CAD 2: Inverter Standard Cell

EE 476 | University of Washington

Notes

1. Where students are required to submit circuits netlists, circuit pin names **should be named as they appear in each Part**. Students can look at the provided `specifications.json` file, which lists the names of the circuits and their associated pins, in addition to running the checker script that is provided.

2. Remember to add the path to this CAD's library at the top of the speicifcations.json file.

3. Your design library must be completely self-contained. I.E. When starting a new library, make sure that you copy all cells being used in that library. You must NOT be accessing cells from other design libraries. An easy way to test you have done this correctly is to:

    a. Comment out (using #) the lines corresponding to other CADs in your cds.lib file that can be found in your cadence directory.

    b. Click View -> Refresh in your library manager

    c. Running check and save on your schematic and making sure there are no broken references.

4. Layout will be graded only if the previous two instructions are followed.

5. The document for this CAD is significantly longer than normal, since there is an integrated tutorial, and more detail in the directions given. Later CAD assignments assume students are more independent, with the documents between 4-7 pages in length.

Setup

Use the same general directory structure as in CAD 1 (also described in *Tutorial 1*).

CAD 2 Overview

CAD 2 involves the design and implementation of two inverter standard cells. While CAD 1 explored the characteristics of MOSFET devices, CAD 2 takes the next step and asks students to build a basic logic circuit. As in CAD 1, this CAD asks students to build and characterize their circuits with standard industry tools (Cadence Virtuoso, HSPICE, etc.). In addition to schematic-level design, students are introduced to physical layout, and simulation of layout-level circuits.

1 Standard Cells as Building Blocks

A *standard cell* is a digital logic gate that is designed to be reused within and across different designs (Figure 1). A group of standard cells is called a *standard cell library*. Typically, a standard cell library is used in conjunction with logic synthesis tools to automatically pick logic cells and perform the physical layout of cells and connections. In this way, a standard cell can be thought of as an atomic building block for a digital design. While students will not use synthesis tools in EE476, standard cells are still useful for custom layout (layout done by hand).

In order for a design to be composed of standard cells, the cells must be compatible with each other. While it is too early to describe what "compatibility" means in precise terms, in general it means that cells should be able to be laid side-by-side, and tiled vertically (which involves flipping the cell to make the horizontal Vdd or Vss trunks stack on top of each other. Note that the contacts also have to overlap perfectly to avoid contact spacing DRC errors - more on this later), without violating any design rules. Throughout the layout portion of this CAD, pay attention to the spacing and symmetry of the different mask layers and try to figure out what dimensions are (or are not) important for cell compatibility.

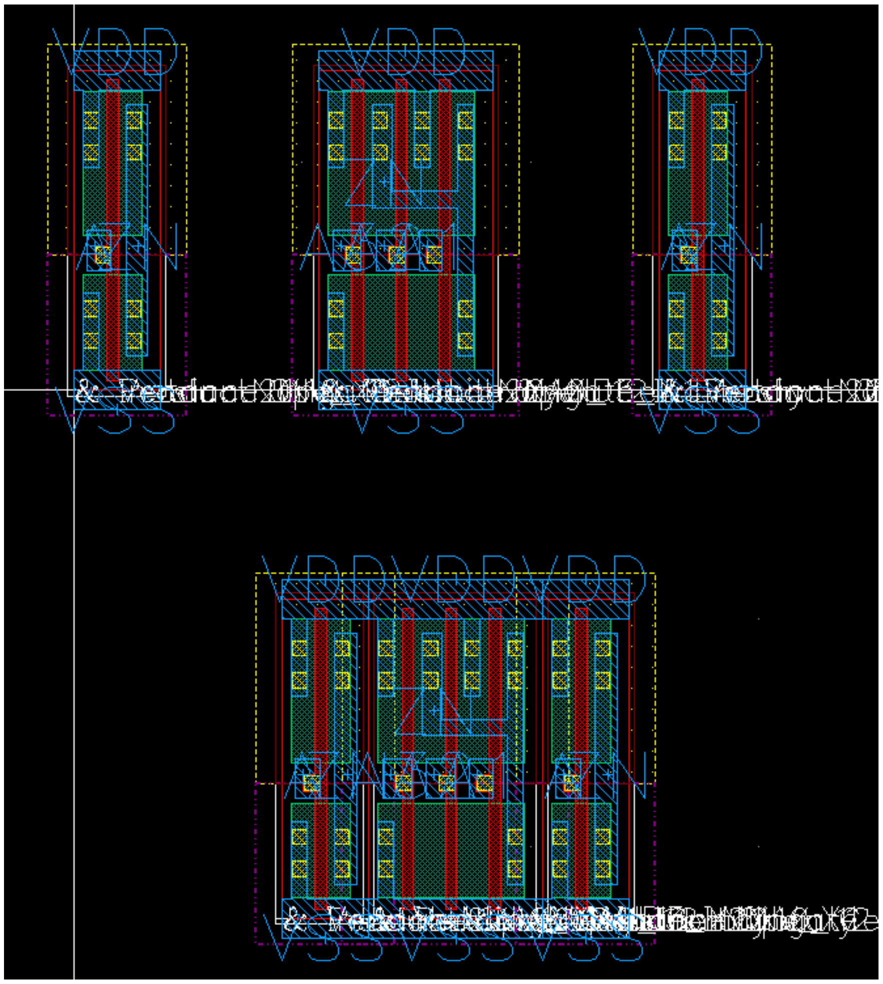


Figure 1: An inverter, 3-input AND, and another inverter, placed separately to see individual standard cells (top row), and side by side as they would typically be used (bottom row).

2 Schematic-Level Inverter Creation and Test

a. Create the circuit from Figure 2 in Virtuoso, and call it `INVD1`. The PMOS device has L/W = 50nm/0.4um and the NMOS device has L/W = 50nm/0.3um . In order to use this schematic in other circuits, we must also create a "schematic symbol" - students should create a symbol similar to that in Figure 3. The capacitor can be found in `analogLib` under the name `cap`. As with all CADs in this class, the body connection for all PMOS and NMOS devices is vdd! and vss! respectively.

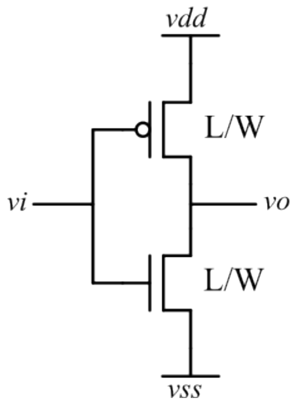


Figure 2

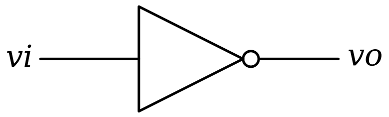


Figure 3

Once the inverter symbol has been created, it can be instantiated in other schematic designs. With the newly created symbol, create a new schematic called `loaded_inverter`, and build the circuit in Figure 4. For this CAD, it is **critical** that the NMOS uses `VSS`, and the capacitor load uses `gnd`[1].

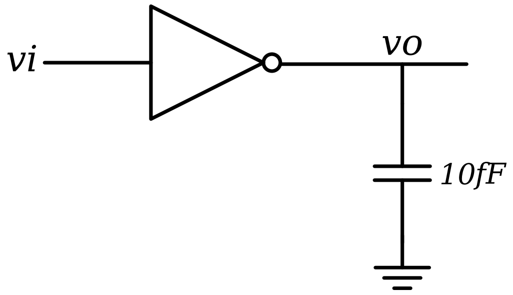


Figure 4

b. Generate the netlist for the `loaded_inverter` circuit, and use HSPICE to collect the measurements in Table 1 **at both VDD = 0.8V and VDD = 1.2V** (the HSPICE `SWEEP` command is useful here). For this part and the rest of the CAD, the input signal should switch from 0V to VDD. This is the only part in this CAD that requires a VDD other than 1.2V.

[1]The reason is that the grading scheme measures the capacitor current through `VSS` and `VDD`. If the capacitor also used `VSS`, the net current through `VSS` would be zero when the capacitor discharges through the NMOS.

Measurement	**Description**
`rise_delay`	Rise delay, $v_i = 50\%$ to $v_o = 50\%$ (a.k.a. propagation delay low-to-high)
`fall_delay`	Fall delay, $v_i = 50\%$ to $v_o = 50\%$ (a.k.a. propagation delay high-to-low)
`rise_time`	Rise time, $v_o = 20\%$ to $v_o = 80\%$
`fall_time`	Fall time, $v_o = 80\%$ to $v_o = 20\%$

Table 1

Note: As always, "rise" or "fall" in measurements like `rise_delay` indicate that the measurement is to be taken when the *output* is rising/falling.

Delivery

a. None (included in b).

b. The generated `loaded_inverter` netlist (named `loaded_inverter.ckt`), and the measurements in Table 1 (fill in the provided `specifications.json` file).

Integrated Tutorial: Physical Layout (a.k.a. Tutorial 3)

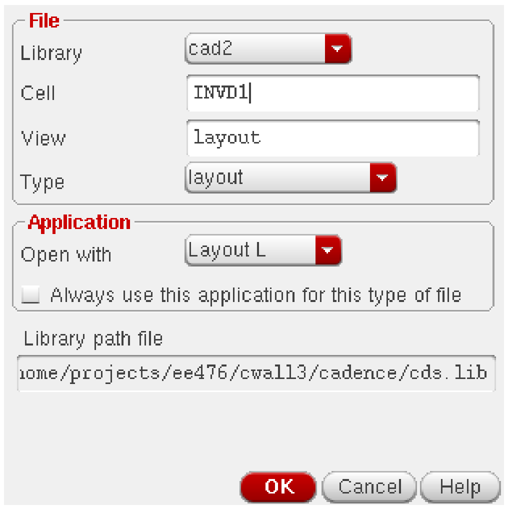
This tutorial walks through the process of creating a physical layout, performing design-rule checks (DRC)[2], and passing formal equivalence checks (layout versus schematic, or "LVS"). This tutorial also includes a section on how to extract circuit parasitics and generate a post-layout netlist.

