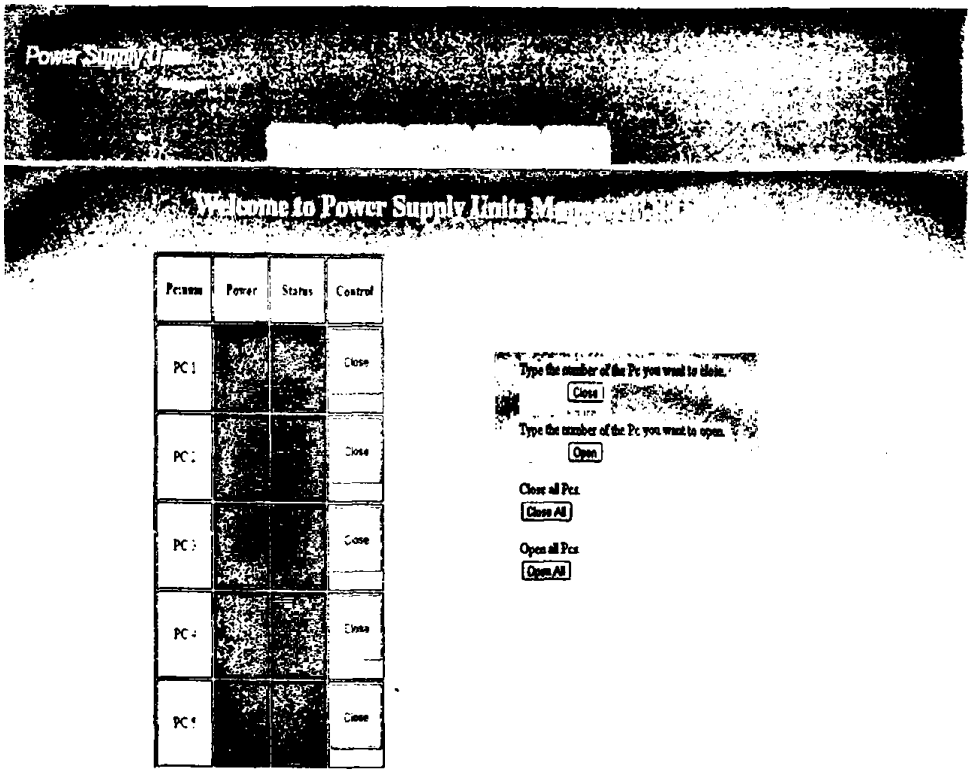
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΦΥΣΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟΥΠΟΛΗ ΙΩΑΝΝΙΝΩΝ, ΙΩΑΝΝΙΝΑ

ΠΜΣ-ΣΗΤ Ιανουάριος 2013

Ανάπτυξη συστήματος σε FPGA απομακρυσμένης διαχείρισης και ελέγχου της τροφοδοσίας μιας συστοιχίας ηλεκτρονικών υπολογιστών.

Ξουπλίδης Γ. Παύλος

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Παπαδόπουλος Ιωάννης



ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Περίληψη στα ελληνικά

Στην διπλωματική εργασία που ακολουθεί, αναλύεται η σχεδίαση, η υλοποίηση και η κατασκευή ενός συστήματος απομακρυσμένου ελέγχου και διαχείρισης της τροφοδοσίας και της λειτουργίας μιας συστοιχίας ηλεκτρονικών υπολογιστών. Η ανάπτυξη της Κεντρικής Μονάδας Ελέγχου που ελέγχει και διαχειρίζεται το δίκτυο του συστήματος αναπτύχθηκε σε FPGA (*Field Programmable Gate Array*). Σε κάθε υπολογιστή του δικτύου εγκαθίσταται ένα κύκλωμα διευθυνσιοδότησης που επικοινωνεί με την Κεντρική Μονάδα Ελέγχου μέσω καλωδίων Ethernet . Ο χρήστης επικοινωνεί μέσω της Ιστοσελίδας Διαχείρισης με την Κεντρική Μονάδα Ελέγχου και έχει τη δυνατότητα να ελέγχει την κατάσταση της τροφοδοσίας αλλά και την κατάσταση της λειτουργίας κάθε υπολογιστή του δικτύου.

Abstract

In this thesis, we present the design and construction of a system for the remote management and control of the power supply units of an array of computers. The development of the Master Control Unit which manages and controls the power supply units was developed in FPGA (Field Programmable Gate Array). In every computer an address circuit is installed that communicates with the master controller via ethernet cable. The user communicates through the Management Website with the Master Control Unit with the ability to check the status of the power supply units and the operation of the computers. Finally, the user can manage the operation of the computers.

Πρόλογος

Η διπλωματική εργασία εκπονήθηκε στα πλαίσια απόκτησης του μεταπτυχιακού διπλώματος ειδίκευσης στις Σύγχρονες Ηλεκτρονικές Τεχνολογίες στη φυσική του Τμήματος Φυσικής του Πανεπιστημίου Ιωαννίνων. Περιγράφει την ανάπτυξη, σχεδίαση και υλοποίηση ενός συστήματος που αναλαμβάνει τον έλεγχο και την διαχείριση από απόσταση μιας συστοιχίας ηλεκτρονικών υπολογιστών.

Ευχαριστώ πάρα πολύ τον επιβλέποντα καθηγητή μου κ Ιωάννη Παπαδόπουλο για την πάρα πολύ καλή συνεργασία που είχαμε όλα αυτά τα χρόνια. Θέλω να ευχαριστήσω επίσης τον Γιάννη Λουκόπουλο μεταπτυχιακό φοιτητή του ΠΜΣ-ΣΗΤ για την βοήθεια στην σχεδίαση των πλακετών και τέλος τον τεχνικό τού Εργαστηρίου Υψηλών Ενεργειών Στάθη Μπλέτσα για την βοήθεια στην κατασκευή.

Θέλω να αφιερώσω την διπλωματική εργασία στην οικογένεια μου(μεγάλη υποστήριξη), στην κοπέλα μου (μεγάλη υπομονή) και στους φίλους μου(μεγάλοι καφέδες).

Ημερομηνία 18/01/2013

Συγγραφέας

Ξουπλίδης Γ. Παύλος

Πίνακας περιεχομένων

Εισαγωγή

Ε.1 Αντικείμενο της διπλωματικής εργασίας

Αντικείμενο της διπλωματικής είναι η σχεδίαση, η υλοποίηση και η κατασκευή ενός συστήματος το οποίο θα ελέγχει και θα διαχειρίζεται από απόσταση την τροφοδοσία και τη λειτουργία μιας συστοιχίας ηλεκτρονικών υπολογιστών.

Το σύστημα αποτελείται από τρία μέρη. Τις πλακέτες που εγκαθίστανται σε κάθε υπολογιστή (κύκλωμα διευθυνσιοδότησης) για τον έλεγχο και την διαχείριση, την πλατφόρμα υλοποίησης SPARTAN-3E 1600E Development Board (κεντρική μονάδα ελέγχου) στην οποία είναι συνδεδεμένες οι πλακέτες και την ιστοσελίδα (ιστοσελίδα διαχείρισης) στην οποία εμφανίζονται τα αποτελέσματα του ελέγχου.

Το σύστημα ελέγχει τους υπολογιστές μιας συστοιχίας και εμφανίζει τα αποτελέσματα στην ιστοσελίδα. Από την ιστοσελίδα διαχείρισης ο διαχειριστής του συστήματος ενημερώνεται για την κατάσταση της τροφοδοσίας και λειτουργίας των υπολογιστών. Μπορεί επίσης να ενεργοποιήσει ή να απενεργοποιήσει τους υπολογιστές. Τέλος, μπορούν να ενεργοποιηθούν ή να απενεργοποιηθούν όλοι οι υπολογιστές της συστοιχίας ταυτόχρονα.

Ε.2 Δομή της διπλωματικής εργασίας

Στο πρώτο κεφάλαιο περιγράφεται η βασική ιδέα και ο σχεδιασμός του συστήματος. Αναφέρονται οι λόγοι που οδήγησαν στην υλοποίηση και τη χρήση ενός τέτοιου συστήματος. Εξηγείται η λειτουργία του και περιγράφεται η τοποθέτηση του. Τέλος, υπολογίζεται το κόστος.

Στο δεύτερο κεφάλαιο περιγράφεται η λειτουργία και η υλοποίηση της κεντρικής μονάδας ελέγχου, της βασικής δομής του συστήματος.

Στο τρίτο κεφάλαιο περιγράφεται η λειτουργία και η υλοποίηση του κυκλώματος

διευθυνσιοδότησης.

Στο τέταρτο κεφάλαιο περιγράφεται η λειτουργία και η υλοποίηση της ιστοσελίδας διαχείρισης.

Στο πέμπτο κεφάλαιο παρουσιάζονται τα αποτελέσματα των ελέγχων που διενεργήθηκαν στο σύστημα, τα συμπεράσματα που εξάγονται από τα αποτελέσματα και οι μελλοντικές επεκτάσεις του συστήματος.

E.3 Ορισμοί και Ορολογία

► **Κεντρική Μονάδα Ελέγχου**: αποτελείται από την πλατφόρμα υλοποίησης SPARTAN-3E 16000E Development Board. Συνδέεται μέσω της ενσωματωμένης στην πλατφόρμα θύρας Ethernet με τον υπολογιστή που διαχειρίζεται την ιστοσελίδα διαχείριση. Επίσης, συνδέεται με τα κυκλώματα διευθυνσιοδότησης μέσω καλωδίου τύπου ethernet.

► **Κύκλωμα Διευθυνσιοδότησης:** Οι πλακέτες που εγκαθίστανται σε κάθε υπολογιστή και παρέχουν μια μοναδική διεύθυνση στον υπολογιστή στο δίκτυο του συστήματος.

► **Καλώδιο Τύπου Ethernet** : Τα καλώδια που συνδέουν τα κυκλώματα διευθυνσιοδότησης με την κεντρική μονάδα ελέγχου. Είναι καλώδια τύπου ethernet ειδικά τροποποιημένα στις ανάγκες του συστήματος.

► **Δίκτυο Συστήματος :** Κάθε υπολογιστής που είναι συνδεδεμένος με την κεντρική μονάδα είναι μέλος το δικτύου του συστήματος ελέγχου και διαχείρισης. Το δίκτυο δεν έχει κάποια γνωστή αρχιτεκτονική. Οι υπολογιστές δεν επικοινωνούν μεταξύ τους, ούτε ανταλλάσσουν δεδομένα παρά μόνο με την κεντρική μονάδα ελέγχου.

Κεφάλαιο 1

Το σύστημα

Στο κεφάλαιο, περιγράφεται η βασική ιδέα του συστήματος απομακρυσμένου ελέγχου και διαχείρισης. Ποιες ανάγκες οδήγησαν στη χρήση ενός τέτοιου συστήματος και ποια η λειτουργία του. Πως προέκυψε ο σχεδιασμός και ποια η διαδικασία τοποθέτησής του. Πως αλληλεπιδρά ο χρήστης με το σύστημα και τέλος ποιο το κόστος ενός τέτοιου συστήματος.

1.1 Χρήση και λειτουργία

Το σύστημα απομακρυσμένου ελέγχου και διαχείρισης δημιουργήθηκε λόγω της εγκατάστασης μιας συστοιχίας ηλεκτρονικών υπολογιστών στο εργαστήριο υψηλών ενεργειών του τμήματος φυσικής του πανεπιστημίου Ιωαννίνων. Η συστοιχία συνδέεται με το υπολογιστικό πλέγμα [1] και εκτελεί εργασίες επεξεργασίας και αποθήκευσης δεδομένων. Λόγω της συνεχούς λειτουργίας των υπολογιστών της συστοιχίας σε 24ωρη βάση καθίσταται απαραίτητη η χρήση ενός συστήματος το οποίο θα παρακολουθεί την κατάσταση της λειτουργία τους σε πραγματικό χρόνο. Προβλήματα του δικτύου ρεύματος, όπως αυξομειώσεις τάσης λόγω καιρικών φαινομένων και διακοπές ρεύματος, οδήγησαν στην ανάγκη συνεχούς παρακολούθησης της τροφοδοσίας των υπολογιστών.

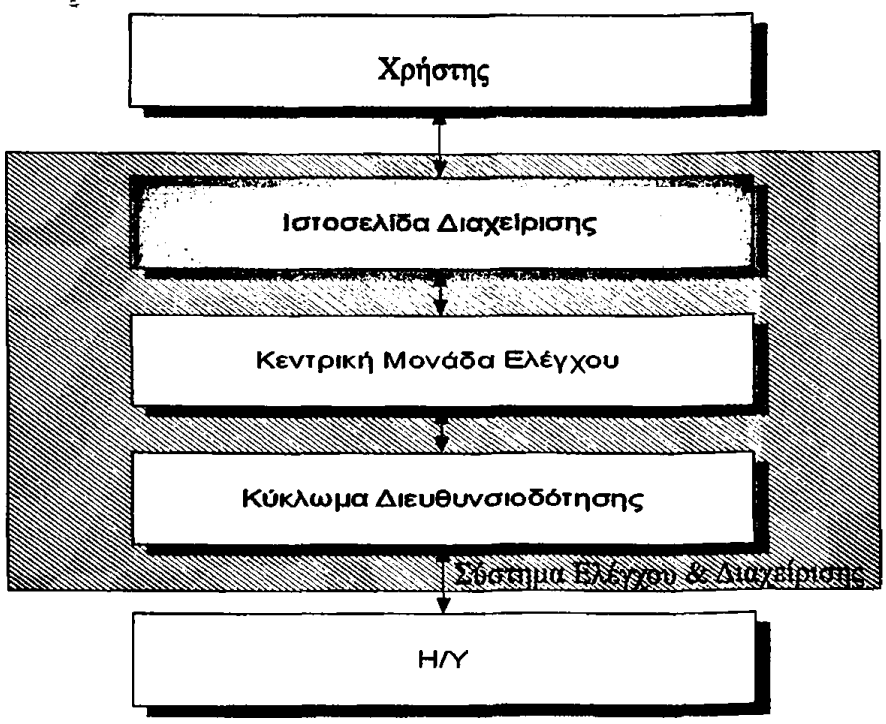
Η ικανότητα του συστήματος (πέραν της παρακολούθησης της τροφοδοσίας και της λειτουργίας) να διαχειρίζεται τη λειτουργία των υπολογιστών, κρίθηκε απαραίτητη λόγω ορισμένων προβλημάτων τα οποία δεν μπορούν να λυθούν με μια απλή επανεκκίνηση. Ένα τέτοιου είδους πρόβλημα, για παράδειγμα, είναι το SATA bus Freeze, αποτέλεσμα του οποίου είναι η διακοπή της λειτουργίας των σκληρών δίσκων τεχνολογίας SATA [2] λόγω της.

Αποτέλεσμα των παραπάνω αναγκών και προβλημάτων κατά την διάρκεια λειτουργίας των υπολογιστών της συστοιχίας ήταν η υλοποίηση ενός συστήματος το οποίο θα παρακολουθεί συνεχώς την τροφοδοσία και τη λειτουργία των υπολογιστών της συστοιχίας σε πραγματικό χρόνο και θα δίνει τη δυνατότητα στον διαχειριστή της συστοιχίας να επεμβαίνει στην λειτουργία των υπολογιστών.

Στην περίπτωση διακοπής ρεύματος το σύστημα εκκινείται αυτόματα και συνεχίζει και πάλι τη λειτουργία του όταν επανέλθει το ρεύμα.

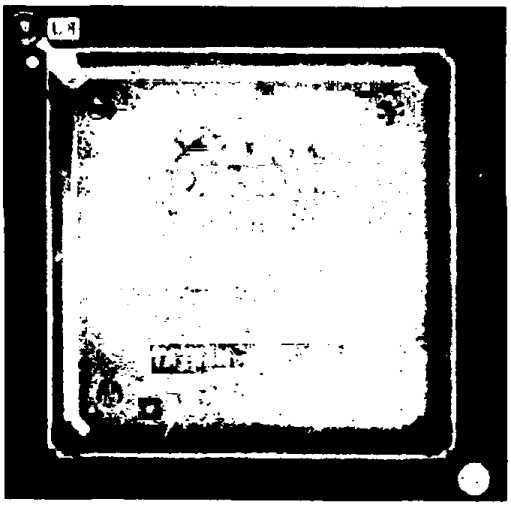
1.2 Σχεδιασμός και Αλληλεπίδραση

Το σύστημα ελέγχου και διαχείρισης αποτελείται από τρία τμήματα, όπως φαίνεται στην εικόνα 1. Την κεντρική μονάδα ελέγχου, τα κυκλώματα διευθυνσιοδότησης και την ιστοσελίδα διαχείρισης.



Εικόνα 1: Δομή του Συστήματος Ελέγχου και Διαχείρισης.

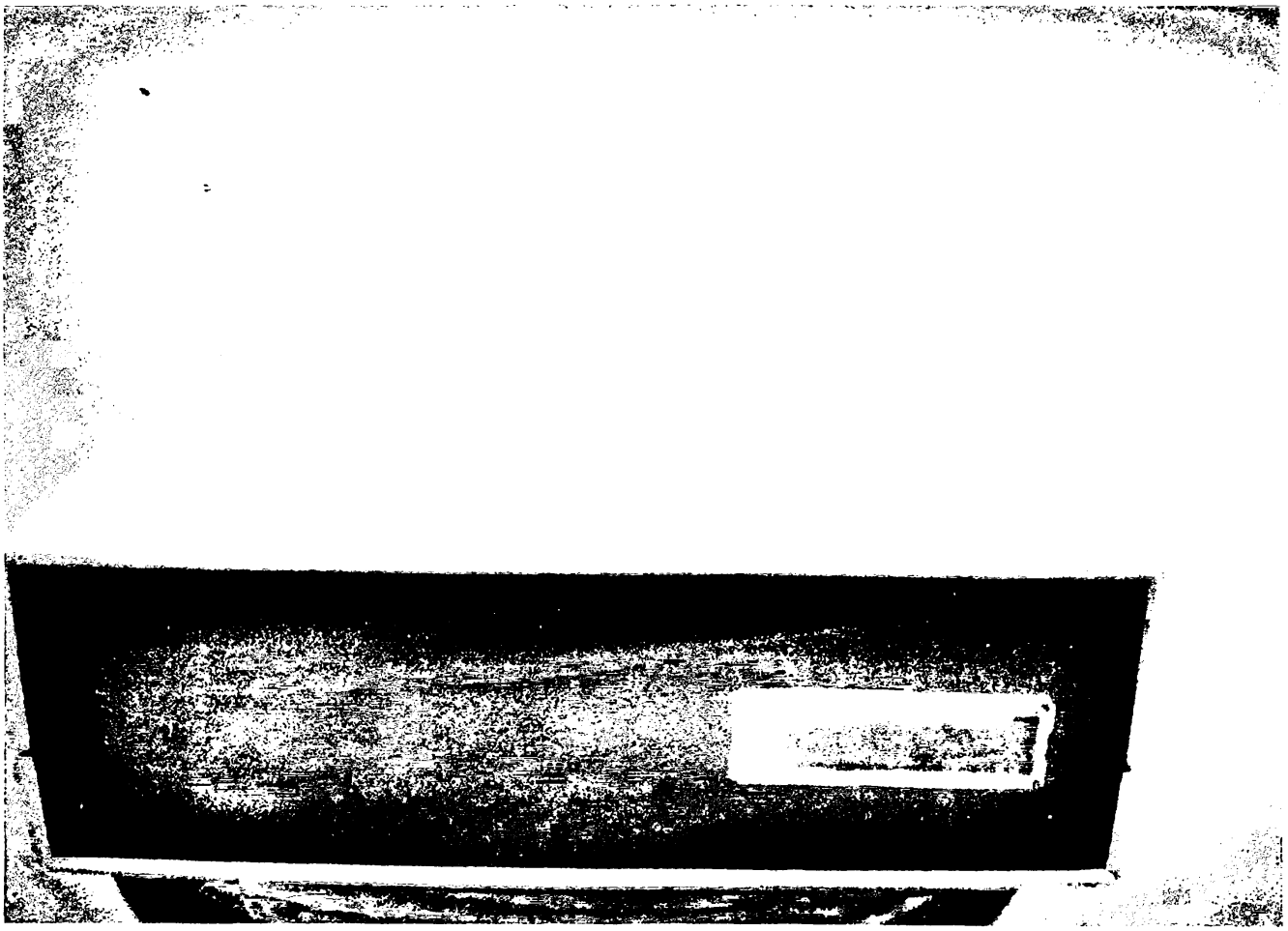
Ο σχεδιασμός της κεντρικής μονάδας βασίστηκε στην τεχνολογία των FPGA (Field-Programmable Gate Array) (εικόνα 2). Η τεχνολογία των FPGA έγινε γνωστή στην αγορά το 1985 από τους Ross Freeman και Bernard Vonderschmitt συν-ιδρυτές της εταιρίας Xilinx.inc [3], της πρώτης εταιρίας που κατασκεύασε ένα εμπορικό FPGA με τον κωδικό XC2064. Ένα FPGA, σύμφωνα με τον κατασκευαστή, είναι ένα ολοκληρωμένο κύκλωμα το οποίο μπορεί να προγραμματιστεί από τον πελάτη. Καθώς ένα FPGA χάνει τον προγραμματισμό του εκτός τροφοδοσίας μπορεί να προγραμματιστεί εκ νέου. Αποτελείται από ένα σύνολο προγραμματιζόμενων πυλών, οι διασυνδέσεις των οποίων είναι και αυτές προγραμματιζόμενες. Ο αριθμός των κυκλωμάτων που μπορούν να υλοποιηθούν σε ένα FPGA είναι ανάλογος του αριθμού των πυλών από τις οποίες είναι κατασκευασμένο.



