ΣΤΟΙΧΕΙΑ ΠΟΥ ΣΥΜΠΛΗΡΩΝΕΙ Ο ΦΟΙΤΗΤΗΣ

|  |  |
| --- | --- |
| **ΟΝΟΜΑΤΕΠΩΝΥΜΟ** | ΔΗΜΗΤΡΗΣ ΜΠΑΡΜΠΑΚΟΣ |
| **Α/Α ΓΡΑΠΤΗΣ ΕΡΓΑΣΙΑΣ** | 1η ΕΡΓΑΣΙΑ |
| **ΗΜΕΡΟΜΗΝΙΑ ΑΠΟΣΤΟΛΗΣ** | Τρίτη, 19 Νοεμβρίου 2019 |
| **ΣΧΟΛΙΑ ΠΡΟΣ ΚΑΘΗΓΗΤΗ** |  |

**Ημερομηνία ανακοίνωσης εργασίας: 5/11/2019**

**Ημερομηνία παράδοσης εργασίας: 4/12/2019**

ΣΤΟΙΧΕΙΑ ΠΟΥ ΣΥΜΠΛΗΡΩΝΕΙ Ο ΚΑΘΗΓΗΤΗΣ

|  |  |
| --- | --- |
| **ΟΝΟΜΑΤΕΠΩΝΥΜΟ** |  |
| **ΗΜΕΡΟΜΗΝΙΑ ΑΞΙΟΛΟΓΗΣΗΣ** | **Click here to enter a date.** |
| **ΒΑΘΜΟΣ** | <*αριθμητικώς*> (<*ολογράφως*>) |

**ΣΧΟΛΙΑ ΠΡΟΣ ΦΟΙΤΗΤΗ / ΦΟΙΤΗΤΡΙΑ:**

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 Α, Β, Cin και εξόδου Sum , Cout να είναι τύπου std\_logic. Ονομάστε την οντότητα (entity) του κυκλώματος FA1bit.



**Απάντηση**

***-------------------------------------------------------***

***-- Design Name : Askisi\_1 [ Skip-carry full adder ]***

***-- File Name : FA1bit.vhd***

***-- Function : 1bit full adder structural design***

***-- whois : Dimitris BARMPAKOS (VHDL)***

***-------------------------------------------------------***

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**library** work**;**

**use** work**.all;**

***-- /libraries -----------------------------------------***

**entity** FA1bit **is**

**port** **(**

A\_1bitFA**,** B\_1bitFA**,** Cin\_1bitFA **:** **in** std\_logic**;**

Sum\_1bitFA**,** Cout\_1bitFA **:** **out** std\_logic

**);**

**end** FA1bit**;**

***-- /entity --------------------------------------------***

**architecture** structure **of** FA1bit **is**

***-- one signal for each NOR2 output***

**signal** io **:** std\_logic\_vector**(**9 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**begin**

io**(**0**)** **<=** A\_1bitFA NOR A\_1bitFA**;**

io**(**1**)** **<=** A\_1bitFA NOR B\_1bitFA**;**

io**(**2**)** **<=** B\_1bitFA NOR B\_1bitFA**;**

io**(**3**)** **<=** io**(**0**)** NOR io**(**2**);**

io**(**4**)** **<=** io**(**3**)** NOR io**(**1**);**

io**(**5**)** **<=** io**(**4**)** NOR io**(**4**);**

io**(**6**)** **<=** io**(**4**)** NOR Cin\_1bitFA**;**

io**(**7**)** **<=** Cin\_1bitFA NOR Cin\_1bitFA**;**

io**(**8**)** **<=** io**(**5**)** NOR io**(**7**);**

io**(**9**)** **<=** io**(**3**)** NOR io**(**8**);**

Sum\_1bitFA **<=** io**(**8**)** NOR io**(**6**);**

Cout\_1bitFA **<=** io**(**9**)** NOR io**(**9**);**

***-- /logic -----------------------------------------***

**end** structure**;**

***-- /architecture --------------------------------------***

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 Α(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**

