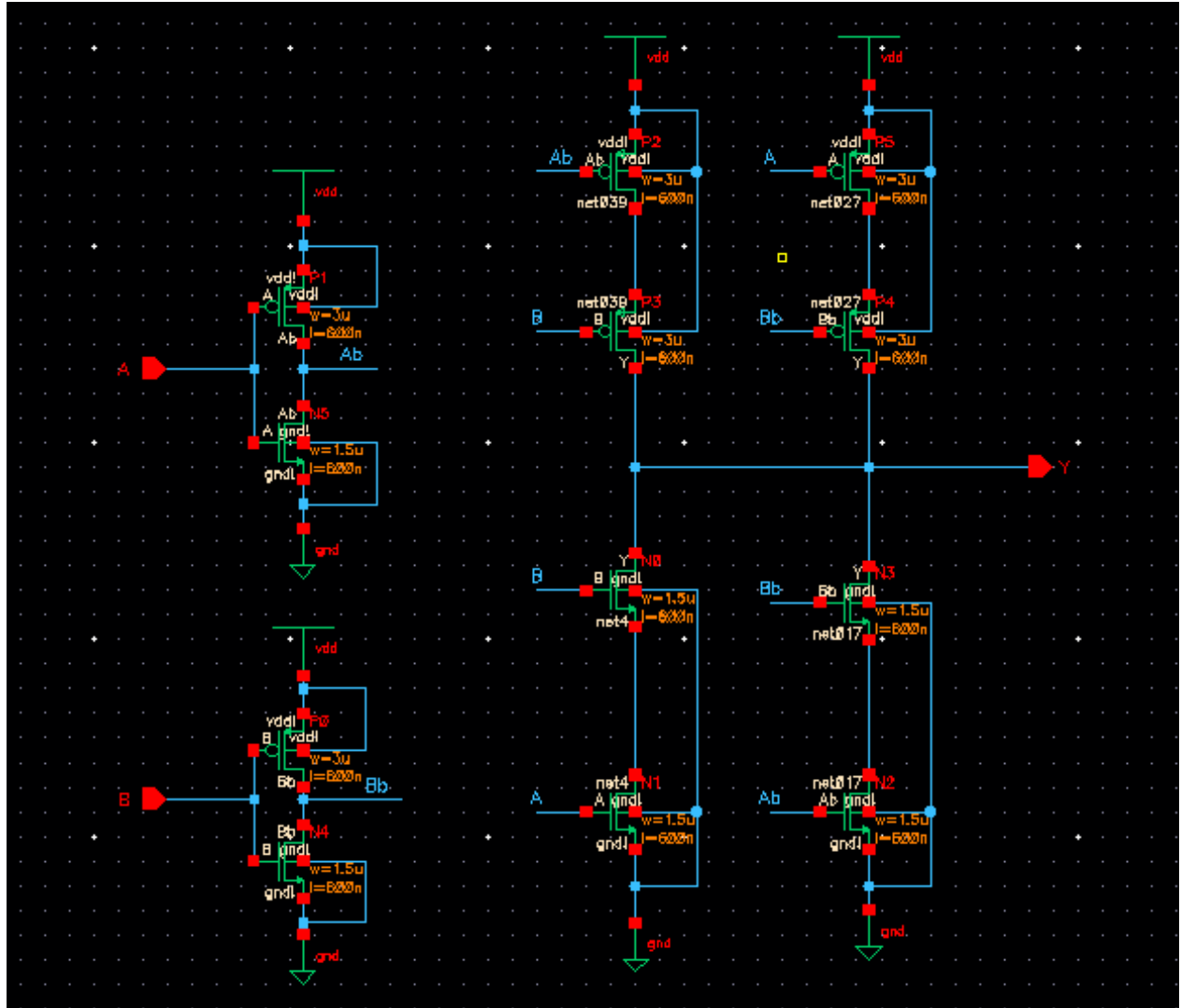


## ECE520: Lab 10 XOR cell layout

Name: Arijit Sengupta, ID: 001441748

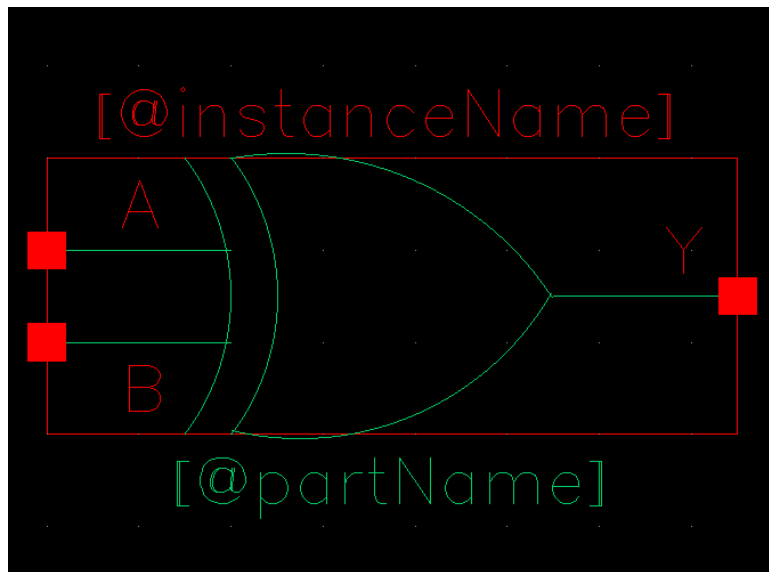
### 1) Designing an XOR2X1 cell



XOR2X1 schematic

pmos  $W/L = (3.0\mu\text{m}) / (0.6\mu\text{m}) = 5$   $W/L=5$ . This is a size 2 pmos

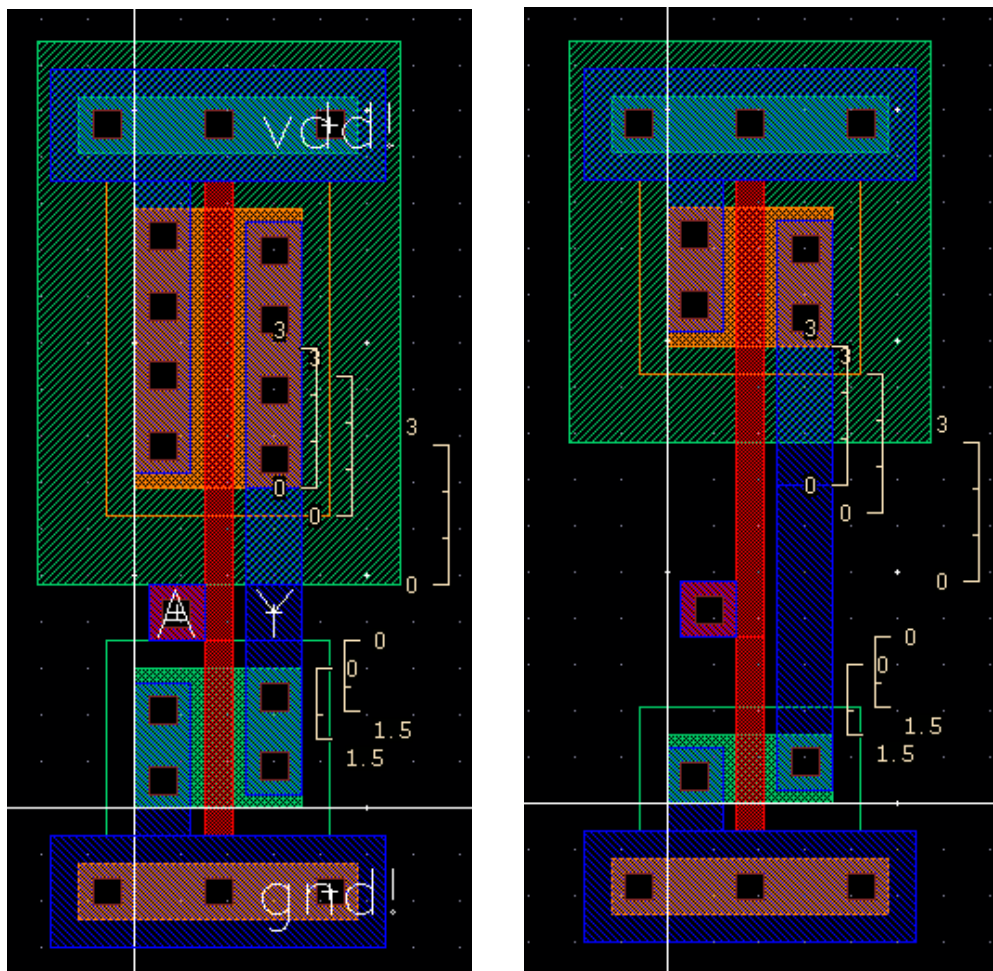
nmos  $W/L = (1.5\mu\text{m}) / (0.6\mu\text{m}) = 2.5$   $W/L=2.5$ . This is a size 1 nmos



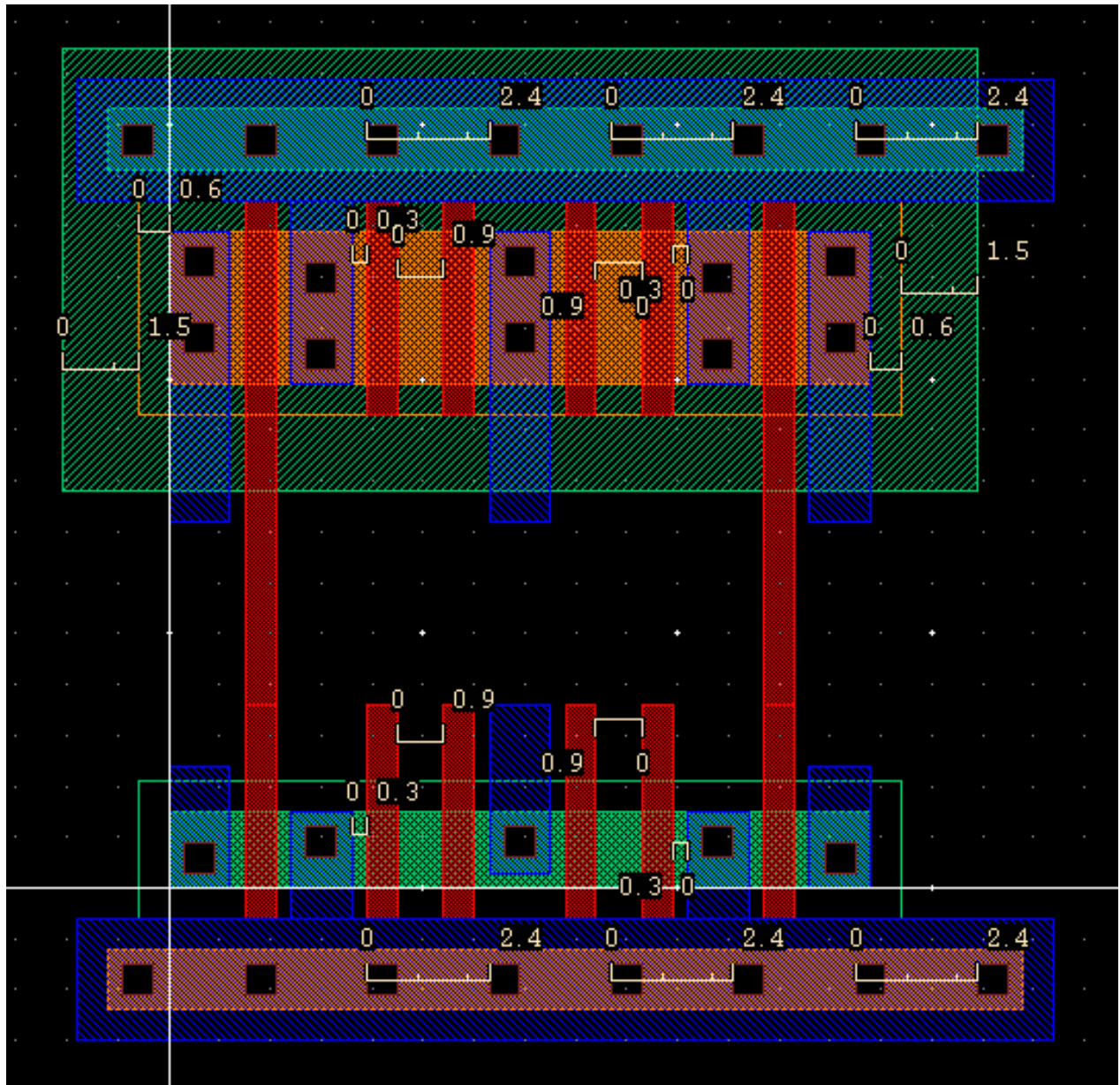
XOR2X1 Symbol

For drawing the XOR2X1 layout, we will start by copying the INVX2 layout from Lab 8.

INVX2 cell has a  $6\mu\text{m}$  pmos and a  $3\mu\text{m}$  nmos. We shrink this to  $3\mu\text{m}$  (pmos) and  $1.5\mu\text{m}$  (nmos), pushing the transistors towards the power rails and leaving the middle area open. This should leave an additional  $4.5\mu\text{m}$  space in the middle to work with. Also, we remove the labels and pins for A, Y, vdd! and gnd!



From this starting point, we now draw the nmos and pmos transistors. We only draw the active areas and their connections to gnd! and vdd! rails initially.

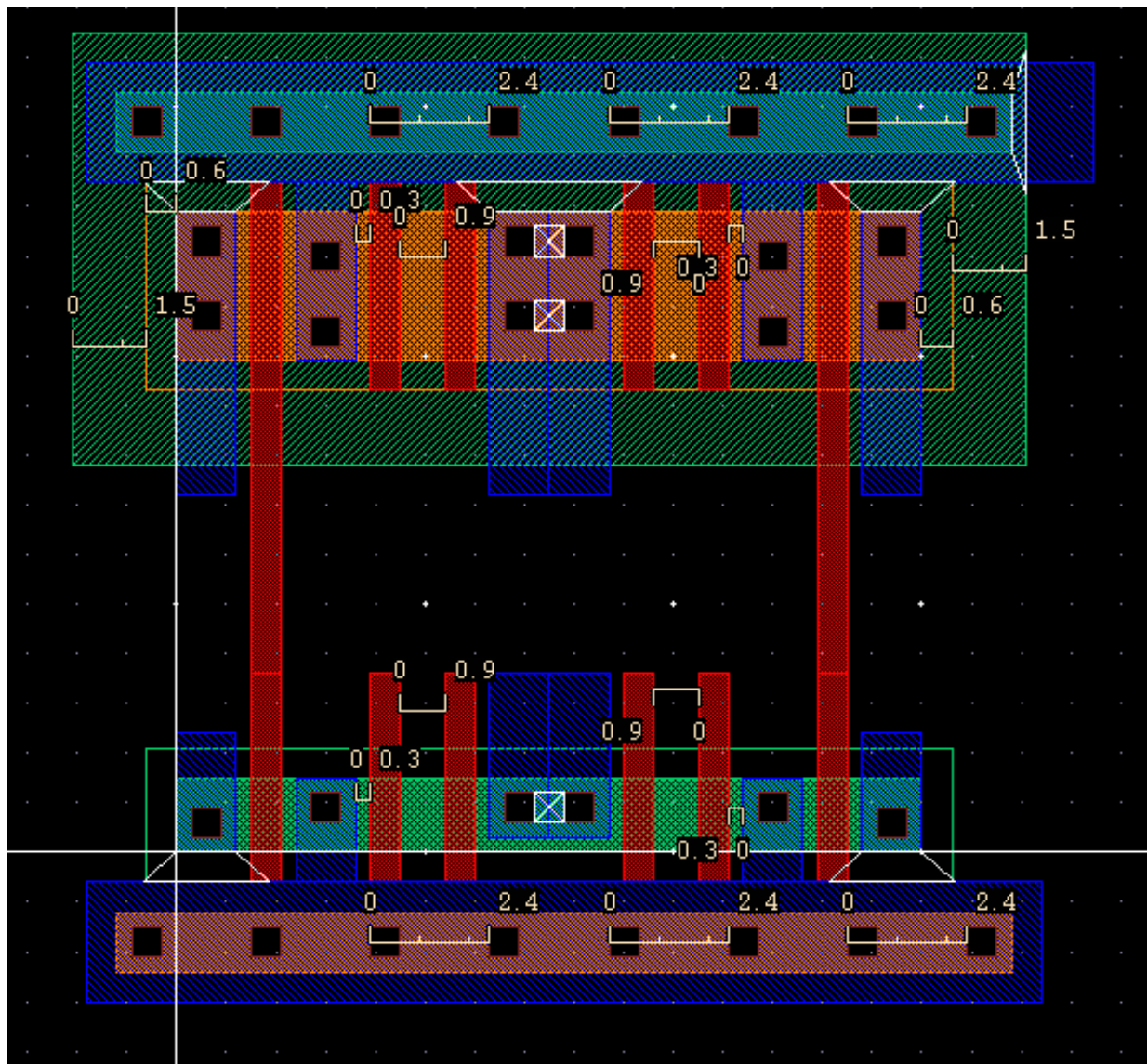


We want to push everything to the right, but we don't want to change a few things: first, we don't want to change the spacing between poly strips and we don't want to change the spacing between poly and the passive contacts.

The easiest way to achieve these goals is to replicate a set of contacts smack in the middle (metal1 and passive-metal1 contacts).

We complete the design as given below and run DRC to get the errors as given below:

```
***** Summary of rule violations for cell "XOR2X1 layout" *****
# errors  Violated Rules
    2  (SCMOS Rule 2.4) substrate/well contact active to well edge: 0.90 um
    5  (SCMOS Rule 7.2) metal1 spacing: 0.90 um
    3  (SCMOS_SUBM Rule 6.3) active contact spacing: 0.90 um
   10  Total errors found
```



**(SCMOS Rule 2.4)** substrate/well contact active to well edge: 0.90  $\mu\text{m}$ . In the above vdd! rail the pactive region is too close to the nactive region. We move it 0.60 $\mu\text{m}$  to the right to keep the minimum spacing 0.9 $\mu\text{m}$  ( $3\lambda$ ).

**(SCMOS Rule 7.2)** metal1 spacing: 0.90  $\mu\text{m}$ . On both sides, this should make you clearly understand now why our transistor metal1's looked asymmetric a little bit. For example: the Drain (D) of the top left pmos is 0.6 $\mu\text{m}$  ( $2\lambda$ ) away from the vdd! rail's metal1, which violates the SCMOS Rule 7.2; minimum-metal1-metal1-spacing: 0.9 $\mu\text{m}$  ( $3\lambda$ ). Remedy is very simple: we push the metal1 and the contact down by 0.3 $\mu\text{m}$  ( $\lambda$ ).

**(SCMOS\_SUBM Rule 6.3)** active contact spacing: 0.90  $\mu\text{m}$ . This tells us that minimum-active contact-spacing must be 0.9 $\mu\text{m}$  ( $3\lambda$ ). Remedy: push the second set of contacts to the right by another  $\lambda$ . We also stretched everything sufficiently to add another contact to the vdd! rail.

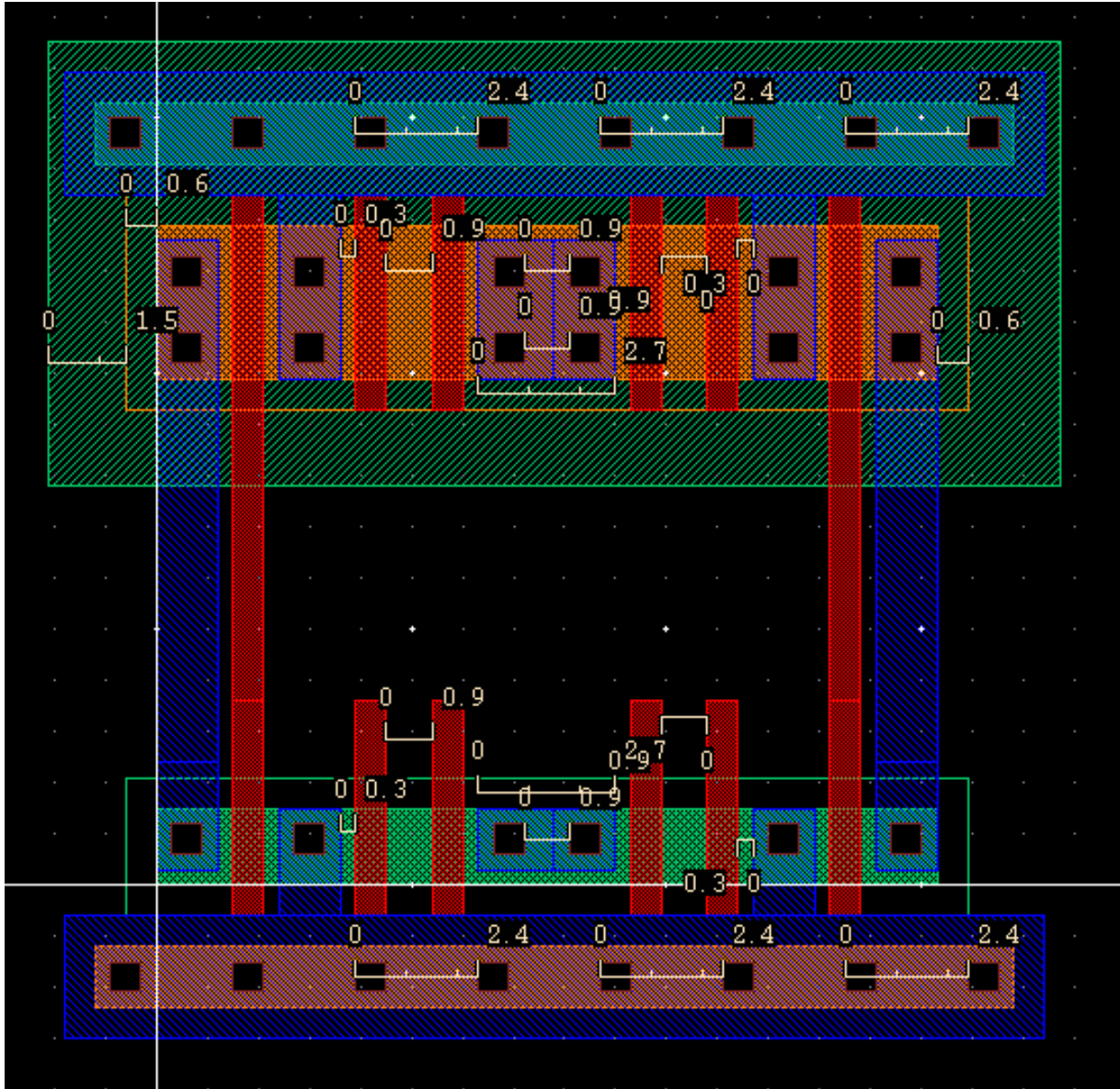
We also removed the existing metal1 strips around the contacts (for the Y output connection) and redrew a single small piece of metal1. The dimensions of this metal1 strip ended up being 2.7 $\mu\text{m}$  x 3 $\mu\text{m}$  ( $9\lambda$  x  $10\lambda$ ) for pmos and 2.7 $\mu\text{m}$  x 1.5 $\mu\text{m}$  ( $9\lambda$  x  $5\lambda$ ) for nmos. Furthermore, we joined the Ab (inverted A) and Bb (inverted B) metal1 strips with a single piece of metal1. We run DRC again, and it returns 0 errors!



```

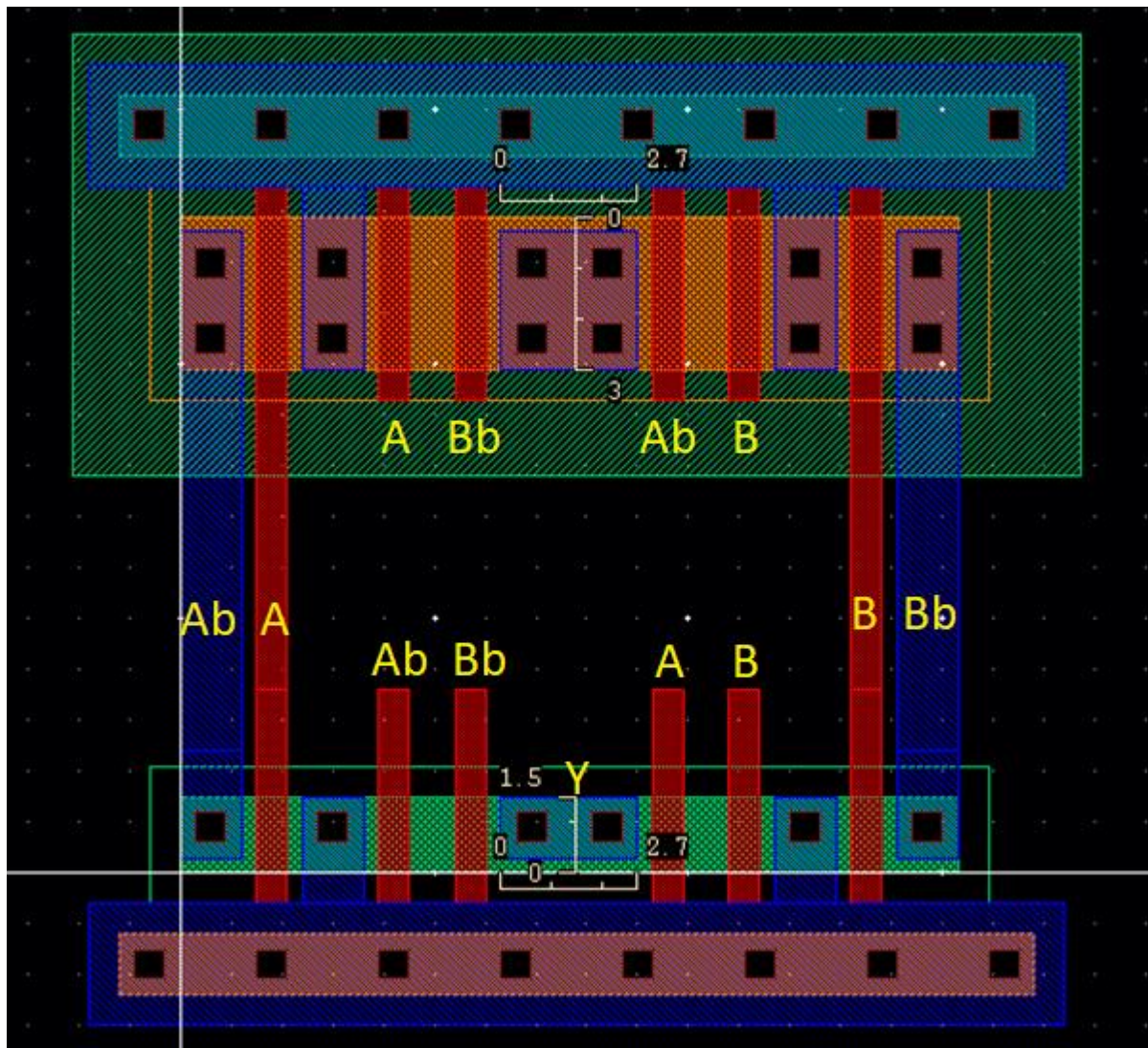
DRC started.....Fri Nov 29 18:19:38 2019
completed ....Fri Nov 29 18:19:38 2019
CPU TIME = 00:00:00 TOTAL TIME = 00:00:00
***** Summary of rule violations for cell "XOR2X1 layout" *****
Total errors found: 0

