Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Ψηφιακά Συστήματα **VLSI**

1η Εργαστηριακή Άσκηση

Αλικάρης Αντώνιος Δημήτριος (03118062)

A2: Δυαδικός αποκωδικοποιητής 3 σε 8

Παρακάτω παρουσιάζονται οι κώδικες σε dataflow και behavioral περιγραφή, καθώς και ο κώδικας του testbench.

Κώδικας dataflow

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;

ENTITY dec3to8 IS
    PORT (
        enc : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
        dec : OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END dec3to8;

ARCHITECTURE dataflow OF dec3to8 IS
BEGIN
    WITH enc SELECT
        dec <=  "00000001" WHEN "000",
                "00000010" WHEN "001",
                "00000100" WHEN "010",
                "00001000" WHEN "011",
                "00010000" WHEN "100",
                "00100000" WHEN "101",
                "01000000" WHEN "110",
                "10000000" WHEN OTHERS;
END dataflow;
```

Κώδικας behavioral

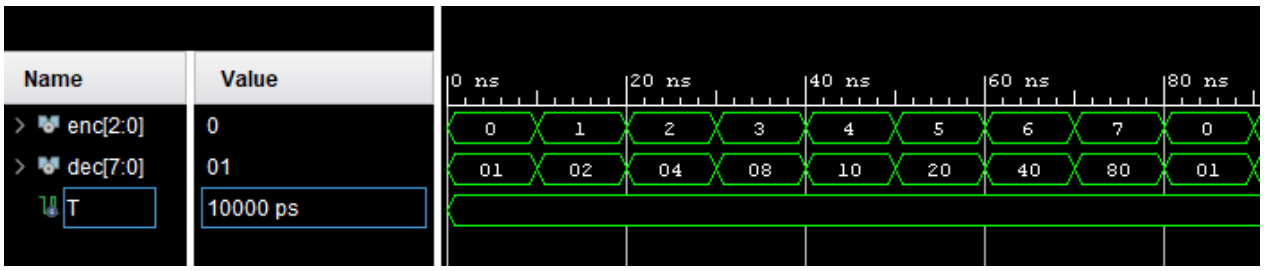
```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;

ENTITY dec3to8 IS
    PORT (
        enc : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
        dec : OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END dec3to8;

ARCHITECTURE behavior OF dec3to8 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 OTHERS => dec <= "10000000";

        END CASE;
    END PROCESS;
END behavior;
```

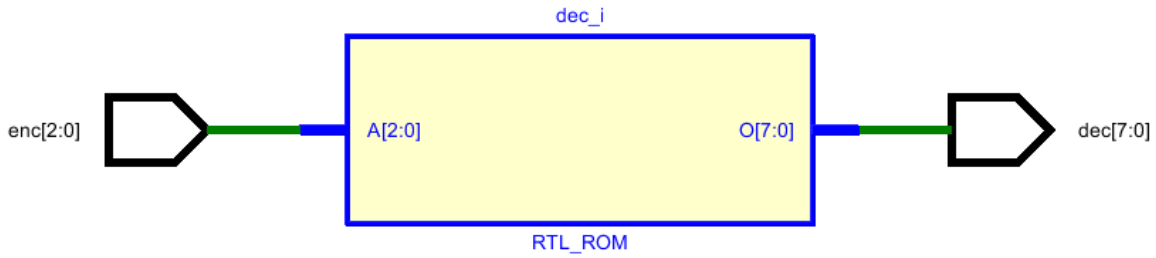
Κώδικας testbench

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;
USE IEEE.std_logic_unsigned.ALL;

ENTITY dec3to8_tb IS
END ENTITY;

ARCHITECTURE bench OF dec3to8_tb IS
    -- Define component
    COMPONENT dec3to8 IS
        PORT (
            enc : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
            dec : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
        );
    END COMPONENT;
    -- Test signals
    SIGNAL enc : STD_LOGIC_VECTOR(2 DOWNTO 0) := "000";
    SIGNAL dec : STD_LOGIC_VECTOR(7 DOWNTO 0);
    CONSTANT T : TIME := 10 ns;
BEGIN
    uut : dec3to8
    PORT MAP(
        enc => enc,
        dec => dec
    );

    stimulus : PROCESS
    BEGIN
        WAIT FOR T;
        enc <= enc + 1;
    END PROCESS;
END ARCHITECTURE;
```

Έξοδος προσομοίωσης



RTL Ανάλυση



B2: Καταχωρητής ολίσθησης των 4 bits με παράλληλη φόρτωση

Παρακάτω παρουσιάζονται ο κώδικας σε behavioral περιγραφή, καθώς και ο κώδικας του testbench. Για την υλοποίησή του, προστέθηκε στον υπάρχοντα κώδικα ένα επιπλέον σήμα sh που καθορίζει αν θα κάνει δεξιά (sh = 0) ή αριστερή (sh = 1) ολίσθηση. Στο testbench για όλους τους αριθμούς του din, αρχικά κάνουμε παράλληλη φόρτωση σε λειτουργία right shift με είσοδο 0 για 4 περιόδους και έπειτα με είσοδο 1 για 8.Στην συνέχεια ξανακάνουμε παράλληλη φόρτωση σε λειτουργία left shift με είσοδο 1 για 4 περιόδους και είσοδο 0 για άλλες 8.