***-------------------------------------------------------***

***-- Design Name : Askisi\_1 [ Skip-carry full adder ]***

***-- File Name : adderBypass4.vhd***

***-- Function : Structural 4bit adder with skip-carry***

***-- whois : Dimitris BARMPAKOS (VHDL)***

***-------------------------------------------------------***

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**library** work**;**

**use** work**.all;**

***-- /libraries -----------------------------------------***

**entity** adderBypass4 **is**

**port** **(**

A\_In**,** B\_In **:** **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**;**

***-- /entity --------------------------------------------***

**architecture** structural **of** adderBypass4 **is**

**component** FA1bit **is**

**port** **(**

A\_1bitFA**,** B\_1bitFA**,** Cin\_1bitFA **:** **in** std\_logic**;**

Sum\_1bitFA**,** Cout\_1bitFA **:** **out** std\_logic

**);**

**end** **component;**

**component** carryMux2t1 **is**

**port** **(**

A\_cMux2t1**,** B\_cMux2t1**,** Sel\_cMux2t1 **:** **in** std\_logic**;**

Z\_cMux2t1 **:** **out** std\_logic

**);**

**end** **component;**

***-- /components ------------------***

***-- P is XOR results, carryTemp is FAs'1bit results***

**signal** P**,** carryTemp **:** std\_logic\_vector**(**3 **downto** 0**);**

**signal** and\_out **:** std\_logic**;**

**begin**

P**(**3**)** **<=** A\_In**(**3**)** XOR B\_In**(**3**);**

P**(**2**)** **<=** A\_In**(**2**)** XOR B\_In**(**2**);**

P**(**1**)** **<=** A\_In**(**1**)** XOR B\_In**(**1**);**

P**(**0**)** **<=** A\_In**(**0**)** XOR B\_In**(**0**);**

and\_out **<=** **(**P**(**0**)** AND P**(**1**)** AND P**(**2**)** AND P**(**3**));**

