**PART 1: XILINX**

**TITLE: PROGRAM FOR 4-BIT ALU**

**PROGRAM:**

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 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;

entity vr1 is

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

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

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

c : out STD\_LOGIC\_VECTOR (4 downto 0));

end vr1;

architecture Behavioral of vr1 is

begin

Process(a,b,Op)

Begin

Case op is

When "000" =>

c <= ('0' & a) + ('0' & b);

When "001" => c <= ('0' & a) - ('0' & b);

When "010" => c <= ('0' & a) AND ('0' & b);

When "011" => c <= ('0' & a) OR ('0' & b);

When "100" => c <= ('0' & a) XOR ('0' & b);

When "101" => c <= ('0' & a) NAND ('0' & b);

When "110" => c <= ('0' & a) XNOR ('0' & b);

When others => c <= '0' & (not a);

End case;

End process;

end Behavioral;

**UCF FILE**

ALU 4 BIT #SW 0 to SW 3

net "a" LOC = "P75";

net "a" LOC = "P64";

net "a" LOC = "P58";

net "a" LOC = "P6";

#SW 8 to SW 11 net "b" LOC = "P116";

net "b" LOC = "P67";

net "b" LOC = "P61";

net "b" LOC = "P8";

#SW 15 to SW 12 net "op" LOC = "P126";

net "op" LOC = "P133";

net "op" LOC = "P139"

; ALU UCF FILE # LI 1 to LI 5 net "c" LOC = "P2";

net "c" LOC = "P142";

net "c" LOC = "P138";

net "c" LOC = "P134";

net "c" LOC = "P124";

**TITLE: PROGRAM FOR MOD N COUNTER**

**PROGRAM:**

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 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;

entity counter is

Port ( clk,reset : in STD\_LOGIC;

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

end counter;

architecture Behavioral of counter is

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

begin

process (clk,reset)

begin

if(reset='1')then

count\_t<="0000";

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

count\_t<=count\_t+"0001";

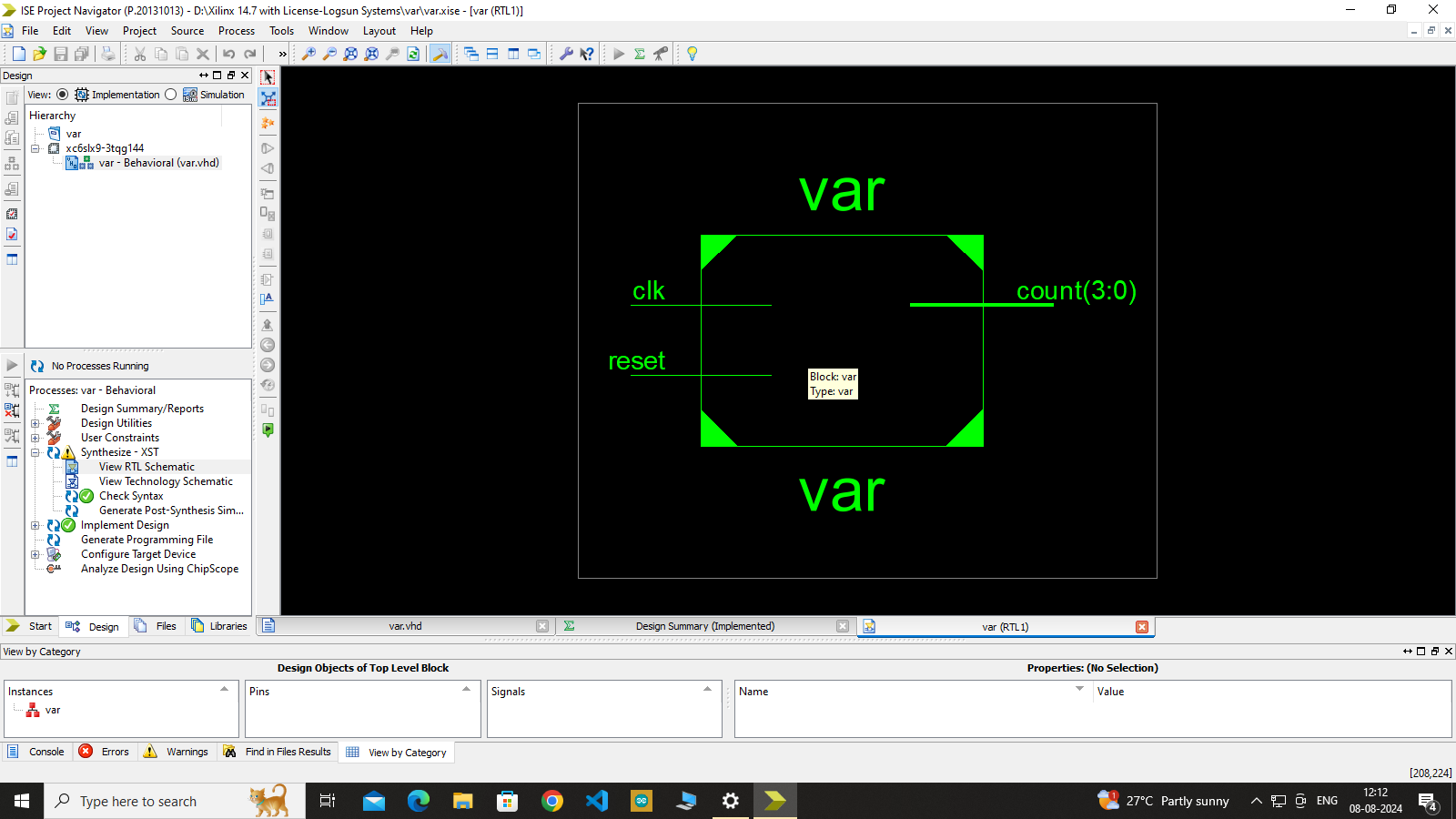
end if;

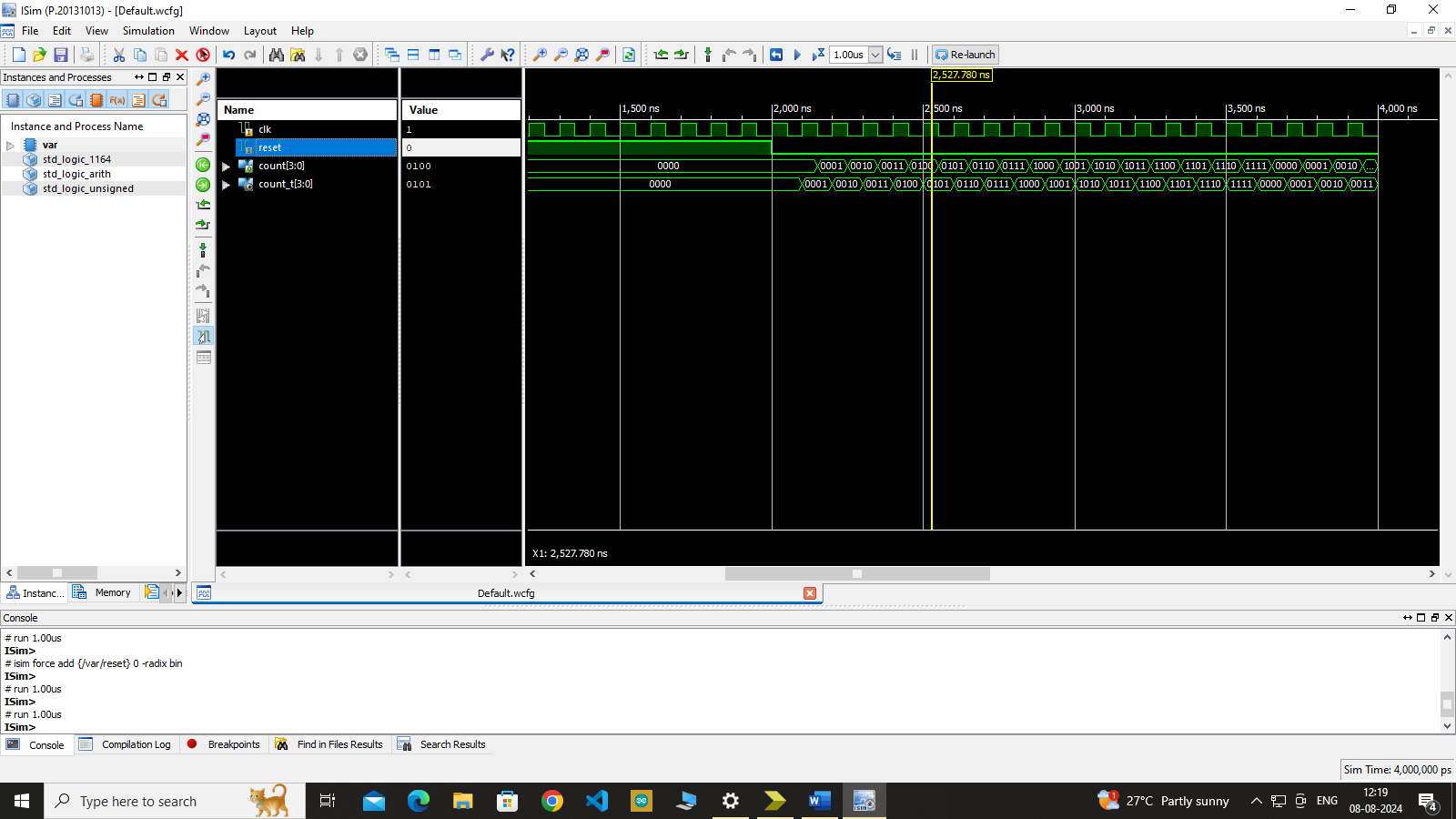
count<=count\_t;

end process ;

end Behavioral;

**OUTPUT:**





**TITLE: PROGRAM FOR SHIFT REGISTER**

**PROGRAM:**

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;

-- Uncomment the following library declaration if instantiating

-- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity shift is

Port ( clk, clr, lin, rin : in STD\_LOGIC;

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

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

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

end shift;

architecture Behavioral of shift is

begin

process(clk, clr, lin, rin, s, d)

begin

if(clr='0') then q<="0000"

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

if(s="00") then q<=q;

elsif(s="01") then q<=rin&q(3 downto 1);

elsif(s="10") then q<=q(2 downto 0) &iin;

elsif(s="11" then q<=d;

