**Expt 1 (ALU 4-BIT)**

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.numeric\_std.all;

use IEEE.std\_logic\_unsigned.all;

--use UNISIM.VComponents.all;

entity Hi is

Port ( a : in STD\_LOGIC\_VECTOR (3 downto 0);

b : in STD\_LOGIC\_VECTOR (3 downto 0);

sel : in STD\_LOGIC\_VECTOR (2 downto 0);

y : out STD\_LOGIC\_VECTOR (3 downto 0));

end Hi;

architecture Behavioral of Hi is

begin

process(a,b,sel)

begin

case sel is

when "000"=>y<=a + b;

when "001"=>y<=a - b;

when "010"=>y<=a and b;

when "011"=>y<=a or b;

when "100"=>y<=a nand b;

when "101"=>y<=a nor b;

when "110"=>y<= not b;

when "111"=>y<=a;

when others=> null;

end case;

end process;

end Behavioral;

TEST BENCH

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

--USE ieee.numeric\_std.ALL;

ENTITY HIII IS

END HIII;

ARCHITECTURE behavior OF HIII IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT Hi

PORT(

a : IN std\_logic\_vector(3 downto 0);

b : IN std\_logic\_vector(3 downto 0);

sel : IN std\_logic\_vector(2 downto 0);

y : OUT std\_logic\_vector(3 downto 0)

);

END COMPONENT;

--Inputs

signal a : std\_logic\_vector(3 downto 0) := (others => '0');

signal b : std\_logic\_vector(3 downto 0) := (others => '0');

signal sel : std\_logic\_vector(2 downto 0) := (others => '0');

--Outputs

signal y : std\_logic\_vector(3 downto 0);

-- No clocks detected in port list. Replace <clock> below with

-- appropriate port name

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: Hi PORT MAP (

a => a,

b => b,

sel => sel,

y => y

);

-- Clock process definitions

-- <clock>\_process :process

--begin

--<clock> <= '0';

---wait for <clock>\_period/2;

--<clock> <= '1';

--wait for <clock>\_period/2;

--end process;

-- Stimulus process

stim\_proc: process

begin

-- hold reset state for 100 ns.

a<="0101";

b<="0100";

sel<="000";

wait for 100 ns;

a<="0101";

b<="0100";

sel<="001";

wait for 100 ns;

a<="0101";

b<="0100";

sel<="010";

wait for 100 ns;

a<="0101";

b<="0100";

sel<="011";

wait for 100 ns;

a<="0101";

b<="0100";

sel<="100";

wait for 100 ns;

a<="0101";

b<="0100";

sel<="101";

wait for 100 ns;

a<="0101";

b<="0100";

sel<="110";

wait for 100 ns;

a<="0101";

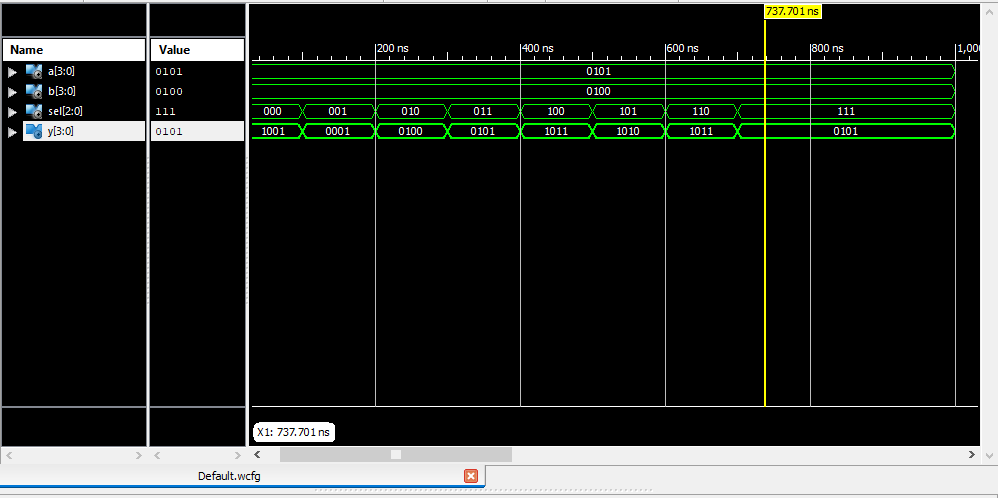
b<="0100";

sel<="111";

wait;

end process;

END;



**Expt 2 ( USR SISO, PISO,SIPO,PIPO)**

library IEEE;

use IEEE STD\_LOGIC\_1164 ALL;

entity shiftreg is

Port ( si: in STD\_LOGIC;

clk: in STD LOGIC,

so: out STD LOGIC;

pin: in STD\_LOGIC\_VECTOR (3 downto 0);

po: out STD\_LOGIC\_VECTOR (3 downto 0);

sel: in STD\_LOGIC\_VECTOR (1 downto 0));

end shiftreg,

architecture Behavioral of shiftreg is

signal temp:STD\_LOGIC\_VECTOR( 3 downto 0);

begin

process(clk)

begin

if(clk'event and clk='1')then

case sel is

when"00"=> temp<= si&temp(3 downto 1);

so <=temp(3);

when"01"=> temp<= si&temp(3 downto 1);

po<=temp;

when others=>null;

end case;

end if;

end process;

end Behavioral;

TEST BENCH

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY shifttest IS

END shifttest;

ARCHITECTURE behavior OF shifttest IS

COMPONENT shiftreg

PORT(

si: IN std\_logic;

clk: IN std\_logic;

so: OUT std\_logic;

pin: IN std\_logic\_vector(3 downto 0);

po: OUT std\_logic\_vector(3 downto 0);

sel: IN std\_logic\_vector(1 downto 0) );

END COMPONENT,

--Inputs

signal si: std\_logic := '0';

signal clk: std\_logic := '0';

signal pin: std\_logic\_vector(3 downto 0) := (others => '0');

signal sel: std\_logic\_vector(1 downto 0) := (others => '0');

--Outputs

signal so: std\_logic,

signal po: std\_logic\_vector(3 downto 0);

constant cik\_period: time := 10 ns;

BEGIN

uut: shiftreg PORT MAP (

si =>si,

clk => clk,

so =>so,

pin => pin,

po =>po,

sel => sel

);

clk\_process:process

begin

clk <= '0';

wait for clk\_period/2;

clk <= ‘1’;

wait for clk\_period/2;

end process;

--Stimulus process

stim\_proc: process

begin

sel<="00";

si <=’1’;

-hold reset state for 100 ns.

wait for 100 ns;

sel<="01";

si <=’1’;

wait for 100 ns;

sel<="10";

pin<="1100";

wait for 100 ns;

sel<="11";

pin<="1100";

--=wait for clk\_period\*10;

wait;

end process;

END;

Experiment No: 3 ( Mod N Counter )

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

useIEEE.STD\_LOGIC\_unsigned.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entitymodncount is

Port ( clk : in STD\_LOGIC;

clr : in STD\_LOGIC;

q :inout STD\_LOGIC\_VECTOR (2 downto 0));

endmodncount;

architecture Behavioral of modncount is

signal count: std\_logic\_vector(2 downto 0);

begin

process(clk)

begin

if (clr='1') then count <= "000";

elsif (rising\_edge (clk)) then

if (count="100")

then count<= "000";

else

count<=count+ 1;

end if;

end if;

end process;

q<=count;

end Behavioral;

**Testbench:**

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--USE ieee.numeric\_std.ALL;

ENTITY hhh IS

END hhh;

ARCHITECTURE behavior OF hhh IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT modncount

PORT(

clk : IN std\_logic;

clr : IN std\_logic;

q : INOUT std\_logic\_vector(2 downto 0)

);

END COMPONENT;

--Inputs

signalclk : std\_logic := '0';

signalclr : std\_logic := '0';

--BiDirs

signal q : std\_logic\_vector(2 downto 0);

-- Clock period definitions

-- constantclk\_period : time := 10 ns;

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: modncount PORT MAP (

clk =>clk,

clr =>clr,

q => q

);

-- Clock process definitions

clk\_process :process

begin

clk<= '0';

wait for 10 ns;

clk<= '1';

wait for 10 ns;

end process;

-- Stimulus process

stim\_proc: process

begin

clr<='1';

-- hold reset state for 100 ns.

wait for 20 ns;

clr<='0';

-- hold reset state for 100 ns.

wait for 20 ns;

-- hold reset state for 100 ns.

