Due on Monday, Feb. 7th at 5pm. Please submit electronically in CANVAS.

Inverter Layout and Extraction Tutorial

EE473 CAD Lab 3
Winter 2022

Introduction

Transistor technology has come a long way since the work of Bardeen, Brattain, and Shockley in 1947. Integrated circuit technology packs so many devices in such a small scale that a single MOSFET device is smaller than a virus! For this reason, the physical design of integrated circuits must be performed through computing tools and mechanized interfaces.

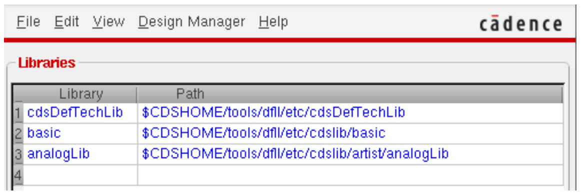
The final stage of any chip design is called "layout," when designers organize the physical structure of the silicon, doping, metal, and insulators that make up an integrated circuit. This step is crucial, as the placement and connection of devices has dramatic impact on performance. The wires and devices on a chip form an array of physical capacitors, resistors, and inductors, which are not captured by schematic-level simulation. Automated analysis of a layout allows for estimation of these "parasitic" elements and their impact on circuit functionality, a process called "extraction."

This tutorial walks through the layout, extraction of a CMOS inverter.

Library Setup

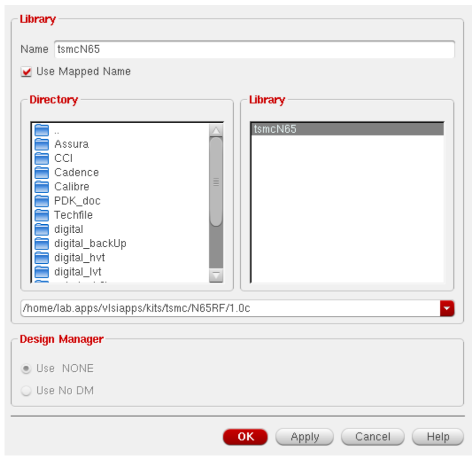
Integrated circuit technologies come in industry-standard libraries called Process Design Kits (PDKs). These kits contain the schematic-level models of the available devices, their corresponding layouts, and mask data for the layers and process steps that go into fabricating a chip. In order to create a physical layout for this tutorial, we will have to first reference the foundry-provided 65nm PDK.

To do so, open up Cadence in the working directory created in the first lab. From the Library Manager drop-down menus, select "Edit: Library Path" to open the Library Path Editor, which looks like this:



	Library	Path
1	cdsDefTechLib	$CDSHOME/tools/dfII/etc/cdsDefTechLib
2	basic	$CDSHOME/tools/dfII/etc/cdslib/basic
3	analogLib	$CDSHOME/tools/dfII/etc/cdslib/artist/analogLib
4		

Select "Edit: Add Library" from the drop-down. Navigate to /home/lab.apps/vlsiapps/kits/tsmc/N65RF/1.0c/ and select the tsmcN65 library, as shown below:


Library

Name tsmcN65

[✓ Use Mapped Name]

Directory

..
Assura
CCI
Cadence
Calibre
PDK_doc
Techfile
digital
digital_backUp
digital_hvt
digital_lvt

Library

tsmcN65

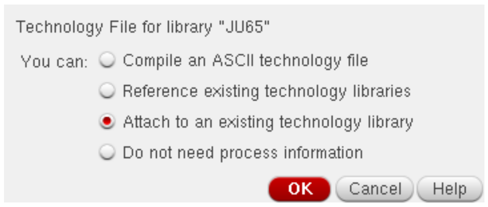
/home/lab.apps/vlsiapps/kits/tsmc/N65RF/1.0c

Design Manager

[● Use NONE]
Use No DM

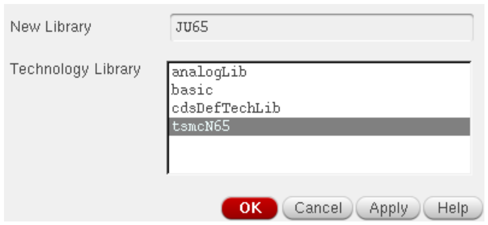
Hit OK, save and close the Library Path Editor, and you will notice tsmcN65 has been added to the list of libraries. Next, create a new library for your design in the 65nm process. Name it something descriptive, indicating the library owner and process node. My library is named JU65 for my initials and the 65nm process.

After the library is named, you will be asked about a technology file for the library. This is where you specify that this design library will be in the 65nm process. Select "Attach to an existing technology library," as shown:


Technology File for library "JU65"

You can:
Compile an ASCII technology file
Reference existing technology libraries
[● Attach to an existing technology library]
Do not need process information

Select the tsmcN65 library and press OK, as shown, to complete the library setup.


New Library JU65

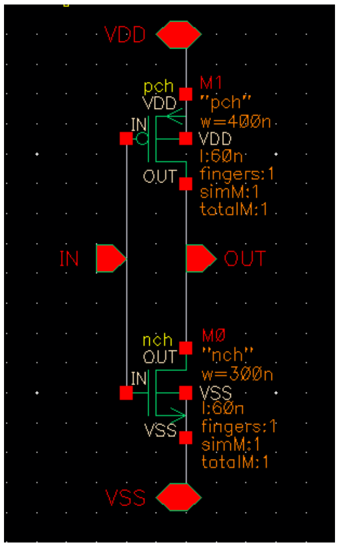
Technology Library

analogLib
basic
cdsDefTechLib
[✓ tsmcN65]

Schematic and Symbol Creation

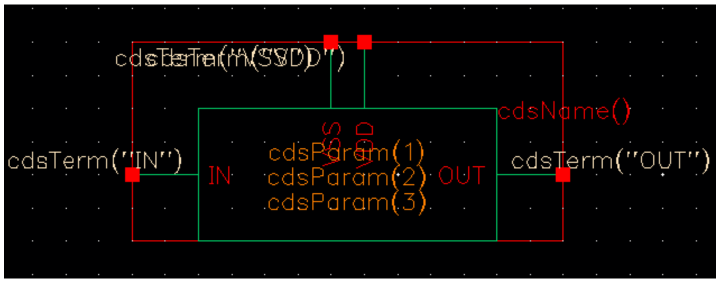
Next, create a schematic cell view in the newly created library called "INVD1." This is standard naming convention for an inverter cell with single-gate MOSFETs. In this schematic, instantiate a PMOS (cell "pch" in library tsmcN65) and NMOS (cell "nch" in library tsmcN65). Size these devices so that the PMOS has $\frac{W_P}{L_P} = \frac{400n}{60n}$ and the NMOS has $\frac{W_N}{L_N} = \frac{300n}{60n}$. This sizing skew is introduced to balance the inverter, due to PMOS devices having less transconductance for a given gate width in this process.

Create pins for "IN," "OUT," "VDD," and "VSS" using the "p" hotkey. These pins are used to create a symbol that compresses this schematic into a single device for higher-level hierarchies. Connect the device gates to "IN," device drains to "OUT," NMOS source and body to "VSS," and PMOS source and body to "VDD." Make sure you make "IN" an input pin, "OUT" an output pin, and "VDD" and "VSS" input/output pins. The resulting schematic should look similar to the one below:

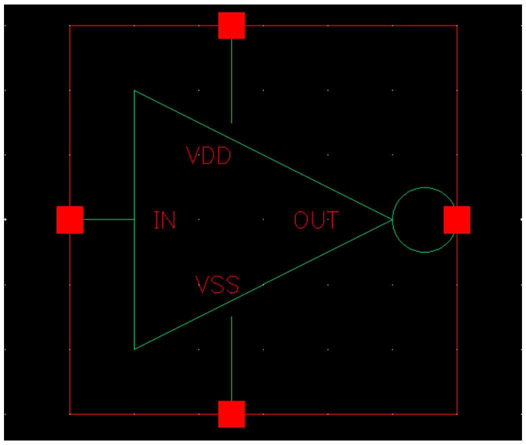


Check and save the schematic, and we can move on to creating a circuit symbol.

To automatically create a circuit symbol from the saved schematic, select "Create:Cellview:From Cellview" in the schematic editor dropdown menus. This uses the pins placed in the schematic to automatically create a symbol with those same pins. Click OK through the following pop-up menus. This will open an automatically generated symbol, as shown below:



The automatically generated symbols are not at all descriptive. It is good practice for circuit symbols to imply the underlying functionality, to make the top-level schematic clear and easy to interpret. Edit the automatically generated symbol to resemble a traditional inverter, similar to the one shown below:

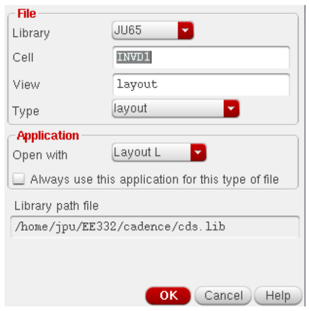


Check and save your finished symbol. The check feature here ensures that the pins in your symbol and schematic match. This finished symbol can be instantiated in other designs, just like the PDK devices. Once you pass the check, close the symbol editor.

Physical Layout

This section walks through the process of creating a physical layout, performing design-rule checks (DRC), passing formal equivalence checks (layout versus schematic, or "LVS"), and extracting circuit parasitics (parasitic extraction, or "PEX").