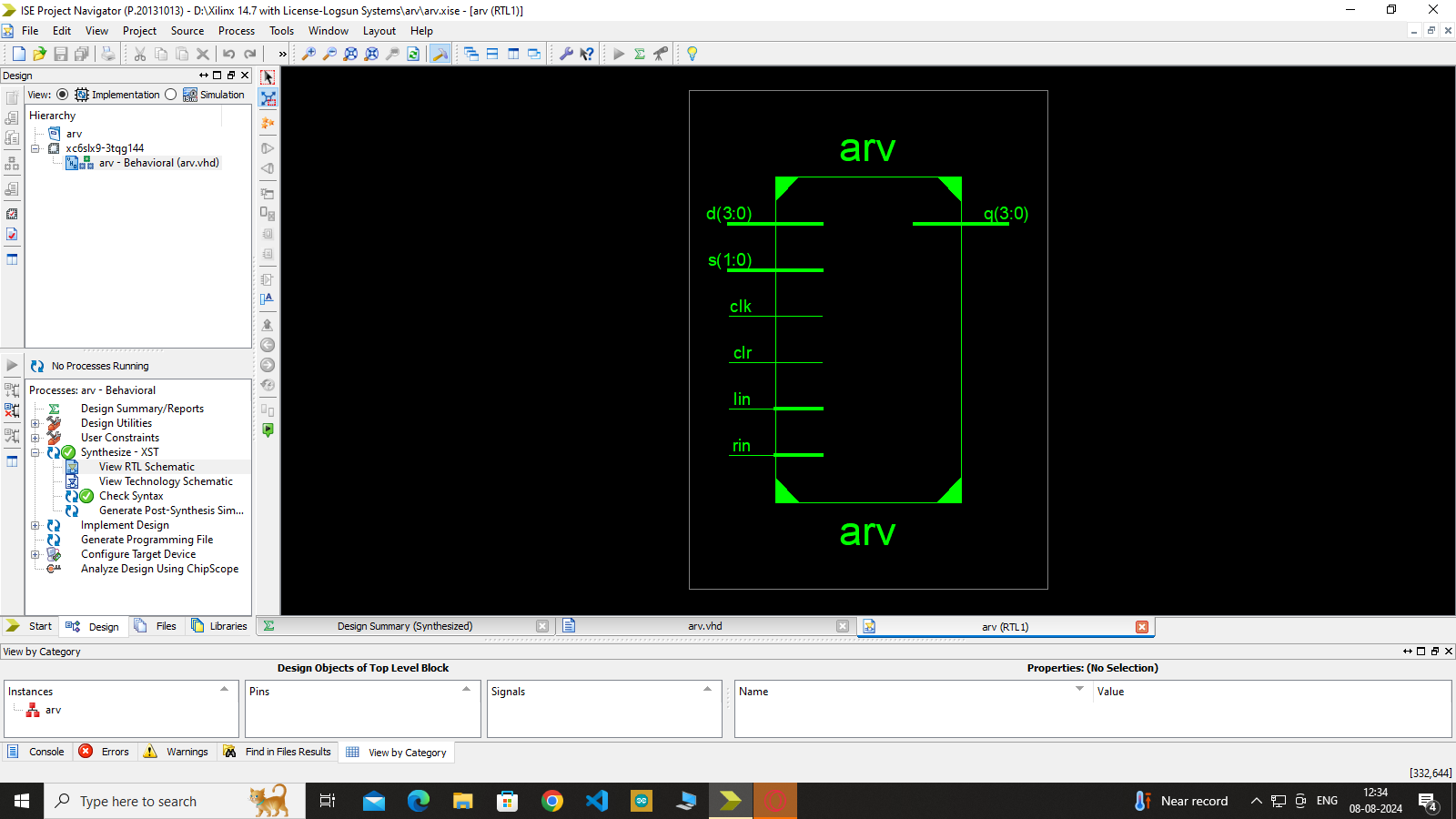
end if;

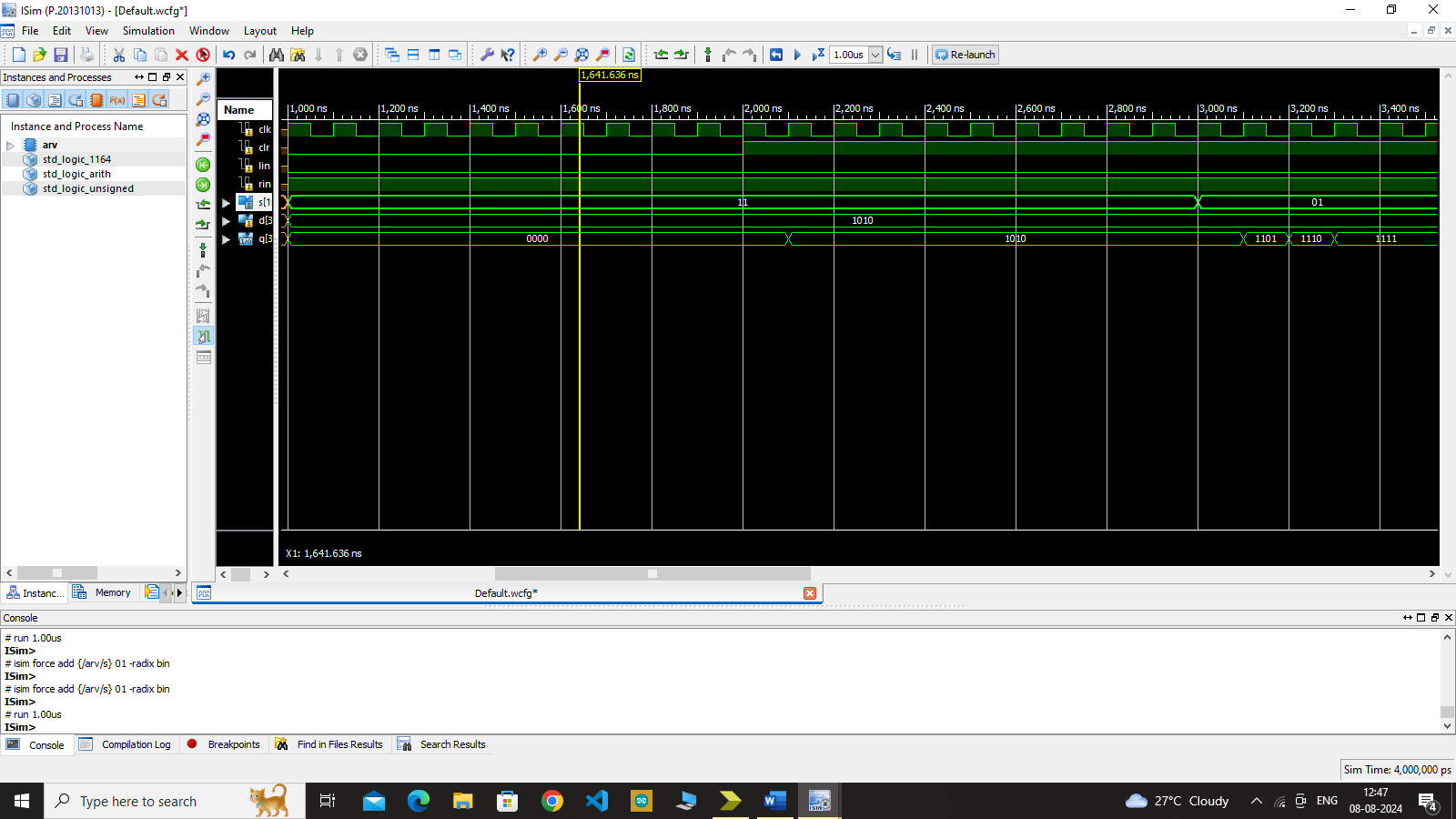
end if;

end process;

end Behavioral;

**OUTPUT:**





# Program Name: Kepad interfacing with seven segment display

## Program:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL; use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity keypad is

Port ( clk : in STD\_LOGIC; rst : in STD\_LOGIC;

row : inout STD\_LOGIC\_VECTOR (3 downto 0); col : inout STD\_LOGIC\_VECTOR (3 downto 0);

digit : out STD\_LOGIC\_VECTOR (3 downto 0); display: out STD\_LOGIC\_VECTOR (7 downto 0));

end keypad;

architecture Behavioral of keypad is

TYPE STATE\_TYPE IS ( Col1Set, Col2Set, Col3Set, Col4Set ); SIGNAL coltest : STATE\_TYPE;

SIGNAL data:STD\_LOGIC\_VECTOR (7 downto 0);

SIGNAL rowm:STD\_LOGIC\_VECTOR (3 downto 0); SIGNAL clock:STD\_LOGIC;

begin

Process for clock divide by 10

Process(clk,rst)

variable temp:integer range 1 to 10; begin

if(rst='0') then temp:=1;

elsif(rising\_edge(clk)) then temp:=temp+1; if(temp=10) then

clock<= not clock; temp:=1;

end if; end if;

end process;

Process for Keypad scan & Display

process(clock,rst) begin

if (rst='0') then coltest<=col1set; rowm<="1111"; data<=x"00";

elsif rising\_edge (clock) then digit<="1111";

case coltest is

when col1set=> rowm<="0111";

case col is

when "0111"=>data<= x"C6"; C

when "1011"=>data<= x"A1"; D

when "1101"=>data<= x"86"; E

when "1110"=>data<= x"8E"; F

when others=>coltest<=col2set; end case;

when col2set=> rowm<="1011";

case col is

when "0111"=>data<= x"C0"; 0

when "1011"=>data<= x"F9"; 1

when "1101"=>data<= x"A4"; 2

when "1110"=>data<= x"B0"; 3

when others=>coltest<=col3set; end case;

when col3set=> rowm<="1101";

case col is

when "0111"=>data<= x"99"; 4

when "1011"=>data<= x"92"; 5

when "1101"=>data<= x"82"; 6

when "1110"=>data<= x"F8"; 7

when others=>coltest<=col4set; end case;

when col4set=> rowm <="1110";

case col is

when "0111"=>data<= x"80"; 8

when "1011"=>data<= x"90"; 9

when "1101"=>data<= x"88"; A

when "1110"=>data<= x"83"; B

when others=>coltest<=col1set; end case;

end case; end if;

end process; display<=data; row<=rowm; end Behavioral;

## UCF file:

NET "clk" LOC = "P56"; NET "rst" LOC = "P22";

