EESM5000 Project Report

ALU Design

Group Members:

Hsu, Yung-Hsiang 20734841

Chio Yat Hei 20765204

a) ALU design

Our team has designed a 2-bit full adder and utilized its capabilities to implement subtraction and division as well. For subtraction, we treat it as the addition of a positive number and a negative number (i.e. A+(−B)). To achieve this, we convert input BBB into its 2' s complement. We implement this by using an inverter to flip the bits of BBB and setting the carry-in bit of the first adder to 1. To control the functionality, we connect a PMOS transistor at the output of the inverter (for B_0' and B_1') and at the input (for B0 and B1), which are managed by the control signals SEL0 and SEL_0'.

However, the carry-out from the second full adder may not reflect the correct value because the third bit of BBB is not considered during the addition, potentially resulting in an incorrect output. To ensure that both the adder and subtractor produce accurate results simultaneously, we have added a NAND gate between the output of the third bit of the adder and the subtractor.

For the divider, since we are only performing 2-bit division, we can treat it as a series of subtractions with an additional increment. If the output of the third bit of the subtractor is 1, it indicates that B is greater than A, resulting in an output of zero. There is one exception: when A is 11 and B is 10, which leaves a remainder. In this case, we need to subtract one, so we use a NAND gate and an NMOS transistor to ensure the division output is correct under this condition. By reusing the adder with different inputs, we are able to save many transistors.

For the multiplier, we use the approximate multiplier instead of traditional multiplier. Under this condition, the only difference will be when A and B are all 11 and output of it will become 111 instead of 1001 because it only output three bit instead of four bits. Therefore, we add a string of nmos to make sure that it will pull it down to 0 when A and B are 11, then use a inverter to invert back to one, then we will have the correct output of forth bit of multiply. We also add a nmos at the output of second and third bit of the multiplier control with the forth bit, to pull it down at A and B is 11.

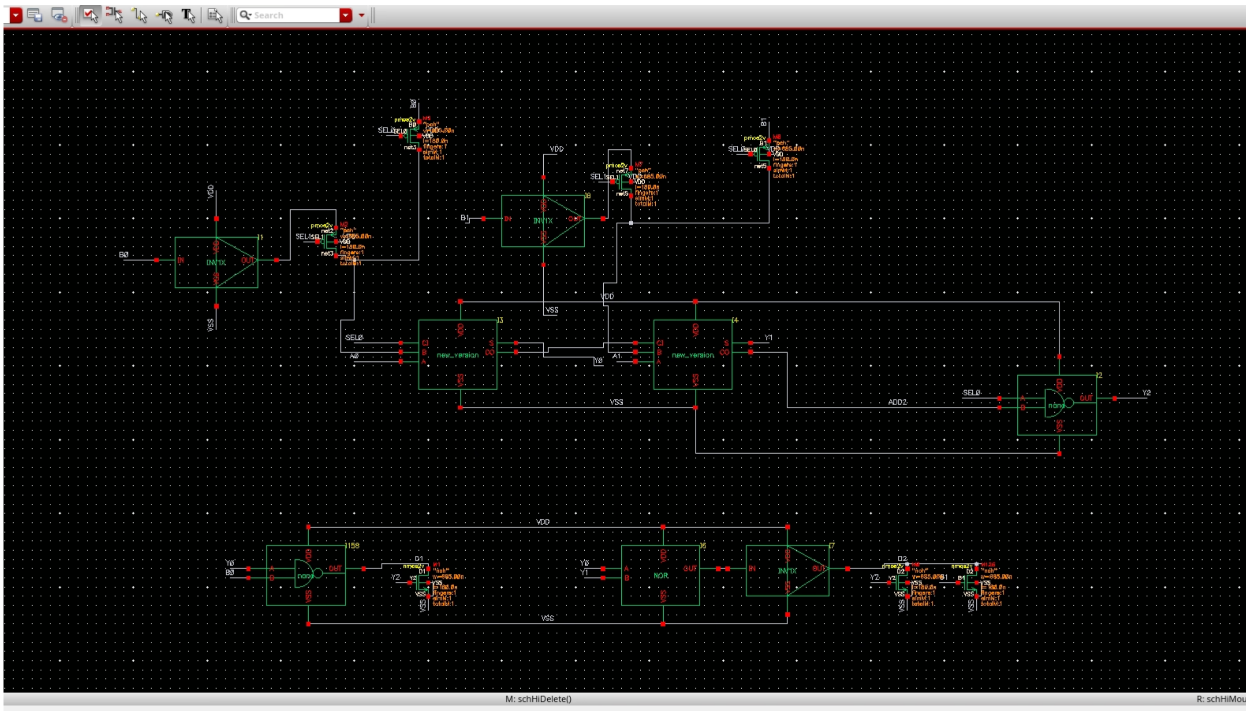
For the bitwise nand, we just get the value of nand gate of the multiplier as the input of them is the same.

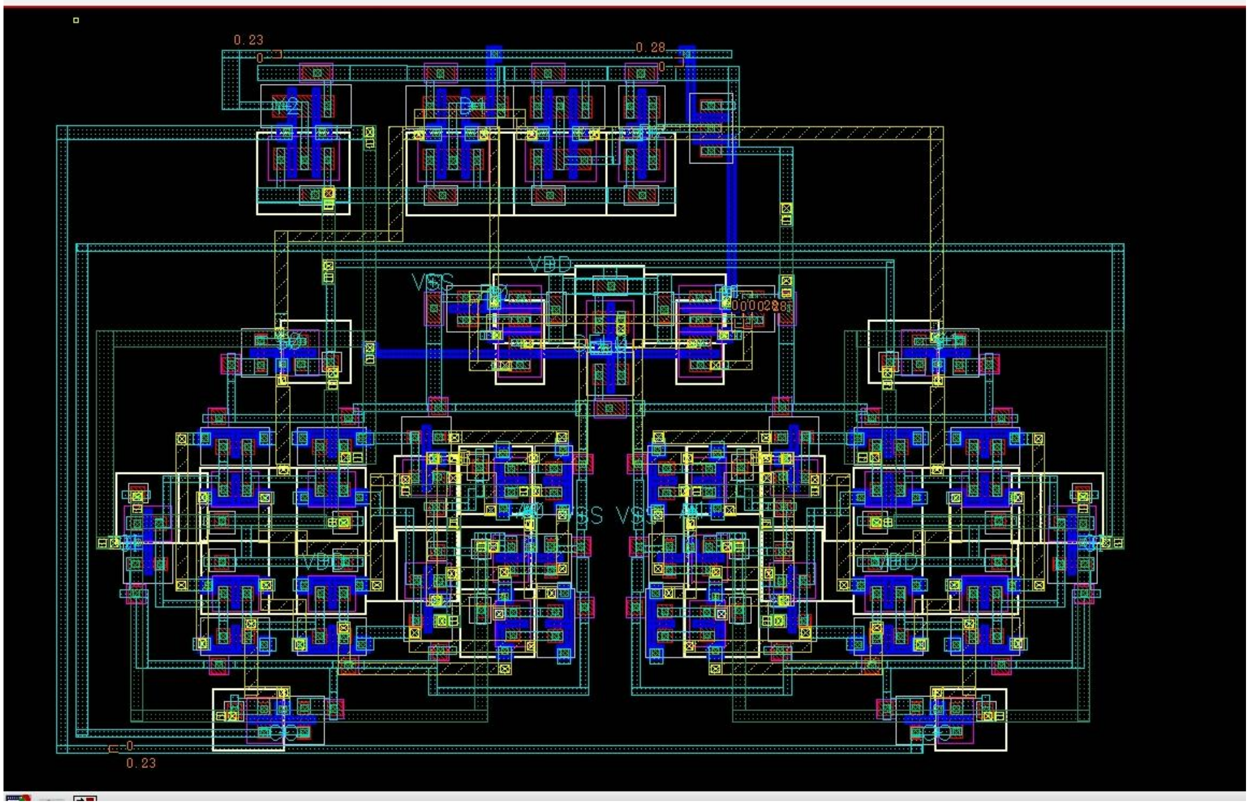
For the left logical shifter, we use transmission gate to find where A0 and A1 shift to separately. If we do it together, their output data will influence each other, so we add or gate at the second third and forth bit of it to get the final output. On top of that, the VDD of the transmission gate have been change to A0 and A1 to reduce the transistor. Also, the input of control signal have been modified too. Under this condition, we only use 38 transistors to do so, however, the output of first bit and forth bit might become a weak 0 if it have been pull up to 1 before