-- wait for 100 ns;

--

-- wait for clk\_period\*10;

-- insert stimulus here

wait;

end process;

END;

**UCF**

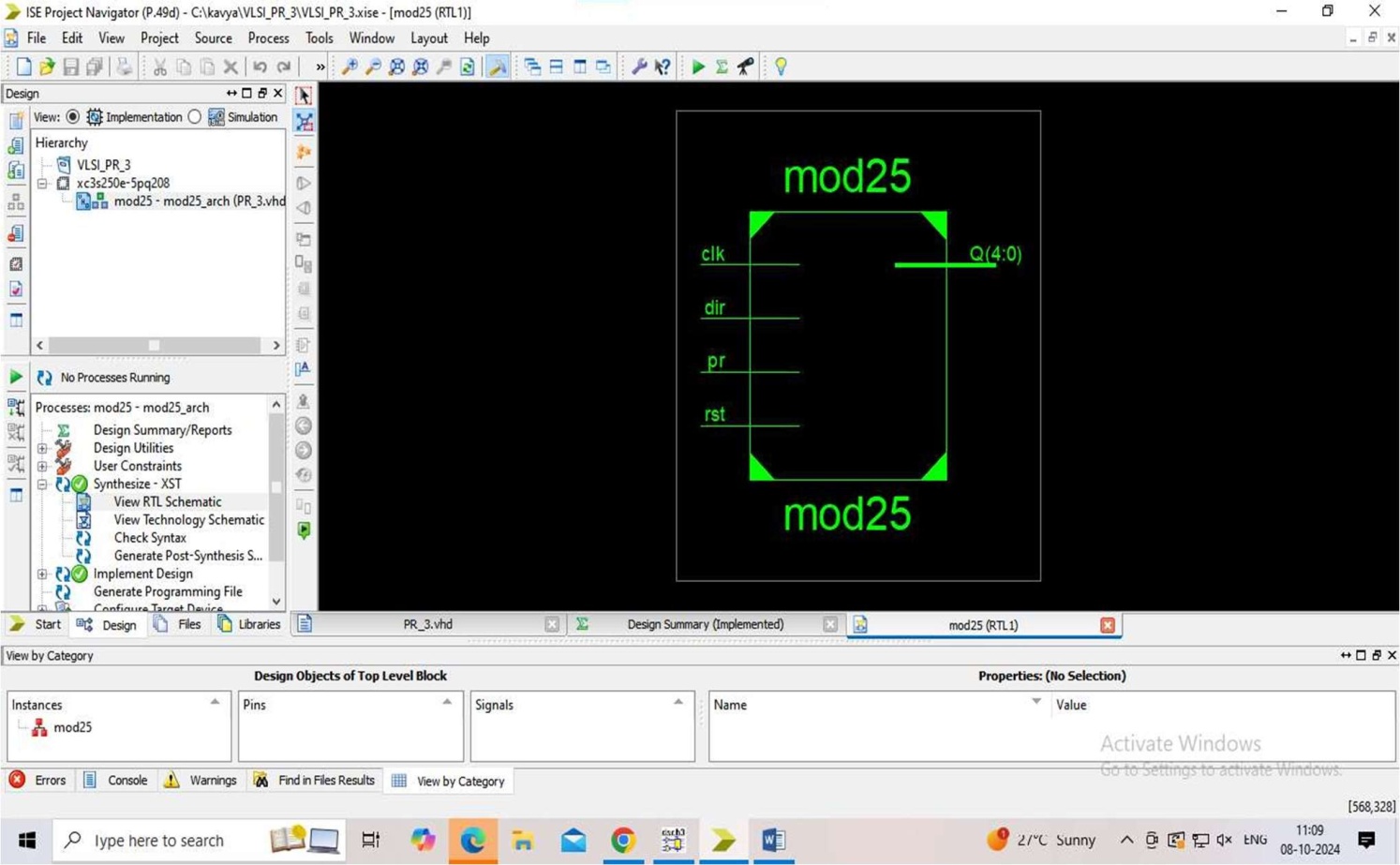
Net clkloc=”p182”;

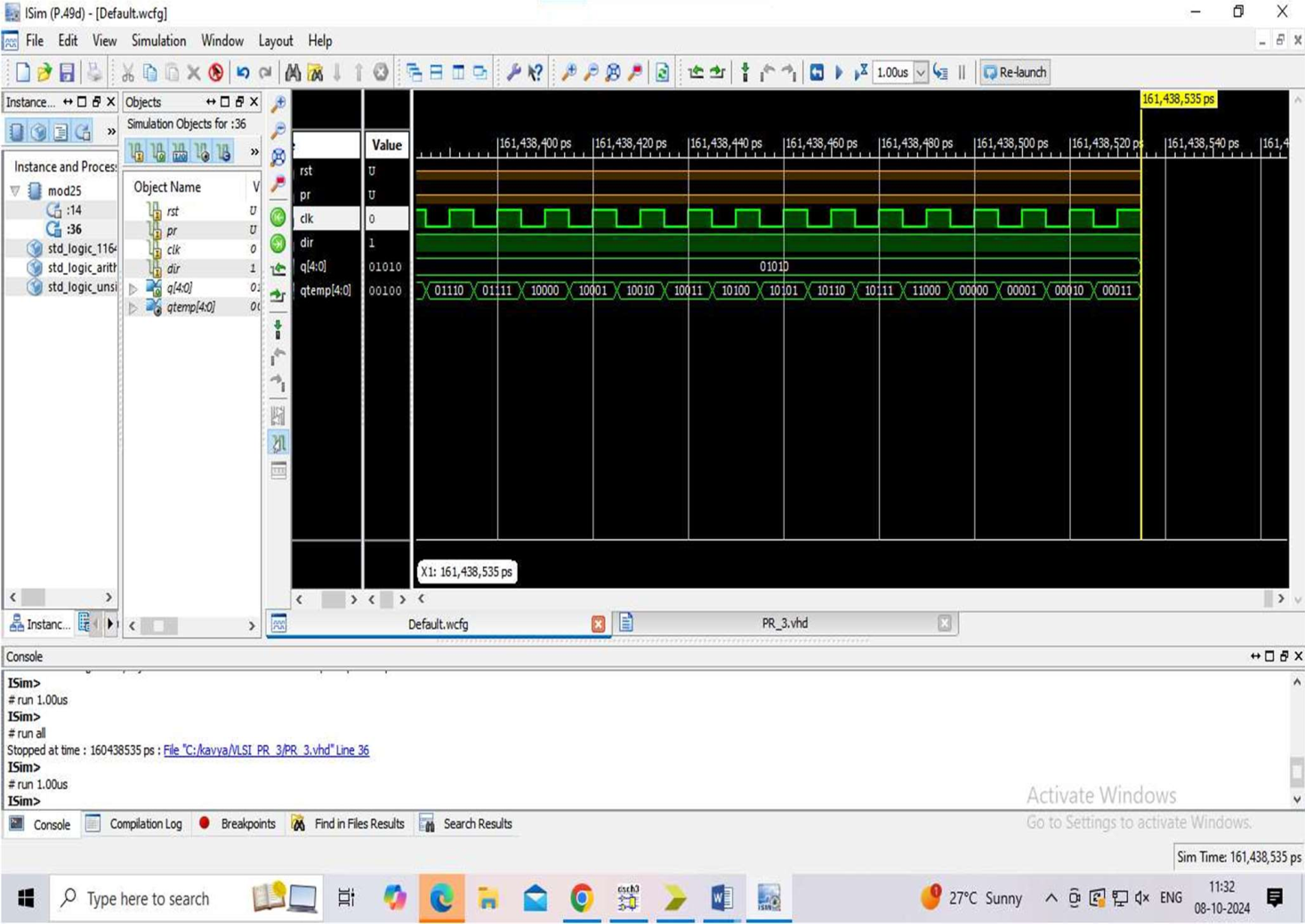
Net clrloc=”p102”;

Net q(2) loc=”p165”;

Net q(2) loc=”p166”;

Net q(2) loc=”p167”;





**Experiment 4 ( FIFO )**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating

---- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity FIFO is

Port ( DI : in std\_logic\_vector(3 downto 0);

DO : out std\_logic\_vector(3 downto 0);

RW : in std\_logic;

FULL : out std\_logic;

RS : in std\_logic;

CLK : in std\_logic);

end FIFO;

architecture Behavioral of FIFO is

begin

process(RS,CLK)

type memory is Array( 0 to 3)of std\_logic\_vector(3 downto 0);

variablemem: memory;

variable r\_p:integer range 0 to 3;

variable w\_p:integer range 0 to 3;

variable overwrite :boolean;

begin

if RS='1' then

DO<="0000";

elsif(CLK'event and CLK='1') then

if RW='1'then

if (overwrite=False OR w\_p/=r\_p) then

mem (w\_p):=DI;

ifw\_p=3 then

w\_p:=0 ;

else

w\_p:=w\_p+1;

overwrite:=False;

end if;

end if;

elseif RW='0' then

if (overwrite=False OR w\_p/=r\_p) then

DO<=mem(r\_p);

if (r\_p=3 ) then

r\_p:=0 ;

overwrite:=true;

else

r\_p:=r\_p+1;

end if;

end if;

end if;

if w\_p=r\_p then

if overwrite=true then

FULL<='1';

else

FULL<='0';

end if;

else

FULL<='0';

end if;

end if;

end process;

end Behavioral;

