

## 4 BIT ALU

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.all;

use IEEE.STD_LOGIC_UNSIGNED.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

net "a[0]" LOC = "P75";

net "a[1]" LOC = "P64";

net "a[2]" LOC = "P58";

net "a[3]" LOC = "P6";

# SW 8 to SW 11

net "b[0]" LOC = "P116";

net "b[1]" LOC = "P67";

net "b[2]" LOC = "P61";

net "b[3]" LOC = "P8";

# SW 15 to SW 12

net "op[0]" LOC = "P126";

net "op[1]" LOC = "P133";

net "op[2]" LOC = "P139";

# LI 1 to LI 5

net "c[0]" LOC = "P2";

net "c[1]" LOC = "P142";

net "c[2]" LOC = "P138";

net "c[3]" LOC = "P134";

net "c[4]" LOC = "P124";

## MOD N COUNTER

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_arith.ALL;

use IEEE.STD_LOGIC_unsigned.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;
```

## SHIFT REGISTER

```
library IEEE;
use IEEE.STD_LOGIC_1164.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) & lin;
            elsif(s = "11") then
                q <= d;
            end if;
        end if;
    end process;
end Behavioral;
```

## KEYPAD SEVEN SEGMENT DISPLAY

```
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";           -- Assign "clk" signal to pin P56
NET "rst" LOC = "P22";           -- Assign "rst" signal to pin P22
NET "clk" CLOCK_DEDICATED_ROUTE = true;    -- Enable dedicated clock routing for "clk"
NET "rst" CLOCK_DEDICATED_ROUTE = FALSE;    -- Disable dedicated clock routing for "rst"

-- Mapping display outputs to pins
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";

-- Mapping digit outputs to pins
NET "digit[0]" LOC = "P46";
NET "digit[1]" LOC = "P43";
NET "digit[2]" LOC = "P44";
NET "digit[3]" LOC = "P40";

-- Mapping column inputs to pins with pull-up resistors and drive strength
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;
```



-- Mapping row inputs to pins

NET "row[0]" LOC = "P64";

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

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

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

## KEYPAD INTERFACING WITH LCD

```
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",
        x"4B", x"45", x"59", x"20", x"50", x"52", x"45", x"53", x"53", x"3A" );
```

```

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 =>
                    if i < 5 then
                        R_W <= '0';
                        R_S <= '0';
                    end if;
                end case;
            end process;
        end if;
    end process;
end

```

```
else
    R_S <= '1';
end if;
Ldata <= datas(i);
E_N <= '1';
step <= 1;
when 1 =>
    i <= i + 1;
    step <= 2;
when 2 =>
    if i = 16 then
        step <= 3;
    else
        step <= 0;
    end if;
when 3 =>
    R_S <= '0';
    E_N <= '1';
    step <= 4;
when 4 =>
    step <= 5;
when 5 =>
    R_S <= '1';
    Ldata <= x"8B";
    E_N <= '0';
    Ldata <= data;
    E_N <= '0';
    E_N <= '1';
    step <= 6;
```

```

        when 6 =>
            step <= 3;
        end case;
    end if;
end process;
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";
                    when "1011" => data <= x"44";
                    when "1101" => data <= x"45";
                    when "1110" => data <= x"46";
                    when others => rowtest <= 1;
                end case;
            when 1 =>
                rowm <= "1011";
                case COL is
                    when "0111" => data <= x"30";
                    when "1011" => data <= x"31";
                    when "1101" => data <= x"32";
                    when "1110" => data <= x"33";
                end case;
            end case;
        end case;
    end if;
end process;

```

```

        when others => rowtest <= 2;
    end case;
when 2 =>
    rowm <= "1101";
    case COL is
        when "0111" => data <= x"34";
        when "1011" => data <= x"35";
        when "1101" => data <= x"36";
        when "1110" => data <= x"37";
        when others => rowtest <= 3;
    end case;
when 3 =>
    rowm <= "1110";
    case COL is
        when "0111" => data <= x"38";
        when "1011" => data <= x"39";
        when "1101" => data <= x"41";
        when "1110" => data <= x"42";
        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 and Reset 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;

# LCD Data Pins

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 Lines

NET "RS" LOC = "P88";

NET "EN" LOC = "P84";

NET "RW" LOC = "P140";

# Column Pins with Pull-up and Drive Strength

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;

# Row Pins

NET "ROW[0]" LOC = "P64";

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

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

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

## LCD INTERFACING

```
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

    -- Clock divider to slow down the clock

    process(rst, clk)

        variable temp: integer range 0 to 999999;

    begin

        if (rst = '0') then

            temp := 0;

            clock <= '0';

        elsif (clk'event and clk = '1') 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 (rst = '0') then
        count <= 0;
    elsif (rising_edge(clock)) then
        if (count < 47 and flag = '0') then
            count <= count + 1;
        end if;
    end if;
end process;

-- Process to allocate output on counter values and write on LCD in HEX format
process(count, rst, clock)
begin
    if (rst = '0') then
        RW <= '0';
        RS <= '0';
        EN <= '0';
        LCD <= (others => '0');
        flag <= '0';
    elsif (rising_edge(clock)) then
        case count is

```

when 0 =>

    RW <= '0';

    EN <= '0';

when 1 =>

    RW <= '0';

    EN <= '0';

when 2 =>

    RW <= '0';

when 3 =>

    EN <= '0';

when 4 =>

    RW <= '0';

when 5 =>

    EN <= '0';

when 6 =>

    RW <= '0';

    RS <= '0';

    LCD <= x"38"; -- init in 2 line mode

    EN <= '1';

when 7 =>

    EN <= '0';

when 8 =>

    RW <= '0';

when 9 =>

    EN <= '0';

when 10 =>

    RW <= '0';

    EN <= '0';

when 11 =>

```
RW <= '0';  
when 12 =>  
    EN <= '0';  
when 13 =>  
    RW <= '0';  
when 14 =>  
    EN <= '0';  
    RS <= '0';  
    LCD <= x"81"; -- set cursor to 1st line, 2nd position  
    EN <= '1';  
when 15 =>  
    RS <= '1';  
    LCD <= x"4A"; -- J  
    EN <= '1';  
when 16 =>  
    RS <= '1';  
    LCD <= x"53"; -- S  
    EN <= '1';  
when 17 =>  
    RS <= '1';  
    LCD <= x"50"; -- P  
    EN <= '1';  
when 18 =>  
    RS <= '1';  
    LCD <= x"4D"; -- M  
    EN <= '1';  
when 19 =>  
    RS <= '1';  
    LCD <= x"20"; -- space
```

```
EN <= '1';  
when 20 =>  
    RS <= '1';  
    LCD <= x"47"; -- G  
    EN <= '1';  
when 21 =>  
    RS <= '1';  
    LCD <= x"52"; -- R  
    EN <= '1';  
when 22 =>  
    RS <= '1';  
    LCD <= x"4F"; -- O  
    EN <= '1';  
when 23 =>  
    RS <= '1';  
    LCD <= x"55"; -- U  
    EN <= '1';  
when 24 =>  
    RS <= '1';  
    LCD <= x"50"; -- P  
    EN <= '1';  
when 25 =>  
    RS <= '1';  
    LCD <= x"20"; -- space  
    EN <= '1';  
when 26 =>  
    RS <= '1';  
    LCD <= x"20"; -- space  
    EN <= '1';
```

when 27 =>

RS <= '1';

LCD <= x"20"; -- space

EN <= '1';

when 28 =>

RS <= '1';

LCD <= x"20"; -- space

EN <= '1';

when 29 =>

RS <= '0';

LCD <= x"C6"; -- 2nd line

EN <= '1';

when 30 =>

RS <= '1';

LCD <= x"50"; -- P

EN <= '1';

when 31 =>

RS <= '1';

LCD <= x"55"; -- U

EN <= '1';

when 32 =>

RS <= '1';

LCD <= x"4E"; -- N

EN <= '1';

when 33 =>

RS <= '1';

LCD <= x"45"; -- E

EN <= '1';

when 34 =>

```

        flag <= '1';

    when others =>

        null; -- No action

    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;

# Pins 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 lines for LCD

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

```