Κύριος Κώδικας

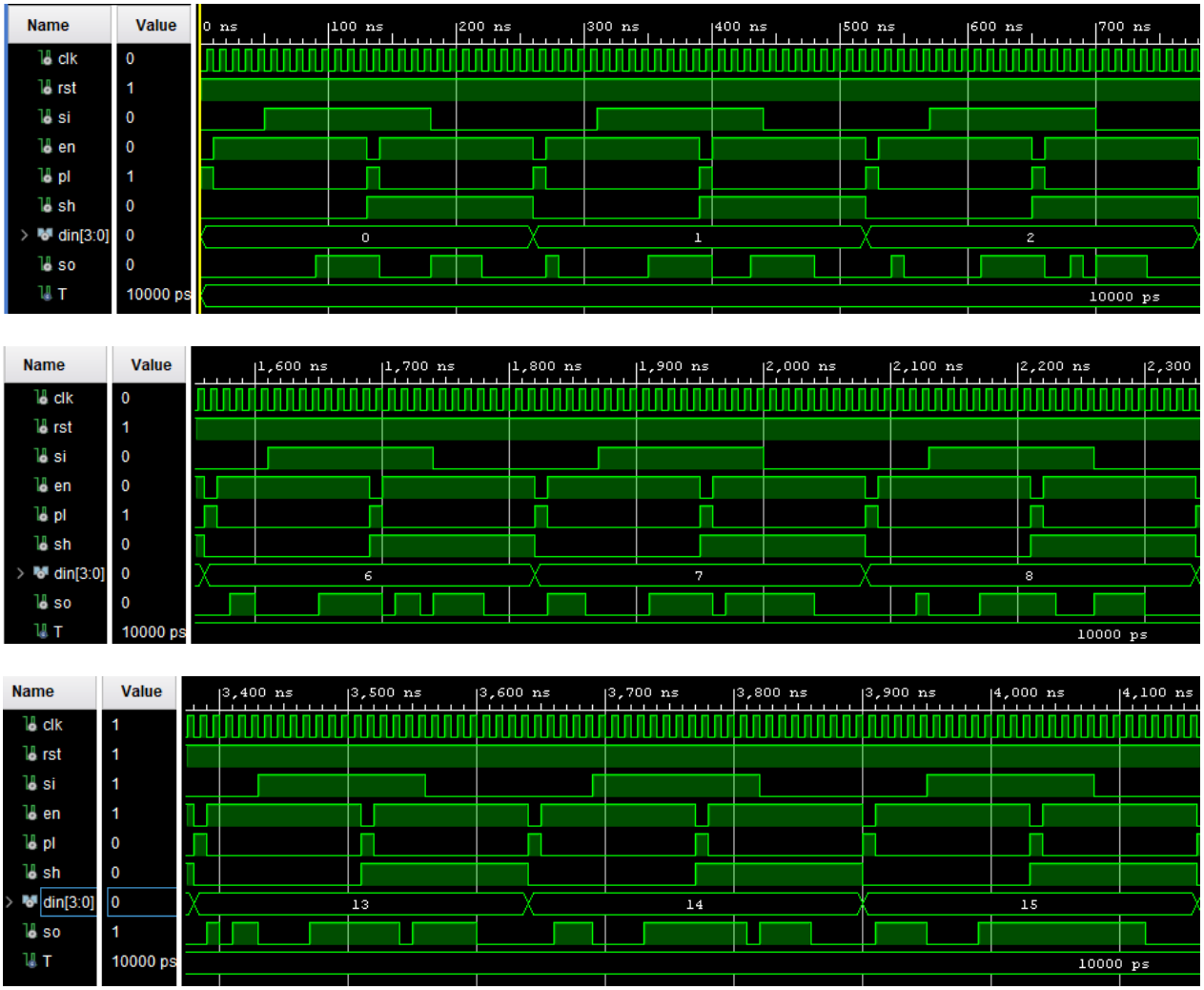
```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
ENTITY lrshift_reg3 IS
    PORT (
        clk, rst, si, en, pl, sh : IN STD_LOGIC; --sh = '0' means right shift
        din : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
        so : OUT STD_LOGIC
    );
END lrshift_reg3;
ARCHITECTURE behavioral OF lrshift_reg3 IS
    SIGNAL dff : STD_LOGIC_VECTOR(3 DOWNTO 0) := "0000";
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 sh = '0' THEN
                    dff <= si & dff(3 DOWNTO 1);
                ELSE
                    dff <= dff(2 DOWNTO 0) & si;
                END IF;

            END IF;
        END IF;

        CASE(sh) IS

            WHEN '0' => so <= dff(0);
            WHEN OTHERS => so <= dff(3);
        END CASE;
    END PROCESS;
END behavioral;
```

