support design variables.

Issue: It is not recommended to use design variable when MOS width is less than 0.12um (Mos dog-bone). The parameters of MOS dog-bone do not support design variables in this version.

(Responsible: IPDS/DF/FDK)

This one answers why you get different results when using variables and small devices. But also why the abutment and modification of transistors smaller than 120nm does not properly work.

IC615 Virtuoso off-grid issues of Flip vertical/ horizontal

In IC615, add 2 more options in layout editor options

In the Editor control (left side of the form), these options are

1. Rotate Around Combined Center

2. Flip Around Combined Center

These 2 option might cause off-grid issue if the option is checked. Unfortunately it was set to on by default in IC615.

Please turn these options to be off and see if the off grid issue still exists.

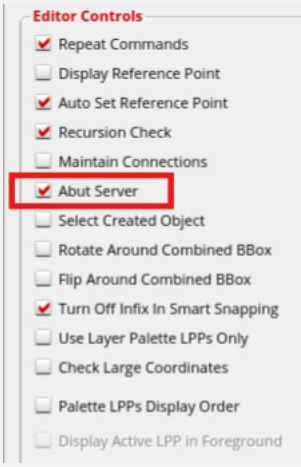
Or you can set in .cdsenv

layout flipAroundCombinedCenter boolean nil

layout rotateAroundCombinedCenter boolean nil

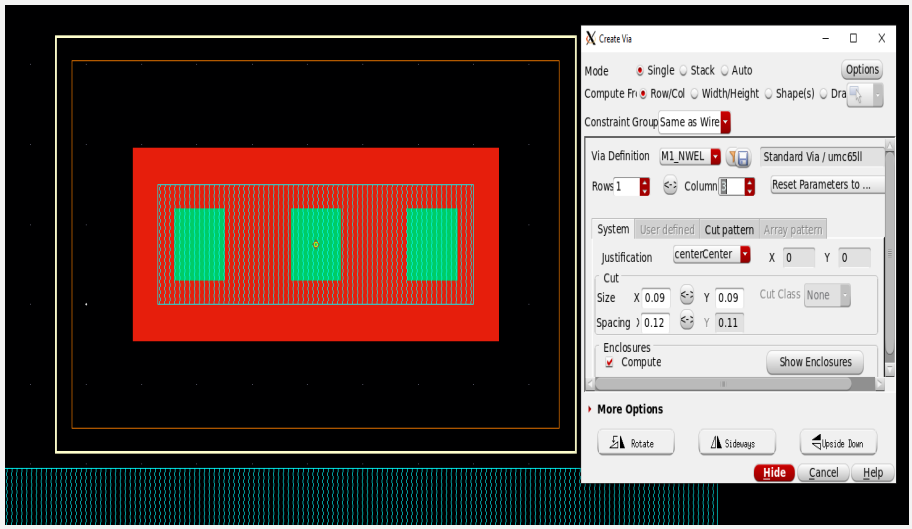
Page 32. This issue is critical. And we fixed it for you. You can check the content of the .cdsenv file in your working directory.

Cadence virtuoso features an automatic merging function for transistors, which is not fully supported in this PDK. You can enable it by ticking the "Abut Server" function in the Option>Editor (Shift+e) dropdown menu of the layout editor. **Be careful with this option it tends to not properly abut transistors if smaller than 120nm.**

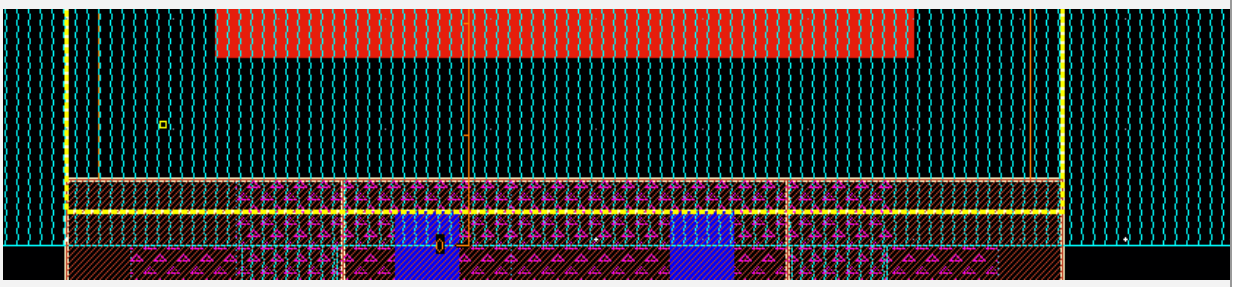


Step 16 - Creating the N-Well Contact:

- ✓ If you remember your classes about MOS transistors, transistors are actually 4-terminal devices. Their substrate needs to be biased. Ideally, all of your NMOS substrate should be biased to GND while all of your PMOS substrates should be biased to VDD. To do so, you will need to connect the N-WELL of your PMOS to VDD, and the PSUB of your NMOS to GND. Let's start with the N-WELL.
- ✓ Now, we need to create a contact for the N-Well. Press ***o*** and select M1_NWEL as the ***Via Definition***, and set the number of columns to 3.

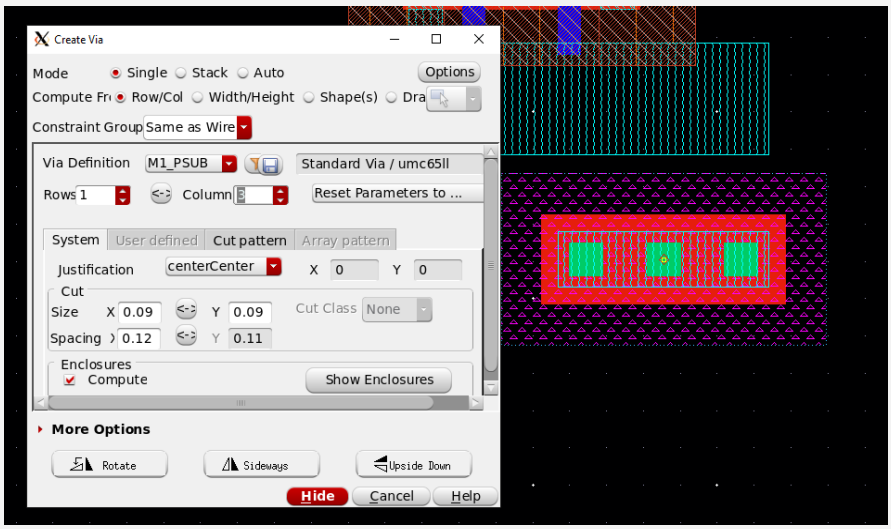


- ✓ Center the N-Well contact (horizontally it should be the same width than the PMOS cells NWEL) and place it (vertically) so that the PPLUS layer (purple) perfectly touches the NPLUS (Orange).

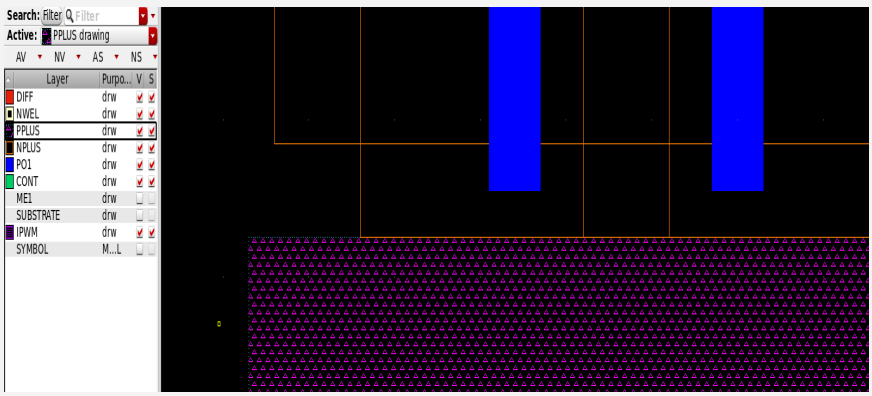


Step 16 - Creating the Substrate Contact:

✓ Now, we need to create a contact for the substrate. Press ***o*** and select M1_PSUB as the ***Via Definition***, and set the number of columns to 3.



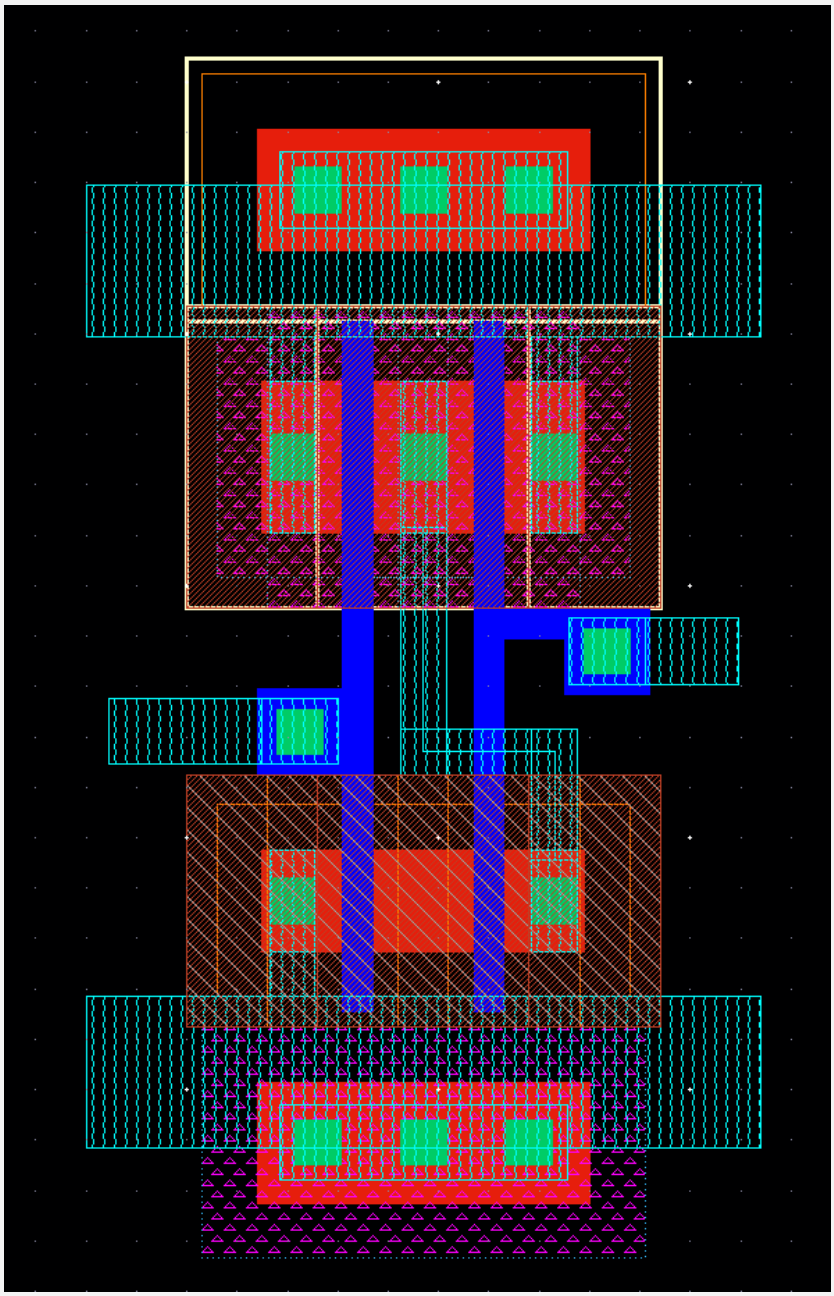
✓ Center the substrate contact (horizontally) and place it (vertically) so that PPLUS layer (purple) touches the NPLUS layer (orange).



QUESTION 3-1: Let's say your polysilicon are not aligned between the PMOS and NMOS transistors. How would you connect them ? What do you need to be careful about ? feel free to describe your approach with a drawing.

Step 17 - Verify your Layout

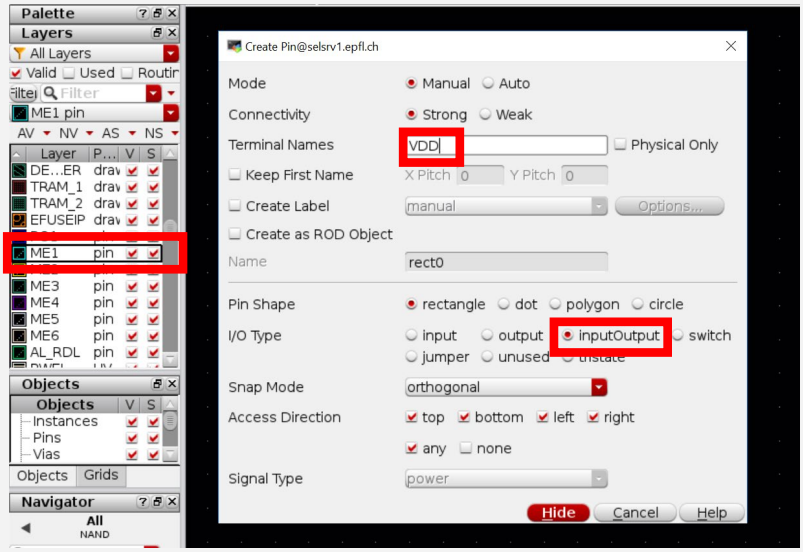
✓ So far your layout should approximately look as follows:



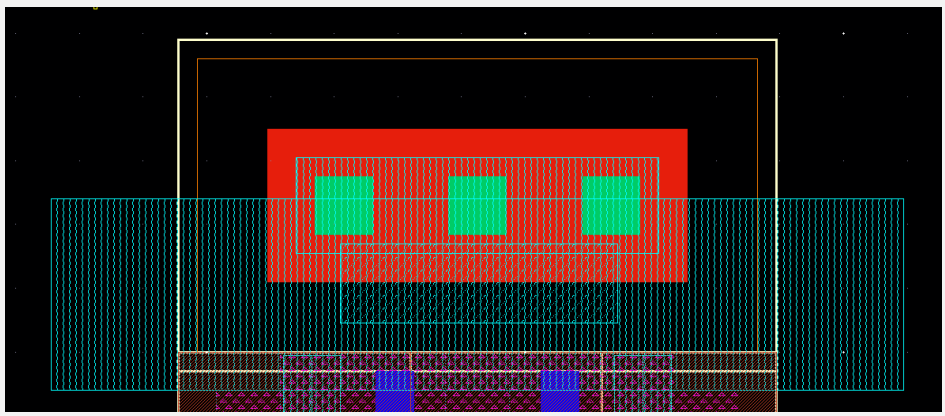
✓ Check your layout and make sure that everything is in place.