.

For bitwise nor and tri-state, we didn' t optimize much just stay as conventional design.

Circuit schematics:



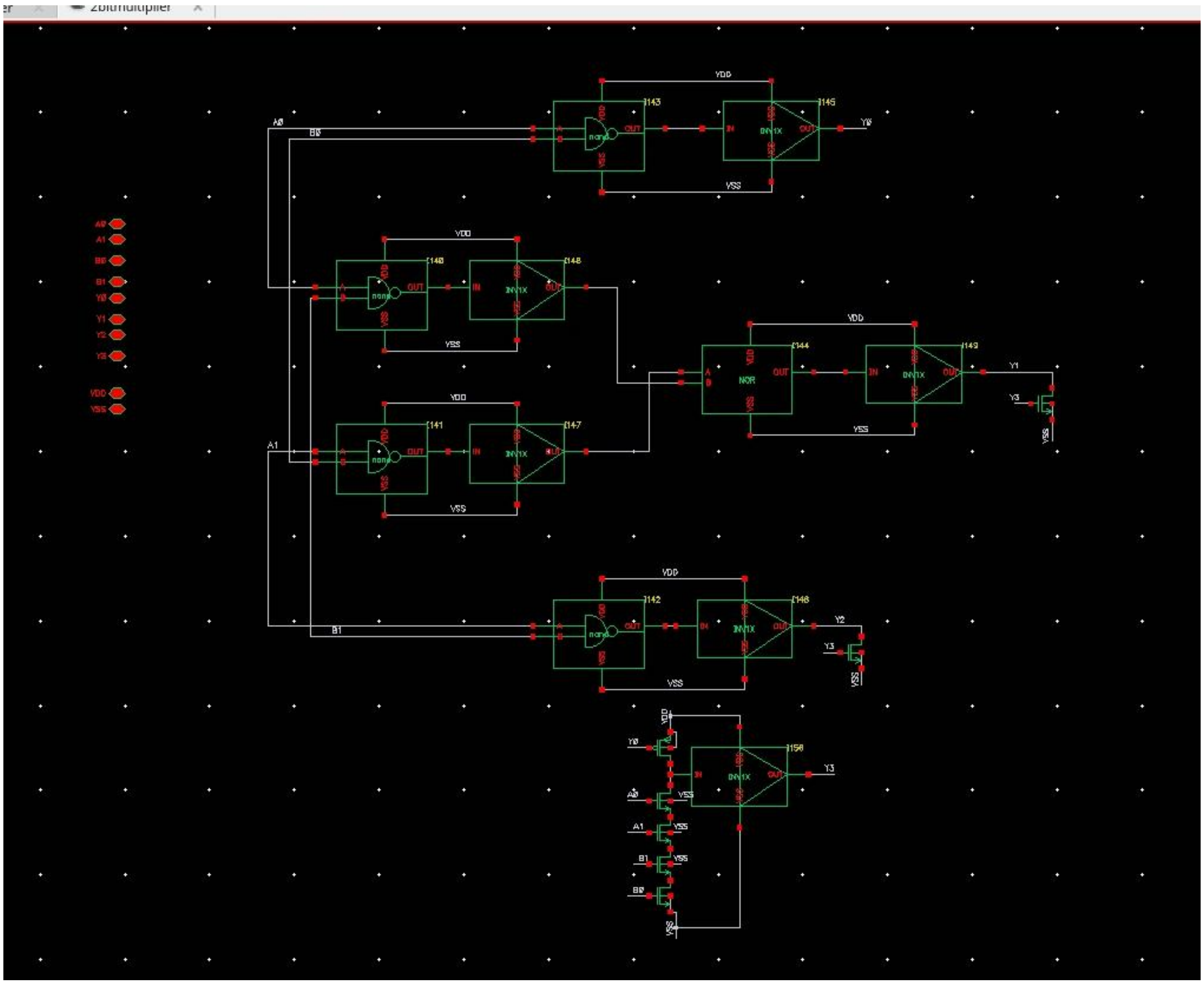


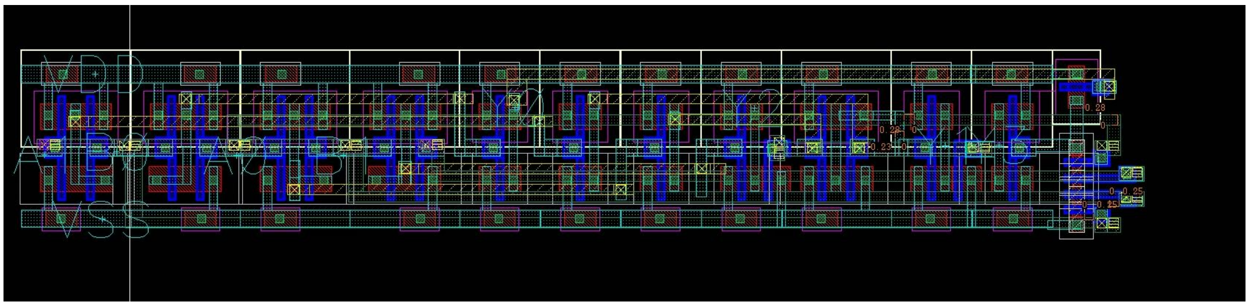
This is the circuit for adder subtractor and divider the outputs are

