ENSC450: VLSI Design

Lab Tutorial: Manual layout

ENSC450 focuses on building computation systems with CMOS logic.

While most of the course focuses on the design of complex DSP Logic, the design of any complex Multi Kgate logic comes down to selecting (Synthesis), placing, synchronizing (Clock Tree Synthesis) and Interconnecting (Routing) basic functionalities (Gates). Such gates are simple CMOS circuits. CMOS circuits are implemented in the silicon substrate drawing appropriate doping areas, SiO2 insulator layers, and metal/polysilicon interconnect on top of those areas. We have designed and simulated several cells with HSpice, but drawing and simulating a CMOS circuit at spice level is an abstraction. We have to make this abstraction real by designing, physically, the transistors we have been utilizing on the silicon:

Starting from a reference CMOS circuit, we need to draw the diffusion (n+ and p+) areas for each transistor, the Polysilicon Gate, and the Metal connection that route the I/O Signals and Vdd/Gnd to the cell pins. After we have done that, we shall extract a new circuit from the layout. This is done in order to verify that the layout we have designed actually corresponds to the starting circuit (this step is called LVS Layout-versus-Schematic check) . Also, after layout, we will have correct information about cell area, pin position, and parasitic RC values that could only be guessed at circuit level.

For our labs we will be using the Cadence Virtuoso IC development suite, that is by far the most utilized manual layout platform both in industry and academia. Virtuoso is not to be confused with Cadence encounter, that is a different software we will be using for implementing automated layout starting from VHDL descriptions.

The fundamental difference between the Virtuoso suite and the Cadence Encounter suite, is that

a) Virtuoso is used for the implementation of small designs starting from Circuit (Spice) representations, while Encounter is used for the implementation of large designs starting from HDL representations
b) Virtuoso requires the user to manually specify design areas and perform manual metal interconnect (there are ways to use automatic routing but they are suboptimal and they will not be utilized during this lab). Encounter only requires the user to specify Floorplan and constraints while performing placing and routing automatically

Most of the layout work, in academia and industry, is done manually, clicking and dragging over the silicon space the desired areas (Diffusion, Wells, Contacts, Polysilicon). That require very specific training and expertise, that goes way beyond the scope of a generic course in VLSI CMOS. Nevertheless, you need to be aware of the basic steps and guidelines for this kind of activity.

The following Tutorial will guide you through the most significant steps that layout designer need to master, for a very simple basic Inverter cell

The basic step followed by a designer in the definition of a given circuit layout are the following:

1) ***Schematic Entry:*** Based on the notions you acquired in Unit 2, the desired circuit (for us it will be a Standard Cell) is realized by means of an abstract schematic composed of nMOS and pMOS transistors. We will have to choose reference transistor parameters, that we will then try to match in our layout. If that will not be possible we will get back to the schematic to refine those values.
2) ***Layout*** : Using the schematic as reference, and based on the notions you acquired in unit 3, the desired circuit is implemented designing the actual mask patterns on the silicon space drawing polygons. The interaction between polygons will describe the transistors in the schematic and the relative interconnections
3) ***DRC (Design Rule Check):*** Unit 3 described how Layout design is driven by Technology specific Design Rules, that guarantee that the drawn polygons will behave as expected, and that lithography mask will be able to guarantee a faithful reproduction of the shapes we mean to draw***. Rules describe how far each of the layout shapes must be from another, and the legal dimensions of the shapes we can design.*** DRC is a design step in which our polygons and their spacing is verified. If DRC is not successful, we need to go back to the layout and fix all our Design Rules violations. This step is very costly and takes up the largest part of the design effort: you will feel very, very relieved when your complete design will undergo a successful DRC check
4) ***LVS (Layout Versus Schematic):*** It is not sufficient that your design is compliant to design rules. It must also implement the same physical behavior as described by the schematic. LVS is an automated formal check that verifies that the geometries realized during Layout design correspond to the schematic drawn in all aspects. LVS is divided in two steps: *extraction*, where the layout shapes are analyzed in order to determine which electrical devices they represent. *LVS proper*, where this extraction is compared to a reference schematic to verify that the user has really implemented on the layout his/her initial intentions. If there is no match, the designer can either go back to the layout and fix it, or change the schematic in order to match the layout constraints.

Note that design rules and layout coordinates are in multiples of 0.005 microns in this technology (the manufacturing grid value). This is specified by the supplier of the technology.

1. Environment Setup

Before we set off designing our cells we need to create a suitable environment:

As described in previous tutorial each time we set off with a new activity it is advisable to build a new terminal, name it appropriately, and configure it in order to utilize only the set of tools required by the activity.

Right Mouse Clock -> Open Terminal

Terminal -> Set Title *< Choose a name that makes sense with this activity, such as Layout or Virtuoso >*

In the case of Layout, we need access to some confidential technology information (Screen captures or file-copying is severely prohibited, except for handing in the Assignment report).

We will be using 180 nm TSMC technology provided by CMC Microsystems.

source */ensc/fac1/fcampi/setup/CDS_setup.csh*

cp ~fcampi/.simrc ~
cp ~fcampi/.cdsplotinit ~
cp ~fcampi/.cdsinit ~
mkdir <Your Working Folder for this Tutorial>/CDS
cd !$
cp /ensc/fac1/fcampi/Tutorials/Layout/CDS/cds.lib .
mkdir LVS
cd !$
cp /ensc/fac1/fcampi/Tutorials/Layout/CDS/LVS/.simlocal .
cp /ensc/fac1/fcampi/Tutorials/Layout/CDS/si.env .

