Φυλλάδιο

Εργαστηριακών Ασκήσεων

Μιχάλης Ψαράκης

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ - ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΜΑΘΗΜΑ: «ΣΧΕΔΙΑΣΗ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ»

Εισαγωγή

Οι εργαστηριακές ασκήσεις του μαθήματος "Σχεδίαση Υπολογιστικών Συστημάτων» θα σας βοηθήσουν στην εκμάθηση του εργαλείου Xilinx Vivado για την εισαγωγή σχεδίασης (design entry), την προσομοίωση (simulation), την σύνθεση (synthesis) ψηφιακών κυκλωμάτων και την υλοποίηση τους σε εκπαιδευτικές πλατφόρμες FPGA (Field Programmable Gate Arrays). Οι στόχοι του εργαστηρίου είναι:

- να εξοικειωθείτε με ένα ολοκληρωμένο περιβάλλον σχεδίασης, προσομοίωσης και σύνθεσης κυκλωμάτων (σημείωση: αυτό το περιβάλλον θα χρησιμοποιήσετε και για να υλοποιήσετε την εργασία του μαθήματος).
- να διδαχτείτε μέσα από μια σειρά εργαστηριακών ασκήσεων με κλιμακούμενη πολυπλοκότητα την σχεδίαση υπολογιστικών συστημάτων με χρήση της γλώσσας περιγραφής υλικού VHDL.
- να έρθετε σε επαφή με μια εκπαιδευτική και αναπτυξιακή πλατφόρμα υλικού και χρησιμοποιώντας την τεχνολογία προγραμματιζόμενης λογικής (programmable logic) να υλοποιήσετε τα κυκλώματά σε συσκευές FPGAs (Field Programmable Gate Arrays).

Οι εργαστηριακές ασκήσεις χωρίζονται στις παρακάτω ενότητες:

- Εισαγωγή στο εργαλείο: Θα εξοικειωθείτε με τη χρήση του εργαλείου Xilinx Vivado σχεδιάζοντας και προσομοιώντας στοιχειώδη κυκλώματα.
- Εξοικείωση με την χρήση της εκπαιδευτικής πλακέτας FPGA: Θα χρησιμοποιήσετε μια εκπαιδευτική πλατφόρμα (Basys 3 Artix-7 FPGA board) και θα εξοικειωθείτε με την χρήση της υλοποιώντας κάποια στοιχειώδη κυκλώματα.
- Υλοποίηση απλών κυκλωμάτων: Θα ασχοληθείτε με την σχεδίαση, προσομοίωση και υλοποίηση στην πλακέτα FPGA απλών συνδυαστικών (π.χ. πολυπλέκτες, αποκωδικοποιητές, αθροιστές, κτλ.) και ακολουθιακών κυκλωμάτων (π.χ. μετρητές, καταχωρητές ολίσθησης, κτλ.).
- Υλοποίηση σύνθετων κυκλωμάτων: Θα ασχοληθείτε με την σχεδίαση, προσομοίωση και υλοποίηση στην πλακέτα FPGA πιο σύνθετων κυκλωμάτων (π.χ. μηχανές πεπερασμένων καταστάσεων, μνήμες, κτλ.)

Το εργαλείο Xilinx Vivado υποστηρίζει διάφορες γλώσσες περιγραφής υλικού (hardware description languages). Οι εργαστηριακές ασκήσεις επικεντρώνονται στην γλώσσα περιγραφής υλικού VHDL.

Σημείωση: Το παρόν φυλλάδιο αφορά την έκδοση του εργαλείου Xilinx Vivado 18.3.

Εργαστηριακή Άσκηση 1: Εισαγωγή στο εργαλείο

Σε αυτήν την εργαστηριακή άσκηση θα εξοικειωθείτε με το περιβάλλον Xilinx Vivado. Πιο συγκεκριμένα θα μάθετε:

- Πως να σχεδιάζετε κυκλώματα με τη χρήση μιας γλώσσας περιγραφής υλικού (π.χ. VHDL)
- Πώς να προσομοιώνετε το κύκλωμα (functional simulation) με χρήση του προσομοιωτή Vivado.

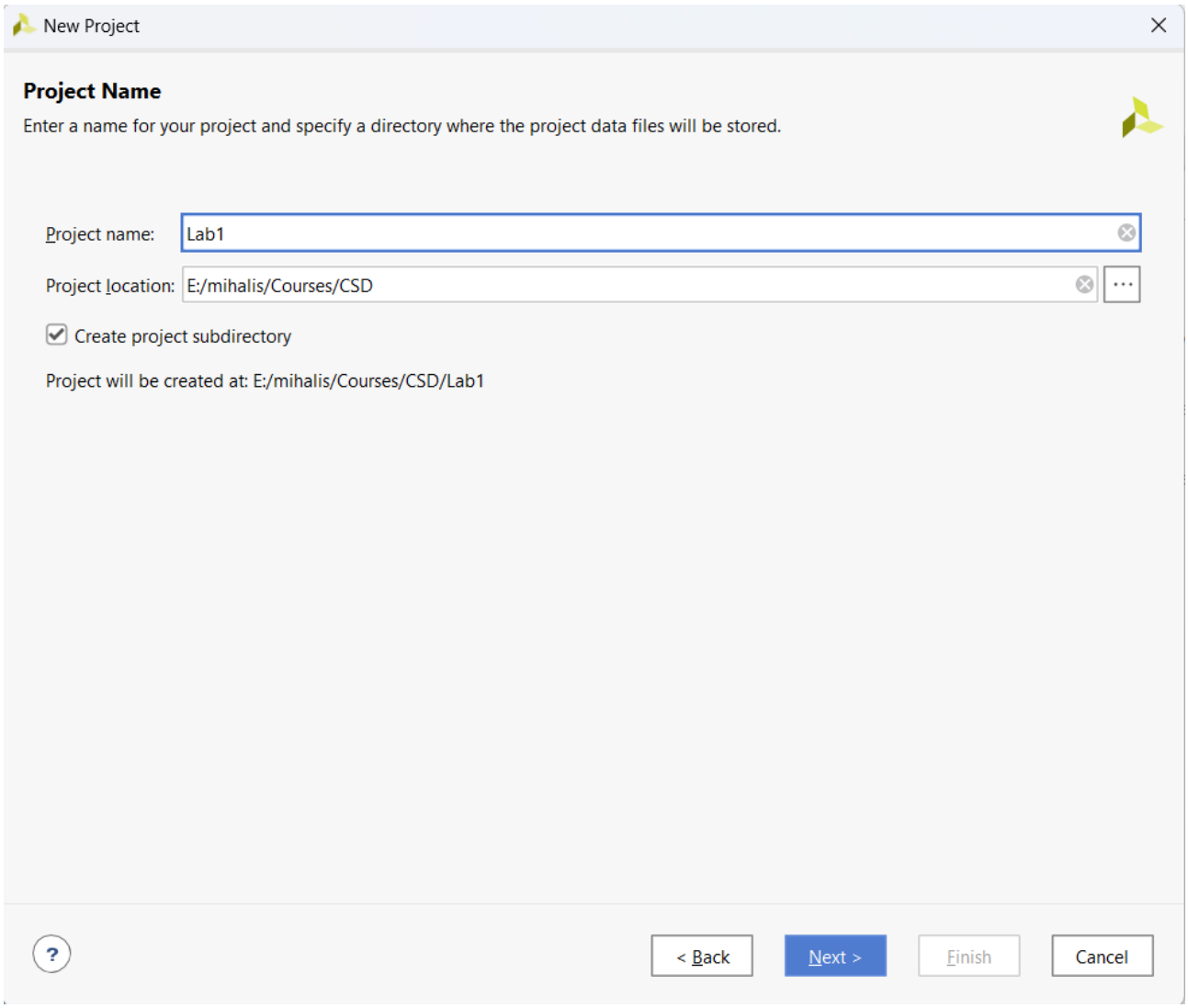
Άσκηση 1: 4-bit parity generator (με VHDL)

Το κύκλωμα που θα υλοποιήσετε σε αυτήν την εργαστηριακή άσκηση είναι ένα απλό κύκλωμα υπολογισμού της άρτιας ισοτιμίας ενός μηνύματος των 4-bit.

1.1 Δημιουργία νέου έργου (new project)

Επιλέξτε All Programs→Xilinx Design Tools→Vivado 2018.3→Vivado 2018.3. Επιλέξτε Create New Project. Θα εμφανιστεί το πρώτο πλαίσιο διαλόγου του New Project, όπως φαίνεται στην Εικόνα 1.

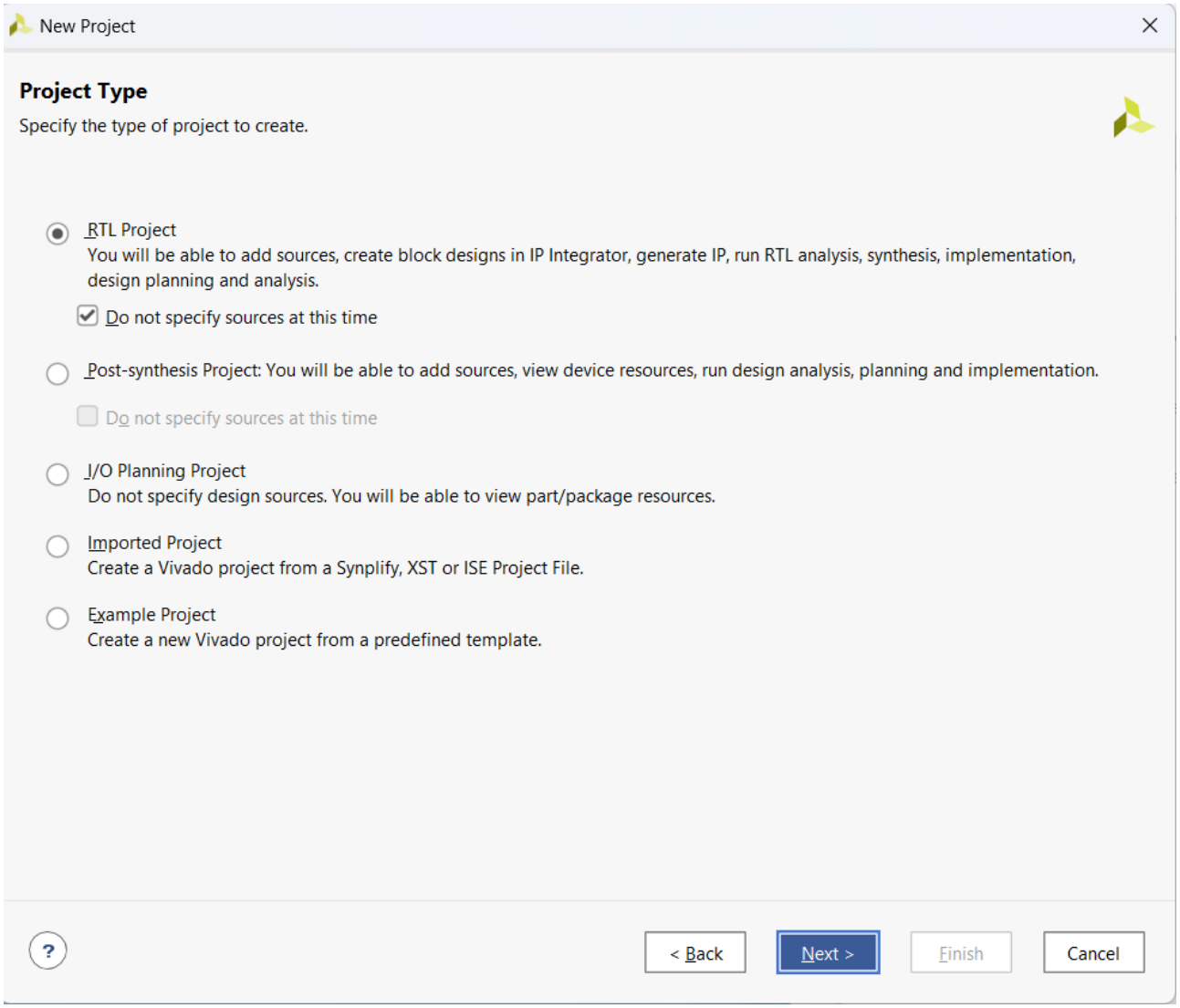
Το πλαίσιο διαλόγου σας προτρέπει να εισάγετε το όνομα και τον κατάλογο του έργου, όπως φαίνεται στην Εικόνα 1. Αφού συμπληρώσετε τα στοιχεία, πατήστε Next.



