Ain Shams University, Faculty of Engineering,

ECE212: Digital Design

Arithmetic and Logical Unit Design Project

Table of Contents

ALU Code

```
--                              4-bit ALU

--      sel             Operation                               Unit
--      0000            increment a

--      0001            decrement a

--      0010            transfer b

--      0011            increment b                             Arithmetic
--      0100            decrement b

--      0101            transfer a

--      0110            add a and b

--      0111            multiply a by 2

--      1000            complement a (1s complement)

--      1001            complement b

--      1010            AND

--      1011            OR

--      1100            XOR                                     Logic
--      1101            XNOR

--      1110            NAND

--      1111            NOR

library ieee;
use ieee.std_logic_1164.all; use
ieee.std_logic_signed.all;
 entity ALU
is  port(
      a,b,sel: IN  STD_LOGIC_VECTOR(3 downto 0);
y : out STD_LOGIC_VECTOR(4 downto 0)
); end
ALU;

architecture Alu of Alu is
 begin
process(a,b,sel)

Variable yav : STD_LOGIC_VECTOR(4 downto 0);
-- variable to hold the output in arithmetic operations
 variable ylv : STD_LOGIC_VECTOR(3 downto
0);
-- variable to hold the output in logic operations

variable temp : STD_LOGIC_VECTOR(4 downto 0);

begin
```

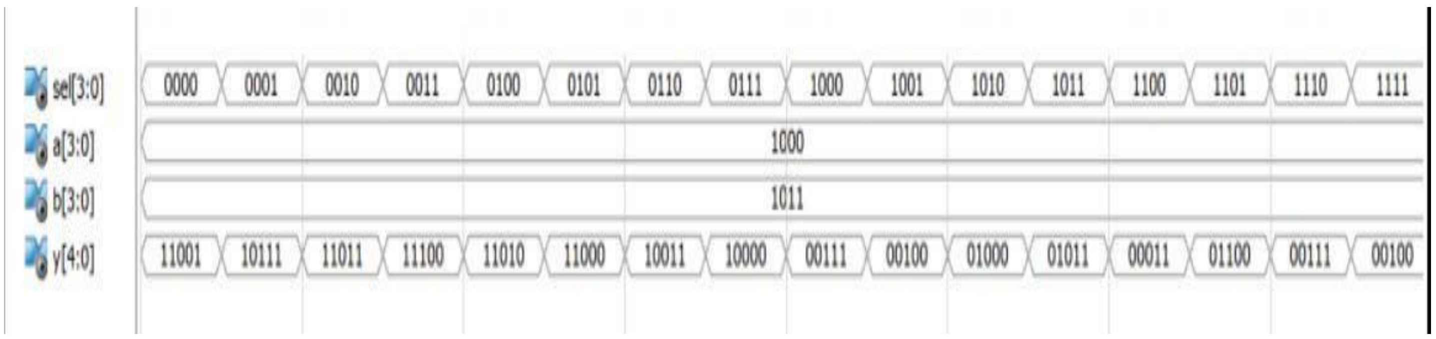
```
                    -- Arithmetic Operations

      if (sel = "0000") then                   --Increment a
              temp(4 downto 0) := ("00000" + a + "0001");
              yav := temp;



     elsif (sel = "0001") then    -- decrement a          temp(4
downto 0) := ("00000" + a + "1111");
              yav := temp;

      elsif (sel = "0010") then                -- transfer b
              yav := ("00000" + b);

      elsif (sel = "0011") then                -- increment b
              temp(4 downto 0) := ("00000" + b + "0001");
              yav := temp;

     elsif (sel = "0100") then            -- decrement b
temp(4 downto 0) := ("00000" + b + "1111");
              yav := temp;

      elsif (sel = "0101") then                -- transfer a
              yav := "00000" + a;

      elsif (sel = "0110") then                -- Add a and b
              temp(4 downto 0) := ("00000" + a + b);
              yav := temp;

      elsif (sel = "0111") then    -- multiply (a) by 2 or shift left 1 bit
              temp(4 downto 0) := ("00000" + a + a);
              yav := temp;



                    -- Logic Operations

      elsif (sel = "1000") then    -- 1's complement of a (invert a)
              ylv := NOT a;

      elsif (sel = "1001") then    -- 1's complement of b (invert b)
              ylv := NOT b;

     elsif (sel = "1010") then    -- a AND b
     ylv := (a AND b);

     elsif (sel = "1011") then    -- a OR b
ylv := (a OR b);
```

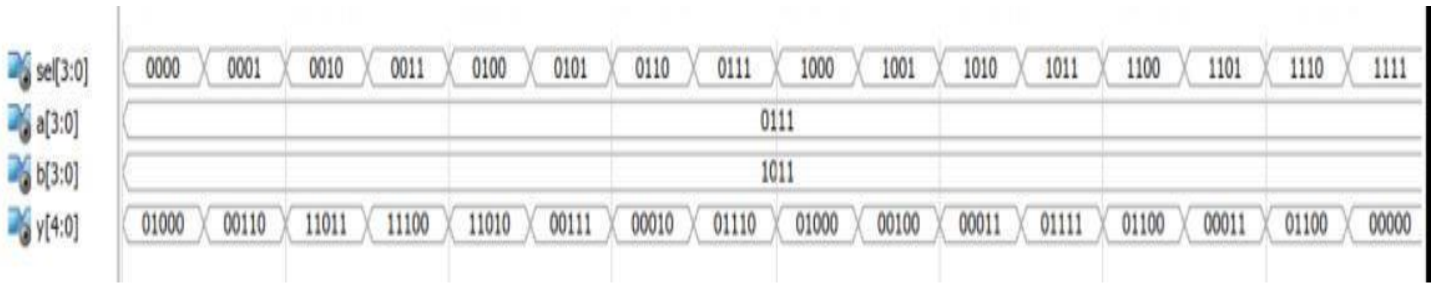
```
        elsif (sel = "1100") then    -- a XOR b
        ylv := (a XOR b);

        elsif (sel = "1101") then    -- a XNOR b
        ylv := (a XNOR b);

        elsif (sel = "1110") then    -- a NAND b
        ylv := (a NAND b);

        elsif (sel = "1111") then    -- a NOR b
        ylv := (a NOR b);
                                                       end if;




          if (sel(3) = '0') then
                    y <= yav;
        elsif (sel(3) = '1') then
y <= ("0" & ylv);
          end if;
 end
process; end
Alu;
```

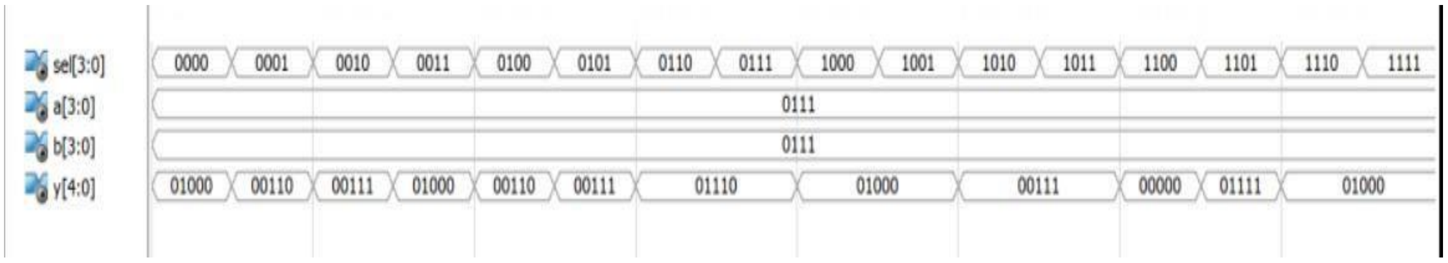
Testbench

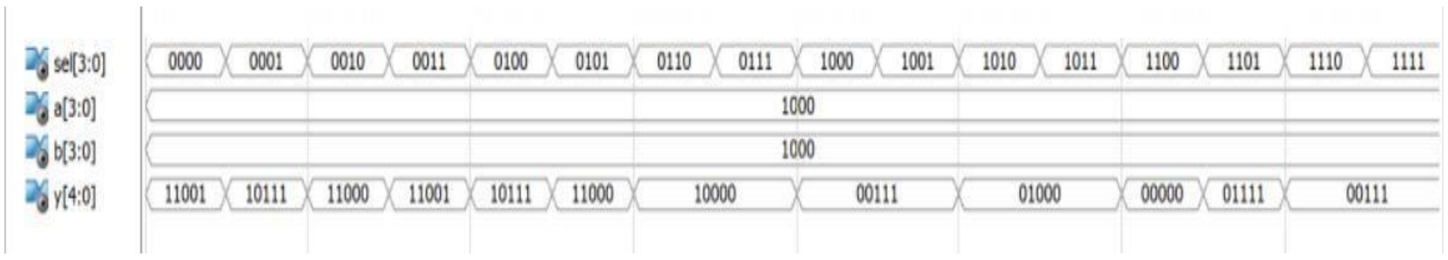
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL; use
ieee.std_logic_signed.all;

ENTITY tst IS
END tst;

ARCHITECTURE behavior OF tst IS

COMPONENT Alu PORT(        sel : IN
std_logic_vector(3 downto 0);        a : IN
std_logic_vector(3 downto 0);        b : IN
std_logic_vector(3 downto 0);        y : OUT
std_logic_vector(4 downto 0)
     );

END COMPONENT;

       --Inputs
      signal sel : std_logic_vector(3 downto 0) := (others => '0');
signal a : std_logic_vector(3 downto 0)  := (others => '0');
signal b : std_logic_vector(3 downto 0)  := (others => '0');

       --Outputs
       signal y : std_logic_vector(4 downto 0);



BEGIN      uut: Alu
PORT MAP (
sel => sel,
a => a,            b =>
b,           y => y
        );



   -- Stimulus process
stim_proc: process    begin

      a <= "1000";           -- input a to change   b <=
"1011";          -- input b to change
      sel <= "0000";
wait for 100 ns;
      for i in 0 to 15 loop
sel <= sel + "0001";         wait
for 100 ns;
       end loop;
 ----    wait;
end process;

