Παράρτημα Γ΄

Σχεδίαση σε επαναδιαμορφώσιμη Αρχιτεκτονική Altera FPGA

Όπως έχει γίνει κατανοητό στα προηγούμενα κεφάλαια, το οικοσύστημα των ενσωματωμένων συστημάτων καλύπτει ένα πολύ μεγάλο εύρος απαιτήσεων και για αυτό το λόγο χρησιμοποιούνται πλήθος τεχνολογιών υλοποίησης. Εκτός από τους επεξεργαστές των 8bit και των 32bit, αρκετά συχνά χρησιμοποιούνται και οι επαναδιαμορφώσιμες αρχιτεκτονικές, όπως Altera και Xilinx. Οι αρχιτεκτονικές αυτές χρησιμοποιούνται τόσο στα πρώτα στάδια του σχεδιασμού για τη γρήγορη προτυποποίηση και τη δημιουργία του πρωτοτύπου, όσο και στην τελική υλοποίηση, σε προβλήματα που δε μπορούν να υλοποιηθούν οικονομικά και αξιόπιστα με άλλες επιλογές. Σε αυτό το παράρτημα, θα γίνει η παρουσίαση της αναπτυξιακής πλακέτας της Altera DE2-115 και στη συνέχεια θα ακολουθήσουν οι ασκήσεις εμβάθυνσης σε αυτή την πλατφόρμα.

Γ΄.1 Εισαγωγή

Η αναπτυξιακή πλακέτα FPGA που θα χρησιμοποιήσουμε είναι η Altera DE2-115 (Εικόνα Γ΄.1), η οποία αν και είναι σχεδιασμένη κατά βάση για ακαδημαϊκούς σκοπούς (διδασκαλία και έρευνα), εντούτοις δεν υπολείπεται ως προς τη λειτουργικότητά της και μπορεί να χρησιμοποιηθεί ακόμη και σε πραγματικές εφαρμογές ενσωματωμένων συστημάτων. Εξάλλου, η διαδικασία ανάπτυξης ενός ενσωματωμένου συστήματος σε μια πλακέτα FPGA είναι η ίδια σε πλακέτες χαμηλής ή υψηλής πολυπλοκότητας ακόμη και σε διαφορετικές εταιρίες.

Η πλακέτα αυτή διαθέτει ένα ισχυρά επαναδιαμορφώσιμο ολοκληρωμένο κύκλωμα το Altera Cyclone IV και όλα τα σημαντικά της στοιχεία είναι συνδεδεμένα με τους ακροδέκτες (pins) του Cyclone, παρέχοντας τη δυνατότητα στο χρήστη να μπορεί να ρυθμίσει την σύνδεση πολλαπλών στοιχείων με τον

τρόπο που επιθυμεί. Για την εύκολη διεξαγωγή πειραμάτων, η πλακέτα αυτή περιλαμβάνει έναν ικανοποιητικό αριθμό διακοπτών, φωτεινών LED, και οθόνες 7 στοιχείων (7-segment displays). Για πιο προχωρημένες εφαρμογές, περιλαμβάνει μνήμες SRAM, SDRAM και επίσης μη πτητική μνήμη FLASH που διατηρεί τα δεδομένα ή την επαναδιαμόρφωση. Για τα προβλήματα που απαιτούν προγραμματιζόμενο επεξεργαστή και συσκευές εισόδου εξόδου, μπορεί να χρησιμοποιηθεί ο επεξεργαστής μαλακού πυρήνα (soft-core) που περιγράφεται σε HDL, Nios II της Altera, ο οποίος μπορεί με τον κατάλληλο προγραμματισμό να εκμεταλλευτεί όλες τις διεπαφές της αναπτυξιακής πλακέτας. Επίσης περιλαμβάνει δυνατότητες όπως τυποποιημένους υποδοχείς για επεξεργασία σημάτων εικόνας και ήχου, καθώς και τη δυνατότητα σύνδεσης πλακετών, σχεδιασμένων από το χρήστη, μέσω δύο κεφαλών επέκτασης τα οποία όμως είναι προχωρημένα θέματα και δε θα εξεταστούν σε αυτό το παράρτημα.

Τα παρακάτω στοιχεία περιλαμβάνονται στην πλακέτα DE2-115:

- FPGA
  - Cyclone V SoC 5CSEMA5F31 / EPCQ256 256-Mbit serial configuration device
- I/O Διεπαφές
  - Εσωτερική USB-Blaster για την επαναδιαμόρφωση του FPGA
  - Είσοδος/Έξοδος γραμμής ήχου, είσοδος μικροφώνου (24-bit)
  - Έξοδος Video (VGA 24-bit DAC)
  - Είσοδος Video (NTSC/PAL/SECAM)
  - Θύρα υπερύθρων
  - Σειριακή θύρα RS232 DB9
  - 10/100/1000 Ethernet (2 θύρες)
  - Δυο θύρες USB 2.0 Host (Type A/B)
  - PS/2 πληκτρολόγιο και mouse
  - Αναμονές επέκτασης (40-pin headers)
- Μνήμη
  - 128MB SDRAM, 2MB SRAM, 8MB Flash, 32Kbit EEPROM
  - Υποδοχή για Micro SD κάρτα μνήμης
- Απεικόνιση

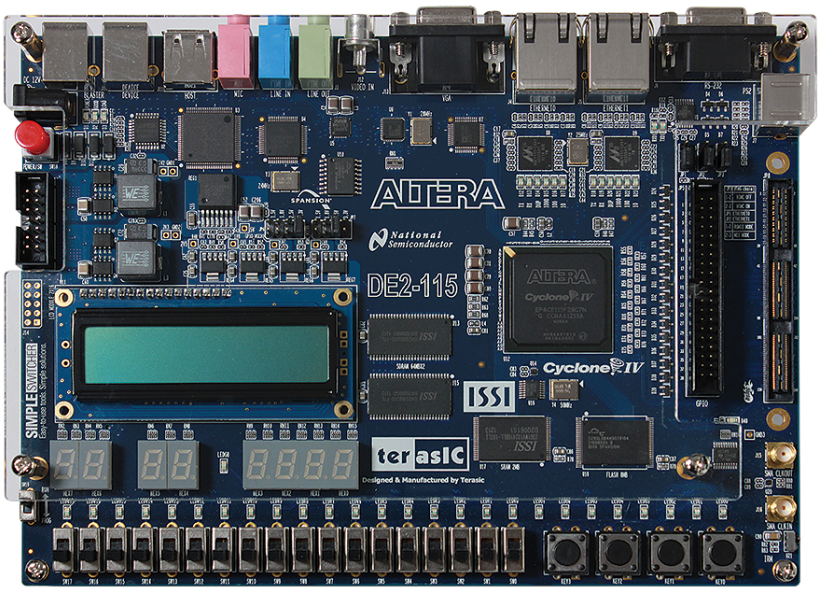
  - 8 οθόνες 7-segment
  - 1 οθόνη LCD 2x16

- Switches and LEDs
  - 18 διακόπτες
  - 18 LEDs
  - 4 διακόπτες με αποκλυδωνισμό

- Ρολόγια
  - 50 MHz clock (3 ρολόγια)

Για να αναπτυχθεί ένα ενσωματωμένο σύστημα σε οποιαδήποτε πλακέτα της Altera χρησιμοποιείται το λογισμικό Quartus. Υπάρχουν 2 εκδόσεις του προγράμματος, που μπορούν να μεταφορτωθούν δωρεάν από τον ιστοχώρο της Altera (www.altera.com). Η δωρεάν έκδοση (web), η οποία είναι μεν πλήρης άλλα φέρει κάποιους περιορισμούς ως προς τις δυνατότητες των εργαλείων (π.χ. δεν υποστηρίζεται η παράλληλη εκτέλεση, και όλα εκτελούνται σε έναν μόνο πυρήνα), και η επαγγελματική έκδοση (subscription), η οποία απαιτεί ένα αρχείο με άδειες που έχει αγοράσει κάποιος από την Altera και οι οποίες καθορίζουν ακριβώς τα εργαλεία, τα διαθέσιμα δομοστοιχεία (IP) και τις δυνατότητες που έχει στη διάθεσή του ο σχεδιαστής. Στα πλαίσια των εργαστηριακών ασκήσεων θα χρησιμοποιήσουμε τη web έκδοση, η οποία μας καλύπτει πλήρως. Η τελευταία έκδοση του Quartus (Ιούλιος 2015) είναι η 15, αλλά στα πλαίσια αυτών των ασκήσεων χρησιμοποιούμε την έκδοση 13, η οποία είναι διαθέσιμη για μεταφόρτωση και δεν έχει πολλές διαφοροποιήσεις σε σχέση με την 15. Όμως, το πιο σημαντικό στοιχείο είναι ότι αυτή είναι η τελευταία έκδοση που υποστηρίζει 32bit και 64bit συστήματα, από Windows XP και άνω, και έτσι μπορεί να χρησιμοποιηθεί σε όλους τους υπολογιστές που κατασκευάστηκαν από το 2001 και μετά. Κατά την αγορά της πλακέτας Altera DE2-115 υπάρχει το λογισμικό σε CD και έτσι δε χρειάζεται να μεταφορτωθεί από το Internet. Μαζί με το λογισμικό, υπάρχουν και αρκετά αρχεία επίδειξης των δυνατοτήτων της πλακέτας.

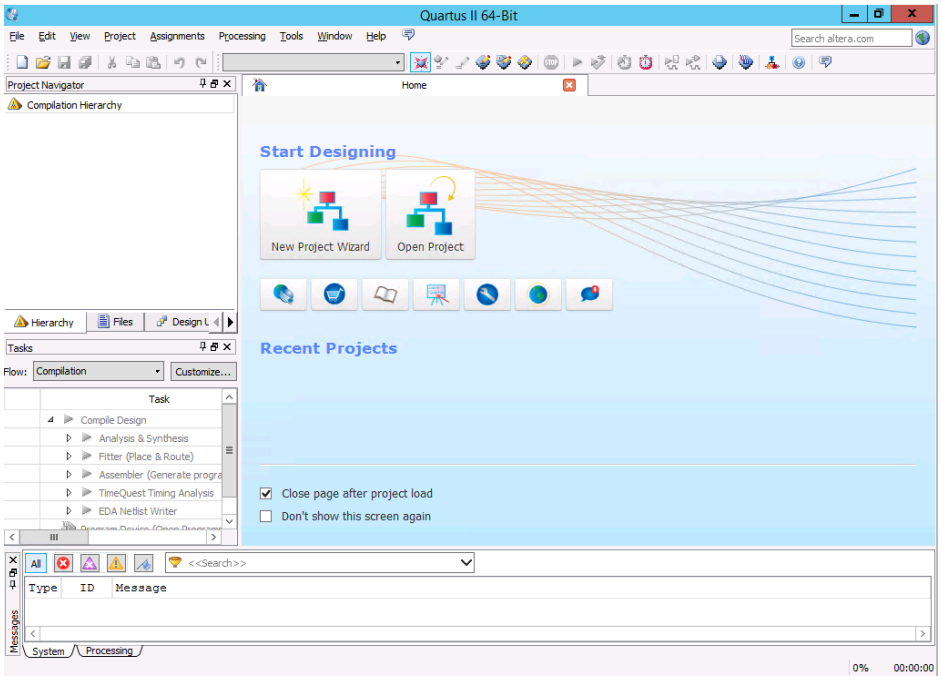
Όπως προαναφέρθηκε, οι δυο μεγαλύτεροι πάροχοι επαναδιαμορφώσιμων αρχιτεκτονικών Altera και Xilinx, υποστηρίζουν πλήρως τις πλακέτες που κατασκευάζουν με τα ολοκληρωμένα λογισμικά ανάπτυξης, επεξεργασίας και μεταφόρτωσης κώδικα HDL ή σχηματικού. Η βασική διαδικασία είναι η ίδια, με τη μόνη διαφορά την χρήση διαφορετικών περιβαλλόντων σχεδίασης και διαφορετικών ονομάτων στα εργαλεία. Κατά τη διαδικασία ανάπτυξης ενός ενσωματωμένου συστήματος σε μια πλακέτα FPGA, χρησιμοποιείται συνήθως μια



Σχήμα Γ΄.1: Η αναπτυξιακή πλακέτα που χρησιμοποιείται στα εργαστήρια είναι η Altera DE2-115 που μπορεί να την προμηθευτεί κάποιος σε χαμηλή ακαδημαϊκή τιμή από το διαδίκτυο.

γλώσσα περιγραφής υλικού όπως Verilog ή VHDL, η οποία αρχικά προσομοιώνεται και επιβεβαιώνεται η ορθή της λειτουργία, στη συνέχεια δημιουργείται το αρχείο περιορισμών (π.χ. χρονικών περιορισμών) και η αντιστοίχηση στους ακροδέκτες του FPGA (σύμφωνα με την αναπτυξιακή πλακέτα) και τέλος η μεταφόρτωση στην πλακέτα.

Το Altera Quartus αποτελεί ένα πλήρες περιβάλλον για τον σχεδιασμό συστήματος σε επαναπρογραμματιζόμενο chip (System-On-a-Programmable-Chip - SOPC). Περιλαμβάνει εργαλεία για όλες τις φάσεις σχεδιασμού της πλακέτας. Διαθέτει δικό του γραφικό περιβάλλον (GUI) καθώς και διεπαφή γραμμής εντολών (command-line interface) επιτρέποντας την χρήση οποιοδήποτε εργαλείου που περιέχει και με τους 2 τρόπους.



Σχήμα Γ΄.2: Στην περιοχή Tasks του εργαλείου Altera Quartus, ο χρήστης μπορεί να επιλέξει να εκτελέσει ξεχωριστά το εργαλείο που χρειάζεται.