```

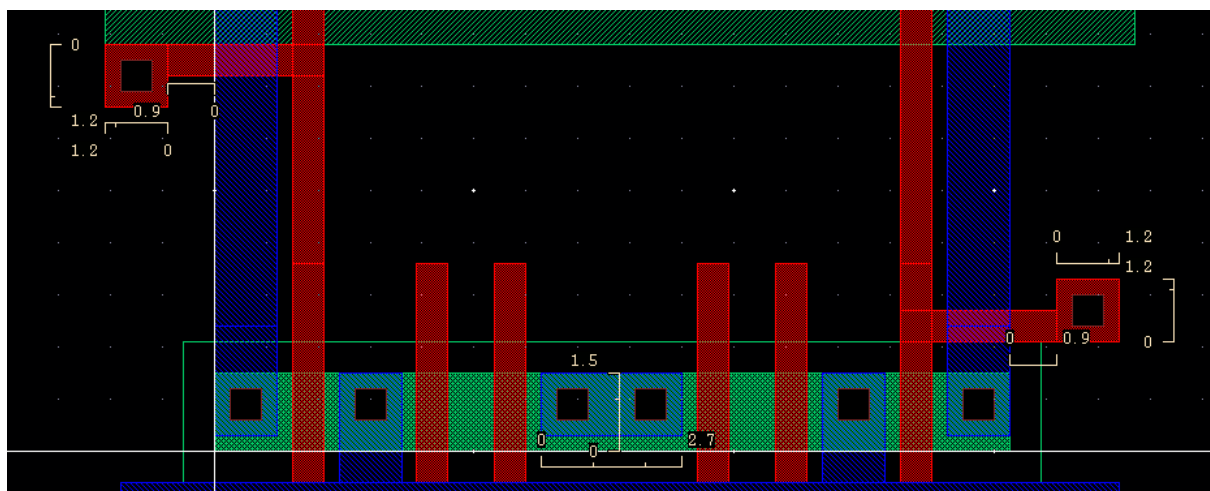


Here is where we are so far in the layout:

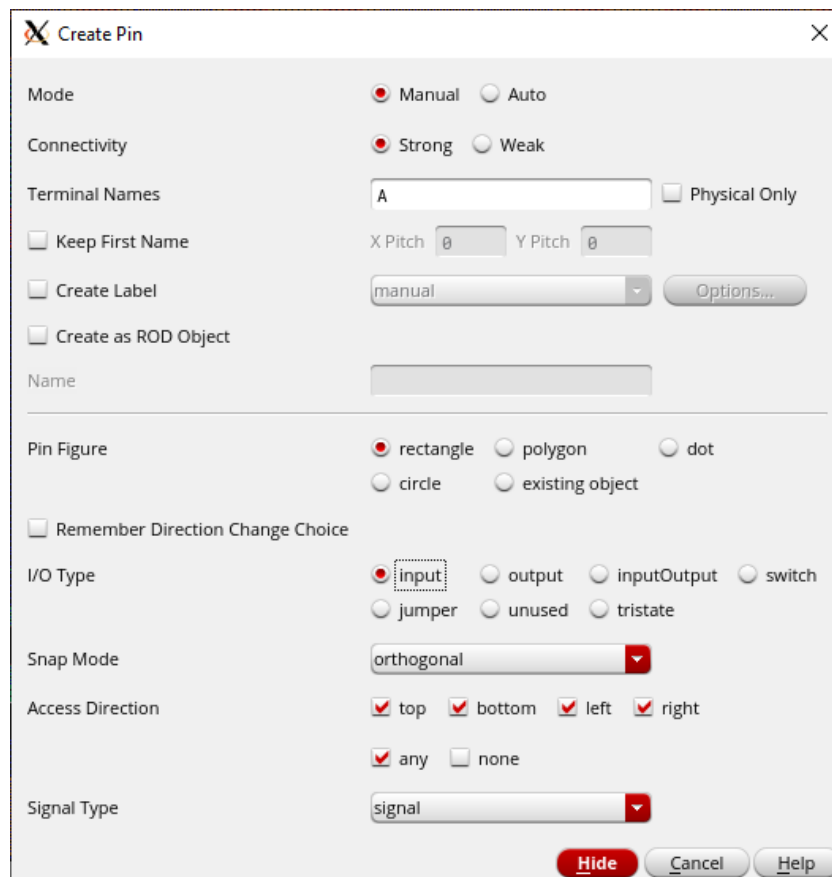
- We need 2 input (A B) pads and one output pad (Y). These are the “pins” of this XOR2X1 cell.
- Y outputs are there individually, but need to be connected to each other
- 8 poly lines need a connection for all 8 transistors: A, Bb, Ab, B for pmos ... Ab Bb A B for nmos



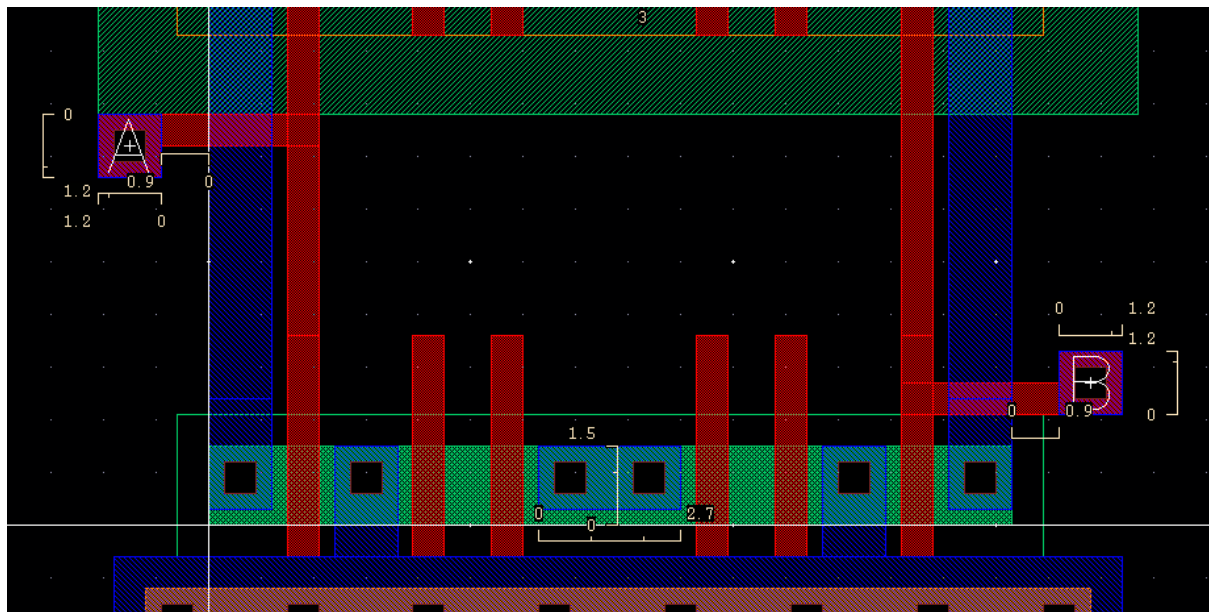
First, we start out by drawing our input pads. They will be on the sides of the cell and will be placed in the least interfering possible position. We will place them in such a way that we will not block the horizontal and vertical routing of the A, B, Y pads when this cell is instantiated. The best way to do this is to place on as close to the top possible and the other as close to the bottom as possible.





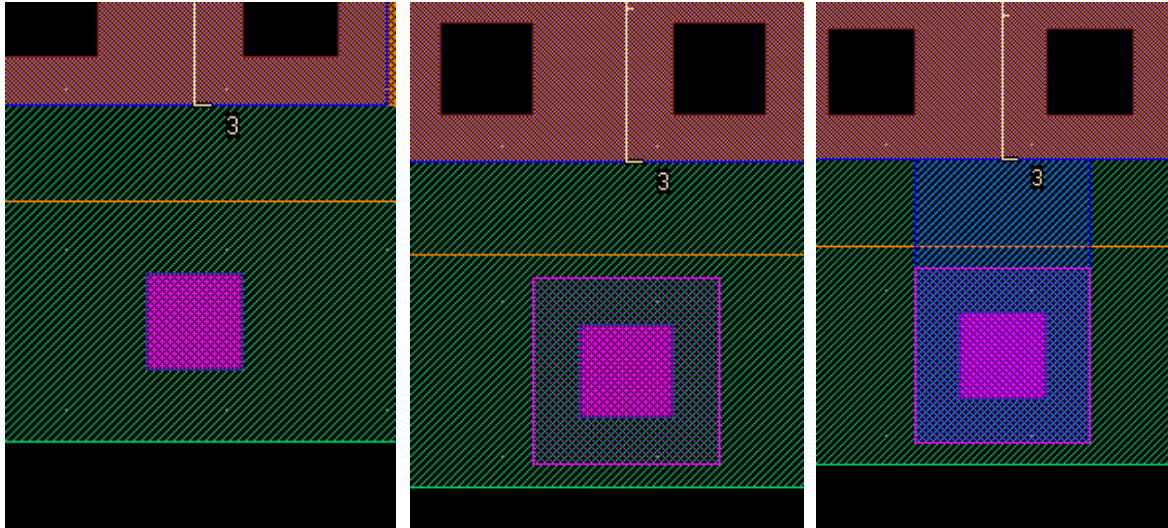


After placing the poly pads and the A B input pins, this is what the layout look like:



We will now connect the two separate Y output metal1 strips and add a Y output pin using metal2 and via to connect it with metal1 since allows a connection from metal1 to metal2 is a via.

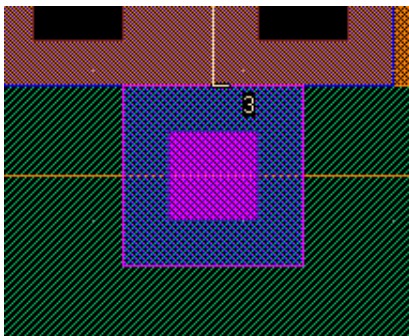
A via is nothing more than a contact, so, it follows the same rules as a contact. It must be a  $0.6\mu\text{m} \times 0.6\mu\text{m}$  ( $2\lambda \times 2\lambda$ ) square. We surround it by a  $4\lambda \times 4\lambda$  metal2, also surround it by a  $4\lambda \times 4\lambda$  metal1 and connect to the Y output using metal1.



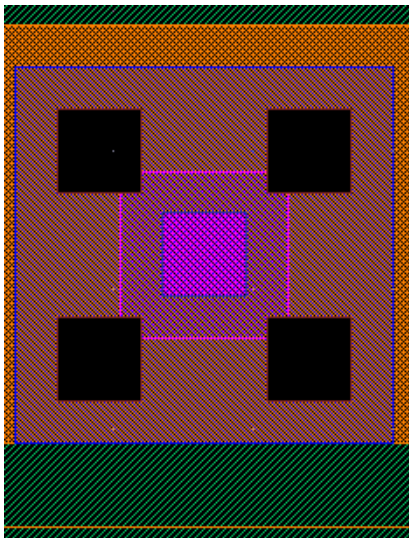
We run DRC and get no errors.

```
DRC started.....Fri Nov 29 20:02:49 2019
  completed ....Fri Nov 29 20:02:49 2019
  CPU TIME = 00:00:00  TOTAL TIME = 00:00:00
*****  Summary of rule violations for cell "XOR2X1 layout"  *****
  Total errors found: 0
```

This definitely makes us wonder if we could have pushed the via a lot farther above. So we try pushing it above and check DRC for errors.



```
DRC started.....Fri Nov 29 20:06:52 2019
  completed ....Fri Nov 29 20:06:52 2019
  CPU TIME = 00:00:00  TOTAL TIME = 00:00:00
*****  Summary of rule violations for cell "XOR2X1 layout"  *****
  Total errors found: 0
```



```
DRC started.....Fri Nov 29 20:09:51 2019
  completed ....Fri Nov 29 20:09:51 2019
  CPU TIME = 00:00:00  TOTAL TIME = 00:00:00
*****  Summary of rule violations for cell "XOR2X1 layout"  *****
  Total errors found: 0
```

We get no errors!

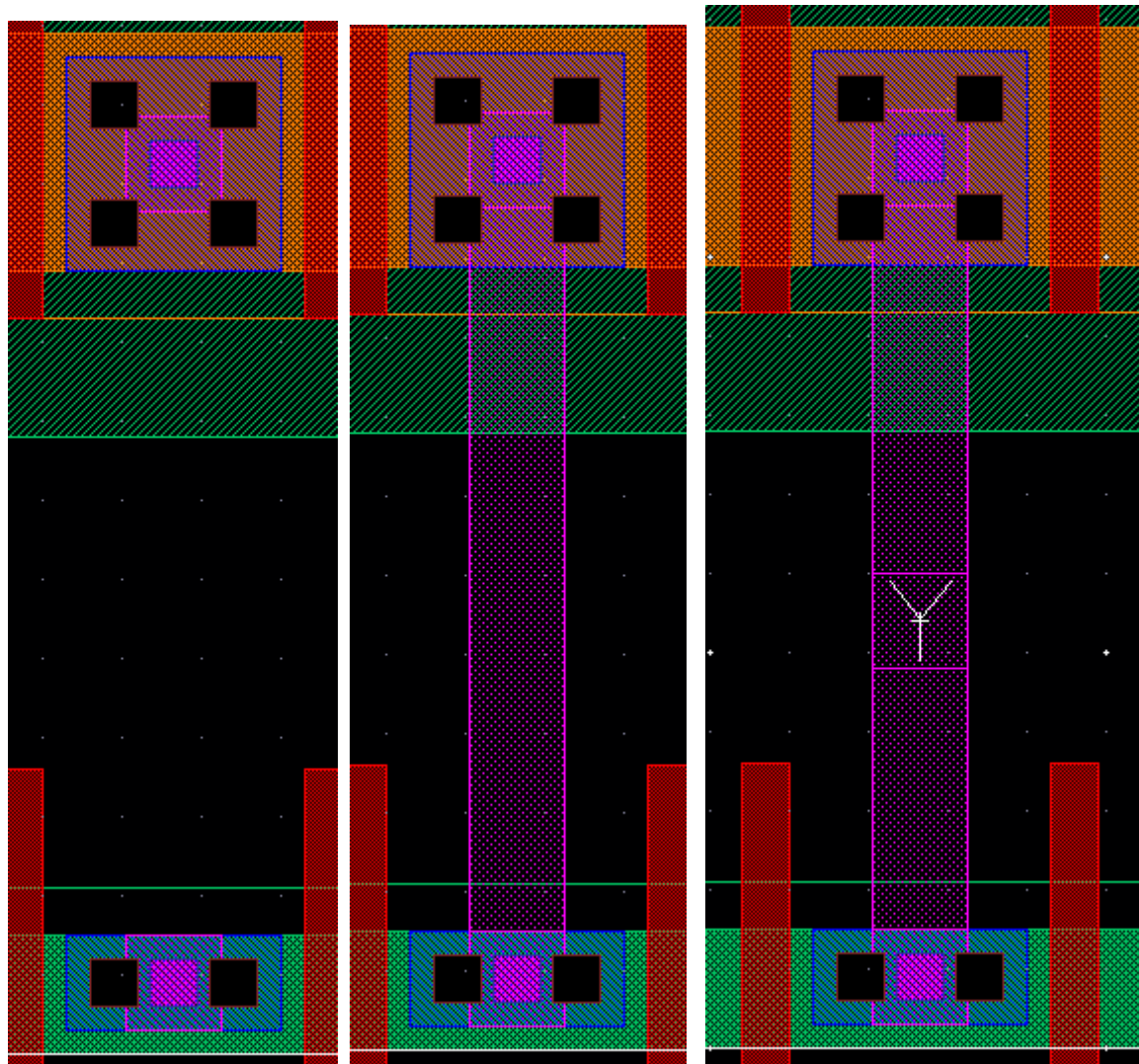


The moral of the story is that the via (metal1-metal2 connection) is so far higher than the pactive-metal1 contact in the lithography layers that we can plop the via right on top of the other cc (pactive-metal1) contact without causing a DRC error.

Moving the Y output line so far above has pros and cons:

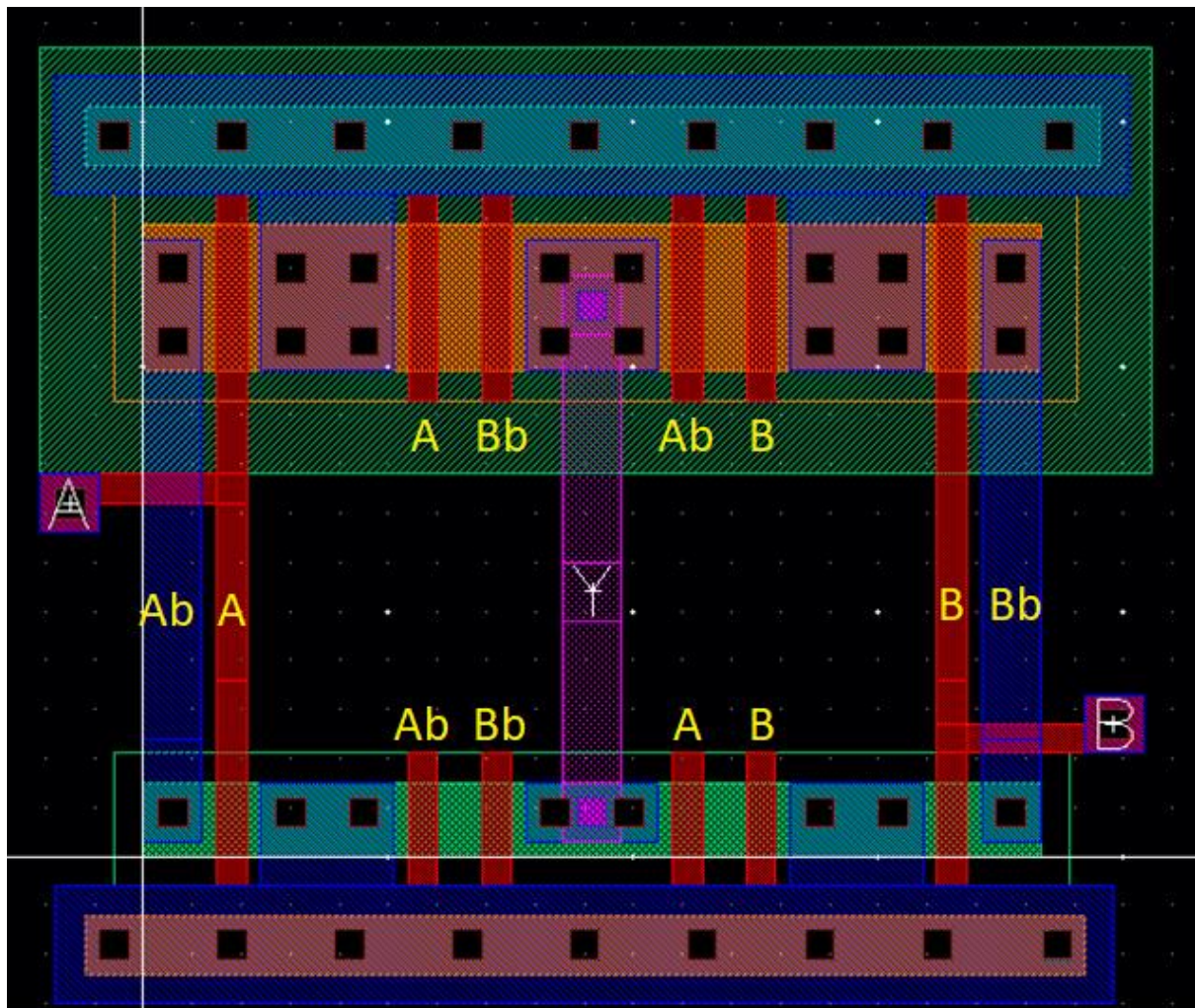
Pros: it will allow comfortable routing from the top or bottom using a metal2 line

Cons: The farther the Y output lines from each other, the longer metal2 line is necessary, going from the top to the bottom, which can block important cell-to-cell connections. However, we will choose to do this anyway and our design will look like this at this point.



We know that we will need a little more elbow room during the routing. So, we create doubled contacts where we are going to the vdd! rail and gnd! rail using the same trick as before to create a second set of contacts for the two vdd! rail connections and two gnd! rail connections.