Step 18 - Creating the Pins:

✓ In the Layers sub-window choose ***ME1 pin*** layer, and press ***Ctrl+p***. (Type ***ME1*** into ***Filter*** for easy access.) If not defined yet, define the **Signal Type** as **Power**.

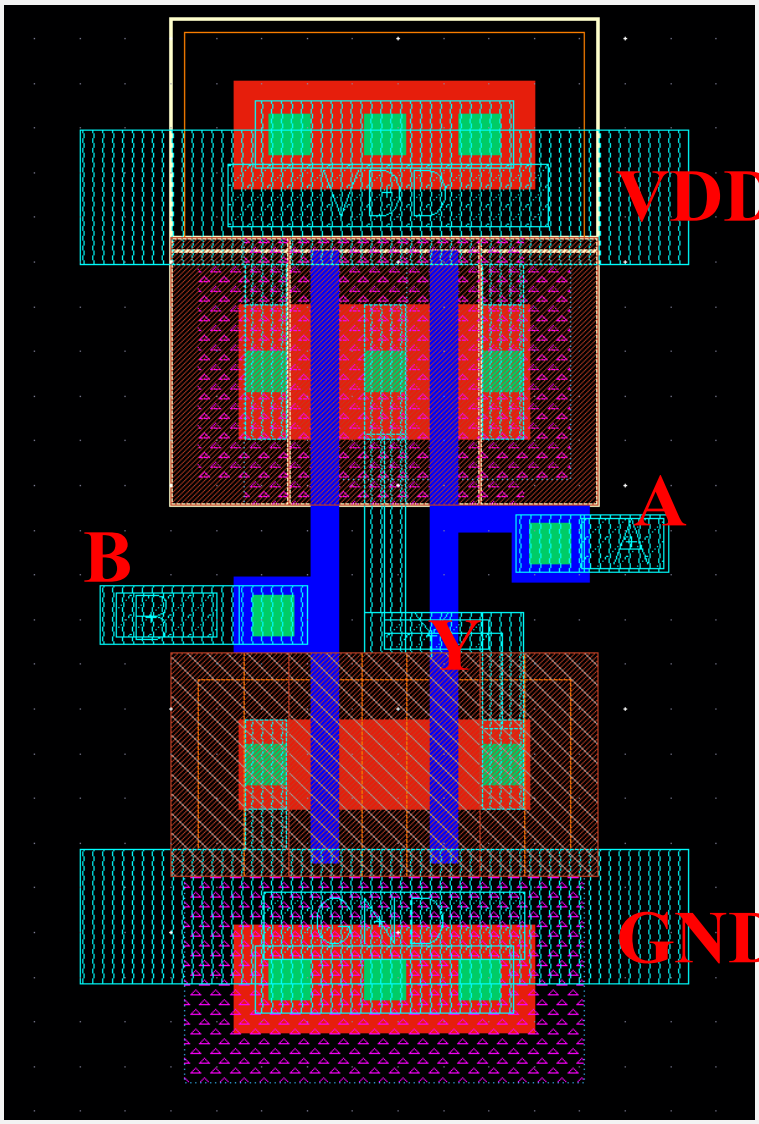


✓ Set the pin name to ***VDD*** and the ***I/O Type*** to ***inputOutput***. Press Hide and draw the small rectangle on the top (VDD) metal line as in the figure:



Step 19 - Creating the Pins:

✓ Repeat the procedure for all the remaining pins. Remember to define ***GND*** as ***inputOutput***, ***A*** and ***B*** as ***input***, and ***Y*** as ***output***. Draw the rectangles at the correct places.

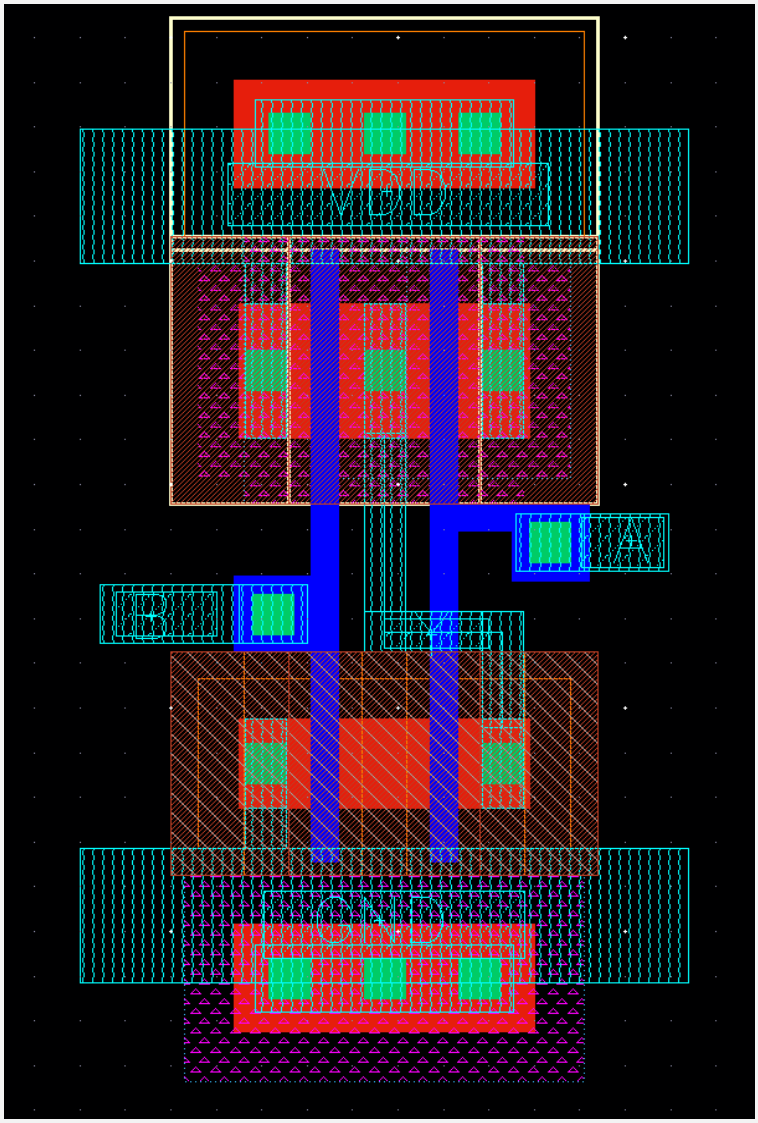


Step 20 - Creating the Labels:

✓ Now we will define the labels for our pins. In the layers sub-window choose ***M1_CAD text*** layer. Press ***l*** and enter ***VDD*** for the label. Put font height to be 0.1 and place the center of the label (+ sign) **<u>on top of the *VDD* pin rectangle</u>**. Repeat the procedure for all the remaining pins.

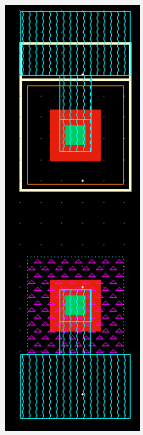
Step 21 - Verify the Layout:

✓ So far, your layout should look like this:

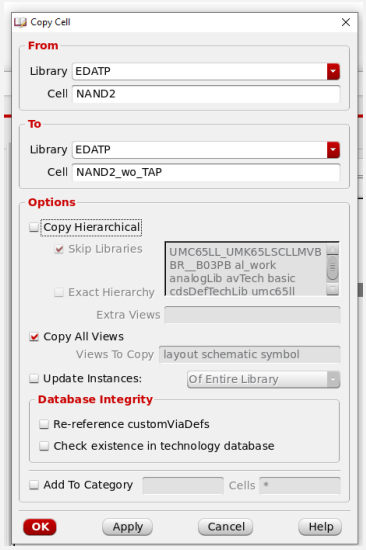


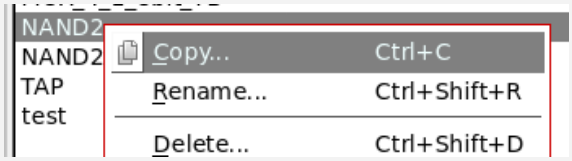
Step 22 – Optimize your substrate biasing with TAP Cells:

- ✓ In digital design, cells are abutted in rows along the VDD and GND rails. Adding substrate biasing on top may not be the best way to operate, as it will introduce irregularities that will make the design harder (i.e., all the cells are not always aligned vertically, nor make the same width, and thereby this will induce DRC errors). In this context, using TAP cells is a good alternative.
- ✓ A TAP cell is a substrate/Nwell biasing cell that has the same height than a regular cell, but instead of containing transistors, it contains a connection between VDD and the N-Well and GND and the substrate. TAP cells are placed regularly along the rows to bias the substrate and wells.
- ✓ Create a new layout cell called TAP. Make it the same height that the NAND2 cell you created before. Make the NWELL the same height as well. as shown in the following figure :



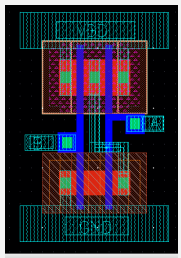
- ✓ The top contact is the same than in step 16 while the bottom one step 17.
- ✓ Save it and close it.
- ✓ Copy your NAND2 cell in a new cell called NAND2_wo_TAP (right click>copy).





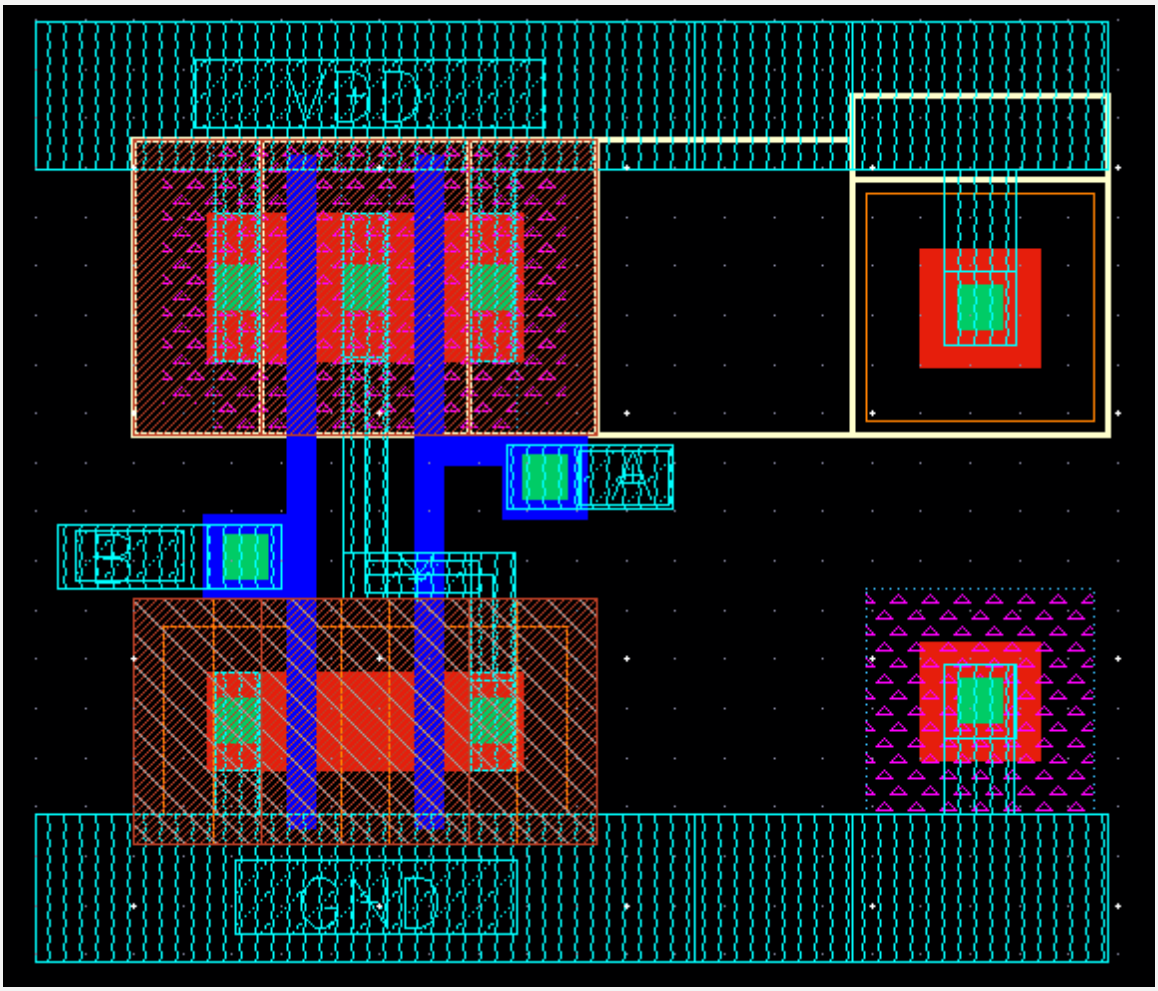
To copy a cell and include its hierarchy from one library to another one, use the hierarchical copy.