Εικόνα 1: New Project – Project name (1 από 4)

Σημείωση: Μην χρησιμοποιείτε ονόματα αρχείων ή φακέλων που περιέχουν διαστήματα και ονόματα φακέλων στα ελληνικά.

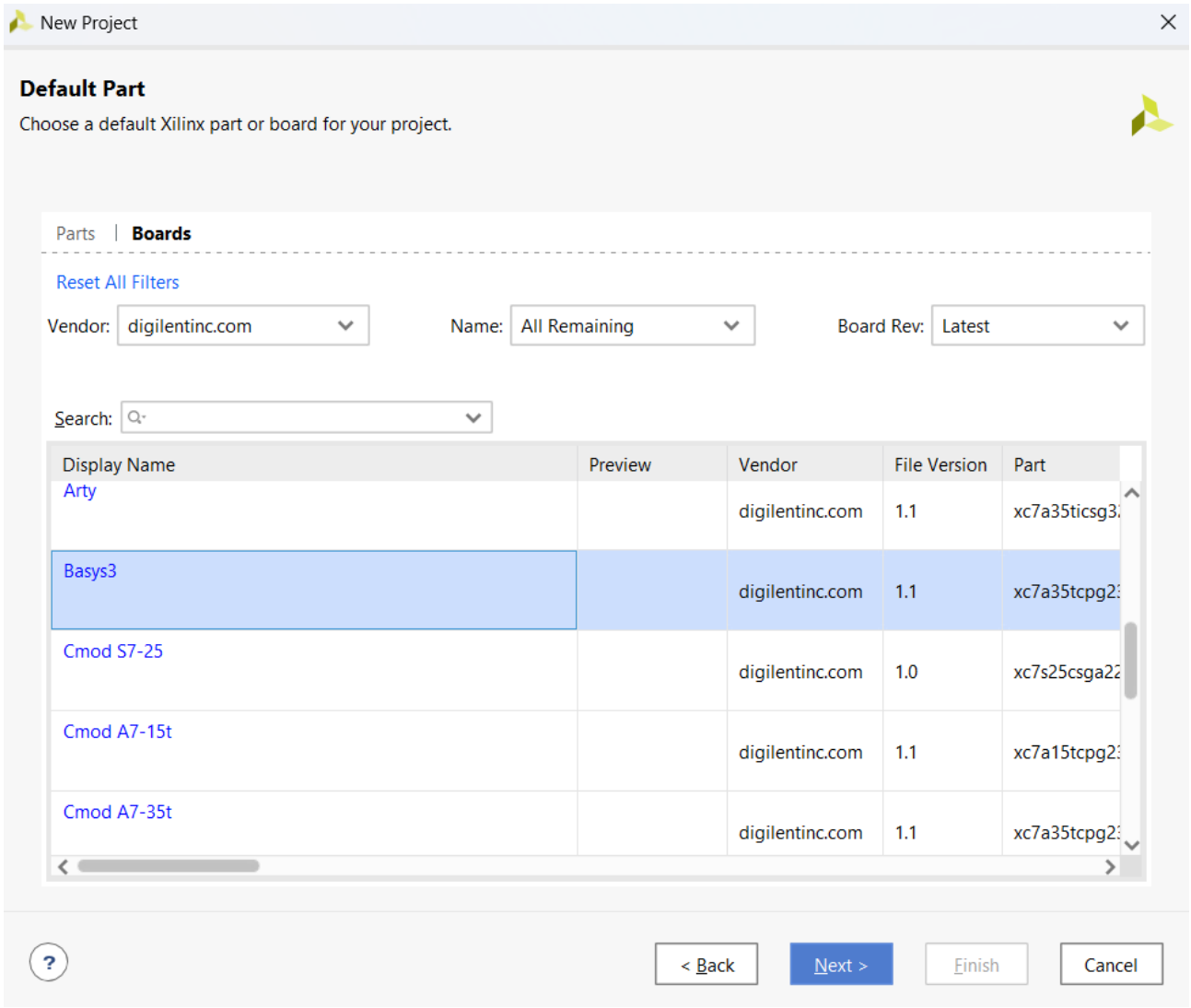
Το επόμενο πλαίσιο διαλόγου σας επιτρέπει να επιλέξετε τον τύπο του έργου. Επιλέξτε τύπο RTL Project και ενεργοποιήστε την επιλογή Do not specify sources at this time, όπως φαίνεται στην Εικόνα 3, και πατήστε Next.



Εικόνα 2: New Project – Project type (2 από 4)

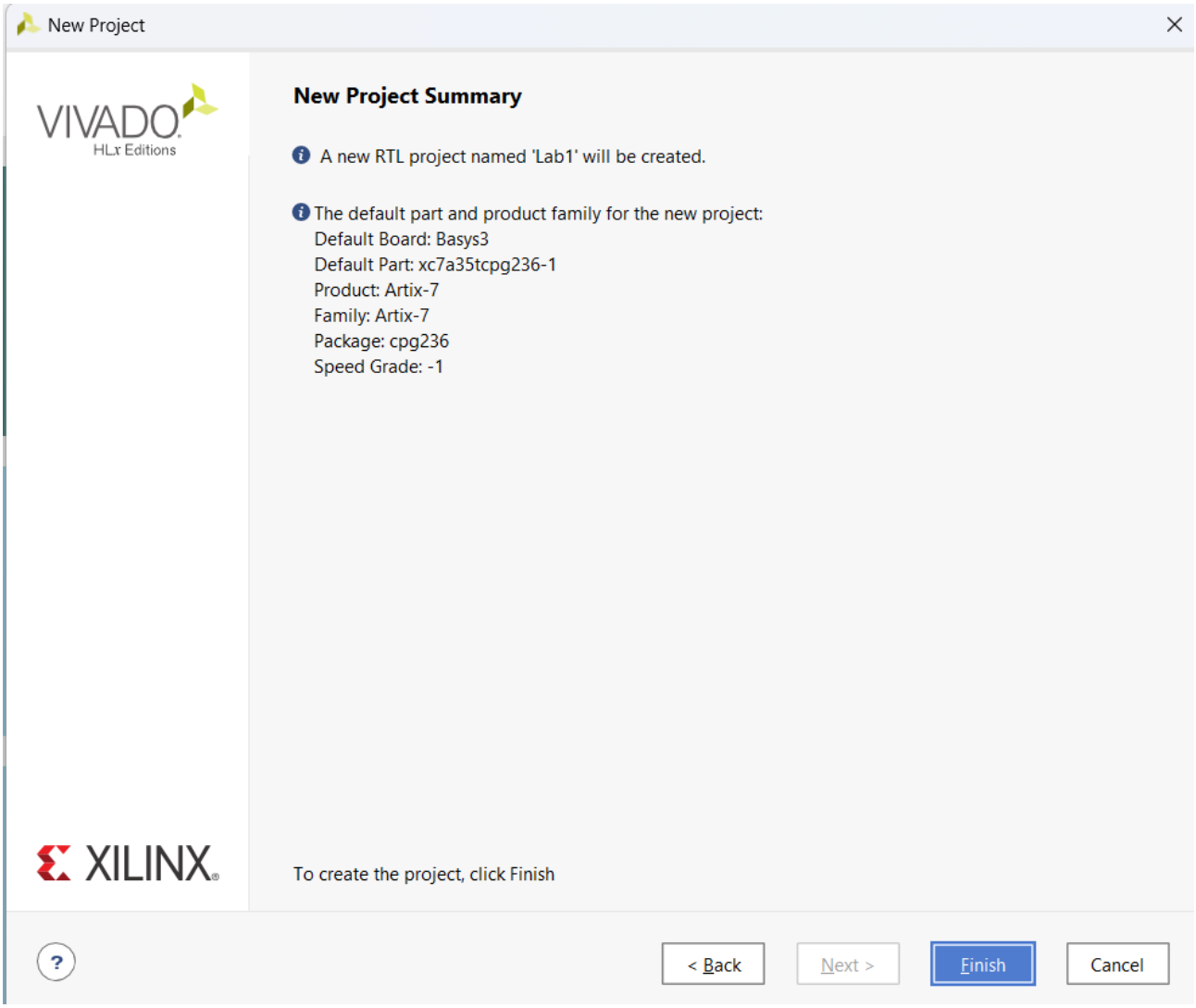
Το επόμενο πλαίσιο διαλόγου σας επιτρέπει καθορίσετε τον τύπο της συσκευής FPGA που θα χρησιμοποιήσετε. Επιλέξτε Boards και στο πεδίο Filter/Preview επιλέξτε Vendor: digilentinc.com. Έπειτα, επιλέξτε Display Name: Basys3, όπως στην Εικόνα 3. Αφού επιλέξετε τον τύπο της συσκευής, πατήστε Next.

Σημείωση: Εάν οι πλακέτες της εταιρείας (vendor) digilentinc.com δεν είναι διαθέσιμες, επιλέξτε Parts και έπειτα στο πεδίο Filter επιλέξτε Family: Artix-7, στο πεδίο Package: cpg236, και στο πεδίο Speed grade: -1. Τέλος, επιλέξτε την συσκευή xc7a35tcpg236 -1.



