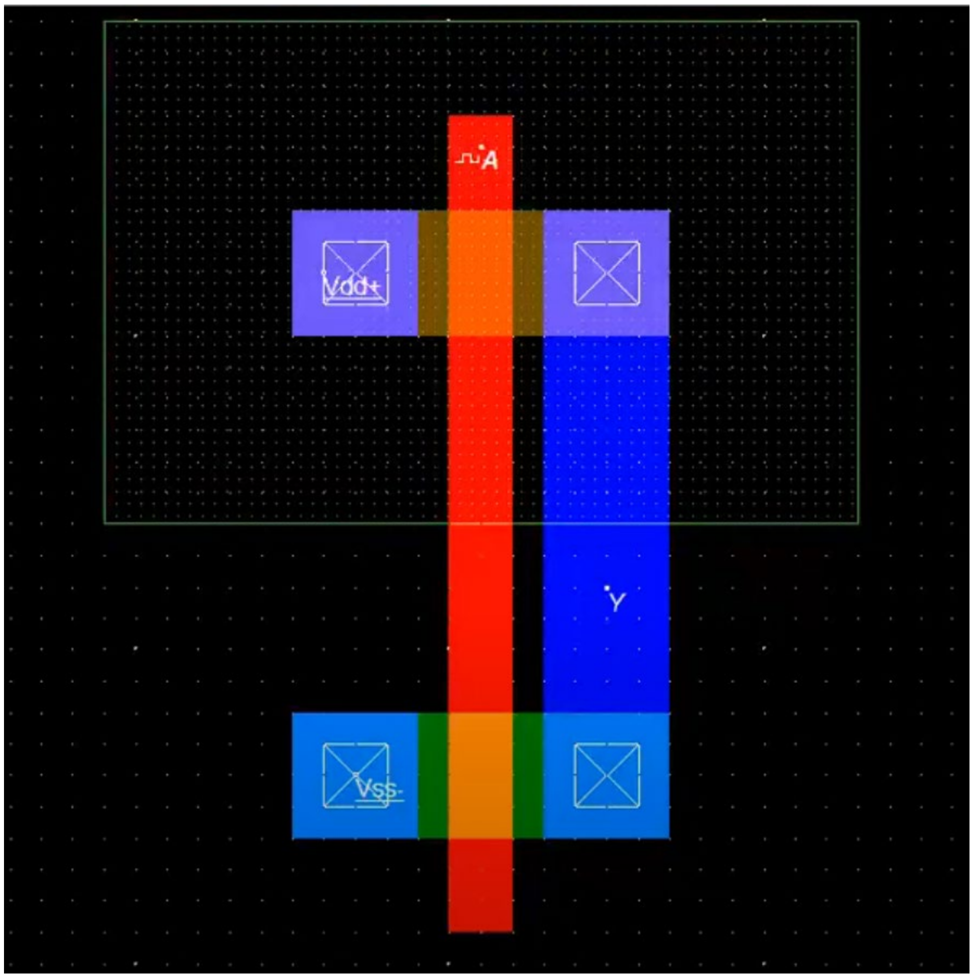
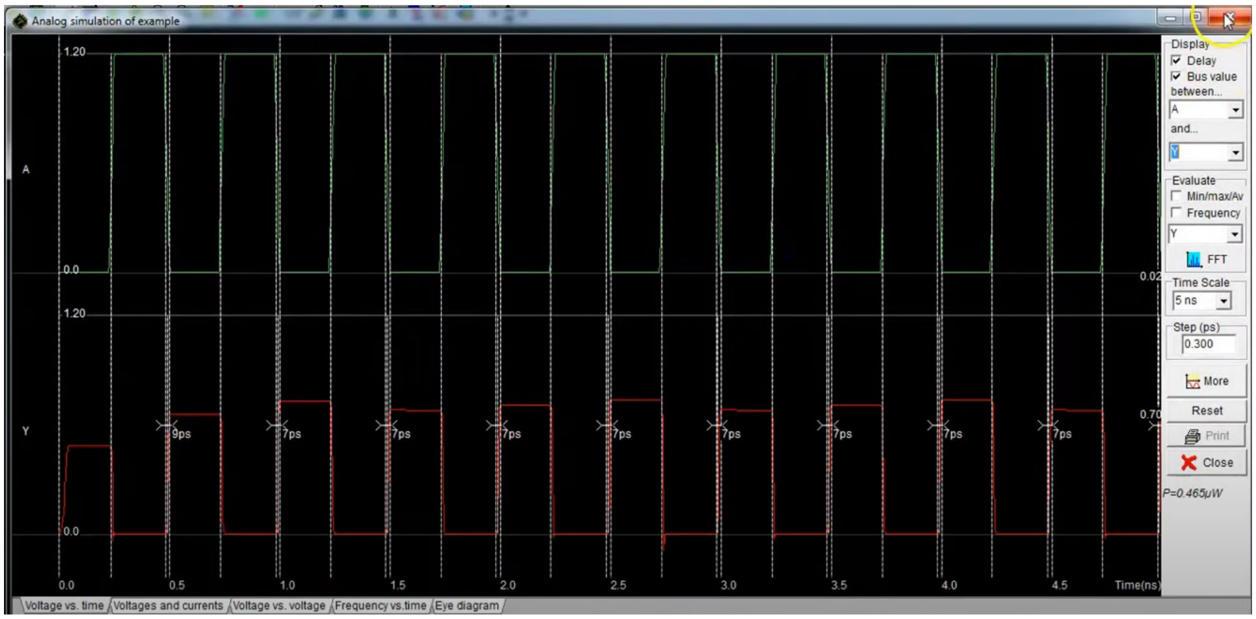