**Testbench:**

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--USE ieee.numeric\_std.ALL;

ENTITY SSSA\_vhd IS

END SSSA\_vhd;

ARCHITECTURE behavior OF SSSA\_vhd IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT fifo

PORT(

DI : IN std\_logic\_vector(3 downto 0);

RW : IN std\_logic;

RS : IN std\_logic;

CLK : IN std\_logic;

DO : OUT std\_logic\_vector(3 downto 0);

FULL : OUT std\_logic

);

END COMPONENT;

--Inputs

SIGNAL RW :std\_logic := '0';

SIGNAL RS :std\_logic := '0';

SIGNAL CLK :std\_logic := '0';

SIGNAL DI :std\_logic\_vector(3 downto 0) := (others=>'0');

--Outputs

SIGNAL DO :std\_logic\_vector(3 downto 0);

SIGNAL FULL :std\_logic;

constant CLK\_period : time := 10 ns;

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: fifo PORT MAP(

DI => DI,

DO => DO,

RW => RW,

FULL => FULL,

RS => RS,

CLK => CLK

);

process

begin

CLK<='0' ;

wait for CLK\_period/2;

CLK<='1' ;

wait for CLK\_period/2;

end process;

tb : PROCESS

BEGIN

RS<='1';

wait for 100 ns;

RS<='0';

RW<='1';

DI<="0011";

-- Wait 100 ns for global reset to finish

wait for 10 ns;

RW<='1';

DI<="0110";

-- Wait 100 ns for global reset to finish

wait for 10 ns;

RW<='1';

DI<="1100";

-- Wait 100 ns for global reset to finish

wait for 10 ns;

RW<='1';

DI<="1001";

-- Wait 100 ns for global reset to finish

wait for 10 ns;

-- Place stimulus here

RW<='0';

-- Wait 100 ns for global reset to finish

wait for 10 ns;

RW<='0';

-- Wait 100 ns for global reset to finish

wait for 10 ns;

RW<='0';

-- Wait 100 ns for global reset to finish

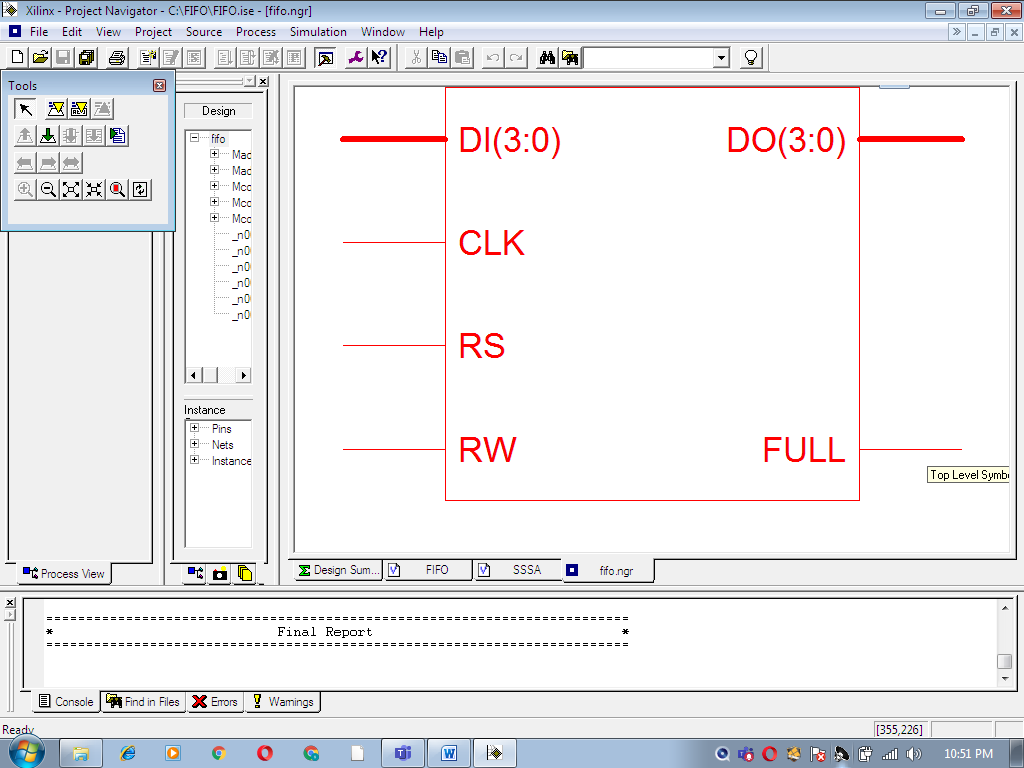
wait for 10 ns;

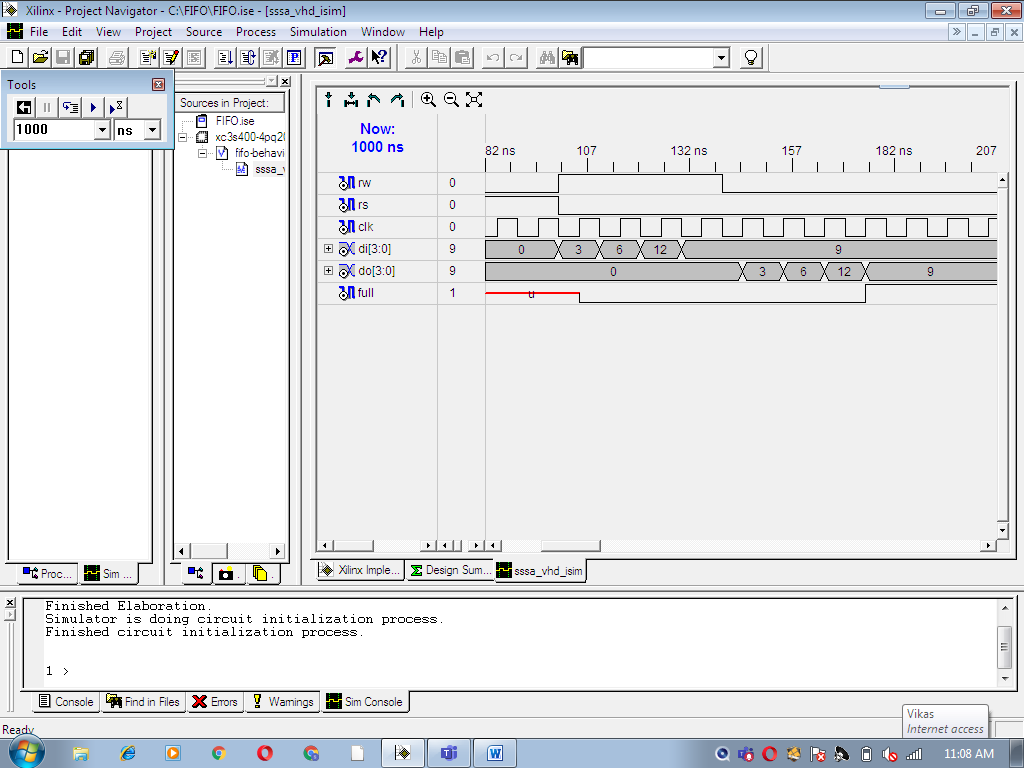
WAIT FOR CLK\_period\*10;

wait; **-- will wait forever**

END PROCESS;

END;





**Experiment 5 ( LCD )**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity lcd is

Port ( clk,reset : in std\_logic;

RS,EN,RW : out std\_logic;

data : out std\_logic\_vector(7 downto 0));

end lcd;

architecture Behavioral of lcd is

type state\_type is (s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,

s12,s13,s14,s15,s16,s17,s18,s19,s20,s21,s22,s23);

signal state:state\_type;

SIGNAL count:std\_logic\_vector(22 downto 0);

SIGNAL clk1:std\_logic;

begin

process(Clk,Reset)

begin

if(Clk' event AND Clk='1')then

count<=count+"0001";

end if;

clk1<=count(20);

end process;

RW<='0';

process(clk1, reset)

begin

if reset = '1' then

state <= s0;

elsif rising\_edge(clk1) then

if state = s0 then

state <= s1;

RS<='0'; -- Write commonds to LCD.

