Ομάδα Α10:

Παγώνης Γεώργιος : 03117030

Δημήτριος - Σταμάτιος Μπούρας : 03117072

8ο εξάμηνο - ΨΗΦΙΑΚΑ ΣΥΣΤΗΜΑΤΑ VLSI

ΣΗΜΜΥ

1η σειρά ασκήσεων

Άσκηση Α.2

Η περιγραφή του δυαδικού αποκωδικοποιητή 3 σε 8 και της αρχιτεκτονικής του σε dataflow VHDL είναι η εξής :

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


entity dec3to8_dataflow is
port(
    dec_in :    in std_logic_vector(2 downto 0);
    dec_out :   out std_logic_vector(7 downto 0)
);
end dec3to8_dataflow;


architecture dataflow_arch of dec3to8_dataflow is
```

```
begin

    with dec_in select dec_out <=

        "00000001" when "000",

        "00000010" when "001",

        "00000100" when "010",

        "00001000" when "011",

        "00010000" when "100",

        "00100000" when "101",

        "01000000" when "110",

        "10000000" when "111",

        "--------" when others;

end dataflow_arch;
```

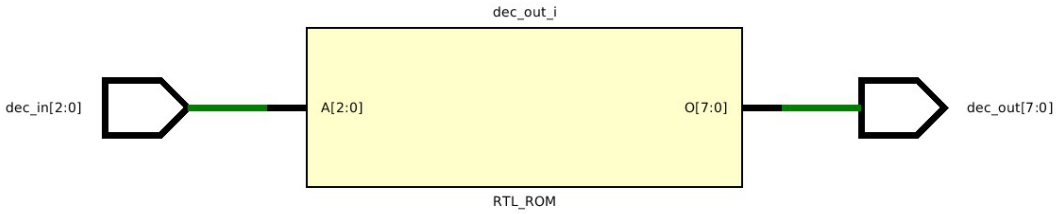
Το σχετικό testbench είναι το εξής:

```
library ieee;

use ieee.std_logic_1164.all;

use ieee.numeric_std.all;



entity dec3to8_dataflow_tb is

end entity;



architecture bench of dec3to8_dataflow_tb is



  component dec3to8_dataflow is

    port(

  dec_in :  in std_logic_vector(2 downto 0);

  dec_out : out std_logic_vector(7 downto 0)



);
```

```
  end component;

  signal input_A  : std_logic_vector(2 downto 0) := (others => '0');
  signal output_B  : std_logic_vector(7 downto 0);

  constant TIME_DELAY : time := 10 ns;

begin

  uut : dec3to8_dataflow
    port map (
              dec_in  => input_A,
              dec_out  => output_B

              );

  stimulus : process
  begin
    input_A <= (others => '0');

    wait for TIME_DELAY;
    ----- Stimulus example 1 -----
    input_A <= "000";
    wait for TIME_DELAY;
    input_A <= "001";

    wait for TIME_DELAY;
```

```
    input_A <= "010";

    wait for TIME_DELAY;
    input_A <= "011";

    wait for TIME_DELAY;
    input_A <= "100";

    wait for TIME_DELAY;
    input_A <= "101";

    wait for TIME_DELAY;
    input_A <= "110";

    wait for TIME_DELAY;
    input_A <= "111";

    wait for TIME_DELAY;
    input_A <= (others => '0');

    wait;
  end process;

end architecture;
```

Το RTL σχηματικό του κυκλώματος που προκύπτει είναι το εξής:



Η περιγραφή του δυαδικού αποκωδικοποιητή 3 σε 8 και της αρχιτεκτονικής του σε behavioral VHDL είναι η εξής :

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



entity dec3to8_behavioral is
port(
    dec_in :    in std_logic_vector(2 downto 0);
    dec_out :   out std_logic_vector(7 downto 0);
    e :in std_logic
);
end dec3to8_behavioral;


------------------------------------------------
```

```
architecture behavioral_arch of dec3to8_behavioral is
begin



    process (dec_in,e)
    begin

       if  e ='1' then
        case dec_in is
            when "000" => dec_out <= "00000001";
            when "001" => dec_out <= "00000010";
            when "010" => dec_out <= "00000100";
            when "011" => dec_out <= "00001000";
            when "100" => dec_out <= "00010000";
            when "101" => dec_out <= "00100000";
            when "110" => dec_out <= "01000000";
            when "111" => dec_out <= "10000000";
            when others => dec_out <= "--------";
        end case;
        else
            dec_out <= "00000000";
        end if ;




    end process;
```

```
end architecture;
```

Το σχετικό testbench είναι το εξής:

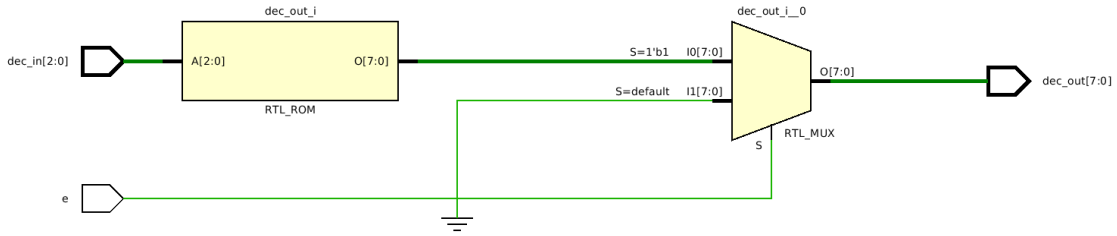
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;


entity dec3to8_behavioral_tb is
end entity;


architecture bench of dec3to8_behavioral_tb is


  component dec3to8_behavioral is
    port(
  dec_in :  in std_logic_vector(2 downto 0);
  dec_out : out std_logic_vector(7 downto 0);
  e :in std_logic
);
  end component;


  signal input_A  : std_logic_vector(2 downto 0) := (others => '0');
  signal output_B  : std_logic_vector(7 downto 0);
  signal input_C : std_logic := '0'  ;


  constant TIME_DELAY : time := 10 ns;
```

```
begin

  uut : dec3to8_behavioral
    port map (
              dec_in  => input_A,
              dec_out  => output_B,
              e => input_C
              );

  stimulus : process
  begin
    input_A <= (others => '0');
    input_C <= '0';
    wait for TIME_DELAY;
    ----- Stimulus example 1 -----
    input_A <= "000";
    wait for TIME_DELAY;
    input_A <= "001";

    wait for TIME_DELAY;
    input_A <= "010";

    wait for TIME_DELAY;
    input_A <= "011";

    wait for TIME_DELAY;
    input_A <= "100";
```

```
wait for TIME_DELAY;
input_A <= "101";

wait for TIME_DELAY;
input_A <= "110";


wait for TIME_DELAY;
input_A <= "111";


wait for TIME_DELAY;
input_A <= "000";
input_C <=  '1';

wait for TIME_DELAY;
input_A <= "001";

wait for TIME_DELAY;
input_A <= "010";

wait for TIME_DELAY;
input_A <= "011";


wait for TIME_DELAY;
input_A <= "100";

wait for TIME_DELAY;
input_A <= "101";

wait for TIME_DELAY;
```

```
    input_A <= "110";


    wait for TIME_DELAY;
    input_A <= "111";

    wait for TIME_DELAY;
    input_A <= (others => '0');
    input_C <= '0';
    wait;

  end process;


end architecture;
```

Το RTL σχηματικό του κυκλώματος που προκύπτει :



Άσκηση Β.2

Η περιγραφή της οντότητας του καταχωρητή ολίσθησης με μια επιπλέον είσοδο και της αρχιτεκτονικής του σε behavioral VHDL είναι η εξής :

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;



entity right_left_shift_reg is
    port (
    clk,rst,en,pl: in std_logic;
    new_bit :in std_logic;
    right_or_left : in std_logic;
    din: in std_logic_vector(3 downto 0);
    output_bit: out std_logic;
    output_din : out std_logic_vector(3 downto 0)
);
end right_left_shift_reg;


architecture behavioral_arch of right_left_shift_reg is


    signal reg_bits: std_logic_vector(3 downto 0);
    begin
        edge: process (clk,rst)
            begin
                if rst='0' then
                    reg_bits<=(others=>'0');
                elsif clk'event and clk='1' then
                        if pl='1' then
                            reg_bits <= din;
                        elsif en='1' then
                            case right_or_left is
                                when '0'=>  output_bit <= reg_bits(0);
                                            reg_bits <= new_bit & reg_bits(3 downto 1);
```

```
                                    output_din <= reg_bits;


                            when '1'=>  output_bit <= reg_bits(3);
                                        reg_bits <= reg_bits(2 downto 0) & new_bit;
                                        output_din <= reg_bits;

                            when others =>   reg_bits <="XXXX";

                            end case;


                        end if;
                end if;


        end process;

end behavioral_arch;
```

Το σχετικό testbench είναι το εξής:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;


entity right_left_shift_reg_tb is
end entity;
--2,250ns


architecture bench of right_left_shift_reg_tb is
```

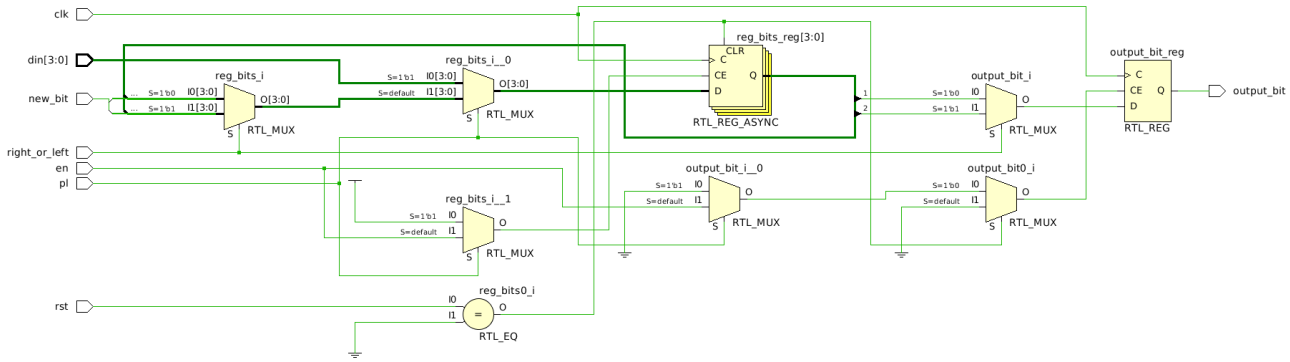
```
  component right_left_shift_reg is
    port(
        clk,rst,en,pl: in std_logic;
        new_bit :in std_logic;
        right_or_left : in std_logic;
        din: in std_logic_vector(3 downto 0);
        output_bit: out std_logic;
        output_din : out std_logic_vector(3 downto 0)
        );
  end component;
    signal input_clk : std_logic ;
    signal input_rst : std_logic := '0'  ;
    signal input_en : std_logic := '0'  ;
    signal input_pl : std_logic := '0'  ;
    signal input_new_bit : std_logic := '0'  ;
    signal input_right_or_left : std_logic := '0'  ;
    signal input_din : std_logic_vector(3 downto 0) := (others => '0');
    signal output_output_bit  : std_logic;
    signal output_output_din : std_logic_vector(3 downto 0) ;


  constant TIME_DELAY : time := 80 ns;
  constant CLOCK_PERIOD : time := 10 ns;

begin

    module:right_left_shift_reg
```

```
port map (
    clk =>input_clk,
    rst=>input_rst,
    en=>input_en,
    pl=>input_pl,
    new_bit =>input_new_bit,
    right_or_left =>input_right_or_left,
    din =>input_din,
    output_bit =>output_output_bit,
    output_din =>output_output_din
);


stimulus : process
begin
  input_rst <= '1';
  input_new_bit <= '0';
  input_en <= '0';
  input_pl <= '0';
  input_right_or_left <= '0';
  input_din <= (others => '0');
  wait for CLOCK_PERIOD;


  for i  in std_logic range '0' to '1' loop -- for shift_left=0 shift_right=1
    input_right_or_left <= i;
    input_en <= '1';

    input_new_bit <= '0';
    for n in 0 to 15 loop
      input_pl <= '1';
      input_din <= std_logic_vector(to_unsigned(n, 4));
```

```
            wait for CLOCK_PERIOD;
            input_pl <= '0';
            wait for TIME_DELAY;
          end loop;


          input_new_bit <= '1';

          for n in 0 to 15 loop
            input_pl <= '1';
            input_din <= std_logic_vector(to_unsigned(n, 4));
            wait for CLOCK_PERIOD;
            input_pl <= '0';
            wait for TIME_DELAY;
          end loop;
        end loop;