Το αναπτυξιακό περιβάλλον της Altera, παρέχει σε ένα ενοποιημένο πρόγραμμα όλες τις λειτουργίες που απαιτούνται για το σχεδιασμό ενός ολοκληρωμένου κυκλώματος (Εικόνα Γ΄.2) Το Quartus περιλαμβάνει έναν αρθρωτό μεταγλωττιστή ο οποίος αποτελείται από τα εξής τμήματα (modules):

- Ανάλυση & Σύνθεση (Analysis & Synthesis)
- Συγχώνευση Κατατμήσεων (Partition Merge)
- Εφαρμοστή FPGA (Fitter)
- Συναρμολογητή (Assembler)
- Αναλυτή Χρονισμού (TimeQuest)
- Βοηθό Σχεδιασμού (Design Assistant)
- Δημιουργία αρχείου netlist (EDA Netlist Writer)
- HardCopy Netlist Writer

Για την πλήρη μεταγλώττιση, η οποία περιλαμβάνει όλα τα ανωτέρω (modules), αφού εκτελέσουμε το Quartus II, και δημιουργήσουμε όλα τα αρχεία του project μας, επιλέγουμε το "Start Compilation" που βρίσκεται στο μενού Processing.

Σε περίπτωση που επιθυμούμε να εκτελέσουμε κάθε module ξεχωριστά, κάνουμε κλικ στο (Start) από το ίδιο μενού και επιλέγουμε το εργαλείο που θέλουμε να χρησιμοποιήσουμε. Εναλλακτικά, αυτό μπορεί να γίνει και από το παράθυρο (Tasks) από το οποίο μπορούμε επίσης να αλλάξουμε τις ρυθμίσεις ή να δούμε το αρχείο αναφοράς για το εργαλείο που μας ενδιαφέρει (Εικόνα Γ΄.3).

Τα παρακάτω βήματα περιγράφουν τη βασική ροή σχεδιασμού στο Altera Quartus:

1. Για την δημιουργία νέου project και το καθορισμό της συσκευής ή της οικογένειας συσκευών, κάνουμε κλικ στο New Project Wizard στο μενού File.

2. Μπορούμε να χρησιμοποιήσουμε τον ενσωματωμένο επεξεργαστή κειμένου (Text Editor) για να δημιουργήσουμε το κύκλωμά μας σε μια γλώσσα περιγραφής υλικού (harware description language, HDL) όπως τις προτυποποιημένες (Verilog, VHDL) ή την Altera Hardware Descrition Language (AHDL).

3. Χρησιμοποιούμε τον επεξεργαστή block (Block Editor) για να δημιουργήσουμε ένα διάγραμμα μπλοκ με σύμβολα που αναπαριστούν άλλα αρχεία σχεδιασμού ή για να δημιουργήσουμε ένα σχηματικό (schematic).

4. Με το εργαλείο (MegaWizard Plug-in Manager), μπορούμε να δημιουργήσουμε δικές μας παραλλαγές από συναρτήσεις υλοποιημένες σε υλικό (megafunctions) και δομοστοιχεία IP (IP functions).

5. Για το σχεδιασμό συστήματος-πάνω-σε-ψηφίδα χρησιμοποιούμε το (SOPC Builder ή QSYS) ή τον DSP Builder.

6. Για την αντιστοίχηση του κυκλώματος πάνω σε ακροδέκτες (pin) του FPGA chip, μπορούμε να καθορίζουμε τους σχεδιαστικούς περιορισμούς (constraints) με τον Assignment Editor, ή το Pin Planner.

7. Μπορεί προαιρετικά να γίνει πρόωρη εκτίμηση των χρονισμών πριν τη διαδικασία υλοποίησης και σύνθεσης (fitting).

8. Η σύνθεση του σχεδίου γίνεται με το Analysis & Synthesis.

9. Σε περίπτωση που το σχέδιο είναι χωρισμένο σε διαμερίσματα (partitions), μπορούμε να τα συγχωνεύσουμε με το εργαλείο Partition Merge.

10. Προαιρετικά μπορεί να γίνει λειτουργική προσομοίωση με χρήση του προσομοιωτή Modelsim Starter που περιλαμβάνεται (και είναι ελεύθερο στη χρήση) στην εγκατάσταση.

11. Η τοποθέτηση και δρομολόγηση του σχεδίου μας γίνεται με το εργαλείο (Fitter).

12. Μπορεί να γίνει εκτίμηση και ανάλυση της κατανάλωσης με το εργαλείο (PowerPlay Power Analyzer).

13. Με το TimeQuest Timing Analyzer γίνεται ανάλυση των χρονισμών.

14. Η διόρθωση κάποιον προβλημάτων χρονισμού μπορεί να γίνει με τη συνδυασμένη χρήση των εργαλείων Chip Planner, LogicLock και Assignment Editor.

15. Το αρχείο προγραμματισμού του chip δημιουργείται με τον Assembler και έπειτα προγραμματίζεται με το Programmer.

16. Μπορεί να γίνει αποσφαλμάτωση με την παρακολούθηση των σημάτων πάνω στην πλακέτα με το εργαλείο SignalTrap II Logic Analyzer, ή έναν εξωτερικό λογικό αναλυτή.

Αξίζει να αναφερθεί ότι όλα τα εργαλεία υποστηρίζουν πλήρως τη γραμμή εντολών κάτι που βοηθάει την ομαδοποιημένη εκτέλεση λειτουργιών (batch operations).

Προκειμένου να εκμεταλλευτεί καλύτερα κάποιος τα εργαλεία, προτείνεται ο έλεγχος και η προσομοίωση του κώδικα να γίνεται σε ένα περιβάλλον όμως το modelsim (που περιλαμβάνεται στην ελεύθερη web έκδοση), και μόνο όταν επιτευχθεί η επιβεβαίωση της ορθής λειτουργίας, να χρησιμοποιείται το Quartus για τη σύνθεση. Προτείνεται λοιπόν να αποφεύγεται η σύνθεση κώδικα που δεν έχει επιβεβαιωθεί, γιατί η σύνθεση είναι μια διαδικασία αρκετά χρονοβόρα (η δωρεάν έκδοση του Quartus δεν υποστηρίζει πολλαπλούς επεξεργαστές). Όσες λιγότερες φορές επαναληφθεί λοιπόν η σύνθεση, τόσο περισσότερο χρόνο θα έχει ο σχεδιαστής για να ασχοληθεί με κάποιο άλλο θέμα. Επίσης, η συγγραφή του κώδικα μπορεί να γίνει στο Quartus όπου από το μενού EDIT, επιλογή Insert Templates, παρέχονται στο σχεδιαστή αρκετοί σκελετοί κώδικα (σε Verilog ή VHDL, κτλ) που μπορούν να τοποθετηθούν στο ανοιχτό έγγραφο άμεσα πατώντας το κουμπί Insert. Ο σχεδιαστής απλώς θα πρέπει να τροποποιήσει σε

ελάχιστα σημεία τις γραμμές που τον ενδιαφέρουν και να απομακρύνει τις υπόλοιπες, γλυτώνοντας πολύ χρόνο σε σύγκριση με το να πληκτρολογούσε όλες τις γραμμές.

Γ΄.2 Εργαστηριακή Άσκηση 1

Σε αυτή την άσκηση γνωριμίας, θα δημιουργήσετε ένα απλό project στο Quartus και θα ακολουθήσετε όλη τη ροή σχεδιασμού έως τη μεταφορά στην πλακέτα και την επιβεβαίωση ορθής λειτουργίας.

Δημιουργία Νέου Project

- Επιλέξτε New Project Wizard από το μενού File.
- Στο παράθυρο του Wizard θα σας ζητηθούν (Εικόνα Γ΄.4):
  - Κατάλογος Εργασίας
  - Όνομα Project
  - Προσδιορισμός του ονόματος της top-level οντότητας.
  - Τον τύπο της συσκευής ή την οικογένεια συσκευών, το package type κλπ

Στην 1η οθόνη του οδηγού, ο χρήστης συμπληρώνει 3 πεδία, τον κατάλογο εργασίας του project, το όνομα του project και το όνομα του top-level design entity για το project, το οποίο συνήθως είναι το ίδιο με το όνομα του project και για αυτό προσυπληρώνεται από το Quartus με το όνομα του project. Το όνομα του top-level design είναι μία παράμετρος που μπορεί να τροποποιηθεί, κατά την ανάπτυξη του project. Μάλιστα, μια καλή πρακτική σε ένα project που αναπτύσσεται από την αρχή, είναι η τοποθέτηση στο όνομα του top-level design (οντότητα σχεδιαστικής κορυφής), του πρώτου component που θα αναπτυχθεί, και μόλις γίνει η σύνθεση και επιβεβαιωθεί η ορθή λειτουργία, να τροποποιηθεί στην επόμενη ιεραρχικά οντότητα που θα δημιουργηθεί, ώστε το project να ακολουθεί τη διαδικασία ανάπτυξης bottom-up, δηλαδή από τα χαμηλότερα σχεδιαστικά modules προς τα υψηλότερα.

Στη 2η οθόνη του οδηγού υπάρχει η προτροπή για την προσθήκη των αρχείων ή των βιβλιοθηκών (Εικόνα Γ΄.5). Σε αυτό το σημείο μπορούν να προστεθούν οτιδήποτε τύπου σχεδιαστικά αρχεία υποστηρίζει το Quartus, όπως αρχεία VHDL, Verilog, netlists, σχηματικά κτλ. Επίσης, σε περίπτωση που απαιτούνται κατάλογοι με υποστηρικτικές βιβλιοθήκες, μπορούν να προστεθούν από την επιλογή User Libraries.

Η επόμενη οθόνη (οθόνη 3) επιτρέπει στο χρήστη να επιλέξει τα χαρακτηριστικά τεχνολογίας για το project του και συγκεκριμένα την οικογένεια και το FPGA chip. Για να χρησιμοποιήσει την πλακέτα Altera DE2-115, ο χρήστης θα πρέπει να επιλέξει το Device Family Cyclone IV E και να περιορίσει τα αποτελέσματα που εμφανίζονται χρησιμοποιώντας το Name filter, όπως φαίνεται στην Εικόνα Γ΄.6. Το ολοκληρωμένο κύκλωμα FPGA έχει τυπωμένο το όνομα του, οπότε κάποιος μπορεί να το δει στην πλακέτα. Συνήθως, η Altera DE2-115 φέρει το EP4CE115F29C7 ή EP4CE115F29C8 (όποιο και να επιλεχθεί θα λειτουργήσει ορθά σε αυτές τις εργαστηριακές ασκήσεις, αφού είναι συμβατά ως προς τις τάσεις και τον αριθμό των στοιχείων).

Στο τελευταίο βήμα (4ο) των ρυθμίσεων, ρυθμίζουμε τα εργαλεία που χρησιμοποιήσαμε ή θα χρησιμοποιήσουμε για την ανάπτυξη του project. Συνήθως, στο Simulation επιλέγουμε το ModelSim-Altera, και στο Design Entry/Synthesis, επιλέγουμε Custom/VHDL (Εικόνα Γ΄.7). Μόλις πατήσουμε NEXT, εμφανίζεται μια οθόνη σύνοψης των πληροφοριών του project (Summary) και μόλις ο χρήστης πατήσει Finish, ολοκληρώνεται η διαδικασία δημιουργίας νέου project.

Μία δυνατότητα που συναντάται σε κάθε ολοκληρωμένο περιβάλλον ανάπτυξης επαναδιαμορφώσιμων αρχιτεκτονικών, είναι η δυνατότητα σύνδεσης συγκεκριμένων ακροδεκτών του FPGA chip με συγκεκριμένες διεπαφές εισόδου-/εξόδου του κυκλώματος του. Για παράδειγμα, αν κάποιος θέλει να ενεργοποιηθεί ένα led όταν η λογική έξοδος του κυκλώματος του ledout λάβει την τιμή 1, θα πρέπει να συνδέσει τη λογική έξοδο ledout της HDL οντότητάς του, με το pin του FPGA που συνδέεται σε αυτό το led. Κάθε ολοκληρωμένο κύκλωμα έχει πολλούς ακροδέκτες, και για κάθε αναπτυξιακή πλακέτα υπάρχει μία αναλυτική λίστα με τις συνδέσεις όλων των ακροδεκτών. Η αντιστοίχηση των λογικών διεπαφών με τις φυσικές διεπαφές επιτυγχάνεται με τους παρακάτω τρόπους:

1. Με τον επεξεργαστή συνδέσεων ακροδεκτών (Pin Assignment Editor) (Εικόνα Γ΄.8)

2. Με την οπτική σύνδεση ακροδεκτών, όπου φαίνονται όλοι οι ακροδέκτες του chip (Εικόνα Γ΄.9)

3. Με την χρήση του CSV αρχείου που συνοδεύει την πλακέτα (DE2_115_pin_assignments.csv) και εισαγωγή στο project από το μενού Assignments, επιλογή Import Assignments, και στη συνέχεια χρήση των ονομάτων που φέρουν οι ακροδέκτες στο αρχείο αυτό, μέσα στο ανώτερο κύκλωμα υλοποίησης (top level design). Για παράδειγμα, μέσα σε αυτό το αρχείο υπάρχει η γραμμή CLOCK_50,Input,PIN_Y2,2,B2_N0,3.3-V LVTTL, οπότε αν στο κύκλωμα μας, χρησιμοποιήσουμε σε ένα σημείο την είσοδο CLOCK_50, τότε αυτή θα συνδεθεί άμεσα με το pin Y2 που στην Altera DE2-115 αντιστοιχεί το ρολόι των 50 Mhz.

Σε περίπτωση που ο χρήστης δώσει λάθος ακροδέκτη (π.χ. έναν ακροδέκτη που δεν υπάρχει), τότε θα εμφανιστεί κατά τη διαδικασία της τοποθέτησης και της δρομολόγησης, το μήνυμα Cannot Recognize Value Pin (Εικόνα Γ΄.10) Τέλος, να σημειωθεί ότι όταν πατηθεί ένα πλήκτρο (KEY[0] έως KEY[3]) παράγεται ένα λογικό '0' (και όχι '1' όπως θα περίμενε κάποιος), ενώ ο διακόπτης (slide switch), παράγει '1' όταν είναι στην άνω θέση και '0' όταν είναι στην κάτω.