One thing to note is that the layout created in the tutorial below has a cell-height of 1.4um. The cell-height is defined as the distance between the top edge of the top row of contacts, to the top edge of the bottom row of contacts (each row of contacts should be aligned). **In this and all future CAD's, standard cells should have a cell-height of 1.4um**.

Creating a layout view

In Cadence Library Manager, select the inverter design that was created earlier in this CAD (*not* the loaded inverter). Then select "File → New → Cell View"; the window that appears should look like that in Figure 5. In the dropdown menu labeled "Type", select "layout", and press "OK". The layout editor should open with a blank canvas, as in Figure 6.

[2]Chapter 3.3 of Weste & Harris provides a good overview of layout design rules



File

Library	cad2
Cell	INVD1
View	layout
Type	layout

Application

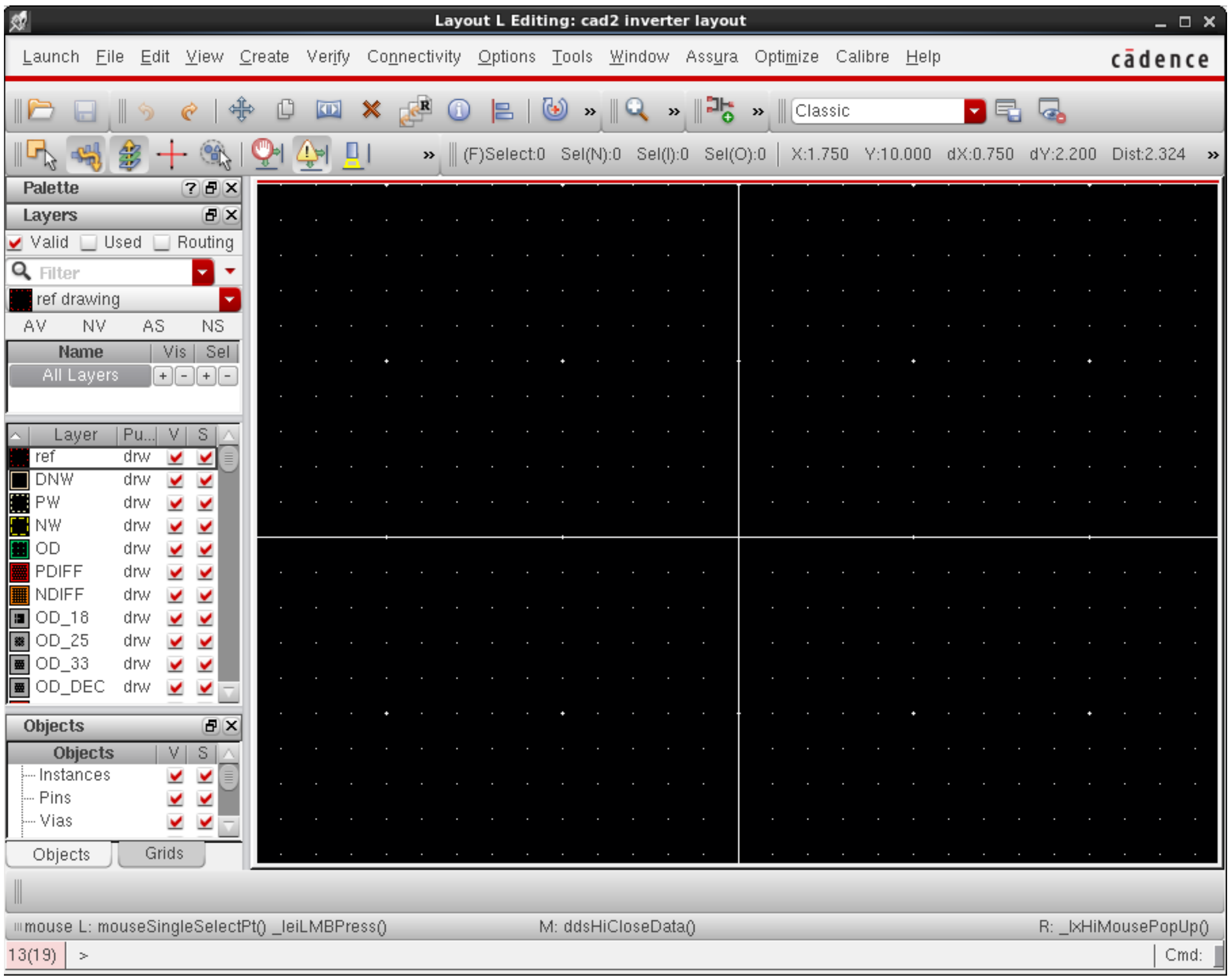
Open with	Layout L

☐ Always use this application for this type of file

Library path file

ıome/projects/ee476/cwall3/cadence/cds.lib

Figure 5



Layout L Editing: cad2 inverter layout

(F)Select:0 Sel(N):0 Sel(I):0 Sel(O):0 X:1.750 Y:10.000 dX:0.750 dY:2.200 Dist:2.324

✓ Valid ☐ Used ☐ Routing

Filter

ref drawing

AV NV AS NS

Name	Vis	Sel
All Layers	+ -	+ -

Layer	Pu...	V	S
ref	drw	✓	✓
DNW	drw	✓	✓
PW	drw	✓	✓
NW	drw	✓	✓
OD	drw	✓	✓
PDIFF	drw	✓	✓
NDIFF	drw	✓	✓
OD_18	drw	✓	✓
OD_25	drw	✓	✓
OD_33	drw	✓	✓
OD_DEC	drw	✓	✓

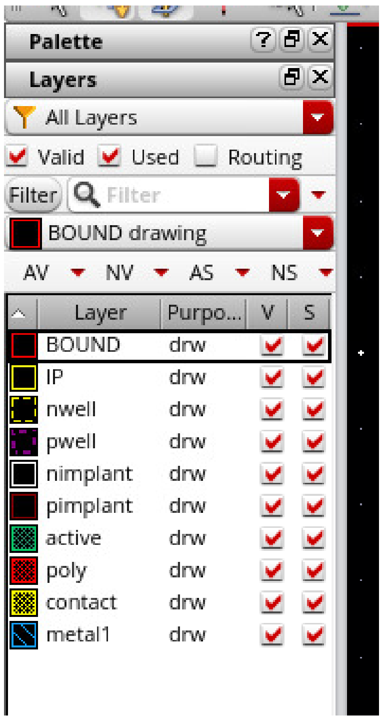
Objects	V	S
Instances	✓	✓
Pins	✓	✓
Vias	✓	✓

```
mouse L: mouseSingleSelectPt() _leiLMBPress()          M: ddsHiCloseData()          R: _lxHiMousePopUp()
13(19)  >                                                                              Cmd:
```

Figure 6

Layout Drawing

The inverter in this tutorial is defined by seven photolithographic mask layers[3]: n-well (NW), poly-silicon (PO), n+ diffusion (NP), p+ diffusion (PP), contact (CO), oxide diffusion (OD), and metal-layer 1 (M1) (Figure 7).



✓ Valid ✓ Used ☐ Routing

Layer	Purpo...	V	S
BOUND	drw	✓	✓
IP	drw	✓	✓
nwell	drw	✓	✓
pwell	drw	✓	✓
nimplant	drw	✓	✓
pimplant	drw	✓	✓
active	drw	✓	✓
poly	drw	✓	✓
contact	drw	✓	✓
metal1	drw	✓	✓

Figure 7

1. DRD Notify and DRD Enforce

DRD Notify and DRD Enforce are two features of Cadence Virtuoso that are generally helpful (but not necessary) during layout. Clicking the DRD Notify icon ( ) will enable graphical warnings in Virtuoso for when a design rule is violated. DRD Enforce ( ) is more restrictive than DRD Notify, and prevents the placement of shapes that violate design rules. Using one of these tools is recommended, since they can help prevent late changes in layout that would otherwise be very time consuming.

[3]Review the lecture notes if needed, Weste & Harris Ch.1.5 also presents an overview of CMOS device fabrication

2. Power straps and n-well

Use Metal-1 (metal1) to create power straps with width of 0.38um, height of 0.17, and with a pitch of 1.4um. The pitch refers to the distance between the top of the lower metal strap and the top of the higher metal strap. 1.4um is the pitch used by standard library, so for future cooperation between standard library and your design, please set pitch to 1.4um. Following the power straps, place a large region of n-well (width and height ~1.5um) as shown in Figure 8.

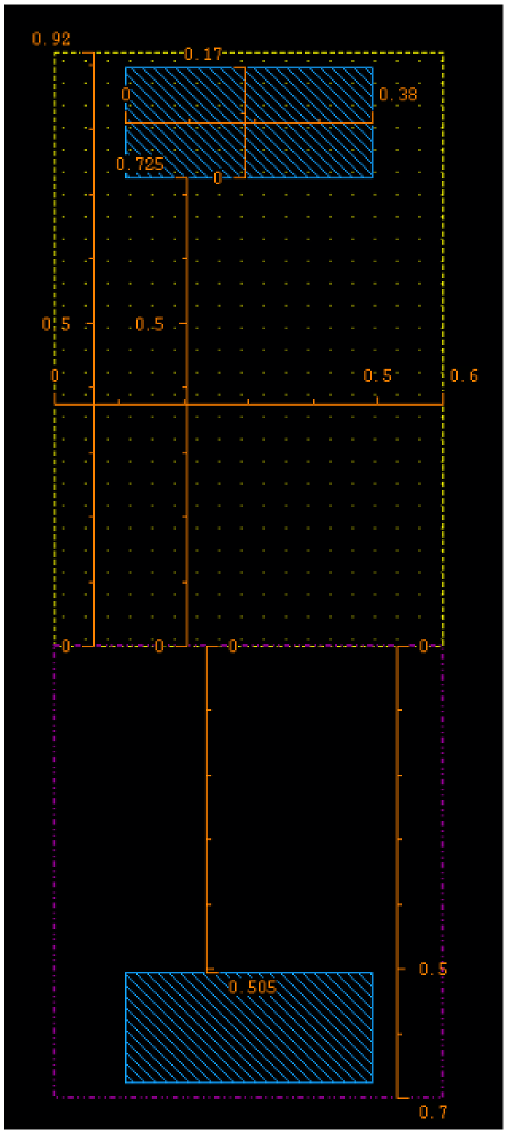


Figure 8

2. N-plus and p-plus

Draw the p-plus (pimplant) and n-plus (nimplant) regions as shown in Figure 9. These masks serve to distinguish whether the region exposed by the OD mask will be n-doped or p-doped. In order to meet min spacing design rules, the edges of adjacent NP and PP regions should touch but not overlap.