- ✓ In the newly created NAND2_wo_TAP cell, suppress the substrate connections on the top and bottom of the cells.



- ✓ Copy the NAND2_wo_TAP cell into a new cell called NAND2_TAP
- ✓ In the schematic import your TAP cell in the layout using the **i** key, and place it next to your NAND2 layout. Connect the NWELL and metals (both VDD and GND) with rectangles (**r** key).

✓ Your final cell should look like this:



✓ How does this compare to the previous cell you designed (with the biasing well connections on the top and bottom)?

ℹ Note that here, we did not optimize the spacing between the TAP cell and the logic gate. Also, you do not need one TAP cell per gate, but one TAP cell per row of gates. The DRC will usually tell you if you miss TAP cells by complaining about a floating substrate for e.g.

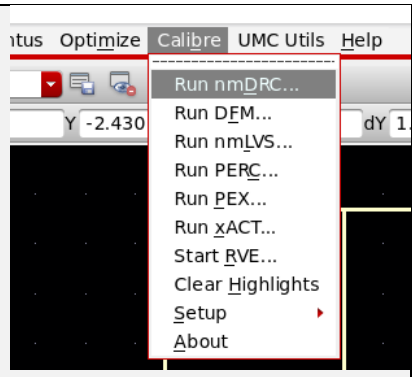
QUESTION 3-2: Why do you need to add TAP cells ? list the different ways of handling substrate biasing which we just explored.

3.3. DESIGN RULE CHECK

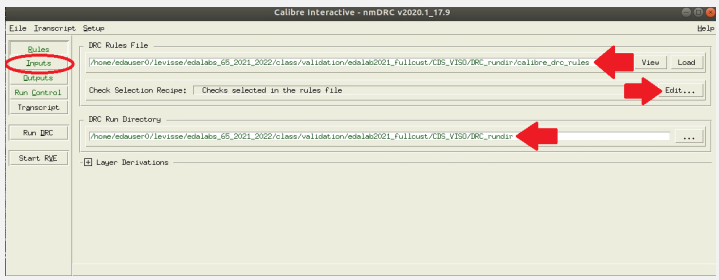
After you finish drawing the layout, it is necessary to perform the final **D**esign **R**ules **C**heck (**DRC**).

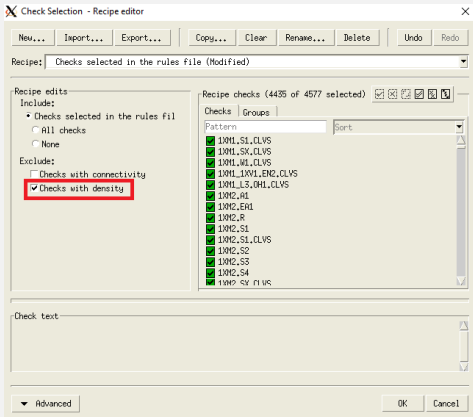
In this lab, we propose you to use a tool developed by a company called Mentor Graphics (acquired by Siemens in 2017). This tool, Calibre, is commonly used in the industry for DRC, LVS and parasitic Extraction (PEX) in technologies down to sub-20nm. Note that the steps described here are specific to this tool. However, the general methodology will be equivalent with other tools. Other compagnies such as Cadence and Synopsys also propose equivalent tools (e.g. Cadence Assura, Cadence PVS, or Synopsys StarRC).

- ✓ Save your layout ***(File> Save)***.
- ✓ Use ***Calibre >Run nmDRC***… from the Layout Editor Menu.
- ✓ Select the **Rules** panel.
- ✓ Define the DRC rule file and the DRC run directory as in the following screenshot (you need to import **calibre_drc_rules** – these rules are just a renamed version of the rules called *G-DF-LOGIC_MIXED_MODE65N-1P10M2T2F-LL-Calibre-drc-1.14-P1*). You can find all the different rules in ./RuleDecks/Calibre/DRC/



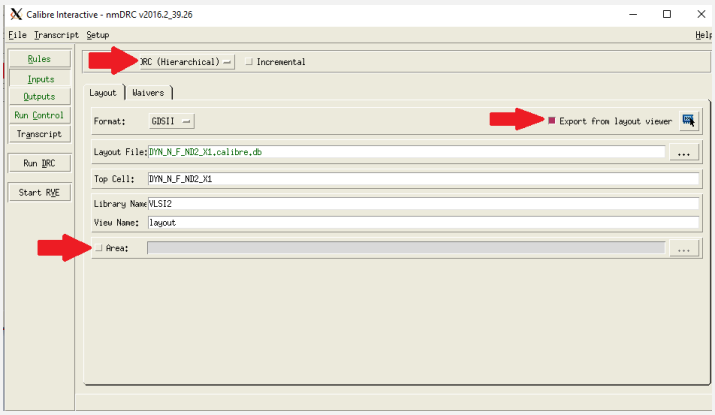
- ✓ Make sure that the "**check selection recipe**" is defined as "**checks selected in the rule file**" and click on "**Edit…**"
- ✓ Check the box corresponding to the exclusion of density checks. Click OK.

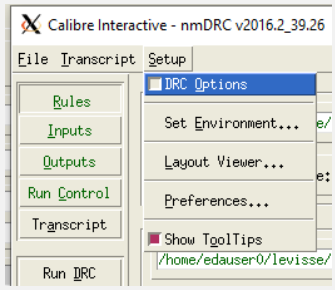




- *We disabled density checks to ease the design during this lab. In real life, when fabricating a circuit, each layer must have a minimum density to ensure that the fabrication process can be performed. One would usually add what is commonly called fillers to make sure that a minimum density is achieved in the design.*
- ✓ In the Input panel, make sure that you are extracting the layout from the layout viewer in the GDSII format.

- ✓ More options are available by clicking on **Setup>DRC Options**. There is no need to change any options in it for this Lab.





Other relevant points of interest:

- ✓ On the top, on the scroll down menu, the default option is "DRC (hierarchical)". This option maintains the hierarchy in the design rule checks and makes the evaluation faster. In some cases, if the errors are referring to lower hierarchy levels, you can put it to "DRC (flat)". The run will take more time but will give you the position of the DRC problems regardless of their position in the layout hierarchy.
- ✓ Ticking the "**area**" button will allow you to select the area in which you want your DRC to be run (by clicking on the "…" and then making a square around your block of interest). This can enable you to avoid some errors related to the edged of your contribution, or just focus in one area of interest inside your layout.
- ℹ *As a designer, you should always, by yourself, do a hierarchical DRC check. i.e., before integrating the layout of a cell in a higher hierarchy level, you should always make sure that your cell is "DRC clean". Consider the following statement :* ***"Everything that has not been tested is not working"***

- ✓ Once this is done, click on **Run DRC**.

After you finish drawing the layout, it is necessary to perform the final **D**esign **R**ules **C**heck (**DRC**).

Notice at this point that the DRC ***Run Directory*** is set to ***./DRC_rundir*** within your project directory. This is the location where your DRC-run data will be stored. Depending on the design-kit, similar as for the simulation directory, the default location can be different. If needed, it is possible to store DRC data in the different place by specifying another path manually. However, be careful about where you store different data. **Never use the same directory for storing simulation data and DRC data (same is the case for LVS data in the next session). As a rule, you should always use a dedicated and a different directory for the different types of data.**

- ℹ *Note here that for this technology, and this lab, we provide you with the rule files (in the corresponding DRC_rundir, LVS_rundir and PEX_rundir folders). You can check the different subfolders in ./RuleDecks/Calibre/ . They contain the rule files. Most of the PDKs would usually just request you to provide the rule files and run the DRC from a working DRC folder. The version of the UMC65 PDK you use in this lab will require you to have the rule files, or at least symbolic links to the different required files in your Run Directory.*

```
[edauser0@selsrv1 CDS_VISO]$ ls RuleDecks/Calibre/
calview.cellmap  DRC  LPE  LVS
[edauser0@selsrv1 CDS_VISO]$ ls RuleDecks/Calibre/DRC/
65nm_layers_v2.6_P2.cal                                 G-DF-LOGIC_MIXED_MODE65N-1P7M1T2F-LL-Calibre-drc-1.14-P1
Application_Note_LOGIC_MIXED_MODE65N-LL-Calibre-drc-1.0-P1.pdf  G-DF-LOGIC_MIXED_MODE65N-1P7M1T2H-LL-Calibre-drc-1.14-P1
DRC_QA_Report_G-DF-LOGIC_MIXED_MODE65N-LL-CALIBRE-DRC-Ver.1.14_P1.pdf  G-DF-LOGIC_MIXED_MODE65N-1P7M2T0F1U-LL-Calibre-drc-1.14-P1
G-DF-LOGIC_MIXED_MODE65N-1P10M2T1F1U-LL-Calibre-drc-1.14-P1  G-DF-LOGIC_MIXED_MODE65N-1P7M2T0F-LL-Calibre-drc-1.14-P1
G-DF-LOGIC_MIXED_MODE65N-1P10M2T2F-LL-Calibre-drc-1.14-P1  G-DF-LOGIC_MIXED_MODE65N-1P7M2T1F1U-LL-Calibre-drc-1.14-P1
G-DF-LOGIC_MIXED_MODE65N-1P10M2T2H-LL-Calibre-drc-1.14-P1  G-DF-LOGIC_MIXED_MODE65N-1P7M2T1F-LL-Calibre-drc-1.14-P1
G-DF-LOGIC_MIXED_MODE65N-1P4M0T0F1U-LL-Calibre-drc-1.14-P1  G-DF-LOGIC_MIXED_MODE65N-1P7M2T1H-LL-Calibre-drc-1.14-P1
G-DF-LOGIC_MIXED_MODE65N-1P4M0T1F-LL-Calibre-drc-1.14-P1  G-DF-LOGIC_MIXED_MODE65N-1P7M2T2F-LL-Calibre-drc-1.14-P1
```

- *If you want to create a new Run Directory for DRC, LVS or PEX, you can follow these steps :*

In your CDS_VISO folder, you will find a directory called ***calibre_refDirs****. Inside this directory are you will find three ready to use DRC, LVS and PEX folders containing the required rules. Just copy the one you need in your working directory and rename it the way you want.*

- ✓ Check the rule files in the *DRC_rundir, LVS_rundir* and *PEX_rundir* folders. You can open them with gedit. Do not modify them ! As you can see, these files are quite complex and embed a lot of options. For each technology feature, options can be enabled/disabled. The following box describes some rules and options that will be considered during the DRC check process.

- **Switches:** Design rule check file contains many rules related to layout constrains, but also rules related to further processing and production steps. In different phases of the design, different rules may be needed or used. At this stage of the design, we need to check if our layout fulfills the geometry constrains and the most relevant design rules. **Note that the design rules as well as the switches can be very different from one technology process to the other.** The naming of the switches can also be very different, even if they relate to generally similar design rules. Some are however very common and easily recognized such as: layer coverage, antenna errors, Electrical Rule Checks (ERC) or grid alignment.

  Some typical switches are explained here:

- **Layer coverage *(check_density)*** – IC fabrication process requires diffusion, polycrystalline silicon and metal layers to be sufficiently inhabited in order to improve the uniformity of the implantation and deposition, quality of polishing and overall reliability of the final design. Typically, the metal coverage of the final chip is around 30% of the total area, top metals usually require higher coverage, while polycrystalline silicon requires typically around 15%. The diffusion coverage is generally needed for higher gate-oxide uniformity. Here, the goal is to check only a small part of the design, so no coverage check is needed in our case.

- **ERC *(Skip_Soft-Connect_Checks)*** – Electric Rule Check is used to check the design for all possible connections that may lead to the final circuit malfunctions. Typically, ERC verifies for proper substrate and well contacts, looks for gates connected to pads or supplies (possible breakdowns), disconnected or unconnected inputs, floating and shorted outputs, etc. Generally, ERC check is needed more in the final stages of the design. Here it will be enabled. However, checking ERC should usually be performed after each larger design block (note that if you do not choose the switch, these rules will be included).

- **Metal Slots *(check_slots)* –** Similar as for coverage (and for the same reasons), checks the uniformity of metal patterns. The layers needed to fill the required coverage are in most cases generated automatically. This part of the DRC check validates if the automatically generated layers are properly placed and if they satisfy the design rules. It also checks for very wide lines in the design and suggests the possible changes, in order to avoid non-uniformity and stress due to large metals lines. This is important for the final chip, and we will not use it for our gate.

