

```
1.ALU MAIN CODE
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.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 alcl is
    Port ( a : in  STD_LOGIC_VECTOR (3 downto 0);
          b : in  STD_LOGIC_VECTOR (3 downto 0);
          c : in  STD_LOGIC_VECTOR (2 downto 0);
          y : out STD_LOGIC_VECTOR (3 downto 0));
end alcl;

architecture Behavioral of alcl is

begin
process(a,b,c)
begin
    case c is
        when "000"=>Y<=a+b;
        when "001"=>Y<=a-b;
        when "010"=>Y<=a and b;
        when "011"=>Y<=a nand b;
        when "100"=>Y<=a xor b;
        when "101"=>Y<=a xnor b;
        when "110"=>Y<=a or b;
        when "111"=>Y<=a;
        when others=>Y<="0000";
    end case;
end process;

end Behavioral;
```

```
ALU TB
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 tbc IS
END tbc;

ARCHITECTURE behavior OF tbc IS

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

    COMPONENT alcl
    PORT(
        a : IN  std_logic_vector(3 downto 0);
        b : IN  std_logic_vector(3 downto 0);
        c : 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 c : 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

    constant <clock>_period : time := 10 ns;

BEGIN

    -- Instantiate the Unit Under Test (UUT)
    uut: alcl PORT MAP (
        a => a,
        b => b,
        c => c,
        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<="1011";
    b<="1001";
    c<="000"; --result for addition
    wait for 100 ns;
    c<="001"; --result for subtraction
    wait for 100 ns;
    c<="010"; --result of and
    wait for 100 ns;
    c<="011"; --result of xor
    wait for 100 ns;
    c<="100"; --result for xnor
    wait for 100 ns;
    c<="101"; --result of or
    wait for 100 ns;
    c<="111"; --result of a
    wait for 100 ns;

    wait for 100 ns;

    --wait for <clock>_period*10;

    -- insert stimulus here

    wait;
end process;

END;
```

```
2. USR MAIN CODE
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 sdfg is
    Port ( D : in  STD_LOGIC_VECTOR (3 downto 0);
          Y : out  STD_LOGIC_VECTOR (3 downto 0);
          clk : in  STD_LOGIC;
          ch : in  STD_LOGIC_VECTOR (1 downto 0));
end sdfg;

architecture Behavioral of sdfg is
    signal temp:STD_LOGIC_VECTOR(3 downto 0);

begin
    process(D,clk)
    begin
        if(clk'event and clk='1') then
            if (ch= "00") then
                temp(0)<=D(0);
                temp(3 downto 1)<=temp(2 downto 0);
                Y(3)<=temp(3);
            elsif(ch ="01") then
                temp(0)<=D(0);
                temp(3 downto 1)<=temp(2 downto 0);
                Y<=temp;
            elsif(ch = "10") then
                temp<=D;
                temp(3 downto 1)<=temp(2 downto 0);
                Y(3)<=temp(3);
            elsif (ch ="11") then
                temp <= D;
                temp(3 downto 1)<=temp(2 downto 0);
                Y<=temp;
            end if;
        end if;
    end process;
end Behavioral;
```

```
USR TB

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 fgrt IS
END fgrt;

ARCHITECTURE behavior OF fgrt IS

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

    COMPONENT sdfg
    PORT(
        D : IN  std_logic_vector(3 downto 0);
        Y : OUT  std_logic_vector(3 downto 0);
        clk : IN  std_logic;
        ch : IN  std_logic_vector(1 downto 0)
    );
    END COMPONENT;

    --Inputs
    signal D : std_logic_vector(3 downto 0) := (others => '0');
    signal clk : std_logic := '0';
    signal ch : std_logic_vector(1 downto 0) := (others => '0');

    --Outputs
    signal Y : std_logic_vector(3 downto 0);

    -- Clock period definitions
    constant clk_period : time := 10 ns;

BEGIN

    -- Instantiate the Unit Under Test (UUT)
    uut: sdfg PORT MAP (
        D => D,
        Y => Y,
        clk => clk,
        ch => ch
    );

    -- Clock process definitions
```

```
-- clock process definitions
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
    D <= "1000";
    ch <= "00";
    -- hold reset state for 100 ns.
    wait for 500 ns;
    ch <= "01";
    wait for 500 ns;
    D <= "0101";
    ch <= "10";
    wait for 500 ns;
    ch <= "11";
    wait for 500 ns;
    wait for clk_period*10;
-- insert stimulus here

    wait;
end process;

END;
```

```
3.KEYPAD MAIN CODE
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 sdfg is
    Port ( rows : in  STD_LOGIC_VECTOR (3 downto 0);
          column : in  STD_LOGIC_VECTOR (3 downto 0);
          v : out STD_LOGIC_VECTOR (7 downto 0));
```

```
        y <= std_logic_vector (/ downto 0));
    end sdfg;
```

