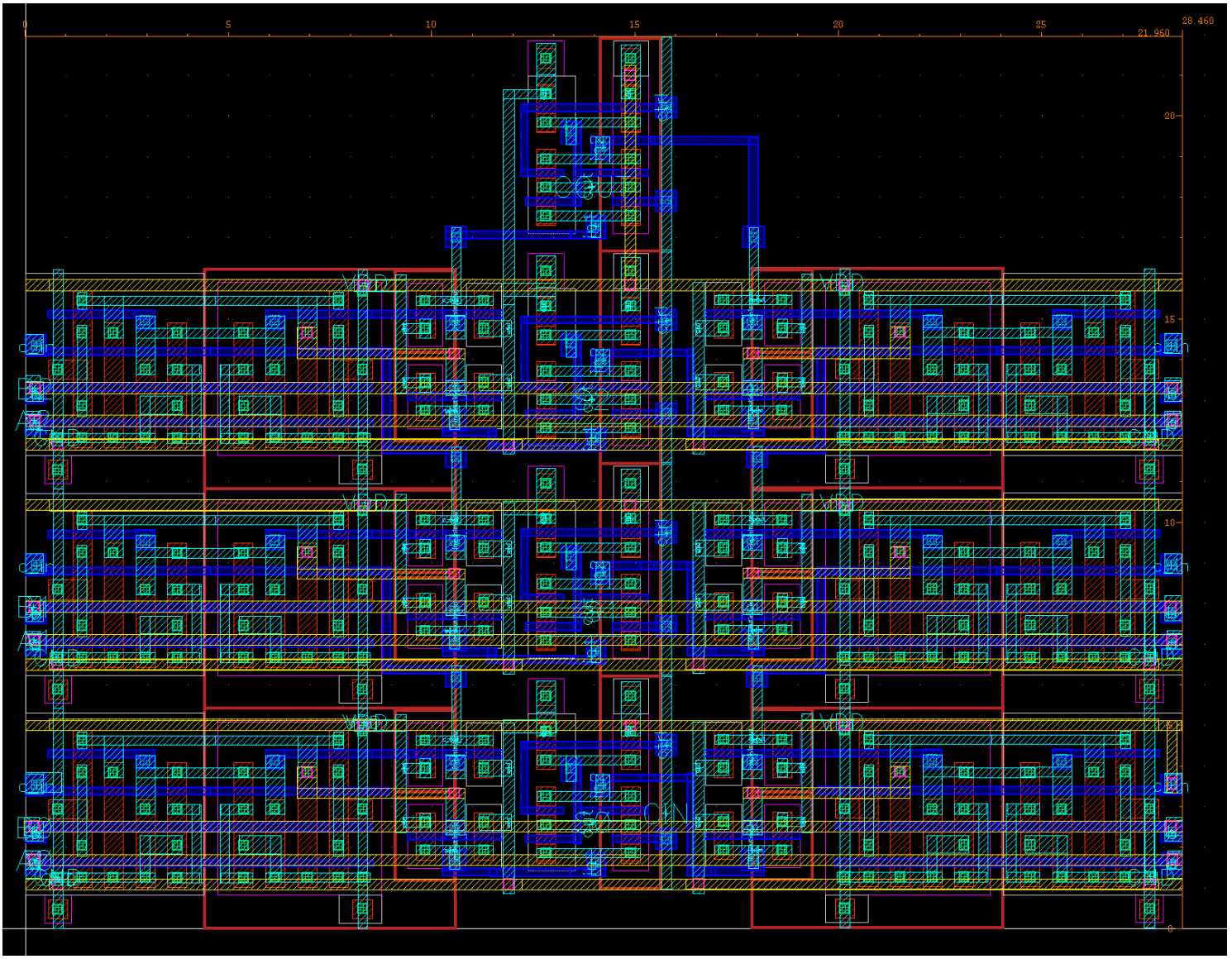
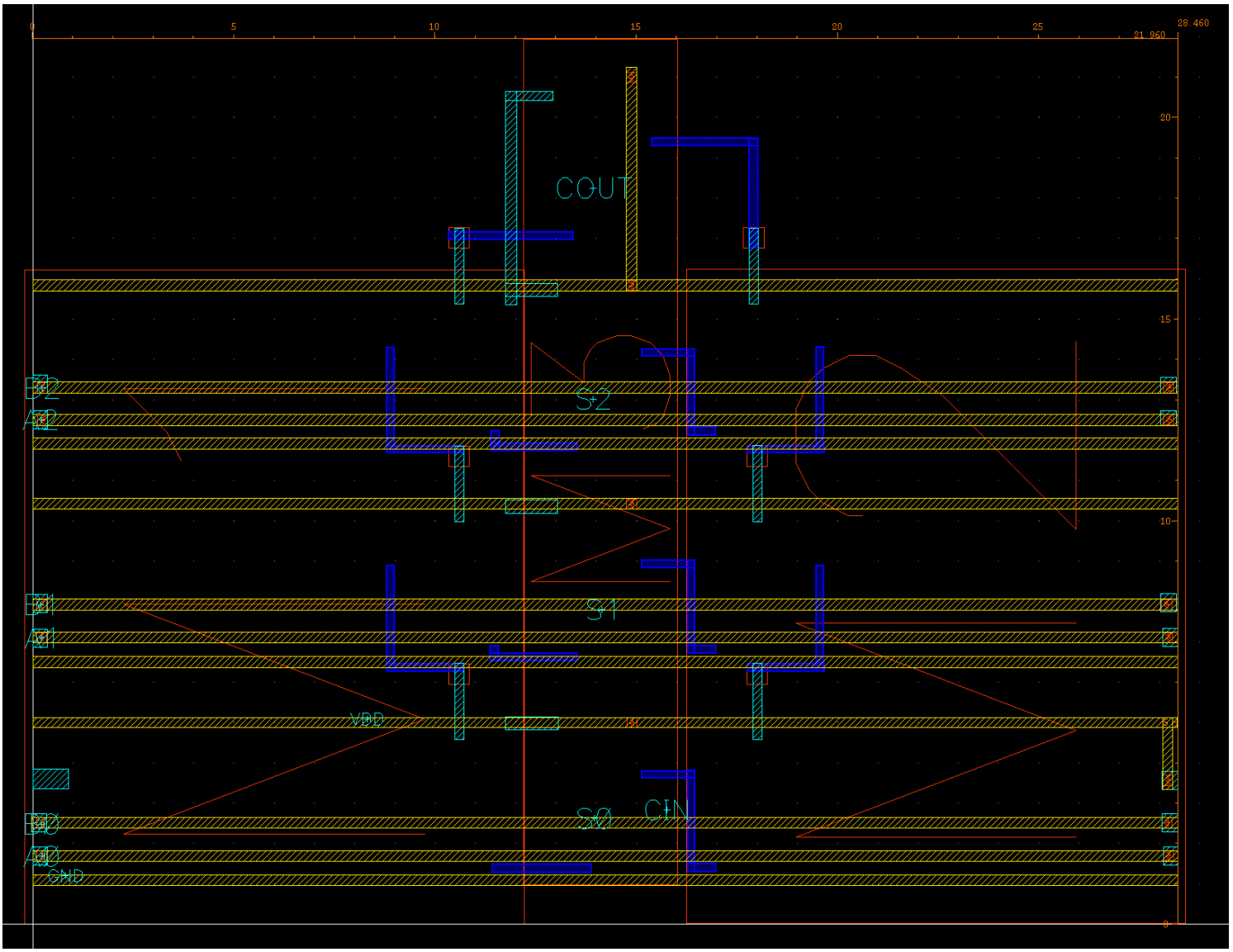
# 1. screenclips