NET "clk" CLOCK\_DEDICATED\_ROUTE = true; NET "rst" CLOCK\_DEDICATED\_ROUTE = FALSE; NET "display[0]" LOC = "P48";

NET "display[1]" LOC = "P45"; NET "display[2]" LOC = "P51"; NET "display[3]" LOC = "P47"; NET "display[4]" LOC = "P57"; NET "display[5]" LOC = "P50"; NET "display[6]" LOC = "P59"; NET "display[7]" LOC = "P55"; NET "digit[0]" LOC = "P46";

NET "digit[1]" LOC = "P43";

NET "digit[2]" LOC = "P44";

NET "digit[3]" LOC = "P40";

NET "col[0]" LOC = "P58" | PULLUP | DRIVE = 2; NET "col[1]" LOC = "P62" | PULLUP | DRIVE = 2; NET "col[2]" LOC = "P61" | PULLUP | DRIVE = 2; NET "col[3]" LOC = "P66" | PULLUP | DRIVE = 2; NET "row[0]" LOC = "P64";

NET "row[1]" LOC = "P16";

NET "row[2]" LOC = "P117"; NET "row[3]" LOC = "P75

# Program Name : Keypad interfacing with LCD

## VHDL code

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity key\_Lcd is

Port ( clk : in STD\_LOGIC; rst : in STD\_LOGIC;

RS : out STD\_LOGIC:='0'; EN : out STD\_LOGIC:='0'; RW : out STD\_LOGIC;

LCD : out STD\_LOGIC\_VECTOR (7 downto 0); ROW : inout STD\_LOGIC\_VECTOR (3 downto 0); COL : inout STD\_LOGIC\_VECTOR (3 downto 0));

end key\_Lcd;

architecture Behavioral of key\_Lcd is SIGNAL rowtest : integer range 0 to 3:=0; constant N: integer:=15;

SIGNAL data:STD\_LOGIC\_VECTOR (7 downto 0);

SIGNAL Ldata:STD\_LOGIC\_VECTOR (7 downto 0);

SIGNAL rowm:STD\_LOGIC\_VECTOR (3 downto 0); signal step: integer range 0 to 6:=0;

signal i: integer range 0 to 16:=0; signal R\_S,E\_N,R\_W : std\_logic:='0'; SIGNAL clock:STD\_LOGIC;

type arr is array (1 to N) of std\_logic\_vector(7 downto 0);

constant datas : arr := (x"38",x"0E",x"06",x"01",x"80",--init command for 8 bit mode

x"4B",x"45",x"59",x"20",x"50",x"52",x"45",x"53",x"53",x"3A"); --KEY PRESS:

begin process(rst,clk)

variable temp: integer range 0 to 999999; begin

if(rst='0')then temp:=0; clock<='0';

elsif(rising\_edge(clk))then temp:=temp+1;

if(temp =25000)then clock<= not clock; temp:=0;

end if; end if;

end process; Process(clock,rst,step,data) begin

if(rst='0') then

R\_W<='0';

E\_N<='0';

R\_S<='0';

Ldata <= x"00"; i<=0;

elsif(rising\_edge(clock)) then

case step is

when 0=> R\_W <= '0';

if(i<5) then --init command upto 4 in array >> RS=0; R\_S<='0';

else R\_S <= '1'; --for data string RS=1; end if;

Ldata <= datas(i); --send array element one by one E\_N <= '1'; --enable pulse

step<=1;

when 1=> E\_N <= '0';

i<=i+1; --increament array pointer

step<=2;

when 2=> if(i=16) then --if i=16 indicate end of array element then go to step 3 step<=3; --else go to step 1

else step<=0; end if;

when 3 => Ldata <= x"8B"; --set lcd cursor location at 0x8b to print key preseed R\_S <= '0';

E\_N <= '1';

step<=4;

when 4 => E\_N <= '0'; step<=5;

when 5 => Ldata <= data; --print key R\_S <= '1'; --rs=1 data mode

E\_N <= '1'; --enable pulse

step<=6;

when 6 => E\_N <= '0'; --repeat from step 3 to print new data like while(1) loop step<=3;

end case; end if;

end process;

Process for Keypad scan & Display

process(clock,rst) begin

if (rst='0') then rowtest<=0; rowm<="1111"; data<=x"FF";

elsif rising\_edge (clock) then case rowtest is

when 0=> rowm<="0111"; case COL is

when "0111"=>data<= x"43"; C

when "1011"=>data<= x"44"; D

when "1101"=>data<= x"45"; E

when "1110"=>data<= x"46"; F

when others=>rowtest<=1;

end case; when 1=> rowm<="1011";

case COL is

when "0111"=>data<= x"30"; 0

when "1011"=>data<= x"31"; 1

when "1101"=>data<= x"32"; 2

when "1110"=>data<= x"33"; 3

when others=>rowtest<=2; end case;

when 2=> rowm<="1101";

case COL is

when "0111"=>data<= x"34"; 4

when "1011"=>data<= x"35"; 5

when "1101"=>data<= x"36"; 6

when "1110"=>data<= x"37"; 7

when others=>rowtest<=3; end case;

when 3=> rowm <="1110";

case COL is

when "0111"=>data<= x"38"; 8

when "1011"=>data<= x"39"; 9

when "1101"=>data<= x"41"; A

when "1110"=>data<= x"42"; B

when others=>rowtest<=0; end case;

end case; end if;

end process;

ROW<=rowm;

LCD<=Ldata;

EN<=E\_N;

RW<=R\_W;

RS<=R\_S;

end Behavioral;

## UCF file:

# clock pin for Basys2 Board

NET "clk" LOC = "P56"; # Bank = 0, Signal name = MCLK NET "rst" LOC = "P22"; # Bank = 2, Signal name = Reset NET "rst" CLOCK\_DEDICATED\_ROUTE = TRUE;

# Pin Connected to LCD data NET "LCD[0]" LOC = "P85";

NET "LCD[1]" LOC = "P82";

NET "LCD[2]" LOC = "P83";

NET "LCD[3]" LOC = "P80";

NET "LCD[4]" LOC = "P81";

NET "LCD[5]" LOC = "P78";

NET "LCD[6]" LOC = "P79";

NET "LCD[7]" LOC = "P74";

#Control line

NET "RS" LOC = "P88"; NET "EN" LOC = "P84"; NET "RW" LOC = "P140";

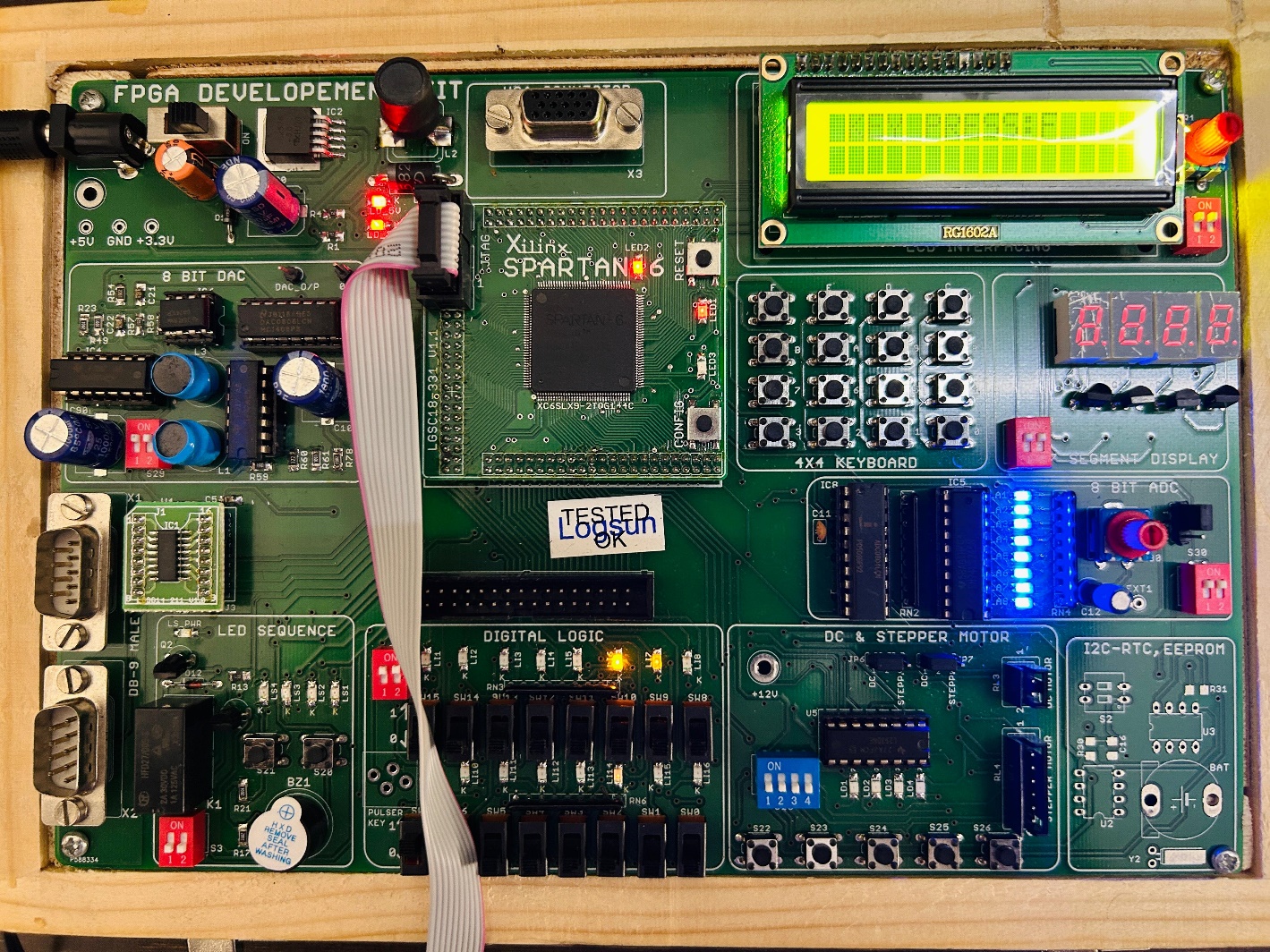
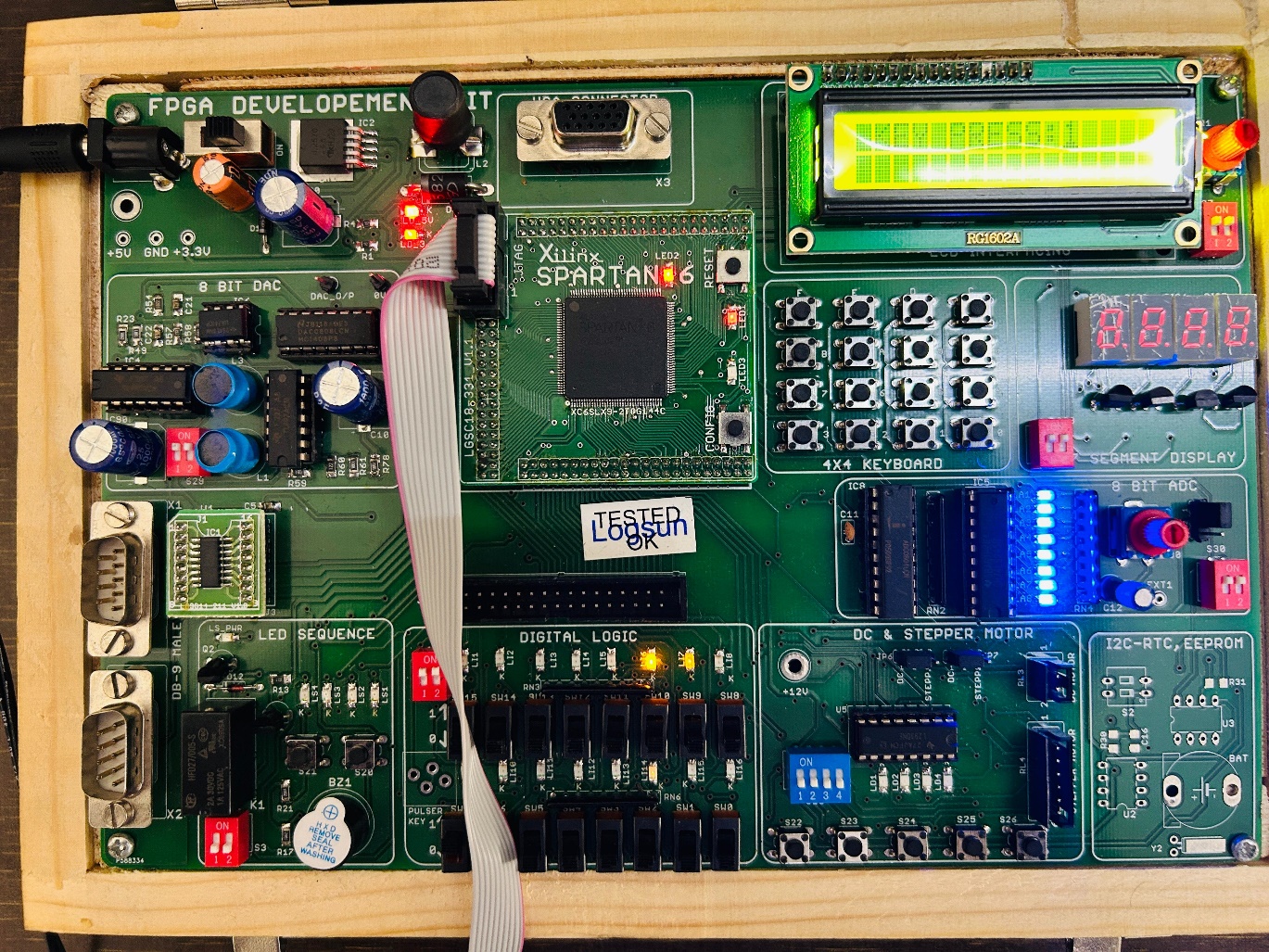
NET "COL[0]" LOC = "P58" | PULLUP | DRIVE = 2; NET "COL[1]" LOC = "P62" | PULLUP | DRIVE = 2; NET "COL[2]" LOC = "P61" | PULLUP | DRIVE = 2; NET "COL[3]" LOC = "P66" | PULLUP | DRIVE = 2; NET "ROW[0]" LOC = "P64";