EN <= '1';

data <= "00110000"; -- Function set for 8 bit interface, 1 line mode and 5x7 dot matrix.

end if;

if state = s1 then

state <= s2;

EN <= '0';

end if;

if state = s2 then

state <= s3;

EN <= '1';

data <= "00001111"; -- Display cursor and blinking ON.

end if;

if state = s3 then

state <= s4;

EN <= '0';

end if;

if state = s4 then

state <= s5;

EN <= '1';

data <= "00000001"; -- Clear display.

end if;

if state = s5 then

state <= s6;

EN <= '0';

end if;

if state = s6 then

state <= s7;

EN <= '1';

data <= "10000100"; -- Display address.

end if;

if state = s7 then

state <= s8;

EN <= '0';

end if;

if state = s8 then

RS <= '1'; -- Write data to LCD.

state <= s9;

EN <= '1';

data <= "00101010"; --(\*)

end if;

if state = s9 then

state <= s10;

EN <= '0';

end if;

if state = s10 then

state <= s11;

EN <= '1';

data <= "01010011"; --S

end if;

if state = s11 then

state <= s12;

EN <= '0';

end if;

if state = s12 then

state <= s13;

EN <= '1';

data <= "01001011"; --K

end if;

if state = s13 then

state <= s14;

EN <= '0';

end if;

if state = s14 then

state <= s15;

EN <= '1';

data <= "01001110"; --N

end if;

if state = s15 then

state <= s16;

EN <= '0';

end if;

if state = s16 then

state <= s17;

EN <= '1';

data <= "01000011"; --C

end if;

if state = s17 then

state <= s18;

EN <= '0';

end if;

if state = s18 then

state <= s19;

EN <= '1';

data <= "01001111"; --O

end if;

if state = s19 then

state <= s20;

EN <= '0';

end if;

if state = s20 then

state <= s21;

EN <= '1';

data <= "01000101"; --E

end if;

if state = s21 then

state <= s22;

EN <= '0';

end if;

if state = s22 then

state <= s23;

EN <= '1';

data <= "00101010"; --(\*)

end if;

if state = s23 then

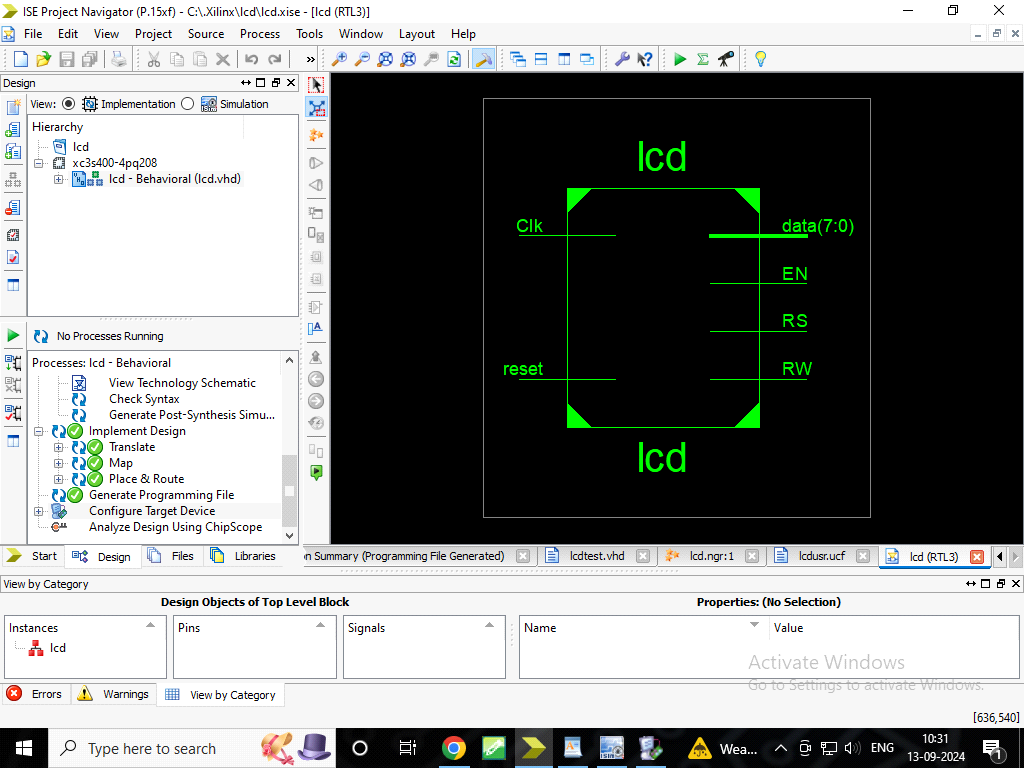
EN <= '0';

end if;

end if;

end process;

end Behavioral;



**Testbench:**

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--USE ieee.numeric\_std.ALL;

ENTITY lcdtest IS

END lcdtest;

ARCHITECTURE behavior OF lcdtest IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT lcd

PORT(

clk1 : IN std\_logic;

reset : IN std\_logic;

RS : OUT std\_logic;

EN : OUT std\_logic;

RW : OUT std\_logic;

data : OUT std\_logic\_vector(7 downto 0)

);

END COMPONENT;

--Inputs

signal clk1 : std\_logic := '0';

signal reset : std\_logic := '0';

--Outputs

signal RS : std\_logic;

signal EN : std\_logic;

signal RW : std\_logic;

signal data : std\_logic\_vector(7 downto 0);

-- Clock period definitions

constant clk1\_period : time := 10 ns;

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: lcd PORT MAP (

clk1 => clk1,

reset => reset,

RS => RS,

EN => EN,

RW => RW,

data => data

);

-- Clock process definitions

clk1\_process :process

begin

clk1 <= '0';

wait for clk1\_period/2;

clk1 <= '1';

wait for clk1\_period/2;

end process;

-- Stimulus process

stim\_proc: process

begin

-- hold reset state for 100 ns.

reset<='0';

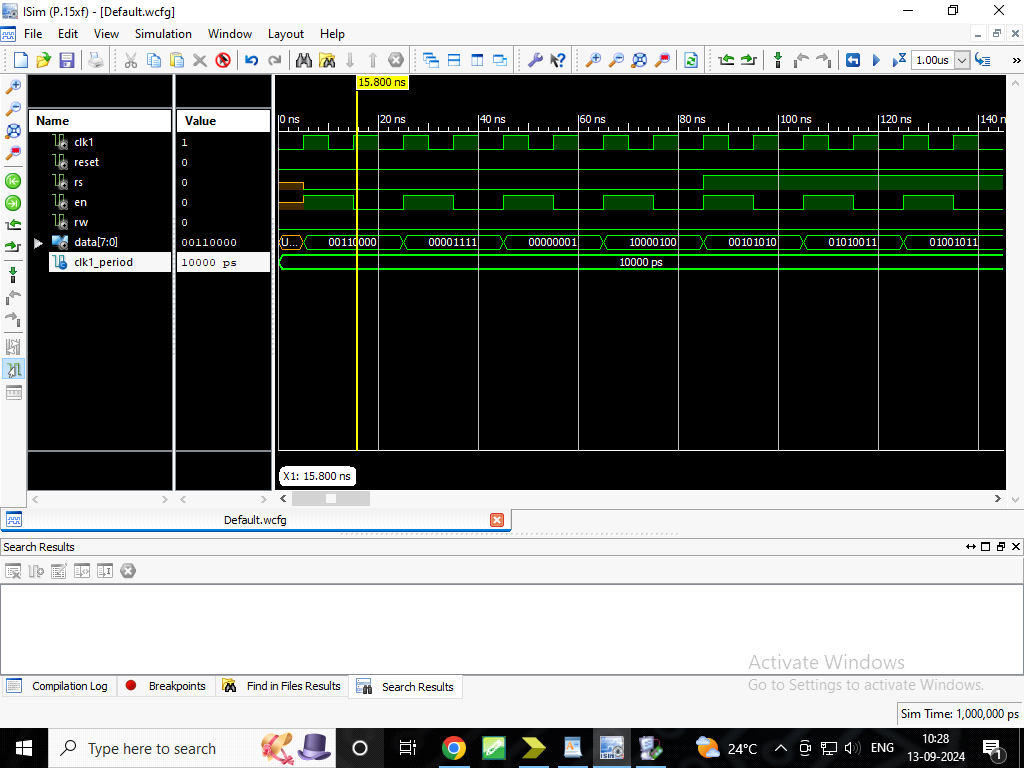
wait for clk1\_period\*10;

-- insert stimulus here

wait;

end process;

END;



**UCF:**

NET data(0) LOC =P62;

NET data(1) LOC =P63;

NET data(2) LOC =P64;

NET data(3) LOC =P65;

NET data(4) LOC =P67;

NET data(5) LOC =P68;

NET data(6) LOC =P71;

NET data(7) LOC =P72;

NET Clk LOC =P183;

