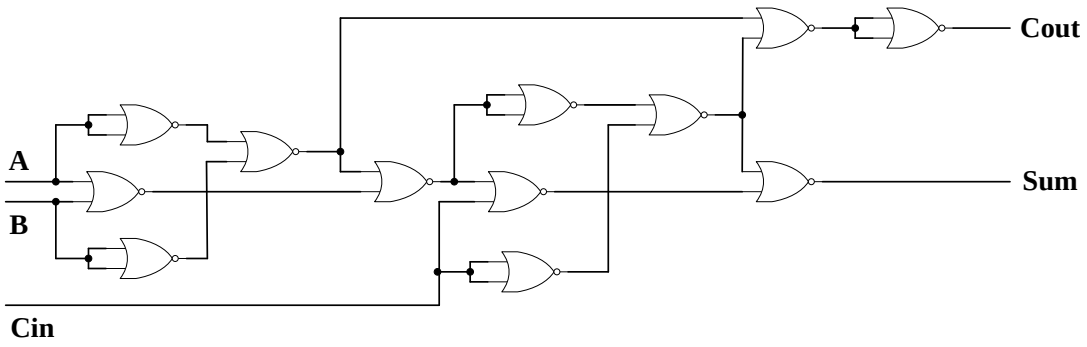
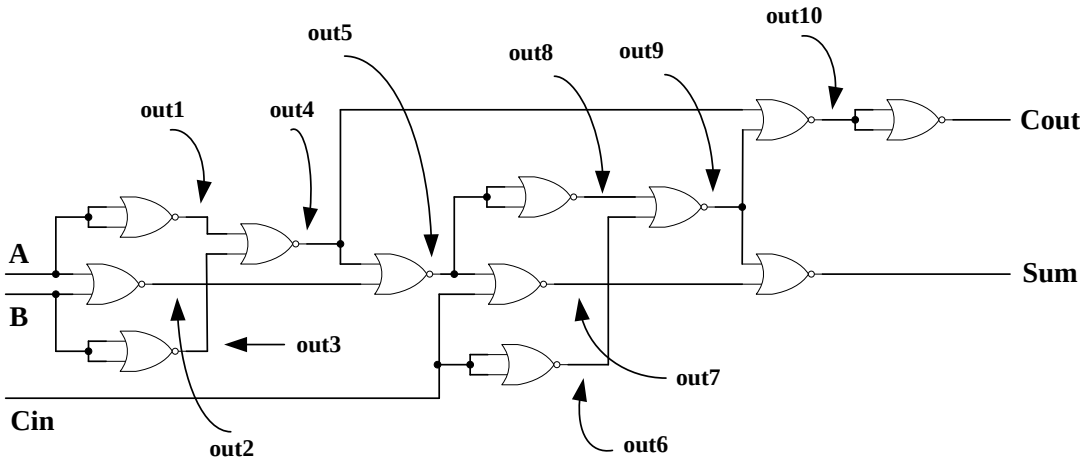
ΑΣΚΗΣΗ 1 (5 + 15 + 10 = 30 ΜΟΝΑΔΕΣ)

1) Να γραφεί ο κώδικας VHDL του πλήρους αθροιστή με πύλες NOR, του παρακάτω σχήματος. Τα δεδομένα εισόδου A, B, Cin και εξόδου Sum , Cout να είναι τύπου std_logic. Ονομάστε την οντότητα (entity) του κυκλώματος FA1bit.



