SKY130 DAY 3: Design Library Cell Using Magic Layout and NGSPICE characterisation

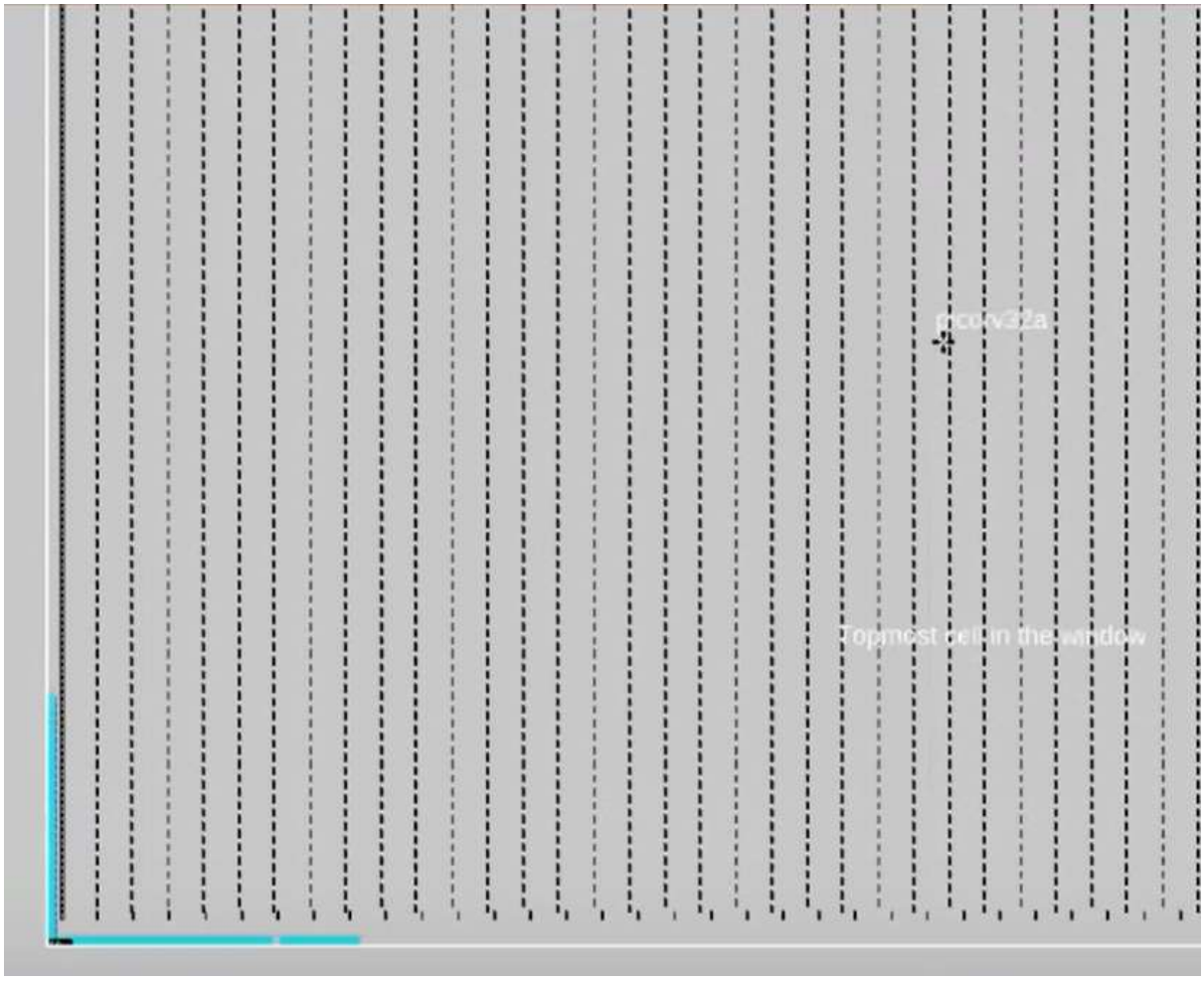
Labs for CMOS Inverter NGSPICE Simulations

IO Placer Revision

OpenLANE configurations can be changed on the fly inside the shell. To change the IO mode from the default equidistant (1) to mode 2, use this command after running floorplan: set ::env(FP_IO_MODE) 2 . This will adjust the IO pins to a non-equidistant arrangement.

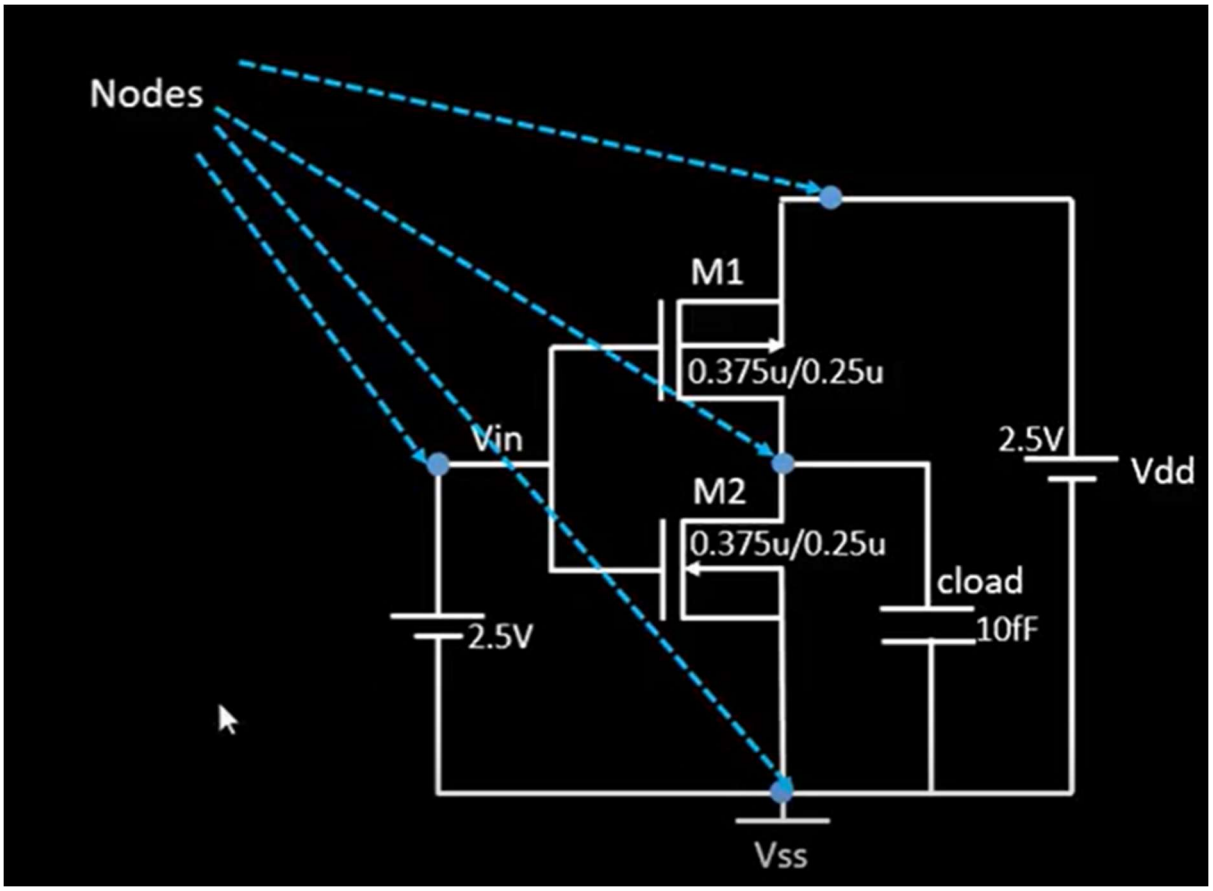
After running the command, re-run the floorplan. You'll notice the pins are now placed using the Hungarian algorithm, stacked one over the other.

Note: Changing the configuration on the fly will not affect runs/config.tcl. The changes will only apply to the current session.



SPICE Deck Creation For CMOS Inverter

The SPICE deck contains netlist connectivity, input specifications, output taps, and component values. For example, the PMOS typically has a channel length of 0.25µm and width of 0.375µm. Ideally, the PMOS width is 2-3 times that of the NMOS to compensate for slower hole carriers and match rise/fall times. The next steps involve identifying and naming the nodes.