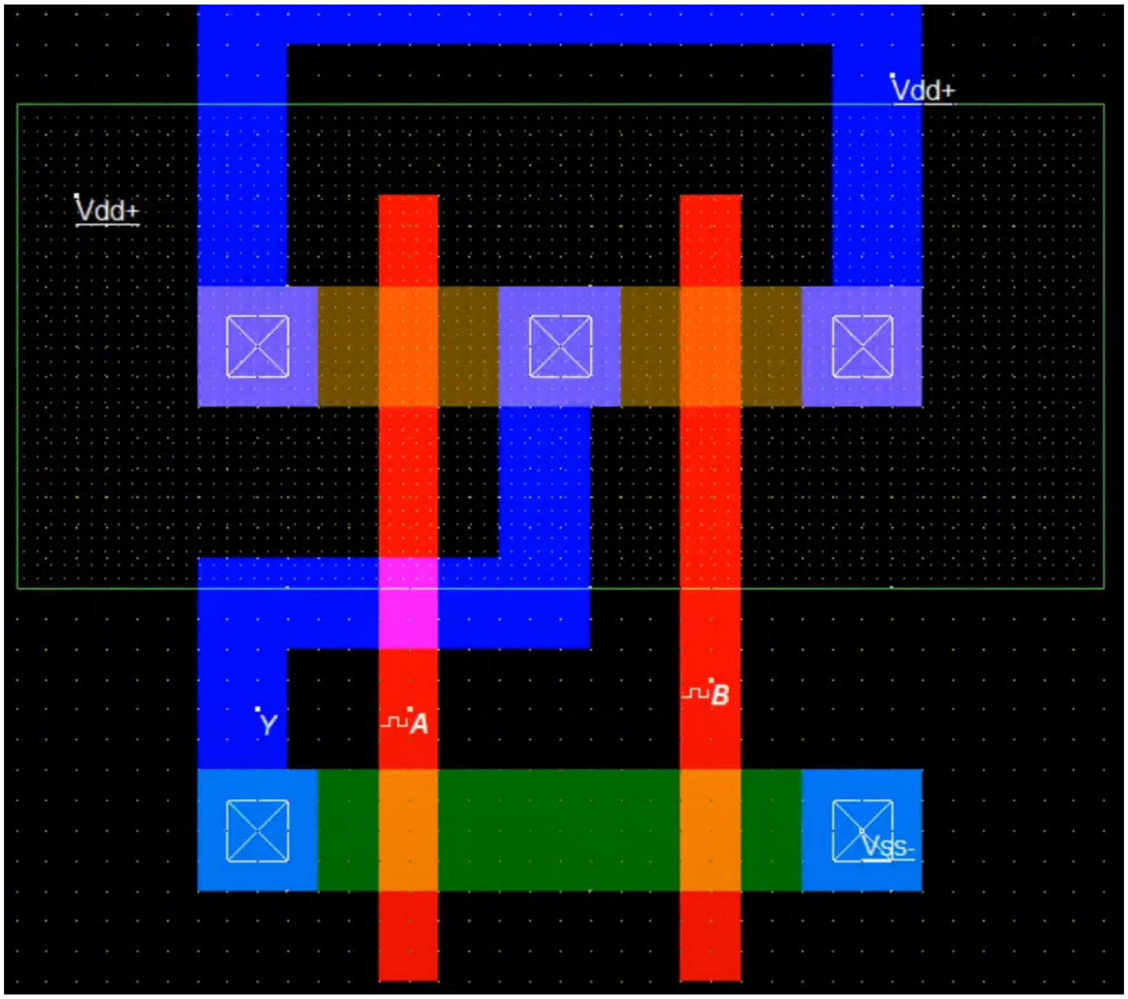
EXP 6:

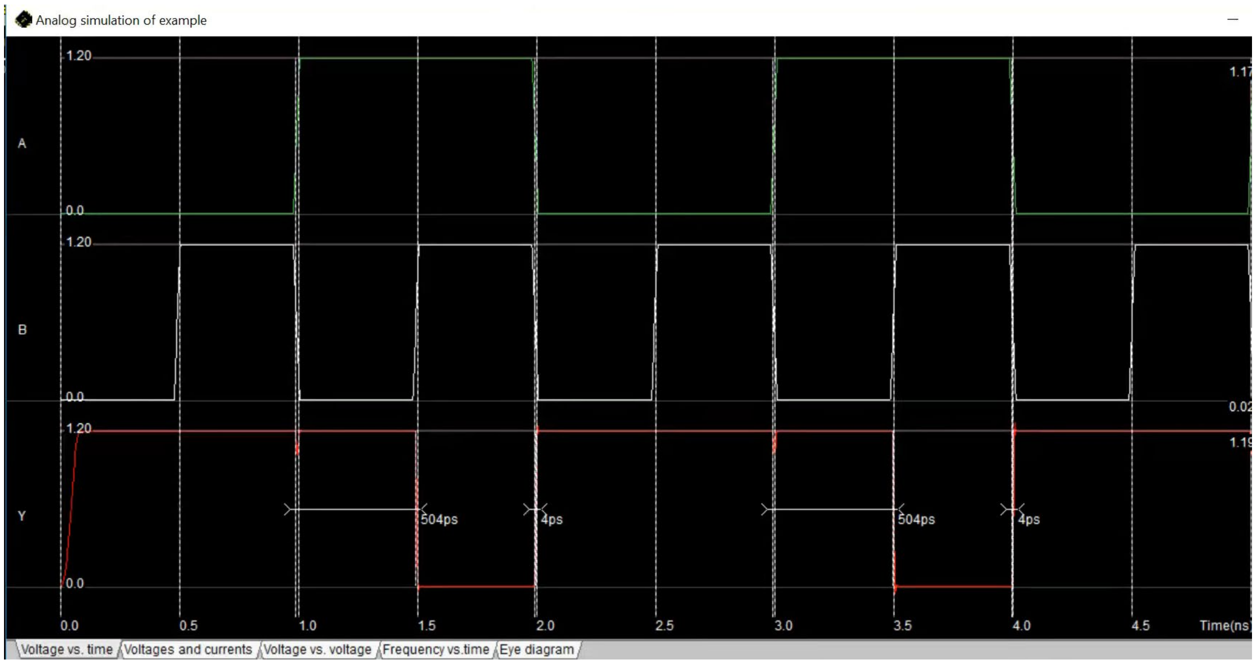
Inverter



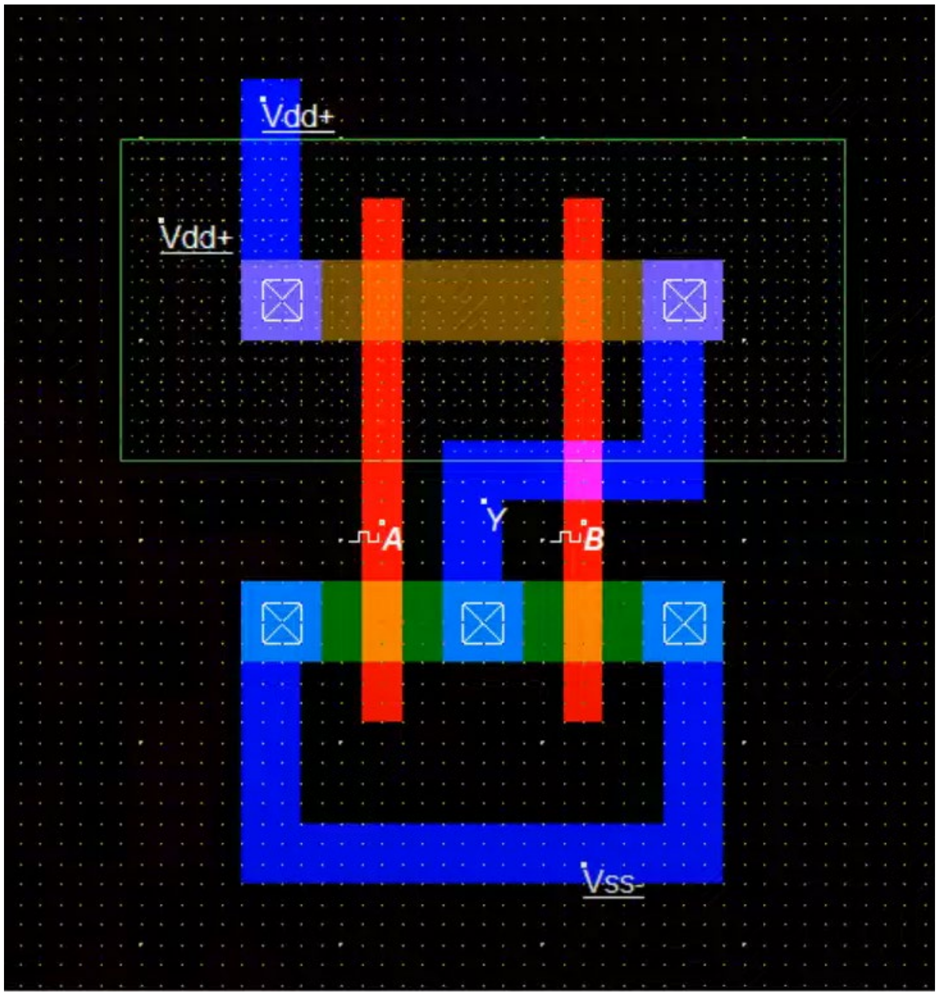


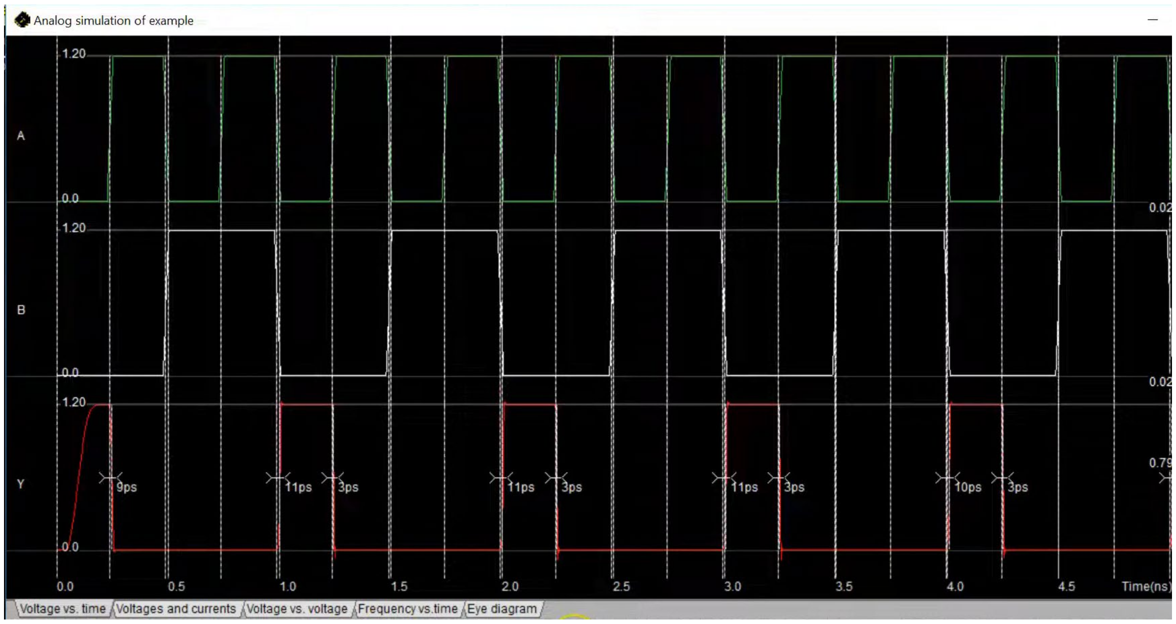
NAND


Vdd+
Vdd+
Y
A
B
Vss-

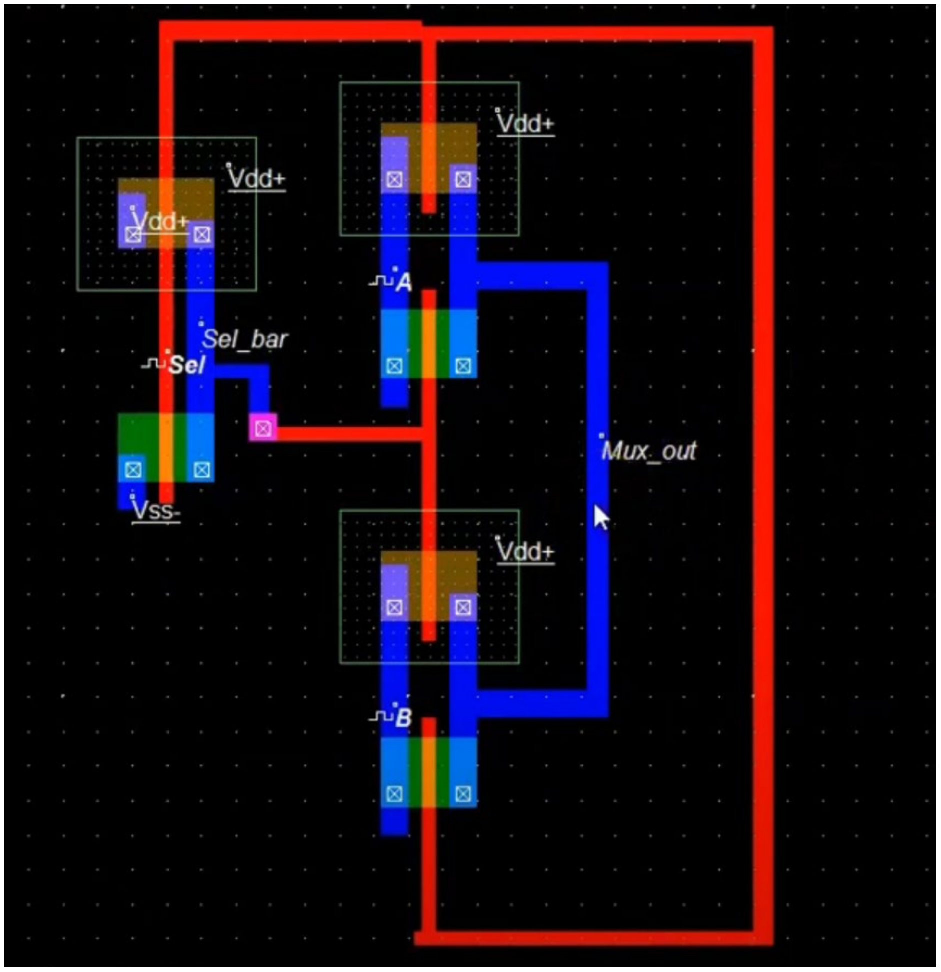

Analog simulation of example
1.20
0.0
A
B
Y
504ps
4ps
504ps
4ps
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
Time(ns
Voltage vs. time
Voltages and currents
Voltage vs. voltage
Frequency vs.time
Eye diagram