carryMux2t1\_0 **:** carryMux2t1 **port** **map** **(**

A\_cMux2t1 **=>** carryTemp**(**3**),**

B\_cMux2t1 **=>** Cin**,**

Sel\_cMux2t1 **=>** and\_out**,**

Z\_cMux2t1 **=>** Cout

**);**

FA1bit\_0 **:** FA1bit **port** **map** **(**

A\_1bitFA **=>** A\_In**(**3**),**

B\_1bitFA **=>** B\_In**(**3**),**

Cin\_1bitFA **=>** Cin**,**

Sum\_1bitFA **=>** S**(**3**),**

Cout\_1bitFA **=>** carryTemp**(**0**)**

**);**

FA1bit\_1 **:** FA1bit **port** **map** **(**

A\_1bitFA **=>** A\_In**(**2**),**

B\_1bitFA **=>** B\_In**(**2**),**

Cin\_1bitFA **=>** carryTemp**(**0**),**

Sum\_1bitFA **=>** S**(**2**),**

Cout\_1bitFA **=>** carryTemp**(**1**)**

**);**

FA1bit\_2 **:** FA1bit **port** **map** **(**

A\_1bitFA **=>** A\_In**(**1**),**

B\_1bitFA **=>** B\_In**(**1**),**

Cin\_1bitFA **=>** carryTemp**(**1**),**

Sum\_1bitFA **=>** S**(**1**),**

Cout\_1bitFA **=>** carryTemp**(**2**)**

**);**

FA1bit\_3 **:** FA1bit **port** **map** **(**

A\_1bitFA **=>** A\_In**(**0**),**

B\_1bitFA **=>** B\_In**(**0**),**

Cin\_1bitFA **=>** carryTemp**(**2**),**

Sum\_1bitFA **=>** S**(**0**),**

Cout\_1bitFA **=>** carryTemp**(**3**)**

**);**

***-- /logic -----------------------------------------***

**end** structural;

***-- /architecture --------------------------------------***

***-------------------------------------------------------***

***-- Design Name : Askisi\_1 [ Skip-carry full adder ]***

***-- File Name : carryMux2t1.vhd***

***-- Function : 2x1 Multiplexer block for adderBypass4***

***-- Sel\_cMux2t1 is controlled by carry AND result***

***-- whois : Dimitris BARMPAKOS (VHDL)***

***-------------------------------------------------------***

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**library** work**;**

**use** work**.all;**

***-- /libraries -----------------------------------------***

**entity** carryMux2t1 **is**

**port** **(**

A\_cMux2t1**,** B\_cMux2t1**,** Sel\_cMux2t1 **:** **in** std\_logic**;**

Z\_cMux2t1 **:** **out** std\_logic

**);**

**end** carryMux2t1**;**

***-- /entity --------------------------------------------***

**architecture** struct **of** carryMux2t1 **is**

**signal** A\_io**,** B\_io **:** std\_logic**;**

**begin**

A\_io **<=** A\_cMux2t1 AND **(**NOT **(**Sel\_cMux2t1**));**

B\_io **<=** B\_cMux2t1 AND Sel\_cMux2t1**;**

Z\_cMux2t1 **<=** A\_io OR B\_io**;**

***-- /logic -----------------------------------------***

**end** struct**;**

***-- /architecture --------------------------------------***

3) Να εξομοιώσετε τo 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 πολλαπλασιαστή.

**Απάντηση**

***-------------------------------------------------------***

***-- Design Name : Askisi\_1 [ Skip-carry full adder ]***

***-- File Name : AdderBypass4\_tb.vhd***

***-- Function : Tests four cases of addition for***

***-- 4bit adder with skip-carry function***

***-- whois : Dimitris BARMPAKOS (VHDL)***

***-------------------------------------------------------***

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**library** work**;**

**use** work**.all;**

***-- /libraries -----------------------------------------***

**entity** adderBypass4\_tb **is**

**end** adderBypass4\_tb**;**

***-- /entity --------------------------------------------***

**architecture** adderBypass4\_testbench **of** adderBypass4\_tb **is**

**component** adderBypass4 **is**

**port** **(**

A\_In**,** B\_In **:** **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;**

**signal** A**,** B**,** S **:** std\_logic\_vector**(**3 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**signal** Cin**,** Cout **:** std\_logic **:=** '0'**;**

**begin**

uut **:** adderBypass4 **port** **map** **(**

A**,** B**,** Cin**,** S**,** Cout

**);**

stim\_proc **:** **process**

**begin**

Cin **<=** '0'**;**

A **<=** "0000"**;**

B **<=** "1111"**;**

**wait** **for** 100 ns**;**

A **<=** "0101"**;**

B **<=** "0000"**;**

**wait** **for** 100 ns**;**

A **<=** "0011"**;**

B **<=** "1110"**;**

**wait** **for** 100 ns**;**

A **<=** "1111"**;**

B **<=** "1001"**;**

**wait** **for** 100 ns**;**

A **<=** "0000"**;**

B **<=** "0000"**;**

**wait** **for** 100 ns**;**

**wait;**

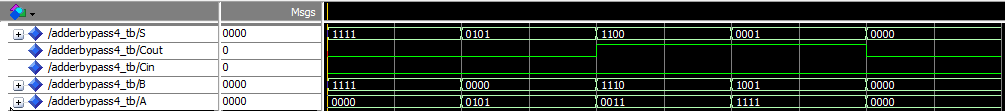
**end** **process;**

***-- /logic -----------------------------------------***

**end** adderBypass4\_testbench**;**

***-- /architecture --------------------------------------***

***Wave output:***



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 reset και η έξοδός του η Out(7:0). Ονομάστε την οντότητα (entity) του κυκλώματος counter4.