[edit the si.env file with a text editor specifying correspondence File to point to your CDS directory]

cd <your path>/CDS

2. Opening Virtuoso

In this section we will use existing Skill code to design an Inverter. You can start the Virtuoso Layout environment from the CDS directory with the command

startCds -t cmosp18 -w . &

-w specifies the working library, that should be your CDS directory

-t specifies the technology that in our case will be cmosp18 -> TSMC 180nm

Shortly, you should have the

- **icfb** console or command input window (CIW), with a blinking cursor on the command line. This has a menu function banner, a scrolling log window, and a command input line.
- **A "Library Manager" (LM)** window. If you don't get the library manager automatically you can start it by pressing "F6" or clicking Tools-> Library Manager . The LM window will show libraries that were included in your cds.lib file. If you left-click on a library name, you will see a list of *cell views*. In particular, vst_n18 is a standard cell library. Each cell has several views, such as symbol, schematic, layout, etc. You do not have write access to any of this, so only open such views in "ro" (read-only) mode. If you open a layout cell view, you will have only "*abstract*" views: this means that there isn't much there but some metal1 (blue) and a few pin names (white). This is because we don't have access to the real layouts, the abstract views will be substituted with their "real" contents after a design is submitted to foundry for fabrication. You may get warnings in the CIW window during opening of a standard cell. Ignore them if the cell opens.

You may also get a "What's new" information window that you can happily ignore and close.

You need to create your own library from the Library Manager as follows:

- Select File>New>Library in the LM window.
- In the pop-up, enter your <*libname*>, then click OK.
- The next pop-up asks about Technology. Select "*Attach to an existing techfile*"; Select cmosp18fix3
- Select View>Refresh in the LM, and make sure your library is there.

3. Designing Schematics

Virtuoso is a complete design suite, that support the user in every step of the circuit design. We should keep in mind that Virtuoso is a platform of reference also for Analog design, where the complexity of the circuits and the criticality of the Spice level design is way higher than in the case of digital cells.
So, it is possible and very often convenient to edit a Spice schematic in Virtuoso. This is done for two reasons:

1) The schematic can be used as a reference during layout to drive the layout implementation
2) At the end of the layout activity, a schematic will be extracted from the layout. Then, a specific utility called LVS will be used to compare the original schematic with the one extracted from the layout. This will ensure that we did not introduce any functional error in our layout

This section will guide you in designing an inverter cell schematic:

Select your library name in the LM. From the Library Manager window type

- File>New>Cell View
  - Library Name = <Your Library Name>
  - Cell Name = <Your Cell Name>
  - View Name = schematic
  - Tool = Composer - Schematic

You should get a new window for drawing your schematic:

Basic Navigation rules (Valid for Schematic AND Layout Editing)

- ***In all Virtuoso tools, Type esc any time you need to exit from a specific functionality and go back to the main window.***
- ***You can zoom in a selected area with right mouse button. You can zoom out typing SHFT-Z***
- ***You can zoomout back to full image typing 'f' (Fit).***

- ***To Move an instance (or a layout feature) select it and press 'm'. (It may take a while, be patient) and you will get a pop-up window guiding you***
- ***To Copy an instance (or a layout feature) press 'c' and use the same methodology as in "move"***
- ***To delete a wire/instance or layout feature select it and press the delete button***

Adding Transistor Instances

From that window we start adding our inverter components, starting from the Pull-Down PMOS

- Type 'I' (Or Add>Instance from the GUI)
- Browse for library cmosp18 (**not** cmosp18fix3);
- Select pfet.

  - In the Add Instance pop-up set View = symbol;
  - Choose a Spice name for the PMOS if you like, otherwise the tool will chose a name for you (*It is the spice name of a mosfet, it needs to start with M*)

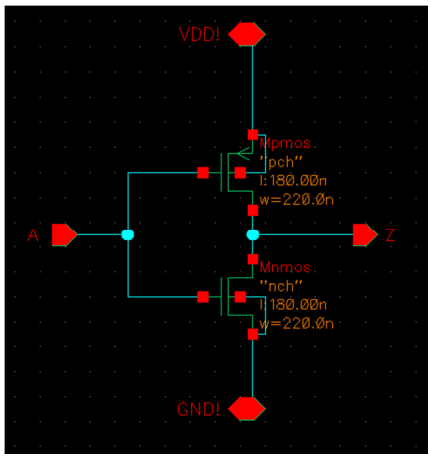
- Set the W Parameter (We are not changing L in this tutorial, and we do not need to change any of the other physical parameters). They represent the chosen physical parameters of the transistor, that we can set to get the desired behavior and that we will have to match when we move on to layout. If during the layout we realize we can not physically fit the W/L we chose, we can always go back to the schematic and fix it.

- Move mouse over drawing window and click button to place your PMOS transistor. Clock Right mouse button to rotate it if needed

If you want to exit from the "instance" menu (or from any other functionality menu) type ESC. Be patient, the tool can be at times a little slow.
We can repeat same procedure for NMOS setting Cell = nfet; Place it, set the desired width, then press Esc key to exit the "Add>Instance" functionality.

Building wire connections

Note when you move the mouse over a red square (Pin) it gets highlighted (also when mouse is over the transistor symbol). To design connections between instances, press 'w' (wire). Move the Add Wire pop-up out of the way (if necessary) by dragging with your mouse or hide it with the relative button, then click on the source and destination pin for your wire connection. Draw all necessary connections to form the inverter.



Adding Pins and Power Rail Connections

Pins are added very similarly to instances. Then specific Wires must be designed to connect the pins to appropriate instances. Power rail connections are added using specific pin names "VDD!" and GND!", and power rail pin directions are specified as InputOutput.