END;
```

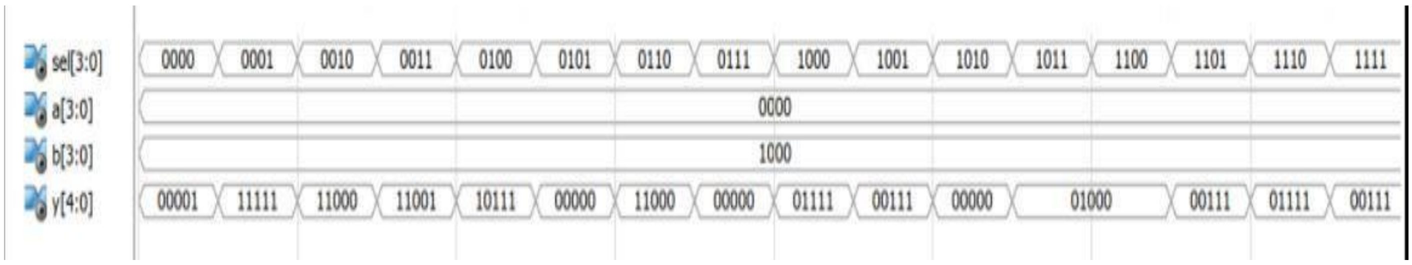
Waveforms:

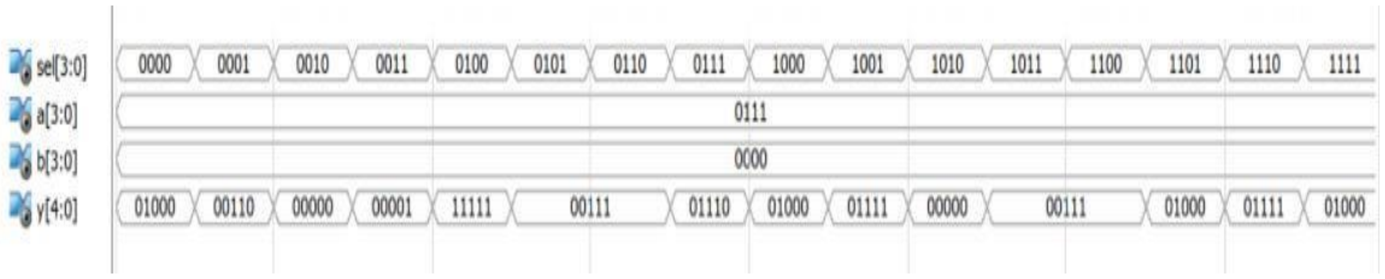

sel[3:0] 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
a[3:0] 1000
b[3:0] 1011
y[4:0] 11001 10111 11011 11100 11010 11000 10011 10000 00111 00100 01000 01011 00011 01100 00111 00100

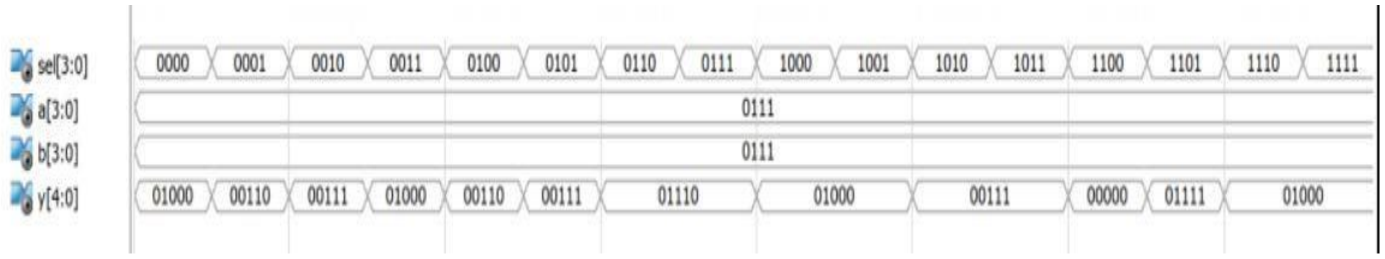

sel[3:0] 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
a[3:0] 0111
b[3:0] 1011
y[4:0] 01000 00110 11011 11100 11010 00111 00010 01110 01000 00100 00011 01111 01100 00011 01100 00000

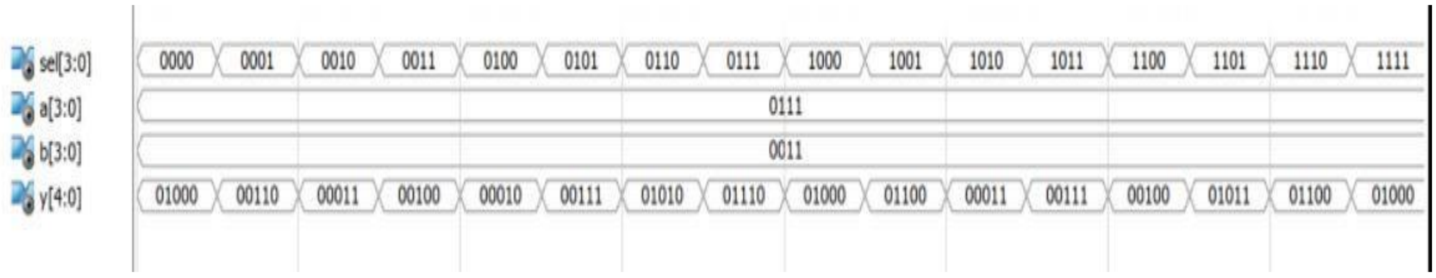

sel[3:0] 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
a[3:0] 0111
b[3:0] 0111
y[4:0] 01000 00110 00111 01000 00110 00111 01110 01000 00111 00000 01111 01000