Κώδικας Testbench

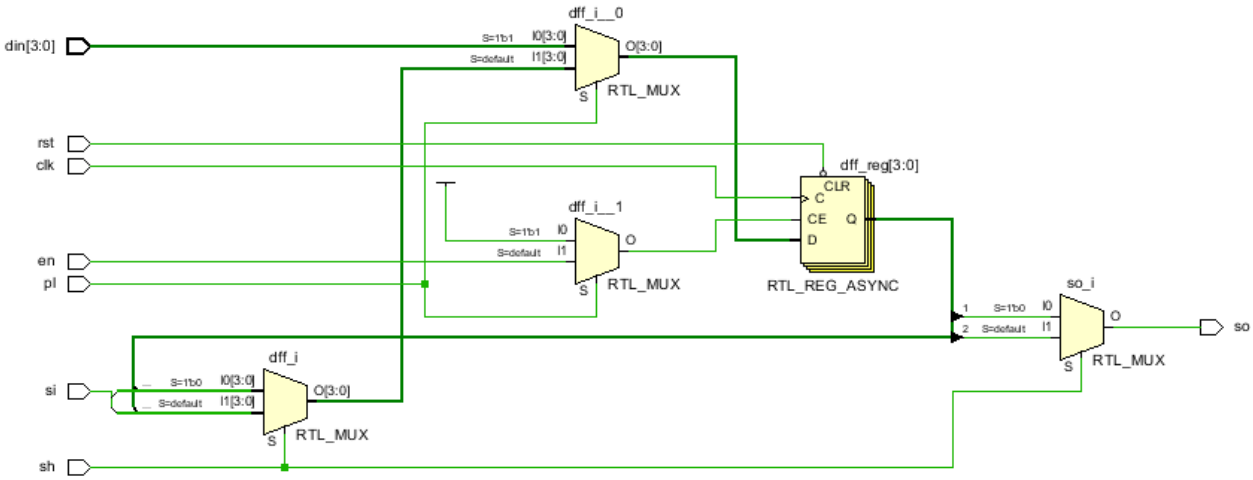
```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;
USE IEEE.std_logic_unsigned.ALL;

ENTITY lrshift_reg3_tb IS
END ENTITY;

ARCHITECTURE bench OF lrshift_reg3_tb IS

    -- Define component
    COMPONENT lrshift_reg3 IS
        PORT (
            clk, rst, si, en, pl, sh  : IN STD_LOGIC;
            din : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
            so : OUT STD_LOGIC
        );
    END COMPONENT;

    -- Test signals
    SIGNAL clk, rst, si, en, pl, sh : STD_LOGIC;
    SIGNAL din : STD_LOGIC_VECTOR(3 DOWNTO 0) := "0000";
    SIGNAL so : STD_LOGIC;

    CONSTANT T : TIME := 10 ns;

BEGIN

    uut : lrshift_reg3
    PORT MAP(
        clk => clk,
        rst => rst,
        si => si,
        en => en,
        pl => pl,
        sh => sh,
        din => din,
        so => so
    );

    stimuli : PROCESS
    BEGIN
        -- read din, so = LSB

        en <= '0';
        si <= '0';
```

```
47          rst <= '1';
48          sh <= '0';
49          pl <= '1';
50          WAIT FOR T;
51
52          -- s0 = reverse din with input 0
53          en <= '1';
54          pl <= '0';
55          WAIT FOR 4 * T;
56
57          -- input = 1
58          si <= '1';
59          WAIT FOR 8 * T;
60
61
62          --read din, s0 = MSB
63          si <= '1';
64          sh <= '1';
65          en <= '0';
66          pl <= '1';
67          WAIT FOR T;
68
69          -- s0 = din with input 1
70          en <= '1';
71          pl <= '0';
72          WAIT FOR 4 * T;
73
74          --input = 0
75          si <= '0';
76          WAIT FOR 8 * T;
77
78          din <= din + 1;
79
80
81      END PROCESS;
82
83      generate_clock : PROCESS
84      BEGIN
85          clk <= '0';
86          WAIT FOR T/2;
87          clk <= '1';
88          WAIT FOR T/2;
89      END PROCESS;
90  END ARCHITECTURE;
```

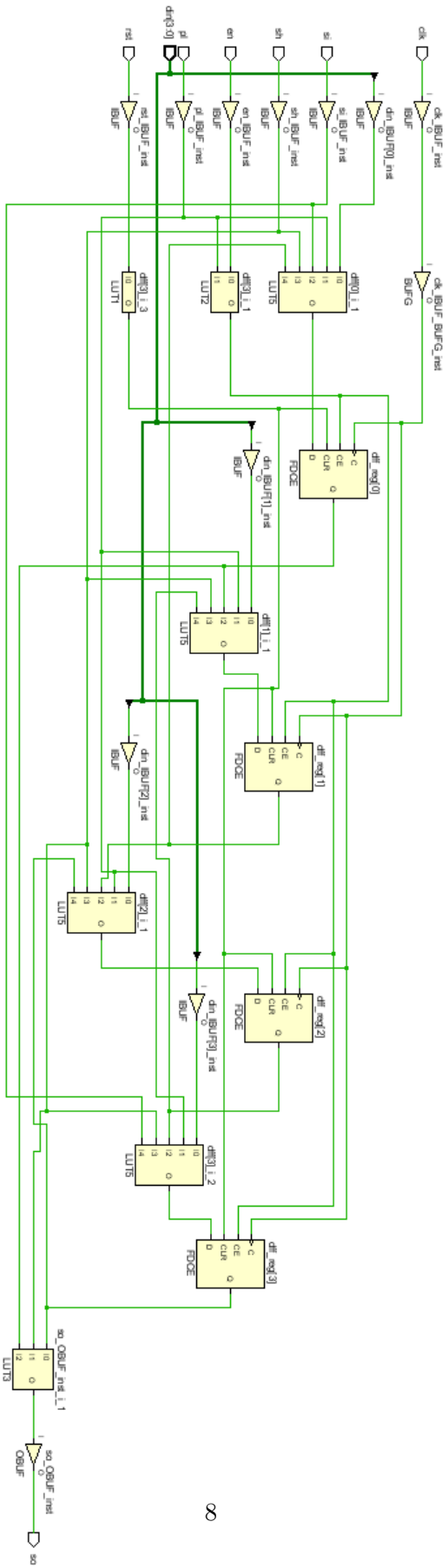
Έξοδος προσομοίωσης



RTL Ανάλυση



Σχηματικό Σύνθεσης



Στην σύνθεση υπάρχουν διαφορές από το σχήμα της εκφώνησης, καθώς έχουν προστεθέι 4 στοιχεία από τα οποία περνάει το σήμα sh προκειμένου να καταλήξει στους 4 καταχωρητές.

B3: Μετρητής 3 bit με είσοδο ενεργοποίησης και κρατούμενο εξόδου

1.

Παρακάτω παρουσιάζονται ο κύριος κώδικας καθώς και ο κώδικας του testbench. Η λειτουργία του βασίζεται στον μετρητή του παραδείγματος, με αρχιτεκτονική limit (έλεγχος πριν την υπερχείλιση). Προστέθηκε μία επιπλέον είσοδος ελέγχου, η up, η οποία όταν είναι 1 έχουμε μέτρηση προς τα πάνω και όταν είναι 0 έχουμε μέτρηση προς τα κάτω. Ο έλεγχος για υπερχείλιση επεκτάθηκε και σε έλεγχο μηδενισμού (κατά τη μέτρηση προς τα κάτω).

Κύριος Κώδικας

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_unsigned.ALL;

ENTITY updown_counter_3bit IS
    PORT (
        clk, reset_bar, count_en, up : IN STD_LOGIC;
        sum : OUT STD_LOGIC_VECTOR(2 DOWNTO 0);
        cout : OUT STD_LOGIC
    );
END;

-- LIMIT
ARCHITECTURE rtl_limit OF updown_counter_3bit IS
    SIGNAL count : STD_LOGIC_VECTOR(2 DOWNTO 0) := "000";