Εικόνα 3: New Project – Default Part (3 από 4): Επιλογή πλακέτας Zybo

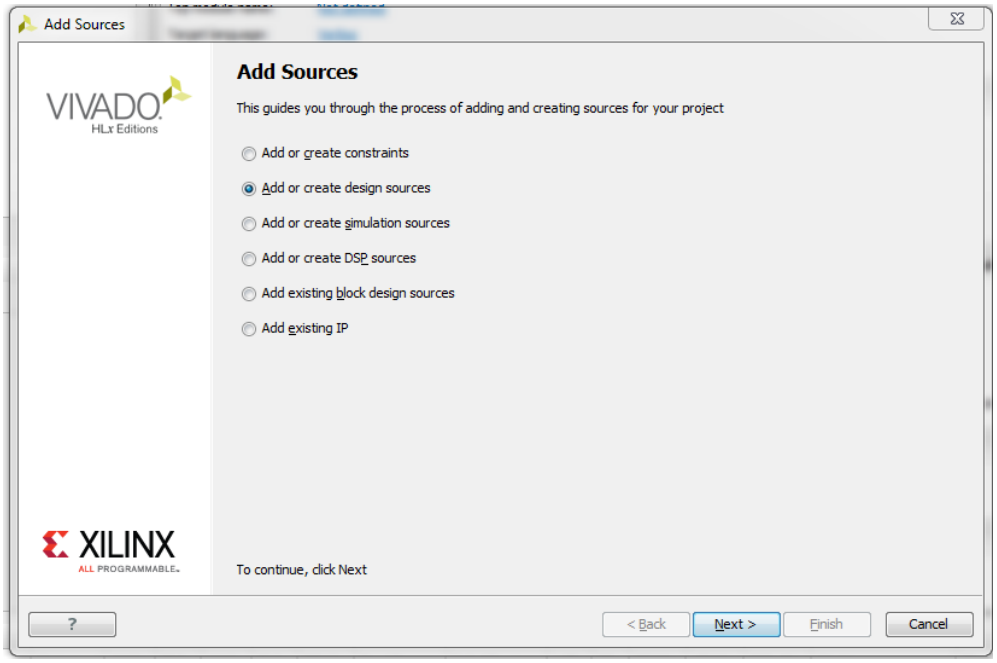
Το τελικό πλαίσιο διαλόγου στη διαδικασία δημιουργίας νέου έργου, που φαίνεται στην Εικόνα 4, παρέχει μια περίληψη του έργου που το Vivado θα δημιουργήσει βασισμένο στις ρυθμίσεις σας. Ελέγξτε την περίληψη για να σιγουρευτείτε ότι ταιριάζει με ότι φαίνεται στην Εικόνα 4. Εάν όχι, πατήστε Back για να διορθώσετε οποιοδήποτε λάθος. Διαφορετικά, πατήστε Finish για να ολοκληρώσετε τη διαδικασία.



Εικόνα 4: New Project – Summary (4 από 4)

1.2 Εισαγωγή σχεδίασης

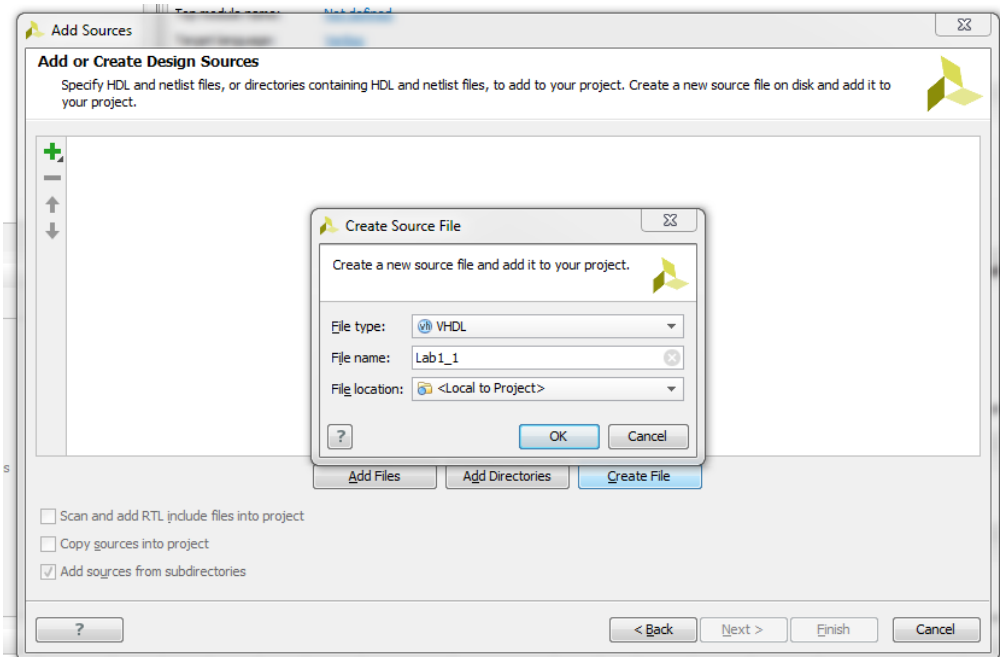
Σε αυτό το σημείο, το έργο που έχει δημιουργηθεί δεν περιέχει κανένα πηγαίο αρχείο. Δημιουργήστε ένα νέο πηγαίο αρχείο για τη σχεδίαση του κυκλώματος υπολογισμού άρτιας ισοτιμίας. Στο παράθυρο Project Manager επιλέξτε Add Sources. Στα επόμενα βήματα θα πρέπει να καθορίσετε το πηγαίο αρχείο. Το πρώτο από τα νέα κουτιά διαλόγου, σας ζητάει να καθορίσετε το τύπο του αρχείου, όπως φαίνεται στην Εικόνα 5. Επιλέξτε Add or create design sources.



Εικόνα 5: Add Sources (1 από 3)

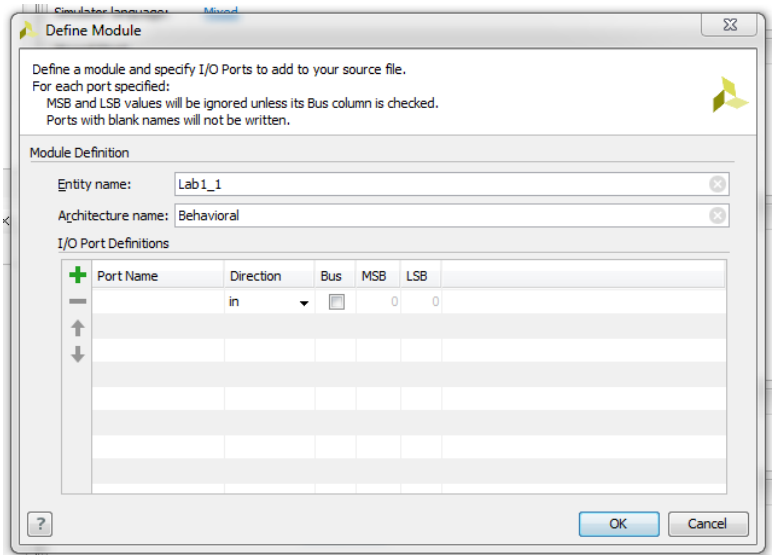
Στην συνέχεια μπορείτε να επιλέξετε να προσθέσετε νέο αρχείο (Add Files), νέο κατάλογο όπου περιέχονται ένα ή περισσότερα πηγαία αρχεία (Add Directories) ή να δημιουργήσετε νέο αρχείο (Create File). Επιλέξτε Create File.

Το επόμενο πλαίσιο διαλόγου σας καλεί να δηλώσετε τον τύπο και το όνομα του νέου αρχείου. Επιλέξτε τύπο VHDL και δώστε όνομα Lab1_1, όπως στην Εικόνα 6.



Εικόνα 6: Add Sources (2 από 3)

Το επόμενο παράθυρο σας επιτρέπει (προαιρετικά) να καθορίσετε τις θύρες (ports) της μονάδας. Αυτό μπορεί επίσης να γίνει στον επεξεργαστή κειμένου, κατά την επεξεργασία της μονάδας, έτσι αγνοήστε το σε αυτή τη φάση. Απλά επιβεβαιώστε ότι οι ρυθμίσεις ταιριάζουν με εκείνες που φαίνονται στην Εικόνα 7 και πατήστε Next.



Εικόνα 7: Add Sources (3 από 3)

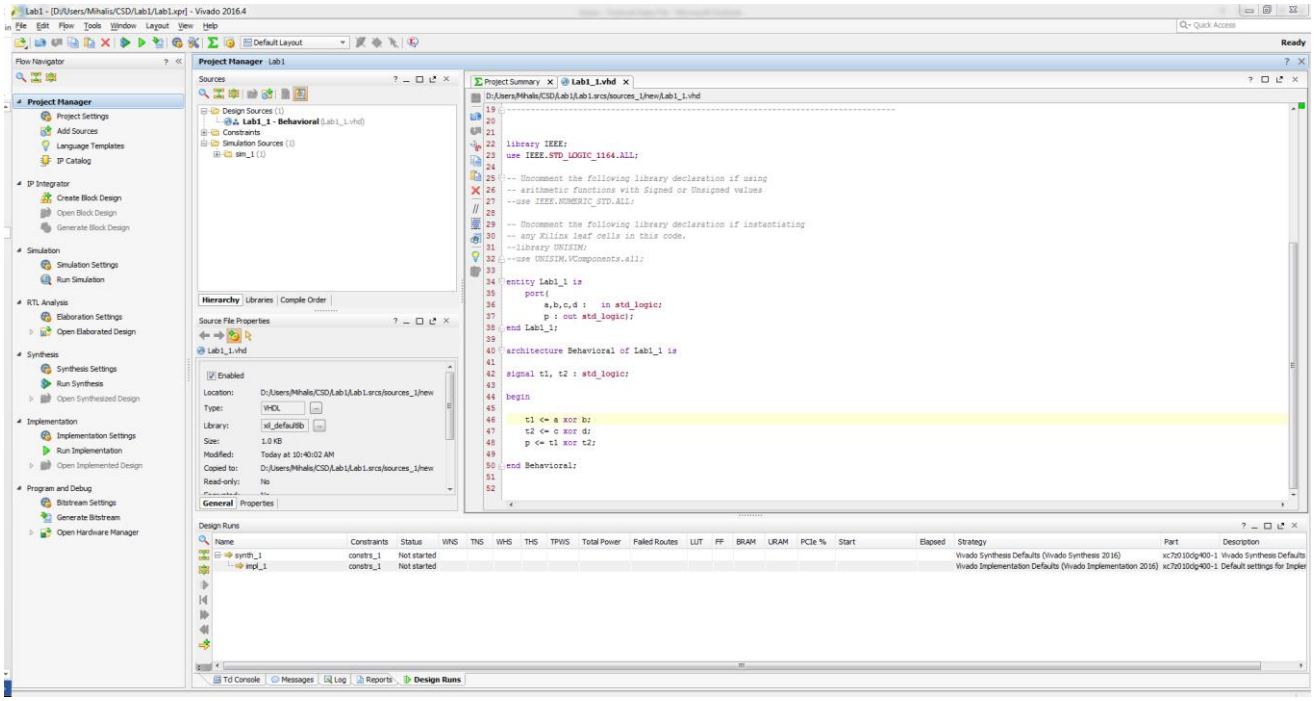
Στον επεξεργαστή κειμένου, μερικές από τη βασικές δομές του VHDL αρχείου είναι ήδη γραμμένες. Οι λέξεις κλειδιά (keywords) της γλώσσας απεικονίζονται με μπλε χρώμα, οι τύποι δεδομένων με κόκκινο, τα σχόλια με γκρι, και οι τιμές με μαύρο. Αυτή η κωδικοποίηση ανάλογα με το χρώμα ενισχύει την αναγνωσιμότητα του VHDL αρχείου και την εύρεση τυπογραφικών λαθών. Τώρα, εισάγετε την περιγραφή του κυκλώματος υπολογισμού άρτιας ισοτιμίας.