First, create a new layout cell view for the INVD1 inverter, as shown:

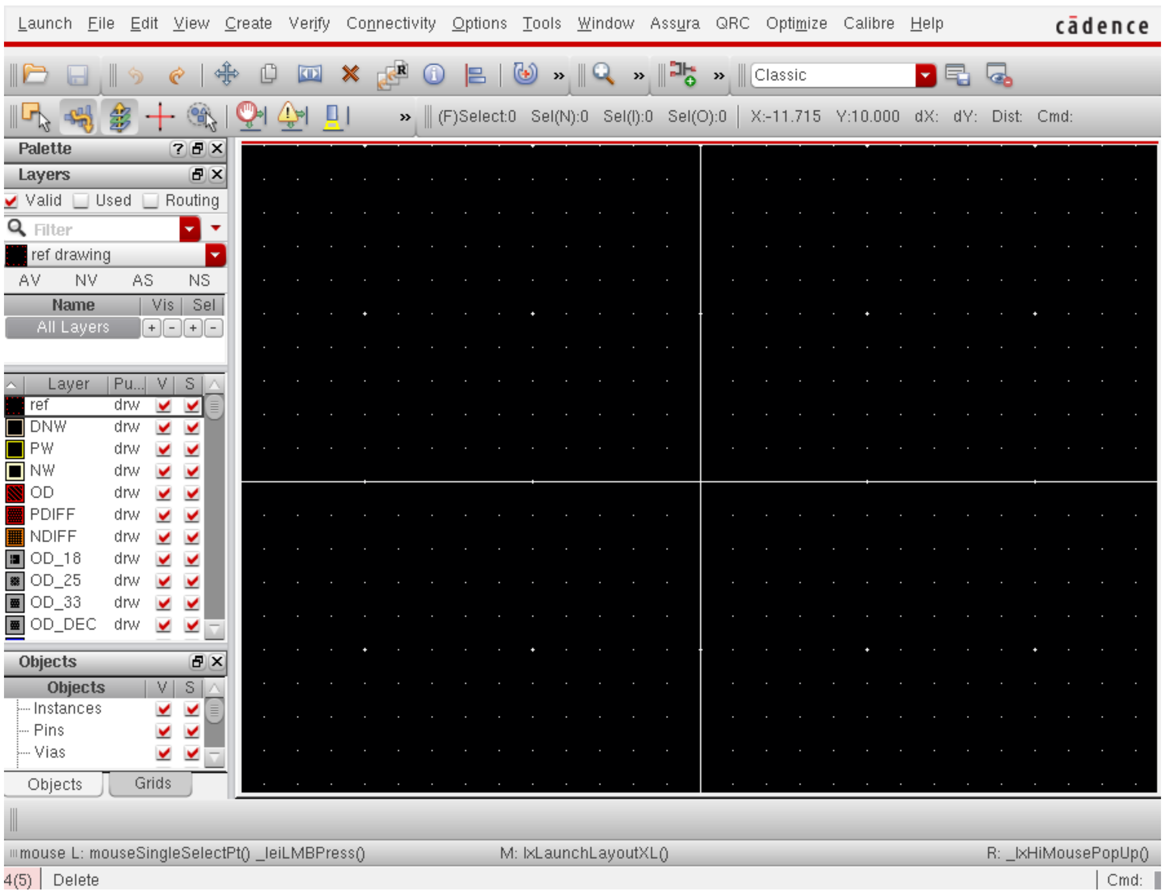


File
Library: JU65
Cell: INVD1
View: layout
Type: layout

Application
Open with: Layout L
Always use this application for this type of file: ☐

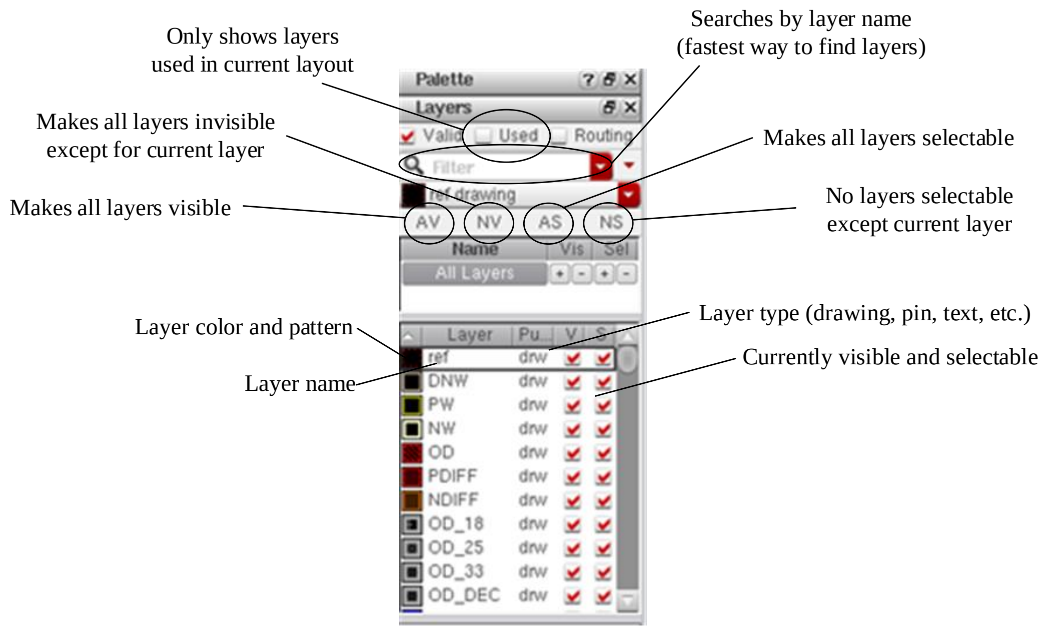
Library path file: /home/jpu/EE332/cadence/cds.lib

This opens a blank layout editor screen, as shown. Think of this as a top-down view of a silicon wafer.

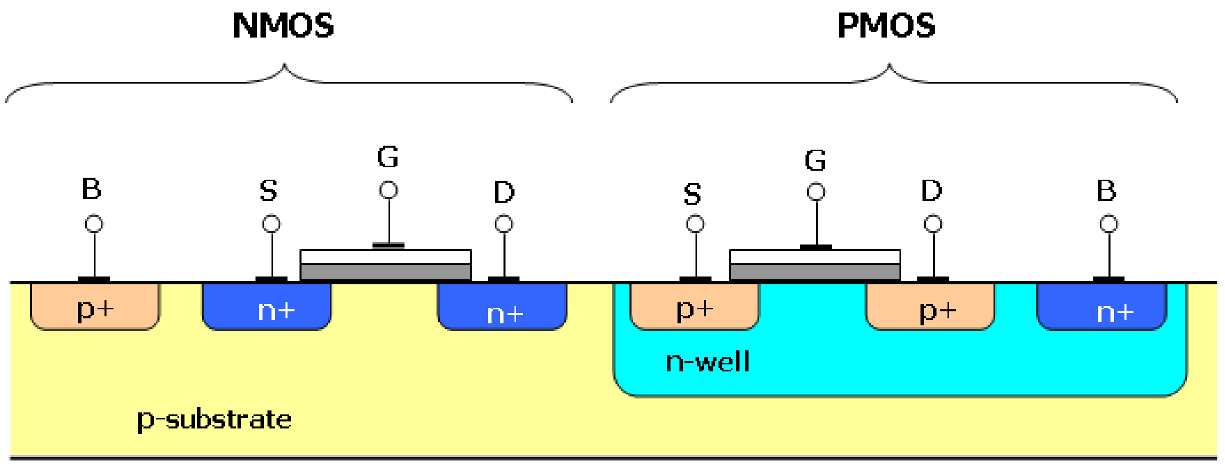


mouse L: mouseSingleSelectPt() _leiLMBPress()   M: lxLaunchLayoutXL()   R: _lxHiMousePopUp()
4(5) | Delete | Cmd:

An important part of the layout editor window is the layer palette, located along the left side. This is used to select the current drawing layer and toggle visibility and selectability for all layers. The layer palette functionality is summarized below:


Only shows layers used in current layout
Searches by layer name (fastest way to find layers)
Makes all layers invisible except for current layer
Makes all layers selectable
Makes all layers visible
No layers selectable except current layer
Layer color and pattern
Layer type (drawing, pin, text, etc.)
Layer name
Currently visible and selectable
Palette
Layers
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

The CMOS inverter that you will be creating consists of seven photolithographic layers, n-well (NW), poly-silicon (PO), n+ diffusion (NP), p+ diffusion (PP), tungsten contact (CO), oxide diffusion (OD), and copper metal-layer 1 (M1). Refer to the traditional MOSFET cross-sectional diagram below for reference as to the location of each layer in the physical structure of a transistor. The locations of the n-well , n+, and p+ diffusion regions are obvious. The OD layer is placed over the entire source, gate, and drain of the device to define the gate oxide and active transistor regions. The tungsten CO contacts are used to connect the substrate and diffusion regions to the metal 1 layer, not shown in the diagram. Poly-silicon is used as the conductor above the gate oxide and also requires a CO contact to reach metal 1.



Layout Tips

DRD Notify and DRD Enforce are two features of Cadence Virtuoso that are generally helpful (but mnot 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.

The default background grid is by default large. To change the grid, go to the “Options: Display” dropdown in the layout editor and change “Minor Spacing” and “Major Spacing” under “Grid Controls” to something more reasonable, such as 0.1 and 0.5.

By default, the layout editor only allows moving polygons orthogonally (vertical or horizontal only). This slows down the editing process, and can be changed in the “Options: Display” dropdown under “Snap Modes.” Change the “Edit” option to “anyAngle” to be able to freely move created polygons.

Frequently used layout hotkeys include:

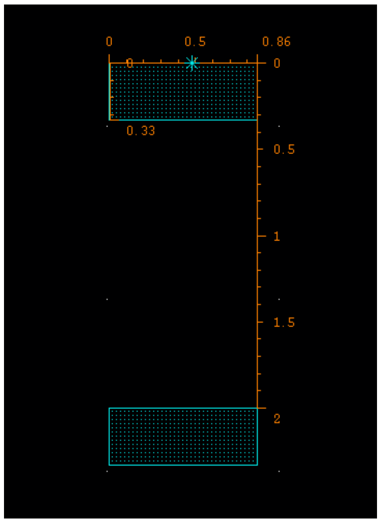
f:	fit
shift-f:	in hierarchical layout show all levels as if flat
ctrl-f:	hide all hierarchy and show only outline of instances
r:	rectangle
q:	property of an object
ctrl-z:	zoom in
shift-z:	zoom out
f2:	save
t:	tap: if you select a layer, saw NW in layout and press tap, that layer gets selected in LSW (layer selection window). Then you can use r to draw rectangles of that layer: Normally we can select a layer in LSW and when we press “r” a rectangle of that layer gets drawn. But if we press “t” (called tap) and then select a shape/rectangle, the layer of that shape/rectangle gets selected in LSW and then pressing r cretes rectangle of that layer
p:	path: makes a min width path of the layer selected in LSW : If some layer is selected in LSW, then “p” starts to create a path of that layer with width same as the min width for that layer defined in the drc (if it is loaded into icfb)
ctrl-a:	select all
ctrl-d:	deselect all
c:	copy
m:	move: move a whole rectangle
s:	stretch: can stretch just a side of a rectangle
f:	fit
k:	ruler
i:	add an instance
u:	undo
shift-U:	redo
shift-r:	reshape: use it to reshape a layer so as to make it bigger, e.g., a turn in a metal wire
shift-c:	chop: chop a rectangle, i.e., reduce its size as you want
shift-m:	merge all rectangles selected as per their layer-purpose name (lpp) (layer purpose pair) for example (“met1” “drawing”) is a 2 value array (pair) having value of layer (met1) and its purpose (drawing), i.e. all selected and touching rectangles or paths in same layer-purpose pair (say all met1 drawings) (my above comment will clarify this too) get merged into one.
e:	display options like grid size, snap size etc
f6:	redraw

rLayout Drawing

This tutorial walks through drawing the two transistors step-by-step. Typically, designers work with pre-made transistors provided in the PDK. However, drawing each photolithographic mask individually will help you understand exactly how transistors are formed.