architecture Behavioral of sdfg is

```
begin
process(rows, column)
    begin
        if (rows = "0001") then
            if (column = "0001") then
                y <= "00111111";
            elsif (column = "0010") then
                y <= "00000110";
            elsif (column = "0100") then
                y <= "01011011";
            elsif (column = "1000") then
                y <= "01001111";
            end if;
        elsif (rows = "0010") then
            if (column = "0001") then
                y <= "01100110";
            elsif (column = "0010") then
                y <= "01101101";
            elsif (column = "0100") then
                y <= "01111101";
            elsif (column = "1000") then
                y <= "00000111";
            end if;
        elsif (rows = "0100") then
            if (column = "0001") then
                y <= "01111111";
            elsif (column = "0010") then
                y <= "01100111";
            elsif (column = "0100") then
                y <= "01110111";
            elsif (column = "1000") then
                y <= "01111100";
            end if;
        elsif (rows = "1000") then
            if (column = "0001") then
                y <= "00111001";
            elsif (column = "0010") then
                y <= "01011110";
            elsif (column = "0100") then
                y <= "01111001";
            elsif (column = "1000") then
                y <= "01110001";
            end if;
        else
            y <= (others => '0'); -- default output
        end if;
    end process;
end Behavioral;
```

```
KEYPAD_TB
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 rg IS
END rg;

ARCHITECTURE behavior OF rg IS

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

    COMPONENT sdfg
    PORT(
        rows : IN  std_logic_vector(3 downto 0);
        column : IN  std_logic_vector(3 downto 0);
        y : OUT  std_logic_vector(7 downto 0)
    );
    END COMPONENT;

    --Inputs
    signal rows : std_logic_vector(3 downto 0) := (others => '0');
```

```
signal column : std_logic_vector(3 downto 0) := (others => '0');

--Outputs
signal y : std_logic_vector(7 downto 0);
-- No clocks detected in port list. Replace <clock> below with
-- appropriate port name

constant <clock>_period : time := 10 ns;

BEGIN

    -- Instantiate the Unit Under Test (UUT)
    uut: sdfg PORT MAP (
        rows => rows,
        column => column,
        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
        begin
            rows<="0001";
            coloumn<="0001";
            wait for 100 ns;

            rows<="0001";
            coloumn<="0010";
            wait for 100 ns;

            rows<="0001";
            coloumn<="1000";
            wait for 100 ns;

            rows<="0010";
            coloumn<="0001";
            wait for 100 ns;

rows<="0010";
            coloumn<="0010";
            wait for 100 ns;

            rows<="0010";
            coloumn<="0100";
            wait for 100 ns;

            rows<="0010";
            coloumn<="1000";
            wait for 100 ns;

            rows<="0100";
            coloumn<="0001";
            wait for 100 ns;

            rows<="0100";
            coloumn<="0010";
            wait for 100 ns;

rows<="0100";
            coloumn<="0100";
            wait for 100 ns;

            rows<="0100";
            coloumn<="1000";
            wait for 100 ns;

            rows<="1000";
            coloumn<="0001";
            wait for 100 ns;

            rows<="1000";
            coloumn<="0010";
            wait for 100 ns;

rows<="1000";
            coloumn<="0100";
            wait for 100 ns;

            rows<="1000";
            coloumn<="1000";
            wait for 100 ns;
            --wait for <clock>_period*10;

        end;
```

```
wait;
end process;
```

```
END;
```

4.FIFO MAIN CODE

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
```

```
entity fifo is
  generic (
    DEPTH : integer := 8 -- FIFO depth
  );
  Port (
    clk      : in  STD_LOGIC;
    reset    : in  STD_LOGIC;
    enr      : in  STD_LOGIC; -- read enable
    enw      : in  STD_LOGIC; -- write enable
    data_in  : in  STD_LOGIC_VECTOR (7 downto 0);
    data_out : out STD_LOGIC_VECTOR (7 downto 0);
    fifo_empty : out STD_LOGIC;
    fifo_full  : out STD_LOGIC
  );
end fifo;
```

architecture Behavioral of fifo is

```
-- memory declaration
type memory_type is array (0 to DEPTH-1) of STD_LOGIC_VECTOR(7 downto 0);
signal memory : memory_type := (others => (others => '0'));
signal readptr : integer := 0;
signal writeptr : integer := 0;
signal empty : STD_LOGIC := '1';
signal full : STD_LOGIC := '0';
signal num_elem : integer := 0;
```

begin