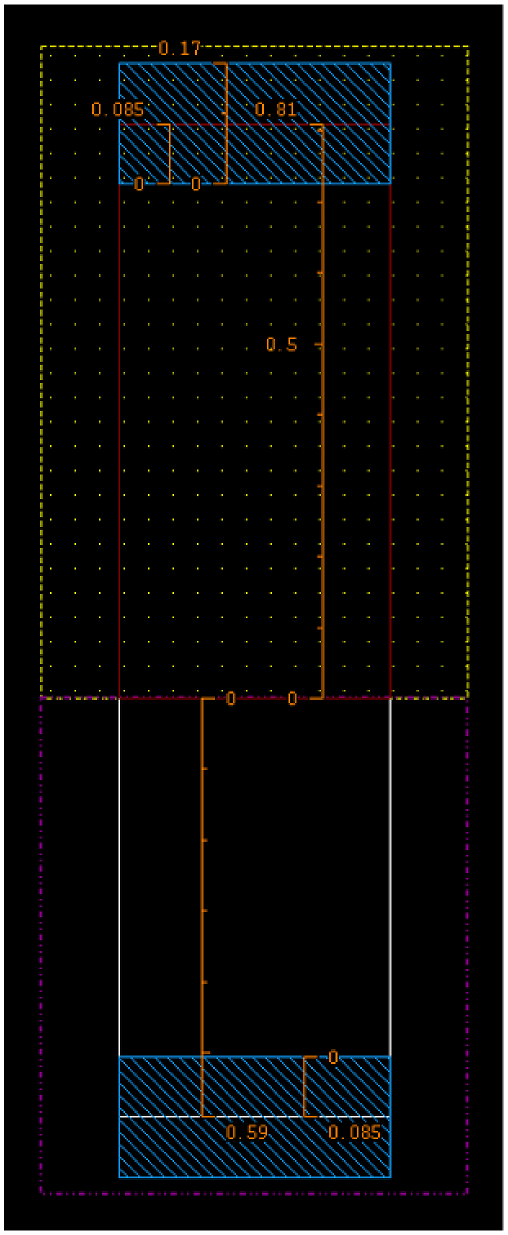


Figure 9

3. Diffusion

The diffusion (or active) region represents the area that is exposed for the diffusion of n-type or p-type dopants. The height of the OD shape determines the channel width of the NMOS or PMOS device. Mismatch between the drawn shape and the width specified in the schematic will result in an LVS error. Diffusion regions are also required to make an ohmic connection between the body and power straps.

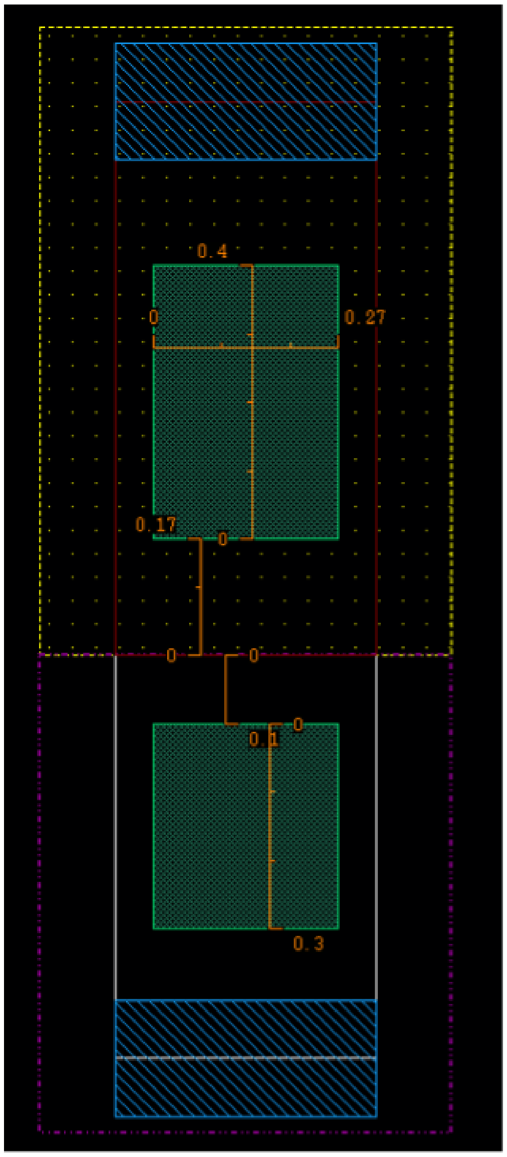


Figure 10

4. Polysilicon

Polysilicon, a.k.a 'poly', is the material used for the transistor gate. Create the inverter gate as in Figure 11, making sure to include the small protruding rectangle, which will be used later to house a metal contact. (Note: there is an intentional DRC violation here, in order to demonstrate fixing DRC violations later.)

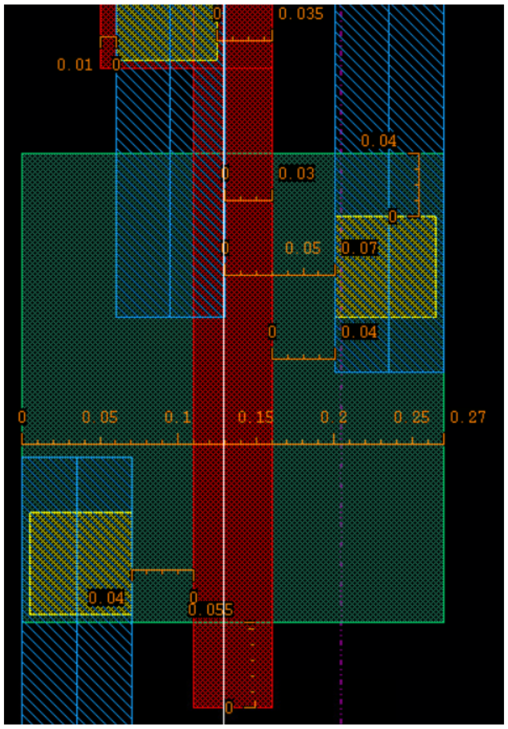


Figure 11

5. Contacts and metal

Place contacts and metal connections to complete the inverter as shown in Figures 12 and 13. Contacts must have dimensions of 0.065x0.065um in our FreePDK45 technology. Also notice the 0.075um space between pairs of contacts, and the 0.035um space from the leftmost and rightmost powerstrap contacts to the OD edge (Figure 12). 0.075um is the minimum spacing between contacts (per DRC rules), and the 0.005um space from the OD edge ensures that when cells are place side-by-side the 0.075um contact spacing is maintained.

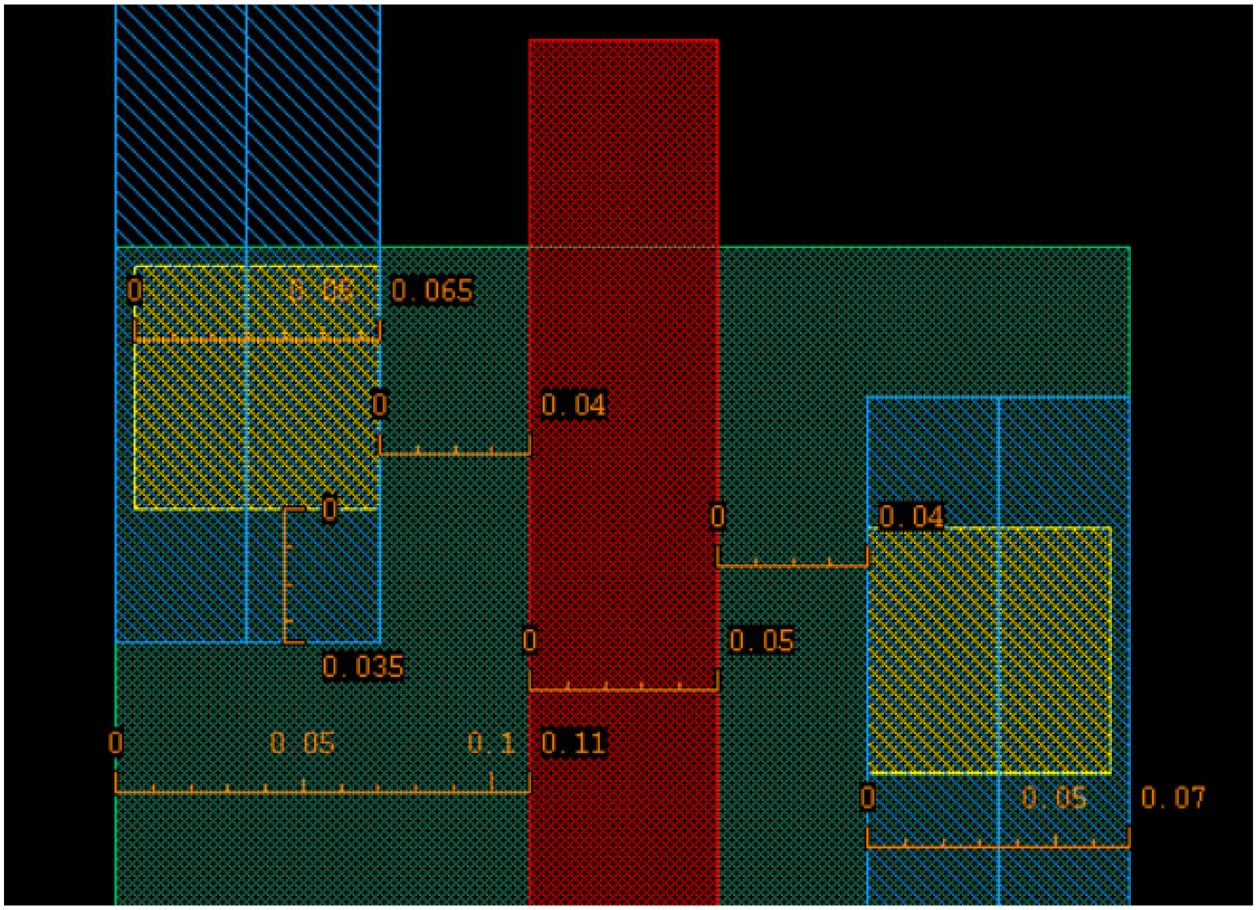


Figure 12

For metal on contacts the DRC rule is straightforward - the metal must extend at least 0.035um over at least two opposite sides of the contact. (In Figure 13, for each contact, one side of the metal extends 0.035um while the opposite side of the metal extends greater than 0.035um.)