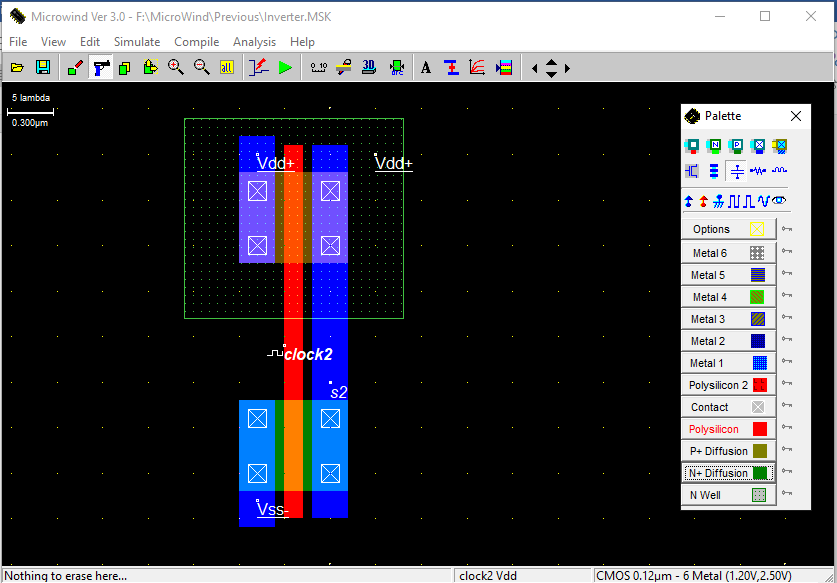
NET reset LOC =P102;

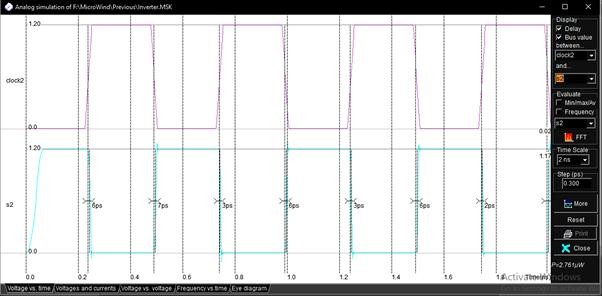
NET RS LOC =P57;

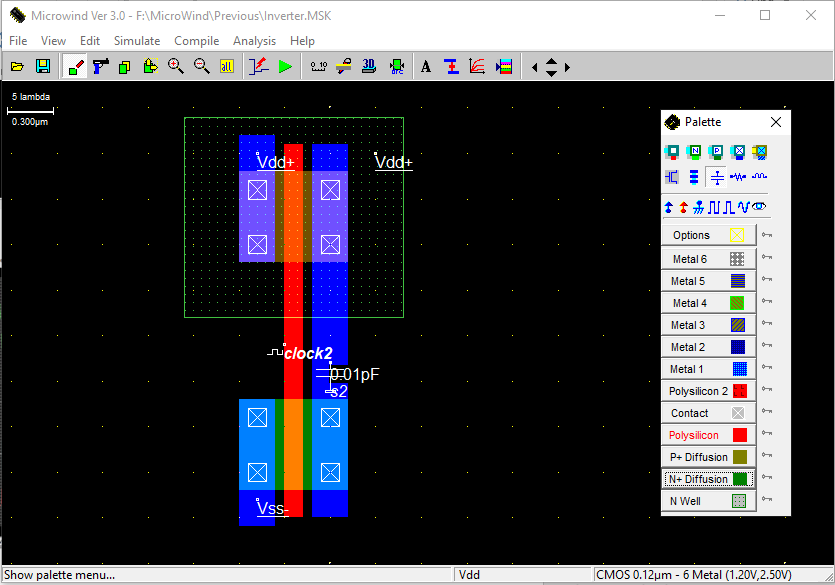
NET EN LOC =P61;

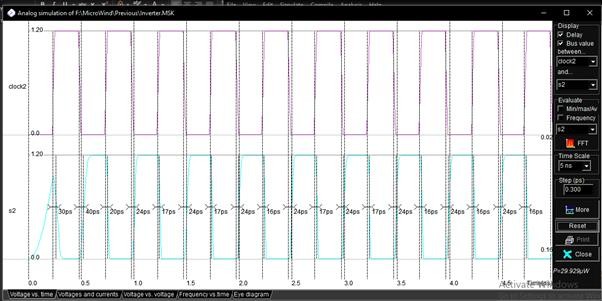
NET RW LOC =P58;

INVERTER (NOT)