Αφού ολοκληρωθεί με επιτυχία η σύνθεση, τότε ο χρήστης μπορεί να επιλέξει από τις συντομεύσεις ενεργειών TASKS ή από το μενού Tools την επιλογή Programmer. Αν έχουν εγκατασταθεί οι οδηγοί της πλακέτας (κάτι που γίνεται αυτόματα κατά την εγκατάσταση του Quartus II, εφόσον ο χρήστης επιλέξει υποστήριξη για τη συγκεκριμένη πλακέτα), θα υπάρχει η συσκευή USB BLASTER, η οποία είναι το υλικό πάνω στην πλακέτα που θα δεχτεί το bitstream και θα επαναπρογραμματίστει το FPGA (Εικόνα Γ΄.11).

Γ΄.3 Εργαστηριακή Άσκηση 2

Σε αυτή την άσκηση θα γίνει μια πιο εκτενής γνωριμία με το περιβάλλον Quartus της Altera. Συγκεκριμένα, θα γίνουν ασκήσεις πάνω στα περιφερειακά της πλακέτας. Αρχικά, δημιουργήστε ένα νέο project για την Altera DE2-115 και τοποθετήστε τον κώδικα Γ΄.3 σε ένα αρχείο με όνομα book01.vhd. Στο Project Navigator, επιλέξτε το αρχείο που μόλις έχετε δημιουργήσει και πατήστε δεξί κλικ και Set as Top Level Entity, για να γίνει η σύνθεση του συγκεκριμένου component. Να σημειώσετε ότι η ονομασία του αρχείου είναι πολύ σημαντική, επειδή όταν επιλεχθεί και τεθεί ως κορυφή της ιεραρχίας σχεδίασης, το Quartus σημειώνει το όνομα του αρχείου και περιμένει να βρει ένα entity με αυτό το όνομα. Οπότε, αν το όνομα του αρχείου χωρίς την κατάληξη .vhd είναι διαφορετικό από το όνομα του entity που περιλαμβάνει, δημιουργείται ένα πρόβλημα μη εύρεσης του top level component. Το συγκεκριμένο component, όπως φαίνεται, κάνει reset όταν πατηθεί ένα κουμπί, ενώ σε διαφορετική περίπτωση μεταφέρει την τιμή του διακόπτη σε ένα led κάθε 2.62ms (επειδή ο μετρητής αυξάνεται κατά 1 με συχνότητα 50 Mhz και μετράει ως 2**17-1, οπότε ο χρόνος υπολογίζεται ως $(2^{17}-1) * \frac{1}{50000000} = 0.00262142$.

Για να ολοκληρωθεί η σύνθεση, θα πρέπει να γίνει η αντιστοίχηση των θυρών του ανώτερου επιπέδου σχεδιασμού στα pins του FPGA. Συγκεκριμένα, θα πρέπει να γίνει η αντιστοίχηση των pins clk, button, switch, led.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity book01 is
6  port (clk: in std_logic;
7  button: in std_logic;
```

```
8  switch: in std_logic;
9  led: out std_logic);
10 end book01;
11
12 architecture behavioral of book01 is
13 signal counter: natural range 0 to 2**17-1; --0 every 2.62ms
14 begin
15 process(clk,button)
16 begin
17 if button='1' then
18    counter<=0;
19 elsif clk'event and clk='1' then
20    counter<=counter+1;
21    if counter=2**17-1 then
22       led<=switch;
23       counter<=0;
24    end if;
25 end if;
26 end process;
27 end behavioral;
```

Από το μενού Assignments επιλέγουμε τον Assignment Editor. Στο παράθυρο που ανοίγει ορίζουμε την αντιστοίχιση από σημείο σε σημείο και δίνουμε μία ονομασία στο σήμα μας όπως το ορίσαμε και στο αρχείο VHDL. Εναλλακτικά, η αντιστοίχηση μπορεί να γίνει δημιουργώντας ένα αρχείο qcf. Για την συγκεκριμένη άσκηση θα χρησιμοποιήσουμε ένα button για reset, ένα slide switch ως είσοδο, το ρολόι των 50Mhz, κι ένα led για έξοδο. Επιλέγουμε τα pins που θα χρησιμοποιήσουμε ως εξής: Από τα κουμπιά της πλακέτας επιλέγουμε το KEY3, από τα LED επιλέγουμε αυτό που αναγράφεται ως LEDG1, και από τους διακόπτες, επιλέγουμε τον SW11. Από το αρχείο περιορισμών της πλακέτας (DE2_115_pin_assignments.csv) βρίσκουμε τις γραμμές που μας ενδιαφέρουν, και είναι:

KEY[3],Input,PIN_R24,5,B5_N0,2.5 V,
SW[11],Input,PIN_AB24,5,B5_N2,2.5 V,
LEDG[1],Output,PIN_E22,7,B7_N0,2.5 V,
CLOCK_50,Input,PIN_Y2,2,B2_N0,3.3-V LVTTL,

Από τις παραπάνω γραμμές διαπιστώνουμε τα pins που πρέπει να αντιστοιχηθούν στα ports. Αυτό μπορεί να γίνει με 2 τρόπους: (α) με επεξεργασία του αρχείου qsf (quartus settings file) που βρίσκεται στον κατάλογο του project και φέρει όλες τις ρυθμίσεις του project μας, είτε (β) με τον επεξεργαστή συσχετίσεων. Αν επιλέξουμε τον (α) τρόπο, τότε θα πρέπει να προσθέσουμε τις παρακάτω γραμμές (αφορούν μόνο την Altera DE2-115, άλλη πλακέτα έχει άλλες αντιστοιχίσεις) στο τέλος:

set_location_assignment PIN_Y2 -to CLK
set_location_assignment PIN_R24 -to BUTTON

```
set_location_assignment PIN_AB24 -to SWITCH
set_location_assignment PIN_E22 -to LED
```

Αν επιλέξουμε τον (β) τρόπο, τότε θα πρέπει να τοποθετήσουμε τις 4 αναθέσεις στο τέλος της λίστας, όπως φαίνεται στην εικόνα Γ΄.12. Επιβεβαιώστε την ορθή λειτουργία του κυκλώματος, αφού προγραμματίστε την πλακέτα.

Επίσης, μπορούμε να δώσουμε κάποιους σχεδιαστικούς περιορισμούς, ώστε η σύνθεση να γίνει με τρόπο που να τους ικανοποιεί. Οι σχεδιαστικοί περιορισμοί εισάγονται με το File→New→Synopsis Design Constraints File. Στο παράθυρο που ανοίγει τοποθετούμε τη γραμμή + create_clock -period 20 [get_ports clk]+++, όπου ορίζουμε ότι το ρολόι θα πρέπει να έχει ελάχιστη περίοδο 20, και το αποθηκεύουμε στον κατάλογο του project με κατάληξη .sdc. Θα διαπιστώσουμε ότι το αρχείο μόλις αποθηκευτεί θα έχει προστεθεί στο παράθυρο project navigator στην καρτέλα Files.

Για να κάνουμε compile και synthesize το project που δημιουργήσαμε, τρέχουμε τον compiler επιλέγοντας Processing→Start Compilation. Καθώς η διαδικασία προχωράει από στάδιο σε στάδιο, αριστερά (παράθυρο tasks), βλέπουμε την αναφορά της πορείας του κάθε χρονική στιγμή. Μετά την ολοκλήρωσή του, εμφανίζεται ένα παράθυρο που μας πληροφορεί αν το compile έγινε επιτυχώς ή όχι και αν υπήρχαν warnings ή errors. Σε περίπτωση που πήγαν όλα καλά, μπορούμε να προσομοιώσουμε το σχέδιό μας ή να περάσουμε κατευθείαν στη διαδικασία προγραμματισμού της FPGA.

Αν επιθυμούμε να κάνουμε προσομοίωση του σχεδίου μας, ανοίγουμε τον τον Waveform Editor επιλέγοντας File→New. Κάνουμε κλικ στην καρτέλα Verification/Debugging Files, επιλέγουμε University Program VWF, πατάμε OK και σώζουμε το αρχείο με τη μορφή <project name>.vwf, στον κατάλογο του project (για να τοποθετηθεί αυτόματα στα files). Για να γίνει η προσομοίωση ορίζουμε τους χρόνους από το μενού του παραθύρου του Waveform Editor, Edit→End Time στα όρια που μας επιτρέπει. Μετά τοποθετούμε τα nodes (Edit→Insert Node or Bus) και είτε γράφουμε ένα ένα τα ονόματα των ports, είτε από το Node filter επιλέγουμε αμέσως όλες τις πόρτες. Στη συνέχεια επιλέγουμε Simulation→Functional Simulation ή Simulation→Timing Simulation, ανάλογα με την προσομοίωση που θέλουμε να γίνει, και στο τέλος μας εμφανίζονται τα αποτελέσματα σε ένα νέο παράθυρο.

Σε περίπτωση που το compile ολοκληρώθηκε επιτυχώς, μπορούμε να προγραμματίσουμε την FPGA. Η διαδικασία μεταφόρτωσης του σχεδίου μας στην FPGA καλείται FPGA Configuration. Για το configuration της Altera DE2-115 το Quartus υποστηρίζει 2 μεθόδους.

Η πρώτη καλείται Active Serial Programming και με αυτή το configuration bit stream μεταφορτώνεται στην quad serial configuration device (EPCQ256) με

μη πτητικό τρόπο που σημαίνει ότι ακόμη και αν απενεργοποιηθεί και ξαναενεργοποιηθεί η FPGA, οι πληροφορίες του configuration παραμένουν.

Η δεύτερη, την οποία και θα χρησιμοποιήσουμε σ' αυτήν την εργασία, καλείται JTAG Programming (Joined Test Action Group με βάση την τυποποίηση της IEEE) με την οποία το bit stream μεταφορτώνεται απευθείας στην FPGA και παραμένει εκεί μέχρις ότου την απενεργοποιήσουμε. Μόλις πατηθεί το reset χάνεται το configuration και επαναφορτώνεται αυτό που ήταν αποθηκευμένο στη μνήμη FLASH.

Για να γίνει το JTAG Programming ακολουθείστε τα εξής βήματα:

1. Ανοίξτε τον Programmer από το μενού Tools και κάντε κλικ στο Auto Detect

2. Επιλέξτε την εντοπισμένη συσκευή από το Hardware Setup (αν δεν είναι ήδη επιλεγμένη η USB-Blaster USB-0)

3. Πατήστε Start και θα μεταφερθεί αυτόματα ο κώδικας.

Μόλις ολοκληρωθεί η διαδικασία, μπορούμε να επιβεβαιώσουμε τη λειτουργία του σχεδίου μας στην FPGA. Πρέπει όταν πατάμε το switch να ανάβει το led. Όταν πατάμε το button πρέπει να κάνει reset.

Γ΄.4 Εργαστηριακή Άσκηση 3

Σε αυτή την άσκηση θα πρέπει να δημιουργηθεί ένας πολυπλέκτης 2-σε-1, ο οποίος θα χρησιμοποιεί 2 κουμπιά (KEY2, KEY1) για την είσοδο του πολυπλέκτη και ένα διακόπτη SW0 για την επιλογή της εισόδου. Η έξοδος θα εμφανίζεται στο LEDR0.

Το αρχείο VHDL που θα χρησιμοποιηθεί σε αυτή την άσκηση σας δίνεται στον κώδικα Γ΄.4. Αποθηκεύστε το αρχείο στον κατάλογο του project σας με όνομα mux2to1, και μόλις εμφανιστεί στα Files του Project Navigator, πατήστε δεξί κλικ στο αρχείο και Set As Top Level Entity.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  entity mux2to1 is
7  port ( A : in BIT;
8  B : in BIT;
9  S : in BIT;
10 X : out BIT);
11 end mux2to1;
12
13 architecture Behavioral of mux2to1 is
14 begin
```

```
15  X <= not ( (A and not S) or (B and S) );
16  end behavioral;
```

Να σημειωθεί ότι σε αυτή την άσκηση, τοποθετήθηκε η πύλη not στην έξοδο, γιατί τα LED στην Altera DE2-115 ανάβουν όταν τοποθετηθεί το '0', οπότε με την παραπάνω τροποποίηση γίνεται καλύτερα κατανοητό πότε η έξοδος φέρει το '1' δηλαδή την τιμή του κουμπιού που έχει επιλεχθεί (με το S) και είναι πατημένο. Στη συνέχεια βρίσκουμε τις αντιστοιχίσεις, και κάνουμε τις απαραίτητες συσχετίσεις στο αρχείο DE2_115_pin_assignments.csv, όπως προηγουμένως:

KEY[3],Input,PIN_R24,5,B5_N0,2.5 V,
KEY[2],Input,PIN_N21,6,B6_N2,2.5 V,
SW[0],Input,PIN_AB28,5,B5_N1,2.5 V,
LEDR[0],Output,PIN_G19,7,B7_N2,2.5 V,

Στην συγκεκριμένη άσκηση δεν χρειαζόμαστε το ρολόι για αυτό δεν θα δημιουργήσουμε αρχείο χρονισμών. Στη συνέχεια επιλέγουμε Start Compilation από το μενού Processing. Αν ολοκληρωθεί επιτυχώς προχωράμε στον προγραμματισμό της FPGA και επιβεβαιώνουμε την ορθή λειτουργία.