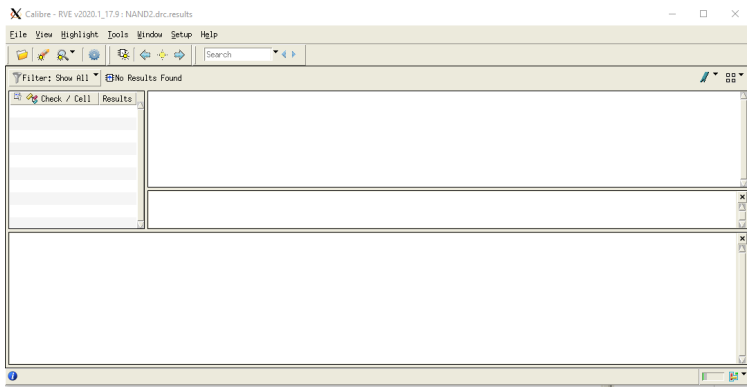
- **Grid *(included by default)* –** The masks that are used in different processing steps are always aligned to the same predetermined grid. This check verifies if the shapes you have placed correspond to the specified grid, or in other words it checks for possible off-grid elements.

- **Antenna Checks** ***(in this technology, antenna rules are in the separate rule file)*** – A very important design rule. It checks for Electrostatic Discharge (ESD) related problems. The most important antenna rule checks if gates of transistors have appropriate path towards VDD and ground (protection diodes) in order to conduct the excess of charge safely. If no protection is used, the accumulated static charge may cause large spikes of the gate voltage, causing the gate-oxide breakdown and circuit malfunction**.**
- **Seal Ring** ***(SR)*** **–** In order to protect the integrated circuit, a seal ring is required that is implemented around the edge of the chip. The seal ring is generally subject to a bit different design rules than the rest of the chip. This switch enables the DRC check for the seal ring. This rule is important for the final chip, and we will not use it for our gate.
- **Die Corner** ***(check_die_corner)*** **–** As for the seal ring, die corners are also subject to specific design rules. This switch enables the check for the die corners to confirm if they satisfy the design rules. This rule is also important for the final chip, and we will not use it for our gate.
- **Top Metal –** Top metal is also subject to specific design rules. It is therefore important to specify which metal is the final metal layer on our chip.

The Calibre DRC runs and gives you 2 output views :

- The **DRC summary report** that contains information on the checks and layers.
- The **RVE view** which contains the different errors. It is important to note that some errors cannot always be solved at your level.

This view shows that you have no DRC errors :



✓ Try to break a rule, for e.g., by drawing a random PO1 rectangle next to one PO1 gate, or moving the right side PO1-ME1 contact closer to the gate (min space rule is 120nm). For this example, we moved the contact on the left, and reduced the spacing to 80nm. Save your design and run the DRC again.

✓ A “Check PLY_F.S2” error should appear. If you click on the “+” it will show you in which cell the error appears. You can use this feature to filter your errors.

✓ The bottom panel gives you information on the rule being broken in the selected check. Here, it says that the minimum spacing for PO1 is 120nm.

✓ The figures on the right panel are reference to all the different DRC errors in your design. As visible here, only one error exists in this design. Double click on the “1”. Switch to your layout view, the DRC error is highlighted and visible. Correct it, save your design and run the DRC again.

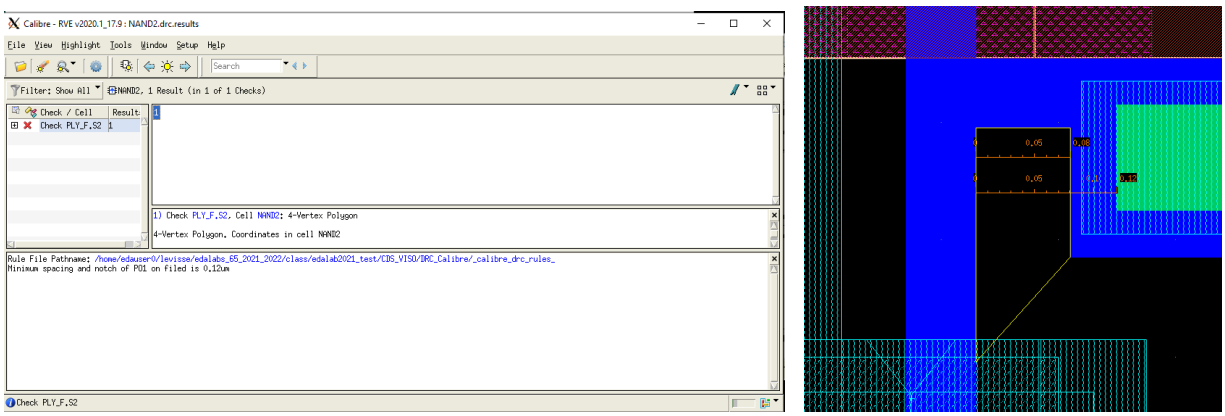


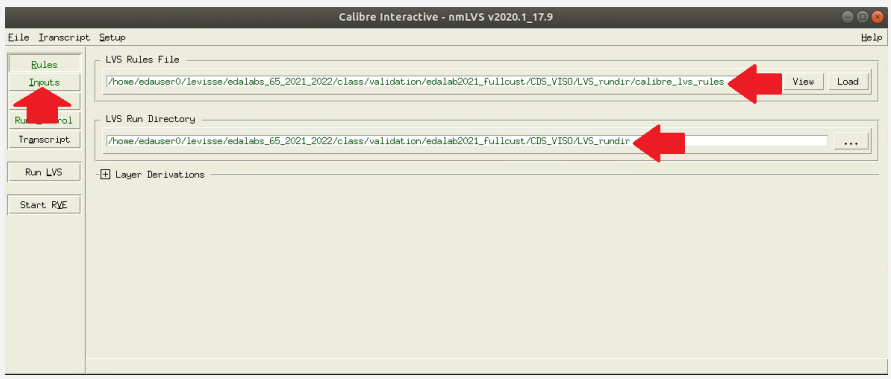
Figure 3 – An example for error in the layout and its description.

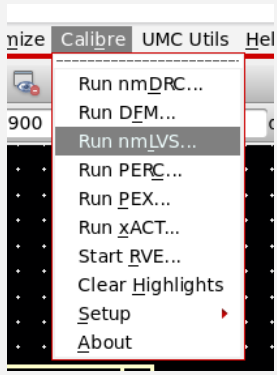
QUESTION 3-3: Explain with your own words what does a DRC does. What does a successful DRC means ? does it means that the circuit is correct ? and why ?

3.4. LAYOUT VERSUS SCHEMATIC (LVS) CHECK

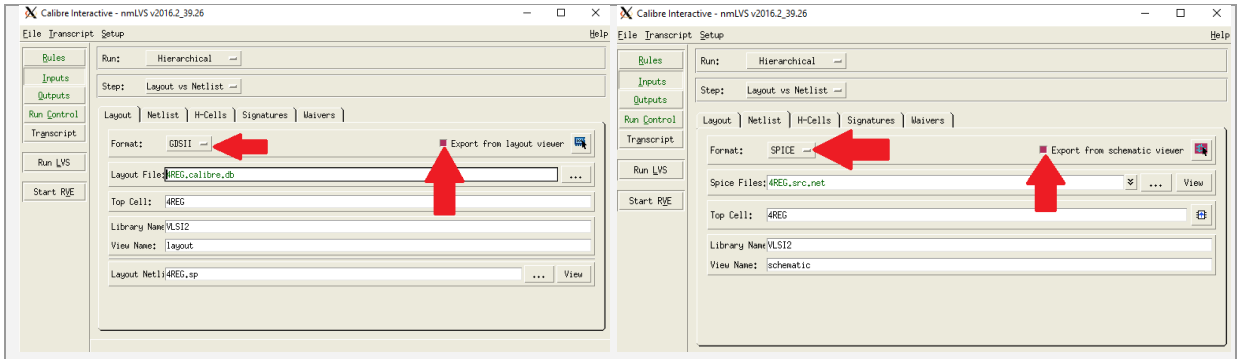
✓ Run Calibre>nmLVS

✓ Set the rules and run directory for the LVS as shown in the screenshot





✓ Click on inputs and make sure that in the “**layout**” and “**netlist**” panels the case “**export from viewer**” is selected. Note here that you could give it a netlist and a gds file. As Calibre is an independent tool (not developed by cadence) and does not need cadence virtuoso to run.



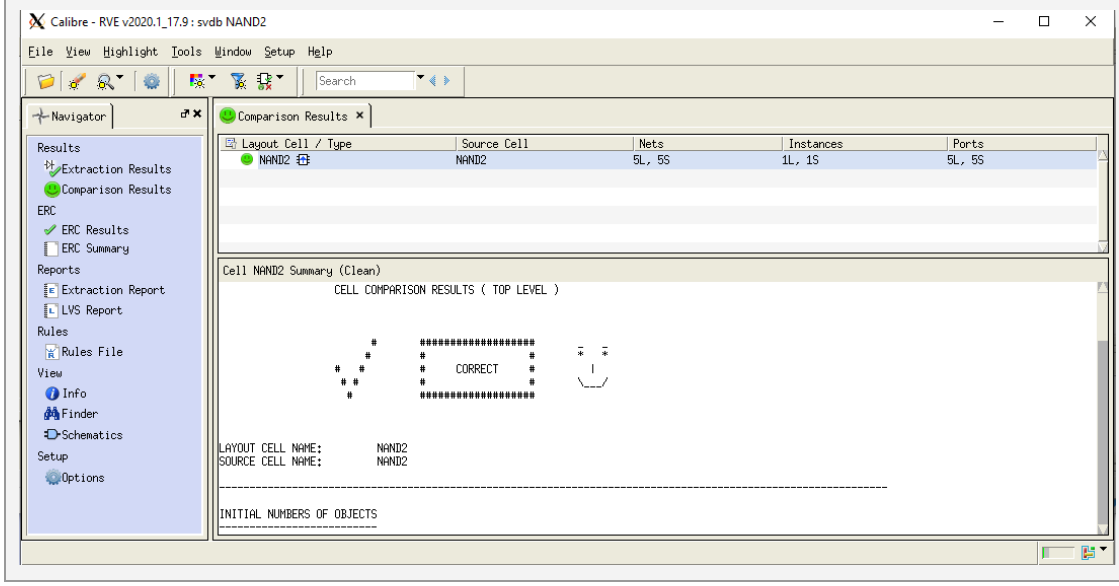
- ✓ You can explore the LVS options by clicking on **setup>LVS options**
- *Some of these options can help you with your design. When evaluating a big block with several vdd and gnd rails which are not yet connected, you can either connect them for the LVS pass, with a wire, or tick the "connect nets named" in the LVS Options>connect panel and input vdd and gnd there. "Connect all by name" can be used as well.*

CAUTION: these techniques may make you forget about some connections! so use it wisely! They should only be used if you know why you are using them ! (i.e., do not use them in this lab, but know they exist)

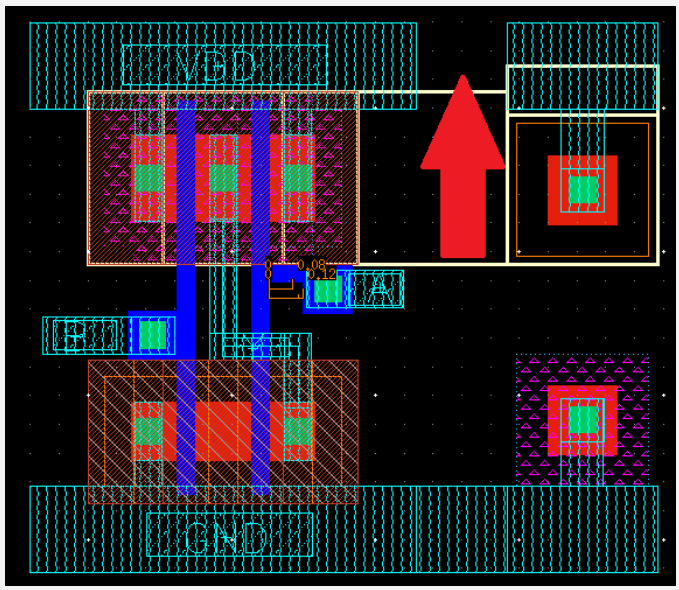
- ✓ Click on "Run LVS"

When the LVS is done, it will show you the calibre RVE view. Make sure that ERC (electrical rule checks) the extraction and comparison results are correct.

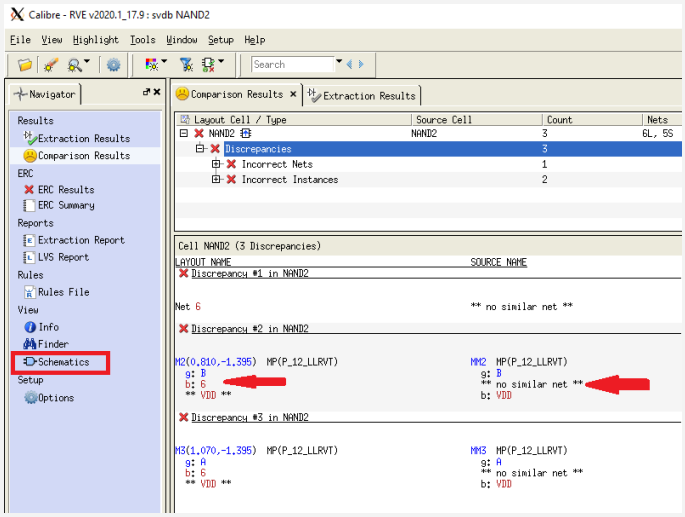
Calibre LVS shows you a nice smile is the comparison is good.