NET "ROW[1]" LOC = "P16";

NET "ROW[2]" LOC = "P117";

NET "ROW[3]" LOC = "P75";

**OUTPUT:**



# Program Name: LCD interfacing

## VHDL code:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL; use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity LCD\_interface is

Port ( rst : in std\_logic;

clk : in std\_logic; RS : out std\_logic; RW : out std\_logic; EN : out std\_logic;

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

end LCD\_interface;

architecture arch\_LCD\_interface of LCD\_interface is signal count: integer range 0 to 47;

signal clock: std\_logic; signal flag : std\_logic:='0'; begin

process(rst,clk)

variable temp: integer range 0 to 999999; begin

--Clock divider, to slow the clock if(not rst='1')then

temp:=0;

clock<='0';

elsif(clk='1' and clk' event)then temp:=temp+1;

if(temp =250000)then

clock<= not clock; temp:=0;

end if; end if;

end process;

## -- counter to go through the stages of LCD writing

process (rst,clock) begin

if (not rst ='1') then COUNT <=0;

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

if (count < 47 and flag='0' ) then count <= count+ 1;

end if; end if;

END PROCESS;

## -- process to allocate ouput on counter vaules

**-- writes on LCD in HEX format**

Process(count,rst,clock) begin

if (NOT rst ='1') then

RW<='0';

RS<='0';

EN<='0';

LCD <= "00000000";

flag<='0'; elsif(rising\_edge(clock)) then

## -- first few counts are to initialize LCD

case count is

when 0 => RW <= '0';

RS <= '0';

LCD <= x"38"; -- init in 2 lin mode EN <= '1';

when 1 => EN <= '0';

when 2 => RW <= '0';

RS <= '0';

LCD <= x"0E";-- display on, cursor blink EN <= '1';

when 3 => EN <= '0';

when 4 => RW <= '0';

RS <= '0';

LCD <= x"01"; -- clear display EN <= '1';

when 5 => EN <= '0';

when 6 => RW <= '0';

RS <= '0';

LCD <= x"06";-- shift cursor to right EN <= '1';

when 7 => EN <= '0';

when 8=> RW <= '0';

RS <= '0';

LCD <= x"81"; -- initialize to 1st line, 2nd position EN <= '1';

when 9=> EN <= '0';

## -- Data wriing starts

when 10 => RW <= '0';

RS <= '1';

LCD <=x"4A"; --J EN <= '1';

when 11 => EN <= '0';

when 12 => RW <= '0';

RS<= '1';

LCD<= x"53"; --S EN <= '1';

when 13 => EN <= '0';

when 14 => RW <= '0';

RS<= '1';

LCD<= x"50"; --P EN <= '1';

when 15 => EN <= '0';

when 16 => RW <= '0';

RS<= '1';

LCD<= x"4D"; --M EN <= '1';

when 17 => EN <= '0';

when 18 => RW <= '0';

RS<= '1';

LCD<= x"20"; --space EN <= '1';

when 19 => EN <= '0';

when 20 => RW <= '0';

RS<= '1';

LCD<= x"47"; --G EN <= '1';

when 21 => EN <= '0';

when 22 => RW <= '0';

RS<= '1';

LCD<= x"52"; --R EN <= '1';

when 23 => EN <= '0';

when 24 => RW <= '0';

RS<= '1';

LCD<= x"4F"; --O EN <= '1';

when 25 => EN <= '0';

when 26 => RW <= '0';

RS<= '1';

LCD<= x"55"; --U EN <= '1';

when 27 => EN <= '0';

when 28 => RW <= '0';

RS<= '1';

LCD<= x"50"; --P EN <= '1';

when 29=> EN <= '0';

when 30 => RW <= '0';

RS<= '1';

LCD<= x"20"; --space EN <= '1';

when 31=> EN <= '0';

when 32 => RW <= '0';

RS<= '1';

LCD<= x"20"; --space EN <= '1';

when 33=> EN <= '0';

when 34 => RW <= '0';

RS<= '1';

LCD<= x"20"; --space EN <= '1';

when 35=> EN <= '0';

when 36 => RW <= '0';

RS<= '1';

LCD<= x"20"; --space EN <= '1';

when 37 => EN <= '0';

when 38 => RW <= '0';

RS<= '0';

LCD<= x"C6"; --2nd line(0xC5) EN <= '1';

when 39 => EN <= '0';

when 40 => RW <= '0';

RS<= '1';

LCD<= x"50"; --P EN <= '1';

when 41 => EN <= '0';

when 42 => RW <= '0';

RS<= '1';

LCD<= x"55"; --U EN <= '1';

when 43 => EN <= '0';

when 44 => RW <= '0';

RS<= '1';

LCD<= x"4E"; --N EN <= '1';

when 45 => EN <= '0';

when 46 => RW <= '0';

RS<= '1';

LCD<= x"45"; --E EN <= '1';

when 47 => EN <= '0';

flag<='1';

END CASE;

end if;

END PROCESS;

end arch\_LCD\_interface;

## UCF file:

**# clock pin for FPGA Board**

NET "clk" LOC = "P56"; # Bank = 0, Signal name = MCLK NET "rst" LOC = "P22"; # Bank = 2, Signal name = Reset NET "rst" CLOCK\_DEDICATED\_ROUTE = FALSE;

# Pin Connected to LCD data NET "LCD[0]" LOC = "P85";

NET "LCD[1]" LOC = "P82";

NET "LCD[2]" LOC = "P83";

NET "LCD[3]" LOC = "P80";

NET "LCD[4]" LOC = "P81";

NET "LCD[5]" LOC = "P78";