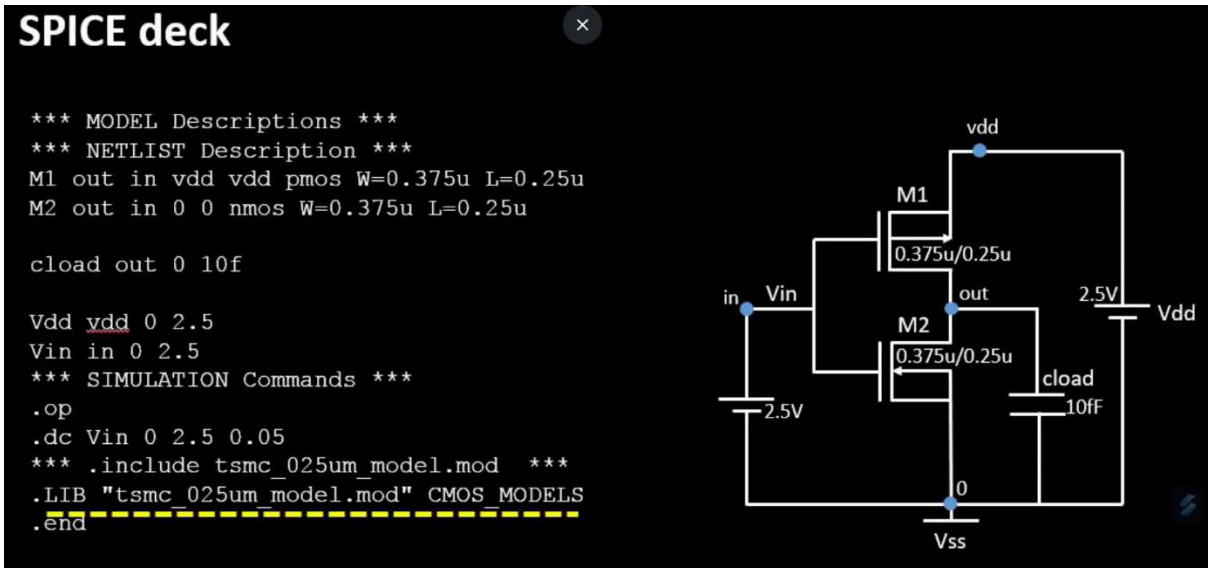
The SPICE deck netlist syntax for PMOS and NMOS is as follows:

[component name] [drain] [gate] [source] [substrate] [transistor type] W=[width] L=[length]

Example Syntax:

M1 OUT IN VDD VDD PMOS W=.375U L=.25U

For a CMOS inverter SPICE simulation, the simulation begins with .op. In this, Vin will be swept from 0 to 2.5V with 0.05V steps. The model file tsmc_025um_model.mod contains all the technology parameters for 0.25µm NMOS and PMOS.

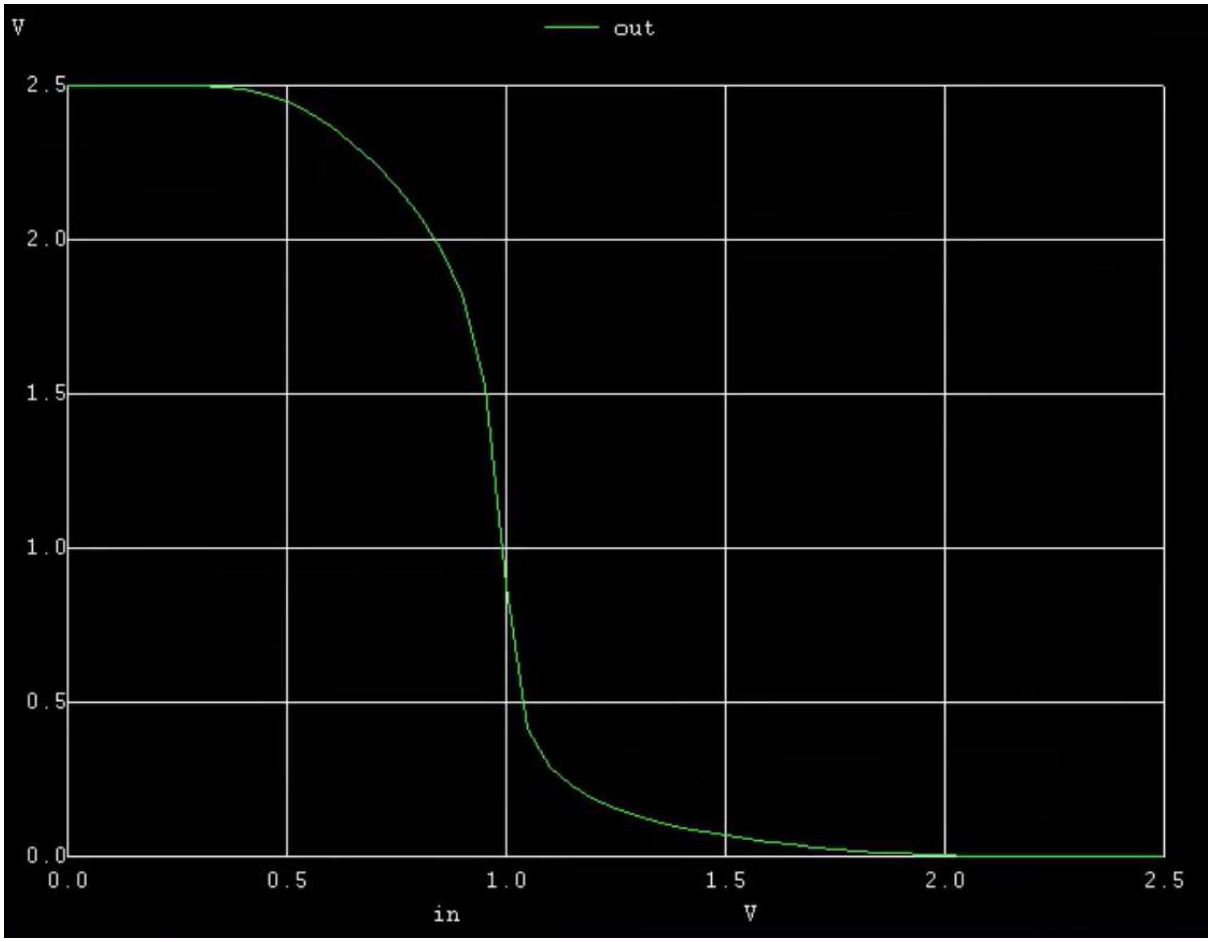


For SPICE simulation, there are various steps :-

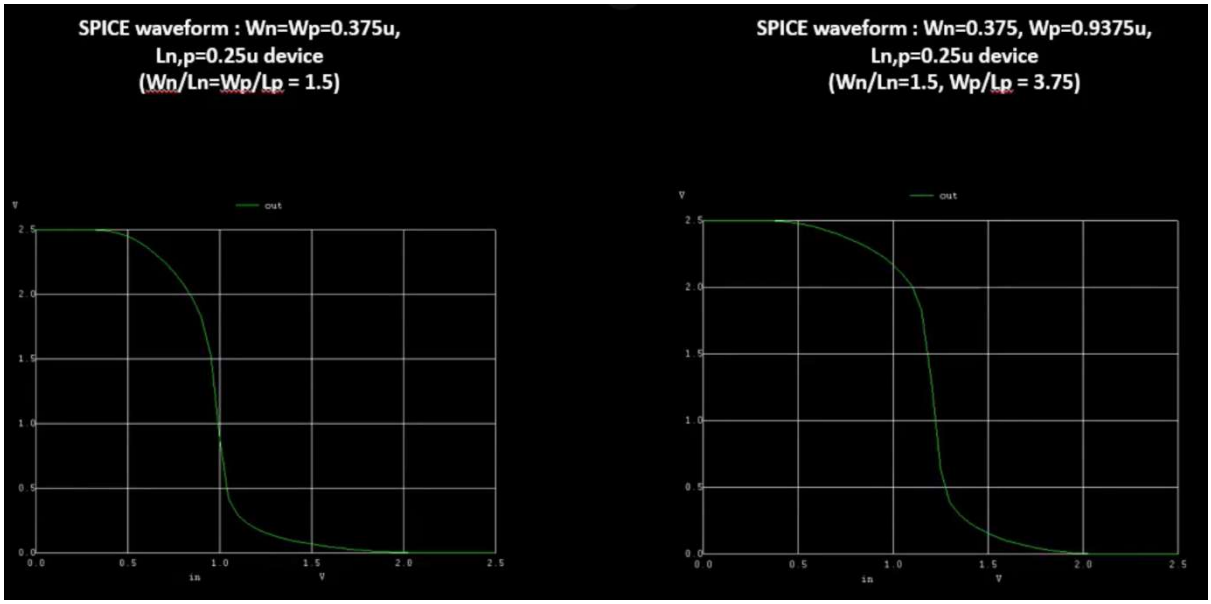
1. **Open the NGSPICE simulator**
2. **Source the Circuit File through *source* command**
3. **To execute the simulation, use the command:**

```
arduino
CopyEdit
run
```
Then, use `setplot` to view the available plots from the simulations in the SPICE deck. This will provide you with options to select which simulation to run.