Εικόνα 2: Ένα FPGA της εταιρείας Xilinx.inc

Για την υλοποίηση του συστήματος, αρχικά χρησιμοποιήθηκε το FPGA της οικογένειας SPARTAN-3E [4] με κωδικό XCS3S500E κατασκευασμένο από την εταιρία Xilinx.inc. Το συγκεκριμένο FPGA βρισκόταν ενσωματωμένο στην πλατφόρμα SPARTAN-3E 500E Starter Kit της εταιρίας Digilent.inc [5]. Ο σχεδιασμός στην πλατφόρμα αυτή αποκλείστηκε λόγω των χαρακτηριστικών του FPGA και της περιορισμένης ενσωματωμένης μνήμης της πλατφόρμας (βλέπε κεφάλαιο 2).

Το FPGA της εταιρίας Xilinx με κωδικό XC3S1600E της οικογένειας SPARTA ενσωματωμένο στην πλατφόρμα SPARTAN-3E 1600E Development Board της εταιρίας Digile ήταν η επιλογή για το σχεδιασμό του συστήματος. Το σημαντικό πλεονέκτημα του XC3S1600E ότι υποστηρίζει την τεχνολογία Microblaze [6]. Η τεχνολογία Microblaze αναπτύχθηκε απ εταιρία Xilinx.inc και πρόκειται για την υλοποίηση ενός 32-bit Soft-Core επεξεργαστή στο F βασισμένο στην αρχιτεκτονική RISC του πανεπιστημίου του Harvard [7]. (Αναλυτικότερα γι τεχνολογία Microblaze βλέπε κεφάλαιο 2).

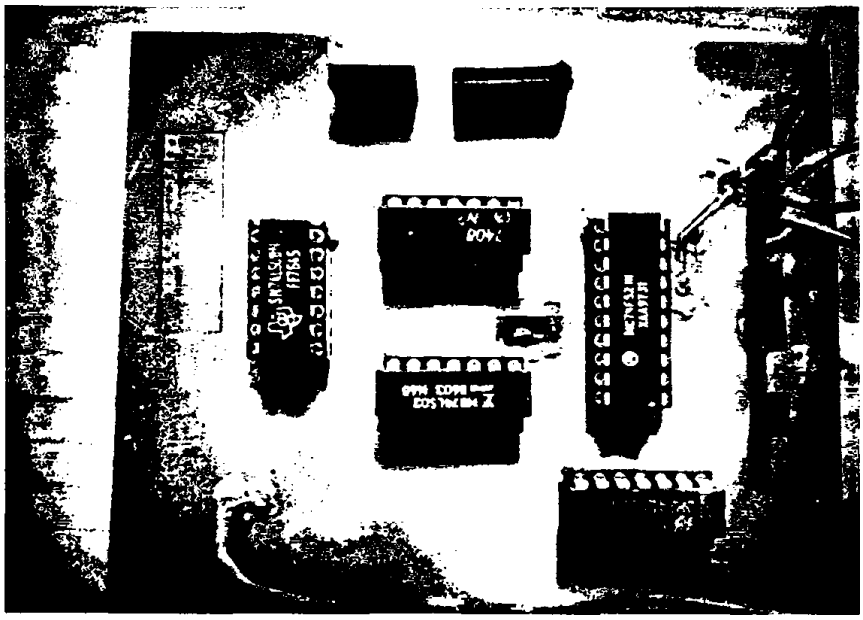


Εικόνα 3: Η Κεντρική Μονάδα Ελέγχου (Εμπρόσθια όψη)

Στην πλατφόρμα SPARTAN-3E 1600E Development Board έγινε η υλοποίηση της κεντ μονάδας ελέγχου του συστήματος η εμπρόσθια όψη της οποία φαίνεται στην εικόνα 3. Η κεν μονάδα ελέγχου είναι ο διαμεσολαβητής μεταξύ της ιστοσελίδας διαχείρισης και των κυκλωμ διευθυνσιοδότησης. Λαμβάνει δεδομένα από τα κυκλώματα διευθυνσιοδότησης και ενημερώνε

ιστοσελίδα του συστήματος για την κατάσταση της τροφοδοσίας και λειτουργίας των υπολογιστών. Από την ιστοσελίδα διαχείρισης γίνεται η διαχείριση και ο έλεγχος της λειτουργίας των υπολογιστών. Η επιλογή της ιστοσελίδας έγινε με κριτήριο την εύκολη πρόσβαση στο σύστημα από οποιοδήποτε μέσο με σύνδεση στο διαδίκτυο, π.χ. Η/Υ, κινητό τηλέφωνο, tablet. Εκτός από την ιστοσελίδα πρόσβασης στο σύστημα, υποστηρίζεται και η δυνατότητα πρόσβασης μέσω πρωτοκόλλου Telnet [8].

Το κύκλωμα διευθυνσιοδότησης το οποίο φαίνεται στην εικόνα 4 εγκαθίσταται σε κάθε υπολογιστή που πρόκειται να συνδεθεί με το σύστημα (βλέπε ενότητα 1.3) παρέχοντας του μια μοναδική διεύθυνση στο δίκτυο του συστήματος ελέγχου και διαχείρισης. Τα κυκλώματα διευθυνσιοδότησης ενεργοποιούνται από την κεντρική μονάδα ελέγχου βάσει της διεύθυνσής τους στο δίκτυο του συστήματος. Όταν ενεργοποιηθούν, αποστέλλουν την πληροφορία για την κατάσταση της τροφοδοσίας και της λειτουργίας του υπολογιστή και ενεργοποιούν ή απενεργοποιούν τον υπολογιστή όταν λάβουν την εντολή από την κεντρική μονάδα.



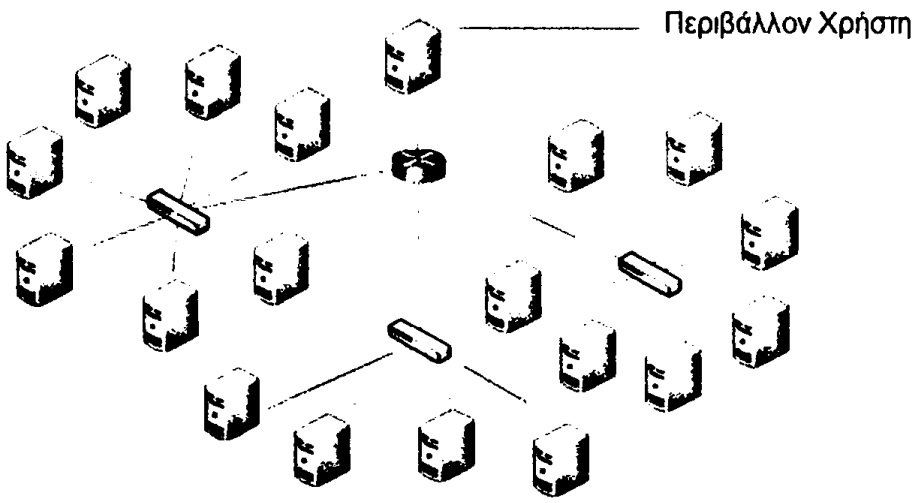
Εικόνα 4: Κύκλωμα Διευθυνσιοδότησης.

Η πρόσβαση στο σύστημα γίνεται από την ιστοσελίδα διαχείρισης. Η ιστοσελίδα διαχείρισης επικοινωνεί με την κεντρική μονάδα ελέγχου και εμφανίζει την κατάσταση της τροφοδοσίας και της λειτουργίας των υπολογιστών της συστοιχίας. Στην ιστοσελίδα διαχείρισης υπάρχει και η δυνατότητα

ενεργοποίησης ή απενεργοποίησης των υπολογιστών.

1.3 Τοποθέτηση και κόστος συστήματος

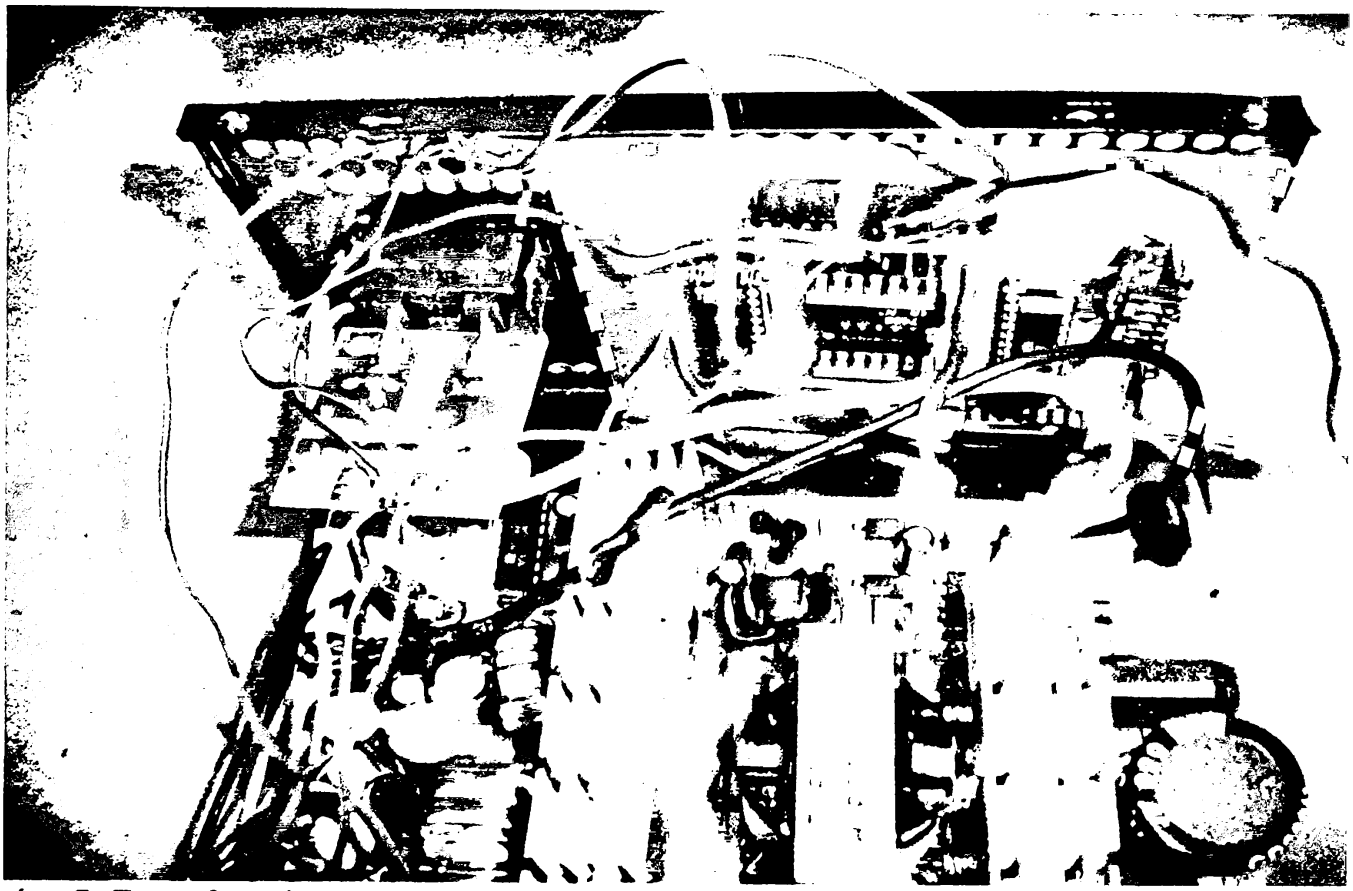
Το σχεδιάγραμμα του συστήματος διαχείρισης και ελέγχου φαίνεται στην εικόνα 5. Το σύστημα αποτελείται από τον υπολογιστή που φιλοξενεί την ιστοσελίδα διαχείρισης (web server), την κεντρική μονάδα ελέγχου και τα κυκλώματα διευθυνσιοδότησης που εγκαθίστανται σε κάθε υπολογιστή της συστοιχίας . Ο υπολογιστής που φιλοξενεί την ιστοσελίδα διαχείρισης μπορεί να είναι συνδεδεμένος με το σύστημα ελέγχου είτε άμεσα (LAN) είτε μέσω διαδικτύου.



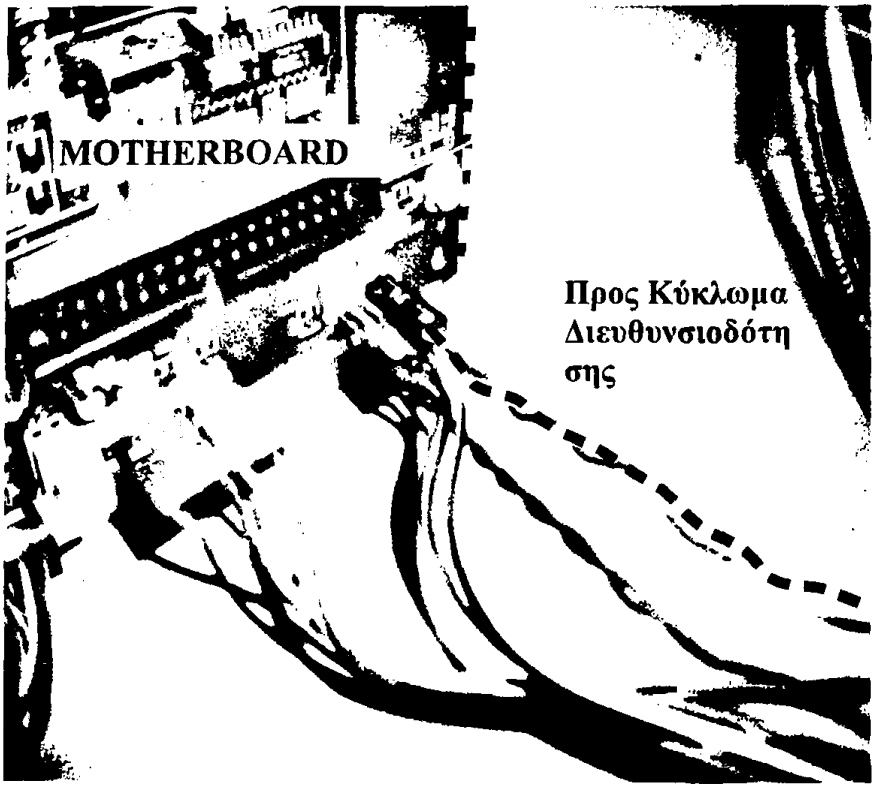
Εικόνα 5: Το σύστημα απομακρυσμένου ελέγχου και διαχείρισης

Το κύκλωμα διευθυνσιοδότησης τοποθετείται σε μια υποδοχή PCI του υπολογιστή, όπως φαίνεται στην εικόνα 6 ή ενσωματώνεται στο τροφοδοτικό, όπως φαίνεται στη εικόνα 7. Το κύκλωμα ελέγχει το τροφοδοτικό του υπολογιστή. Η σύνδεση του κυκλώματος διευθυνσιοδότησης φαίνεται στην εικόνα 8. Το κουμπί εκκίνησης του υπολογιστή συνδέεται με το κύκλωμα, ενώ το κύκλωμα συνδέεται στον διακόπτη εκκίνησης της μητρικής πλακέτας.

Εικόνα 6: Κύκλωμα Διευθυνσιοδότησης σε μορφή PCI, τοποθετημένο στο κουτί ενός Η/Υ.



Εικόνα 7: Τροφοδοτικό (PSU) στο οποίο έχει ενσωματωθεί το Κύκλωμα Διευθυνσιοδότησης.

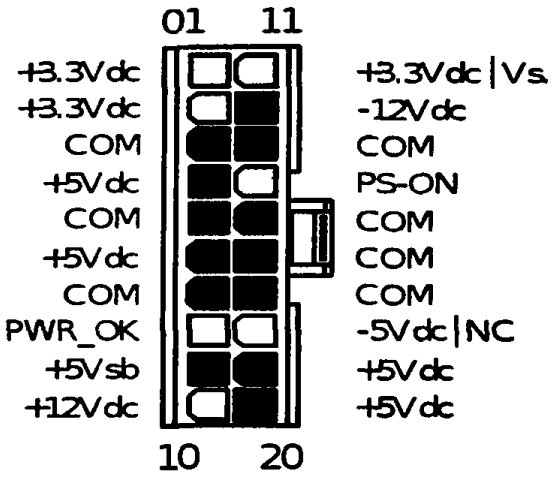


Εικόνα 8: Σύνδεση του Κυκλώματος Διευθυνσιοδότησης στη μητρική πλακέτα.

Στην εικόνα 9 φαίνεται μία τυπική διασύνδεση τροφοδοτικού 20-pins [9]. Το κύκλωμα διευθυνσιοδότησης συνδέεται στο μοβ καλώδιο για τον έλεγχο της τροφοδοσίας του υπολογιστή, καθώς και για την τροφοδοσία του ίδιου του κυκλώματος. Με την σύνδεση του κυκλώματος στο γκρι καλώδιο λαμβάνεται η πληροφορία για την κατάσταση της λειτουργίας του υπολογιστή.

Τα κυκλώματα διευθυνσιοδότησης συνδέονται με την κεντρική μονάδα ελέγχου μέσω καλωδίου τύπου ethernet που χρησιμοποιείται με ειδικό τρόπο. Το καλώδιο τύπου ethernet αποτελείται από οκτώ καλώδια. Το σύστημα ελέγχου και διαχείρισης χρησιμοποιεί τα τέσσερα καλώδια για τη διευθυνσιοδότηση των κυκλωμάτων, τα δύο για τη λήψη δεδομένων από τα κυκλώματα και το ένα για την αποστολή του bit ενεργοποίησης ή απενεργοποίησης. Το σύστημα χρησιμοποιεί το 5 καλώδιο για την επέκταση της διευθυνσιοδότησης για μεγαλύτερο πλήθος υπολογιστών.

Ο υπολογιστής που φιλοξενεί την ιστοσελίδα διαχείρισης συνδέεται μέσω πρωτοκόλλου telnet με την κεντρική μονάδα ελέγχου.



Εικόνα 9: 20-pin ATX Connector

Θα περίμενε κανείς ότι το κόστος ενός τέτοιου συστήματος θα ήταν μεγάλο. Όμως ο σχεδιασμός του συστήματος και η χρήση της τεχνολογίας των FPGA κατέστησε το σύστημα πολύ οικονομικό σε σχέση με τις δυνατότητες που παρέχει. Στον πίνακα 1, βλέπουμε αναλυτικά το κόστος του συστήματος για τη διαχείριση δύο υπολογιστών.

Υλικά	Κόστος (€)
SPARTAN-3E 1600E Development Board	250
Κουτί κατασκευής κεντρικής μονάδας	15
Πύλες	2.3
Comparator	1.6
Καλώδια ethernet	4,5
Κατασκευή πλακετών	6
Σύνολο	280

Πίνακας 1: Αναλυτικό κόστος συστήματος.