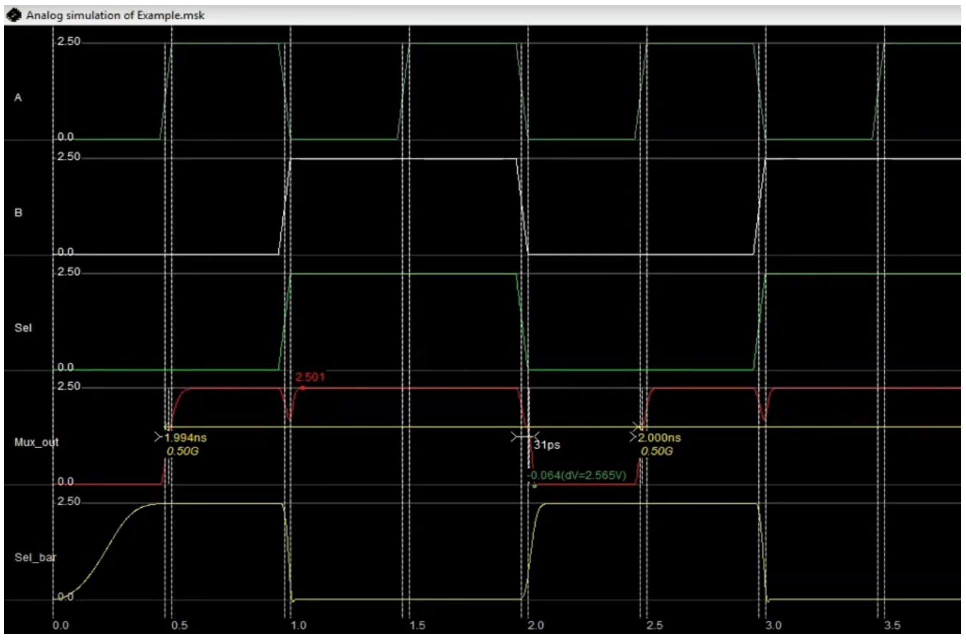
NOR


Vdd+
Vdd+
A
Y
B
Vss-

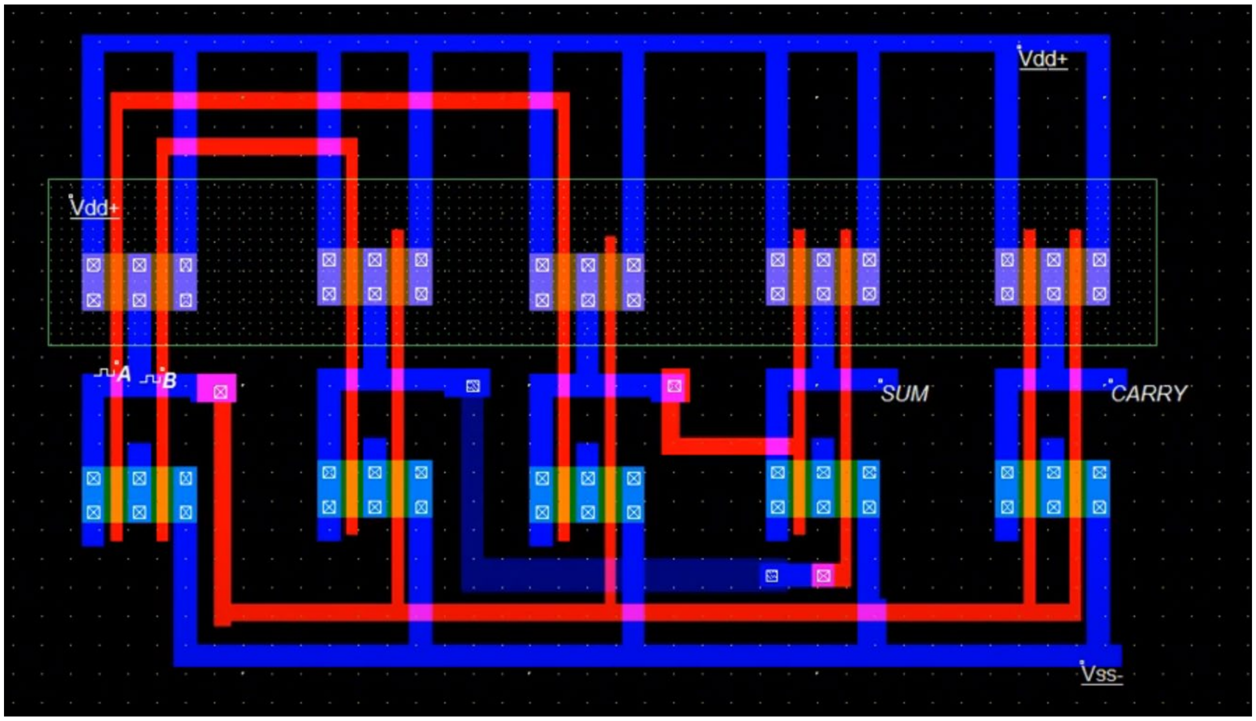

Analog simulation of example
A
B
Y
1.20
0.0
9ps
11ps
3ps
10ps
0.02
0.79
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
Time(ns)
Voltage vs. time
Voltages and currents
Voltage vs. voltage
Frequency vs.time
Eye diagram