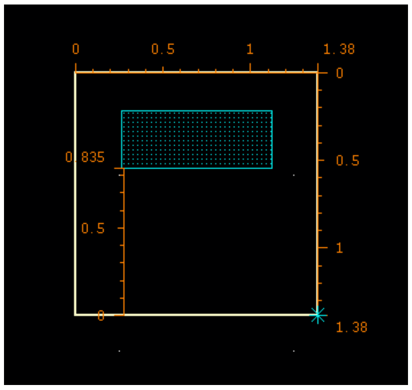
1. Power Straps

Select the M1 metal layer in the layer palette. Using the "r" hotkey, draw two rectangles with a width of 0.86µm, height of 0.33µm, and a vertical pitch of 2µm, as shown in the figure below. Use the "k" hotkey to draw rulers for checking the dimensions. You can type in the rectangle height and width directly by pulling up the rectangle properties. This can be done with the "q" hotkey with a rectangle selected.



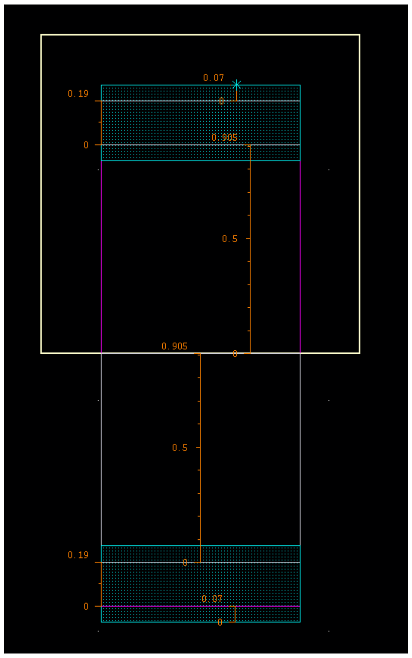
1. N-Well

Draw a 1.38µm by 1.38µm rectangle of NW (nwell) to define the region for the PMOS transistor. Center this rectangle horizontally around the metal power rails with the align hotkey "a". To align the edge of the nwell with the center of the inverter, place it 0.835µm away from the bottom of the top rail, as shown:



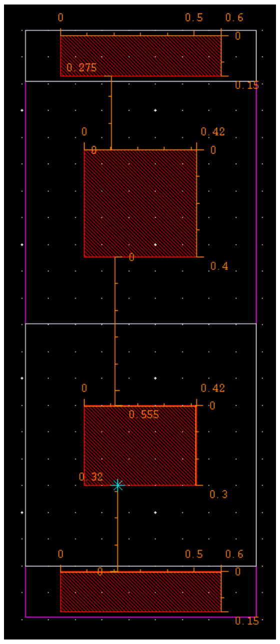
2. P-Plus and N-Plus

Next, define the p-plus (PP) and n-plus (NP) regions as shown. These masks distinguish whether the region exposed by the oxide diffusion (OD) mask will be p-doped or n-doped. Define a PP body contact under the top power rail, a NP region for the PMOS transistor, a PP region for the NMOS transistor, and an NP body contact under the bottom power rail. These regions should all be touching (but not overlapping) and extend to the horizontal edges of the power rails.



3. Active Diffusion

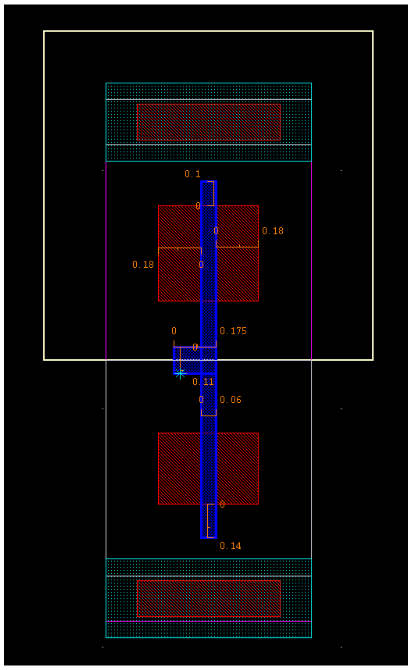
The diffusion (or active) region represents the area that is exposed for the diffusion of n-type or p-type dopants. The height of the OD shape determines the channel width of the NMOS or PMOS device. Mismatch between the drawn shape and the width specified in the schematic will result in an LVS error. Diffusion regions are also required to make an ohmic connection between the body and power straps. Draw four OD rectangles with the specified dimensions, ensuring that the PMOS rectangle is 0.4µm tall and the NMOS rectangle is 0.3µm tall, to match our desired transistor sizes. Center the body contact OD under the M1 rails and center the transistor OD on their respective NP and PP regions.



4. Polysilicon

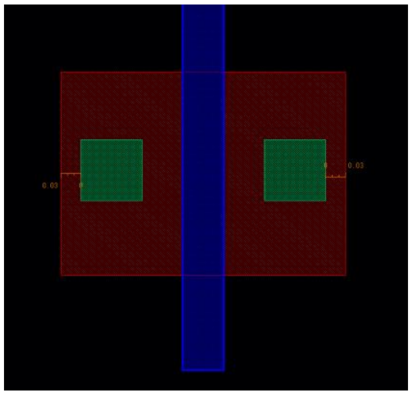
Polysilicon, a.k.a. “poly,” is the material used for the transistor gate. It is easily grown on top of the silicon substrate and gate oxide, making it the preferred conductor for the transistor gate. Create the inverter gate out of the PO layer, following the dimensions shown. The protruding rectangle in the center is for a gate contact to connect to metal.

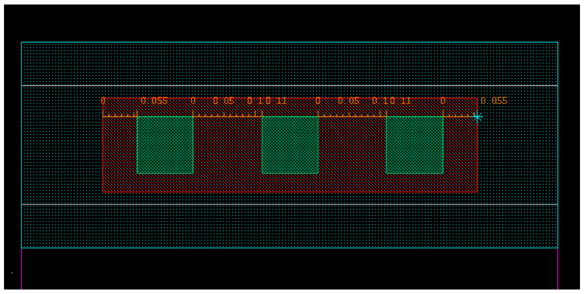
There is an intentional DRC error here. We will fix it later, to demonstrate how DRC identifies errors.

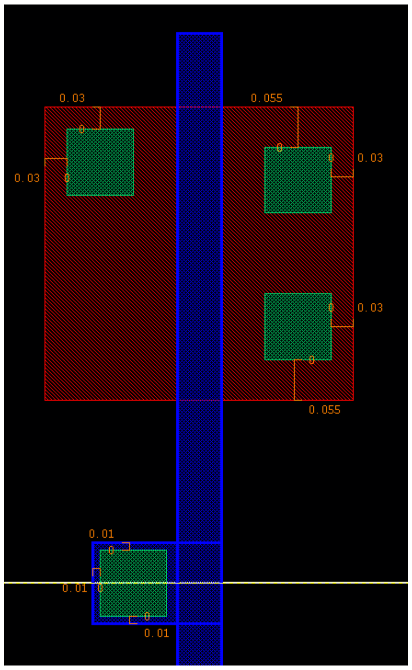


5. Contacts

Place contacts to connect the active diffusion and polysilicon to the metal conductor above. In the 65nm process, contacts must be square with dimensions of exactly 0.09µm x 0.09µm. These contacts are fairly resistive, so it is good practice to place as many as possible in a design. Instead of drawing individual rectangles every time, draw one and copy it to place the rest, with the "c" hotkey. There is a mandatory 0.11µm minimum spacing between contacts. Arrange the contacts to look like the following figures.

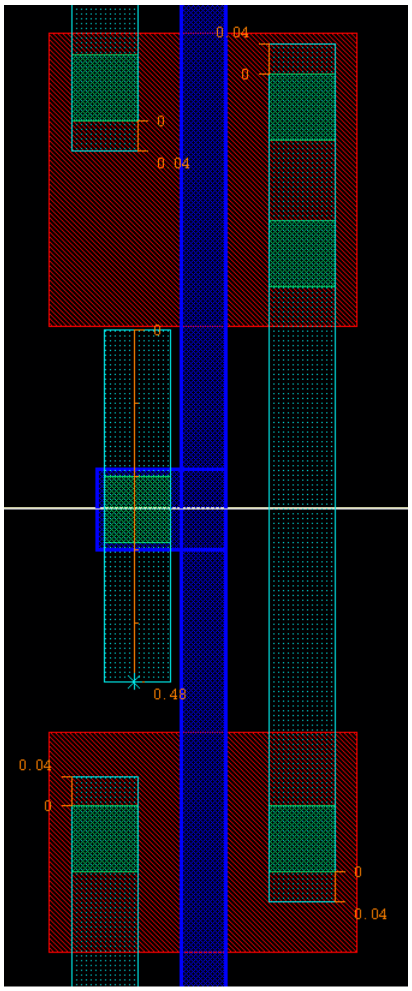






6. Metal Interconnects

Now the two devices are completely defined. Make the final interconnections on M1 to complete the inverter drawing. The design rule for metal on contacts is simple, the metal must extend 0.04µm over the contact on two sides. Draw metal rectangles connecting the sources to the power rails and the drains together. Also add metal above the gate contact, which would be used later for connecting to other cells.



The top-down layout view of each layer gives the deceiving appearance that the layers are flat. Rather, the polysilicon, contacts, and metal layers have appreciable height. For instance, the contact squares are taller (out of the screen in the layout viewer) then they are long or wide. The M1 layer is about 0.18µm tall, two times the minimum width used for the traces in this layout.

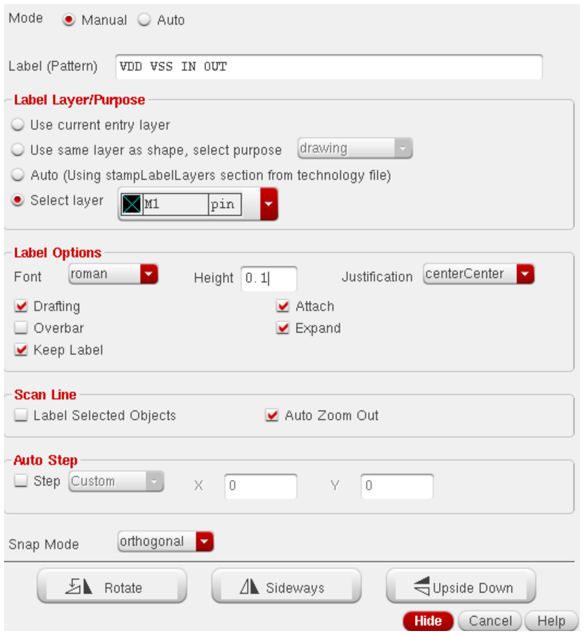
7. Pin Labels

In order for the tools to perform various design checks, as well as for netlist creation, pins must be defined on the physical layout. Press the "l" hotkey to open up the label popup menu, as shown.

You can place all of the pins in one operation, so input all of the pin names separated by spaces.

Make sure you click the "Select layer" radio button and select the "M1 pin" layer from the drop-down. Only this layer will be correctly interpreted as a pin by LVS.

Format your label as 0.1µm tall and roman font, to make your layout easier to read.