Κεφάλαιο 2

Κεντρική Μονάδα Ελέγχου

Στο κεφάλαιο αυτό περιγράφεται η υλοποίηση και η λειτουργία της κεντρικής μονάδας ελέγχου στο SPARTAN-3E 1600E Development Board. Το κεφάλαιο ξεκινά με την παρουσίαση των χαρακτηριστικών της πλατφόρμας και του επεξεργαστή Microblaze που υλοποιήθηκε σε αυτή. Στη συνέχεια του κεφαλαίου, αναφέρεται η διαδικασία υλοποίησης του λογισμικού. Πιο συγκεκριμένα, περιγράφεται η υλοποίηση του λειτουργικού Standalone [10], της εφαρμογής LwIP [11] και του πρωτοκόλλου δικτύου Telnet.

2.1 Εργαλεία Υλοποίησης

2.1.1. Το SPARTAN-3E 1600E Development Board

Το SPARTAN-3E 1600E Development Board [12] φαίνεται στην εικόνα 10. Πρόκειται για μια προηγμένη αυτόνομη πλατφόρμα σχεδιασμού, διανομής της εταιρίας Digilent. Η πλατφόρμα έχει ενσωματωμένο το FPGA με κωδικό XC3S1600E της οικογένειας SPARTAN-3E, κατασκευάστρια της οποίας είναι η εταιρία Xilinx.inc. Το XC3S1600E διαθέτει 1600K πύλες και είναι κατασκευασμένο για τη σχεδίαση μεγάλων και σύνθετων ψηφιακών συστημάτων, όπως του Microblaze RISC soft processor 32-Bit με τις διασυνδέσεις της ενσωματωμένης μνήμης DDR.

Ο προγραμματισμός της πλατφόρμας και η ενσωμάτωση του σχεδίου στο FPGA γίνεται μέσω των ενσωματωμένων θυρών USB2, καθώς και των θυρών RS232 DTE και DTC. Οι εξωτερικές μνήμες Platform Flash, Intel StrataFlash, ST Microelectonics Serial Flash παρέχουν επιπλέον δυνατότητες αποθήκευσης και αρχικοποίησης του FPGA.

Τα χαρακτηριστικά της πλατφόρμας είναι τα εξής:

- **Ολοκληρωμένα της εταιρίας Xilinx:**
  - Spartan-3E FPGA (XC3S500E-4FG320C)

- CoolRunner™-II CPLD (XC2C64A-5VQ44C)
- Platform Flash (XCF04S-VO20C)

•Ρολόι χρονισμού:

- 50 MHz crystal clock oscillator

• Μνήμη:

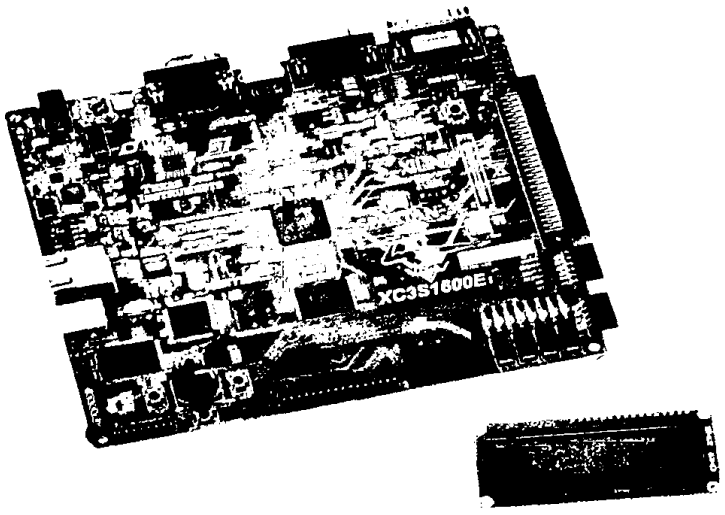
- 128 Mbit Parallel Flash
- 16 Mbit SPI Flash
- 64 MByte DDR SDRAM

• Θύρες επικοινωνίας και θύρες εισόδου/εξόδου:

- SMSC LAN83C185 Ethernet PHY
- JTAG USB download
- Two 9-pin RS-232 serial port
- PS/2- style mouse/keyboard port, rotary encoder with push button
- Four slide switches
- Eight individual LED outputs
- Four momentary-contact push buttons
- 100-pin expansion connection ports
- Three 6-pin expansion connectors

• Οθόνη:

- 16 character - 2 Line LCD



Εικόνα 10: SPARTAN-3E 1600E Development Board

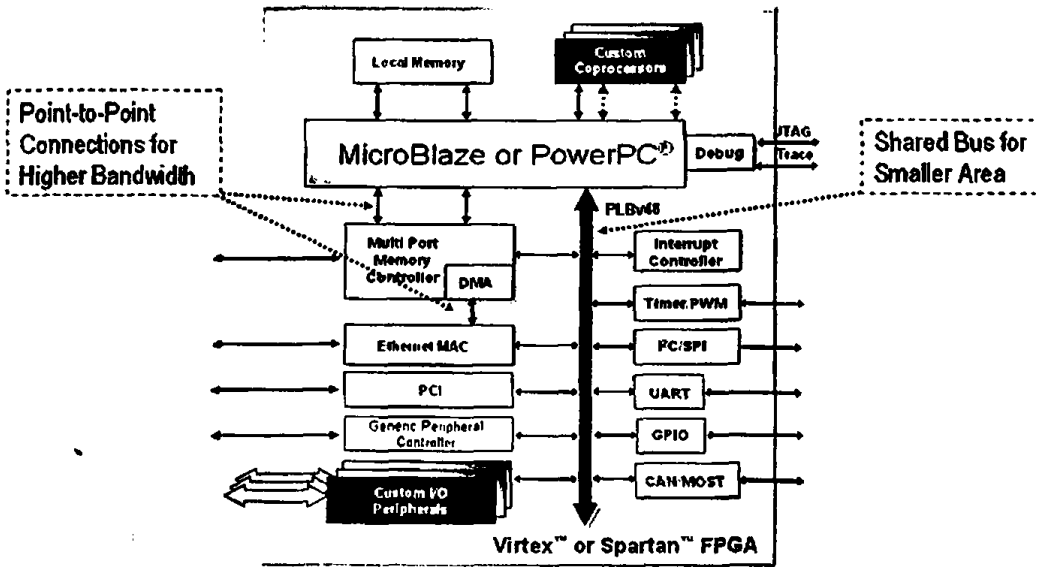
2.1.2. *Επεξεργαστής Microblaze*

Ο επεξεργαστής MicroBlaze (εικόνα 11) είναι ένας Soft-Core επεξεργαστής 32-bit βασισμένος στην αρχιτεκτονική RISC του πανεπιστημίου Harvard. Ένας Soft-Core επεξεργαστής συντίθεται χρησιμοποιώντας λογικά blocks του FPGA. Ο επεξεργαστής Microblaze μας επιτρέπει μεγάλο βαθμό ευελιξίας και παραμετροποίησης. Η υλοποίηση του επεξεργαστή γίνεται από το λογισμικό EDK (Embedded Development Kit) [13]. Το EDK είναι μέρος της σχεδιαστικής πλατφόρμας, που μας προσφέρει η εταιρία Xilinx.inc για το σχεδιασμό ενσωματωμένων συστημάτων.

Τα χαρακτηριστικά του επεξεργαστή Microblaze είναι:

- Low Latency Interrupt Mode
- Fault Tolerance, including Error Correction Codes (ECC) and Lockstep support
- LMB BRAM memory
- Parity protection on internal BRAMs and caches
- Floating Point Unit (FPU)
- IEEE 754 compatible
- Single precision
- Memory Management Unit (MMU)
- Full MMU with Virtual Memory supported by Linux
- MPU mode for region protection for secure RTOS applications
- Instruction and Data Caches
- Cache size configurable: 2kB - 64kB (Block RAM based)
- Microcache size configurable: 64B – 1024B (Distributed RAM based)
- Direct mapped write-through or write-back operation
- Victim cache size configurable: 2, 4 or 8 cache lines
- Instruction Stream Buffers
- Branch Optimizations
- Branch prediction logic
- Branch target cache
- Execution Hardware Acceleration

- Barrel Shifter (1 cycle operation)
- Integer Divide (32 cycle operation)
- Multiply (1 cycle operation)
- Instruction Set Extensions
- Pattern Compare Instructions
- Machine Status Register Set and Clear
- Atomic Access
- Endian Conversion Support
- Hardware Exception Support
- Unaligned access
- Illegal instruction
- Data bus error
- Instruction bus error
- Divide Exception
- Floating Point Exception
- FSL Exception
- MMU Exception
- Interrupt Signaling – Edge or level
- Debug Logic
- JTAG control via a debug support core
- Up to 8 hardware break points
- Up to 8 hardware watch points



Εικόνα 11: Δομή του επεξεργαστή Microblaze

2.1.3. Λειτουργικό Standalone

Το λειτουργικό Standalone [14] είναι ένα απλό λειτουργικό σύστημα το οποίο επιτρέπει την πρόσβαση σε απλές λειτουργίες ενός επεξεργαστή. Κάποιες από τις λειτουργίες που μπορεί να εκτελέσει κανείς χρησιμοποιώντας το λειτουργικό Standalone είναι η διαχείριση του χρονισμού, της μνήμης, του δικτύου, των I/O pins και των άλλων περιφερειακών του επεξεργαστή.

2.1.4. Light Weight IP (LwIP)

Η εφαρμογή LwIP [15] που υλοποιήθηκε στον επεξεργαστή Microblaze είναι μία μικρή ανεξάρτητη εφαρμογή του πρωτοκόλλου TCP / IP που έχει αναπτυχθεί από τον Adam Dunkels στο Εργαστήριο Υπολογιστών και Δικτύων του Σουηδικού Ινστιτούτου Πληροφορικής. Η εφαρμογή LwIP διανέμεται δωρεάν υπό την άδεια BSD [16].

Η βασική ιδέα της υλοποίησης της εφαρμογής LwIP είναι η μείωση της χρήσης των πόρων του

συστήματος από το πρωτόκολλο TCP/IP, ενώ ταυτόχρονα διατηρείται ένα πλήρες πακέτο λειτουργιών. Αυτό καθιστά την LwIP κατάλληλη για χρήση σε ενσωματωμένα συστήματα με δεκάδες kilobytes ελεύθερης μνήμης RAM και χώρο για περίπου 40 kilobytes του κώδικα ROM.

Τα χαρακτηριστικά της εφαρμογής LwIP είναι τα εξής:

- IP (Internet Protocol) με προώθηση πακέτων σε πολλαπλά δίκτυα.
- ICMP (Internet Control Message Protocol) για αποσφαλμάτωση και συντήρηση του δικτύου.
- UDP (User Datagram Protocol) πειραματικές "ελαφριές" εφαρμογές του πρωτοκόλλου UDP.
- TCP (Transmission Control Protocol) έλεγχο συμφόρησης, εκτίμηση RTT, γρήγορη αναμετάδοση.
- Specialized raw API για βελτιωμένες επιδόσεις.
- Berkeley-alike socket API.
- DHCP (Dynamic Host Configuration Protocol).
- PPP (Point-to-Point Protocol).
- ARP (Address Resolution Protocol) for Ethernet.

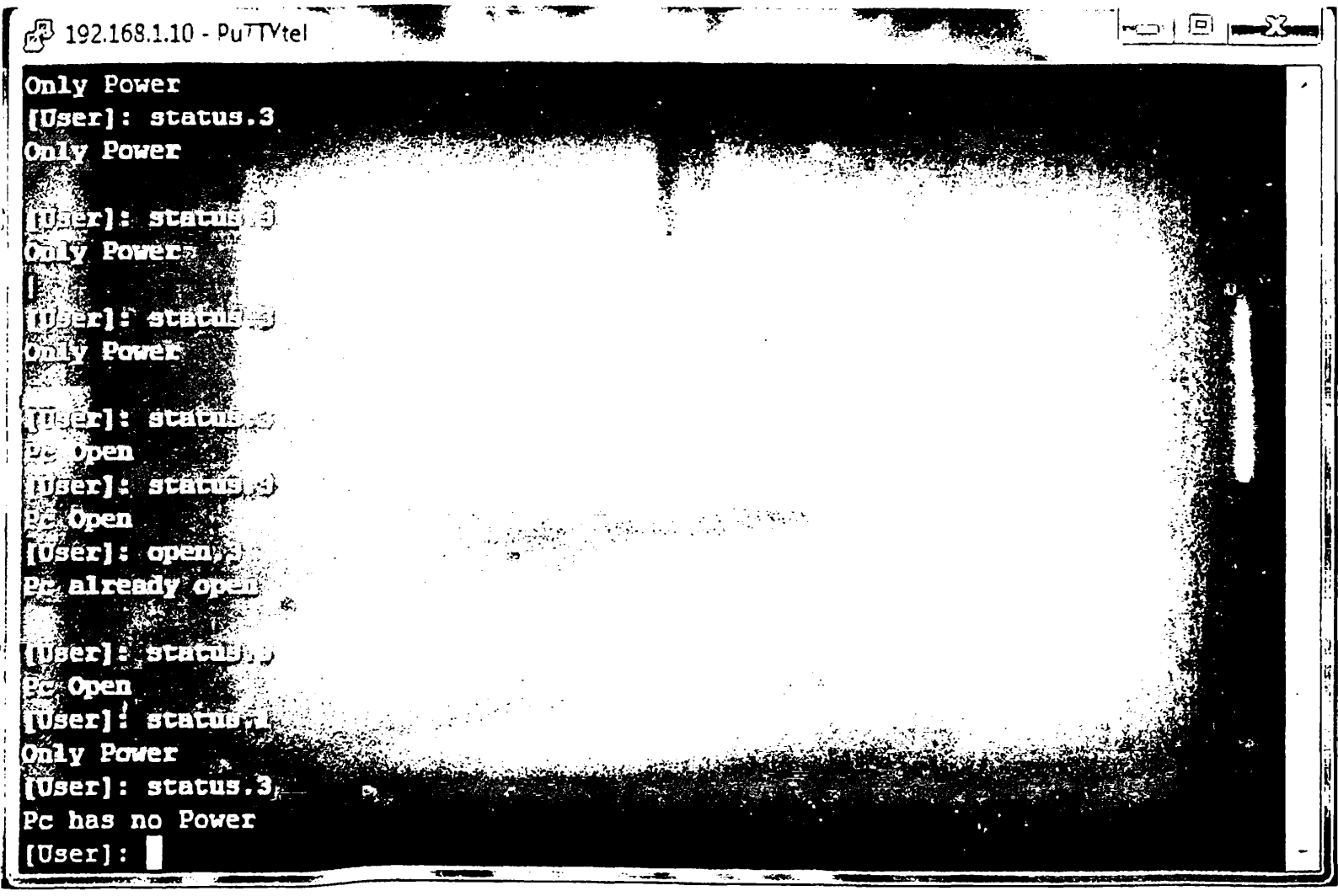
2.1.5. Πρωτόκολλο TELNET

Το Telnet είναι μια υπηρεσία του Internet που μας επιτρέπει να συνδεόμαστε με έναν απομακρυσμένο υπολογιστή και να δουλεύουμε χρησιμοποιώντας τα προγράμματά του σαν να είμαστε άμεσα συνδεδεμένοι μαζί του. Με άλλα λόγια, το δικό μας τερματικό - προσωπικός υπολογιστής, workstation, τερματικό ενός UNIX συστήματος, κλπ. - μετατρέπεται σε τερματικό του απομακρυσμένου υπολογιστή ο οποίος ανταποκρίνεται στις εντολές μας.

Το Telnet βασίζεται στην αρχιτεκτονική Client/Server: για να χρησιμοποιήσουμε το Telnet, εκτελούμε στον υπολογιστή μας ένα πρόγραμμα πελάτη για Telnet (Telnet client), ενώ στον απομακρυσμένο υπολογιστή εκτελείται ένα πρόγραμμα που ονομάζεται εξυπηρετητής Telnet (Telnet server). Ο Telnet server διαχειρίζεται ταυτόχρονα της αιτήσεις δημιουργώντας μία επεξεργασία στον επεξεργαστή για κάθε αίτηση. Βασικό χαρακτηριστικό του πρωτοκόλλου επικοινωνίας telnet είναι ότι υποστηρίζει την μεταδώσει δεδομένα μέσω πακέτων πρωτοκόλλου TCP/IP.

2.2 Λειτουργία της κεντρικής μονάδας ελέγχου

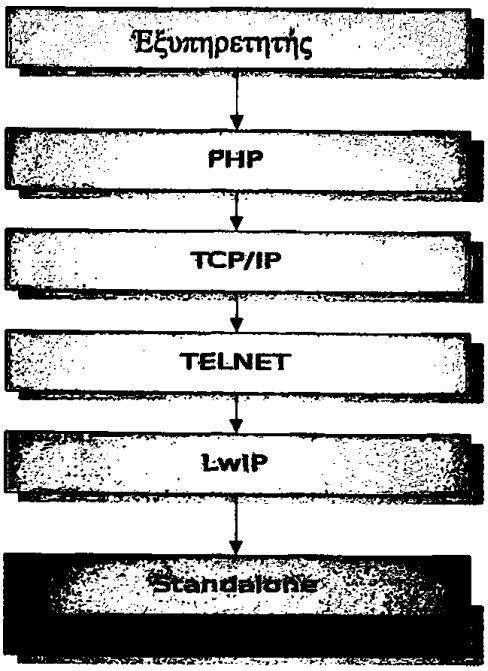
Η επικοινωνία μεταξύ του επεξεργαστή και της ιστοσελίδας διαχείρισης η οποία εκτελε
ον εξυπηρετητή κάποιου υπολογιστή ήταν το πιο χρονοβόρο κομμάτι της υλοποίησης
στήματος. Το πρωτόκολλο TELNET επελέγη για την επικοινωνία αυτή. Η ιστοσελίδα διαχείρι
νδέεται, μέσω του πρωτοκόλλου, με τον επεξεργαστή και του αποστέλλει εντολές (εικόνα 12)
ήστη. Ο επεξεργαστής λαμβάνει τις εντολές και εκτελεί τις λειτουργίες που του ζητήθηκαν.
τολές που υποστηρίζονται, καθώς και η λειτουργία τους φαίνονται στον πίνακα 2.



Εικόνα 12: Σύνδεση με την κεντρική μονάδα μέσω telnet και εκτέλεση εντολών.