Γ΄.5 Εργαστηριακή Άσκηση 4

Σε αυτή την εργαστηριακή άσκηση θα δημιουργηθεί ένας μετρητής Up/Down. Συγκεκριμένα, θα χρησιμοποιείται το slide switch (SW0) το οποίο όταν είναι 1, η μέτρηση θα επιτρέπεται αλλιώς ο μετρητής θα σταματάει. Η μέτρηση θα ανανεώνεται με συχνότητα 1.49 Hz, ενώ θα χρησιμοποιείται ένα κουμπί για να κάνει reset. Η τιμή του μετρητή των 22 ψηφίων, θα εμφανίζεται στα 22 κόκκινα LED της πλακέτας. Θα χρησιμοποιηθεί ο κώδικας Γ΄.5. Δημιουργήστε ένα νέο project και τοποθετήστε τον κώδικα Γ΄.5. Θα πρέπει να εισάγεται το CSV αρχείο με τις αντιστοιχίσεις της πλακέτας, και επίσης θα πρέπει να δώστε την αντιστοίχηση για το κουμπί RST προς ένα οποιοδήποτε KEY της πλακέτας (KEY[0]/KEY[1]/KEY[2]/KEY[3]). Σημειώστε ότι δε χρειάζεται αντιστοίχηση για τη θύρα CLOCK_50, επειδή φέρει το ίδιο όνομά σε μια γραμμή του αρχείου των συσχετίσεων, δηλαδή

CLOCK_50,Input,PIN_Y2,2,B2_N0,3.3-V LVTTL,

Ομοίως, δε χρειάζεται να τοποθετηθεί συσχέτιση για τις θύρες SW ή το LEDR. Αποθηκεύστε το αρχείο με όνομα counter8.vhd στον κατάλογο του project, ώστε να τοποθετηθεί αυτόματα στο Project Navigator στην καρτέλα Files, και στη συνέχεια πατήστε δεξί κλικ και "Set as top Level Entity" και ολοκληρώστε τη

σύνθεση. Σημειώστε ότι τα κουμπιά της πλακέτας παράγουν το ’0’ όταν πατηθούν, οπότε για αυτό έχουμε προσδιορίσει τον κώδικα + if RST=’0’ then+++ για να δηλώσουμε όταν πατηθεί το συγκεκριμένο κουμπί.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  entity counter8 is
7  port(CLOCK_50: in std_logic; --clock
8  RST: in std_logic; --reset
9  SW: in std_logic_vector(1 downto 0);
10 LEDR: out std_logic_vector(7 downto 0));
11 end counter8;
12
13 architecture behavioral of counter8 is
14 signal counter1Hz49: std_logic_vector(22 downto 0);--50MHz/2**23=1.49Hz
15 signal direction: std_logic; --0: up, 1: down
16 signal enable: std_logic; --counter enable
17 signal counter: std_logic_vector(7 downto 0);
18 begin
19
20
21 process(CLOCK_50, RST)
22 begin
23  if RST='0' then
24     counter1Hz49<=(others=>'0');
25  elsif CLOCK_50'event and CLOCK_50='1' then
26     counter1Hz49<=counter1Hz49+1;
27 end if;
28 end process;
29
30 process(CLOCK_50, RST)
31 begin
32  if RST='0' then
33     counter<=(others =>'0');
34     direction<='0';
35  elsif CLOCK_50'event and CLOCK_50='1' then
36     enable<=SW(0);
37     direction<=SW(1);
38     if counter1Hz49=0 and enable='1' then
39        if direction='0' then
40           counter<=counter+1;
41        else
42           counter<=counter-1;
43        end if;
44     end if;
45 end if;
46 end process;
47
48
49 LEDR<=counter;
50 end behavioral;
```

Και τέλος δημιουργούμε το αρχείο timings.sdc, στον κατάλογο του project, στο οποίο ορίζουμε τη συχνότητα το ρολογιού όπως φαίνεται παρακάτω: +create_clock -period 20 [get_ports CLOCK_50]+++

Επιβεβαιώστε την ορθή λειτουργία της υλοποίησης.

Γ΄.6 Εργαστηριακή Άσκηση 5

Σε αυτήν την άσκηση θα γίνει εισαγωγή στον σχεδιασμό ενσωματωμένου συστήματος σε προγραμματιζόμενο chip χρησιμοποιώντας το εργαλείο Altera Qsys System Integration Tool και το Altera Program Monitor. Η διαδικασία του σχεδιασμού χωρίζεται σε δύο στάδια:

1. Hardware Development
2. Software Development

Hardware Development

Μέχρι στιγμής έχουμε δει τον κλασσικό τρόπο σχεδιασμού ψηφιακών κυκλωμάτων. Μπορούμε να χρησιμοποιήσουμε τις ίδιες μεθόδους για να κατασκευάσουμε ένα ενσωματωμένο σύστημα. Ωστόσο, όσο η πολυπλοκότητα των συστημάτων αυξάνεται, γίνεται δύσκολο και χρονοβόρο να κατασκευάσουμε οτιδήποτε από το μηδέν. Θα εξετάσουμε μία μέθοδο που χρησιμοποιεί προσχεδιασμένα modules από την Altera για να διευκολύνουμε τον σχεδιασμό και την διαδικασία υλοποίησης. Αυτά τα προσχεδιασμένα modules είναι γνωστά ως Intellectual Property (IP) Cores. Ένα ενσωματωμένο σύστημα, γενικά αποτελείται από τον επεξεργαστή, την μνήμη και τα υποστηριζόμενα περιφερειακά.

Στην καρδιά ενός ΕΣ υπάρχει ένας (ή μπορεί και περισσότεροι) επεξεργαστής ο οποίος εποπτεύει και διαχειρίζεται τις κύριες διεργασίες που απαιτούνται. Σε αυτό το εργαστήριο θα χρησιμοποιήσουμε τον Nios II της Altera. Ο επεξεργαστής Nios II αποτελεί ιδιοκτησία της Altera και είναι ειδικά σχεδιασμένος και υλοποιημένος για τις FPGA της Altera. Ο Nios II είναι ένας soft-processor (επεξεργαστής μαλακού πυρήνα) επειδή έχει δημιουργηθεί με χρήση VHDL ή Verilog. Ο επεξεργαστής μπορεί να παραμετροποιηθεί, ώστε να πληρεί συγκεκριμένες απαιτήσεις και ανάγκες.

Ένα παράδειγμα ενσωματωμένου συστήματος παρουσιάζεται στο Σχήμα Γ΄.13, όπου εμφανίζεται ο παραμετροποιήσιμος επεξεργαστής μαλακού πυρήνα (soft core) της Altera NIOS II. Κατά το σχεδιασμό ενός ΕΣ συστήματος με NIOS II, ο σχεδιαστής μπορεί να προσθέσει ή αφαιρέσει ή τροποποιήσει πλήθος δομοστοιχείων (modules), προκειμένου να καλύψει τις σχεδιαστικές του ανάγκες. Για παράδειγμα, αν μια σχεδιαστική ανάγκη είναι να έχει χαμηλή επιφάνεια, τότε μπορεί να απομακρύνει όλα τα μη απαραίτητα δομοστοιχεία, ενώ αν υπάρχει ανάγκη για πράξεις με πραγματικούς αριθμούς μπορεί να προσθέσει μια μονάδα FPU (floating point unit).

Το διάγραμμα μπλοκ του Nios II παρουσιάζεται στο Σχήμα Γ΄.14. Τα κύρια χαρακτηριστικά του είναι

- Αρχιτεκτονική load/store 32 bit
- Είσοδος/Εξοδος με απεικόνιση σε διευθύνσεις μνήμης (memory mapped)
- 32 επίπεδα διακοπτών
- 32 καταχωρητές γενικού σκοπού
- Τρεις εκδόσεις με διαφορετικές δυνατότητες
  - Nios-II /f: fast (γρήγορος) 6 επίπεδα διασωλήνωσης, βέλτιστη απόδοση.
  - Nios-II /s: standard (τυπικός) 5 επίπεδα διασωλήνωσης, σχετικά καλή απόδοση.
  - Nios-II /e: economy (οικονομικός) 1 στάδιο διασωλήνωσης, πολύ μικρός και ελεύθερος στη χρήση (δεν απαιτείται άδεια).

Θα δημιουργήσουμε ένα πλήρες ΕΣ χρησιμοποιώντας το εργαλείο της Altera με όνομα "Qsys system integration tool, QSYS" που βρίσκεται στο Quartus II. Σε αυτό το εργαστήριο, δημιουργούμε ένα βασικό ενσωματωμένο σύστημα με τον επεξεργαστή NIOS II/e, που αποτελείται από τον επεξεργαστή, μια μνήμη RAM 8KB που φέρει το πρόγραμμα, και μια σειριακή διεπαφή UART εισόδου εξόδου. Σε αυτό το εργαστήριο δεν χρησιμοποιούμε κάποιο λειτουργικό σύστημα, αλλά χρησιμοποιούμε προγραμματισμό "γυμνού΄΄ υλικού (bare metal programming), δηλαδή δημιουργείται ο κώδικας μηχανής από το πρόγραμμά μας σε C, και ο επεξεργαστής εκτελεί άμεσα αυτόν τον κώδικα.

Τα σημαντικά βήματα αυτού του εργαστηρίου είναι:

- Δημιουργία project στο Quartus.
- Δημιουργία ενός NIOS II ενσωματωμένου συστήματος από το Qsys.
- Δημιουργία ενός αρχείου VHDL ως σχεδιαστική κορυφή, του BSP και του προγράμματος.
- Δημιουργία του συνολικού κώδικα προγραμματισμού της πλακέτας και προγραμματισμός της πλακέτας.

Δημιουργία project στο Quartus

Το πρώτο βήμα είναι να δημιουργήσουμε ένα νέο project που αφορά την FPGA στην πλακέτα DE2-115. Δημιουργούμε ένα project για το FPGA chip που φέρει η πλακέτα μας (π.χ. EP4CE115F29C7). Στο project δεν προσθέτουμε κάποια αρχεία προς το παρόν. Θα δημιουργήσουμε την HDL περιγραφή από το

πρόγραμμα QSYS, και το αρχείο της μνήμης από το SDK. Το επόμενο βήμα είναι η δημιουργία της HDL περιγραφής, το οποίο είναι απαραίτητο ώστε ο compiler να γνωρίζει τις δυνατότητες του ενσωματωμένου συστήματος. Για παράδειγμα, αν στο ενσωματωμένο σύστημα τοποθετήσουμε κάποια μονάδα FPU, τότε ο compiler θα γνωρίζει πως μπορεί να χρησιμοποιήσει εντολές FPU για πράξεις με πραγματικούς αριθμούς, και δε θα καταφύγει στην προσομοίωση με λογισμικό αυτών των δυνατοτήτων, μέσω ακέραιων πράξεων. Για αυτό, αμέσως μετά τη δημιουργία του υλικού, δημιουργείται το Board Support Package, που περιγράφει τις δυνατότητες του υλικού.

Γ΄.6.1 Δημιουργία ενός NIOS II ενσωματωμένου συστήματος από το Qsys

Από το μενού Tools του Quartus, επιλέγουμε το εργαλείο QSYS, που αν και δεν το αναφέρει η Altera, σημαίνει Quick System Integration, δηλαδή παρέχει έναν εύκολο τρόπο να δημιουργήσει κάποιος ένα ενσωματωμένο σύστημα με προγραμματιζόμενο επεξεργαστή. Σε αυτό το περιβάλλον, ο χρήστης επιλέγει components (κάποια είναι ελεύθερα και δωρεάν, ενώ κάποια απαιτούν ειδική άδεια) από το παράθυρο στα αριστερά με τίτλο “Component Library” και τα τοποθετεί στο παράθυρο στα δεξιά, που καταλαμβάνει το μεγαλύτερο μέρος και φέρει καρτέλες με ονόματα “System Contents”, “Address Map” κτλ. (Σχήμα Γ΄.15). Το πρώτο στοιχείο που θα προσθέσουμε είναι η onchip RAM μνήμη. Πληκτρολογήστε “On-chip memory” στο εύρεση κάτω από το component library, επιλέξτε το δομοστοιχείο RAM or ROM, και μετά το Add ή διπλό κλικ. Στο παράθυρο ρυθμίσεων της μνήμης που θα ανοίξει αυτόματα, επιλέξτε RAM (Writable) και μέγεθος 8192. Με αυτόν τον τρόπο θα χρησιμοποιηθεί η onchip μνήμη του FPGA chip (η Altera DE2-115, έχει και 2 chip των 64 Mbytes). Στις επιλογές του Memory Initialization δε θα αλλάξουμε κάτι, επειδή δεν είναι δυνατό να δημιουργηθεί ο κώδικας μηχανής (σε μορφή hex initialization file), αφού δεν υπάρχει το board support package. Με την επιλογή Finish προστίθεται το component στο “System Contents”. Ο χρήστης μπορεί με διπλό κλικ σε ένα component να τροποποιήσει κάποια παράμετρο. Κατά την προσθήκη components, εμφανίζονται κάποια μηνύματα σφάλματος (Errors) στο κάτω παράθυρο του εργαλείου, που όμως δεν μας απασχολούν για την ώρα, αφού δεν έχει ολοκληρωθεί το σύστημα. Ως αυτό το σημείο, υπάρχουν δυο components: clk_0 και onchip_memory2_0. Τα components μπορούν με δεξί κλικ να μετονομαστούν, αλλά δεν είναι απαραίτητο.

Το επόμενο βήμα είναι η προσθήκη του επεξεργαστή. Στο πεδίο εύρεσης components, γράψτε το nios και από τη λίστα των αποτελεσμάτων επιλέξτε το “Nios II Processor”. Ο NIOS II μπορεί να ρυθμιστεί με λίγα, ενδιάμεσα ή πολλά

χαρακτηριστικά, με τις επιλογές Nios II/e, Nios II/s ή NIOS II/f αντίστοιχα. Η μόνη ελεύθερη έκδοση που δεν απαιτεί άδεια, είναι η Nios II/e, και πρέπει αυτή να επιλεγεί. Με το Finish, προστίθεται το component στην λίστα, οπότε το System Contents φέρει 3 στοιχεία: clk_0, onchip_memory2_0, nios2_qsys_0.

Το τελευταίο στοιχείο του κυκλώματος μας είναι η σειριακή διεπαφή UART. Γράψτε UART στο component library search, και επιλέξτε το UART (RS-232 Serial Port). Οι παράμετροι προεπιλογής είναι N/8/1/2 στα 115200 fixed, οπότε δεν τροποποιούνται (αν αλλάξετε κάποια από αυτές τις παραμέτρους θα πρέπει να τοποθετήσετε τις ίδιες και στο πρόγραμμα τερματικού σειριακής σύνδεσης στο τέλος της άσκησης). Επειδή θα πρέπει να συνδέσουμε αυτό το περιφερειακό σε φυσική διεπαφή, θα πρέπει να γίνει εξαγωγή ως θύρα. Επιλέξτε το external_connection του uart_0 και πατήστε στη στήλη Export (double-click to export) και θα δείτε ότι εξάγεται ως θύρα με όνομα "uart_0_external_connection". Επίσης, θα πρέπει να συνδεθεί το interrupt request (IRQ) στον επεξεργαστή. Επιλέξτε τον επεξεργαστή, δεξί κλικ connections→ nios2_qsys_0.d_irq→uart_0.irq.

Σε αυτό το σημείο υπάρχουν 8 errors, επειδή δεν έχουν ολοκληρωθεί οι διασυνδέσεις. Πατήστε στη στήλη των connections σε κάθε component τη διασύνδεση με το ρολόι (clk). Με τις 3 συνδέσεις, τα λάθη έχουν μειωθεί από 8 σε 5. Ομοίως, συνδέστε το clk_reset με το αντίστοιχο reset. Με τις νέες 3 συνδέσεις τα λάθη έχουν μειωθεί σε 2. Το data_master και το instruction_master του επεξεργαστή πρέπει να συνδεθεί στη θύρα s1 της μνήμης, όπως και το s1 της uart_0 με το data_master του επεξεργαστή, και με το s1 της μνήμης. Πατήστε διπλό κλικ στον επεξεργαστή, και στη συνέχεια στο Reset Vector όπως και στο Exception Vector, επιλέξτε στο πεδίο vector memory onchip_memory2_0.s1. Πατήστε finish. Επίσης, ο επεξεργαστής μας έχει το JTAG που θα πρέπει να συνδεθεί σε κάθε περιφερειακό για την εύκολη αποσφαλμάτωση. Οπότε, συνδέστε το jtag_debug_module_reset του επεξεργαστή στις θύρες reset των στοιχείων uart_0, nios2_qsys_0 και onchip_memory2_0. Το ίδιο μπορεί να επιτευχθεί και από το μενού System→Create Global Reset Network. Επίσης, για λόγους αποσφαλμάτωσης θα προσθέσουμε και το περιφερειακό JTAG UART (ώστε να έχουμε εύκολη πρόσβαση εισόδου/εξόδου για την αποσφαλμάτωση του λογισμικού). Στο περιφερειακό αυτό, συνδέετε το clk και το reset, ενώ το avalon_jtag_slave συνδέεται στο δίαυλο δεδομένων (data) που καταλήγει στη θύρα s1 της μνήμης εντός ολοκληρωμένου κυκλώματος. Επίσης, στη στήλη IRQ θα πρέπει να γίνει η σύνδεση με το κανάλι IRQ του επεξεργαστή, οπότε η κατανομή των IRQ είναι η εξής: το περιφερειακό UART λαμβάνει το IRQ0, ενώ το JTAG UART λαμβάνει το IRQ1. Σε αυτό το σημείο έχουμε 4 σφάλματα, τα οποία οφείλονται σε συγκρούσεις διευθύνσεων. Κάθε στοιχείο του συστήματος θα πρέπει να βρίσκεται σε μια ξεχωριστή και μοναδική διεύθυνση. Με την επιλογή System→assign base addresses, αυτοματοποιείται η ανάθεση των πόρων.

Τα σφάλματα και οι προειδοποιήσεις έχουν μειωθεί στο 0 (Σχήμα Γ΄.16).

Αποθηκεύστε το project στο QSYS στον κατάλογο του κεντρικού project στο Quartus και στη συνέχεια αποεπιλέξτε το Generate από την καρτέλα Generation. Μετά από λίγη ώρα θα δημιουργηθεί ένα αρχείο qip στον υποκατάλογο synthesis που βρίσκεται ως υποκατάλογος σε ένα κατάλογο που φέρει το όνομα του QSYS project. Μπορείτε να κλείσετε το εργαλείο QSYS.

Το λογισμικό αλληλεπιδρά με τις συσκευές I/O μέσω των memory - mapped καταχωρητών. Σε αυτό το σχήμα, οι καταχωρητές ελέγχου και Ε/Ε μιας συσκευής απεικονίζονται στον χώρο διευθύνσεων της κύριας μνήμης. Τα Software Programs μπορούν να γραφούν σε ASSEMBLY ή σε C. Θα χρησιμοποιήσουμε τη γλώσσα C σε αυτήν την εργασία. Για να επικοινωνήσουν περιφερειακά και συσκευές στα συστήματα, θα προσπελάσουμε τους καταχωρητές τους. Από τη στιγμή που αυτοί οι καταχωρητές απεικονίζονται στον χώρο διευθύνσεων της μνήμης του συστήματος, θα γίνει εκτενής χρήση των C pointers.

Με την μέθοδο πρόσβασης χαμηλού επιπέδου, οι διευθύνσεις μνήμης που έχουν αντιστοιχηθεί στους καταχωρητές των συσκευών Ε/Ε στο Qsys Tool μπορούν να θεωρηθούν ως τιμές των pointer. Έτσι εμείς μπορούμε να χρησιμοποιήσουμε τις μεταβλητές-pointers για να χειριστούμε τους καταχωρητές. Για παράδειγμα το Σχήμα Γ΄.16 φαίνεται η βασική διεύθυνση της σειριακής θύρας είναι η 0x5000.

Οι δηλώσεις:

```
volatile char* uart=0x0005000;
volatile char* switches=0x0003000;
volatile char* ledg=0x0003010;
```

ορίζουν τρεις pointers που δείχνουν σε περιοχές τύπου δεδομένων char (8 bits). Η λέξη volatile γνωστοποιεί στον μεταγλωττιστή ότι η τιμή του αντικειμένου μπορεί να τροποποιηθεί χωρίς την επέμβαση του επεξεργαστή και για αυτό δεν πρέπει να γίνουν βελτιστοποιήσεις. Εναλλακτικά, το QUARTUS δίνει τη δυνατότητα να χρησιμοποιηθούν pre-defines για τις διευθύνσεις, ώστε αν κάποια στιγμή τροποποιηθούν οι διευθύνσεις βάσης κάποιου περιφερειακού, να μη χρειάζεται να τροποποιηθεί ο κώδικας C, απλώς να γίνει ένα νέο compile. Για παράδειγμα, η διεύθυνση βάσης της σειριακής θύρας UART_0 μπορεί να χρησιμοποιηθεί ως + volatile uint32_t *uart = (volatile uint32_t*) UART_0_BASE;+++ Έτσι, αν αλλαχθεί κάποια στιγμή η διεύθυνση της σειριακής θύρας, αρκεί μια νέα μεταγλώττιση του πηγαίου προγράμματος σε C και θα λειτουργεί απρόσκοπτα. Οι διευθύνσεις αυτές δημιουργούνται κατά την παραγωγή του board support package, με το BSP Generate, όπως θα δούμε σε επόμενη παράγραφο, και τοποθετούνται στο αρχείο system.h που βρίσκεται μέσα στον υποκατάλογο του bsp, που βρίσκεται μέσα στον κατάλογο software του Project μας. Μπορείτε να ανοίξετε το system.h με έναν επεξεργαστή κειμένου και να δείτε όλα τα

defines. Για να χρησιμοποιήσετε αυτά τα defines, θα πρέπει ασφαλώς να έχετε μια γραμμή + #include ”system.h” +++ στο αρχείο σας.

Βασική Αρχιτεκτονική Προγράμματος σε C Η απλούστερη αρχιτεκτονική ελέγχου, όπως έχουμε δει στη θεωρία, είναι ένας ατέρμων βρόχος στον οποίο οι διεργασίες εκτελούνται ακολουθιακά. Ο κώδικας ενός βασικού προγράμματος σε C είναι:

```
main() {
//initialization
while (1) //infinite loop {
task_1();
task_2();
.
.
.
task_n();
}
}
```

Το σύστημα αρχικοποιείται μία φορά στην αρχή του προγράμματος. Έπειτα, η κάθε διεργασία εκτελείται με την σειρά. Σε αυτό το παράδειγμα το πρόγραμμα δεν τερματίζεται ποτέ.

Γ΄.6.2 Δημιουργία ενός αρχείου VHDL ως σχεδιαστική κορυφή, του BSP και του προγράμματος

Με την επιλογή generate δημιουργείται ένα αρχείο qip (q intellectual property) στον υποκατάλογο synthesis, που βρίσκεται στον υποκατάλογο που φέρει το όνομα του QSYS project. Στο Quartus Project Navigator, περιοχή Files, προσθέστε το qip αρχείο που έχει δημιουργηθεί. Επίσης, πατήστε δεξί κλικ και Set as Top Level Entity, για να ρυθμίστε τη σχεδιαστική κορυφή.

Το επόμενο βήμα είναι να δημιουργηθεί το αρχείο περιγραφής του υλικού (board support package), ώστε να γνωρίζει ο συμβολομεταφραστής τις δυνατότητες του συστήματος. Αυτό επιτυγχάνεται από το μενού Tools→Nios II Software Build Tools for Eclipse. Την πρώτη φορά που θα εκτελεστεί το εργαλείο θα σας ζητηθεί η επιλογή του workspace. Μπορείτε να το αφήσετε ως προεπιλογή, αφού δε θα χρησιμοποιηθεί.

Στο πρόγραμμα του Eclipse (Εικόνα Γ΄.17) δημιουργείται η BSP περιγραφή από το μενού File→New→Nios II Application and BSP from Template και τοποθετήστε το SOPC file και ένα όνομα για το project. Πατήστε finish (και όχι Next)

και θα δημιουργηθούν όλα τα κατάλληλα αρχεία, τόσο για την εφαρμογή όσο και το BSP (Εικόνα Γ΄.18).

Για να ρυθμίσουμε καλύτερα τις προδιαγραφές της αρχιτεκτονικής, θα επιλέξουμε το BSP με δεξί κλικ, και από το αναδυόμενο μενού θα ακολουθήσουμε το NIOS II→BSP Editor. Στο νέο παράθυρο θα απλοποιήσουμε το BSP (επομένως και τον κώδικα που θα καταλαμβάνει το τελικό αρχείο αρχικοποίησης της μνήμης) με το να επιλέξουμε το settings και να επιβεβαιώσουμε ότι υπάρχουν tick επιλογής μόνο στις παρακάτω ρυθμίσεις:

- enable_lighweight_device_driver_api
- enable_reduced_device_drivers
- enable_small_c_library
- allow_code_at_reset

Τέλος, θα πρέπει το stderr, stdin, stdout να έχουν την επιλογή jtag_uart_0 (ή αν έχετε σειριακό καλώδιο null modem συνδεδεμένο στον υπολογιστή, την επιλογή uart_0), ώστε να εμφανίζονται τα μηνύματα σε μια συσκευή εξόδου. Πατήστε το GENERATE και μετά το EXIT.

Όπως μπορείτε να καταλάβετε, η εφαρμογή αυτή εκτυπώνει το μήνυμα Hello from NIOS II στη σειριακή θύρα, κάτι που φαίνεται στο αρχείο hello_world.c. Πατήστε Project→Build ALL για να δημιουργηθεί και το BSP και το πρόγραμμα Hello World. Επίσης, απαιτείται η δημιουργία του αρχείου αρχικοποίησης της μνήμης. Πατήστε δεξί κλικ στο όνομα του subproject για το hello world (όχι το subproject για το BSP), και Make Target→Build→mem_init_generate και μετά build. Αυτό θα δημιουργήσει ένα αρχείο με κατάληξη hex μέσα στον υποκατάλογο του project mem_init, το οποίο θα πρέπει να ενσωματωθεί στον κώδικα προγραμματισμού της πλακέτας.

Γ΄.6.3 Δημιουργία του συνολικού κώδικα προγραμματισμού της πλακέτας και προγραμματισμός της πλακέτας

Μεταβείτε στο Quartus, και προσθέστε στα αρχεία, το αρχείο αρχικοποίησης της μνήμης από τον κατάλογο software, υποκατάλογος mem_init μέσα στον φάκελο του project λογισμικού (όχι του BSP). To File of type, πρέπει να έχει την επιλογή Memory Files (*.mif, *.hex).

Το επόμενο βήμα, είναι η τοποθέτηση των περιορισμών. Όπως και στα προηγούμενα εργαστήρια, δημιουργήστε ένα αρχείο sdc (Synopsis Design Constraints File) μέσα στο φάκελο του project, που να ρυθμίζει ένα ρολόι των 40Mhz με τις γραμμές:

```
create_clock -name {clk} -period 40.000 -waveform { 0.000 20.000 } [get_ports {clk_clk}]
derive_clock_uncertainty
```

Μόλις αποθηκευτεί το αρχείο .sdc στον κατάλογο του project, θα προστεθεί στη λίστα των αρχείων του project. Σε αυτό το σημείο η λίστα των αρχείων θα αποτελείται από τρεις καταχωρήσεις: Το αρχείο του επεξεργαστή, το αρχείο αρχικοποίησης μνήμης και το αρχείο των σχεδιαστικών περιορισμών. Το τελευταίο βήμα είναι η σύνδεση των ακροδεκτών.

Για να βρεθούν οι ακροδέκτες που απαιτούνται να συνδεθούν, πρέπει το Quartus να ολοκληρώσει τη διαδικασία του elaboration. Αυτό επιτυγχάνεται με Processing→Start→Start Analysis and elaboration.

Στη συνέχεια, από το μενού Assignments→Pin Planner θα εμφανιστεί μια οθόνη με τους ακροδέκτες. Από το αρχείο csv βρίσκουμε τις γραμμές που μας αφορούν, δηλαδή τις παρακάτω γραμμές:

```
CLOCK_50,Input,PIN_Y2,2,B2_N0,3.3-V LVTTL,
KEY[0],Input,PIN_M23,6,B6_N2,2.5 V,
UART_RXD,Input,PIN_G12,8,B8_N1,3.3-V LVTTL,
UART_TXD,Output,PIN_G9,8,B8_N2,3.3-V LVTTL,
```

Από το εργαλείο pin planner, γίνεται η αντιστοίχηση των λογικών ακροδεκτών του ενσωματωμένου συστήματος με τους φυσικούς ακροδέκτες της πλακέτας (Εικόνα Γ΄.19). Μόλις ολοκληρωθεί η αντιστοίχηση, από το Processing→Start Compilation, δημιουργείται το αρχείο ρυθμίσεων της επαναδιαμορφώσιμης αρχιτεκτονικής (bitstream), το οποίο στη συνέχεια θα μεταφερθεί στην πλακέτα από το Program Device (Open Programmer).

Να σημειωθεί ότι αν γίνει κάποια αλλαγή στο QSYS, π.χ. προστεθεί κάποιο component, και γίνει πάλι generate, τότε θα πρέπει στο eclipse να ξαναδημιουργηθεί το BSP. Αυτό επιτυγχάνεται με το να επιλεχθεί το BSP subproject, και στη συνέχεια δεξί κλικ, επιλογή NIOS II→Generate BSP. Στη συνέχεια, δεξί κλικ στο project και επιλογή Build this project. Ασφαλώς μετά θα πρέπει να γίνει build και το λογισμικό μας (build this project, αφού επιλεχθεί το project του λογισμικού), όπως και το αρχείο αρχικοποίησης της μνήμης.

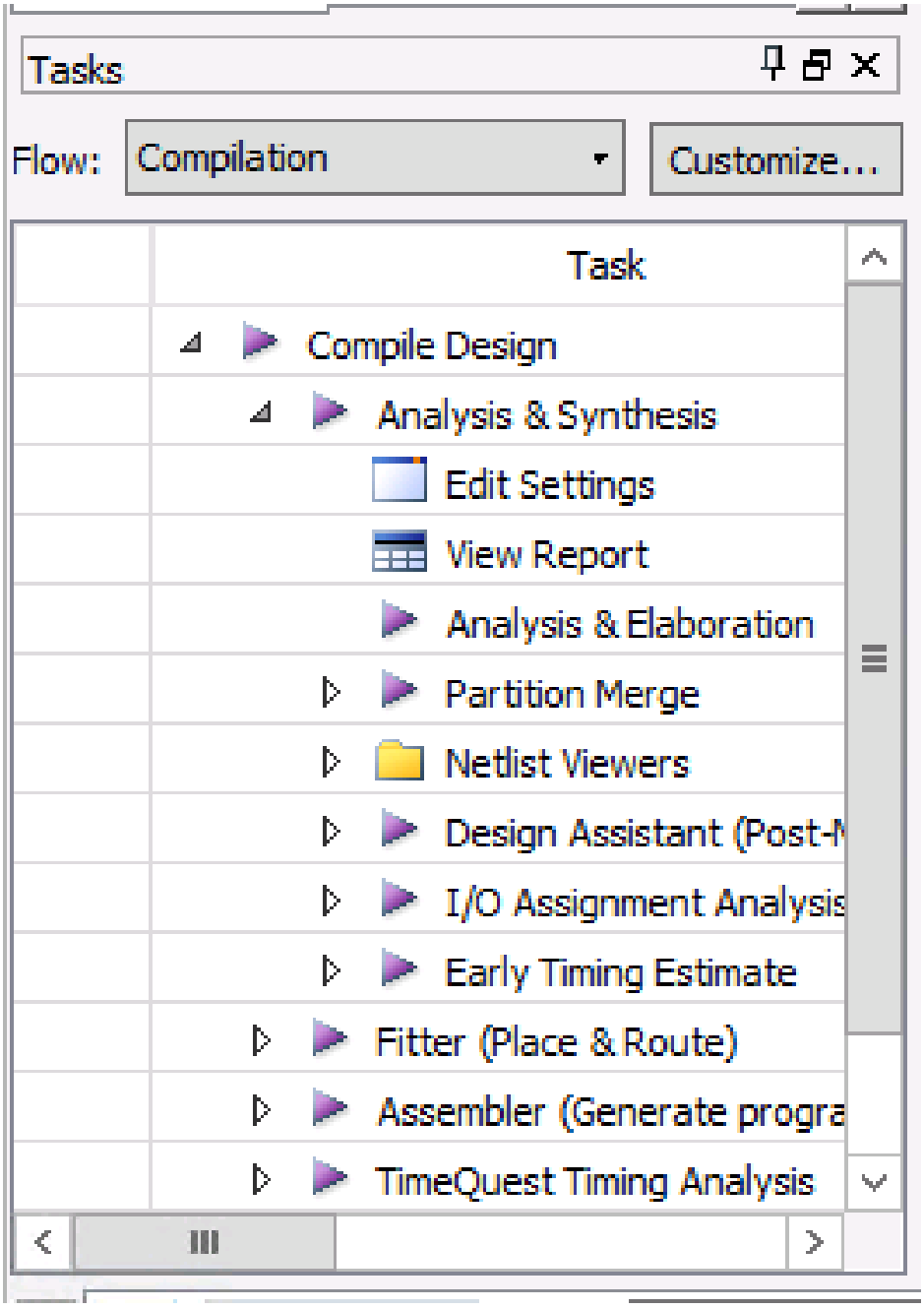
Επίσης, από τη στιγμή που θα δημιουργηθεί το hardware με υποστήριξη JTAG μπορείτε μέσα από το περιβάλλον του Eclipse να μεταφορτώνετε τον κώδικα άμεσα στο FPGA, χωρίς να χρειάζεται να δημιουργήσετε ένα νέο bitstream. Αρκεί να επιλέξετε από το Eclipse το hardware και μετά Run as ή Debug as και επιλογή NIOS II Hardware. Βέβαια για να μπορεί να γίνει αυτό θα πρέπει να έχετε επιλέξει στο Run as →Run Configuration το Refresh Connection για να βρεθεί το hardware και μετά να έχετε ενεργοποιημένες τις επιλογές “Ignored mismatched System ID” και “Ignored mismatched System timestamp”, γιατί απαιτούν δυνατότητες από το hardware που δεν έχουν συμπεριληφθεί.

Εκτός από το πρόγραμμα hello world, μπορείτε να χρησιμοποιήσετε και το πρόγραμμα Γ΄.6.3 στο οποίο εκτυπώνονται ταυτόχρονα και στην τυπική έξοδο του JTAG αλλά και στο παράθυρο τερματικού (Εικόνα Γ΄.20) Σε περίπτωση που χρησιμοποιήσετε σειριακό καλώδιο, θα πρέπει να γνωρίζετε ότι η θύρα της Altera DE2-115 είναι μία θύρα σειριακή τύπου DTE (data terminal equipment) σε FEMALE υποδοχή (συνήθως οι DTE χρησιμοποιούν MALE υποδοχές, οπότε δημιουργείται κάποια σύγχυση σε αυτό το σημείο), ίδιου τύπου με τη σειριακή θύρα του PC. Για να συνδεθεί μια θύρα DTE με DTE, θα πρέπει να χρησιμοποιηθεί ένα cross-over serial cable ή αλλιώς null modem cable. Αν χρησιμοποιηθεί ένα απλό καλώδιο serial Male-Female δε θα εμφανιστεί τίποτα στην οθόνη, γιατί αυτό το καλώδιο χρησιμοποιείται για σύνδεση σειριακής θύρας τύπου DTE με τύπου DCE (data computing equipment). Μπορείτε επίσης να προσθέσετε και τους ακροδέκτες CTS/RTS στο UART (από το QSYS) για να έχετε καλύτερο έλεγχο της ροής των δεδομένων (για να μη γίνει υπερχείλιση κάποιου buffer, αν μεταφέρονται πολλά δεδομένα), οπότε θα πρέπει να κάνετε και τη αντίστοιχη συσχέτιση των 2 επιπρόσθετων ακροδεκτών στο pin planner.

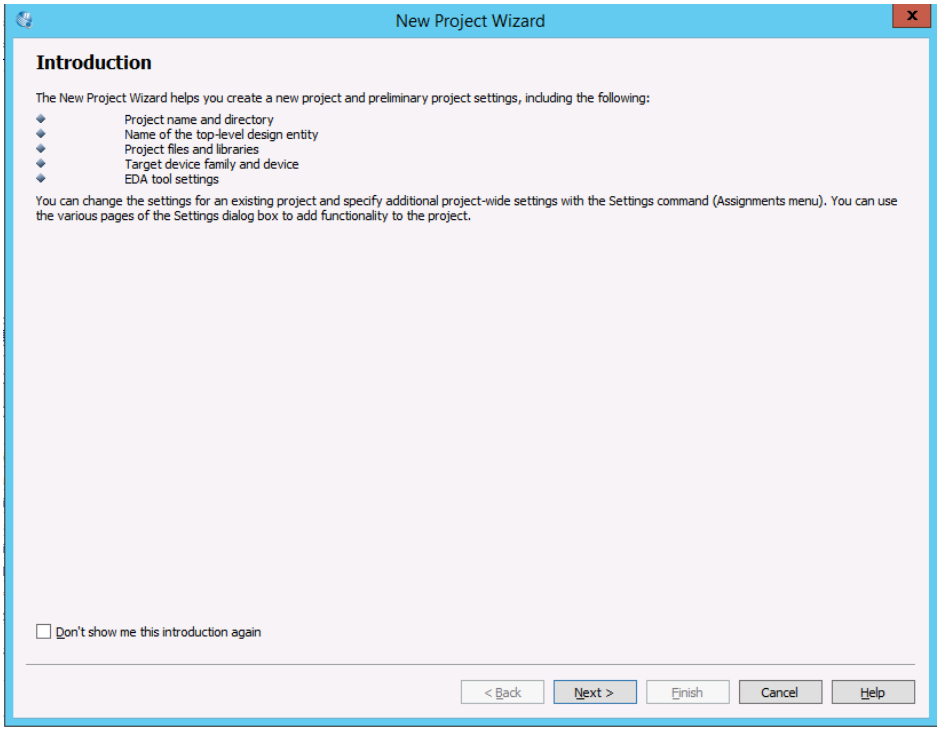
```
#include <stdio.h>
#include <stdint.h>
#include "system.h"

int main()
{
   int delay;
   char *str = "Hello Serial from NIOS II\n";

   volatile uint32_t *uart = (volatile uint32_t*) UART_0_BASE;

      printf("Start Printing to Serial Port");
      while (1)
      {
         printf("New Iteration\n");

        char *ptr = str;
         while (*ptr != '\0')
         {
            while ((uart[2] & (1<<6)) == 0);
            uart[1] = *ptr;
            ptr++;
         }