Mode [● Manual] ○ Auto

Label (Pattern) VDD VSS IN OUT

Label Layer/Purpose
- ○ Use current entry layer
- ○ Use same layer as shape, select purpose drawing
- ○ Auto (Using stampLabelLayers section from technology file)
- [● Select layer] M1 pin

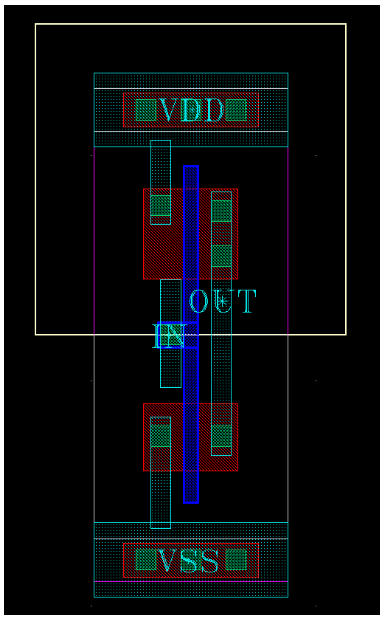
Label Options
- Font roman
- Height 0.1
- Justification centerCenter
- ✓ Drafting
- ☐ Overbar
- ✓ Keep Label
- ✓ Attach
- ✓ Expand

Scan Line
- ☐ Label Selected Objects
- ✓ Auto Zoom Out

Auto Step
- ☐ Step Custom X 0 Y 0

Snap Mode orthogonal

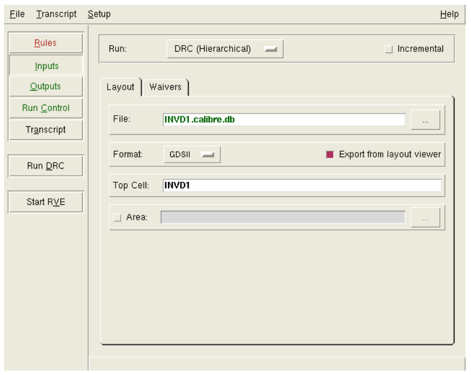
Place the pins as shown. Your inverter is now complete and ready for DRC and LVS checks.



Design Rule Checking (DRC)

Although the designer draws layouts as perfect polygons, the shapes are manufactured with a variety of physical processes that are all subject to their own limitations. These physical limits define the size, shape, and spacing of shapes on each mask layer. A PDK comes with a several-hundred-page document defining every design rule for the process. A designer cannot possibly follow every rule with hand-checking, so DRC was developed to automatically a layout geometry for coincidence with design rules.

In the layout editor, open Calibre (program that runs DRC and LVS) with the "Calibre: Run DRC" dropdown. The following window will appear (hit "Cancel" for the "Load Runset File" window). If a warning about "Undefined Packets" appears, you can likely ignore it.

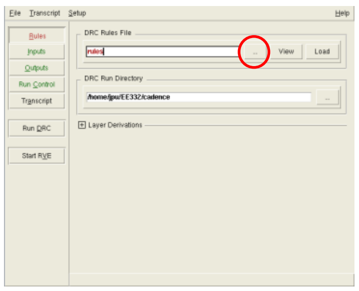


Run: DRC (Hierarchical)
Incremental
File: INVD1.calibre.db
Format: GDSII
Export from layout viewer
Top Cell: INVD1
Area:

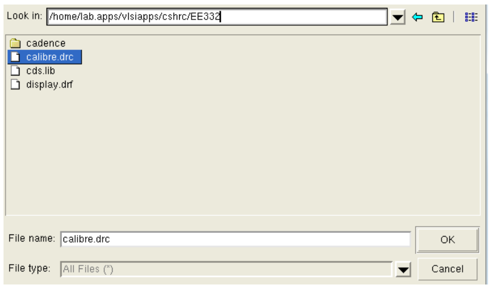
Click on the "Rules" tab on the left.

Click the "…" browse to locate the rules file for the 65nm process, as shown below:

Select /home/lab.apps/vlsiapps/cshrc/EE332/calibre.drc



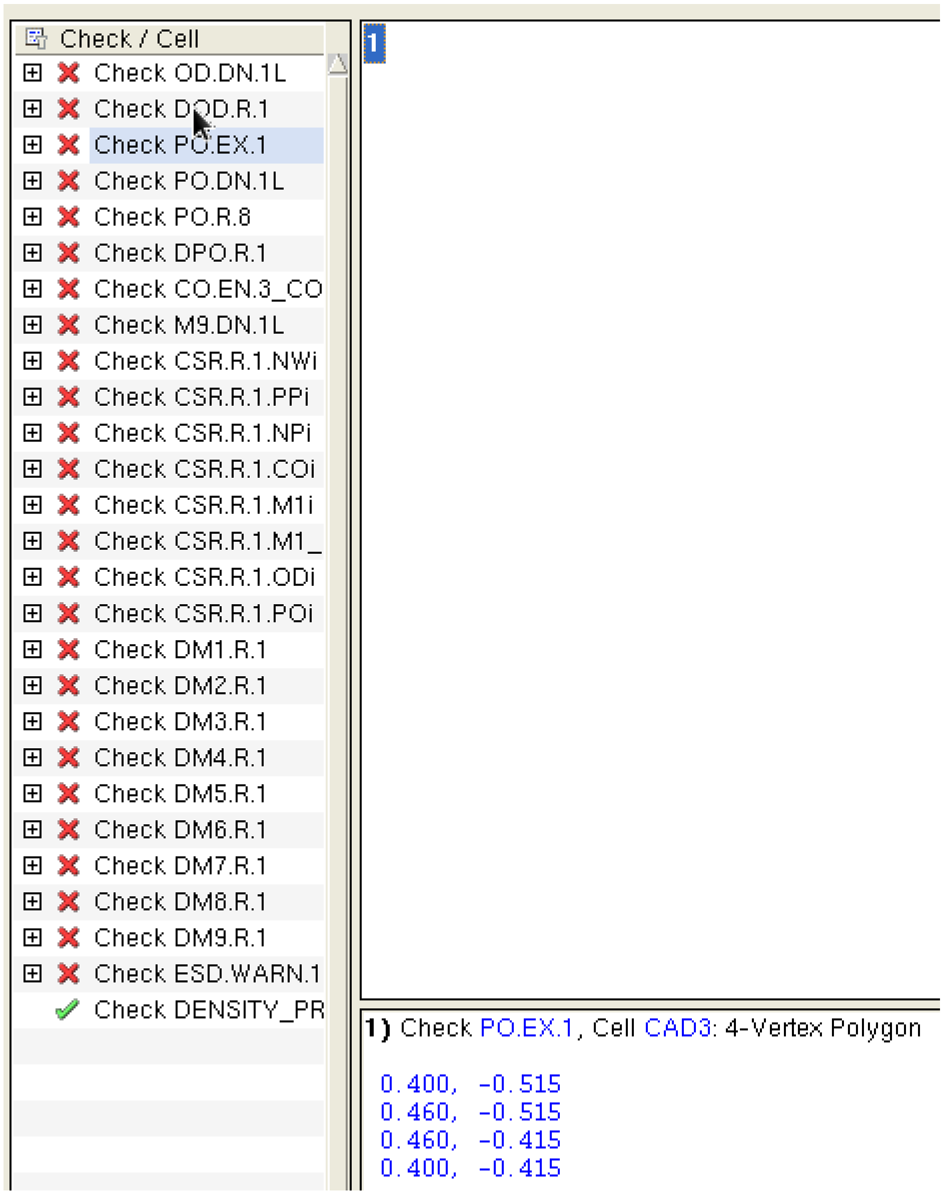
DRC Rules File: rules
DRC Run Directory: /home/jpu/EE332/cadence



Look in: /home/lab.apps/vlsiapps/cshrc/EE332
cadence
calibre.drc
cds.lib
display.drf
File name: calibre.drc
File type: All Files (*)

NOTE: If the Calibre option does not appear in Virtuoso Layout, please see Appendix A for a possible fix.

Hit “Run DRC” to perform the check. An error report window will pop up:



```
Check / Cell
⊞ ✗ Check OD.DN.1L
⊞ ✗ Check DOD.R.1
⊞ ✗ Check PO.EX.1
⊞ ✗ Check PO.DN.1L
⊞ ✗ Check PO.R.8
⊞ ✗ Check DPO.R.1
⊞ ✗ Check CO.EN.3_CO
⊞ ✗ Check M9.DN.1L
⊞ ✗ Check CSR.R.1.NWi
⊞ ✗ Check CSR.R.1.PPi
⊞ ✗ Check CSR.R.1.NPi
⊞ ✗ Check CSR.R.1.COi
⊞ ✗ Check CSR.R.1.M1i
⊞ ✗ Check CSR.R.1.M1_
⊞ ✗ Check CSR.R.1.ODi
⊞ ✗ Check CSR.R.1.POi
⊞ ✗ Check DM1.R.1
⊞ ✗ Check DM2.R.1
⊞ ✗ Check DM3.R.1
⊞ ✗ Check DM4.R.1
⊞ ✗ Check DM5.R.1
⊞ ✗ Check DM6.R.1
⊞ ✗ Check DM7.R.1
⊞ ✗ Check DM8.R.1
⊞ ✗ Check DM9.R.1
⊞ ✗ Check ESD.WARN.1
  ✓ Check DENSITY_PR

1

1) Check PO.EX.1, Cell CAD3: 4-Vertex Polygon

 0.400,  -0.515
 0.460,  -0.515
 0.460,  -0.415
 0.400,  -0.415
```

Although there are a number of errors, only two of them are the real problem to us at this point, while the rest can be completely ignored.

All these currently negligible can be categorized into 4 groups:

1. **Density issue**: floating metal fill or dummy material layers need to be added into the chip-level design to reduce variation in dielectric thickness due to dishing and erosion, as well as to increase planarity and maintain nearly uniform pattern density on the scale of the chemical mechanical planarization (CMP) process. In other words, most of the metal and semiconductor layers are required to be placed uniformly on the entire chip with a certain density. The following errors are related to density violation:

OD.DN.1L – diffusion layer density violation

PO.DN.1L – polysilicon layer density violation

M9.DN.1L – 9th metal layer density violation

DOD.R.1 – lacking dummy diffusion layer

DPO.R.1 – lacking dummy polysilicon layer

DM1.R.1 – lacking 1st dummy metal layer

DM2.R.1 – lacking 2nd dummy metal layer

DM3.R.1 – lacking 3rd dummy metal layer

DM4.R.1 – lacking 4th dummy metal layer

DM5.R.1 – lacking 5th dummy metal layer

DM6.R.1 – lacking 6th dummy metal layer

DM7.R.1 – lacking 7th dummy metal layer

DM8.R.1 – lacking 8th dummy metal layer

DM9.R.1 – lacking 9th dummy metal layer

2. **Floating gate**: this error is necessary for a chip-level design and not important in the local DRC. It is basically saying the gates of the MOSFETs are not connected to the pads of the chip anywhere.

   PO.R.8 – floating gate error

3. **Seal ring**: again, this error is for a whole chip design. An IC chip will need a seal ring, which is not required here.