4. **After running the simulation, type: `display` . This will give you a choice of nodes to plot. Once you type `plot out vs in`, the graph will be displayed.**



Switching Threshold Vm

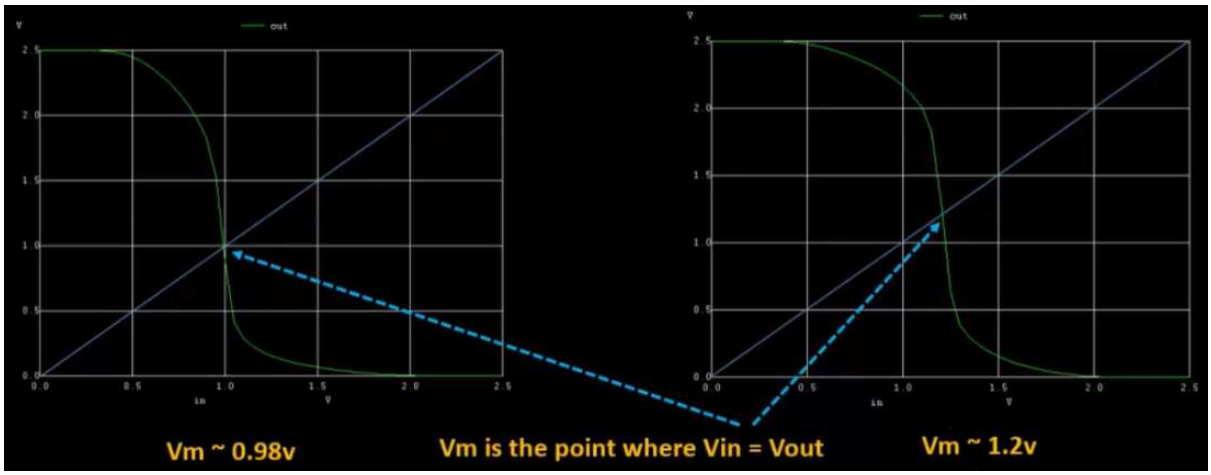


POINTS TO BE NOTED:

- **The shapes of the graphs are almost identical, which leads to the conclusion that CMOS is a robust device.**

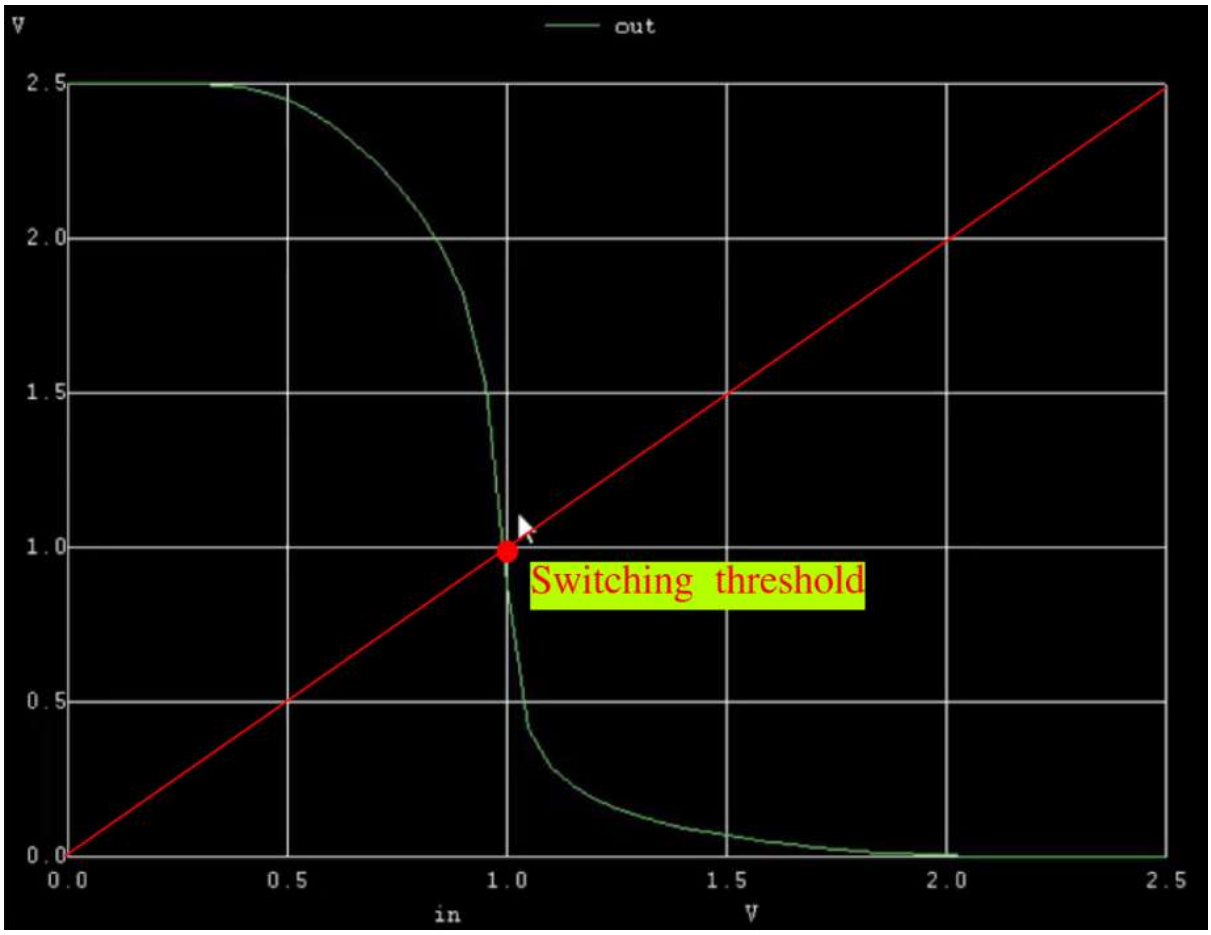
- **The robustness of CMOS is determined by two key parameters: the switching threshold and the propagation delay.**

The Switching Threshold is the point where the input voltage equals the output voltage, and both PMOS and NMOS are in saturation. When both are on, there is a high chance of leakage, causing current to flow directly from VDD to GND, which can lead to a short circuit.