         //add a little delay
         delay=0;
         while (delay<2000000)
         {
            delay++;
         }
      }
      return 0;
}
}
```

Γ΄.7 Εργαστηριακή Άσκηση 5

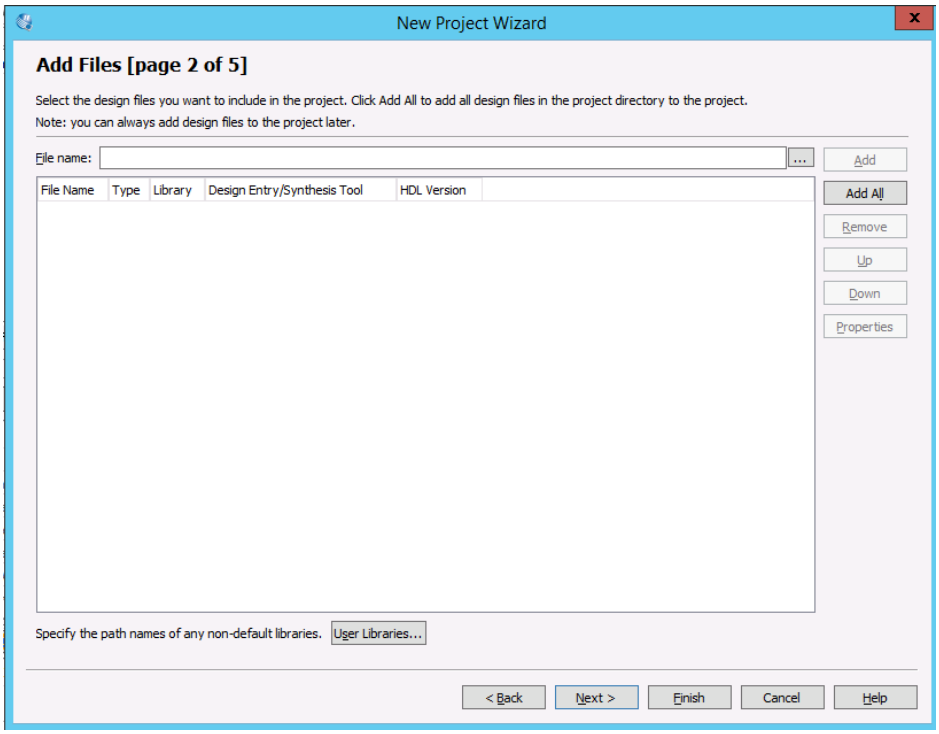
Βασιζόμενοι στις δεξιότητες που αποκομίσατε από τις προηγούμενες εργαστηριακές ασκήσεις, καλείστε να υλοποιήσετε ένα ενσωματωμένο σύστημα που βασίζεται στο NIOS επεξεργαστή, και χρησιμοποιεί εκτός από τα προηγούμενα στοιχεία και το στοιχείο PIO (Parallel I/O). Θα προσθέσετε αυτό το στοιχείο 2 φορές: μία για είσοδο και μία για έξοδο ενώ θα το ρυθμίσετε στα 8 bit. Η 8 bit είσοδος θα συνδεθεί σε 8 διακόπτες SW[0] έως SW[7], ενώ η 8 bit έξοδος θα συνδεθεί σε 8 led: LEDR[0] έως LEDR[1]. Η μνήμη RAM του συστήματος θα ρυθμιστεί σε 32KB.



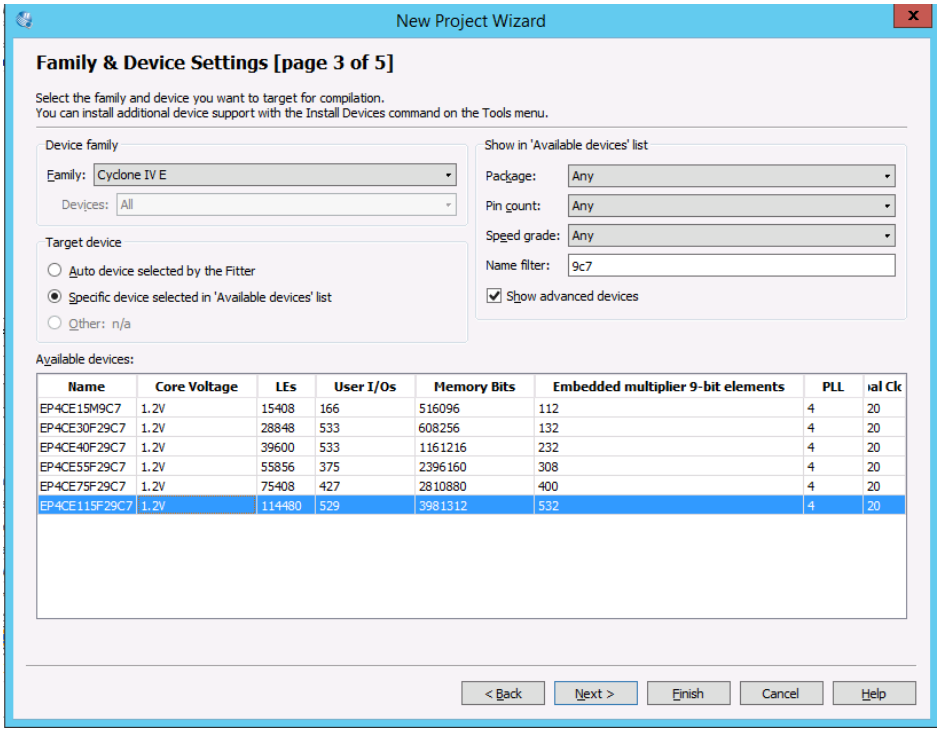
Σχήμα Γ΄.3: Στην περιοχή Tasks του εργαλείου Altera Quartus, ο χρήστης μπορεί να επιλέξει να εκτελέσει ξεχωριστά το εργαλείο που χρειάζεται.



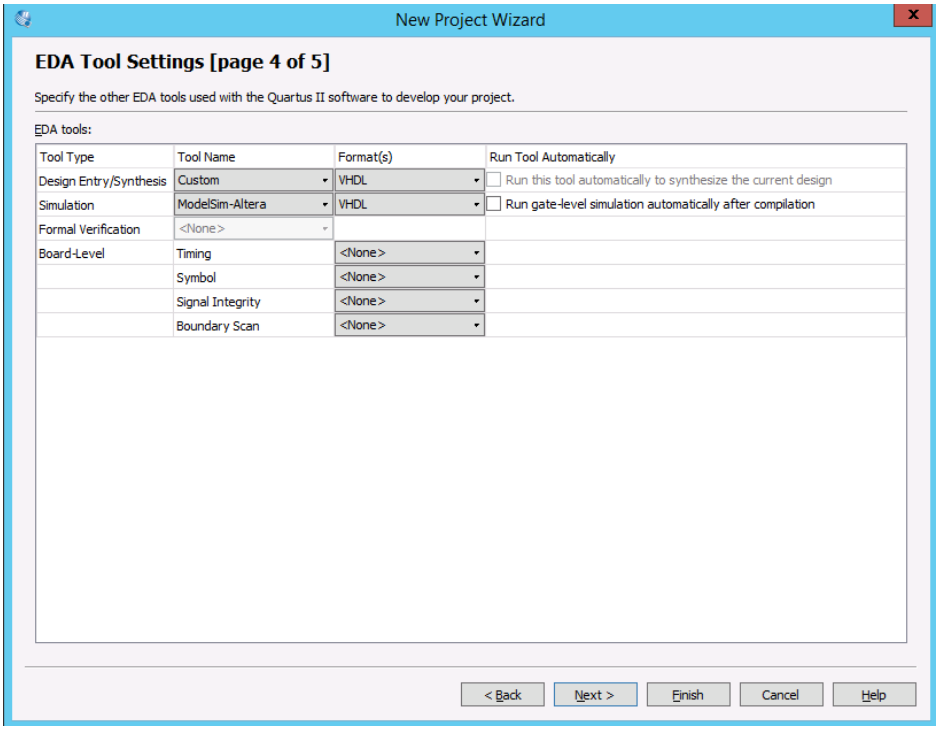
Σχήμα Γ΄.4: Μόλις ξεκινήσει το New Project Wizard στο Altera Quartus, ο χρήστης ενημερώνεται για τις πληροφορίες που θα εισάγει.



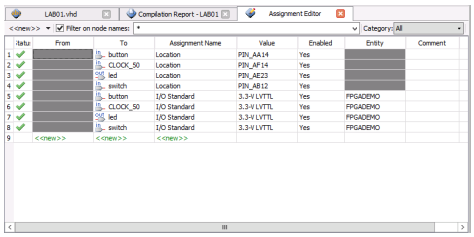
Σχήμα Γ΄.5: Στη 2η οθόνη του New Project Wizard, ο χρήστης μπορεί να προσθέσει αρχεία στο project.



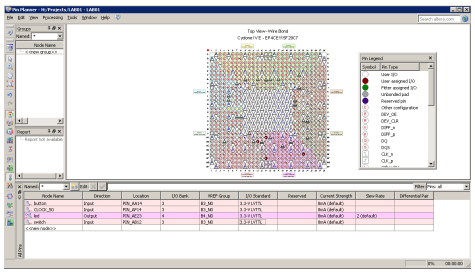
Σχήμα Γ΄.6: Στην 3η οθόνη του New Project Wizard, ο χρήστης επιλέγει το FPGA chip.



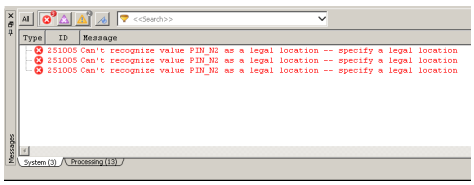
Σχήμα Γ΄.7: Στην 4η οθόνη του New Project Wizard, ο χρήστης ρυθμίζει τα εργαλεία του project.

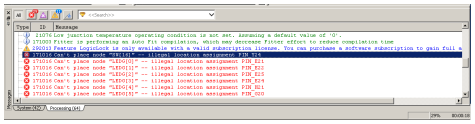


Σχήμα Γ΄.8: Ο επεξεργαστής συνδέσεων ακροδεκτών.

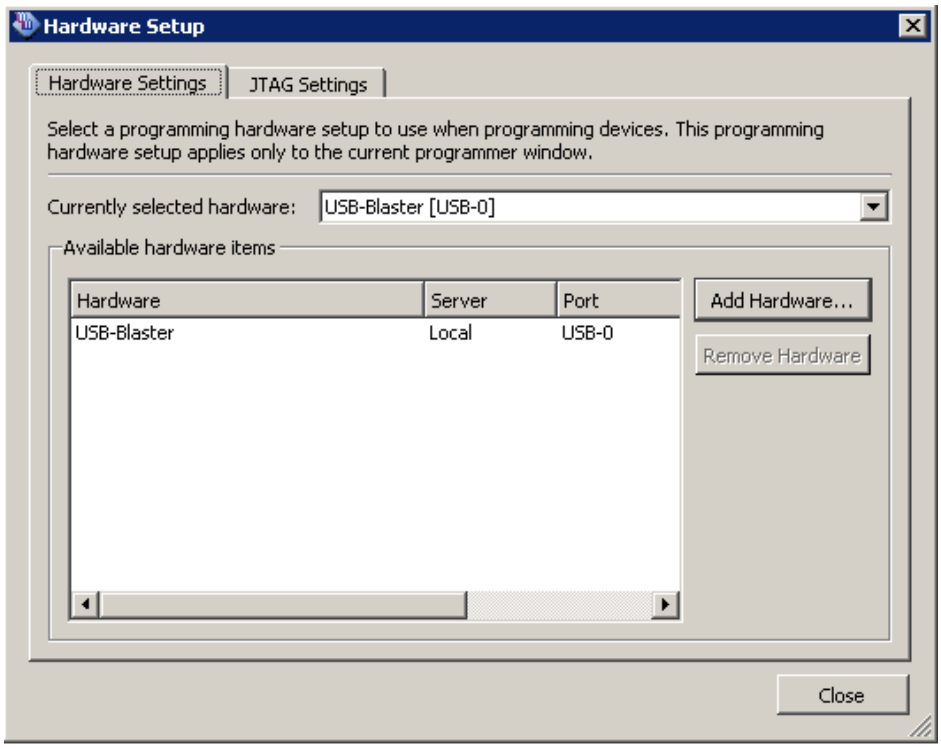