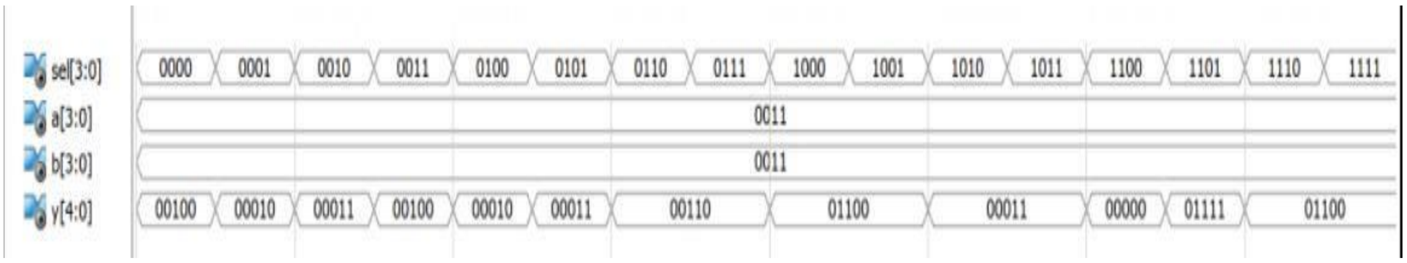

sel[3:0] 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
a[3:0] 1000
b[3:0] 1000
y[4:0] 11001 10111 11000 11001 10111 11000 10000 00111 01000 00000 01111 00111

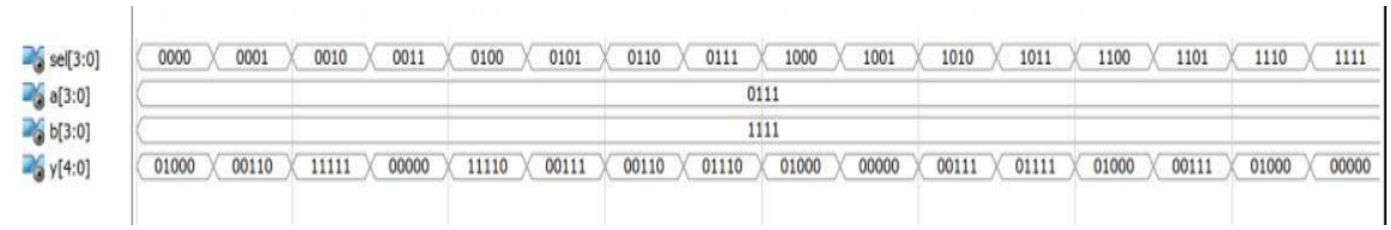

sel[3:0] 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
a[3:0] 0000
b[3:0] 1000
y[4:0] 00001 11111 11000 11001 10111 00000 11000 00000 01111 00111 00000 01000 00111 01111 00111


sel[3:0] 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
a[3:0] 0111
b[3:0] 0000
y[4:0] 01000 00110 00000 00001 11111 00111 01110 01000 01111 00000 00111 01000 01111 01000

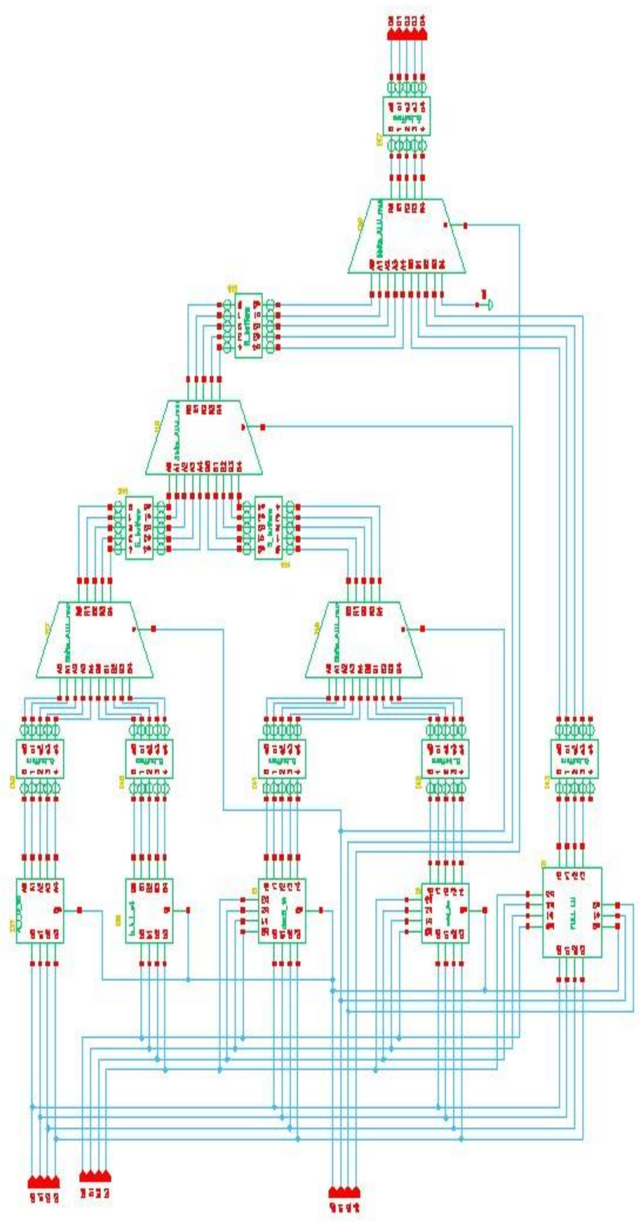

sel[3:0] 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
a[3:0] 0111