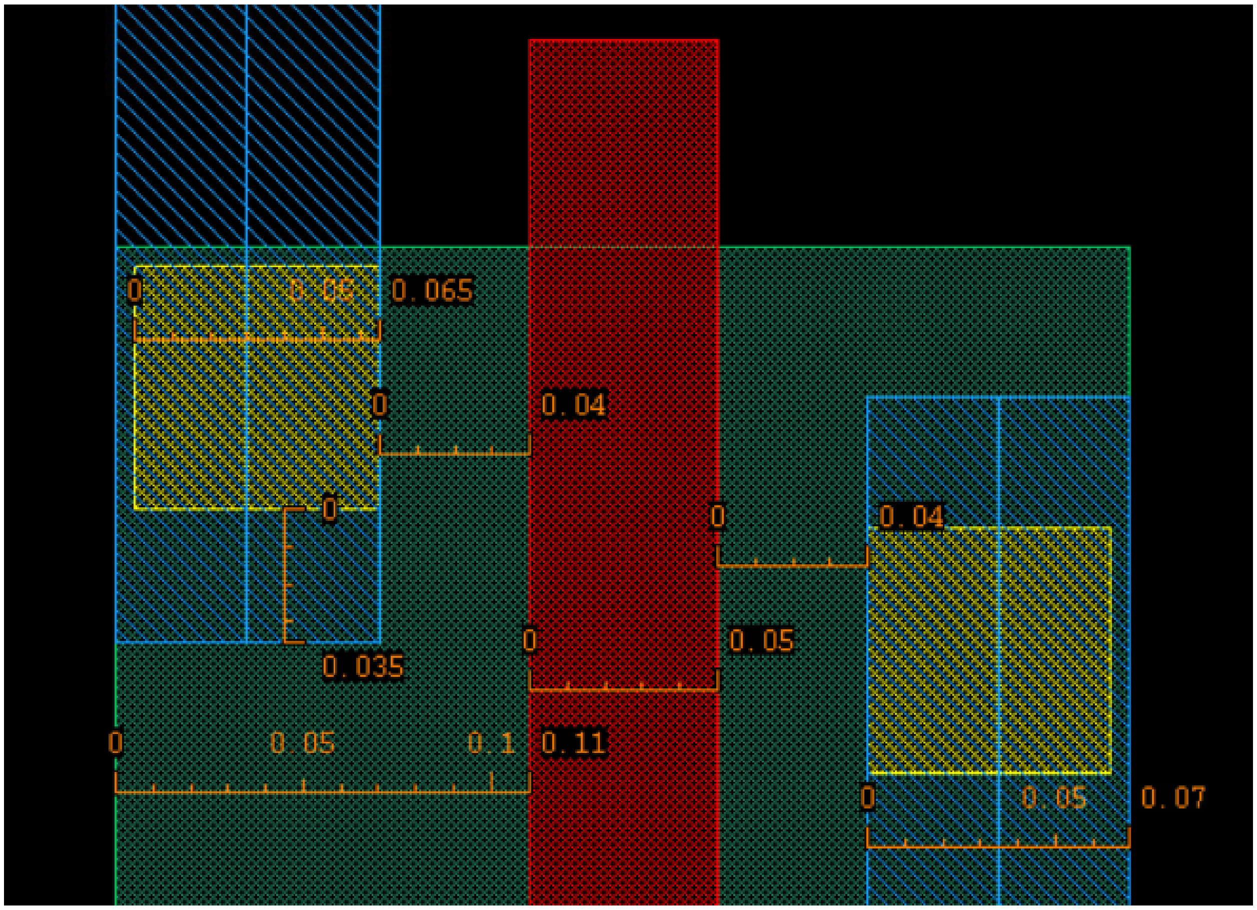
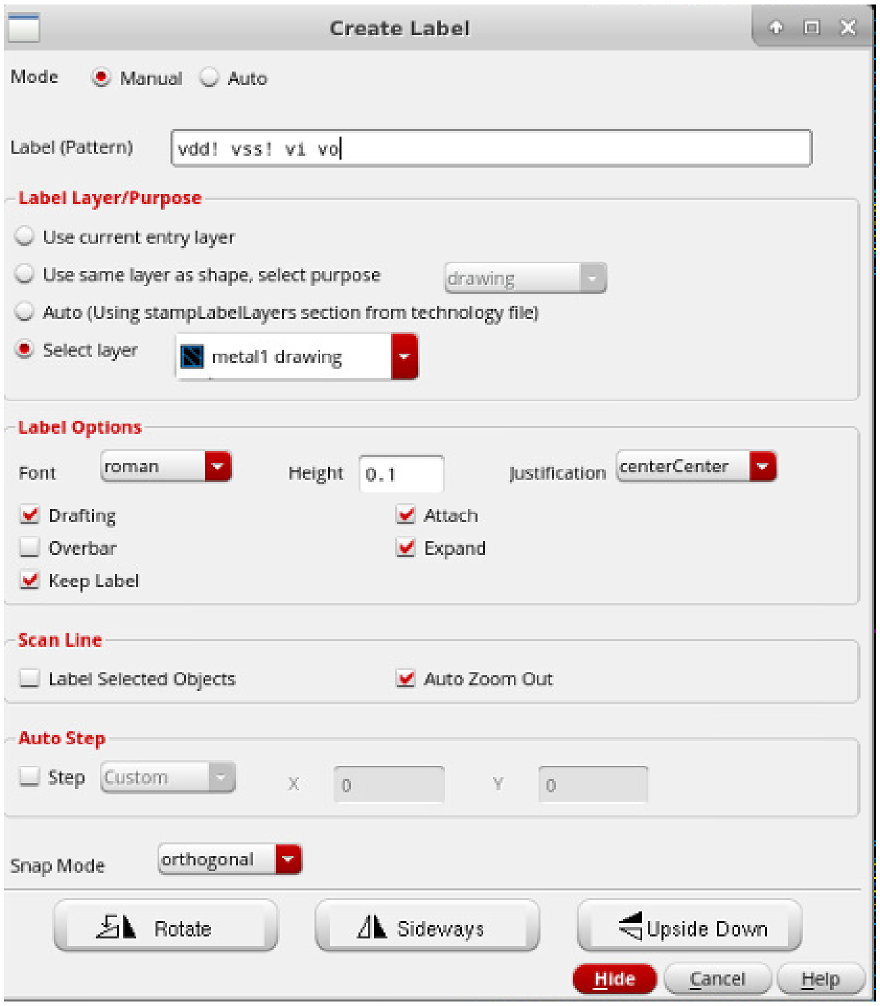


Figure 13

6. Pin placement

In order for the tools to perform various design checks, as well as for netlist creation, the metal shapes that correspond to circuit pins must be labeled. Select "Create → Label" to bring up the window in Figure 14. The radio button that is selected under "Label Layer/Purpose" heading should be set to "Select layer", where "metal1 drw" should be selected in the drop-down menu.

For power and ground nets, the name of the pin should be suffixed with an exclamation mark, e.g., 'vss!'. This is done to match the inverter schematic, where nets connected to the 'vss' and 'vdd' symbols are automatically given global status (indicated by "!"). After the pin labels have been placed onto their corresponding metal shapes, the layout should look like that in Figure 15.


Create Label

Mode [✓ Manual] Auto

Label (Pattern) vdd! vss! vi vo

Label Layer/Purpose
- Use current entry layer
- Use same layer as shape, select purpose drawing
- Auto (Using stampLabelLayers section from technology file)
- [✓ Select layer] metal1 drawing

Label Options
Font roman   Height 0.1   Justification centerCenter
- [✓ Drafting]
- Overbar
- [✓ Keep Label]
- [✓ Attach]
- [✓ Expand]

Scan Line
- Label Selected Objects
- [✓ Auto Zoom Out]