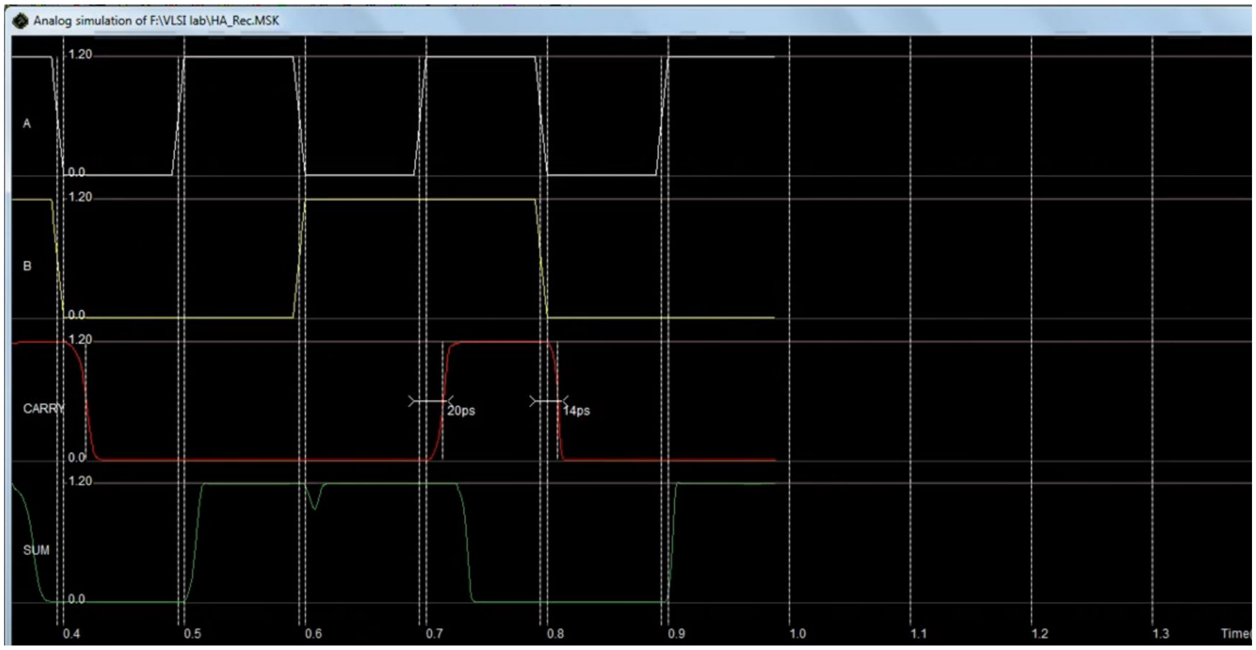
EXP 7


Vdd+
Vdd+
Vdd+
Vdd+
A
Sel_bar
Sel
Vss-
Mux_out
B

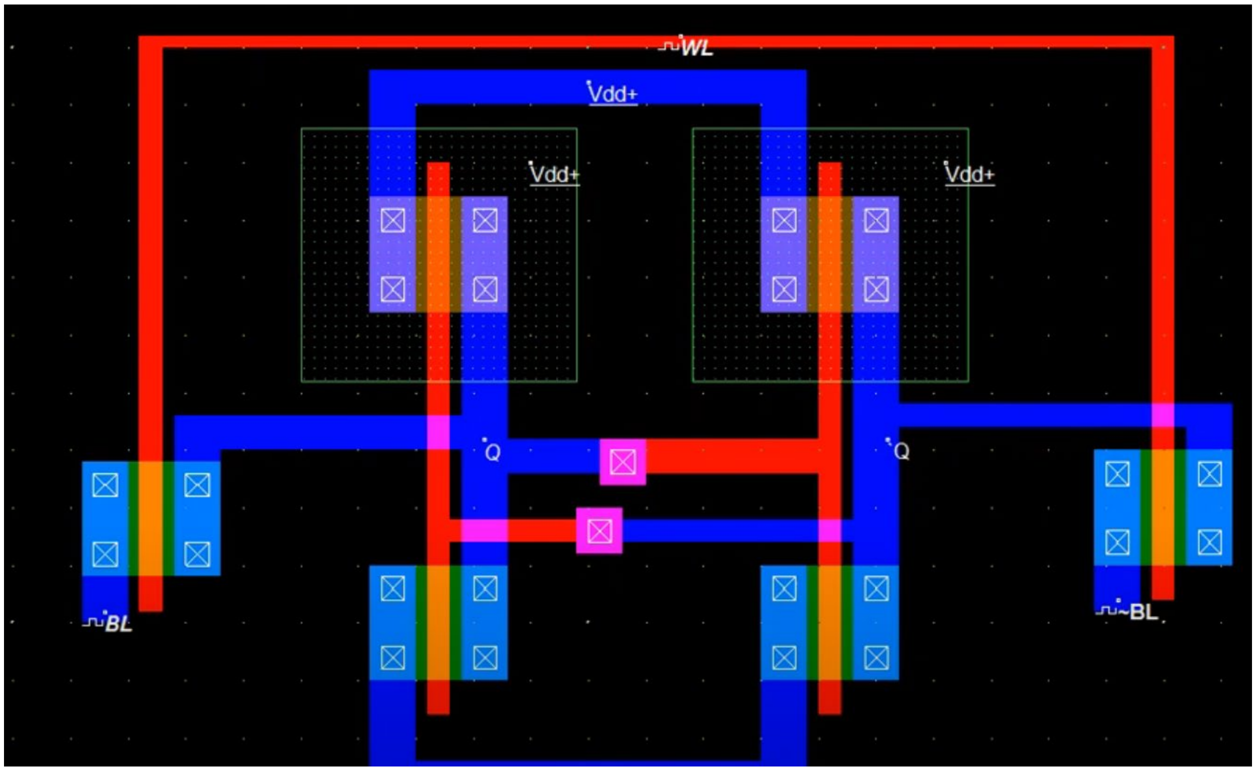

Analog simulation of Example.msk
A
B
Sel
Mux_out
Sel_bar
2.50
0.0
2.501
1.994ns
0.50G
31ps
-0.064(dV=2.565V)
2.000ns
0.50G