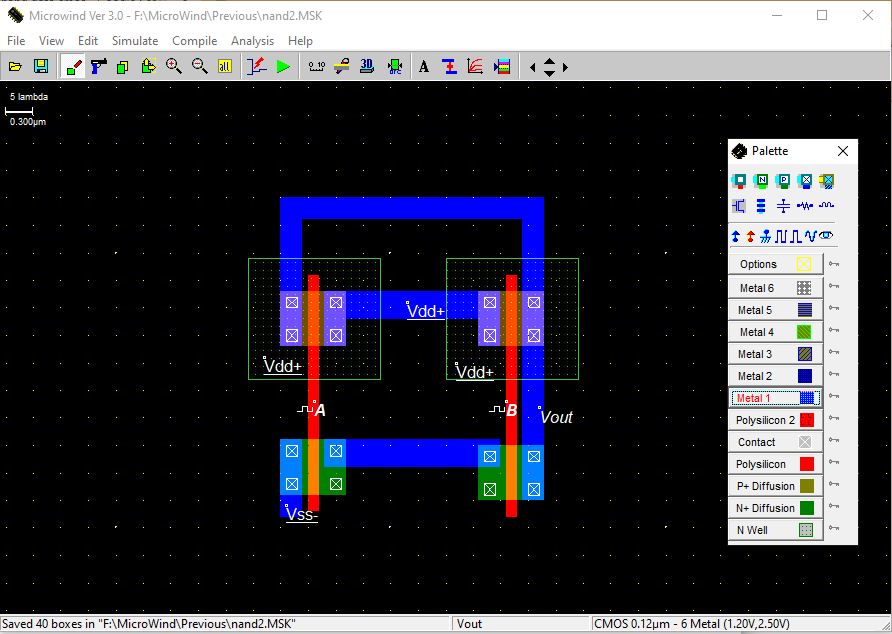


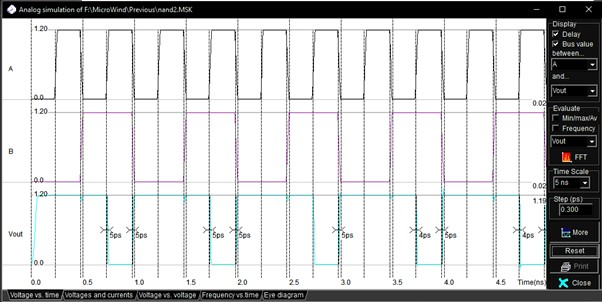


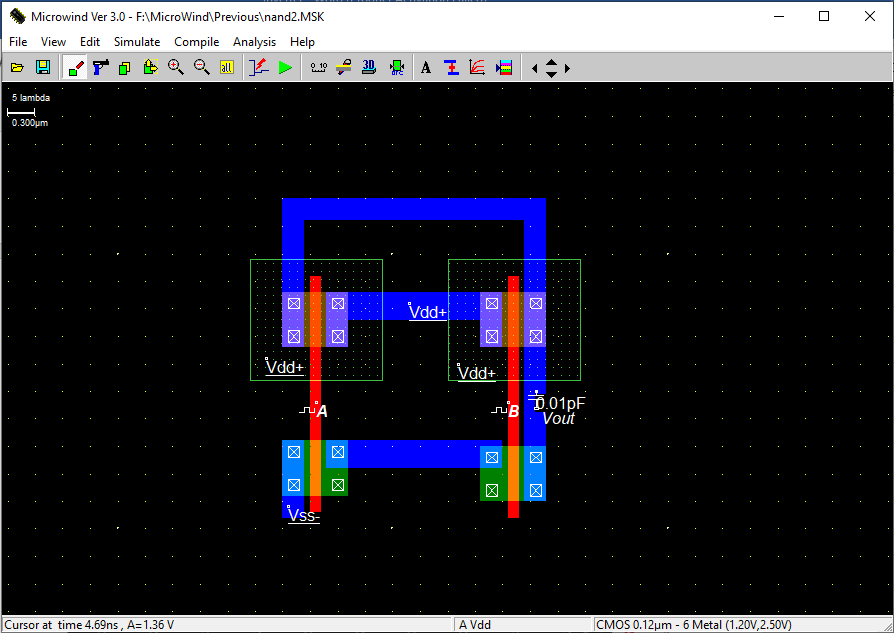
****

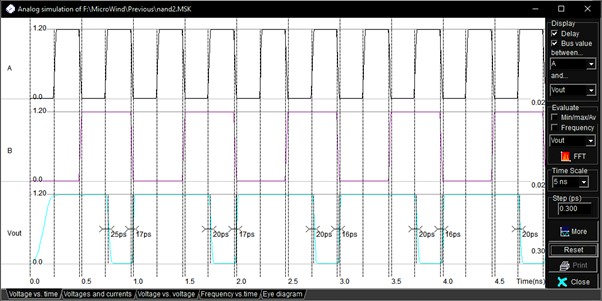


NAND

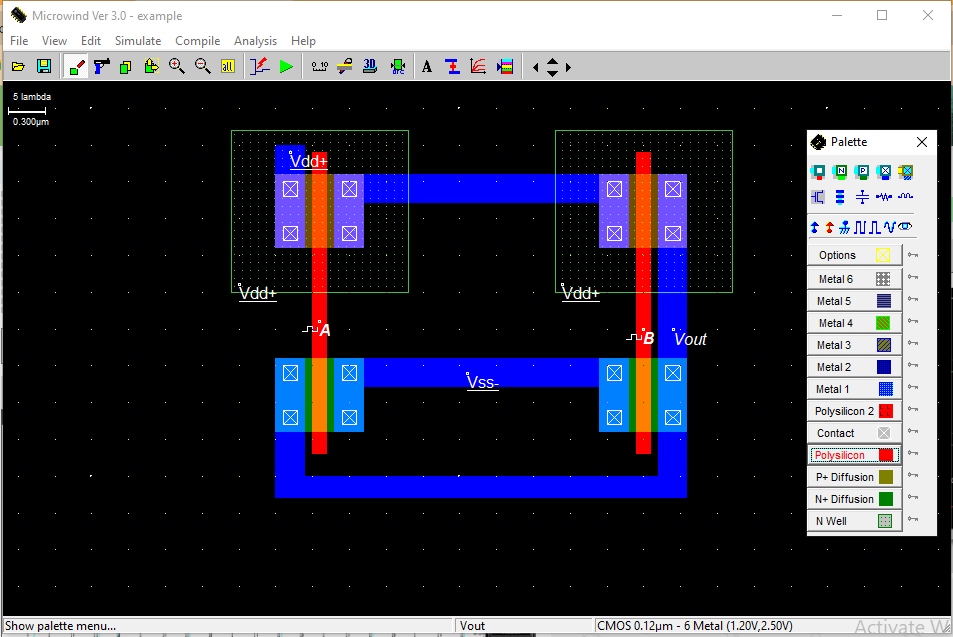


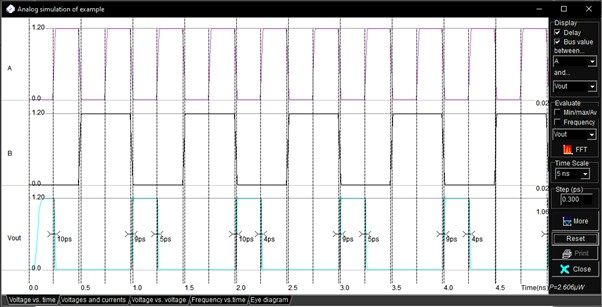


****

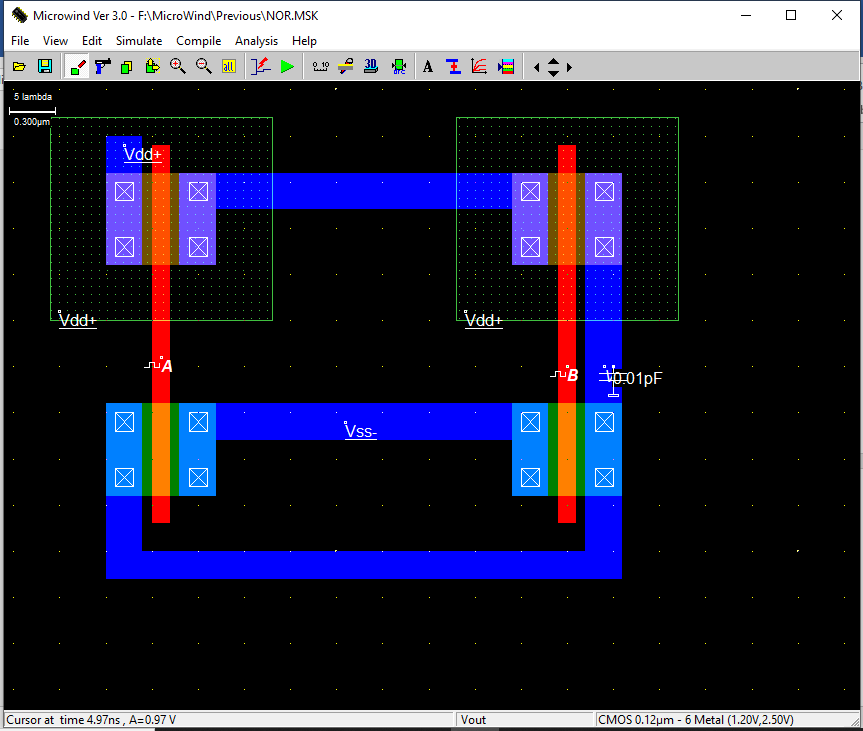


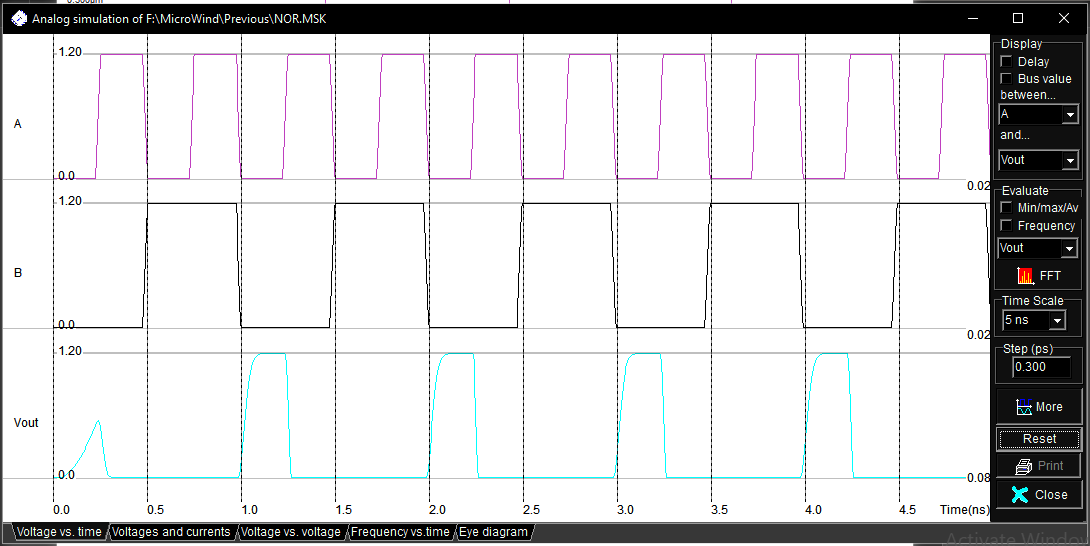
NOR



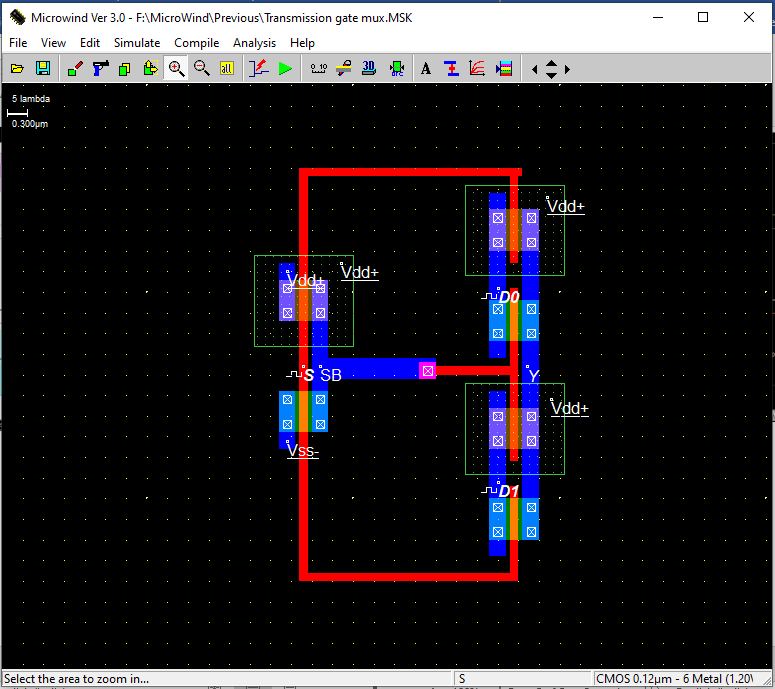


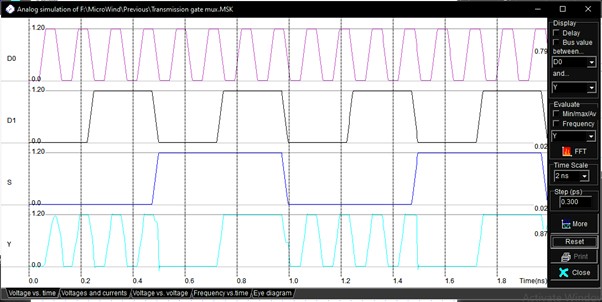
With Capacitor



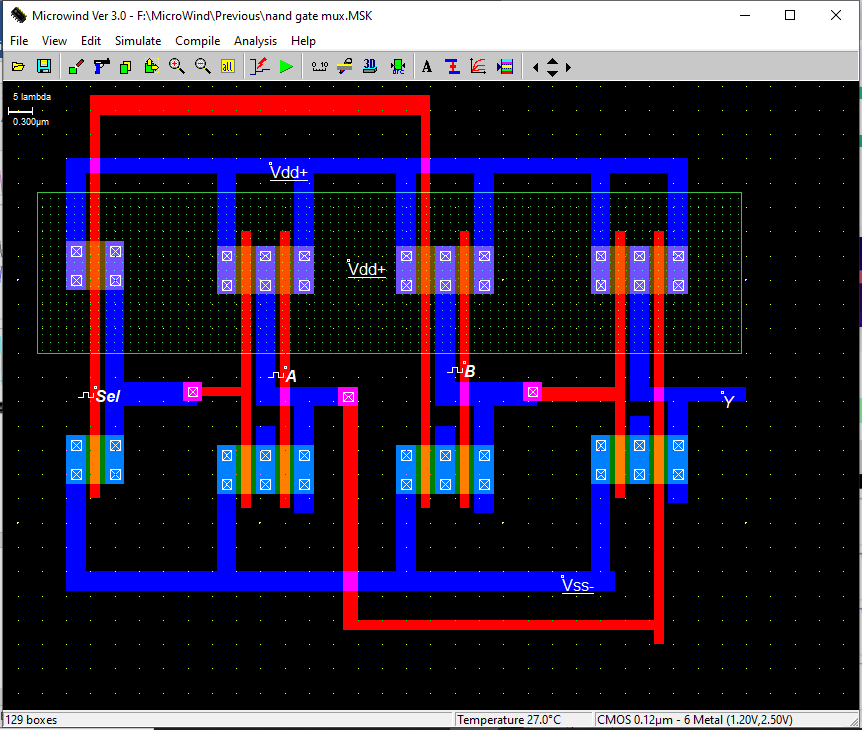


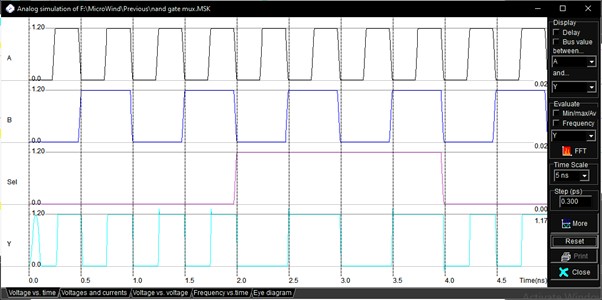