Σχήμα Γ΄.9: Η οπτική σύνδεση ακροδεκτών.

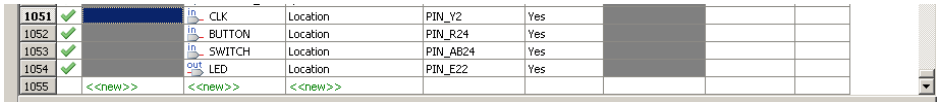




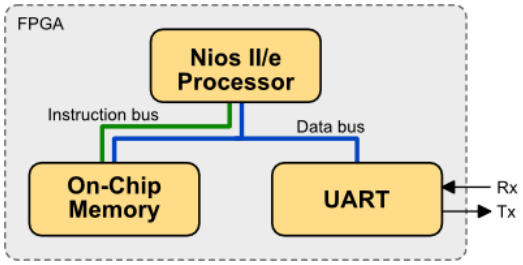
Σχήμα Γ΄.10: Απαιτείται προσοχή κατά τη διαδικασία της σύνδεσης των ακροδεκτών του FPGA, ώστε να μη δημιουργηθούν σφάλματα.



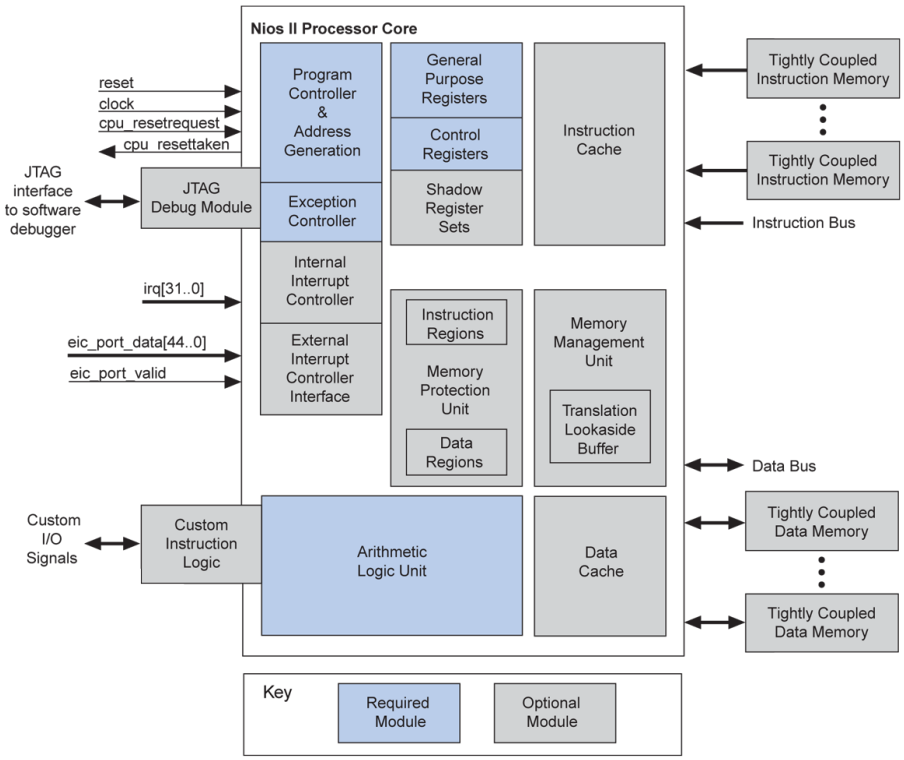
Σχήμα Γ΄.11: Στη διαδικασία του προγραμματισμού του FPGA, θα πρέπει να έχει επιλεχθεί η συσκευή USB BLASTER.



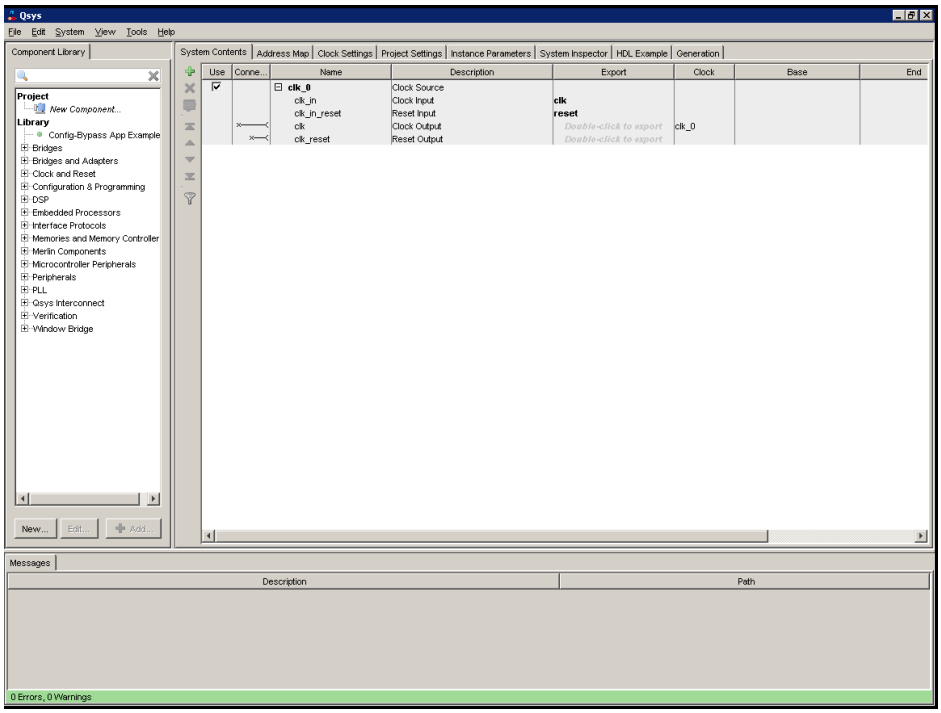
Σχήμα Γ΄.12: Τοποθέτηση των συσχετίσεων των ακροδεκτών του FPGA με τις θύρες εισόδου εξόδου του Top Level Design, της πρώτης άσκησης.



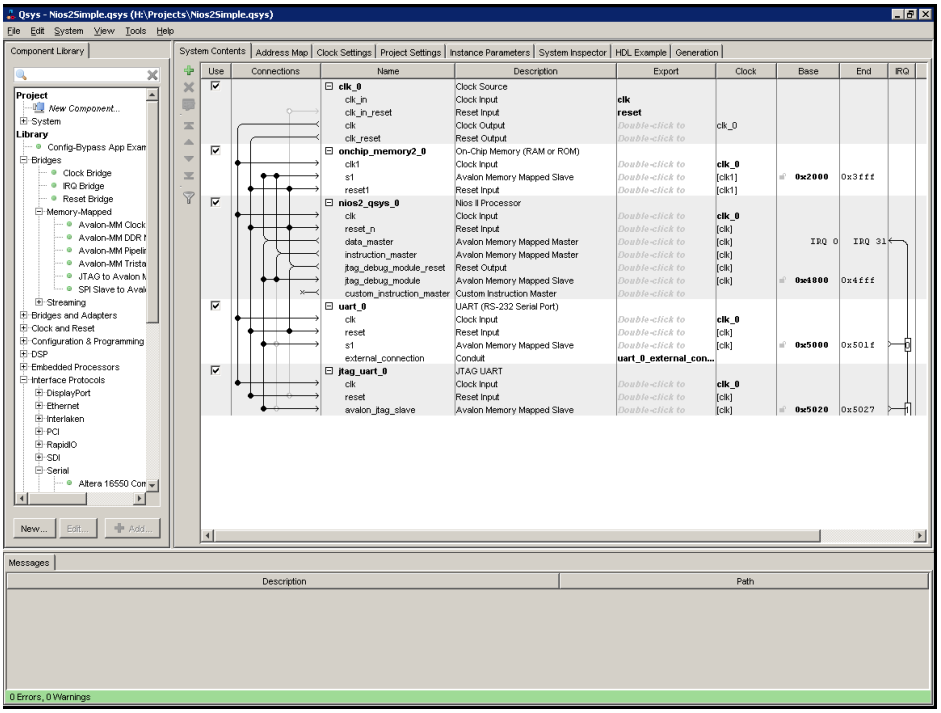
Σχήμα Γ΄.13: Ένα απλό ενσωματωμένο σύστημα με τον επεξεργαστή NIOS II, τη μνήμη, και τη σειριακή διασύνδεση εισόδου εξόδου.



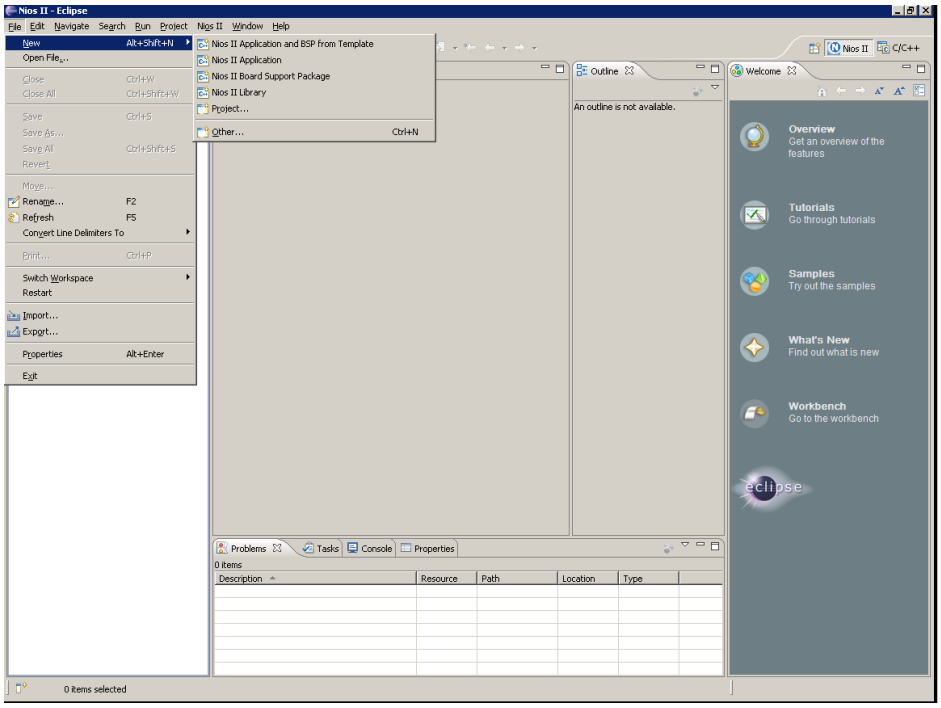
Σχήμα Γ΄.14: Ο μαλακός επεξεργαστής NIOS II της Altera είναι πλήρως παραμετροποιήσιμος, αφού είναι αρθρωτός.



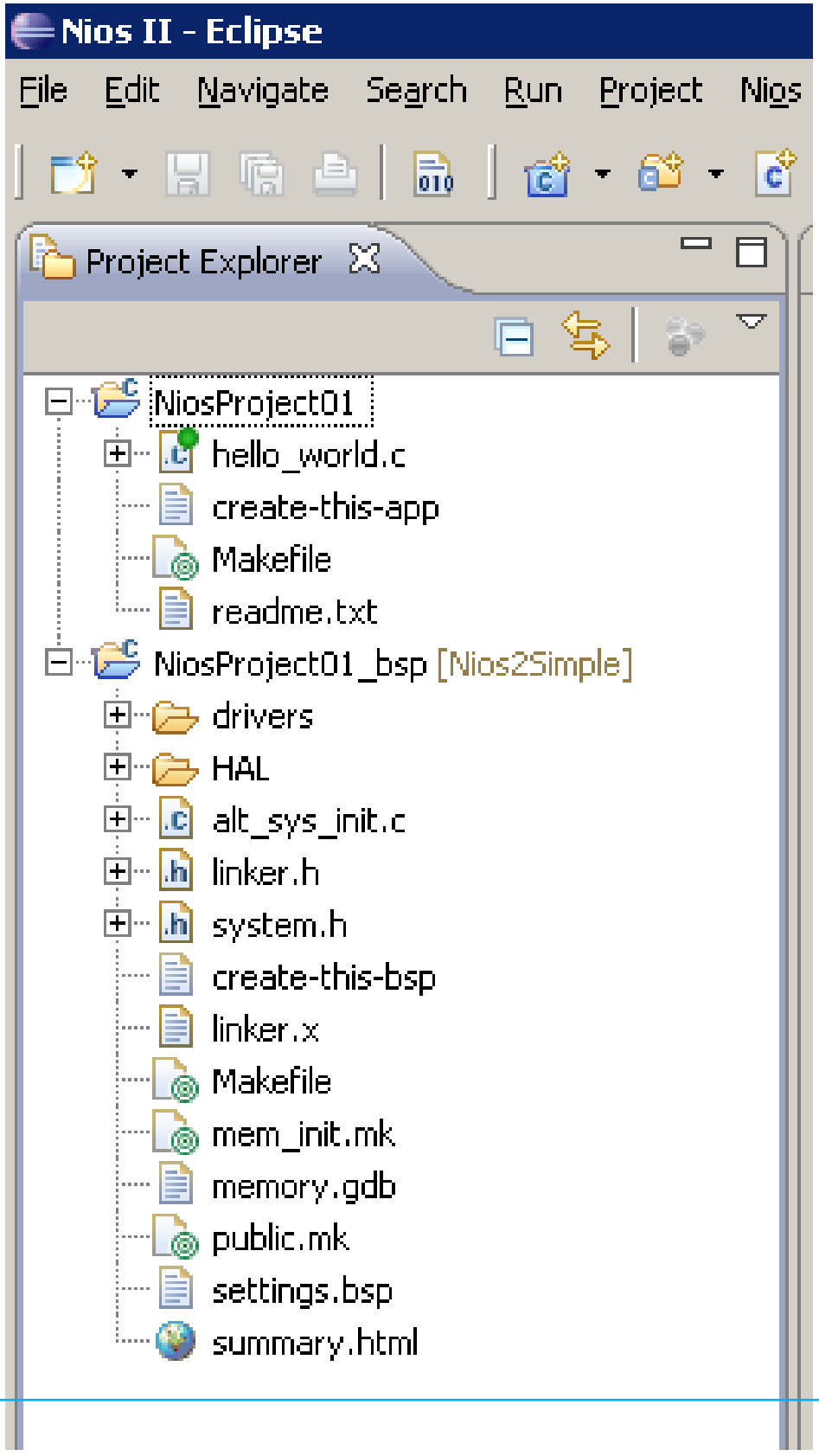
Σχήμα Γ΄.15: Με το εργαλείο QSYS αυτοματοποιείται η δημιουργία ενός ενσωματωμένου συστήματος που βασίζεται σε επεξεργαστή μαλακού πυρήνα.



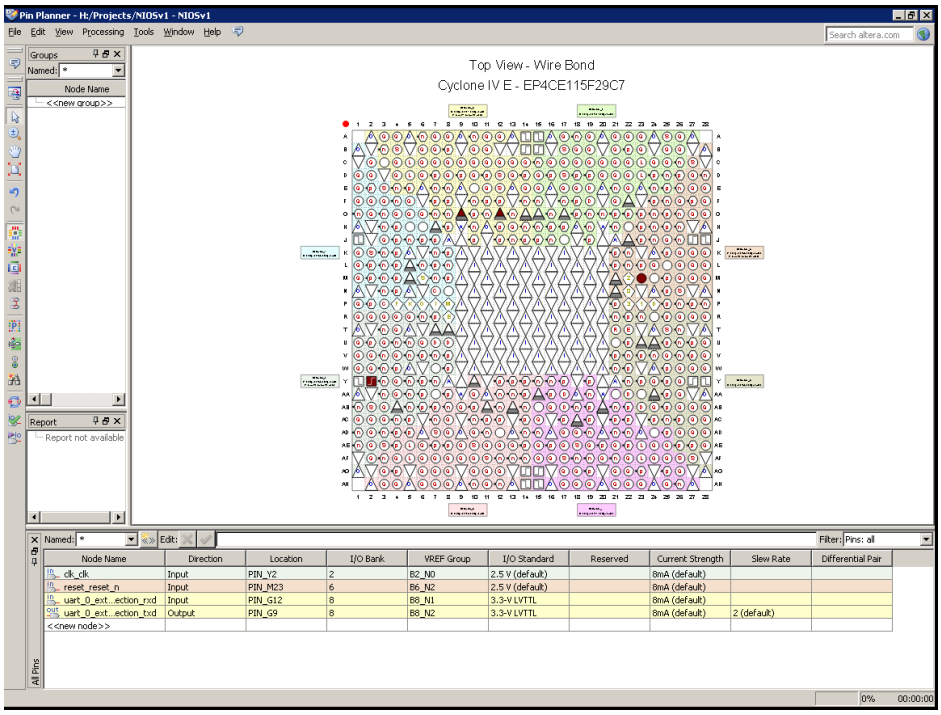
Σχήμα Γ΄.16: Η τελική μορφή του απλού ενσωματωμένου συστήματος.



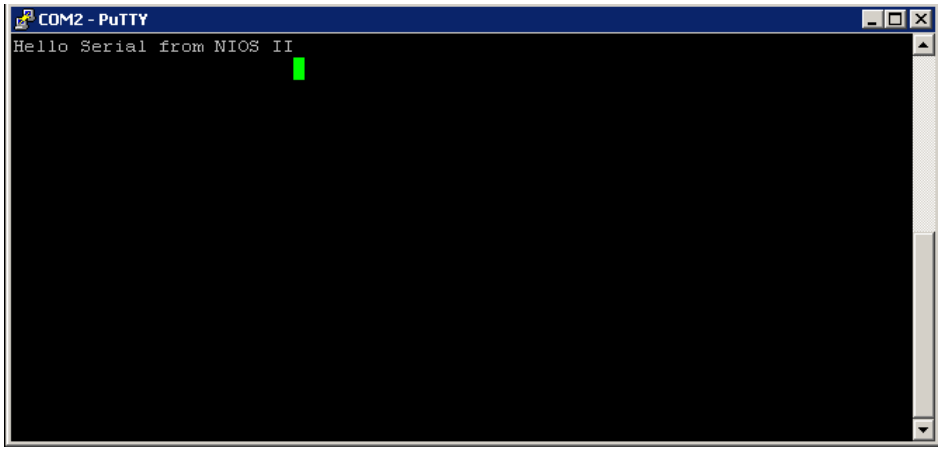
Σχήμα Γ΄.17: Η δημιουργία του BSP όπως και η ανάπτυξη του λογισμικού γίνεται από το Eclipse με υποστήριξη NIOS II



Σχήμα Γ΄.18: Με τη δημιουργία της εφαρμογή και του BSP, εμφανίζονται δυο subprojects στο Project Explorer του Eclipse



Σχήμα Γ΄.19: Στο Pin Planner πρέπει να γίνει η αντιστοίχηση των ακροδεκτών.



Σχήμα Γ΄.20: Έξοδος του προγράμματος σε ένα τερματικό σειριακής οθόνης.