Απάντηση

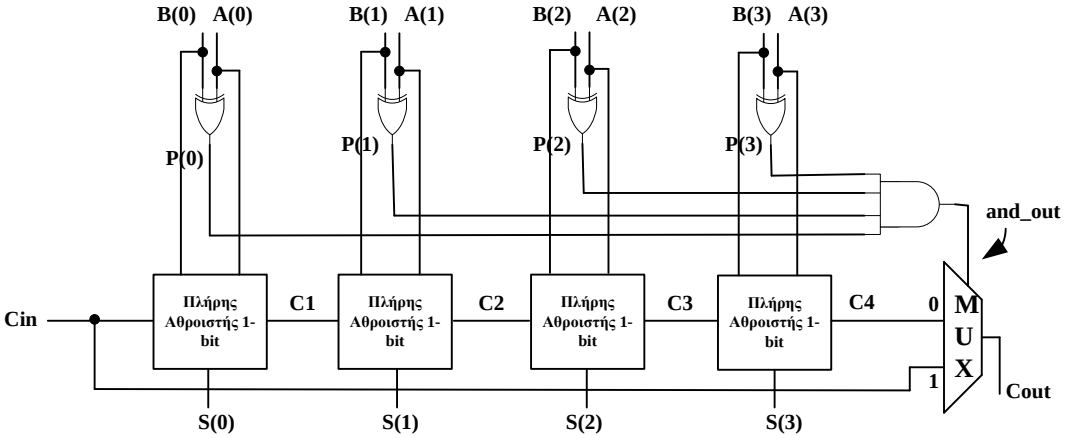
Οι ακροδέκτες εισόδων και εξόδων θα είναι τύπου std_logic, άρα θα πρέπει να δηλωθεί το πακέτο std_logic_1164.all της IEEE στο αντίστοιχο τμήμα του κώδικα. Άρα, θεωρώντας τα εσωτερικά σήματα του παρακάτω σχήματος, ο κώδικας VHDL του πλήρους αθροιστή του 1 bit είναι:



```
LIBRARY ieee;
```

```
USE ieee.std_logic_1164.all;

ENTITY FA1bit IS
PORT (A, B: IN STD_LOGIC;
        Cin: IN STD_LOGIC;
        Sum, Cout: OUT STD_LOGIC);
END FA1bit;
-------------------------------------------------
ARCHITECTURE FA1bit _rtl OF FA1bit IS
SIGNAL out1, out2, out3, out4, out5, out6, out7, out8, out9, out10: STD_LOGIC; -- δήλωση
των εσωτερικών σημάτων
BEGIN

out1 <= A NOR A;
out2 <= A NOR B;
out3 <= B NOR B;
out4 <= out1 NOR out3;
out5 <= out2 NOR out4;
out6 <= Cin NOR Cin;
out7 <= Cin NOR out5;
out8 <= out5 NOR out5;
out9 <= out6 NOR out8;
out10 <= out4 NOR out9;
Sum <= out7 NOR out9;
Cout <= out10 NOR out10;
END FA1bit _rtl;
```

2) Να γραφεί ο κώδικας VHDL του αθροιστή παράκαμψης κρατουμένου των 4-bit, του παρακάτω σχήματος, χρησιμοποιώντας τέσσερις πλήρεις αθροιστές του 1-bit του προηγούμενου ερωτήματος (structural τρόπος σχεδίασης). Τα δεδομένα εισόδου A(3:0), B(3:0) και Cin και εξόδου S(3:0) και Cout να είναι τύπου std_logic και std_logic_vector. Ονομάστε την οντότητα (entity) του κυκλώματος adderbypass4.



Απάντηση

Ο Αθροιστής Παράκαμψης Κρατουμένου είναι ένα συνδυαστικό κύκλωμα που μπορεί να κατασκευαστεί με την διαδοχική σύνδεση n μονάδων πλήρων αθροιστών του 1-bit. Για τις εισόδους των τελεστέων των 4-bit (A, B) καθώς και την έξοδο (S) επίσης των 4-bit θα γίνει χρήση διαύλου των 4-bit οι οποίοι θα περιγραφούν στη VHDL σαν Vector τύπου std_logic. Το κρατούμενο εισόδου (Cin) και το κρατούμενο εξόδου (Cout) θα είναι τύπου std_logic. Επίσης, το σήμα P των 4-bit θα είναι δίαυλος των 4-bit και θα δηλωθεί ως Vector τύπου std_logic. Τέλος τα εσωτερικά σήματα C1, C2, C3, C4 και and_out που θα χρησιμοποιηθούν ως σήματα κρατουμένων μεταξύ των αθροιστών του 1-bit και ως έξοδος της πύλης AND των 4-bit αντίστοιχα (δείτε το σχήμα παραπάνω) θα είναι επίσης τύπου std_logic. Για τους λόγους αυτούς απαιτείται η χρήση του πακέτου std_logic_1164.all της βιβλιοθήκης IEEE.

Για την υλοποίηση του αθροιστή παράκαμψης κρατουμένου απαιτείται ο κώδικας του πολυπλέκτη 2 σε 1, ο οποίος θα μπορούσε να είναι ο παρακάτω.

```
----------------------------------------------------
Library IEEE;
Use IEEE.std_logic_1164.all;
----------------------------------------------------
ENTITY mux2_1 IS
port(in1,in2 : in std_logic;
sel  : in std_logic;
output :out std_logic);
END mux2_1;

ARCHITECTURE behavioral OF mux2_1 IS
BEGIN

Process(sel,in1,in2)
BEGIN

Case sel is
```

```
when ‘0’ =>
output<=in1;
when others =>
output<=in2;
End case;

END process;
END behavioral;
```

Οπότε ο ζητούμενος κώδικας του αθροιστή είναι ο παρακάτω.

```
----------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY adderbypass4 IS
PORT (A, B: IN STD_LOGIC_VECTOR(3 downto 0);
          Cin: IN STD_LOGIC;
          S: OUT STD_LOGIC_VECTOR(3 downto 0);
          Cout: OUT STD_LOGIC);
END adderbypass4;
-----------------------------------------------
ARCHITECTURE adderbypass4_rtl OF adderbypass4 IS

COMPONENT FA1bit -- Χρήση του αθροιστή του 1-bit
PORT (A, B: IN STD_LOGIC;
          Cin: IN STD_LOGIC;
          Sum, Cout: OUT STD_LOGIC);
END COMPONENT;

COMPONENT mux2_1 -- Χρήση του πολυπλέκτη 2 σε 1 του 1-bit
port(in1,in2 : in std_logic;
sel  : in std_logic;
output :out std_logic);
END COMPONENT;

SIGNAL C1, C2, C3, C4, and_out: STD_LOGIC; -- δήλωση των εσωτερικών σημάτων
SIGNAL P : std_logic_vector(3 downto 0);

BEGIN

P(0) <= A(0) xor B(0);
P(1) <= A(1) xor B(1);
P(2) <= A(2) xor B(2);
P(3) <= A(3) xor B(3);
```

```
Block0: adder1bit port map (A=>A(0),B=>B(0),Cin=>Cin,Sum=>S(0),Cout=>C1); --Δήλωση του δεξιότερου αθροιστή του 1-bit
Block1: adder1bit port map (A=>A(1),B=>B(1),Cin=>C1,Sum=>S(1),Cout=>C2);
Block2: adder1bit port map (A=>A(2),B=>B(2),Cin=>C2,Sum=>S(2),Cout=>C3);
Block3: adder1bit port map (A=>A(3),B=>B(3),Cin=>C3,Sum=>S(3),Cout=>C4); --Δήλωση του αριστερότερου αθροιστή του 1-bit
and_out <= P(0) and P(1) and P(2) and P(3);
Block4: mux2_1 port map (in1=>C4, in2=>Cin, sel=>and_out, output=>Cout);

END adderbypass4_rtl;
```

3) Να εξομοιώσετε το κύκλωμα του αθροιστή παράκαμψης κρατουμένου των 4-bit με το ModelSim. Να δοκιμάστε τουλάχιστον τρία διαφορετικά ζεύγη αριθμών (αθροιστέοι) των 4-bit. Να παραδώσετε τον κώδικα του testbench που αναπτύξατε και χαρακτηριστικά στιγμιότυπα των κυματομορφών, όπου επιδεικνύεται η λειτουργικότητα του αθροιστή.