- Press 'CTRL-p' or from the GUI menu Add>Pin
  - Pin Name=vdd!; Direction=inputOutput

  - Add>Pin
  - Pin Name=gnd!; Direction=inputOutput

  - Add>Pin
  - Pin Name=In; Direction=input

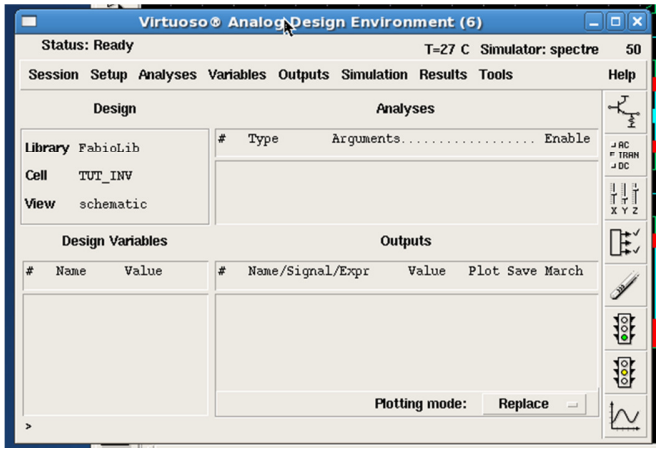
- Add>Pin
- Pin Name=Out; Direction=Output

To save your schematic use Design>Check and Save. Results of the check will appear in the command window. Keep fixing eventual errors until you have a clean check in the CW log

4. Simulating the Schematic Netlist

We want to simulate the schematic we just wrote in order to verify its behavior both from the functional and electrical point of view.

On the Virtuoso GUI click Tools -> Analog Environment



You will get a simulation Window, choose Setup -> Simulator/Directory/Host

- Choose Hspice-S as reference simulator
- Choose a project directory that you like (you can use the default, or more appropriately build a simulation directory in your CDS work directory)

Be patient, the initialization of the environment takes a while. From the Analog Environment window you can then perform Simulation->Netlist-> Create Final .

Note that you are getting only the circuit Hspice netlist, not the stimuli or the initialization. It is possible to perform the whole spice simulation in the Virtuoso environment, although this is not described in the current tutorial. In this case, we can simply paste the ASCII netlist we just generated in the HSPICE environment of the HSPICE Tutorial: please open a separate terminal window to do that, in order to avoid confusion with the cadence environment.

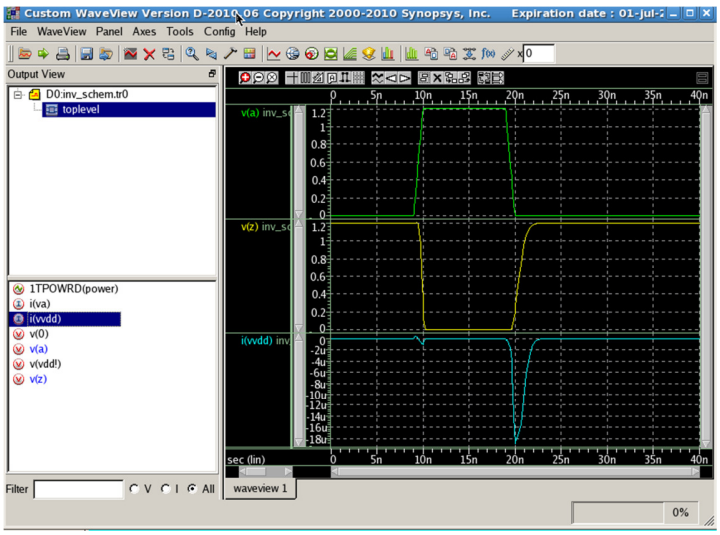
Note that the ASCII file containing the netlist is located in the path described on the top of the netlist window *(<simulation folder>/<cell name>/hspiceS/schematic/netlist/hspiceFinal)*

To simulate the netlist you can use the example available in /ensc/fac1/fcampi/Tutorials/Layout/hspice/inv_schem.cir

The tutorial example displays the schematic waveforms and measures Propagation delay and transition times for the schematic we just built.

If we are satisfied with the behavior of the schematic we designed, we can move on to Layout. Note that, while the circuit evaluation can and should be done using Hspice at ASCII level, we are still supposed to define a schematic of our circuit in order to support automated LVS. There would be ways to define the circuit at ASCII level and use it for LVS, but in the context of our course we will stick to the standard methodology of using the GUI schematic editor.

When you are done with the Analog Design Environment close the window (Session->Quit), as it takes lots of memory space and may slow down the following design steps.



5. Layout Design

Building a circuit from scratch is very complicated and time-consuming: even for such a small cell as an inverter, that will take you a lot of time and interaction with the tool. Try not to perform everything in one session, and not to rush through all the step in a hurry, else you will never be able to complete the activity

You can create an empty layout view for your schematic with

- File>New>Cell View
  - Library Name = <You Library Name>
  - Cell Name = <Your Cell Name>
  - View Name = Virtuoso

Basic Navigation rules (Valid for Schematic AND Layout Editing)

- ***In all Virtuoso tools, Type esc any time you need to exit from a specific functionality and go back to the main window.***
- ***You can zoom in with right mouse button. You can zoom out typing SHFT-Z***
- ***If you zoom down too much you can go back to full image typing 'f' (Fit).***

- ***To Move an instance (or a layout feature) select it and press 'm'. (It may take a while, be patient) and you will get a pop-up window guiding you***
- ***To Copy an instance (or a layout feature) press 'c' and use the same methodology as in "move"***
- ***To delete a wire/instance or layout feature select it and press the delete button***

After opening a layout, even an "empty" one, you will see the "Layer Selection Window" (LSW) showing all technology layer names and purpose layers. This LSW is a fundamental resource for the designer: when performing layout, we are rendering a 3Dimensional plot in 2 dimensions, so all physical and virtual layers of our layout will overlap in the Virtuoso GUI. Often, to make sense of what we are inspecting, we want to remove some of the uninteresting layers to really focus on what we need to check/Draw/Correct. LSW allows us to make some layers invisible or un-selectable thus greatly enhancing our inspection.

The big challenge in designing a layout is to be compliant to all design rules, and there are quite a few of them. In order to make this procedure simpler, it is advisable to design step by step and check every structure incrementally, rather than designing the entire circuit and checking it afterwards, as if the design is too advanced it may be impossible to fix all DRC errors. In the following, you will be guided to the full design of the inverter and Design Rules will be introduced step-by-step and highlighted in Purple.