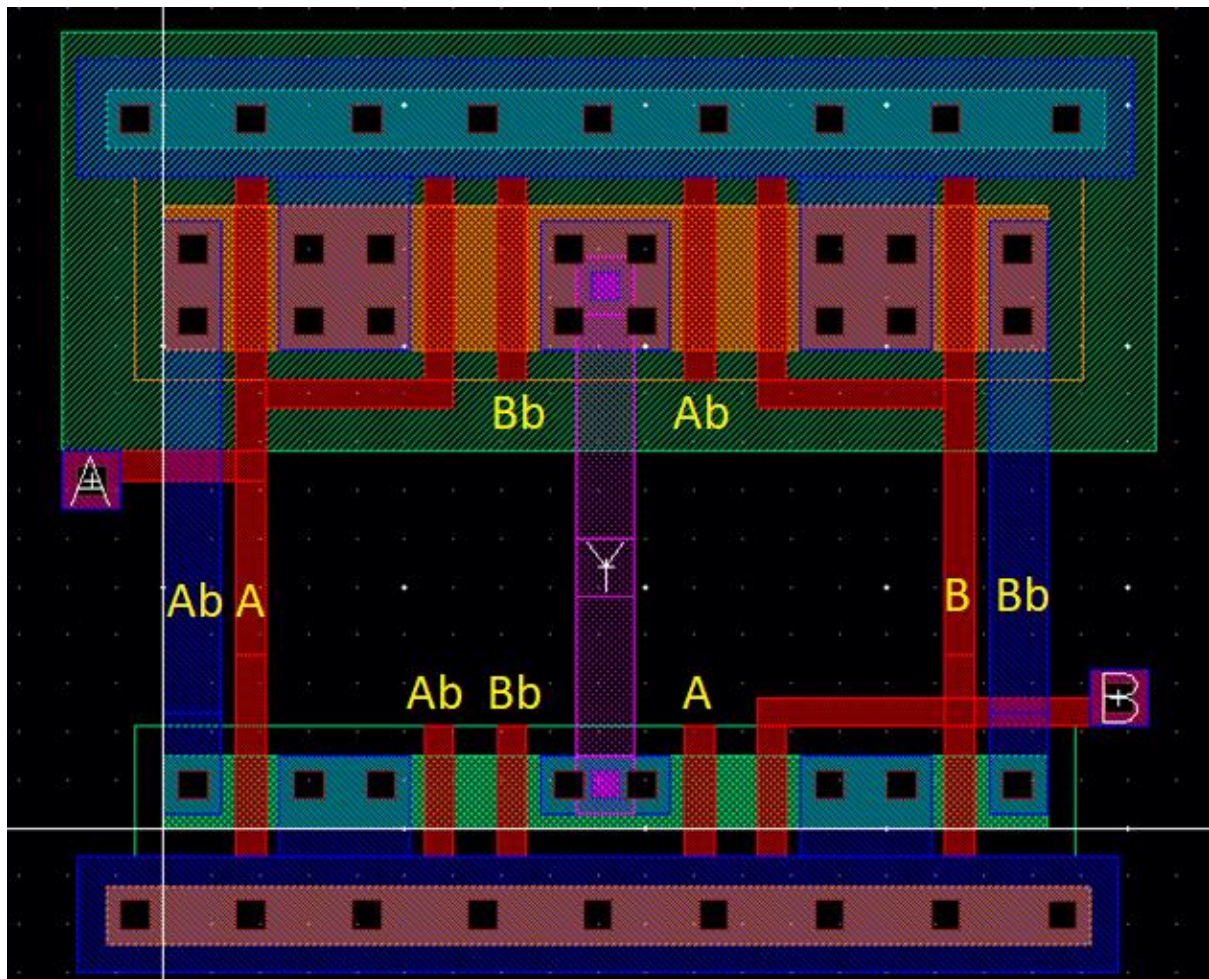
The double-padded width is  $2.7\mu\text{m}$ , whereas the single-padded width is  $1.2\mu\text{m}$ . So, each one will extend the cell towards the right by  $1.5\mu\text{m}$ . Combined, the cell will extend by  $3\mu\text{m}$  to the right.



All that is left is the 8 transistor Gate (G) poly lines that needs to be connected to their intended connection.

We will connect these 8 lines one at a time. At each step, we will make a connection that won't get us trapped or the future lines. The easiest ones are two B lines on the right and the A line on the left. They are right next to the A and B inputs and these three connections are extremely easy.



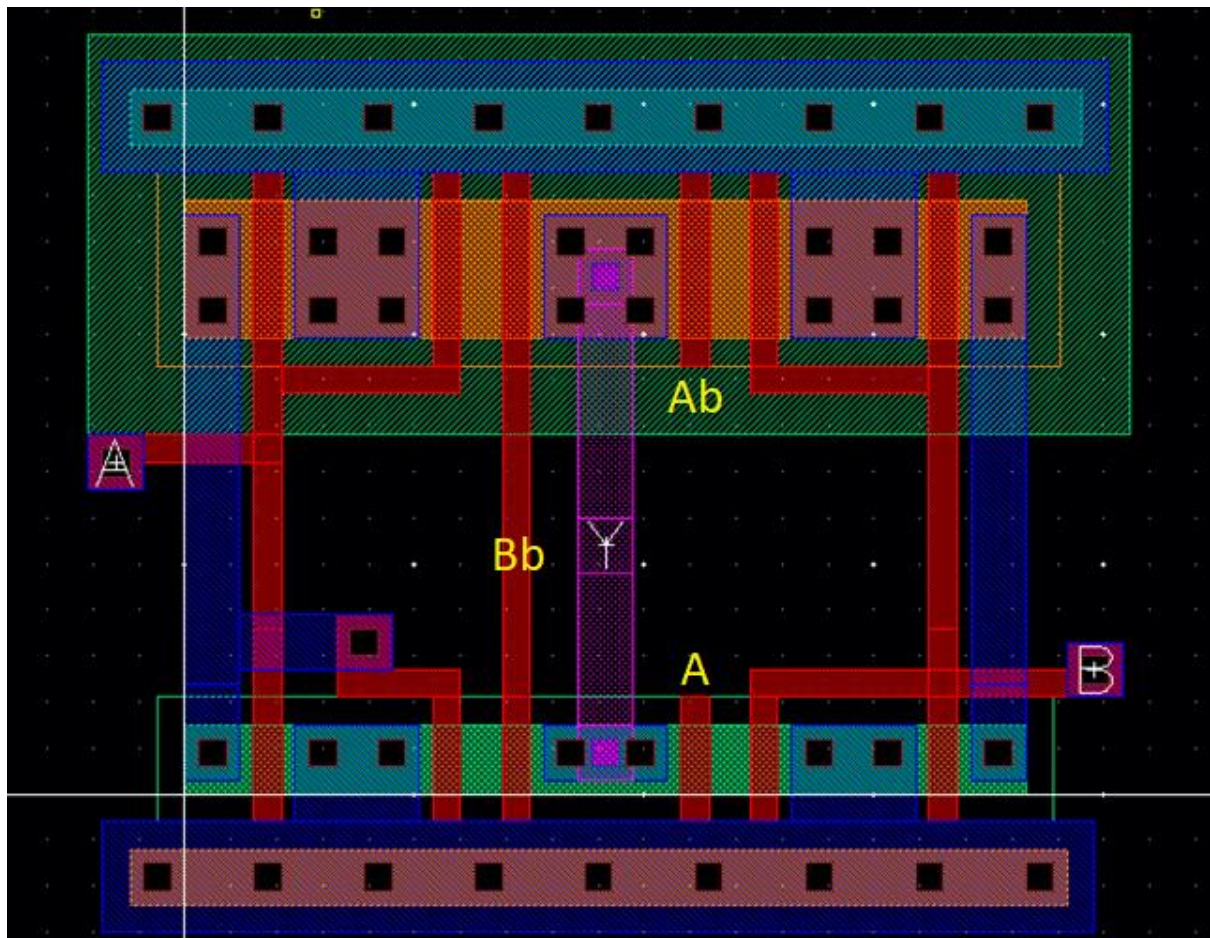


Four of the connections seem to be criss-crossing each other (Bb, Bb, Ab, A).

Ab on the bottom left is a little more isolated.

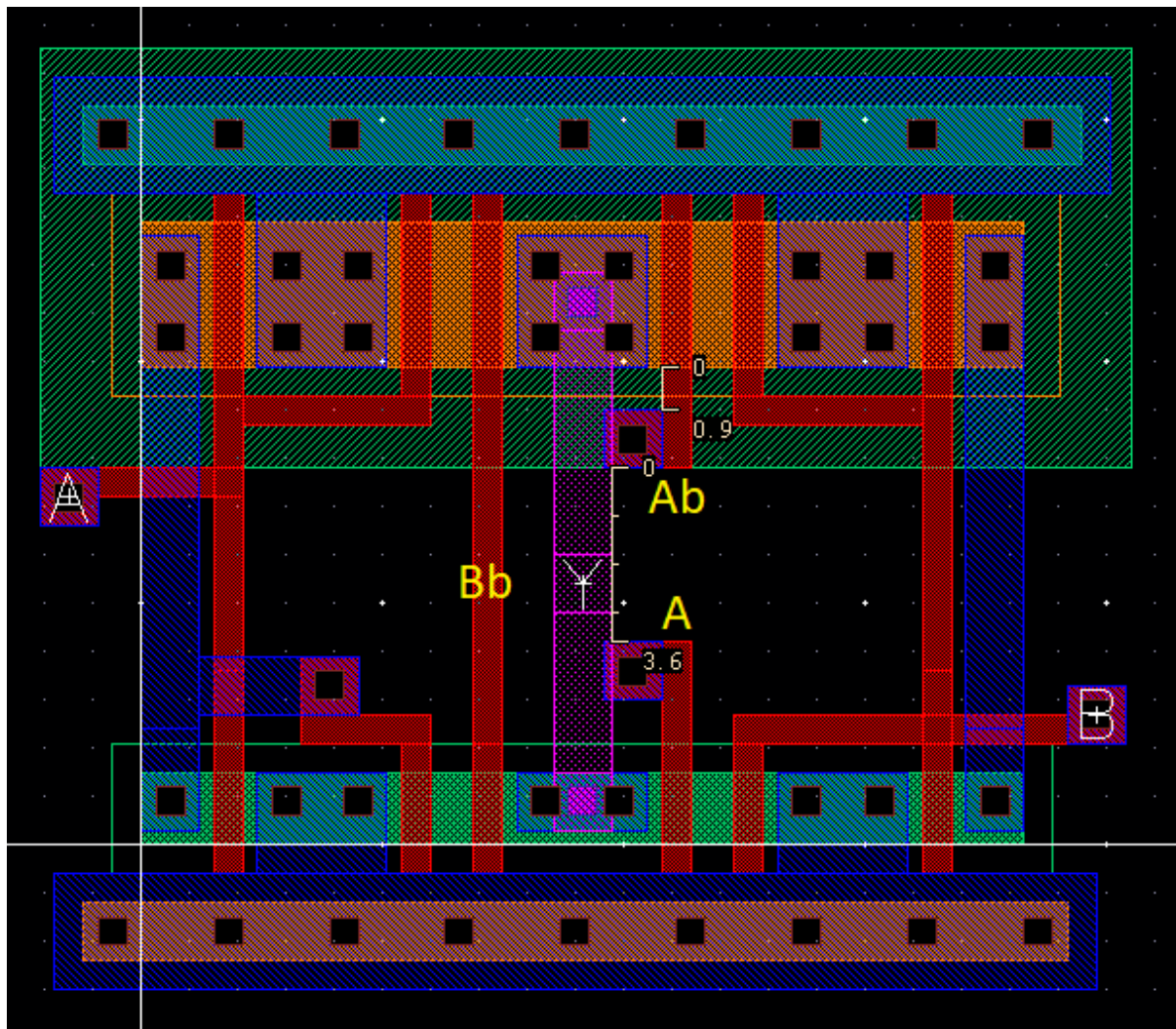
Ab is on poly and the Ab output of the inverter is on metal1. So, we will need a poly contact. Thus we put that contact as far to the left as possible, so, other poly contacts do not get too close to it. Also, we connect the two Bb lines to each other with a poly.





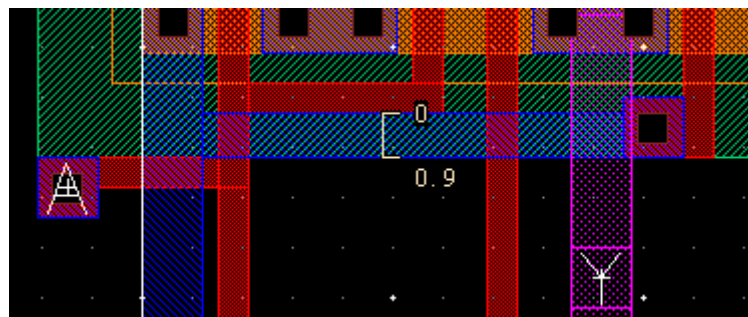
We will make a poly-metal1 pad/contact for both A and Ab poly lines.

In adding these pads, we make them as far from each other as possible.



Poly-metal1 pad on top made it easy to connect it to Ab metal1 line on the left, running vertically. The rule for metal1 thickness is that it has to be minimum  $3\lambda$ . We can make them thicker without violating a DRC rule however,  $3\lambda$  suffices ( $0.9\mu\text{m}$ ).

When running long lines, we avoid poly-metal1 overlaps, as this will create high parasitic capacitances and will degrade your circuit's performance. We used a  $3\lambda$  thick metal1 line to run the Ab to the left and thus also poly-metal1 overlaps are minimum.

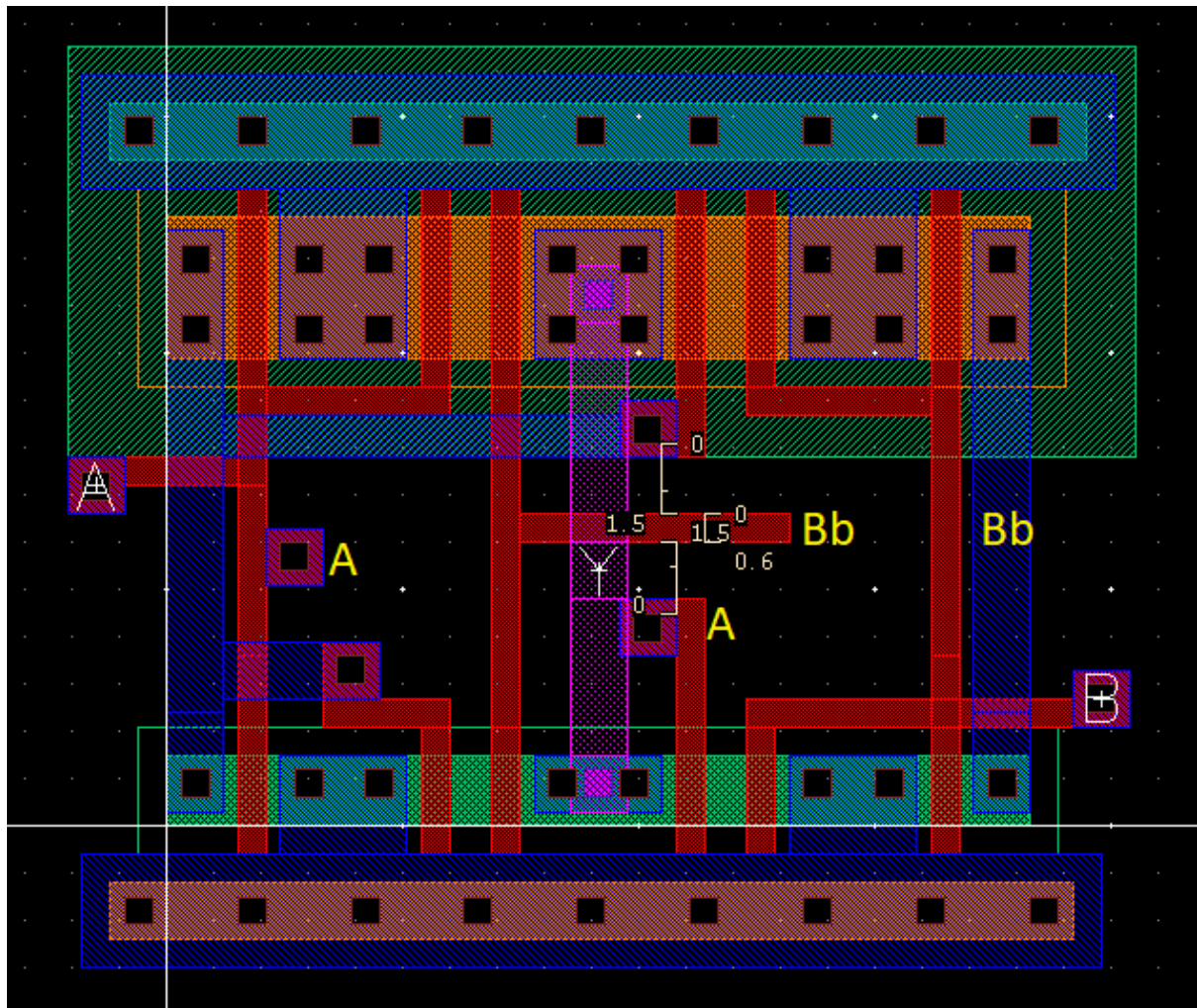


Next, we will run the A connection to the left. To run the A connection to the left it will be a metal1 running in the horizontal direction. So, it is a good idea to get the metal1-poly pad ready on the left side (A input).



For the Bb connection, we run the vertical poly line to the right and passed it under metal2. In a second, we will connect a pad to it and will connect to the metal 1 inverter output (Bb connection) on the right.

The metal1-poly pad (A) is also pushed upwards a little bit since the only DRC rule we are concerned about is the poly-poly contact spacing, which has to be  $1.5\mu\text{m}$  ( $5\lambda$ ). As seen from the rulers, the middle poly line is steering exactly  $1.5\mu\text{m}$  away from the two pads. We run DRC and it returns no errors!

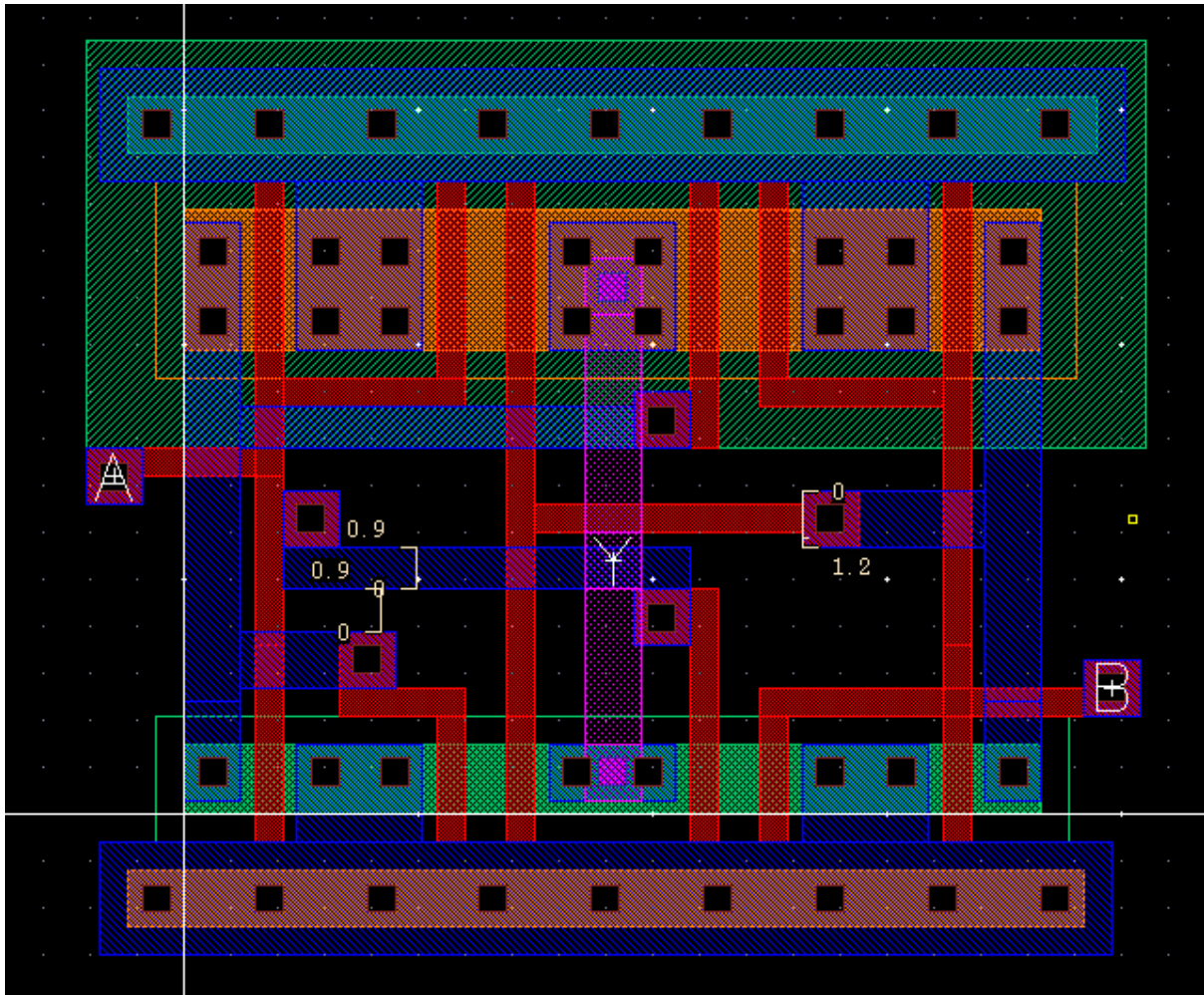


```
DRC started.....Sat Nov 30 03:17:18 2019
completed ....Sat Nov 30 03:17:18 2019
CPU TIME = 00:00:00 TOTAL TIME = 00:00:00
***** Summary of rule violations for cell "X0R2X1 layout" *****
Total errors found: 0
```

We run a  $0.9\mu\text{m}$  ( $3\lambda$ ) wide metal1 from the A input to the left moving the A input pad above a little bit, so it looks nice and symmetric. We put a poly-metal1 pad at the end of the Bb line and continued running it to the right using metal1.