Σημείωση: Μπορείτε να κατεβάσετε το παρακάτω μοντέλο (Lab1_1.vhd) από την ιστοσελίδα του μαθήματος (περιέχεται στο Lab1.zip).

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Lab1_1 is
   port(  a,b,c,d :     in std_logic;
          p :    out std_logic);
end Lab1_1;

architecture Behavioral of Lab1_1 is

signal t1, t2 : std_logic;

begin

   t1 <= a xor b;
   t2 <= c xor d;
   p <= t1 xor t2;
```

```
end Behavioral;
```

Σε αυτό το σημείο, πρέπει να καταλήξετε με ένα παράθυρο που μοιάζει με αυτό που φαίνεται στην Εικόνα 8. Μόλις τελειώσετε, αποθηκεύστε το αρχείο και κλείστε το παράθυρο. Υπάρχουν επιλογές στο κυρίως μενού για να σώσετε είτε μεμονωμένα αρχεία είτε όλο το έργο.



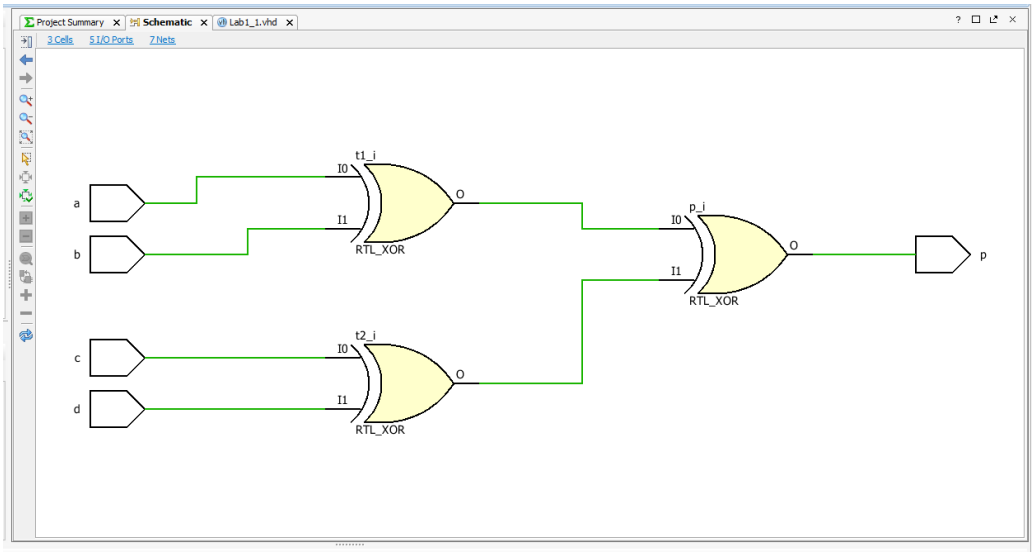
Εικόνα 8: Ολοκληρωμένη σχεδίαση

1.3 Προσομοίωση (Simulation)

Μπορείτε να κατεβάσετε το μοντέλο (Lab1_bench.vhd) από την ιστοσελίδα του μαθήματος (περιέχεται στο Lab1.zip).

1.4 Ανάλυση κυκλώματος (RTL Analysis)

Στο παράθυρο RTL Analysis επιλέξτε Open Elaborated Design. Παρατηρήστε το σχηματικό διάγραμμα του κυκλώματα ισοτιμίας (αποτελείται από 3 πύλες XOR των 2-εισόδων).



Εικόνα 9: Σχηματικό διάγραμμα

Εργαστηριακή Άσκηση 2: Προγραμματισμός της συσκευής FPGA

Σε αυτήν την εργαστηριακή άσκηση θα εξοικειωθείτε με την χρήση και τον προγραμματισμό της εκπαιδευτικής πλακέτας FPGA. Πιο συγκεκριμένα θα μάθετε:

- Τα χαρακτηριστικά της εκπαιδευτικής και αναπτυξιακής πλακέτας (Digilent Basys-3 FPGA board) που θα χρησιμοποιήσετε στο εργαστήριο
- Πώς να δημιουργείτε ένα αρχείο προγραμματισμού FPGA
- Πώς να προγραμματίζετε μια συσκευή FPGA

Άσκηση 1: Υλοποίηση του κυκλώματος 4-bit parity generator στην πλακέτα FPGA

Θα υλοποιήσετε το κύκλωμα που υπολογισμού της άρτιας ισοτιμίας ενός μηνύματος των 4-bit (που σχεδιάσατε στην εργαστηριακή άσκηση 1) στην πλακέτα FPGA. Οπότε ανοίξτε το έργο της εργαστηριακής άσκησης 1 (Lab1 project).

Στην εργαστηριακή άσκηση 1 ασχοληθήκατε με τα βήματα: εισαγωγή σχεδίασης, προσομοίωση και ανάλυση κυκλώματος. Τα επόμενα βήματα ώστε να προγραμματίσετε την συσκευή FPGA είναι: σύνθεση (synthesis), υλοποίηση (implementation) και προγραμματισμός FPGA και αποσφαλμάτωση (Program and Debug).

1.1 Εκπαιδευτική πλακέτα Zybo

Πρώτα όμως εξοικειωθείτε με την πλακέτα FPGA που θα χρησιμοποιήσετε. Στην ιστοσελίδα του εργαστηρίου μπορείτε να βρείτε εγχειρίδιο χρήσης της πλακέτας.

Εικόνα 10 Basys-3 board

Η πλακέτα Basys-3 περιέχει τα εξής (στα πλαίσια του εργαστηρίου θα χρησιμοποιήσετε ότι είναι με bold):

- **Artix-7 35T FPGA**
- **16 user switches**
- **16 user LEDs**
- **5 user pushbuttons**
- **4-digit 7-segment di**splay
- Three Pmod ports
- Pmod for XADC signals
- 12-bit VGA output
- USB-UART Bridge
- Serial Flash
- **Digilent USB-JTAG port for FPGA programming and communication**
- USB HID Host for mice, keyboards and memory sticks
- Και άλλα συστατικά που δεν θα χρησιμοποιηθούν στα πλαίσια του εργαστηρίου

1.2 Σύνθεση σχεδίασης (design synthesis)

Μετά την προσομοίωση και την επαλήθευση ότι το κύκλωμά σας λειτουργεί σωστά, το επόμενο βήμα είναι να χρησιμοποιήσετε ένα εργαλείο σύνθεσης (synthesis tool) για να μετασχηματίσετε την περιγραφή σας σε μια λίστα συνδέσεων (netlist). Μια λίστα συνδέσεων (netlist) είναι μια σχηματική αναπαράσταση που μπορεί να διαβαστεί από αυτόματα εργαλεία.

Στο παράθυρο Synthesis επιλέξτε Run Synthesis.

Όταν ολοκληρωθεί η διαδικασίας της σύνθεσης επιλέξτε View Report. Στο παράθυρο Reports επιλέξτε Utilization Report και δείτε τα resources του FPGA device που χρησιμοποιεί το κύκλωμά σας. Πόσα Slice LUTs, πόσα Slice Registers, πόσα Memory blocks (Block RAMs), πόσα DSPs, πόσα IOB (Input-Output Blocks)?

Έπειτα στο παράθυρο Synthesis επιλέξτε Schematic και δείτε το σχηματικό διάγραμμα του κυκλώματός σας με χρήση των programmable resources του FPGA. Επιλέξτε το LUT4 και δείτε το Truth Table.

Κλείστε το Synthesized Design.

1.3 Υλοποίηση σχεδίασης (design implementation)

Η υλοποίηση της σχεδίασης είναι η ακολουθία γεγονότων που μεταφράζει τη λίστα συνδέσεων της σχεδίασης που έχετε ήδη συνθέσει (synthesized design netlist) σε ένα αρχείο προγραμματισμού για τη συσκευή FPGA. Η περιγραφή του κυκλώματός σας, που έχετε συνθέσει τώρα, έχει έναν αριθμό θυρών (ports) στο υψηλότερο επίπεδο (top level). Τα εργαλεία υλοποίησης (implementation tools) πρέπει να γνωρίζουν πώς θα αναθέσουν τις θύρες στο υψηλότερο επίπεδο της σχεδίασής σας στους φυσικούς ακροδέκτες (pins) του FPGA, οι οποίοι συνδέονται με διάφορους πόρους της πλακέτας. Εάν δεν ορίσετε ρητές αναθέσεις, τα εργαλεία θα ορίσουν τυχαία τους ακροδέκτες για σας. Προφανώς, αυτό είναι μια κακή ιδέα αφού οι τυχαίες αναθέσεις θα είναι λανθασμένες.

Το υψηλότερο επίπεδο της σχεδίασης του παραδείγματος έχει 4 θύρες εισόδου (a, b, c, d), και μια θύρα εξόδου (p). Άρα, θέλουμε να **έχουμε 4 διακόπτες, SW0, SW1, SW2 και SW3**, που συνδέονται με τις εισόδους. Επιπλέον, θέλουμε την έξοδο να συνδέεται με μια ενδεικτική λυχνία (LED) έτσι ώστε να μπορούμε να την παρατηρήσουμε – το **LD0** είναι κατάλληλο για αυτόν τον σκοπό.

Εάν επιθεωρήσετε την πάνω πλευρά της πλακέτας σας, θα παρατηρήσετε ότι σχεδόν κάθε πόρος έχει σχολιαστεί με κάποιο κείμενο που προσδιορίζει με ποιους ακροδέκτες του FPGA συνδέεται. Αυτές οι

πληροφορίες είναι επίσης διαθέσιμες στον οδηγό χρήσης της πλακέτας (User Guide). Προσπαθήστε να προσδιορίσετε στην πλακέτα σας ποιοι ακροδέκτες του FPGA χρησιμοποιούνται για τα SW0, SW1, SW2, SW3 και LD0, και ελέγξτε έπειτα τα αποτελέσματά σας με αυτά που παρουσιάζονται παρακάτω:

SW0 → FPGA Pin V17

SW1 → FPGA Pin V16

SW2 → FPGA Pin W16

SW3 → FPGA Pin W17

LD0 → FPGA Pin U16

Έχετε τώρα αρκετές πληροφορίες για να δημιουργήσετε αυτό που ονομάζεται αρχείο περιορισμών του χρήστη **(user constraint file), ή UCF**. Αυτό το αρχείο περιέχει τους περιορισμούς της σχεδίασης που δεν καθορίσατε στην περιγραφή VHDL, όπως οι περιορισμοί θέσης των ακροδεκτών (pin location) και απόδοσης της σχεδίασης (design performance). Είναι βολικό να παρασχεθούν σε ένα UCF παρά στην VHDL περιγραφή. Για παράδειγμα, εάν κάνετε ένα λάθος στις αναθέσεις των ακροδεκτών, δεν χρειάζεται να επιστρέψετε και να επανασυνθέσετε το κύκλωμά σας.

Κατεβάστε από το gunet το XDC constraint file της πλακέτας σας (Basys-3-Master.xdc).

Μπορείτε να προσθέσετε ένα UCF στο έργο χρησιμοποιώντας την ίδια διαδικασία που χρησιμοποιήσατε για την προσθήκη της σχεδίασης. Στο παράθυρο Project Manager επιλέξτε Add Sources → Add or create constraints. Στο επόμενο παράθυρο επιλέξτε Add Files και διαλέξτε το Basys-3-Master.xdc file.

Ανοίξτε το xdc file και uncomment τις γραμμές που αναφέρονται στα 4 switches που θέλετε να χρησιμοποιήσετε και το ένα LED. Δώστε στα αντίστοιχα σήματα τα ονόματα εισόδου και εξόδου του κυκλώματός σας, δηλαδή a αντί για sw[0], b αντί για sw[1], c αντί για sw[2], d αντί για sw[3], p αντί για led[0],

Τώρα που έχετε ένα αρχείο περιορισμών στο έργο σας, μπορείτε να υλοποιήσετε τη σχεδίαση. Στο παράθυρο Implementation επιλέξτε Run Implementation. Όταν ολοκληρωθεί η διαδικασίας της υλοποίησης επιλέξτε View Report.

1.4 Προγραμματισμός του FPGA (Program and Debug)

Σε αυτό το σημείο, είστε έτοιμοι να προγραμματίσετε το FPGA με τη σχεδίασή σας.

Στο παράθυρο Program and Debug επιλέξτε Generate Bitstream. Όταν ολοκληρωθεί η διαδικασία επιλέξτε View Reports.

Συνδέστε την πλακέτα σας μέσω του USB καλωδίου. Ανοίξτε την τροφοδοσία της πλακέτας (power off). Επιλέξτε Open Hardware Manager. Επιλέξτε Open Target. Επιλέξτε Program Device.

Τώρα, μπορείτε να δοκιμάσετε τη σχεδίασή σας στο υλικό. Εντοπίστε τους διακόπτες SW0-SW3 στην πλακέτα, και εξετάστε το κύκλωμά σας δοκιμάζοντας τους 16 πιθανούς συνδυασμούς των τιμών των διακοπτών και παρατηρώντας το LD0. Το κύκλωμα σας συμπεριφέρεται όπως αναμένετε; Εάν όχι, ζητήστε τη βοήθεια του καθηγητή. Εάν λειτουργεί σωστά, έχετε ολοκληρώσει με επιτυχία την άσκηση.

Εργαστηριακή Άσκηση 3: Συνδυαστικά κυκλώματα

Σε αυτήν την εργαστηριακή άσκηση θα ασχοληθείτε με την σχεδίαση συνδυαστικών κυκλωμάτων. Πιο συγκεκριμένα θα σχεδιάσετε:

- Ένα κύκλωμα πλειοψηφίας των 3 εισόδων
- Έναν αθροιστή των 4-bit
- Έναν αποκωδικοποιητή 3-σε-8
- Έναν συγκριτή δύο αριθμών των 4-bit
- Έναν μετατροπέα δυαδικού αριθμού σε BCD αριθμό

Άσκηση 1: Κύκλωμα πλειοψηφίας των 3 εισόδων

Σχεδιάστε ένα κύκλωμα πλειοψηφίας των 3 εισόδων χρησιμοποιώντας τους διακόπτες SW2-SW0 ως εισόδους και το LED0 ως έξοδο.

- Δημιουργήστε ένα νέο project.
- Δημιουργήστε ένα νέο αρχείο (vhdl module) με όνομα lab3_majority.vhd.
- Η οντότητα της μονάδας είναι η εξής:

```
entity lab3_majority is
   port (
        a, b, c : in STD_LOGIC;
        f : out STD_LOGIC
   );
end lab3_majority;
```

- Για να υπολογίσετε την λογική συνάρτηση της εξόδου f, χρησιμοποιήστε πίνακα αληθείας και χάρτη Karnaugh.
- Δημιουργήστε ένα αρχείο δοκιμής (testbench) για το project και εισάγετε όλα τα πιθανά διανύσματα δοκιμής του κυκλώματος.
- Προσομοιώστε το κύκλωμα.
- Υλοποιήστε το κύκλωμα εισάγοντας το κατάλληλο ucf file.
- Προγραμματίστε την πλακέτα και ελέγξτε την λειτουργία του κυκλώματος.

Άσκηση 2: Αθροιστής των 4-bit

Σχεδιάστε ένα αθροιστή που προσθέτει δύο απρόσημους αριθμούς των 4-bit A=A3 A2 A1 A0 και B= B3 B2 B1 B0 και παράγει άθροισμα των 4-bit S= S2 S2 S1 S0 και κρατούμενο εξόδου C. Χρησιμοποιήστε τους διακόπτες SW7-SW0 για τις εισόδους A και B και τα LEDs LED4-LED0 για τις εξόδους S και C.

- Δημιουργήστε ένα νέο project.
- Δημιουργήστε ένα νέο αρχείο (vhdl module) με όνομα lab3_adder.vhd.
- Η οντότητα της μονάδας είναι η εξής:

```
entity lab3_adder is
   port (
        A, B  : UNSIGNED(3 DOWNTO 0);
        S     : UNSIGNED(3 DOWNTO 0);
        C     : STD_LOGIC
   );
end lab3_adder;
```

- Προσθέστε στο αρχείο την παρακάτω βιβλιοθήκη (περιέχει συναρτήσεις για τον τύπο δεδομένων UNSIGNED):

```
USE ieee.numeric_std.ALL;
```

- Δημιουργήστε ένα αρχείο δοκιμής (testbench) για το project και εισάγετε διάφορα σενάρια δοκιμής του κυκλώματος.
- Προσομοιώστε το κύκλωμα.
- Υλοποιήστε το κύκλωμα εισάγοντας το κατάλληλο ucf file.
- Προγραμματίστε την πλακέτα και ελέγξτε την λειτουργία του κυκλώματος.

Άσκηση 3: Αποκωδικοποιητής 3-σε-8

Σχεδιάστε έναν αποκωδικοποιητή 3-σε-8 (3x8 decoder) χρησιμοποιώντας τους διακόπτες SW2-SW0 ως εισόδους και τα leds LED7-LED0 ως εξόδους.

Σημείωση: Ένας αποκωδικοποιητής n-σε-2^n για κάθε συνδυασμό εισόδου ενεργοποιεί (θέτει στην ενεργή τιμή, π.χ. 1) μία έξοδο και απενεργοποιεί (θέτει στο 0) όλες τις υπόλοιπες.

- Δημιουργήστε ένα νέο project.
- Δημιουργήστε ένα νέο αρχείο (vhdl module) με όνομα lab3_decoder_3x8.vhd.
- Η οντότητα της μονάδας είναι η εξής:

```
entity lab3_decoder_3x8 is
   port (
        d : in STD_LOGIC_VECTOR (2 downto 0);
        q : out STD_LOGIC_VECTOR (7 downto 0)
   );
end lab3_decoder_3x8;
```

- Προσθέστε στο αρχείο τις παρακάτω βιβλιοθήκες (περιέχουν συναρτήσεις για τον τύπο δεδομένων std_logic):

```
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

- Εισάγετε την σχεδίαση του αποκωδικοποιητή χρησιμοποιώντας είτε την εντολή ανάθεσης **with-select** είτε την εντολή ανάθεσης **when-else**.
- Εισάγετε στο project το testbench lab3_decoder_3x8_tb.vhd. Σημείωση: Μπορείτε να κατεβάσετε το testbench από την ιστοσελίδα του μαθήματος (περιέχεται στο Lab3.zip).
- Προσομοιώστε το κύκλωμα.
- Υλοποιήστε το κύκλωμα εισάγοντας το κατάλληλο ucf file.
- Προγραμματίστε την πλακέτα και ελέγξτε την λειτουργία του κυκλώματος.

Άσκηση 4: Συγκριτής δυαδικών αριθμών

Σχεδιάστε έναν συγκριτή 2 απρόσημων δυαδικών αριθμών των 4-bit. Το κύκλωμα θα συγκρίνει τους δυαδικούς αριθμούς A=A3 A2 A1 A0 (για την είσοδο A χρησιμοποιήστε τους διακόπτες SW7, SW5, SW5, SW4) και B=B3 B2 B1 B0 (για την είσοδο B χρησιμοποιήστε τους διακόπτες SW3, SW2, SW1, SW0) και θα παράγει 3 εξόδους: LT (μικρότερο από, LED2), GT (μεγαλύτερο από, LED1) και EQ (ίσο, LED0).

- Σχεδιάστε το κύκλωμα χρησιμοποιώντας την εντολή **when-else**.
- Συνθέστε και υλοποιήστε το κύκλωμα εισάγοντας το κατάλληλο ucf file.
- Προγραμματίστε την πλακέτα και ελέγξτε την λειτουργία του κυκλώματος.