Auto Step
- Step Custom   X 0   Y 0

Snap Mode orthogonal

Figure 14

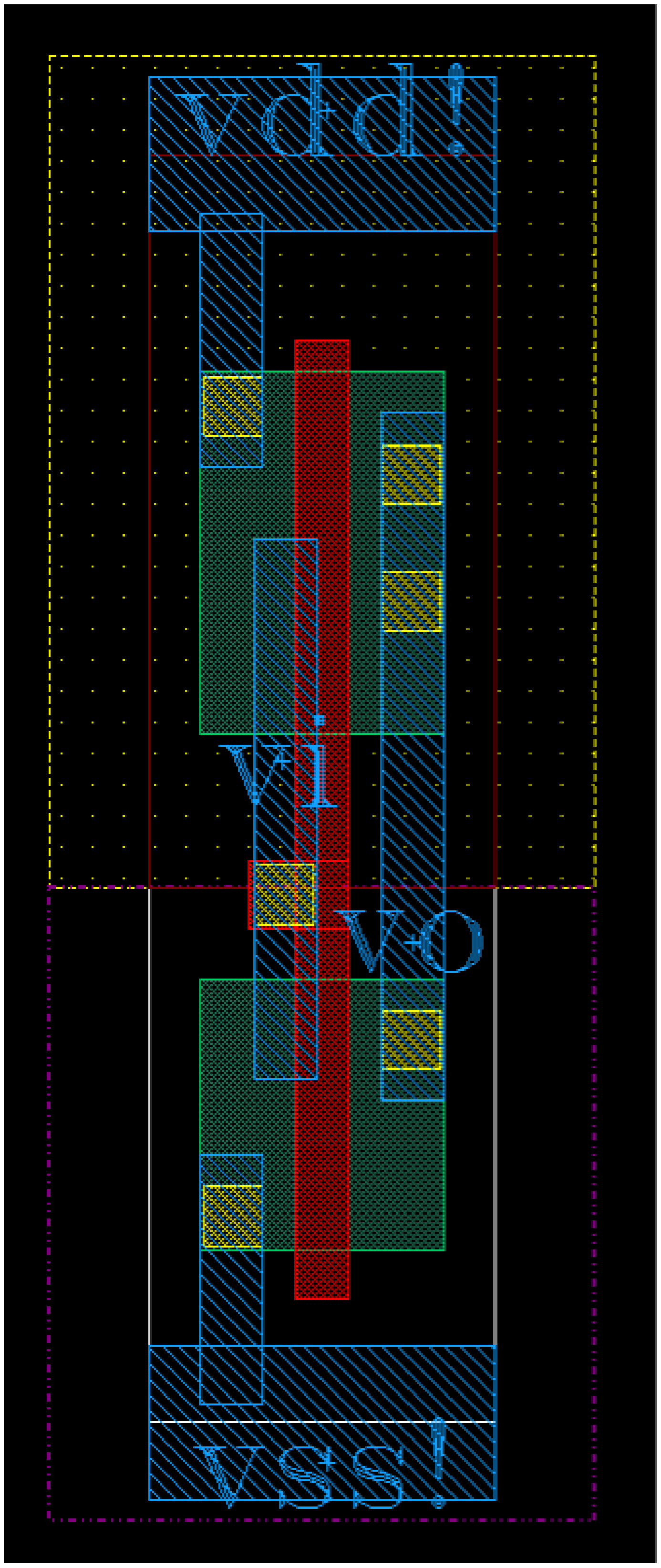


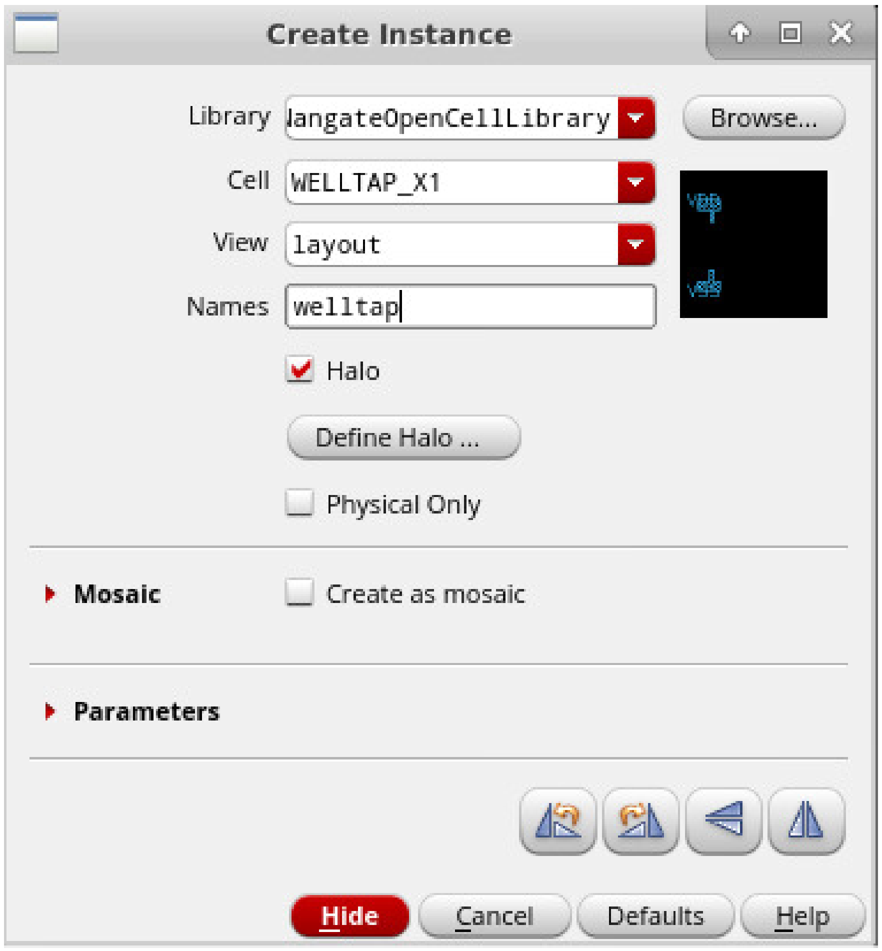
Figure 15: Pin placement, with Roman font and 0.1 text height for demonstration

7. Well tap

Well taps are placed to connect n-well to VDD and p-well to VSS. Due to Nangate standard library being a tap-less standard design, well taps must be placed adjacent to designs. Press "I" to create an instance, then select WELLTAP_X1 from NangateOpenCellLibrary as shown in Figure 16. In the layout editor window, press Shift+F to show instance details instead of an empty red box as shown in Figure 17. Place the tap cell adjacent to the inverter so that the powerstraps, wells, and doped regions connect as shown in 18.

Note: If reusing layout for a larger design and the layout passes DRC and LVS, remove the well tap until the final layout of the design is constructed.

At this point, the layout for the inverter is complete and ready to be checked for design rule violations, and equivalence to the inverter schematic.



Library NangateOpenCellLibrary
Cell WELLTAP_X1
View layout
Names welltap
[✓] Halo
[ ] Physical Only
[ ] Create as mosaic

Figure 16

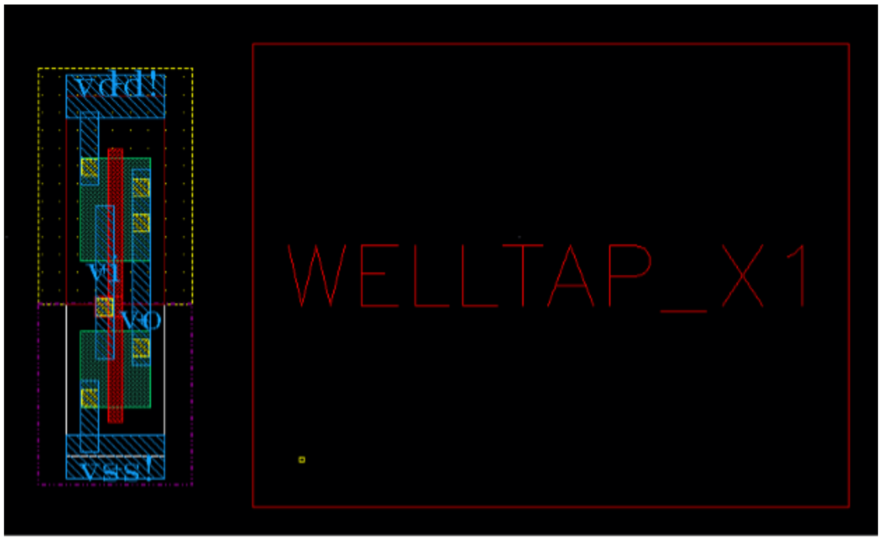


Figure 17: Press Shift+F to show contents of instances. Press CTRL+F to hidden details.

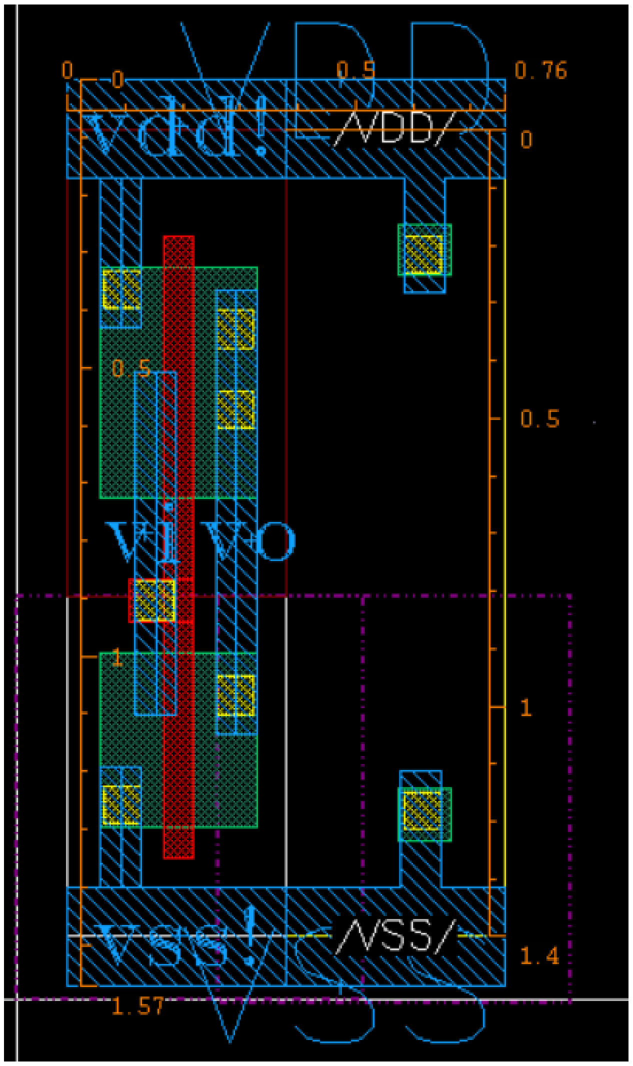


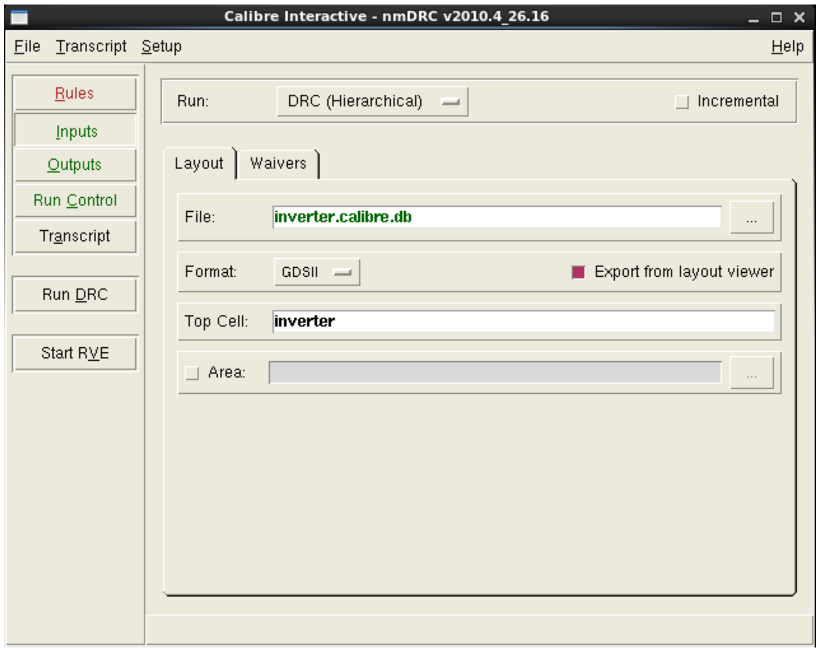
Figure 18: Inverter with well tap

Design rule checking (DRC)

While layout shapes are drawn and represented as perfect polygons, in reality the shapes and characteristics of the deposited materials are subject to variation. Design rules are sets of constraints that ensure a given chip will function correctly after it is manufactured in spite of variations[4].