The TSMC 0.180 um Design rule manual is available at

/ensc/Lnx_STC/kits/cmosp18/doc/CMOSP18designRulesLogic.pdf

In order to set your design environment, with the menu Options-> Display make sure that Pin Names are ON and the grid control X/Y Snaps are .01 (Which means that the max resolution of your cursor will be 10 nm, that is 2 times the reference grid step in this technology).

First of all, we know that, in order to be suitable for Place & Route, our circuits must be "abbuttable", which means that :

1) **the height of the cell must not be changed (6.5 u in our case)**
2) **Power rails must touch both sides of the cell and have the same width. (Power rails are .9 u wide in our case)**

These are not real design rules (and are not checked by DRC) but they must be respected as they are imposed by the fact that being an inverter our circuit must be part of a Standard Cell library

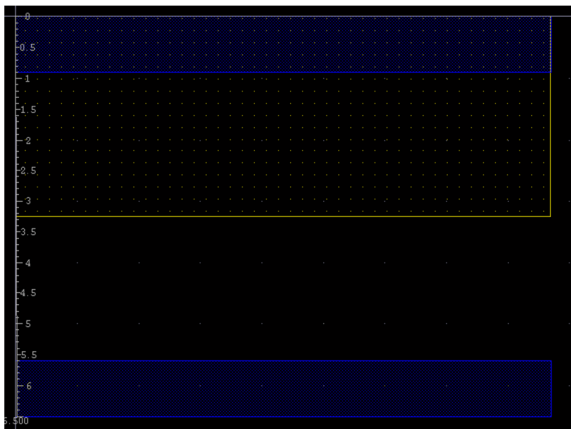
- ***In order to measure a distance select the starting point, press 'k' (Ruler) and stretch it to the target point. Clock on the target point if you want the ruler to stick, you can remove all rulers if you like with 'shft-K'***
- ***To build a rectangle shape on the layout choose the appropriate layer on the LSW, place the cursor in the desired starting point, and press 'r' (Rectangle). You do not need to specify a net name. Keep an eye on the dimensions of the rectangle you are drawing described by dX and dY on the top of the layout drawing widow***
- ***In case you want to stretch an existing rectangle (this happens very often when you try to match Design Rules) you need to select not a whole polygon but only one side of it. This feature is activated by the key "F4". Press "F4" until you see in the command window the display "geTogglePartialSelect t". "geTogglePartialSelect nil" meas that the feature is deactivated. Then press 's' and pull the side of the rectangle to the desired position***
- ***In case you want to know the layer a given polygon belongs to you can select the polygin with left mouse, and use the 'q' (Query) command. With the same command you can also change the layer of a polygon, this is quite useful if copy-pasting an nMOS into a pMOS and viceversa, or a Nwell-tap into a Pwell-tap) and viceversa***
- ***If you want to see/select a layer that is buried under other polygons USE THE LSW. LSW allows you to visualize, or select, only one or a subset of layers***

So we can start by Designing the two power rails at the require height, we have no constraints on the width except to make it as small as possible to save are, so we will stretch it or reduce it later on depending on our needs. It is advisable to draw a cursor to keep an eye on the distances.

You can build power lines selecting the METAL-1 layer on the LSW (Layer selection window)

Note: Each time you are into a functionality (such as 'm' move, 'c' copy, 'r' rectangle 's' stretch or 'k' ruler) you get an annoying pop-up window that helps you with the command. In particular, you can choose the direction of your move/copy/stretch operations (Only up/down or every angle). Be patient, at times it takes a few tens of seconds for it to pop-up: don't click around while you are waiting. When it eventually pops-up, you can move it or hide it if it gets in the way. Press ESC to exit from the specific functionality. Try to be aware of what functionality you are in all the time, and be patient as the tool can be slow at times.

You can undo every action with 'u'.

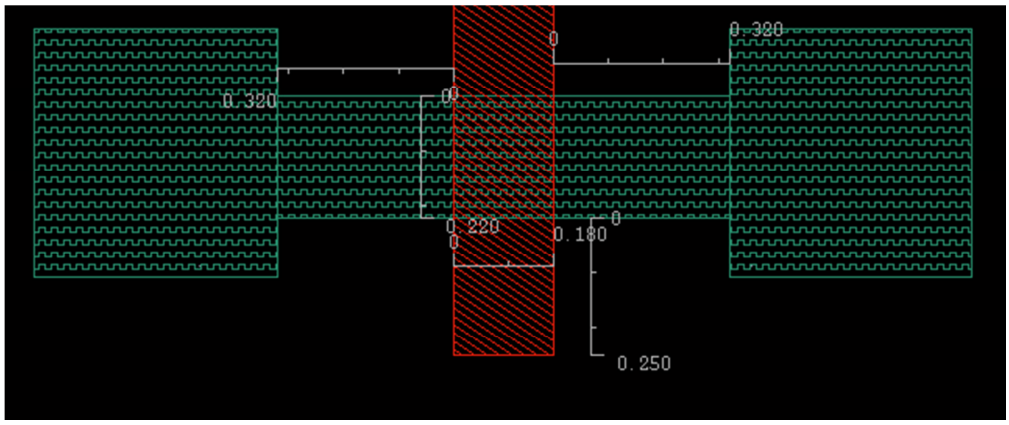


We can also build our n-well rectangle covering half of the cell in height.

After this preparatory setup, we can focus on designing our transistors: we will start with the nMOS. Transistors design involves respecting several design rules.