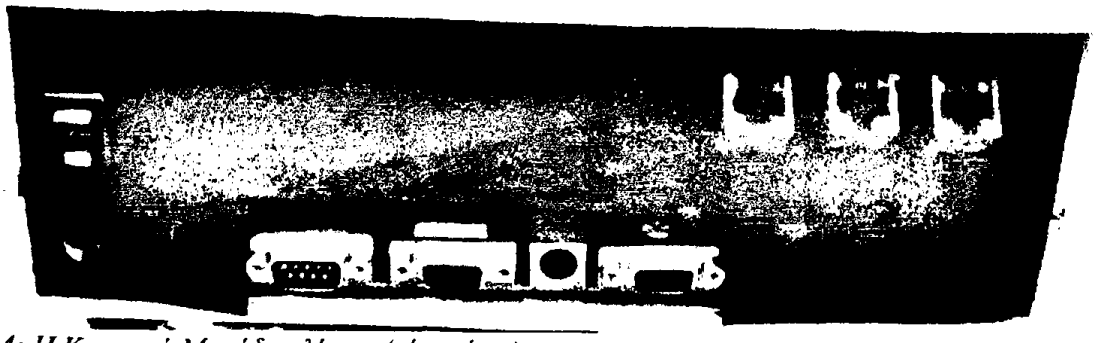
Εντολή	Λειτουργία
status.<αριθμός υπολογιστή>	Ενημέρωση για την κατάσταση της τροφοδοσίας και της λειτουργίας του υπολογιστή με βάση τον αριθμό του
closepc.<αριθμός υπολογιστή>	Απενεργοποίηση του υπολογιστή με βάση τον αριθμό του
openpc.<αριθμός υπολογιστή>	Ενεργοποίηση του υπολογιστή με βάση τον αριθμό του
wakeall	Ενεργοποίηση όλων των υπολογιστών
killall	Απενεργοποίηση όλων των υπολογιστών
help	Βοήθεια
exit	Έξοδος

Πίνακας 2: Εντολές Κεντρικής Μονάδας Ελέγχου



Εικόνα 13: Επικοινωνία Ιστοσελίδας με τον Επεξεργαστή

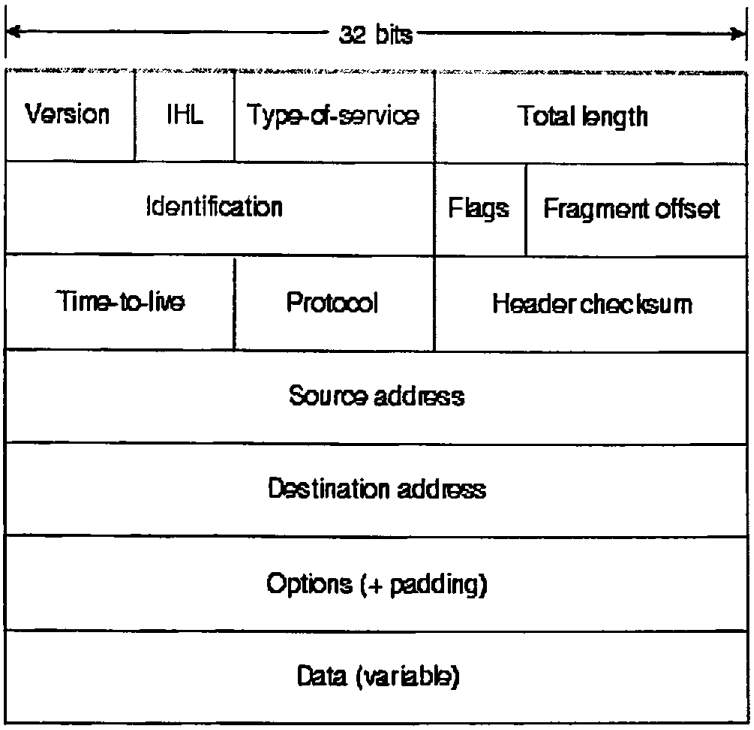
Στην εικόνα 13 βλέπουμε τη διαδικασία επικοινωνίας μεταξύ του επεξεργαστή της κεντρικής μονάδας και της ιστοσελίδας διαχείρισης. Ο διαχειριστής του συστήματος αποστέλλει μια εντολή από το γραφικό περιβάλλον της ιστοσελίδας διαχείρισης προς το σύστημα. Η εφαρμογή telnet η οποία έχει υλοποιηθεί με την χρήση της γλώσσας προγραμματισμού PHP συνδέεται μέσω του πρωτοκόλλου TELNET με τον επεξεργαστή της κεντρικής μονάδας ελέγχου. Λαμβάνει τα δεδομένα από τον χρήστη τα ενσωματώνει σε ένα πακέτο της μορφής TCP/IP (εικόνα 15) και τα αποστέλλεται στην κεντρική μονάδα ελέγχου. Η εφαρμογή telnet περιμένει την απόκριση από την αποστολή του πακέτου. Ο επεξεργαστής διαβάζει το πακέτο, εκτελεί την εντολή, ενσωματώνει την έξοδο της εντολής σε ένα πακέτο της μορφής TCP/IP, μέσω της εφαρμογής LwIP (παρ 2.1.4), και το αποστέλλει στην ιστοσελίδα διαχείρισης. Η εφαρμογή Telnet διαβάζει το πακέτο, κλείνει τη σύνδεση με τον επεξεργαστή και προβάλλει την έξοδο της εντολής στο γραφικό περιβάλλον της ιστοσελίδας διαχείρισης.



Εικόνα 14: Η Κεντρική Μονάδα ελέγχου(πίσω όψη)

Αφού λάβει την εντολή από την ιστοσελίδα διαχείρισης ο επεξεργαστής microblaze της κεντρικής μονάδας ελέγχου (εικόνα 14) αποστέλλει τον ζητούμενο αριθμό της συσκευής στο δίκτυο όπου είναι συνδεδεμένα τα κυκλώματα διευθυνσιοδότησης των υπο έλεγχο Η/Υ. Ο αριθμός έχει μέγεθος 4-bit και μεταδίδεται στο δίκτυο μέσω μιας συνάρτησης, η οποία επικοινωνεί με τους εξωτερικούς ακροδέκτες (βλ. Κεφάλαιο 2, ενότητα 2.2.1). Οι εξωτερικοί ακροδέκτες συνδέονται απευθείας με τους ακροδέκτες των συγκριτών όλων των κυκλωμάτων διευθυνσιοδότησης. Ο συγκριτής του κυκλώματος διευθυνσιοδότησης ενεργοποιεί το κύκλωμα. Το κύκλωμα λαμβάνει την κατάσταση της τροφοδοσίας και της λειτουργίας του υπολογιστή και την μεταδίδει στην κεντρική μονάδα ελέγχου.

Ο επεξεργαστής microblaze της κεντρικής μονάδας ελέγχου (PSUM) λαμβάνει τα δεδομένα από το κύκλωμα διευθυνσιοδότησης. Όταν κάποιο κύκλωμα διευθυνσιοδότησης ενεργοποιηθεί, αποστέλλει δεδομένα με την μορφή αριθμού 2-bit. Ο επεξεργαστής λαμβάνει τον αριθμό 2-bit και τον μεταφράζει. Το πρώτο bit του αριθμού είναι η κατάσταση της τροφοδοσίας του υπολογιστή, ενώ το δεύτερο bit είναι η κατάσταση της λειτουργίας του υπολογιστή. Η κεντρική μονάδα ενημερώνει την ιστοσελίδα διαχείρισης για την κατάσταση του υπολογιστή.



Εικόνα 15: Δομή πακέτου TCP/IP

2.3 Υλοποίηση

2.3.1. Το λογισμικό Embedded Development Kit

Το λογισμικό Embedded Development Kit (EDK) [18] είναι ένα ολοκληρωμένο περιβάλλον εργασίας για τον σχεδιασμό και την ενσωμάτωση συστημάτων επεξεργασίας σε FPGA. Το λογισμικό EDK έχει' εγκατεστημένες τις βιβλιοθήκες δημιουργίας και ενσωμάτωσης του soft-core επεξεργαστή Microblaze. Επίσης, διαθέτει μια πληθώρα βιβλιοθηκών για τη σύνδεση του επεξεργαστή με τις περιφερειακές συσκευές που προσφέρονται. Τέλος, περιλαμβάνει το Software development Kit για τη δημιουργία του λογισμικού διαχείρισης του επεξεργαστή.

Το Embedded Development Kit συνοπτικά περιλαμβάνει:

- Το Xilinx Platform Studio (XPS) Tool Suite, που περιλαμβάνει: το γραφικό περιβάλλον και τη γραμμή εντολών για το σχεδιασμό των ενσωματωμένων συστημάτων, τον οδηγό γρήγορης ρύθμισης υλικού για τα FPGA της εταιρίας Xilinx.inc και τέλος, το XPS περιλαμβάνει και άλλους έξυπνους οδηγούς που βοηθούν στις ρυθμίσεις των παραμέτρων της αρχιτεκτονικής του επεξεργαστή, της μνήμης και των περιφερειακών .

- Το λογισμικό Software Development Kit (SDK) για τον επεξεργαστή MicroBlaze περιλαμβάνει: τον GNU C/C++ compiler and debugger για τη δόμηση και αποσφαλμάτωση των προγραμμάτων, την Xilinx Microprocessor Debug (XMD) target server για την αποσφαλμάτωση απευθείας στο FPGA και το πρωτόκολλο Data2MEM για την επικοινωνία με τον επεξεργαστή και την ενσωμάτωση του λογισμικού.

- Το Real-Time Operating System and Embedded OS Support περιλαμβάνει την υποστήριξη για συστήματα σχεδιασμένα σε FPGA άλλων κατασκευαστών, συνεργαζόμενων με την εταιρία Xilinx.inc, όπως οι Wind River, Green Hills, Mentor, LynuxWorks.

- Το Processing IP and MicroBlaze Soft Processor Core περιλαμβάνει τις απαραίτητες βιβλιοθήκες για την ενσωμάτωση του επεξεργαστή Microblaze και τους οδηγούς διασύνδεσης του επεξεργαστή με ποικιλία περιφερειακών συσκευών για περισσότερη ευελιξία στη σχεδίαση συστημάτων.

2.3.2. *Υλοποίηση του επεξεργαστή Microblaze*

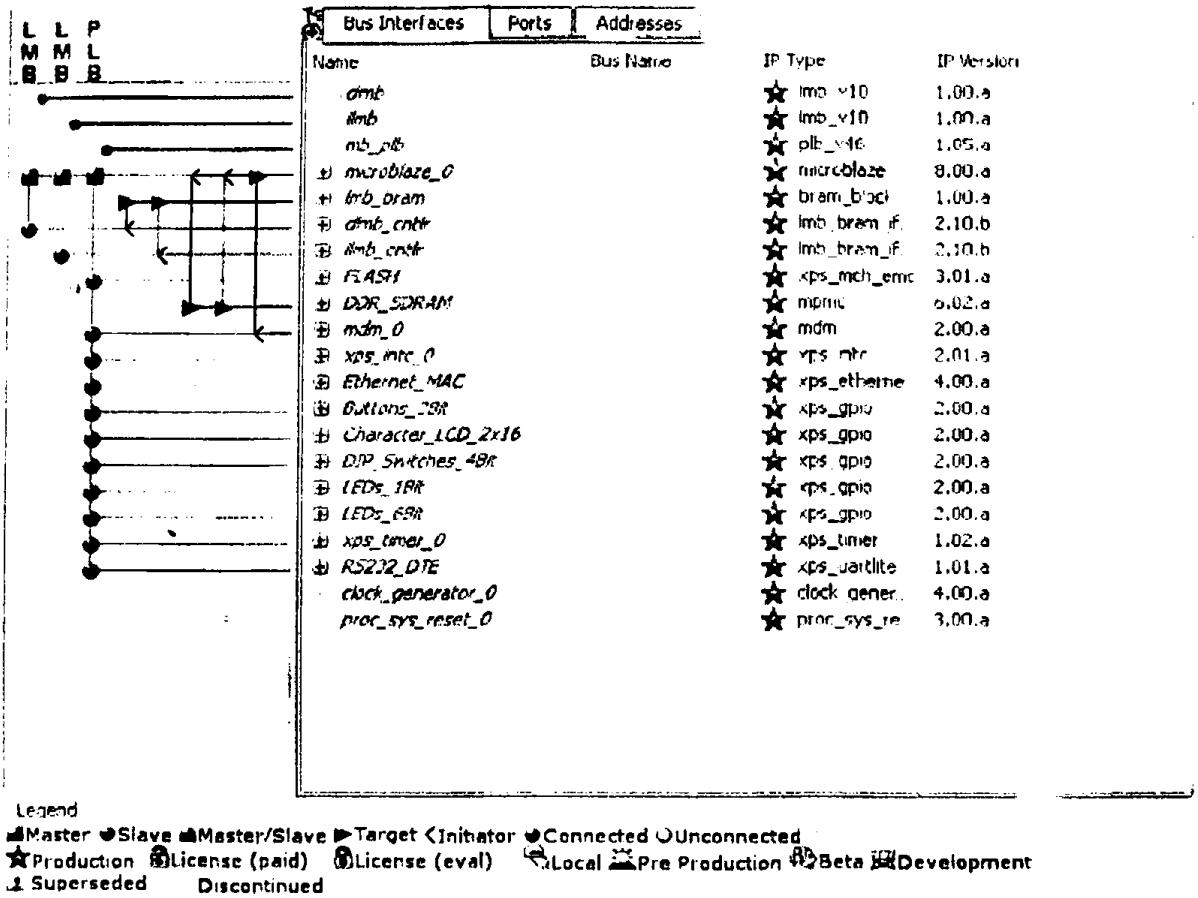
Η υλοποίηση του επεξεργαστή Microblaze ξεκινάει με τον σχεδιασμό της δομής του επεξεργαστή. Χρησιμοποιώντας τον οδηγό υλοποίησης που παρέχεται από το EDK, επιλέγουμε τα χαρακτηριστικά του επεξεργαστή που χρειαζόμαστε για την υλοποίηση του συστήματος.

Στον οδηγό υπάρχει η δυνατότητα υλοποίησης και δεύτερου επεξεργαστή στο ίδιο FPGA. Η επιλογή αυτή δεν χρησιμοποιείται σε αυτό το σύστημα. Λόγω της αρχιτεκτονικής του επεξεργαστή θα υλοποιήσουμε στο FPGA δύο block μνήμης, την instruction-cache και την data-cache. Οι μνήμες υλοποιούνται στο ίδιο το FPGA και είναι ενσωματωμένες στον επεξεργαστή.

Ο χρονισμός του επεξεργαστή γίνεται από το ενσωματωμένο στην πλατφόρμα υλοποίησης ρολόι των 50 MHZ. Ο χρονισμός των περιφερειακών συσκευών είναι επίσης 50 MHZ. Χρησιμοποιήθηκαν δύο timers, ένας για την διαχείριση των εντολών του δικτύου και ένας για τη διαχείριση των εντολών του επεξεργαστή.

Η πλατφόρμα υλοποίησης παρέχει μεγάλη ποικιλία περιφερειακών. Για την υλοποίηση του συστήματος χρησιμοποιήθηκαν: ο Ethernet Controller για την επικοινωνία του επεξεργαστή με το δίκτυο, ο Memory Controller για την επικοινωνία του επεξεργαστή με την εξωτερική μνήμη Flash και ο Gpio Controller για την επικοινωνία του επεξεργαστή με την οθόνη LCD και τους εξωτερικούς ακροδέκτες I/O J-pin. Ένας πλήρης οδηγός βήμα προς βήμα παρέχεται στο παράρτημα II.B. Ο οδηγός περιγράφει τη χρήση του οδηγού υλοποίησης που παρέχεται από το EDK.

Στην εικόνα 16 φαίνεται ο επεξεργαστής με τις διασυνδέσεις της εσωτερικής μνήμης, της εξωτερικής μνήμης και των περιφερειακών. Για τη σύνδεση της ενσωματωμένης μνήμης χρησιμοποιείται το LMB bus, που στην εικόνα φαίνεται με μπλε χρώμα. Η σύνδεση της εξωτερικής μνήμης και των περιφερειακών γίνεται μέσω του PLB bus που στη εικόνα φαίνεται με κίτρινο χρώμα.



Εικόνα 16: Διασυνδέσεις του Επεξεργαστή Microblaze

Το LMB (Local Memory Bus) [19] είναι η υλοποίηση ενός local memory bus χρησιμοποιώντας τα λογικά blocks του FPGA. Αποτελεί τη διασύνδεση της εσωτερικής μνήμης, που υλοποιήθηκε στο FPGA, με τον επεξεργαστή Microblaze. Το LMB χρησιμοποιεί ελάχιστους πόρους του FPGA, έχει ξεχωριστά read/write busses και είναι ένας αποτελεσματικός τρόπος διασύνδεσης της εσωτερικής μνήμης απευθείας με τον επεξεργαστή, χωρίς την παρουσία ελεγκτή μνήμης.

Το PLB [20] (Processor Local Bus) είναι η υλοποίηση μιας εξωτερικής 128-bit διασύνδεσης για την επικοινωνία του επεξεργαστή με τις περιφερειακές συσκευές. Περιλαμβάνει ένα bus control unit, έναν watchdog timer, ξεχωριστά read/write data paths και ξεχωριστό data path για τις διευθύνσεις. Αναλυτικά, τα χαρακτηριστικά του είναι:

- Arbitration support for a configurable number of
- PLB master devices

- PLB address and data steering support for all masters
- 128-bit, 64-bit, and 32-bit support for masters and slaves
- PLB address pipelining (supported in shared bus mode or point-to-point configuration)
- Three-cycle arbitration
- Four levels of dynamic master request priority
- Selectable round robin or fixed priority arbitration
- Configurable optimization for point-to-point topology
- PLB watchdog timer
- PLB architecture compatible
- Complete PLB bus structure provided
- Supports a configurable number of slave devices
- No external OR gates required for PLB slave input signals
- PLB Reset circuit
- PLB Reset generated synchronously to the PLB clock
- PLB Reset generated synchronously from external reset when external reset provided
- Provides vectorized reset signal to reduce system fanout for improved timing
- Active state of external reset selectable via a design parameter

2.3.3. *Υλοποίηση στο λογισμικό SDK*

Το λογισμικό Software Development Kit (SDK) [21] για τον επεξεργαστή MicroBlaze περιλαμβάνει: τον GNU C/C++ compiler version 4.5.2 and debugger για την δόμηση και αποσφαλμάτωση των προγραμμάτων, την Xilinx Microprocessor Debug (XMD) target server για την αποσφαλμάτωση απευθείας από το FPGA και το πρωτόκολλο Data2MEM για την επικοινωνία του επεξεργαστή με το λογισμικό SDK και την ενσωμάτωση του λειτουργικού σε αυτόν. Χρησιμοποιώντας το SDK αναπτύχθηκαν τρεις συναρτήσεις (η getstatus, η closepc και η openpc) για τον έλεγχο των τροφοδοτικών.

2.3.4. *Συνάρτηση getstatus*

Η συνάρτηση που ελέγχει την κατάσταση των υπολογιστών ονομάζεται **getstatus**. Η συνάρτηση **getstatus** λαμβάνει σαν όρισμα έναν ακέραιο αριθμό, ο οποίος αντιστοιχεί στον αριθμό του υπολογιστή (εν γένει) του οποίο θέλουμε να ελέγξουμε την κατάσταση. Αποτελείται από δύο υποσυναρτήσεις την **Xgpio_DiscreteWrite** και την **XGpio_DiscreteRead**, οι οποίες παρέχονται από τις βιβλιοθήκες του SDK και διαχειρίζονται τα ενσωματωμένα J-pins της πλατφόρμας υλοποίησης.

Η **Xgpio_DiscreteWrite** δέχεται σαν όρισμα ένα δεκαεξαδικό αριθμό ο οποίος αντιστοιχεί στη διεύθυνση του περιφερειακού που αναφερόμαστε και ενός δεκαεξαδικού, η τιμή του οποίου θα εμφανιστεί στα τέσσερα εξωτερικά pins της πλατφόρμας. Έτσι, στέλνει το 4-bit ID, βάσει του ακεραίου που λαμβάνει από την **getstatus**, στα κυκλώματα διευθυνσιοδότησης που είναι συνδεδεμένα στο δίκτυο.

Η **XGpio_DiscreteRead** λαμβάνει δεδομένα με τη μορφή δυαδικού αριθμού από τα εξωτερικά pins και τα μετατρέπει σε μορφή δεκαεξαδικού αριθμού. Το δεύτερο όρισμα της συνάρτησης επιλέγει το περιφερειακό από το οποίο θα ληφθούν τα δεδομένα. Έτσι, λαμβάνει τον 2-bit αριθμό από το κύκλωμα διευθυνσιοδότησης και τον αποθηκεύει.

Τέλος, η **getstatus** αναλύει τον 2-bit αριθμό και αποστέλλει την κατάσταση του υπολογιστή στη συνάρτηση απόκρισης πακέτου, η οποία με τη σειρά της ενσωματώνει τα δεδομένα σε πακέτο της μορφής TCP/IP και τα αποστέλλει μέσω του πρωτοκόλλου telnet στην ιστοσελίδα διαχείρισης.

2.3.5. *Συνάρτηση closepc*

Η συνάρτηση που απενεργοποιεί τους υπολογιστές ονομάζεται **closepc**. Η **closepc** λαμβάνει σαν όρισμα έναν ακέραιο αριθμό, ο οποίος αντιστοιχεί στον αριθμό του υπολογιστή (εν γένει) που θέλουμε να απενεργοποιήσουμε. Η **closepc** αρχικά καλεί την **getstatus** για να "μάθει" την κατάσταση του υπολογιστή. Αν ο υπολογιστής δεν είναι ενεργοποιημένος, τότε η συνάρτηση **closepc** αγνοεί την εντολή απενεργοποίησης και ειδοποιεί τον χρήστη αναλόγως για την κατάσταση του υπολογιστή. Στην αντίθετη περίπτωση, εκτελεί την εντολή στέλνοντας μέσω της συνάρτησης **XGpio_DiscreteWrite** τον 6-bit αριθμό στα κυκλώματα διευθυνσιοδότησης που είναι συνδεδεμένα με το σύστημα. Για παράδειγμα, αν θέλουμε να απενεργοποιήσουμε τον υπολογιστή με αριθμό 5, ο οποίος είναι ενεργοποιημένος, τότε η συνάρτηση **XGpio_DiscreteWrite** θα στείλει στα κυκλώματα τον δυαδικό αριθμό 101001 για τρία δευτερόλεπτα για την απενεργοποίηση του υπολογιστή με τον αριθμό 5.