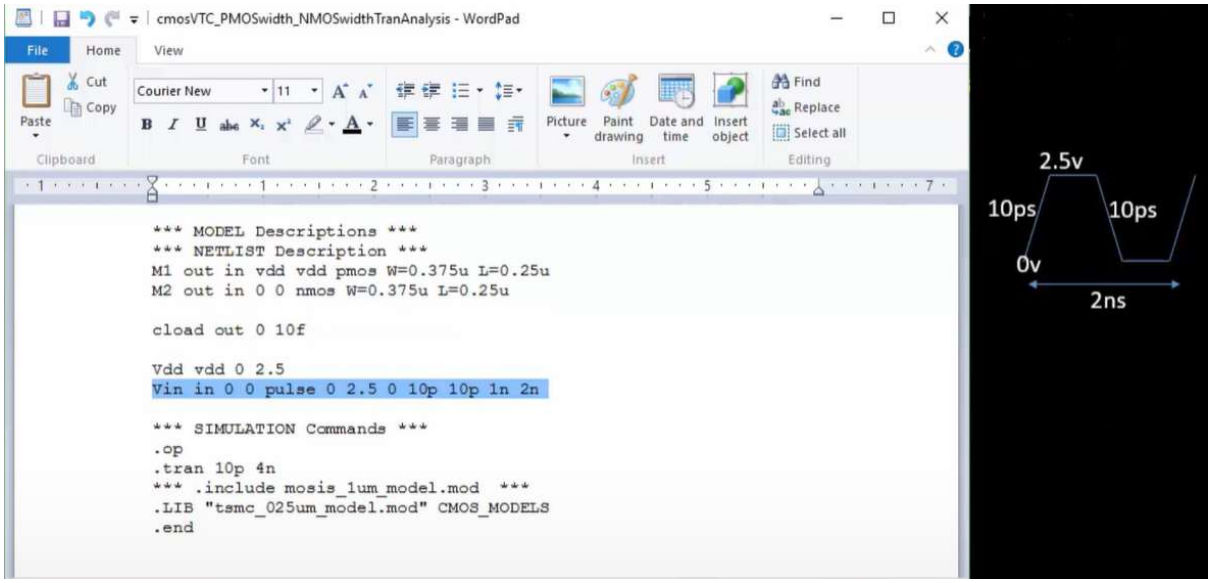


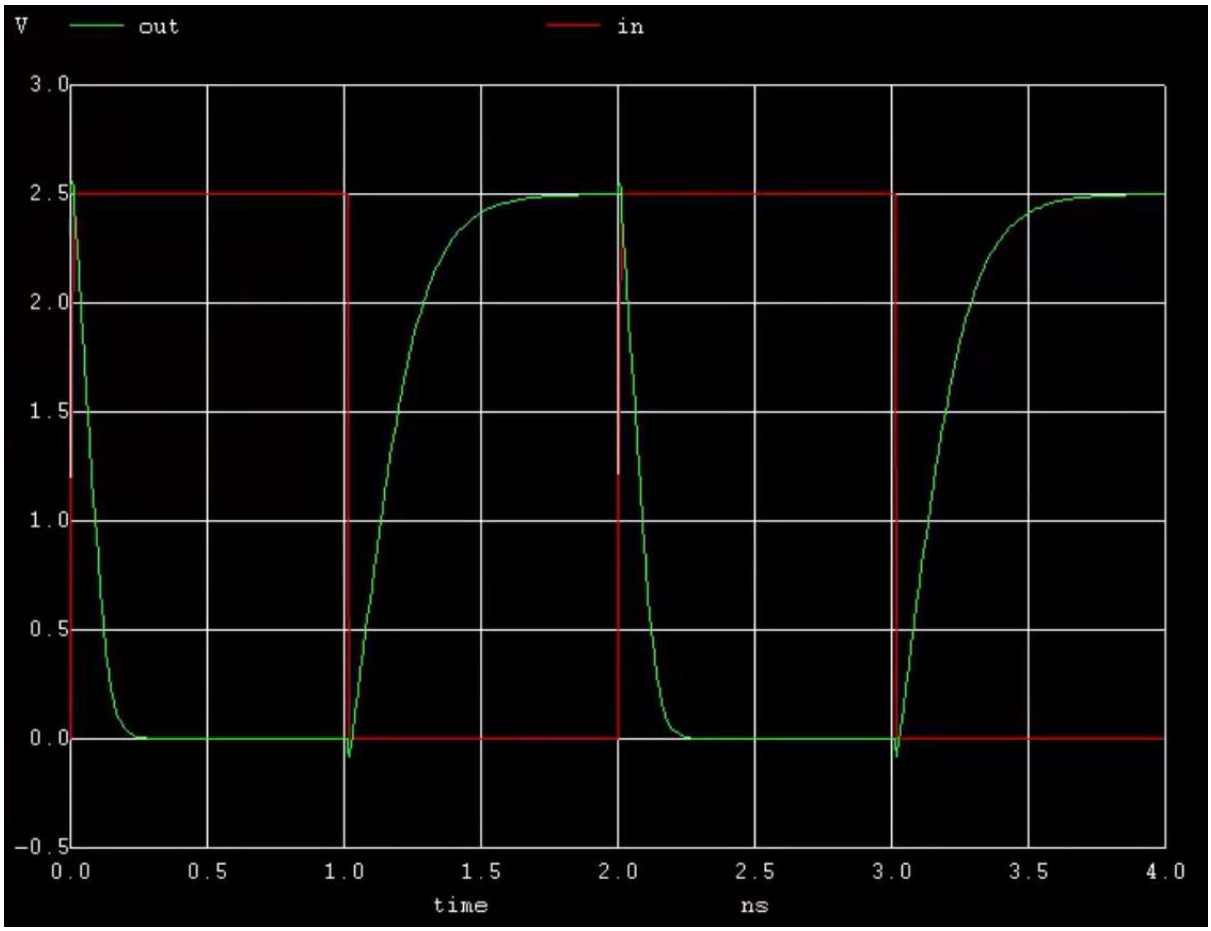
Static and Dynamic Simulation of CMOS Inverter

To find the switching threshold (Vm), we use DC Transfer Analysis. The simulation sweeps the input voltage from 0V to 2.5V in 0.05V steps.



To find the propagation delay, we use transient analysis by applying a pulse to the CMOS and measuring the delay between the input and output transitions.





Lab Steps to GitClone VSDSTD Cell Design

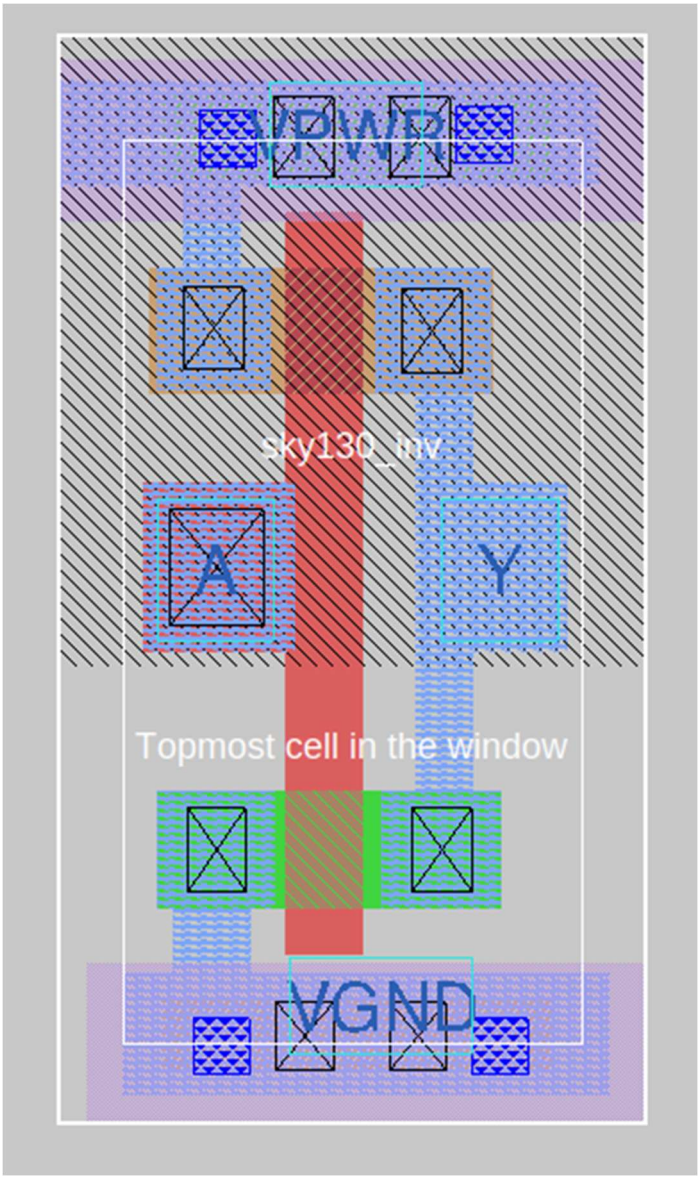
The GitHub repository containing the inverter files can be accessed at this link. Clone the custom inverter standard cell design from the GitHub repository shared above

1. **Clone the repository with the custom inverter design through the command *git clone***

2. **Subsequently, copy the tech file to the vsdstdcelldesign directory using the following command:**

cp home/vsduser/Desktop/work/tools/openlane_working_dir/pdks/sky130A/libs.tech/magic/sky130A.tech /home/vsduser/Desktop/work/tools/openlane_working_dir/openlane/vsdstdcelldesign/

3. To open the custom inverter layout in MAGIC, use the following command: ***magic -T sky130A.tech sky130_inv.mag &cp***



Inception of Layout and CMOS Fabrication Process

The 16 MASK CMOS Fabrication process is as follows:

Create Active Regions

1. **The first step is to select a substrate, which serves as the foundation for your entire design. The most commonly used substrate is a P-doped Silicon Substrate, which is typically less doped than its wells.**
2. **The next step is creating an active region for transistors. It's important to ensure isolation between the pockets, which can be achieved through.**
    - **Growing 40nm of Silicon Dioxide**
    - **Depositing 80nm of Silicon Nitride.**
    - **Depositing a layer of photoresist**
    3. **Deposit mask-1 layer on top of the photoresist. This layer covers the photoresist areas that must not be etched away, protecting the two transistor active regions.**
    - **Applying UV light to remove the layers on the unmasked regions**
    - **Removing mask-1 and photoresist layers**

- **Placing the chip in the furnace to grow the oxide in other areas**

4. **Remove the Si3N4 layer using hot phosphoric acid, leaving only the p-substrate and SiO2 layer behind.**

Formation of N and P well

3. **P well and N well formation**

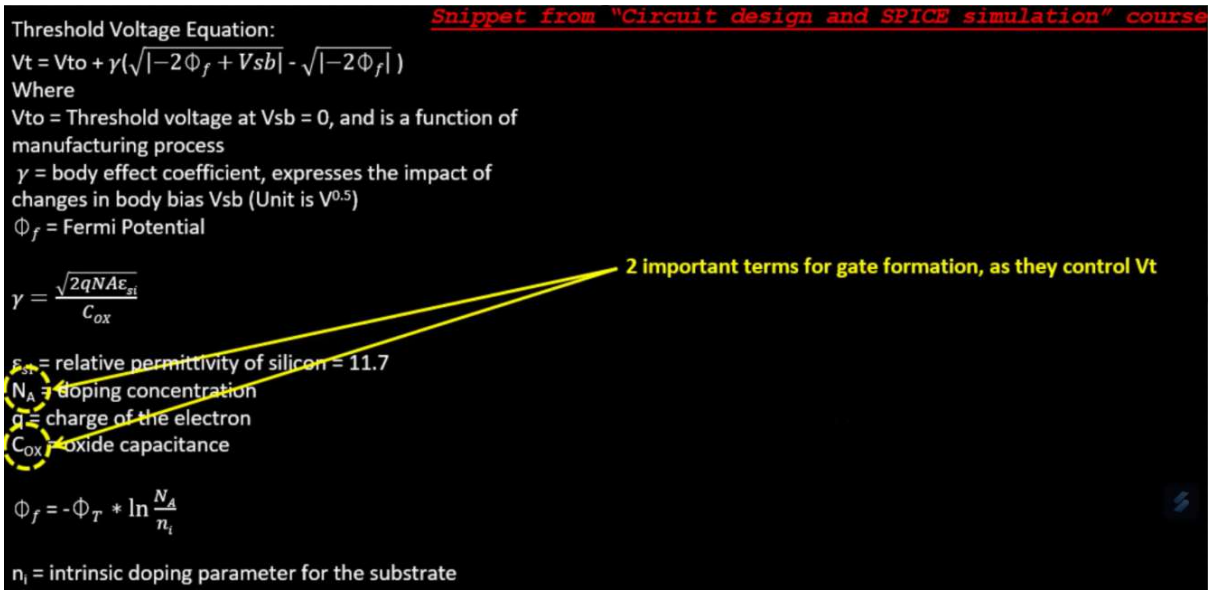
   - **Deposit the photoresist layer and define the areas to protect by applying mask-2 and mask-3. Mask-2 protects the N-Well (PMOS side) while the P-Well (NMOS side) is being fabricated, and Mask-3 protects the P-Well while the N-Well is being formed.**