BEGIN
    PROCESS (clk, reset_bar)
    BEGIN
        IF reset_bar = '0' THEN
            -- Asynchronous reset
            count <= (OTHERS => '0');
        ELSIF rising_edge(clk) THEN
            IF count_en = '1' THEN
                -- Count only if count_en = 1
                IF up = '1' THEN
                    -- If up = 1 count up
                    IF count /= 7 THEN
                        -- Increment counter if it hasn't reached 7
                        count <= count + 1;
                    ELSE
                        -- else clear all
                        count <= (OTHERS => '0');
                    END IF;
                ELSIF up = '0' THEN
```

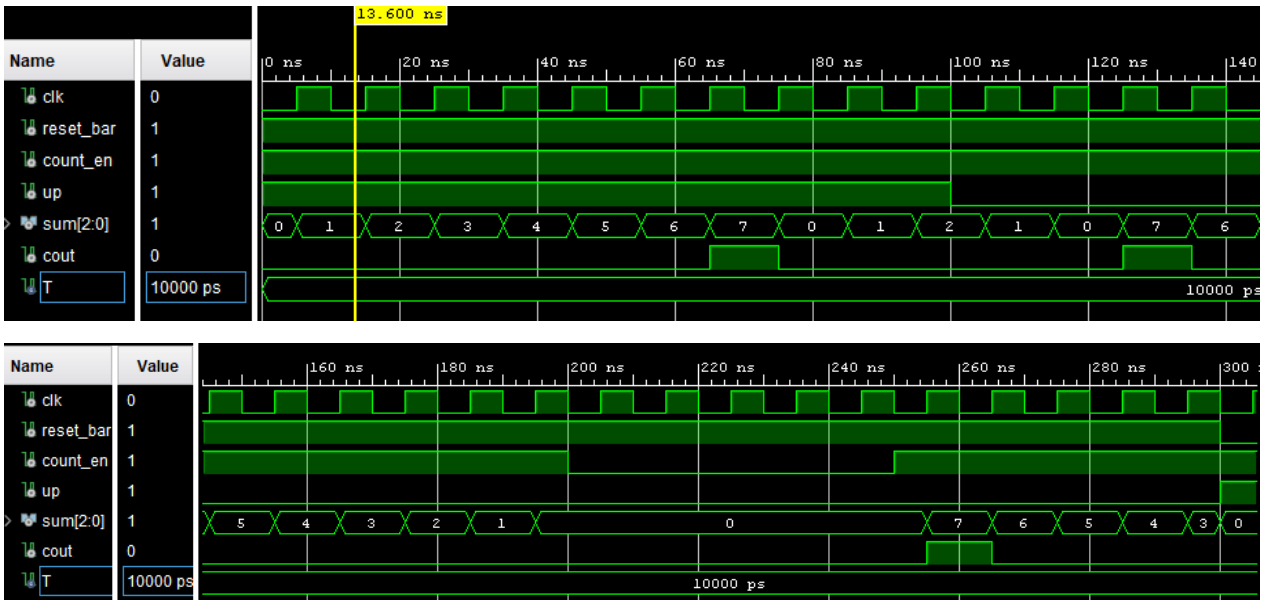
```
                    -- Count down
                    IF count /= 0 THEN
                        -- Decrement counter if it hasn't reached 0
                        count <= count - 1;
                    ELSE
                        -- else set all
                        count <= (OTHERS => '1');
                    END IF;
                END IF;
            END IF;
        END IF;
    END PROCESS;
    sum <= count;
    cout <= '1' WHEN count = 7 AND count_en = '1' ELSE
        '0';
END;
```

Κώδικας testbench

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;

ENTITY updown_counter_3bit_tb IS
END ENTITY;

ARCHITECTURE bench OF updown_counter_3bit_tb IS

    -- Define component
    COMPONENT updown_counter_3bit IS
        PORT (
            clk, reset_bar, count_en, up : IN STD_LOGIC;
            sum : OUT STD_LOGIC_VECTOR(2 DOWNTO 0);
            cout : OUT STD_LOGIC
        );
    END COMPONENT;

    -- Test signals
    SIGNAL clk, reset_bar, count_en, up : STD_LOGIC;
    SIGNAL sum : STD_LOGIC_VECTOR(2 DOWNTO 0);
    SIGNAL cout : STD_LOGIC;

    CONSTANT T : TIME := 10 ns;

BEGIN

    uut : updown_counter_3bit
```

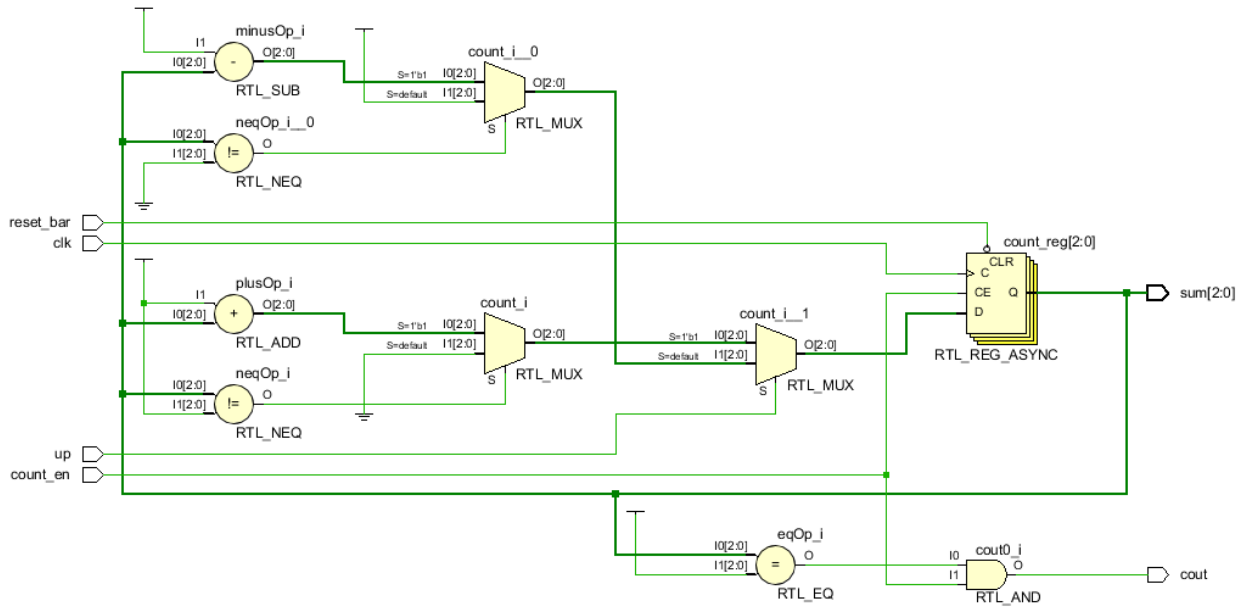
```
    PORT MAP(
        clk => clk,
        reset_bar => reset_bar,
        count_en => count_en,
        up => up,
        sum => sum,
        cout => cout
    );

    stimulus : PROCESS
    BEGIN
        -- Start by counting up
        reset_bar <= '1';
        count_en <= '1';
        up <= '1';
        WAIT FOR 10 * T;

        -- Then count down
        up <= '0';
        WAIT FOR 10 * T;

        -- Stop counting
        count_en <= '0';
        WAIT FOR 5 * T;

        -- Restart
        count_en <= '1';
        WAIT FOR 5 * T;

        -- Reset and count up
        reset_bar <= '0';
        up <= '1';
        WAIT FOR 5 * T;
        WAIT;
    END PROCESS;

    generate_clock : PROCESS
    BEGIN
        clk <= '0';
        WAIT FOR T/2;
        clk <= '1';
        WAIT FOR T/2;
    END PROCESS;

END ARCHITECTURE;
```