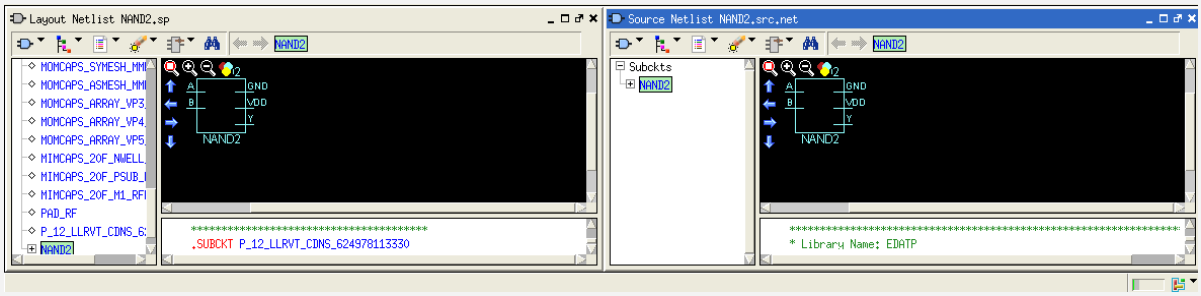
- ✓ Create an error in your layout to make the schematic and the layout different. Cut the connection between the VDD rail and the NWELL substrate biasing contacts. Save your design and run the LVS.



- ✓ The introduced error here makes the PMOS substrate disconnected from VDD.

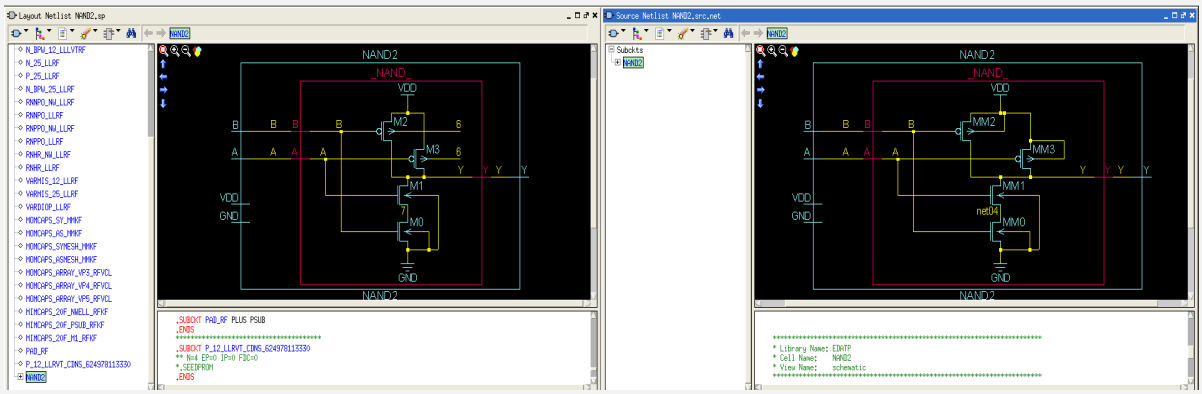


- ✓ The Layout Vs Schematic comparison found 6 nets on the Layout and only 5 on the schematic, Why ?
- ✓ When looking in details (bottom view), the two PMOS (M2 and M3) have their bulk (b) node connect to a node named “6” in the layout (left column) and VDD in the schematic (right column). Net “6” is referred as “no similar net” in the schematic column (called “SOURCE NAME”).
- ✓ If still not clear, you can click on the “schematics” menu in the left panel.



The left view shows the schematic extracted from the layout while the right view the original schematic.

✓ Double click on the two symbol cells, and then on the NAND2 symbol. Use the  key to center the view on the schematic.



✓ The missing connections are here clearly visible. Interpreting the LVS errors from the report, and being able to identify what to solve is an important quality of a full custom designer.

LVS errors will not always be as straightforward as here. As the LVS extracts the schematic from the layout, it is important that you perform hierarchical LVS steps. As for the DRC, consider that while you do not have a LVS clean, your block is not working.

Here, when disconnecting the TAP cell, we created LVS errors. If you want to use TAP cells in your design, you will not use one TAP cell per logic gate, but rather include 1 TAP cell every once in a while in the NWELL rail. You should thereby, expect your LVS to fail if the TAP cell is not included in your LVS check. One possible design flow is to design the cell without the TAP cell, and check for the DRC. Then, as the LVS does not check DRC rules, you can add the TAP cell in your layout, and validate the functionality. You can finally remove the TAP cell and integrate the cell within the next hierarchy level.

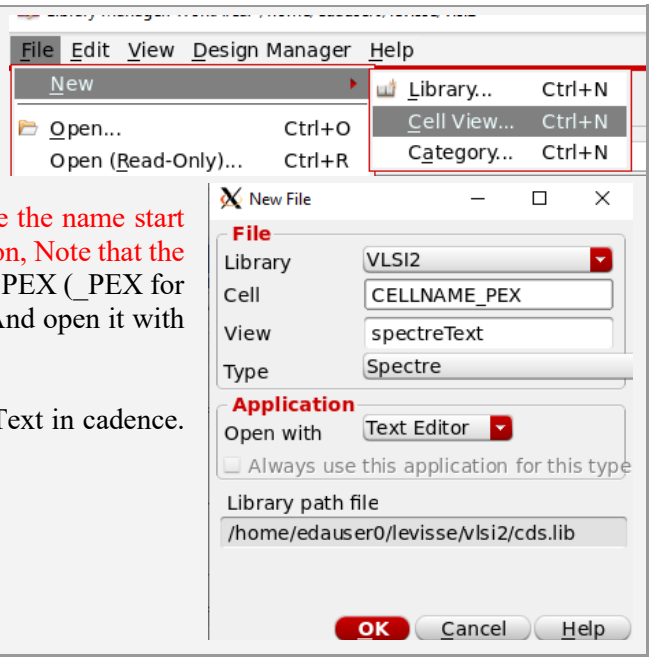
QUESTION 3-4: Explain with your own words what does a LVS does. What does a successful LVS means ? does it means that the circuit can be fabricated ? and why ?

QUESTION 3-5: In your opinion, could a DRC be wrong and a LVS correct ? and vice-versa ?

3.5. LAYOUT PARASITICS EXTRACTION

In this section you will learn how to use Calibre PEX for creating an extracted netlist of the layout. The extracted view of a layout consists of parasitic and wiring capacitances and resistances (depending on what is being selected) in addition to the elements in the schematic view. The extracted netlist will later be used for post-layout simulation.

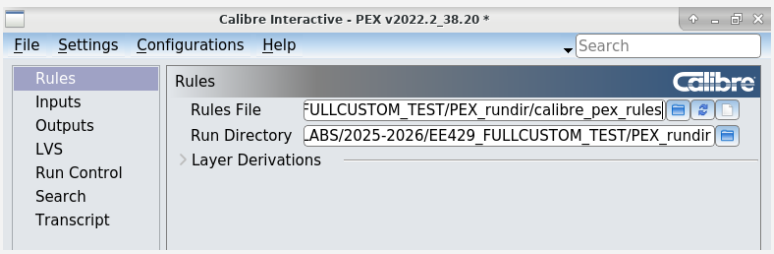
- ✓ First, create a cellview, in which you will host the extracted netlist. Select the cell that you want to extract and click New>Cell View…
- ✓ Give it a name that correspond to your current naming methodology (do NOT make the name start with a number, you may have troubles later on, Note that the names are cap-sensitive), with a reference to PEX (_PEX for e.g.). Make it a spectre type (spectreText). And open it with the text Editor.
- ✓ Click ok
- ✓ The new cellview now appears as a spectreText in cadence. Close it.

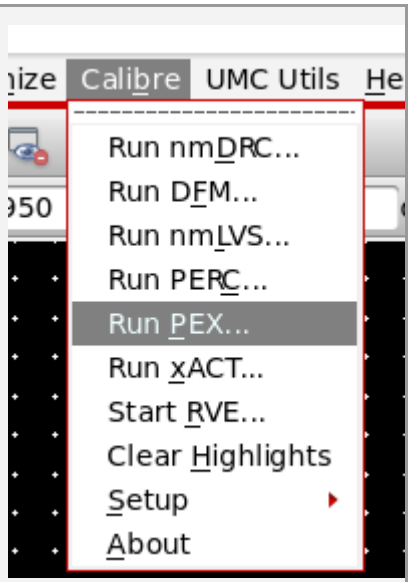


- ✓ Switch to the layout view of the cell you want to extract.

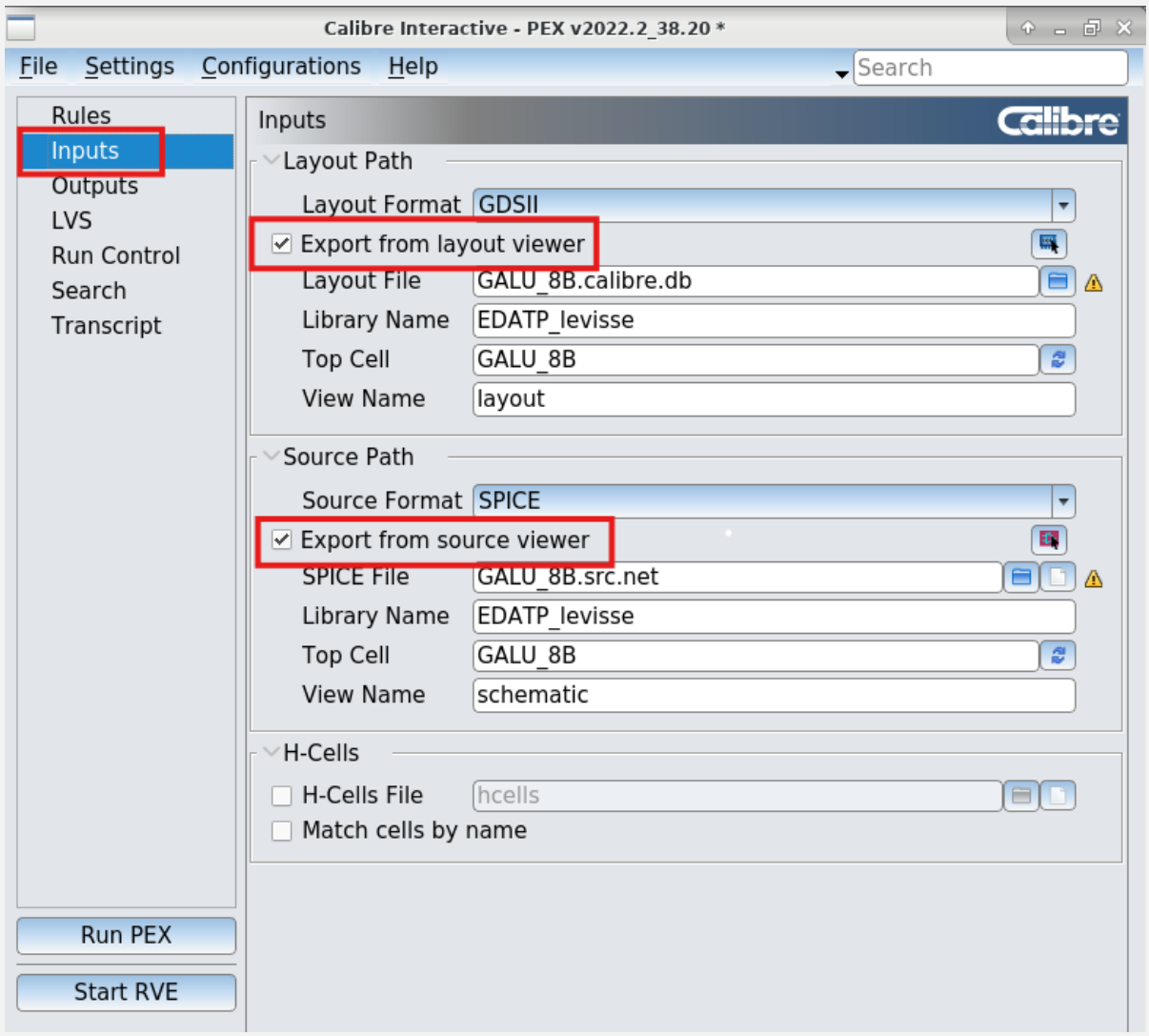
- ✓ Open Calibre>Run PEX
- *Parasitic extraction does something a lot alike the LVS, but then does generate a new netlist that can then be simulated and compared to the pre-layout netlist (schematic simulation).*

- ✓ Select the **calibre_pex_rules** file inside the PEX_rundir folder. As for the LVS, select the Run directory as being the directory holding the rule file (in this example, PEX_rundir).

