Εργαστηριακές ασκήσεις: σχεδίαση κυκλωμάτων σε FPGA με VHDL

Μιχάλης Ψαράκης, Δημήτριος Αγιακάτσικας, Ιωάννα Σουβατζόγλου

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ - ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΠΜΣ ΚΥΒΕΡΝΟΑΣΦΑΛΕΙΑ ΚΑΙ ΕΠΙΣΤΗΜΗ ΔΕΔΟΜΕΝΩΝ

ΜΑΘΗΜΑ: «ΣΧΕΔΙΑΣΗ ΕΝΣΩΜΑΤΩΜΕΝΩΝ ΣΥΣΤΗΜΑΤΩΝ»

Εισαγωγή

Οι εργαστηριακές ασκήσεις του μαθήματος "Σχεδίαση Ενσωματωμένων Συστημάτων» θα σας εισάγουν στην *σχεδίαση και υλοποίηση κυκλωμάτων σε τεχνολογία (Field Programmable Gate Arrays) FPGA με χρήση της γλώσσας περιγραφής υλικού (hardware description language) VHDL*. Για τις ανάγκες των εργαστηριακών ασκήσεων θα χρησιμοποιήσετε το εργαλείο λογισμικού Xilinx Vivado και την αναπτυξιακή πλακέτα (Digilent Zybo board).

Οι στόχοι των εργαστηριακών ασκήσεων είναι:

- να εξοικειωθείτε με ένα ολοκληρωμένο περιβάλλον σχεδίασης, προσομοίωσης και σύνθεσης κυκλωμάτων με τεχνολογία FPGA
- να σας εισάγει στην γλώσσα περιγραφής υλικού VHDL μέσα από μια σειρά απλών εργαστηριακών ασκήσεων
- 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).
- να υλοποιήσετε έναν επεξεργαστή ανοικτού κώδικα (RISC-V) σε FPGA και να εξοικειωθείτε με τα εργαλεία του για την ανάπτυξη λογισμικού. Ο συγκεκριμένος επεξεργαστής θα χρησιμοποιηθεί στο μάθημα του εαρινού εξαμήνου «Αξιόπιστα Ενσωματωμένα Συστήματα».

Οι εργαστηριακές ασκήσεις χωρίζονται στις παρακάτω ενότητες:

- Ενότητα 1 - Εξοικείωση με το εργαλείο Xilinx Vivado και με την χρήση πλακέτας FPGA: Θα εξοικειωθείτε με τη χρήση του εργαλείου Xilinx Vivado και την εκπαιδευτική πλακέτα Digilent Zybo board σχεδιάζοντας, προσομοιώντας και υλοποιώντας στην πλακέτα στοιχειώδη κυκλώματα.
- Ενότητα 2 - Σχεδίαση απλών συνδυαστικών κυκλωμάτων με VHDL: Θα ασχοληθείτε με την σχεδίαση, προσομοίωση και υλοποίηση στην πλακέτα FPGA απλών συνδυαστικών κυκλωμάτων (π.χ. αθροιστές, συγκριτές, κτλ.).
- Ενότητα 3 - Σχεδίαση απλών ακολουθιακών κυκλωμάτων με VHDL: Θα ασχοληθείτε με την σχεδίαση, προσομοίωση και υλοποίηση στην πλακέτα FPGA απλών ακολουθιακών κυκλωμάτων (π.χ. μετρητών, συσσωρευτών, κτλ.).
- Ενότητα 4 – Υλοποίηση του επεξεργαστή RISC-V "neorv32" στην πλακέτα Zybo Z7 board χρήση των εργαλείων του επεξεργαστή για την ανάπτυξη λογισμικού.

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

Εργαστηριακή Άσκηση 1: Εξοικείωση με το εργαλείο και την πλακέτα FPGA

Σε αυτήν την εργαστηριακή άσκηση θα εξοικειωθείτε με το περιβάλλον Xilinx Vivado και την χρήση της εκπαιδευτικής πλακέτας FPGA. 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.
- Τα χαρακτηριστικά της εκπαιδευτικής και αναπτυξιακής πλακέτας (Digilent Zybo board) που θα χρησιμοποιήσετε στο εργαστήριο
- Πώς να δημιουργείτε ένα αρχείο προγραμματισμού FPGA
- Πώς να προγραμματίζετε μια συσκευή FPGA