Adder:Y0 Y1 ADD2

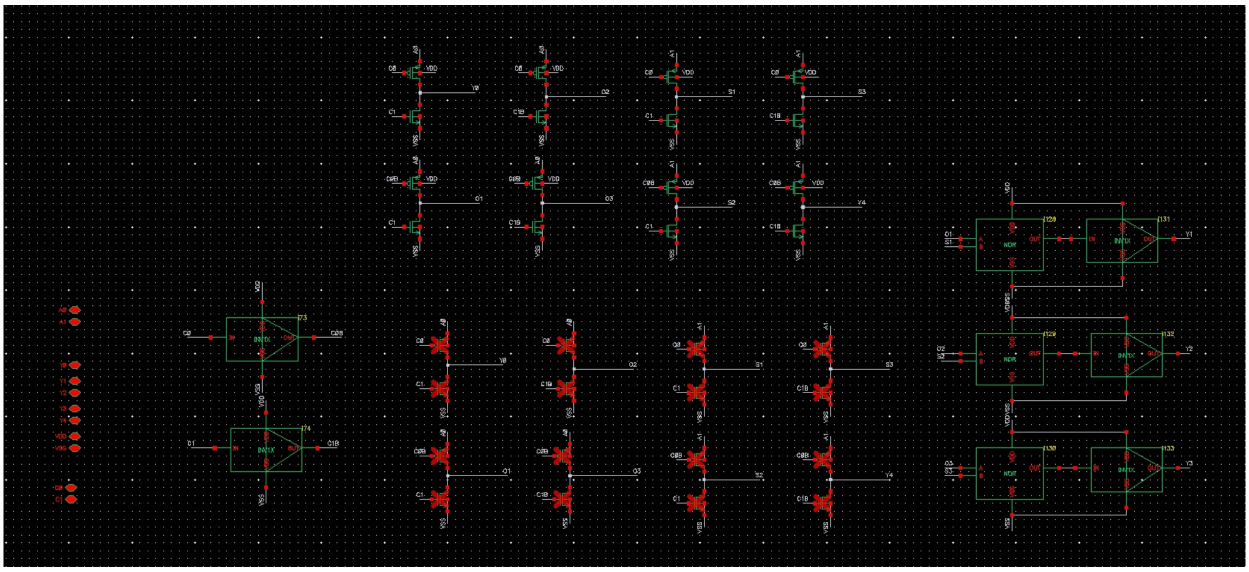
Subtractor: Y0 Y1 Y2

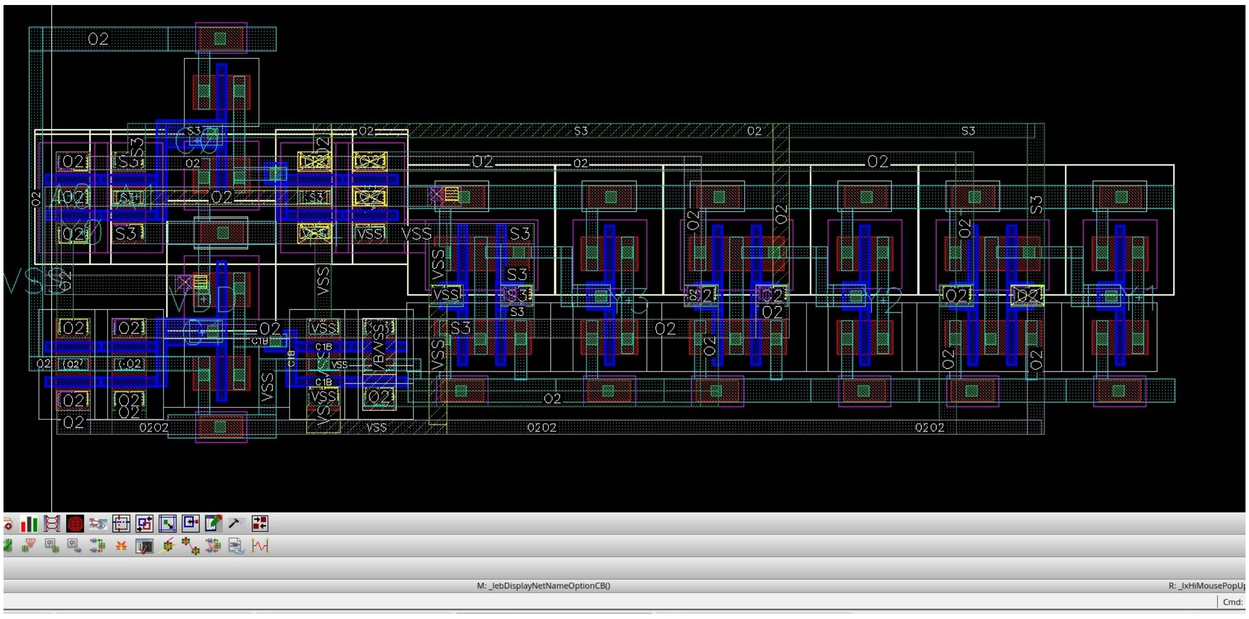
Divider: D1 D2



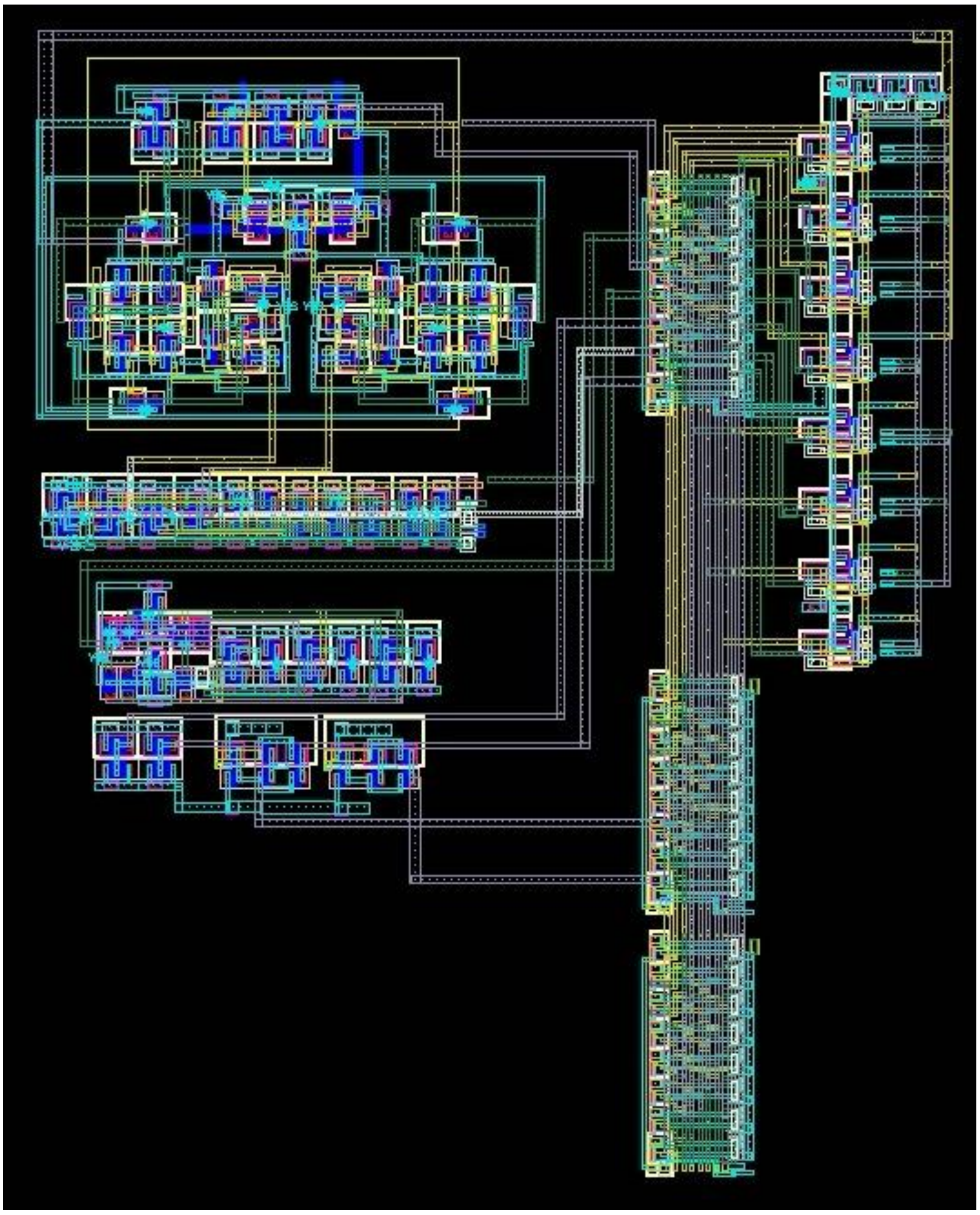