Άσκηση 5: Μετατροπέας δυαδικού σε BCD

Σχεδιάστε ένα κύκλωμα που μετατρέπει έναν δυαδικό αριθμό των 4-bit B = B3B2B1B0 στον ισοδύναμο δεκαδικό αριθμό (BCD code) με 2 ψηφία D = D1D0. Ο παρακάτω πίνακας απεικονίζει την μετατροπή:

B3B2B1B0	D1	D0
0000	0	0
0001	0	1
0010	0	2
…	…	
1001	0	9
1010	1	0
1011	1	1
1100	1	2
1101	1	3
1110	1	4
1111	1	5

- Σχεδιάστε το κύκλωμα.
- Συνθέστε και υλοποιήστε το κύκλωμα εισάγοντας το κατάλληλο ucf file. Για τις 4 εισόδους χρησιμοποιήστε τους διακόπτες SW3-SW0 και για τα 2 δεκαδικά ψηφία τα leds της πλακέτας, π.χ. D1 (LED7-LED4) και D0 (LED3-LED0)
- Προγραμματίστε την πλακέτα και ελέγξτε την λειτουργία του κυκλώματος

Σημείωση: Για την σχεδίαση του κυκλώματος μην χρησιμοποιήσετε πολύπλοκες with-select, when-else, case, if-then-else εντολές. Ο σκοπός της άσκησης είναι να υλοποιήσετε το κύκλωμα χρησιμοποιώντας απλές λογικές συναρτήσεις. Μπορείτε να χρησιμοποιήσετε μόνο λογικούς (Boolean) τελεστές, έναν αθροιστή 4-bit και έναν πολυπλέκτη 2x1 (για να υλοποιήσετε τον πολυπλέκτη χρησιμοποιήστε μια εντολή when-else).

Για την σχεδίαση του κυκλώματος βασιστείτε στην εξής παρατήρηση. Όταν η είσοδος B είναι μικρότερη από 10 τότε η έξοδος D1 είναι 0 και η έξοδος D0 είναι ίση με B. Όταν η είσοδος B είναι μεγαλύτερη από 9 τότε η έξοδος D1 είναι 1 και η έξοδος D0 είναι ίση με B+6 (αγνοήστε το κρατούμενο).

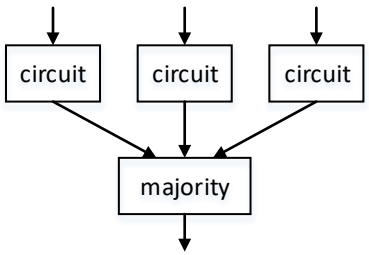
Εργαστηριακή Άσκηση 4: Περιγραφή δομής

Σε αυτήν την εργαστηριακή άσκηση θα ασχοληθείτε με την σχεδίαση συνδυαστικών κυκλωμάτων με χρήση περιγραφής δομής (structural description). Πιο συγκεκριμένα θα σχεδιάσετε:

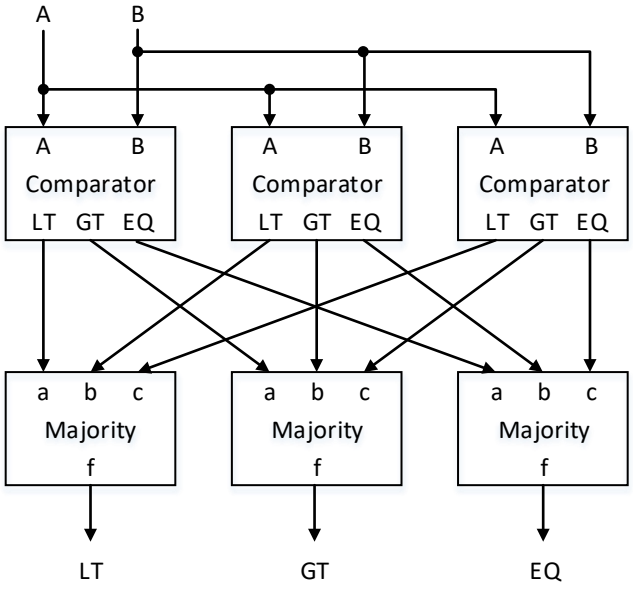
- Τον συγκριτή δύο αριθμών των 2-bit της Άσκησης 3.4 με χρήση της τεχνικής πλεονασμού TMR (Triple Modular Redundancy)

Άσκηση 1: Συγκριτής δυαδικών αριθμών με χρήση της τεχνικής TMR

Η τεχνική TMR (Triple Modular Redundancy) χρησιμοποιείται για την σχεδίαση αξιόπιστων συστημάτων (reliable systems) και βασίζεται στον τριπλασιασμό του υπό σχεδίαση κυκλώματος και την χρήση ενός κυκλώματος πλειοψηφίας των 3 εισόδων (majority voter). Η βασική αρχή του TMR απεικονίζεται στην παρακάτω Εικόνα. Εάν ένα από τα 3 αντίγραφα του κυκλώματος παρουσιάσει σφάλμα τότε το κύκλωμα πλειοψηφίας επιλέγει την σωστή έξοδο.



- Δημιουργήστε ένα νέο project και εισάγετε το vhdl μοντέλο (comparator) που είχατε υλοποιήσει στην Άσκηση 3.4.
- Εισάγετε ένα νέο vhdl unit (comparator_TMR) που έχει τις ίδιες εισόδους & εξόδους με τον comparator (της Άσκησης 3.4). Σχεδιάστε τον συγκριτή δύο αριθμών των 2-bit χρησιμοποιώντας την τεχνική TMR. Χρησιμοποιείστε δομική περιγραφή, όπου θα εισάγετε 3 αντίγραφα (instances) του comparator και για κάθε έξοδο ένα κύκλωμα πλειοψηφίας (χρησιμοποιήστε το κύκλωμα πλειοψηφίας της Άσκησης 3.1), όπως φαίνεται στην παρακάτω Εικόνα.



ΤΟ κύκλωμα θα συγκρίνει τους δυαδικούς αριθμούς A=A1A0 (για την είσοδο A χρησιμοποιήστε τους διακόπτες SW3, SW2) και B=B1B0 (για την είσοδο B χρησιμοποιήστε τους διακόπτες SW1, SW0) και θα παράγει 3 εξόδους: LT (μικρότερο από, LED2), GT (μεγαλύτερο από, LED1) και EQ (ίσο, LED0)

- Προσομοιώστε το κύκλωμα (χρησιμοποιήστε το testbench της Άσκησης 3.4).
- Υλοποιήστε το κύκλωμα (χρησιμοποιήστε το ucf file της Άσκησης 3.4).
- Προγραμματίστε την πλακέτα και ελέγξτε την λειτουργία του κυκλώματος.

Για να ελέγξετε την λειτουργία του μηχανισμού TMR, δημιουργήστε ένα κύκλωμα comparator_faulty, που θα υλοποιεί την ίδια συνάρτηση με τον comparator, αλλά θα περιέχει ένα σφάλμα σε μία από τις εξόδους του (υλοποιήστε ένα λογικό σφάλμα, όπου μία έξοδος θα έχει λανθασμένη τιμή για κάποιους συνδυασμούς εισόδου).

- Αντικαταστήστε μία από τις τρεις μονάδες του comparator_TMR με το κύκλωμα comparator_faulty. Υλοποιήστε και δοκιμάστε το κύκλωμα.
- Αντικαταστήστε δύο από τις τρεις μονάδες του comparator_TMR με το κύκλωμα comparator_faulty. Υλοποιήστε και δοκιμάστε το κύκλωμα.

Εργαστηριακή Άσκηση 5: Διεργασίες και ακολουθιακές εντολές

Σε αυτήν την εργαστηριακή άσκηση θα ασχοληθείτε με την σχεδίαση συνδυαστικών κυκλωμάτων χρησιμοποιώντας διεργασίες (process) και ακολουθιακές εντολές (if-then-else, for-loop). Πιο συγκεκριμένα θα σχεδιάσετε:

- Έναν συγκριτή δύο αριθμών των 2-bit
- Έναν απαριθμητή άσσων σε έναν δυαδικό αριθμό

Άσκηση 1: Συγκριτής δυαδικών αριθμών

Σχεδιάστε τον συγκριτή 2 απρόσημων δυαδικών αριθμών των 2-bit που είχατε σχεδιάσει και στην άσκηση 3.4 χρησιμοποιώντας όμως την εντολή if-then-else (αντί της εντολής when-else που είχατε χρησιμοποιήσει στο εργαστήριο 3).

Το κύκλωμα θα συγκρίνει τους δυαδικούς αριθμούς A=A1A0 (για την είσοδο A χρησιμοποιήστε τους διακόπτες SW3, SW2) και B=B1B0 (για την είσοδο B χρησιμοποιήστε τους διακόπτες SW1, SW0) και θα παράγει 3 εξόδους: LT (μικρότερο από, LED2), GT (μεγαλύτερο από, LED1) και EQ (ίσο, LED0).

- Σχεδιάστε το κύκλωμα χρησιμοποιώντας την εντολή **if-then-else**.
- Συνθέστε και υλοποιήστε το κύκλωμα εισάγοντας το κατάλληλο ucf file.
- Προγραμματίστε την πλακέτα και ελέγξτε την λειτουργία του κυκλώματος.

Άσκηση 2: Απαριθμητής άσσων σε έναν δυαδικό αριθμό

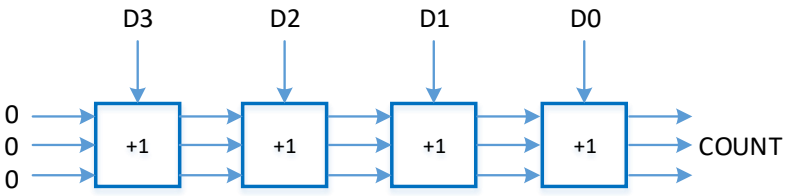
Σχεδιάστε ένα κύκλωμα που θα απαριθμεί τον αριθμό των άσσων (‘1’) που περιλαμβάνονται σε έναν δυαδικό αριθμό των 4-bit.

- Η οντότητα της μονάδας είναι η εξής:

```
entity ones_counter is
   port (
        D     : in  STD_LOGIC_VECTOR(3 DOWNTO 0);
        COUNT : out UNSIGNED(2 DOWNTO 0)
   );
end ones_counter;
```

- Για την έξοδο COUNT μπορείτε να χρησιμοποιήσετε και τους τύπους integer ή std_logic_vector.
- Σχεδιάστε το κύκλωμα χρησιμοποιώντας την εντολή **for loop**.
- Συνθέστε και υλοποιήστε το κύκλωμα εισάγοντας το κατάλληλο ucf file (για την είσοδο D χρησιμοποιήστε τους διακόπτες SW3-SW0 και για την έξοδο COUNT τα LED2-LED0).
- Προγραμματίστε την πλακέτα και ελέγξτε την λειτουργία του κυκλώματος.

Σχεδιάστε το ίδιο κύκλωμα χρησιμοποιώντας την δομική περιγραφή του παρακάτω Σχήματος. Σχολιάστε πως σχετίζεται η δομική περιγραφή με την αρχική υλοποίηση με χρήση της εντολής for loop.