      -- for i  in std_logic range '0' to '1' loop -- input_rst loop
        -- for j in std_logic range '0' to '1' loop -- input_new_bit loop
          --     for k in std_logic range '0' to '1' loop -- input_en loop
            --       for l in std_logic range '0' to '1' loop -- input_pl loop
              --         for m in std_logic range '0' to '1' loop -- input_right_or_left
loop
                ---          for n in 0 to 15 loop -- input_din loop
                    --            input_rst <= i;
                      --          input_new_bit <= j;
                        --        input_en <= k;
                          --      input_pl <= l;
                            --    input_right_or_left <= m;
                              --  input_din <= std_logic_vector(to_unsigned(n, 4));
                                --wait for TIME_DELAY;
```

```
    --                    end loop;
      --               end loop;
        --         end loop;
          --    end loop;
      --  end loop;
    --end loop;
      input_rst <= '0';
      input_new_bit <= '0';
      input_en <= '0';
      input_pl <= '0';
      input_right_or_left <= '0';
      input_din <= (others => '0');
      wait;
    end process;


  generate_clock : process
  begin
    input_clk <= '0';
    wait for CLOCK_PERIOD/2;
    input_clk <= '1';
    wait for CLOCK_PERIOD/2;
  end process;
  end architecture;
```

Το RTL σχηματικό του κυκλώματος που προκύπτει :



Άσκηση Β.3

Ζητούμενο 1)

Η περιγραφή της οντότητας μετρητή up/down των 3 bits και της αρχιτεκτονικής του σε behavioral VHDL είναι η εξής :

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;



entity counter_up_down_3bit is
port(
    clk, resetn , count_en : in std_logic;
    add_or_sub : in std_logic;
    sum : out std_logic_vector(2 downto 0);
    cout : out std_logic
    );
end entity;


architecture behavioral_arch of counter_up_down_3bit is
    signal count : std_logic_vector(2 downto 0);
    begin
        process(clk, resetn)
```

```
begin
    if resetn='0' then
    -- Ασύγχρονος μηδενισμός
        count <= (others=>'0');
    elsif clk'event and clk='1' then


            if count_en = '1' then
            -- Μέτρηση μόνο αν count_en=’1’
                case add_or_sub is
                when '0' =>
                    if count/=7 then
                        -- Αυξάνουμε το μετρητή μόνο αν
                        -- δεν είναι 7
                            count <= count+1;
                            cout <= '0' ;
                        else
                            -- Αλλιώς τον μηδενίζουμε
                            count<=(others=>'0');
                            cout <= '1' ;
                        end if;
                when '1'=>
                    if count/=0 then
                        -- Μειώνουμε το μετρητή μόνο αν
                        -- δεν είναι 0
                            count <= count-1;
                            cout <= '0' ;
                        else
                            -- Αλλιώς τον κάνουμε 7
                            count<=(others=>'1');
                            cout <= '1' ;
```

```
                                end if;
                            when others =>
                                count <= (others => '-');
                                cout <= '-';
                            end case;
                        end if;
                end if;
            end process;
            sum<= count;
end architecture;
```