Mux 2:1 using Transmission gate



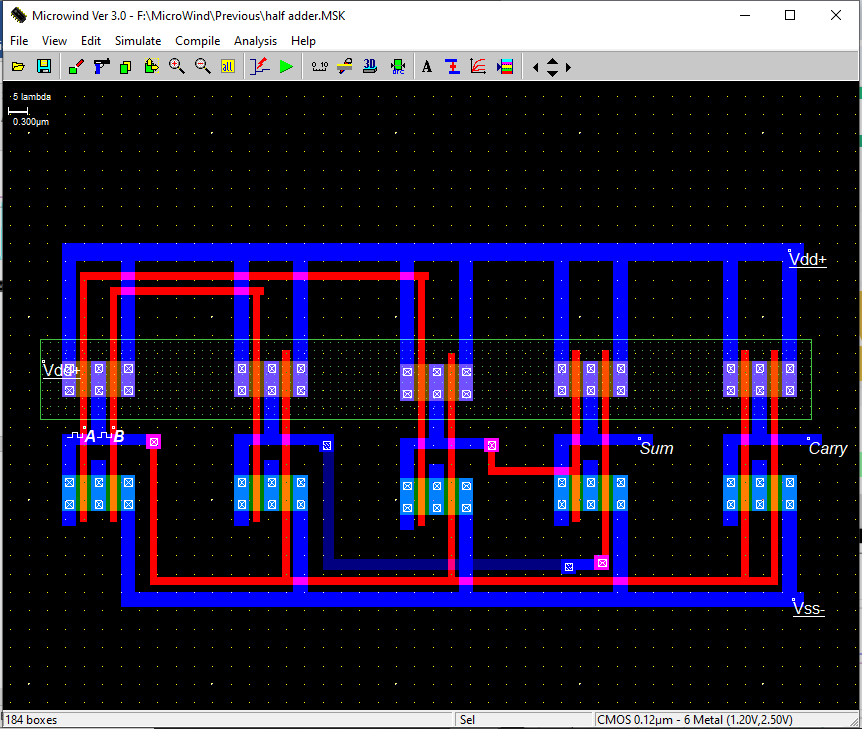


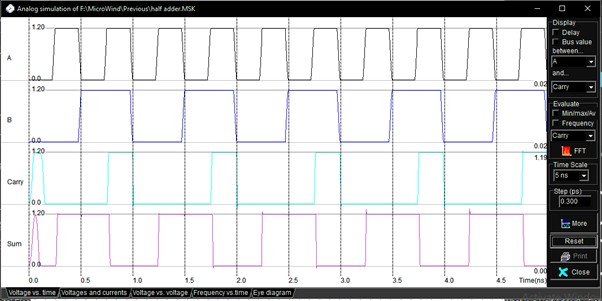
Mux 2:1 using Conventional gate (NAND gate)



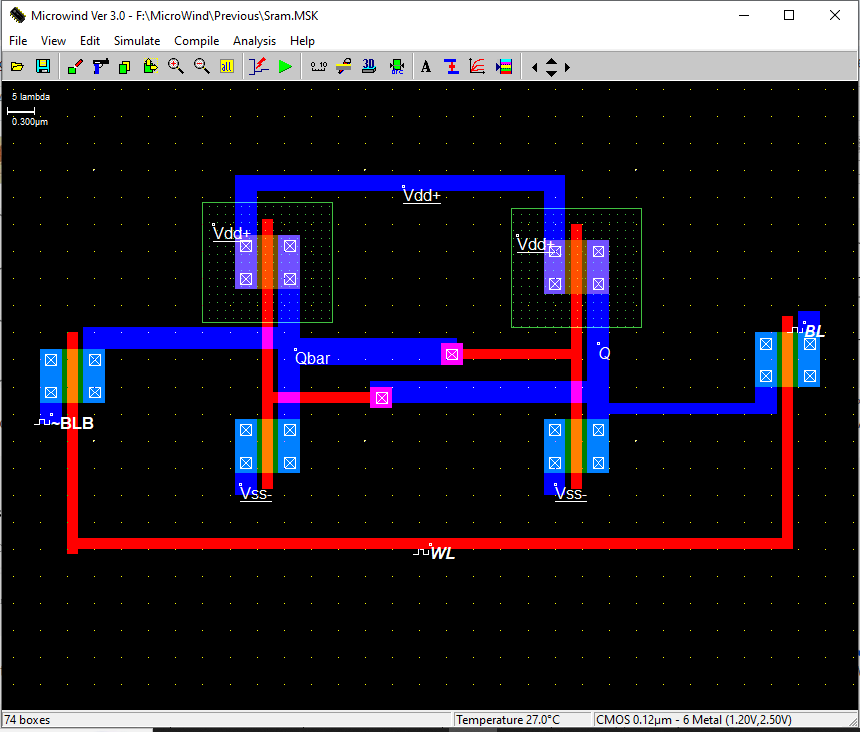


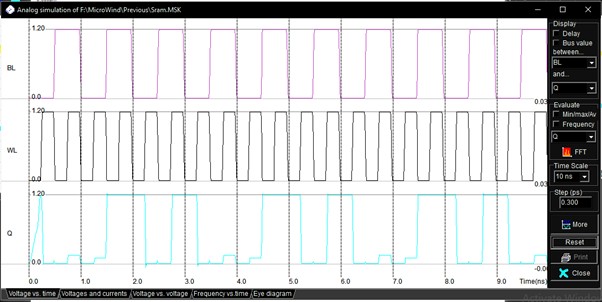
Half adder





SRAM





EXPERIMENT NO : 10

**Half adder**

library IEEE;

use IEEE.STD\_LOGIC 1164.ALL;

entity adder is

Port (a: in STD\_LOGIC;

b: in STD\_LOGIC;

s:out STD\_LOGIC;

c:out STD\_LOGIC);

end adder;

architecture Behavioral of adder is

begin

s<=a xor b;

c<=a and b;

end Behavioral;