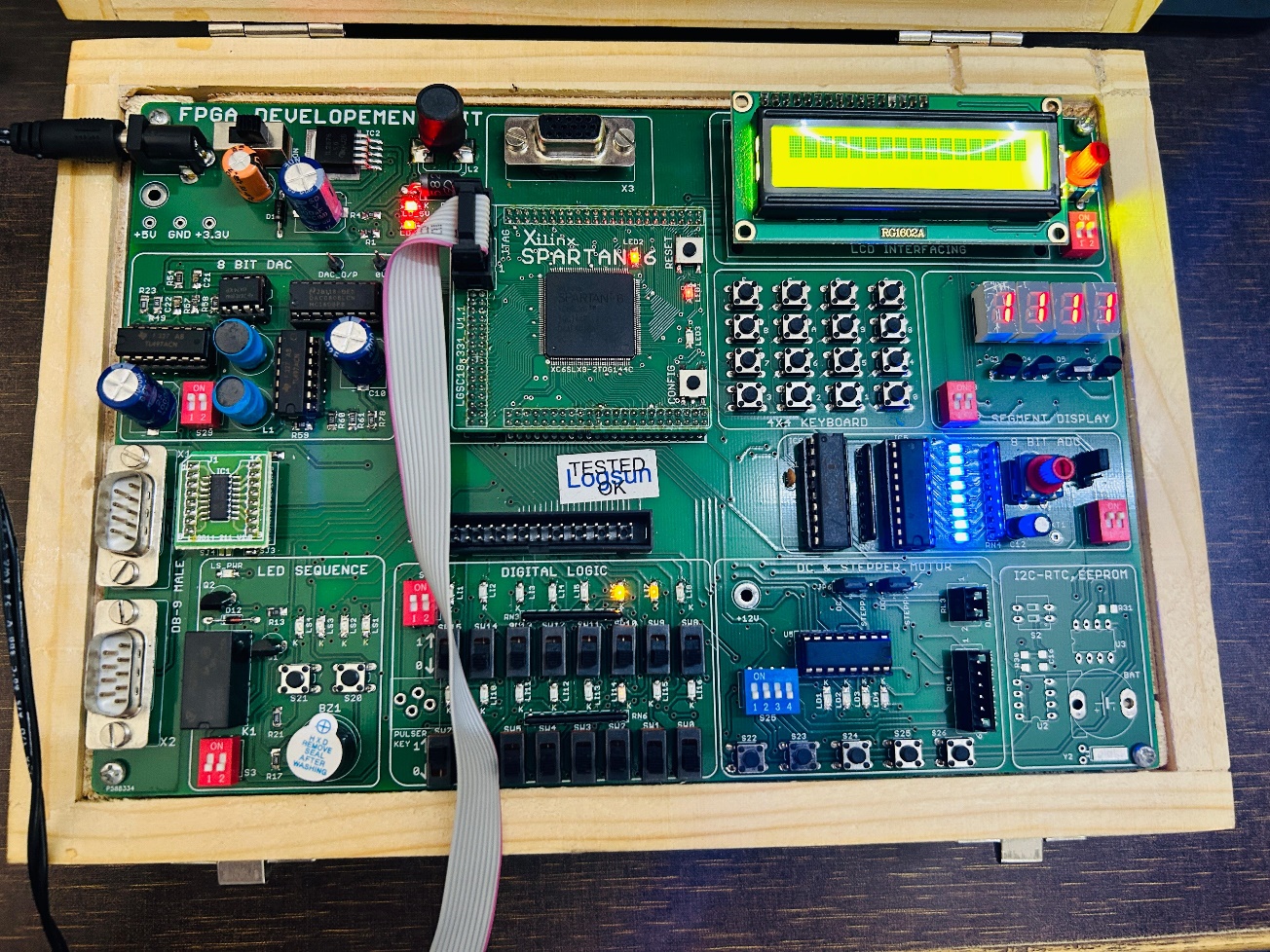
NET "LCD[6]" LOC = "P79";

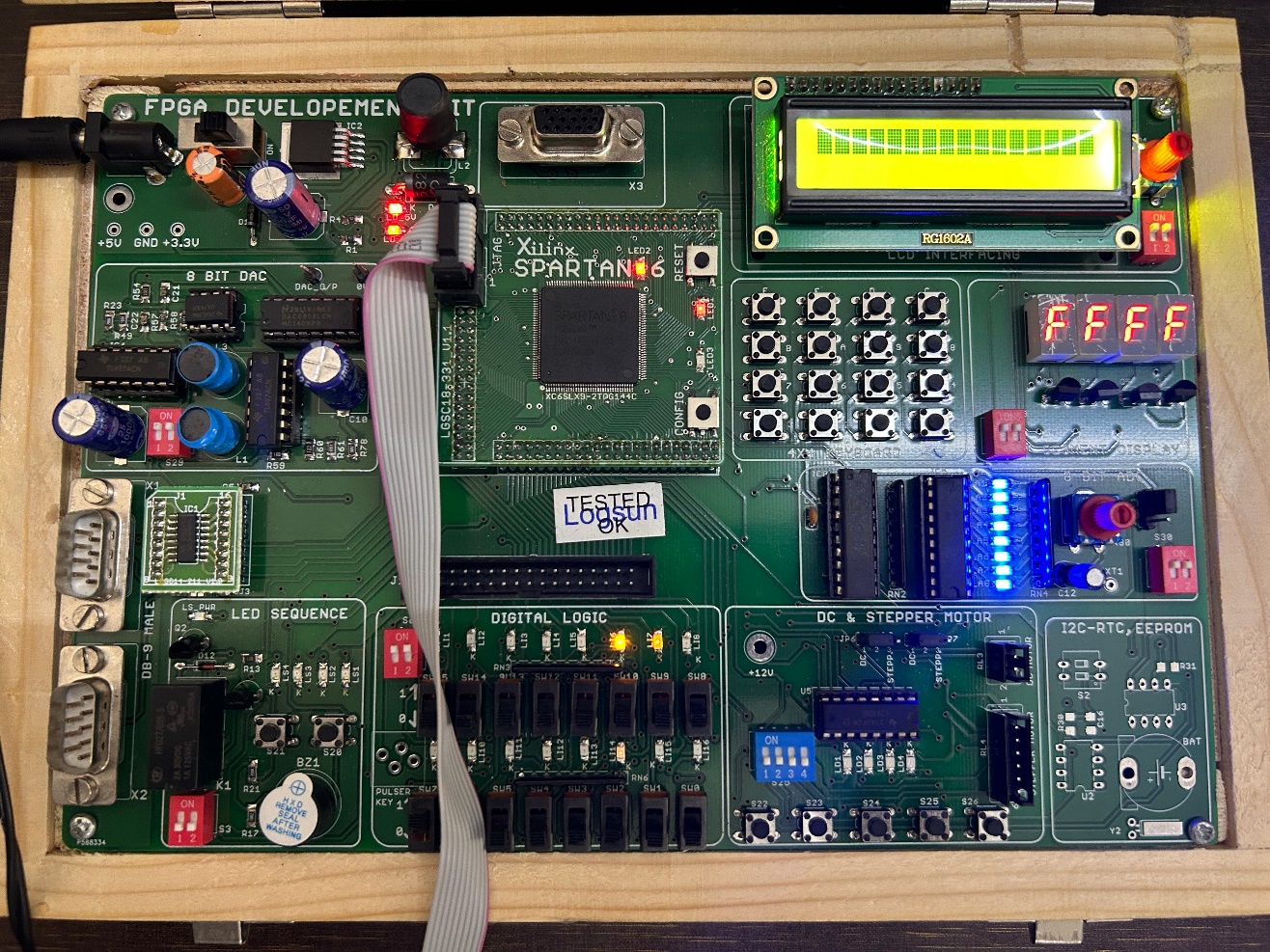
NET "LCD[7]" LOC = "P74";

## #Control line

NET "RS" LOC = "P88"; NET "EN" LOC = "P84"; NET "RW" LOC = "P2";

**OUTPUT:**

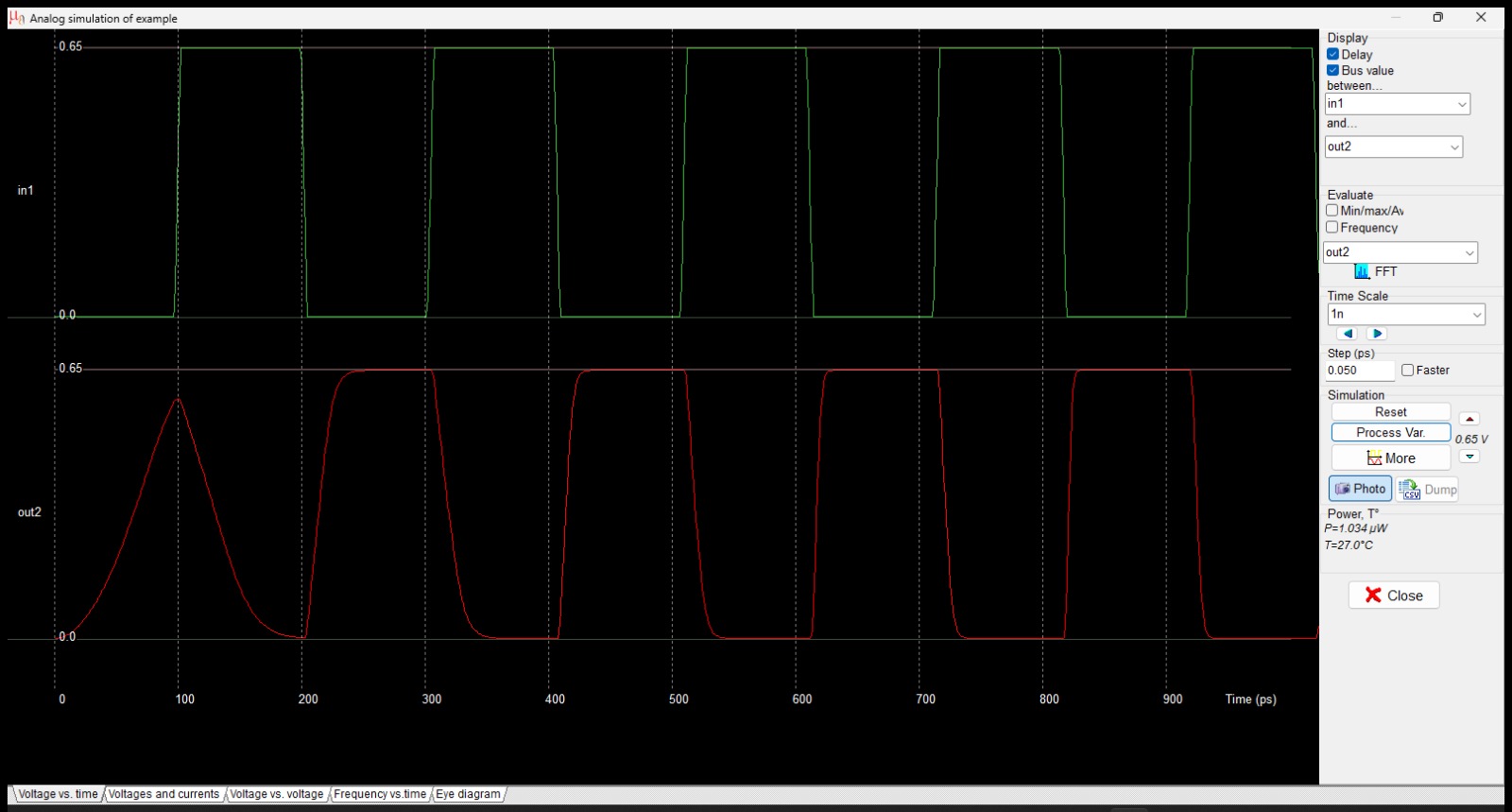
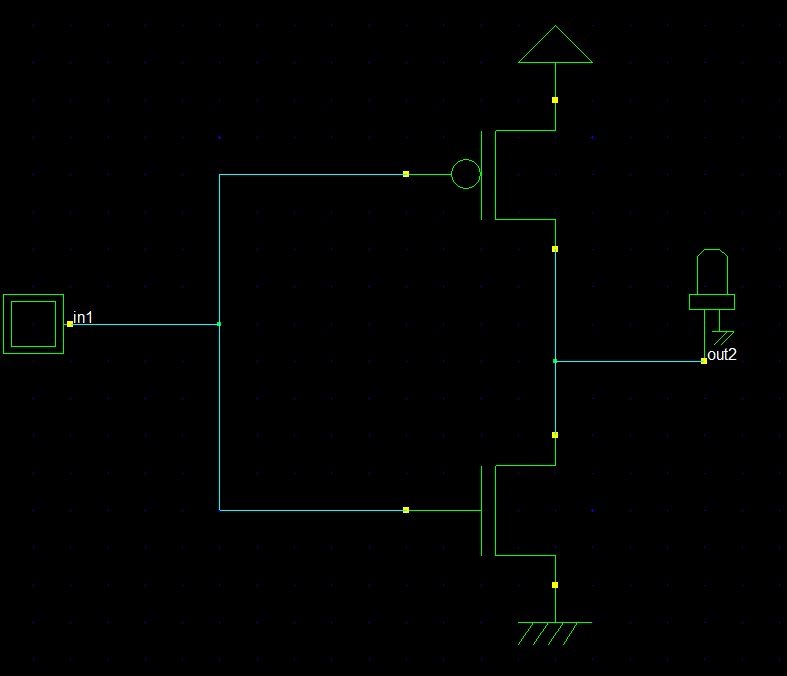