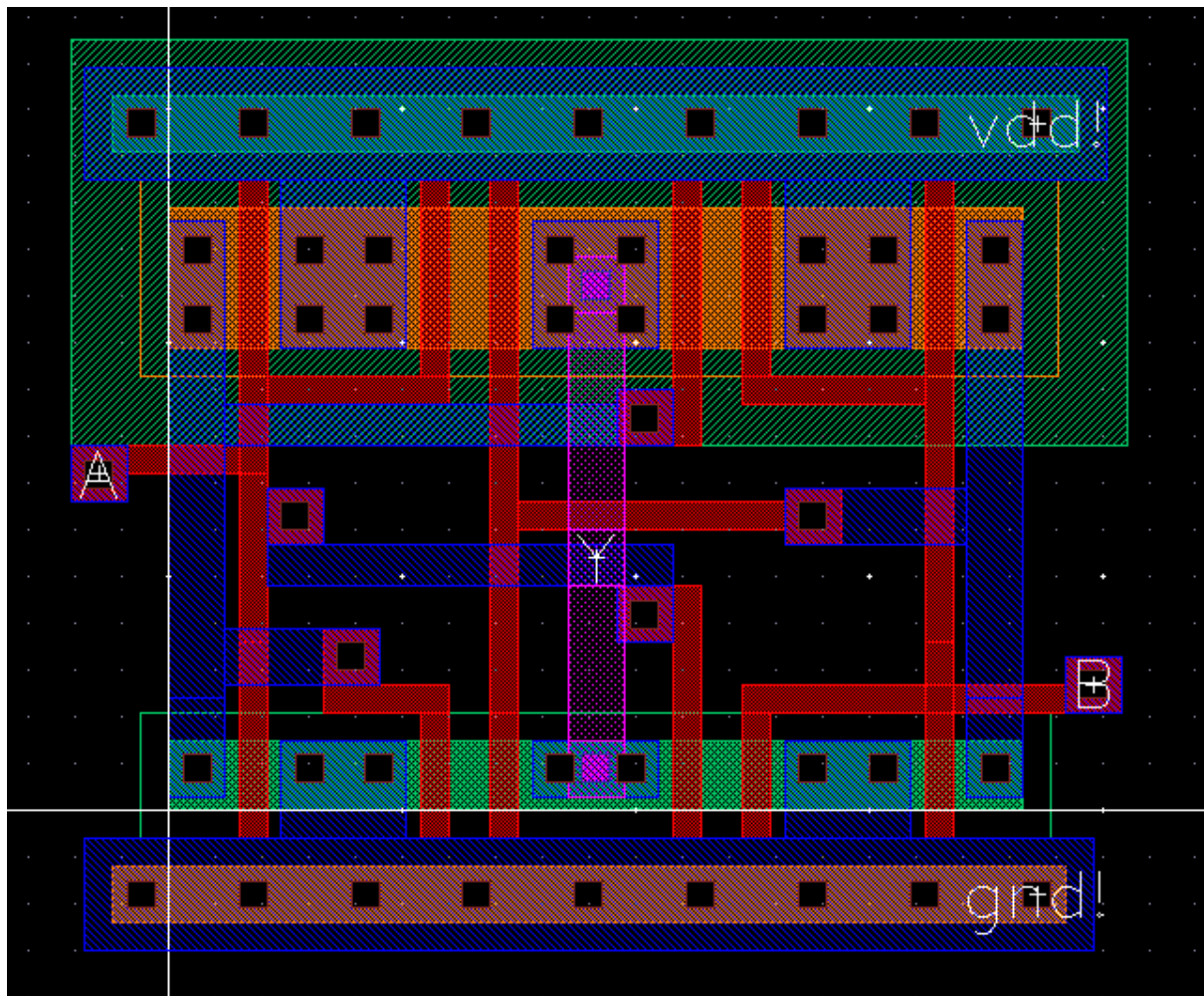
The finished layout is shown below. We run DRC. It is all good!





```
DRC started.....Sat Nov 30 03:28:49 2019
completed ....Sat Nov 30 03:28:49 2019
CPU TIME = 00:00:00  TOTAL TIME = 00:00:00
*****  Summary of rule violations for cell "XOR2X1 layout"  *****
Total errors found: 0
```

The last step is to create the vdd! and gnd! pins and label them. The final XOR2X1 layout looks like this:



We will now create the extracted view and run LVS.

Extractor
✕

Extract Method    ☒ flat    ☐ macro cell    ☐ full hier    ☐ incremental hier

View Names    Extracted     Excell

Switch Names       

Run-Specific Command File    ☐

Inclusion Limit        Limit Rule Errors    ☐

Join Nets With Same Name    ☒    Limit Run Errors    ☐

Echo Commands    ☐

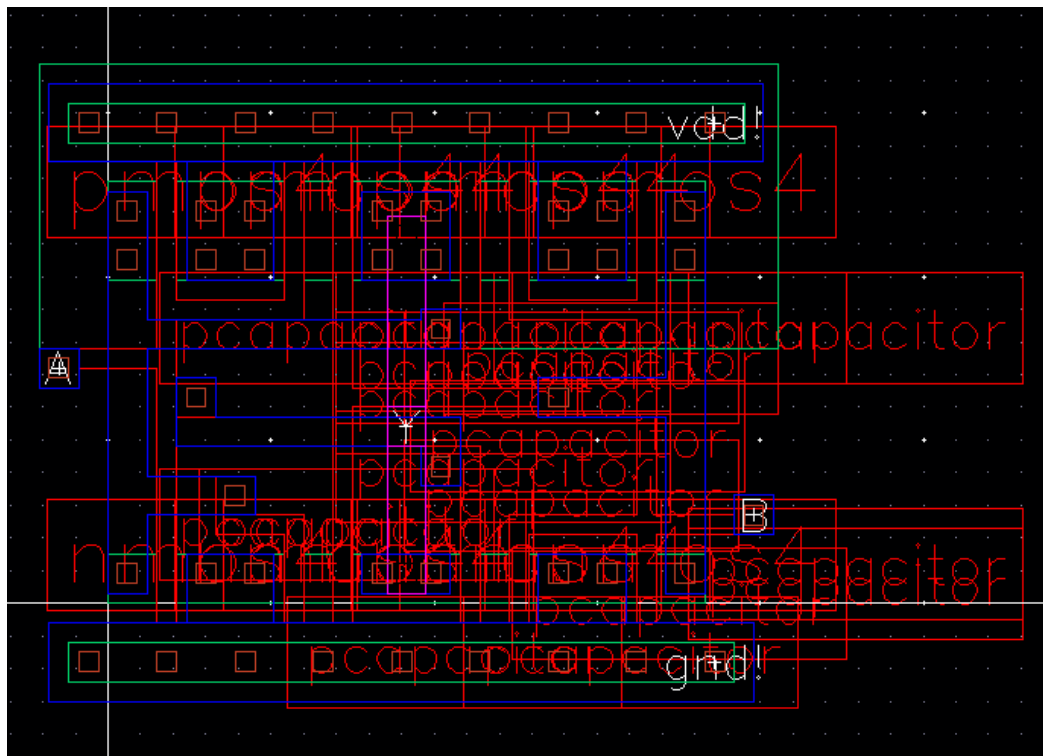
Rules File   

Rules Library    ☒

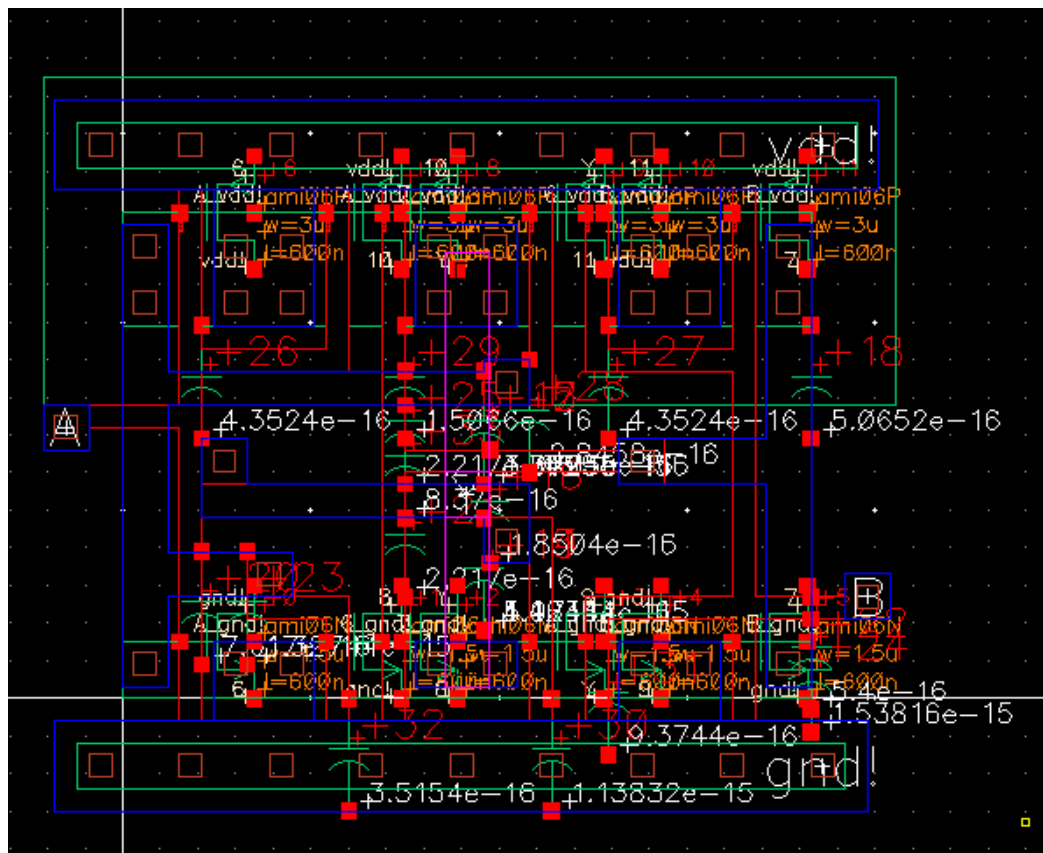
Machine    ☒ local    ☐ remote    Machine

Ignore Missing Cell Masters    ☐

saving rep LAB10/XOR2X1/extracted  
 Getting layout proper bagGetting layout proper bagLoading techComp.cxt

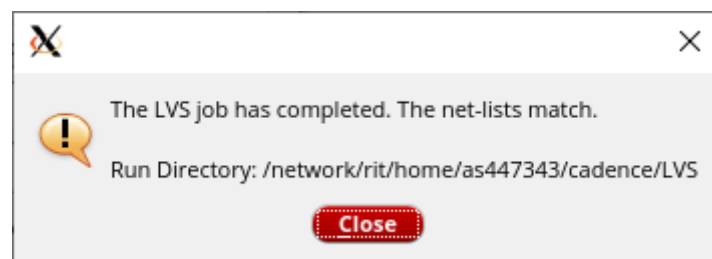


XOR2X1 extracted view (Top Level)

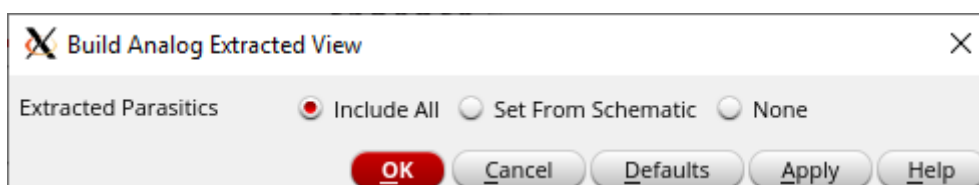


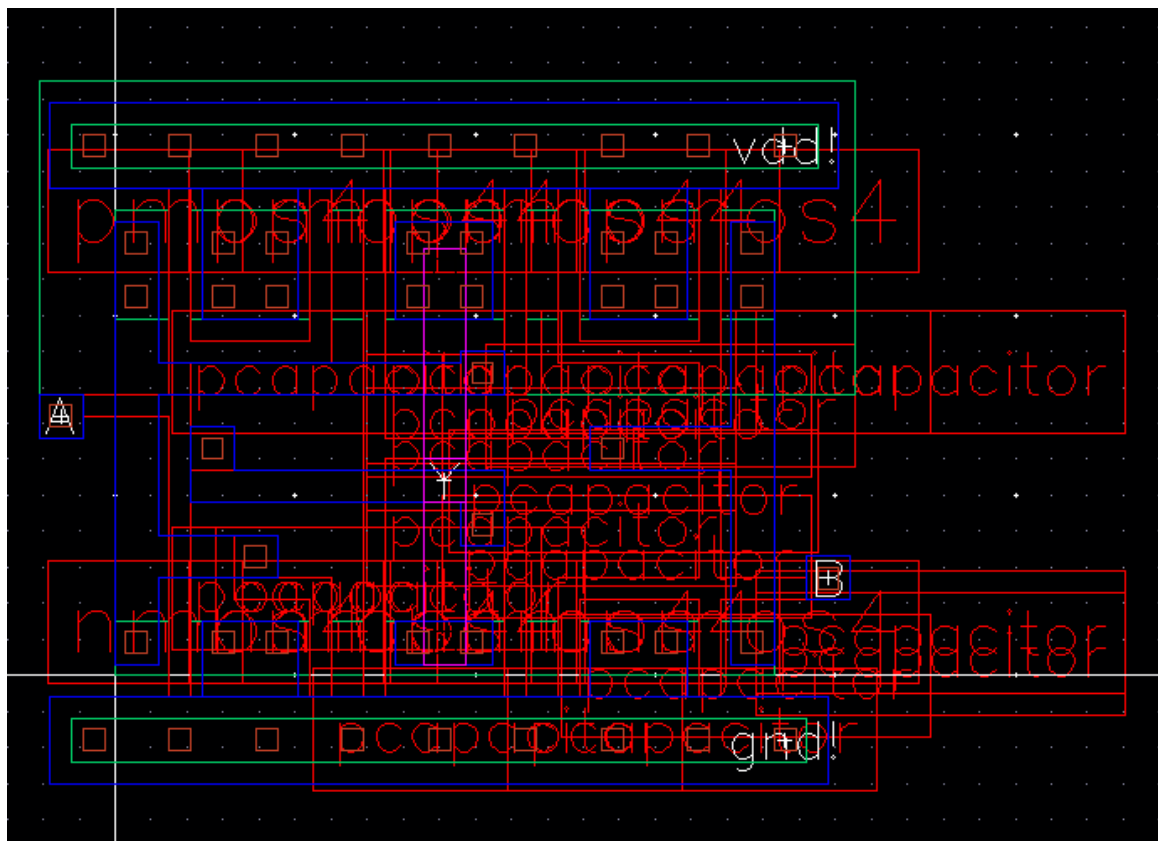
XOR2X1 extracted view (Lower Level)





The net-lists match. So we create the analog\_extracted view which will be later used in the testbench.



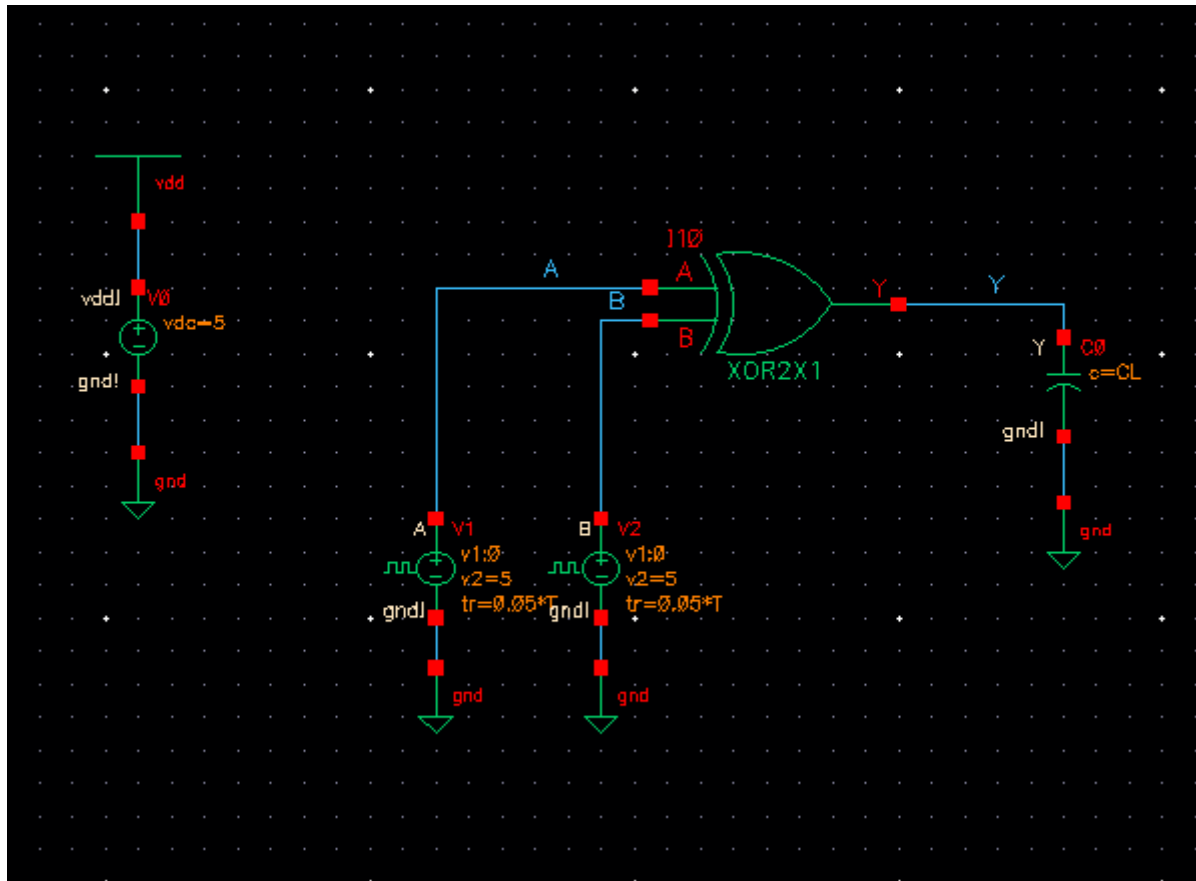


XOR2X1 analog\_extracted view

Now, we will build a test bench to test this cell. We create a TBxor cell schematic view and place labels A, B, Y on it, make cap value parametric: CL, make vdc 5V fixed, use two vpulse generators with parametric values:

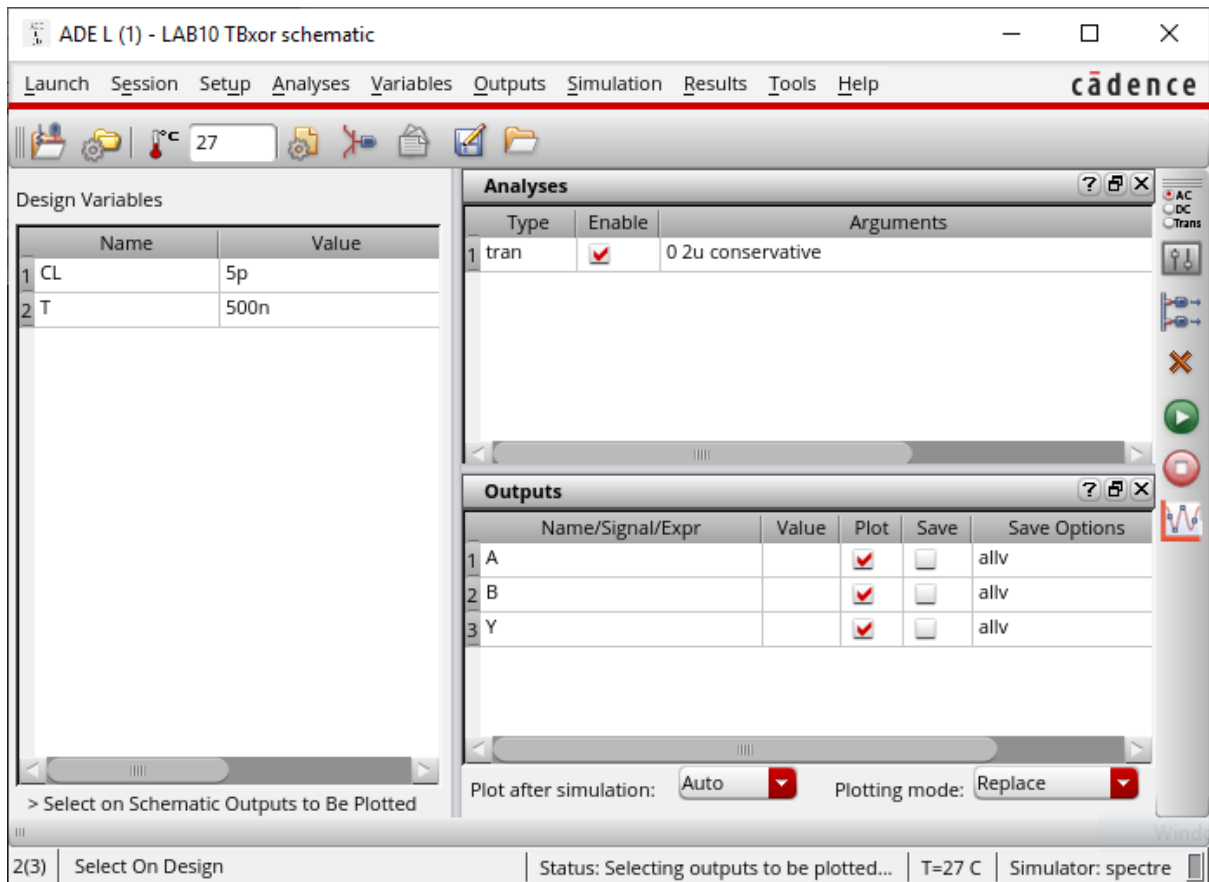
First vpulse: Delay=0, Period=T, Rise Time=0.05\*T, Fall Time=0.05\*T, Pulse Width=0.45\*T

Second vpulse: Delay=0, Period=2\*T, Rise Time=0.05\*T, Fall Time=0.05\*T, Pulse Width=0.95\*T

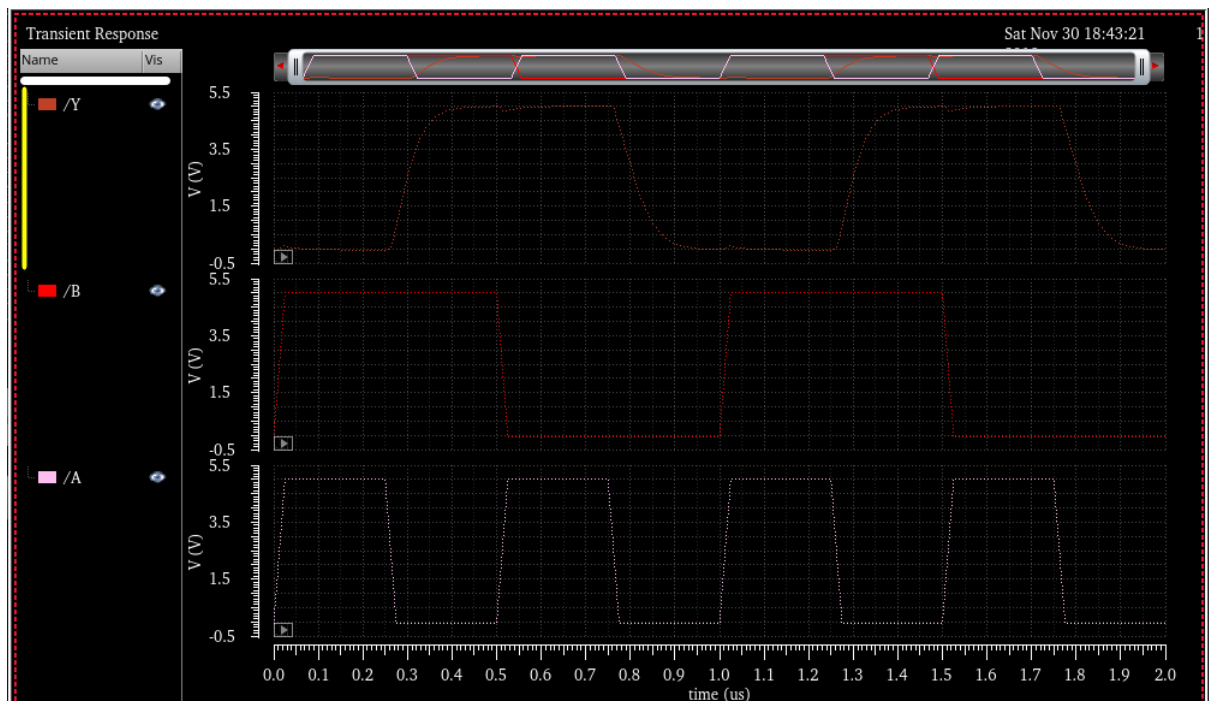


TBxor schematic





TBxor schematic ADE L

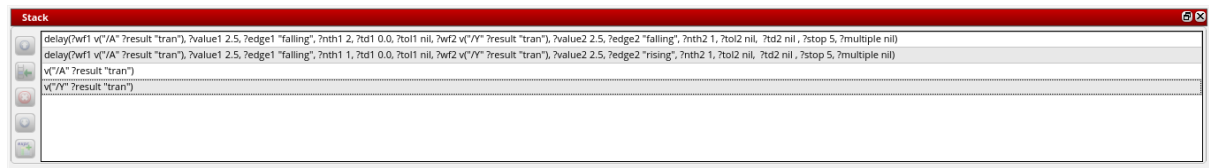


TBxor schematic output waveform (2MHz)

Now we measure the propagation delays between the inputs and the output at 50% during transition.

