ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΜΜΥ

ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ & ΥΛΙΚΟΥ

ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ ΓΙΑ ΤΟ ΜΑΘΗΜΑ:

ΗΡΥ 203 - ΠΡΟΧΩΡΗΜΕΝΗ ΛΟΓΙΚΗ ΣΧΕΔΙΑΣΗ

ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2020

Εργαστήριο 1

ΕΞΟΙΚΕΙΩΣΗ ΜΕ ΤΗ ΓΛΩΣΣΑ ΠΕΡΙΓΡΑΦΗΣ ΥΛΙΚΟΥ VHDL ΚΑΙ ΤΗΝ ΙΕΡΑΡΧΙΚΗ ΣΧΕΔΙΑΣΗ

ΕΚΠΟΝΗΣΗ: Καθ. Α. Δόλλας, Δρ. Κ. Παπαδημητρίου, Δρ. Ε. Σωτηριάδης, Μ. Κιμιωνής

ΣΥΝΕΡΓΑΤΕΣ: Δρ. Ε.Σωτηριάδης, Μ.Κιμιωνής, Ι.Πολογιώργη

ΕΚΔΟΣΗ : 11

Χανιά

Σκοπός - Βήματα

Είναι η εξοικείωση με τη γλώσσα περιγραφής υλικού VHDL(VHSICHardware Description Language) για απλά συνδυαστικά κυκλώματα, και με την πλήρη σχεδιαστική ροή απλών ψηφιακών συστημάτων. Με τη χρήση του εργαλείου Xilinx ISE θα δημιουργήσετε ένα project στο οποίο θα δημιουργήσετε αρχεία, θα κάνετε σύνθεση του κώδικα σας, και στη συνέχεια προσομοίωση(simulation). Επαληθεύσετε ότι το κύκλωμα λειτουργεί σύμφωνα με ότι παρατηρήσατε στην προσομοίωση. Για την προσομοίωση δημιουργήσετε δικό σας αρχείο testbench για να επαληθεύσετε όλες τις περιπτώσεις λειτουργίας της σχεδίασης. Τα επόμενα βήματα της σχεδιαστικής ροής, δηλαδή της μετάφρασης (translation), αποτύπωσης (map) και της τοποθέτησης και διασύνδεσης (place and route) και της δημιουργίας του αρχείου προγραμματισμού (.bitfile) της αναδιατασσόμενης συσκευής (FPGA), δεν θα τα κάνετε στο Εργαστήριο 1.

Για να σας βοηθήσουμε, έχουμε ετοιμάσει το σχετικό υλικό, το οποίο βρίσκεται στο Παράρτημα 1, και το οποίο περιέχει μία αντίστοιχη απλή σχεδίαση, με σωστή τεκμηρίωση (σχόλια), με testbench, και σωστή ιεραρχική δομή (η δομή του Project δηλαδή). Υλοποιήσετε τις σχεδιάσεις αυτές για να εξοικειωθείτε με τα εργαλεία, αλλά όταν γίνει αυτό δημιουργήσετε νέο projectκαι γράψετε κώδικα μόνοι σας.

Προεργασία (30%)

Πριν την προσέλευση σας στο εργαστήριο να έχετε:
α)υπολογίσει τις συναρτήσεις,
β)σχεδιάσει το σχετικό κύκλωμα (block diagram),
γ)υλοποιήσει σε γλώσσα VHDL τα παρακάτω κυκλώματα, με σωστή τεκμηρίωση (σχόλια) στα Αγγλικά
δ)δημιουργήσει testbench για κάθε κύκλωμα και να έχετε επαληθεύσει την λειτουργία των κυκλωμάτων σας.

Προφανώς μέρος της προεργασίας είναι η πραγματική εξοικείωση σας με τα εργαλεία (δηλαδή όχι απλά να τα δείτε αλλά να μπορείτε να τα χρησιμοποιήσετε). Κατά τη διάρκεια της εξέτασης θα πρέπει να δημιουργείτε νέο Project λόγω ασυμβατότητας των εργαλείων στο εργαστήριο με τις τρέχουσες εκδόσεις, επιπλέον θα πρέπει να μπορείτε να κάνετε απλές αλλαγές στο κύκλωμα ή στο testbench, ή να γράψετε ένα πολύ απλό κύκλωμα. Οι ασυμβατότητες δεν αφορούν τον κώδικα που έχετε γράψει, και ο οποίος σε καμία περίπτωση δεν επηρεάζεται από την έκδοση του εργαλείου, αλλά όλα τα υπόλοιπα αρχεία του Project που δημιουργούνται αυτόματα.

