Lab 7

Pragaash Mohan

November 6, 2020

Part 2

Top level entity

```
--Task 2
--Creating modulo-k counter
--modified 8bit counter


library ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;

entity part_2 is
        port
        (
                KEY         : in std_logic_vector(1 downto 0);
                CLOCK_50 : in std_logic;
                o_put                : out std_logic_vector(11 downto 0);
                HEX0, HEX1, HEX2        : out std_logic_vector(0 to 6)

        );

end part_2;

architecture behav of part_2 is
        component counter
        generic
        (
                n : integer := 4;
                k : integer := 8

        );
```

```
        port
        (

                clk         : in std_logic;
                R                   : in std_logic;
                q                   : out std_logic_vector(n-1 downto 0)
                --rollover : out std_logic
        );

        end component;

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

        end component;

signal CYCLES : std_logic_vector(25 downto 0);

--Hex output
signal s0 : std_logic_vector(3 downto 0);
signal s1 : std_logic_vector(3 downto 0);
signal s2 : std_logic_vector(3 downto 0);

--1s, 10s, 100s input
signal ons,tns,hunds : std_logic;

--base and new
signal x1,y1,z1 : std_logic;
signal x2,y2,z2 : std_logic;



        begin

        --output for hex 0-2
        o_put(3 downto 0) <= s0;
        o_put(7 downto 4) <= s1;
        o_put(11 downto 8) <= s2;
```

```
--Base and new. Change when KEY(0) is registered
x2 <= x1 and KEY(0);
y2 <= y1 and KEY(0);
z2 <= z1 and KEY(0);

CT                : counter
        generic map ( n => 26, k => 50000000 )
        port map ( CLOCK_50,KEY(0),CYCLES );

C1                : counter
        generic map ( n => 4, k => 11 )
        port map ( ons, x2, s0 );

C10         : counter
        generic map ( n => 4, k => 11 )
        port map ( tns, y2, s1 );

C100  : counter
        generic map ( n => 4, k => 11 )
        port map ( hunds, z2, s2 );


process (CLOCK_50) begin

if rising_edge(CLOCK_50) then

        if (CYCLES = 49999999)then
                ons <= '1';
        else
                ons <= '0';
        end if;

        if (s0 = 10) then
                tns <= '1';
                x1        <= '0';
        else
                tns <= '0';
                x1 <= '1';
        end if;

        if (s1 = 10) then
                hunds <= '1';
                y1 <= '0';
        else
```

```
                        hunds <= '0';
                        y1 <= '1';
                end if;

                z1 <= '1';

        end if;

end process;


d_ones          : Hex7 PORT MAP (s0, HEX0);
d_tens          : Hex7 PORT MAP (s1, HEX1);
d_hundreds : Hex7 PORT MAP (s2, HEX2);


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;
```

Counter

```
library ieee;
use ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;


entity counter is
        generic
        (
                n : integer := 4;
                k : integer := 8
        );

        port
        (
                clk        : in std_logic;
                R                : in std_logic;
                q                : out std_logic_vector(n-1 downto 0)
                rollover : out std_logic
        );

end counter;

architecture behav of counter is
        signal q0 : std_logic_vector(n-1 downto 0);

        begin
                process(clk, R)
                begin
                        if R = '0' then
                                q0 <= (others => '0');
                                rollover <= '0';

                        elsif rising_edge(clk) then

                                if q0 >= k-1 then
                                        q0 <= (others => '0');
                                        rollover <= '1';
                                else
                                        q0 <= q0 + 1;

                                end if;
```

```
                end if;

        end process;
        q <= q0;

end behav;
```

