HACETTEPE UNIVERSITY

Electrical and Electronics Engineering

ELE 419 Integrated Circuit Design
Term Project
Full Custom ALU Desgin

Student Name : Ertuğrul Tiyek

Student ID : 21828916

Instructor: Associate Professor Dinçer Gökcen

Table of Contents

Design:

Firstly I want to introduce my design concerns. I aimed a different design in this project.

- I tried to design each module in standardized size. This allowed me to put every element together without wasting space. I first specified a height to module to fit every design from inverter to whole ALU. I specified this height as 55.5 after a few design.

- The other aim was scaling whole ALU from 1 bit to whatever size that needed. The whole ALU can be scaled by adding more ALU's like construction toys (LEGO). This also allowed me not to waste space with scaling modules and assembling different sized modules with wide bus connections.

- I used **Only 2 Layers of Metal** to reduce production cost.

- I like to use complex logic, but the complex logic needs different sized transistors, so using complex logic would destroy my standardized size aim. I didn't use complex logic in this project. Instead of that, I used multi input gates and combinational gates.

- For sizing, I mostly preserve 2:1 ratio. But according to simulation results, I changed some of the sizes. For example transmission gates are implemented using 10:10 sizing.

As following my aims, I am able to design the whole circuit took only **1789 x 969.5** squares of space.

Logic Design:

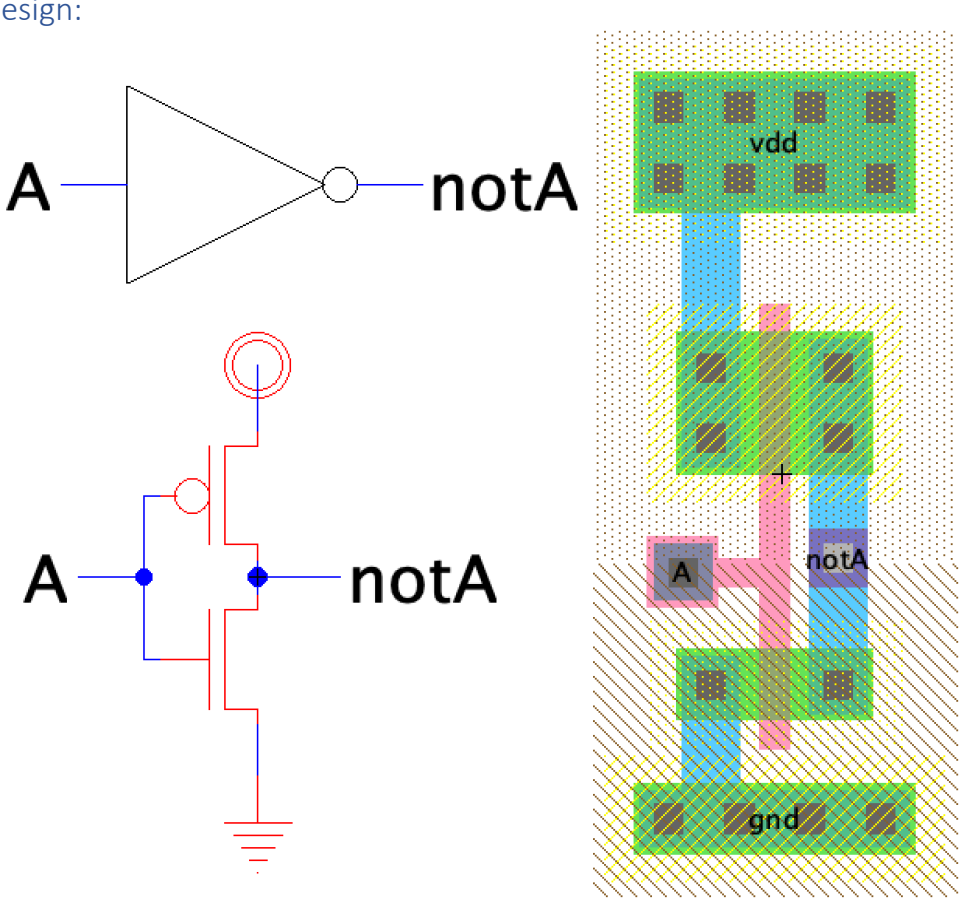
S2	S1	S0	CIN	B_inp	C_inp	OP
0	0	0	0	X	X	A
0	0	0	1	X	X	A'
0	0	1	0	X	X	AND
0	0	1	1	X	X	OR
0	1	0	0	B	0	XOR
0	1	0	1	0	1	INC
0	1	1	0	B	0	ADD
0	1	1	1	B	1	ADD+C
1	0	0	0	X	X	
1	0	0	1	B	1	SUB
1	0	1	0	B	0	SUB-B
1	0	1	1	X	X	
1	1	0	0	0	0	DEC
1	1	0	1	X	X	
1	1	1	0	X	X	<<
1	1	1	1	X	X	>>

Karnaugh Map for Arithmetic Unit B input:

	00	01	11	10
00	X	X	X	X
01	B	0	B	B
11	0	X	X	X
10	X	B	X	B

$$B \leftarrow (S_2S_1 + S_2{'}S_0{'}C_{in})$$

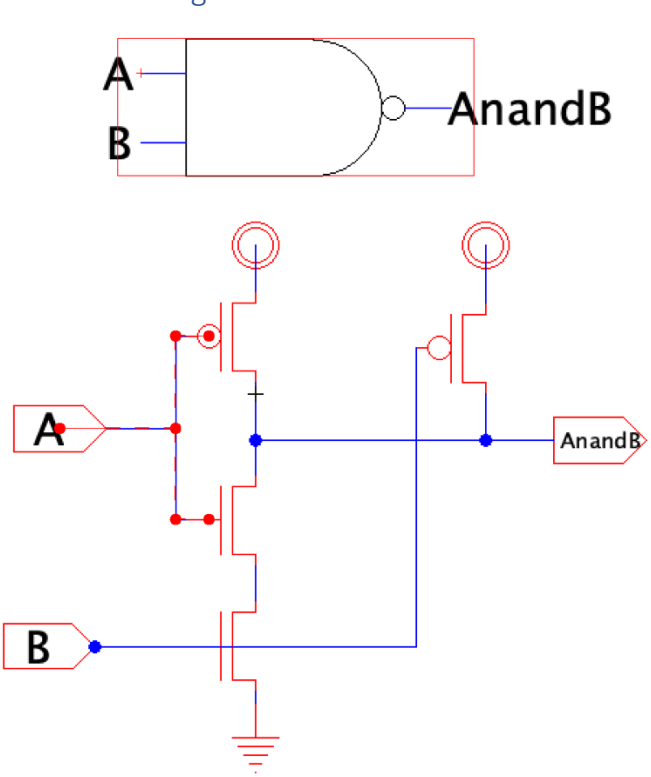
Inverter Design:

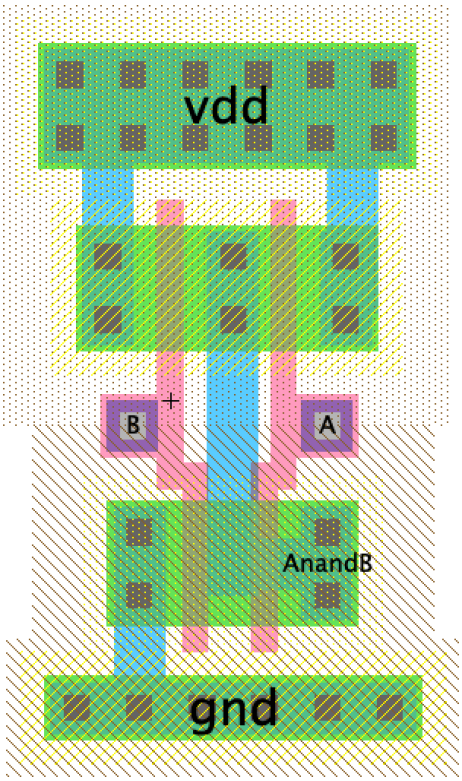


Although the inverter is the smallest gate, I am able to design all gates in the height of an inverter.

Transistor Sizing:
Pmos: 10
Nmos: 5
Ratio: 2:1

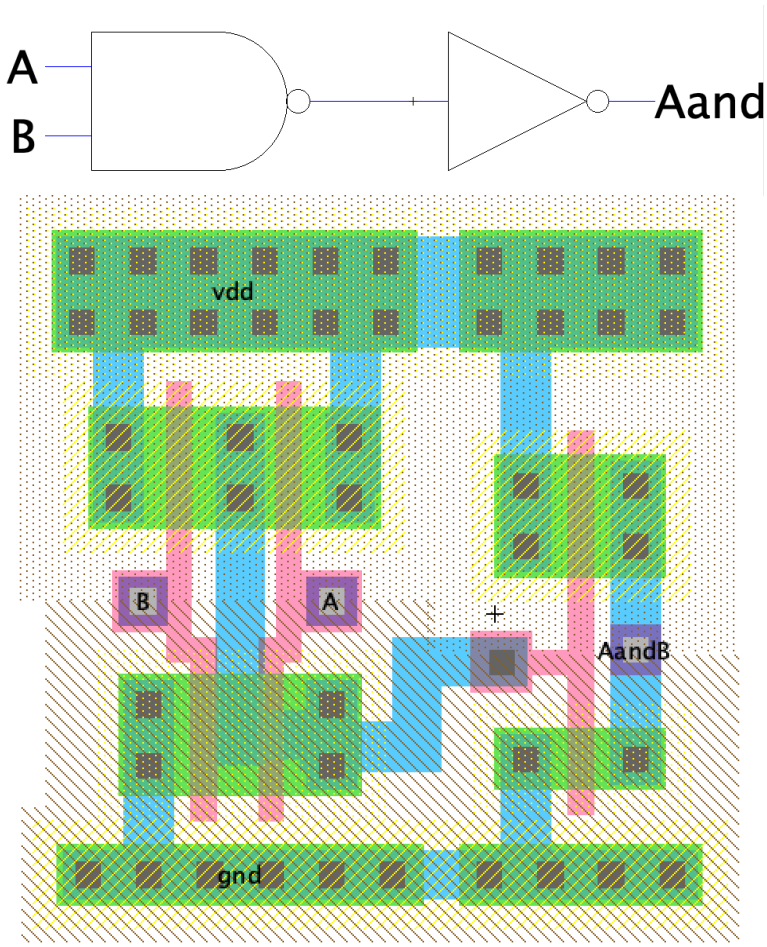
Nand Gate Design:



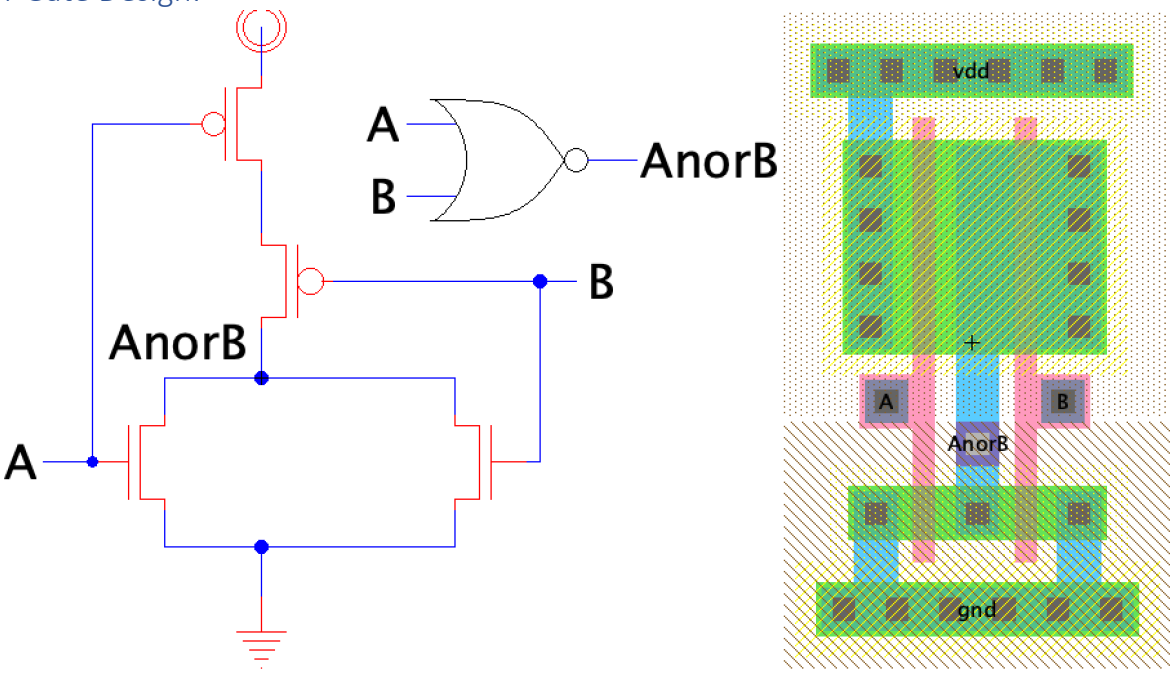


Transistor sizes :
Pmos: 10
Nmos: 10
Ratio: 2:1

And Gate Design:

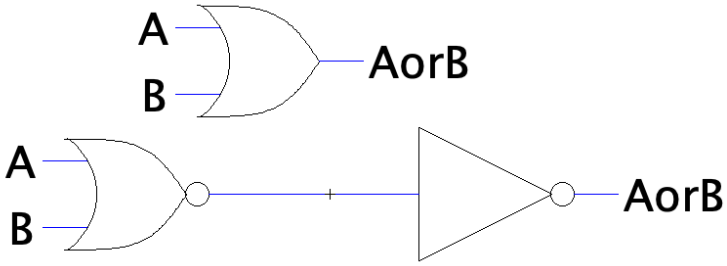


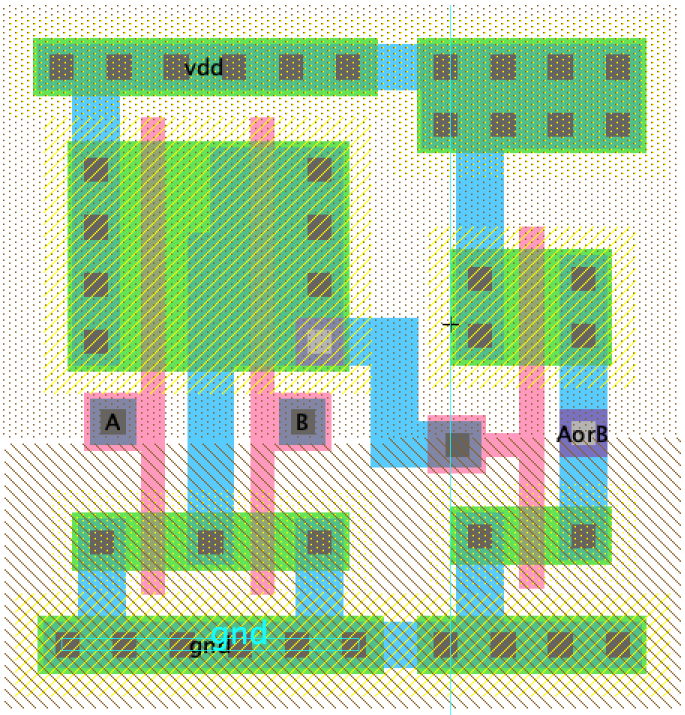
Nor Gate Design:



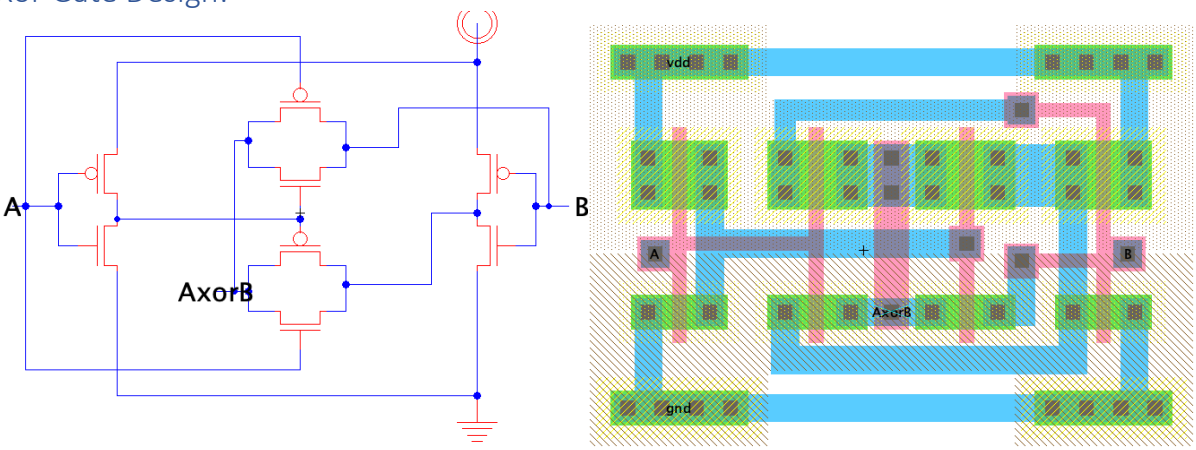
Transistor Sizes:
Pmos: 20
Nmos: 5
Ratio: 2:1

Or Gate Design:



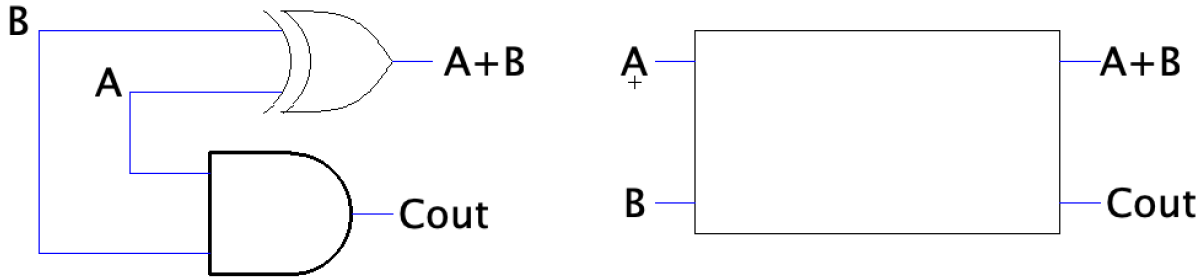


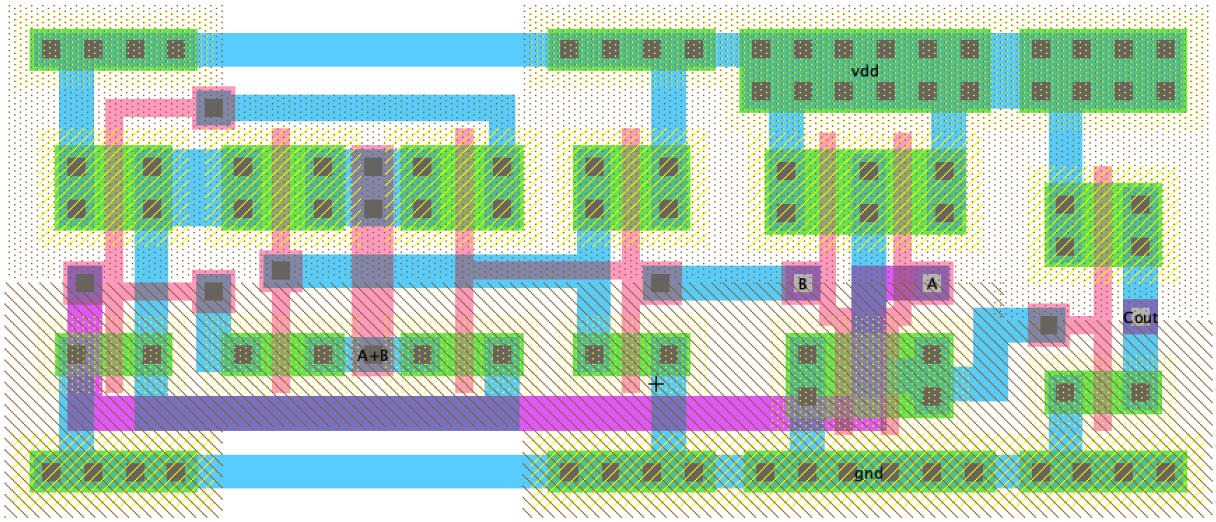
Xor Gate Design:



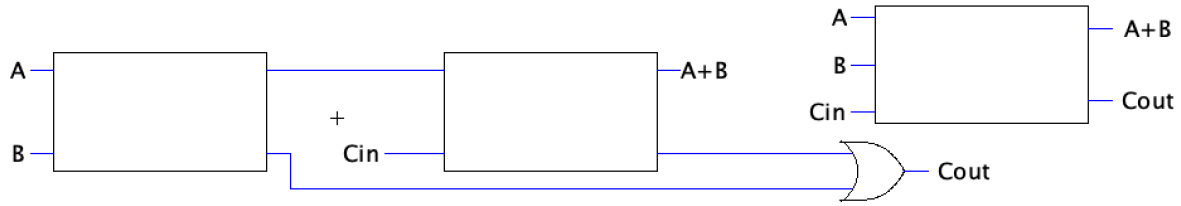
I designed Xor Gate using **Transmission Gate** technology. Transmission gates are useful technology for saving resource and space, but because they are not supplied by direct voltage sources, it causes attenuations. Because I paid much attention not to cascade transmission gates. Using transmission gates with combination of other gates is a useful thing to both not losing signal quality and saving space and resources.

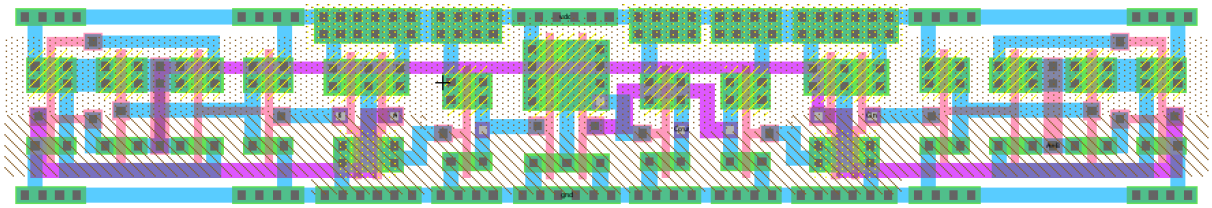
Half Adder Design:





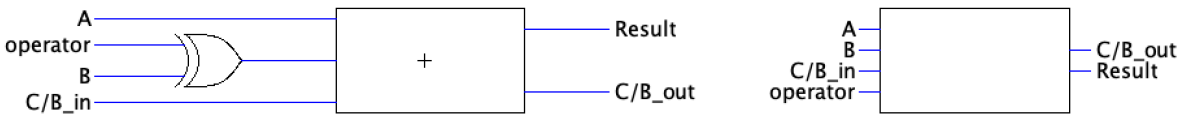
Full Adder Design:

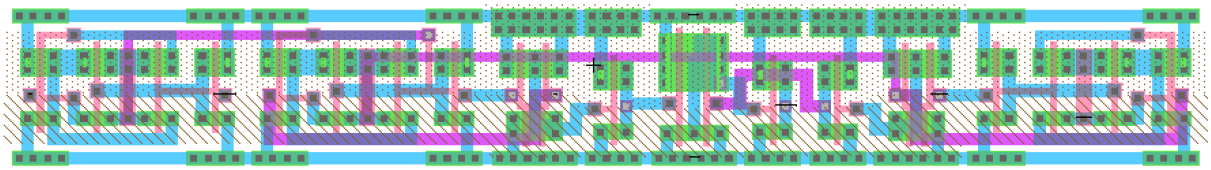




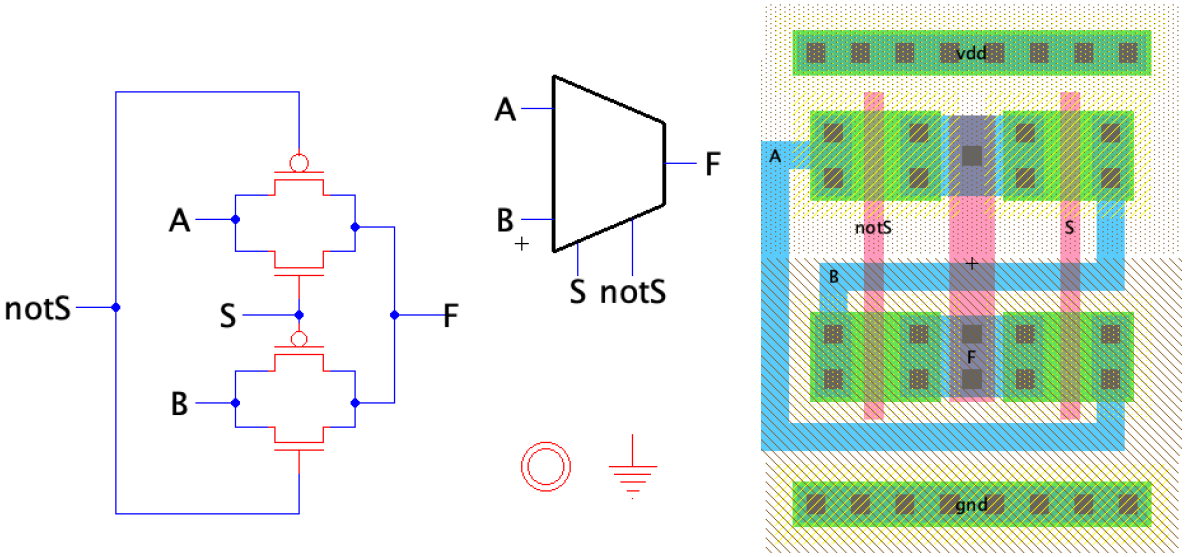
Again I didn't choose using complex logic to implement Full Adder. Because it would not fit in my standards. Instead I used traditional two half adder method.

Adder Subtractor Design:





Multiplexer Design:



I used transmission technology to implement multiplexer. Sizing is done by simulation results. I used 10:10 ratio.

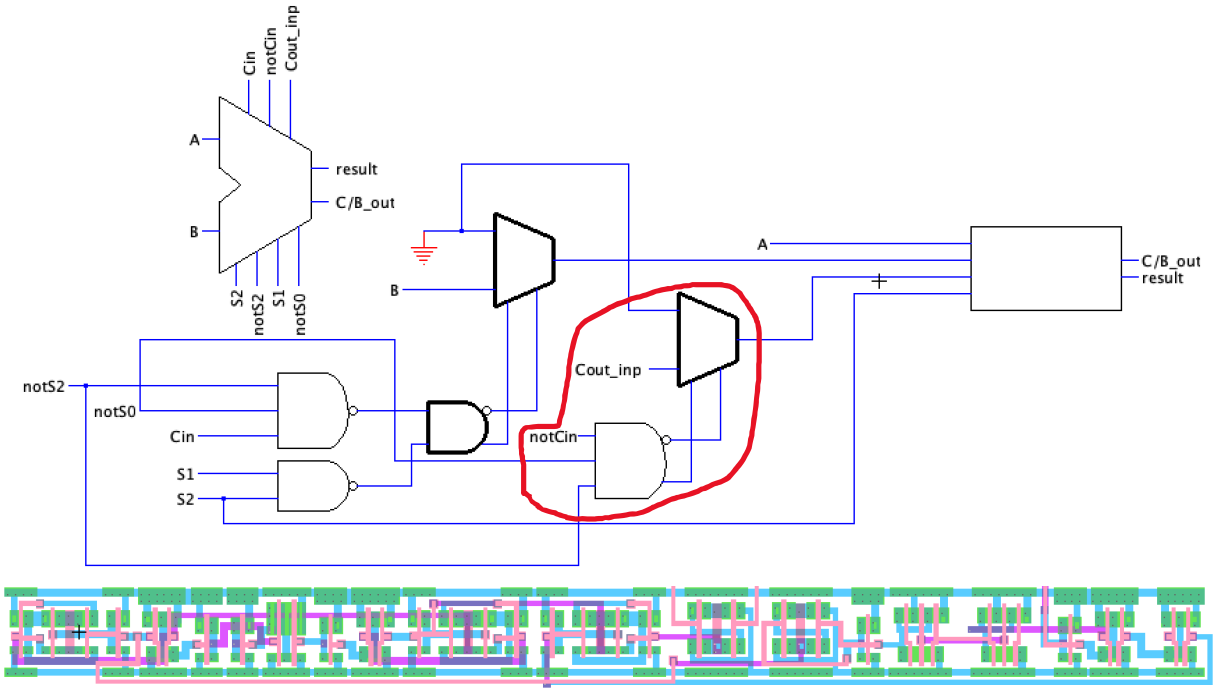
Arithmetic Unit Design:

I designed two different Arithmetic Units to perform 7 of the instructions that are required from ALU. These are:

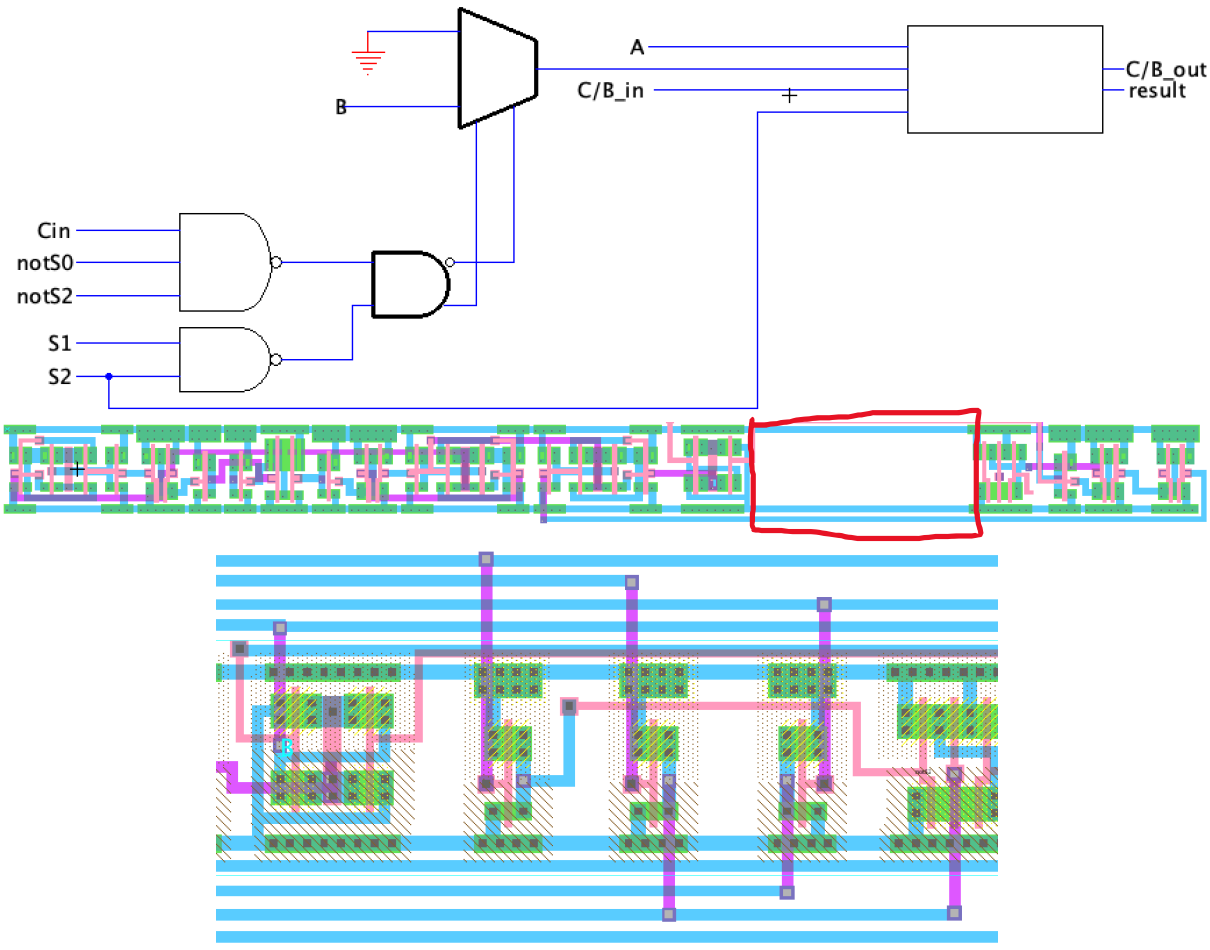
0	1	0	0	B	0	XOR
0	1	0	1	0	1	INC
0	1	1	0	B	0	ADD
0	1	1	1	B	1	ADD+C
1	0	0	0	X	X	
1	0	0	1	B	1	SUB
1	0	1	0	B	0	SUB-B
1	0	1	1	X	X	
1	1	0	0	0	0	DEC

The reason I designed two of the Arithmetic units is the first bit of the ALU needed less logic than other bits, so I removed C/B_in logic from first bit, than I opened some space to insert other logic (inverters for selection bits). This get me out of putting three inverters for each bit. I used all three inverted common selection signals to all bits.

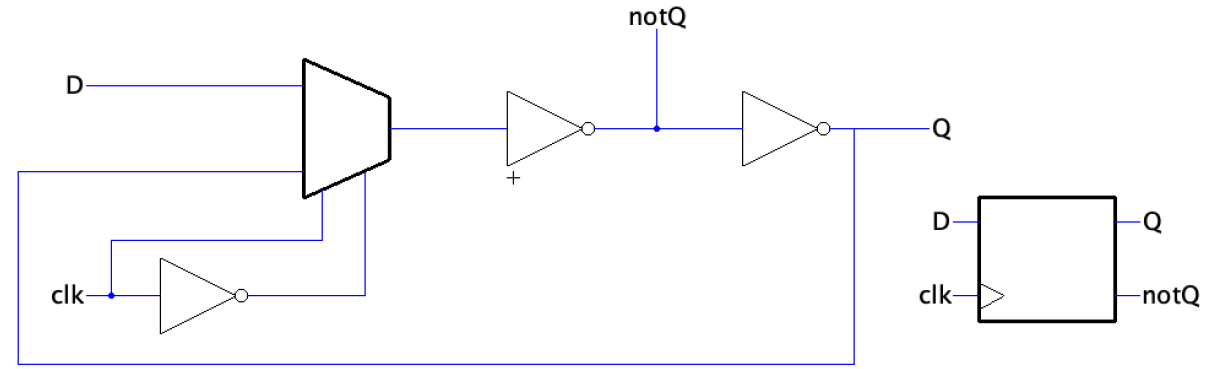
Arithmetic Unit for normal bits:

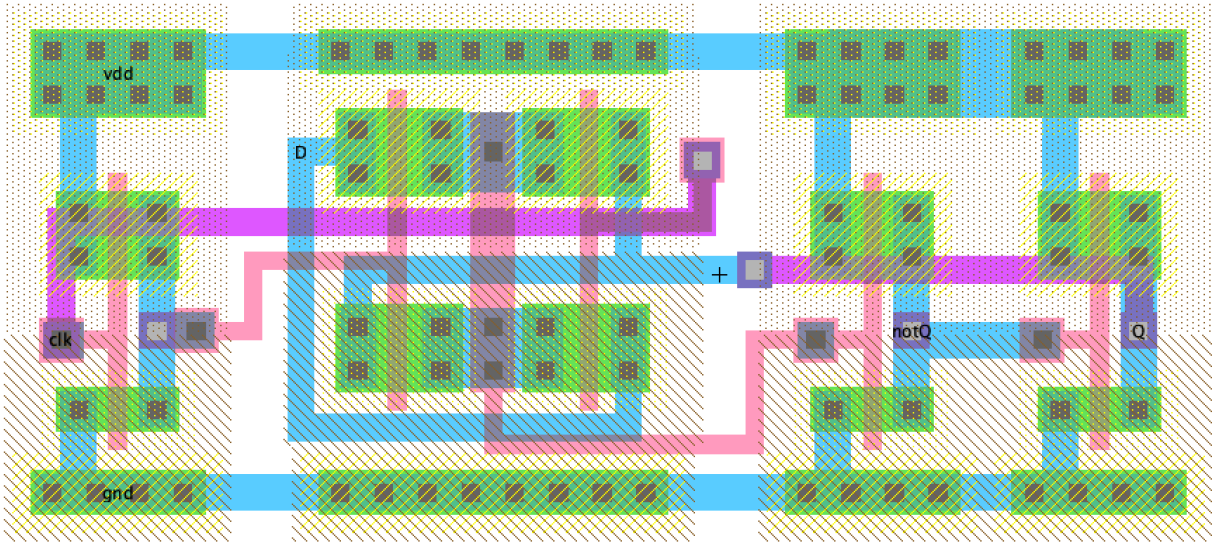


Arithmetic Unit for First Bit:

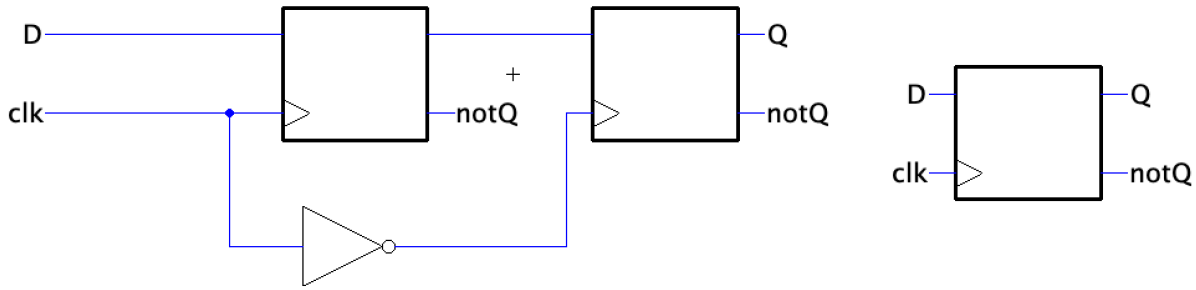


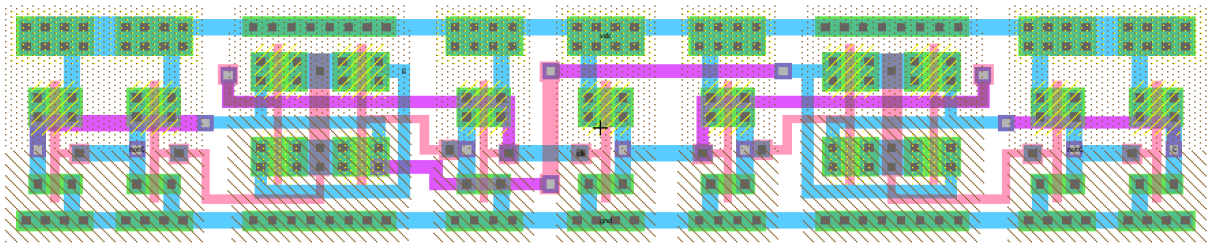
D-Latch Design:



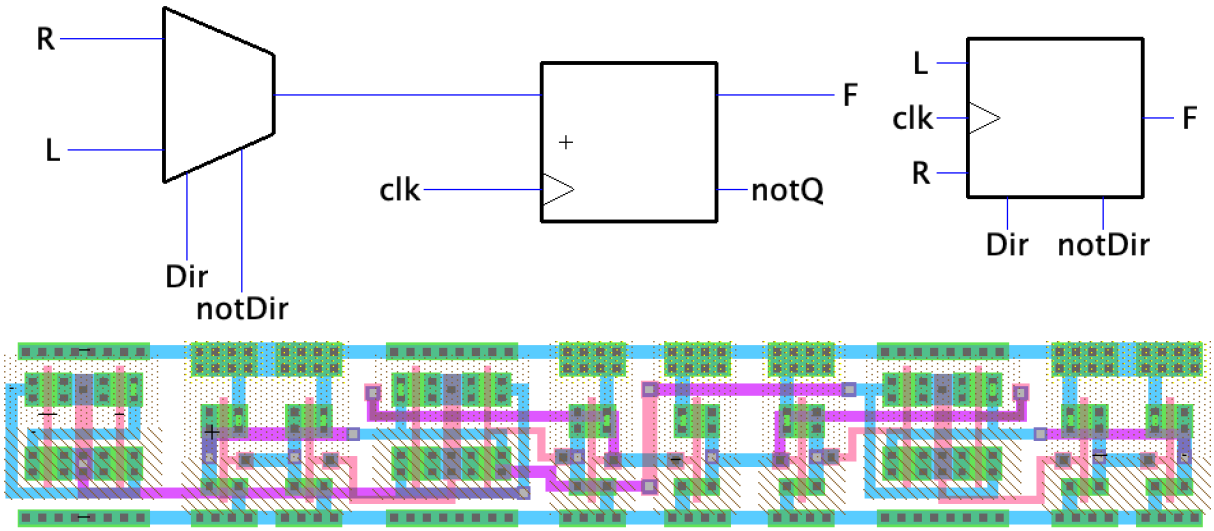


Edge Triggered D-FlipFlop Design:





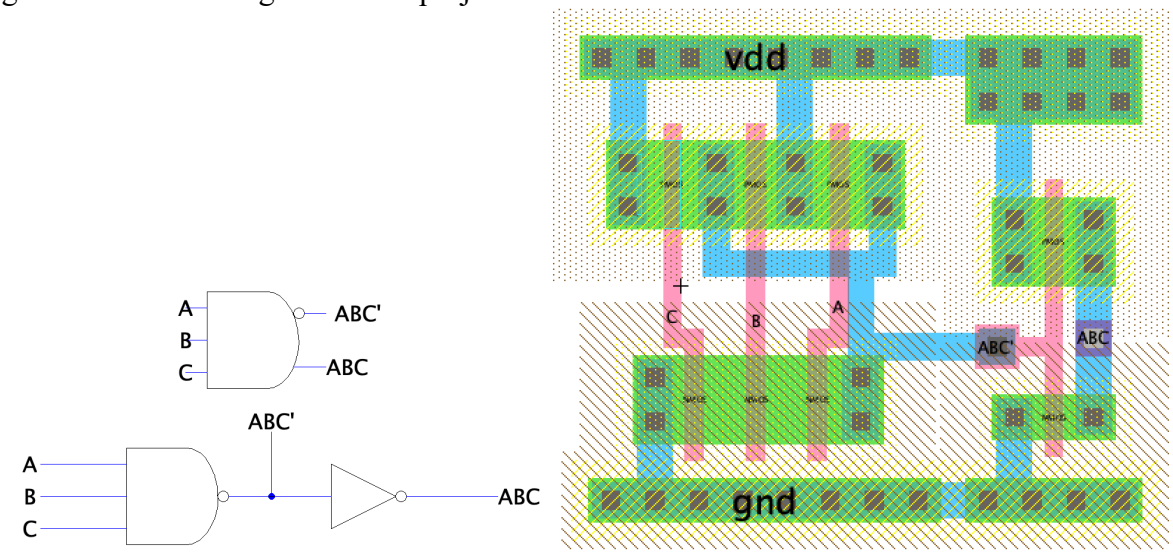
Shift Register Design:



Other Logic Designs for Midparts:

Dual Output Gates:

Putting output exports before inverting the signal does the job. I used NAND/AND gates and NOR/OR gates in this project.



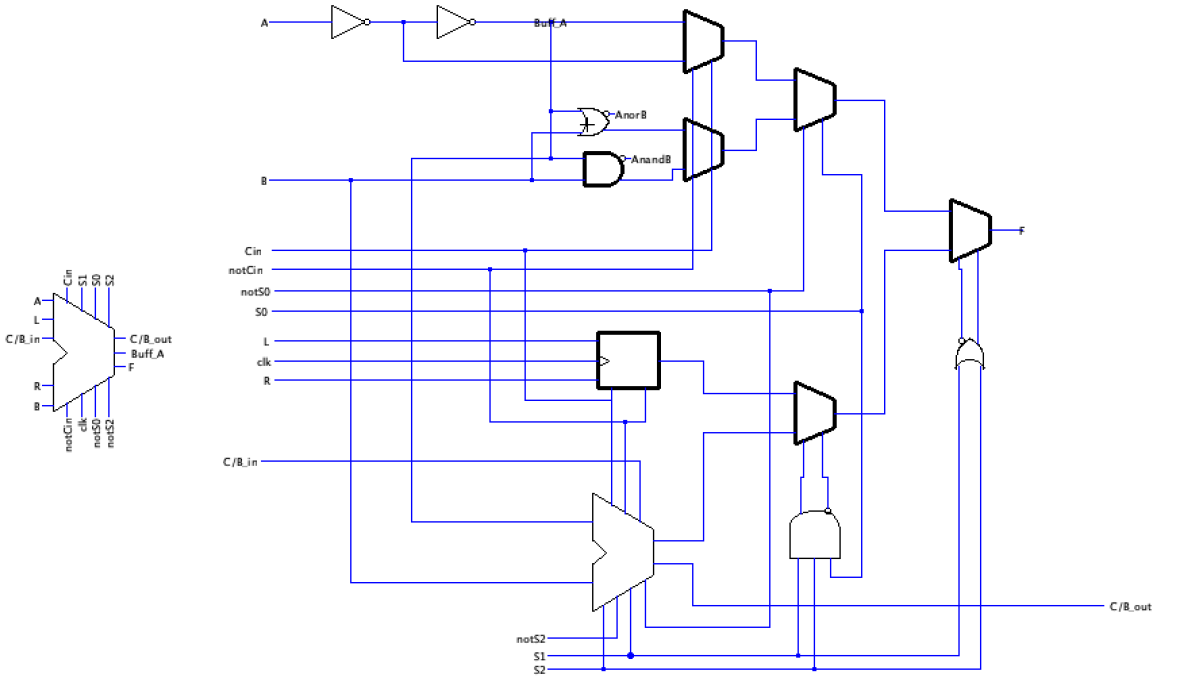
Three Input Gates:

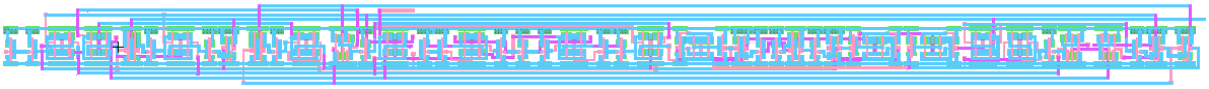
Logic gates can have more than two inputs. Adding extra inputs increases number of transistors and size of the gate. But it is efficient than cascading multiple gates, so I used 3 input gates on some of my designs.

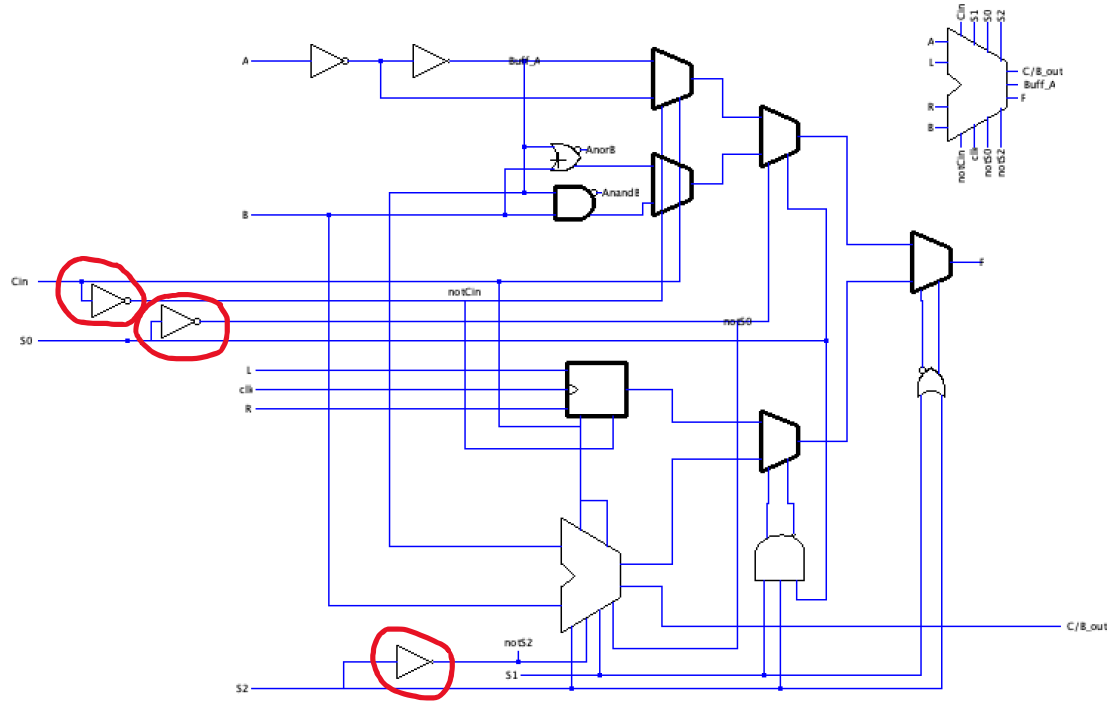
Gates With Different Layouts:

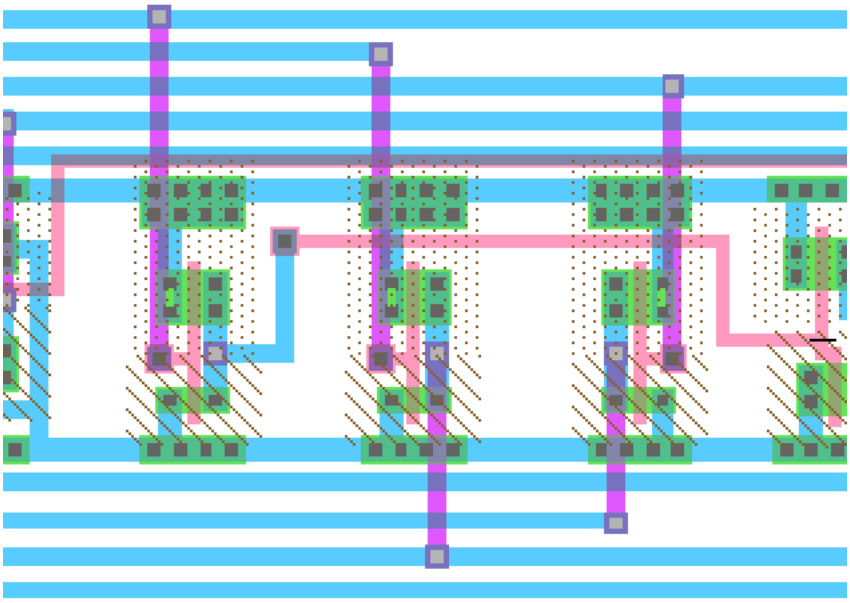
To improve overall designs, I sometimes change the layout of modules that I designed earlier. Sometimes I change the location of the input and output ports, sometimes I change the size and path of a line.

1-Bit ALU Design:

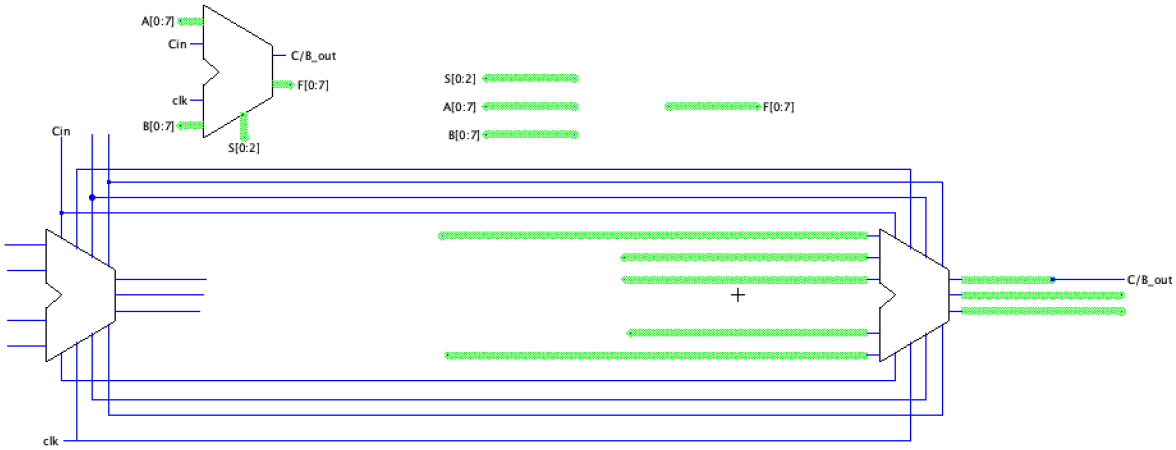


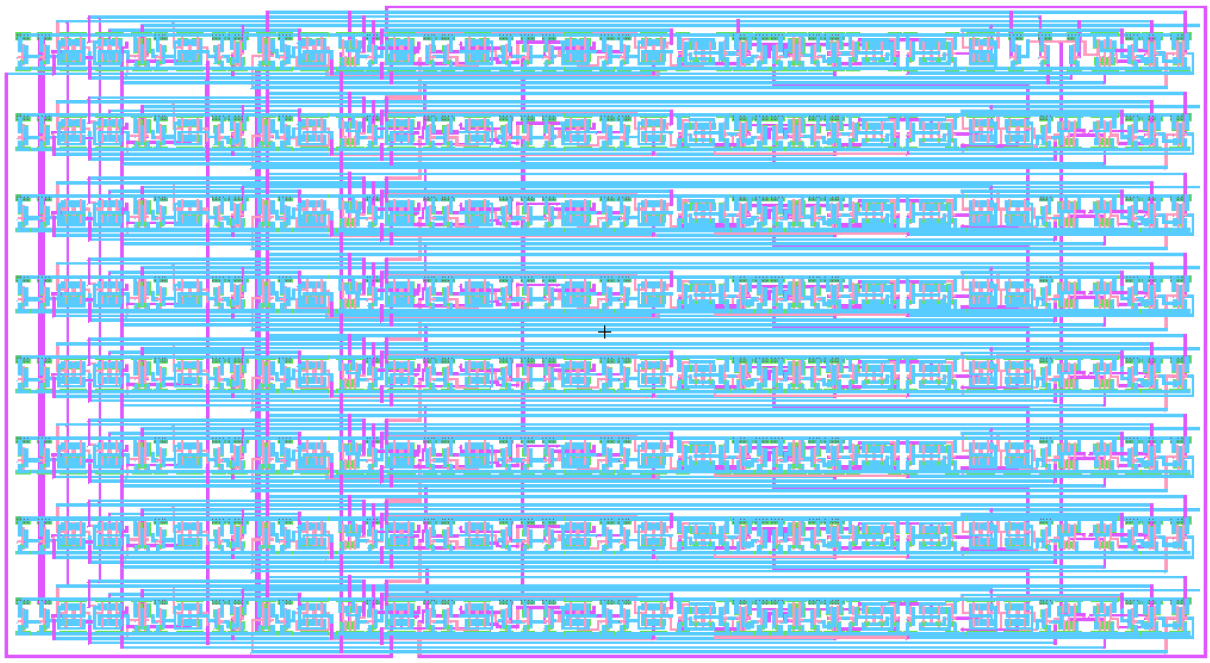




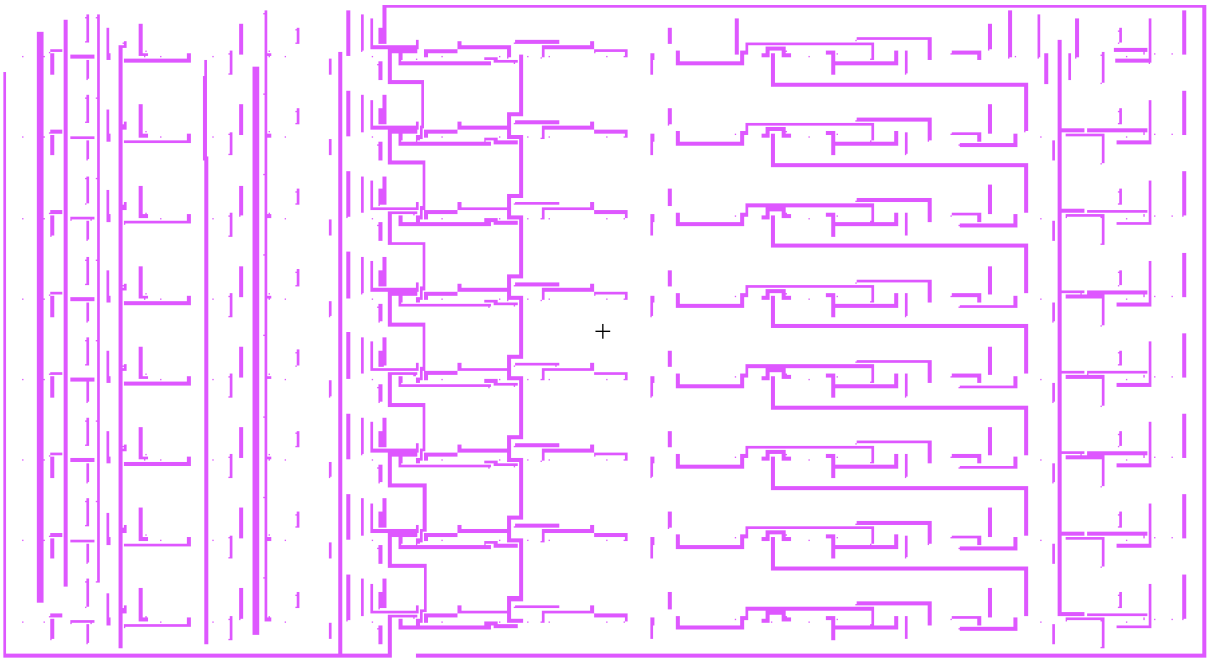


Scaling ALU to 8 BIT:





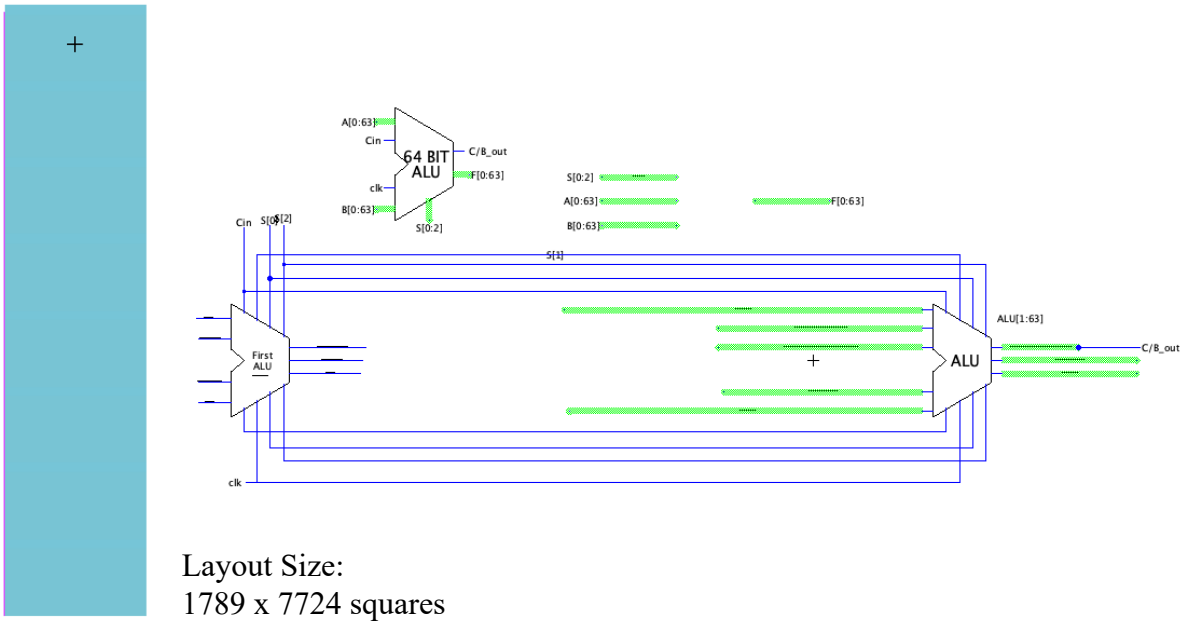
I used metal 2 only when I needed while designing submodules. The Metal 2 is used to connect common signals like Selection bits as long vertical lines. Metal 2 layer is shown below and long vertical lines carry power lines and common signals. Mostly horizontal lines are used inside of submodules to connect internal signals.



With this design, I am able to fit the whole 8 bit ALU inside of **1789x969.5** squares. It can be shrinkable by adding a third layer of metal and using common signals jointly. But for this project the size is small enough.

Scaling ALU to 64 BIT:

I applied same scaling process up to 64 bits. Although I didn't complete the routing, but process is straight forward. You can examine the layout of 64bit version ALU in the library. The screenshot of the 64 bit ALU is shown even nothing can be distinguished.

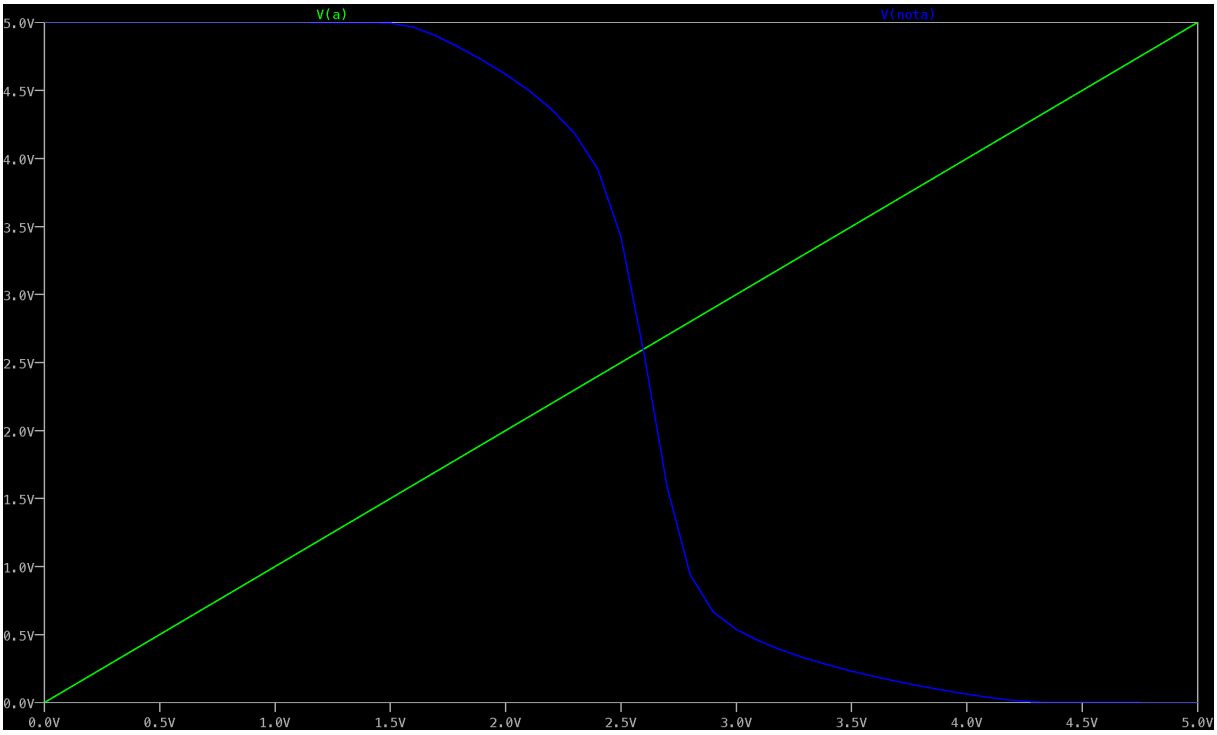


Layout Size:
1789 x 7724 squares

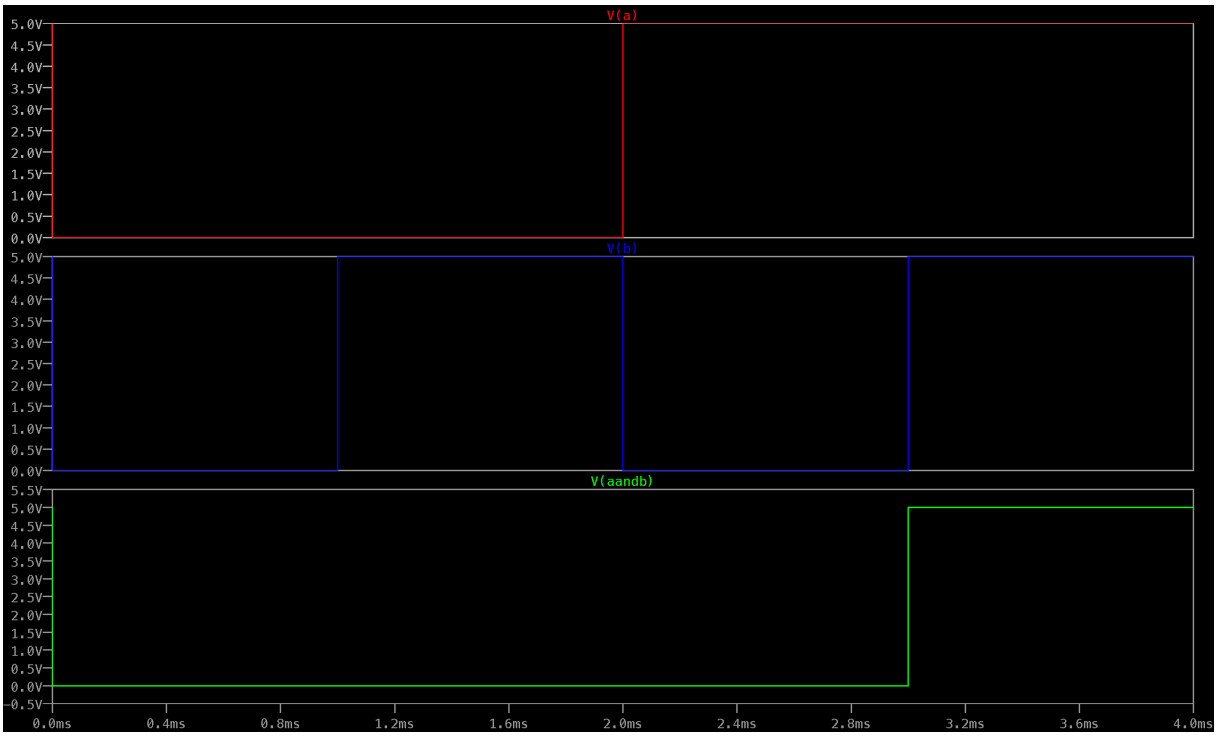
Simulation:

Although I designed 64 bit version of the ALU, The simulating whole circuit is same with 8 bit one. So I am simulating all operations on 8 bit version. All facts are valid for 64 bit version.