The nMOS transistor is composed by a L=0.18 u Polysilicon line crossing an active diffusion area of width W. Active diffusion is called "active" in Virtuoso and OD in design rules. The Active region must be fully immersed in N+ diffusion

We could design our transistor with horizontal poly and vertical diffusion or vice-versa. In this tutorial we will use horizontal diffusion to be compliant with stick diagrams introduced in class:



TRANSISTOR Rules: Poly/Active

Minimum Gate Length: 0.180 u [Note: In some libraries, to decrease leakage, it is possible to have larger Gate L. In our lab activity, we will stick to unity Length transistors]

Minimum Gate Width: 0.220 u ;

Poly endcap of gate .250;

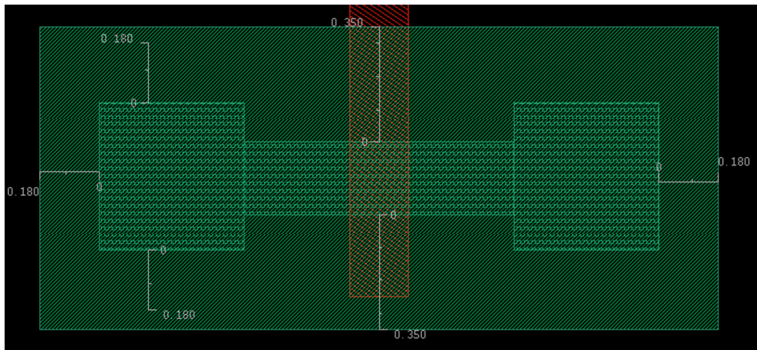
Active min area >= .2025u^2 ; Active overhang of gate >= .320u; Active to Active >= 280u;

The following rules must be taken into account only in case of active geometries (such as the active box around the contact) that are larger than the active with under the gate

Active to Gate >= .160u ; Active to poly >= .100u

Transistor Rules: N Diffusion

Nplus extension over active >= .180u ; Nplus extension over channel >= .350u



It is advisable to run DRC now (*See following section for a guide to DRC*) if we haven't done it before, in order to check if we haven't caused any errors so far. Since our design is incomplete, we should get a few errors which we can ignore for the moment, ***but the rules introduced so far should not be violated.***

The rules we can violate now are:

1) Need substrate contact within 5u of nMOS [we don't have substrate Tap yet]
2) 1 floating poly as we did not add contacts so far

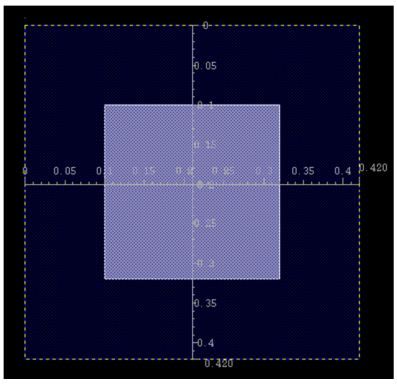
LAYER / Contact Rules

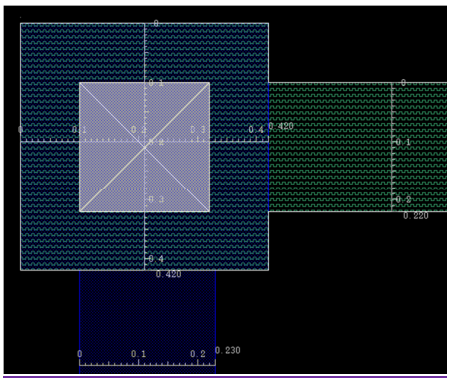
We must now contact the S / D / G / B terminal of our transistor: we will need to contact them all to M1, in this standard cell library we are required to have all pins in Metal1.

For each of the basic layers, we have specifications in terms of Width, Box, and Spacing. Box refers to the space that hosts a contact, Width refers to un-contacted polygons, and spacing is the distance between two parallel polygons of the same layer:

Note that contacts are used for communication between Active regions and Poly or M1

Contact to Gate >= .160u ; Contact to Contact >= .470u





Contact Box >= .220u ; Contact Overlap of Active/Poly/Metal1 >= .100 u;

Metal1 Box >= .420u ; Metal1 Width >= .230u ; Metal1 Spacing >= .230u ;

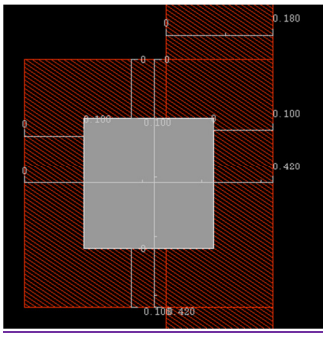
Active Box >= .420u ; Active Width >= .220u ;

Poly line & Poly Box:

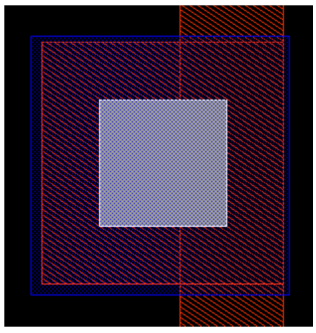
The gate is composed by Polysilicon. In order to be contacted to the inverter input pin the poly line must feature a Poly box and a Contact. In order to be considered a valid pin, in this technology, the gate must be contacted with a Metal1 Box. Please note that a single Metal1 Box must have area >.2025 so must have side >= 450u.

Poly Box >= .420 u ; Poly Width >= .180u ; Poly Spacing >= .250u (Gate to Gate Distance) ;

Min Metal1 Area >= .2025u^2 [sqrt(.2025)=.450u]

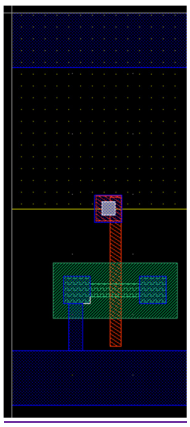


=> => =>



The final stdcell with the complete nMOS layout looks like the following picture.

Since we don't have specific constraints, we can enjoy symmetry and place the gate/poly contact box in the very middle of the cell. That is not necessary, if we were constrained in terms of area we could place it anywhere it doesn't violate specific DRC rules.