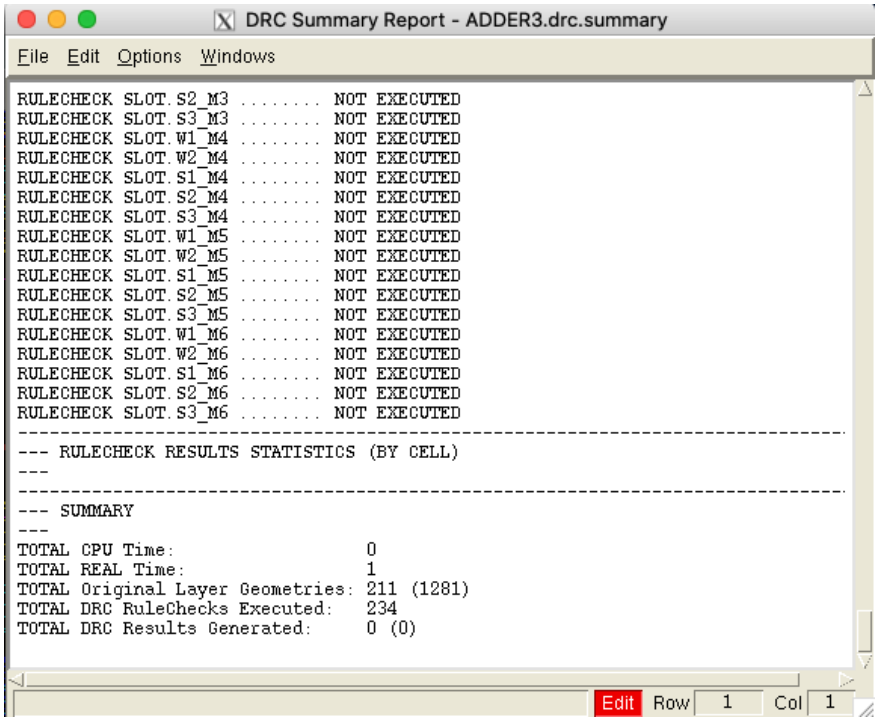
area: 624.9816
width: 28.46
height: 21.96
Level 10 cell view



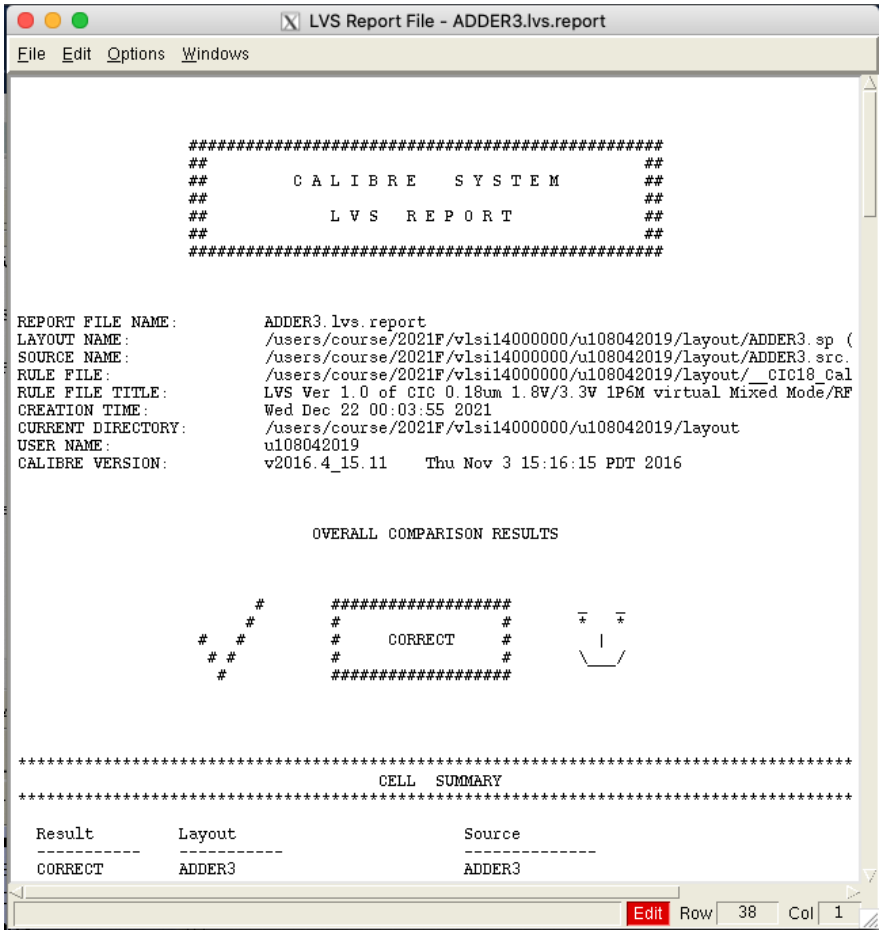
Level 0 cell view:



DRC report:




LVS report:



Schematic capture: (also reference on attached files)

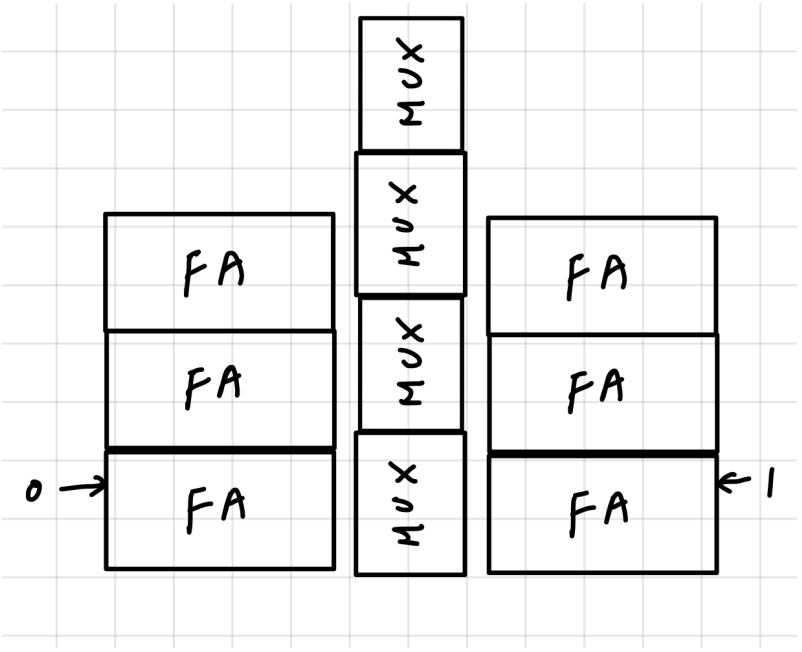
```
.subckt ADDER3 A0 A1 A2 B0 B1 B2 CIN S0 S1 S2 COUT VDD GND
x_fa0_in0 A0 B0 GND SUM0 COUT0 VDD GND FA
x_fa0_in1 A0 B0 VDD SUM0_b COUT0_b VDD GND FA
x_fa1_in0 A1 B1 COUT0 SUM1 COUT1 VDD GND FA
x_fa1_in1 A1 B1 COUT0_b SUM1_b COUT1_b VDD GND FA
x_fa2_in0 A2 B2 COUT1 SUM2 COUT2 VDD GND FA
x_fa2_in1 A2 B2 COUT1_b SUM2_b COUT2_b VDD GND FA
x_mux0 SUM0 SUM0_b CIN S0 VDD GND MUX21
x_mux1 SUM1 SUM1_b CIN S1 VDD GND MUX21
x_mux2 SUM2 SUM2_b CIN S2 VDD GND MUX21
x_mux3 COUT2 COUT2_b CIN COUT VDD GND MUX21
.ends

$ ======================================Y
.subckt MUX21 in1 in2 ctrl out VDD GND
$inv
mp1 s_bar ctrl VDD VDD P_18 w=0.47u l=0.18u
mn1 s_bar ctrl GND GND N_18 w=0.47u l=0.18u

$TG
mp2 out s_bar in2 VDD P_18 w=0.47u l=0.18u
mp3 out ctrl in1 VDD P_18 w=0.47u l=0.18u
mn2 out ctrl in2 GND N_18 w=0.47u l=0.18u
mn3 out s_bar in1 GND N_18 w=0.47u l=0.18u

.ends
$ ======================================Y
.subckt FA a b c_in sum c_out VDD GND
$cout_ p
mp1_1 p01 a VDD VDD P_18 w=0.47u l=0.18u
mp1_2 p01 b VDD VDD P_18 w=0.47u l=0.18u
mp1_3 cout_ c_in p01 VDD P_18 w=0.47u l=0.18u

mp2_1 p02 a VDD VDD P_18 w=0.47u l=0.18u
mp2_2 cout_ b p02 VDD P_18 w=0.47u l=0.18u

$cout_ n
mn1_1 cout_ c_in n01 GND N_18 w=0.47u l=0.18u
mn1_2 n01 a GND GND N_18 w=0.47u l=0.18u
mn1_3 n01 b GND GND N_18 w=0.47u l=0.18u

mn2_1 cout_ b n02 GND N_18 w=0.47u l=0.18u
mn2_2 n02 a GND GND N_18 w=0.47u l=0.18u

$s_ p
mp3_1 p03 a VDD VDD P_18 w=0.47u l=0.18u
mp3_2 p03 b VDD VDD P_18 w=0.47u l=0.18u
mp3_3 p03 c_in VDD VDD P_18 w=0.47u l=0.18u
mp3_4 s_ cout_ p03 VDD P_18 w=0.47u l=0.18u

mp4_1 p04 a VDD VDD P_18 w=0.47u l=0.18u
mp4_2 p05 b p04 VDD P_18 w=0.47u l=0.18u
mp4_3 s_ c_in p05 VDD P_18 w=0.47u l=0.18u

$s_ n
mn3_1 s_ cout_ n03 GND N_18 w=0.47u l=0.18u
mn3_2 n03 a GND GND N_18 w=0.47u l=0.18u
mn3_3 n03 b GND GND N_18 w=0.47u l=0.18u
mn3_4 n03 c_in GND GND N_18 w=0.47u l=0.18u

mn4_1 s_ c_in n05 GND N_18 w=0.47u l=0.18u
mn4_2 n05 b n04 GND N_18 w=0.47u l=0.18u
mn4_3 n04 a GND GND N_18 w=0.47u l=0.18u

x_s_inv s_ sum VDD GND INV
x_p_inv cout_ c_out VDD GND INV
.ends

.subckt INV inv_in inv_out vdd gnd
mp inv_out inv_in vdd vdd P_18 w=0.47u l=0.18u
mn inv_out inv_in gnd gnd N_18 w=0.47u l=0.18u
.ends
~
```

# 2. approach

## (1) Think about whole design arrangement

In order to minimize area, we can

- align pairs of FA that have the same inputs, so inputs can run through the whole design by metal2.
- align 2 groups of 3 FA side by side vertically. -> We can use array method
- align 3 MUX21 side by side vertically. -> also by array method




## (2) make own cells

(a) FA(full adder) cell with Mirror Adder (MA) design

(i) subcell: MA_MOS ( details in (2)(a) )

(ii) INV

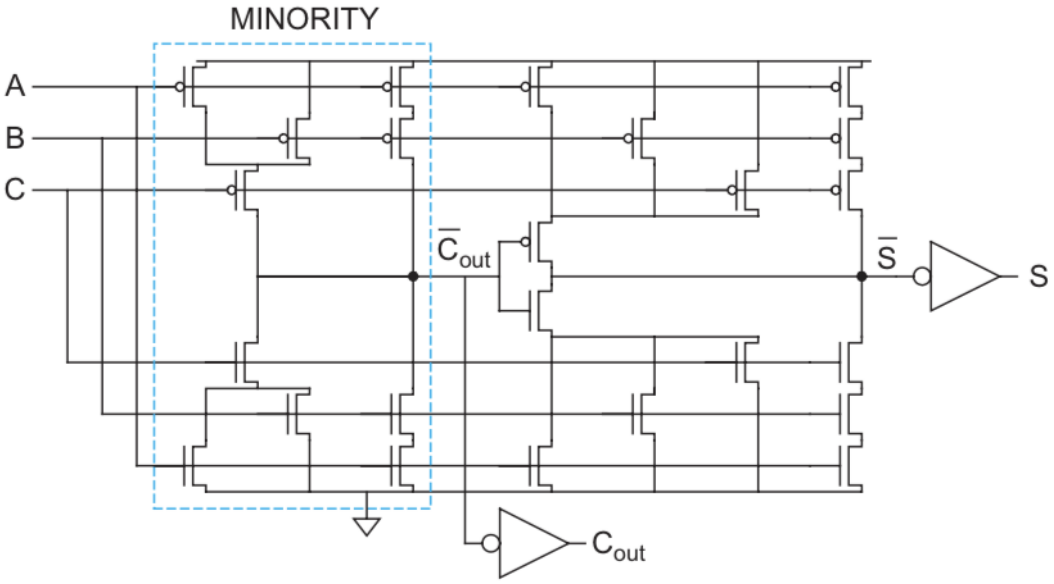
(b) MUX21 cell ( details in (3) )

(c) cell of repeated objects for layout convenience

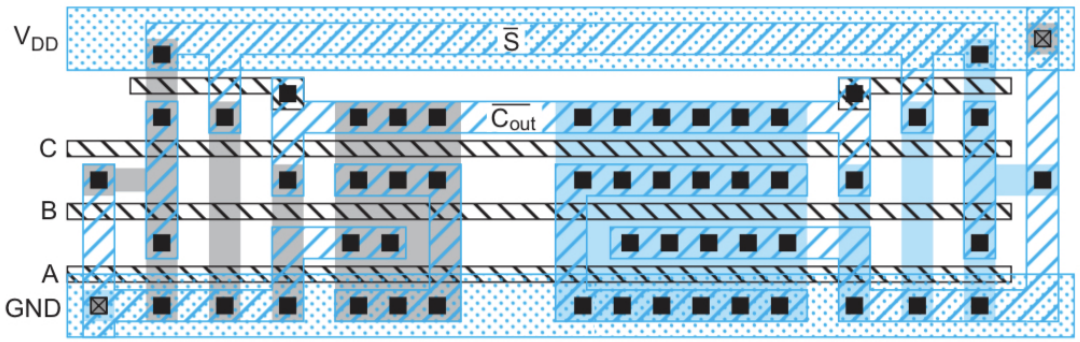
- CNT(contact only)
- CNT_DIF(contact with diff)
- CNT_POLY(contact with poly)
- nwell_pickup(pickup for PMOS)
- p_pickup(pickup for NMOS)

(3) FA: by using mirror adder design

2* MA_MOS cell = adder with inverted outputs of FA
2* MA_MOS cell + 2* INV cell = ideal FA that we want




<mirror FA schematic, which can be made by 2 MA_MOS cell+ 2 INV>
referenced from book **CMOS VLSI Design: A Circuits and Systems Perspective, p432**

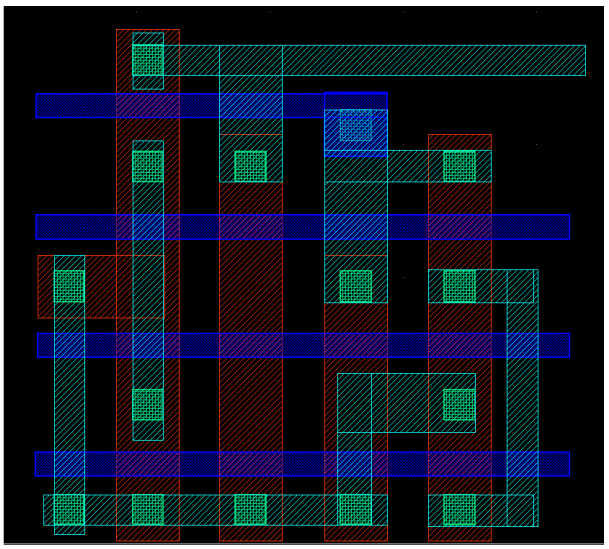



<2 MA_MOS cell layout, which outputs sum_bar and cout_bar>
referenced from book **CMOS VLSI Design: A Circuits and Systems Perspective, p433**

(a) make MA_MOS cell

Since in a mirror adder, NMOS and PMOS are horizontally symmetric, I make a MA_MOS cell with one side, and later instance another MA_MOS horizontally flipped.

Note that MA_MOS only contains contact, diff, poly, metal1. Distance between two poly can't be too close, since we haven't put poly-contact of each inputs.

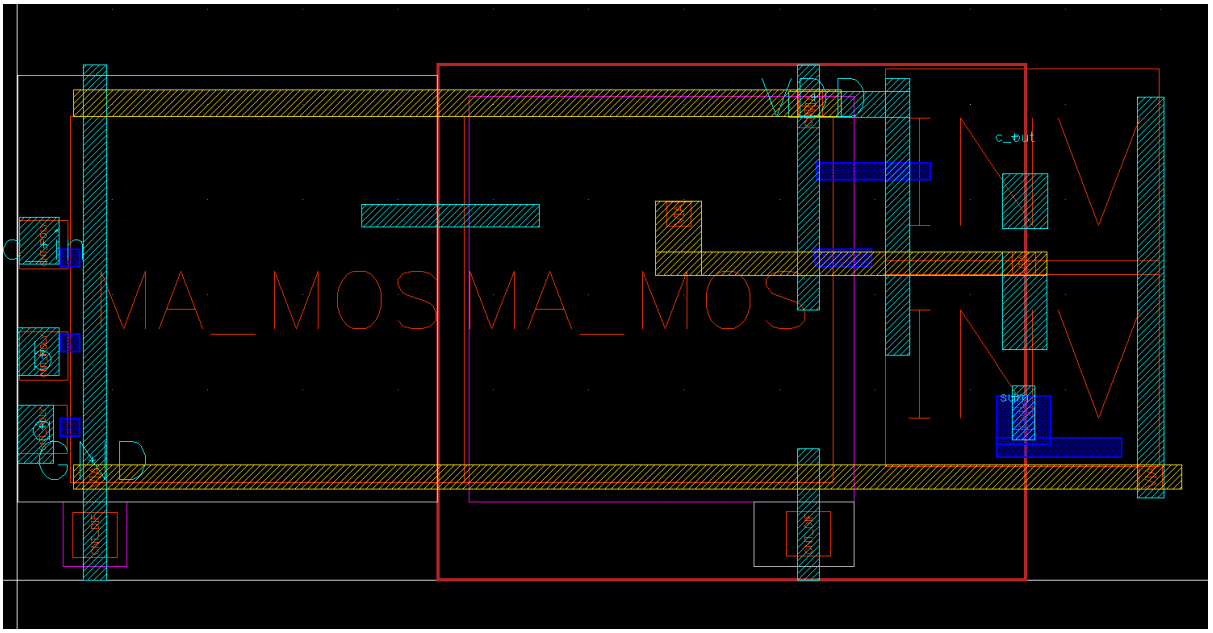


<MA_MOS cell>

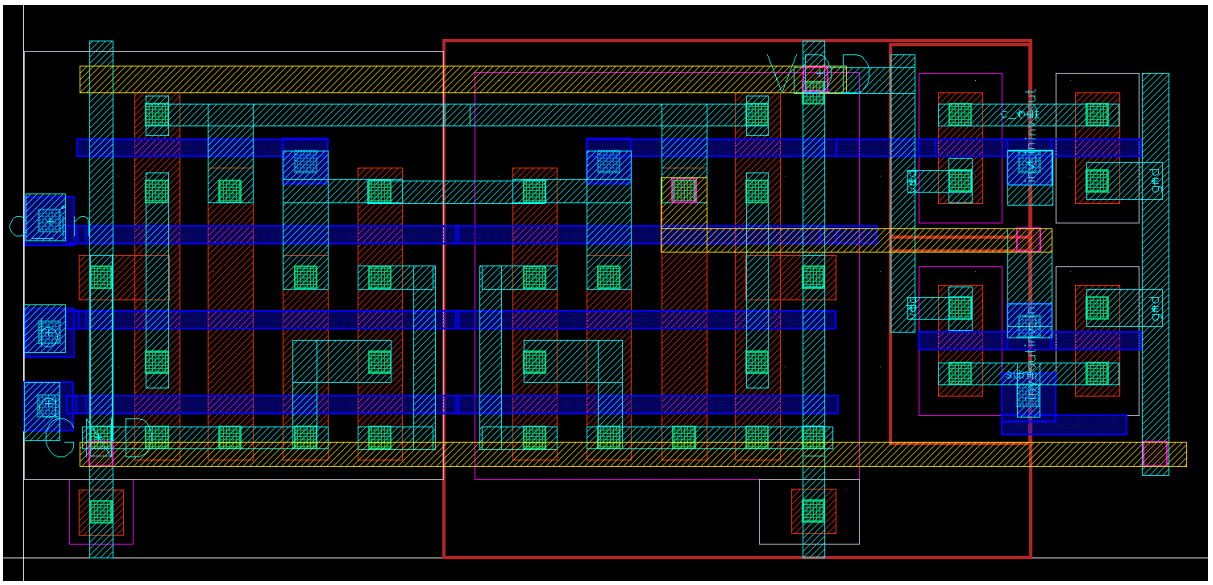
(b) Add objects on top
Add N+, P+, NWELL, pickups to two MA_MOS cell, making them real NMOS and PMOS.
Add poly-contact to input. Connect source with metal2. Then we have a part of FA that has output `sum_bar` and `cout_bar`. Hence we need to add two inverter to make it FA.

(c) Instance two INV to invert those two signals mentioned in (b). Connect signals by poly and metal2. Align `cout_bar` with INV that would invert this signal.

(d) Source : Connect sources with metal2 horizontally . Connect some sources by extending metal1 vertically.
Basically, a FA made with mirror adder design is done by this time.



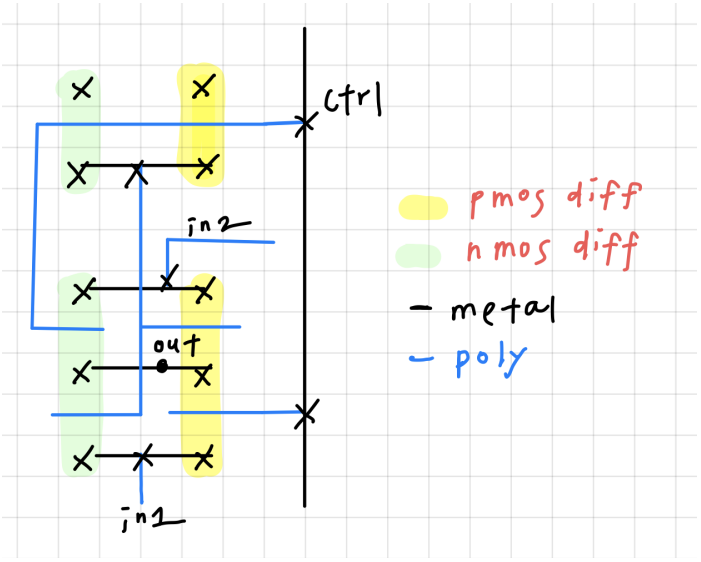

<FA level0>

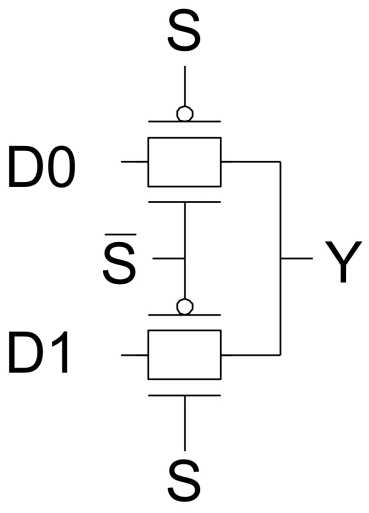



<FA level10>

(4) MUX21 : use inverter+transmission gate to make MUX21

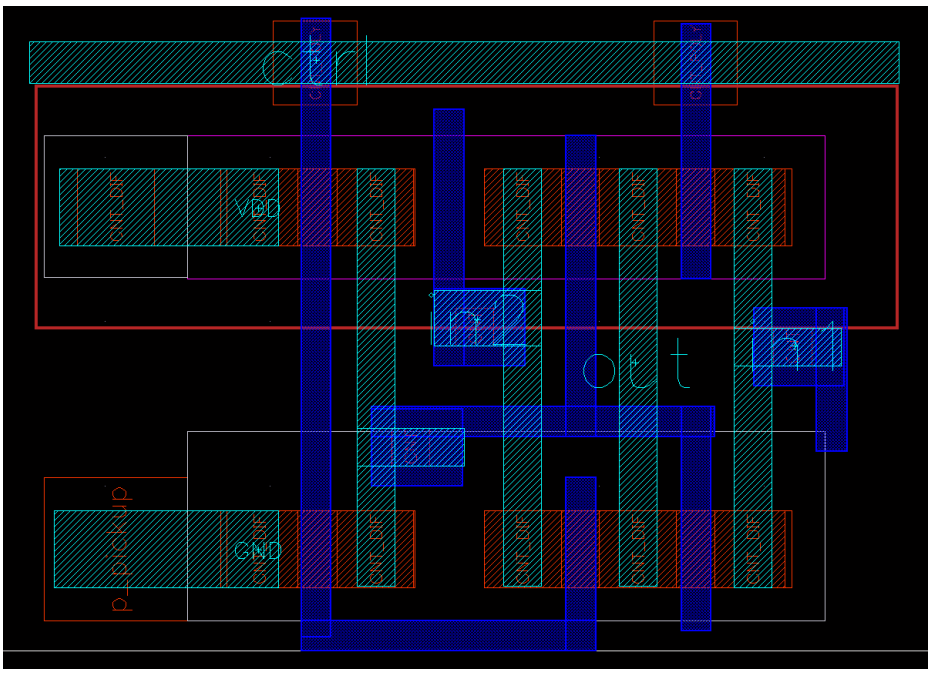
(a) design MUX21 by TG mentioned in lecture, which would need 2+4=6 transistor



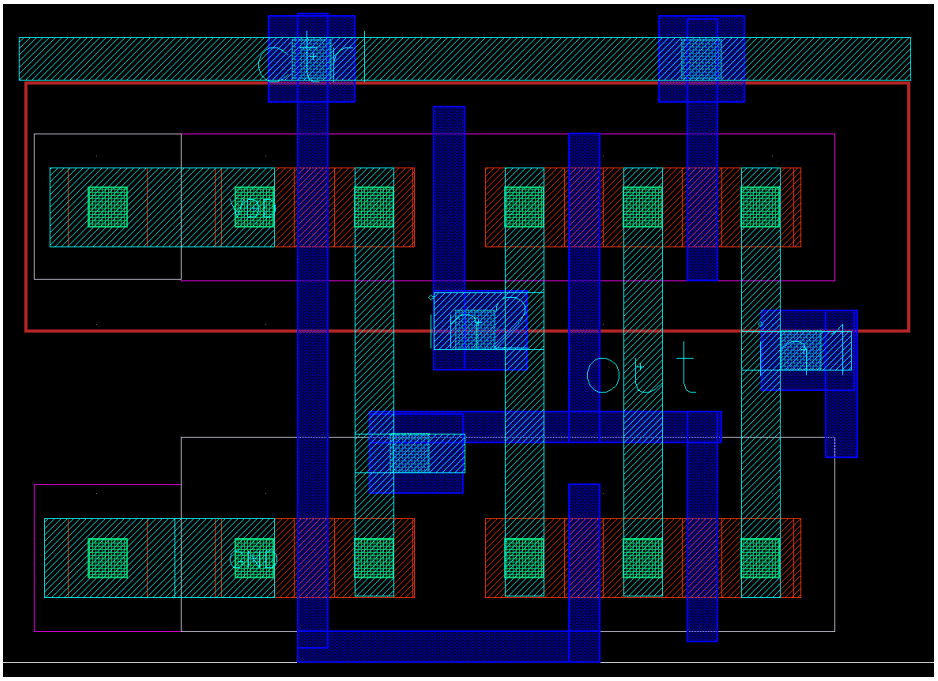




(b) Connect 2 gate input `ctrl` with metal1 and poly-contact. Extend its metal1 as long as the MUX21 cell. By doing so, we can connect `ctrl` signals for every MUX21 without adding more metal1 in ADDER3.

(c) using CNT_DIF, CNT_POLY and pickup cells that we made.