   CSR.R.1.NWi

   CSR.R.1.PPi

   CSR.R.1.NPi

   CSR.R.1.COi

   CSR.R.1.M1i

   CSR.R.1.M1_reali

   CSR.R.1.ODi

   CSR.R.1.Poi

4. **Electrostatic discharge (ESD) protection**: to an IC chip, some sorts of ESD protection are necessary, and this warning is excited from the lack of it.

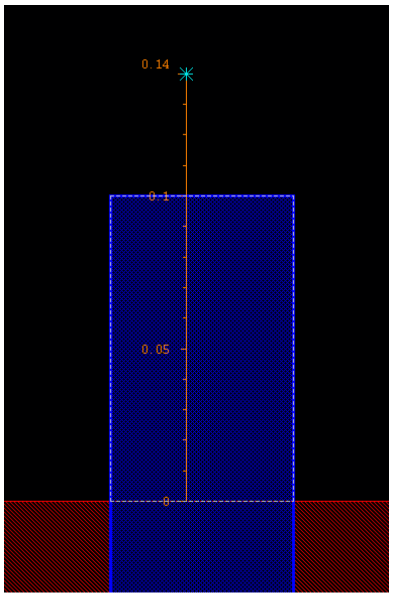
   ESD.WARN.1– lacking ESD protection

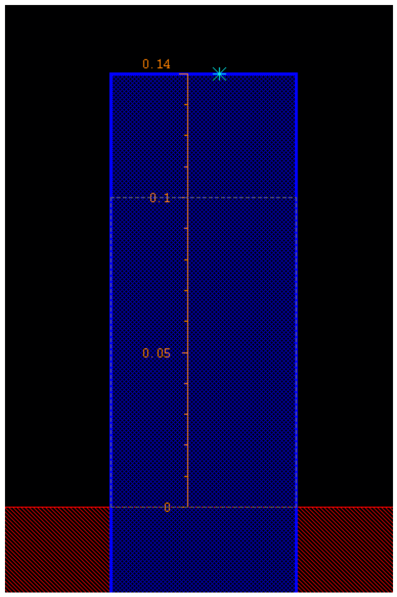
With the above explanation, we actually only need to deal with two errors: *PO.EX.1* and *CO.EN.3_CO*.

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 "PO.EX" with the details:

> PO.EX.1 { @ Extension on OD (end-cap) >= 0.14 (ENC OD POLY < PO_EX_1 ABUT < 89.5 OPPOSITE SINGULAR REGION) NOT INSIDE LOGO }

The information to be gleaned from the message is that the PO must extend at least 0.14um past the end of the OD ("@ Extension on OD (end-cap) >= 0.14"). Stretch the extension of poly-silicon so that it extends 0.14µm past the OD rectangle to fix this error.

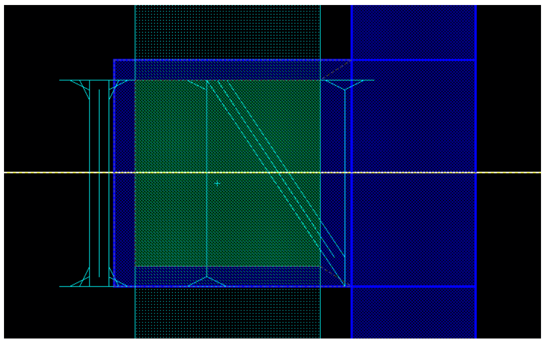


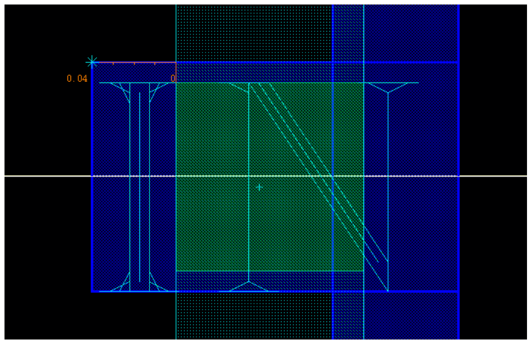


The next error, "CO.EN," has the details:

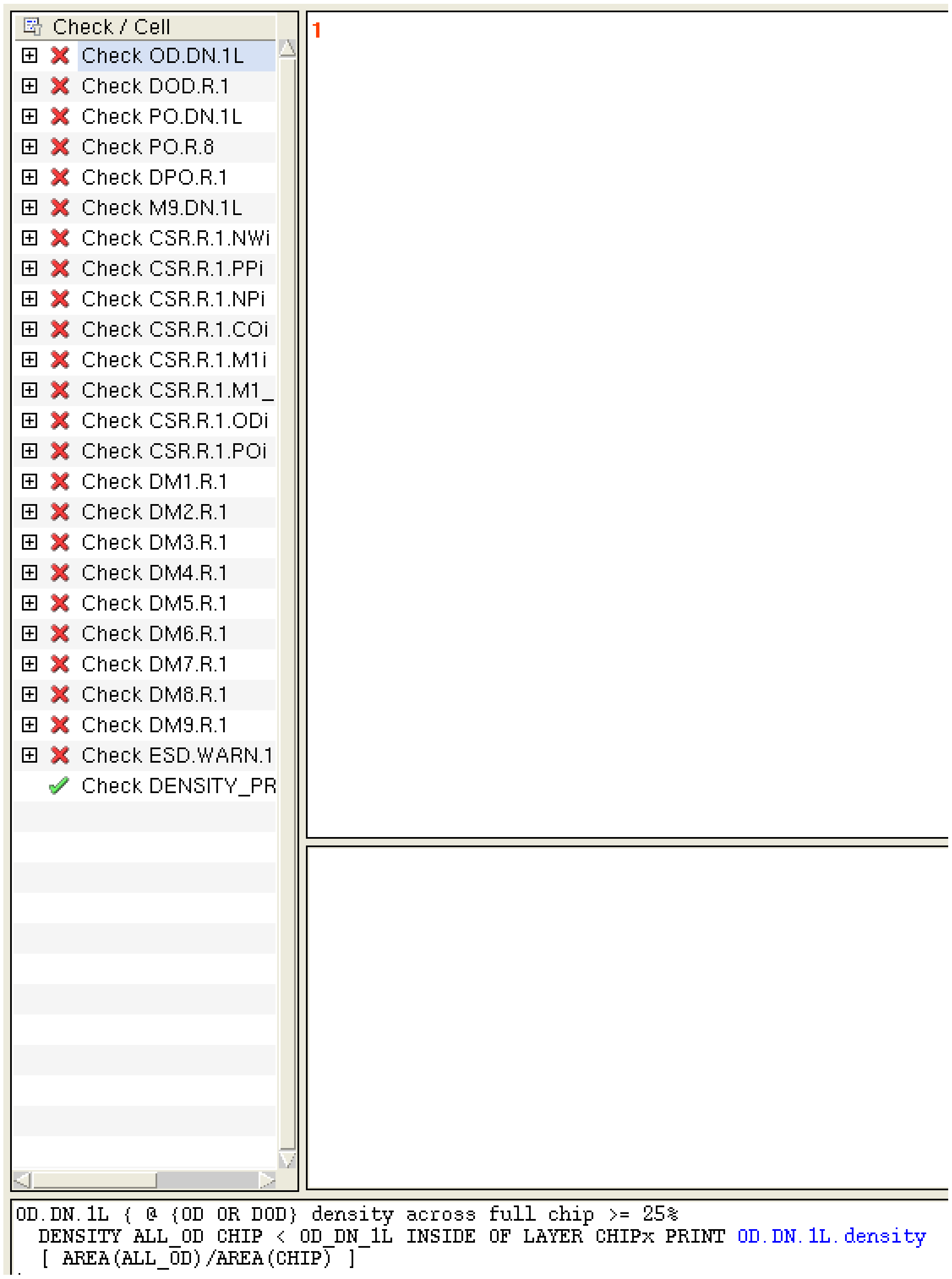
> CO.EN3_CO.EN.4 { @ Enclosure by PO [at least two opposite sides] >= 0.04 ,or [all sides] >= 0.03 X = RECTANGLE ENCLOSURE CO POLYs ABUT >0 < 90 SINGULAR GOOD CO_EN_2 CO_EN_3 OPPOSITE CO_EN_2 CO_EN_3 OPPOSITE ENC X POLYs < CO_EN_4 ABUT < 90 SINGULAR REGION }

This is similar to the M1/contact enclosure rule that was followed drawing the metal interconnects. There must be at least 0.04µm of PO overhang on opposite edges of the CO contact. To fix this error, move the gate contact, and its connected M1 line and pin 0.03µm to the right.





Re-run DRC, which should now yield a "clean" report:



Check / Cell

- ⊞ ✗ Check OD.DN.1L
- ⊞ ✗ Check DOD.R.1
- ⊞ ✗ Check PO.DN.1L
- ⊞ ✗ Check PO.R.8
- ⊞ ✗ Check DPO.R.1
- ⊞ ✗ Check M9.DN.1L
- ⊞ ✗ Check CSR.R.1.NWi
- ⊞ ✗ Check CSR.R.1.PPi
- ⊞ ✗ Check CSR.R.1.NPi
- ⊞ ✗ Check CSR.R.1.COi
- ⊞ ✗ Check CSR.R.1.M1i
- ⊞ ✗ Check CSR.R.1.M1_
- ⊞ ✗ Check CSR.R.1.ODi
- ⊞ ✗ Check CSR.R.1.POi
- ⊞ ✗ Check DM1.R.1
- ⊞ ✗ Check DM2.R.1
- ⊞ ✗ Check DM3.R.1
- ⊞ ✗ Check DM4.R.1
- ⊞ ✗ Check DM5.R.1
- ⊞ ✗ Check DM6.R.1
- ⊞ ✗ Check DM7.R.1
- ⊞ ✗ Check DM8.R.1
- ⊞ ✗ Check DM9.R.1
- ⊞ ✗ Check ESD.WARN.1
- ✓ Check DENSITY_PR

1

```
OD.DN.1L { @ {OD OR DOD} density across full chip >= 25%
  DENSITY ALL_OD CHIP < OD_DN_1L INSIDE OF LAYER CHIPx PRINT OD.DN.1L.density
  [ AREA(ALL_OD)/AREA(CHIP) ]
```

Layout versus schematic (LVS)