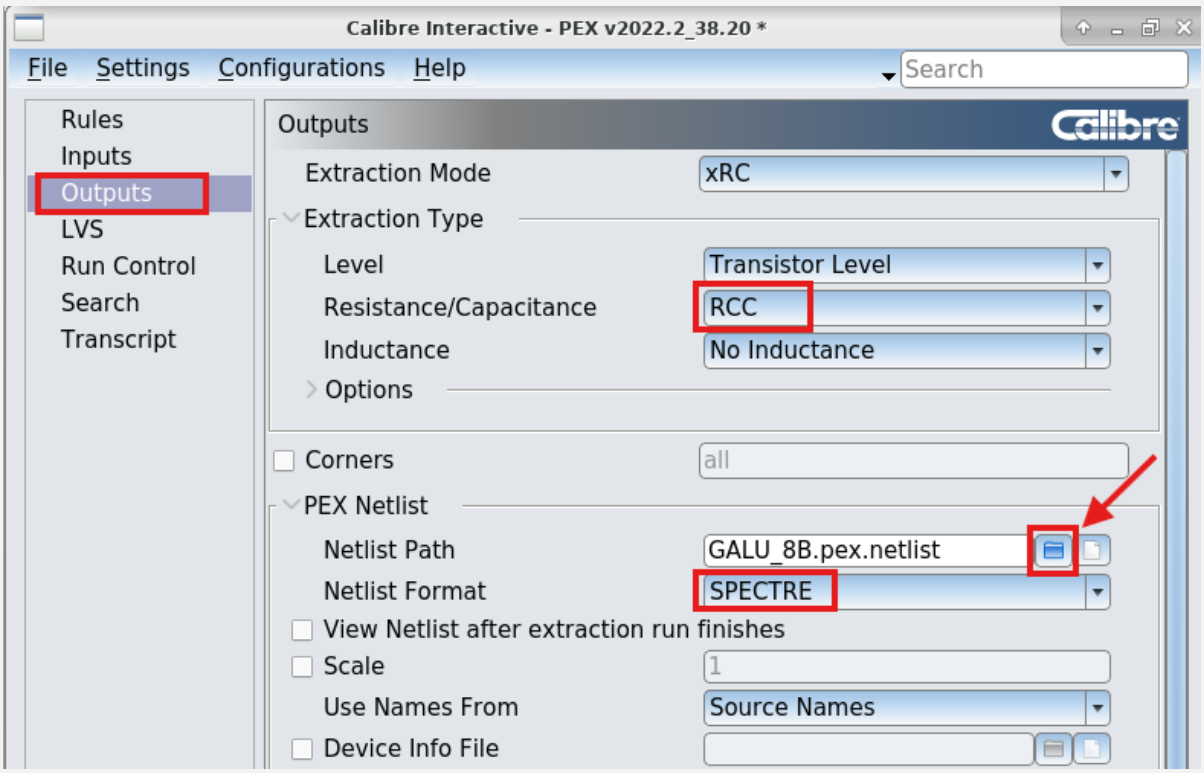




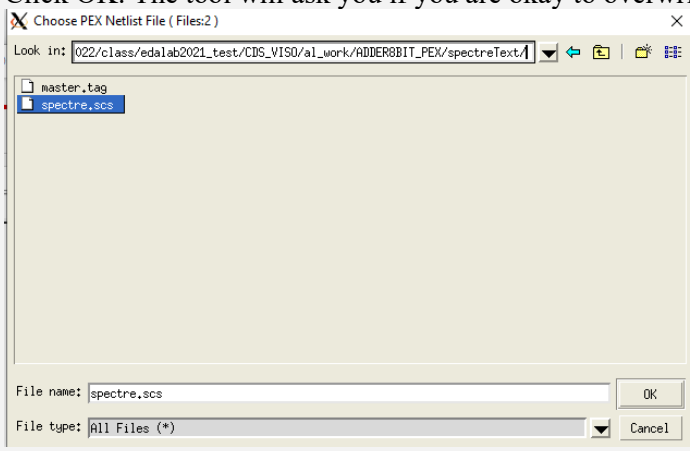
✓ In the Input page, under the layout tab, make sure that the box export from layout viewer is selected. Same for the netlist tab.



✓ In the Outputs page, in the netlist tab, make sure that format is set to SPECTRE
✓ Set the extraction type to Transistor Level, and RCC, and No Inductance.



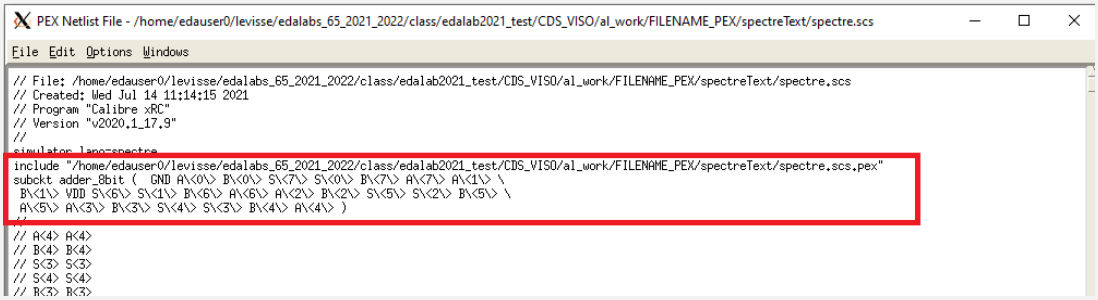
✓ In the Netlist Path field, select the spectre.scs file that corresponds to the spectreText cell you just created inside virtuoso. In this example, the library is called "al_work" and the cell is "ADDER8BIT_PEX". Just chose the good path corresponding to your newly created cellview. Click OK. The tool will ask you if you are okay to overwrite the file. Click Yes.



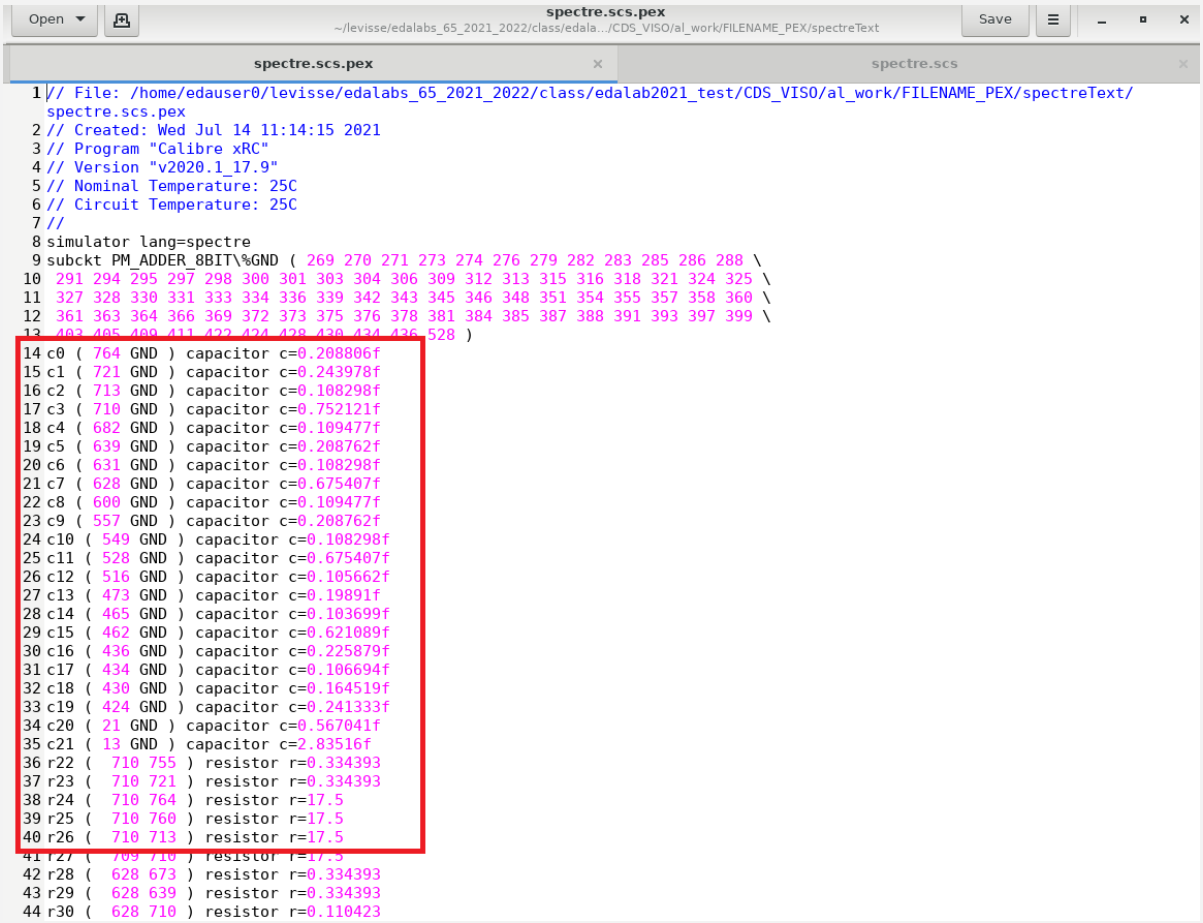
✓ Click on Run PEX

ℹ *If you ticked the "View netlist after PEX finishes"on the output panel, it will show you the extracted netlist at the end of the process. Spectre netlists are not generated in a single file. The main file defines the transistors and nets.*

ℹ The Spectre.scs.pex file contains all the parasitics elements.



Note that there are no resistance and capacitances values lower than the values you define as limits in the calibre PEX options.



✓ Close the window and open the netlists with a text editor from your terminal (you can also open the spectre.scs netlist from the library manager window, but the editor is not really convenient to use).

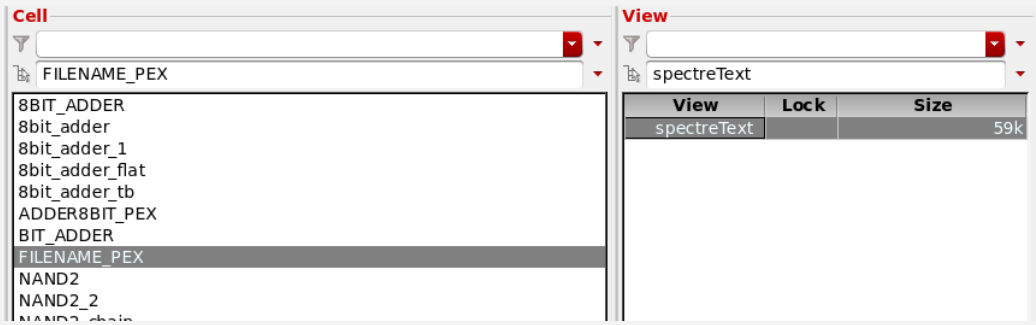
[edauser0@selsrv1 CDS_VISO]$ gedit EDATP/FILENAME_PEX/spectreText/spectre.scs &

✓ Correct the name of the subckt created by Calibre PEX with the name of the cell at the beginning (line 8) and at the end of the file. You could also do this step by opening the spectreText view in the library manager.

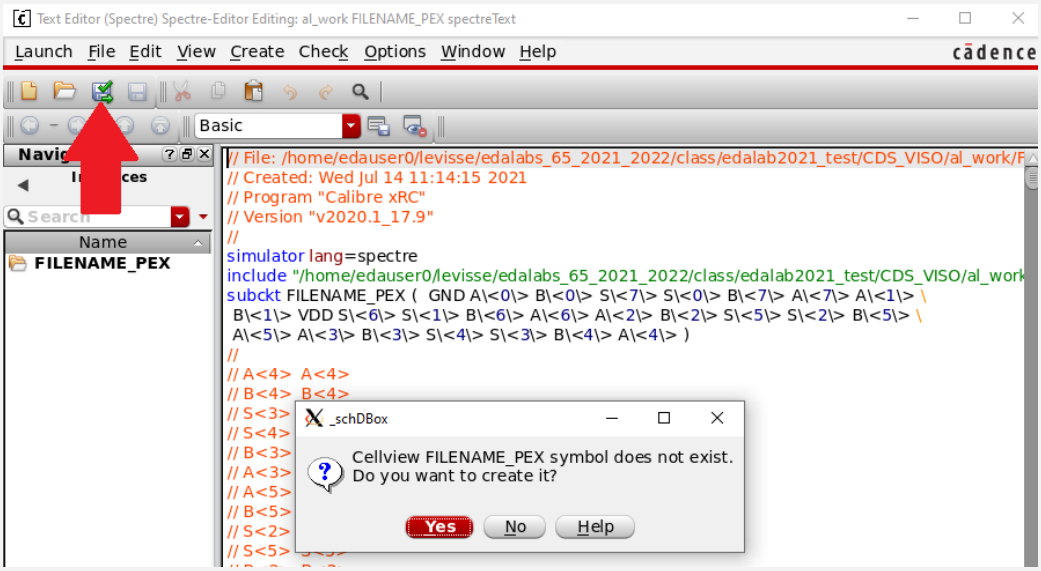
Using gedit, take a look at the other files that are being included inside the spectre.scs file. What do you see there ? ***do not modify them !***

```
 8 subckt FILENAME_PEX (  GND A\<0\> B\<0\> S\<7\> S\<0\> B\<7\> A\<7\> A\<1\> \
 9  B\<1\> VDD S\<6\> S\<1\> B\<6\> A\<6\> A\<2\> B\<2\> S\<5\> S\<2\> B\<5\> \
10  A\<5\> A\<3\> B\<3\> S\<4\> S\<3\> B\<4\> A\<4\> )
11 //
931 N_VDD_XI7\/MPM13_b ) P_12_LLRVT l=6e-08 w=2.4e-07 ad=3.84e-14 as=2.4e-14 \
932 pd=8e-07 ps=4.4e-07 nrd=0 nrs=0 sa=1.6e-07 sb=1.35e-06 sca=16.52 scb=0.019587 \
933 scc=0.000974964
934 //
935 include "/home/edauser0/levisse/edalabs_65_2021_2022/class/edalab2021_test/CDS_VISO/al_work/FILENAME_PEX/spectreText/
    spectre.scs.ADDER_8BIT.pxi"
936 //
937 ends FILENAME_PEX
938 //
```

✓ Save the file and close. If you used gedit to edit the netlist, go back to your library manager. Select your cellview as shown in the next screenshot. Double click on the spectreText to open it.