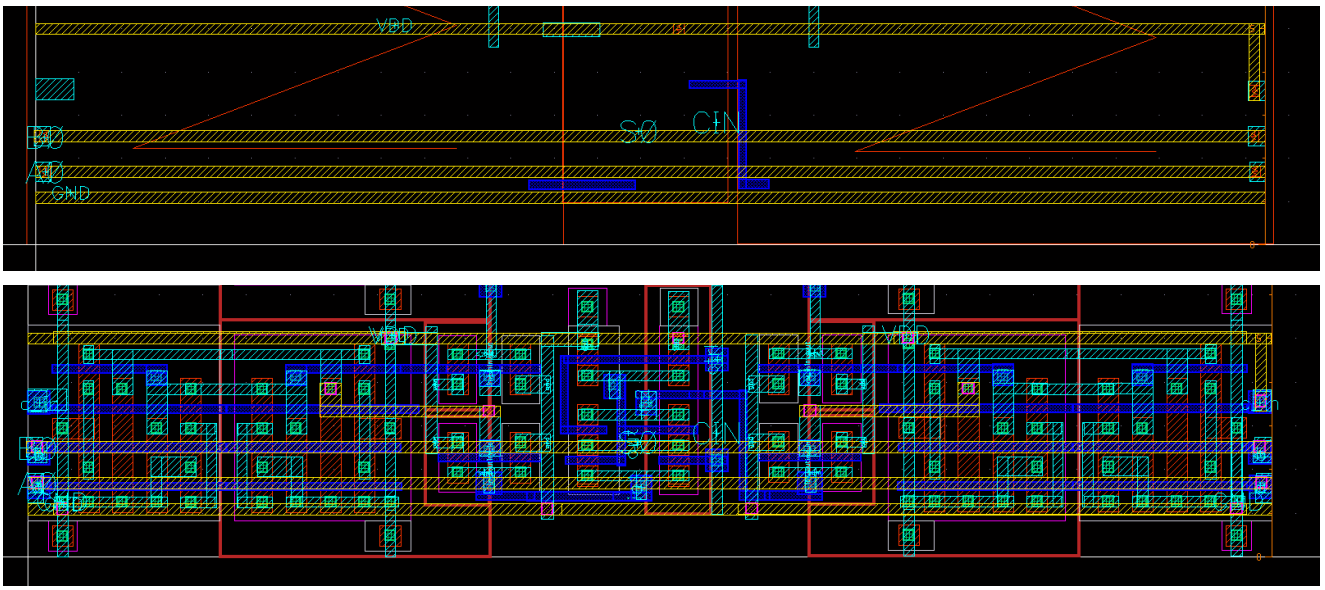

<MUX21 level0>



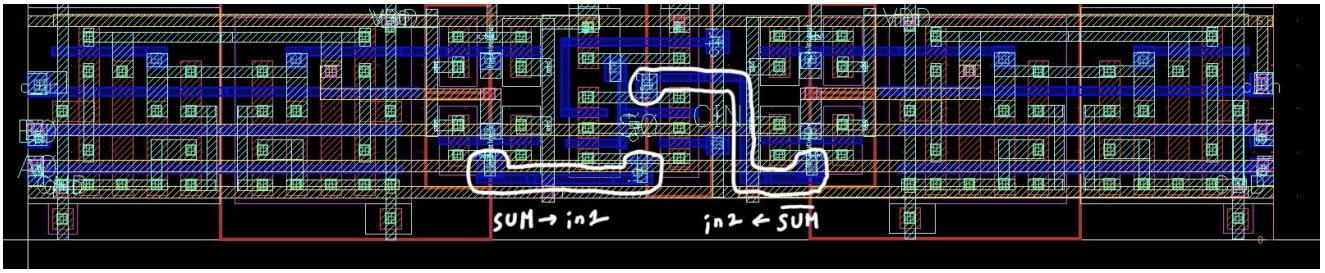

<MUX21 level10>

(5) align FA that have same inputs

- Horizontally connect source and inputs by metal2
  - also connect source(GND) to MUX21 by metal1 !
- For first bit, connect c_in of both FA to corresponding source( c_in=0 connect to GND, c_in=1 to VDD)
- connect `sum[x], sum[x]_bar` (x for 0,1,2) from both FA to MUX21
- put M1 text label on inputs and outputs
- Do LVS of first bit




<first bit adder, with 2FA & 1MUX>




<SUM->MUX21 inputs>

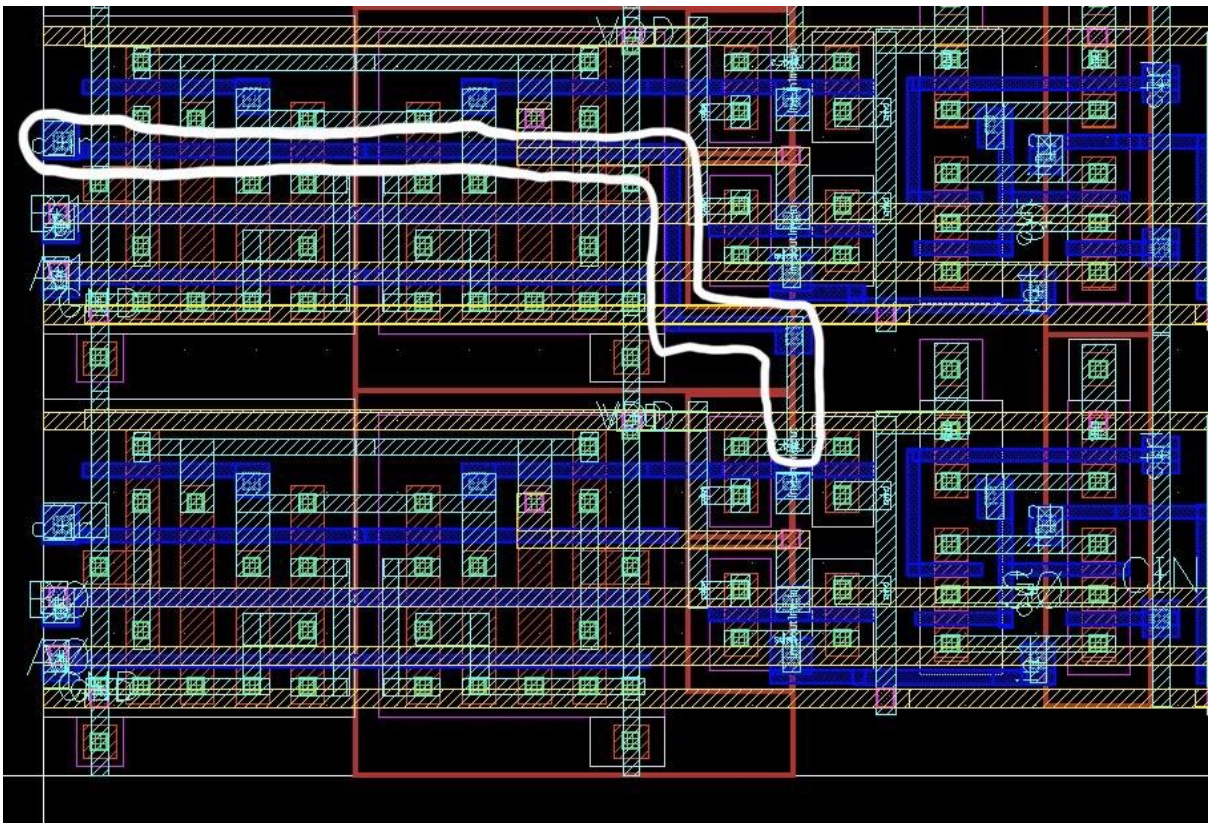
(6) arrange cells vertically by using array method (and do propagate c_out from FA)