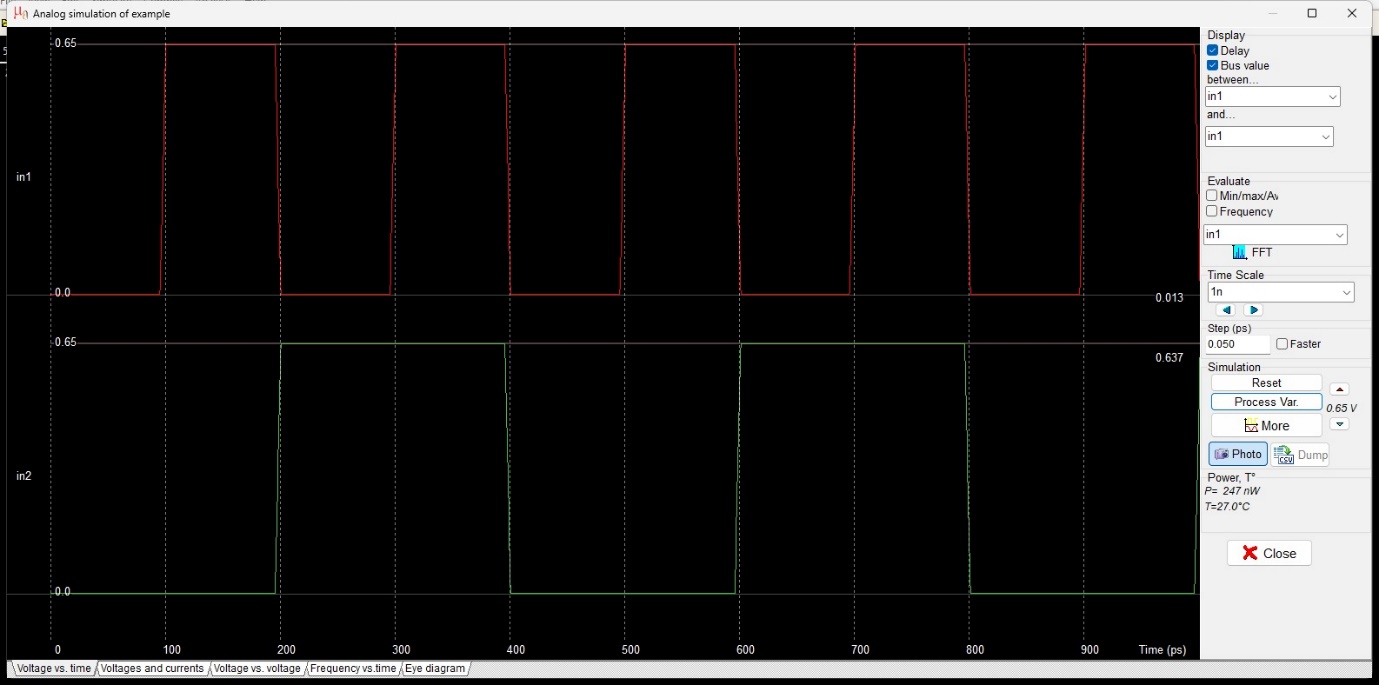
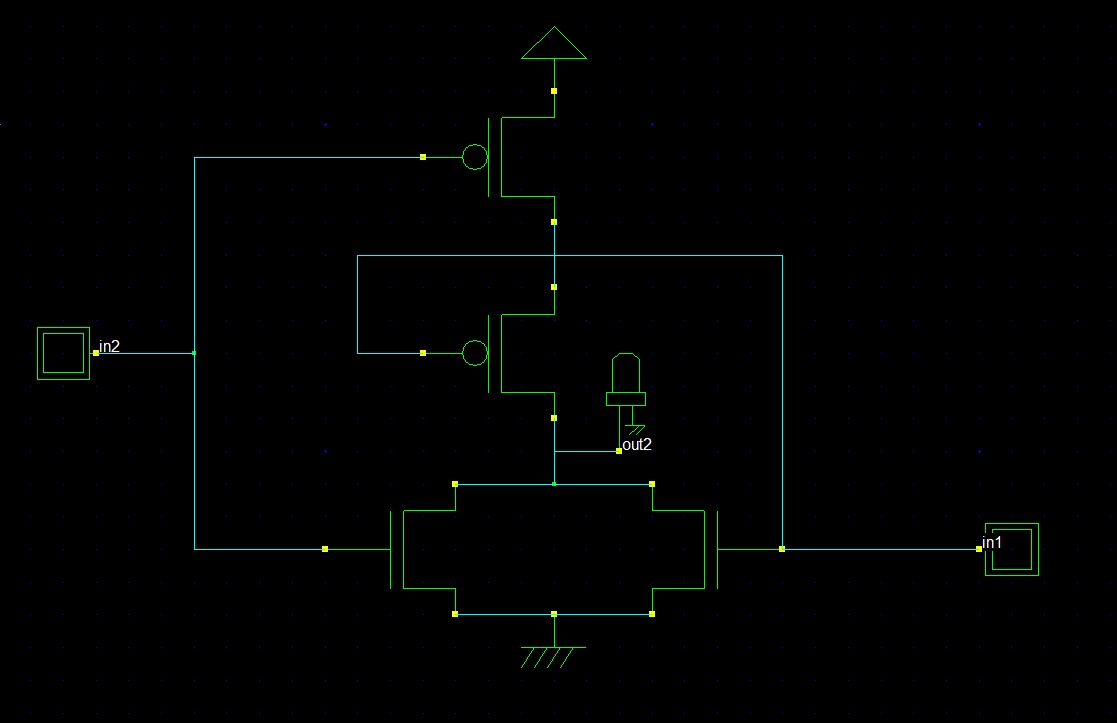
**PART 2: MICROWIND OUTPUTS**

