ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Ψηφιακά VLSI
1η Εργαστηριακή Άσκηση

Ομάδα: 17
Μέλη: Κωνσταντίνος Ιωάννου (ΑΜ: 03119840)
Εμμανουήλ – Αναστάσιος Σερλής (ΑΜ: 03118125)

Ημερομηνία εργαστηριακής εξέτασης: 22/3/2023
Ημερομηνία υποβολής: 2/4/2023

Άσκηση Α2

Σε αυτό το ερώτημα το ζητούμενο είναι να περιγράψουμε σε γλώσσα περιγραφής VHDL έναν αποκωδικοποιητή 3 σε 8 (dec 3 to 8) τόσο σε αρχιτεκτονική dataflow όσο και σε behavioral.

Κώδικας για dataflow:

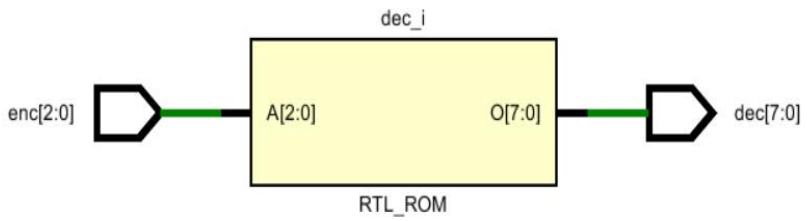
```
entity decoder_8 is
port(
 enc : in std_logic_vector(2 downto 0); -- input
 dec : out std_logic_vector(7 downto 0) -- ouput
);
end entity;


architecture dataflow_arch of decoder_8 is
begin

dec(0) <= not enc(0) and not enc(1) and not enc(2);
dec(1) <=  enc(0) and not enc(1) and not (enc(2));
dec(2) <= not enc(0) and  enc(1) and not enc(2);
dec(3) <=  enc(0) and  enc(1) and not enc(2);
dec(4) <= not enc(0) and not enc(1) and  enc(2);
dec(5) <= enc(0) and not enc(1) and  enc(2);
dec(6) <= not enc(0) and  enc(1) and   enc(2);
dec(7) <= enc(0) and  enc(1) and  enc(2);

end architecture;
```

Κώδικας για behavioral :

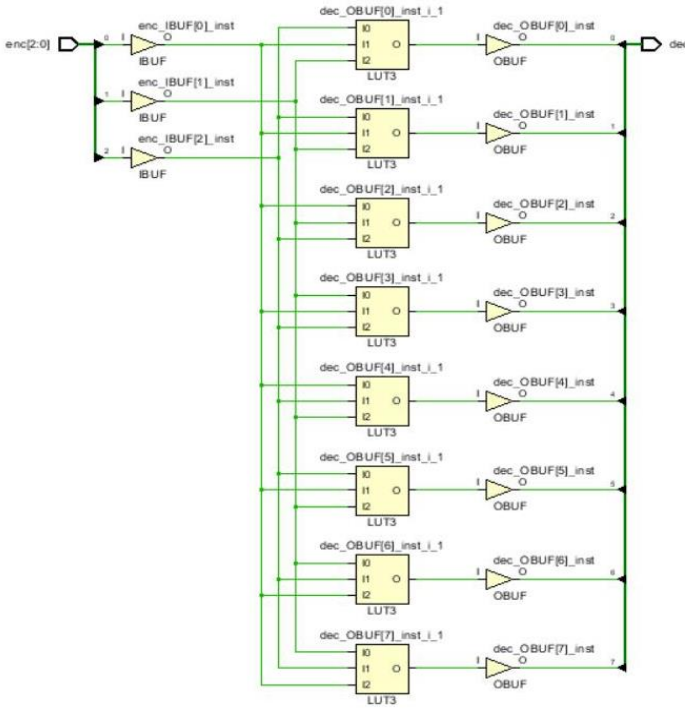
```
architecture Behavioral_arch of decoder_8 is
begin

process(enc)
 begin

 case enc is

   when  "000" =>
   dec <= "00000001";

   when  "001" =>
   dec <= "00000010";

   when  "010" =>
   dec <= "00000100";

   when  "011" =>
   dec <= "00001000";

   when  "100" =>
   dec <= "00010000";

   when  "101" =>
   dec <= "00100000";

   when  "110" =>
   dec <= "01000000";

   when  "111" =>
   dec <= "10000000";

   when others =>
   dec <= (others => '-');

   end case;
  end process;

end Behavioral_arch ;
```