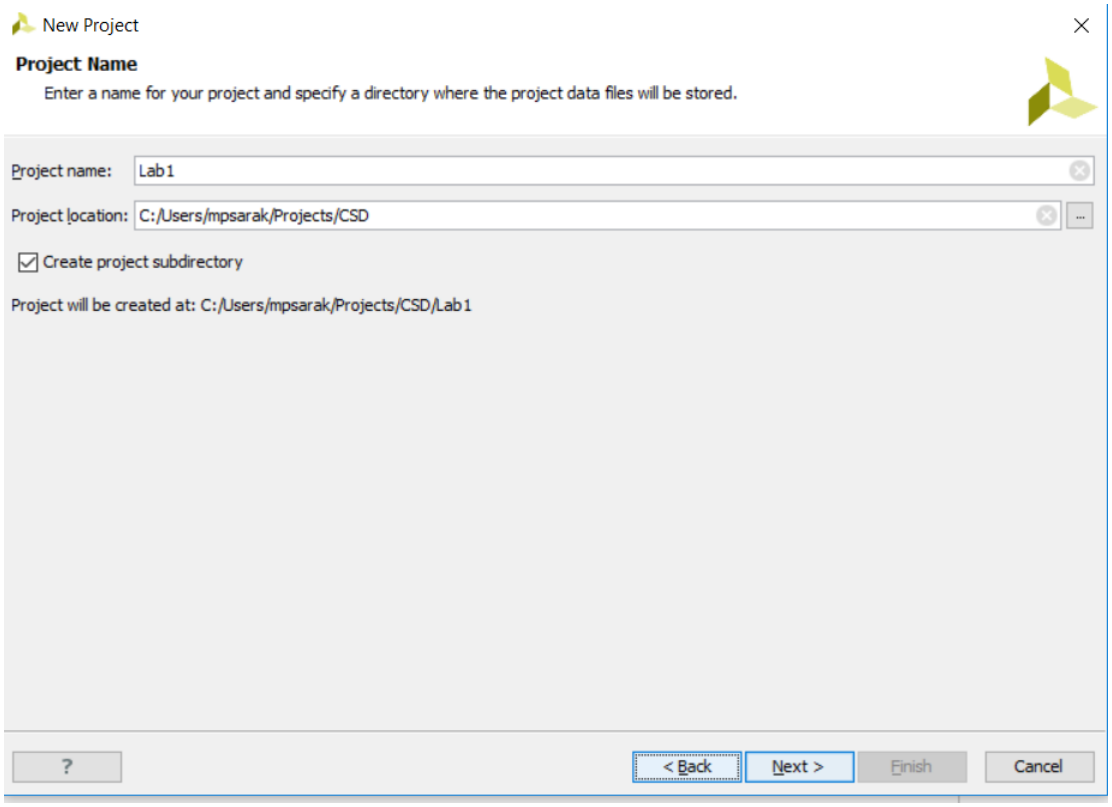
Άσκηση 1: 4-bit parity generator (με VHDL)

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 2016.4→Vivado 2016.4. Επιλέξτε Create New Project. Θα εμφανιστεί το πρώτο πλαίσιο διαλόγου του New Project, όπως φαίνεται στην Εικόνα 1.

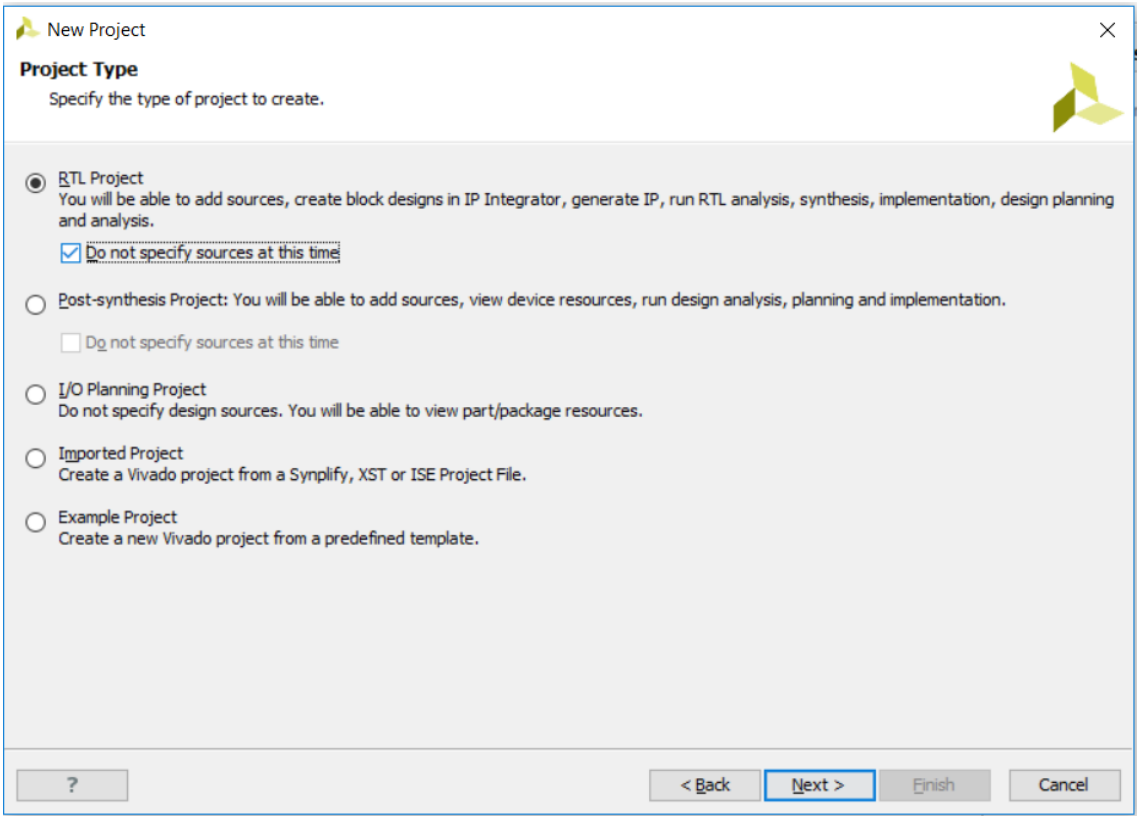
Το πλαίσιο διαλόγου σας προτρέπει να εισάγετε το όνομα και τον κατάλογο του έργου, όπως φαίνεται στην Εικόνα 1. Αφού συμπληρώσετε τα στοιχεία, πατήστε Next.