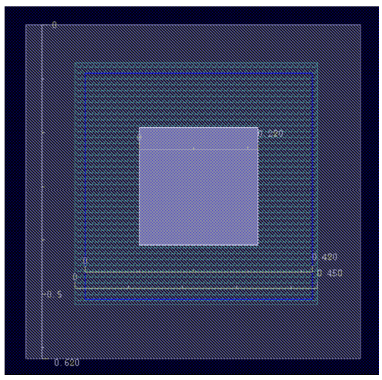
If we were to run DRC now, we would see one error, related to the minimum distance between a MOS and the nearest Well/Substrate Tap. This is imposed in order to avoid Latch-Up and guarantee an uniform Body Biasing.

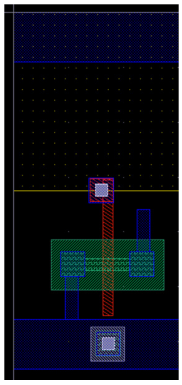
MOS to Well Tap spacing < 5u

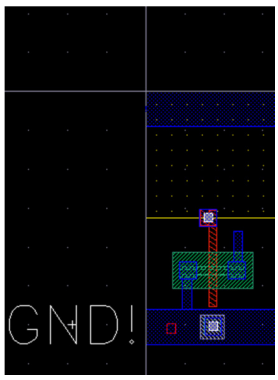
If we were using an optimized StdCell library with separate Tap cells, we could ignore this rule. In our case, we will try to satisfy it adding a Substrate Tap. A P-substrate Tap bust be composed by the stack (Pile of layers) N-diff + Active + Contact + Metal1 (Gnd). The well must respect all the M1 / Active rules already applied for the transitor contacts. Moreover, being the tap a small diffusion area, we must make sure to respect the minimum Active box area [Abx] .2025 u^2 and Metal1 box area [Mbx] .2025 u^2 rules already introduced above.

On top of that, for well taps, we need to ensure the following

Pplus/Nplus Diffusion min width .440u ; Pplus/Nplus Diffusion min area .3844u^2 [sqrt(.3844)=.620u]







Building the P+ / Active / Contact / Metal1 structure above, DRC is still not happy because the ground stripe is still not marked as ground. We can clean this last DRC issue adding a ground pin on the GND stripe:

From the GUI type Create-> Pin (or CTRL-p), Locate the pin on the stripe add place the name nearby (Pin and pin name are not real layers, are called “Purpose layers” as they exist only in the CAD environment to Drive DRC and LVS.

Pins refer to a specific layer. In this specific design, make sure to select the Metal1 Layer before placing pins, as all pins must be in Metal1.

Note that Vdd and Ground pins must have the conventional names VDD! and GND! . Vdd and Gnd pins must be defined as input/output.

It may be handy to force the display of the pin names on the layout choosing from the Virtuoso GUI Options -> Display and ticking "Display Controls -> Pin Names", and ticking if necessary Display pin Name in the pin creation pop up menu.

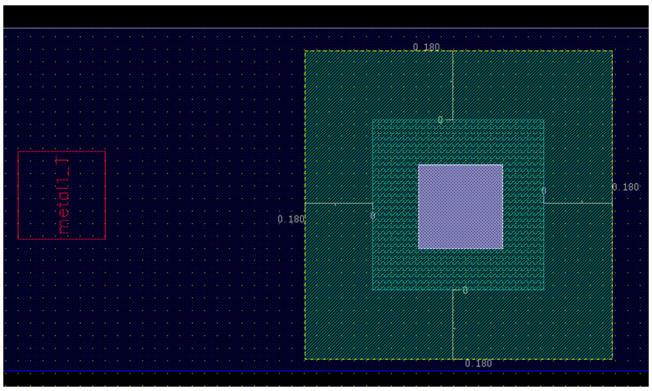
We have now a DRC free nMOS transistor. We can build a similar pMOS transistor following the same rules. Since we spent quite a bit of effort in building DRC-clean geometries for the nMOS, we should reuse the effort as much as possible in the pMOS, copy-pasting most of the polygons and only changing diffusion features.

We can copy any stack of polygons selecting them all together and applying, as usual, the 'c' (Copy) command. We can change the layer of a polygon selecting it and using the command 'q' (Query).

Although it makes sense to place the n+ well tap for anchoring the pMOS substrate to Vdd under the Vdd rail line, this poses an additional challenge:

Nplus extension over p-Active >= .180 u when –nplus distance to Nwell boundary < .430u

Hence, we need to extend the Nwell tap N+ box to extent .180u over the active region.

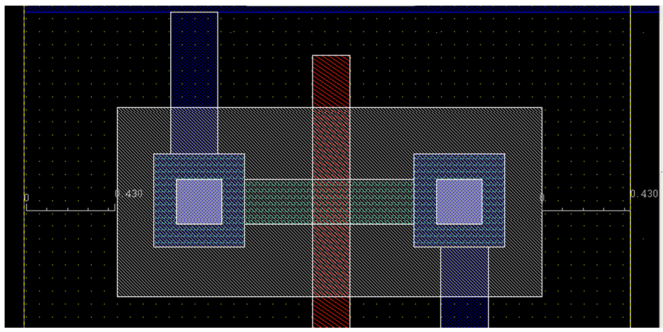


All the pMOS is 100% equivalent to the nMOS. In this tutorial we will build it with Wp = Wn = Min Active Width = Wm#, but this will obviously lead to asymmetrical behavior of the Inverter between Pull-Up and Pull-Down.

We must make sure to respect the minimum distance between P+ diffusion and Nwell:

Nwell overlap of Pplus >= .430u

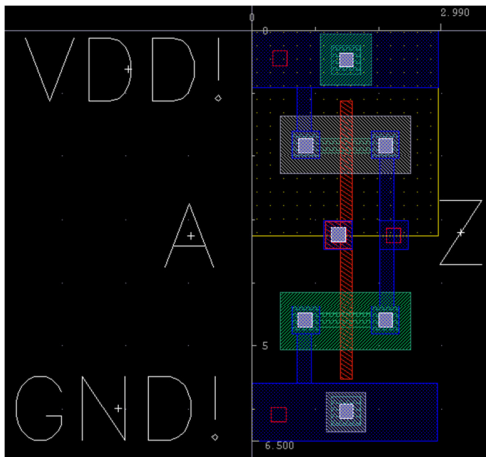
This rule will also determine the width of our inverter (This would not be the case if we decide to orientate our MOS vertically rather than horizontally).