```
fifo_empty <= empty;
fifo_full <= full;

process(clk, reset)
begin
  if (reset = '1') then
    memory <= (others => (others => '0'));
    data_out <= (others => '0');
    empty <= '1';
    full <= '0';
    readptr <= 0;
    writeptr <= 0;
    num_elem <= 0;
    elsif rising_edge(clk) then
      -- READ
      if (enr = '1' and empty = '0') then
        data_out <= memory(readptr);
        readptr <= (readptr + 1) mod DEPTH;
        num_elem <= num_elem - 1;
      end if;

      -- WRITE
      if (enw = '1' and full = '0') then
        memory(writeptr) <= data_in;
        writeptr <= (writeptr + 1) mod DEPTH;
        num_elem <= num_elem + 1;
      end if;

      -- FLAGS
      if (num_elem = 0) then
        empty <= '1';
      else
        empty <= '0';
      end if;

      if (num_elem = DEPTH) then
        full <= '1';
      else
        full <= '0';
      end if;
    end if;
  end process;
```

```
end Behavioral;
```

FIFO TB

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
```

```
entity fifo_tb is
end fifo_tb;
```

architecture Behavioral of fifo_tb is

```
-- Component Declaration for the Unit Under Test (UUT)
component fifo
  generic (
    DEPTH : integer := 8
```

```
);
Port (
    clk      : in  STD_LOGIC;
    reset    : in  STD_LOGIC;
    enr      : in  STD_LOGIC;
    enw      : in  STD_LOGIC;
    data_in  : in  STD_LOGIC_VECTOR (7 downto 0);
    data_out  : out STD_LOGIC_VECTOR (7 downto 0);
    fifo_empty : out STD_LOGIC;
    fifo_full  : out STD_LOGIC
);
end component;

-- Testbench signals
signal clk_tb      : STD_LOGIC := '0';
signal reset_tb    : STD_LOGIC := '0';
signal enr_tb      : STD_LOGIC := '0';
signal enw_tb      : STD_LOGIC := '0';
signal data_in_tb  : STD_LOGIC_VECTOR (7 downto 0) := (others => '0');
signal data_out_tb  : STD_LOGIC_VECTOR (7 downto 0);
signal fifo_empty_tb : STD_LOGIC;
signal fifo_full_tb  : STD_LOGIC;

-- Clock period definition
constant CLK_PERIOD : time := 10 ns;

begin

-- Instantiate the Unit Under Test (UUT)
 uut: fifo
  port map (
    clk      => clk_tb,
    reset    => reset_tb,
    enr      => enr_tb,
    enw      => enw_tb,
    data_in  => data_in_tb,
    data_out => data_out_tb,
    fifo_empty => fifo_empty_tb,
    fifo_full => fifo_full_tb
  );

-- Clock generation
clk_process : process
begin
    clk_tb <= '0';
    wait for CLK_PERIOD/2;
    clk_tb <= '1';
    wait for CLK_PERIOD/2;
end process;

-- Stimulus process
stim_proc: process
begin
    -----
    -- RESET
    -----
    reset_tb <= '1';
    wait for 20 ns;
    reset_tb <= '0';
    wait for 20 ns;
    -----

    -- WRITE some data
    -----
    for i in 0 to 4 loop
        enw_tb <= '1';
        data_in_tb <= std_logic_vector(to_unsigned(i + 10, 8)); -- 10,11,12,13,14
        wait for CLK_PERIOD;
    end loop;
    enw_tb <= '0';

    wait for 30 ns;
    -----

    -- READ back the data
    -----
    for i in 0 to 4 loop
        enr_tb <= '1';
        wait for CLK_PERIOD;
    end loop;
    enr_tb <= '0';

    -----

    -- Finish simulation
    -----
    wait for 50 ns;
    assert false report "Simulation Finished Successfully" severity note;
    wait;
end process;

end Behavioral;
```

-- Design Name: LCD FSM Controller
-- Module Name: LCD_FSM - Behavioral
-- Description: Finite State Machine for LCD initialization and data display

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

entity LCD_FSM is
    Port (
        rst      : in  STD_LOGIC;           -- reset
        clk_12Mhz : in  STD_LOGIC;          -- high frequency clock
        lcd_rs    : out STD_LOGIC;          -- LCD RS control
        lcd_en    : out STD_LOGIC;          -- LCD Enable
        lcd_data  : out STD_LOGIC_VECTOR(7 downto 0) -- LCD Data port
    );
end LCD_FSM;
```

```
architecture Behavioral of LCD_FSM is

    -- internal signals
    signal div      : STD_LOGIC_VECTOR(15 downto 0) := (others => '0');
    signal clk_fsm  : STD_LOGIC;
    signal lcd_rs_s : STD_LOGIC;
    signal dataout_s : STD_LOGIC_VECTOR(7 downto 0);