1. Open Calibre

Click "Calibre → Run DRC". The window in Figure 19 should appear (hit "Cancel" for the "Load Runset File" window).


Calibre Interactive - nmDRC v2010.4_26.16

Run: DRC (Hierarchical)

File: inverter.calibre.db

Format: GDSII

Export from layout viewer

Top Cell: inverter

Figure 19

2. Runsets

After starting Calibre, there will be a notification that asks to load a runset file. A runset is a file that saves the session settings, so that they may be reused later. Since DRC has never been run before, this window may be closed (although it will save time to use runsets in the future).

3. Load the design rules

Run the terminal commands below to copy the DRC rules into the local directory:

```
cd <local_Cadence_dir>/
mkdir Calibre
cd Calibre
```

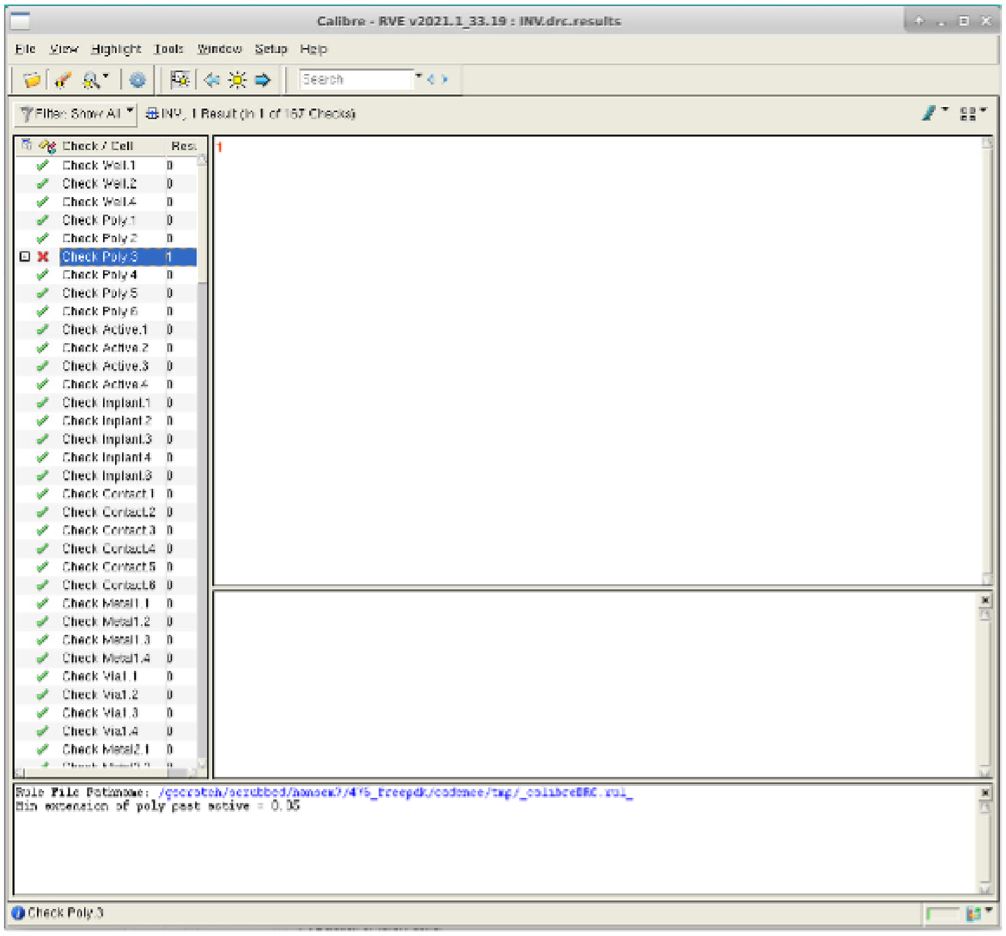
[4]Realistically it impossible to achieve 100% yield, so design rules for a given semiconductor process are typically constructed to balance device performance and yield

```
For hyak:
cp /gscratch/ece/courses/476_aut2022/common/freepdk45_opencelllibrarypdk45/FreePDK45/n
For linux lab:
cp /home/projects/ee476.2022aut/common/freepdk45_opencelllibrarypdk45/FreePDK45/ncsu_b
```

Load the DRC rules into Calibre by selecting the "Rules" tab, and "..." to browse for the `calibreDRC.drc` file that was copied earlier.

4. *Running DRC*

Select the "Run DRC" button; there should be a DRC error as shown in Figure 20.


Calibre - RVE v2021.1_33.19 : INV.drc.results
File View Highlight Tools Window Setup Help
Search
Filter: Show All
INV, 1 Result (in 1 of 157 Checks)
1

	Check / Cell	Res.
✓	Check Well.1	0
✓	Check Well.2	0
✓	Check Well.4	0
✓	Check Poly.1	0
✓	Check Poly.2	0
✗	Check Poly.3	1
✓	Check Poly.4	0
✓	Check Poly.5	0
✓	Check Poly.6	0
✓	Check Active.1	0
✓	Check Active.2	0
✓	Check Active.3	0
✓	Check Active.4	0
✓	Check Implant.1	0
✓	Check Implant.2	0
✓	Check Implant.3	0
✓	Check Implant.4	0
✓	Check Implant.5	0
✓	Check Contact.1	0
✓	Check Contact.2	0
✓	Check Contact.3	0
✓	Check Contact.4	0
✓	Check Contact.5	0
✓	Check Contact.6	0
✓	Check Metal1.1	0
✓	Check Metal1.2	0
✓	Check Metal1.3	0
✓	Check Metal1.4	0
✓	Check Via1.1	0
✓	Check Via1.2	0
✓	Check Via1.3	0
✓	Check Via1.4	0
✓	Check Metal2.1	0

```
Rule File Pathname: /gscratch/scrubbed/hansem7/476_freepdk/cadence/tmp/_calibreDRC.rul_
Min extension of poly past active = 0.05
```

Check Poly.3

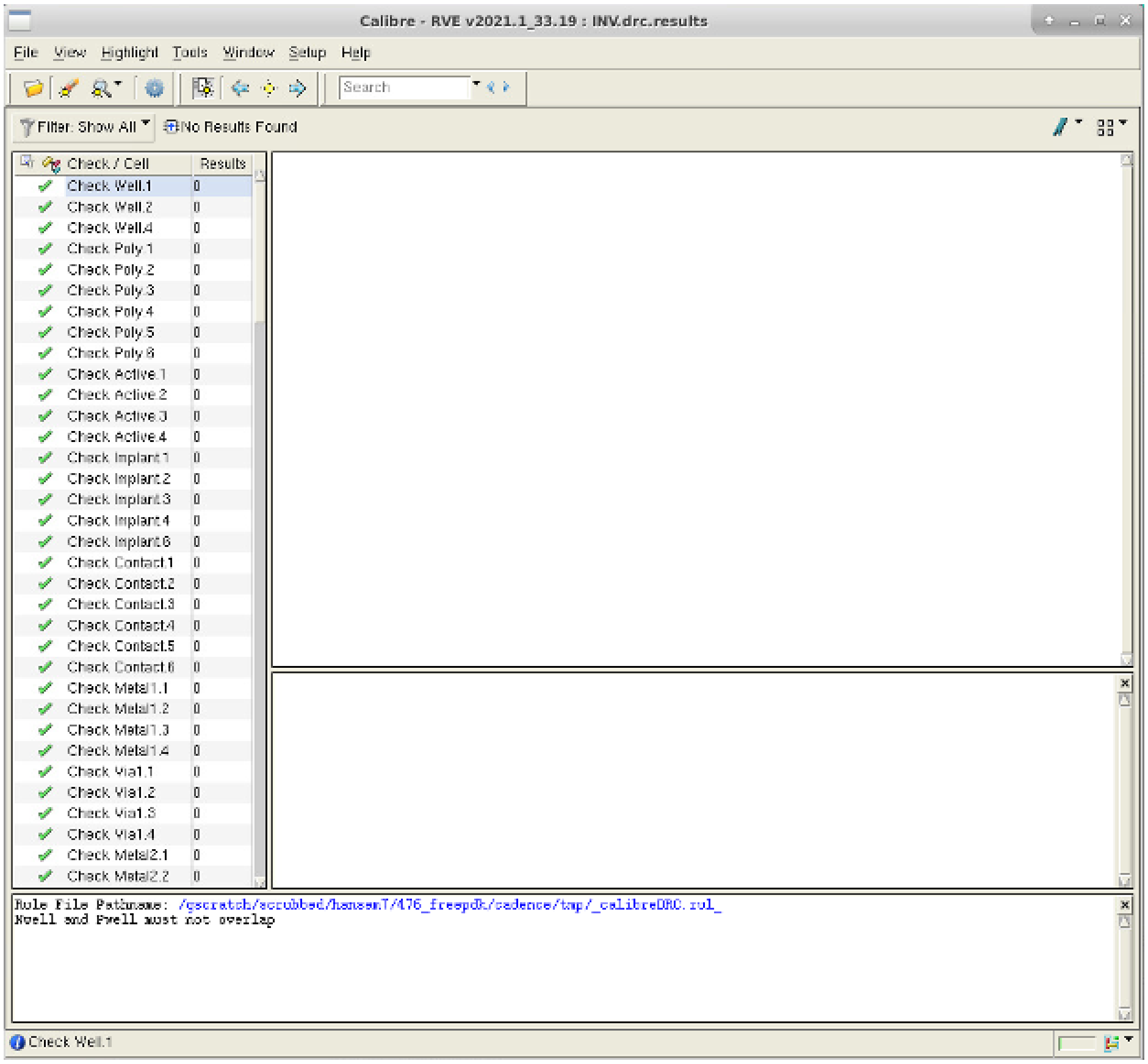
Figure 20

5. *Fixing violations*

Select the design rule violation in the window that appears to see more information. Double clicking the number (e.g., '1') will highlight the error in the layout editor. In this case, the error should be "Check Poly.3" with the details

```
Min extension of poly past active = 0.05
```

Alternatively, we can find more detail on NCSU wiki design rules and see that POLY.3 has the minimum extension set to 55 nm (0.055um). In this case, since there is a discrepancy, we will go with 55 nm extension even though 50 nm is sufficient to pass DRC. Correcting for this, we see in Figure 21 that DRC passes.