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

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

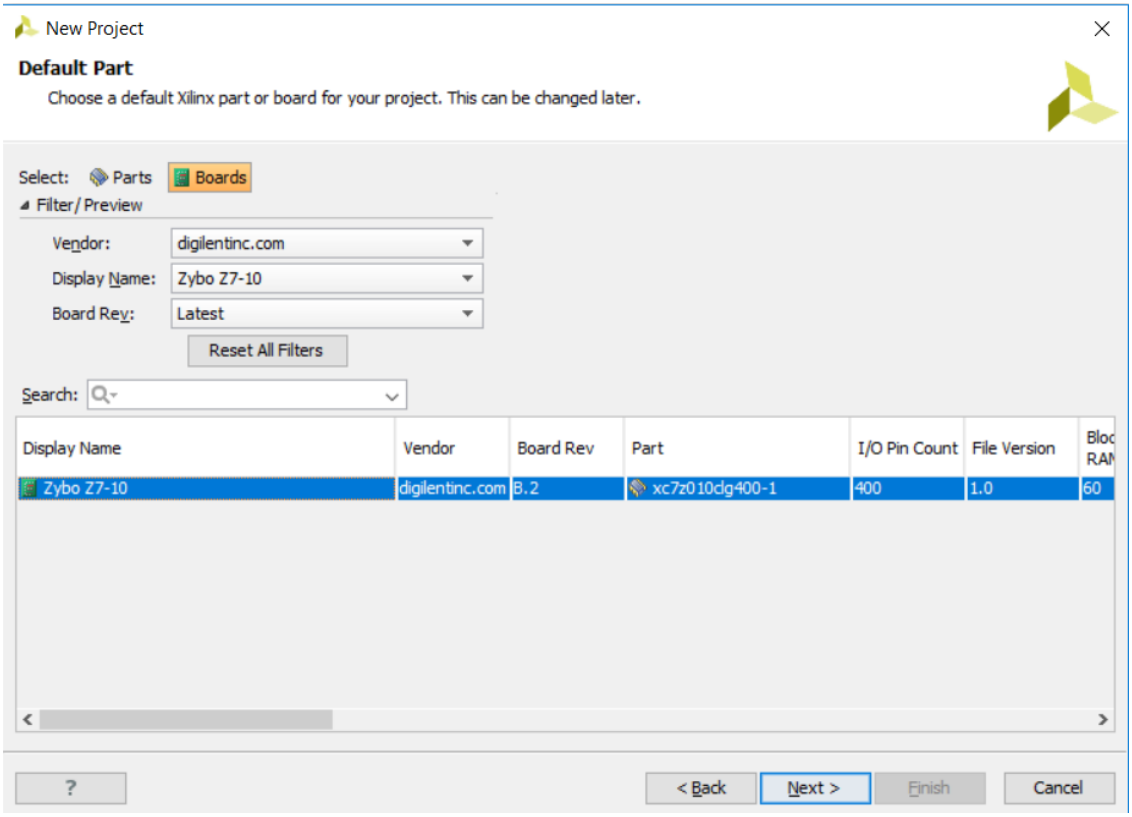
Το επόμενο πλαίσιο διαλόγου σας επιτρέπει να επιλέξετε τον τύπο του έργου. Επιλέξτε τύπο RTL Project και ενεργοποιήστε την επιλογή Do not specify sources at this time, όπως φαίνεται στην Εικόνα 3, και πατήστε Next.



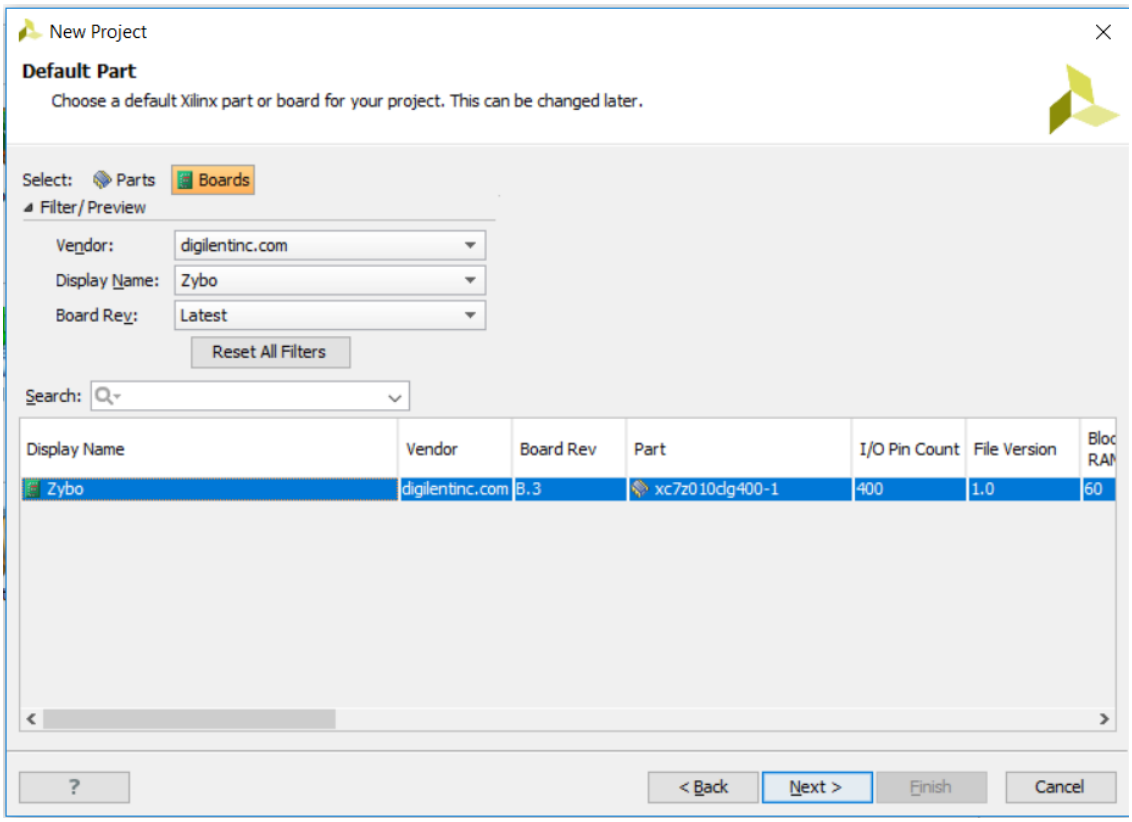
Εικόνα 2: New Project – Project type (2 από 4)