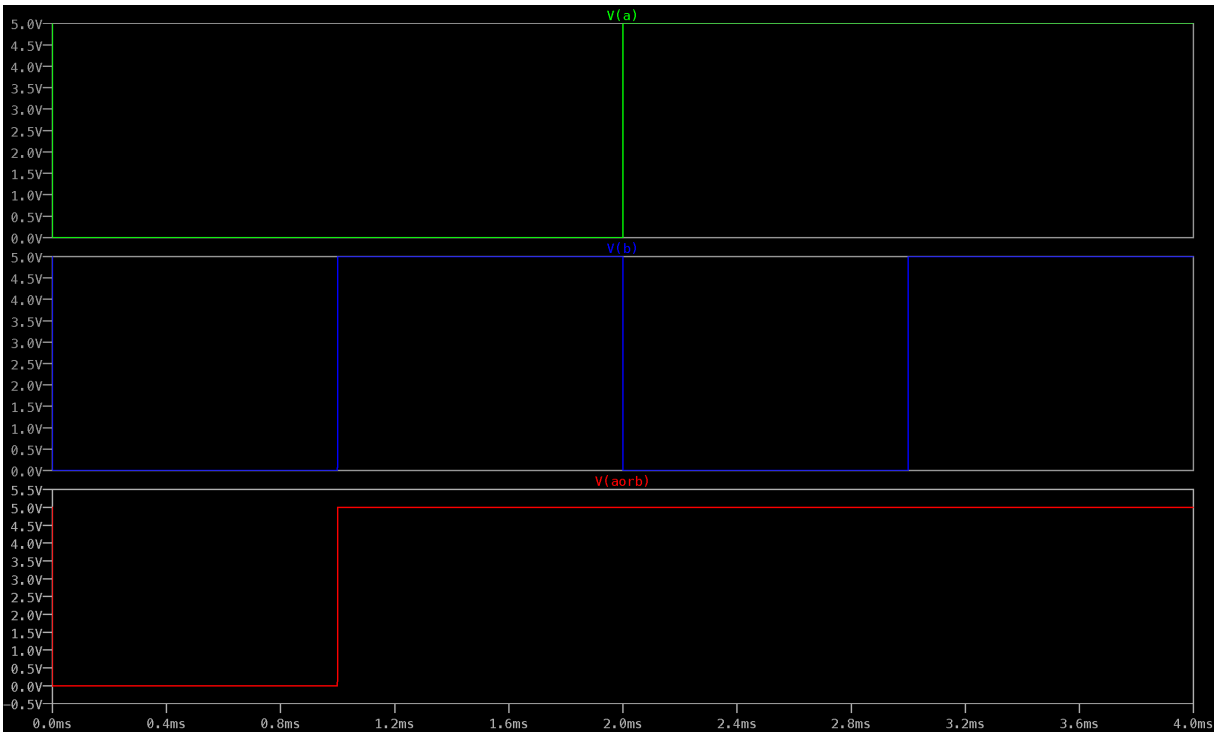
Inverter Simulation:



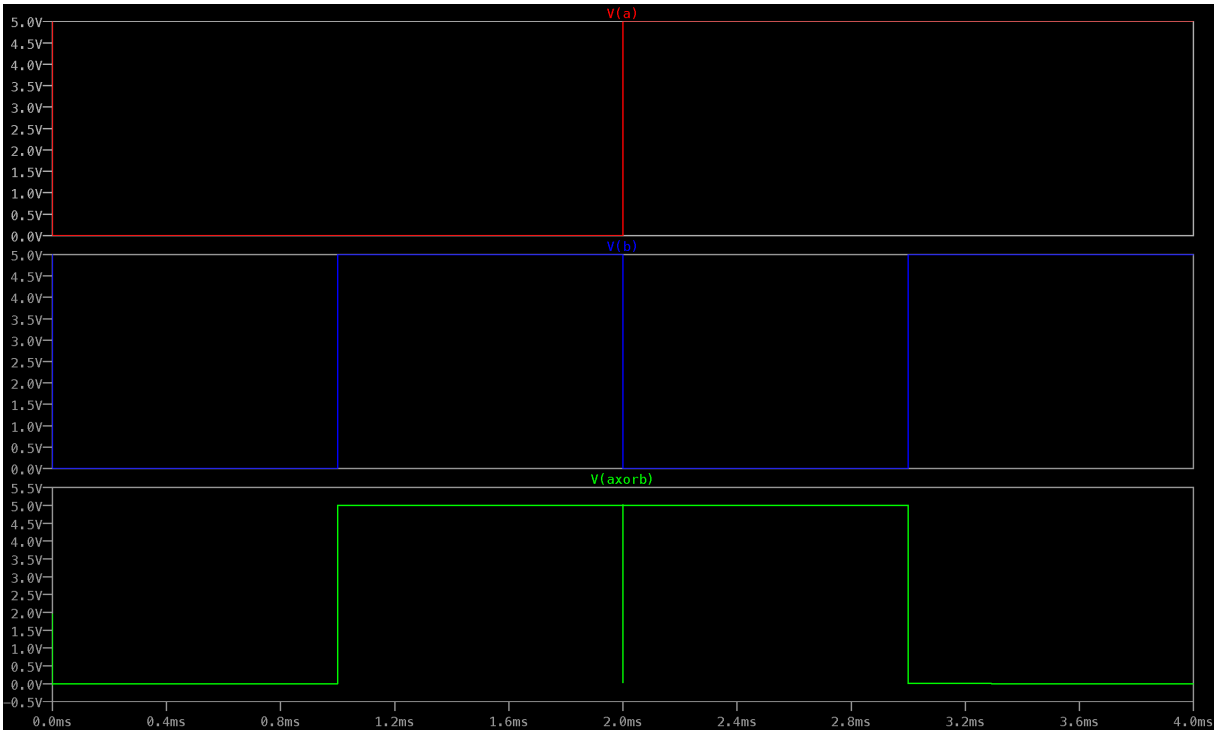
And Gate Simulation:



Or Gate Simulation:



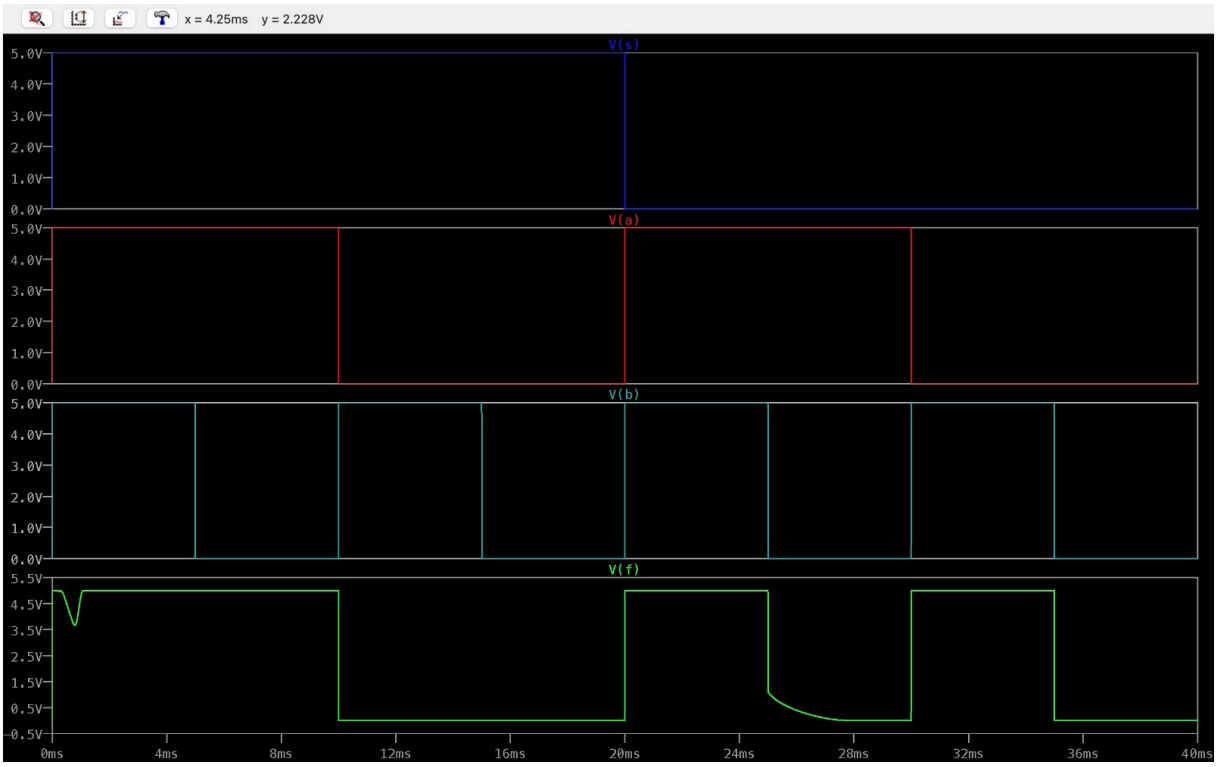
Xor Gate Simulation:



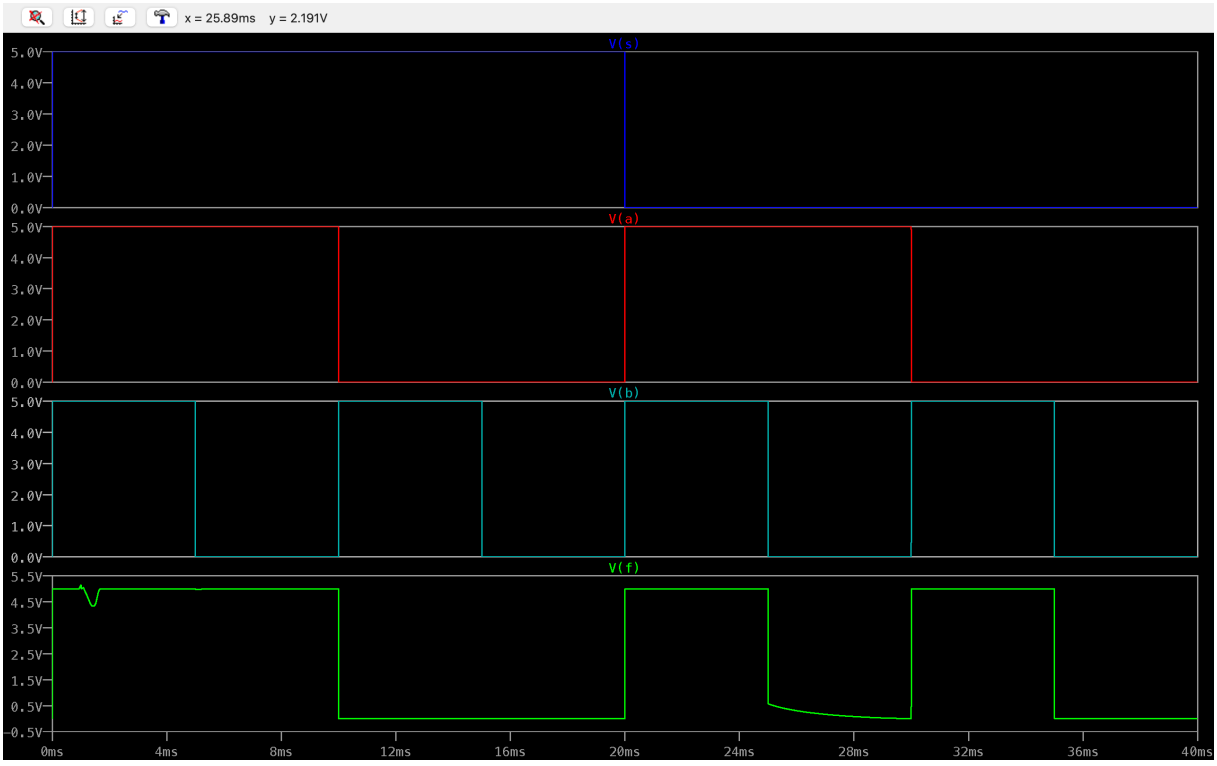
Multiplexer Simulation:

2x1 transmission mux spice sim results:

10:5

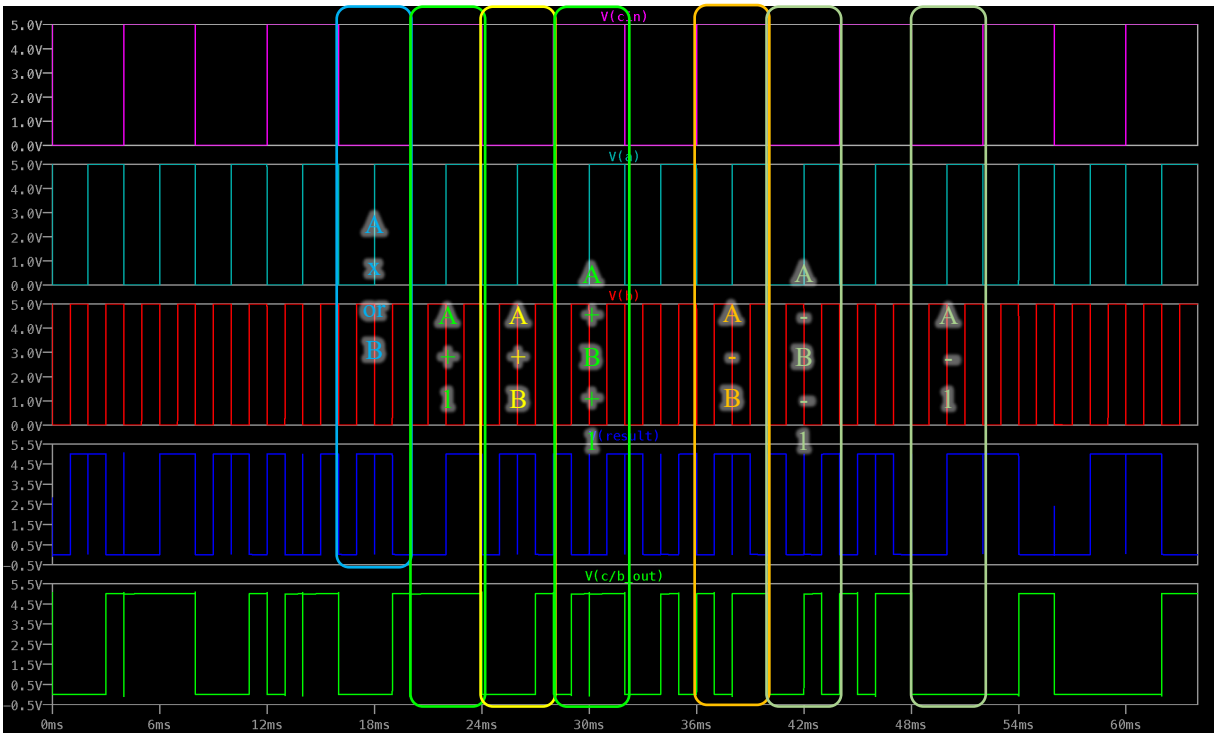


10:10

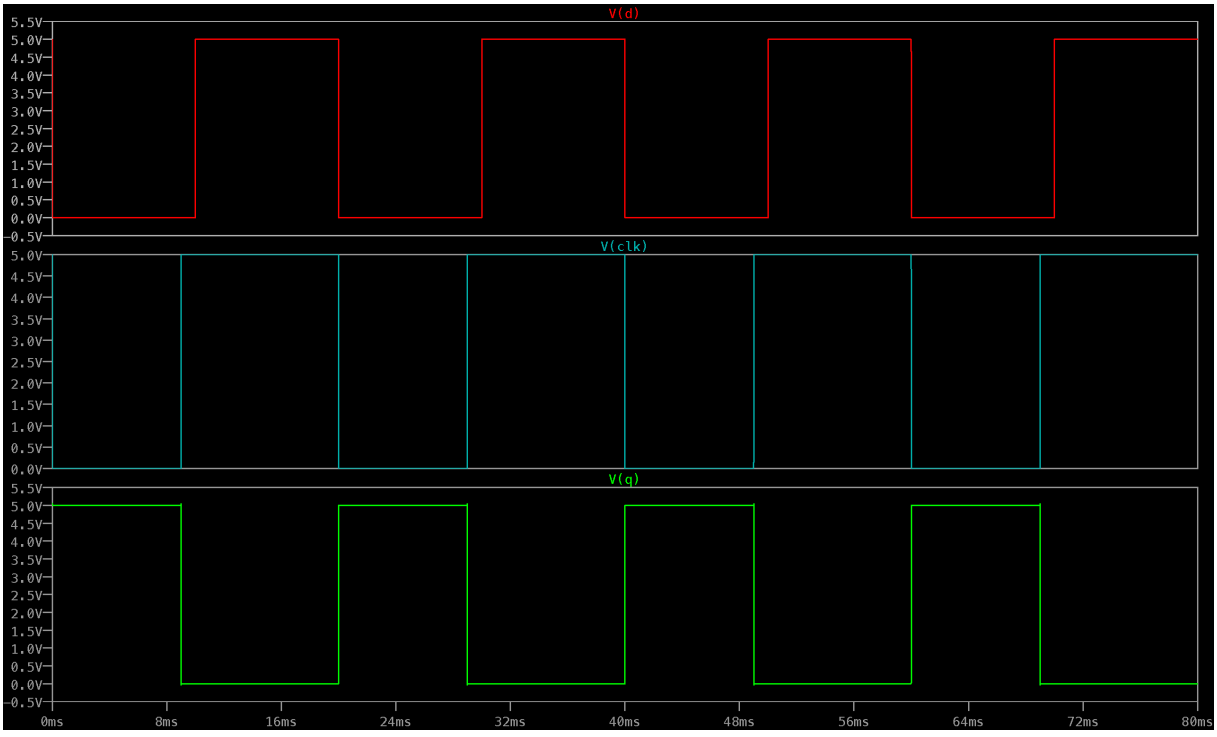


To prevent ripples and transition delays on 10:5 transistor sizing, I used 10:10 ratio by simulation results.

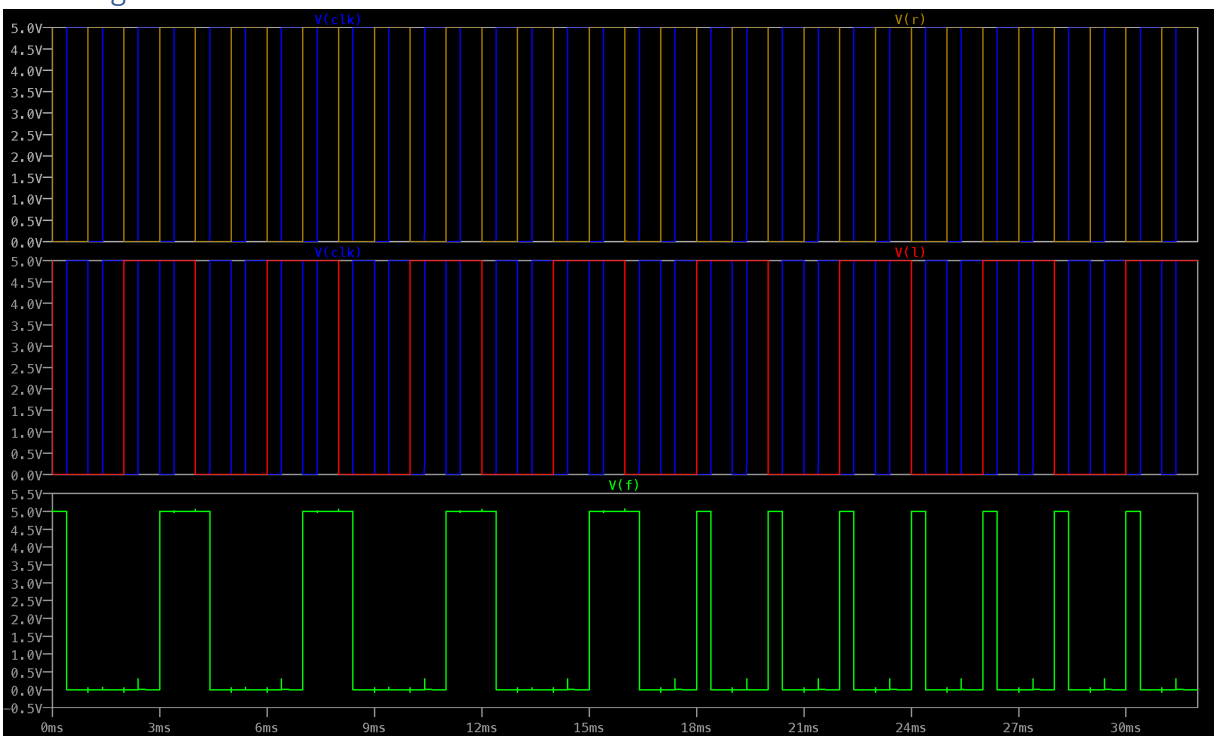
Arithmetic Unit Simulation:



Edge Triggered D-FlipFlop Simulation:

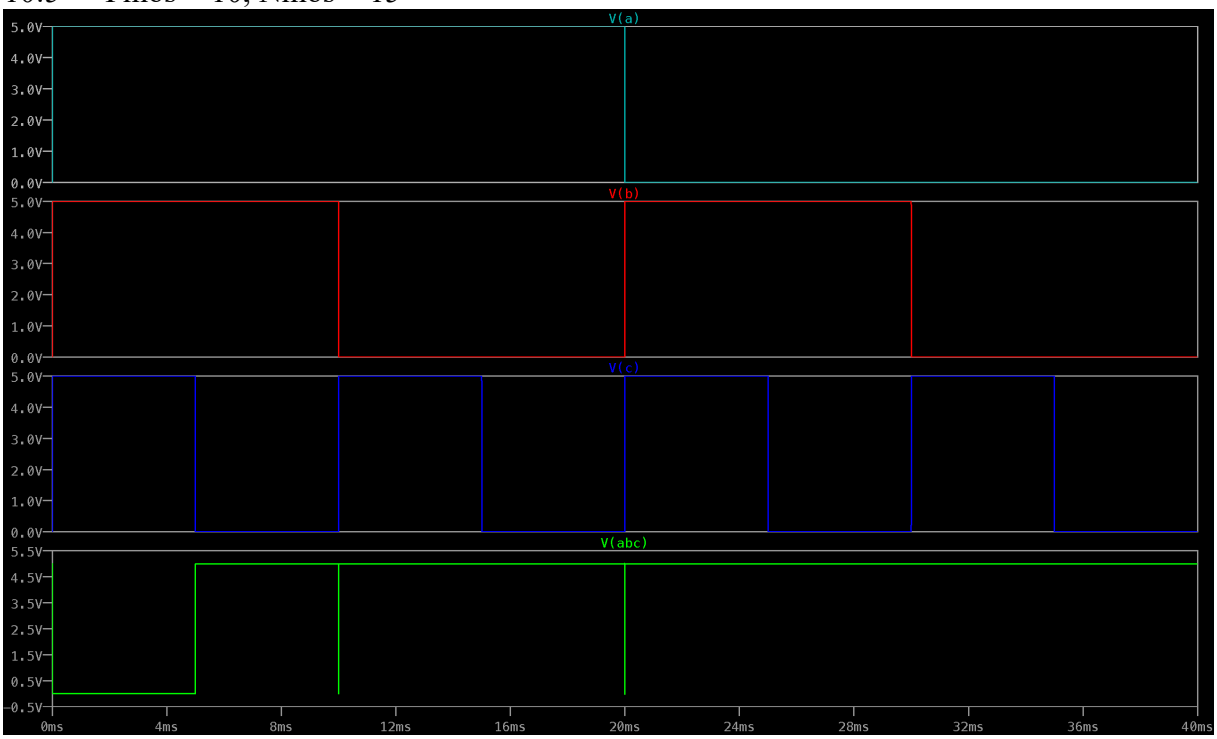


Shift Register Simulation:

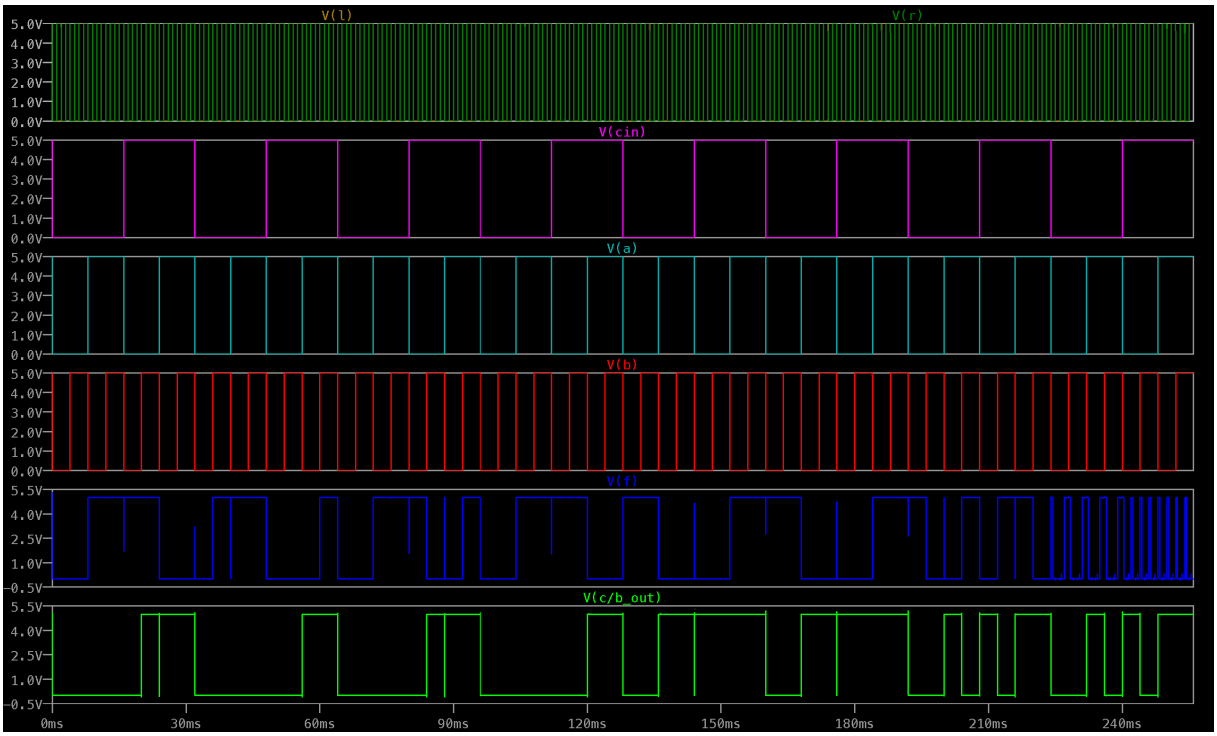


Other Gate Simulations:

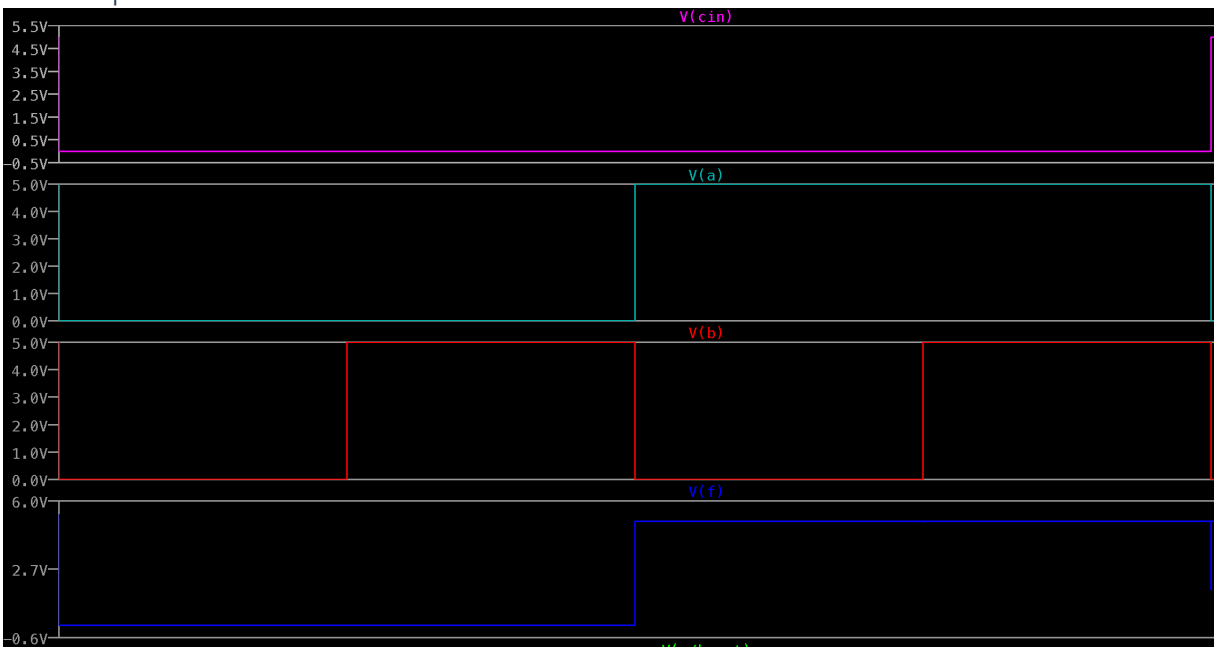
3 input Nand gate spice simulation results:
10:5 -> Pmos = 10, Nmos = 15



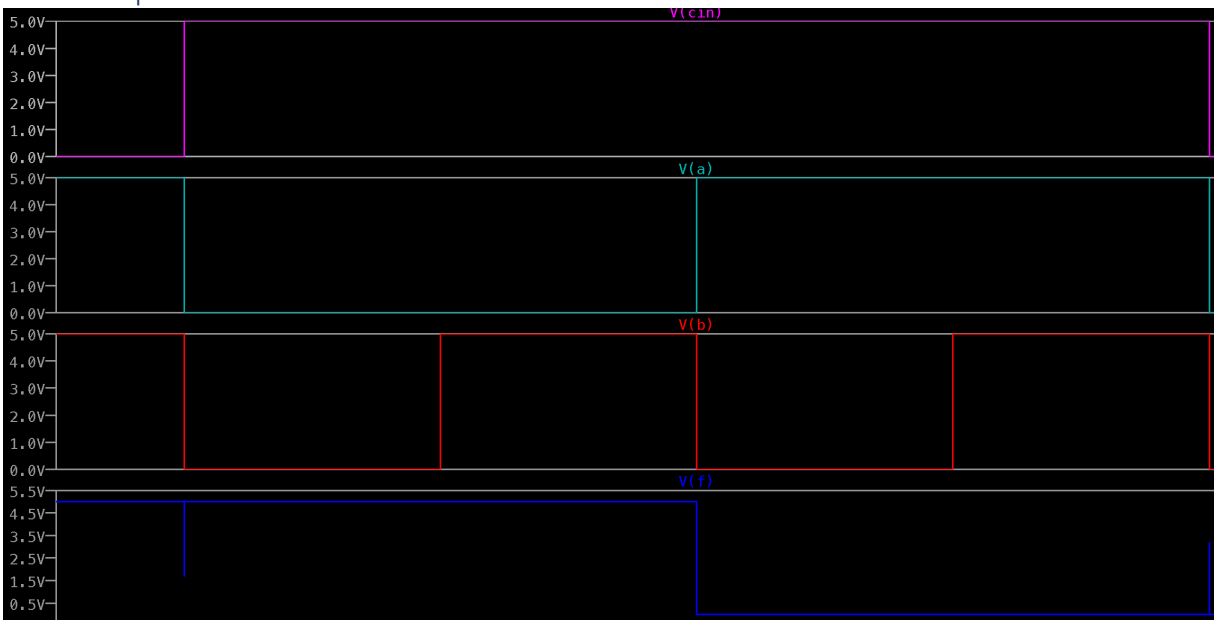
1-Bit ALU Simulation:



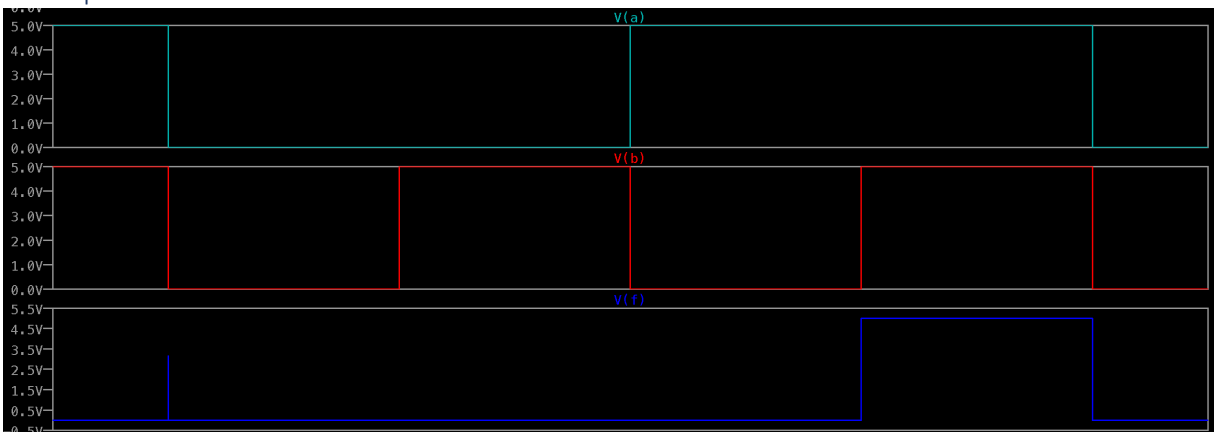
Buffer Operation



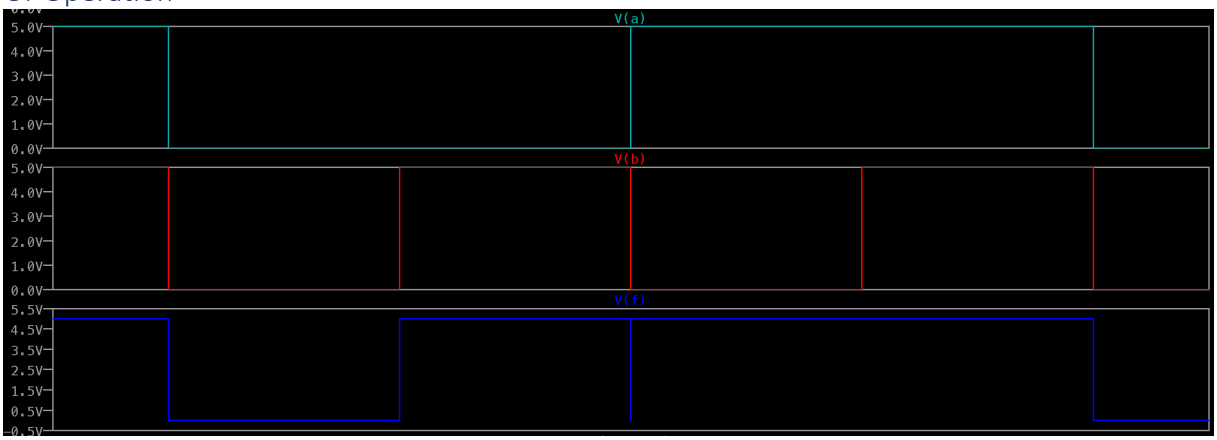
Inverter Operation



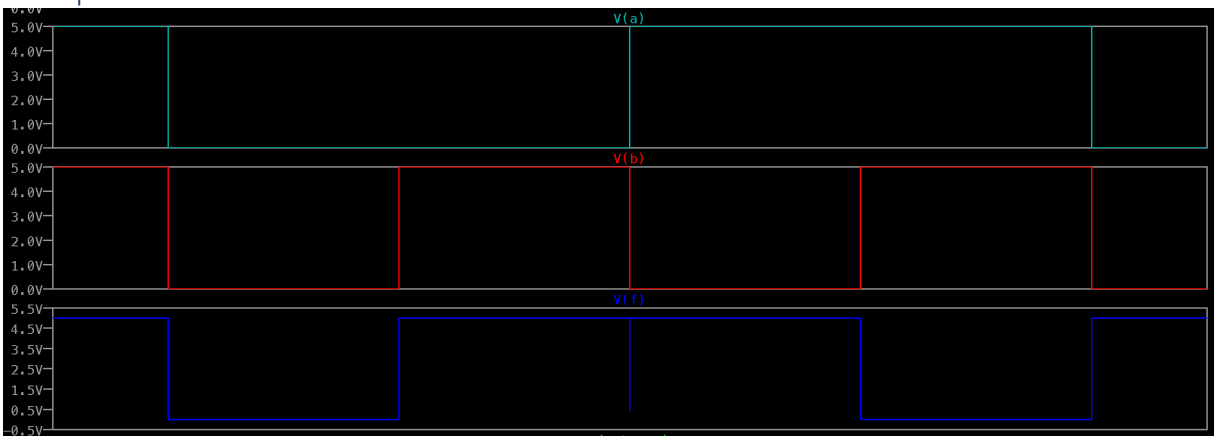
And Operation



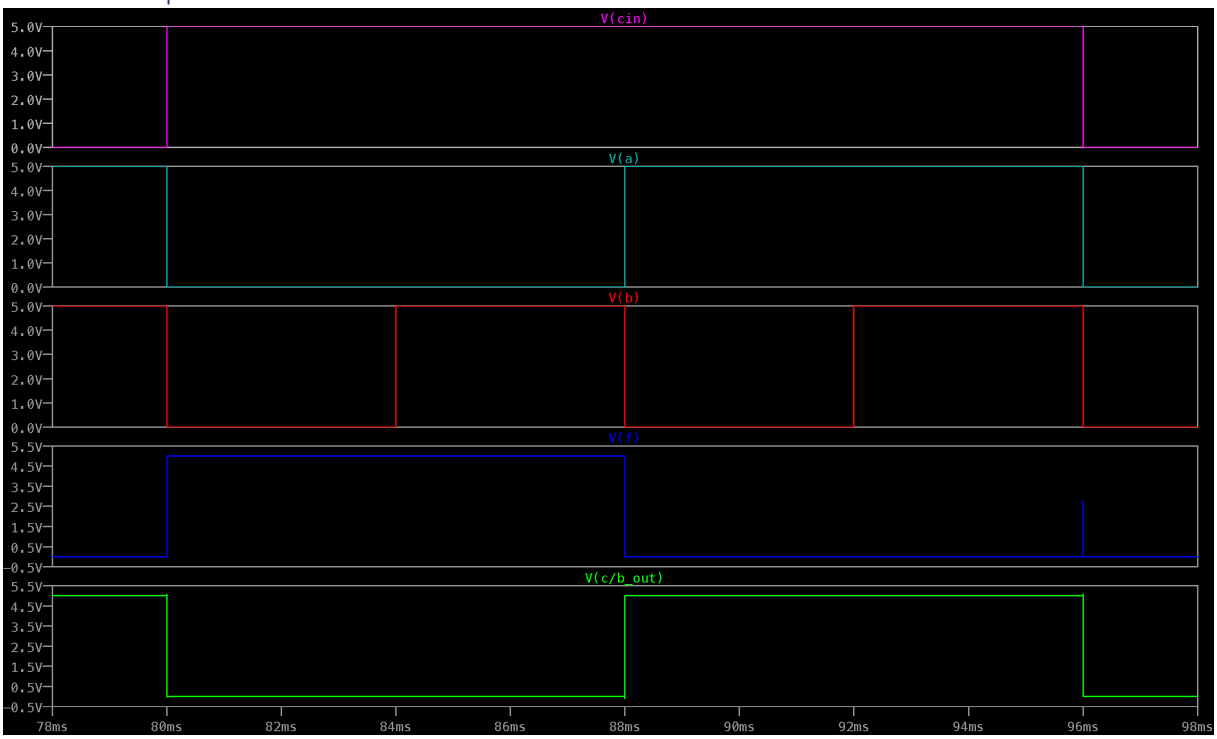
Or Operation



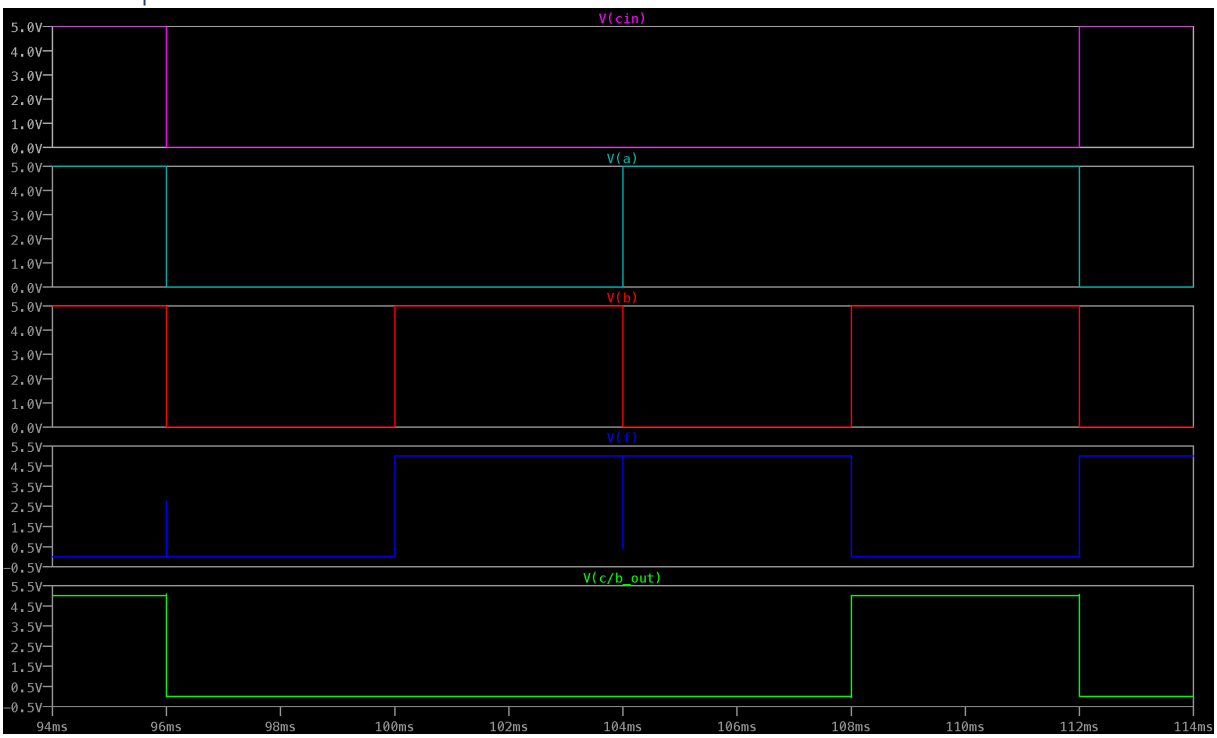
Xor Operation



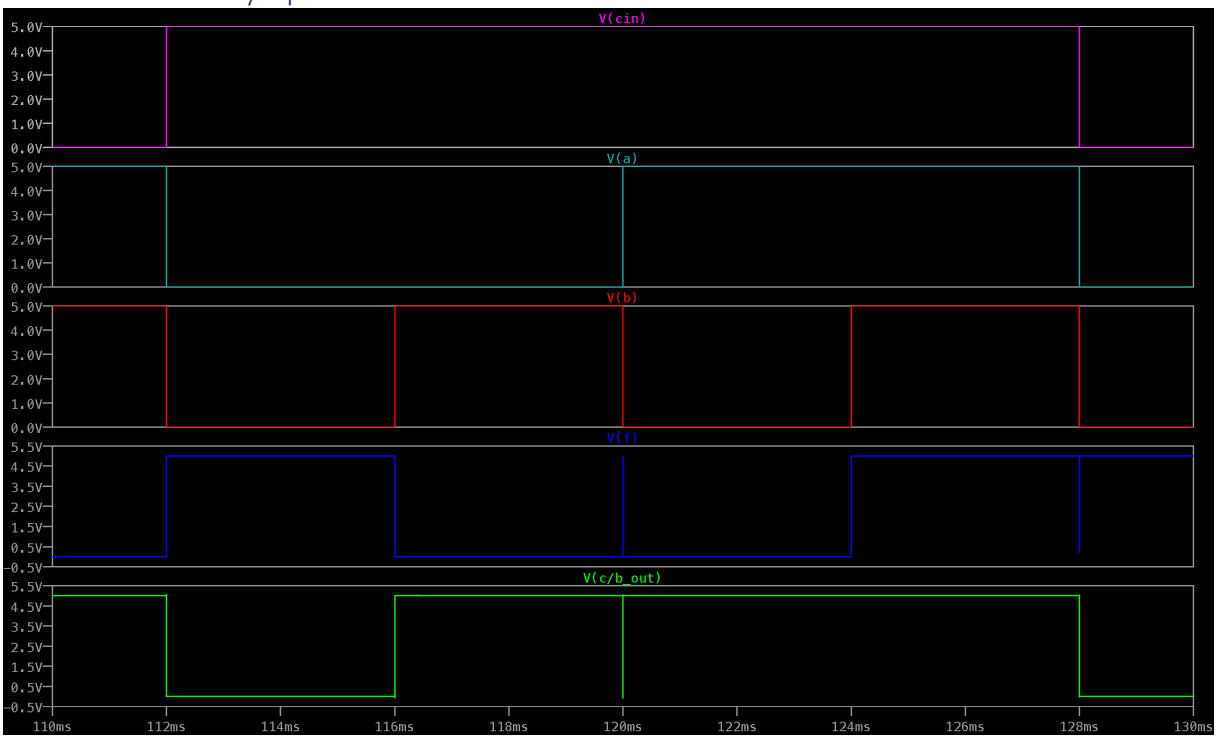
Increment Operation



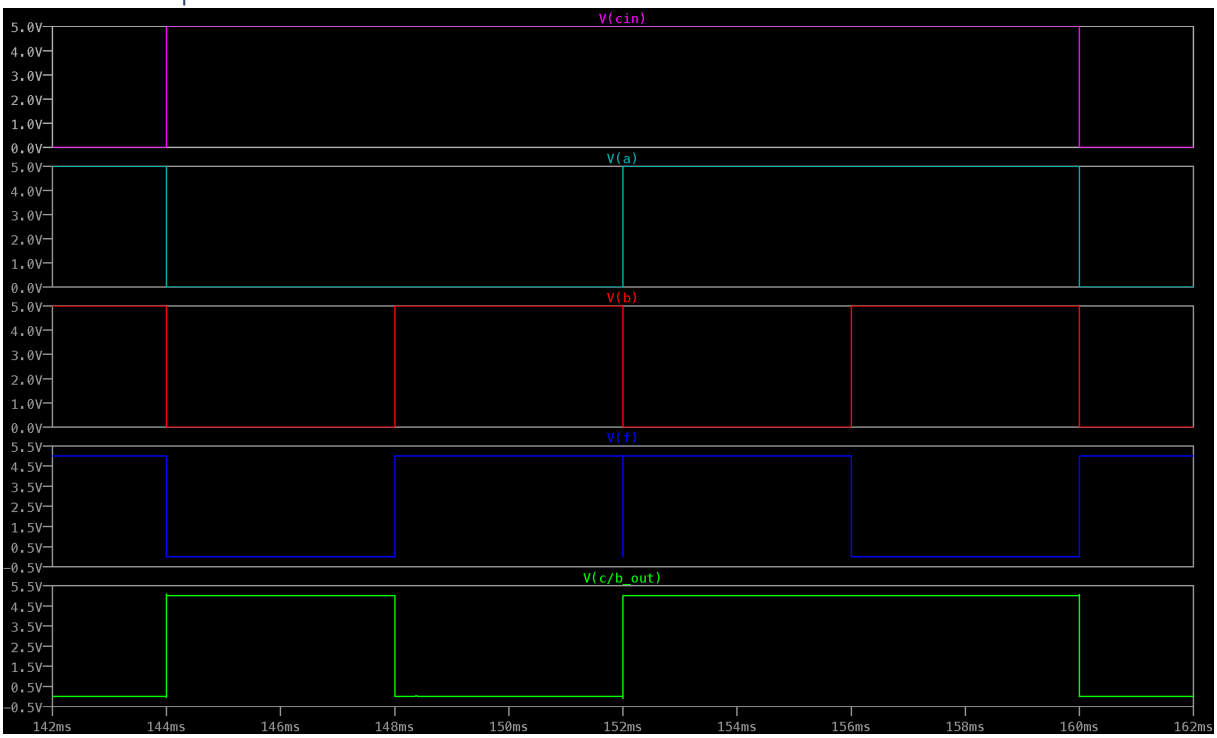
Addition Operation



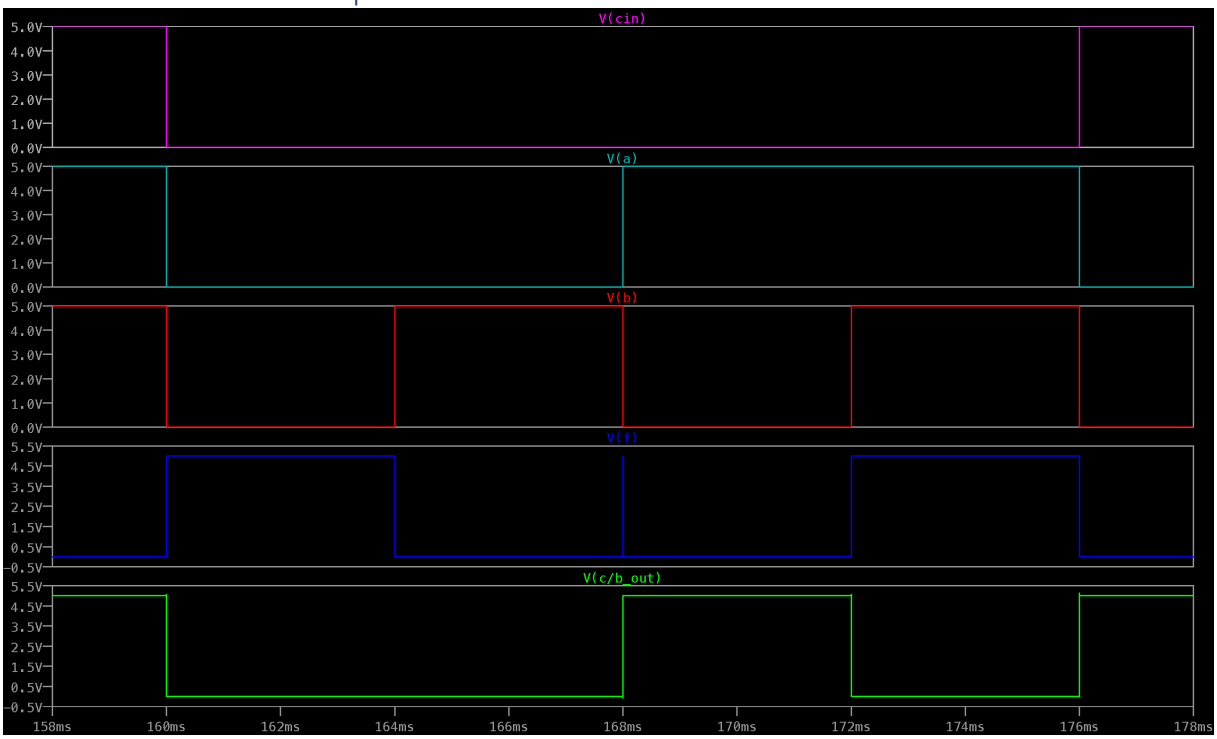
Addition With Carry Operation



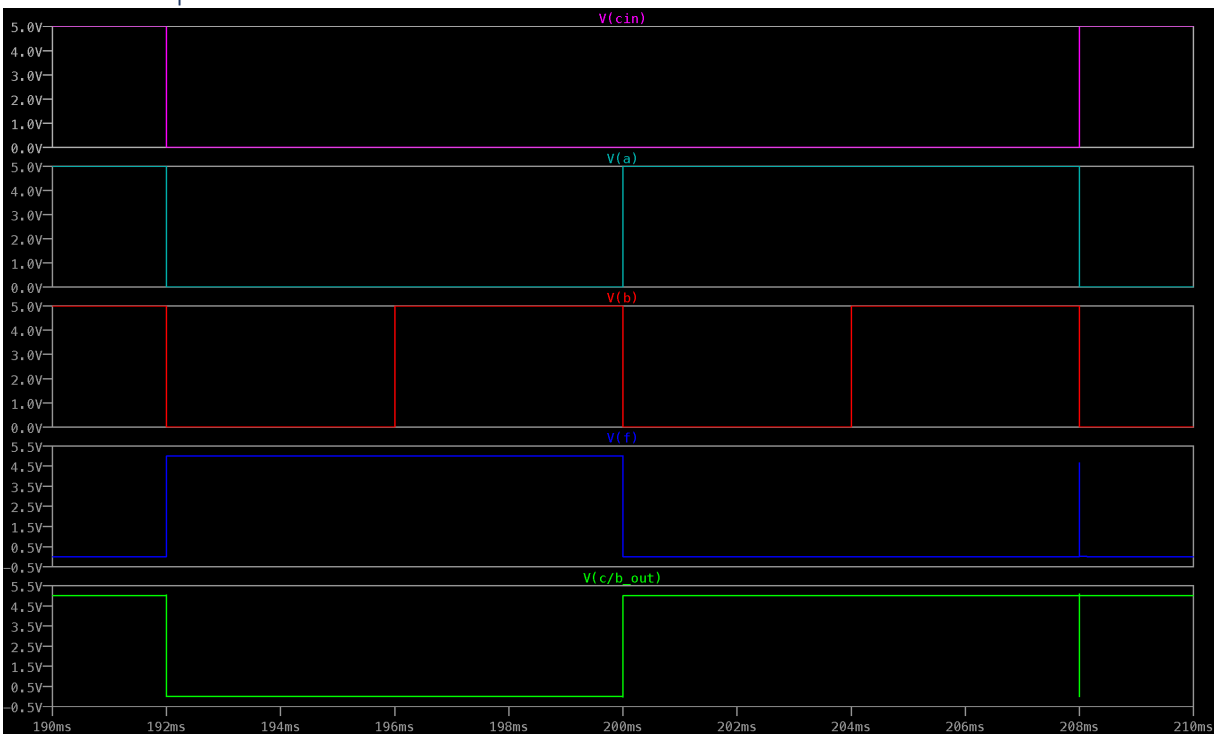
Subtraction Operation



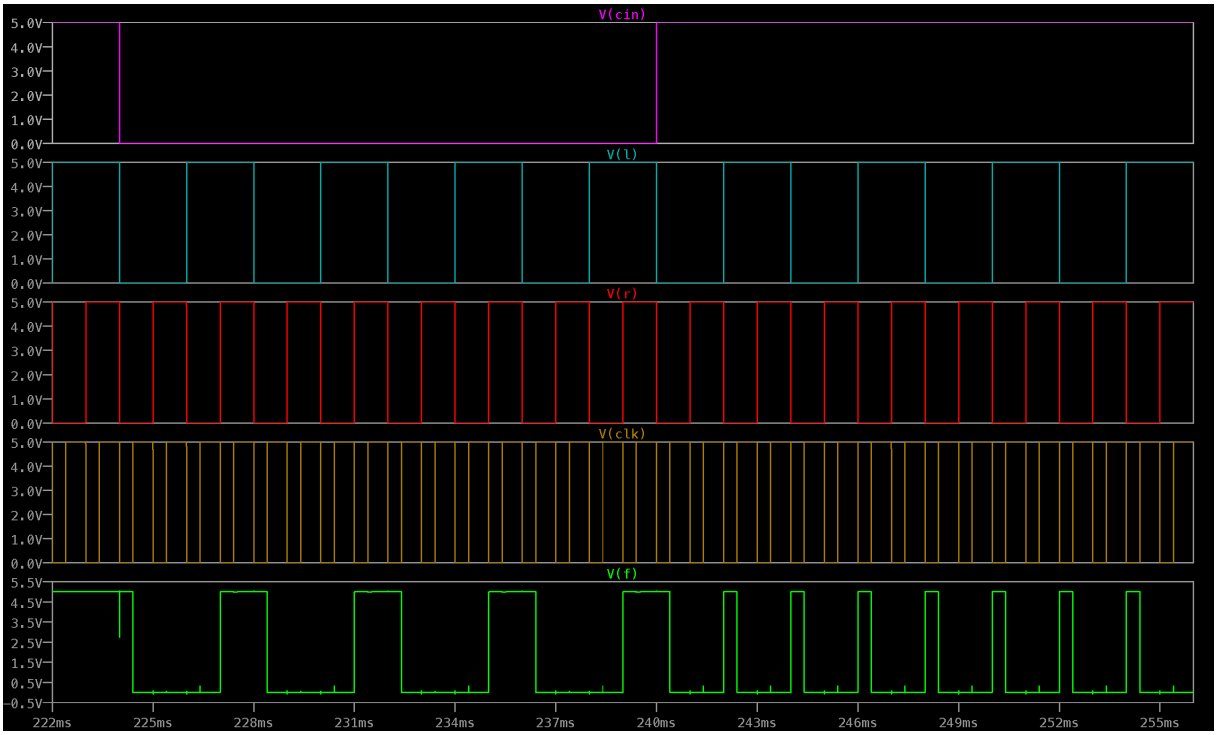
Subtraction With Borrow Operation



Decrement Operation



Shift Operations



8Bit Overall ALU Simulation:

Simulation Spice Code:

```
* SUPPLY VOLTAGE
vdd vdd 0 dc 5
* OPERATION SELECTIONS
vs2 S[2] 0 dc 0 pulse 5 0 1n 1n 1n 8m 16m
vs1 S[1] 0 dc 0 pulse 5 0 1n 1n 1n 4m 8m
vs0 S[0] 0 dc 0 pulse 5 0 1n 1n 1n 2m 4m
vcin Cin 0 dc 5 pulse 5 0 1n 1n 1n 1m 2m
* INPUT A <- 00000110
va0 A[0] 0 DC 0
va1 A[1] 0 DC 5
va2 A[2] 0 DC 5
va3 A[3] 0 DC 0
va4 A[4] 0 DC 0
va5 A[5] 0 DC 0
va6 A[6] 0 DC 0