Τόσο για την αρχιτεκτονική dataflow όσο και για την behavioural έχουμε τα ίδια αποτελέσματα:

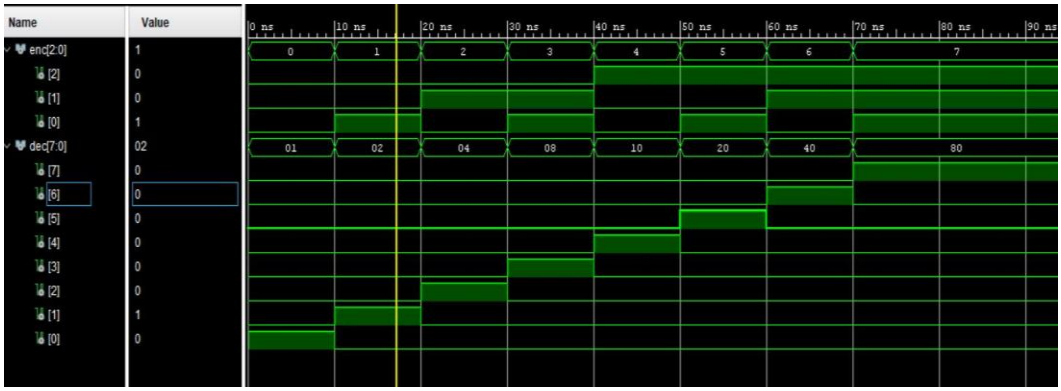
- RTL ANALYSIS



- Synthesis



- Στην συνέχεια γράφοντας ένα απλό **testbench** έχουμε στο **simulation**:



Παρατηρούμε λοιπόν από την προσομοίωση ότι έχουμε τα αναμενόμενα αποτελέσματα για τον decoder 3 σε 8.

Άσκηση Β2

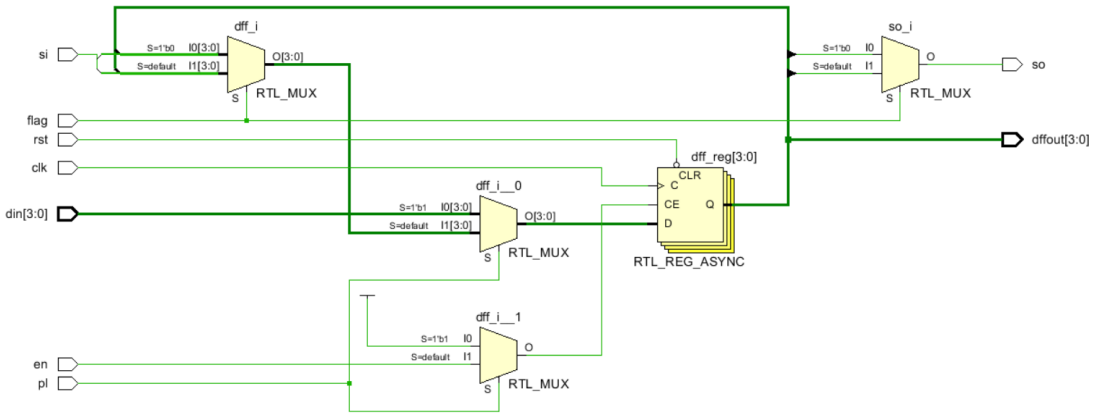
Το ζητούμενο σε αυτό το ερώτημα είναι να τροποποιήσουμε έναν καταχώρηση δεξιάς ολίσθησης 4 bits με παράλληλη φόρτωση ώστε να μπορούμε μέσω ενός flag να επιλέξουμε αν το λογικό κύκλωμα θα κάνει δεξιά ή αριστερή ολίσθηση.