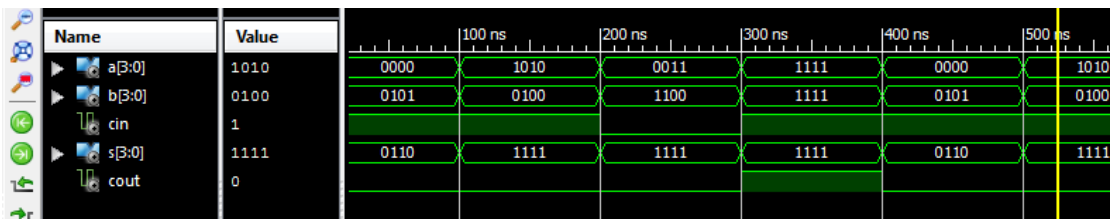
Απάντηση

Το testbench για τον αθροιστή παράκαμψης κρατουμένου των 4-bit με τέσσερα ζεύγη τιμών είναι το παρακάτω:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY adderbypass4_tb IS
END adderbypass4_tb;

ARCHITECTURE behavior OF adderbypass4_tb IS

  -- Component Declaration for the Unit Under Test (UUT)

  COMPONENT adderbypass4
  PORT(
      A : IN  std_logic_vector(3 downto 0);
      B : IN  std_logic_vector(3 downto 0);
      Cin : IN  std_logic;
      S : OUT  std_logic_vector(3 downto 0);
      Cout : OUT  std_logic
     );
  END COMPONENT;


 --Inputs
 signal A : std_logic_vector(3 downto 0) := (others => '0');
 signal B : std_logic_vector(3 downto 0) := (others => '0');
 signal Cin : std_logic := '0';
```

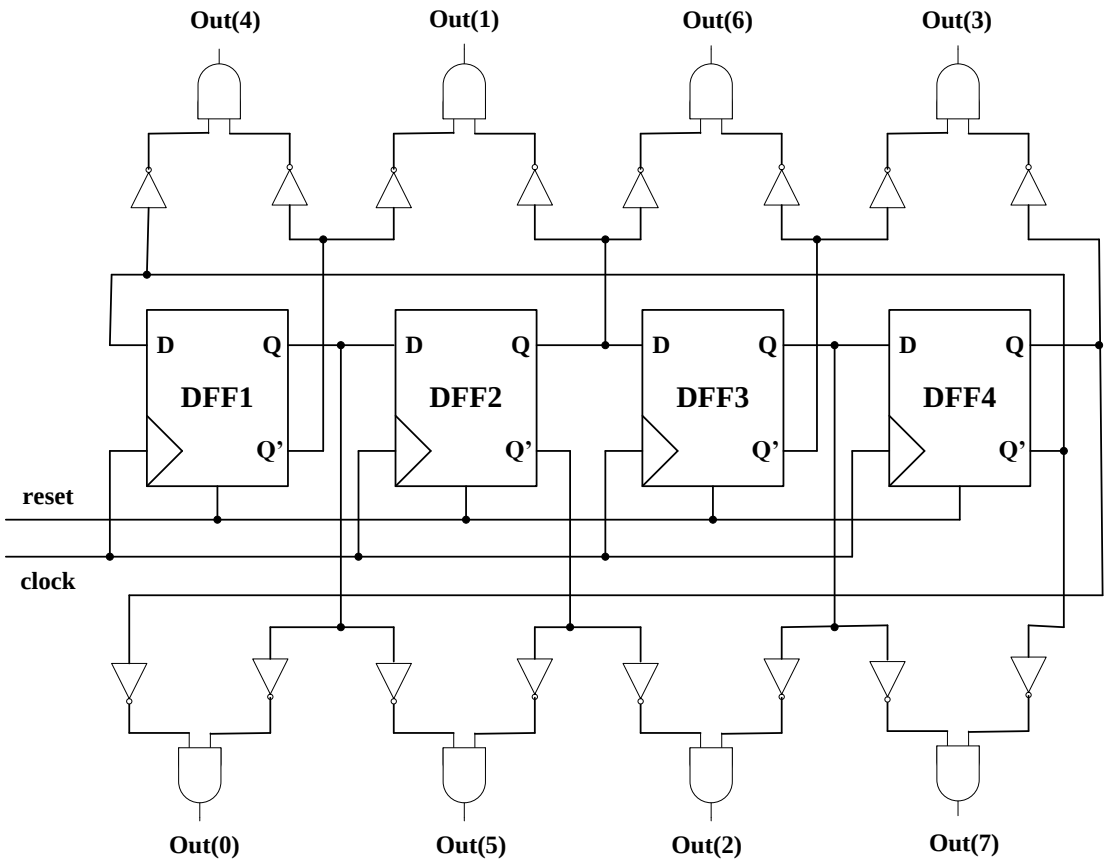
```
        --Outputs
  signal S : std_logic_vector(3 downto 0);
  signal Cout : std_logic;