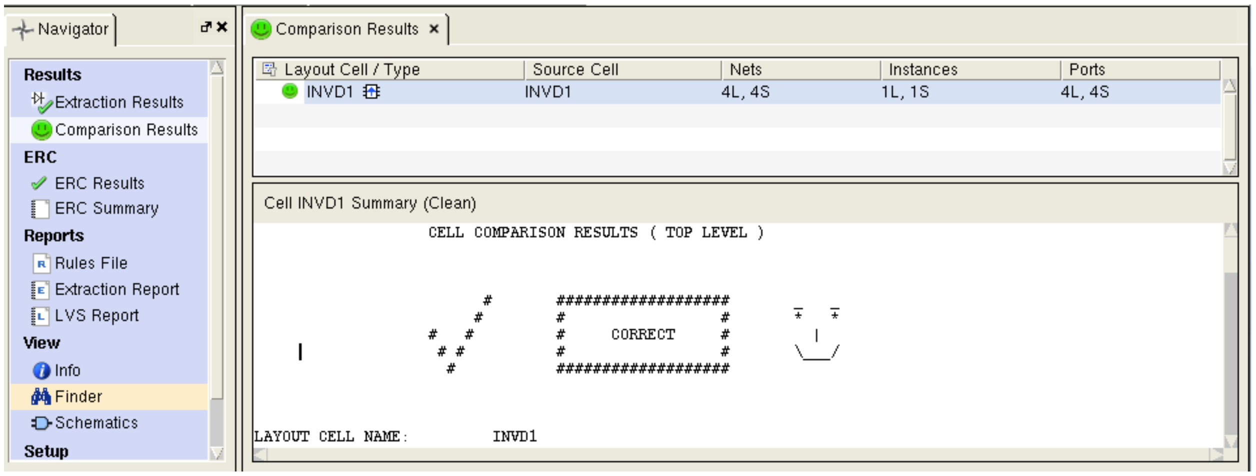
DRC ensures that the layout shapes meet manufacturability requirements, but it does not check that the layout is logically equivalent to the schematic. That is, LVS checks whether the layout performs the same mathematical function as the schematic, and that the physical properties of the transistors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

As with DRC, select "Calibre: Run LVS." There will be another prompt to load a runset file, cancel this dialog again. Just as before for DRC, select the following rules file for LVS:

/home/lab.apps/vlsiapps/cshrc/EE332/calibre.lvs

Select "Inputs" on the left side of the main LVS window, and make sure that "Export from layout viewer" is selected under the "Layout" tab and "Export from schematic viewer" is selected under the "Netlist" tab. This allows LVS to automatically generate netlists from the most current versions of the schematic and netlist.

Select "Run LVS" to perform the check. If your schematic and layout match, the following window will appear. This simple design does not have the potential for many errors. More complicated layouts almost always end up with LVS errors, including: misplaced labels, incorrect MOSFET sizes, accidental shorts/opens, and misplaced vias (connections between metal layers).



Layout Cell / Type	Source Cell	Nets	Instances	Ports
INVD1	INVD1	4L, 4S	1L, 1S	4L, 4S

Cell INVD1 Summary (Clean)

```
                    CELL COMPARISON RESULTS ( TOP LEVEL )


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


LAYOUT CELL NAME:          INVD1
```

Netlist and Parasitics Extraction

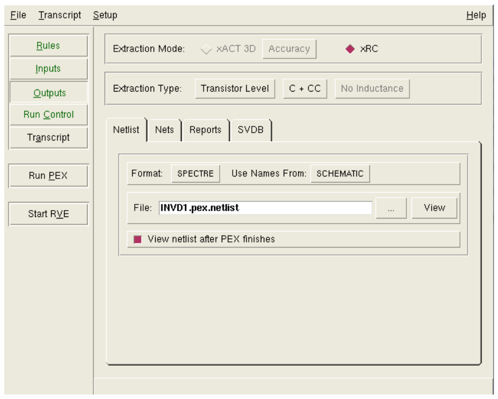
Extraction is the translation of an integrated circuit layout back into the electrical circuit (netlist) it is intended to represent. The extracted circuit is needed for various purposes including circuit simulation, timing analysis, power analysis, and more. Compared to a schematic netlist, the layout 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.

As with DRC and LVS, open PEX with "Calibre: Run PEX" in the layout viewer window.

Input the PEX rules file in the Rules tab. The file is located at:

/home/lab.apps/vlsiapps/cshrc/EE332/calibre.rcx

In the Outputs tab, select the "C+CC" extraction option. This extracts capacitance to ground and coupling capacitance between each net. Also choose the "SPECTRE" output option; this determines the output netlist format.



Extraction Mode: xACT 3D Accuracy [✓ xRC]

Extraction Type: Transistor Level C + CC No Inductance

Format: SPECTRE Use Names From: SCHEMATIC

File: INVD1.pex.netlist

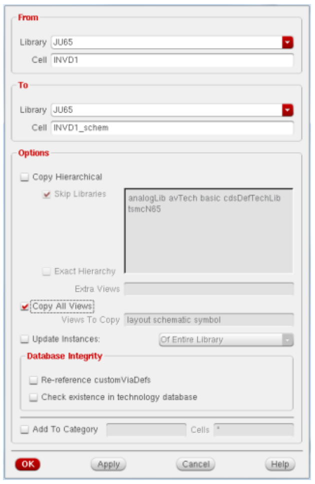
[✓] View netlist after PEX finishes

Now, select "Run PEX" to extract the parasitics and generate a spectre netlist. The netlist will appear once PEX is finished. You can now close all Calibre windows, we will reference this netlist in our simulation setup later.

Simulating the Inverter: Schematic vs Post-Layout

Now we will simulate the schematic symbol inverter alongside the extracted netlist and compare the differences in delay and power consumption due to parasitic elements. First, we must organize our library so that the simulator knows the difference between the designs.

In the Library Manager, copy the INVD1 cell view to a cell named INVD1_schem. Make sure you "Copy All Views"



From
Library JU65
Cell INVD1

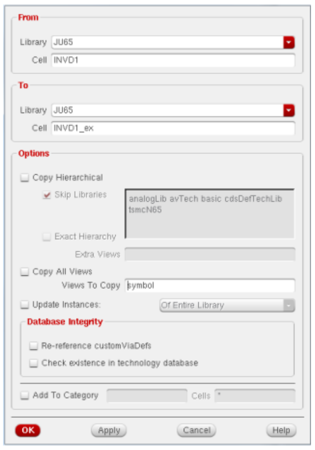
To
Library JU65
Cell INVD1_schem

Options
[ ] Copy Hierarchical
[✓] Skip Libraries: analogLib avTech basic cdsDefTechLib tsmcN65
[ ] Exact Hierarchy
Extra Views
[✓] Copy All Views
Views To Copy: layout schematic symbol
[ ] Update Instances: Of Entire Library

Database Integrity
[ ] Re-reference customViaDefs
[ ] Check existence in technology database

[ ] Add To Category Cells *

Next, copy the INVD1 cell view again to a cell named INVD1_ex. Only include the symbol view this time.



From
Library JU65
Cell INVD1

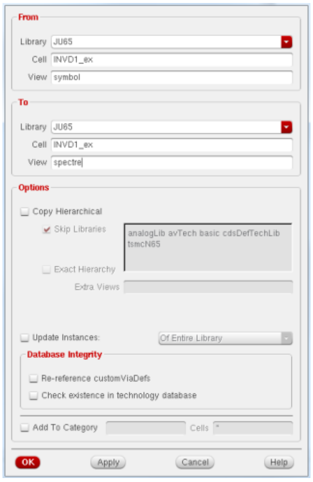
To
Library JU65
Cell INVD1_ex

Options
[ ] Copy Hierarchical
[✓] Skip Libraries: analogLib avTech basic cdsDefTechLib tsmcN65
[ ] Exact Hierarchy
Extra Views
[ ] Copy All Views
Views To Copy: symbol
[ ] Update Instances: Of Entire Library

Database Integrity
[ ] Re-reference customViaDefs
[ ] Check existence in technology database

[ ] Add To Category Cells *

Then, copy the INVD1 symbol view to a "spectre" view, which serves as a placeholder for the netlist generated in the PEX extraction.



From

Library	JU65
Cell	INVD1_ex
View	symbol

To

Library	JU65
Cell	INVD1_ex
View	spectre

Options

☐ Copy Hierarchical

✓ Skip Libraries: analogLib avTech basic cdsDefTechLib tsmcN65

☐ Exact Hierarchy

Extra Views

☐ Update Instances: Of Entire Library

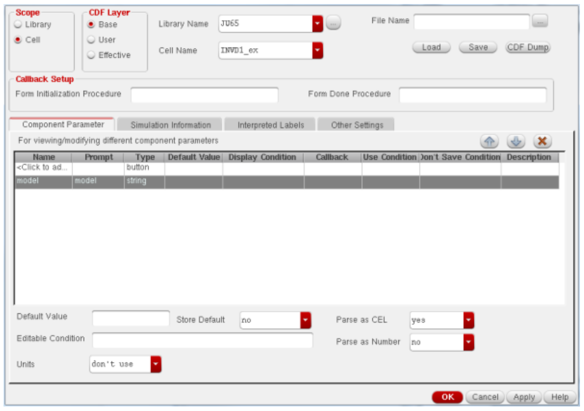
Database Integrity

☐ Re-reference customViaDefs

☐ Check existence in technology database

☐ Add To Category   Cells *

Now, go to the main Cadence CIW window. Select "Tools: CDF: Edit" from the drop-downs. Fill out the form as follows: Edit the Base Cell CDF for the INVD1_ex cell in your library. Add a "Component Parameter" named "model," and select the string type. Make sure to select "yes" under "Parse as CEL."



Scope: Library [✓ Cell]

CDF Layer: [✓ Base] User Effective

Library Name: JU65

Cell Name: INVD1_ex

File Name:

Callback Setup

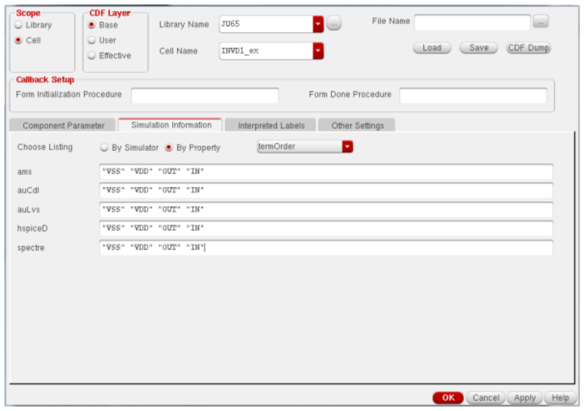
Form Initialization Procedure: Form Done Procedure:

For viewing/modifying different component parameters

Name	Prompt	Type	Default Value	Display Condition	Callback	Use Condition	Don't Save Condition	Description
<Click to ad...		button						
model	model	string						

Default Value		Store Default	no
Parse as CEL	yes	Parse as Number	no
Editable Condition		Units	don't use

Under the "Simulation Information" tab, select the "By Property" radio button and "termOrder" under the dropdown. We must make the pin order match that of our extracted netlist, or the simulator will mix up the input/output pins. The pin order should match the order from the file INVD1.pex.netlist in your working directory. Open this file in a text editor to check the order and input the pins in quotations, separated by spaces, in the order from your .pex.netlist file.