va7 A[7] 0 DC 0
* INPUT B <- 00001001
vb0 B[0] 0 DC 5
vb1 B[1] 0 DC 0
vb2 B[2] 0 DC 0
vb3 B[3] 0 DC 5
vb4 B[4] 0 DC 0
vb5 B[5] 0 DC 0
vb6 B[6] 0 DC 0
vb7 B[7] 0 DC 0
* CLK FOR SHIFT OPERATION
vclk clk 0 DC 0 pulse 5 0 1n 1n 1n 0.9m 2m
.tran 0 16m
.include /Applications/electric_vlsi/process/C5_models.txt
```

I used square wave with different periods to perform each operation in ALU. The operations occurred respectively in the waveform. Input signals are constant 8 bit signals. Where:

A <= 00000110b (6)

B <= 00001001b (9)

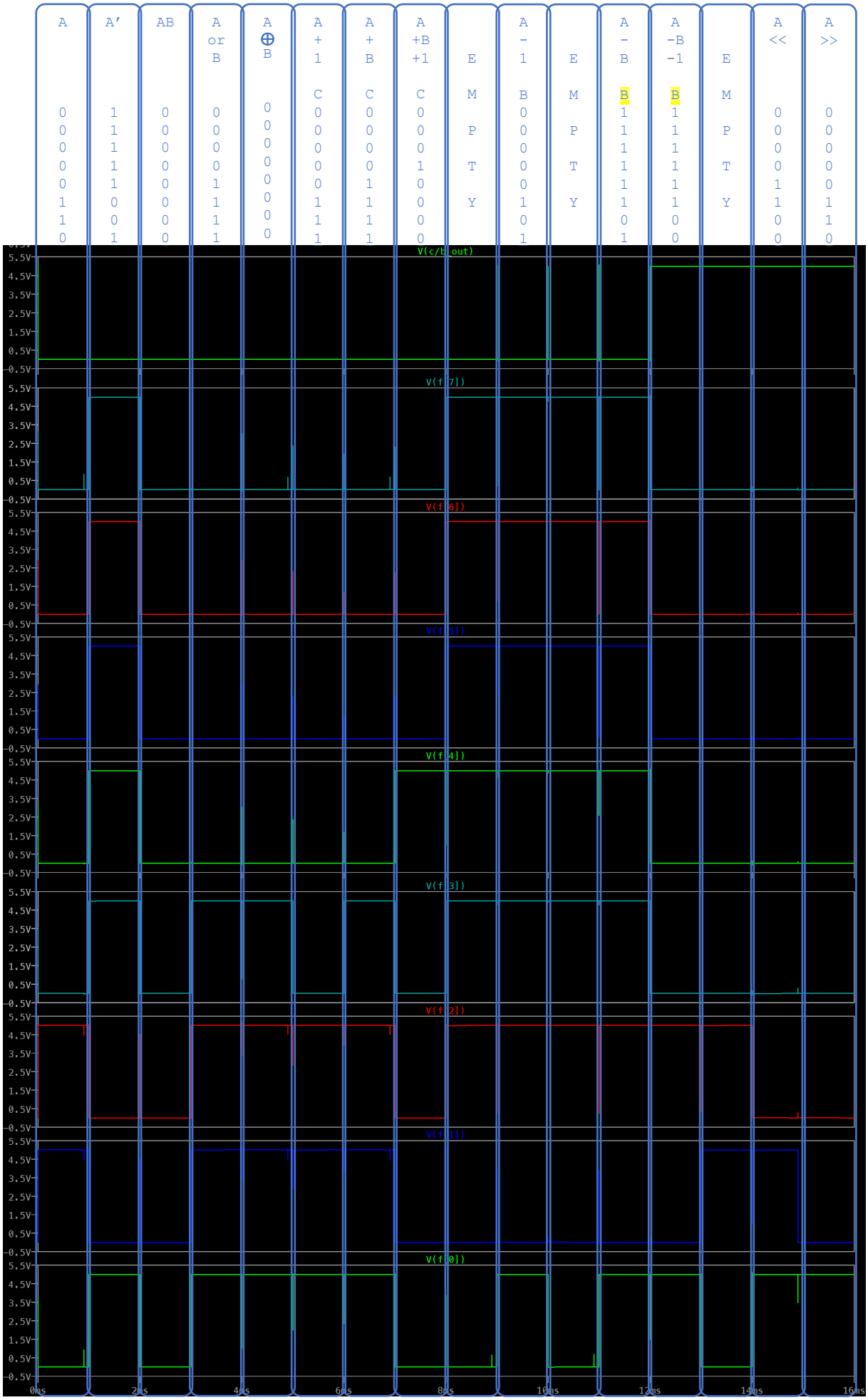
⊕ = XOR

C = No carry occurs

C = Carry occurs

B = No borrow occurs

B = Borrow occurs


A
0
0
0
0
1
1
0
A'
1
1
1
1
0
0
1
AB
0
0
0
0
0
0
0
A or B
0
0
0
0
1
1
1
A ⊕ B
0
0
0
0
0
0
0
A + 1
C
0
0
0
0
1
1
1
A + B
C
0
0
0
0
1
1
1
A +B +1
C
0
0
0
1
0
0
0
E M P T Y
A - 1
B
0
0
0
0
1
0
1
E M P T Y
A - B
B
1
1
1
1
1
1
0
A -B -1
B
1
1
1
1
1
1
0
E M P T Y
A <<
0
0
0
0
1
1
0
0
A >>
0
0
0
0
1
1
0
0
V(c/B_out)
V(f[7])
V(f[6])
V(f[5])
V(f[4])
V(f[3])
V(f[2])
V(f[1])
V(f[0])
5.5V
4.5V
3.5V
2.5V
1.5V
0.5V
-0.5V
0ns
2ns
4ns
6ns
8ns
10ns
12ns
14ns
16ns