**Απάντηση:**

***-------------------------------------------------------***

***-- Design Name : Askisi\_2 [ 4bit binary counter ]***

***-- File Name : my\_package.vhd***

***-- Function : D-FlipFlop Package***

***-- whois : Dimitris BARMPAKOS (VHDL)***

***-------------------------------------------------------***

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**library** work**;**

**use** work**.all;**

***-- /libraries -----------------------------------------***

**package** my\_package **is**

**procedure** DFF

**(**

**signal** CLK**,** RST**,** D **:** **in** std\_logic**;**

**signal** Q**,** notQ **:** **out** std\_logic

**);**

**end** **package;**

***-- /package header ------------------------------------***

**package** **body** my\_package **is**

**procedure** DFF

**(**

**signal** CLK**,** RST**,** D **:** **in** std\_logic**;**

**signal** Q**,** notQ **:** **out** std\_logic

**)** **is**

**begin**

**if** RST **=** '0' **then**

Q **<=** '0'**;**

notQ **<=** '1'**;**

**elsif** **(**CLK'**event** and CLK**=**'1'**)** **then**

Q **<=** D**;**

notQ **<=** NOT D**;**

**end** **if;**

**end** DFF**;**

***-- /logic -----------------------------------------***

**end** my\_package**;**

***-- /package body --------------------------------------***

***-------------------------------------------------------***

***-- Design Name : Askisi\_2 [ 4bit binary counter ]***

***-- File Name : counter4.vhd***

***-- Function : 4bit binary counter using***

***-- custom D FlipFlop package***

***-- whois : Dimitris BARMPAKOS (VHDL)***

***-------------------------------------------------------***

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**library** work**;**

**use** work**.all;**

**use** work**.**my\_package**.all;**

***-- /libraries -----------------------------------------***

**entity** counter4 **is**

**port** **(**

CLK**,** RST **:** **in** std\_logic**;**

Output **:** **out** std\_logic\_vector**(**7 **downto** 0**)**

**);**

**end** counter4**;**

***-- /entity --------------------------------------------***

**architecture** struct **of** counter4 **is**

**signal** Qs**,** notQs **:** std\_logic\_vector**(**3 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**signal** Outputs **:** std\_logic\_vector**(**7 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**begin**

***-- DFF is defined in my\_package***

DFF**(**CLK**,** RST**,** notQs**(**3**),** Qs**(**0**),** notQs**(**0**));**

DFF**(**CLK**,** RST**,** Qs**(**0**),** Qs**(**1**),** notQs**(**1**));**

DFF**(**CLK**,** RST**,** Qs**(**1**),** Qs**(**2**),** notQs**(**2**));**

DFF**(**CLK**,** RST**,** Qs**(**2**),** Qs**(**3**),** notQs**(**3**));**

**process** **(**CLK**,** RST**,** Qs**,** notQs**)** **is**

**begin**

Outputs**(**0**)** **<=** **(**NOT Qs**(**3**))** AND **(**NOT Qs**(**0**));**

Outputs**(**1**)** **<=** **(**NOT notQs**(**0**))** AND **(**NOT Qs**(**1**));**

Outputs**(**2**)** **<=** **(**NOT notQs**(**1**))** AND **(**NOT Qs**(**2**));**

Outputs**(**3**)** **<=** **(**NOT notQs**(**2**))** AND **(**NOT Qs**(**3**));**

Outputs**(**4**)** **<=** **(**NOT notQs**(**3**))** AND **(**NOT notQs**(**0**));**

Outputs**(**5**)** **<=** **(**NOT notQs**(**1**))** AND **(**NOT Qs**(**0**));**

Outputs**(**6**)** **<=** **(**NOT notQs**(**2**))** AND **(**NOT Qs**(**1**));**

Outputs**(**7**)** **<=** **(**NOT notQs**(**3**))** AND **(**NOT Qs**(**2**));**