b[3:0] 0111
y[4:0] 01000 00110 00111 01000 00110 00111 01110 01000 00111 00000 01111 01000


sel[3:0] 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
a[3:0] 0111
b[3:0] 0011
y[4:0] 01000 00110 00011 00100 00010 00111 01010 01110 01000 01100 00011 00111 00100 01011 01100 01000

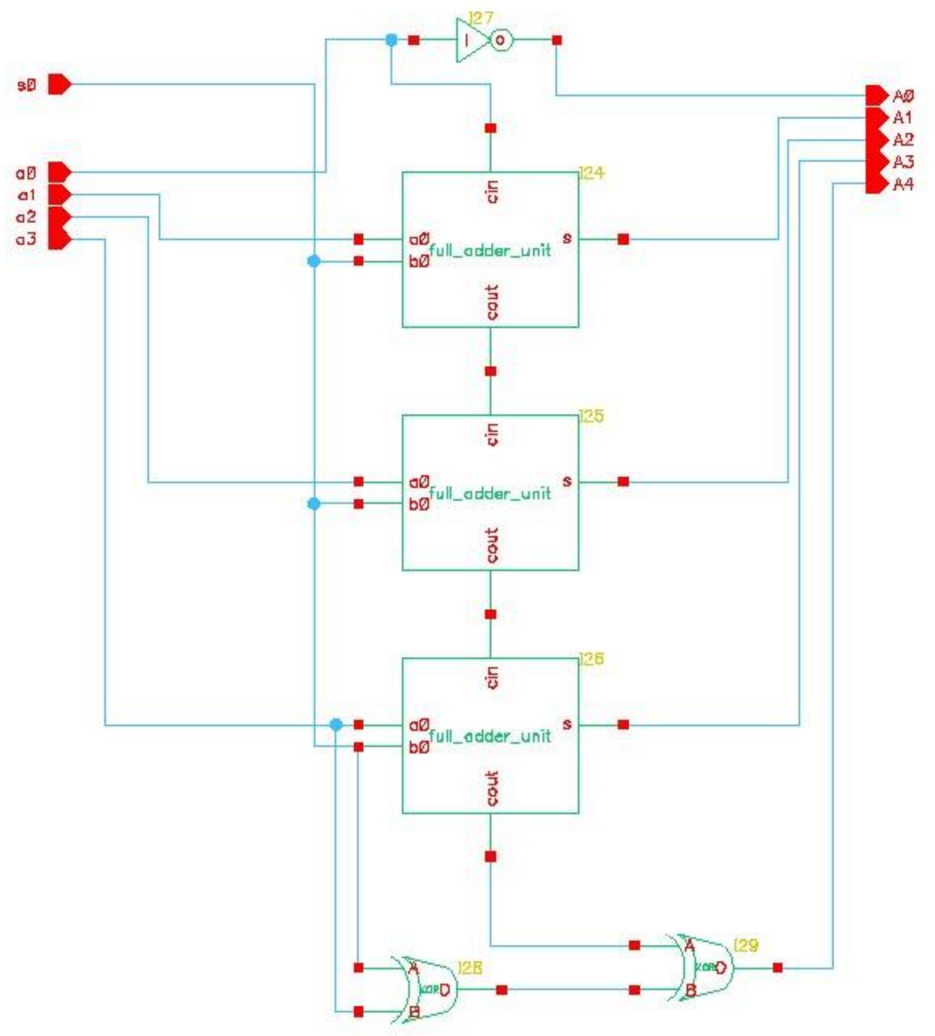




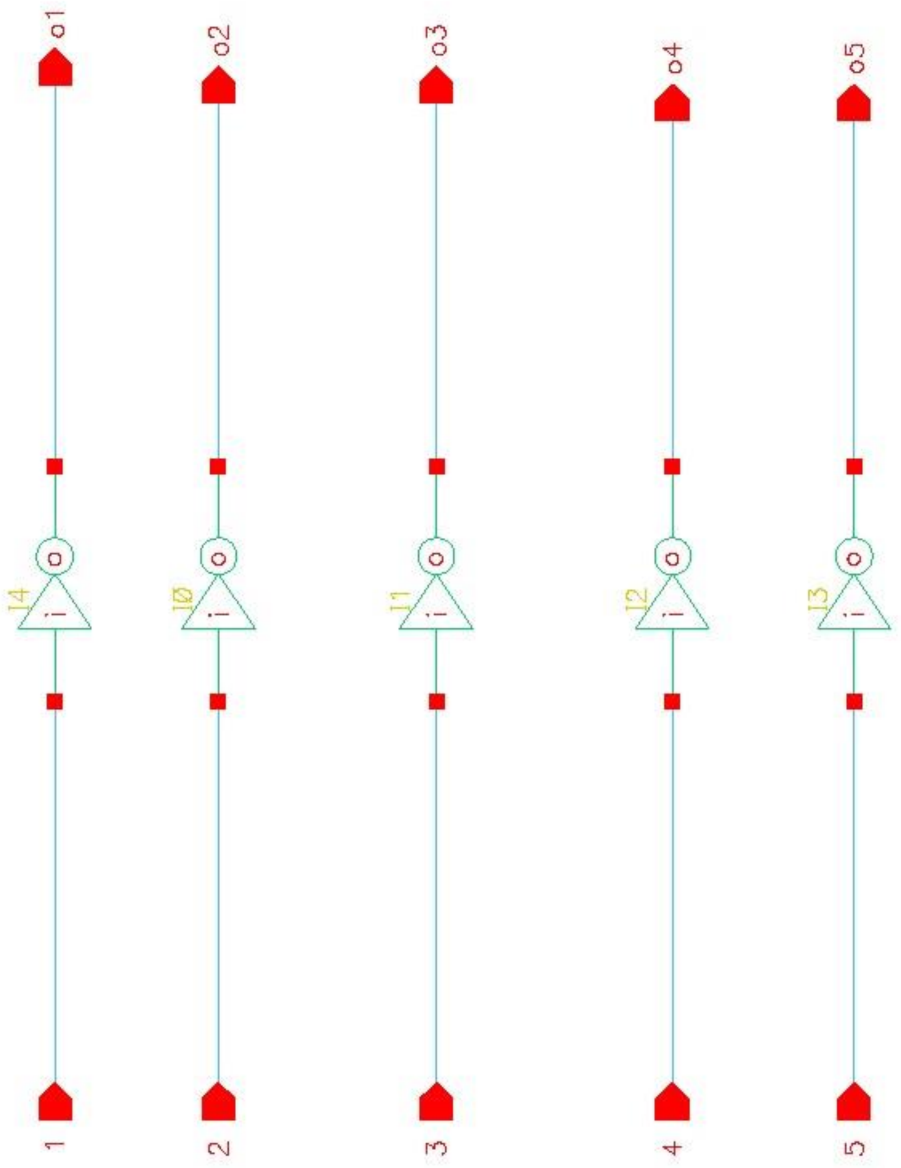
Circuit Schematics

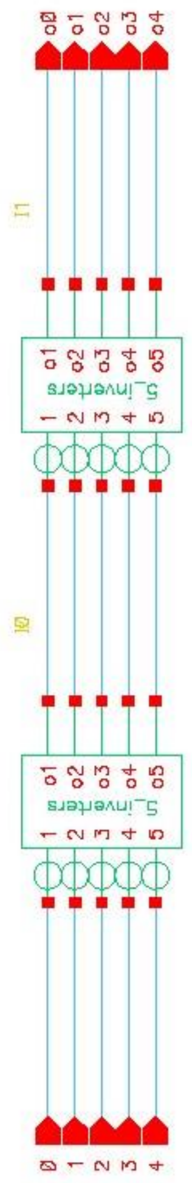


The full design of 4 bits ALU With sign extension

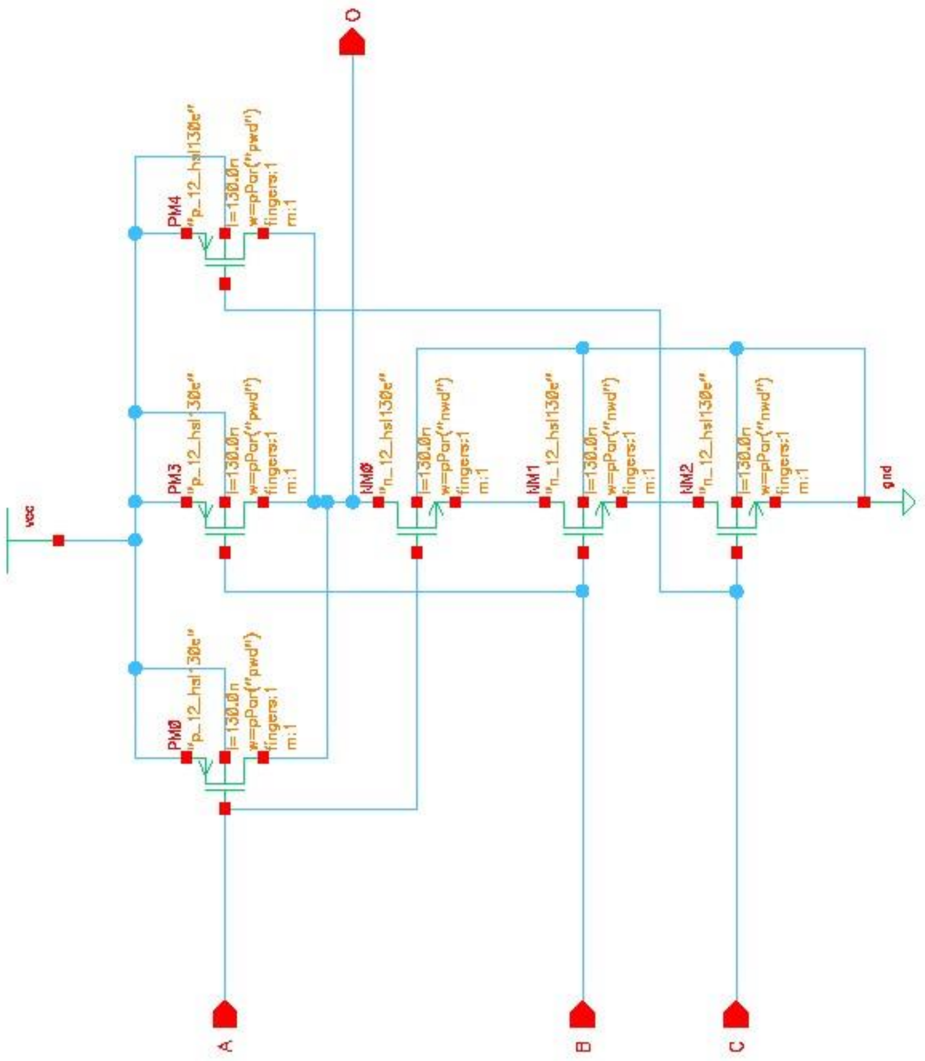


The increase and decrease of A circuit design

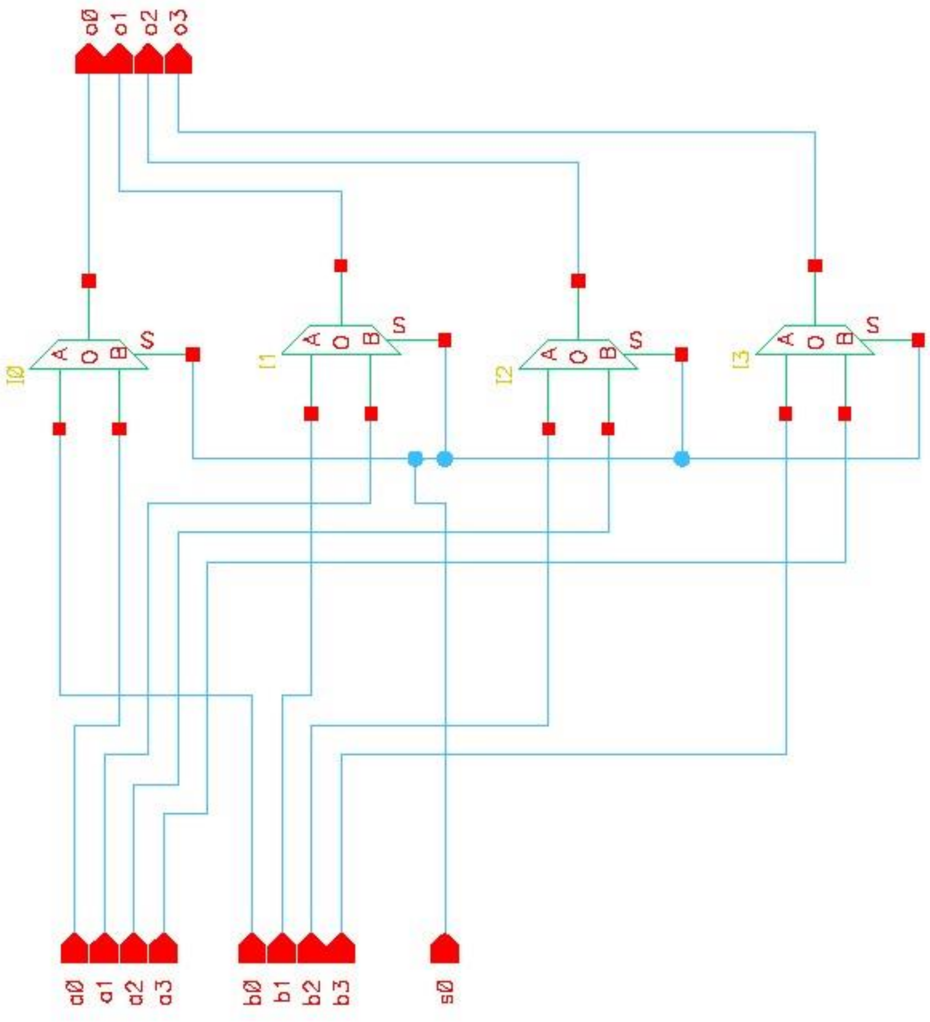

o1
o2
o3
o4
o5
I4
IØ
I1
I2
I3
i
o
1
2
3
4
5


o0
o1
o2
o3
o4
I1
5_inverters
1
2
3
4
5
o1
o2
o3
o4
o5
I0
5_inverters
1