When the pMOS is also complete and DRC-free, we need to add the signal pins A (input) and Z (output). Pins must be accommodated on metal1 boxes (remember that as introduced previously a Metal1 polygon must have area>=.2025 u^2. If we build a square box, the side must be >=.45u .

Make sure to choose correctly Input/Output direction for A/Z, and to place all pins in Metal1, (Including A, that is then carried down to polysilicon by a specific contact). The pin direction and its layer are selected in the Create Pin Pop-Up menu. Note that the pin layer is different from the wire layer: Pin layer is Metal1_T (EDA purpose layer), while wires are Metal1-Drawing (real Process layer).

If you choose the wrong terminal name, or the wrong direction (You may realize that later during LVS) you can fix that selecting the pin, pressing ‘q’ (query), and choosing the “connectivity” menu in the pop-up window.



5. DRC (Design Rule Check)

After we designed a custom circuit layout (or a portion of the same), we need to make sure that our implementation does not violate design rules. This is done by the utility *design-rule-checking* (DRC). We ***must*** run DRC on any layout, to make sure that the layout complies to the rules of the technology. If DRC is violated

1) The Mask may not be able to provide the expected design, so there must be unplanned shorts or opens
2) The Spice Models could be inaccurate, thus making our simulation unreliable. If the SPICE simulation is unreliable, then the STA on our synthesis tools will be unreliable, so our IC will not work due to Setup or Hold violation that we don't see in the netlist. Such violations are induced by DRC errors, and are caused by misalignment between the real silicon shapes and the transistor behavior modeled in spice.

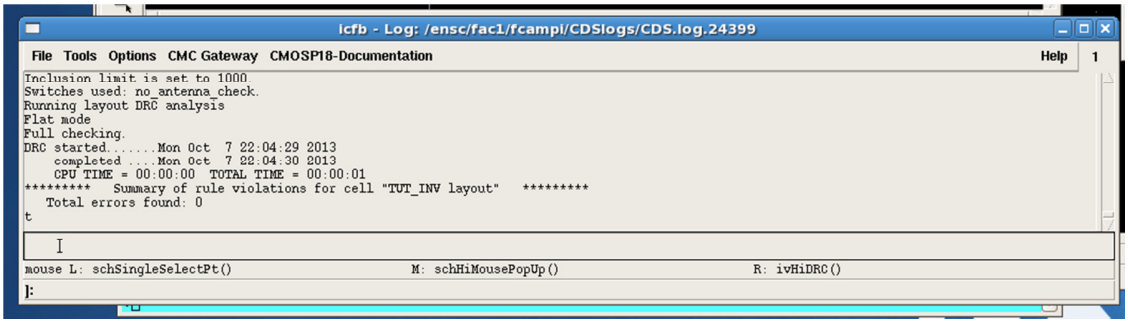
To perform DRC open the layout cell view in the Virtuoso GUI then:

- Verify-> DRC
    - Checking Method: flat
    - Checking limit: full
    - Switch Names: no_antenna_check (browse via Set Switches and select, OK)
    - Join Nets with same name: On
    - Echo commands: Off

Your layout window will now show annoying flashing violation markers describing the area that represent DRC violations. The Command window will also describe a textual report of the DRC, that will likely be more readable and easier to handle.

- Verify>Markers>Explain (or Find, Delete, etc). If you select Explain, then LMC on the marker, you will get more info about the error in a pop-up window. After Verify>Markers>Find, you can use "Next" to help you cycle through all the markers (in this case there should be only one).
- Verify>Markers>Delete All gets rid of the flashing markers. They get saved if you do a Design>Save

It is very hard to complete a layout resolving all DRC violation, so it is recommended that you run DRC checks often during your design iterating DRC checks as you make your way through your design implementation.



6. LVS (Layout Versus Schematic)

The Layout of the standard cell is a fundamental information for digital design:

1) it specifies the shape of the standard cells as well as all pin positions, that will be used by automated P&R tools to perform connections during placing/routing
2) From the layout, we can extract a new spice circuit for the cell containing the real physical parameters of the layout. Since the layout has been drawn starting form a Spice circuit
    a. We need to check the functional equivalence between the starting circuit and the extracted one to make sure we actually accomplished our design task
    b. From the extracted circuit we can have a way more precise estimation of the circuit parameters that can allow detailed simulations

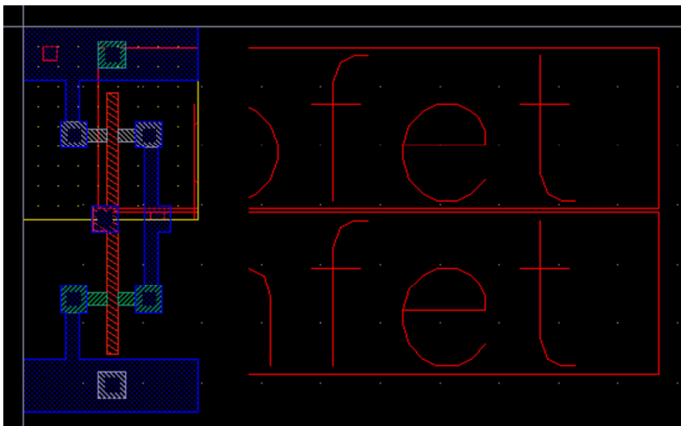
The step of extracting a circuit schematic from a layout and comparing it against the reference schematic used to pilot the layout is called LVS. LVS is an essential mandatory step in the design of integrated circuits!