This is our 2 bit multiplier .

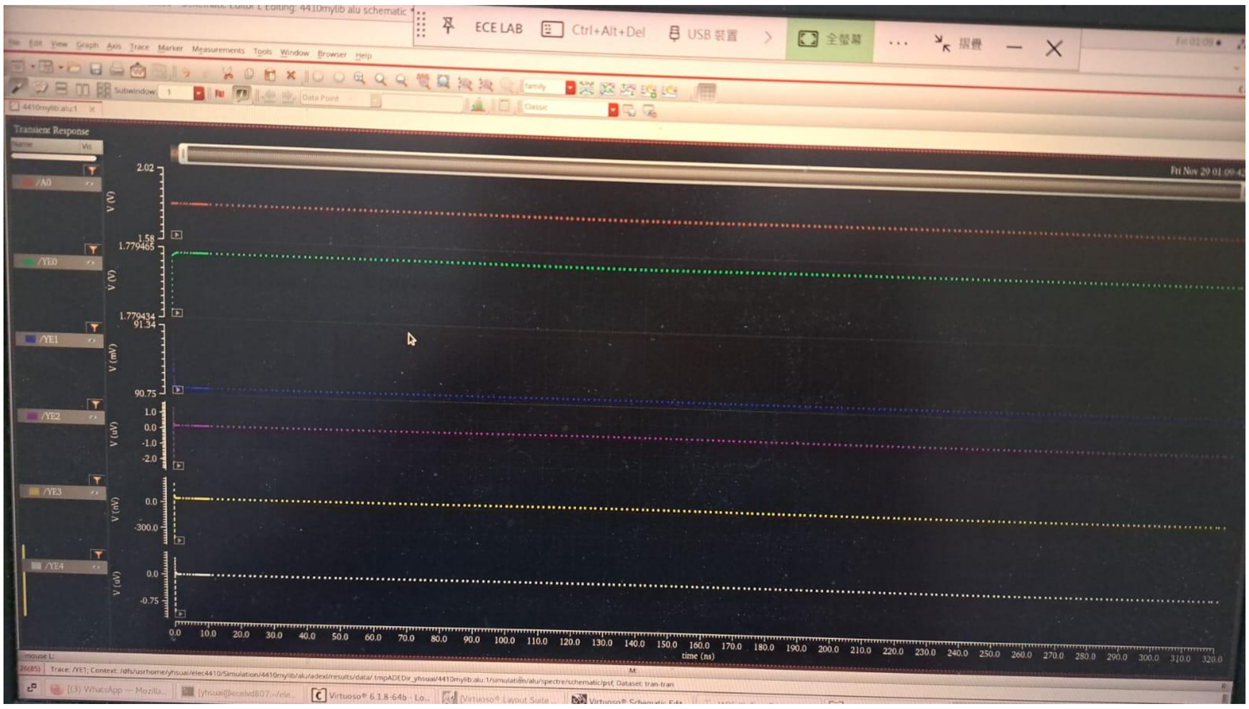




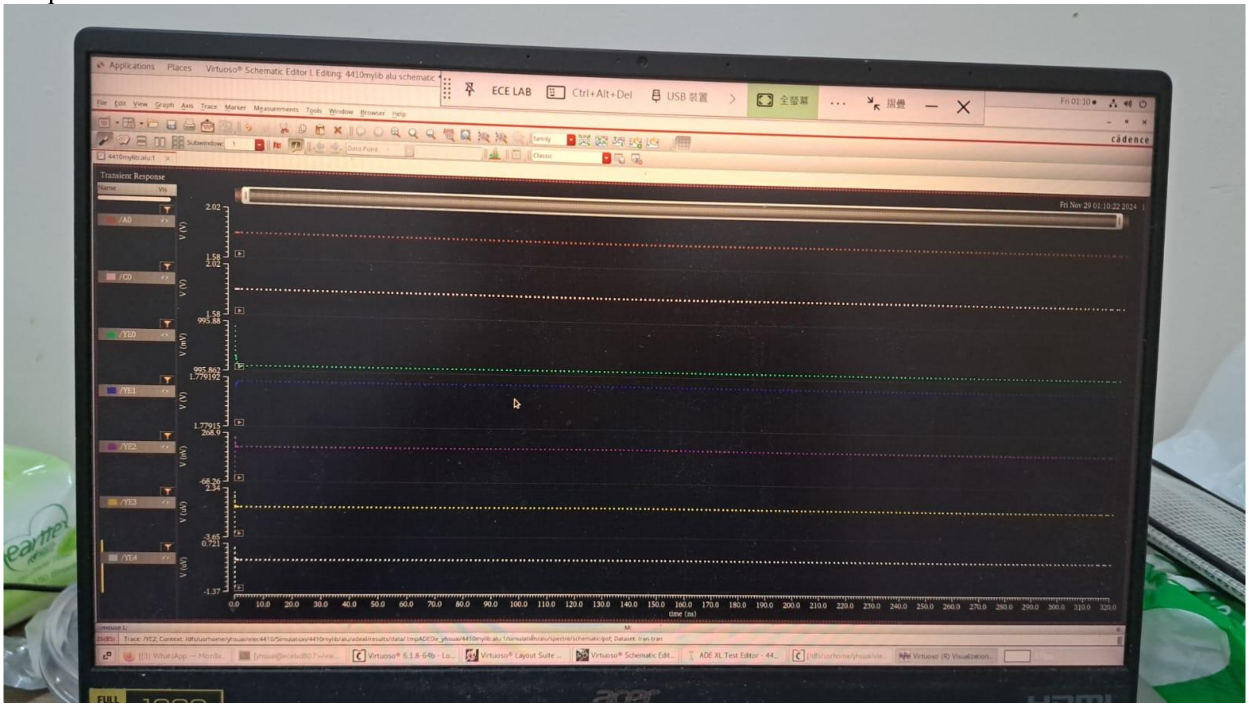
This is our logical left shifter.