✓ Click on the "**Build a database of instances, nets and pins found in file**" button. 

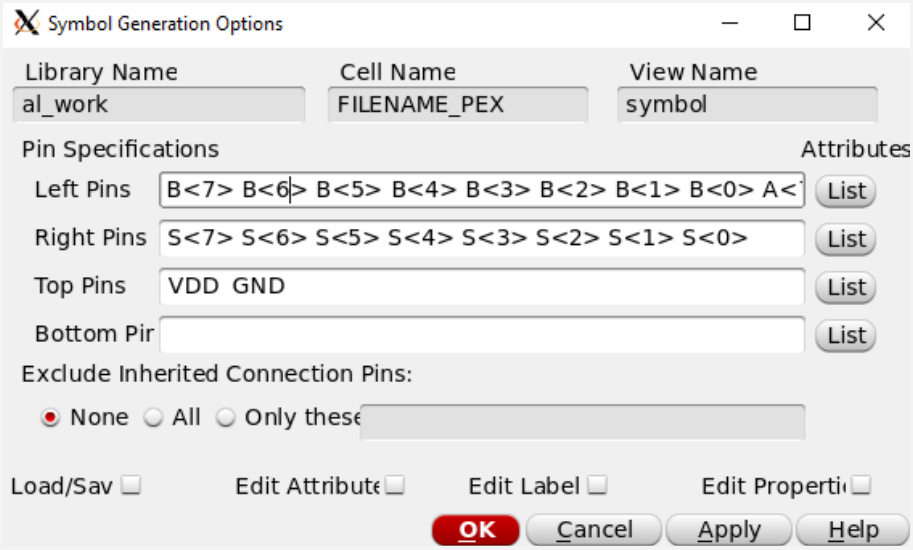


✓ At this point, the tool will ask you if you want to create a symbol for your cellview. Click yes.

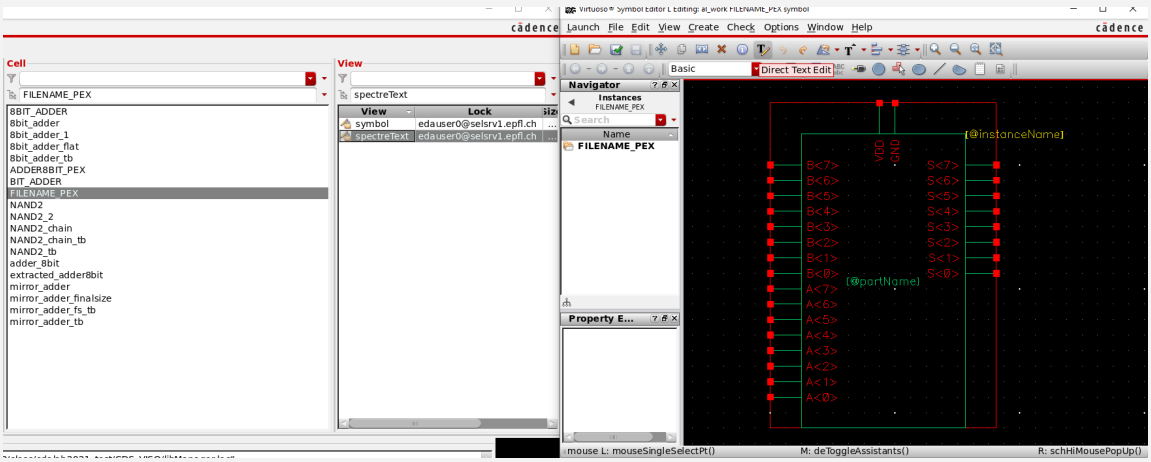
If the tool tells you that you have a syntax error, check that the name you gave for the subckt at the beginning end at the end the file is the same than the cellview name. if you used gedit to modify the file, make sure that the file is not being opened/locked in virtuoso when saving it. Also, make sure that your subckt name does not start with a number and that you respect the capital letters in both the name and cellview name. This will make the simulator/parser crash. Finally, if you renamed the cellview, make sure that you updated the paths associated with the

"include" statements to account for the new name. If none of these solutions work, try to follow the flow again with a new cellview. ***Then, if none of this works,*** *you can call a TA.*

✓ Create the symbol as shown in the screenshot. A good practice is to keep input on the left, outputs on the right. For the postPEX symbol creation, do **not** merge the pins (e.g. A<7:0>). Click OK.



✓ A new symbol is not being created. Check in the library manager that you now have a "symbol" view. Save and close it. The example in the screenshot is for a 8bit adder.



✓ Call a TA to check your circuit with you.

ℹ Here, you will note that the calibre PEX tool does create as many pins as there are wires. You can select the instance and press "space" to avoid naming them by hand. And then, with the "l" key, create a bus label (e.g., A<0:7>) that you attach to a single wire, to make all these wires a bus again. Also, note that you could create a wrapper cellview in which the wires are connected to a bus.

QUESTION 3-6: Explain with your own words what does a PEX does.

QUESTION 3-7: what does the resistance and capacitance filter means for the precision of the simulation? Describe the trade off.

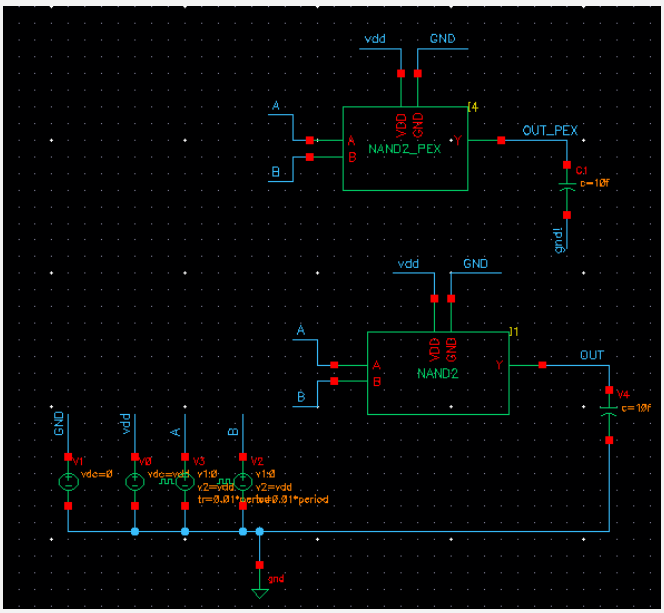
QUESTION 3-8: Metal corners exist. In the lab, we make you use the typical corner. How do you think parasitic metal corners are being defined ? hint. They consider capacitance and resistance.

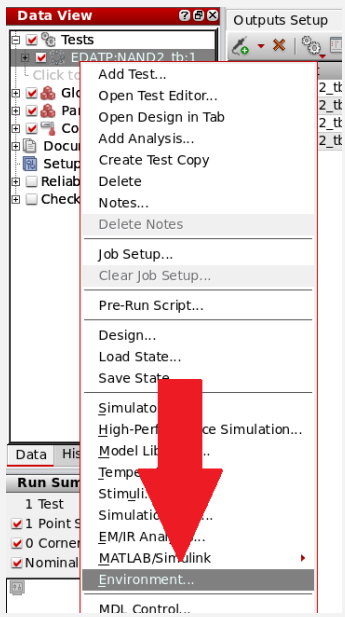
3.6. POST LAYOUT (EXTRACTED) SIMULATION

Throughout this section you will learn how to run a post layout simulation from the extracted netlist.

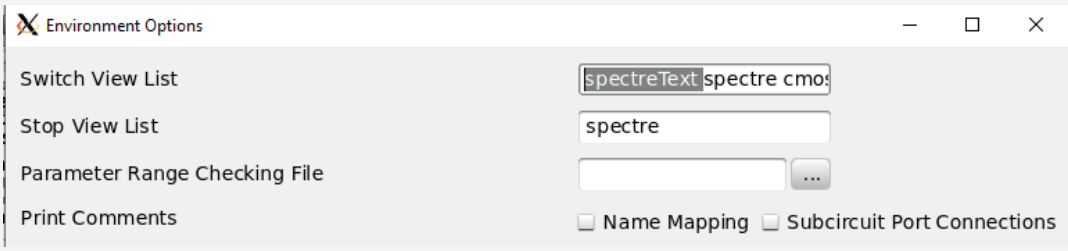
- ✓ Open the testbench you designed for your NAND2 gate.
- ✓ Import the NAND2_PEX cell and connect -it to the same inputs A and B. connect a 10fF capacitance to the OUT and OUT_PEX signals. Do NOT connect them together !

  ***Small tip**: select the instance and press spacebar. Virtuoso will automatically create wires with corresponding labels. This is really useful when dealing with big instances that have many input-ouputs. Just rename the labels you need to rename.*





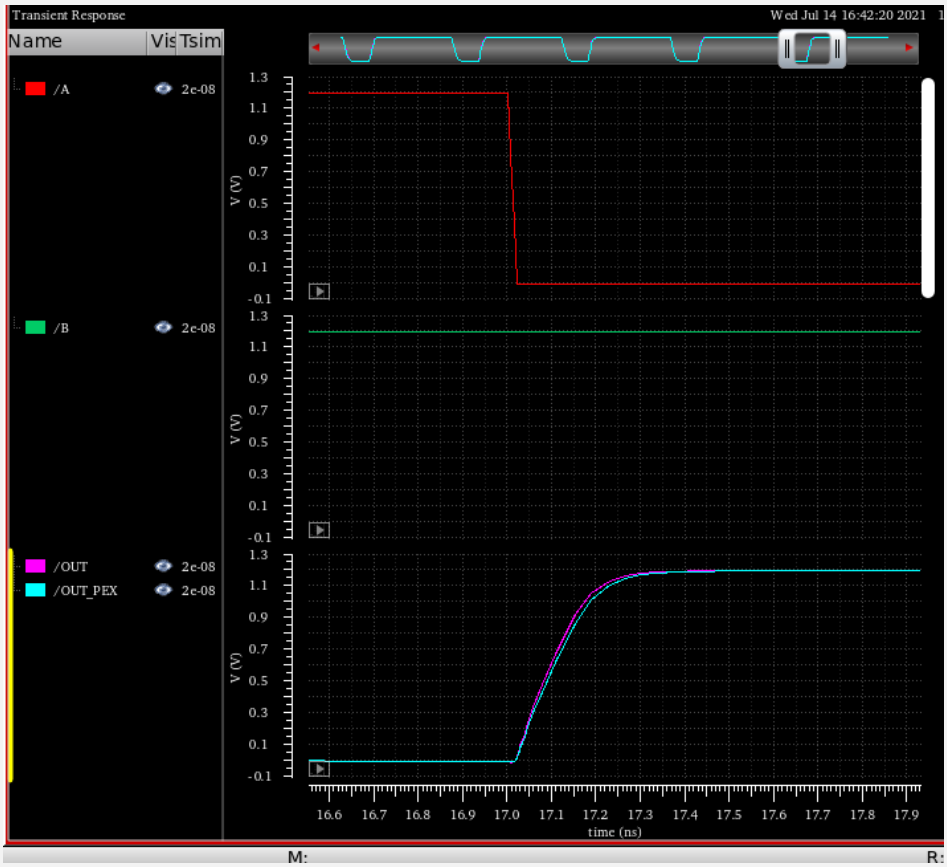
- ✓ Open your ADE XL view for the testbench (NAND2_tb).
- ✓ You must tell ADE that he will have to look into both the schematic view and the spectreText views. Right click on the transient test you will run. In the dropdown menu, select **Environment**
- ✓ Add spectreText in the first position of the "Switch View List". Click OK. *Be careful with the capital letters, make sure you have it properly written.*



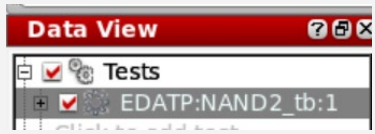
✓ Setup your test outputs and run the simulation.