   - **Application of UV Light to remove the exposed photoresist**

   - **Place the chip in a furnace to diffuse boron and phosphorus, forming the wells. This process is called the Twintub process. Boron [B] is used to form P-Well and Phosphorus [P] is used to form N-well**

Formation of Gate Terminal

Gate Terminal is where Threshhold Voltage is controlled - as seen below:



4. **Formation of Gate**

   - **Deposit photo resist layer to define the areas to be protected, and then subsequently deposit mask-4. Then, UV light is applied, and the exposed area of photoresist is removed**

   - **Then, implantation of low energy boron at the surface of p-well using mask-4 to control the threshold occurs**

   - **Similarly, implantation of phosphorous/arsenic for n-well using mask-5 occurs**

   - **Fixing the oxide which is damaged by implantation steps by removing extra SiO2 using the hydroflouric acid and re-grow high quality SiO2 on p-substrate to contol the oxide thickness occurs next**

   - **Addition of polysilicon film subsequently occurs**

   - **Then, mask-6 is added and etching using photolithography occurs**

   - **Then, mask 6 is etched off to form the gate terminal**

Lightly Doped Drain [LDD] Formation

5. **LDD Formation: Lateral Diffused Drain (LDD) regions are created to prevent hot electrons from breaking Si-Si bonds or creating excessive voltage that exceeds the 3.2eV barrier, causing issues with doped regions. Additionally, LDDs help prevent the short-channel effect, which can lead to gate malfunction due to the drain field penetrating the channel.**
    - **Mask 7 and 8 are created for NMOS (lightly doped N-type) and PMOS (lightly doped P-type) respectively.**
    - **Heavily doped impurities (N+ for NMOS and P+ for PMOS) are added for the source and drain, while lightly doped impurities are introduced to maintain spacing between the source and drain. This helps prevent hot electron effects and the short-channel effect.**
    - **To protect the lightly doped regions, we also add SiO2 and create spacers using *plasma anisotropic* etching**

Source and Drain Formation

6. **Source and Drain Formation**
    - **Thin screen oxide is added to avoid channeling during. Channeling is when implantations dig too deep into substrate which is very problematic**
    - **We create Mask-9 is for N+ implantation and Mask-10 for P+ implantation**
    - **The side wall spacers maintain the N-/P- while implanting the N+/P+**
    - **High temperature annealing is done as well**

Local Interconnect Formation

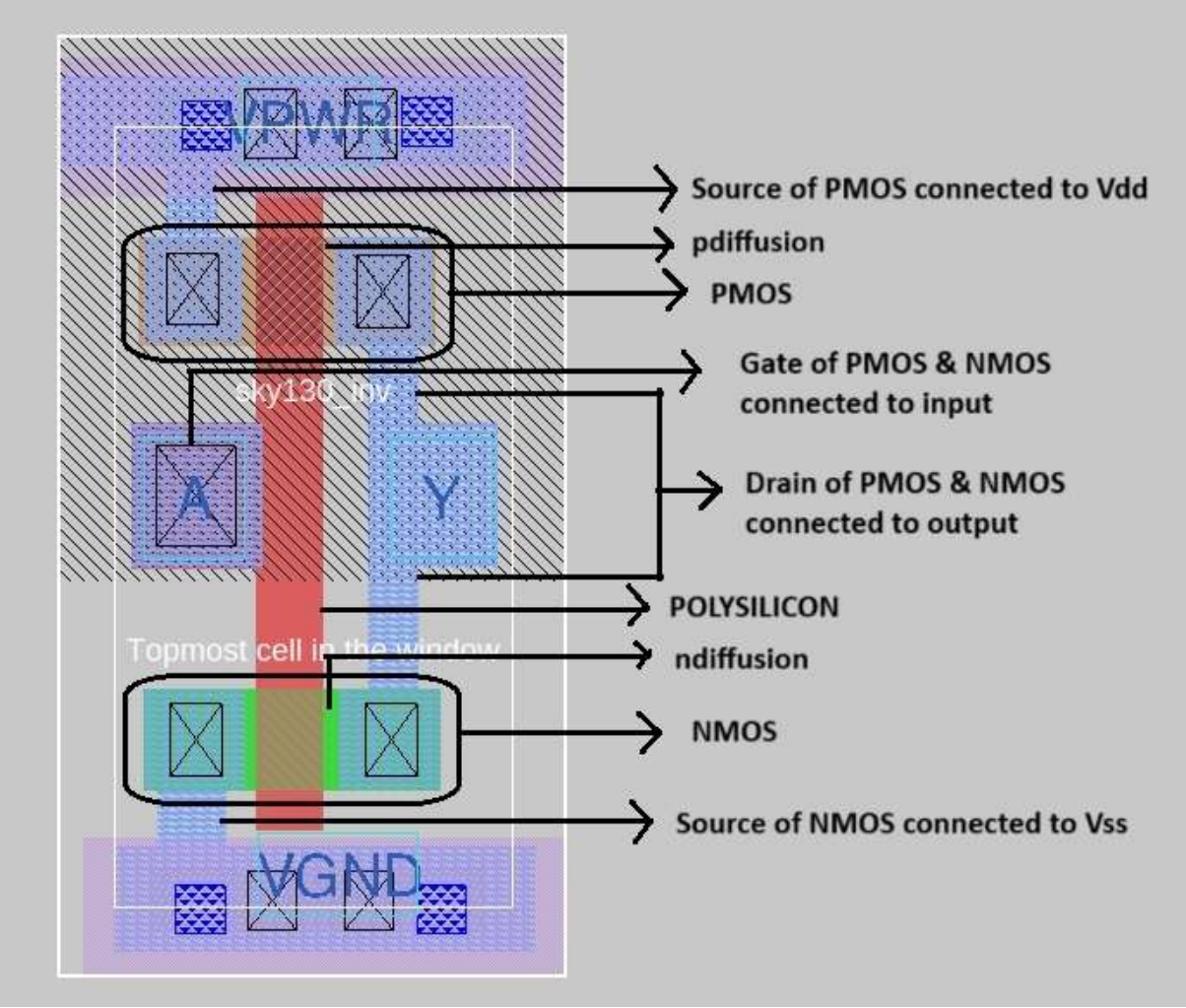
7. **Local connects and interconnects play a key role in controlling electrical characteristics and are the primary components the end user can access. They are formed using metal layers and vias for connections, followed by planarization and contact formation to ensure proper routing and functionality of the chip.**
    - **The thin screen oxide is removed for opening up the source, drain and gate for contact building. We use Titanium as it has less resistance.**
    - **Titanium Diselenide [Ti2Si2] is used for local interconnects**
    - **Mask 11 is formed and Titanium Nitride [Ti N] is etched off by RCA cleaning to create the first level contact**

Higher Level Metal Formation

8. **Higher Level Metal Formation - These steps are very similar to the previous steps and are quite easy to understand.**
    - **The previous steps in the MASK process have created an uneven surface layer. A layer of Silicon Dioxide [SiO2] doped with phosphorous or boron - [boron reduces the temperature] [known as phosphosilicate glass and borophosphosilicate glass] is deposited on the wafer surface.**
    - **Then, the surface is polished using the CMP [Chemical Mechanical Polishing] technique to planarize the surface.**
    - **Contact holes are created through photolithography.**
    - **Various masks are used for the various processes after this:-**
        - **Mask 12 is created for the first contact holes**

- **Mask 13 is used for the first Aluminum contact layer, which the contact holes are connected to.**
- **Mask 14 creates the second contact holes**
- **Mask 15 is similarly, for the second Aluminum contact layer**
- **Finally, we use Mask 16 for making contact to topmost layer**

Lab Introduction to SKY130 Basic Layers Layout and LEF using Inverter



In Sky130A, the layers are arranged as follows: the first layer is the LDD or local-i, followed by m1, m2, and other metal layers. Power and ground lines are placed in m1. When polysilicon crosses ndiffusion, an NMOS is created, and when it crosses pdiffusion, a PMOS is formed. The output of the layout process is a LEF file, which contains an abstract layout of the standard cell, including the location of the standard cell pins. This file is used by the router in the APR process to ensure proper routing.