Ο τροποποιημένος κώδικας λοιπόν φαίνεται παρακάτω:

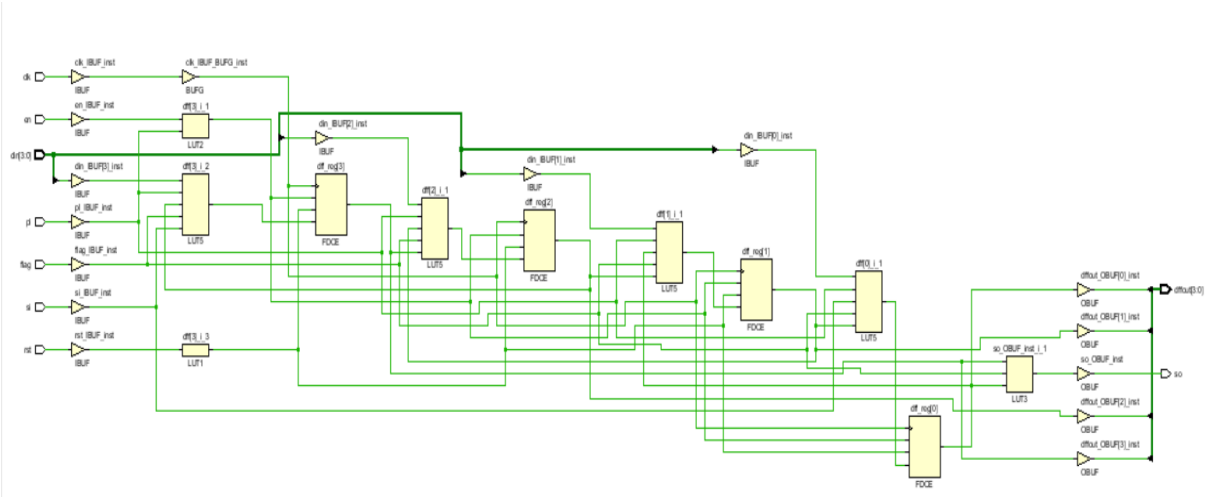
```
architecture rtl of lab01_ex02 is
signal dff: std_logic_vector(3 downto 0);
begin
edge: process (clk,rst)

begin


if rst='0' then
dff<=(others=>'0');

elsif clk'event and clk='1' then
if pl='1' then
dff<=din;

elsif en='1' then
  if flag = '0' then
  dff<=si&dff(3 downto 1) ;


  elsif flag ='1' then
  dff<=dff(2 downto 0)&si;


end if;
end if;
end if;

end process;

dffout <= dff;
so <= dff(0) WHEN flag = '0' ELSE dff(3) WHEN flag = '1';
end rtl;
```

Για flag =0 έχουμε δεξιά ολίσθηση, ενώ για flag =1 έχουμε αριστερή ολίσθηση Τέλος, όταν pl =1 φορτώνουμε την τιμή που θέλουμε να ολισθήσουμε.