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. Όσοι φοιτητές έχουν την πλακέτα Zybo Z7-10, επιλέξτε Display Name: Zybo Z7-10, όπως στην Εικόνα 3.α, ενώ όσοι έχουν την πλακέτα Zybo, επιλέξτε Display Name: Zybo, όπως στην Εικόνα 3.β. 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: Zynq-7000, στο πεδίο Package: clg400, και στο πεδίο Speed grade: -1. Τέλος, επιλέξτε την συσκευή xc7z010clg400-1.



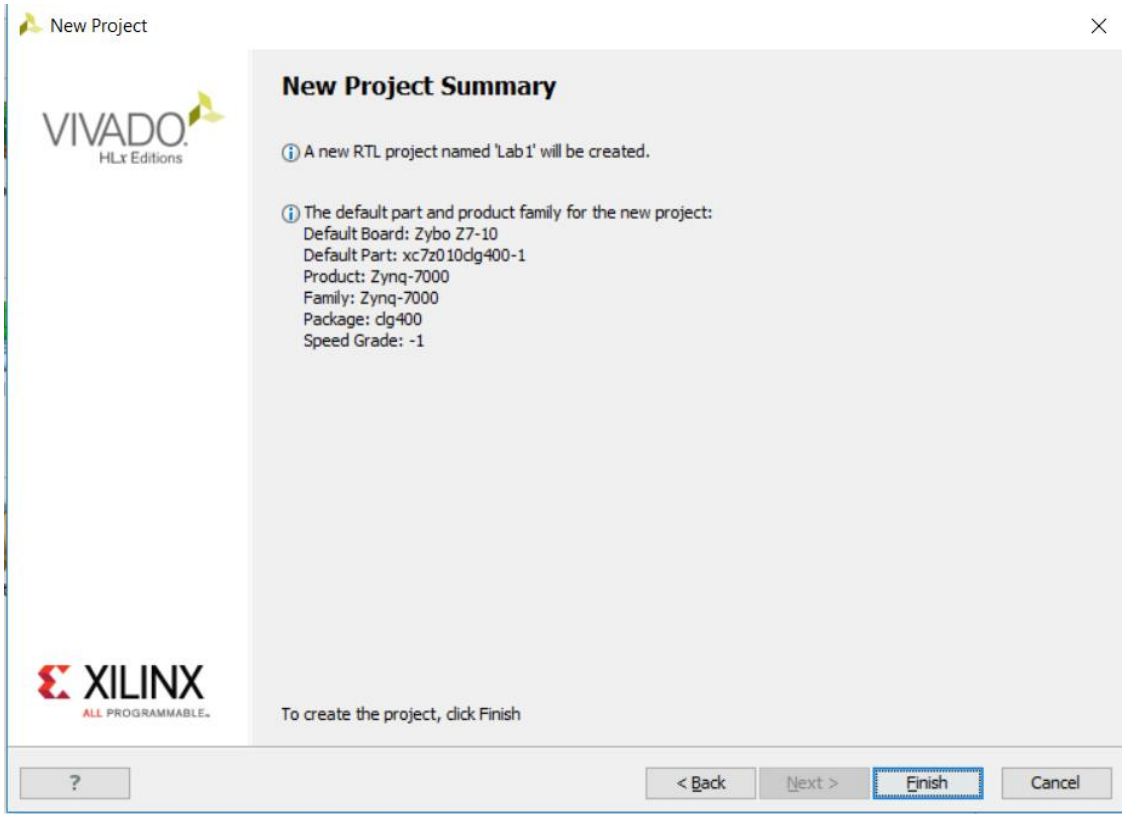
Εικόνα 2.α: Επιλογή πλακέτας Zybo Z7-10



Εικόνα 3.β: Επιλογή πλακέτας Zybo

Εικόνα 3: New Project – Default Part (2 από 4)