Έξοδος προσομοίωσης



Η λειτουργία του είναι αρκετά απλή: παρουσιάζεται η μέτρηση άνω, η μέτρηση κάτω, η παύση όταν το enable είναι 0 και το ασύγχρονο reset.

RTL Ανάλυση



2.

Παρακάτω παρουσιάζονται ο κύριος κώδικας καθώς και ο κώδικας του testbench. Η λειτουργία του βασίζεται στον μετρητή του παραδείγματος, με αρχιτεκτονική limit (έλεγχος πριν την υπερχείλιση). Προστέθηκε μία επιπλέον είσοδος τριών bit, η modulo, η οποία καθορίζει το άνω όριο μέτρησης. Σε κάθε ακμή του ρολογιού γίνεται έλεγχος αν η τιμή του μετρητή έφτασε στο modulo-1 και αν ναι, τότε μηδενίζεται.

Κύριος Κώδικας

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_unsigned.ALL;

ENTITY mod_counter_3bit IS
    PORT (
        clk, reset_bar, 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;

-- LIMIT
ARCHITECTURE rtl_limit OF mod_counter_3bit IS
    SIGNAL count : STD_LOGIC_VECTOR(2 DOWNTO 0) := "000";

BEGIN
    PROCESS (clk, reset_bar)
    BEGIN
        IF reset_bar = '0' THEN
            -- Asynchronous reset
            count <= (OTHERS => '0');
        ELSIF rising_edge(clk) THEN
            IF count_en = '1' THEN
                IF count /= 7 AND count < modulo - 1 THEN
                    count <= count + 1;
                ELSE
                    count <= (OTHERS => '0');
                END IF;
            END IF;
        END IF;
    END PROCESS;
    sum <= count;
    cout <= '1' WHEN count = 7 AND count_en = '1' ELSE
        '0';
END;
```

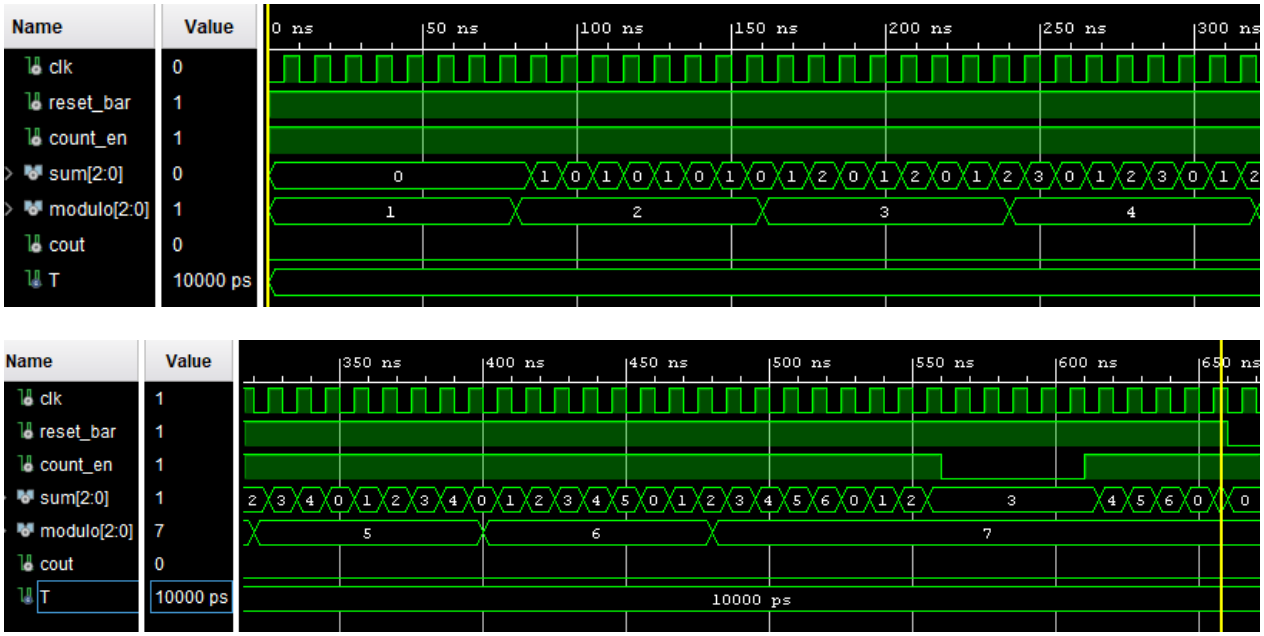
Κώδικας testbench

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;

ENTITY mod_counter_3bit_tb IS
END ENTITY;