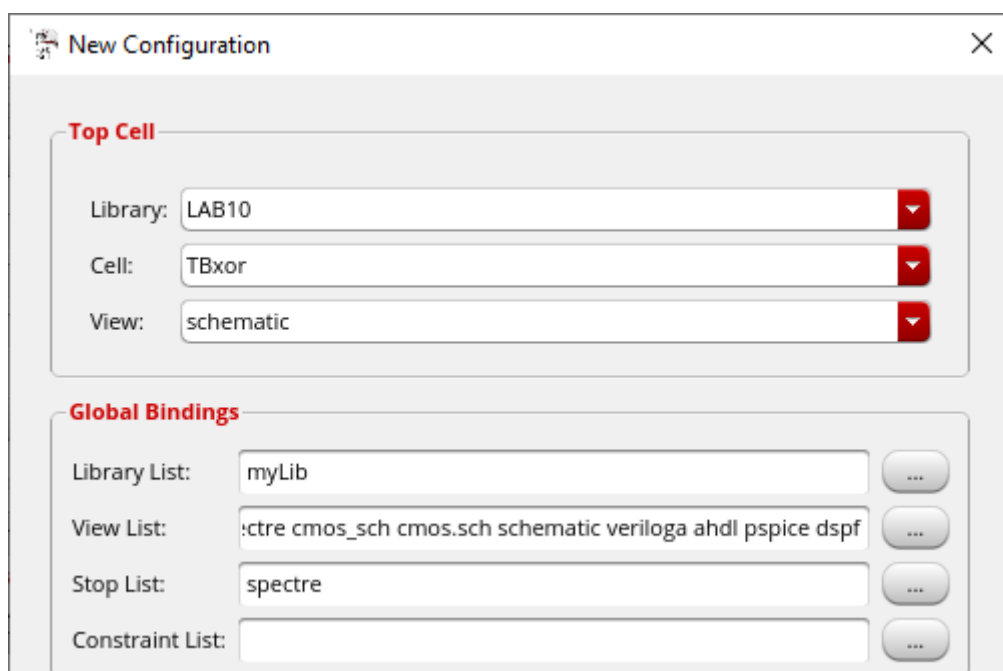
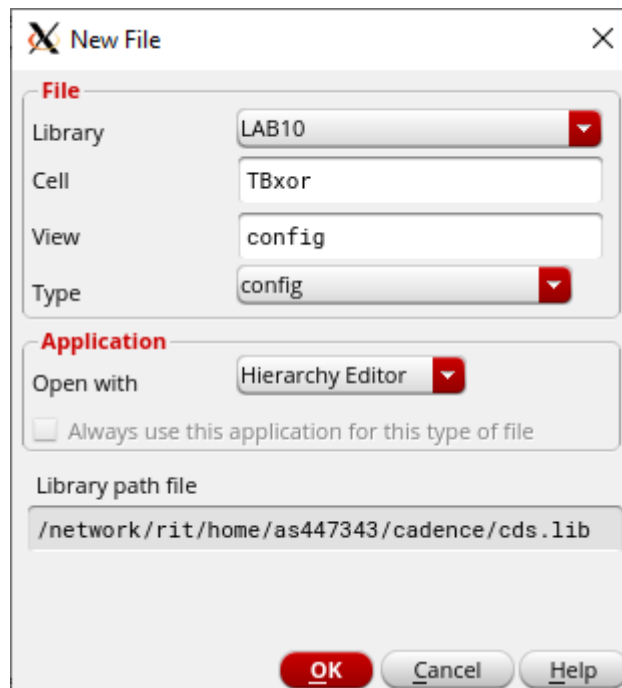
$t_{pd}(11 \rightarrow 01) = 36.9\text{E-}9 = \mathbf{36.9\text{ns}}$

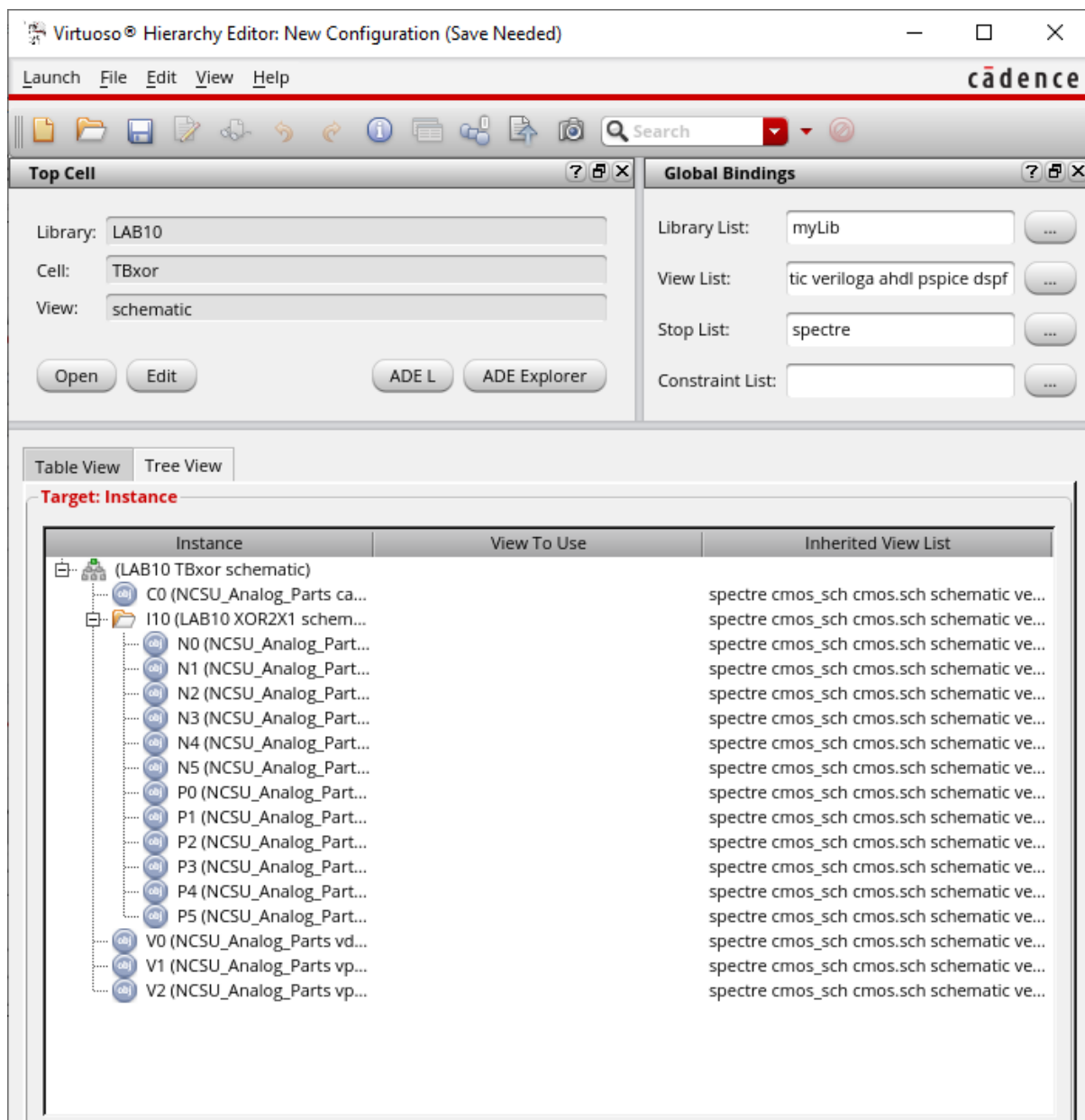
$t_{pd}(10 \rightarrow 00) = 46.76\text{E-}9 = \mathbf{46.76\text{ns}}$



This was the schematic-based simulation. It completely ignored the layout-based parasitic information.

We will now create the config view to take into account the parasitic information.



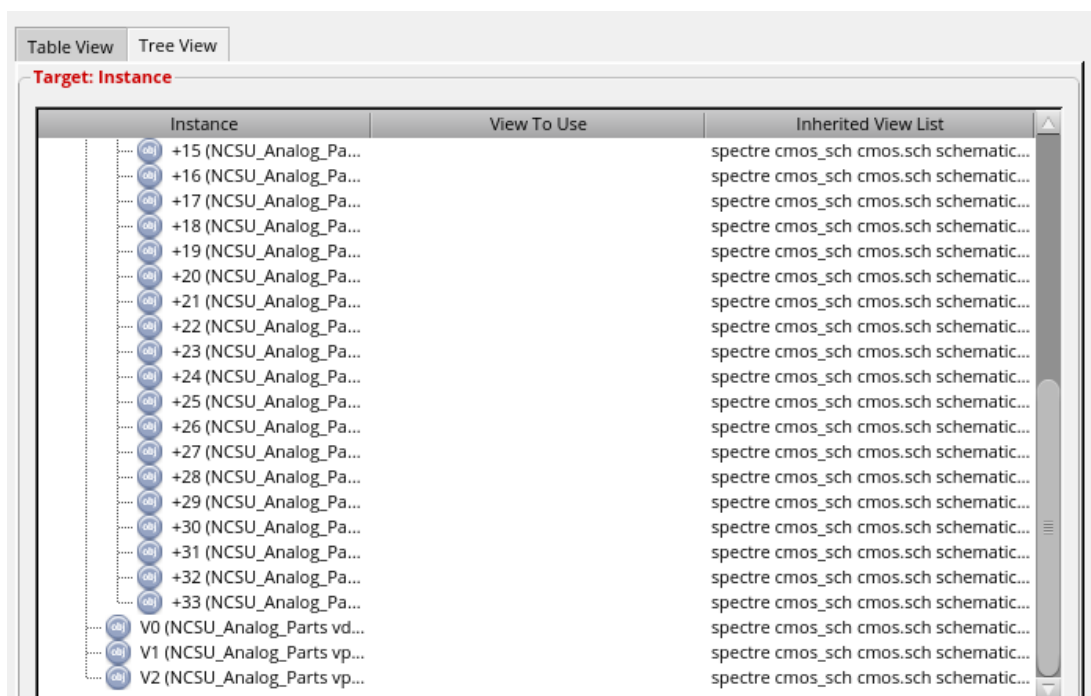
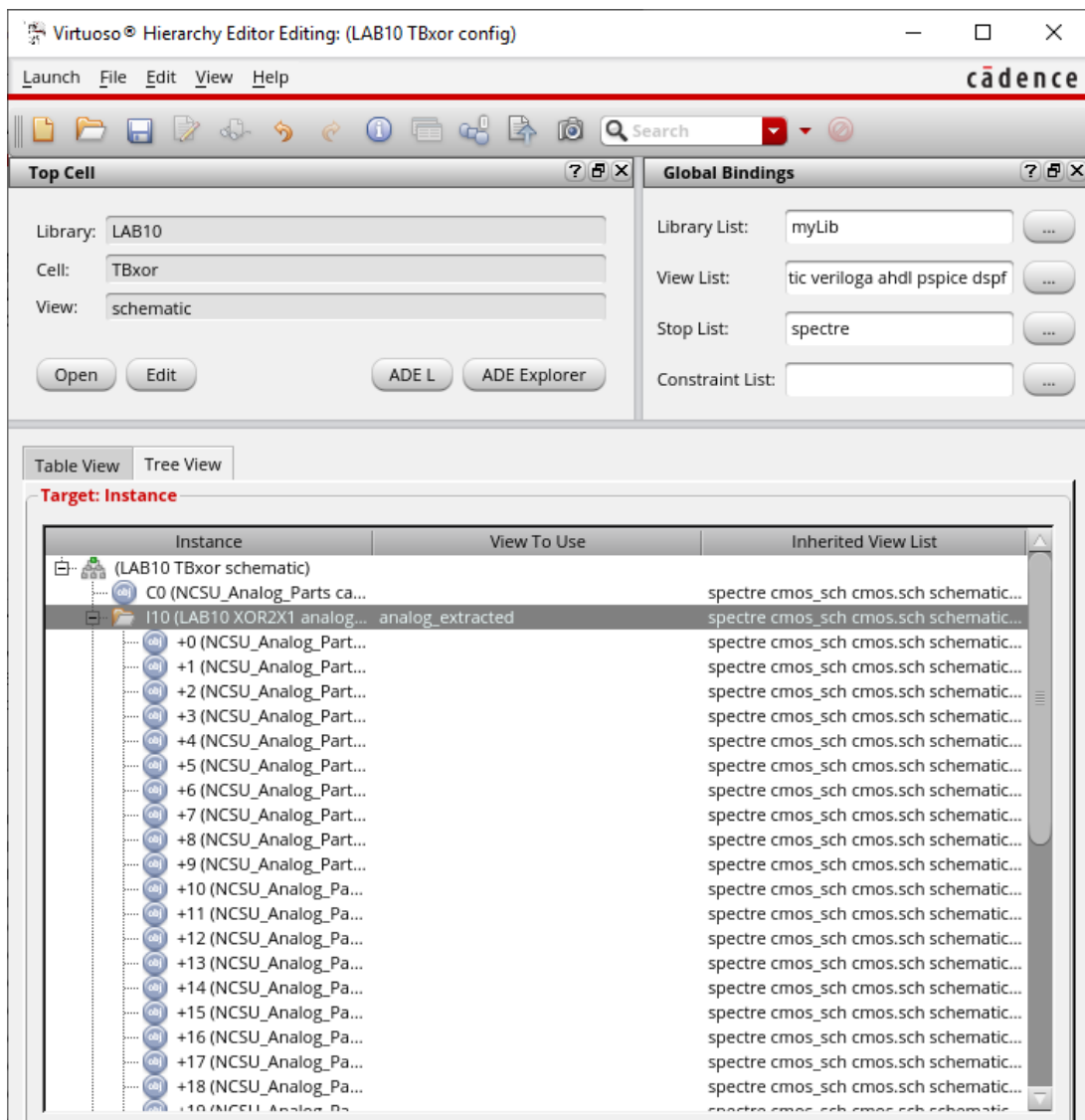


The XOR2X1 schematic has 12 transistors.

We right click XOR2X1. Set Instance View -> change from schematic to analog\_extracted.

Now we see that the analog\_extracted view has the same 12 transistors, however, has a bunch of parasitic capacitors too, which are totally related to the layout.



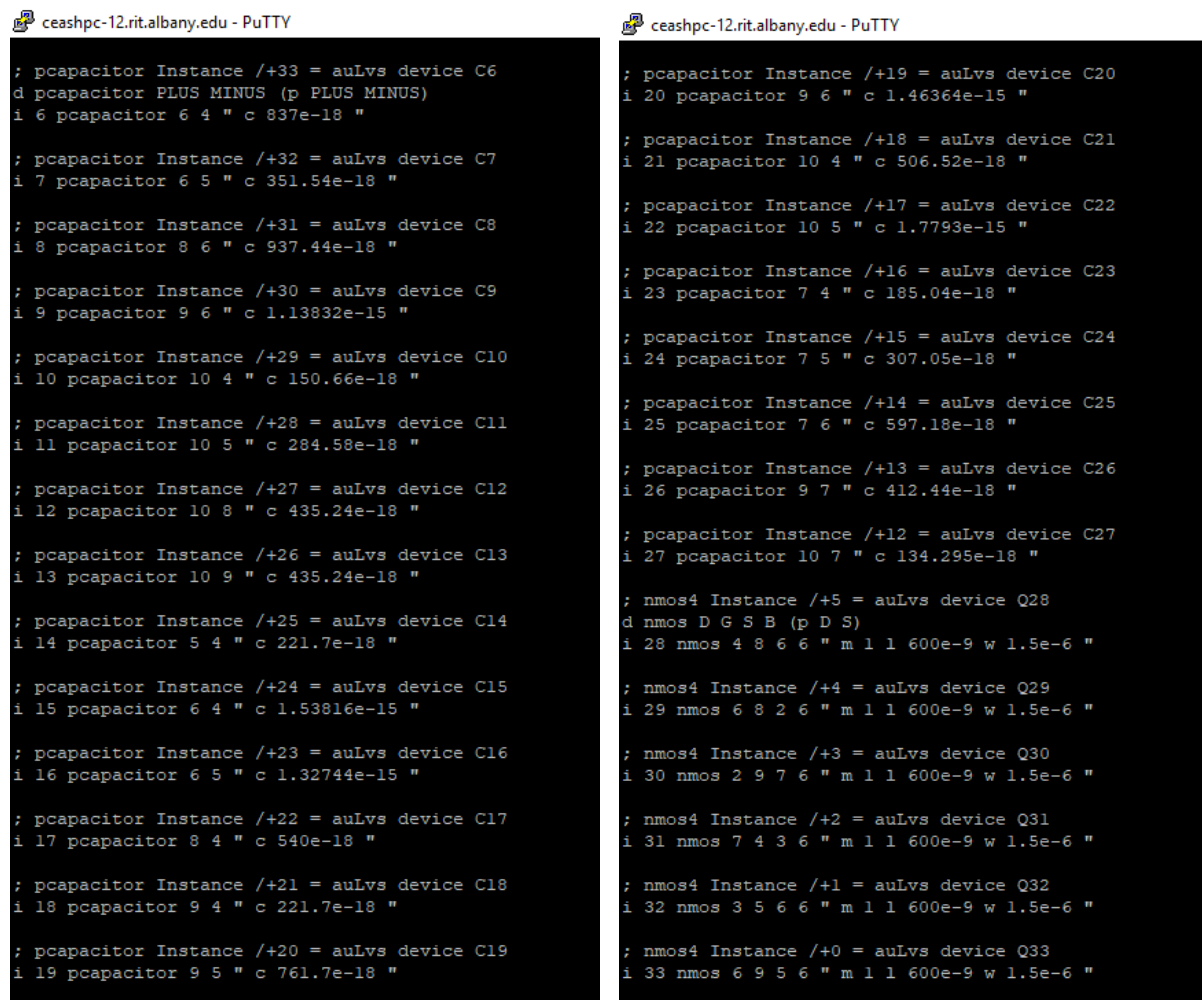


We want to know the value of the parasitic capacitances.

So we run the following commands in the PuTTY terminal:

```
cd ~/cadence/LVS/layout
```

```
cat extNetlist
```



The image shows two side-by-side PuTTY terminal windows. The left window displays the output of the 'cat extNetlist' command, showing a series of 'pcapacitor' instances (C6 to C19) with their respective values in scientific notation. The right window shows the output of the same command, displaying 'pcapacitor' instances (C20 to C27) and 'nmos4' instances (Q28 to Q33) with their respective values in scientific notation.

```
; pcapacitor Instance /+33 = auLvs device C6
d pcapacitor PLUS MINUS (p PLUS MINUS)
i 6 pcapacitor 6 4 " c 837e-18 "

; pcapacitor Instance /+32 = auLvs device C7
i 7 pcapacitor 6 5 " c 351.54e-18 "

; pcapacitor Instance /+31 = auLvs device C8
i 8 pcapacitor 8 6 " c 937.44e-18 "

; pcapacitor Instance /+30 = auLvs device C9
i 9 pcapacitor 9 6 " c 1.13832e-15 "

; pcapacitor Instance /+29 = auLvs device C10
i 10 pcapacitor 10 4 " c 150.66e-18 "

; pcapacitor Instance /+28 = auLvs device C11
i 11 pcapacitor 10 5 " c 284.58e-18 "

; pcapacitor Instance /+27 = auLvs device C12
i 12 pcapacitor 10 8 " c 435.24e-18 "

; pcapacitor Instance /+26 = auLvs device C13
i 13 pcapacitor 10 9 " c 435.24e-18 "

; pcapacitor Instance /+25 = auLvs device C14
i 14 pcapacitor 5 4 " c 221.7e-18 "

; pcapacitor Instance /+24 = auLvs device C15
i 15 pcapacitor 6 4 " c 1.53816e-15 "

; pcapacitor Instance /+23 = auLvs device C16
i 16 pcapacitor 6 5 " c 1.32744e-15 "

; pcapacitor Instance /+22 = auLvs device C17
i 17 pcapacitor 8 4 " c 540e-18 "

; pcapacitor Instance /+21 = auLvs device C18
i 18 pcapacitor 9 4 " c 221.7e-18 "

; pcapacitor Instance /+20 = auLvs device C19
i 19 pcapacitor 9 5 " c 761.7e-18 "

; pcapacitor Instance /+19 = auLvs device C20
i 20 pcapacitor 9 6 " c 1.46364e-15 "

; pcapacitor Instance /+18 = auLvs device C21
i 21 pcapacitor 10 4 " c 506.52e-18 "

; pcapacitor Instance /+17 = auLvs device C22
i 22 pcapacitor 10 5 " c 1.7793e-15 "

; pcapacitor Instance /+16 = auLvs device C23
i 23 pcapacitor 7 4 " c 185.04e-18 "

; pcapacitor Instance /+15 = auLvs device C24
i 24 pcapacitor 7 5 " c 307.05e-18 "

; pcapacitor Instance /+14 = auLvs device C25
i 25 pcapacitor 7 6 " c 597.18e-18 "

; pcapacitor Instance /+13 = auLvs device C26
i 26 pcapacitor 9 7 " c 412.44e-18 "

; pcapacitor Instance /+12 = auLvs device C27
i 27 pcapacitor 10 7 " c 134.295e-18 "

; nmos4 Instance /+5 = auLvs device Q28
d nmos D G S B (p D S)
i 28 nmos 4 8 6 6 " m 1 1 600e-9 w 1.5e-6 "

; nmos4 Instance /+4 = auLvs device Q29
i 29 nmos 6 8 2 6 " m 1 1 600e-9 w 1.5e-6 "

; nmos4 Instance /+3 = auLvs device Q30
i 30 nmos 2 9 7 6 " m 1 1 600e-9 w 1.5e-6 "

; nmos4 Instance /+2 = auLvs device Q31
i 31 nmos 7 4 3 6 " m 1 1 600e-9 w 1.5e-6 "

; nmos4 Instance /+1 = auLvs device Q32
i 32 nmos 3 5 6 6 " m 1 1 600e-9 w 1.5e-6 "

; nmos4 Instance /+0 = auLvs device Q33
i 33 nmos 6 9 5 6 " m 1 1 600e-9 w 1.5e-6 "
```