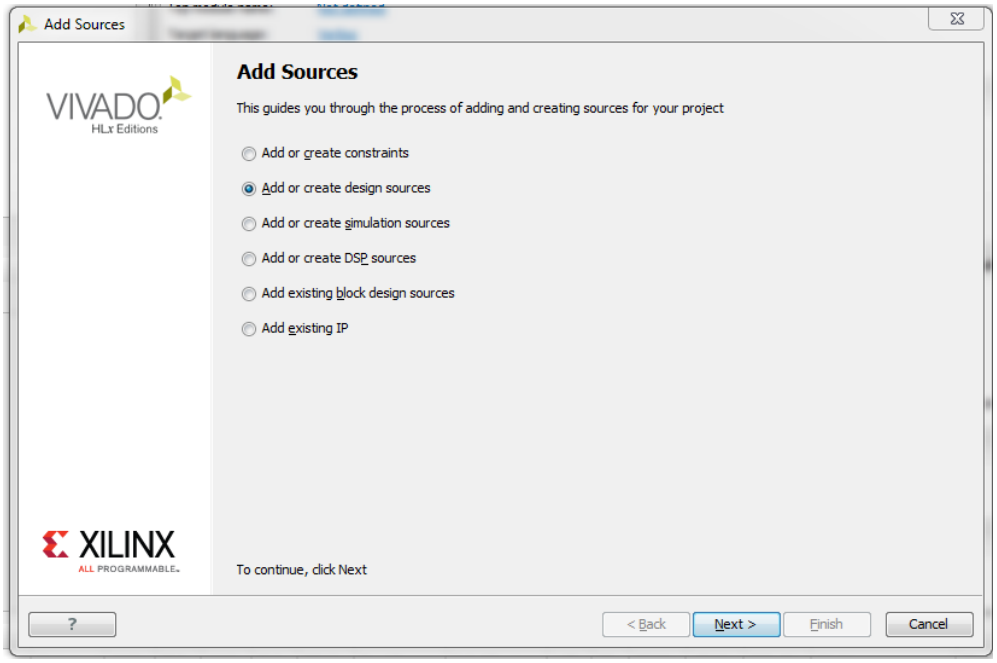
Το τελικό πλαίσιο διαλόγου στη διαδικασία δημιουργίας νέου έργου, που φαίνεται στην Εικόνα 4, παρέχει μια περίληψη του έργου που το Vivado θα δημιουργήσει βασισμένο στις ρυθμίσεις σας. Ελέγξτε την περίληψη για να σιγουρευτείτε ότι ταιριάζει με ότι φαίνεται στην Εικόνα 4. Εάν όχι, πατήστε Back για να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



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

1.2 Εισαγωγή σχεδίασης

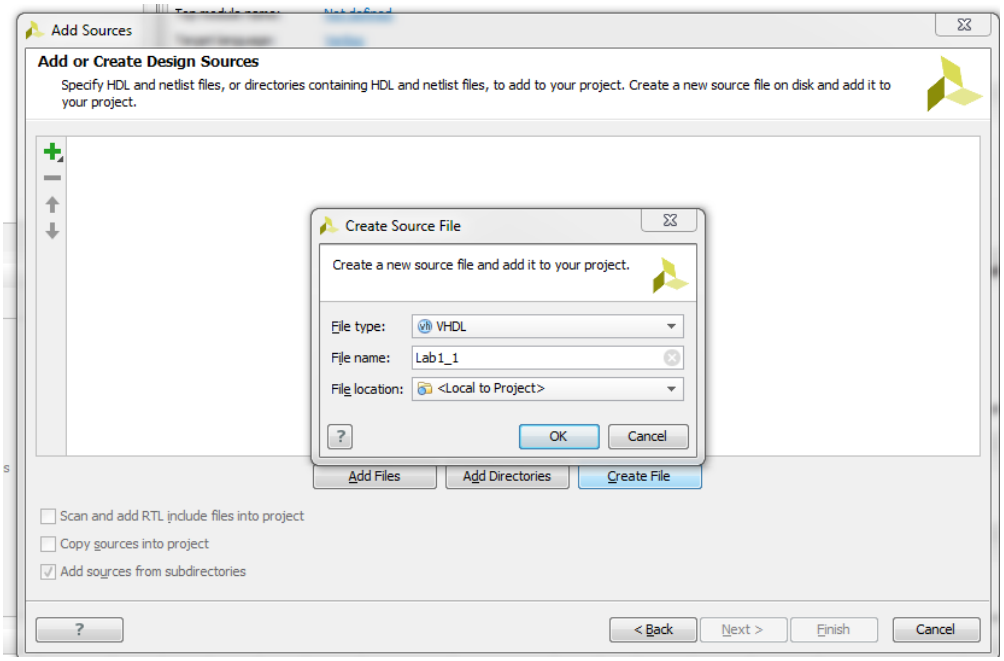
Σε αυτό το σημείο, το έργο που έχει δημιουργηθεί δεν περιέχει κανένα πηγαίο αρχείο. Δημιουργήστε ένα νέο πηγαίο αρχείο για τη σχεδίαση του κυκλώματος υπολογισμού άρτιας ισοτιμίας. Στο παράθυρο Project Manager επιλέξτε Add Sources. Στα επόμενα βήματα θα πρέπει να καθορίσετε το πηγαίο αρχείο. Το πρώτο 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)

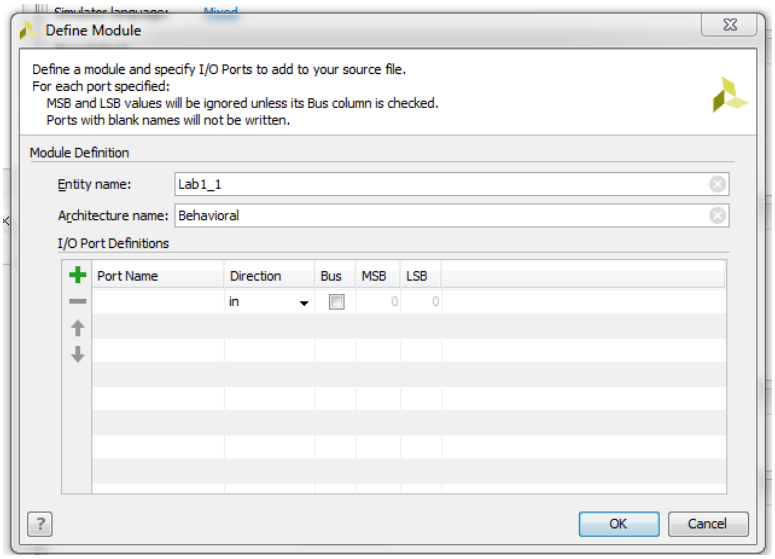
Στην συνέχεια μπορείτε να επιλέξετε να προσθέσετε νέο αρχείο (Add Files), νέο κατάλογο όπου περιέχονται ένα ή περισσότερα πηγαία αρχεία (Add Directories) ή να δημιουργήσετε νέο αρχείο (Create File). Επιλέξτε Create File.