Κύκλωμα 1

Ζητούμενα

Να σχεδιάσετε και να υλοποιήσετε κύκλωμα που έχει εισόδους και εξόδους όπως στον Πίνακα 1.

Όνομα Σήματος	in/out	Πλάτος σε bit
A	in	1
B	in	1
C0	in	1
C1	in	1
C2	in	1
C3	in	1
C4	in	1
C5	in	1
Result	out	6

Πίνακας 1: Είσοδοι – έξοδοι κυκλώματος

Το κύκλωμα λειτουργεί ως εξής:

1)Το RESULT[0] είναι το αποτέλεσμα της A **NAND** B, αν το C0 είναι πατημένο (δηλαδή '1'), διαφορετικά είναι '0'.

2)Το RESULT[1] είναι το αποτέλεσμα της A **NOR** B αν το C1 είναι πατημένο (δηλαδή '1'),διαφορετικά είναι '0'.

3)Το RESULT[2] είναι το αποτέλεσμα της A **AND** B αν το C2 είναι πατημένο (δηλαδή '1'),διαφορετικά είναι '0'.

4)Το RESULT[3] είναι το αποτέλεσμα της A **XOR** B αν το C3 είναι πατημένο (δηλαδή '1'),διαφορετικά είναι '0'.

5)Το RESULT[4] είναι το αποτέλεσμα της (A **AND** B) **OR** (A' **AND** B') αν το C4 είναι πατημένο (δηλαδή '1'),διαφορετικά είναι '0'.

6)Το RESULT[5] είναι το αποτέλεσμα της (A'**AND** B) **XOR**(A' **OR** B) αν το C5 είναι πατημένο (δηλαδή '1'),διαφορετικά είναι '0'.

Κύκλωμα 2

Ζητούμενα

Να εξοικειωθείτε με το κύκλωμα ημιαθροιστή (HalfAdder) με τη χρήση λογικών πυλών, που σας δίνουμε στο Παράρτημα. Έπειτα με χρήση του ημιαθροιστή ως υποκύκλωμα, εξοικειωθείτε με τον πλήρη αθροιστή (FullAdder) 1 bit που επίσης σας δίνουμε στο Παράρτημα, δηλαδή με εισόδους A,B,Cin και εξόδους S,Cout. Στη συνέχεια με τη χρήση του πλήρους αθροιστή 1 bitνα σχεδιάσετε μόνοι σας έναν αθροιστή 2 bit.Κάνετε χρήση των εντολών component - portmap. Οι είσοδοι/έξοδοι του τελικού κυκλώματος που αθροίζει τελεστέους 2 Bitφαίνονται στον Πίνακα 2.

Όνομα	in/out	Πλάτος σε bit
A	in	2
B	in	2
Cin	in	1
RESULT	out	2
Cout	out	1

Πίνακας 2: Είσοδοι–έξοδοικυκλώματος

Διαδικασία στο εργαστήριο

Στη διάρκεια του εργαστηρίου πρέπει να κάνετε τα ακόλουθα

1) Δημιουργήσετε νέο project για κάθε ένα από τα κυκλώματα με τα αρχεία που έχετε φέρει μαζί σας.
2) Εξηγήσετε τα σημεία του κώδικα που θα σας ζητηθεί.
3) Δείξετε με το κατά το δυνατόν πιο εξαντλητικό τρόπο την ορθή λειτουργία του κυκλώματος στο simulation.
4) Κάντε τις αλλαγές που θα σας ζητηθούν στον κώδικα και επαναλάβετε το 3)

Παρατηρήσεις/Σημειώσεις

(1) Τα κυκλώματα της άσκησης αυτής είναι συνδυαστικά.
(2) Επαληθεύσετε τη λειτουργία κάθε υποκυκλώματος αφού το σχεδιάσετε ξεχωριστά, με χρήση ξεχωριστού testbench, δηλ. για κάθε πύλη φτιάξτε ένα testbench και προσομοιώσετε την. Το ίδιο κάνετε για τον ημιαθροιστή, και για τον πλήρη αθροιστή.
(3) Η σύνδεση των modules γίνεται ιεραρχικά, δηλ. έχουμε modules που συνδέονται μεταξύ τους μέσα σε άλλο moduleπου βρίσκεται σε 1 παραπάνω επίπεδο. Αυτό με τη σειρά του μπορεί να συνδέεται με άλλα modules, σε module που βρίσκεται σε ακόμη παραπάνω επίπεδο κ.ο.κ. Σκεφτείτε πως θα συνδέσετε half-adders για τη δημιουργία του full-adder!

Ακόμα και αν δεν συνδέονται μεταξύ τους τα modules, χρειάζεται να φτιάξετε 1 module σε 1 παραπάνω επίπεδο, το οποίο περιέχει τα χαμηλότερα σε ιεραρχία modules, ή, τα instances τους (αυτό συμβαίνει στις περισσότερες περιπτώσεις).
(4) Δημιουργήσετε top Level στο οποίο θα συνδέσετε όλα τα σήματα που θέλετε να «δείτε» στην προσομοίωση. Φτιάξτε το τελικό testbench, και συνδέστε το με το topLevel.

Παραδοτέα:

Πηγαίος κώδικας VHDL με σχόλια, κυματομορφές προσομοίωσης, παρουσίαση κυκλώματος, επίσης να δείξετε τη διαδικασία λύσης, τι χρησιμοποιήσατε από τη θεωρία και πως, π.χ. πίνακας καταστάσεων.

Βαθμολογία: Τα δύο κυκλώματα είναι ισοδύναμα βαθμολογικά

Διεξαγωγή εργαστηρίου	Προετοιμασία:30%
	Προσομοίωση: 70%

ΠΡΟΣΟΧΗ!

1) Η προεργασία να είναι σε ηλεκτρονική μορφή, σε αρχείο pdf. Το αρχείο πρέπει να το έχετε μαζί σας σε USB stick ή να μπορείτε να το κατεβάσετε εκείνη τη στιγμή από το διαδίκτυο.

2) Η έλλειψη προετοιμασίας οδηγεί σε απόρριψη.

3)Η διαπίστωση αντιγραφής σε οποιοδήποτε σκέλος της άσκησης οδηγεί στην απόρριψη από το σύνολο των εργαστηριακών ασκήσεων. Αυτό γίνεται οποιαδήποτε στιγμή στη διάρκεια του εξαμήνου. Ως αντιγραφή νοείται και μέρος της αναφοράς, π.χ. σχήματα.

ΚΑΛΗ ΕΠΙΤΥΧΙΑ!☺

Παράρτημα 1

Ενδεικτική λύση για το κύκλωμα 1 με διαφορετικές συναρτήσεις από τις ζητούμενες

```
library ieee;                --Libraries Declaration
use ieee.std_logic_1164.all;

entity equations is
port (    In0: in std_logic;
                  In1: in std_logic;       --Signal in and out declarations
                  Btn0: in std_logic;
                  Btn1: in std_logic;
                  Btn2: in std_logic;
                  Btn3: in std_logic;
                  Led: out std_logic_vector(5 downto 0)); --bus Declaration
end equations;

architecture dataflow of equations is   --Architecture implementation

begin

LED(0) <= (In0 NAND In1) AND Btn0;     --implementation of In0 AND In1 and Btn0
LED(1) <= (In0 XOR In1) AND Btn1;
LED(2) <= (In0 OR In1) AND Btn2;
LED(3) <= (NOT In0) AND Btn3;
LED(4) <= In0;
LED(5) <= In1;

end dataflow;
```

Ενδεικτική λύση για το testbench του κυκλώματος 1 με διαφορετικές συναρτήσεις από τις ζητούμενες

```
LIBRARY ieee;                                                    --Libraries Declaration
USE ieee.std_logic_1164.ALL;

ENTITY Equations_tb IS
END Equations_tb;

ARCHITECTURE behavior OF Equations_tb IS
   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT equations                --Module for testing declaration
   PORT(
        In0 : IN  std_logic;
        In1 : IN  std_logic;
        Btn0 : IN  std_logic;
        Btn1 : IN  std_logic;
        Btn2 : IN  std_logic;
        Btn3 : IN  std_logic;
        Led : OUT  std_logic_vector(5 downto 0)
       );
   END COMPONENT;
  --Inputs
  signal In0 : std_logic := '0';
  signal In1 : std_logic := '0';
  signal Btn0 : std_logic := '0';
  signal Btn1 : std_logic := '0';
  signal Btn2 : std_logic := '0';
  signal Btn3 : std_logic := '0';

--Outputs
  signal Led : std_logic_vector(5 downto 0);

BEGIN

-- Instantiate the Unit Under Test (UUT)
  uut: equations PORT MAP (                          --Module for testing Mapping
         In0 => In0,
         In1 => In1,
         Btn0 => Btn0,
         Btn1 => Btn1,
         Btn2 => Btn2,
         Btn3 => Btn3,
         Led => Led
        );
```

```
  -- Stimulus process
  stim_proc: process
  begin
    -- hold reset state for 100 ns.

       In0 <='0';            --assign values to ALL inputs
       In1 <='0';
       Btn0 <='1';
       Btn1 <='1';
       Btn2 <='1';
       Btn3 <='1';
       wait for 100 ns;    --wait for 100ns to see the results
       In0 <='0';            --assign new values to ALL inputs
       In1 <='1';
       Btn0 <='1';
       Btn1 <='1';
       Btn2 <='1';
       Btn3 <='1';
       wait for 100 ns;    --wait for 100ns to see the results
       In0 <='1';
       In1 <='0';
       Btn0 <='1';
       Btn1 <='1';
       Btn2 <='1';
       Btn3 <='1';
       wait for 100 ns;
       In0 <='1';
       In1 <='1';
       Btn0 <='1';
       Btn1 <='1';
       Btn2 <='1';
       Btn3 <='1';
--      wait for <clock>_period*10;

     -- insert stimulus here

     wait;
  end process;

END;
```