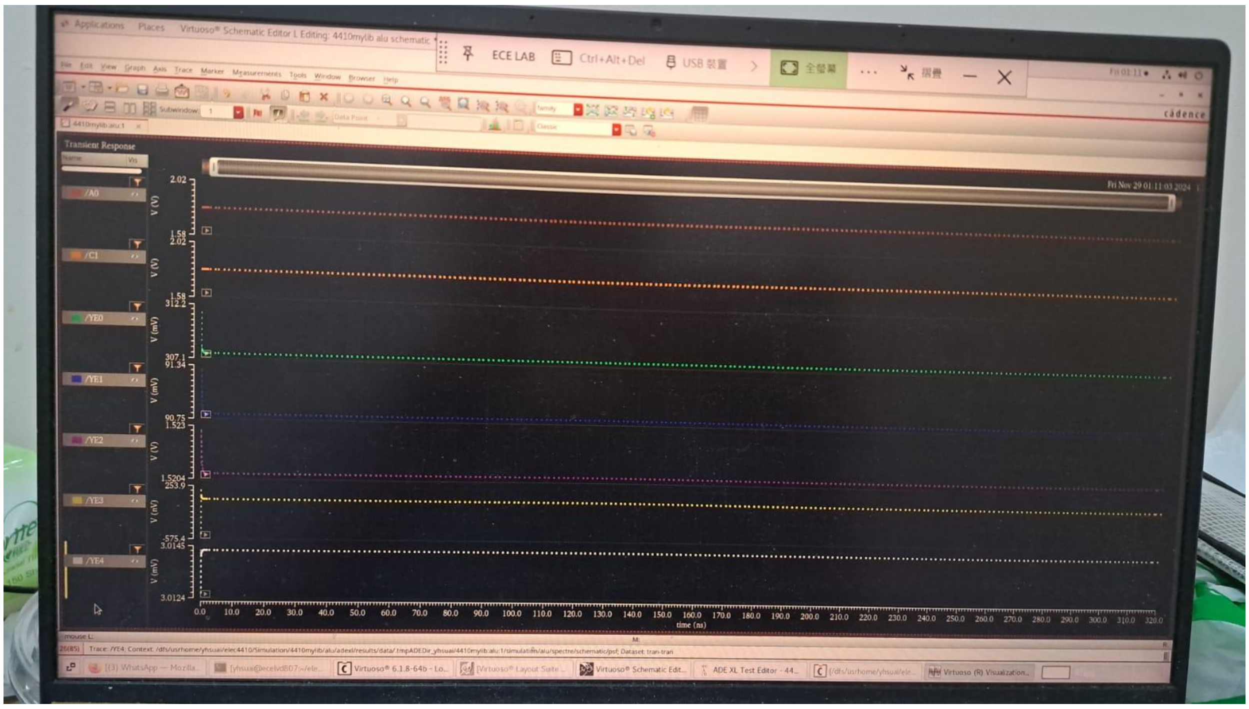
Our layout size till now is 66u*71u.
This is our layout now.