(a) Since we are making a 3 bit adder, array FA with row = 3 for both groups. (group1 with first bit FA with c_in 0, group2 with first bit FA with c_in 1) So we can use `instance cell, row=3` to implement our goal.

(b) array MUX with row = 4 , `instance cell, row=4` to implement our goal.

(c) connect source with metal1 within FA cell

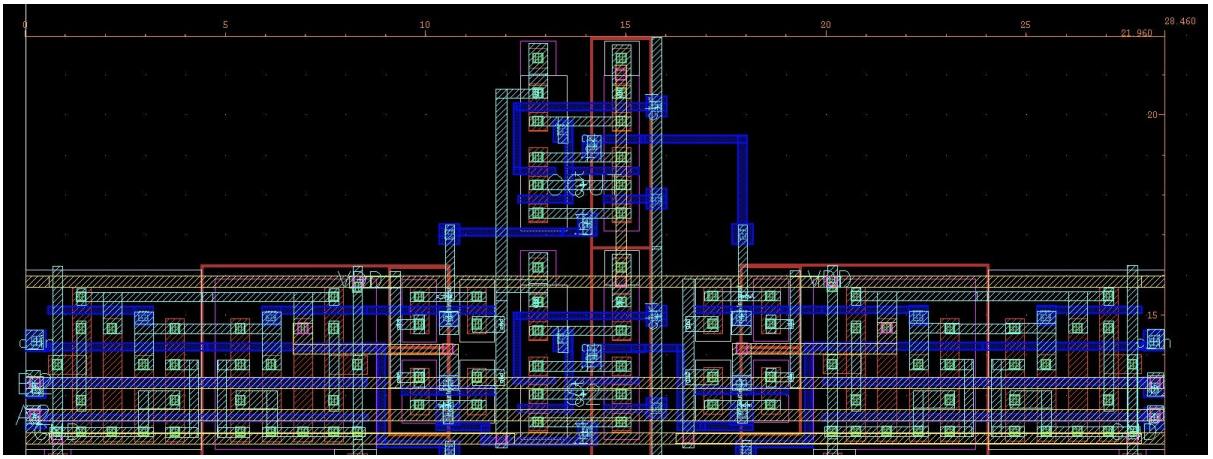
(d) connect `FA[x] c_out` to `FA[x+1] c_in`, (x=0,1) of FA ( circled in white pen in the below picture )



<propagate c_out, FA0 c_out -> FA1 c_in>

(7) use 4th MUX21 to multiplex `COUT2` and `COUT2_bar`.

Note that it is obvious to see that this MUX21 determines the height of the whole design.



(8) to sum up, the whole design includes the following steps (ordered in sequence)

- (a) think about arrangement of whole design, keep in mind that we would eventually connect inputs (by using mt2) that would cross 2FA(horizontally) with same inputs
- (b) build cells to make one FA
  - (i) MA_MOS
  - (ii) INV
  - (iii) little objects that would be reused frequently, such as poly-contact
- (c) build MUX
- (d) connect 1 FA with 1 MUX21

(e) connect 2 FA with 1 MUX21, now we have the one bit select adder
(f) addup (e) with 2FA & 1MUX, now it becomes 2 bit adder, now contains 4 FA+2 MUX
(g) addup (f) with 2FA & 1MUX, now it becomes 3 bit adder, now contains 6 FA + 3 MUX
(h) addup (g) with one more MUX, now contains 6 FA + 4 MUX, a 3 bit carry select adder is done

*note: after every step from (b) to (h) DRC passed, do LVS in each step to make sure that part doesn't contain any LVS error. By doing so, we can also satisfy the question requirement(step1 & step2) that FA and MUX21 cells need to pass DRC and LVS. Moreover, it makes final LVS testing easier, since we don't need to worry about cells having LVS error.

# 3. learned & problems

a. What I've learned
   i. how to use cell-base design
      1. how to include LIB into Virtuoso
      2. how to edit path in Virtuoso if library path changed in user directory
      3. how to call a cell by instance them
      4. how to use cell by array method to precisely put many same cells
   ii. meaning and usage of metal 2 and via
   iii. design method that layout from cell but still considering the whole design by stick diagram and chip plan
   iv. how to write cell-based schematic file
   v. It is possible to use poly as a routing resource. It can minimize the area sometimes, such as cout_bar to INV input.
b. Problems I've met
   i. Can't really understand the LVS report :(
      So I tend to go back to check layout and schematic files. If bit1 had already passed LVS, then if LVS error after adding bit2, it should really be some bug within the bit2 part.
   ii. calling instance but DRC errors
      Then we should go back to the cell and edit inside.
      That is to say, we use cell design to avoid doing repeated works, but we still need to consider problems that would arise when calling instances.