**Test Bench**

library IEEE;

use IEEE.STD\_LOGIC 1164.ALL;

ENTITY hadder IS

END hadder;

ARCHITECTURE behavior OF hadder IS

.. Component Declaration for the Unit Under Test (UUT)

COMPONENT adder

PORT(

a: IN std\_logic;

b: IN std\_logic;

s: OUT std\_logic;

c: OUT std\_logic

);

END COMPONENT;

--Inputs

Signal a: std\_logic := '0';

signal b: std\_logic := '0';

--Outputs

Signal s: std\_logic;

signal c: std\_logic;

BEGIN

uut: adder PORT MAP (

a =>a,

b =>b,

S =>S,

c =>c

);

stim\_proc: process

begin

a <= ‘0';

b<=’’ 0';

wait for 100 ns;

a<=’ 0';

b<=’1';

wait for 100 ns;

a<=’1';

b<=’0';

wait for 100 ns;

a<=’1';

b<=^’1'

wait for 100 ns;

wait;

end process;

END;

**Full Adder**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity fulladder is

Port(x: in STD\_LOGIC;

y: in STD\_LOGIC;

z: in STD\_LOGIC;

s: out STD\_LOGIC;

c:out STD\_LOGIC);

end fulladder;

architecture Behavioral of fulladder is

begin

s<=(x xor y)xor z;

c=((x xor y) and z) or (x and z);

end Behavioral;

**Test Bench**

library IEEE;

use IEEE.STD\_LOGIC 1164.ALL;

ENTITY fadder IS

END fadder;

ARCHITECTURE behavior Of fadder IS

COMPONENT fulladder

PORT(

x: IN std\_logic;

y: IN std\_logic;

z: IN std\_logic;

s: OUT std\_logic;

c: OUT std\_logic

);

END COMPONENT;

-Inputs

signal x: std\_logic := '0’;

signal y: std logic := '0';

signal z: std\_logic := '0';

-Outputs

signals: std\_logic,

signal c: std\_logic;

BEGIN

uut: fulladder PORT MAP (

X => x,

y=> Y,

z =>z,

s =>s,

c=>c

);

stim\_proc: process

begin

x<= ‘ 0';

y<= ‘ 0' ;

z<= ‘ 0' ;

wait for 100 ns;

x<= ‘ 0';

y<= ‘ 1';

z<= ‘ 0';

wait for 100 ns;

x <= ‘ 0’;

y<=‘ 0' ;

z<= ‘ 1';

wait for 100 ns;

x<= ‘ 0';

y<= ‘ 1';

z<= ‘ 1';

wait for 100 ns;

x< = ‘ 1‘;

y<= ‘ 0' ;

z<= ‘ 0' ;

wait for 100 ns;

x<= ‘ 1'

y<= ‘ 0' ;

z<= ‘ 1';

wait for 100 ns;

x<=‘ 1';

y<= ‘ 1';

z<=‘ 0';

wait for 100 ns;

x<=‘ 1';

y<=‘ 1' ;

z<= ‘ 1';wait for 100 ns;

wait;

end process;

END;

**Experiment 10**

**VHDL Code of Half Adder:**

library IEEE;

entity HA is

Port ( a : in STD\_LOGIC;

b : in STD\_LOGIC;

sum : out STD\_LOGIC;

carry : out STD\_LOGIC);

end HA;

architecture Behavioral of HA is

begin

sum<= a xor b;

carry<= a and b;

end Behavioral;

**VHDL Code of Full Adder:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity fa is

Port ( a : in STD\_LOGIC;

b : in STD\_LOGIC;

cin : in STD\_LOGIC;

s : out STD\_LOGIC;

cot : out STD\_LOGIC);

end fa;

architecture Behavioral of fa is

component HA

Port ( a : in STD\_LOGIC;

b : in STD\_LOGIC;

sum : out STD\_LOGIC;

carry : out STD\_LOGIC);

end component;

signal s1, c1, c2: std\_logic

begin

HA1: HA port map( a,b, s1, c1);

HA2: HA port map(s1, cin, s, c2);

cot<= c1 or c2;

end Behavioral;

**Testbench:**

ENTITY bbbb IS

END bbbb;

ARCHITECTURE behavior OF bbbb IS

COMPONENT fa111

PORT(

a : IN std\_logic;

b : IN std\_logic;

cin : IN std\_logic;

s : OUT std\_logic;

cout : OUT std\_logic

);

END COMPONENT;

--Inputs

signal a : std\_logic := '0';

signal b : std\_logic := '0';

signal cin : std\_logic := '0';

--Outputs

signal s : std\_logic;

signal cout : std\_logic;

BEGIN

uut: fa111 PORT MAP (

a => a,

b => b,

cin => cin,

s => s,

cout => cout

);

stim\_proc: process

begin

a<='0';

b<='0';

cin<='0';

wait for 100 ns;

a<='0';

b<='0';

cin<='1';

wait for 100 ns;

a<='0';

b<='1';

cin<='0';

wait for 100 ns;

a<='0';

b<='1';

cin<='1';

wait for 100 ns;

a<='1';

b<='0';

cin<='0';

wait for 100 ns;

a<='1';

b<='0';

cin<='1';

wait for 100 ns;

a<='1';

b<='1';

cin<='0';

wait for 100 ns;

a<='1';

b<='1';

cin<='1';

wait for 100 ns;

a<='0';

b<='0';

cin<='0';

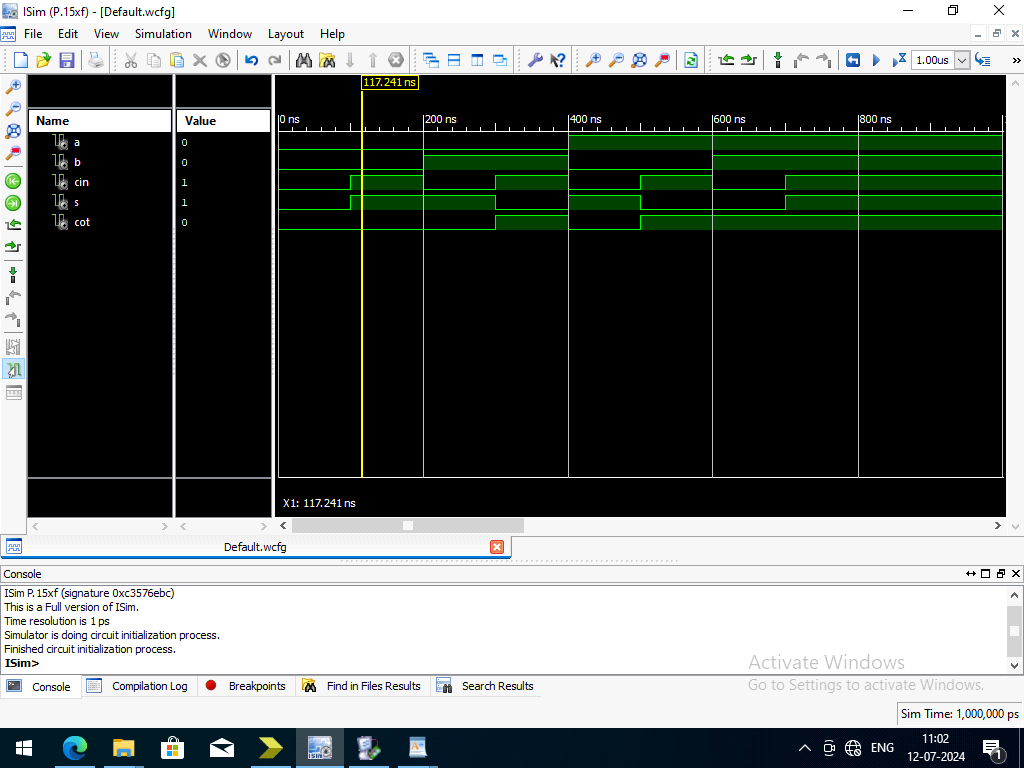
wait for 100 ns;

wait;

end process;

END;

**Simulation:**



**UCF:**

net a loc = p87;

net b loc= p86;

net cin loc= p85;

net s loc = p162;

net cot loc = p165;