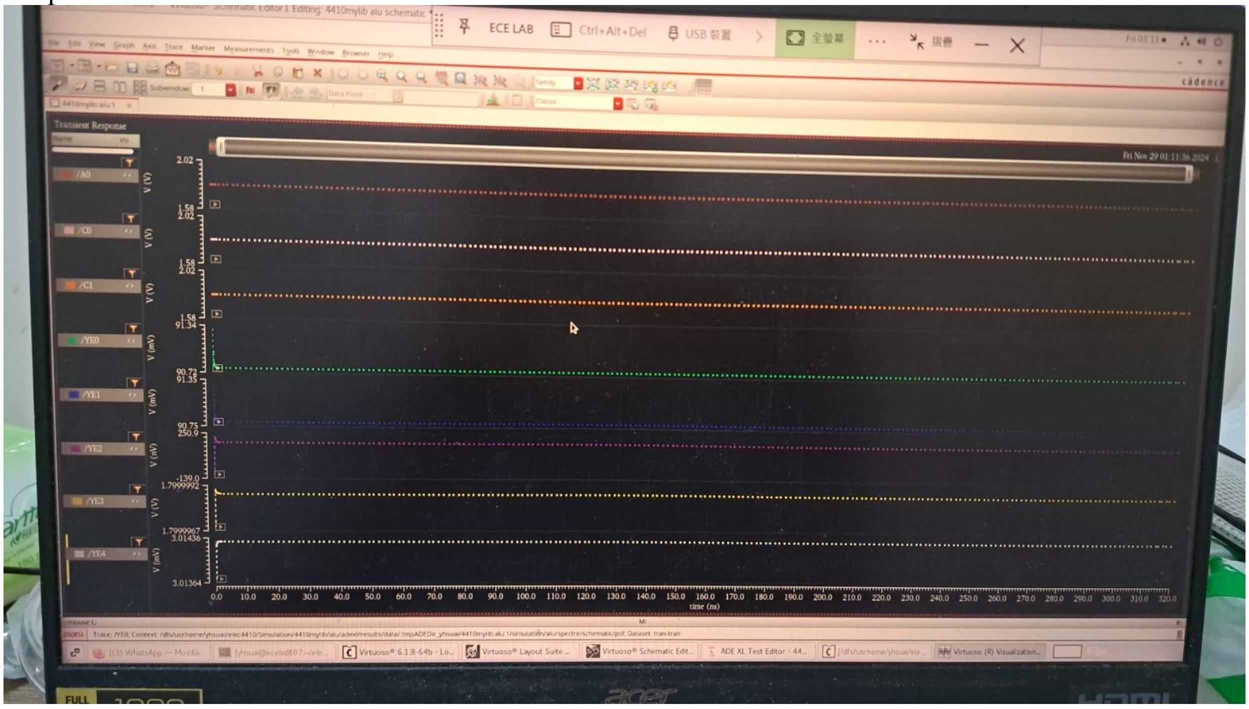


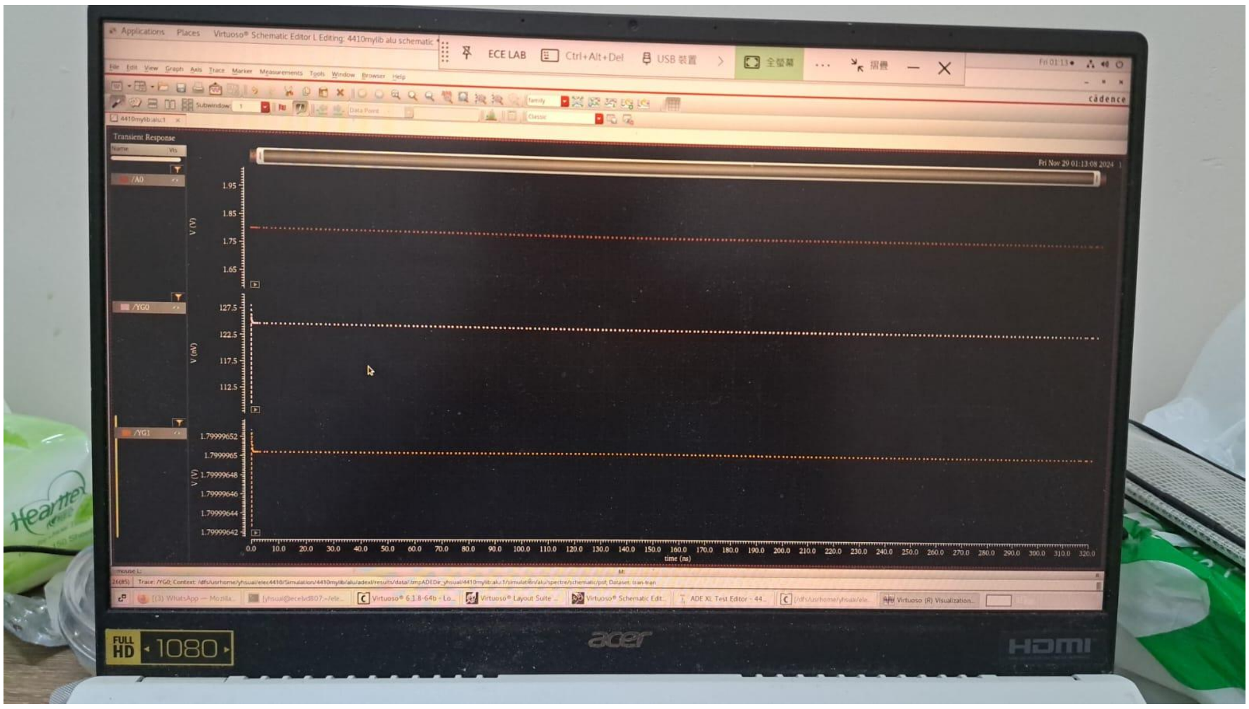
Output of left shifter with C=0

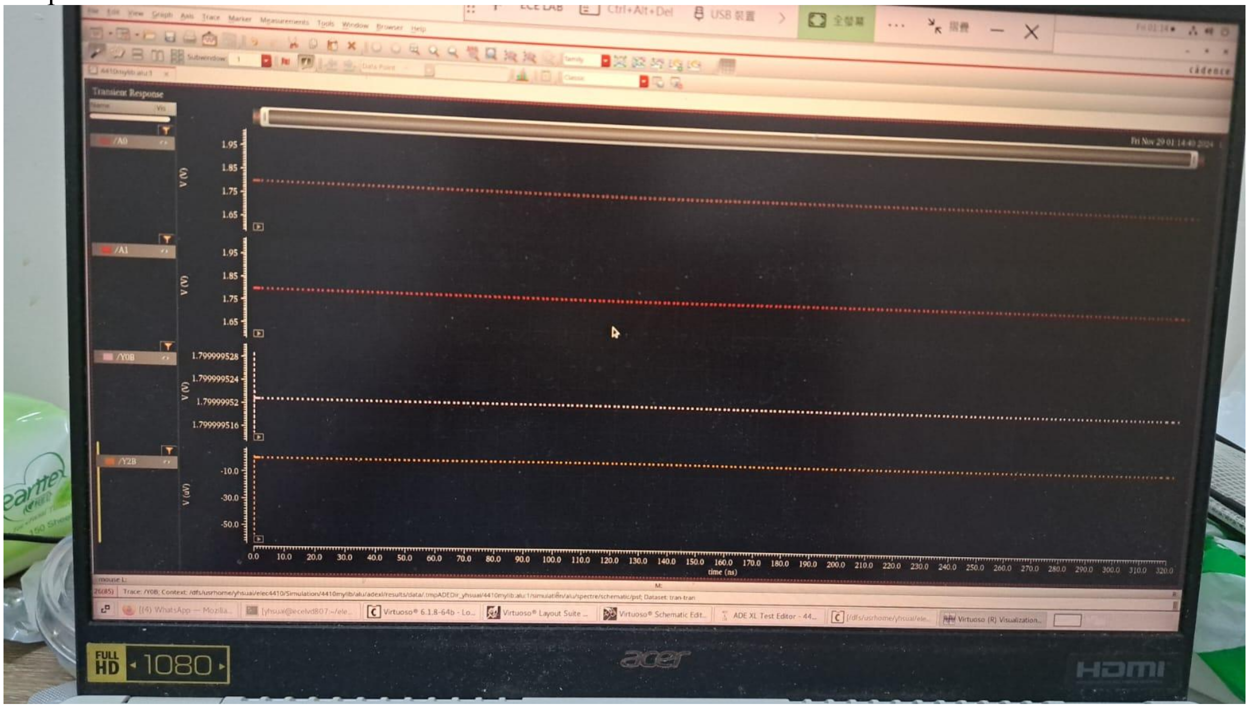


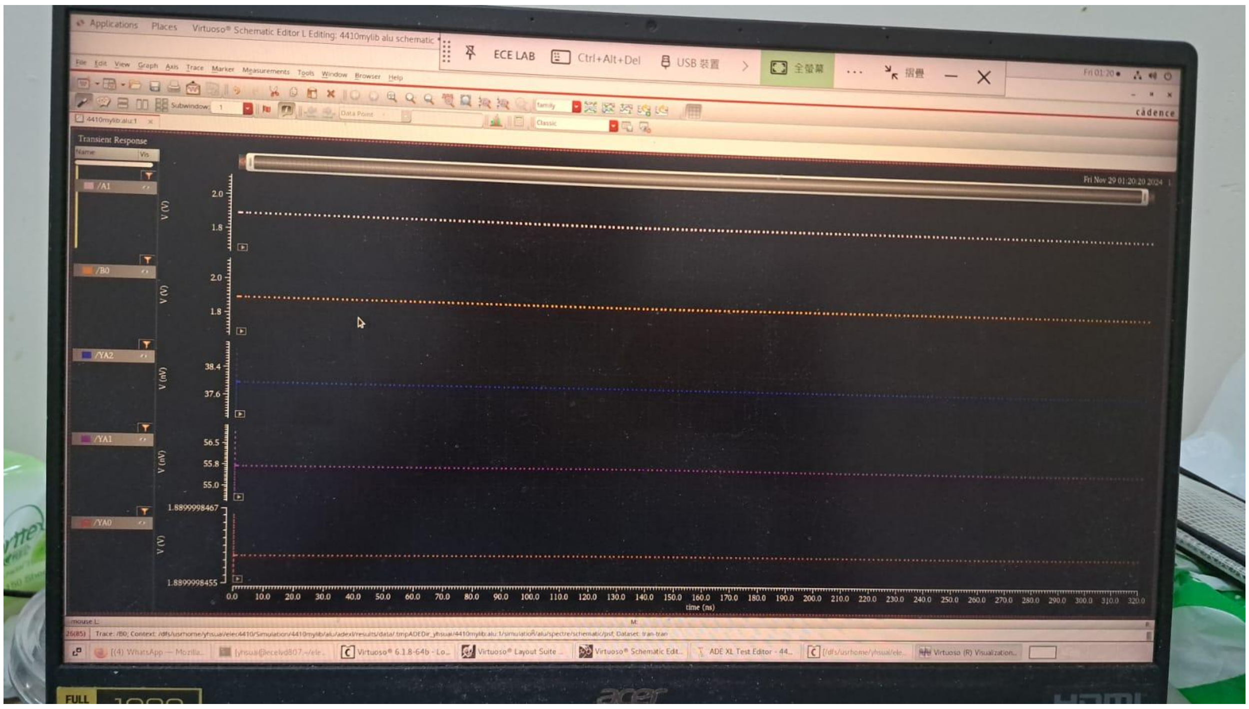
Output of left shifter with C = 01


Output of left shifter with C = 10

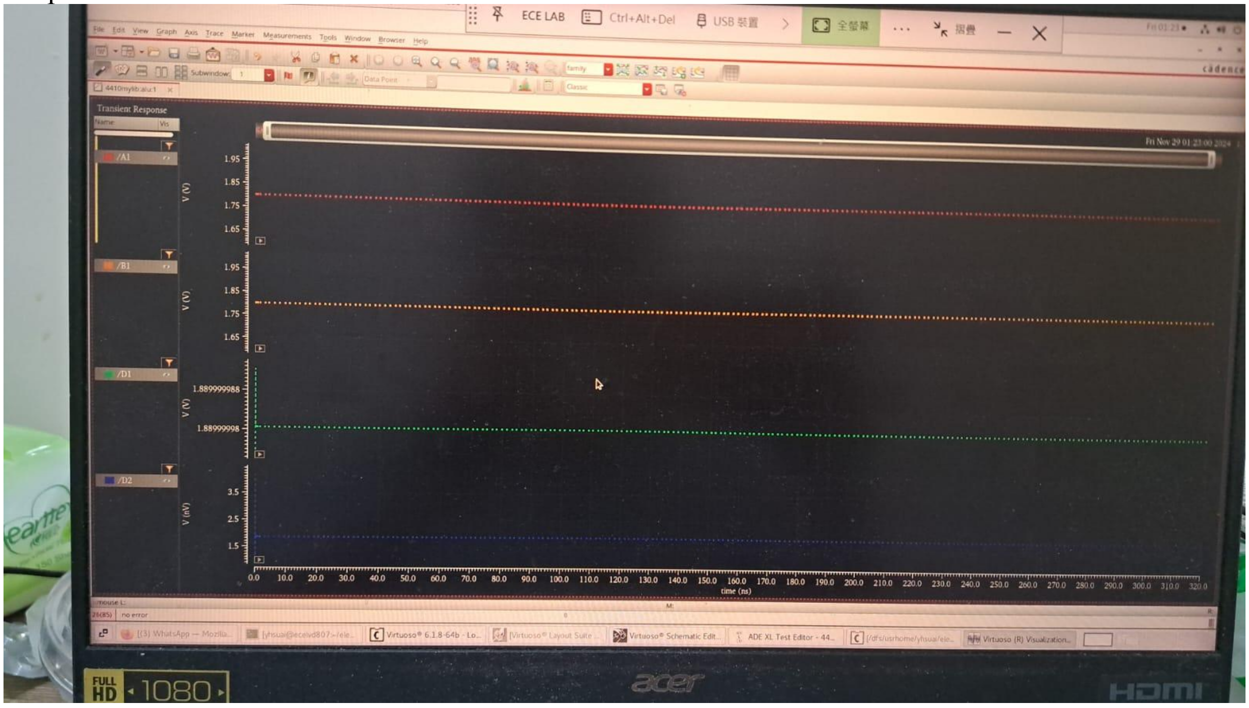

Output of left shifter with C = 11


Output of bitwise nor

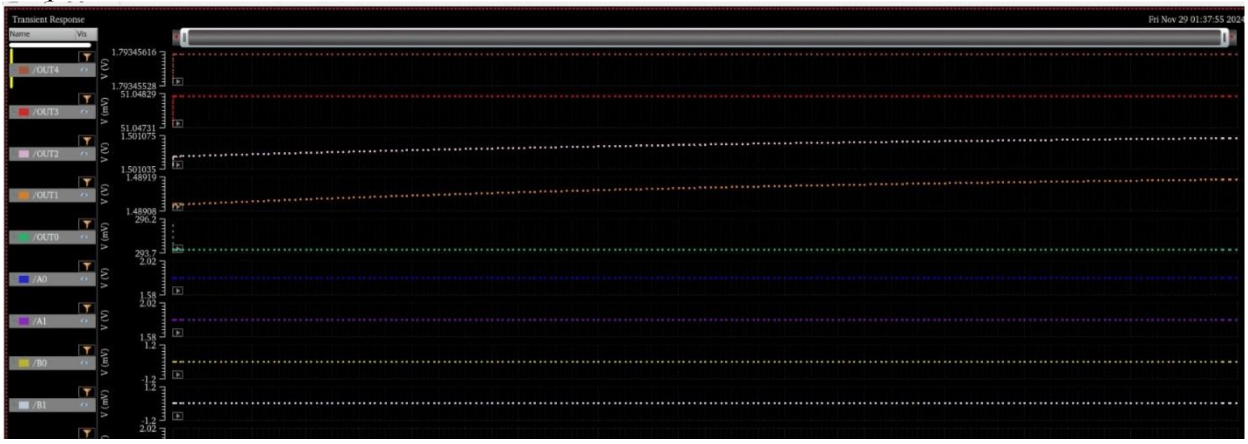

Output of bitwise nand



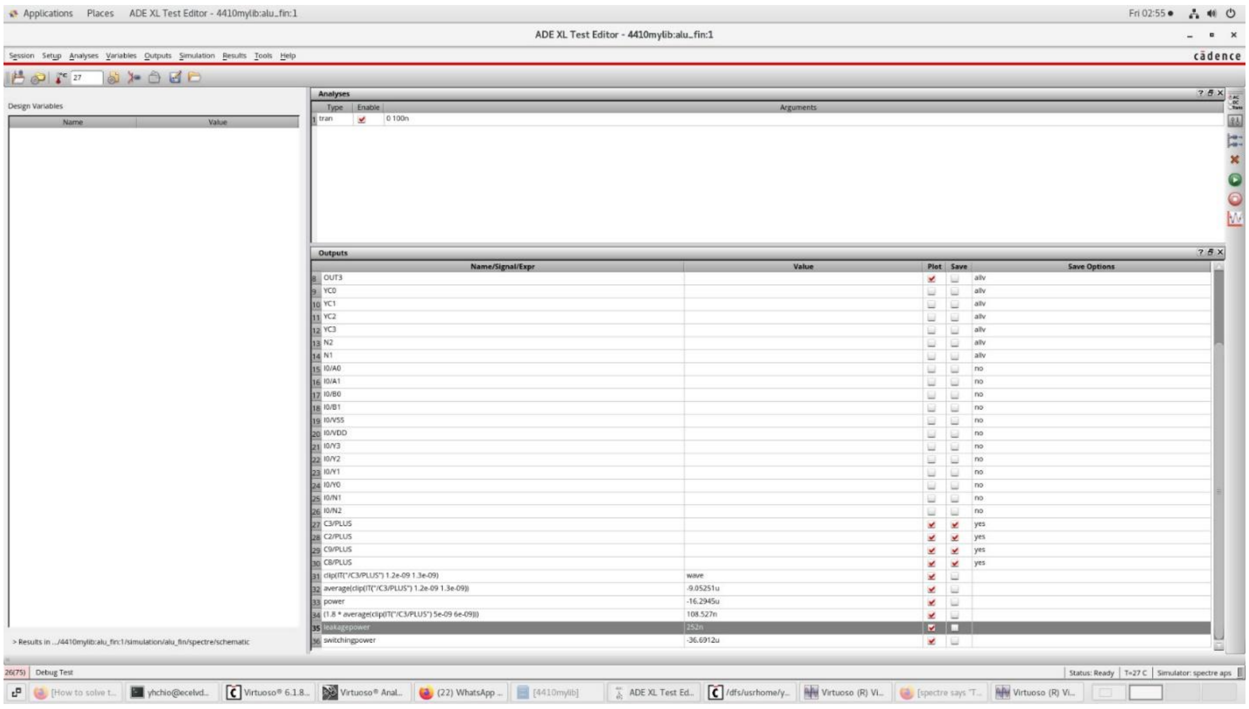
Output of substructor



Output of division



Output of tri-state