2
3
4
5
o1
o2
o3
o4
o5
0
1
2
3
4

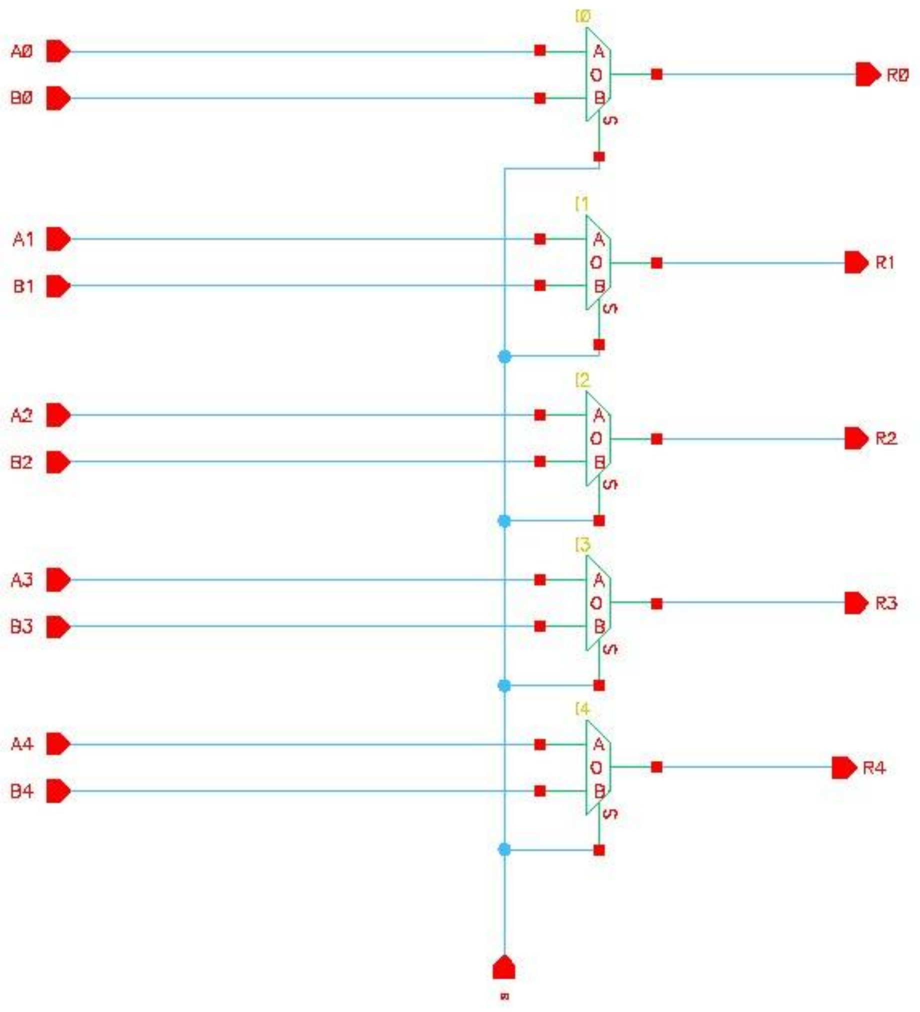
A buffer of 5 input bus



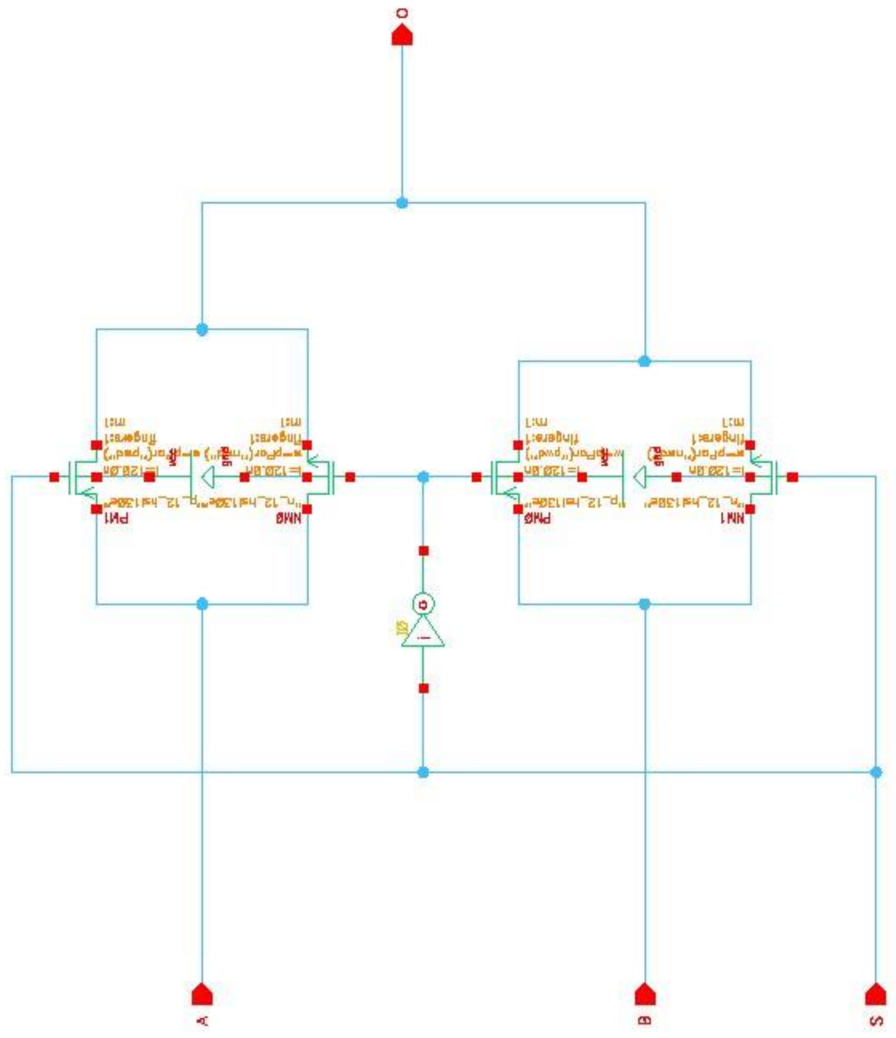
The design of 3 input NAND gate



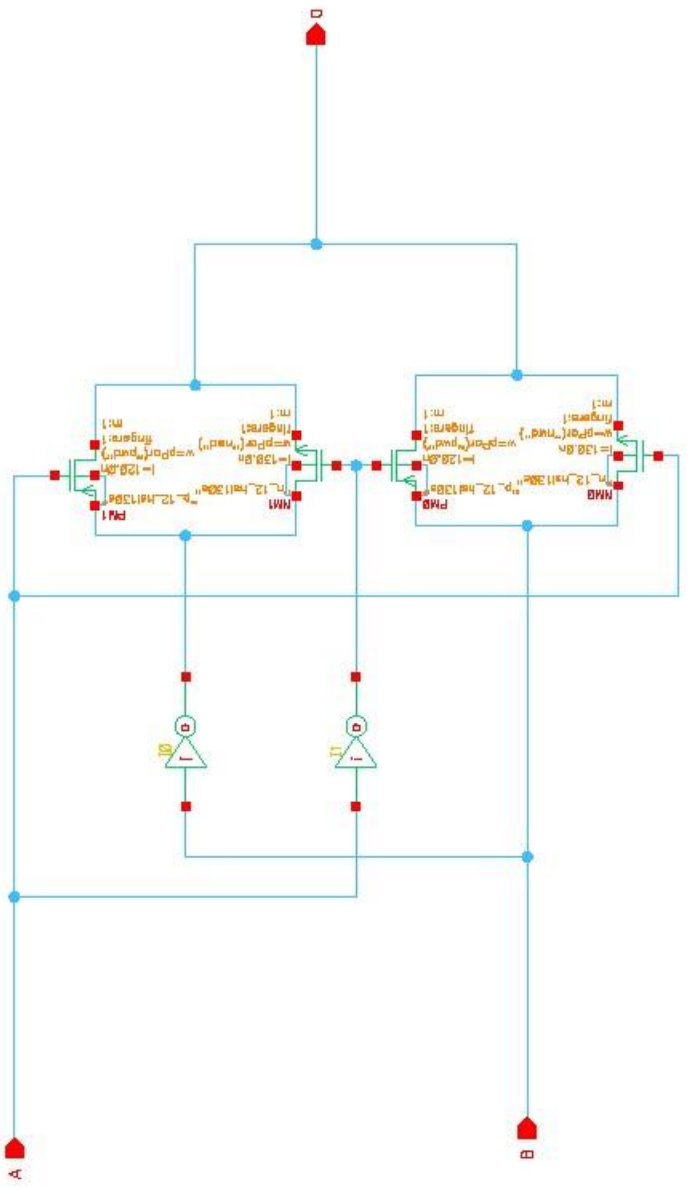
Bus of 4 for two input mux


I0
A0
B0
R0
I1
A1
B1
R1
I2
A2
B2
R2
I3
A3
B3
R3
I4
A4
B4
R4
A
O
B
S

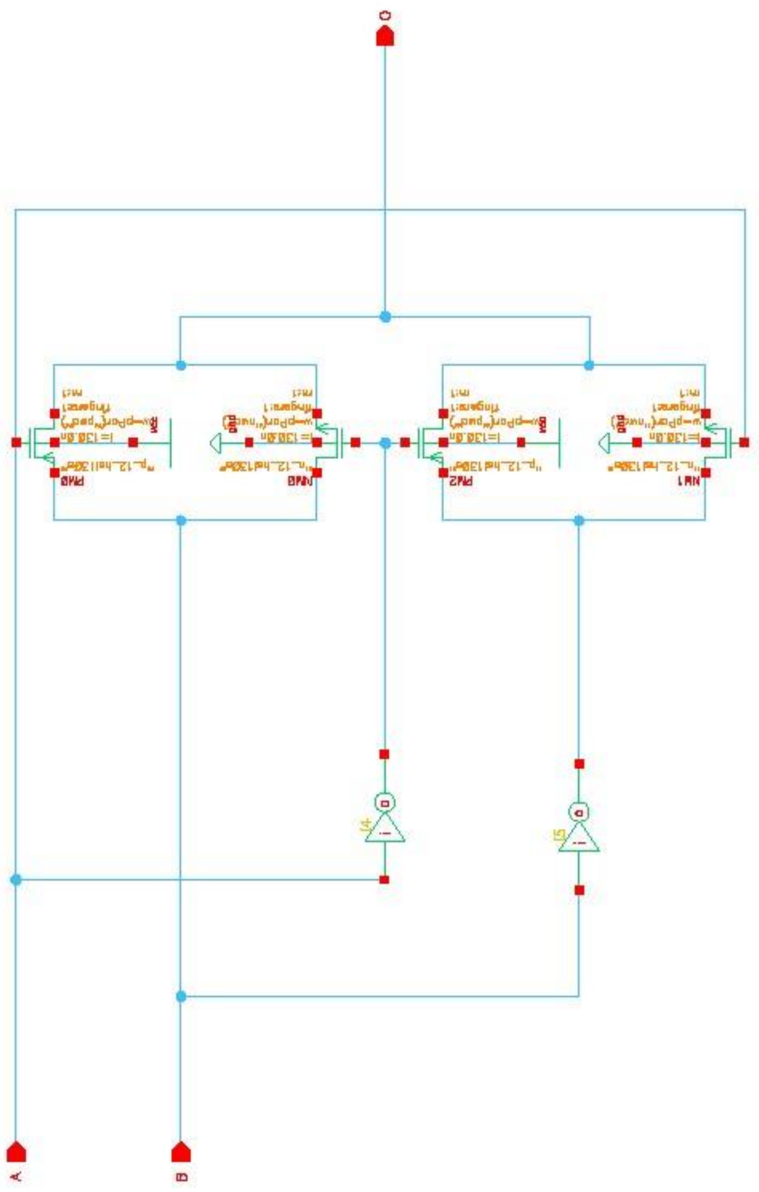
Bus of 5 for two input mux



The Design of the two input Multiplexer using transmission gates and Inverters

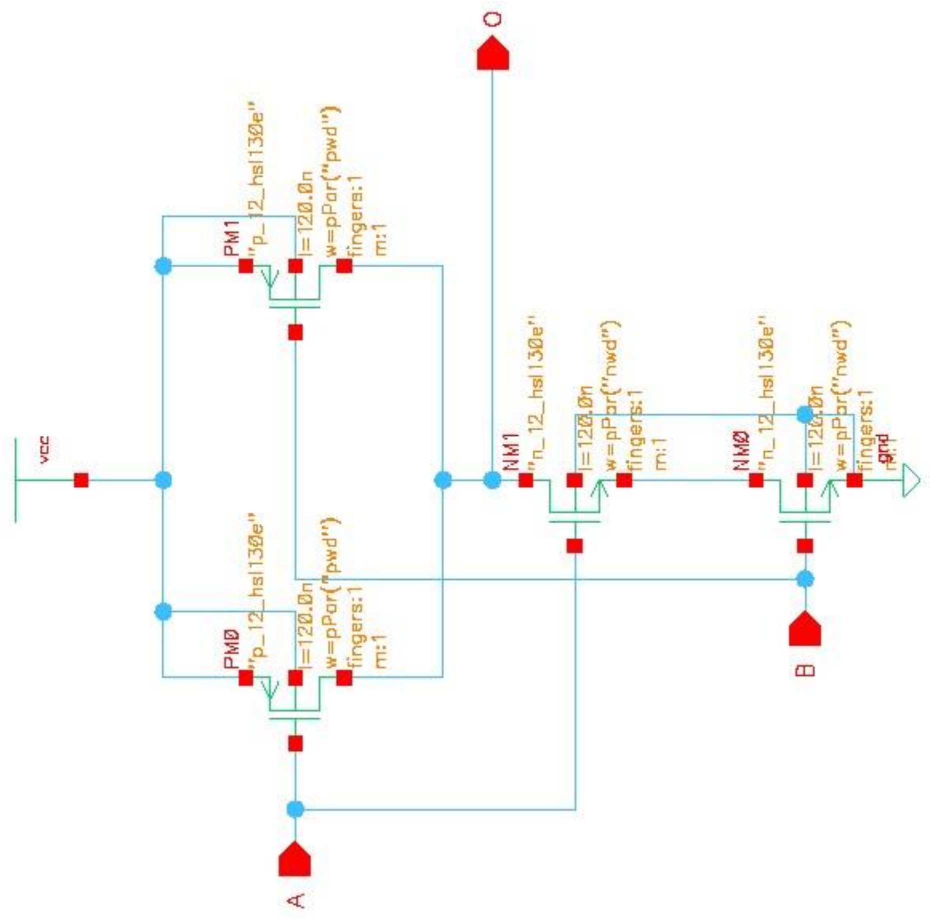


The design of two input XNOR gate using transmission gates and Inverters

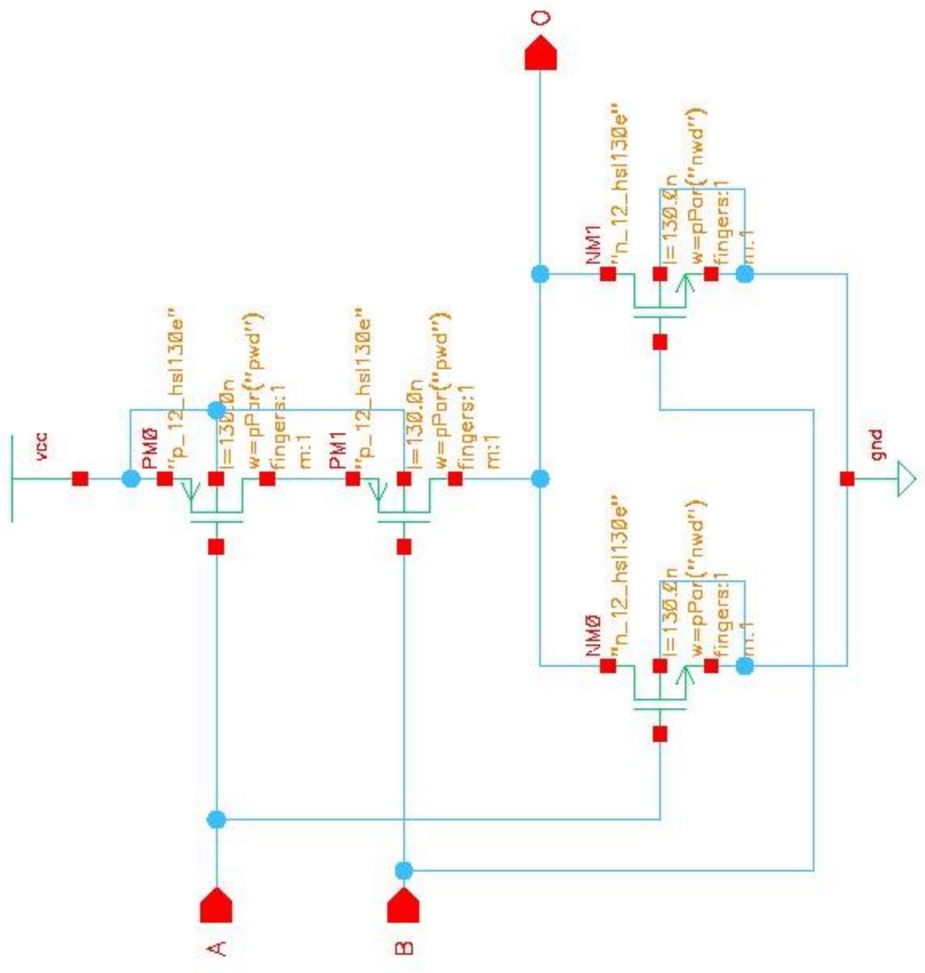


PM0
NM0
PM2
NM1
I4
I5
A
B

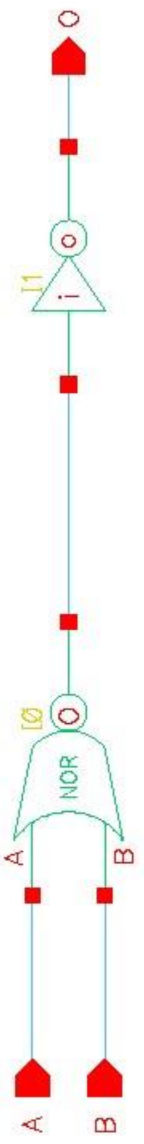
The design of two input XOR gate using transmission gates and Inverters