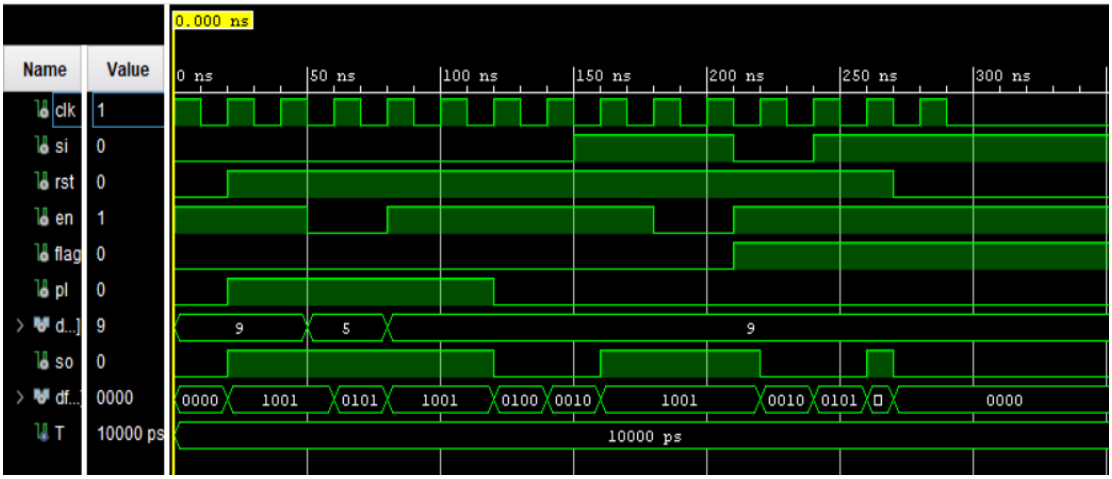
- RTL ANALYSIS



- Synthesis



- Στην συνέχεια γράφοντας ένα **testbench** λαμβάνουμε τα κάτωθι αποτελέσματα:

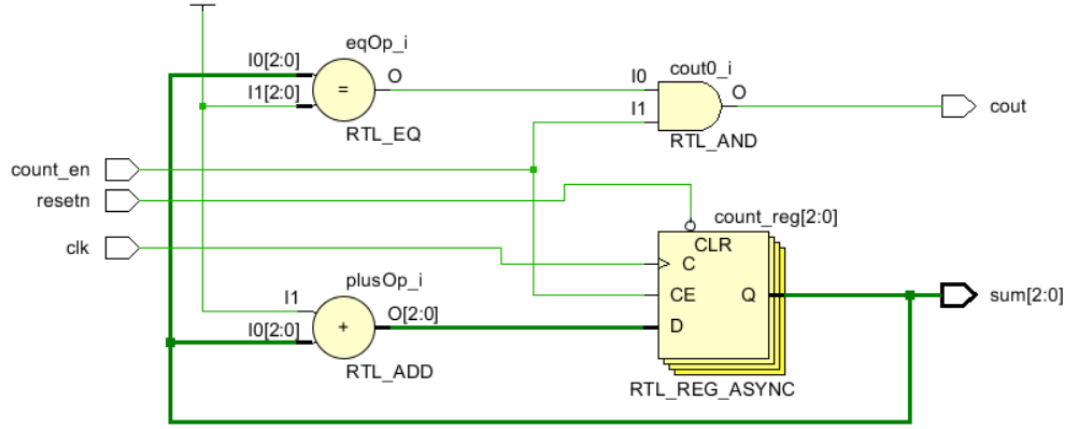


Αρχικά βλέπουμε ότι όταν τίθεται το reset στο 0 έχουμε έξοδο so = 0 ανεξαρτήτως της τιμή των υπόλοιπων σημάτων. Από την άλλη, για reset =1 στις θετικές ακμές του ρολογιού είτε κάνουμε παράλληλη φόρτωση της τιμής όταν pl =1, αλλιώς κάνουμε ολίσθηση δεξιά ή αριστερή ανάλογα με την τιμή του flag . Συνεπώς το λογικό κύκλωμα λειτουργεί όπως αναμένουμε.

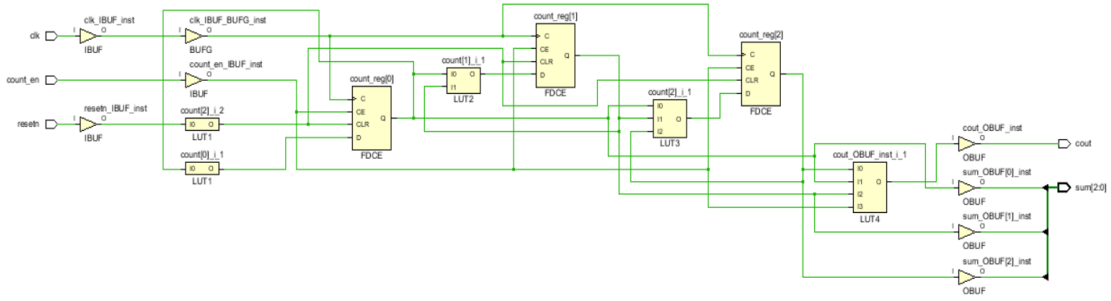
Άσκηση Β3

Σε αυτό το ερώτημα μας δίνεται **μετρητής 3 bit με είσοδο ενεργοποίησης και κρατούμενο εξόδου** όπως φαίνεται παρακάτω:

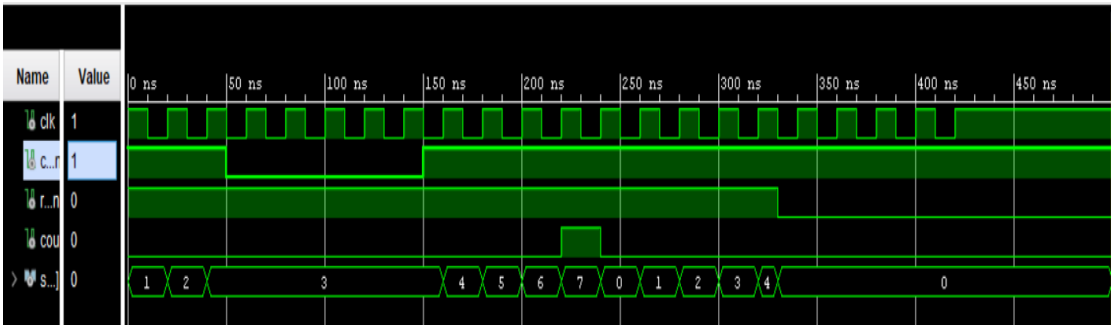
- RTL ANALYSIS



- Synthesis



- Simulation

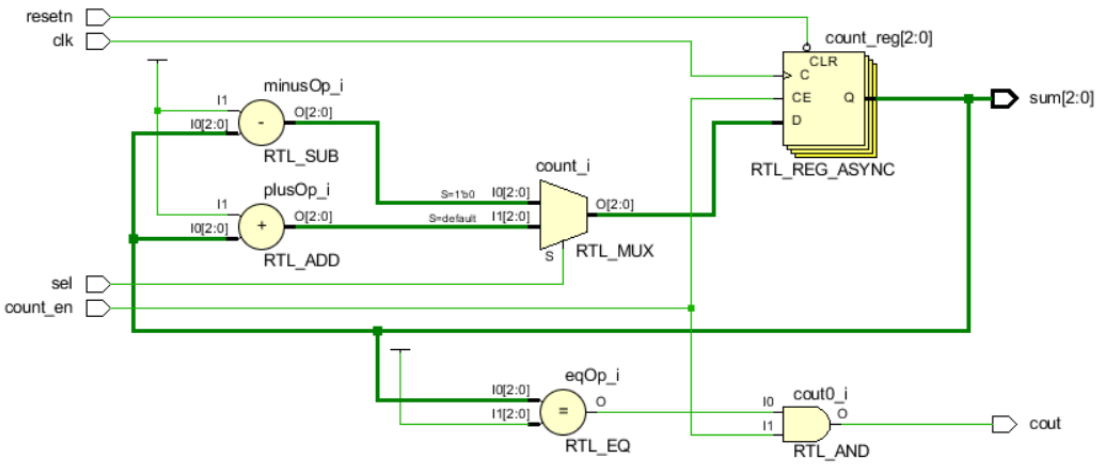


1) Στην συνέχεια με βάση τον κώδικα για τον παραπάνω απλό μετρητή περιγράφουμε έναν **up/down μετρητή**.