Test	Name	Type	Details	EvalType	Plot	Save
EDATP:NAND2_tb:1	A	signal	/A	point	☑	☐
EDATP:NAND2_tb:1	B	signal	/B	point	☑	☐
EDATP:NAND2_tb:1	OUT	signal	/OUT	point	☑	☐
EDATP:NAND2_tb:1	OUT_PEX	signal	/OUT_PEX	point	☑	☐

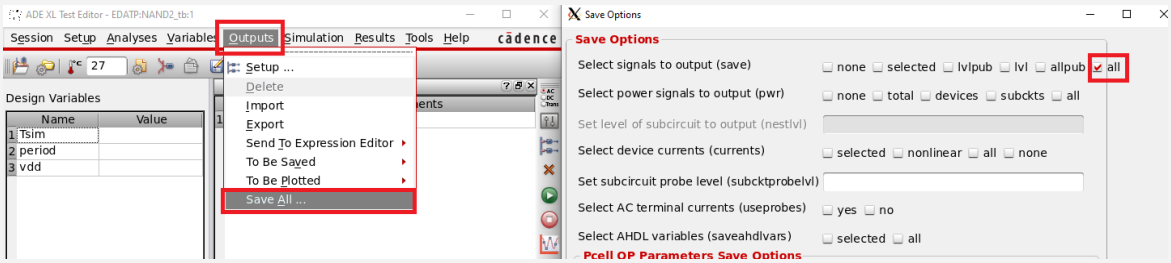
✓ Zoom in the plot outputs to see the difference between the pre and post-PEX simulations. What do you observe ? why is the NAND2_PEX cell slower ? here the difference looks small, but when simulating a larger circuit, you should expect much more parasitic-induced delays.



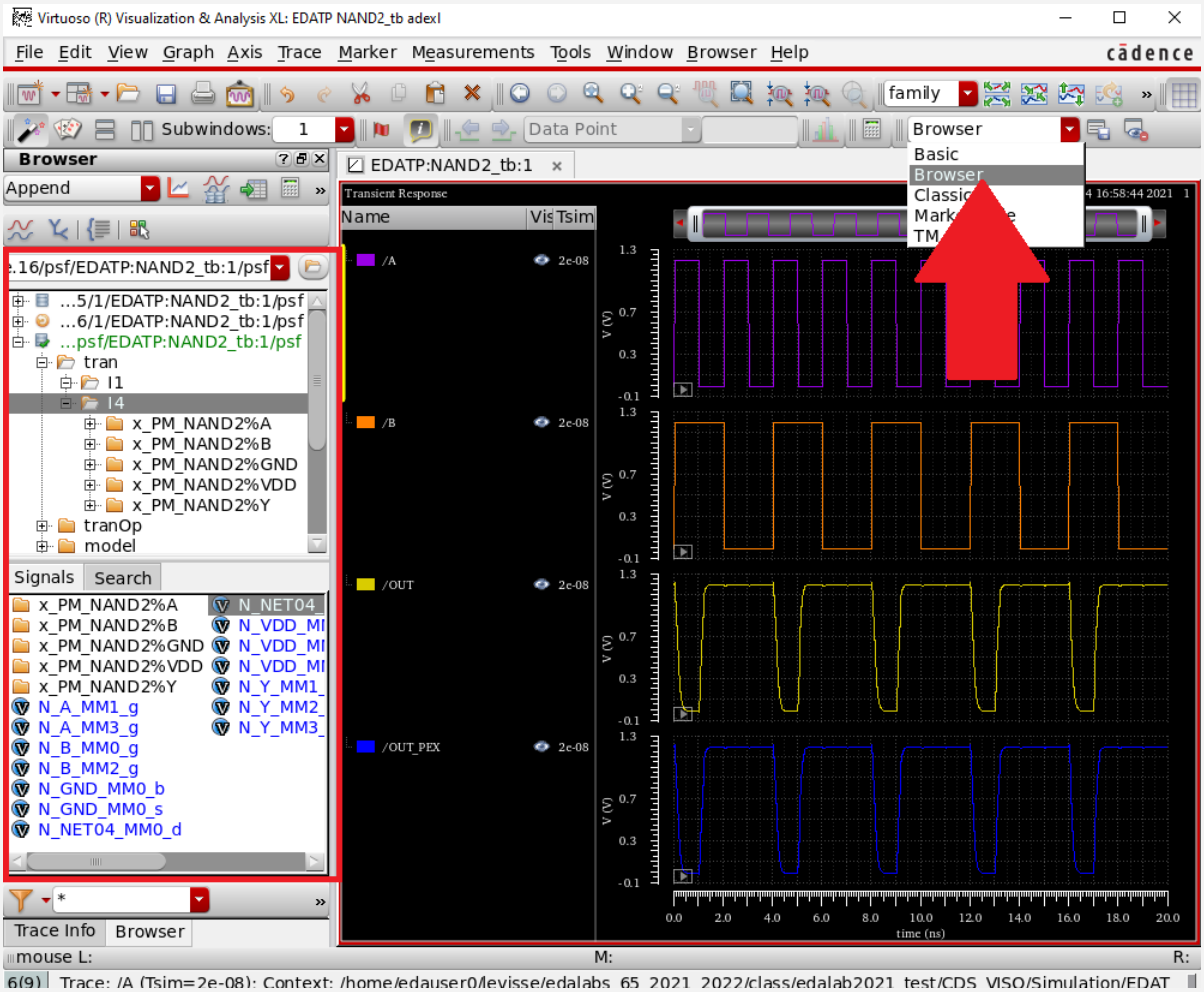
ℹ In order to observe internal nodes inside the extracted cell. You need to tell the simulator to save all the signals so that you can observe them in the wave viewer. In the data view,



double click on the transient test you want to run. Then select outputs>save all… tick “all” and click ok. Close the test editor.



After running the simulation, in the wave viewer. Select the browser mode in the top right corner. And then in the left panel, a hierarchical view of all the signals will appear. You can dive down into it, and plot the signals you need.



4. DESIGNING 2-TO-1 AND 4-TO-1 MULTIPLEXER GATE (OPTIONAL)

- ✓ Open the schematic of either a MUX21. Launch the ***Layout XL*** and design the layout for your 2-to-1 MUX schematic.
- ✓ Using the 2-to-1 multiplexer layout, create the 4-to-1 multiplexer layout.
- ✓ Open the MUX 4-to-1 schematic and launch ***Layout XL***.
- ✓ Instantiate all the MUX 2-to-1 layouts and connect them properly.
- ✓ Launch the DRC and LVS checks to verify the design (you do not need to run a PEX for MUX 4-to-1).

Here are some guidelines on how to design the MUX 2-1:

- Define a height for your cell. Try to take the same for all the cells. It will help you making the final abutment.
- You can start from the dimensions we propose in the design of the NAND gate.
- Make a drawing ! don't jump on the layout editor. Take pen and paper and draw a draft of how you would see it. If TAs are available. Propose them your drawing.
- Avoid stacking PMOS or NMOS. Put the PMOS next to each other and the NMOS next to each other.
- When possible. Try aligning the gates between the NMOS and PMOS.
    - For e.g., the input inverter here is easy to align.
    - In the second stage, gates controlled by A and B are easy to align. Though you would need to cross wires for S and S_INV. There is not much you can do there. Alternatively you could align S and S_INV. And manage for A and B.
    - Don't be too greedy with density. It will take too make your design harder.
- Tons of optimizations techniques can be used. Have a look at the layout from the UMC65LL_UMK65LSCLLMVBBR__B03PB standard cell library. These are extremely dense designs, though some of the tricks applied there can be useful.
- More guidelines will be available in the project document for the layout of the 8bit adder.

5. PROBLEMS, SOLUTIONS, ALTERNATIVE APPROACHES

5.1. INSTANCES UNBINDED

It can happen, and WILL happen, that your instances get "unbinded" between the layout and the schematic. This is related to Layout XL, which "connects" (binds) instances between the layout and the schematic, in order to help you do the layout. Though, if instances get unbinded, then the tool gets confused and complains about nets or instances being incorrect. The bindings are automatically created when you "create from source".

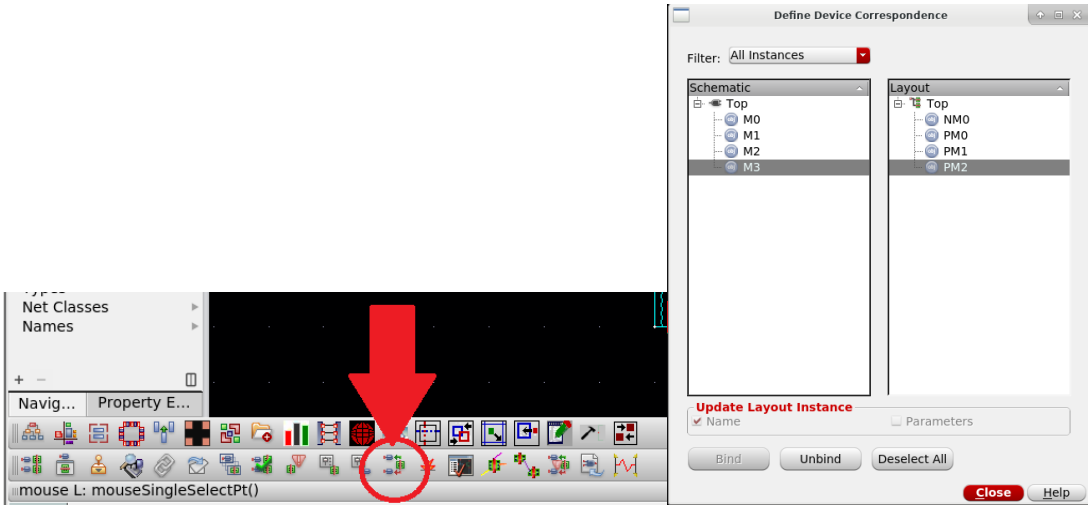
This problem typically happens when you :

- Flatten a transistor. At this point, instance binding are being lost.
- Invert two instances which were supposed to be connected differently.
- Reverse one transistor by inverting drain and source connections.

These situations are NOT problematic, though, as the tool assumes you use its features, it will get confused and mess with your design experience.

Two possibilities arise :

1- For simple designs , sometimes, help from the tool is not mandatory, if you know what you are doing. You can simply use Layout L (instead of XL) which does not feature instances binding. Though it does also not feature “create from source”. For e.g., you could use XL to automatically create the instances, and then switch to L.
2- You can correct the binding. Click on the “define device correspondence” button.

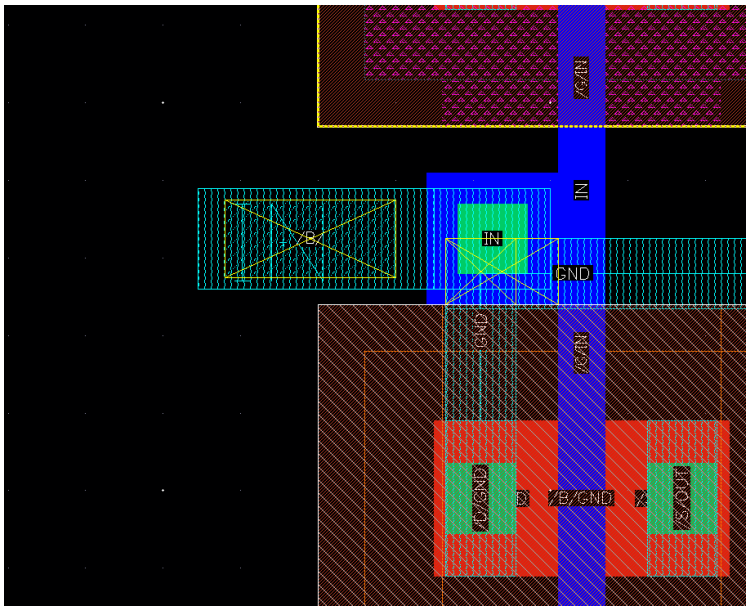


The new window allows you to unbind wrongly binded instances and connect them properly.

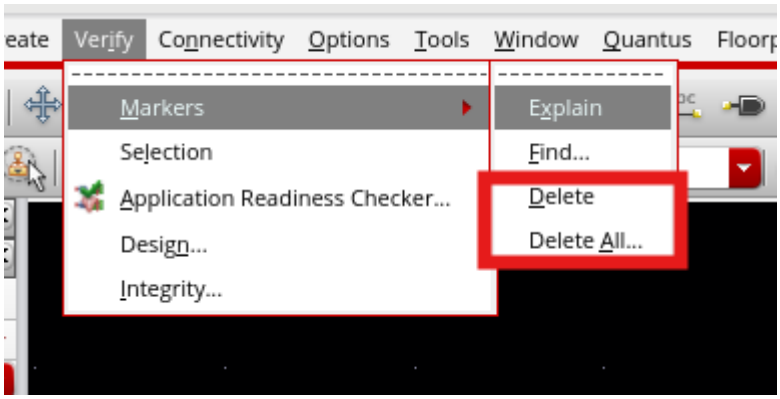
This can be done at various level in the hierarchy. And while it could sometimes be useful to disable the binding inside a small cell design, it is always useful to correct the binding on a higher level of hierarchy. This can help you figuring out where do each wire is supposed to be connected to.

5.2. MARKERS

Unbinded instances will trigger markers in the virtuoso environment. Even if the problem is solved these will not go away.



An easy way is to clean markers with the following approach:



Verify>Markers>Delete or Delete All…

The Delete button allows you to select the markers to remove, while Delete All… is more generous

Note : This feature is accessible in ic_studio, the latest version of virtuoso.

5.3. NETLIST AND EXTRACTION

Parasitic extraction is a good exercise to understand what happens behind virtuoso. In this lab, we purposely make you use a netlist on the output of the PEX. Though, other approach exist which enable you to directly import a schematic-like view on virtuoso. Further more advanced techniques will be explored in the advanced vlsi design EE-490b in the next semester.