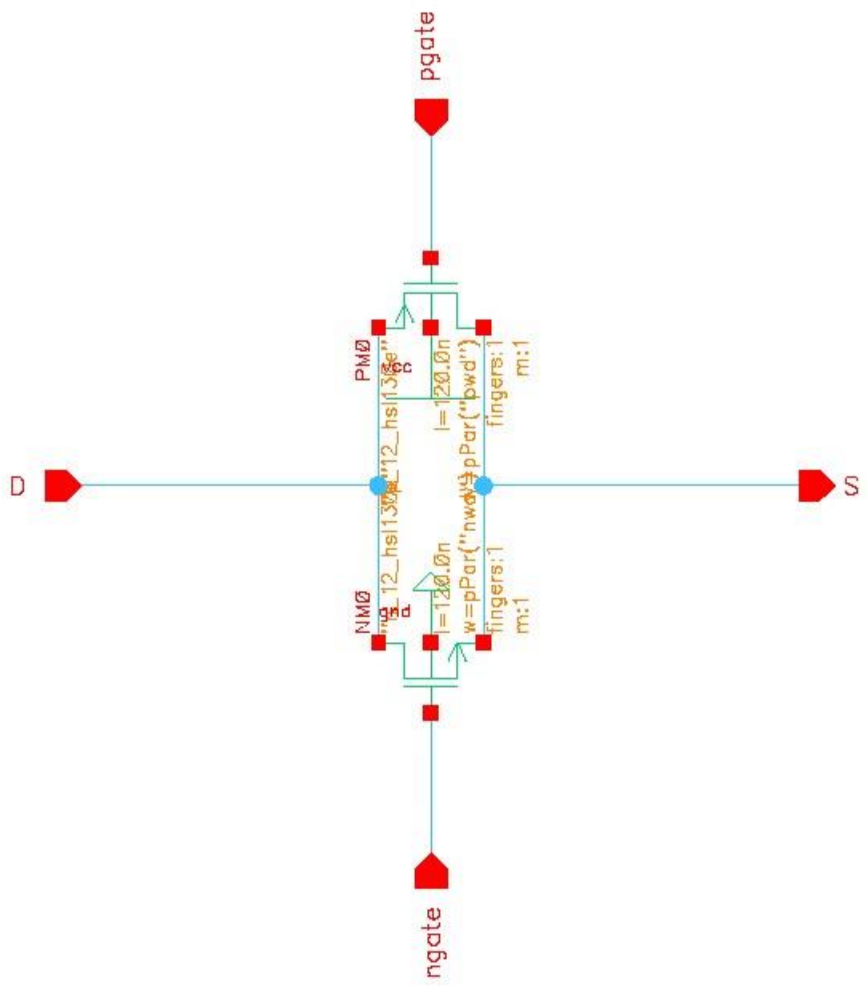
The Design of two input NAND gate



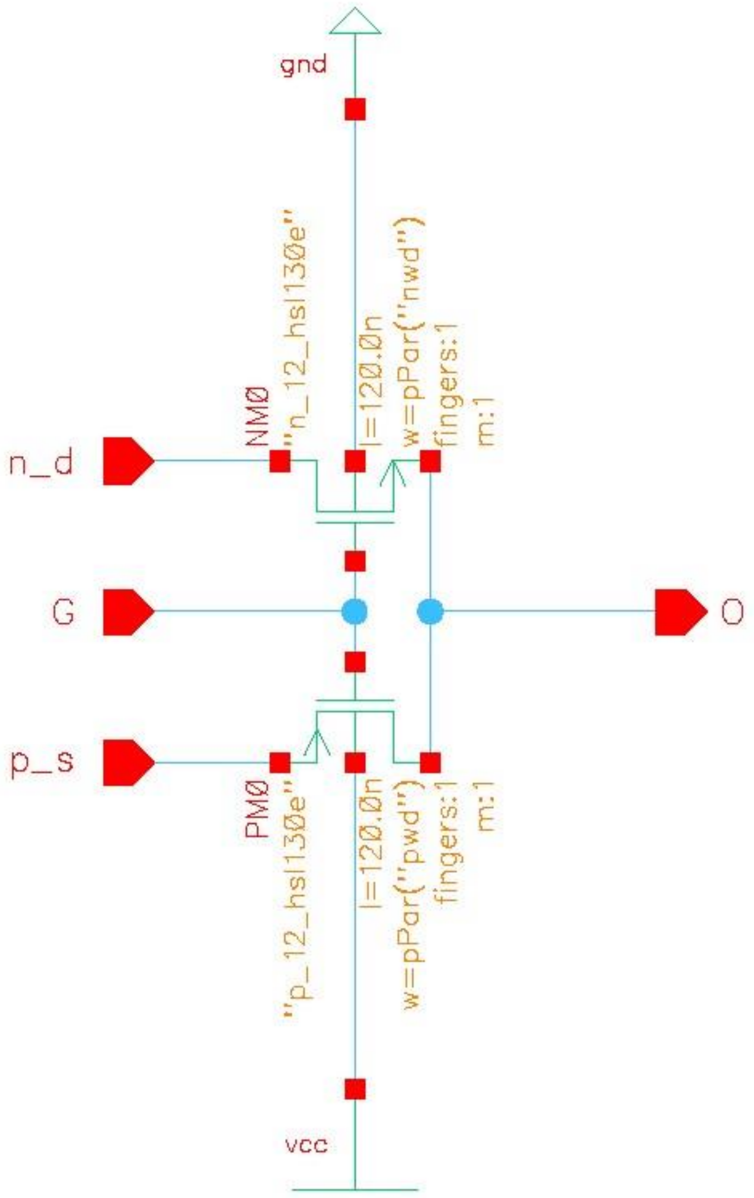
The Design of two input NOR gate

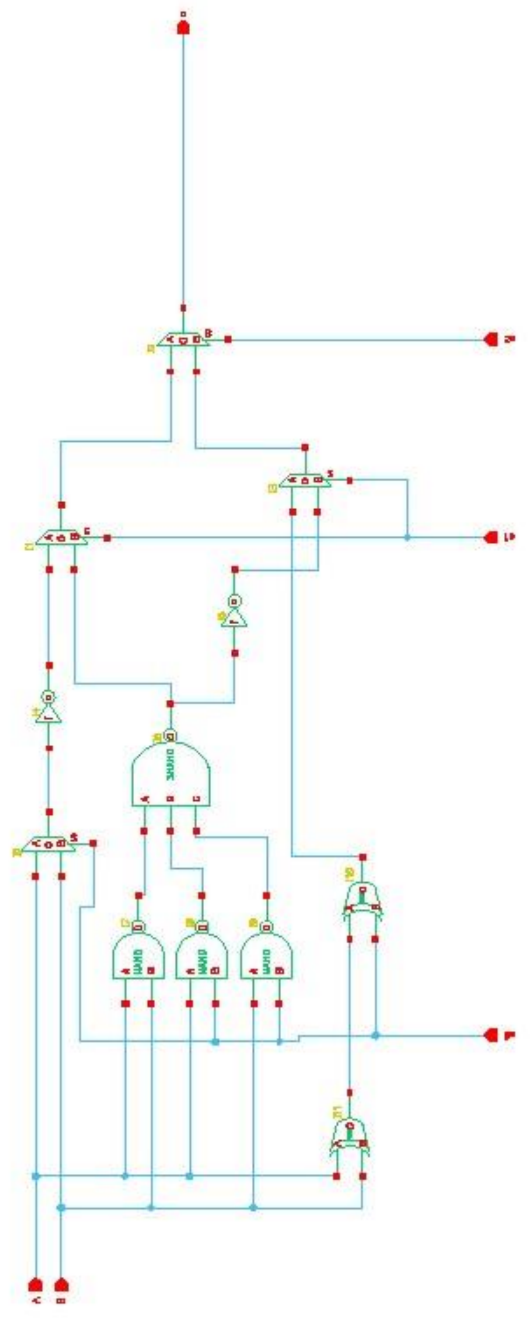


The Design of two input OR using NOR gate and Inverter

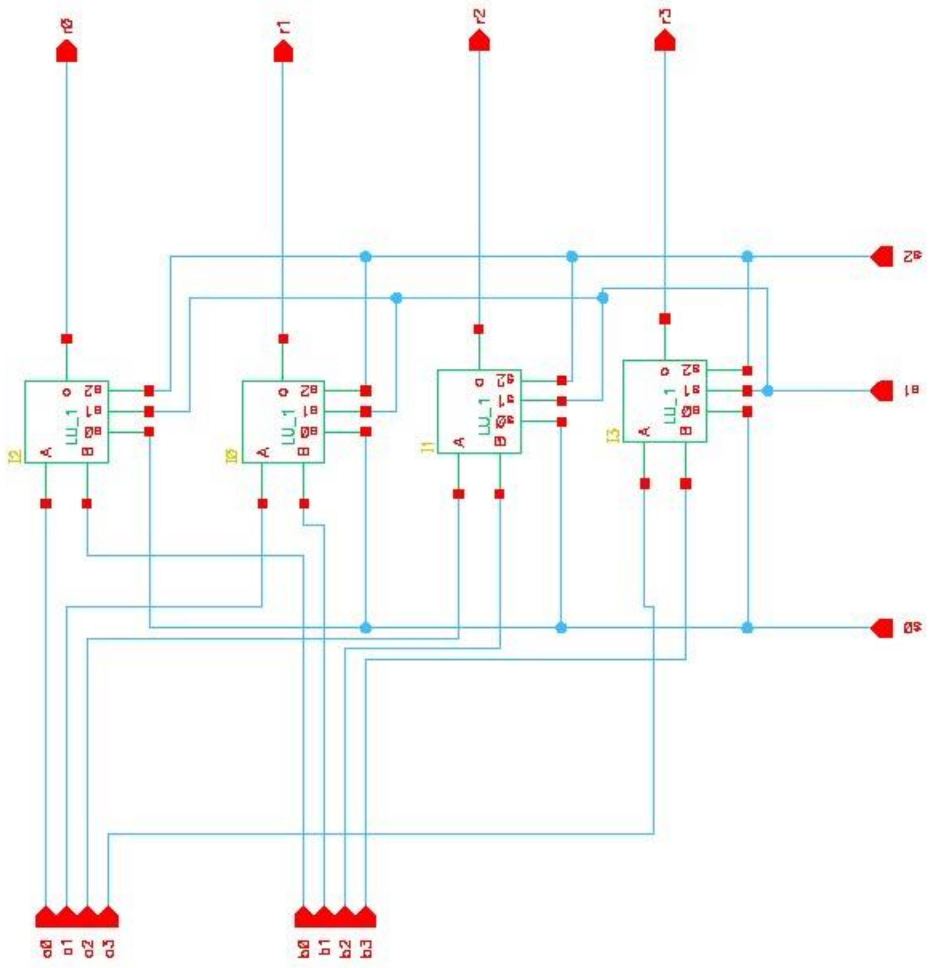


The design of transmission gate

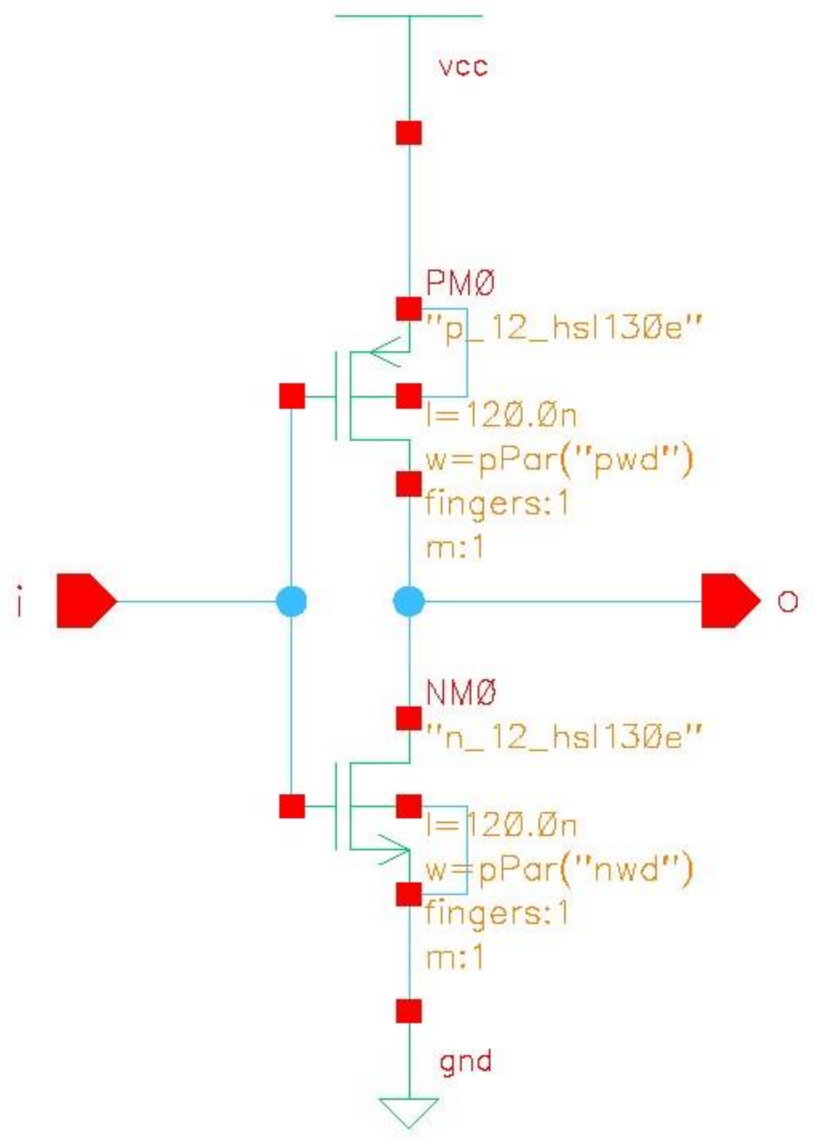




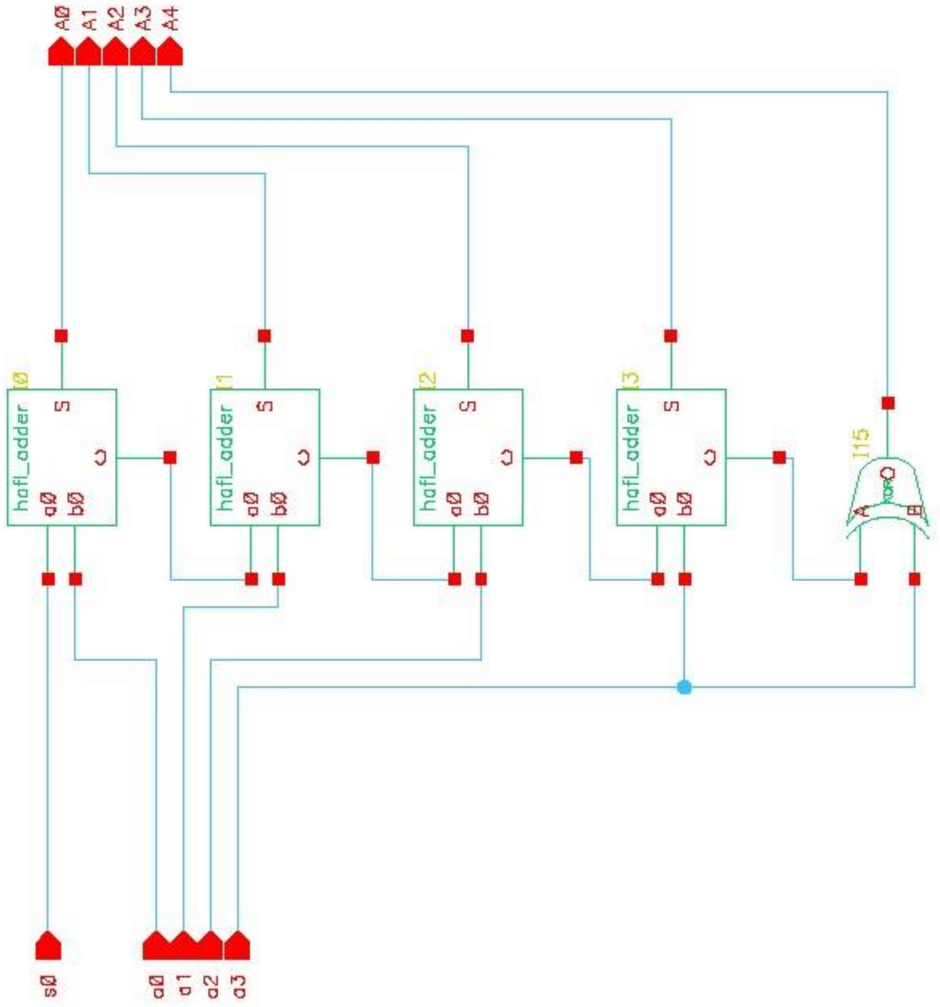
The one Bit Logic unit



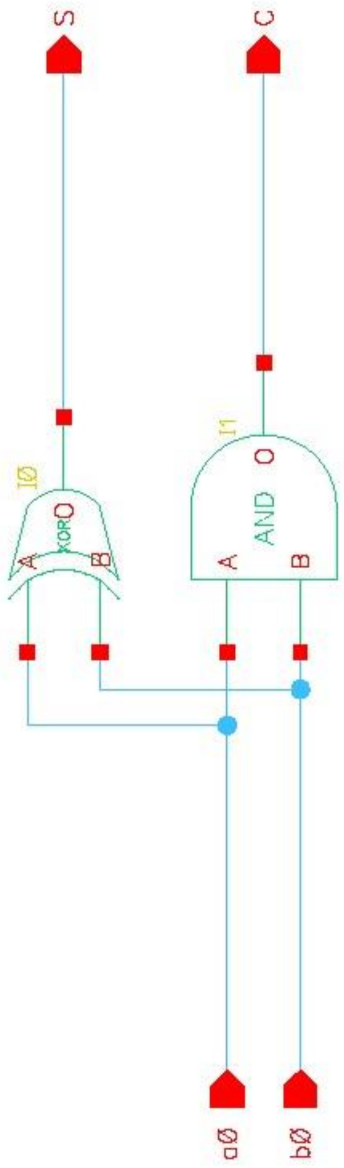
The four(4) Bits logic unit



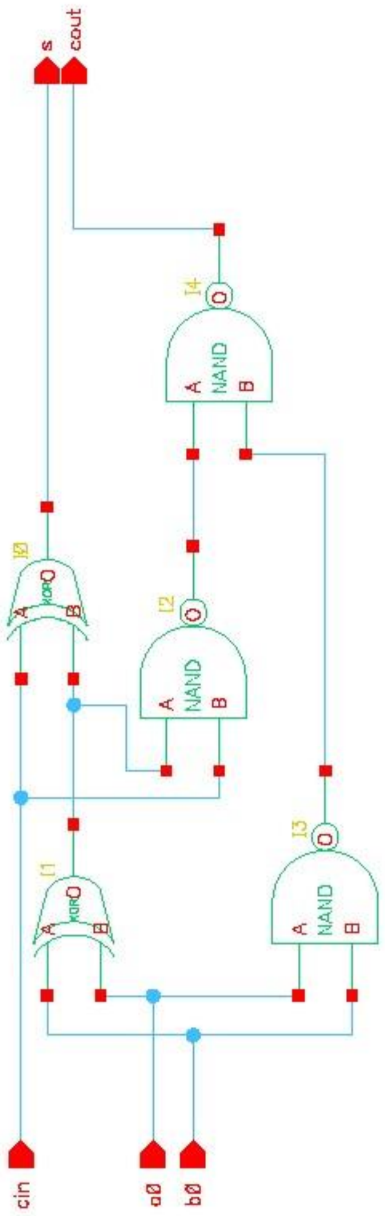
The design of the inverter



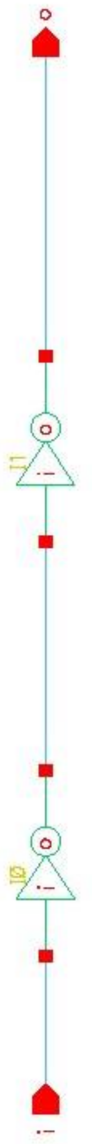
Transfer and increase B



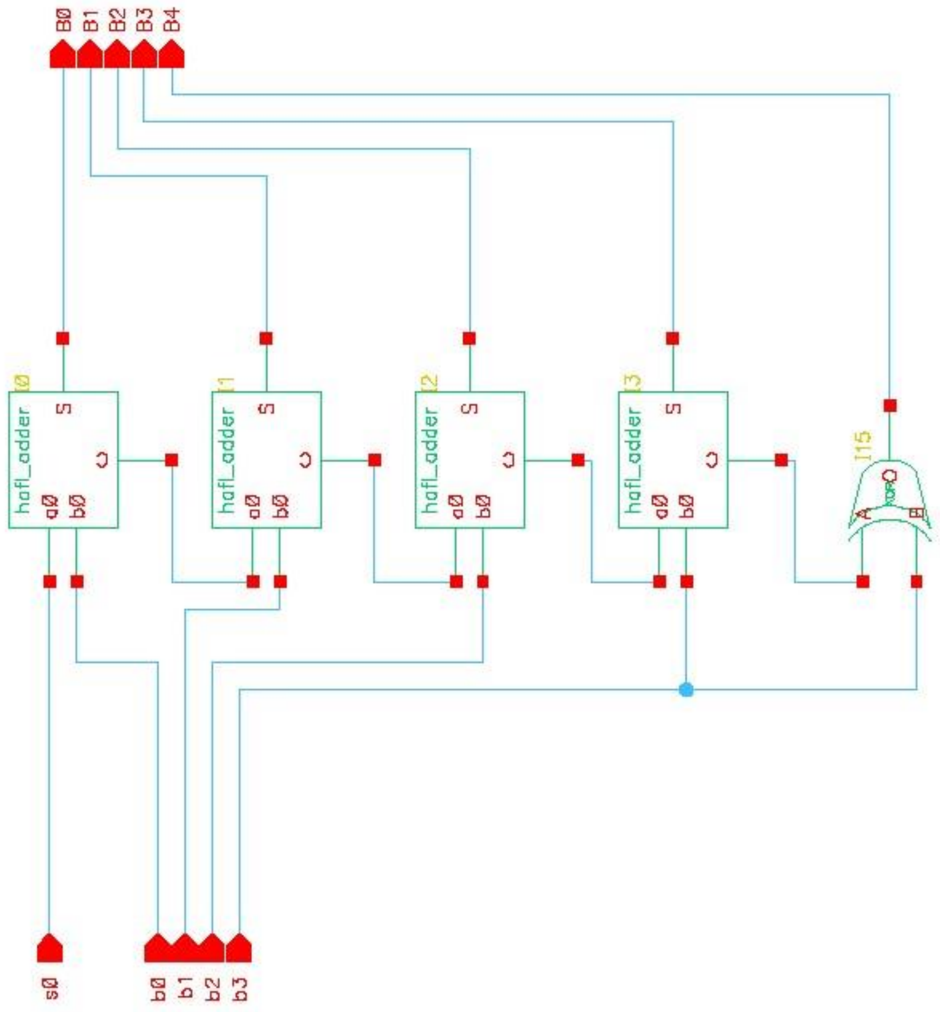