```

```
8  ARCHITECTURE bench OF mod_counter_3bit_tb IS
9
10     -- Define component
11     COMPONENT mod_counter_3bit IS
12         PORT (
13             clk, reset_bar, count_en : IN STD_LOGIC;
14             modulo : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
15             sum : OUT STD_LOGIC_VECTOR(2 DOWNTO 0);
16             cout : OUT STD_LOGIC
17         );
18     END COMPONENT;
19
20     -- Test signals
21     SIGNAL clk, reset_bar, count_en : STD_LOGIC;
22     SIGNAL sum, modulo : STD_LOGIC_VECTOR(2 DOWNTO 0);
23     SIGNAL cout : STD_LOGIC;
24
25     CONSTANT T : TIME := 10 ns;
26
27 BEGIN
28
29     uut : mod_counter_3bit
30     PORT MAP(
31         clk => clk,
32         reset_bar => reset_bar,
33         count_en => count_en,
34         modulo => modulo,
35         sum => sum,
36         cout => cout
37     );
38
39     stimulus : PROCESS
40     BEGIN
41         -- Start by enabling counting
42         reset_bar <= '1';
43         count_en <= '1';
44
45         -- Check all possible modulos
46         FOR i IN 1 TO 7 LOOP
47             modulo <= STD_LOGIC_VECTOR(to_unsigned(i, 3));
48             WAIT FOR 8 * T;
49         END LOOP;
50
51         -- Pause
52         count_en <= '0';
53         WAIT FOR 5 * T;
54
55         -- Restart
```

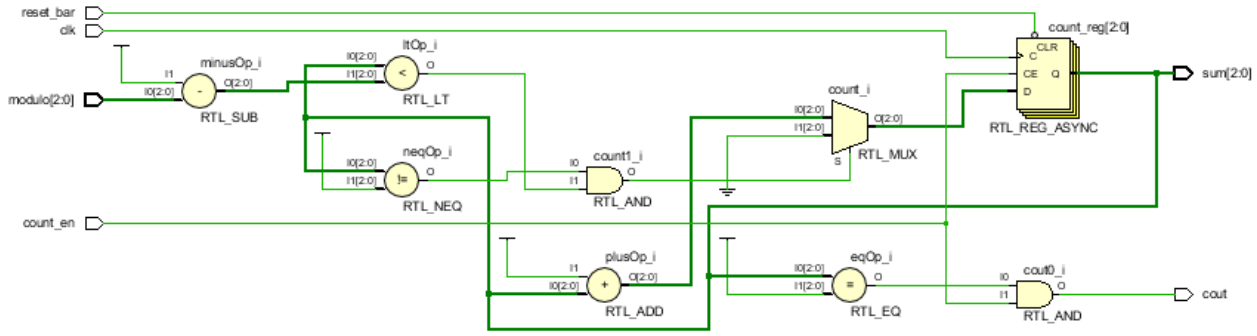
```
56          count_en <= '1';
57          WAIT FOR 5 * T;
58
59          -- Reset
60          reset_bar <= '0';
61          WAIT FOR 5 * T;
62          WAIT;
63      END PROCESS;
64
65      generate_clock : PROCESS
66      BEGIN
67          clk <= '0';
68          WAIT FOR T/2;
69          clk <= '1';
70          WAIT FOR T/2;
71      END PROCESS;
72
73  END ARCHITECTURE;
```

Έξοδος προσομοίωσης



Στο test bench δίνουμε στην είσοδο modulo κάθε δυνατή τιμή (από 1 έως 7) και παρατηρούμε ότι ο μετρητής μετράει προς τα πάνω και παίρνει τιμές έως modulo-1. Παρουσιάζεται επίσης η παύση όταν το enable είναι 0 και το ασύγχρονο reset.