ADE XL Test Editor - 4410mylib:alu_fin:1

Analyses

	Type	Enable	Arguments
1	tran	✓	0 100n

Outputs

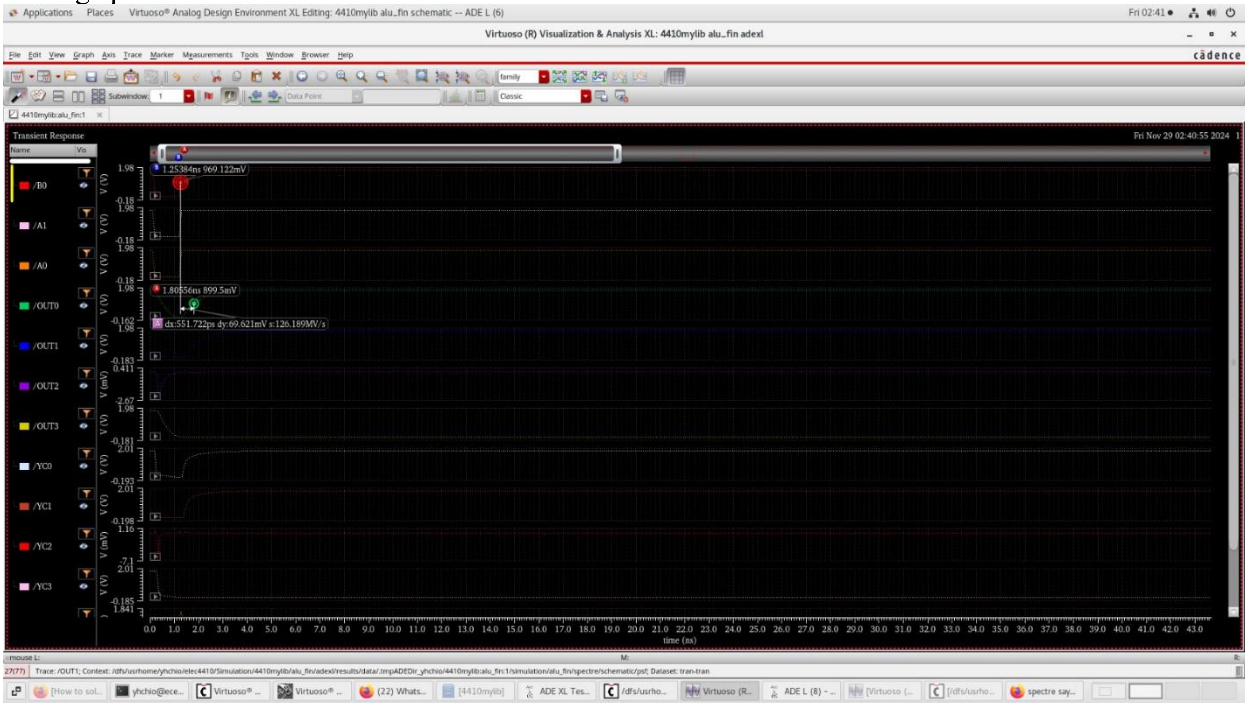
	Name/Signal/Expr	Value	Plot	Save	Save Options
8	OUT3		✓	☐	allv
9	YC0		☐	☐	allv
10	YC1		☐	☐	allv
11	YC2		☐	☐	allv
12	YC3		☐	☐	allv
13	N2		☐	☐	allv
14	N1		☐	☐	allv
15	I0/A0		☐	☐	no
16	I0/A1		☐	☐	no
17	I0/B0		☐	☐	no
18	I0/B1		☐	☐	no
19	I0/VSS		☐	☐	no
20	I0/VDD		☐	☐	no
21	I0/Y3		☐	☐	no
22	I0/Y2		☐	☐	no
23	I0/Y1		☐	☐	no
24	I0/Y0		☐	☐	no
25	I0/N1		☐	☐	no
26	I0/N2		☐	☐	no
27	C3/PLUS		✓	✓	yes
28	C2/PLUS		✓	✓	yes
29	C9/PLUS		✓	✓	yes
30	C8/PLUS		✓	✓	yes
31	clip(IT("/C3/PLUS") 1.2e-09 1.3e-09)	wave	✓	☐	
32	average(clip(IT("/C3/PLUS") 1.2e-09 1.3e-09))	-9.05251u	✓	☐	
33	power	-16.2945u	✓	☐	
34	(1.8 * average(clip(IT("/C3/PLUS") 5e-09 6e-09)))	108.527n	✓	☐	
35	leakagepower	252n	✓	☐	
36	switchingpower	-36.6912u	✓	☐	

> Results in .../4410mylib:alu_fin:1/simulation/alu_fin/spectre/schematic

Status: Ready | T=27 C | Simulator: spectre aps

Switching power is -36.6912u

Leakage power is 252n



Our group delay is 551ps

Our group's FoM = 1/66*71*10^-12*1/36.6912*10^-6*1/252*10^-9*1/551*10^-12=4.18874*10^28