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)

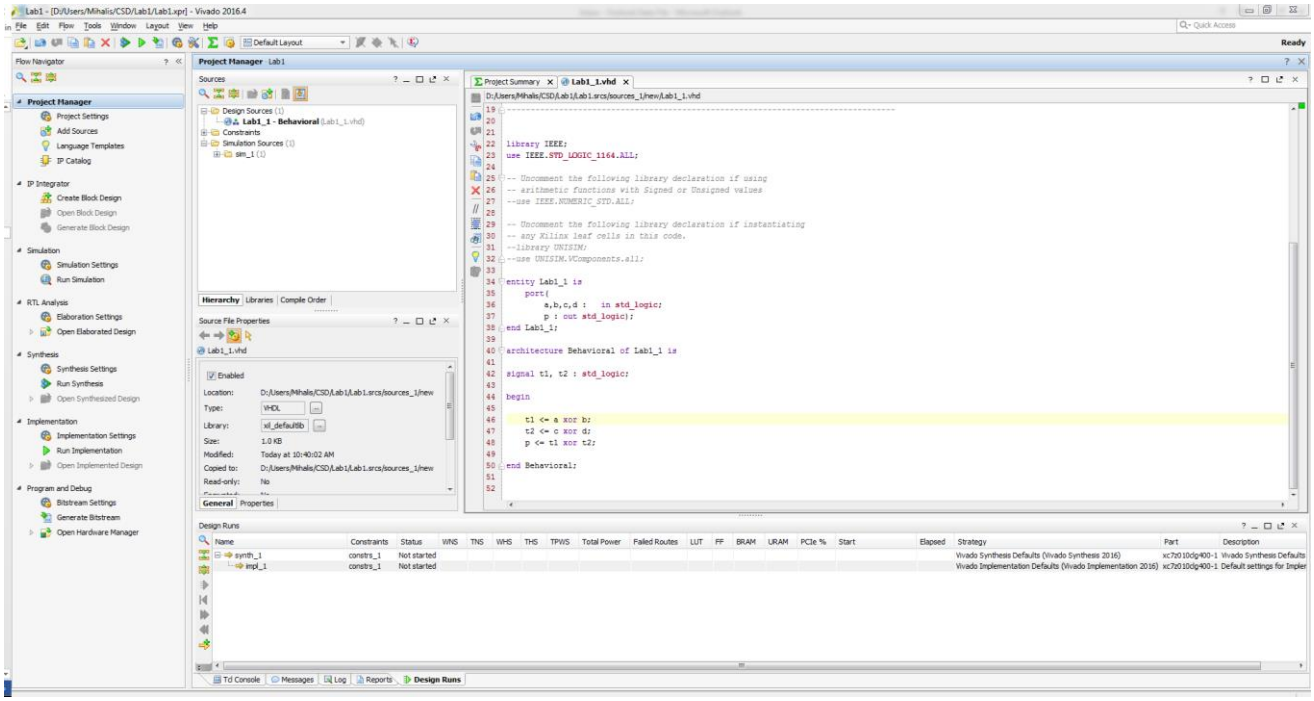
Στον επεξεργαστή κειμένου, μερικές από τη βασικές δομές του VHDL αρχείου είναι ήδη γραμμένες. Οι λέξεις κλειδιά (keywords) της γλώσσας απεικονίζονται με μπλε χρώμα, οι τύποι δεδομένων με κόκκινο, τα σχόλια με γκρι, και οι τιμές με μαύρο. Αυτή η κωδικοποίηση ανάλογα με το χρώμα ενισχύει την αναγνωσιμότητα 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 Lab-vhdl-models.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.