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

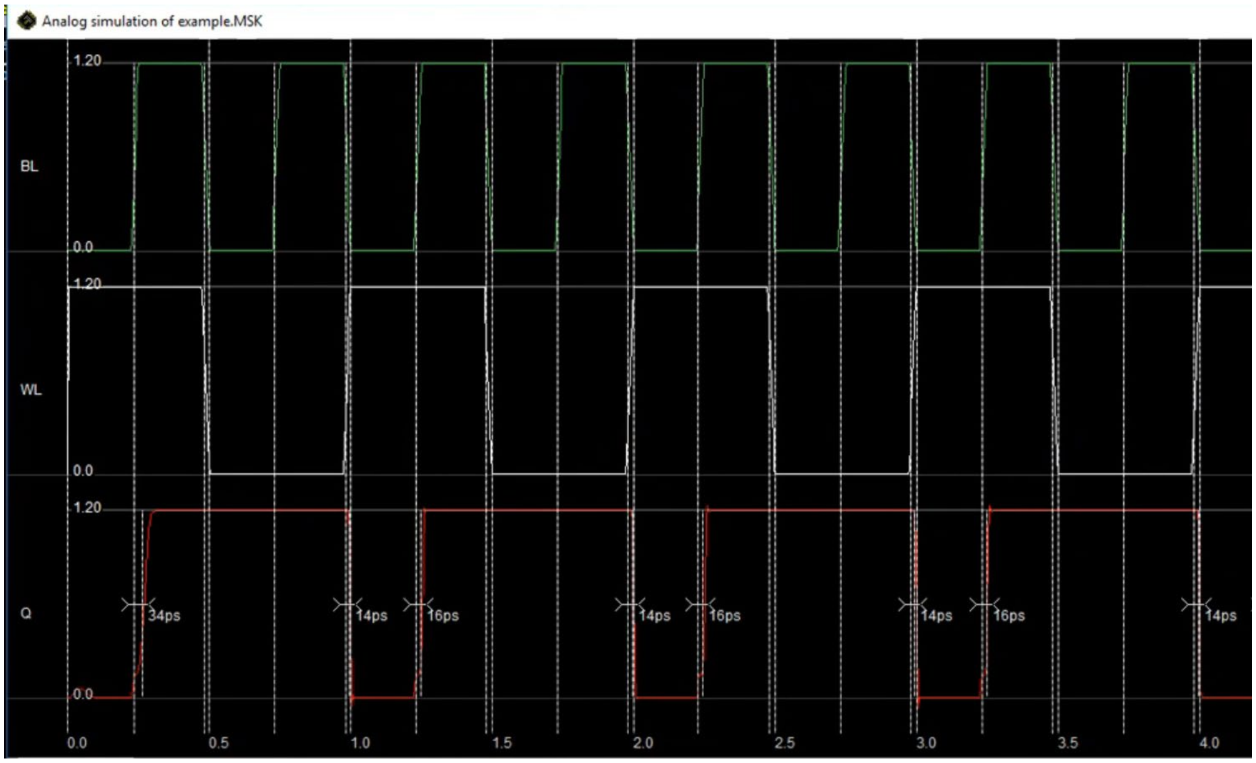
EXP 8:




Analog simulation of F:\VLSI lab\HA_Rec.MSK

EXP 9:


WL
Vdd+
Vdd+
Vdd+
Q
Q
BL
~BL


Analog simulation of example.MSK
BL
WL
Q
1.20
0.0
34ps
14ps
16ps
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