**end** **process;**

Output **<=** Outputs**;**

***-- /logic -----------------------------------------***

**end** struct**;**

***-- /architecture --------------------------------------***

2) Να εξομοιώσετε τo 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***

***-------------------------------------------------------***

***-- Design Name : Askisi\_2 [ 4bit binary counter ]***

***-- File Name : counter4\_tb.vhd***

***-- Function : tests binary 4bit counter with clock pulses and rst***

***-- whois : Dimitris BARMPAKOS (VHDL)***

***--***

***-- I have implemented the question both with package and with structural design.***

***-- Comment / Uncomment the appropriate lines for testing***

***-------------------------------------------------------***

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**library** work**;**

**use** work**.all;**

**use** work**.**my\_package**.all;**

***-- /libraries -----------------------------------------***

**entity** counter4\_tb **is**

**end** counter4\_tb**;**

***-- /entity --------------------------------------------***

**architecture** counter4\_tb\_arch **of** counter4\_tb **is**

***-- uncomment below for structural***

***-- component counter4\_struct is***

**component** counter4

**port** **(**

CLK**,** RST **:** **in** std\_logic**;**

Output **:** **out** std\_logic\_vector**(**7 **downto** 0**)**

**);**

**end** **component;**

**constant** clk\_hp **:** time **:=** 3000 ps**;**

**signal** CLK**,** RST **:** std\_logic **:=** '0'**;**

**signal** Output **:** std\_logic\_vector**(**7 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**begin**

***-- uncomment below for structural***

***-- counter\_instance : counter4\_struct port map(CLK, RST, Output);***

counter\_instance **:** counter4 **port** **map(**CLK**,** RST**,** Output**);**

***-- clk generator ----------------------------------***

clock\_gen\_proc **:** **process** **is**

**begin**

clk **<=** '1'**;**

**wait** **for** clk\_hp**/**2**;** ***-- 50% duty cycle***

clk **<=** '0'**;**

**wait** **for** clk\_hp**/**2**;** ***-- 50% duty cycle***

**end** **process** clock\_gen\_proc**;**

***---------------------------------------------------***

***-- rst issue***

trig**:** **process** **is**

**begin**

**wait** **for** clk\_hp**\***15**;**

rst **<=** '0'**;**

**wait** **for** clk\_hp**\***2**;**

rst **<=** '1'**;**

**wait** **for** clk\_hp**\***30**;**

**end** **process** trig**;**

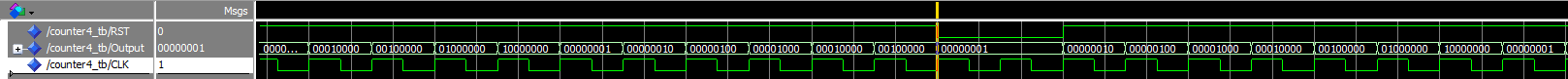
***-- /rst issue***

***-- /logic -----------------------------------------***

**end** counter4\_tb\_arch**;**

***-- /architecture --------------------------------------***

***Wave output:***



Στον κέρσορα παρουσιάζεται η λειτουργία reset.