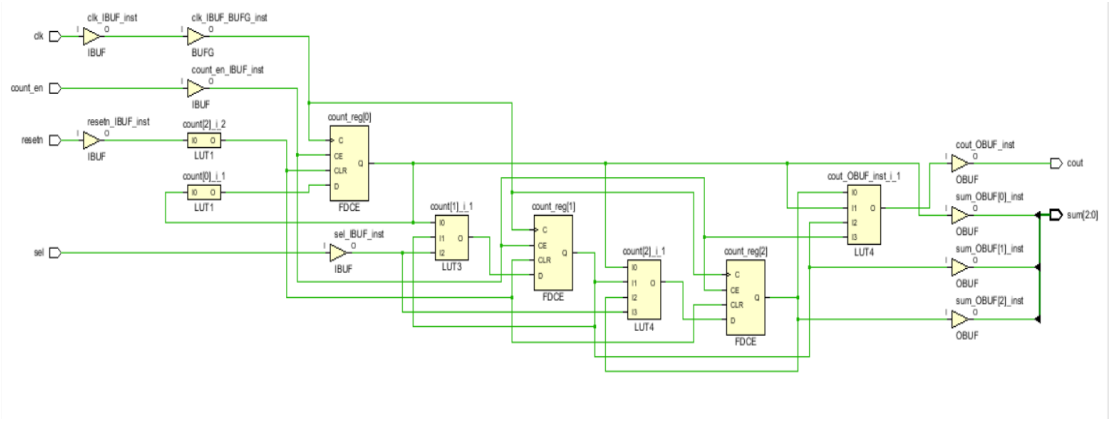
Κώδικας για την αρχιτεκτονική :

```
architecture Behavioral of lab01_ex03_counter_updown is
signal count: std_logic_vector(2 downto 0);
begin
    process(clk, resetn)
    begin
        if resetn='0' then
            count <= (others=>'0'); -- Code for reset
        elsif clk'event and clk='1' then --clock rising edge
            if count_en='1' then
                if sel='0' then  --down_count with zero
                    count<=count-1;
                elsif sel='1' then  --up_count with one
                    count<=count+1;
                else
                    count <= (others=>'0');
                end if;
            end if;
        end if;
    end process;
    sum <= count; -- Output signals
    cout <= '1' when count=7 and count_en='1' else '0';
end Behavioral;
```

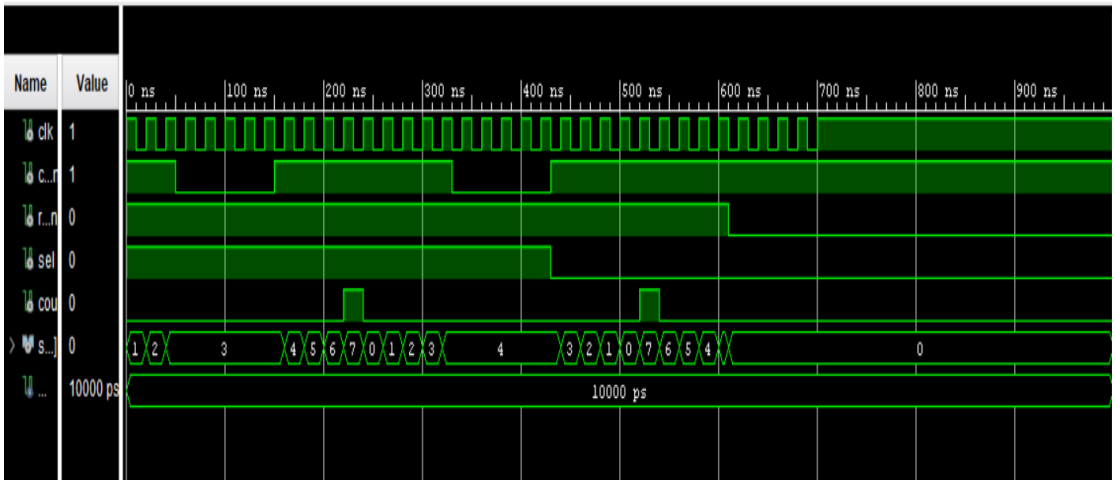
- RTL ANALYSIS



- Synthesis



- Στην συνέχεια γράφοντας ένα **testbench** έχουμε στο **simulation**



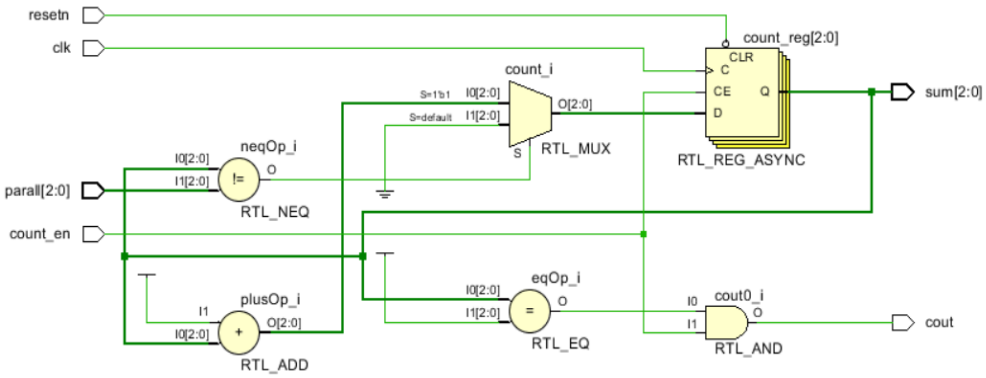
Αρχικά για sel =1 έχουμε up counter ενώ μετά για sel = 0 έχουμε down counter. Επιπλέον, παρατηρούμε ότι όταν η έξοδος sum φτάνει στην ανώτατη τιμή του (sum=7), τότε τίθεται η μεταβλητή cout ίση με 1.

2) Τέλος βασιζόμενοι στον counter που μας δόθηκε περιγράφουμε έναν **up counter των 3 bits με παράλληλη είσοδο modulo** (όριο μέτρησης) των 3bits.

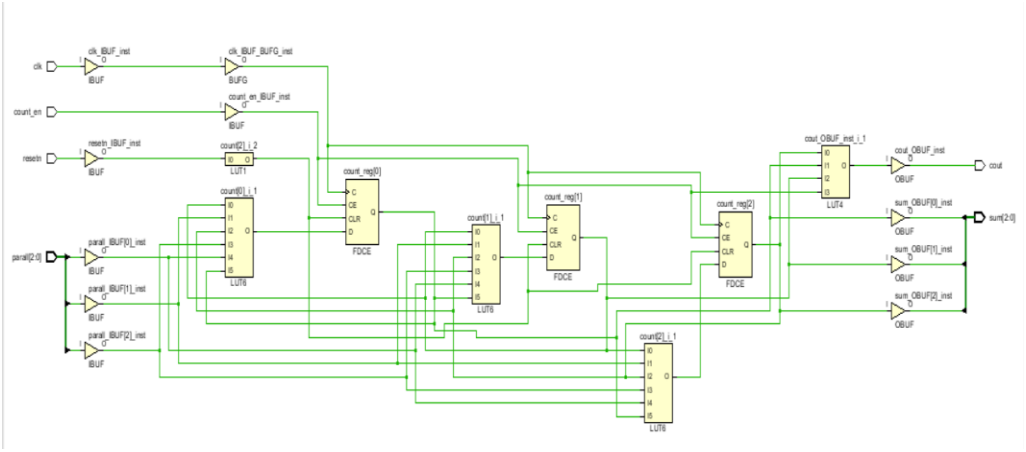
Κώδικας για την αρχιτεκτονική :

```
architecture Behavioral of lab01_ex03_input is
signal count: std_logic_vector(2 downto 0);
begin
    process(clk, resetn)
    begin
        if resetn='0' then
            count <= (others=>'0'); -- Code for reset
        elsif clk'event and clk='1' then --clock rising edge
            if count_en='1' then  -- Count up when count_en=1
               if count/=parall then
                    count<=count+1;
               else  --set to zero when modulo input value is reached
                    count<=(others=>'0');
               end if;
            end if;
        end if;
    end process;
    sum <= count; -- Output signals
    cout <= '1' when count=7 and count_en='1' else '0';
end Behavioral;
```

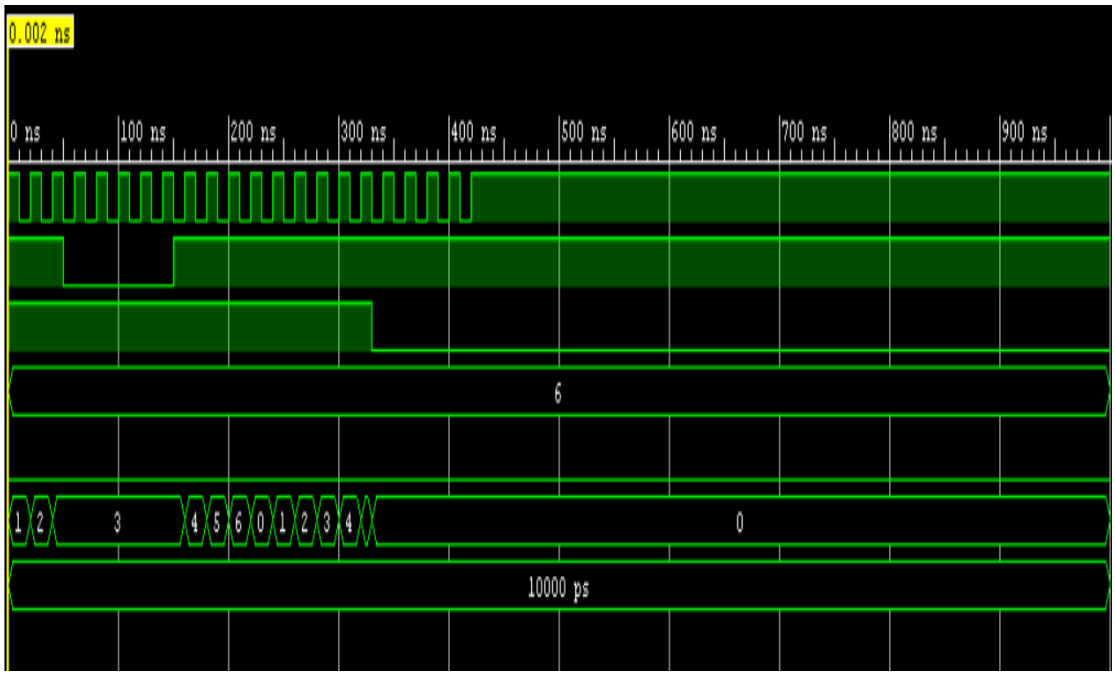
- RTL ANALYSIS



- Synthesis

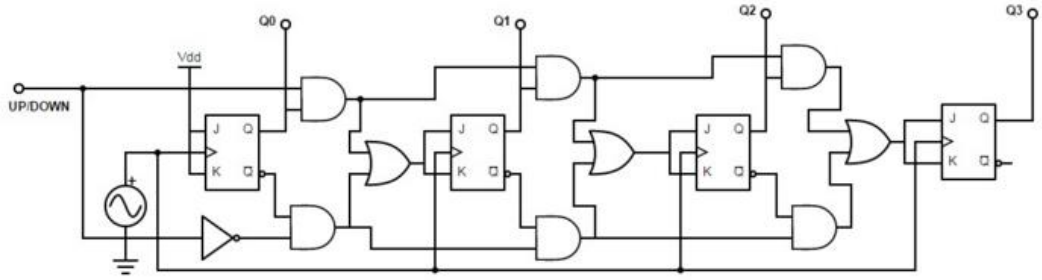


- Στην συνέχεια γράφοντας ένα **testbench** έχουμε στο **simulation**



Παρατηρούμε εκ νέου την ορθότητα της λειτουργίας του input counter, μιας και η τιμή της εξόδου sout φτάνει μέχρι και την τιμή της μεταβλητής input η οποία στο άνωθι simulation έχει τεθεί στο 6. Τέλος, μόλις το reset τεθεί στο μηδέν, η τιμή του sout γίνεται μηδέν (ασύγχρονα με το ρολόι).

3) Όσον αφορά την ύπαρξη απλούστερης υλοποίησης, από synthesis schematic δεν υπάρχει κάποια προφανής βελτίωσή της, όπως θα υπήρχε - για παράδειγμα - στην περίπτωση ύπαρξης πολλαπλών LUTs με κοινές εισόδους. Ωστόσο, μια αισθητά απλοποιημένη μορφή του μετρητή up-down είναι η κάτωθι, η οποία περιλαμβάνει μονάχα λογικές πύλες και JK Flip-Flops:


Q0
Q1
Q2
Q3
Vdd
UP/DOWN
J
Q
K
J
Q
K
J
Q
K
J
Q
K