ALU PROGRAM

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.all;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity ALU_4bit is
    Port ( a,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 ALU_4bit;
Architecture behavioral of ALU_4bit 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;
```

ALU UCF FILE

```
ALU 4 BIT
#SW 0 to SW 3
net "a<3>" LOC = "P75";
net "a<2>" LOC = "P64";
net "a<1>" LOC = "P58";
net "a<0>" LOC = "P6";

#SW 8 to SW 11
net "b<3>" LOC = "P116";
net "b<2>" LOC = "P67";
net "b<1>" LOC = "P61";
net "b<0>" LOC = "P8";

#SW 15 to SW 12
net "op<2>" LOC = "P126";
net "op<1>" LOC = "P133";
net "op<0>" LOC = "P139";
```

```
# LI 1 to LI 5
net "c<4>" LOC = "P2";
net "c<3>" LOC = "P142";
net "c<2>" LOC = "P138";
net "c<1>" LOC = "P134";
net "c<0>" LOC = "P124";
```

SHIFT REGISTER

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity shift_register is
  Port ( CLR                  : in  STD_LOGIC; -- clear input
         SEL                  : in  STD_LOGIC_VECTOR (1 downto 0);
         SHF_LOAD             : in  STD_LOGIC;
         Serial_In  : in  STD_LOGIC; -- serial input
         CLK                  : in  STD_LOGIC;
         D                                : in  STD_LOGIC_VECTOR (3 downto 0); -- Parallel D i/ps
         Q                    : inout  STD_LOGIC_VECTOR (3 downto 0));
end shift_register;
architecture Behavioral of shift_register is
begin
process(CLK,clr)
begin
if (CLR='0')then
  Q<="0000";
  elsif(clk'event and clk='1') then
   if (SHF_LOAD='0')then -- Parallel Load
     Q <= D;
   else
   if(SEL="00")then         ----Shift Right Operation
         -- alternate code
         -- Q <= Serial_In & Q(3 downto 1);
   Q(3)<=Serial_In;
   Q(2)<=Q(3);
   Q(1)<=Q(2);
   Q(0)<=Q(1);
   end if;
   if(SEL="01")then         ----Shift left Operation
         -- alternate code
         -- Q <= Q(2 downto 0) & Serial_In ;
   Q(0)<=Serial_In;
   Q(1)<=Q(0);
   Q(2)<=Q(1);
   Q(3)<=Q(2);
   end if;
   if(SEL="10") then    ----Rotate left
   Q <= D(2 downto 0) & D(3);
   end if;
   if(SEL="11") then        ----Rotate Right
   Q <= D(0) & D(3 downto 1);
   end if;
   end if;
   end if;
   end process;

end Behavioral;
```

SHIFT REGISTER UCF File

```
NET "CLK" LOC = P56;

NET "CLR" LOC = P8; #SW 8
NET "SEL" LOC = P61; #SW 9
NET "SHF_LOAD" LOC = P67; #SW 10
NET "Serial_In" LOC = P116; #SW 11

NET "D[3]" LOC = P6; #SW 0
NET "D[2]" LOC = P58;#SW 1
NET "D[1]" LOC = P64;#SW 2
NET "D[0]" LOC = P75;#SW 3

NET "Q[3]" LOC = P142; # LI 4
NET "Q[2]" LOC = P138; # LI 3
NET "Q[1]" LOC = P134; # LI 2
NET "Q[0]" LOC = P124; # LI 1
```

KEYPAD 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 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;
```

Keypad UCF File

```
Keypad# 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";
```

LCD PROGRAM

```
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;
```

LCD PROGRAM 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";
```

MOD COUNTER PROGRAM

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity nmod is
  port( Clock_enable_B: in std_logic;
            Clock: in std_logic;
            Reset: in std_logic;
            Output: out std_logic_vector(0 to 3));
end nmod;

architecture Behavioral of nmod is
  signal temp: std_logic_vector(0 to 3);
begin   process(Clock,Reset)
  begin
    if Reset='1' then
      temp <= "0000";
    elsif(rising_edge(Clock)) then
      if Clock_enable_B='0' then
        if temp="1001" then
          temp<="0000";
        else
          temp <= temp + 1;
        end if;
      end if;
    end if;
  end process;
  Output <= temp;
end Behavioral;
```

MOD COUNTER UCF File

```
PlanAhead Generated physical constraints

NET "Output[0]" LOC = P7;
NET "Output[1]" LOC = P9;
NET "Output[2]" LOC = P10;
NET "Output[3]" LOC = P11;
NET "Clock" LOC = P79;
NET "Reset" LOC = P43;
```