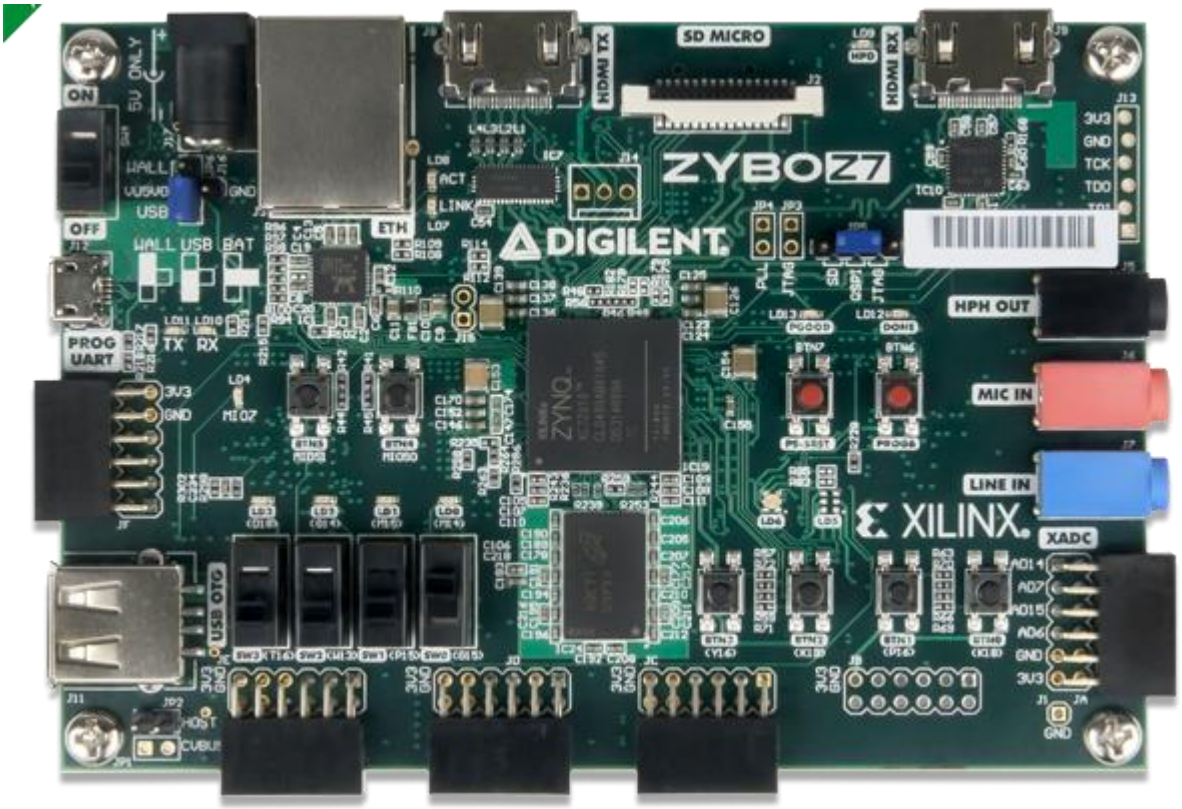
Εικόνα 8: Ολοκληρωμένη σχεδίαση

1.3 Προσομοίωση (Simulation)

Υπό κατασκευή.

1.4 Εκπαιδευτική πλακέτα Zybo

Πρώτα όμως εξοικειωθείτε με την πλακέτα FPGA που θα χρησιμοποιήσετε: Digilent Zybo Z7-10 board (βλέπε Εικόνα 9). Στην ιστοσελίδα του μαθήματος μπορείτε να βρείτε εγχειρίδιο χρήσης της πλακέτας.



Εικόνα 9 Zybo Z7-10 board

Η πλακέτα Zybo Z7-10 περιέχει τα εξής (στα πλαίσια του εργαστηρίου θα χρησιμοποιήσετε ότι είναι με bold):

- ZYNQ Processor, 667 MHz dual-core Cortex-A9 processor
- DDR3L memory controller with 8 DMA channels and 4 High Performance AXI3 Slave ports
- High-bandwidth peripheral controllers: 1G Ethernet, USB 2.0, SDIO
- Low-bandwidth peripheral controllers: SPI, UART, CAN, I2C
- Programmable from JTAG, Quad-SPI flash, and microSD card
- **ZYNQ XC7Z010-1CLG400C**
- Memory
    - 1 GB DDR3L with 32-bit bus @ 1066 MHz
    - 16 MB Quad-SPI Flash with factory programmed 128-bit random number and 48-bit globally unique EUI-48/64™ compatible identifier
    - microSD slot
- USB and Ethernet
    - Gigabit Ethernet PHY
    - **USB-JTAG Programming circuitry**
    - USB-UART bridge
- USB 2.0 OTG PHY with host and device support
- Audio and Video
    - Pcam camera connector with MIPI CSI-2 support
    - HDMI sink port (input) without CEC (Zybo Z7-10)
    - HDMI source port (output) with CEC
    - Audio codec with stereo headphone, stereo line-in, and microphone jacks

- **Switches, Push-buttons, and LEDs**
  - **6 push-buttons (2 processor connected)**
  - **4 slide switches**
  - **5 LEDs (1 processor connected)**
  - **1 RGB LED (Zybo Z7-10)**
- Expansion Connectors
  - 5 Pmod Ports (Zybo Z7-10)
  - 8 Total Processor I/O
  - 32 Total FPGA I/O (Zybo Z7-10)
  - 4 Analog capable 0-1.0V differential pairs to XADC
- Και άλλα συστατικά που δεν θα χρησιμοποιηθούν στα πλαίσια του εργαστηρίου

1.5 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.

1.6 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 G15

SW1 → FPGA Pin P15

SW2 → FPGA Pin W13

SW3 → FPGA Pin T16

LD0 → FPGA Pin M14

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 (επιλέξτε ένα από τα αρχεία Zybo-Master.xdc ή Zybo-Z7-Master.xdc ανάλογα με την πλακέτα που έχετε).

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 Zybo-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.