BEGIN

        -- Instantiate the Unit Under Test (UUT)
  uut: adderbypass4 PORT MAP (
       A => A,
       B => B,
       Cin => Cin,
       S => S,
       Cout => Cout
      );

  -- Stimulus process
  stim_proc: process
  begin
                    A <= "0000";
                    B <= "0101";
                    Cin <= '1';
    wait for 100 ns;
                    A <= "1010";
                    B <= "0100";
                    Cin <= '1';
    wait for 100 ns;
                    A <= "0011";
                    B <= "1100";
                    Cin <= '0';
    wait for 100 ns;
                    A <= "1111";
                    B <= "1111";
                    Cin <= '1';
    wait for 100 ns;
  end process;

END;
```

Το στιγμιότυπο της εξομοίωσης του αθροιστή είναι παρακάτω.



ΑΣΚΗΣΗ 2 (20 + 10 = 30 ΜΟΝΑΔΕΣ)

1) Στο παρακάτω σχήμα δίνεται το κύκλωμα ενός δυαδικού μετρητή των 4-bit. Να γραφεί ο κώδικας VHDL αυτού του μετρητή.



Για το σκοπό αυτό να υλοποιήσετε ένα D Flip Flop θετικά ακμοπυροδότητο το οποίο θα το τοποθετήσετε σε ένα πακέτο (my_package). Για τις λογικές πύλες να χρησιμοποιήσετε τους κατάλληλους τελεστές. Όπως φαίνεται στο σχήμα οι είσοδοι του μετρητή είναι το clock και το res και η έξοδός του η O(7:0). Ονομάστε την οντότητα (entity) του κυκλώματος counter4.

Απάντηση

Ο κώδικας του D Flip Flop είναι ο παρακάτω.

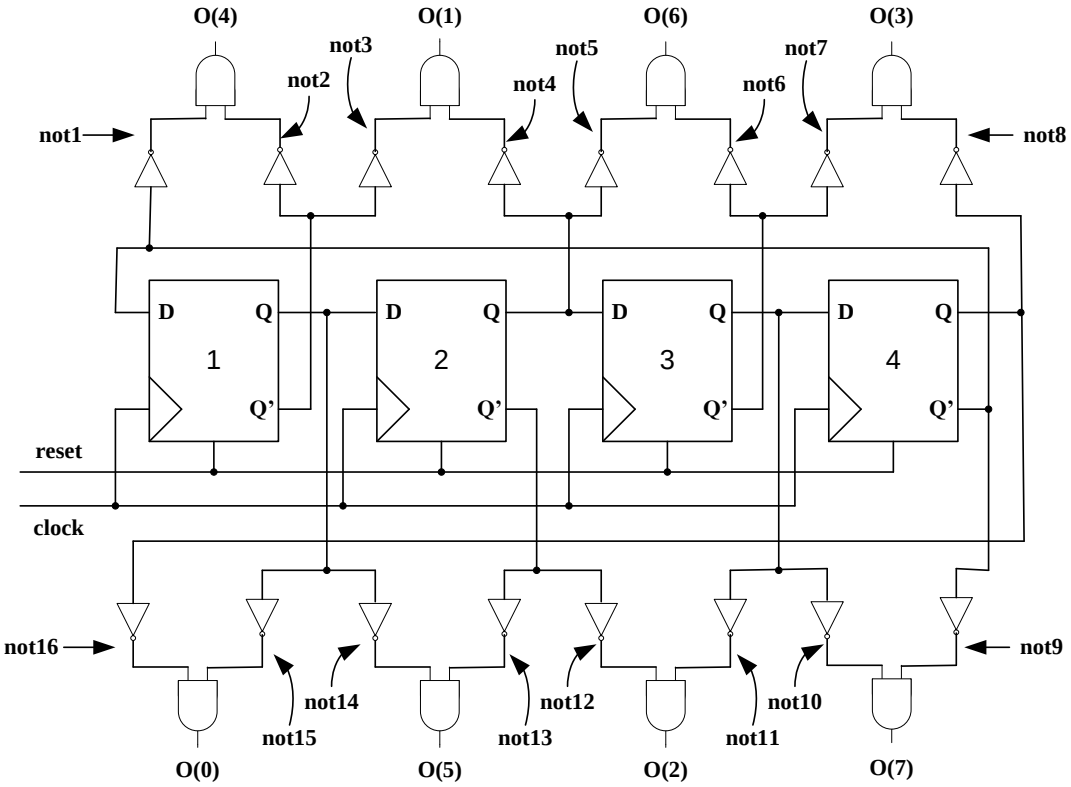
```
Library Ieee;
Use Ieee.std_logic_1164.all;