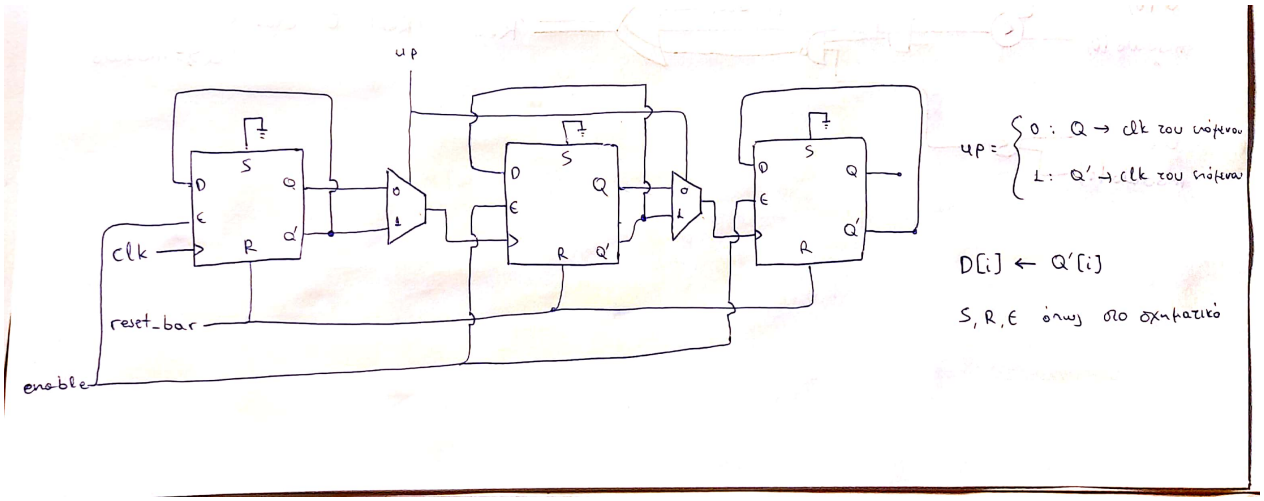
RTL Ανάλυση



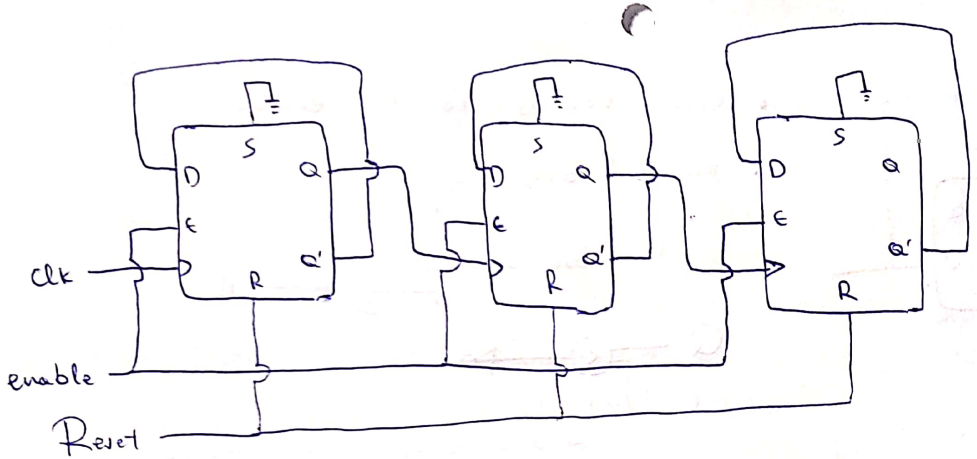
3.

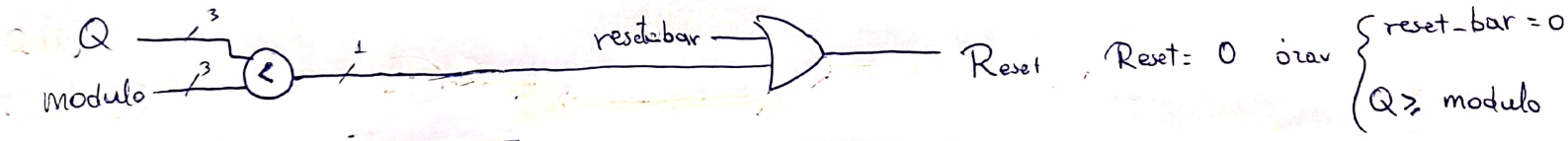
Μια οικονομικότερη σε υλικό υλοποίηση θα ήταν η σχεδίαση με μετρητή ριπής αντί για καταχωρητές και κύκλωμα άθροισης.

Το κύκλωμα για το ζητούμενο 1 χρησιμοποιεί πολυπλέκτες επιλογής είτε το Q είτε του Q' ως είσοδο στην επόμενο καταχωρητή, για μέτρηση άνω ή κάτω.



Το κύκλωμα για το ζητούμενο 2 μετράει μόνο προς τα πάνω, αλλά έχει έναν επιπλέον συγκριτή 3 bits, ο οποίος ενεργοποιεί το reset όταν ο μετρητής φτάσει την τιμή modulo.





Reset: 0 ise $\begin{cases} \text{reset\_bar} = 0 \\ Q \geq \text{modulo} \end{cases}$