2.3.6. *Συνάρτηση openpc*

Η συνάρτηση που ενεργοποιεί τους υπολογιστές ονομάζεται **openpc**. Η **openpc** λαμβάνει σαν όρισμα έναν ακέραιο αριθμό ο οποίος αντιστοιχεί στον αριθμό του υπολογιστή που θέλουμε να ανοίξουμε. Η **openpc** αρχικά καλεί την **getstatus** και ενημερώνεται για την κατάσταση της λειτουργίας του υπολογιστή. Αν ο υπολογιστής δεν είναι απενεργοποιημένος, τότε η συνάρτηση **openpc** αγνοεί την εντολή ενεργοποίησης και ειδοποιεί τον χρήστη ότι ο υπολογιστής είναι ήδη ενεργοποιημένος. Αν ο υπολογιστής είναι απενεργοποιημένος και έχει τροφοδοσία, τότε η **openpc** καλεί την **Xgpio_DiscreteWrite,** η οποία αναλαμβάνει να στείλει τον 4-bit αριθμό και το σήμα απενεργοποίησης στο δίκτυο του συστήματος. Για παράδειγμα, περιγράφοντας τη λειτουργία της **closepc** είχαμε απενεργοποιήσει τον υπολογιστή με αριθμό 5. Ο αριθμός 5 σε δεκαεξαδική μορφή είναι 05, ενώ σε δυαδική μορφή είναι 1010. Αν θέλουμε να τον ενεργοποιήσουμε και πάλι, τότε η συνάρτηση **Xgpio_DiscreteWrite** θα έχει το όρισμα 05 και θα στείλει στο δίκτυο των υπολογιστών του συστήματος τον αριθμό 101001 για ένα δευτερόλεπτο.

Κεφάλαιο 3

Κύκλωμα Διευθυνσιοδότησης

Στο κεφάλαιο αυτό περιγράφονται αρχικά τα ολοκληρωμένα που χρησιμοποιήθηκαν κατά την υλοποίηση του κυκλώματος διευθυνσιοδότησης. Στη συνέχεια, παρουσιάζεται αναλυτικά ο σχεδιασμός και η φιλοσοφία στην οποία κινείται το κύκλωμα. Τέλος περιγράφεται η υλοποίηση της προσομοίωσης του κυκλώματος στο πρόγραμμα Altium Designer Summer Edition 2010 [22].

3.1 Λειτουργία κυκλώματος διευθυνσιοδότησης

Σε κάθε συσκευή που πρόκειται να συνδεθεί στο δίκτυο του συστήματος ελέγχου και διαχείρισης εγκαθίσταται ένα κύκλωμα διευθυνσιοδότησης (εικόνα 17), το οποίο περιέχει έναν συγκριτή 8-bit και έναν διακόπτη 4-bit. Ο διακόπτης κάθε κυκλώματος δίνει τον μοναδικό 4-bit αριθμό της συσκευής στο δίκτυο. Η ρύθμιση γίνεται από τον διαχειριστή του συστήματος ανάλογα με τη δομή διευθυνσιοδότησης που θα επιλέξει. Όταν τα bit που λαμβάνει ο συγκριτής από το δίκτυο είναι ίδια με τα bit που λαμβάνει από τον διακόπτη του κυκλώματος, τότε ενεργοποιείται το επιλεγμένο κύκλωμα.

Ο συγκριτής ελέγχει την ενεργοποίηση και απενεργοποίηση όλου του κυκλώματος. Μόνο όταν τα 4-bit του συγκριτή $A_0A_1A_2A_3$ που συνδέονται με το δίκτυο του συστήματος είναι ίδια με τα 4-bit του εξωτερικού διακόπτη $B_0B_1B_2B_3$, η έξοδος του συγκριτή γίνεται '0'. Όταν η έξοδος του συγκριτή είναι σε κατάσταση λογικού '0', το κύκλωμα της συσκευής ενεργοποιείται και η κεντρική μονάδα ελέγχει το κύκλωμα. Στην αντίθετη περίπτωση, η έξοδος του συγκριτή είναι μόνιμα '1' και το κύκλωμα είναι απενεργοποιημένο. Οι διασυνδέσεις του συγκριτή φαίνονται στο Παράρτημα ΙΙ.Γ.

Όταν ο διαχειριστής επιλέξει ένα κύκλωμα, τότε αυτό ενεργοποιείται. Όταν το κύκλωμα ενεργοποιείται, τότε είναι το μοναδικό που στέλνει δεδομένα στην κεντρική μονάδα ελέγχου. Μια πύλη AND διαβάζει το σήμα για την κατάσταση της τροφοδοσίας από το μοβ καλώδιο του τροφοδοτικού, ενώ μια άλλη πύλη AND διαβάζει το σήμα για την κατάσταση της λειτουργίας του

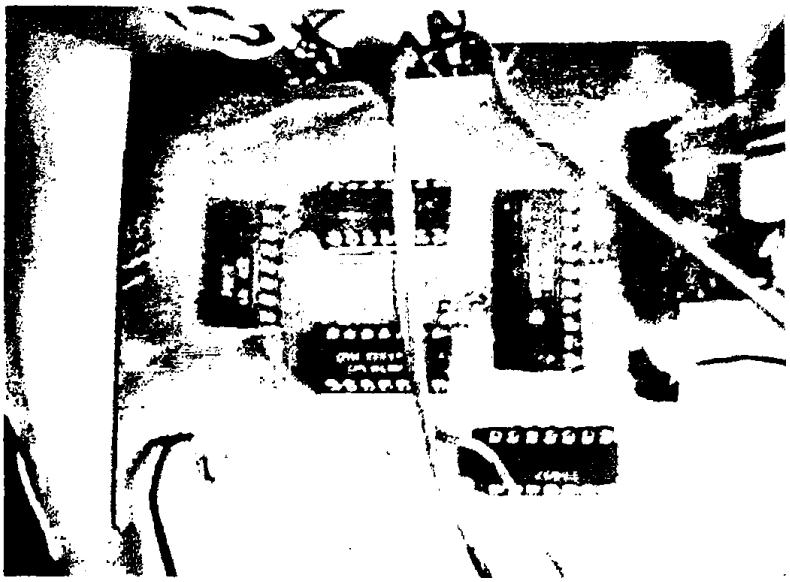
υπολογιστή (γκρι καλώδιο). Οι δύο πύλες τροφοδοτούνται από την πύλη NAND μόνο όταν η έξοδος του συγκριτή είναι '0'. Όταν ενεργοποιηθούν, στέλνουν την πληροφορία στους ακροδέκτες της πλατφόρμας υλοποίησης. Τα δεδομένα έχουν μέγεθος 2-bit. Το πρώτο bit δείχνει την κατάσταση της τροφοδοσίας του υπολογιστή, ενώ το δεύτερο bit την κατάσταση της λειτουργίας του υπολογιστή. Η σημασία του 2-bit αριθμού φαίνεται στον πίνακα 3.

Μορφή 2-bit αριθμού	**Σημασία**
00	Χωρίς Τροφοδοσία (διακοπή ρεύματος)
01	Μόνο Τροφοδοσία
10	Απαγορευμένη κατάσταση
11	Σε λειτουργία

Πίνακας 3: Πίνακας Αριθμού 2-bit.

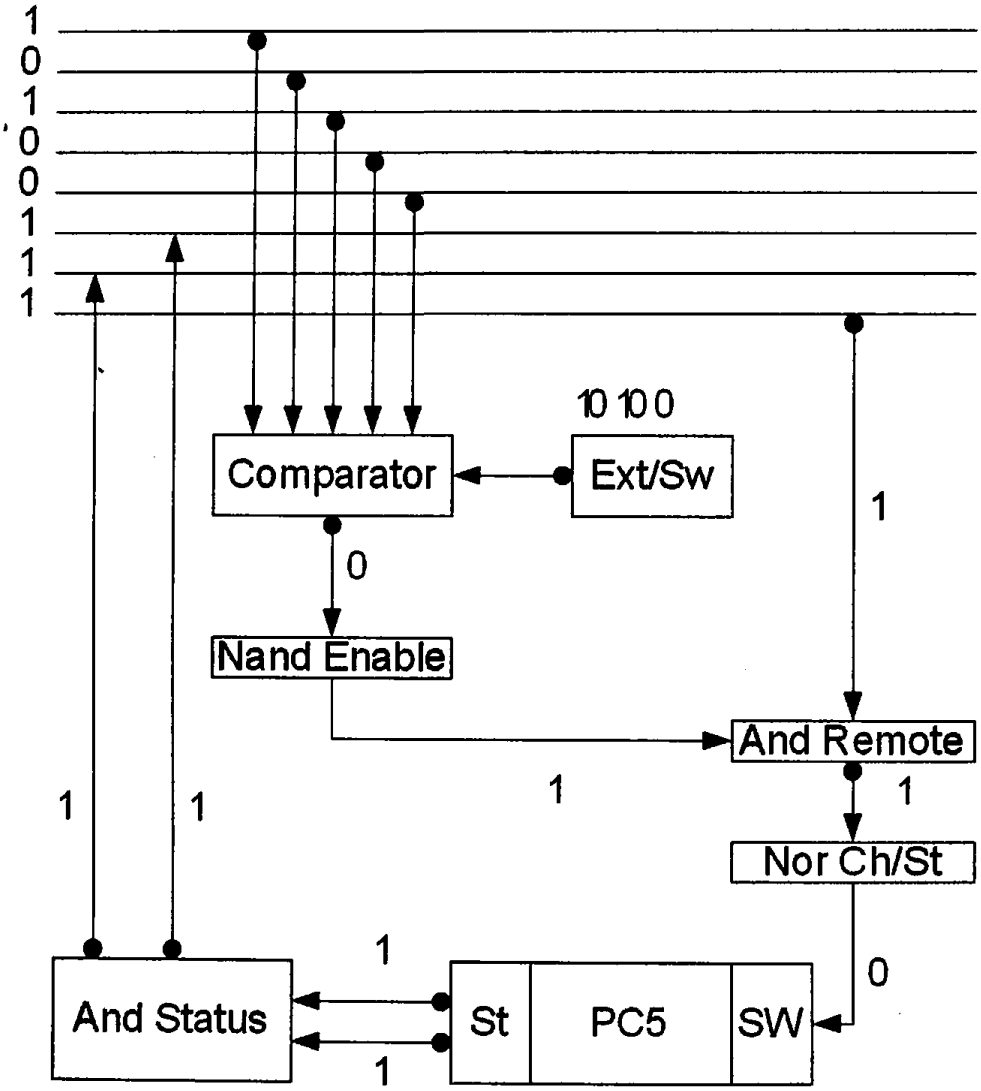
Για την ενεργοποίηση ή την απενεργοποίηση του υπολογιστή θα πρέπει η έξοδος του συγκριτή να είναι σε κατάσταση λογικού '0' και ο ακροδέκτης της πλατφόρμας υλοποίησης που στέλνει το σήμα διαχείρισης να είναι σε κατάσταση λογικού '1'.

Όλα τα κυκλώματα διευθυνσιοδότησης συνδέονται στο ίδιο καλώδιο. Όταν η κεντρική μονάδα αποστέλλει στο καλώδιο τον 4-bit αριθμό, τον λαμβάνουν ταυτόχρονα όλα τα κυκλώματα διευθυνσιοδότησης, ωστόσο μόνο ένα ενεργοποιείται από το συγκεκριμένο 4-bit αριθμό. Μόνο όταν ενεργοποιηθεί ένα κύκλωμα διευθυνσιοδότησης από την κεντρική μονάδα, στέλνει δεδομένα στο καλώδιο. Το σήμα διαχείρισης φτάνει σε όλα τα κυκλώματα διευθυνσιοδότησης (εικόνα 18). Μόνο το



Εικόνα 17: Κύκλωμα Διευθυνσιοδότησης σε λειτουργία

κύκλωμα που είναι ενεργοποιημένο δεν αγνοεί το σήμα διαχείρισης.

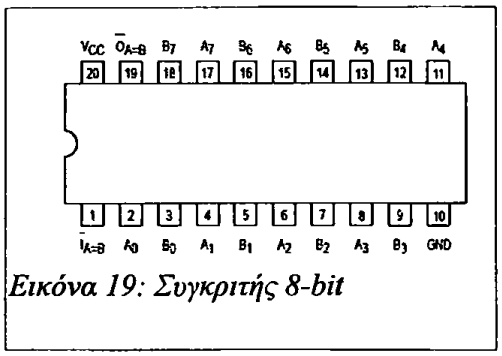


Εικόνα 18: Λειτουργία Κυκλώματος διευθυνσιοδότησης

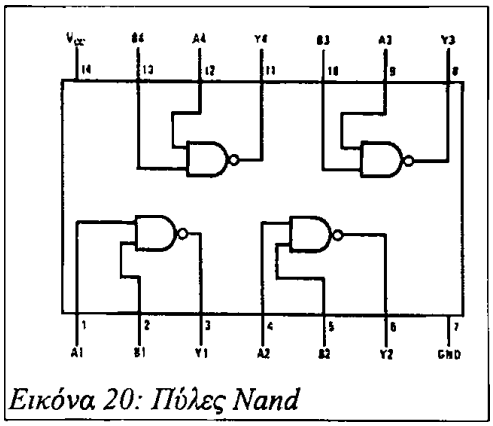
3.2 Υλοποίηση

3.2.1. Λειτουργία ολοκληρωμένων κυκλώματος

Το βασικό ολοκληρωμένο για τη λειτουργία του κυκλώματος διευθυνσιοδότησης είναι ο συγκριτής, 8-bit. Από το ολοκληρωμένο θα χρησιμοποιήσουμε τους ακροδέκτες 1-10,19 και 20. Ο συγκριτής αναλαμβάνει τη διευθυνσιοδότηση του κυκλώματος και την ενεργοποίηση του. Όταν τα 4-bit που αποστέλλονται από την κεντρική μονάδα είναι ίδια με τα 4-bit του διακόπτη του κυκλώματος, η έξοδος του συγκριτή, δηλαδή ο ακροδέκτης 19, είναι σε κατάσταση λογικού '0'.



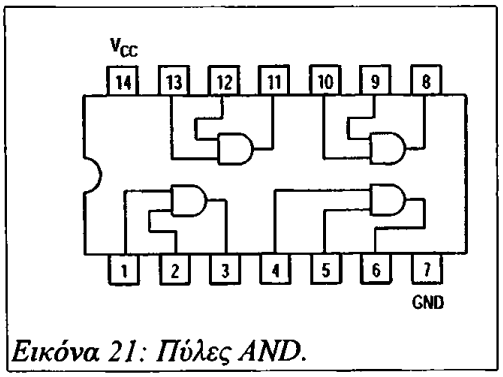
Εικόνα 19: Συγκριτής 8-bit



Εικόνα 20: Πύλες Nand

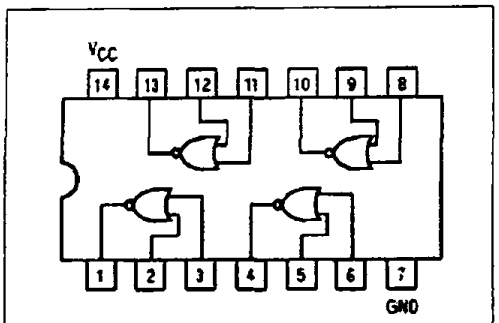
Από το ολοκληρωμένο πυλών NAND χρησιμοποιούμε 2 πύλες NAND. Οι ακροδέκτες που χρησιμοποιήθηκαν είναι οι 1, 2, 3, 4, 5, 6 . Η μία πύλη χρησιμοποιείται για την αντιστροφή του σήματος του συγκριτή και για την τροφοδοσία της πύλης AND Status. Όταν η έξοδος του συγκριτή είναι '0', τότε η έξοδος της πύλης NAND είναι σε κατάσταση λογικού '1'. Τέλος, η δεύτερη πύλη NAND χρησιμοποιείται για την αντιστροφή του σήματος από το κουμπί εκκίνησης του υπολογιστή.

Η *AND Status* είναι ένα από τα δύο ολοκληρωμένα πυλών AND που χρησιμοποιούνται από το κύκλωμα. Η AND status είναι η πύλη μετάδοσης δεδομένων. Η πύλη AND status τροφοδοτείται από την έξοδο της πύλης NAND. Χρησιμοποιούμε δύο από τις τέσσερις πύλες του ολοκληρωμένου. Όταν η AND Status ενεργοποιηθεί, μεταδίδει τα δεδομένα στην κεντρική μονάδα ελέγχου . Το δεύτερο ολοκληρωμένο είναι η πύλη AND Remote. Από το ολοκληρωμένο AND Remote χρησιμοποιούμε μία πύλη για να εξασφαλίσουμε ότι η κατάσταση του υπολογιστή θα αλλάξει, μόνο



Εικόνα 21: Πύλες AND.

όταν έχει ενεργοποιηθεί το κύκλωμα και το bit διαχείρισης είναι '1'.



Εικόνα 22: Πύλες NOR.

Το ολοκληρωμένο πυλών NOR αναλαμβάνει να στείλει τον παλμό αλλαγής κατάστασης στο κύκλωμα ελέγχου της μητρικής πλακέτας του υπολογιστή. Από το ολοκληρωμένο χρησιμοποιείται μία πύλη NOR. Για να αλλάξει κατάσταση ο υπολογιστής θα πρέπει: είτε να πατηθεί το κουμπί εκκίνησης του υπολογιστή, ή να έρθει η εντολή αλλαγής κατάστασης από την κεντρική μονάδα ελέγχου

3.2.2. Συνολικό κύκλωμα

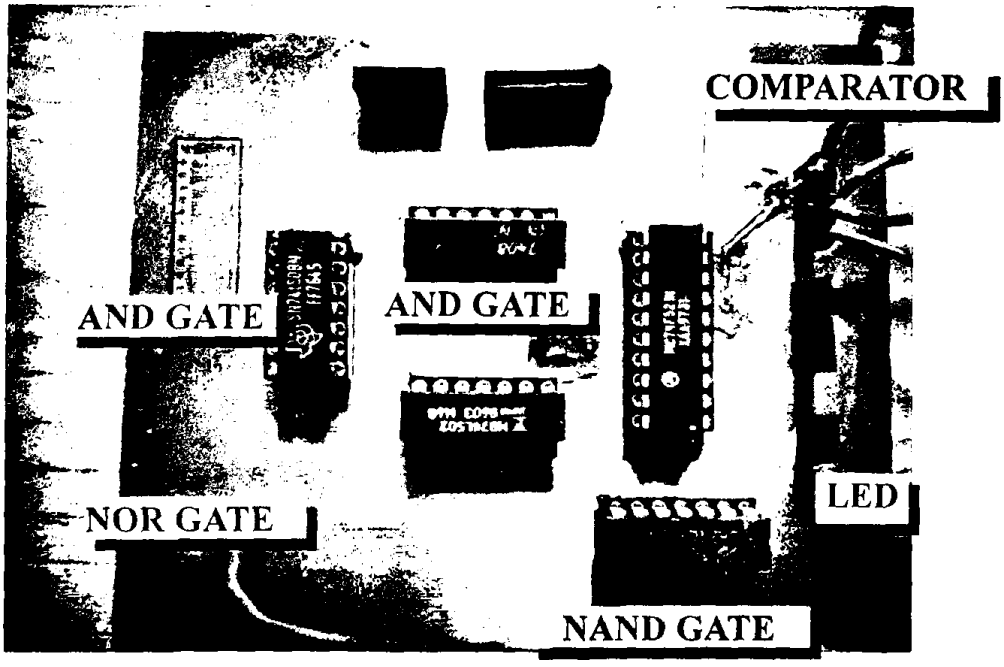
Στην εικόνα 23 βλέπουμε το τελικό κύκλωμα τυπωμένο στην εικόνα 24 βλέπουμε τις διασυνδέσεις των ολοκληρωμένων ενώ. Οι ακροδέκτες 2,4,6,8 του συγκριτή συνδέονται μέσω του καλωδίου τύπου ethernet με τους τέσσερις εξωτερικούς ακροδέκτες της πλατφόρμας υλοποίησης. Οι ακροδέκτες 3,5,7,9 είναι συνδεδεμένοι με τον εξωτερικό 4-bit διακόπτη του κυκλώματος. Όταν οι ακροδέκτες 2,4,6 και 8 είναι ίσοι με τους ακροδέκτες 3,5,7 και 9 αντίστοιχα, τότε η έξοδος του ακροδέκτη 19 είναι σε κατάσταση λογικού '0'. Ο ακροδέκτης 1 είναι συνδεδεμένος με την γείωση, όπως και ο ακροδέκτης 10. Ο ακροδέκτης 20 είναι η τροφοδοσία του ολοκληρωμένου και είναι πάντα σε κατάσταση λογικού '1'.