**pcapacitor** values seem to be:

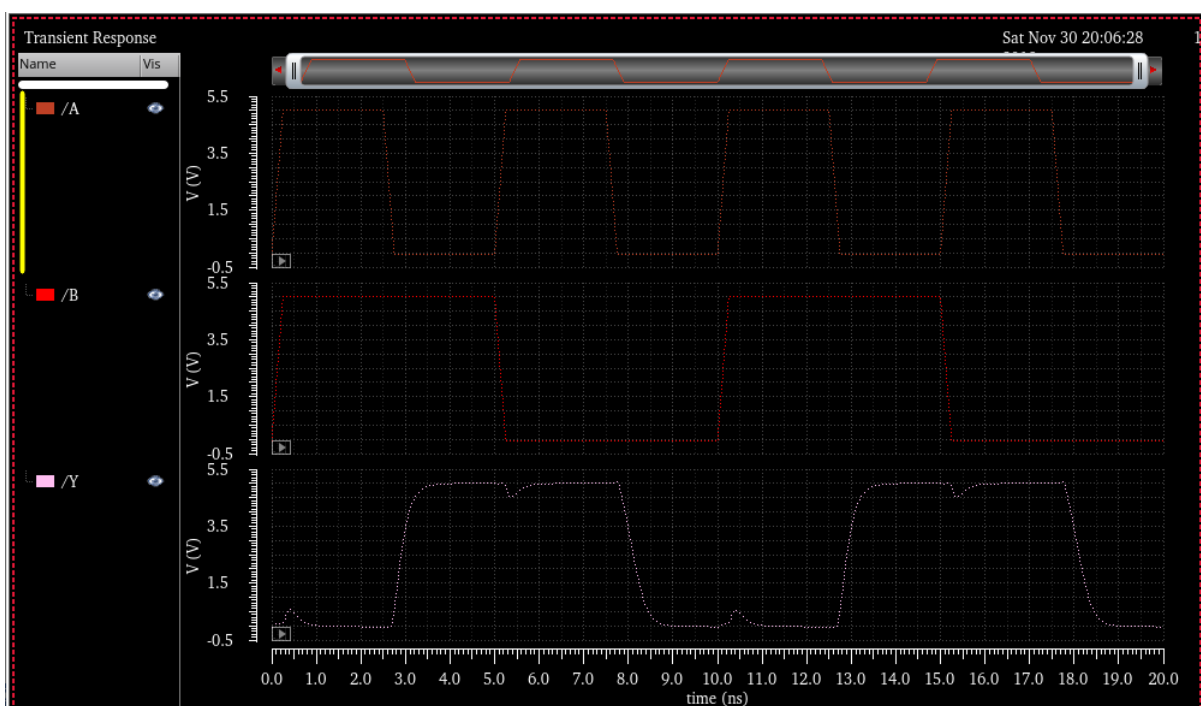
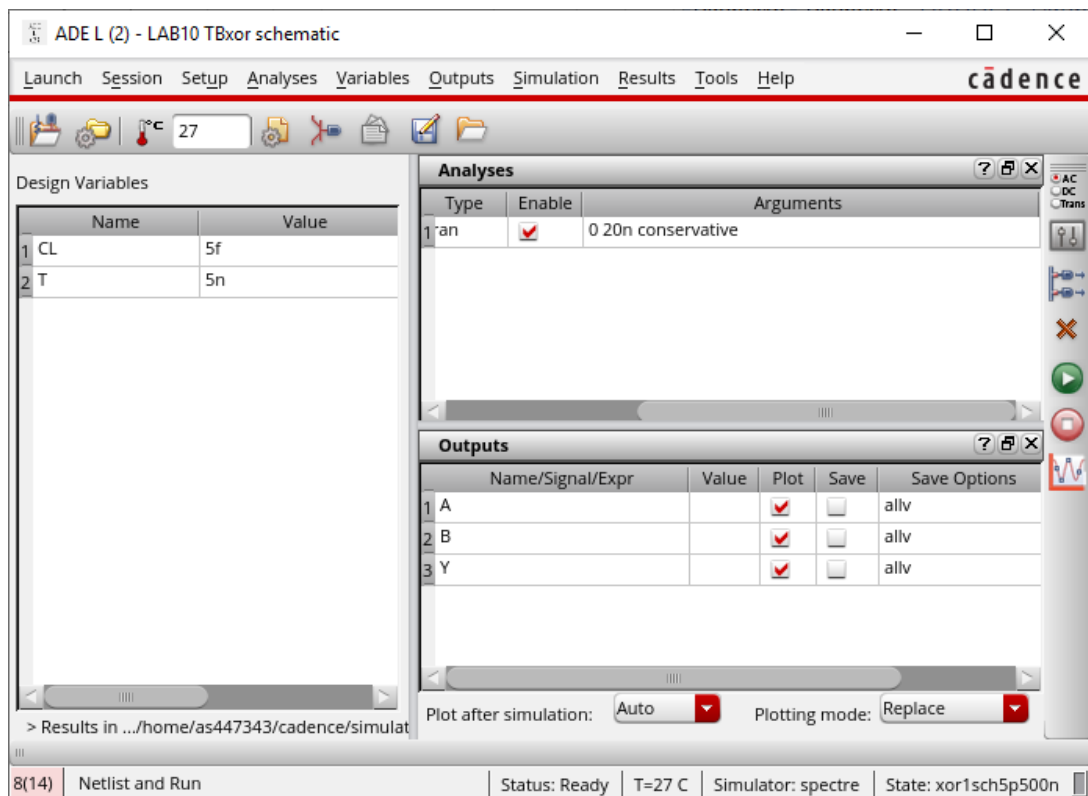
837 aF, 352 aF, 151 aF, 285 aF, 937 aF, 435 aF, 1071 aF, 502 aF, 222 aF, 1538 aF, 1327 aF, 507 aF, 1779 aF, 540 aF, 222 aF, 1464 aF, 185 aF, 307 aF, 597 aF, 134 aF, 412 aF.

Although these are pretty small individually, their effect on the overall performance can be determined by simulating the config view.

The previous simulation was to test the XOR2X1 with a pulse of  $T=500\text{ns}=0.5\mu\text{s}$ , which is 2 MHz.

2MHz is extremely slow for this technology. We can go a lot faster. Hence we try with 200MHz. 200MHz is 100x faster than our previous simulation, which corresponds to  $T=5\text{ns}$ .

In ADE L, we Netlist and Run using  $CL=5\text{f}$ ,  $T=5\text{n}$ .



TBxor schematic output waveform (200MHz)

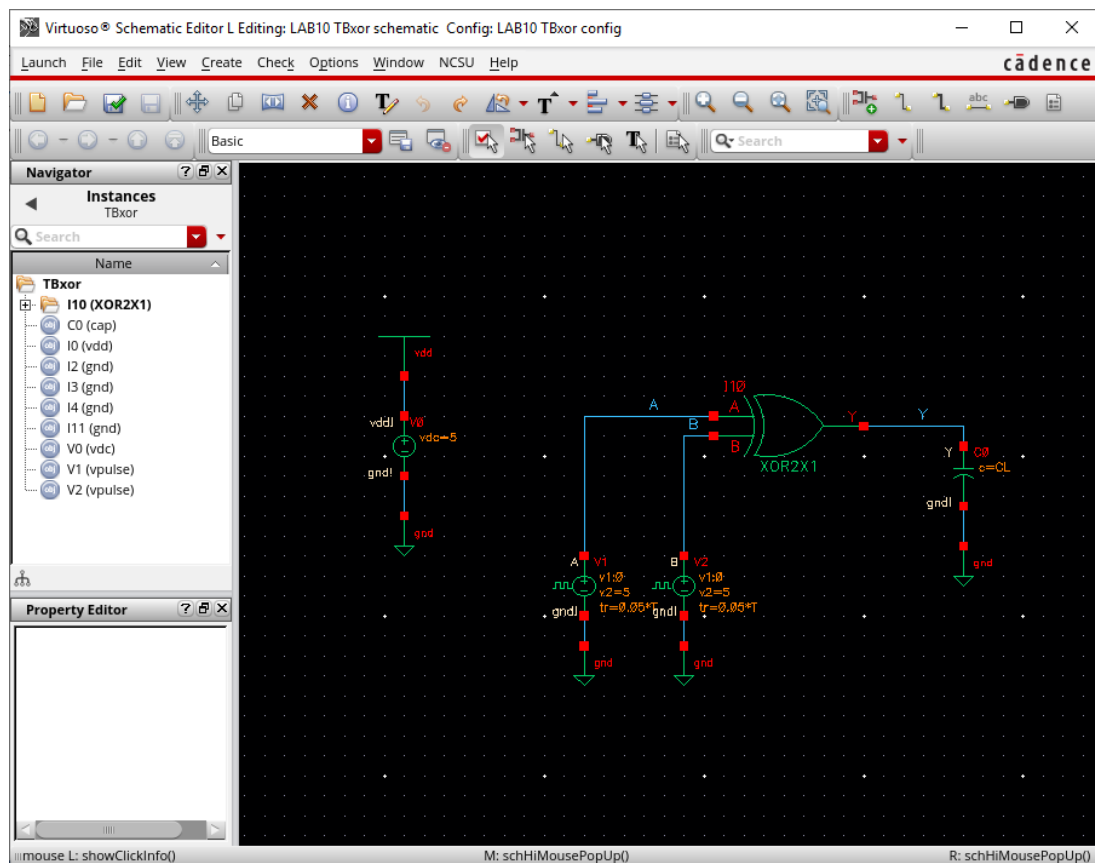
Now we again measure the propagation delays between the inputs and the output at 50% during transition.

$$t_{pd}(11 \rightarrow 01) = 274.1 \text{E-12} = \mathbf{274.1 \text{ps}}$$

$$t_{pd}(10 \rightarrow 00) = 485.7 \text{E-12} = \mathbf{485.7 \text{ps}}$$

Now, we will compare these delays to the simulation using the parasitic capacitances.





TBxor schematic using config

ADE L (1) - LAB10 TBxor config

Launch Session Setup Analyses Variables Outputs Simulation Results Tools Help

Design Variables

Name	Value
1 CL	5f
2 T	5n

Analyses

Type	Enable	Arguments
1 tran	<input checked="" type="checkbox"/>	0.20n conservative

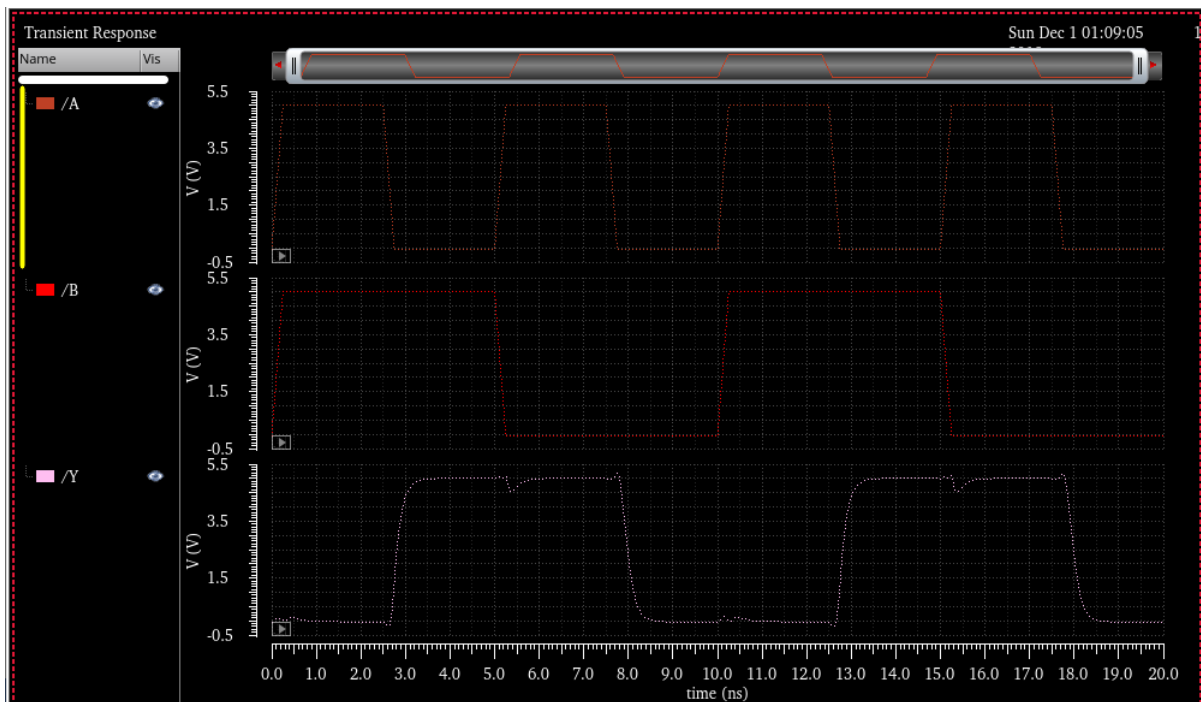
Outputs

Name/Signal/Expr	Value	Plot	Save	Save Options
1 A		<input checked="" type="checkbox"/>	<input type="checkbox"/>	allv
2 B		<input checked="" type="checkbox"/>	<input type="checkbox"/>	allv
3 Y		<input checked="" type="checkbox"/>	<input type="checkbox"/>	allv

Plot after simulation: Auto Plotting mode: Replace

2(4) Netlist and Run Status: Ready T=27 C Simulator: spectre State: xor1sch5f5n

TBxor config ADE L



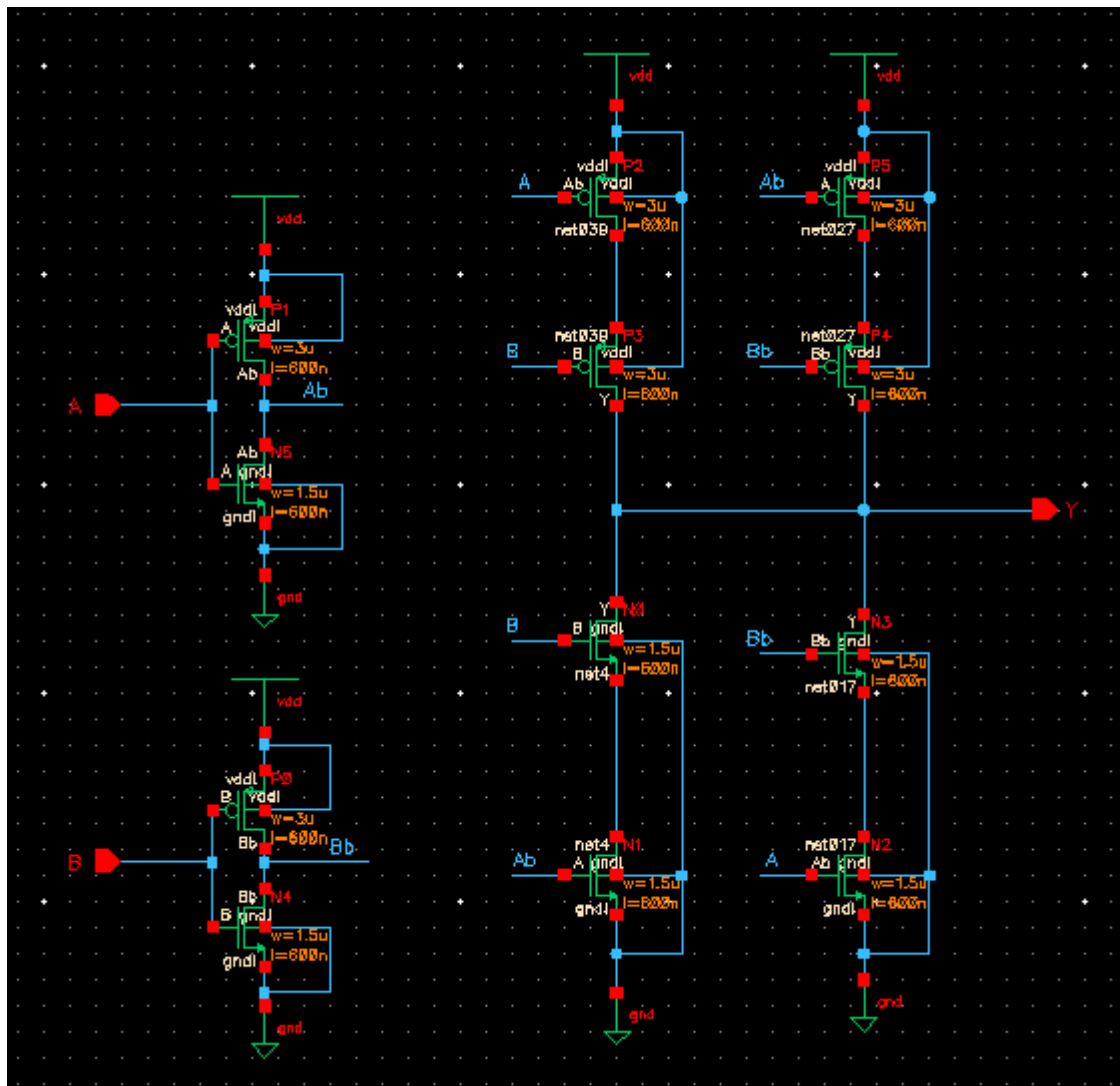
TBxor schematic output waveform (**200MHz with parasitics**)

$$t_{pd}(11 \rightarrow 01) = 181.5\text{E-}12 = \mathbf{181.5ps}$$

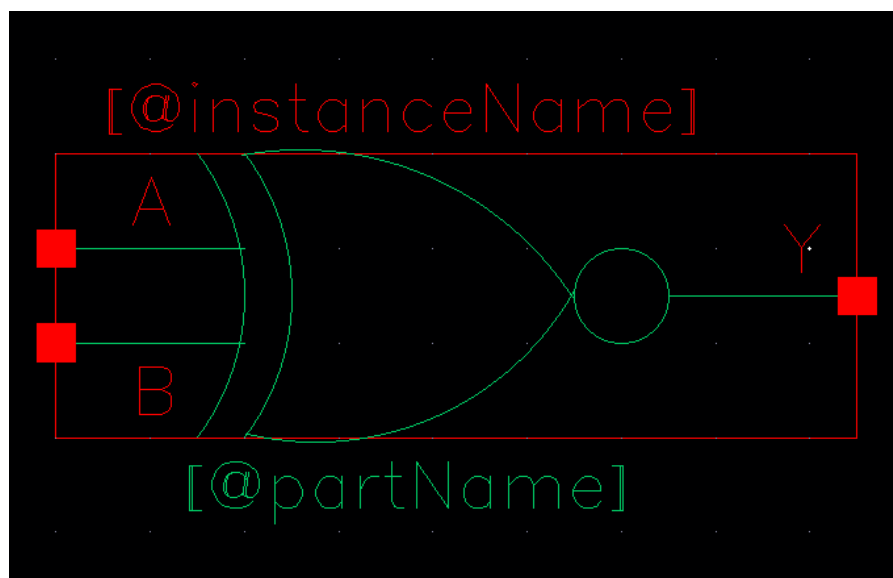
$$t_{pd}(10 \rightarrow 00) = 362.0\text{E-}12 = \mathbf{362ps}$$

This shows an interesting result when the values are compared. The delay is comparatively less with the layout taken into account, even when the parasitic capacitors are added compared to the schematic-only simulation. This is due to the fact that we drew the layout very clean with minimal routing.

## 2) Designing an XNOR2X1 cell



XNOR2X1 schematic

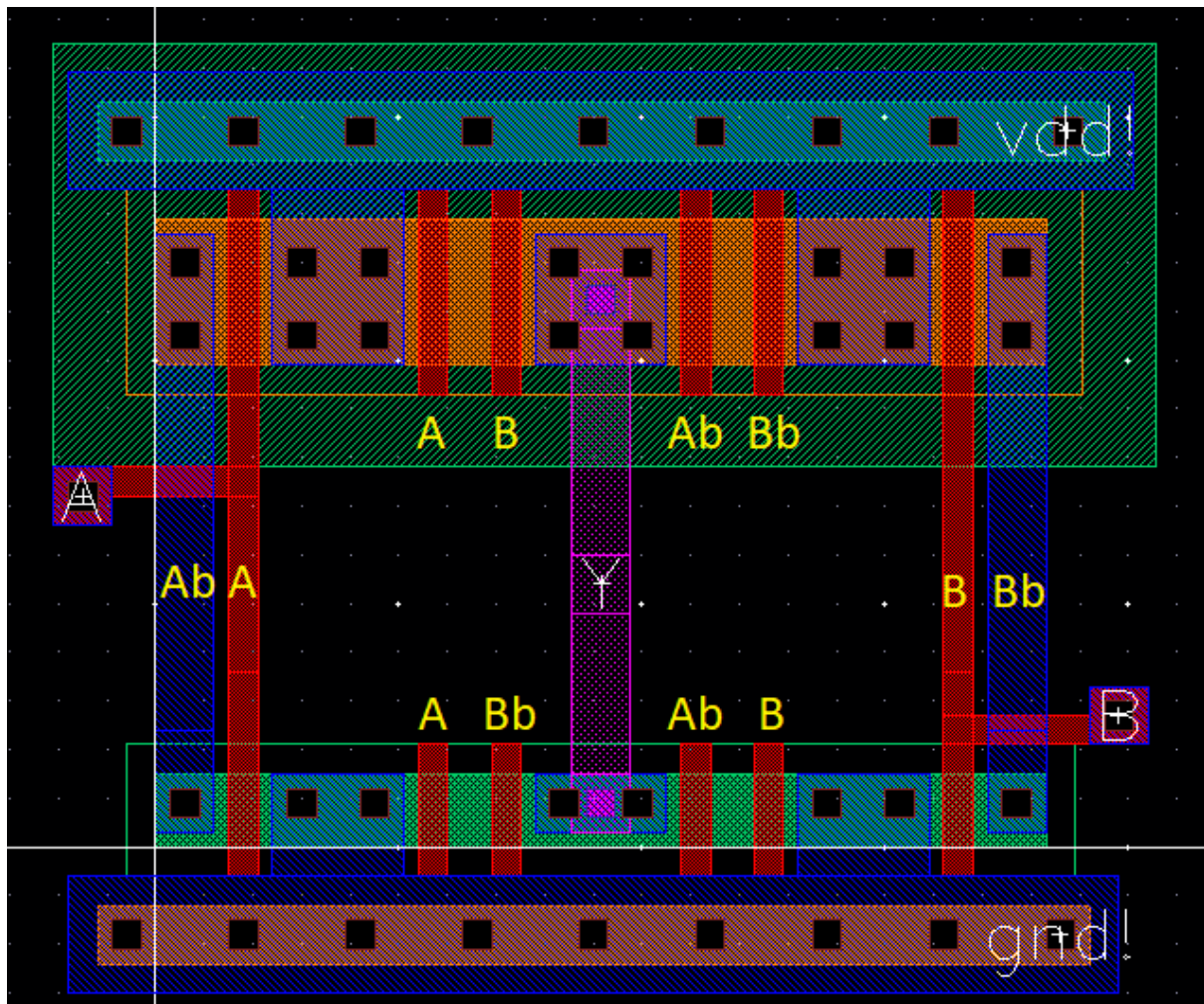


XNOR2X1 Symbol



We start working on the XNOR2X1 layout from the previously saved state.

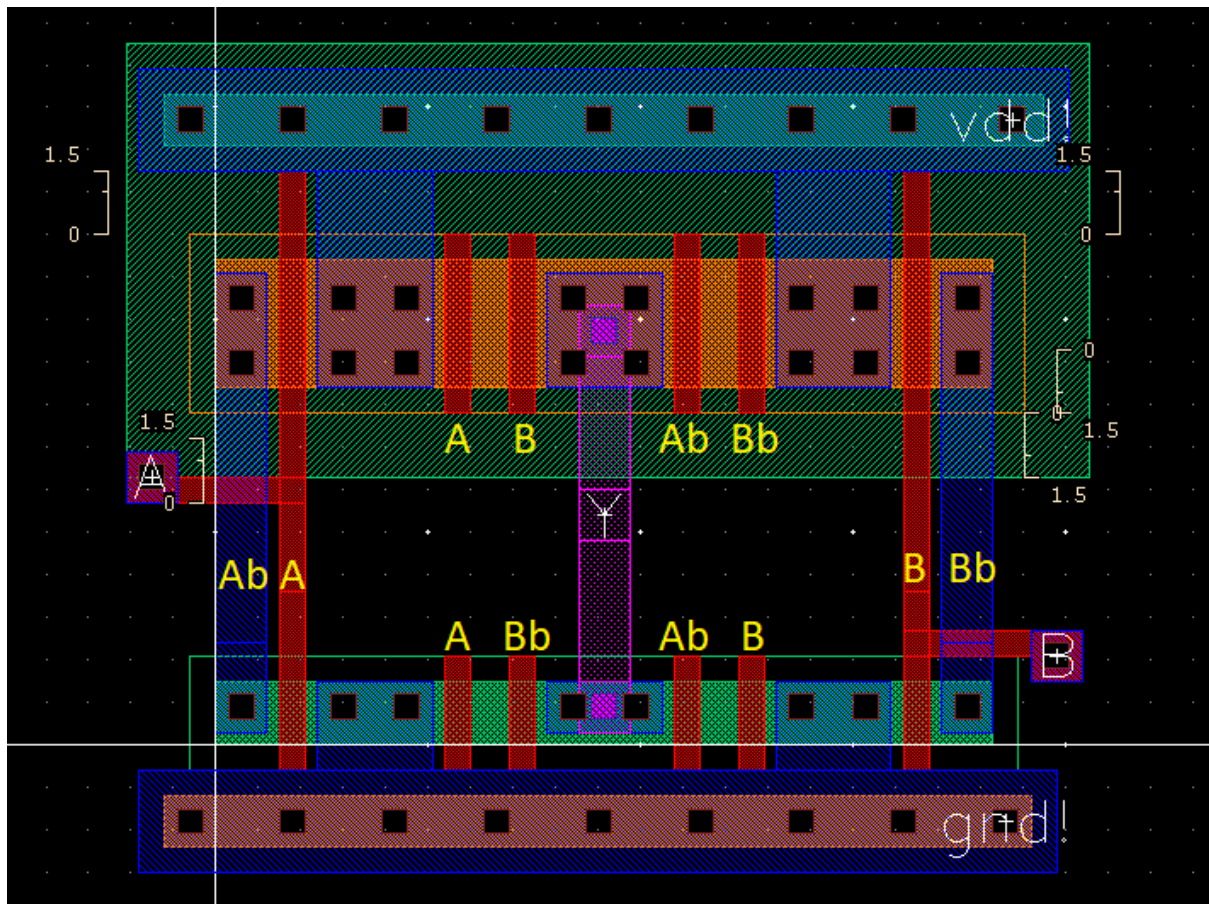
Firstly, we add the vdd! and gnd! inputOutput pins.