Scope: Library [✓ Cell]

CDF Layer: [✓ Base] User Effective

Library Name: JU65

Cell Name: INVD1_ex

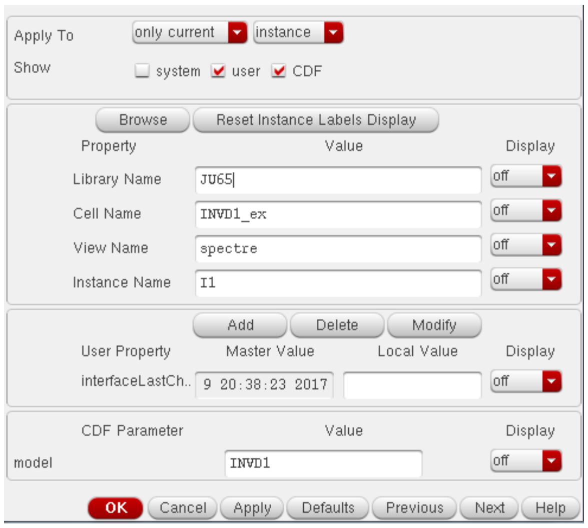
Choose Listing: By Simulator [✓ By Property]   termOrder

ams	"VSS" "VDD" "OUT" "IN"
auCdl	"VSS" "VDD" "OUT" "IN"
auLvs	"VSS" "VDD" "OUT" "IN"
hspiceD	"VSS" "VDD" "OUT" "IN"
spectre	"VSS" "VDD" "OUT" "IN"

Close the CDF editor. Now we're ready to create a simulation test bench.

In the Library Manager, create a new schematic cell view called "tb_INVD1."

Populate this schematic with the "INVD1_schem" symbol view and "INVD1_ex" spectre view from your library. Under the INVD1_ex properties, fill in the model property (this is what was created when editing the CDF) with “INVD1.” This tells the spectre placeholder the name of the model netlist to use for simulation.



Apply To	only current	instance	
Show	☐ system	✓ user	✓ CDF

Browse | Reset Instance Labels Display

Property	Value	Display
Library Name	JU65	off
Cell Name	INVD1_ex	off
View Name	spectre	off
Instance Name	I1	off

Add | Delete | Modify

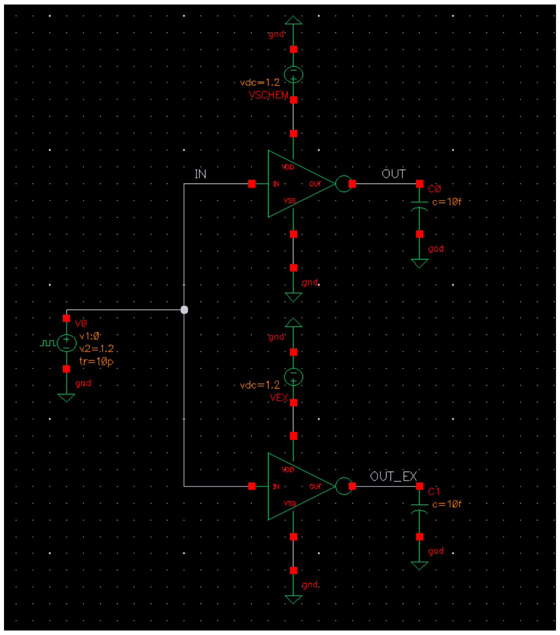
User Property	Master Value	Local Value	Display
interfaceLastCh..	9 20:38:23 2017		off

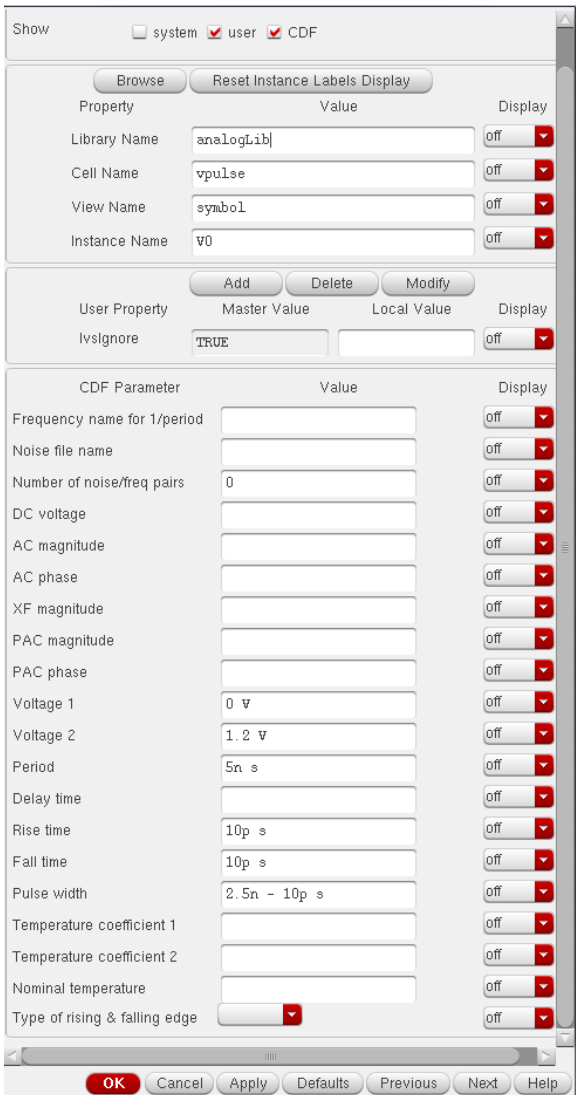
CDF Parameter	Value	Display
model	INVD1	off

OK | Cancel | Apply | Defaults | Previous | Next | Help

The completed schematic should look something like the image below. Add 10fF load capacitors to each inverter, cell “cap” from the “analogLib” library. Power each inverter with a separate 1.2V “vdc” voltage source, so that we can independently measure the power consumptions. I recommend naming the voltage sources “VSCHEM” and “VEX” so that you can tell them apart in the waveform viewer. Ground the VSS pins and the negative terminals of the voltage sources and capacitors with the “gnd” cell from the “analogLib” library.

Finally, add a “vpulse” voltage source from the “analogLib” library to drive the input terminals. Appropriate parameters can be found to the right.





Show [ ] system [✓ user] [✓ CDF]

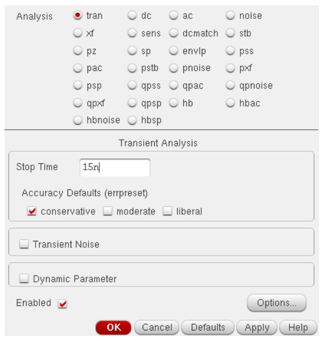
Property	Value	Display
Library Name	analogLib	off
Cell Name	vpulse	off
View Name	symbol	off
Instance Name	V0	off

User Property	Master Value	Local Value	Display
lvsIgnore	TRUE		off

CDF Parameter	Value	Display
Frequency name for 1/period		off
Noise file name		off
Number of noise/freq pairs	0	off
DC voltage		off
AC magnitude		off
AC phase		off
XF magnitude		off
PAC magnitude		off
PAC phase		off
Voltage 1	0 V	off
Voltage 2	1.2 V	off
Period	5n s	off
Delay time		off
Rise time	10p s	off
Fall time	10p s	off
Pulse width	2.5n - 10p s	off
Temperature coefficient 1		off
Temperature coefficient 2		off
Nominal temperature		off
Type of rising & falling edge		off

Label in the input and output pins with descriptive names, and “Check and Save” the schematic. Now launch the simulator with “Launch: ADE L” in the schematic editor dropdowns.

Select the “Analyses: Choose” dropdown and enable transient analysis for 15ns under conservative accuracy. This will capture several transitions of the inverter, as the driver has a period of 5ns.



Analysis: [✓ tran] dc ac noise xf sens dcmatch stb pz sp envlp pss pac pstb pnoise pxf psp qpss qpac qpnoise qpxf qpsp hb hbac hbnoise hbsp

Transient Analysis

Stop Time 15n

Accuracy Defaults (errpreset)

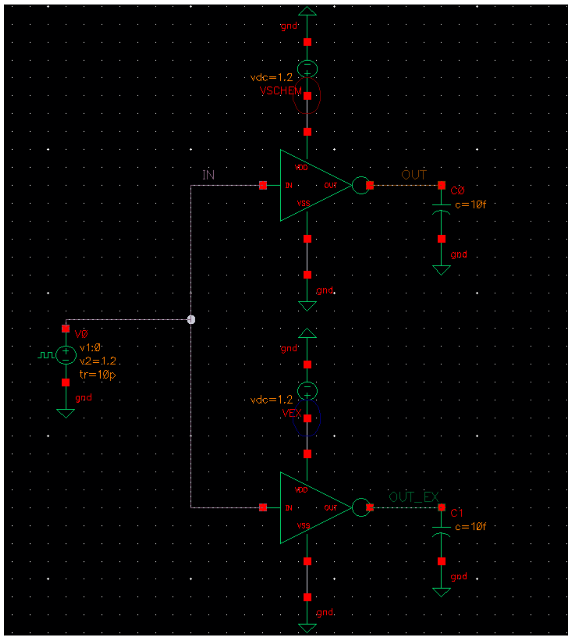
[✓ conservative] [ ] moderate [ ] liberal

[ ] Transient Noise

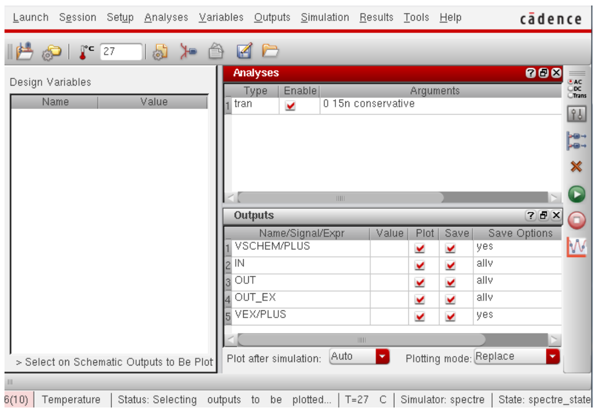
[ ] Dynamic Parameter

Enabled [✓]

Select the "Outputs: Choose on Schematic" dropdown and select the input and output terminals on the schematic. Also click on the positive voltage source pins to measure the current consumed by the inverters. When you're finished, the schematic should look like this:



In the ADE window, make sure you select "Save" next to all of the outputs (or currents won't plot).



27

Analyses

	Type	Enable	Arguments
1	tran	✓	0 15n conservative

Outputs

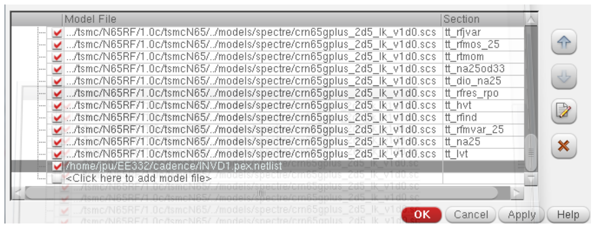
	Name/Signal/Expr	Value	Plot	Save	Save Options
1	VSCHEM/PLUS		✓	✓	yes
2	IN		✓	✓	allv
3	OUT		✓	✓	allv
4	OUT_EX		✓	✓	allv
5	VEX/PLUS		✓	✓	yes

> Select on Schematic Outputs to Be Plot

Plot after simulation: Auto   Plotting mode: Replace

6(10) | Temperature | Status: Selecting outputs to be plotted... | T=27 C | Simulator: spectre | State: spectre_state

Finally, under the "Setup: Model Libraries" dropdown, add a model file to the end of the existing list, the INVD1.pex.netlist file created by PEX. It should be in your working directory. After completing this step, you are ready to run the simulation.