Ο ακροδέκτης 19 του συγκριτή συνδέεται με τους ακροδέκτες 1και 2 της πύλης NAND, καθώς επίσης και με ένα LED. Ο ακροδέκτης 3 της πύλης NAND συνδέεται με τον ακροδέκτη 14 της πύλης AND και της παρέχει τροφοδοσία. Οι ακροδέκτες 4 και 5 συνδέονται με το κουμπί εκκίνησης του υπολογιστή, ενώ ο ακροδέκτης 6 συνδέεται με τον ακροδέκτη 2 της πύλης NOR. Ο ακροδέκτης 7 και ο

ακροδέκτης 14 είναι σε κατάσταση λογικού '0' και '1' αντίστοιχα.

Οι ακροδέκτες 1 και 2 της πύλης AND, που τροφοδοτείται από την πύλη NAND, είναι συνδεδεμένοι με το μοβ καλώδιο του τροφοδοτικού. Ο ακροδέκτης 3 συνδέεται με τον ακροδέκτη της πλατφόρμας υλοποίησης, που λαμβάνει το σήμα για την τροφοδοσία των υπολογιστών. Οι ακροδέκτες 4 και 5 συνδέονται με το γκρι καλώδιο του τροφοδοτικού και ο ακροδέκτης 6 με τον ακροδέκτη της πλατφόρμας υλοποίησης, που λαμβάνει το σήμα για τη λειτουργία των υπολογιστών. Ο ακροδέκτης 7 είναι πάντα σε κατάσταση λογικού '0'.

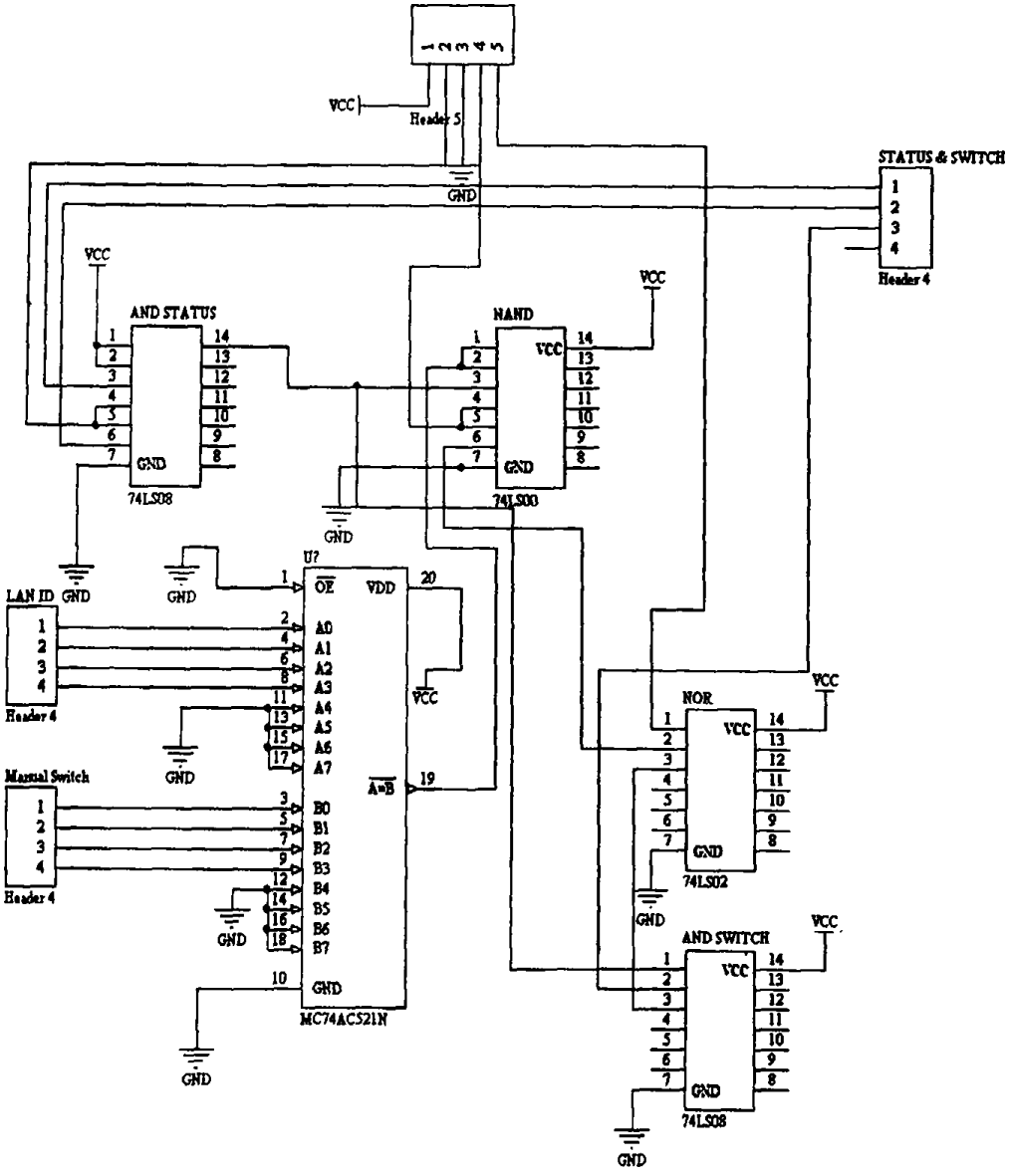
Όπως αναφέραμε στην προηγούμενη παράγραφο, ο ακροδέκτης 2 της πύλης NOR συνδέεται με τον ακροδέκτη 6 της πύλης NAND. Ο ακροδέκτης 1 συνδέεται με τη μητρική πλακέτα του υπολογιστή. Ο ακροδέκτης 3 συνδέεται με τον ακροδέκτη 3 της πύλης AND, που δεν τροφοδοτείται από την NAND, αλλά έχει σταθερή τροφοδοσία. Οι ακροδέκτες 14 και 7 είναι σε κατάσταση λογικού '1' και '0' αντίστοιχα.



Εικόνα 23: Διάταξη ολοκληρωμένων στο κύκλωμα διευθυνσιοδότησης

Ο ακροδέκτης 1 της πύλης AND με σταθερή τροφοδοσία συνδέεται με τον ακροδέκτη 3 της πύλης NAND και ο ακροδέκτης 2 με τον ακροδέκτη της πλατφόρμας υλοποίησης, που στέλνει το

σήμα ενεργοποίησης ή απενεργοποίησης του υπολογιστή. Οι ακροδέκτες 14 και 7 είναι σε κατάσταση λογικού '1' και '0' αντίστοιχα.



*Εικόνα 24: Το κύκλωμα Διευθυνσιοδότησης σχεδιασμένο στο λογισμικό Altium Designer Summer Edition 2010**

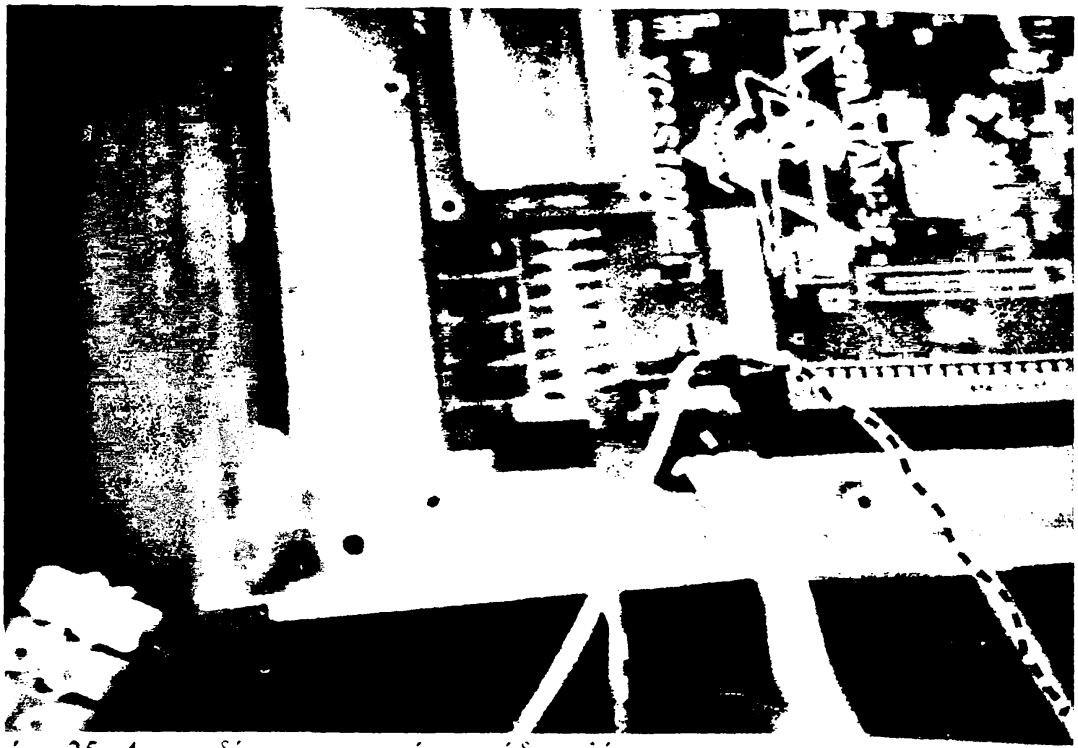
Η διασύνδεση με την ετικέτα PC αποτελείται από 5 pins. Στο pin 1 συνδέετε το μοβ καλώδιο του τροφοδοτικού, στο pin 2 συνδέετε το γκρι καλώδιο του τροφοδοτικού ενώ στο pin 3 συνδέεται η

γειωση του τροφοδοτικού. Στο pin 3 συνδέεται ο διακόπτης ενεργοποίησης του υπολογισ
στο pin 4 συνδέεται η ο διακόπτης της μητρικής πλακέτας του υπολογιστή.

Η διασύνδεση με την ετικέτα STATUS & SWITCH αποτελείται από 4 pins
συνδέεται το pin της κεντρικής μονάδας για την κατάσταση της τροφοδοσίας (εικόνα 2
χρώμα) ενώ στο pin 2 συνδέεται το pin της κεντρικής μονάδας για την κατάσταση της λει
ηλεκτρονικού υπολογιστή (εικόνα 25 με μπλε χρώμα). Στο pin 3 συνδέεται το pin τ
μονάδας που στέλνει της εντολή ενεργοποίησης η απενεργοποίησης του υπολογιστή (ε
κόκκινο χρώμα). Το pin 4 είναι μια βοηθητική γείωση.

Η διασύνδεση με την ετικέτα LAN ID αποτελείται από 4 pin, τα οποία συνδέοντα
της κεντρικής μονάδας ελέγχου που στέλνουν τον αριθμό του υπολογιστή με την μορφή 4
25 με πράσινο χρώμα). Το pin μ ε τον αριθμό 1 είναι το LSB ενώ το pin με τον αριθμό 4 είν

Η διασύνδεση με την ετικέτα MANUAL SWITCH αποτελείται από 4 pins τα οποία
με τον εξωτερικό διακόπτη του κυκλώματος διευθυνσιοδότησης από τον οποίο ο χρήστη
τον αριθμό του υπολογιστή στο δίκτυο του συστήματος ελέγχου και διαχείρισης



Εικόνα 25: Διασυνδέσεις κεντρικής μονάδας ελέγχου

Κεφάλαιο 4

Ιστοσελίδα διαχείρισης

Το κεφάλαιο ξεκινά με την περιγραφή του εξυπηρετητή που είναι εγκατεστημένος στον υπολογιστή που διαχειρίζεται την ιστοσελίδα του συστήματος. Στη συνέχεια του κεφαλαίου περιγράφεται ο σχεδιασμός και η δομή της ιστοσελίδας. Τέλος, περιγράφεται η δομή της ιστοσελίδας διαχείρισης και η υλοποίηση με την χρήση της γλώσσας προγραμματισμού PHP.

4.1 Εργαλεία

4.1.1. Εξυπηρετητής Apache HTTP

Ο Apache HTTP [27] είναι ένας δημοφιλής εξυπηρετητής ιστού (Webserver) που αναπτύσσεται από την "Κοινότητα Ανοιχτού Λογισμικού". Η εποπτεία, υποστήριξη και διάθεση του προγράμματος γίνεται από το Apache Software Foundation [28]. Το πρόγραμμα είναι ανοικτού κώδικα (open source), κάτι που σημαίνει ότι σύμφωνα με την άδεια χρήσης του [29], διατίθεται δωρεάν και μπορούν να γίνουν ελεύθερα από το χρήστη προσθήκες και τροποποιήσεις στον κώδικα του.

Ο Apache HTTP διαθέτει ποικιλία χαρακτηριστικών και μπορεί να υποστηρίξει μια μεγάλη γκάμα εφαρμογών με τις οποίες και συνεργάζεται. Ένα από τα βασικότερα χαρακτηριστικά του όμως, το οποίο και του δίνει μεγάλες δυνατότητες, είναι ότι μπορεί να προσαρμόσει επάνω του πολλές προσθήκες προγραμμάτων (modules), τα οποία με τη σειρά τους παρέχουν διαφορετικές λειτουργίες.

Ένα άλλο χαρακτηριστικό – δυνατότητα του Apache HTTP είναι ότι μπορεί να εγκατασταθεί σε διάφορα λειτουργικά συστήματα. Ο Apache HTTP υποστηρίζει επίσης αρκετές διάσημες εφαρμογές και γλώσσες προγραμματισμού όπως MySQL, PHP, Perl, Python κ.λπ.

Αυτά είναι μερικά από τα χαρακτηριστικά και κάποιες απο τις λειτουργίες που κάνουν τον Apache HTTP τον πιο δημοφιλή Web Server από το 1996 έως τις μέρες μας. Περισσότερο από το 50% των ιστοχώρων του παγκόσμιου ιστού, χρησιμοποιεί τον Apache HTTP ως εξυπηρετητή.

4.1.2. Γλώσσα προγραμματισμού PHP

Η PHP (PHP Hypertext Preprocessor) [30] είναι μία γλώσσα προγραμματισμού για τη δημιουργία σελίδων ιστού με δυναμικό περιεχόμενο. Ο κώδικας PHP εκτελείται από τον εξυπηρετητή μόλις ζητηθεί από τον φυλλομετρητή, ώστε να παραχθεί σε πραγματικό χρόνο το τελικό περιεχόμενο.

Ένα αρχείο με κώδικα PHP θα πρέπει να έχει την κατάλληλη επέκταση (π.χ. *.php, *.php4, *.phtml κ.ά.). Σε αντίθετη περίπτωση, η ενσωμάτωση κώδικα PHP σε ένα αρχείο επέκτασης .html δε θα λειτουργήσει και θα εμφανίσει στον browser τον κώδικα χωρίς καμία επεξεργασία, εκτός αν έχει γίνει η κατάλληλη ρύθμιση στα MIME types του server. Επίσης, ακόμη κι όταν ένα αρχείο έχει την επέκταση .php, θα πρέπει ο server να είναι ρυθμισμένος για να επεξεργάζεται και να μεταγλωττίζει τον κώδικα PHP. Ο διακομιστής Apache HTTP υποστηρίζει εξ ορισμού την εκτέλεση κώδικα PHP, είτε με τη χρήση ενός πρόσθετου (mod_php) ή με την αποστολή του κώδικα προς εκτέλεση σε εξωτερική διεργασία CGI ή FCGI.

4.2 Υλοποίηση και λειτουργία ιστοσελίδας διαχείρισης

Η ιστοσελίδα διαχείρισης αποτελεί το περιβάλλον από το οποίο ο διαχειριστής του συστήματος επικοινωνεί με την κεντρική μονάδα ελέγχου. Είναι ένας εύχρηστος τρόπος από όπου ο διαχειριστής μπορεί με ευκολία να ελέγχει και να διαχειρίζεται σε πραγματικό χρόνο τη λειτουργία των υπολογιστών. Επιλέξαμε η διαχείριση να γίνεται μέσω ιστοσελίδας για την: γρήγορη και εύκολη πρόσβαση, πρόσβαση από διαφορετικές συσκευές (π.χ. PC, κινητό τηλέφωνο, tablet), πρόσβαση από οποιοδήποτε σημείο, ευκολία στην εγκατάσταση χωρίς πρόσθετα προγράμματα.

Για την δημιουργία της ιστοσελίδας χρησιμοποιήθηκε ένα πρότυπο (template). Από τον ιστοχώρο www.freeemplates.com.

Με την είσοδο στην ιστοσελίδα με χρήση του ονόματος χρήστη και του συνθηματικού (σχήμα 19) ο διαχειριστής έχει τον πλήρη έλεγχο του συστήματος. Από την καρτέλα Status, μπορεί να δει την κατάσταση της τροφοδοσίας των υπολογιστών της συστοιχίας, καθώς επίσης και την κατάσταση της λειτουργίας τους. Στην ίδια καρτέλα παρέχεται και η λειτουργία ενεργοποίησης και απενεργοποίησης των υπολογιστών. Η καρτέλα Status όπου αναφέρεται η κατάσταση του συστήματος ανανεώνεται κάθε

ένα λεπτό αυτόματα.

Ο διαχειριστής μπορεί να ενεργοποιήσει ή να απενεργοποιήσει έναν υπολογιστή δίνοντας τον αριθμό του από την καρτέλα extras. Από την ίδια καρτέλα μπορεί να ενεργοποιήσει ή να απενεργοποιήσει ταυτόχρονα όλους τους υπολογιστές, που είναι συνδεδεμένοι με το κύκλωμα διαχείρισης και ελέγχου.

Στην καρτέλα links υπάρχουν κάποιοι χρήσιμοι σύνδεσμοι σχετικά με το σύστημα, ενώ στην καρτέλα contact ο διαχειριστής μπορεί να βρει τους τρόπους επικοινωνίας με την τεχνική υποστήριξη του συστήματος.

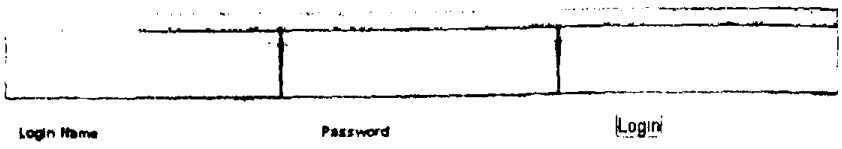
Η υλοποίηση της ιστοσελίδας διαχείρισης του συστήματος ξεκινάει από την εγκατάσταση του εξυπηρετητή Apache HTTP (www.apache.org). Στη συνέχεια, απαιτείται η εγκατάσταση των βιβλιοθηκών της γλώσσας PHP στον εξυπηρετητή (www.php.com).

Όπως αναφέραμε στην ενότητα 4.2 αυτού του κεφαλαίου, η ιστοσελίδα διαχείρισης αποτελείται από πέντε καρτέλες. Την καρτέλα εισόδου (εικόνα 26) , την καρτέλα status (εικόνα 27), την καρτέλα extras (εικόνα 28), την καρτέλα links και την καρτέλα contact.