We will push the entire pmos transistor area down, however, keep the pactive and pselect areas unchanged. This will keep the pmos transistor sizes the same, although it will give us routing elbow room from the top, as we will see in a second. We will have to extend the nwell area towards the bottom by  $1.5\mu\text{m}$  ( $5\lambda$ ). However, the vdd! rail will stay unchanged. We will also have to push the A input pin down by  $1.5\mu\text{m}$  ( $5\lambda$ ) making sure that the vdd! connections are not broken. They should be connecting to the vdd! rail, although the vdd! rail won't move.

We make sure the design is correct up to this point by checking the DRC.

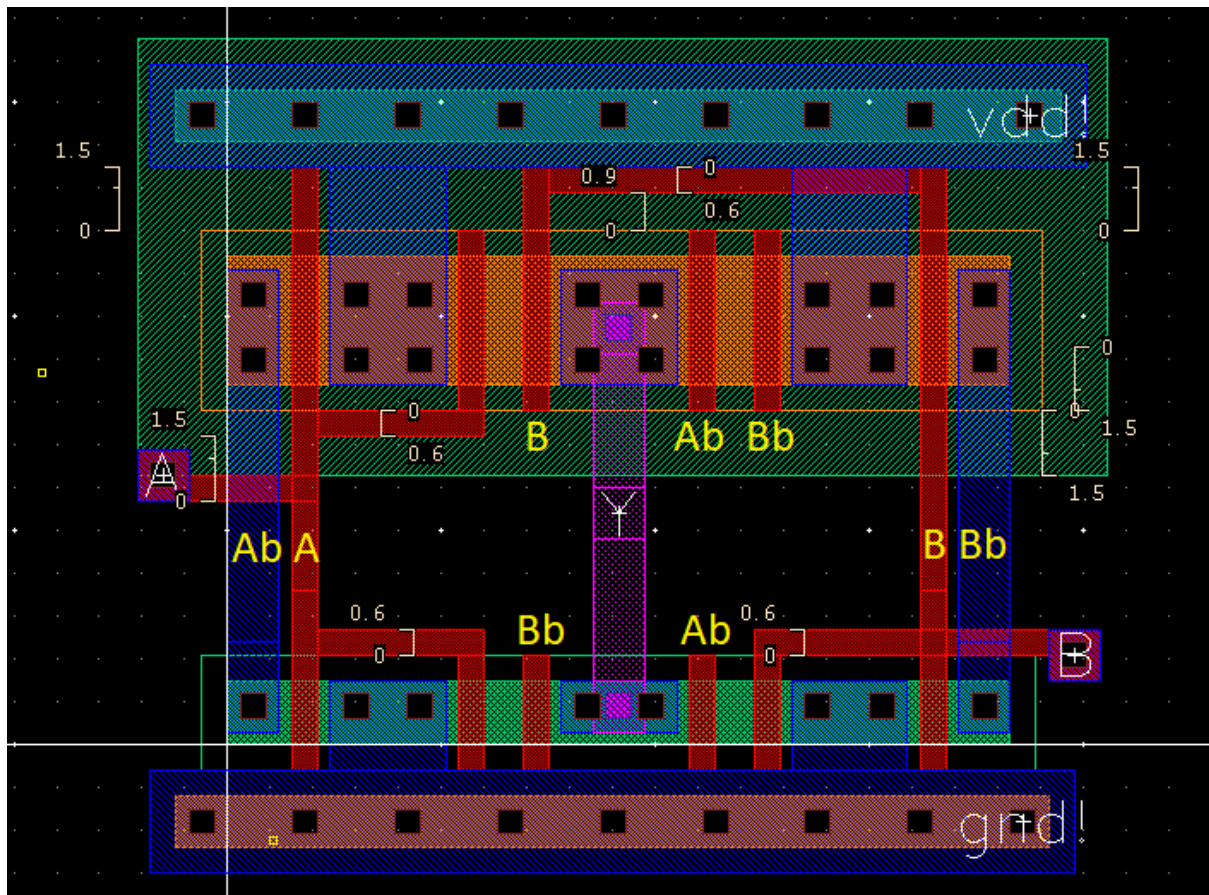
```
DRC started.....Sun Dec 1 02:55:45 2019
completed ....Sun Dec 1 02:55:45 2019
CPU TIME = 00:00:00 TOTAL TIME = 00:00:00
***** Summary of rule violations for cell "XNOR2X1 layout" *****
Total errors found: 0
```



Much like XOR, there are 3 super-easy connections: Two A inputs (poly) on the far left can connect to the poly line of the A input on their left. The B input poly on the bottom right can connect to the B poly line on its right. We make these connections first with as short of a poly like as possible (to avoid additional parasitic capacitance).

After making the 3 obvious connections, we have 5 more connections left. This time, we will route B from above and connect it to the B poly line on the right. Poly lines cannot run above active lines, as they will form a transistor otherwise. So, we cannot leap from the poly over the vdd! rail. We have to route it away from the nactive area that is used in making the nwell taps.





The reason to push the pmos transistor area by  $1.5\mu\text{m}$  ( $5\lambda$ )? The reason is simple; we can see it marked below with the rulers. When routing the B line from above the pmos transistors, it has to be  $3\lambda$  away from the other poly lines. Additionally, the poly line itself is  $2\lambda$ . Add the two up, you get  $5\lambda$ .

We could have saved a  $\lambda$  and ran the poly line overlapping with the metal1 line of the vdd! rail. This is bad for third reasons: first, metal1-poly overlap causes a lot of parasitic capacitance; second, getting so far close to the nactive area, used to make the nwell taps, causes parasitic transistors. Although vdd! rail has a constant voltage, the poly line will cause crosstalk on the vdd! rail and will result in unnecessary glitches; third, we are not short of space. Even with the  $1.5\mu\text{m}$  less vertical space, we still have plenty of room to route everything without a problem.

We have 4 more lines to route. We will run Ab poly line to each other vertically and run it left. We will put a contact on both Bb poly lines and run it to the right as a metal1 line.

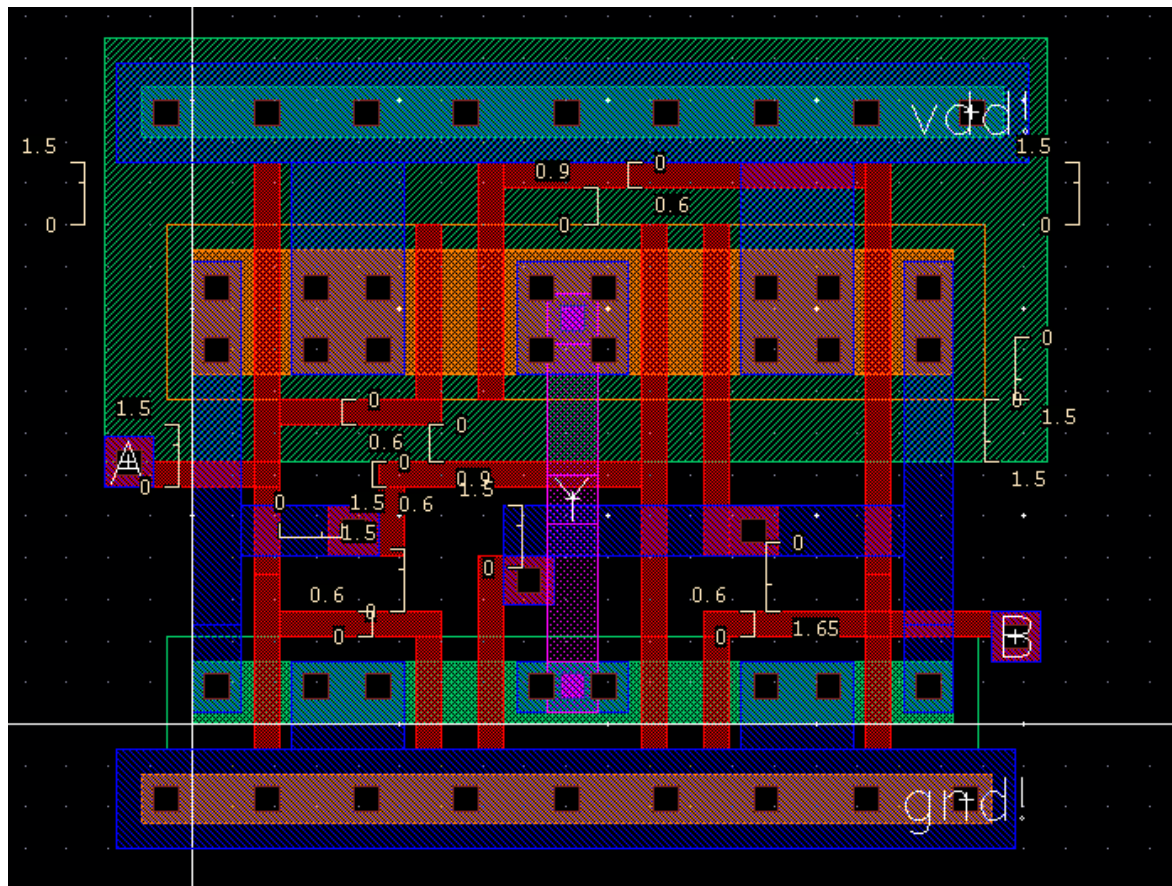
We connected the Ab lines together, so, this left 3 lines to route.

Ab line was in poly. So, we ran it from under metal2 and put a metal1-poly pad in there and ran it to the left, where the metal1 line of Ab resides.

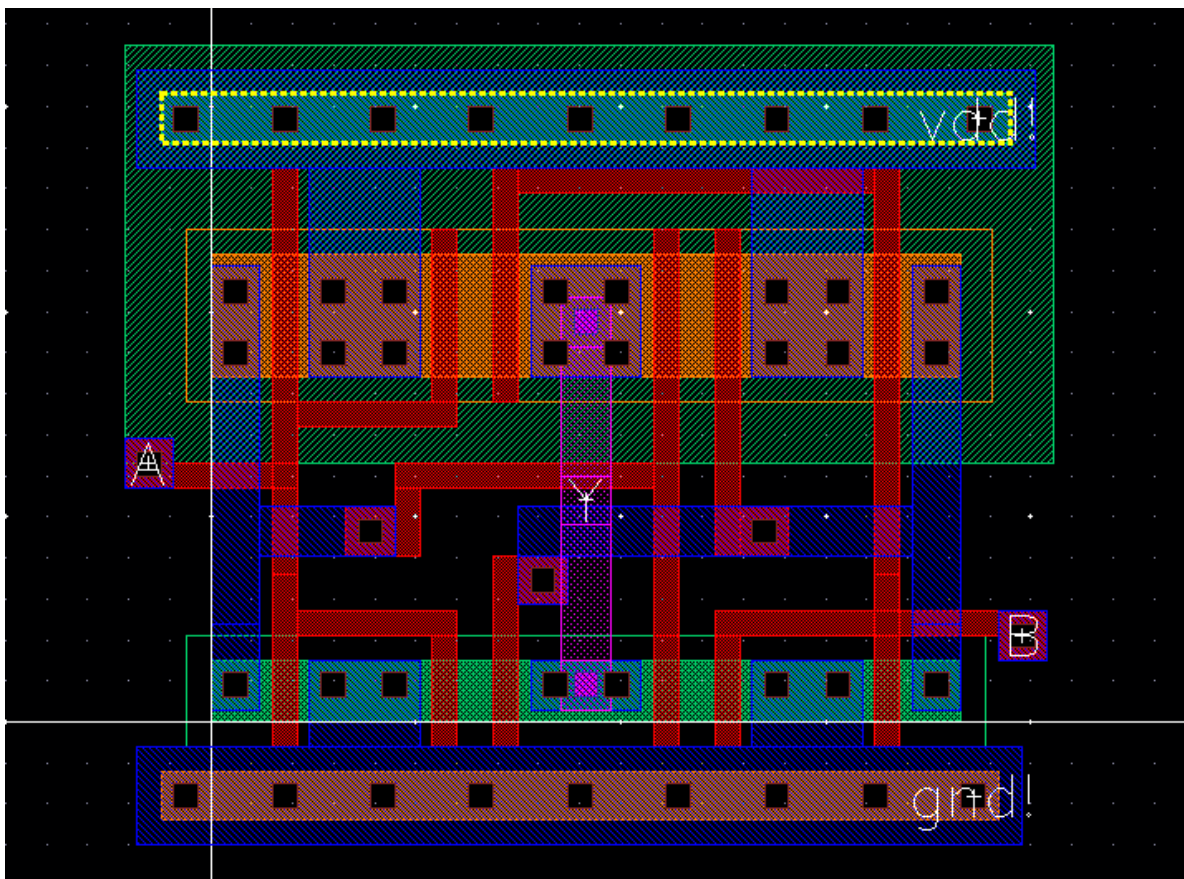
This left two Bb lines to route. The one on the right side was easy. We had to place a metal1-poly pad ( $1.2\mu\text{m} \times 1.2\mu\text{m}$ ) and run it to the right, where the Bb metal1 line resides.

The Bb on the left was a little more interesting. We had to place the metal1-poly pad ( $1.2\mu\text{m} \times 1.2\mu\text{m}$ ) on the left side (slightly under the metal2) and run it to the right in metal1.





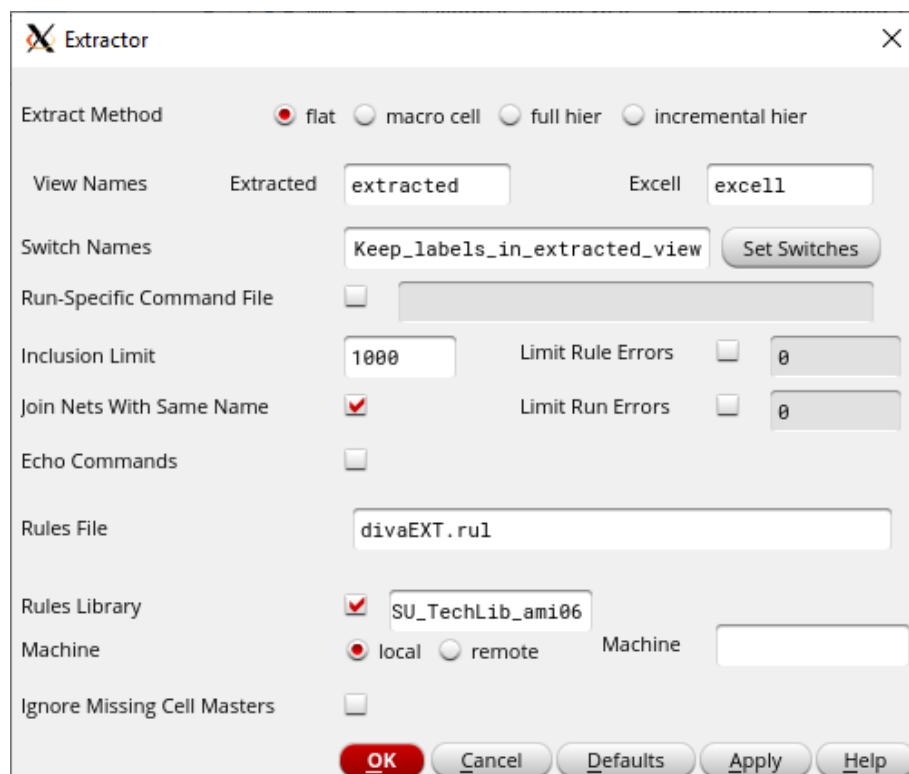
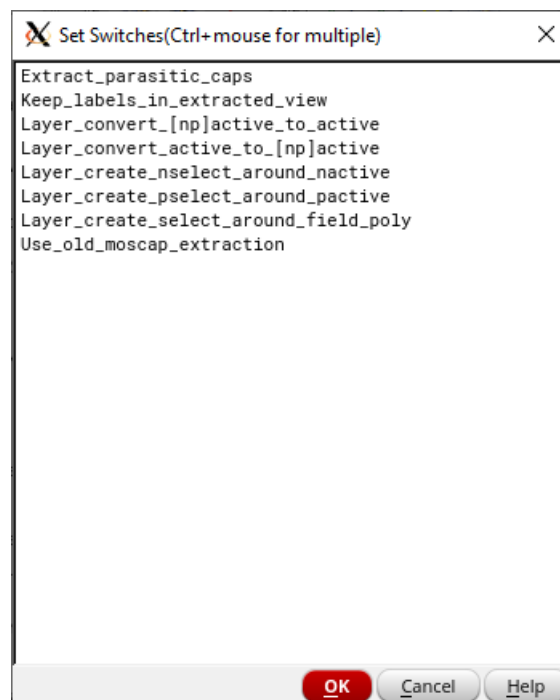
This is what the final layout looks like:



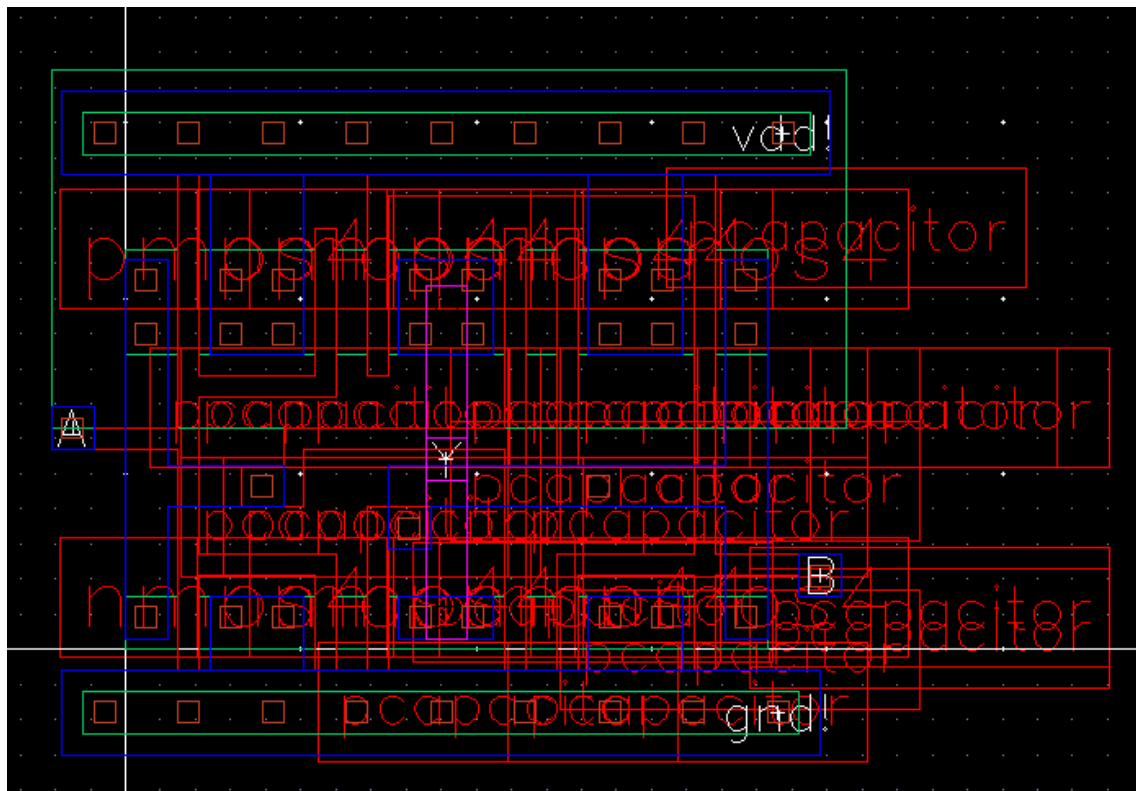
We run DRC and it returns no errors.

```
DRC started.....Sun Dec  1 03:23:46 2019
completed ....Sun Dec  1 03:23:46 2019
CPU TIME = 00:00:00  TOTAL TIME = 00:00:00
***** Summary of rule violations for cell "XNOR2X1 layout" *****
Total errors found: 0
```

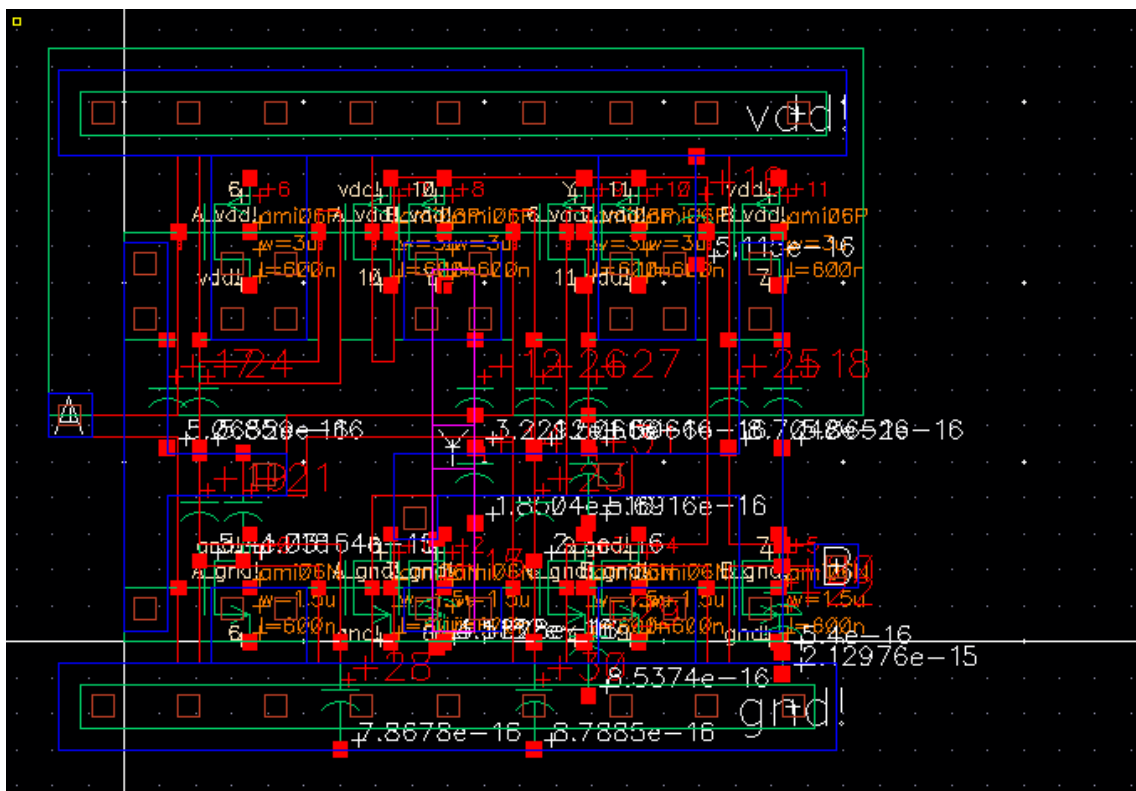
Therefore, we create the extracted view through Verify -> Extract.



saving rep LAB10/XNOR2X1/extracted  
Getting layout proper bagGetting layout proper bagLoading techComp.cxt