1.7 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 2: Συνδυαστικά κυκλώματα με VHDL

Σε αυτήν την εργαστηριακή άσκηση θα ασχοληθείτε με την σχεδίαση απλών συνδυαστικών κυκλωμάτων με χρήση της γλώσσας VHDL. Πιο συγκεκριμένα θα σχεδιάσετε:

- Ένα κύκλωμα πλειοψηφίας των 3 εισόδων
- Έναν αθροιστή των 2-bit
- Έναν συγκριτή δύο αριθμών των 2-bit
- Έναν απαριθμητή άσσων σε έναν δυαδικό αριθμό

Μέρος Α: Χρήση concurrent signal assignments

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 majority.vhd.
- Η οντότητα της μονάδας είναι η εξής:

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

- Υλοποιήστε τη λογική συνάρτηση της εξόδου f με βάση τη συνάρτηση f(a,b,c) = a*b + b*c + a*c
- 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 2-bit

Σχεδιάστε ένα αθροιστή που προσθέτει δύο απρόσημους αριθμούς των 2-bit A=A1 A0 και B=B1 B0 και παράγει άθροισμα των 2-bit S=S1 S0 και κρατούμενο εξόδου C. Χρησιμοποιήστε τους διακόπτες SW3-SW0 για τις εισόδους A και B και τα LEDs LED2-LED0 για τις εξόδους S και C.

- Δημιουργήστε ένα νέο project.
- Δημιουργήστε ένα νέο αρχείο (vhdl module) με όνομα adder_2b.vhd.
- Η οντότητα της μονάδας είναι η εξής:

```
entity adder_2b is
   port (
        A, B  : UNSIGNED(1 DOWNTO 0);
        S     : UNSIGNED(1 DOWNTO 0);
        C     : STD_LOGIC
   );
end adder_2b;
```

- 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: Συγκριτής δυαδικών αριθμών

Σχεδιάστε έναν συγκριτή 2 απρόσημων δυαδικών αριθμών των 2-bit. 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 **when-else**.
- Συνθέστε και υλοποιήστε το κύκλωμα εισάγοντας το κατάλληλο ucf file.
- Προγραμματίστε την πλακέτα και ελέγξτε την λειτουργία του κυκλώματος.

Μέρος Β: Χρήση sequential signal assignments

Άσκηση 4: Συγκριτής δυαδικών αριθμών

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

- 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 5: 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 **if-then-else**. Εναλλακτικά μπορείτε να χρησιμοποιήσετε και 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.

Μέρος Γ: Χρήση δομικής περιγραφής (structural description)

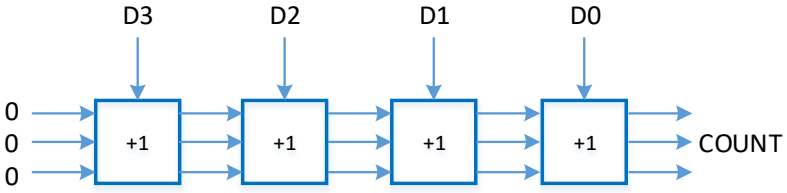
Άσκηση 6: Απαριθμητής άσσων σε έναν δυαδικό αριθμό

Σχεδιάστε τον απαριθμητή άσσων που είχατε σχεδιάσει και στην άσκηση 5 χρησιμοποιώντας την δομική περιγραφή του παρακάτω Σχήματος.

- Σχεδιάστε πρώτα την οντότητα plus1 (+1):

```
entity plus1 is
   port (
        D     : in  STD_LOGIC;
        COUNT : out UNSIGNED(2 DOWNTO 0)
   );
end plus1;
```

- Σχεδιάστε το κύκλωμα χρησιμοποιώντας δομική περιγραφή και τέσσερα instances της οντότητας plus1.



Εργαστηριακή Άσκηση 3: Ακολουθιακά κυκλώματα με VHDL

Σε αυτήν την εργαστηριακή άσκηση θα ασχοληθείτε με την σχεδίαση απλών ακολουθιακών κυκλωμάτων με χρήση της γλώσσας VHDL. Πιο συγκεκριμένα θα σχεδιάσετε:

- Έναν δυαδικό μετρητή των 4-bit
- Ένα κύκλωμα διαίρεσης συχνότητας
- Έναν συσσωρευτή (accumulators) των 4-bit

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 Lab-vhdl-models.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: 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)

- 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.
- 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.

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. Διαφορετικά, για κάθε πάτημα του κουμπιού, όσο και γρήγορο να είναι αυτό, θα προσθέτει πολλές φορές! Για να το πετύχετε αυτό εισάγετε στην αρχιτεκτονική σας το παρακάτω κύκλωμα (προσθέστε και τα αντίστοιχα σήματα στην αρχιτεκτονική).

```
ENABLE_button: process (clk)
begin
     if clk'event and clk = '1' then
          ENABLE_q1 <= ENABLE;
          ENABLE_q2 <= ENABLE_q1;
     end if;
end process;
ENABLE_pulse <= ENABLE_q1 and (not ENABLE_q2);
```

Το παραπάνω κύκλωμα για κάθε πάτημα του κουμπιού ENABLE παράγει έναν παλμό διάρκειας μίας περιόδου του ρολογιού CLK. Χρησιμοποιήστε το σήμα ENABLE_pulse ως σήμα ενεργοποίησης (enable) του συσσωρευτή αντί να χρησιμοποιήσετε απευθείας το σήμα από το κουμπί ENABLE.

Εργαστηριακή Άσκηση 4: 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)

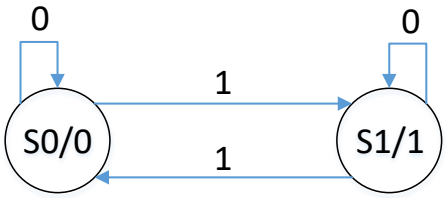
Άσκηση 1: Γεννήτρια ισοτιμίας

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 3 (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 5 και 6: Επεξεργαστής RISC-V neorv-32

To be added