Entity DFF is
Port (D, clock, reset : In std_logic;
```

```
Q,Qbar: Out std_logic);
End DFF;

Architecture behavioral of DFF is
begin

Process(clock,reset)
Variable tmp : std_logic :='0';
begin
        If rising_edge(clock) then
                If reset='1'  then
                        tmp := '0';
                else
                        tmp := D;
                End if;
        End if;
        Q<=tmp;
        Qbar<= not tmp;

End process;
End behavioral;
```

Τότε ο κώδικας του πακέτου my_package είναι ο παρακάτω.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
PACKAGE my_components IS

Component DFF is
Port (D, clock, reset : In std_logic;
Q,Qbar: Out std_logic);
End Component;

END my_components;
```

Τέλος, θεωρώντας τα εσωτερικά σήματα του παρακάτω σχήματος, ο κώδικας VHDL του ζητούμενου μετρητή είναι ο παρακάτω. Επίσης, θεωρείστε τα εσωτερικά σήματα των εξόδων Q και Q' των D Flip Flop (DFF). Του DFF 1 η έξοδος Q είναι το σήμα Q1_out και η έξοδος Q' είναι η Qbar1_out. Ομοίως για τα υπόλοιπα DFF.




```
LIBRARY Ieee;
USE Ieee.std_logic_1164.all;
USE work.my_components.all; -- Δήλωση του πακέτου my_component για χρήση του D Flip Flop
Entity counter4 is
Port (clock, reset : In std_logic;
O : Out std_logic_vector(7 downto 0));
End counter4;

Architecture structural of counter4 is

Signal not1, not2, not3, not4, not5, not6, not7, not8, not9, not10, not11, not12, not13, not14, not15, not16 : std_logic;
Signal Q1_out, Qbar1_out, Q2_out, Qbar2_out, Q3_out, Qbar3_out, Q4_out, Qbar4_out: std_logic;

Begin

block1: DFF port map (D=>Qbar4_out, clock=>clock, reset=>reset, Q=>Q1_out, Qbar=>Qbar1_out); -- Το αριστερότερο D Flip Flop (Το νούμερο 1)
block2: DFF port map (D=>Q1_out, clock=>clock, reset=>reset, Q=>Q2_out, Qbar=>Qbar2_out);
```

```
block3: DFF port map (D=>Q2_out, clock=>clock, reset=>reset, Q=>Q3_out, Qbar=>Qbar3_out);
block4: DFF port map (D=>Q3_out, clock=>clock, reset=>reset, Q=>Q4_out, Qbar=>Qbar4_out); -- Το δεξιότερο D Flip Flop (Το νούμερο 4)

not1 <= not Qbar4_out;
not2 <= not Qbar1_out;
not3 <= not Qbar1_out;
not4 <= not Q2_out;
not5 <= not Q2_out;
not6 <= not Qbar3_out;
not7 <= not Qbar3_out;
not8 <= not Q4_out;
not9 <= not Qbar4_out;
not10 <= not Q3_out;
not11 <= not Q3_out;
not12 <= not Qbar2_out;
not13 <= not Qbar2_out;
not14 <= not Q1_out;
not15 <= not Q1_out;
not16 <= not Q4_out;

O(4) <= not1 AND not2;
O(1) <= not3 AND not4;
O(6) <= not5 AND not6;
O(3) <= not7 AND not8;
O(7) <= not9 AND not10;
O(2) <= not11 AND not12;
O(5) <= not13 AND not14;
O(0) <= not15 AND not16;

End structural;
```