The design of Half Adder



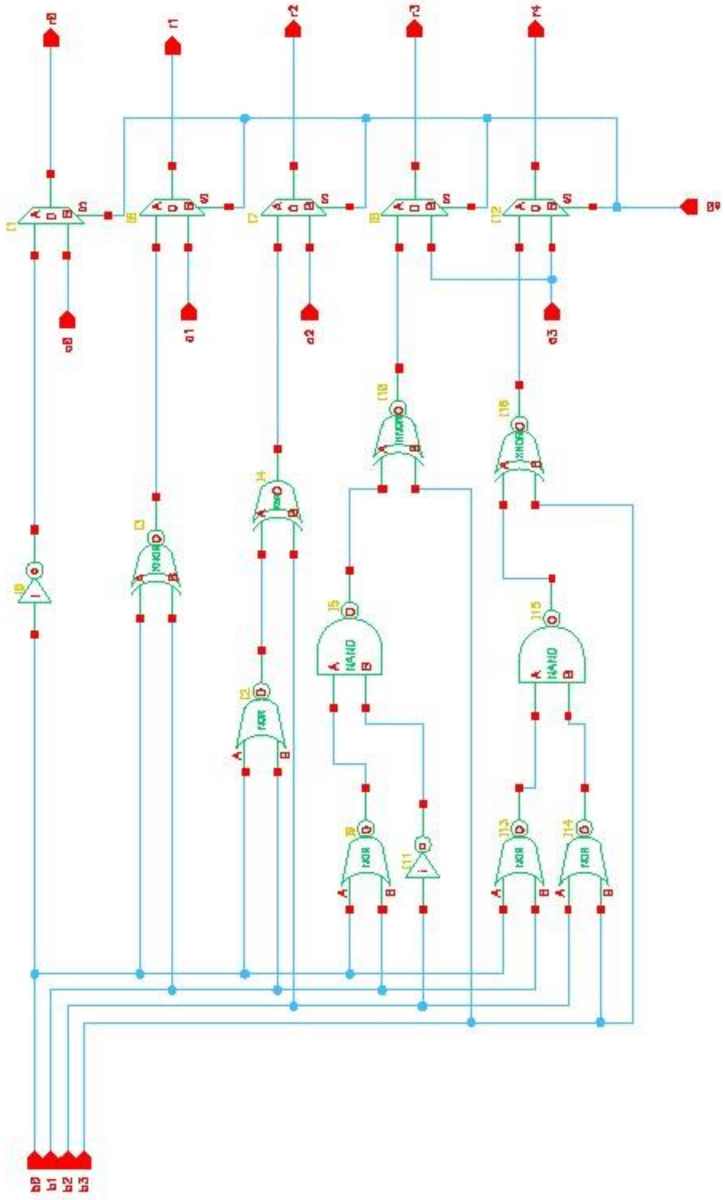
The design of the full adder (using Two Half Adders)



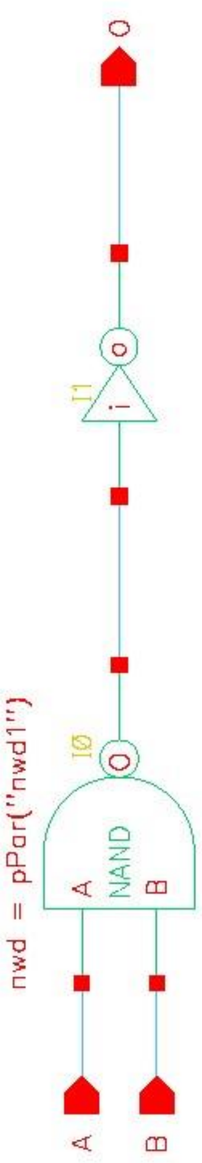
The design of buffer



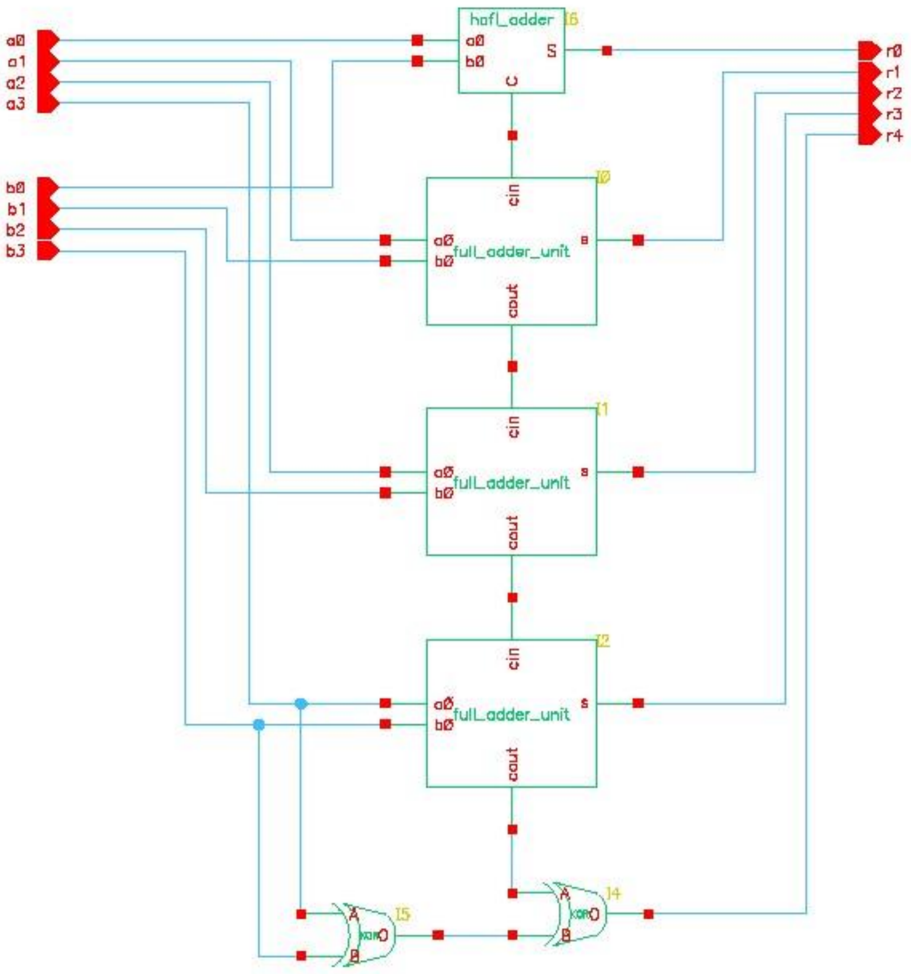
The design of (transfer and increase b)


r0
r1
r2
r3
r4
a0
a1
a2
a3
b0
b1
b2
b3
NAND
NAND
NOR
NOR
NOR
NOR
XNOR
XNOR
XNOR
XNOR

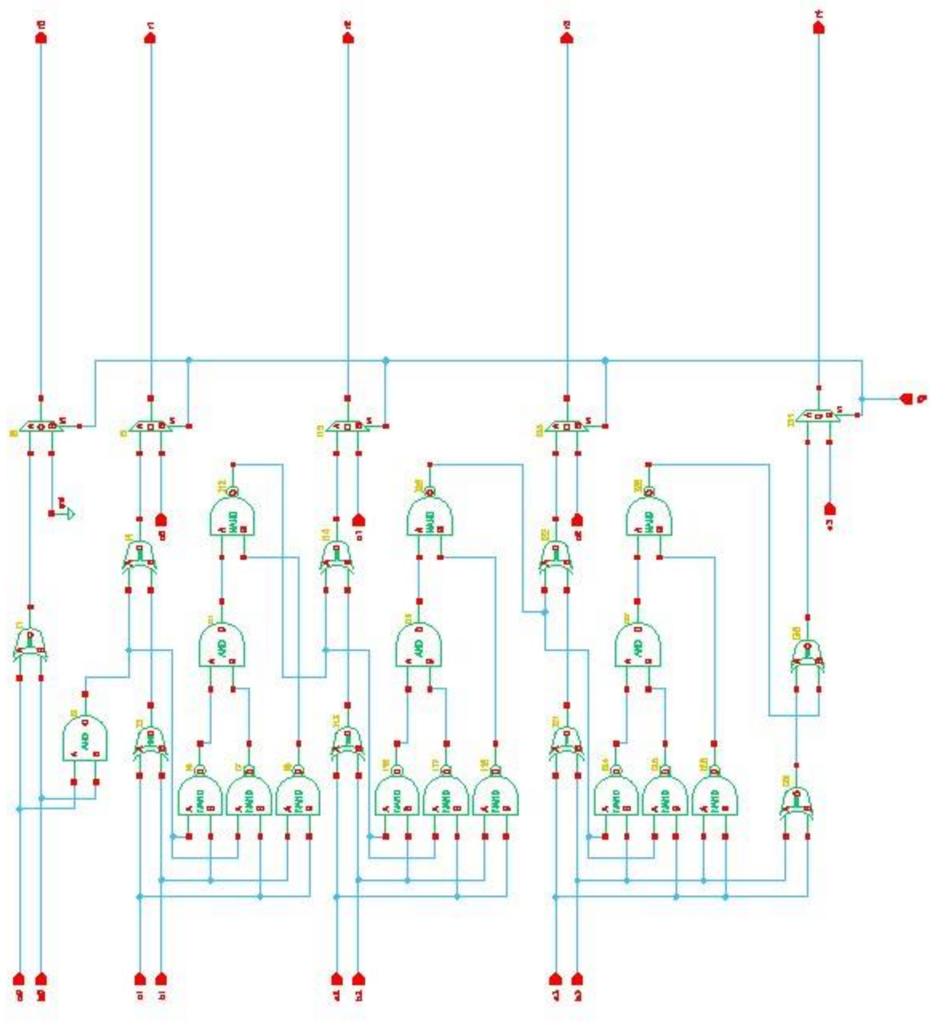
The design of decrease (b) && transfer (a)



The design of AND gate using NAND and Inverter

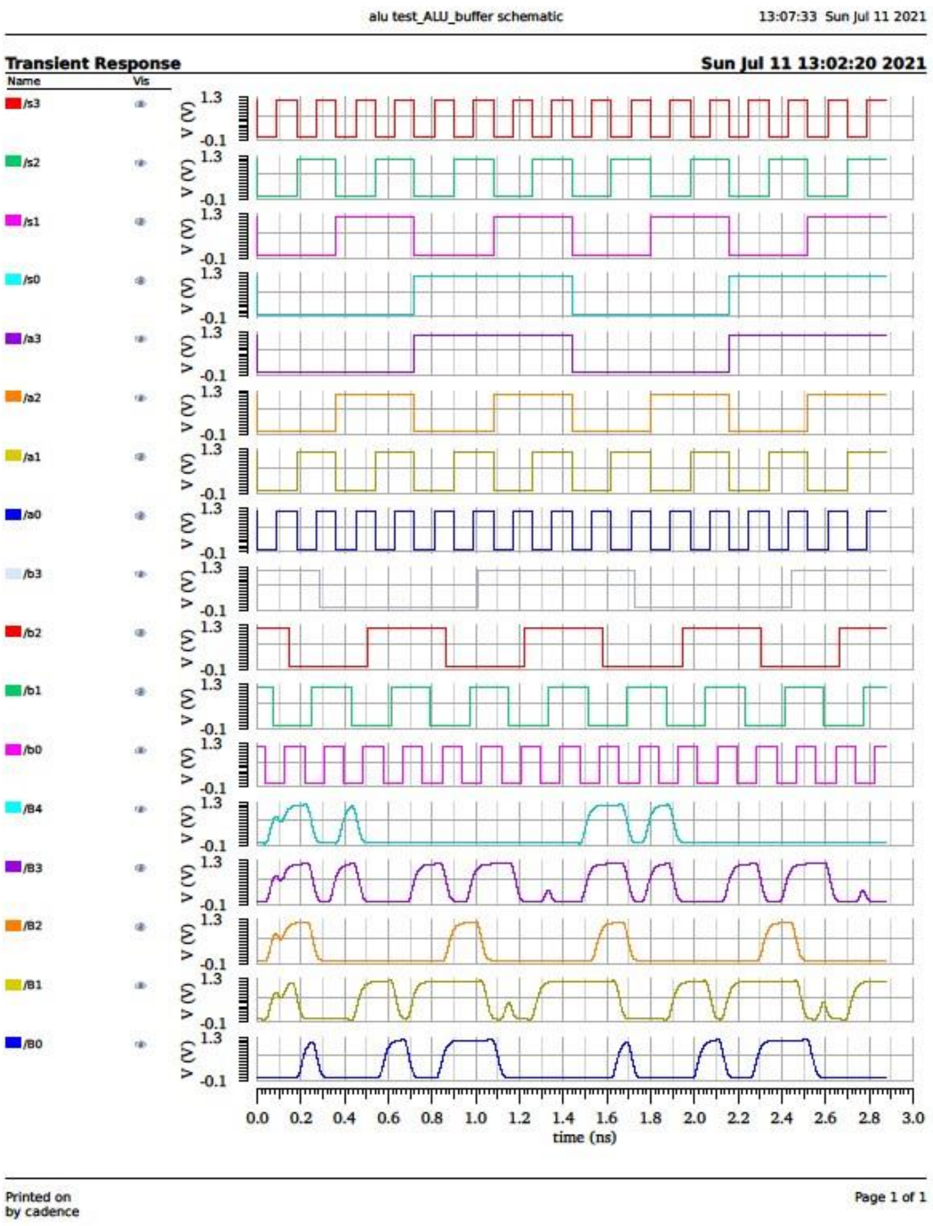


The design of increase and decrease a



The design of add a&b (a+b)   and (the 2*a)

11.7 GHZ

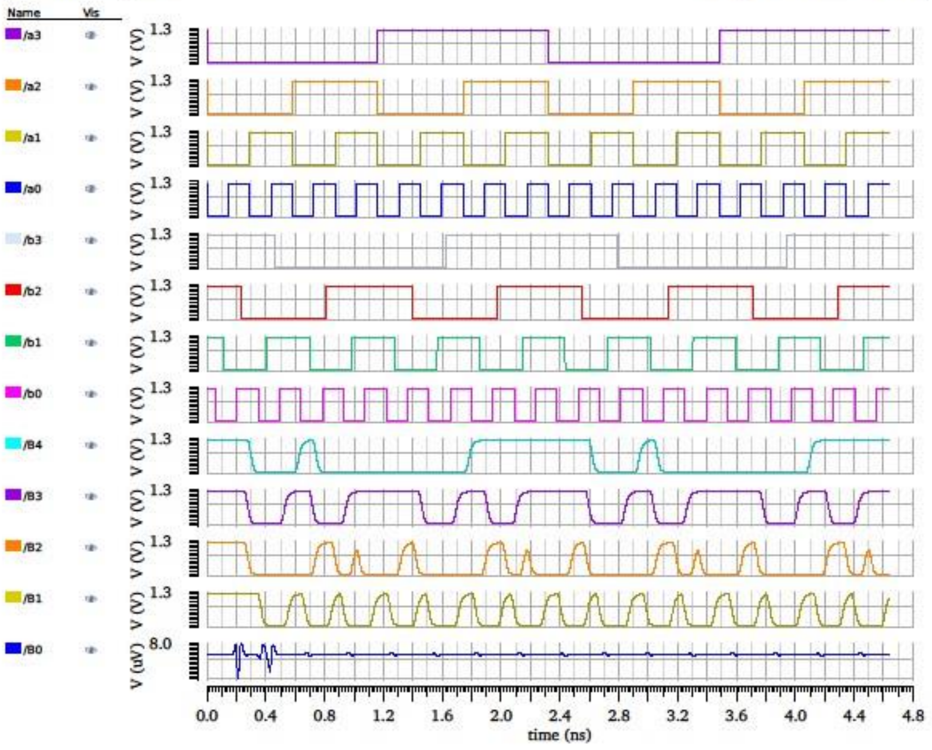


alu test_ALU_buffer schematic
13:07:33 Sun Jul 11 2021