EXP 5

Code:

```
-- lcd_4bit_simple.vhd
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity lcd_4bit_simple is
  generic (CLK_FREQ : integer := 50000000); -- set to your board clock
  port (
    clk    : in  std_logic;
    rst_n  : in  std_logic;        -- active low reset
    lcd_rs : out std_logic;
    lcd_e  : out std_logic;
    lcd_d  : out std_logic_vector(3 downto 0)
    -- Note: Tie RW pin of LCD to GND in hardware (write-only)
  );
end entity;

architecture rtl of lcd_4bit_simple is
  -- timing (in clock cycles)
  constant T15MS : integer := integer(0.015 * real(CLK_FREQ));
  constant T4MS  : integer := integer(0.004 * real(CLK_FREQ));
  constant T40US : integer := integer(0.00004 * real(CLK_FREQ));
  constant PULSE : integer := integer(0.000002 * real(CLK_FREQ));
```

```
type state_t is (
  RESET_WAIT, SEND_INIT1, SEND_INIT2, SET_4BIT,
  FUNC_SET, DISP_ON, CLR, ENTRY, IDLE, SEND_BYTE_H, PULSE_EN, SEND_BYTE_L, WAIT
);
signal state : state_t := RESET_WAIT;

-- small ROM for init & message (commands with RS=0; data with RS=1)
type rom_rec is record data: std_logic_vector(7 downto 0); rs: std_logic; end record;
constant ROM : array(0 to 15) of rom_rec := (
  -- initialization sequence (commands)
  (x"28",'0'), -- Function set: 4-bit, 2 lines, 5x8
  (x"0C",'0'), -- Display ON, cursor off
  (x"01",'0'), -- Clear display
  (x"06",'0'), -- Entry mode: increment
  -- message "HELLO FPGA" (ASCII)
  (x"48",'1'), (x"45",'1'), (x"4C",'1'), (x"4C",'1'), (x"4F",'1'),
  (x"20",'1'), (x"46",'1'), (x"50",'1'), (x"47",'1'), (x"41",'1'),
  (x"20",'1'), (x"20",'1'), (x"00",'0'), (x"00",'0'), (x"00",'0'), (x"00",'0')
);
signal addr : integer range 0 to ROM'length-1 := 0;

signal cnt    : integer := 0;
signal en_cnt : integer := 0;
signal cur8   : std_logic_vector(7 downto 0) := (others=>'0');
signal rs_s   : std_logic := '0';
signal d_s    : std_logic_vector(3 downto 0) := (others=>'0');
```

```
  signal e_s    : std_logic := '0';
  signal hi     : boolean := true;

begin
  lcd_rs <= rs_s;
  lcd_d  <= d_s;
  lcd_e  <= e_s;

  process(clk, rst_n)
  begin
    if rst_n='0' then
      state <= RESET_WAIT;
      cnt <= 0; en_cnt <= 0; addr <= 0; cur8 <= (others=>'0');
      rs_s <= '0'; d_s <= (others=>'0'); e_s <= '0'; hi <= true;
    elsif rising_edge(clk) then
      case state is
        when RESET_WAIT =>
          if cnt < T15MS then
            cnt <= cnt + 1;
          else
            cnt <= 0;
            addr <= 0;
            -- begin sending ROM entries
            cur8 <= ROM(0).data;
            rs_s <= ROM(0).rs;
            hi <= true;
    state <= SEND_BYTE_H;

  end if;


when SEND_BYTE_H =>

  d_s <= cur8(7 downto 4);

  -- pulse E

  if en_cnt = 0 then e_s <= '1'; en_cnt <= en_cnt + 1;

  elsif en_cnt < PULSE then en_cnt <= en_cnt + 1;

  else e_s <= '0'; en_cnt <= 0; state <= SEND_BYTE_L; end if;


when SEND_BYTE_L =>

  d_s <= cur8(3 downto 0);

  if en_cnt = 0 then e_s <= '1'; en_cnt <= en_cnt + 1;

  elsif en_cnt < PULSE then en_cnt <= en_cnt + 1;

  else e_s <= '0'; en_cnt <= 0; cnt <= 0; state <= WAIT; end if;


when WAIT =>

  if cnt < T40US then cnt <= cnt + 1;

  else

    -- advance ROM

    if addr < ROM'length-1 then

      addr <= addr + 1;

      cur8 <= ROM(addr+1).data;

      rs_s <= ROM(addr+1).rs;

      state <= SEND_BYTE_H;

    else
```

```
            state <= IDLE;
          end if;
          cnt <= 0;
        end if;

      when IDLE =>
        -- keep outputs low (or keep last state); nothing to do
        rs_s <= '0';
        d_s <= (others=>'0');
        e_s <= '0';

      when others =>
        state <= RESET_WAIT;
    end case;
  end if;
 end process;
end architecture;
```

Testbranch:

```
-- tb_lcd_4bit_simple.vhd
library ieee;
use ieee.std_logic_1164.all;

entity tb_lcd_4bit_simple is end entity;
```

```
architecture sim of tb_lcd_4bit_simple is
 signal clk    : std_logic := '0';
 signal rst_n  : std_logic := '0';
 signal lcd_rs : std_logic;
 signal lcd_e  : std_logic;
 signal lcd_d  : std_logic_vector(3 downto 0);
 constant CLK_PERIOD : time := 20 ns; -- 50 MHz
begin
 dut: entity work.lcd_4bit_simple
   generic map (CLK_FREQ => 50000000)
   port map (
     clk => clk, rst_n => rst_n, lcd_rs => lcd_rs, lcd_e => lcd_e, lcd_d => lcd_d
   );

 clk_proc: process begin
   while now < 200 ms loop
     clk <= '0'; wait for CLK_PERIOD/2;
     clk <= '1'; wait for CLK_PERIOD/2;
   end loop;
   wait;
 end process;

 stim: process
 begin
   rst_n <= '0'; wait for 5 ms;
   rst_n <= '1';
```

```
    wait for 120 ms; -- let init and message finish
    assert false report "Simulation finished" severity note;
    wait;
  end process;
end architecture;
```