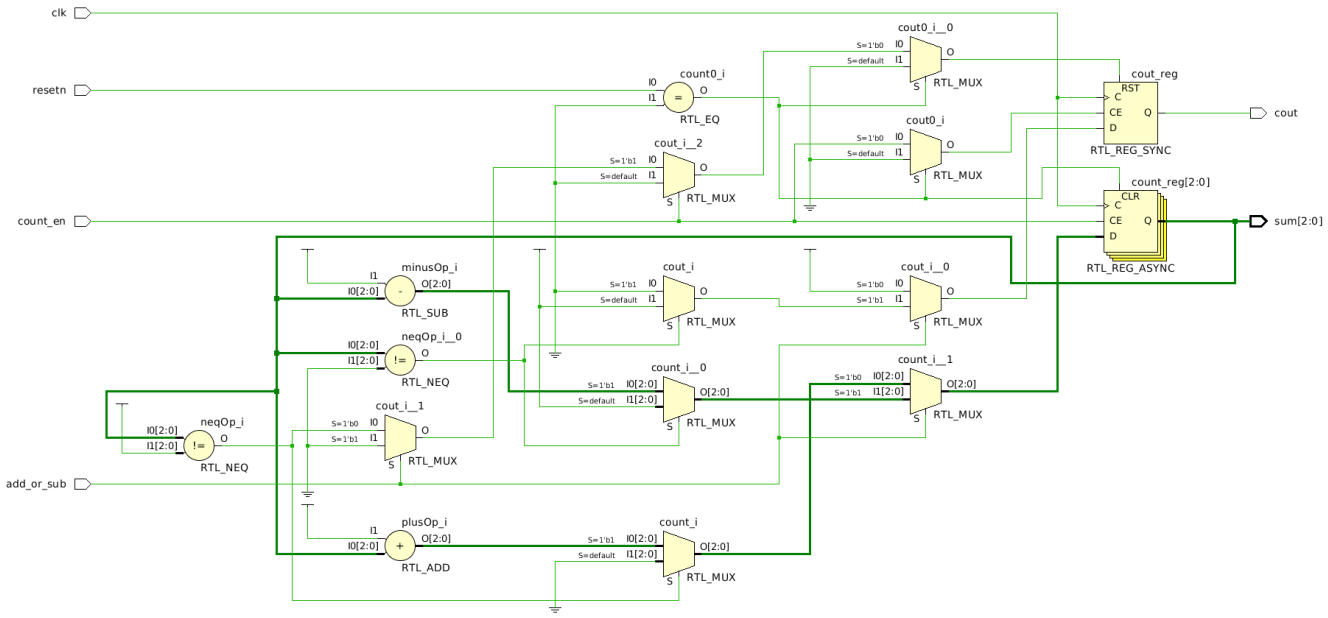
Το σχετικό testbench είναι το εξής:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;



entity counter_up_down_3bit is
port(
    clk, resetn , count_en : in std_logic;
    add_or_sub : in std_logic;
    sum : out std_logic_vector(2 downto 0);
    cout : out std_logic
    );
end entity;


architecture behavioral_arch of counter_up_down_3bit is
    signal count : std_logic_vector(2 downto 0);
    begin
        process(clk, resetn)
```

```
begin
    if resetn='0' then
    -- Ασύγχρονος μηδενισμός
        count <= (others=>'0');
    elsif clk'event and clk='1' then


            if count_en = '1' then
            -- Μέτρηση μόνο αν count_en=’1’
                case add_or_sub is
                when '0' =>
                    if count/=7 then
                        -- Αυξάνουμε το μετρητή μόνο αν
                        -- δεν είναι 7
                            count <= count+1;
                            cout <= '0' ;
                        else
                            -- Αλλιώς τον μηδενίζουμε
                            count<=(others=>'0');
                            cout <= '1' ;
                        end if;
                when '1'=>
                    if count/=0 then
                        -- Μειώνουμε το μετρητή μόνο αν
                        -- δεν είναι 0
                            count <= count-1;
                            cout <= '0' ;
                        else
                            -- Αλλιώς τον κάνουμε 7
                            count<=(others=>'1');
                            cout <= '1' ;
```

```
                    end if;
                when others =>
                    count <= (others => '-');
                    cout <= '-';
                end case;
            end if;
        end if;
    end process;
    sum<= count;
end architecture;
```

Το RTL σχηματικό του κυκλώματος που προκύπτει :



Ζητούμενο 2)

Η περιγραφή της οντότητας ένος up counter των 3 bits με παράλληλη είσοδο modulo (όριο μέτρησης) των 3 bits και της αρχιτεκτονικής του σε behavioral VHDL είναι η εξής :

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;



entity module_counter_3bit is
```

```
port(
    clk, resetn, count_en : in std_logic;
    modulo :in std_logic_vector(2 downto 0);
    sum : out std_logic_vector(2 downto 0);
    cout : out std_logic
    );
end entity;


    architecture behavioral_arch of module_counter_3bit is
        signal count : std_logic_vector(2 downto 0);
            begin


                process(clk, resetn)
                    begin
                        if resetn='0' then
                        -- Ασύγχρονος μηδενισμός
                            count <= (others=>'0');
                        elsif clk'event and clk='1' then
                            if count_en = '1' then
                            -- Μέτρηση μόνο αν count_en=’1’
                                 if count /= modulo then
                                 -- Αυξάνουμε το μετρητή μόνο αν
                                 -- δεν είναι ίσο με το όριο που έχουμε βάλει
                                    count <= count+1;
                                    cout <= '0';
                                else
                                    -- Αλλιώς τον μηδενίζουμε
                                    count<=(others=>'0');
                                    cout <= '1';
                                end if;
```

```
                        end if;
                    end if;
                end process;
            sum<= count;


end architecture;
```