Lab Steps to Create STD Cell Layout and Extract SPICE Netlist

For the SPICE extraction of the custom inverter layout, we can enter the following commands in the TCKON :-

1. ***extract all***
2. ***ext2spice cthresh 0 rthresh 0*** **--> This extracts the parasitic information**
3. ***ext2spice***

```
tkcon 2.3 Main
File  Console  Edit  Interp  Prefs  History  Help
% pwd
/home/vsduser/Desktop/work/tools/openlane_working_dir/openlane/vsdstdcelldesign
% extract all
Extracting sky130_inv into sky130_inv.ext:
% ext2spice cthresh 0 rthresh 0
% ext2spice
exttospice finished.
%
```

```
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane/vsdstdcelldesign$ tree -L 1
.
├── extras
├── Images
├── libs
├── LICENSE
├── README.md
├── sky130A.tech
├── sky130_inv.ext
├── sky130_inv.mag
└── sky130_inv.spice
```

```
sky130_inv.spice (~/Desktop/work/tools/openlane_working_dir/openlane/vsdstdcelldesign) - GVIM
File  Edit  Tools  Syntax  Buffers  Window  Help
* SPICE3 file created from sky130_inv.ext - technology: sky130A

.option scale=10m

.subckt sky130_inv A Y VPWR VGND
X0 Y A VGND VGND sky130_fd_pr__nfet_01v8 ad=1.44n pd=0.152m as=1.37n ps=0.148m w=35 l=23
X1 Y A VPWR VPWR sky130_fd_pr__pfet_01v8 ad=1.44n pd=0.152m as=1.52n ps=0.156m w=37 l=23
C0 VPWR Y 0.117f
C1 A Y 0.0754f
C2 A VPWR 0.0774f
C3 Y VGND 0.279f
C4 A VGND 0.45f
C5 VPWR VGND 0.781f
.ends
~
```

SKY130 Tech File Labs

Lab Steps To Create Final SPICE deck using SKY130 tech

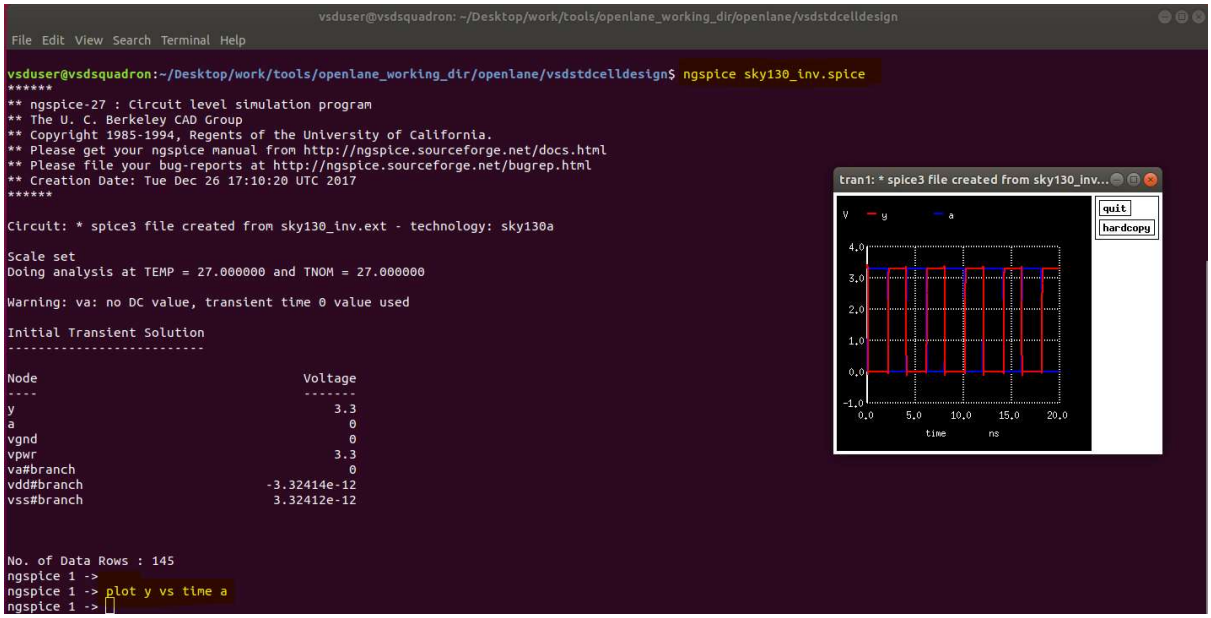
Modify the default Sky130 SPICE deck to plot a transient response, creating the final deck as shown.

```
* SPICE3 file created from sky130_inv.ext - technology: sky130A

.option scale=0.01u
.include ./libs/pshort.lib
.include ./libs/nshort.lib

//.subckt sky130_inv A Y VPWR VGND
M1000 Y A VPWR VPWR pshort_model.0 w=37 l=23
+  ad=1443 pd=152 as=1517 ps=156
M1001 Y A VGND VGND nshort_model.0 w=35 l=23
+  ad=1435 pd=152 as=1365 ps=148
VDD VPWR 0 3.3V
VSS VGND 0 0V
Va A VGND PULSE(0V 3.3V 0 0.1ns 0.1ns 2ns 4ns)
C0 A Y 0.05fF
C1 Y VPWR 0.11fF
C2 A VPWR 0.07fF
C3 Y 0 0.24fF
C4 VPWR 0 0.59fF
//.ends
.tran 1n 20n

.control
run
.endc
.end

~
~
```

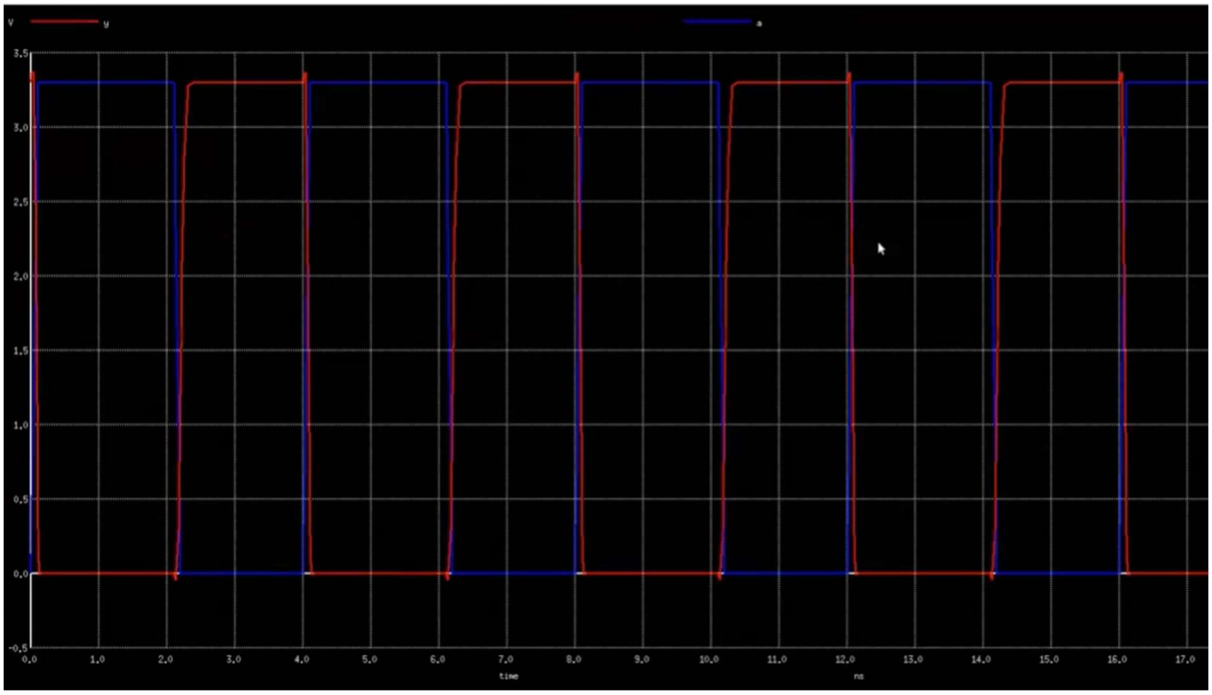
To load the SPICE file for simulation in NGSPICE, type the following command : `ngspice sky130A_inv.spice`

Lab Steps to Characterise Inverter using SKY130 Model Files

After typing this command, you will get a result as follows:-



After measuring the rise transition, to generate a graph, type the following command in NGSPICE:-



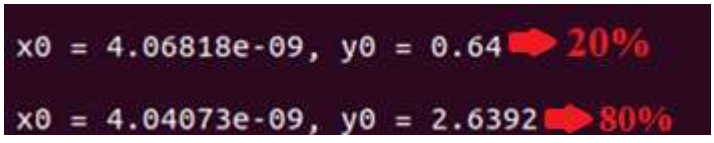
Using this, we can characterise the cell through four parameters -

1. **The rise transition is the time taken for the output waveform to transition from 20% (0.66V) to 80% (2.64V) of the maximum value (Vdd = 3.3V). To measure this, click on the waveform at those voltage points, and the x and y values will appear in the terminal, like this:**

Then, rise transition = 2.2457ns - 2.1819ns = 0.638

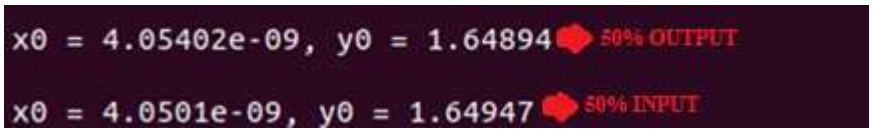
2. **The `fall transition` is the time taken for an output waveform to transition from 80% to 20% of its maximum value.**

   Similarly, 4.06818ns - 4.04073ns = 0.02745ns

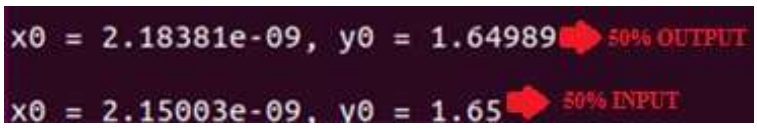


3. **The `fall cell delay` is the time difference between 50% of the input and 50% of the output, representing the time for the output to fall to 50% and the input to rise to 50%.**

   Calculating fall delay => 4.05402ns - 4.0501ns = 0.00392ns



4. **The rise cell delay is the time difference between the 50% points of the input and output signals. It represents the time taken for the output to rise to 50% of its final value, and the time taken for the input to fall to 50%.To calculate the rise delay=> 2.18381ns - 2.15003ns = 0.03378ns**

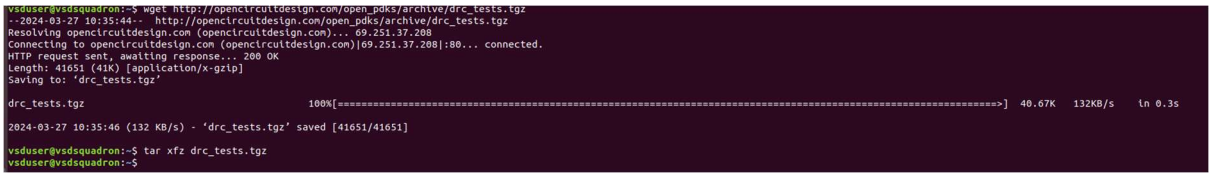


Through this, we have characterised a cell at 27 degrees Celsius successfully :- [cell was plotted with analysis occurring at 27 degrees Celsius]

Lab Introduction to Magic Tool Options and DRC Rules

A documentation shared by the instructor on how MAGIC performs DRC [Design Rule Check] and it's syntax for the rules is at [MAGIC VLSI](#) .