Transient Response
Sun Jul 11 13:02:20 2021
Name
Vis
/s3
/s2
/s1
/s0
/a3
/a2
/a1
/a0
/b3
/b2
/b1
/b0
/B4
/B3
/B2
/B1
/B0
V (V)
1.3
-0.1
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3.0
time (ns)
Printed on
by cadence
Page 1 of 1

3.34GHZ

alu test_ALU_buffer schematic 14:48:43 Sun Jul 11 2021

Transient Response **Sun Jul 11 14:32:45 2021**



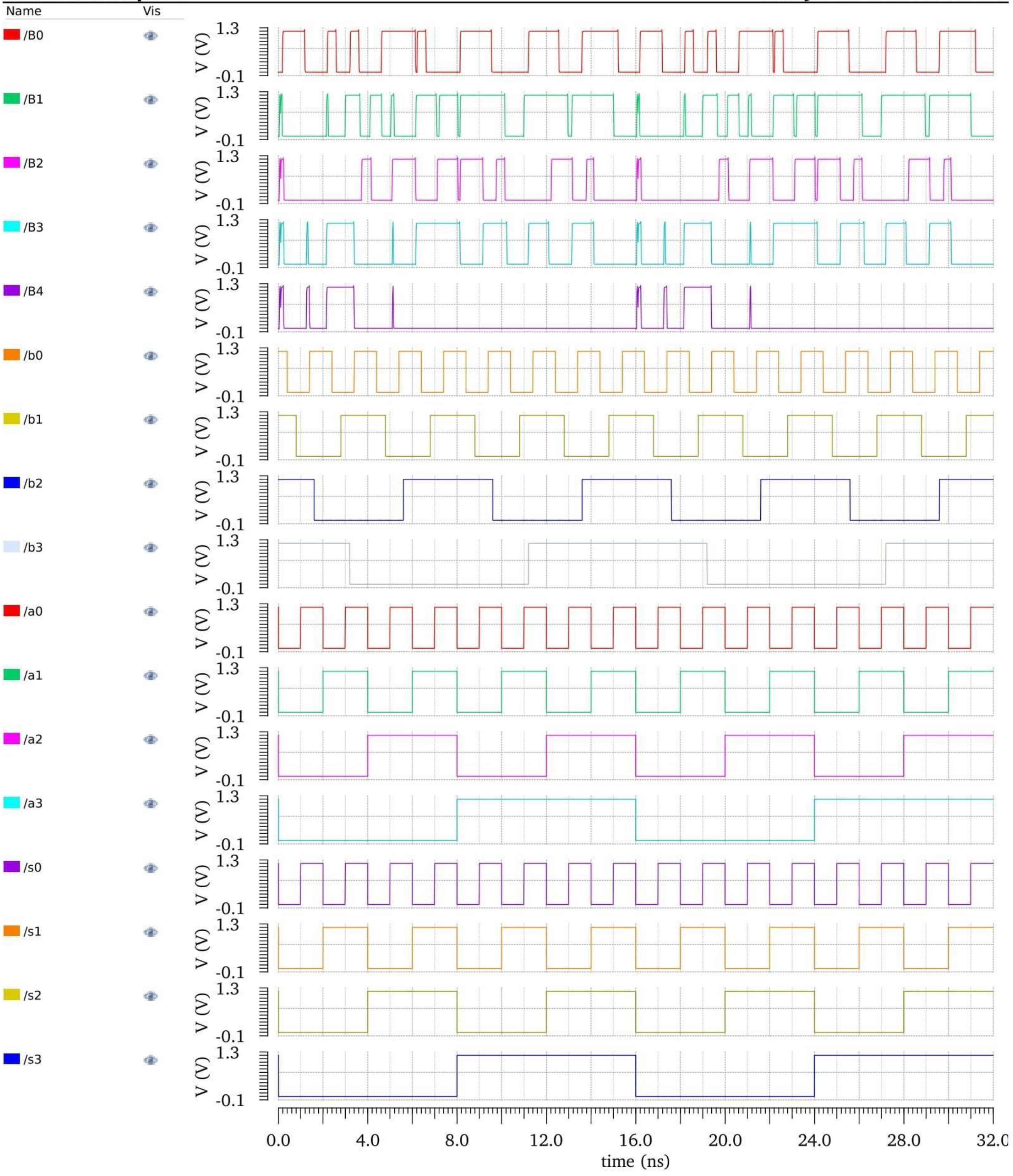
Markers

		V1
	x	10.9161ns
🟪	/a3	None
🟧	/a2	None
🟨	/a1	None
🟦	/a0	None
⬜	/b3	None
🟥	/b2	None
🟩	/b1	None
🟪	/b0	None
🟦	/B4	None
🟪	/B3	None
🟧	/B2	None
🟨	/B1	None
🟦	/B0	None

Printed on
by cadence

Page 1 of 1

Transient Response **Sun Jul 11 09:13:54 2021**


Name
Vis
/B0
/B1
/B2
/B3
/B4
/b0
/b1
/b2
/b3
/a0
/a1
/a2
/a3
/s0
/s1
/s2
/s3
V (V)
time (ns)