Calibre - RVE v2021.1_33.19 : INV.drc.results
File View Highlight Tools Window Setup Help
Search
Filter: Show All
No Results Found

Check / Cell	Results
✓ Check Well.1	0
✓ Check Well.2	0
✓ Check Well.4	0
✓ Check Poly.1	0
✓ Check Poly.2	0
✓ Check Poly.3	0
✓ Check Poly.4	0
✓ Check Poly.5	0
✓ Check Poly.6	0
✓ Check Active.1	0
✓ Check Active.2	0
✓ Check Active.3	0
✓ Check Active.4	0
✓ Check Implant.1	0
✓ Check Implant.2	0
✓ Check Implant.3	0
✓ Check Implant.4	0
✓ Check Implant.6	0
✓ Check Contact.1	0
✓ Check Contact.2	0
✓ Check Contact.3	0
✓ Check Contact.4	0
✓ Check Contact.5	0
✓ Check Contact.6	0
✓ Check Metal1.1	0
✓ Check Metal1.2	0
✓ Check Metal1.3	0
✓ Check Metal1.4	0
✓ Check Via1.1	0
✓ Check Via1.2	0
✓ Check Via1.3	0
✓ Check Via1.4	0
✓ Check Metal2.1	0
✓ Check Metal2.2	0

```
Rule File Pathname: /gscratch/scrubbed/hansenT/476_freepdk/cadence/tmp/_calibreDRC.rul_
Nwell and Pwell must not overlap
```

Check Well.1

Figure 21

Layout versus schematic (LVS)

DRC ensures that the layout shapes meet manufacturability requirements, but it does not check that the layout is logically equivalent to the schematic. That is, LVS checks whether the layout performs the same mathematical function as the schematic, and that the physical properties of the transistors are correct. Some properties LVS uses to check equivalence include transistor type, transistor width, pins, and wires/connections[5].

1. Open Calibre (LVS) and load rules

As with DRC, select "Calibre → Run LVS". There will be another prompt that asks to load a runset file, cancel the dialog box for now.

In the "Rules" tab, select the "”...”" button to browse for the `calibreLVS.rul` file that you copied into your Calibre directory earlier.

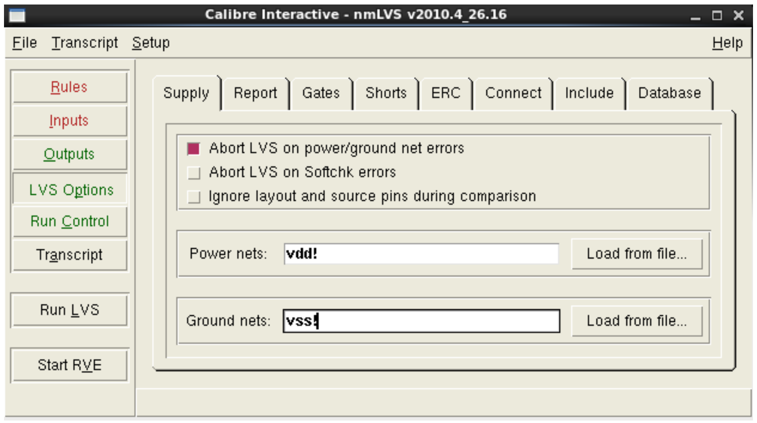
2. Identify netlist

Select "Inputs" in the left side main LVS window, and under the "Layout" tab enable "Export from layout viewer". Under the "Netlist" tab, enable "Export from schematic viewer". This allows LVS to automatically get the netlists for the most recent versions of the schematic and layout.

3. Identify power/ground nets

To suppress certain warnings and strange errors, LVS must be informed about any nets that are used for power and ground connections. At the top of the main window, select "Setup" and check the "LVS Options" box. Select the "LVS Options" item that is now available on the left side of the main window, and enter the names of the power and ground nets in the layout (e.g., "vdd!" and "vss!") (Figure 22).

Note: if you are unable to type into the "Power nets" or "Ground nets" fields, type the text somewhere else and copy it to the clipboard, then right-click in the "Power nets" field and select "Paste".



- [✓] Abort LVS on power/ground net errors
- [ ] Abort LVS on Softchk errors
- [ ] Ignore layout and source pins during comparison

Power nets: vdd!

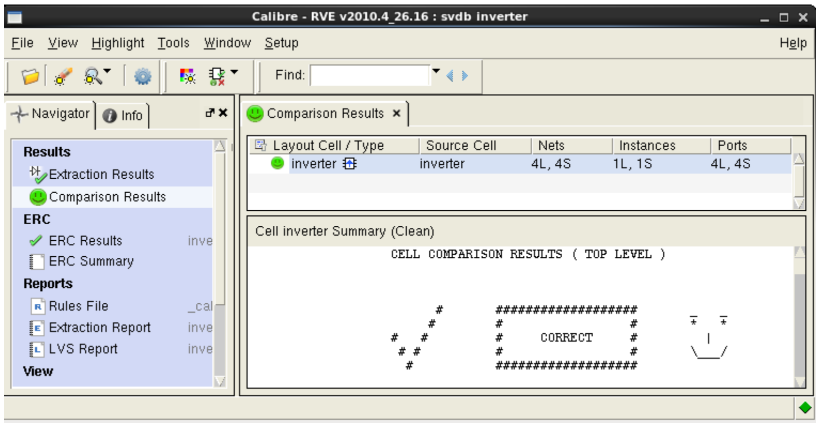
Ground nets: vss!

Figure 22

[5]The details of how LVS tools work is out of the scope of this class, but they essentially check whether the connectivity and property graphs of the schematic and layout are isomorphic

4. Run LVS

Select the "Run LVS" menu item, if there are no errors the window will look like that in Figure 23. The LVS run will also generate a file in the specified run directory called `<name>.lvs.report`, where `<name>` is the name of the design; as with the DRC summary, the LVS report is part of the CAD submission package.



Layout Cell / Type	Source Cell	Nets	Instances	Ports
inverter	inverter	4L, 4S	1L, 1S	4L, 4S

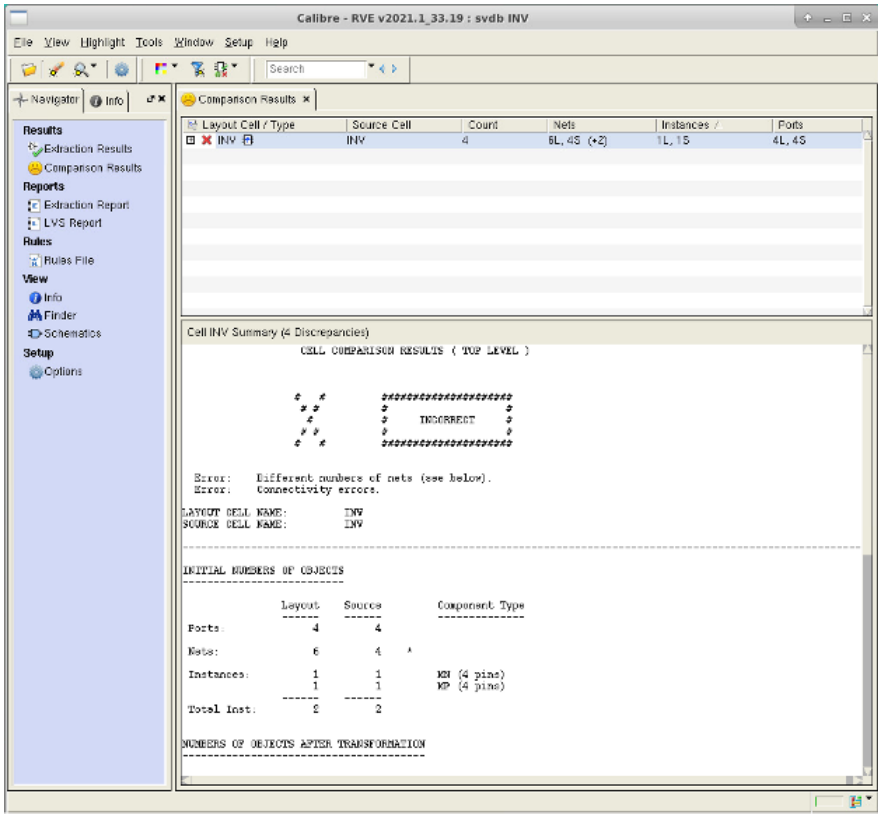
Cell inverter Summary (Clean)

```
                    CELL  COMPARISON  RESULTS  (  TOP  LEVEL  )


                  #            ###################
                 #             #                 #          *   *
             #  #              #     CORRECT     #            |
              # #              #                 #          \___/
               #               ###################
```

Figure 23

If well tap cells were not placed before LVS, then an error will appear as shown in Figure 24.