Ενδεικτική λύση για το κύκλωμα 2 Half Adder

```
library ieee;                   --Libraries Declaration
use ieee.std_logic_1164.all;

entity half_adder is
port (     x: in std_logic;          --Signal in and out declarations
                    y: in std_logic;
                    s: out std_logic;
                    c: out std_logic);
end half_adder;

architecture dataflow of half_adder is          --Architecture implementation

begin

s <= x xor y;                                                   --implementation of  Half
Adder
c <= x and y;

end dataflow;
```

Ενδεικτική λύση για το κύκλωμα 2 Full Adder

```
library ieee;                                   --Libraries Declaration
use ieee.std_logic_1164.all;

entity full_adder is
port (    x: in std_logic;              --Signal in and out declarations
                y: in std_logic;
                z: in std_logic;
                s: out std_logic;
                c: out std_logic);
end full_adder;

architecture struct_dataflow of full_adder is   --Architecture implementation

component half_adder                                    --component declarations
port (    x: in std_logic;
                y: in std_logic;
                s: out std_logic;
                c: out std_logic);
end component;

signal hs, hc, tc: std_logic;                   --internal signals decarations

begin

HA1: half_adder                                                 --port mapping (signals
conection)      of 1st HA
port map (
        x => x,
        y => y,
        s => hs,
        c => hc);

HA2: half_adder                                                 --port mapping (signals
conection)      of 2nd HA
port map (
        x => hs,
        y => z,
        s => s,
        c => tc);

c <= tc or hc;                                                  --carry implementation

end struct_dataflow;
```

Ενδεικτική λύση για το testbench του Full Adder

```
LIBRARY ieee;                          --Libraries Declaration
USE ieee.std_logic_1164.ALL;


ENTITY Full_Adder_tb IS
END Full_Adder_tb;

ARCHITECTURE behavior OF Full_Adder_tb IS

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

   COMPONENT full_adder                        --Module for testing declaration
   PORT(
        x : IN  std_logic;
        y : IN  std_logic;
        z : IN  std_logic;
        s : OUT  std_logic;
        c : OUT  std_logic
       );
   END COMPONENT;


  --Inputs
  signal x : std_logic := '0';
  signal y : std_logic := '0';
  signal z : std_logic := '0';

          --Outputs
  signal s : std_logic;
  signal c : std_logic;


BEGIN

          -- Instantiate the Unit Under Test (UUT)
  uut: full_adder PORT MAP (
         x => x,
         y => y,
         z => z,
         s => s,
         c => c
        );

  -- Stimulus process
  stim_proc: process
  begin

                      x <='0';--assign values to ALL inputs
       y <='0';
       z <='0';
                    wait for 100 ns;          --wait for 100ns to see the results
```

```
                x <='0';                        --assign new values to ALL inputs
        y <='0';
        z <='1';
                wait for 100 ns;        --wait for 100ns to see the results
                x <='0';                        --assign new values to ALL inputs
        y <='1';
        z <='0';
                wait for 100 ns;
                x <='0';
        y <='1';
        z <='1';
                wait for 100 ns;
                x <='1';
        y <='0';
        z <='0';
                wait for 100 ns;
                x <='1';
        y <='0';
        z <='1';
                wait for 100 ns;
                x <='1';
        y <='1';
        z <='0';
                wait for 100 ns;
                x <='1';
        y <='1';
        z <='1';
    -- insert stimulus here

    wait;
  end process;

END;
```