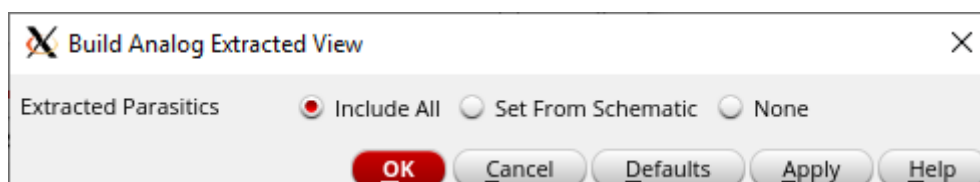
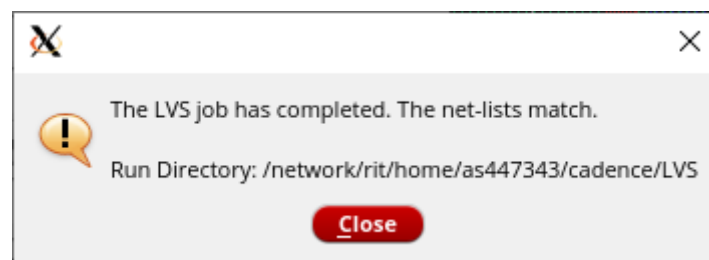
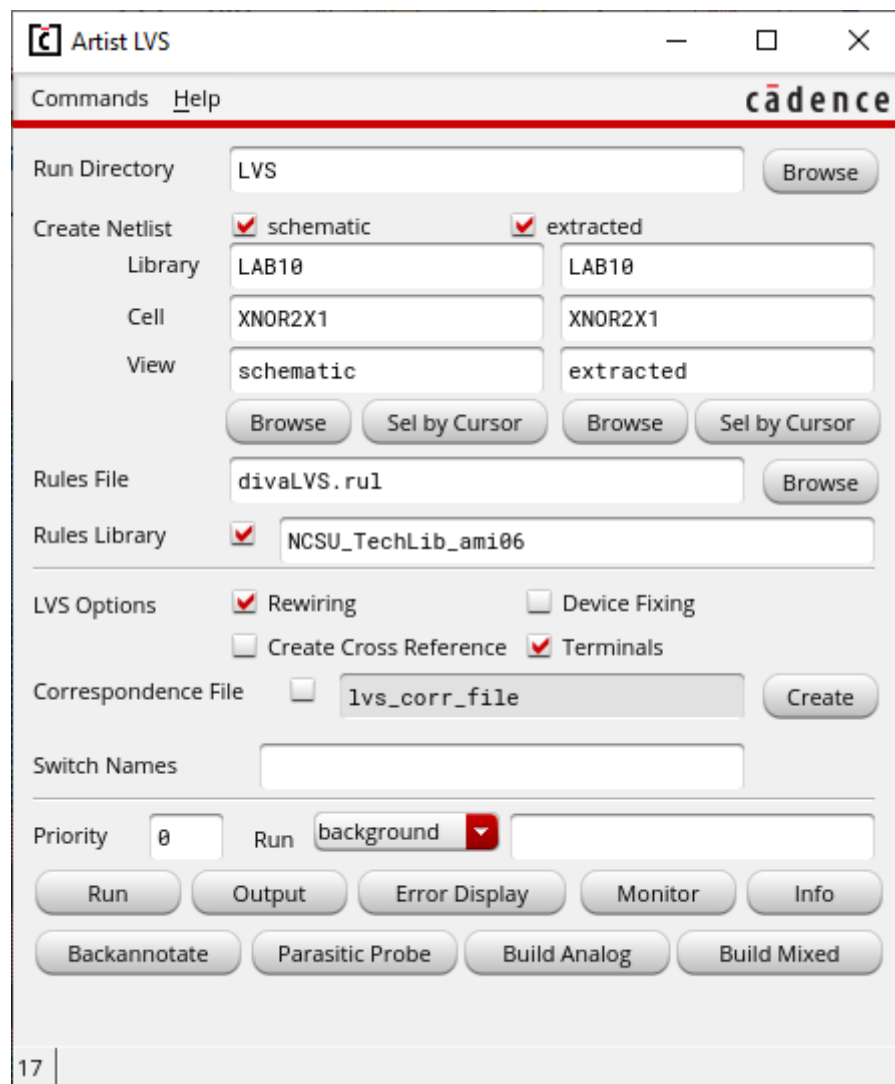


XNOR2X1 extracted view (Top level)

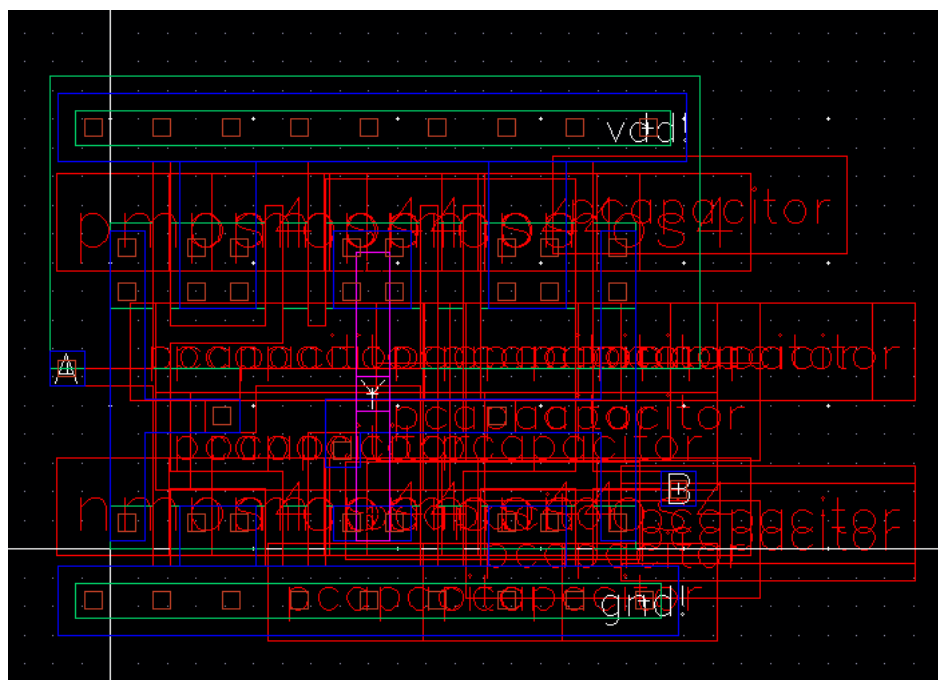


XNOR2X1 extracted view (Lower level)

Next, we do LVS of XNOR2X1 and create the analog\_extracted view.

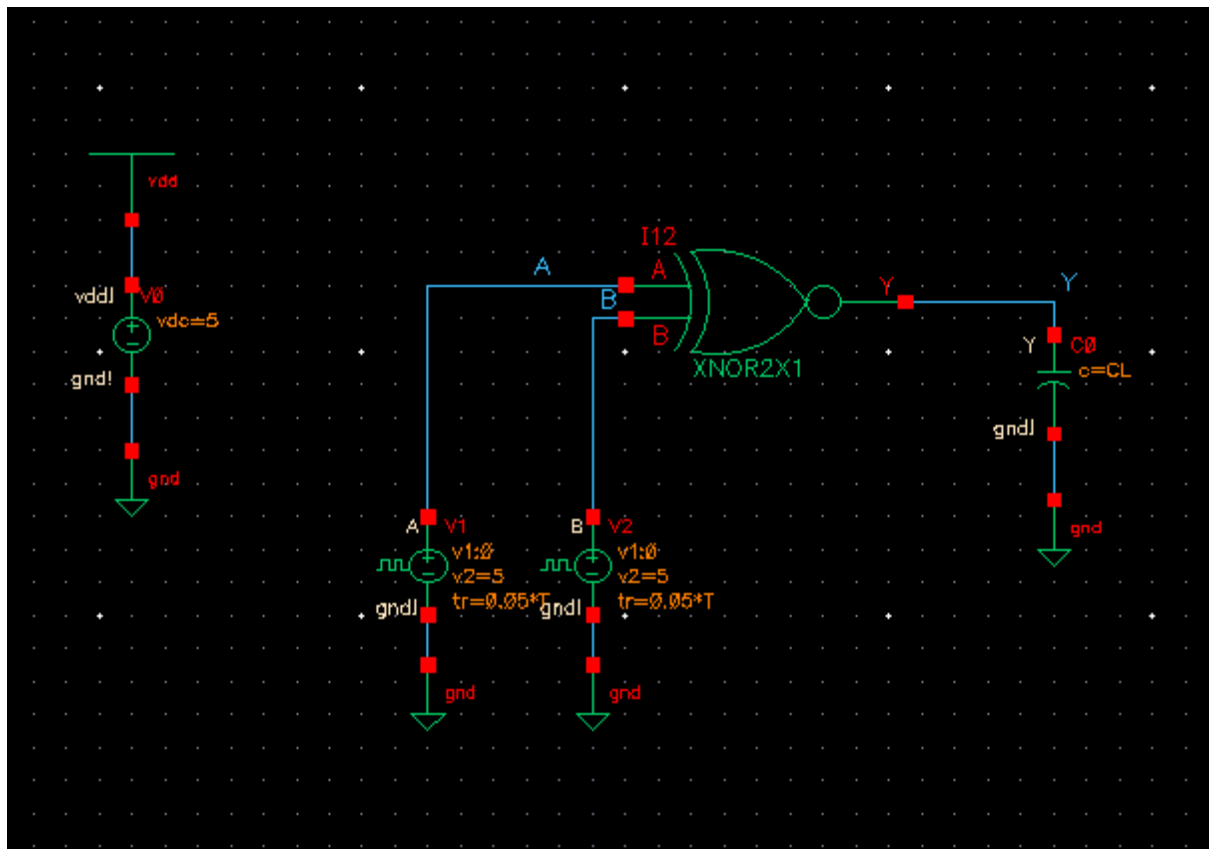




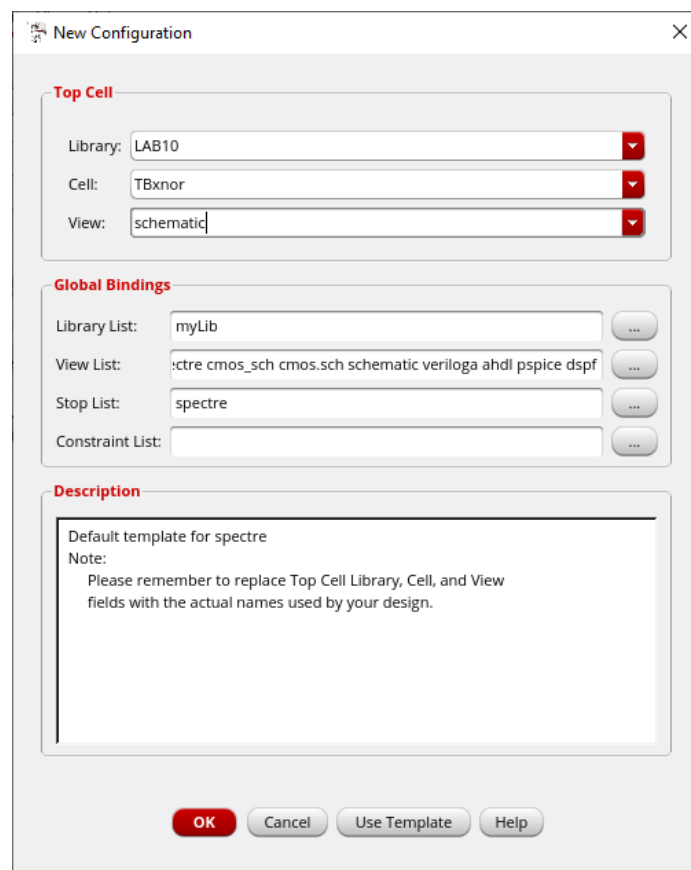


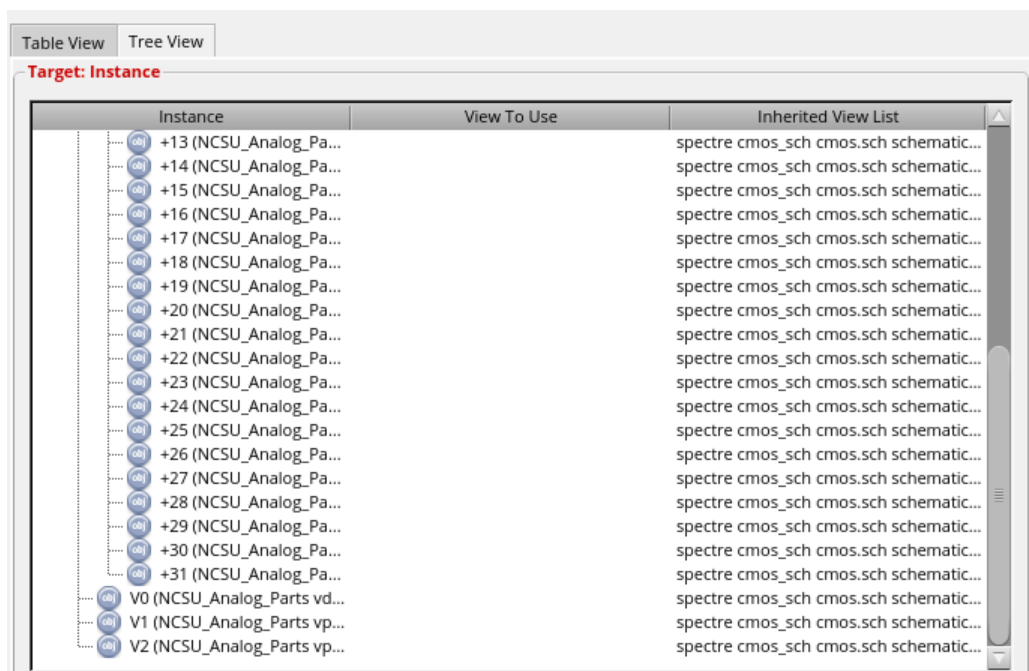
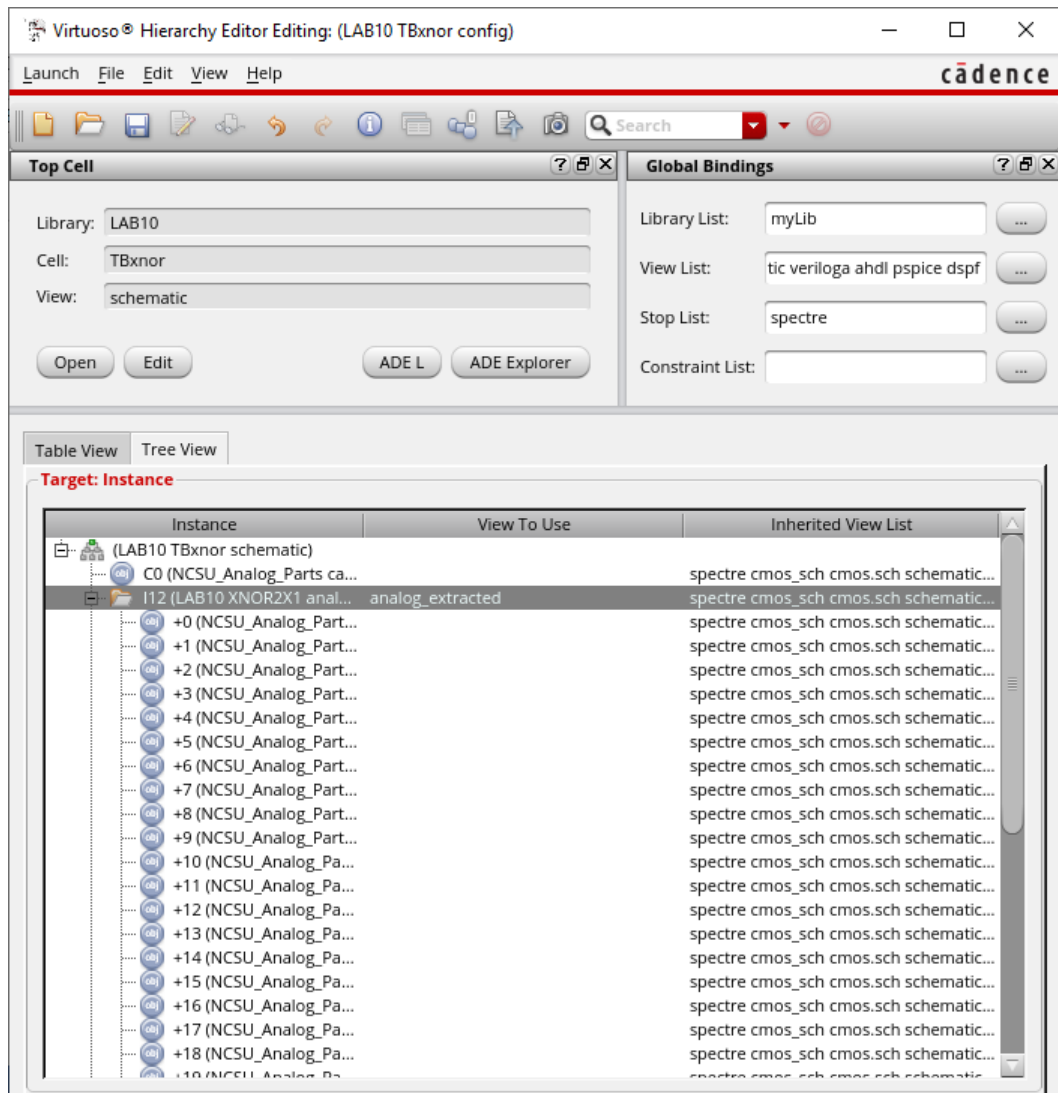
XNOR2X1 analog\_extracted view

After creating the analog\_extracted view, we create a test bench named TBxnor.

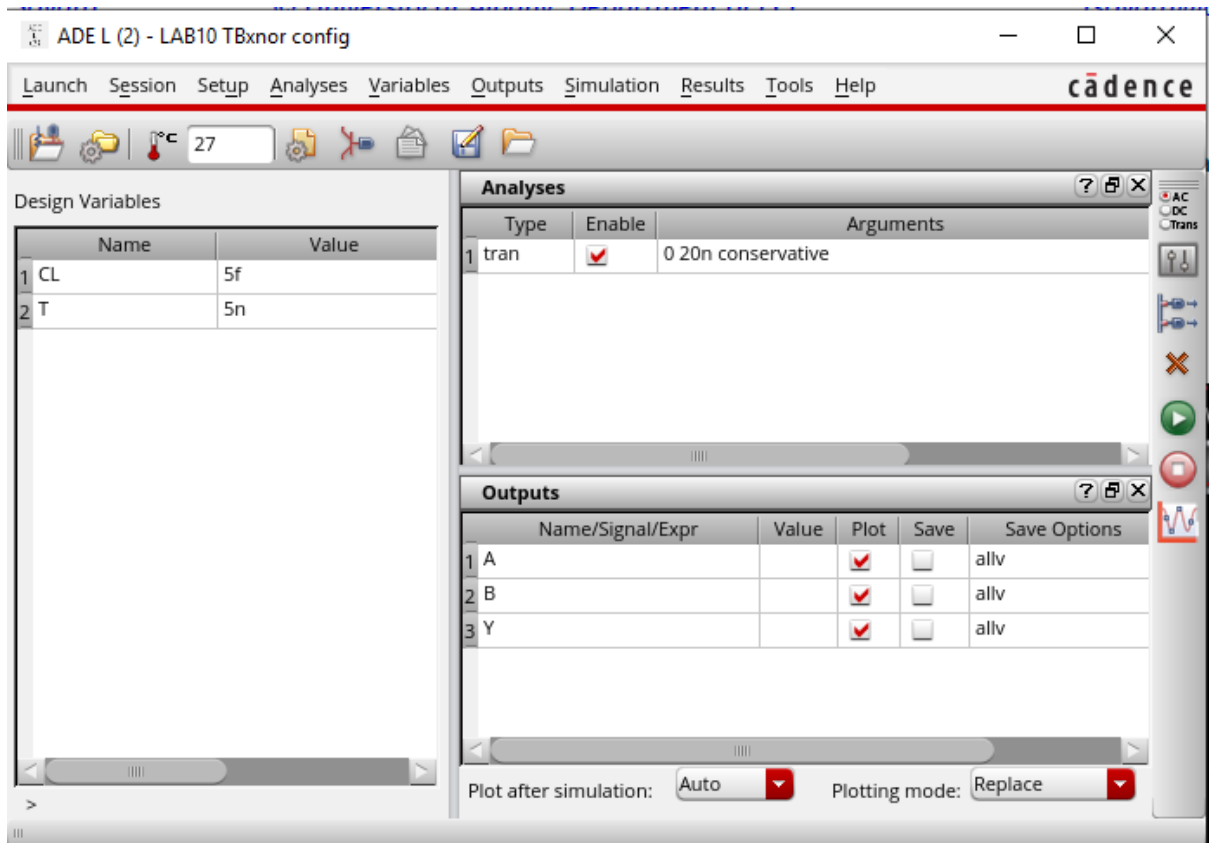


TBxnor schematic

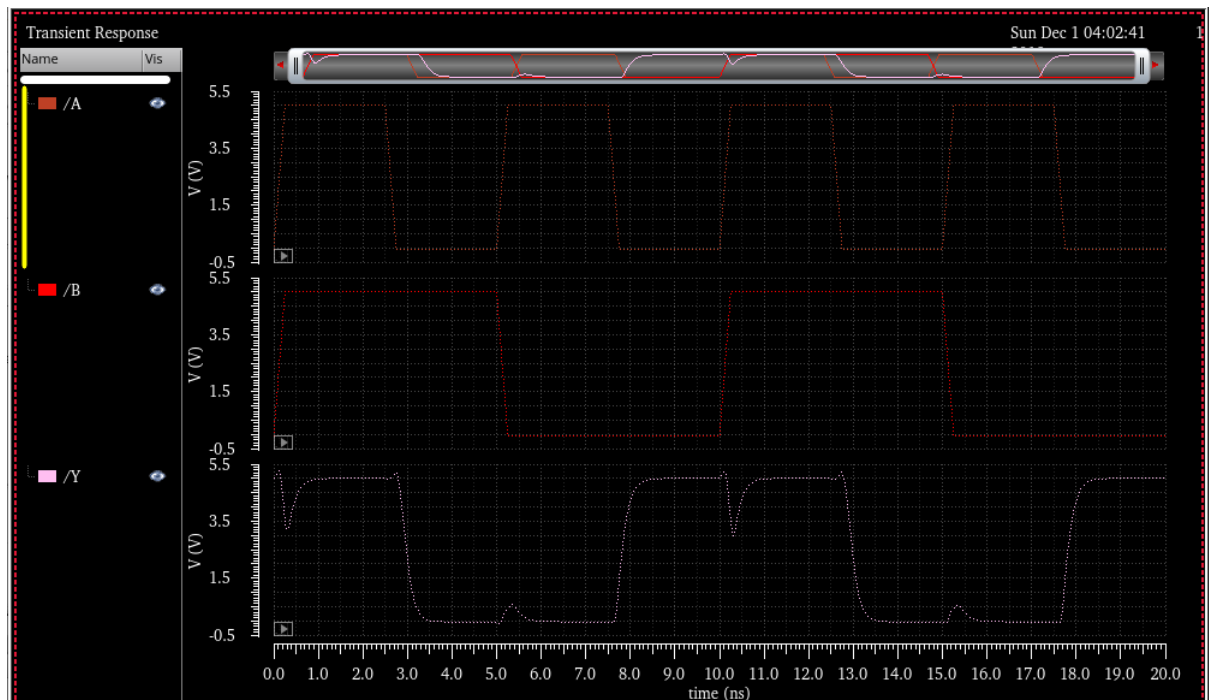




TBxnor tree view



TBxnor config ADE L



TBxnor config output waveform