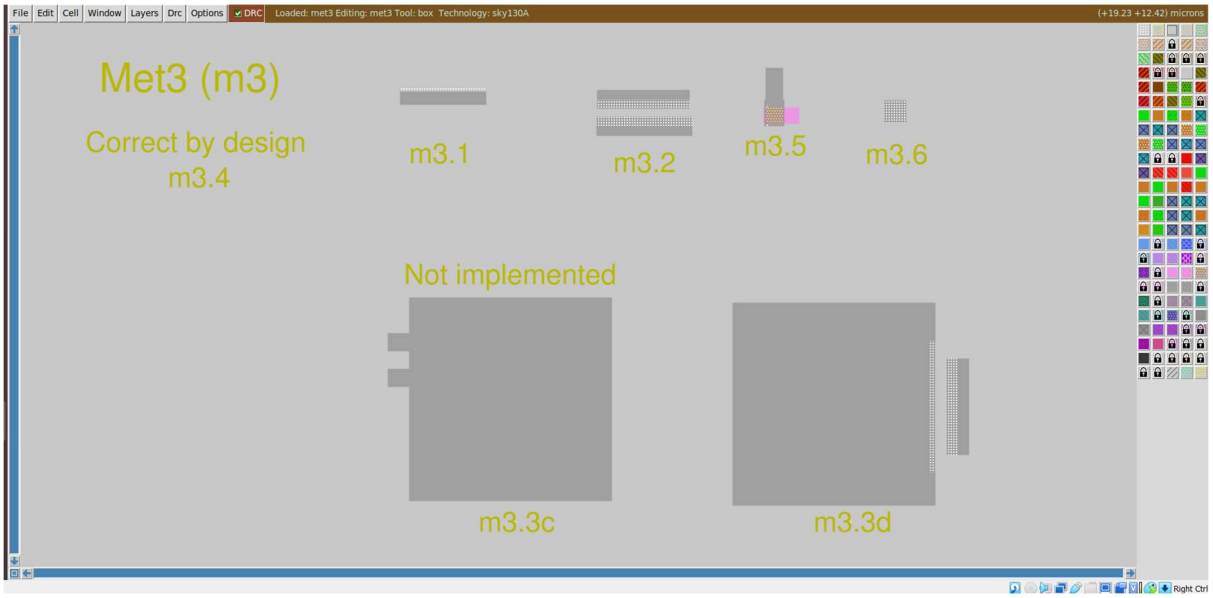
Then, to extract these files, we can type *tar xfz drc_tests.tgz*

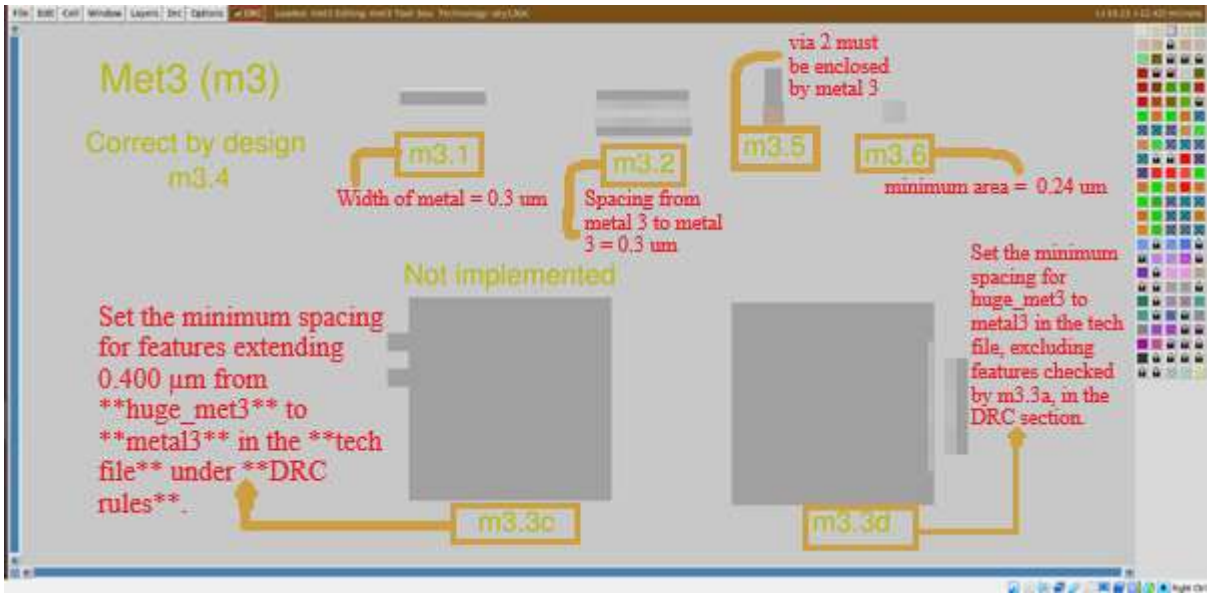


First, navigate to the directory with *`cd drc_tests`*, then list the contents using *`ls -al`*. To open the Magic tool, type *`magic -d XR`*.

Lab Introduction to Magic and Steps to Load SKY130 Tech Rules

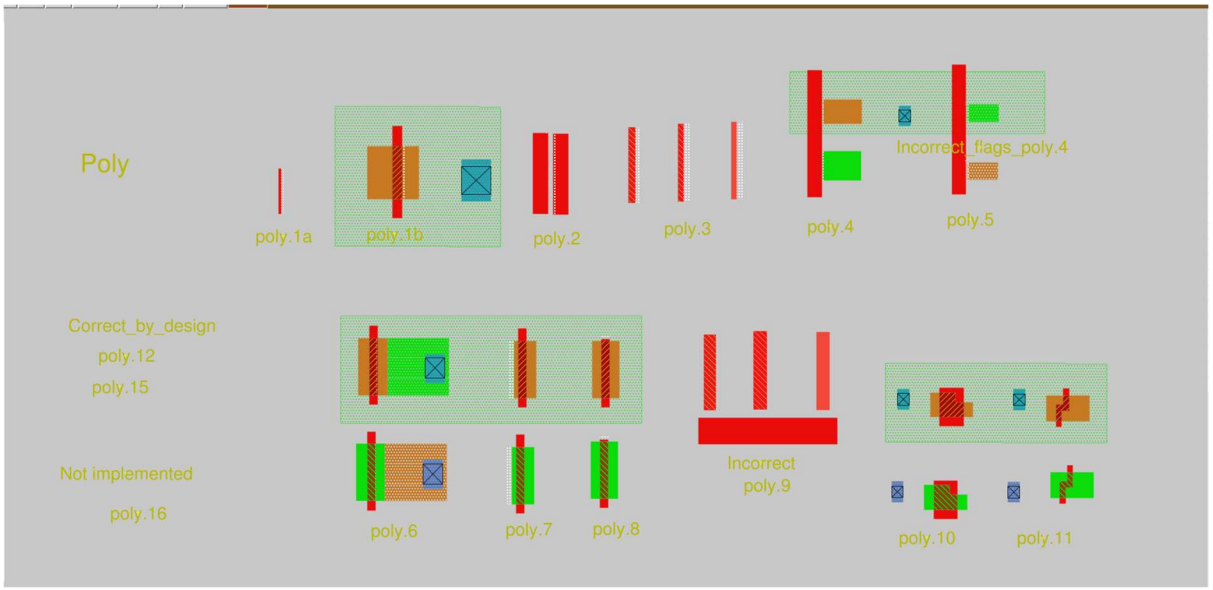
Then, in the empty magic prompt, click "FILE", which is at the top and select *open* and then *`met3.mag`* under that, to obtain this screen view where we can see a number of independent layouts contain certain DRC errors