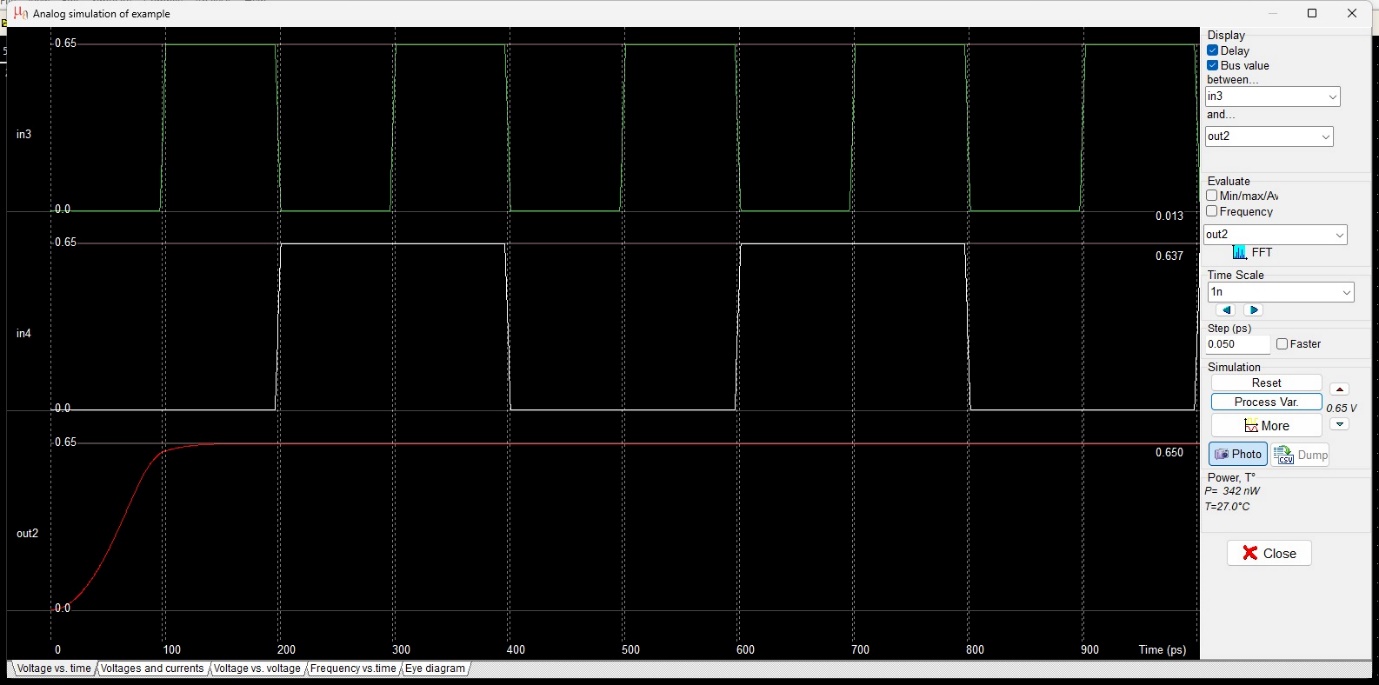
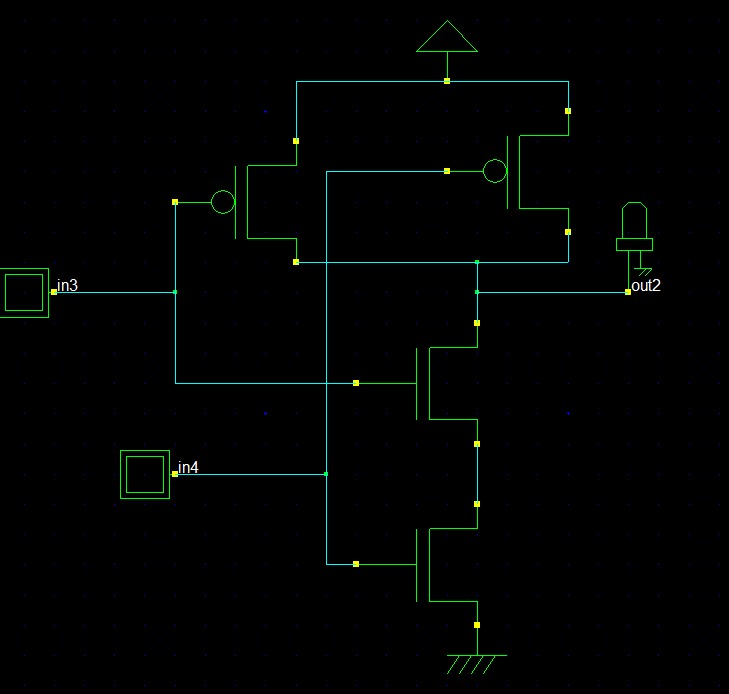
# Program Name: CMOS INVERTER, NAND AND NOR GATES

# INVERTER:

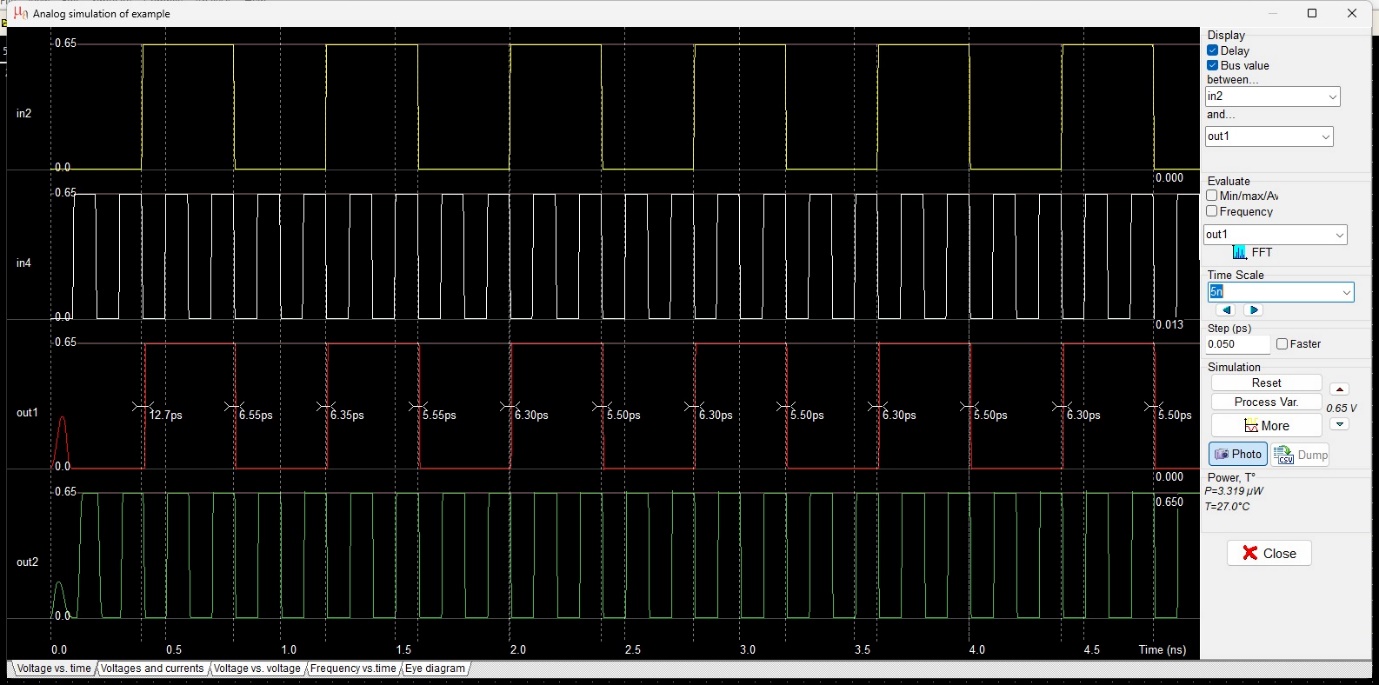
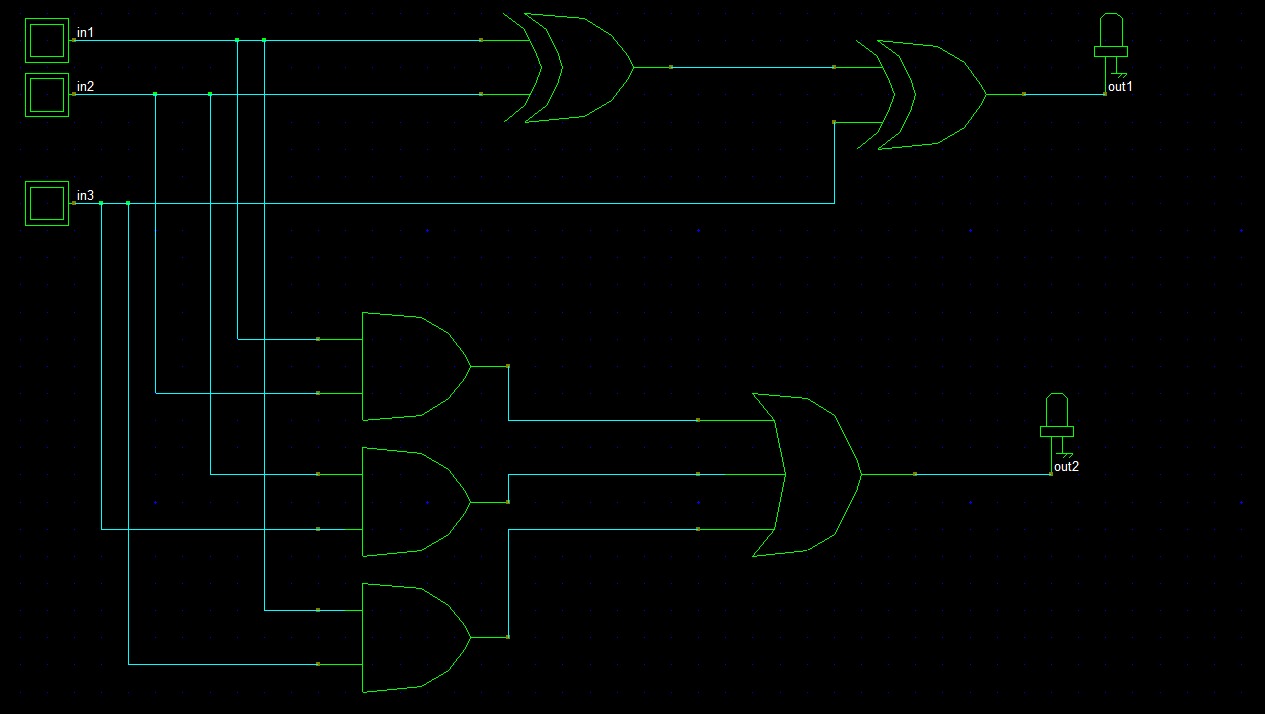
****