Εργαστηριακή Άσκηση 6: Ακολουθιακά κυκλώματα

Σε αυτήν την εργαστηριακή άσκηση θα ασχοληθείτε με την σχεδίαση ακολουθιακών κυκλωμάτων και ειδικότερα την υλοποίηση

- Δυαδικών μετρητών
- Κυκλωμάτων διαίρεσης συχνότητας
- Κυκλωμάτων αποφυγής κλυδωνισμού (debouncing circuits)
- Συσσωρευτών (accumulators)

Σημείωση: Υιοθετήστε τα ονόματα των οντοτήτων (entities) και των θυρών (ports) που αναφέρονται στην εκφώνηση της άσκησης για να μην έχετε προβλήματα ασυμβατότητας με τα vhdl αρχεία που σας δίδονται.

Άσκηση 1: Δυαδικός μετρητής των 4-bit

Σχεδιάστε έναν δυαδικό μετρητή των 4-bit που έχει τις εξής λειτουργίες: (α) μηδενίζεται όταν ενεργοποιηθεί το σήμα εισόδου RESET, (β) μετράει προς τα πάνω ή προς τα κάτω ανάλογα με την τιμή ενός σήματος εισόδου UP_DOWN και (γ) παγώνει όταν ενεργοποιηθεί το σήμα εισόδου FREEZE. Για τις εισόδους και εξόδους του κυκλώματος να χρησιμοποιήσετε τα παρακάτω:

- RESET: BTN0
- FREEZE: BTN1
- UP_DOWN: SW0
- COUNT (counter outputs): LED3-LED0
- CLK: για το ρολόι του μετρητή να χρησιμοποιήσετε το 50MHz ρολόι της πλακέτας

- Δημιουργήστε ένα νέο project.
- Δημιουργήστε ένα νέο αρχείο (vhdl module) με όνομα counter_4b.vhd.
- Η οντότητα της μονάδας είναι η εξής:

```
entity counter_4b is
   port (
        clk, reset      : in STD_LOGIC;
        freeze, up_down : in STD_LOGIC;
        count           : out unsigned(3 downto 0)
   );
end counter_4b;
```

- Σχεδιάστε την αρχιτεκτονική του μετρητή (Σημείωση: Συμβουλευτείτε τις διαφάνειες του μαθήματος).
- Δημιουργήστε ένα vhdl testbench με όνομα counter_4b_tb.vhd και συσχετίστε το με την μονάδα counter_4b. Το ρολόι θα πρέπει να έχει περίοδο 20ns (συχνότητα 50 MHz). Προσομοιώστε το κύκλωμα.
- Υλοποιήστε το κύκλωμα εισάγοντας το κατάλληλο ucf file.
- Προγραμματίστε την πλακέτα και δοκιμάστε όλες τις λειτουργίες του κυκλώματος. Τι παρατηρείτε;

Άσκηση 2: Διαιρέτης ρολογιού

Σε αυτήν την άσκηση θα χρησιμοποιήσετε το αρχείο freq_div.vhd. Η οντότητα freq_div διαιρεί την συχνότητα του ρολογιού.

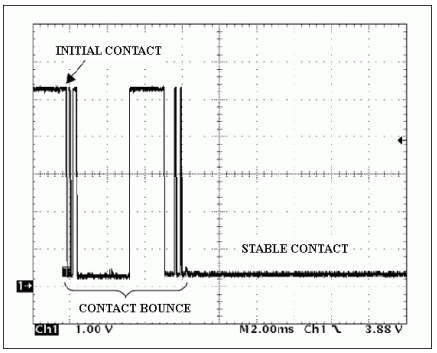
Σημείωση: Μπορείτε να κατεβάσετε τα αρχεία freq_div.vhd και top_level_counter.vhd από την ιστοσελίδα του μαθήματος (περιέχεται στο Lab4.zip).

- Προσθέστε στο project της Άσκησης 1 το αρχείο freq_div.vhd
- Προσθέστε στο project το αρχείο top_level_counter.vhd. Σημείωση: Το αρχείο έχει την ίδια οντότητα (ports) με το counter_4b και περιέχει τα στιγμιότυπα (instances) του counter_4b και του freq_div. Ουσιαστικά συνδέει στον μετρητή το διαιρεμένο ρολόι αντί του ρολογιού των 50MHz της πλακέτας.
- Μπορείτε να ρυθμίσετε τον διαιρέτη της συχνότητας ανάλογα με την τιμή της σταθεράς CLK_DIVISOR. Το ρολόι εισόδου έχει οριστεί ως 50MHz και το ρολόι εξόδου 2Hz.
- Συνθέστε και υλοποιήστε το κύκλωμα χρησιμοποιώντας το ίδιο ucf file.
- Προγραμματίστε την πλακέτα και ελέγξτε την λειτουργία του κυκλώματος.
- Μπορείτε να αλλάξετε τον διαιρέτη του ρολογιού σε άλλη συχνότητα εξόδου και να ελέγξετε την λειτουργία του κυκλώματος.

Άσκηση 3: Κύκλωμα αποφυγής κλυδωνισμού

Σε αυτήν την άσκηση θα χρησιμοποιήσετε ένα κύκλωμα αποφυγής κλυδωνισμού (debouncer).

Το φαινόμενο του κλυδωνισμού (bouncing) εμφανίζεται στους μηχανικούς διακόπτες και έχει ως αποτέλεσμα η έξοδος του διακόπτη να αλλάξει πολλές φορές τιμή μεταξύ του ON και OFF προτού σταθεροποιηθεί στην τελική της τιμή (όπως στην διπλανή εικόνα). Αυτό μπορεί να δημιουργήσει προβλήματα όταν η συχνότητα του ρολογιού του κυκλώματος είναι μεγαλύτερη από την συχνότητα των κλυδωνισμών.



Σχεδιάστε έναν μετρητή των 4-bit όπως στην άσκηση 1 μόνο που αυτήν την φορά ο μετρητής θα αλλάζει τιμές χειροκίνητα (με το πάτημα ενός κουμπιού). Ο μετρητής έχει τις εξής λειτουργίες: (α) μηδενίζεται όταν ενεργοποιηθεί το σήμα εισόδου RESET, (β) μετράει προς τα πάνω όταν πατηθεί το κουμπί UP (για κάθε πάτημα του κουμπιού ο μετρητής θα πρέπει να αυξάνεται μόνο μία φορά) και (γ) μετράει προς τα κάτω όταν πατηθεί το κουμπί DOWN (για κάθε πάτημα του κουμπιού ο μετρητής θα πρέπει να μειώνεται μόνο μία φορά). Για τις εισόδους και εξόδους του κυκλώματος να χρησιμοποιήσετε τα παρακάτω:

- RESET: BTN0
- UP: BTN1
- DOWN: BTN2
- COUNT (counter outputs): LED3-LED0
- CLK: on-board 50MHz clock (Προσοχή: Μην χρησιμοποιήσετε το διαιρεμένο ρολόι)

- Δημιουργήστε ένα νέο project.
- Δημιουργήστε ένα νέο αρχείο (vhdl module) με όνομα counter_m_4b.vhd.
- Η οντότητα της μονάδας είναι η εξής:

```
entity counter_m_4b is
   port (
        clk, reset : in STD_LOGIC;
        up, down   : in STD_LOGIC;
        counter    : out unsigned(3 downto 0)
   );
end counter_m_4b;
```

Για κάθε πάτημα των κουμπιών UP & DOWN θα πρέπει ο μετρητής να αντιλαμβάνεται μόνο ένα παλμό ρολογιού (των 50MHz) ώστε να μετράει μόνο μία φορά. Διαφορετικά, για κάθε πάτημα του κουμπιού θα μετράει πολλές φορές. Για να το πετύχετε αυτό εισάγετε στην αρχιτεκτονική σας το παρακάτω κύκλωμα (προσθέστε και τα αντίστοιχα σήματα στην αρχιτεκτονική).

```
up_button: process (clk)
begin
     if clk'event and clk = '1' then
          up_q1 <= UP;
          up_q2 <= up_q1;
     end if;
end process;
up_pulse <= up_q1 and (not up_q2);

down_button: process (clk)
begin
     if clk'event and clk = '1' then
          down_q1 <= DOWN;
          down_q2 <= down_q1;
     end if;
end process;
down_pulse <= down_q1 and (not down_q2);
```

Το παραπάνω κύκλωμα για κάθε πάτημα του κουμπιού UP ή DOWN παράγει έναν παλμό διάρκειας μίας περιόδου του ρολογιού CLK.

- Χρησιμοποιήστε τα σήματα up_pulse & down_pulse ως σήματα ενεργοποίησης (enable) του μετρητή αντί να χρησιμοποιήσετε απευθείας τα σήματα από τα κουμπιά UP & DOWN.
- Δημιουργήστε ένα vhdl testbench με όνομα counter_m_4b_tb.vhd και συσχετίστε το με την μονάδα counter_m_4b. Το ρολόι θα πρέπει να έχει περίοδο 20ns (συχνότητα 50 MHz). Προσομοιώστε το κύκλωμα. Για τα κουμπιά UP & DOWN προσομοιώστε «πατήματα» διάρκειας 10 παλμών (200 ns).
- Υλοποιήστε το κύκλωμα εισάγοντας το κατάλληλο ucf file.
- Προγραμματίστε την πλακέτα και δοκιμάστε όλες τις λειτουργίες του κυκλώματος.

Εάν παρατηρήσετε ο μετρητής να έχει περίεργη συμπεριφορά, δηλαδή να σε κάθε πάτημα κουμπιού να παίρνει «απροσδιόριστες» τιμές αυτό μπορεί να οφείλεται στο φαινόμενο του κλυδωνισμού. Για να αποφύγετε το φαινόμενο του κλυδωνισμού μπορείτε να χρησιμοποιήσετε το παρακάτω κύκλωμα (τροποποιήστε την διεργασία up_button).

```
constant deb_length: integer :=16;
signal up_q: std_logic_vector(deb_length-1 downto 0);
...
up_button_p1: process (clk)
begin
  if clk'event and clk = '1' then
       up_q(0) <= UP;
       up_q(deb_lenght-1 downto 1) <= up_q(deb_lenght-2 downto 0);
  end if;
end process;

up_button_p2: process(up_q)
variable temp: std_logic;
begin
  temp := up_q(0);
  for i in 1 to deb_length-2 loop
       temp := temp and up_q(i);
  end loop;
  temp := temp and (not up_q(length-1));
  up_pulse <= temp;
end process;
```

- Ομοίως τροποποιήστε και την διεργασία down_button.
- Συνθέστε και υλοποιήστε ξανά το κύκλωμα.
- Προγραμματίστε την πλακέτα και ελέγξτε την λειτουργία του κυκλώματος.
- Μπορείτε να προσαρμόσετε το βάθος του κυκλώματος (deb_length) ανάλογα με την συχνότητα του ρολογιού και την διάρκεια των παλμών κλυδωνισμού.