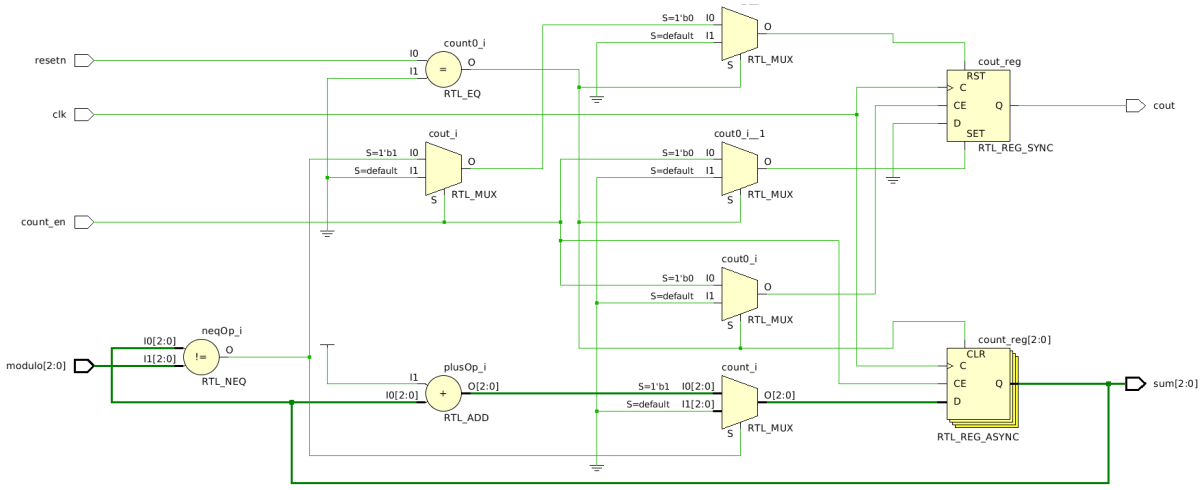
Το σχετικό testbench είναι το εξής:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;



entity counter_up_down_3bit_tb is
end entity;

architecture bench of counter_up_down_3bit_tb is


    component counter_up_down_3bit is
        port(
            clk, resetn , count_en : in std_logic;
            add_or_sub : in std_logic;
            sum : out std_logic_vector(2 downto 0);
            cout : out std_logic
        );
    end component;

    signal input_clk : std_logic ;
```

```
  signal input_resetn : std_logic := '0'  ;
  signal input_count_en : std_logic := '0'  ;
  signal input_add_or_sub : std_logic := '0'  ;
  signal output_sum : std_logic_vector(2 downto 0) ;
  signal output_cout  : std_logic;


  constant TIME_DELAY : time := 10 ns;
  constant CLOCK_PERIOD : time := 10 ns;
  constant CLOCK_PERIOD_2 : time := 100 ns;


begin


      module : counter_up_down_3bit


      port map (
              clk =>input_clk,
              resetn =>input_resetn,
              count_en=>input_count_en,
              add_or_sub=>input_add_or_sub,
              sum=>output_sum,
              cout=>output_cout
      );


      stimulus : process
       begin
      --input_resetn <= '0';
      --input_count_en <= '0';
      input_resetn <= '1';
```

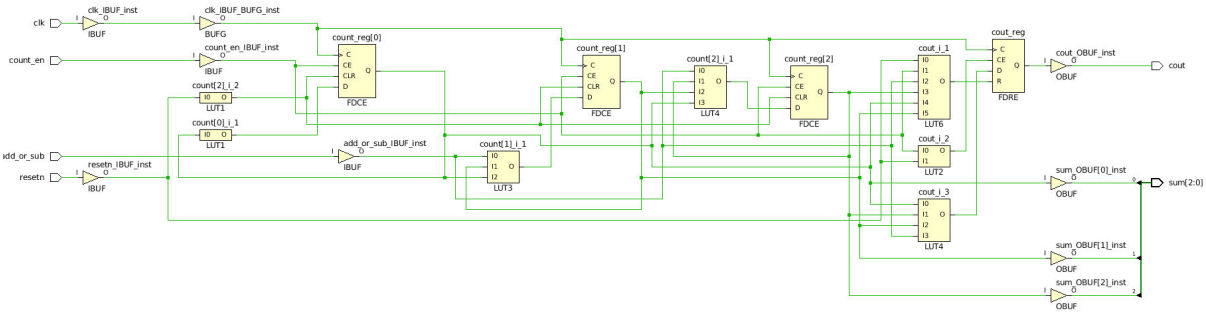
```
        input_count_en <= '1';
        wait for CLOCK_PERIOD;