4.2.1 Καρτέλα Εισόδου

Το αρχείο login.php (Βλ. Παράρτημα II.Β) είναι το αρχείο το οποίο διαχειρίζεται την πρόσβαση στο σύστημα και εκτελεί τις απαραίτητες ενέργειες εισόδου σε αυτό δημιουργόντας την καρτέλα πρόσβασης (εικόνα 26). Το αρχείο login.php λαμβάνει τα δεδομένα από την φόρμα χρήστη στην καρτέλα εισόδου της ιστοσελίδας διαχείρισης, τα συγκρίνει με τα δεδομένα που απαιτούνται για την είσοδο και καλώντας τη συνάρτηση setcookie() του αρχείου chcookie.php, αποθηκεύει το cookie [31] στον υπολογιστή του διαχειριστή. Οι καρτέλες ελέγχου και διαχείρισης διαβάζουν το αποθηκευμένο cookie και ενεργοποιούνται δίνοντας πλήρη πρόσβαση στο σύστημα.

Το αρχείο logout.php (Βλ. Παράρτημα II.Β) είναι το αρχείο που διαχειρίζεται την έξοδο του διαχειριστή από το σύστημα και εκτελεί τις απαραίτητες ενέργειες αποσύνδεσης του από αυτό. Το αρχείο logout.php σβήνει το cookie από τον υπολογιστή του διαχειριστή μόλις πατηθεί το κουμπί log out στην καρτέλα εισόδου.



Εικόνα 26: Καρτέλα Εισόδου

Τέλος, το αρχείο chcookie.php (Βλ. Παράρτημα II.B) είναι το αρχείο που αναλαμβάνει τη δημιουργία και την αποθήκευση του αρχείου cookie, που πιστοποιεί την είσοδο στην ιστοσελίδα διαχείρισης.

4.2.2 Καρτέλα Status

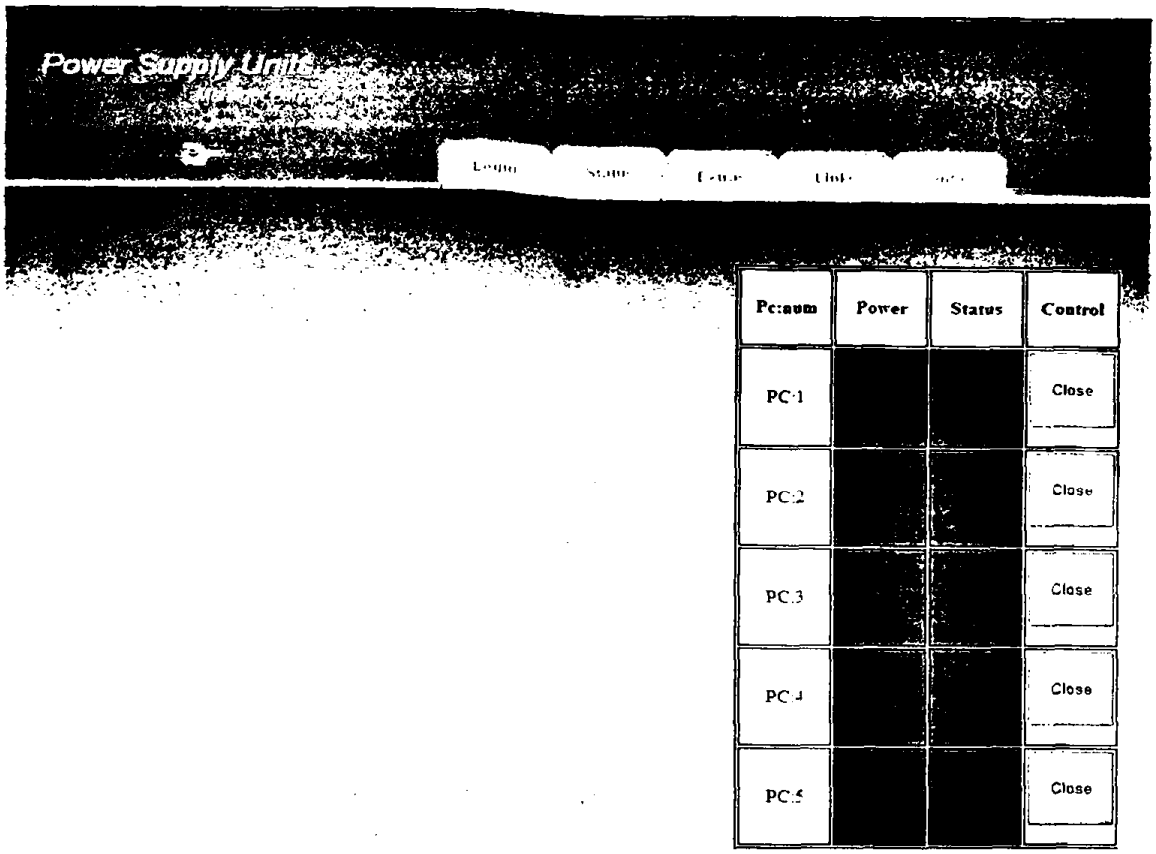
Η βασική δομή της υλοποίησης πραγματοποιείται με το αρχείο status.php (Βλ. Παράρτημα II.B). Είναι το αρχείο που επικοινωνεί με την κεντρική μονάδα ελέγχου μέσω του αρχείου telnet.php και τυπώνει στην οθόνη την κατάσταση της τροφοδοσίας και της λειτουργίας των υπολογιστών.

Το αρχείο status.php αρχικά εκτελεί την εντολή ping [32] για να επιβεβαιώσει την επικοινωνία της ιστοσελίδας διαχείρισης με τον επεξεργαστή της κεντρικής μονάδας ελέγχου. Στην περίπτωση που δεν υπάρχει επικοινωνία τυπώνει ένα μήνυμα σφάλματος. Διαφορετικά, χρησιμοποιώντας τις συναρτήσεις του αρχείου telnet.php, η ιστοσελίδα διαχείρισης συνδέεται με την κεντρική μονάδα ελέγχου. Στη συνέχεια, αποστέλλει την εντολή "*status.<αριθμός υπολογιστή>*". Μόλις λάβει

την απόκριση της εντολής, ελέγχει τα δεδομένα που παρέλαβε με τον τρόπο που φαίνεται στον πίνακα 4 και καλεί τις κατάλληλες συναρτήσεις.

Απόκριση διακομιστή	Λειτουργία status.php
Pc open	Καλεί τη συνάρτηση ok
Only power	Καλεί τη συνάρτηση onlypower
NO power	Καλεί τη συνάρτηση nopower

Πίνακας 4: Εντολές Κεντρικής Μονάδας Ελέγχου



Εικόνα 27: Καρτέλα Status

Η συνάρτηση **ok** δημιουργεί τρία κελιά στον πίνακα ελέγχου από τα οποία τα δύο πρώτα είναι χρώματος πράσινου, υποδεικνύοντας ότι ο υπολογιστής είναι σε λειτουργία, ενώ το τρίτο κελί αποτελείται από ένα κουμπί με την ετικέτα "Close" για την απενεργοποίηση του υπολογιστή . Η συνάρτηση **onlypower** δημιουργεί τρία κελιά στον πίνακα ελέγχου από τα οποία το πρώτο είναι χρώματος πράσινου, το δεύτερο χρώματος κόκκινου, υποδεικνύοντας ότι ο υπολογιστής έχει τροφοδοσία αλλά είναι απενεργοποιημένος, και τέλος το τρίτο κελί αποτελείται από ένα κουμπί με την ετικέτα "Open" για την ενεργοποίηση του υπολογιστή. Η συνάρτηση **nopower** δημιουργεί τρία κελιά στον πίνακα ελέγχου από τα οποία τα δύο πρώτα είναι χρώματος κόκκινου, υποδεικνύοντας ότι ο υπολογιστής δεν έχει τροφοδοσία και αυτονόητα δεν είναι σε λειτουργία, ενώ το τρίτο κελί αποτελείται από ένα κουμπί με την ένδειξη "NO POWER". Σε περίπτωση σφάλματος καλείται η συνάρτηση error(), η οποία δημιουργεί τρία κελιά στον πίνακα ελέγχου με την ένδειξη "ERROR!!!". Η καρτέλα Status φαίνεται στην εικόνα 27.

4.2.3 Καρτέλα Extras

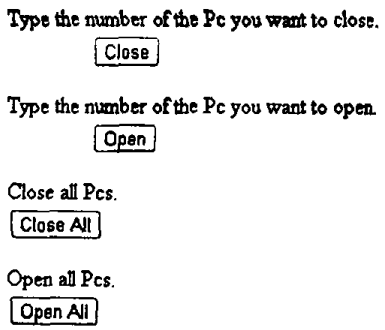
Το αρχείο open.php (Βλ. Παράρτημα II.Β) είναι το αρχείο που αναλαμβάνει να ενεργοποιήσει τους υπολογιστές που είναι συνδεδεμένοι με το σύστημα. Το αρχείο open.php επικοινωνεί με τον επεξεργαστή της κεντρικής μονάδας ελέγχου, μέσω των συναρτήσεων του αρχείου telnet.php, και αποστέλλει την εντολή "open.<αριθμός υπολογιστή>". Μόλις λάβει την απόκριση ότι ο υπολογιστής είναι ενεργοποιημένος, ολοκληρώνει τη λειτουργία του και παραπέμπει στην καρτέλα Status.

Το αρχείο close.php (Βλ. Παράρτημα II.Β) είναι το αρχείο που αναλαμβάνει να απενεργοποιήσει τους υπολογιστές. Το αρχείο close.php επικοινωνεί με τον επεξεργαστή της κεντρικής μονάδας ελέγχου, μέσω των συναρτήσεων του αρχείου telnet.php, και αποστέλλει την εντολή "close.<αριθμός υπολογιστή>". Μόλις λάβει την απόκριση ότι ο υπολογιστής είναι απενεργοποιημένος, ολοκληρώνει τη λειτουργία του και παραπέμπει στην καρτέλα Status.

Το αρχείο closeall.php (Βλ. Παράρτημα II.Β) είναι το αρχείο που αναλαμβάνει να απενεργοποιήσει όλους τους υπολογιστές ταυτόχρονα. Το αρχείο closeall.php επικοινωνεί με τον

επεξεργαστή της κεντρικής μονάδας ελέγχου, μέσω των συναρτήσεων του αρχείου telnet.php, και αποστέλλει την εντολή killall,η οποία απενεργοποιεί όλους τους ηλεκτρονικούς υπολογιστές. Μόλις όλοι οι συνδεδεμένοι στο σύστημα υπολογιστές απενεργοποιηθούν, τότε ολοκληρώνει τη λειτουργία του και παραπέμπει στην καρτέλα Status.

Το αρχείο openall.php (Βλ. Παράρτημα Π.Β) είναι το αρχείο που αναλαμβάνει να ενεργοποιήσει όλους τους υπολογιστές ταυτόχρονα. Το αρχείο openall.php επικοινωνεί με τον επεξεργαστή της κεντρικής μονάδας ελέγχου, μέσω των συναρτήσεων του αρχείου telnet.php, και αποστέλλει την εντολή wakeall, η οποία ενεργοποιεί όλους τους ηλεκτρονικούς υπολογιστές. Μόλις όλοι οι συνδεδεμένοι στο σύστημα υπολογιστές ενεργοποιηθούν, τότε ολοκληρώνει τη λειτουργία του και παραπέμπει στην καρτέλα Status.



Εικόνα 28: Καρτέλα Extras

Το αρχείο settings.php (Βλ. Παράρτημα Π.Β) είναι το αρχείο που διαχειρίζεται την καρτέλα extras. Το αρχείο settings.php δέχεται, από τις φόρμες που υπάρχουν στην καρτέλα διαχείρισης, τον αριθμό του υπολογιστή προς ενεργοποίηση ή απενεργοποίηση και στη συνέχεια, μέσω των

συναρτήσεων του αρχείου telnet.php, αποστέλλει την ανάλογη εντολή στον επεξεργαστή microblaze της κεντρικής μονάδας ελέγχου (PSUM). Η καρτέλα extras φαίνεται στην εικόνα 28.

Κεφάλαιο 5

Αποτελέσματα-Συμπεράσματα

Στην αρχή του κεφαλαίου παρουσιάζονται τα αποτελέσματα των πειραμάτων για τα επιμέρους τμήματα του συστήματος. Τέλος, δίνονται κάποιες μελλοντικές ιδέες για την εξέλιξη, την χρησιμότητα και τη λειτουργία του συστήματος.

5.1 Αποτελέσματα-Συμπεράσματα

Στην ενότητα παρουσιάζουμε τα αποτελέσματα των ελέγχων που διενεργήθηκαν στο σύστημα. Οι έλεγχοι αφορούν την ταχύτητα απόκρισης της υπηρεσίας telnet που υλοποιήσαμε στην πλατφόρμα ανάλογα με το μέγεθος τον πακέτων που λαμβάνει και τις ταχύτητες απόκρισης των εντολών αναλόγα με το πλήθος των υπολογιστών.

Ο πρώτος έλεγχος αφορά τη διαχείριση πακέτων TCP/IP διαφόρων μεγεθών από την εφαρμογή telnet που υλοποιήθηκε στην πλατφόρμα. Ο έλεγχος διενεργήθηκε για πακέτα μεγέθους 32, 64, 128, 512, 1024 και 2056 byte. Τα αποτελέσματα φαίνονται στον πίνακα 5.

Μέγεθος Πακέτου TCP/IP (byte)	Χρόνος Απόκρισης (ms)
32	<1
64	<1
128	<1
512	<1
1024	1
2056	Τέλος Μνήμης

Πίνακας 5: Χρόνοι Απόκρισης Telnet ανάλογα του μεγέθους των πακέτων TCP/IP.

Από τον παραπάνω πίνακα συμπεραίνουμε ότι η διαχείριση των πακέτων από τον επεξεργαστή Microblaze είναι πολύ γρήγορη. Ο επεξεργαστής μέσω του πρωτοκόλλου LwIP μπορεί να διαχειριστεί πακέτα μέχρι 1024 byte. Για το σύστημα το οποίο υλοποιήσαμε δεν είναι απαραίτητη η διαχείριση πακέτων μεγαλύτερου μεγέθους. Μια εντολή του συστήματος ελέγχου και διαχείρισης, όπως η

"status.30" για παράδειγμα, έχει μέγεθος 9 byte. Έτσι το σύστημα μπορεί να διαχειριστεί 113 εντολές σε ένα δευτερόλεπτο.

Για την μελέτη της απόκρισης του συστήματος χρησιμοποιήθηκε ένα κύκλωμα διευθυνσιοδότησης που αποκρίνεται σε όλες τις διευθύνσεις. Έτσι, κατέστη δυνατός ο έλεγχος πολλών εικονικών τροφοδοτικών.

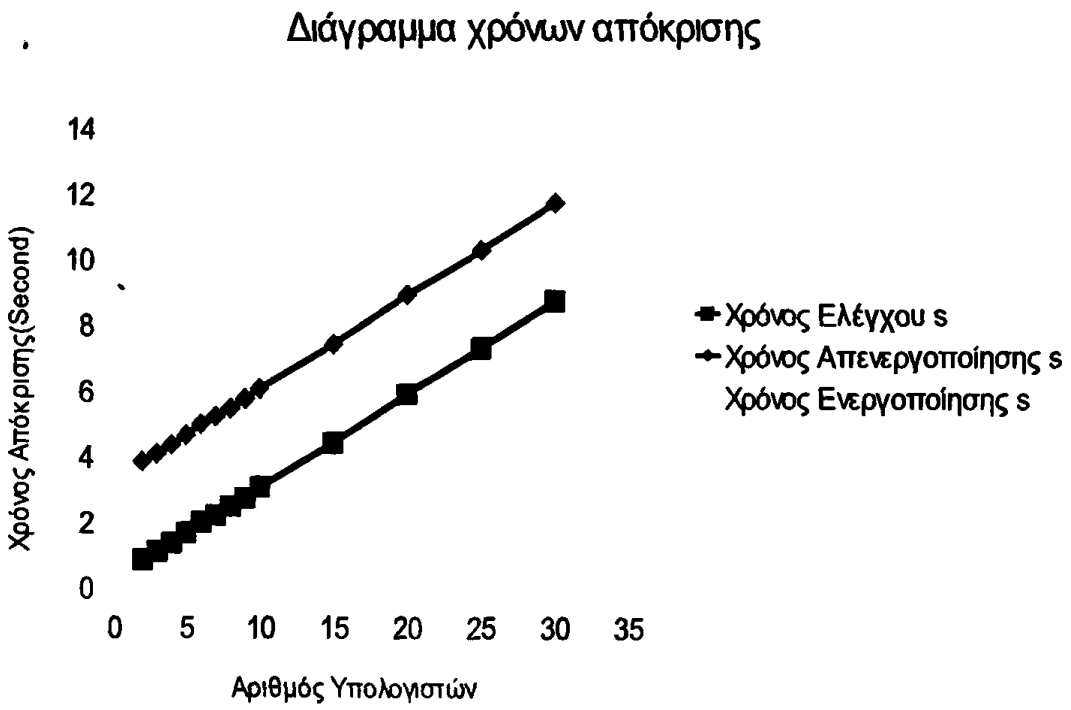
Η προσομοίωση πραγματοποιήθηκε με την εγκατάσταση σε έναν υπολογιστή ενός κυκλώματος διευθυνσιοδότησης που αποκρίνεται σε όλες τις διευθύνσεις. Ο υπολογιστής μετέδιδε την πληροφορία για την κατάσταση της τροφοδοσίας και της λειτουργίας του σε οποιαδήποτε διεύθυνση μετέδιδε η κεντρική μονάδα ελέγχου. Επίσης, η διαχείριση του υπολογιστή επιτρεπόταν σε οποιαδήποτε διεύθυνση μεταδιδόταν.

Πλήθος Υπολογιστών	Χρόνος Ελέγχου(sec)	Χρόνος Ενεργοποίησης(sec)	Χρόνος Απενεργοποίησης(sec)
2	0,84	0,82	3,83
3	1,08	1,09	4,07
4	1,36	1,36	4,35
5	1,65	1,63	4,63
6	1,98	1,97	4,97
7	2,18	2,19	5,19
8	2,46	2,45	5,44
9	2,71	2,71	5,73
10	3,04	3,05	6,03
15	4,39	4,4	7,38
20	5,86	5,84	8,87
25	7,25	7,24	10,23
30	8,68	8,66	11.67

Πίνακας 6: Χρόνοι απόκρισης ανάλογα με το πλήθος των υπολογιστών.

Τα αποτελέσματα των μετρήσεων για τον χρόνο απόκρισης του συστήματος ανάλογα με το πλήθος των υπολογιστών που είναι συνδεδεμένοι σε αυτό, παρουσιάζονται στον πίνακα 6. Η πρώτη στήλη περιέχει το πλήθος των υπολογιστών (Πίνακας 6), η δεύτερη στήλη περιέχει το χρόνο απόκρισης για την εμφάνιση των αποτελεσμάτων ελέγχου, η τρίτη στήλη περιέχει το χρόνο απόκρισης

για την ενεργοποίηση των υπολογιστών και η τελευταία στήλη περιέχει το χρόνο απενεργοποίησης των υπολογιστών. Η γραφική αναπαράσταση των μετρήσεων φαίνεται στην εικόνα 22. Τα αποτελέσματα που αφορούν πλήθος υπολογιστών μεγαλύτερο και ίσο του τέσσερα είναι αποτελέσματα προσομοίωσης.



Εικόνα 22: Διάγραμμα χρόνου απόκρισης ως προς το πλήθος των υπολογιστών

Στην εικόνα 22 φαίνεται η γραφική απεικόνιση των αποτελεσμάτων της απόκρισης του συστήματος συναρτήσει του αριθμού των υπολογιστών που ελέγχονται.

Παρατηρούμε ότι η σχέση των χρόνων απόκρισης είναι γραμμικός ως προς το πλήθος των υπολογιστών. Ο χρόνος ελέγχου φαίνεται με μπλε χρώμα και σχήμα τετραγώνου, ο χρόνος απενεργοποίησης με πορτοκαλί χρώμα και σχήμα ρόμβου, ενώ ο χρόνος ενεργοποίησης με κίτρινο χρώμα και σχήμα τριγώνου ταυτίζεται με τον χρόνο ελέγχου. Αυτό συμβαίνει διότι, για την ενεργοποίηση του υπολογιστή απαιτείται η αποστολή στην μητρική πλακέτα ενός σήματος διάρκειας μικρότερης του δευτερολέπτου, ενώ για την απενεργοποίηση του υπολογιστή απαιτείται η αποστολή ενός σήματος διάρκειας περίπου τριών δευτερολέπτων. Ο περιορισμός αυτός κάνει το σύστημα αρκετά αργό σε σχέση με τις δυνατότητές του στη διαχείριση συσκευών. Στην ενότητα 5.2 του κεφαλαίου

αναφέρονται οι ενδεχόμενες χρήσεις του συστήματος ελέγχου και διαχείρισης.