2) Να εξομοιώσετε το κύκλωμα του μετρητή με το ModelSim όπου θα επιδεικνύονται όλες οι καταστάσεις του μετρητή. Να παραδώσετε τον κώδικα του testbench που αναπτύξατε και χαρακτηριστικά στιγμιότυπα των κυματομορφών, όπου επιδεικνύεται η λειτουργικότητα του απαριθμητή.

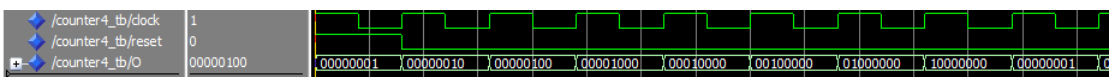
Απάντηση

Το testbench του μετρητή είναι το παρακάτω.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY counter4_tb IS
```

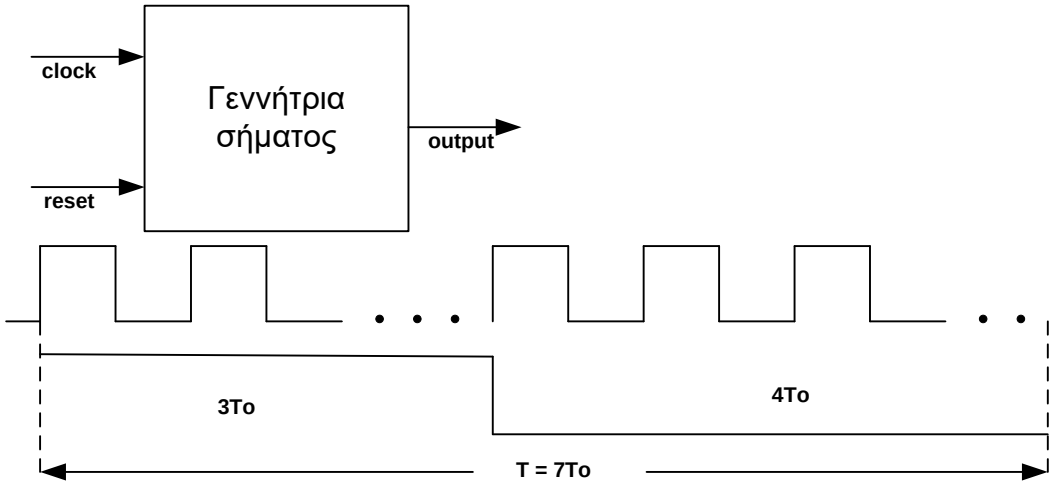
```
END counter4_tb;
ARCHITECTURE counter4_arch OF counter4_tb IS

SIGNAL clock : STD_LOGIC;
SIGNAL reset : STD_LOGIC;
SIGNAL O : STD_LOGIC_VECTOR(7 downto 0);

Component counter4
Port (clock, reset : In std_logic;
O : Out std_logic_vector(7 downto 0));
End component;

BEGIN
        i1 : counter4
        PORT MAP (
        clock => clock,
        reset => reset,
        O => O
        );

clk_generator: PROCESS
                begin
        clock <= '1';
                WAIT FOR 40 ns;
                clock <= '0';
                WAIT FOR 40 ns;
        END PROCESS;


reset_generator: PROCESS
                begin
        reset <='1';
                WAIT FOR 80 ns;
                reset <='0';
                WAIT;
        END PROCESS;