Part 3

Top level entity

```
--Task 3
--Creating a real-time clock
--modified previous assignment
--added some finesse


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

entity part_3 is
        port
        (
                KEY
                CLOCK_50
                SW
                LEDR
                HEX0, HEX1, HEX2, HEX3, HEX4, HEX5        : out std_logic_vector(0 t

        );

end part_3;

architecture behav of part_3 is

        --importing counter
        component counter
        generic
        (
                n        : integer := 4;
                k        : integer := 8
        );

        port
        (
                clk        : in std_logic;
                R                : in std_logic;
                q                : out std_logic_vector(n-1 downto 0)
                --rollover        : out std_logic --not in use
        );
```

```
        end component;

        --importing modified counter
        component counter0
        generic
        (
                n        : integer        :=4;
                k        : integer        :=8
        );

        port
        (
                clk                : in std_logic;
                R                        : in std_logic;
                load                : in std_logic;
                datain          : in std_logic_vector(n-1 downto 0);
                q                 : out std_logic_vector(n-1 downto 0)
                --rollover         : out std_logic --not in use
        );

        end component;



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

        end component;

signal CYCLES : std_logic_vector(25 downto 0);

--pause statement
signal pause_now : std_logic;
--hex output
signal s0,s1  : std_logic_vector(3 downto 0); --1/100, 1/10
signal s2,s3  : std_logic_vector(3 downto 0); --1, 10
signal s4,s5  : std_logic_vector(3 downto 0); --100, 1000
```

```
--1/100, 1/10, 1, 10, 100, 1000 input
signal hunds0, ons0, ons, tns, hunds, tous : std_logic;

--base and new
signal a1, b1, c1, d1, e1, f1        : std_logic;
signal a0, b0, c0, d0, e0, f0 : std_logic;



begin

   --Reset state and base
        a1 <= a0;
        b1 <= b0;
        c1 <= c0;
        d1 <= d0;
        e1 <= e0;
        f1 <= f0;



        cT                : counter
                        generic map (n=> 26, k => 500000) --equates to about 0.01 s
                        port map (CLOCK_50, '1', CYCLES);

        c1s         : counter
                        generic map (n => 4, k => 11)
                        port map (hunds0, a1, s0);

        c2                : counter
                        generic map (n => 4, k => 11)
                        port map (ons0, b1, s1);

        c3                : counter
                        generic map (n => 4, k => 11)
                        port map (ons, c1, s2);

        c4                : counter
                        generic map (n => 4, k => 7)
                        port map(tns, d1, s3);


                        --New counter with modifiable values. Values above 9 is ign
```

```
c5                : counter0
                  generic map (n => 4, k => 11)
                  port map(hunds, e1, KEY(1), SW(3 downto 0), s4);

c6                : counter0
                  generic map (n => 4, k => 7)
                  port map(tous, f1, KEY(1), SW(7 downto 4), s5);



process (CLOCK_50, pause_now)




begin


--process runs only when KEY(0) is not pressed
if (pause_now = '1') then

        if rising_edge(CLOCK_50) then

                if (CYCLES = 499999) then
                        hunds0 <= '1';
                else
                        hunds0 <= '0';
                end if;

                if (s0 = 10) then
                        ons0 <= '1';
                        a0 <= '0';
                else
                        ons0 <= '0';
                        a0 <= '1';
                end if;

                if (s1 = 10) then
                        ons <= '1';
                        b0 <= '0';
                else
```

```
                ons <= '0';
                b0 <= '1';
        end if;

        if (s2 = 10) then
                tns <= '1';
                c0 <= '0';
        else
                tns <= '0';
                c0 <= '1';
        end if;


        if (s3 = 6) then
                hunds <= '1';
                d0 <= '0';
        else
                hunds <= '0';
                d0 <= '1';
        end if;


        if (s4 = 10) then
                tous <= '1';
                e0 <= '0';
        else
                tous <= '0';
                e0 <= '1';
        end if;

        if (s5 = 6) then
                tous <= '0';
                f0 <= '0';
        else
                f0 <= '1';
        end if;


end if;
```

```
        else
        --pause


        end if;


        end process;


d_hundredths   : Hex7 PORT MAP (s0, HEX0);
d_tenths                        : Hex7 PORT MAP (s1, HEX1);
d_ones                    : Hex7 PORT MAP (s2, HEX2);
d_tens              : Hex7 PORT MAP (s3, HEX3);
d_hundreds     : Hex7 PORT MAP (s4, HEX4);
d_thousends    : Hex7 PORT MAP (s5, HEX5);

pause_now <= KEY(0);

--lights up binary values to overwrite
LEDR(3 downto 0) <= SW(3 downto 0);
LEDR(7 downto 4) <= SW(7 downto 4);

end behav;
```

Counter

```
--modified counter for switching

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


entity counter0 is
        generic
        (
                n : integer := 4;
                k : integer := 8
        );

        port
        (
                clk        : in std_logic;
                R                : in std_logic;
                load  : in std_logic;
                datain : in std_logic_vector(n-1 downto 0);
                q                : out std_logic_vector(n-1 downto 0);
                rollover : out std_logic
        );

end counter0;

architecture behav of counter0 is
        signal q0 : std_logic_vector(n-1 downto 0);

        begin
                process(clk, R, load, datain)
                begin

                        if load = '1' then
                                if R = '0' then
                                        q0 <= (others => '0');
                                        rollover <= '0';

                                elsif rising_edge(clk) then
                                        if q0 >= k-1 then
                                                q0 <= (others => '0');
```

```
                                        rollover <= '1';
                                    else
                                        q0 <= q0 + 1;
                                    end if;
                                end if;

                        else
                        q0 <= datain;
                        end if;

                end process;
                q <= q0;


end behav;
```