	Model File	Section
✓	.../tsmc/N65RF/1.0c/tsmcN65/../models/spectre/crn65gplus_2d5_lk_v1d0.scs	tt_rfjvar
✓	.../tsmc/N65RF/1.0c/tsmcN65/../models/spectre/crn65gplus_2d5_lk_v1d0.scs	tt_rfmos_25
✓	.../tsmc/N65RF/1.0c/tsmcN65/../models/spectre/crn65gplus_2d5_lk_v1d0.scs	tt_rtmom
✓	.../tsmc/N65RF/1.0c/tsmcN65/../models/spectre/crn65gplus_2d5_lk_v1d0.scs	tt_na25od33
✓	.../tsmc/N65RF/1.0c/tsmcN65/../models/spectre/crn65gplus_2d5_lk_v1d0.scs	tt_dio_na25
✓	.../tsmc/N65RF/1.0c/tsmcN65/../models/spectre/crn65gplus_2d5_lk_v1d0.scs	tt_rfres_rpo
✓	.../tsmc/N65RF/1.0c/tsmcN65/../models/spectre/crn65gplus_2d5_lk_v1d0.scs	tt_hvt
✓	.../tsmc/N65RF/1.0c/tsmcN65/../models/spectre/crn65gplus_2d5_lk_v1d0.scs	tt_rfind
✓	.../tsmc/N65RF/1.0c/tsmcN65/../models/spectre/crn65gplus_2d5_lk_v1d0.scs	tt_rfmvar_25
✓	.../tsmc/N65RF/1.0c/tsmcN65/../models/spectre/crn65gplus_2d5_lk_v1d0.scs	tt_na25
✓	.../tsmc/N65RF/1.0c/tsmcN65/../models/spectre/crn65gplus_2d5_lk_v1d0.scs	tt_lvt
✓	/home/jpu/EE332/cadence/INVD1.pex.netlist	
	<Click here to add model file>	

Deliverables

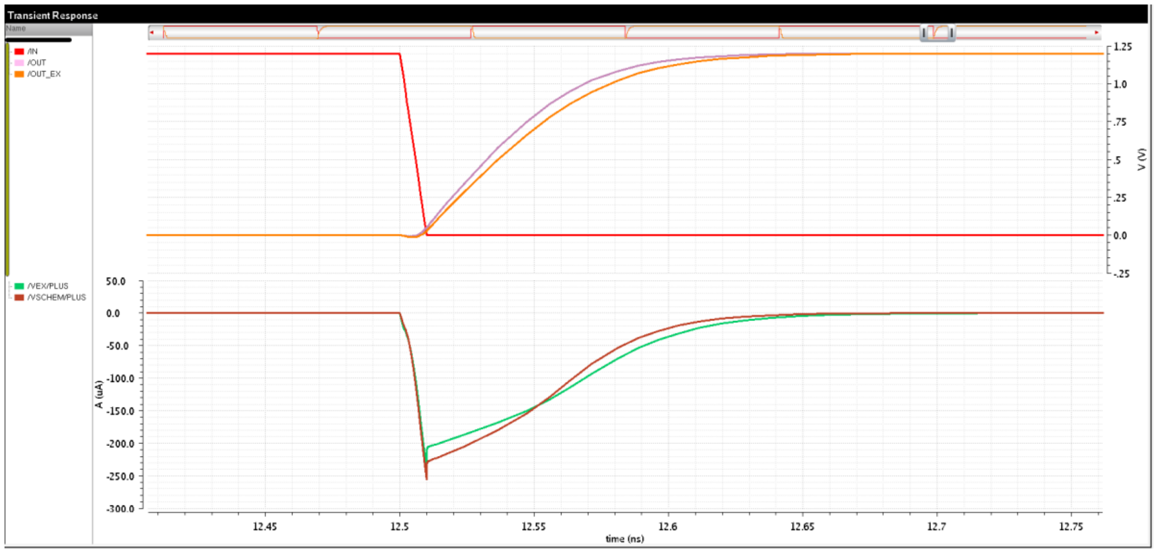
Turning in this lab assignment will consist of several images to verify that you completed the full schematic, layout, and simulation setup. The following images should be included in a document to be handed in:

1. Snapshot of inverter schematic
2. Snapshot of inverter layout
3. Snapshot of DRC clean report (no errors)
4. Snapshot of LVS clean report (smiley faces)
5. Snapshot of testbench schematic

6. Snapshots of rising and falling edges of input and output voltages, with clear differences between the schematic and post-layout waveforms. Also include the different current consumption profiles for each instance. It is recommended to move the current waveforms to a separate strip in the waveform viewer to clearly separate them form the voltages.

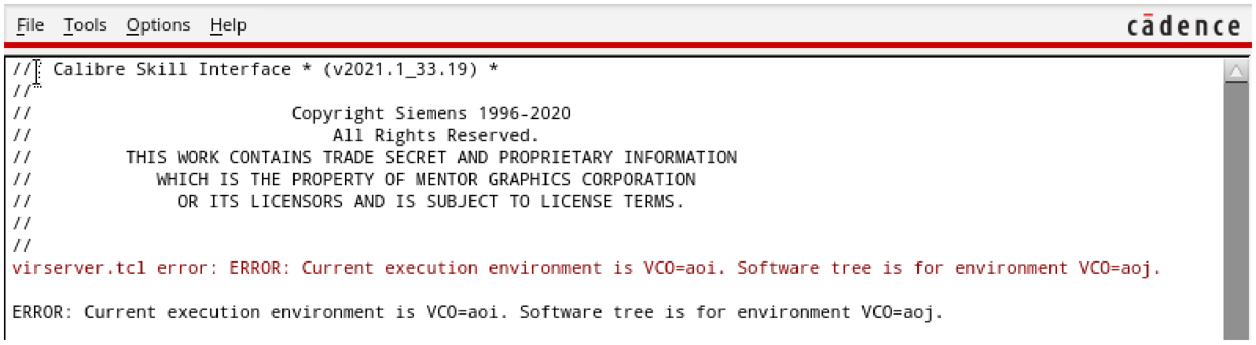
   This should consist of two plots, one for the rising edge, and one for the falling edge. Below is an example of one of these plots, where you can clearly tell which trace is which from the net and voltage source labels.

   In addition to the above items which you will turn in with your report. Write a short paragraph describing each one of the plots, what you observe, relating the results or schematic/layout to the real-world physical significance.



Appendix A

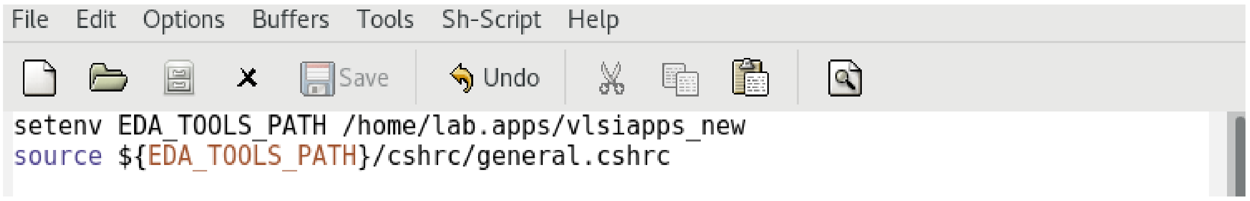
If you get the following error when opening Cadence and the "Calibre" dropdown option is missing when you open a layout cell view, the steps below may fix the issue. NOTE: This fix may only work fully if you are using one of the linux-lab-### machines. At time of writing, it does not fully work for linuxsrv01.



```
// Calibre Skill Interface * (v2021.1_33.19) *
//
//                        Copyright Siemens 1996-2020
//                            All Rights Reserved.
//          THIS WORK CONTAINS TRADE SECRET AND PROPRIETARY INFORMATION
//             WHICH IS THE PROPERTY OF MENTOR GRAPHICS CORPORATION
//               OR ITS LICENSORS AND IS SUBJECT TO LICENSE TERMS.
//
//
virserver.tcl error: ERROR: Current execution environment is VCO=aoi. Software tree is for environment VCO=aoj.

ERROR: Current execution environment is VCO=aoi. Software tree is for environment VCO=aoj.
```

1. Using any editor, open the ".cshrc" file that you usually source when you wish to run Cadence. I use Emacs below. The original file will likely look something like this:



```
setenv EDA_TOOLS_PATH /home/lab.apps/vlsiapps_new
source ${EDA_TOOLS_PATH}/cshrc/general.cshrc
```

2. Add the following lines to the ".cshrc" file:

setenv OA_UNSUPPORTED_PLAT linux_rhel50_gcc44x

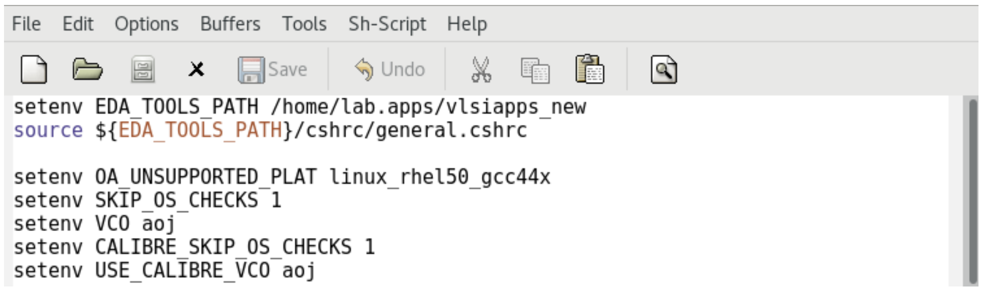
setenv SKIP_OS_CHECKS 1

setenv VCO aoj

setenv CALIBRE_SKIP_OS_CHECKS 1

setenv USE_CALIBRE_VCO aoj

The final file should appear as below. Save the file and run Cadence like normal. The Calibre error message should disappear, and Calibre should now successfully appear when you use Virtuoso Layout.



```
setenv EDA_TOOLS_PATH /home/lab.apps/vlsiapps_new
source ${EDA_TOOLS_PATH}/cshrc/general.cshrc

setenv OA_UNSUPPORTED_PLAT linux_rhel50_gcc44x
setenv SKIP_OS_CHECKS 1
setenv VCO aoj
setenv CALIBRE_SKIP_OS_CHECKS 1
setenv USE_CALIBRE_VCO aoj
```