Extracting Schematic from Layout

From the layout window, we can extract a layout schematic for LVS:

- Verify>Extract (if you don't see Verify, first select Tools > Layout)
- In the Extractor pop-up, choose:
    - Extract Method "flat"
    - turn on "Join Nets With Same Name"
    - turn off "Echo Commands"

Shortly, you should see "Total Errors found: 0" in the CIW window, and see a new view in your library called "extracted" that represents the structures extracted from the layout. You can check visually if it makes sense to you.



LVS

Note: It is possible to run Schematic extraction and LVS even if DRC rules are not met. But, unless you really know what you are doing, it is not advisable as you are extracting a layout that is meaningless, and is not manufacturable, since it does not respect design rules. Most foundries will perform a DRC on the design that you send for fabrication, and they will refuse to build silicon starting from layout that violate DRC. On the contrary, foundries do not perform LVS before fabrication, as they don't have visibility, and do not want to have visibility, of the schematic circuit you started your design from.

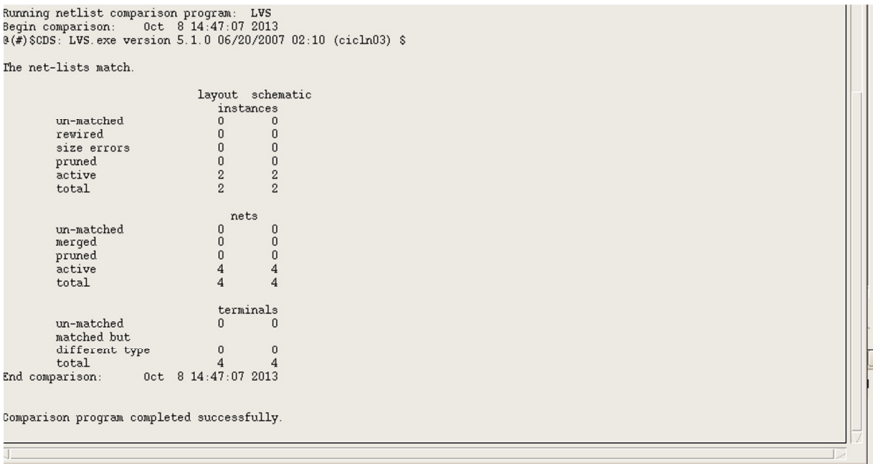
Open the extracted view of <Your CellName> and select Verify>LVS ; You'll get the Artist LVS pop-up window:

- Create Netlist: "schematic" and "extracted" (both selected)
- Library: <Your Libname> for both
- Cell: <Your CellName> for both.
- View: type schematic on left and extracted on right. Watch for typos
- LVS Options: none should be selected, in particular switch off Rewiring and Terminals if they are active

Click "RUN": The CIW log will say "LVS job has started". After a moment, you should get a pop-up that says ".....has succeeded". *This does not mean that LVS matched [You wish!], only that the tool has finished its analysis.*

In order to see details of the run in case of mismatch Select "Info" on the LVS pop-up:

1) If you choose "Log File", you will see a full ASCII log of the LVS activity. ***<u>The magic statement you are looking for is "The net-lists match" (scroll down).</u>*** After that you get a summary of details about your schematics. If the net-lists don't match, something went wrong. Browse the errors, most of the time it will be a missing pin in the layout, a wire not connected properly in the schematic, or something silly related to the naming of the pins or their direction such as "Terminal XZY type in the schematic : input, in the layout InputOutput". Fix the errors (If they are in the layout you will have to run extraction again!)
2) If you choose "Netlist", you can generate a ASCII spice netlist (NOT HSPICE COMPATIBLE!) of both the schematic and the extracted view. In case you are at loss in understanding the reason for LVS mismatch you can try to manually compare the two in order to understand the differences.



```
Running netlist comparison program:  LVS
Begin comparison:     Oct  8 14:47:07 2013
@(#)$CDS: LVS.exe version 5.1.0 06/20/2007 02:10 (cicln03) $

The net-lists match.

                              layout  schematic
                                 instances
        un-matched               0       0
        rewired                  0       0
        size errors              0       0
        pruned                   0       0
        active                   2       2
        total                    2       2

                                   nets
        un-matched               0       0
        merged                   0       0
        pruned                   0       0
        active                   4       4
        total                    4       4

                                 terminals
        un-matched               0       0
        matched but
        different type           0       0
        total                    4       4
End comparison:       Oct  8 14:47:07 2013


Comparison program completed successfully.
```

7. Post Layout Simulation

After having completed a DRC and LVS free layout, we have a valid cell design. Now that we have an extracted schematic, it is useful to simulate it and compare the results against the Schematic version. Especially for circuits a bit more complex than our inverter, it would be good to verify with the physical parameters of the layout if anything has changed in the functional electrical behavior.

To generate the post-layout schematic, we apply the same procedure we used previously but on the extracted view:

1. Open the extracted view from the library manager
2. Open Tools-> Analog Environment
3. On the Analog Environment pop-up window Setup -> Simulator/Directory/Host
4. On the pop-up select your <project simulation directory> and choose hspice-S as simulator
5. On the pop-up select Simulation->Netlist->Create Final
6. Close the Analog Environment (Session->Quit)

Then on the terminal reserved to hspice, open up the extracted netlist file <Your project simulation dir>/simulation/<your cell name>/hspices/extracted/netlist/hspiceFinal, copy and paste the circuit netlist into a spice deck .

Use the file */ensc/fac1/fcampi/Tutorials/Layout/hspice_inv.cir* for reference, pasting in that deck the netlist obtained from your environment. Depending on the names you gave to your cells/terminal, you may have to adapt the file to your needs. Upon simulation, we can see that, although the schematic and extraction yield very similar results, even for such a small circuit as the inverter there are small differences that can be appreciated: using the .measure statements we can quantify differences induced by the layout on the schematic.