Άσκηση 4: Συσσωρευτής των 4-bit

Σχεδιάστε έναν συσσωρευτή των 4-bit (4-bit accumulator) που εκτελεί την πράξη: ACC = ACC + DIN. Η είσοδος DIN έχει μέγεθος 4 bit και η έξοδος ACC έχει μέγεθος 8 bit. Το κύκλωμα θα πρέπει να έχει τις εξής λειτουργίες: (α) μηδενίζεται όταν ενεργοποιηθεί το σήμα εισόδου RESET, (β) έχει είσοδο επίτρεψης (ENABLE) και διαβάζει την είσοδο DIN μόνο όταν η είσοδος ENABLE είναι 1. Για τις εισόδους και εξόδους του κυκλώματος να χρησιμοποιήσετε τα παρακάτω:

- RESET: BTN0
- ENABLE: BTN1
- DIN (accumulator inputs): SW3-SW0
- COUNT (counter outputs): LED7-LED0
- CLK: on-board 50MHz clock (Προσοχή: Μην χρησιμοποιήσετε το διαιρεμένο ρολόι)

Σημείωση: Για κάθε νέα είσοδο που θέλετε να προσθέσει ο συσσωρευτής θα πρέπει να ενεργοποιήσετε την είσοδο (push button) ENABLE αφού πρώτα θέσετε την είσοδο DIN στους διακόπτες SW3-SW0. Για κάθε πάτημα του ENABLE ο συσσωρευτής θα πρέπει να αντιλαμβάνεται μόνο ένα παλμό ρολογιού (των 50MHz) ώστε να διαβάζει και να προσθέτει μόνο μία τιμή. Γι' αυτόν τον σκοπό (αλλά και για την αποφυγή του κλυδωνισμού) χρησιμοποιήστε το κύκλωμα της προηγούμενης άσκησης.

- Δημιουργήστε ένα νέο project.
- Δημιουργήστε ένα νέο αρχείο (vhdl module) με όνομα acc_4b.vhd.
- Η οντότητα της μονάδας είναι η εξής:

```
entity acc_4b is
   port (
        clk, reset      : in STD_LOGIC;
        enable          : in STD_LOGIC;
        din             : in unsigned(3 downto 0);
        acc             : out unsigned(7 downto 0)
   );
end acc_4b;
```

- Σχεδιάστε την αρχιτεκτονική του συσσωρευτή (Σημείωση: Συμβουλευτείτε τις διαφάνειες του μαθήματος).
- Δημιουργήστε ένα vhdl testbench με όνομα acc_4b_tb.vhd και συσχετίστε το με την μονάδα acc_4b. Το ρολόι θα πρέπει να έχει περίοδο 20ns (συχνότητα 50 MHz). Προσομοιώστε το κύκλωμα.
- Υλοποιήστε το κύκλωμα εισάγοντας το κατάλληλο ucf file.
- Προγραμματίστε την πλακέτα και δοκιμάστε την λειτουργία του κυκλώματος.

Σχεδιάστε το ίδιο κύκλωμα με την χρήση σχηματικού διαγράμματος. Χρησιμοποιήστε έναν αθροιστή των 8-bit (add8) και έναν καταχωρητή των 8-bit (fd8re).

Εργαστηριακή Άσκηση 7: Μηχανές πεπερασμένων καταστάσεων

Σε αυτήν την εργαστηριακή άσκηση θα ασχοληθείτε με την σχεδίαση μηχανών πεπερασμένων καταστάσεων (finite state machines, FSMs) και ειδικότερα την υλοποίηση:

- Μιας γεννήτριας ισοτιμίας (parity generator) σειριακής εισόδου
- Ενός ανιχνευτή ακολουθίας (sequence generator)

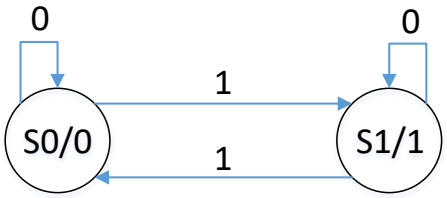
Άσκηση 1: Γεννήτρια ισοτιμίας

Σχεδιάστε ένα ακολουθιακό κύκλωμα που δέχεται μία σειριακή είσοδο DIN και παράγει την ισοτιμία της εισόδου στην σειριακή έξοδο PARITY, δηλαδή παράγει την τιμή 1 όταν λάβει περιττό αριθμό άσσων στην σειριακή είσοδο. Η γεννήτρια ισοτιμίας θα πρέπει να έχει είσοδο μηδενισμού RESET και να λειτουργεί με ρολόι ½ Hz (περίοδο 2 sec). Για τις εισόδους και εξόδους του κυκλώματος να χρησιμοποιήσετε τα παρακάτω:

- RESET: BTN0
- DIN: SW0
- PARITY: LED0
- CLK: on-board 50MHz clock.

Οδηγία: Να χρησιμοποιήσετε το κύκλωμα της άσκησης 2 της εργαστηριακής άσκησης 4 (διαιρέτης ρολογιού) για να διαιρέσετε το ρολόι συχνότητας 50MHz της πλακέτας σε ρολόι συχνότητας ½ Hz. Μπορείτε επίσης να οδηγήσετε το σήμα του διαιρεμένου ρολογιού και σε ένα LED (π.χ. LED7) ώστε να παρακολουθείτε κάθε πότε αλλάζει το ρολόι.

Η λειτουργία της γεννήτριας μπορεί να μοντελοποιηθεί ως μια μηχανή πεπερασμένων καταστάσεων τύπου Moore. (Σημείωση: η γεννήτρια θα μπορούσε να υλοποιηθεί και ως μια μηχανή τύπου Mealy).



- Δημιουργήστε ένα νέο project.
- Δημιουργήστε ένα νέο αρχείο (vhdl module) με όνομα par_gen.vhd.
- Η οντότητα της μονάδας είναι η εξής:

```
entity par_gen is
   port (
        clk, reset : in STD_LOGIC;
        din        : in STD_LOGIC;
        parity     : out STD_LOGIC
   );
end par_gen;
```

- Σχεδιάστε μια αρχιτεκτονική που υλοποιεί την παραπάνω μηχανή πεπερασμένων καταστάσεων (Σημείωση: Συμβουλευτείτε τις διαφάνειες του μαθήματος).
- Δημιουργήστε ένα vhdl testbench με όνομα par_gen_tb.vhd και συσχετίστε το με την μονάδα par_gen. Προσομοιώστε το κύκλωμα.

  Οδηγία: Για τις ανάγκες της προσομοίωσης τροποποιήστε τον διαιρέτη συχνότητας ώστε να διαιρεί το ρολόι εισόδου κατά έναν μικρό διαιρέτη (π.χ. διά 4). Προτού προχωρήσετε στο επόμενο βήμα επαναφέρατε τον διαιρέτη του ρολογιού στην αρχική του τιμή.
- Υλοποιήστε το κύκλωμα εισάγοντας το κατάλληλο ucf file.
- Προγραμματίστε την πλακέτα και δοκιμάστε όλες τις λειτουργίες του κυκλώματος.

Πειραματιστείτε με τις ιδιότητες του εργαλείου σύνθεσης για τον τύπο κωδικοποίησης των FSMs. (Synthesize-XST -> Process Properties -> HDL Options -> FSM Encoding Algorithm). Δοκιμάστε τους τύπους One-Hot & Compact και συγκρίνετε τα αποτελέσματα της σύνθεσης επιλέγοντας View RTL Schematic.

Άσκηση 2: Ανιχνευτής ακολουθίας

Σχεδιάστε ένα ακολουθιακό κύκλωμα που δέχεται μία είσοδο των 2-bit DIN[1:0] και παράγει στην έξοδο DOUT την τιμή 1 όταν ο συνολικός αριθμός των άσσων που έχει ληφθεί στην είσοδο DIN διαιρείται ακριβώς με το 3. Το κύκλωμα θα πρέπει να έχει τις εξής λειτουργίες: (α) μηδενίζεται όταν ενεργοποιηθεί το σήμα εισόδου RESET, (β) έχει είσοδο επίτρεψης (ENABLE) και διαβάζει την είσοδο DIN μόνο όταν η είσοδος ENABLE είναι 1 και (γ) λειτουργεί με ρολόι ½ Hz (περίοδο 2 sec). Για τις εισόδους και εξόδους του κυκλώματος να χρησιμοποιήσετε τα παρακάτω:

- RESET: BTN0
- ENABLE: BTN1
- DIN[1:0]: SW1-SW0
- DOUT: LED0
- CLK: on-board 50MHz clock.

Οδηγία: Να χρησιμοποιήσετε το κύκλωμα της άσκησης 2 της εργαστηριακής άσκησης 4 (διαιρέτης ρολογιού) για να διαιρέσετε το ρολόι συχνότητας 50MHz της πλακέτας σε ρολόι συχνότητας ½ Hz. Μπορείτε επίσης να οδηγήσετε το σήμα του διαιρεμένου ρολογιού και σε ένα LED (π.χ. LED7) ώστε να παρακολουθείτε κάθε πότε αλλάζει το ρολόι.

Μοντελοποιήστε την λειτουργία του κυκλώματος με μια μηχανή πεπερασμένων καταστάσεων τύπου Moore. Πόσες καταστάσεις έχει η μηχανή?

- Δημιουργήστε ένα νέο project.
- Δημιουργήστε ένα νέο αρχείο (vhdl module) με όνομα seq_det.vhd.
- Σχεδιάστε την οντότητα και την αρχιτεκτονική του ανιχνευτή ακολουθίας.
- Δημιουργήστε ένα vhdl testbench με όνομα seq_det_tb.vhd και συσχετίστε το με την μονάδα seq_det. Προσομοιώστε το κύκλωμα.

  Οδηγία: Για τις ανάγκες της προσομοίωσης τροποποιήστε τον διαιρέτη συχνότητας ώστε να διαιρεί το ρολόι εισόδου κατά έναν μικρό διαιρέτη (π.χ. διά 4). Προτού προχωρήσετε στο επόμενο βήμα επαναφέρατε τον διαιρέτη του ρολογιού στην αρχική του τιμή.
- Υλοποιήστε το κύκλωμα εισάγοντας το κατάλληλο ucf file.
- Προγραμματίστε την πλακέτα και δοκιμάστε όλες τις λειτουργίες του κυκλώματος.

Εργαστηριακή Άσκηση 6: Μνήμες

Σε αυτήν την εργαστηριακή άσκηση θα υλοποιήσετε: