Lab 6

Pragaash Mohan

November 1, 2020

Part 1

Top level entity

```
--Part 2
--8bit Accumulator w/add and sub

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;


entity part_2 is
        port(
                KEY : in std_logic_vector(1 downto 0);
                SW  : in std_logic_vector(8 downto 0);
                LEDR : out std_logic_vector(9 downto 0);
                HEX0, HEX1, HEX2, HEX3 : out std_logic_vector(0 to 6)
                );
        end part_2;

-------------------------------------------------------------

architecture behav of part_2 is

                --adder component
                component adc
                generic (n: integer := 8);
                port
                (

                        A               :  in std_logic_vector(n-1 downto 0);
                        B               :  in std_logic_vector(n-1 downto 0);
                        add_sub         :  in std_logic;
```

```
        sum         :  out std_logic_vector(n-1 downto 0);
        carry1          :  out std_logic
    );

    end component;

    --hex display
    component Hex7 port
    (
        c               : in std_logic_vector(3 downto 0);
        disp        : out std_logic_vector(0 to 6)
);
    end component;

    signal w                    : std_logic_vector(7 downto 0);
    signal A, B, S, bchange : std_logic_vector(7 downto 0);

    signal C, carry, overflow : std_logic;


    begin


        f8bit : adc
        generic map (n => 8)
        port map (A,B,SW(8), w, C);



        process(KEY) begin --clock input

        if rising_edge(KEY(1)) then

            A <= SW(7 downto 0);

            S <=  w(7 downto 0);
            carry <= C;
            B <= S;
            overflow <= C XOR (B(7) XOR A(7));

        end if;
```

```
                    if ((KEY(0) = '0')) then --RESET

                            A <= "00000000";
                            B <= "00000000";
                            S <= "00000000";
                            carry <= '0';
                            overflow <= '0';
                    end if;
            end process;




    LEDR(7 downto 0) <= S;
    LEDR(8) <= carry;
    LEDR(9) <= overflow;




    --display what to add
    disp_3 : Hex7 port map (A(7 downto 4), HEX3 );
    disp_2 : Hex7 port map (A(3 downto 0), HEX2 );

    --display sum
    disp_1 : Hex7 port map (S(7 downto 4), HEX1 );
    disp_0 : Hex7 port map (S(3 downto 0), HEX0 );


end behav;
```

Adder

```
--8bit adder w/carry

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;


entity adc is

        generic (n: integer := 8);
        port(

                        A               :  in std_logic_vector(n-1 downto 0);
                        B               :  in std_logic_vector(n-1 downto 0);
                        add_sub         :         in std_logic;
                        sum         :  out std_logic_vector(n-1 downto 0);
                        carry1 : out std_logic

                );

        end adc;


        architecture behav of adc is


        signal temp_1 : std_logic_vector(n downto 0);

        begin

                process (A, B, add_sub) begin
                        if (add_sub = ('1') and B >= A) then
                           temp_1 <= ("0" & B) - ("0" & A);
                else
                                temp_1 <= ("0" & A) + ("0" & B);
                end if;
                end process;

                sum <= temp_1(n-1 downto 0);
                carry1 <= temp_1(n);

        end behav;
```

Hex7

```
--Creating hexadecimal display

library ieee;
use ieee.std_logic_1164.all;

entity Hex7 is
        port
        (
                c               : in std_logic_vector(3 downto 0);
                disp        : out std_logic_vector(0 to 6)
        );
end Hex7;


architecture behav of Hex7 is
begin

        process (c)
        begin
                case c is
                        when "0000" => disp <= "0000001"; -- 0
                        when "0001" => disp <= "1001111"; -- 1
                        when "0010" => disp <= "0010010"; -- 2
                        when "0011" => disp <= "0000110"; -- 3
                        when "0100" => disp <= "1001100"; -- 4
                        when "0101" => disp <= "0100100"; -- 5
                        when "0110" => disp <= "0100000"; -- 6
                        when "0111" => disp <= "0001111"; -- 7
                        when "1000" => disp <= "0000000"; -- 8
                        when "1001" => disp <= "0000100"; -- 9
                        when "1010" => disp <= "0001000"; -- A
                        when "1011" => disp <= "1100000"; -- B
                        when "1100" => disp <= "0110001"; -- C
                        when "1101" => disp <= "1000010"; -- D
                        when "1110" => disp <= "0110000"; -- E
                        when "1111" => disp <= "0111000"; -- F
                        when others => disp <= "1111111";
                end case;
```

```
        end process;

end behav;
```

Part 4

```
--Part 4
--using the same component for hexadecimal display
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;


entity part_4 is

        port
        (
                SW                : in std_logic_vector(9 downto 0);
                KEY         : in std_logic_vector(1 downto 0);
                LEDR          : out std_logic_vector(9 downto 0);
                HEX0, HEX1, HEX2, HEX3 : out std_logic_vector(0 to 6)
        );

end part_4;

architecture behav of part_4 is

        --hex display
        component Hex7 port
                (
                        c                : in std_logic_vector(3 downto 0);
                        disp         : out std_logic_vector(0 to 6)
           );
        end component;

        signal Set : std_logic_vector(1 downto 0);
        signal A,B,sw0 : std_logic_vector(7 downto 0);
        signal S : std_logic_vector(16 downto 0);

        begin

                Set <= SW(9 downto 8);
                sw0 <= SW(7 downto 0);

                process(KEY, A,B)
                variable ivec : std_logic_vector(7 downto 0);
                variable s0 : std_logic_vector(7 downto 0);
```

```
        variable s1        : std_logic_vector(16 downto 0);

        begin
                if rising_edge(KEY(1)) then --manual clock
                        if ((KEY(0) = '0')) then --RESET
                                S <= "00000000000000000";
                                A <= "00000000";
                                B <= "00000000";
                        else
                        if(Set  = "10") then
                                A <= sw0;
                        elsif(Set = "01") then
                                B <= sw0;
                        elsif(Set ="11") then
                                for i in 0 to 7 loop
                                        ivec := (A(7) and B(i), A(6) and B
           S0 := S1((i+7) downto i);
           S1((i+8) downto i) := (("0" & S0) + ("0" & ivec));
        end loop;
                                 S <= s1;
                                 A <= "00000000";
                                 B <= "00000000";
                                end if;
                        end if;
                end if;
        end process;
process(S,set,A,B)
begin
        case Set is
        when "10" => LEDR(7 downto 0) <= A;
        when "01" => LEDR(7 downto 0) <= B;
        when others => LEDR(7 downto 0) <= "00000000";
        end case;
end process;



--Display
disp_0 : Hex7 port map (S(3 downto 0), HEX0);
disp_1 : Hex7 port map (S(7 downto 4), HEX1);

disp_3 : Hex7 port map (S(11 downto 8), HEX2);
disp_4 : Hex7 port map (s(15 downto 12), HEX3);
```

```
end behav;
```