        input_add_or_sub <= '0';
        wait for CLOCK_PERIOD_2;
         input_add_or_sub <= '1';
        wait for CLOCK_PERIOD_2;

        input_resetn <= '0';
        input_count_en <= '0';
        input_add_or_sub <= '0';
        wait;
        end process;

  generate_clock : process
    begin
      input_clk <= '0';
      wait for CLOCK_PERIOD/2;
      input_clk <= '1';
      wait for CLOCK_PERIOD/2;
    end process;

end architecture;
```

Το RTL σχηματικό του κυκλώματος που προκύπτει :

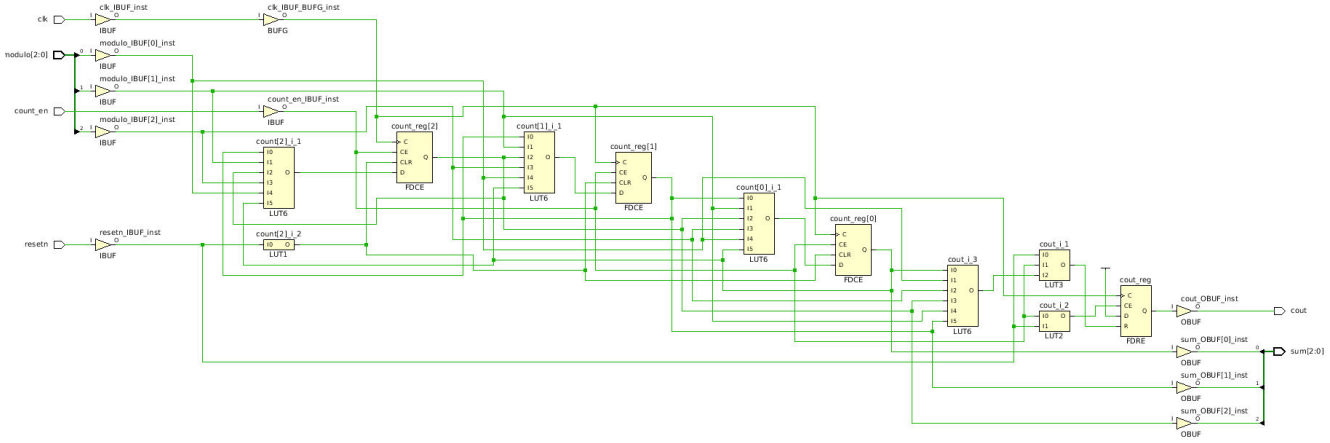


Ζητούμενο 3)

Το κύκλωμα που δίνει ο synthesizer για το ζητούμενο 1:



Το κύκλωμα που δίνει ο synthesizer για το ζητούμενο 2:


clk
clk_IBUF_inst
IBUF
clk_IBUF_BUFG_inst
BUFG
modulo[2:0]
modulo_IBUF[0]_inst
IBUF
modulo_IBUF[1]_inst
IBUF
count_en
count_en_IBUF_inst
IBUF
modulo_IBUF[2]_inst
IBUF
count[2]_i_1
LUT6
count_reg[2]
FDCE
count[1]_i_1
LUT6
count_reg[1]
FDCE
count[0]_i_1
LUT6
count_reg[0]
FDCE
cout_i_3
LUT6
cout_i_1
LUT3
cout_i_2
LUT2
cout_reg
FDRE
cout_OBUF_inst
OBUF
cout
resetn
resetn_IBUF_inst
IBUF
count[2]_i_2
LUT1
sum_OBUF[0]_inst
OBUF
sum_OBUF[1]_inst
OBUF
sum_OBUF[2]_inst
OBUF
sum[2:0]