Layout Cell / Type	Source Cell	Count	Nets	Instances	Ports
INV	INV	4	6L, 4S (+2)	1L, 1S	4L, 4S

Cell INV Summary (4 Discrepancies)

```
                CELL  COMPARISON  RESULTS  (  TOP  LEVEL  )



               #   #        ####################
                # #         #                  #
                 #          #    INCORRECT     #
                # #         #                  #
               #   #        ####################


 Error:    Different numbers of nets (see below).
 Error:    Connectivity errors.

LAYOUT CELL NAME:         INV
SOURCE CELL NAME:         INV

--------------------------------------------------------------------------------

INITIAL NUMBERS OF OBJECTS
--------------------------

                 Layout    Source         Component Type
                 ------    ------         --------------
 Ports:               4         4

 Nets:                6         4    *

 Instances:           1         1         MN (4 pins)
                      1         1         MP (4 pins)
                 ------    ------
 Total Inst:          2         2


NUMBERS OF OBJECTS AFTER TRANSFORMATION
---------------------------------------
```

Figure 24

Types of common LVS errors

Common bugs that cause LVS errors include:

- Misplaced labels (or incorrectly named labels, labels in the wrong layer, etc.)
- Incorrect MOSFET sizes
- Accidental shorts between metals
- Shorts created by misplaced/accidental vias[6]

[6]Vias are the connections between metal layers, for instance to connect M1 to M2.

Netlist and Parasitics Extraction

Extraction is the translation of an integrated circuit layout back into the electrical circuit (netlist) it is intended to represent. The extracted circuit is needed for various purposes including circuit simulation, timing analysis, power analysis, and more. Compared to a schematic netlist, the layout netlist produces a more accurate prediction of the post-manufactured circuit, since the additional capacitances and resistances created by wiring, vias, the shapes of transistors, and more, can be accounted for.

1. Open Calibre (PEX)

As with DRC and LVS, select "Calibre → Run PEX".

2. Load PEX rules and select settings

Load the PEX rules file (`calibrexRC.rul`) as with DRC and LVS. Select "Outputs" and in the window that appears, de-select "R + C + CC" and select "C + CC". This option disables the extraction of parasitic resistances, which helps to reduce simulation times during active development.

3. Run PEX

Select "Run PEX", if there are no DRC or LVS violations a netlist will be generated at the location specified in the PEX output options. PEX generates two files for "C + CC", assuming the design name is `<des_name>`:

1. `<des_name>.pex.netlist`
2. `<des_name>.pex.netlist.<des_name>.pxi`

The first file contains the design sub-circuit, while the second contains additional parasitics[7].

Post-layout simulation

1. Replace the old sub-circuit

To run simulations for the `loaded_inverter` using the post-layout version of the inverter, the `loaded_inverter` netlist can be reused with slight modifications. Specifically, the schematic-level netlist for the inverter sub-circuit can simply be replaced by the new post-layout netlist. Observe the following `loaded_inverter.ckt` netlist:

[7] These files are in human-readable text, and students may want to see how the parasitics are included in the netlist.

```
 1  ** Generated for: hspiceD
 2  ** Generated on: Oct  6 16:25:59 2022
 3  ** Design library name: cad2
 4  ** Design cell name: loaded_inverter
 5  ** Design view name: schematic
 6  .GLOBAL vss! vdd!
 7
 8  .TEMP 25.0
 9  .OPTION
10  +    ARTIST=2
11  +    INGOLD=2
12  +    PARHIER=LOCAL
13  +    PSF=2
14
15  * ==== Schematic - level netlist (commented out) ====
16  ** Library name: cad2
17  ** Cell name: INVD1
18  ** View name: schematic
19  *.subckt INVD1 vi vo
20  *m0 vo vi vss! vss! NMOS_VTL L=50e-9 W=300e-9 AD=31.5e-15 AS=31.5e-15 (...)
21  *m1 vo vi vdd! vdd! PMOS_VTL L=50e-9 W=400e-9 AD=42e-15 AS=42e-15 (...)
22  *.ends INVD1
23  ** End of subcircuit definition.
24
25   ** ==== Post - layout netlist ====
26  .include INVD1.pex.netlist
27
28   ** ==== loaded_inverter circuit ====
29  ** Library name: cad2
30  ** Cell name: loaded_inverter
31  ** View name: schematic
32  xi0 vi vo INVD1
33  c0 vo 0 10e-15
34  .END
```

In the example above, lines 19-22 correspond to the INVD1 schematic-level sub-circuit, which has been commented out. The other two files generated by PEX are included inside the .pex.netlist file, and should be placed in the same directory. One important detail to check is that the port order of the post-layout netlist is identical to that of the schematic-level netlist. Ports are connected positionally, and PEX sometimes generates netlists with different port orderings, or ports that are globally accessible and don't need to be declared (i.e. VDD! and VSS!).

For instance, if the generated INVD1.pex.netlist file looks like this:

```
 1  * File: INVD1.pex.netlist
 2  * Created: Thu Oct  6 23:30:20 2022
 3  * Program "Calibre xRC"
 4  * Version "v2021.3_35.19"
 5  *
 6  .subckt INVD1 VSS! VDD! VO VI
 7  *
 8  MM0 VO VI VSS! VSS! NMOS_VTL L=6e-08 W=3e-07 AD=3.15e-14 AS=3.15e-14 (...)
 9  MM1 VO VI VDD! VDD! PMOS_VTL L=6e-08 W=4e-07 AD=4.2e-14 AS=4.2e-14 (...)
10  + PS=1.01e-06
11  c_4 VI 0 0.0942566f
12  c_8 VO 0 0.0366562f
13  c_12 VSS! 0 0.0747331f
14  c_16 VDD! 0 0.0766818f
15  *
16  .include ‘‘INVD1.pex.netlist.INVD1.pxi’’
17  *
18  .ends
```

then line 6 should be changed to:

```
.subckt INVD1 VI VO
```

2. Run HSPICE

After the sub-circuit substitution mentioned above has been made, HSPICE can be run in the same way as with the schematic-level netlist.

(end of tutorial)

3 Post-layout Inverter Measurements

a. Collect the same measurements from Part 2 but for the post-layout version of the `loaded_inverter` (created in the layout tutorial) at **VDD = 1.2V**.

Delivery

a. The post-layout version of the `loaded_inverter` netlist (named `loaded_inverter.ckt`), and the measurements in Table 1 (fill in the provided `specifications.json` file).

b. The generated DRC and LVS reports for `INVD1` (they should be named `INVD1.drc.summary` and `INVD1.lvs.report`)

4 Double-fingered Inverter and Questions

a. In previous sections, students have gained experience with layout. However, the 1.4um standard cell height limits "wider" MOSFET usage. If the PMOS in Figure 25 has width of 0.8um, it would be impossible to implement in a single inverter layout.

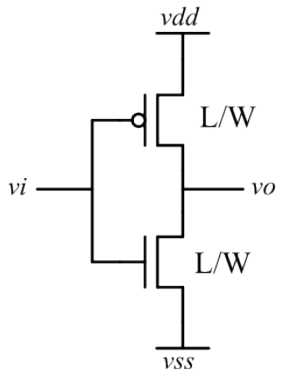


Figure 25

For the layout of this inverter, it will be necessary to "break-up" the PMOS device into two 0.4um / 50nm FETs in parallel in order to get the equivalent width of 0.8um without violating the 1.4um standard cell height. LVS will detect FETs in parallel and “merge” them so that you can still implement your schematic with a single 0.8um/50nm PMOS and 0.3um/50nm NMOS device as illustrated in Figure 25. The stick diagrams in Figures 26 and 27 show two examples for how to layout a double-fingered inverter.

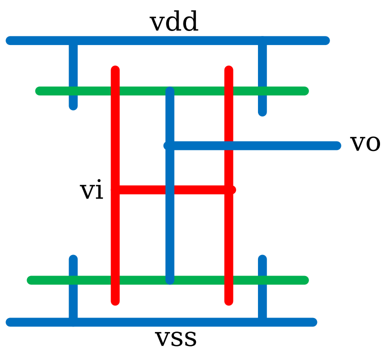


Figure 26

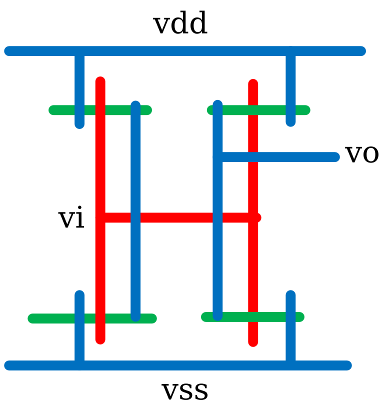


Figure 27

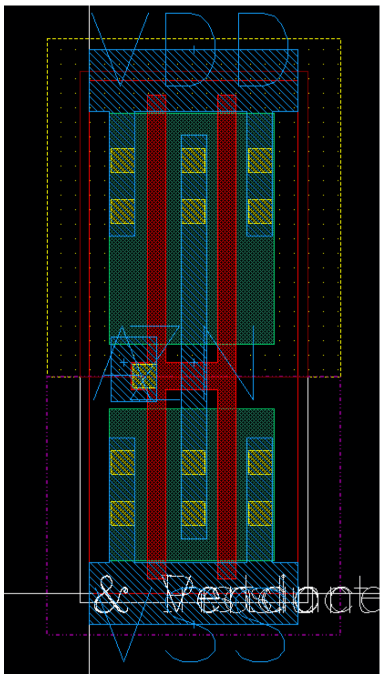


Figure 28

Compare the layout in Figure 28 to Figures 26 and 27 to answer the following questions:

1. Find only one error in these two figures (Figures 26 and 27).
2. Which implementation, Figure 26 or 27, is used in the standard inverter cell? Why is this implementation preferred by the standard library?

Delivery

a. A PDF report answers the questions above.

File Submission

The answer to questions of double-fingered inverter should go in a short report titled "`cad2_report.pdf`". Measurements should be submitted using the provided `specifications.json` file, and files should be in the specified locations (see the *CAD Submission and Grading* document for more information).