**NOR:**

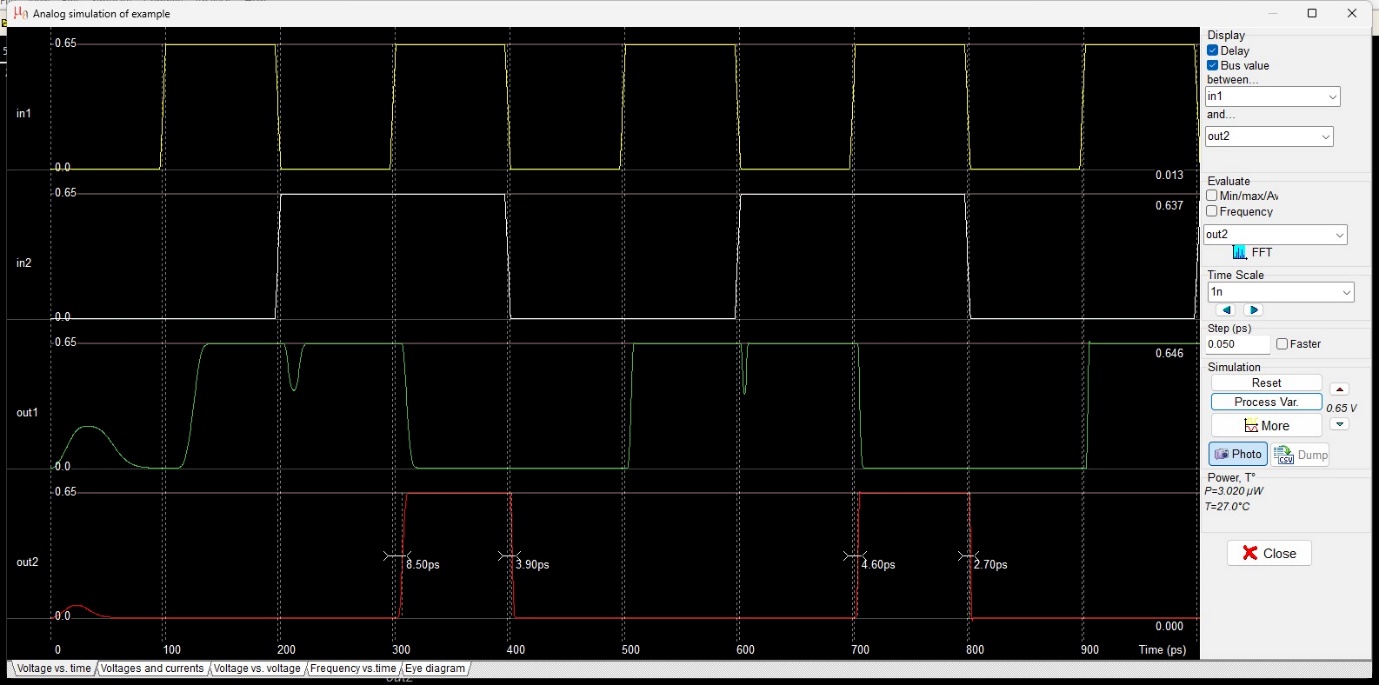
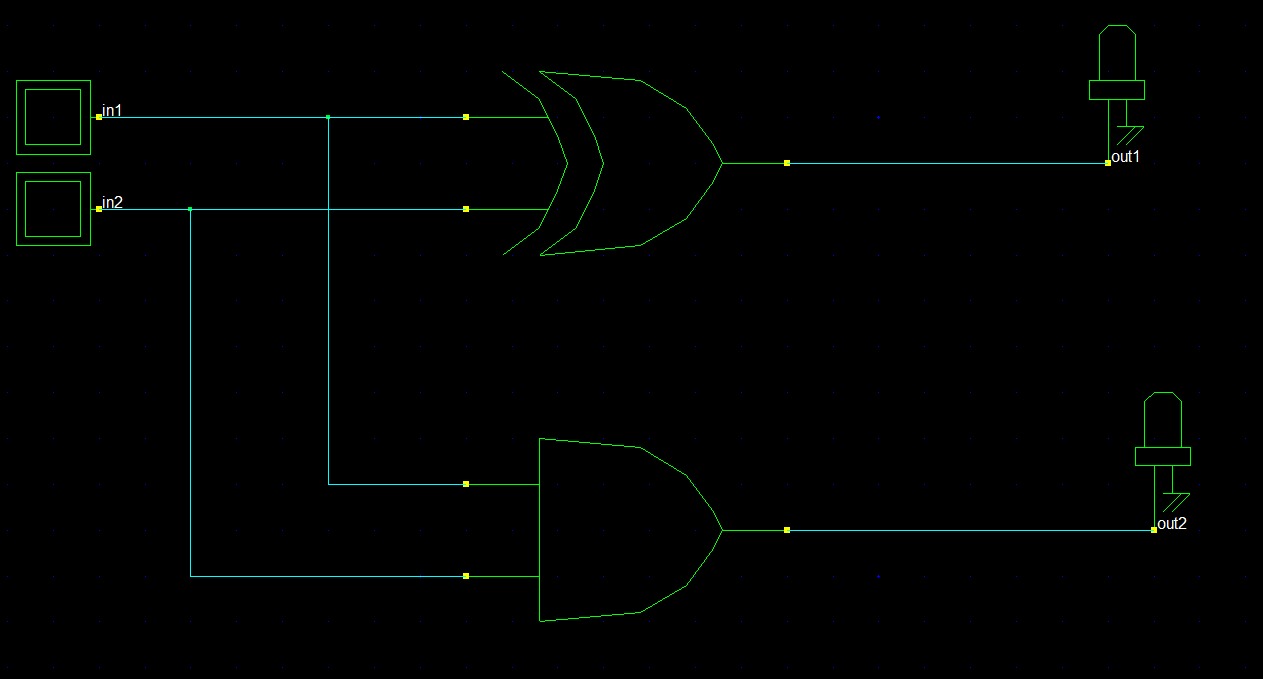


**NAND:**

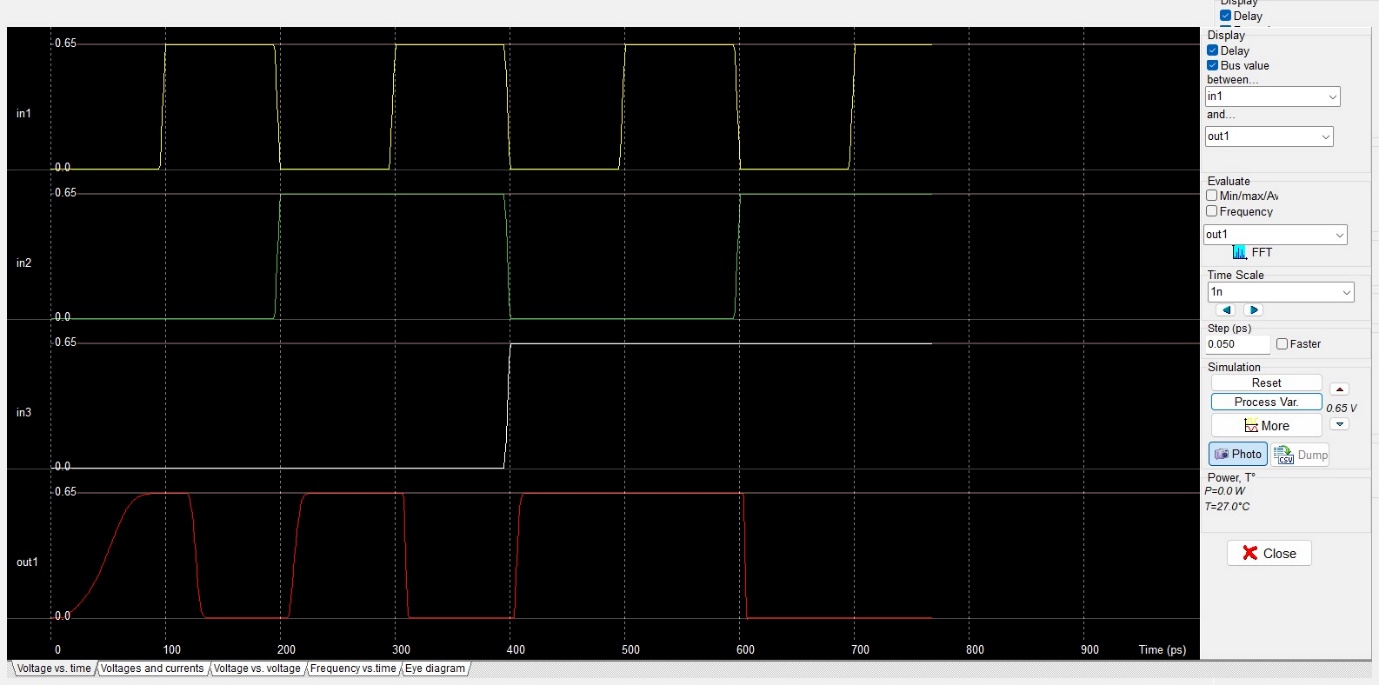
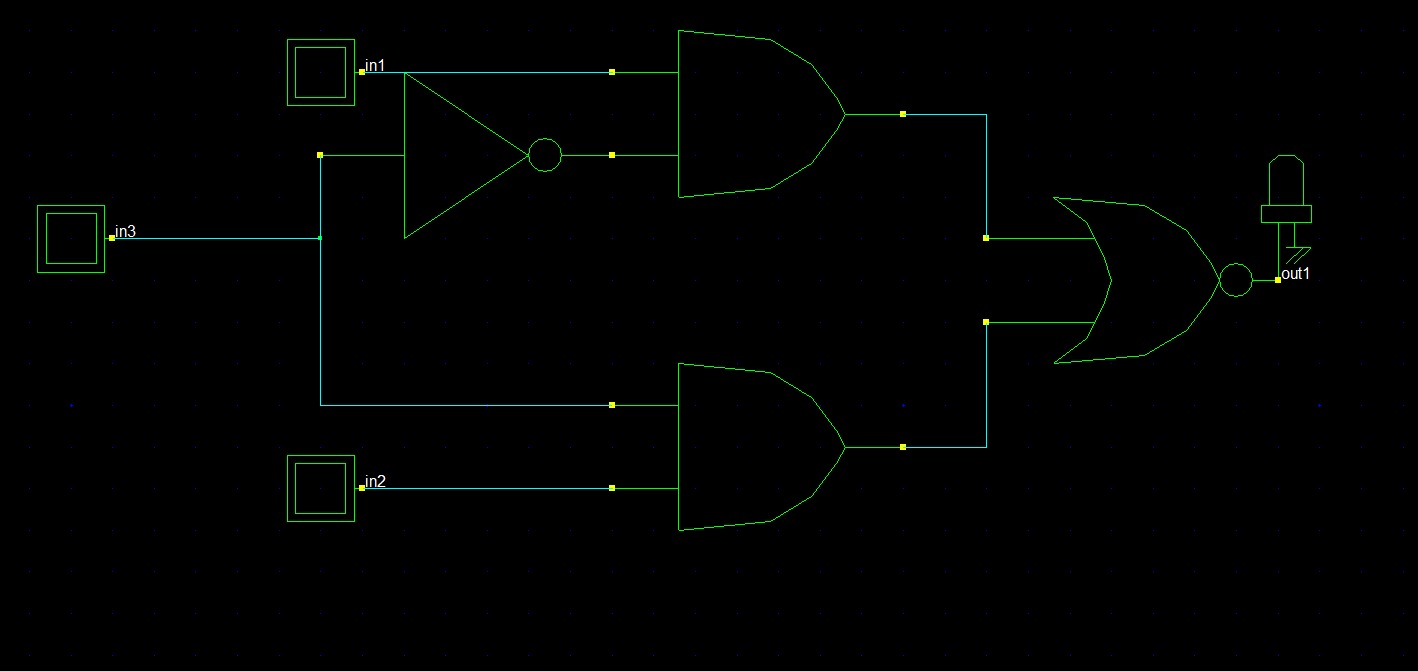
**FULL ADDER:**

****

**HALF ADDER:**

****

**PROGRAM NAME: 2:1 MUX**

****