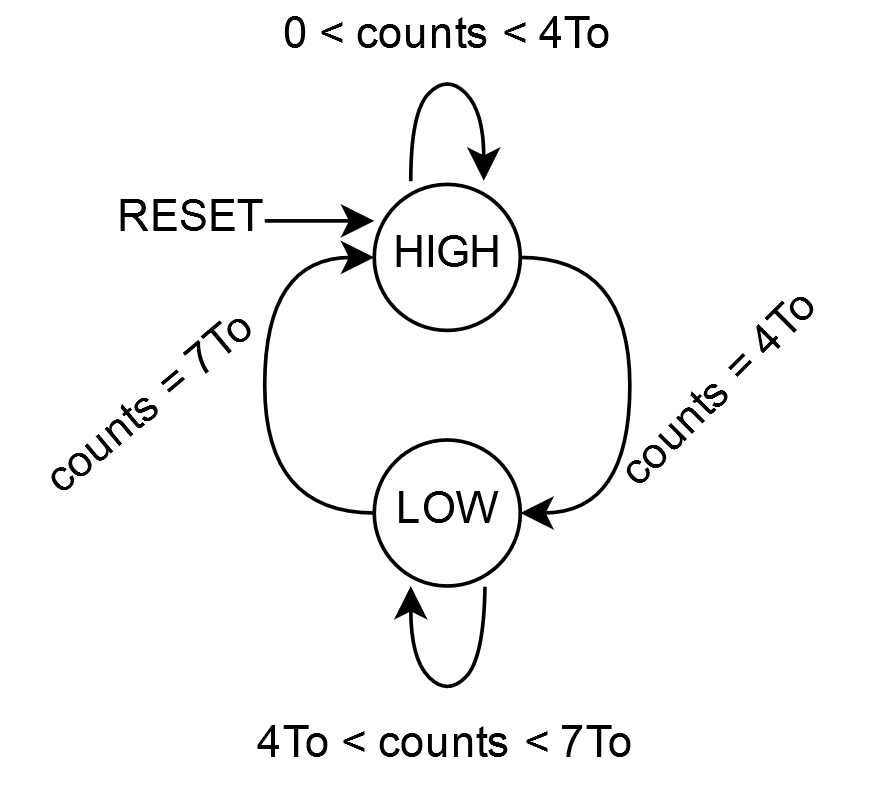
ΑΣΚΗΣΗ 3 (5 + 20 + 15 = 40 ΜΟΝΑΔΕΣ)

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



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

***-------------------------------------------------------***

***-- Design Name : Askisi\_3 [ function generator ]***

***-- File Name : funFSM.vhd***

***-- Function : State Machine with HIGH/LOW (3To/4To)***

***-- whois : Dimitris BARMPAKOS (VHDL)***

***-------------------------------------------------------***

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**use** IEEE**.**std\_logic\_arith**.all;**