END counter4_arch;
```

Και τα στιγμιότυπα από την εξομοίωση, φαίνονται παρακάτω.



ΑΣΚΗΣΗ 3 (5 + 20 + 15 = 40 ΜΟΝΑΔΕΣ)

Έστω μια γεννήτρια παραγωγής σήματος που φαίνεται στα παρακάτω σχήματα.

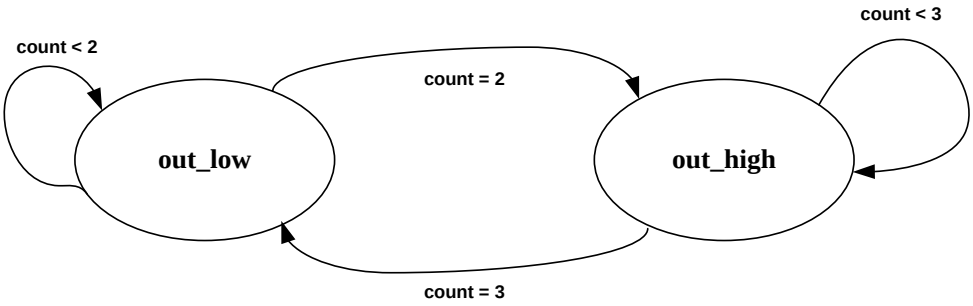


Η έξοδος output είναι ίση με 1 κατά τη διάρκεια των 3 πρώτων παλμών ρολογιού και είναι ίση με 0 κατά τη διάρκεια των 4 επόμενων παλμών.

i) Να σχεδιάσετε το διάγραμμα καταστάσεων της γεννήτριας

Απάντηση

Το διάγραμμα καταστάσεων είναι το παρακάτω.



ii) Να γράψετε τον κώδικα VHDL που περιγράφει τη γεννήτρια. Τα δεδομένα εισόδων και εξόδων να είναι τύπου std_logic.

Απάντηση

Ο κώδικας της γεννήτριας παραγωγής σήματος είναι ο παρακάτω.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY signal_generator IS
```

```
GENERIC (PulsesLow: NATURAL := 3;
PulsesHigh: NATURAL := 4);
PORT (clk, rst: IN STD_LOGIC;
output: OUT STD_LOGIC);
END ENTITY;
-----------------------------------------------------------------
ARCHITECTURE fsm OF signal_generator IS
TYPE state IS (OutputLow, OutputHigh);
SIGNAL pr_state, nx_state: state;
SIGNAL number_of_pulses: NATURAL RANGE 0 TO PulsesHigh;
BEGIN

PROCESS (rst, clk)
VARIABLE count: NATURAL RANGE 0 TO PulsesHigh;
BEGIN
IF (rst='1') THEN
        pr_state <= OutputLow;
ELSIF (clk'EVENT AND clk='1') THEN
        count := count + 1;
        IF (count=number_of_pulses) THEN
                count := 0;
                pr_state <= nx_state;
        END IF;
END IF;
END PROCESS;

PROCESS (pr_state)
BEGIN
CASE pr_state IS
        WHEN OutputLow =>
                output <= '0';
                number_of_pulses <= PulsesLow;
                nx_state <= OutputHigh;
        WHEN OutputHigh =>
                output <= '1';
                number_of_pulses <= PulsesHigh;
                nx_state <= OutputLow;
END CASE;
END PROCESS;
END ARCHITECTURE;
-----------------------------------------------------------------
```

iii) Να εξομοιώσετε το κύκλωμα της γεννήτριας με το ModelSim, όπου θα επιδεικνύεται η επίδραση όλων των σημάτων εισόδου σε όλες τις μεταβάσεις. Να παραδώσετε τον κώδικα του testbench που αναπτύξατε και χαρακτηριστικά στιγμιότυπα των κυματομορφών, όπου επιδεικνύεται η λειτουργικότητα της γεννήτριας.

Απάντηση

Τα στιγμιότυπα από την εξομοίωση, φαίνονται παρακάτω.