    -- FSM states
    type state_type is (s_reset, func, mode, cur, clear, d0, d1, d2, d3, d4, hold);
    signal ps1, nx : state_type;
```

```
begin

    -----
    -- Clock Divider : Generate slow clock for LCD FSM
    -----

    clk_divider : process(rst, clk_12Mhz)
    begin
        if (rst = '1') then
            div <= (others => '0');
        elsif rising_edge(clk_12Mhz) then
            div <= div + 1;
        end if;
    end process;

    clk_fsm <= div(15); -- slow clock for FSM

    -----
    -- Present State Register
    -----

    process(rst, clk_fsm)
    begin
        if (rst = '1') then
            ps1 <= s_reset;
        elsif rising_edge(clk_fsm) then
            ps1 <= nx;
        end if;
    end process;

    -----
    -- Next State and Output Logic
    -----

    process(ps1)
    begin
        case ps1 is

            -----
            when s_reset =>
                nx      <= func;
                lcd_rs_s <= '0';
                dataout_s <= "00111000"; -- 38h: Function set

            -----
            when func =>
                nx      <= mode;
                lcd_rs_s <= '0';
                dataout_s <= "00111000"; -- 38h: 8-bit, 2-line, 5x7 dots

            -----
            when mode =>
                nx      <= cur;
                lcd_rs_s <= '0';
                dataout_s <= "00000110"; -- 06h: Entry mode set

            -----
            when cur =>
                nx      <= clear;
                lcd_rs_s <= '0';
                dataout_s <= "00001100"; -- 0Ch: Display ON, Cursor OFF

            -----
            when clear =>
                nx      <= d0;
                lcd_rs_s <= '0';
                dataout_s <= "00000001"; -- 01h: Clear display
```

```
-----
when d0 =>
    nx          <= d1;
    lcd_rs_s    <= '1';
    dataout_s    <= "01010000";    -- 'P'

-----

when d1 =>
    nx          <= d2;
    lcd_rs_s    <= '1';
    dataout_s    <= "01001001";    -- 'I'

-----

when d2 =>
    nx          <= d3;
    lcd_rs_s    <= '1';
    dataout_s    <= "01000011";    -- 'C'

-----

when d3 =>
    nx          <= d4;
    lcd_rs_s    <= '1';
    dataout_s    <= "01010100";    -- 'T'

-----

when d4 =>
    nx          <= hold;
    lcd_rs_s    <= '1';
    dataout_s    <= "00100000";    -- ' ' (space)

-----

when hold =>
    nx          <= hold;
    lcd_rs_s    <= '0';
    dataout_s    <= "00000000";    -- hold

-----

when others =>
    nx          <= s_reset;
    lcd_rs_s    <= '0';
    dataout_s    <= (others => '0');

    end case;
end process;

-----

-- Output assignments
-----

lcd_en    <= clk_fsm;
lcd_rs    <= lcd_rs_s;
lcd_data <= dataout_s;

end Behavioral;
```

```
5.LCD TO FPGA TB
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
```

```
ENTITY LCD_Test IS
END LCD_Test;
```

```
ARCHITECTURE behavior OF LCD_Test IS
```

```
    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT LCD_FSM
    PORT(
        rst          : IN  std_logic;
        clk_12Mhz    : IN  std_logic;
        lcd_rs       : OUT std_logic;
        lcd_en       : OUT std_logic;
        lcd_data     : OUT std_logic_vector(7 downto 0)
    );
    END COMPONENT;

    -- Inputs
    signal rst          : std_logic := '0';
    signal clk_12Mhz    : std_logic := '0';

    -- Outputs
    signal lcd_rs       : std_logic;
    signal lcd_en       : std_logic;
    signal lcd_data     : std_logic_vector(7 downto 0);

    -- Clock period definition
    constant clk_12Mhz_period : time := 10 ns;
```

```
BEGIN
```

```
    -- Instantiate the Unit Under Test (UUT)
    uut: LCD_FSM
        PORT MAP (
            rst          => rst,
            clk_12Mhz    => clk_12Mhz,
            lcd_rs       => lcd_rs
```

```

    lcd_rs    => lcd_rs,
    lcd_en    => lcd_en,
    lcd_data  => lcd_data
);

-----
-- Clock Generation Process
-----

clk_12Mhz_process : process
begin
    clk_12Mhz <= '0';
    wait for clk_12Mhz_period/2;
    clk_12Mhz <= '1';
    wait for clk_12Mhz_period/2;
end process;

-----
-- Stimulus Process
-----

stim_proc : process
begin
    -- Apply reset
    rst <= '1';
    wait for 20 ns;
    rst <= '0';

    -- Allow simulation to run
    wait for 1000 ns;

    -- End simulation
    assert false report "Simulation completed successfully" severity note;
    wait;
end process;

END behavior;
```