**library** work**;**

**use** work**.all;**

***-- /libraries -----------------------------------------***

**entity** funFSM **is**

**port** **(**

CLK**,** RST **:** **in** std\_logic**;**

OUTPUT **:** **out** std\_logic**;**

Y **:** **out** std\_logic\_vector**(**1 **downto** 0**)** ***-- state output***

**);**

**end** funFSM**;**

***-- /entity --------------------------------------------***

**architecture** funFSM\_arch **of** funFSM **is**

**type** state\_type **is** **(**OUT\_HIGH**,** OUT\_LOW**);**

**signal** present\_state**,** next\_state **:** state\_type**;**

**signal** fsm\_output **:** std\_logic**;**

**signal** counter **:** integer **:=** 0**;**

**begin**

sync\_proc **:** **process(**CLK**,** RST**,** next\_state**,** counter**)**

**begin**

**if** **(rising\_edge(**CLK**))** **then**

present\_state **<=** next\_state**;**

counter **<=** counter **+** 1**;**

**case** present\_state **is**

**when** OUT\_HIGH **=>**

fsm\_output **<=** '1'**;**

**when** OUT\_LOW **=>**

fsm\_output **<=** '0'**;**

**if** **(**counter **=** 6**)** **then**

counter **<=** 0**;**

**end** **if;**

**when** **others** **=>**

fsm\_output **<=** fsm\_output**;**

**end** **case;**

**else**

**if** **(**RST **=** '1'**)** **then**

present\_state **<=** OUT\_HIGH**;**

counter **<=** 0**;**

**end** **if;**

**end** **if;**

**end** **process** sync\_proc**;**

comb\_proc **:** **process(**counter**,** present\_state**)**

**begin**

**case** present\_state **is**

**when** OUT\_HIGH **=>**

**if** **(**counter **=** 2**)** **then**

next\_state **<=** OUT\_LOW**;**

**else** next\_state **<=** present\_state**;**

**end** **if;**

**when** OUT\_LOW **=>**

**if** **(**counter **=** 6**)** **then**

next\_state **<=** OUT\_HIGH**;**

**else** next\_state **<=** present\_state**;**

**end** **if;**

**end** **case;**

**end** **process** comb\_proc**;**

***-- state encoding***

**with** present\_state **select**

Y **<=** "00" **when** OUT\_HIGH**,**

"01" **when** OUT\_LOW**,**

"00" **when** **others;**

***-- /state encoding***

OUTPUT **<=** fsm\_output**;**

***-- /logic -----------------------------------------***

**end** funFSM\_arch**;**

***-- /architecture --------------------------------------***

iii) Να εξομοιώσετε τo 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.

***-------------------------------------------------------***

***-- Design Name : Askisi\_3 [ function generator ]***

***-- File Name : funFSM\_tb.vhd***

***-- Function : State Machine with HIGH/LOW (3To/4To)***

***-- Tests RST issuing and 3To/4To output***

***-- whois : Dimitris BARMPAKOS (VHDL)***

***-------------------------------------------------------***

**library** IEEE**;**

**use** IEEE**.**std\_logic\_1164**.all;**

**library** work**;**

**use** work**.all;**

***-- /libraries -----------------------------------------***

**entity** funFSM\_tb **is**

**end** funFSM\_tb**;**

***-- /entity --------------------------------------------***

**architecture** funFSM\_arch\_tb **of** funFSM\_tb **is**

**component** funFSM **is**

**port** **(**

CLK**,** RST **:** **in** std\_logic**;**

OUTPUT **:** **out** std\_logic**;**

Y **:** **out** std\_logic\_vector**(**1 **downto** 0**)** ***-- state output***

**);**

**end** **component;**

***-- signals / variables ----------------------------***

**signal** clock**,** outPutz **:** std\_logic**;**

**signal** rst **:** std\_logic **:=**'0'**;**

**signal** states\_encoded **:** std\_logic\_vector**(**1 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**constant** clk\_period **:** time **:=** 10 ns**;**

***-- /signals / variables ---------------------------***

**begin**

dut **:** funFSM **port** **map** **(**clock**,** rst**,** outPutz**,** states\_encoded**);**

***-- clock generator***

clk\_gen**:** **process** **is**

**begin**

clock **<=** '1'**;**

**wait** **for** clk\_period**/**2**;**

clock **<=** '0'**;**

**wait** **for** clk\_period**/**2**;**

**end** **process** clk\_gen**;**

***-- /clock generator***

***-- rst issue***

trig**:** **process** **is**

**begin**

**wait** **for** clk\_period**\***15**;**

rst **<=** '1'**;**

**wait** **for** clk\_period**\***2**;**

rst **<=** '0'**;**

**wait** **for** clk\_period**\***30**;**

**end** **process** trig**;**

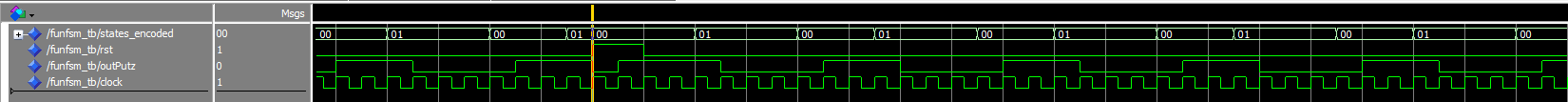
***-- /rst issue***

***-- /logic -----------------------------------------***

**end** funFSM\_arch\_tb**;**

***-- /architecture --------------------------------------***

***Wave output:***

****

Στον κέρσορα επιδεικνύεται η λειτουργία reset, τα states είναι κωδικοποιημένα «00» & «01».