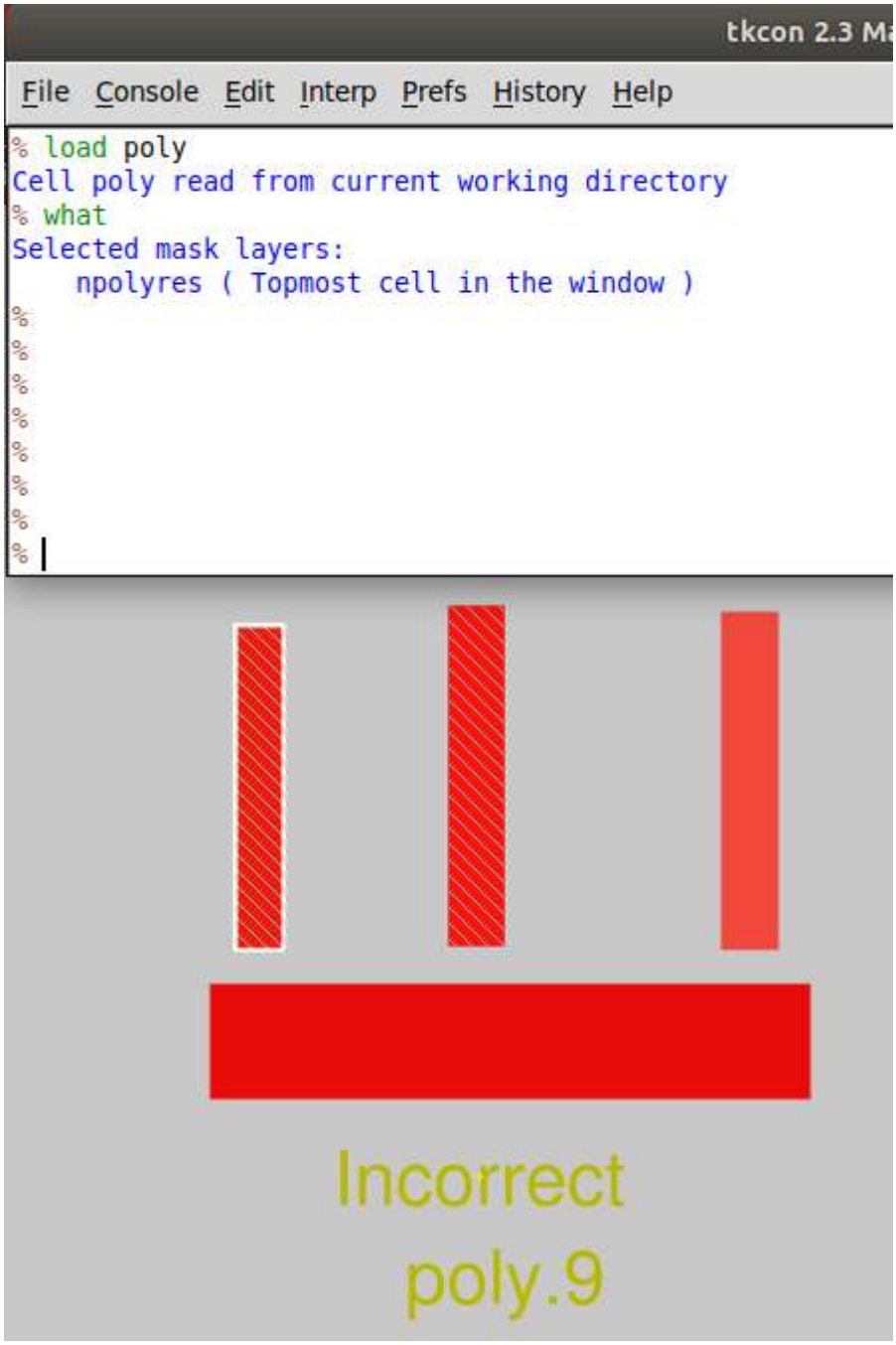


Lab Exercise to Fix poly.9 error in SKY130 Tech-File

Type *load poly* in the tkcon window :-



Now, to focus on the poly 9 rule as explained at this link which states that poly resistor spacing to poly or spacing (no overlap) to diff/tap should be at least 0.48um! Now, we can look at the below screenshot to find that this is not true, here and hence must be fixed by changing the tech file to include this DRC :-



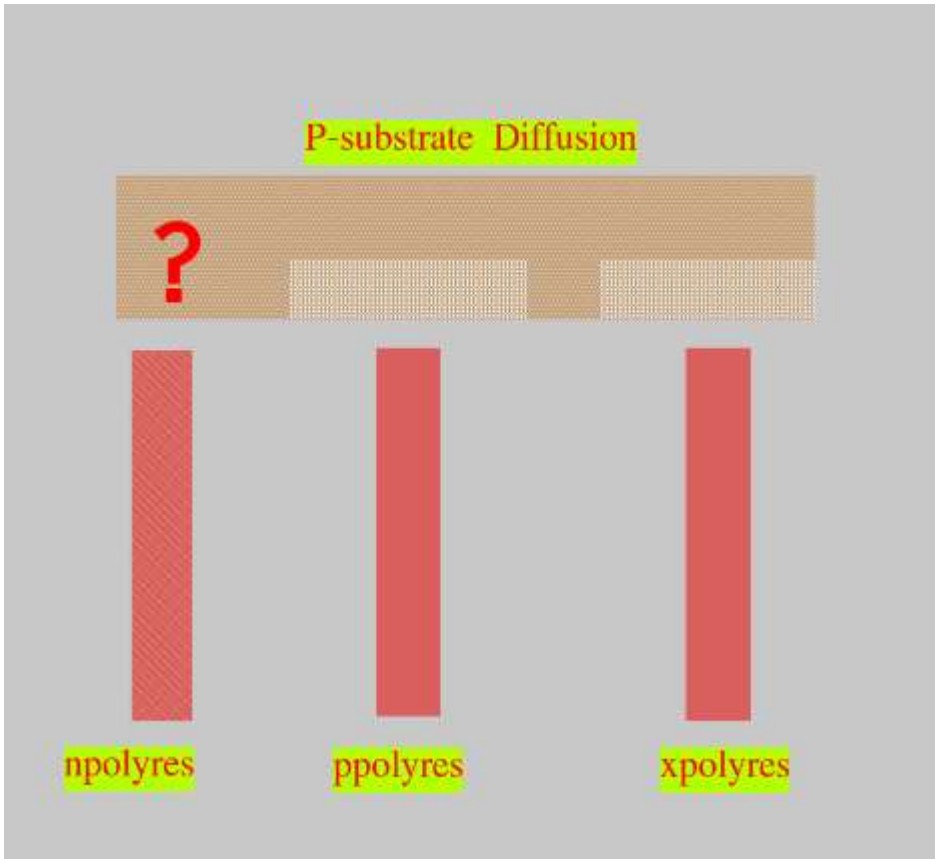
Open sky130A.tech from drc_tests. The poly.9 rules only cover n-poly and p-poly resistors with diffusion. Add new rules for poly resistors with poly non-resistors. The first rule is for p-poly, the second for n-poly. allpolynonres (in the alias section) helps apply these changes.

```
#------------------------------
# POLY
#------------------------------

 width allpoly 150 "poly.width < %d (poly.1a)"
 spacing allpoly allpoly 210 touching_ok "poly.spacing < %d (poly.2)"
 spacing allpolynonfet alldifflvnonfet 75 corner_ok allfets \
        "poly.spacing to Diffusion < %d (poly.4a)"
 spacing npres *nsd 480 touching_illegal \
        "poly.resistor spacing to N-tap < %d (poly.9)"
 spacing npres allpolynonres 480 touching_illegal \
        "poly.resistor spacing to N-tap < %d (poly.9)"
 overhang *ndiff,rndiff nfet,scnfet,npd,npass 250 "N-Diffusion overhang of nmos < %d (poly.7)"
 overhang *mvndiff,mvrndiff mvnfet,mvnnfet 250 \
        "N-Diffusion overhang of nmos < %d (poly.7)"
 overhang *pdiff,rpdiff pfet,scpfet,ppu 250 "P-Diffusion overhang of pmos < %d (poly.7)"
 overhang *mvpdiff,mvrpdiff mvpfet 250 "P-Diffusion overhang of pmos < %d (poly.7)"
 overhang *poly allfets 130 "poly.overhang of transistor < %d (poly.8)"
 rect_only allfets "No bends in transistors (poly.11)"
 rect_only xhrpoly,uhrpoly "No bends in poly resistors (poly.11)"
 extend  xpc/a xhrpoly,uhrpoly 2160 \
        "poly.contact extends poly resistor by < %d (licon.1c + li.5)"
 spacing xhrpoly,uhrpoly xhrpoly,uhrpoly 1240 touching_illegal \
        "Distance between precision resistors < %d (rpm.2 + 2 * rpm.3)"
```

Then, enter commands as shown:

Lab Exercise to Implement Poly-Resistor Spacing to Diff and Tap

There's a spacing rule violation with npolyres. To fix it, update the tech file so poly.9 includes spacing between npolyres and all types of diffusion, not just N-substrate diffusion. This ensures proper separation and prevents design rule violations.



```
spacing allpolynonres alldiffnonres 75 corner_ok affect
        "poly.spacing to Diffusion < %d (poly.4a)"
spacing npres alldiff 480 touching_illegal \
        "poly.resistor spacing to N-tap < %d (poly.9)"
spacing npres allpolynonres 480 touching illegal \
```