5.2 *Μελλοντικές Επεκτάσεις*

Το σύστημα υλοποιήθηκε για τον έλεγχο και τη διαχείριση των υπολογιστών μίας συστοιχίας ηλεκτρονικών υπολογιστών. Το σύστημα μπορεί να χρησιμοποιηθεί στην υπάρχουσα υλοποίηση για τον έλεγχο και τη διαχείριση και άλλων συσκευών, πλην των ηλεκτρονικών υπολογιστών, με τη απαίτηση να υπάρχει μία έξοδος που να ενημερώνει για τη κατάσταση των συσκευών και μια έξοδος για την τροφοδοσία τους.

Έτσι, το σύστημα ελέγχου και διαχείρισης μπορεί να χρησιμοποιηθεί επίσης:

- Στον απομακρυσμένο έλεγχο ανιχνευτών.
- Στην απομακρυσμένη λήψη μετρήσεων, εάν επεκταθεί ο δίαυλος του συστήματος PSUM απο δύο bit σε περισσότερα υποστηρίζοντας επιπλέον σήματα.

Τα επιμέρους τμήματα του συστήματος, δηλαδή η ιστοσελίδα, τα κυκλώματα διευθυνσιοδότησης και η κεντρική μονάδα ελέγχου μπορούν να χρησιμοποιηθούν ξεχωριστά για το σχεδιασμό και την υλοποίηση άλλων συστημάτων. Για παράδειγμα, η ιστοσελίδα διαχείρισης μπορεί να χρησιμοποιηθεί σε ένα σύστημα, ώστε να λαμβάνει και να εμφανίζει δεδομένα που προέρχονται από έναν μικροελεγκτή.

Αναφορές

[01] http://public.web.cern.ch/public/en/spotlight/SpotlightGridFactsAndFiction-en.html

[02] http://www.sata-io.org/

[03] http://www.xilinx.com

[04] http://www.xilinx.com/support/documentation/data_sheets/ds099.pdf

[05] http://www.digilentinc.com/

[06] http://www.xilinx.com/tools/microblaze.htm

[07] http://www.ece.ucdavis.edu/~vojin/CLASSES/EEC180B/Fall99/Writings/RISC-Chaptr.PDF

[08] http://www.telnet.org/htm/faq.htm

[09] http://www.playtool.com/pages/psuconnectors/connectors.html

[10] http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_1/oslib_rm.pdf

[11] http://savannah.nongnu.org/projects/lwip/

[12] http://www.xilinx.com/support/documentation/boards_and_kits/ug257.pdf

[13] http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_3/edk_ctt.pdf

[14] http://opensource.org/licenses/BSD-2-Clause

[15] http://www.xilinx.com/support/documentation/ipembedprocess_memoryinterface_lmb.htm

[16] http://www.xilinx.com/support/documentation/ip_documentation/plb_v46.pdf

[17] http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_3/est_rm.pdf

[18] http://wiki.altium.com/display/ADOH/Getting+Started+with+Altium+Designer

[19] http://www.alldatasheet.com/datasheet-pdf/pdf/158545/MOTOROLA/MC74AC521N.html

[20] http://www.ti.com/lit/ds/symlink/sn74ls08.pdf

[21] http://www.physics.wisc.edu/undergrads/courses/fall2011/623/ds/74LS00.pdf

[22] http://ecee.colorado.edu/~mcclurel/sn74ls02rev5.pdf

[23] http://httpd.apache.org/

[24] http://www.apache.org/

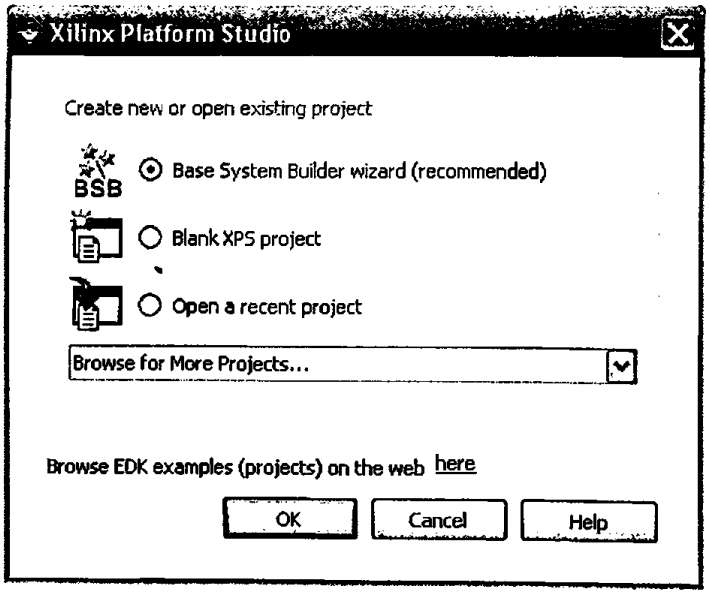
[25] http://www.php.net/

[26] http://web.mit.edu/professional/short-programs/courses/applied_cyber_security.html

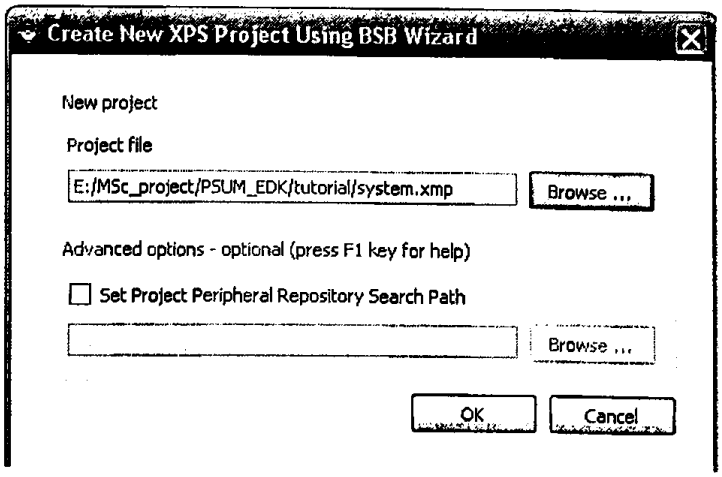
[27] Linux man page, use the command *man ping*

Παράρτημα Ι

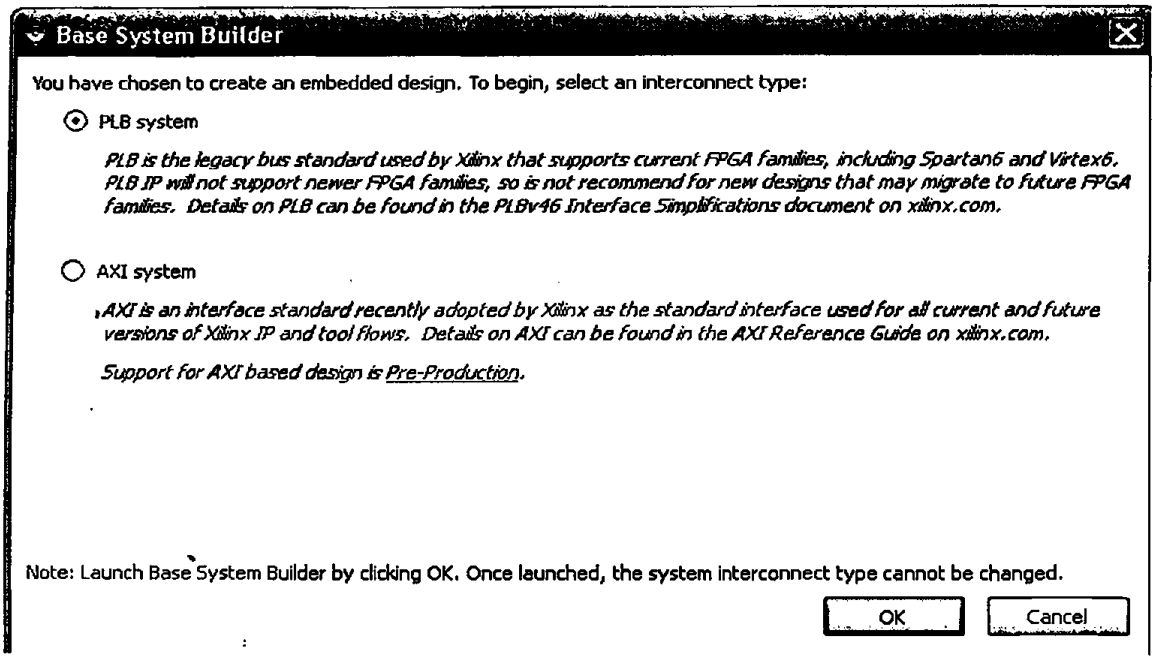
Ι.Α Οδηγός υλοποίησης του επεξεργαστή Microblaze στο EDK



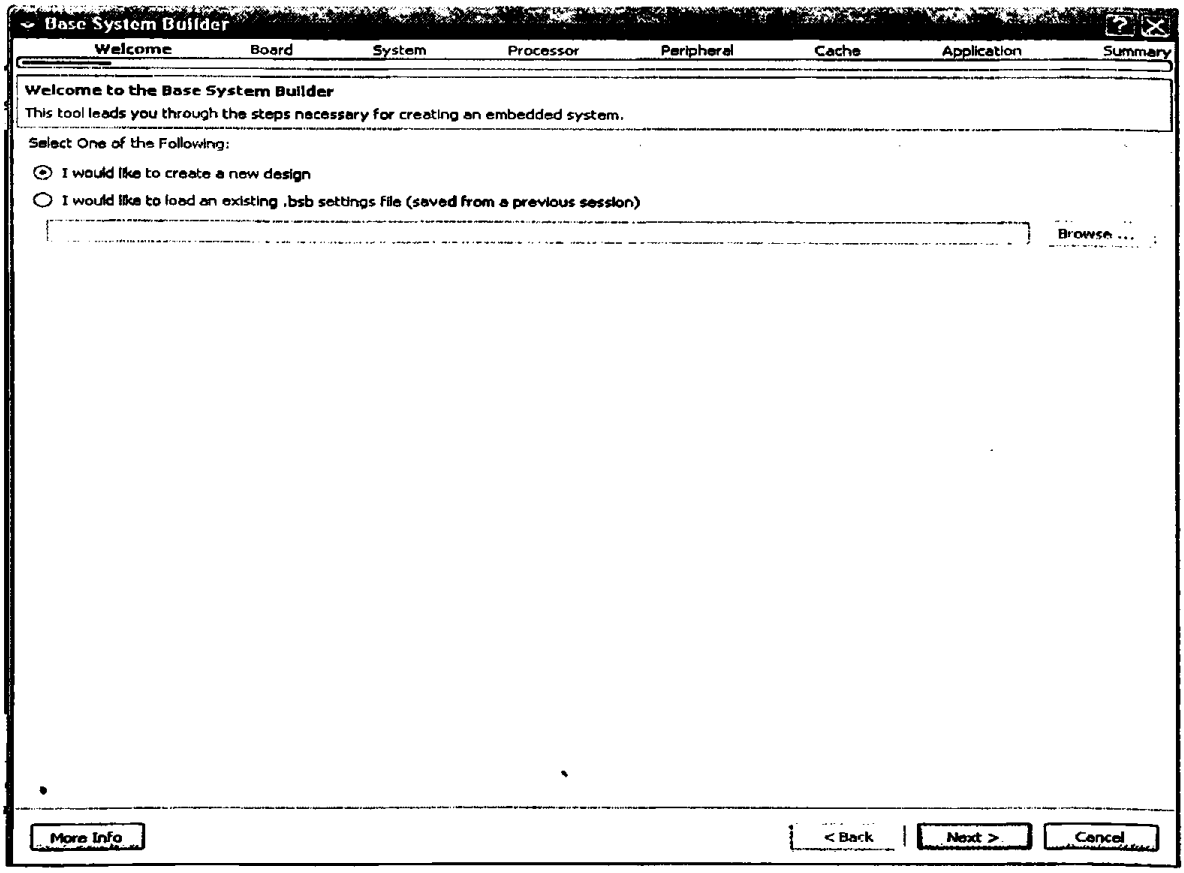
- Επιλέγουμε base system wizard

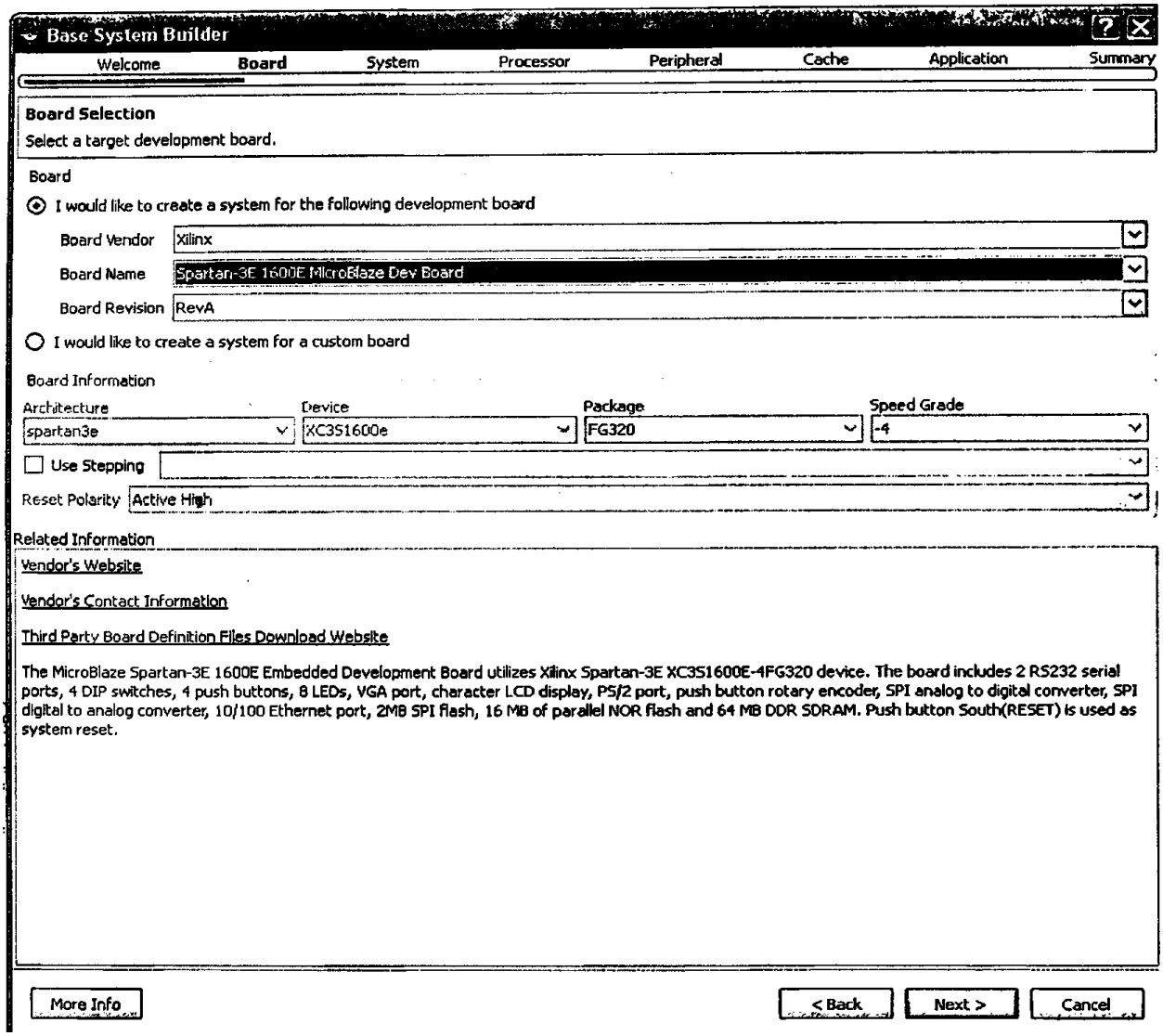


- Ορίζουμε το το project path (no spaces) .

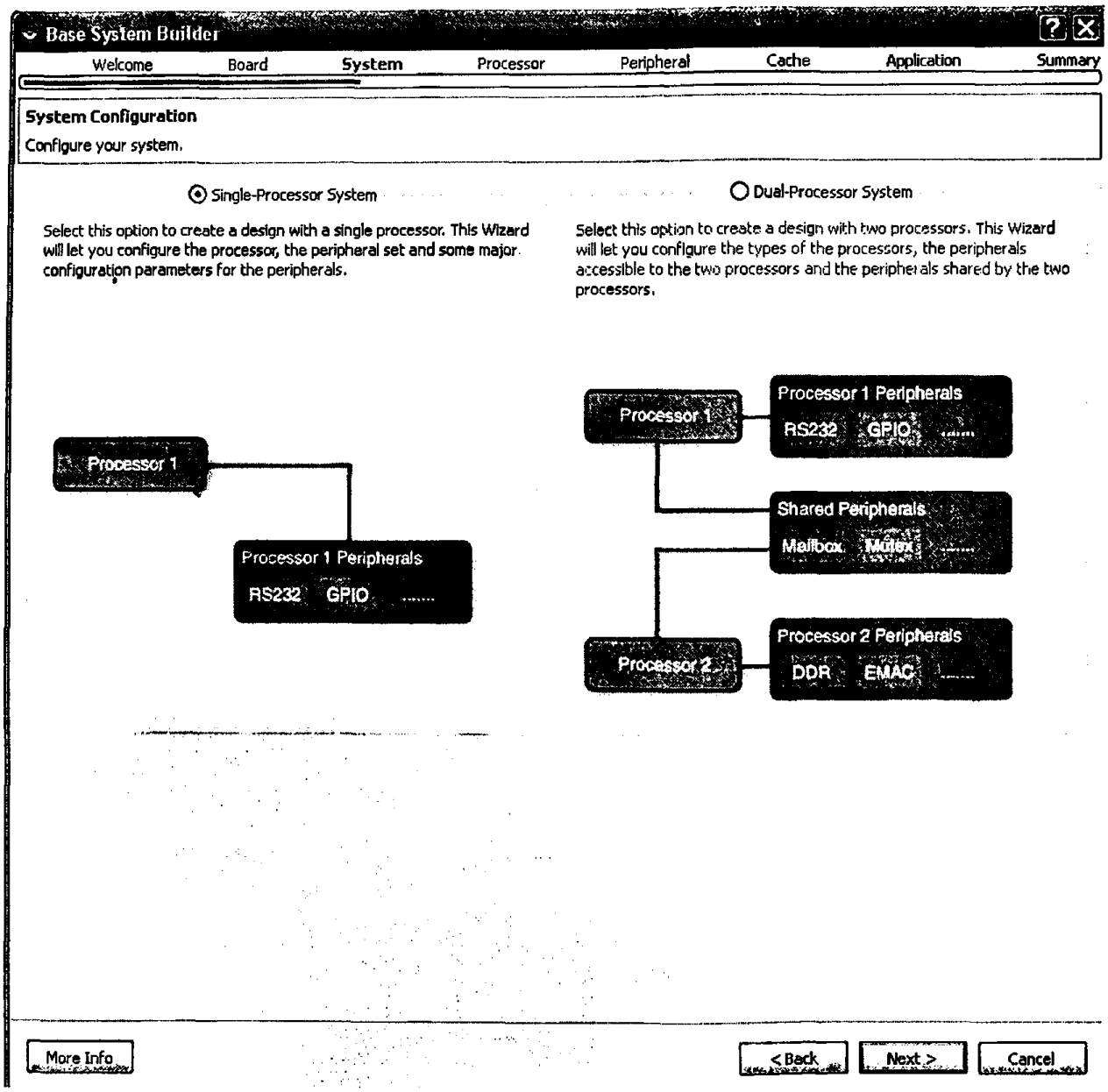


- Επιλέγουμε το PLB system

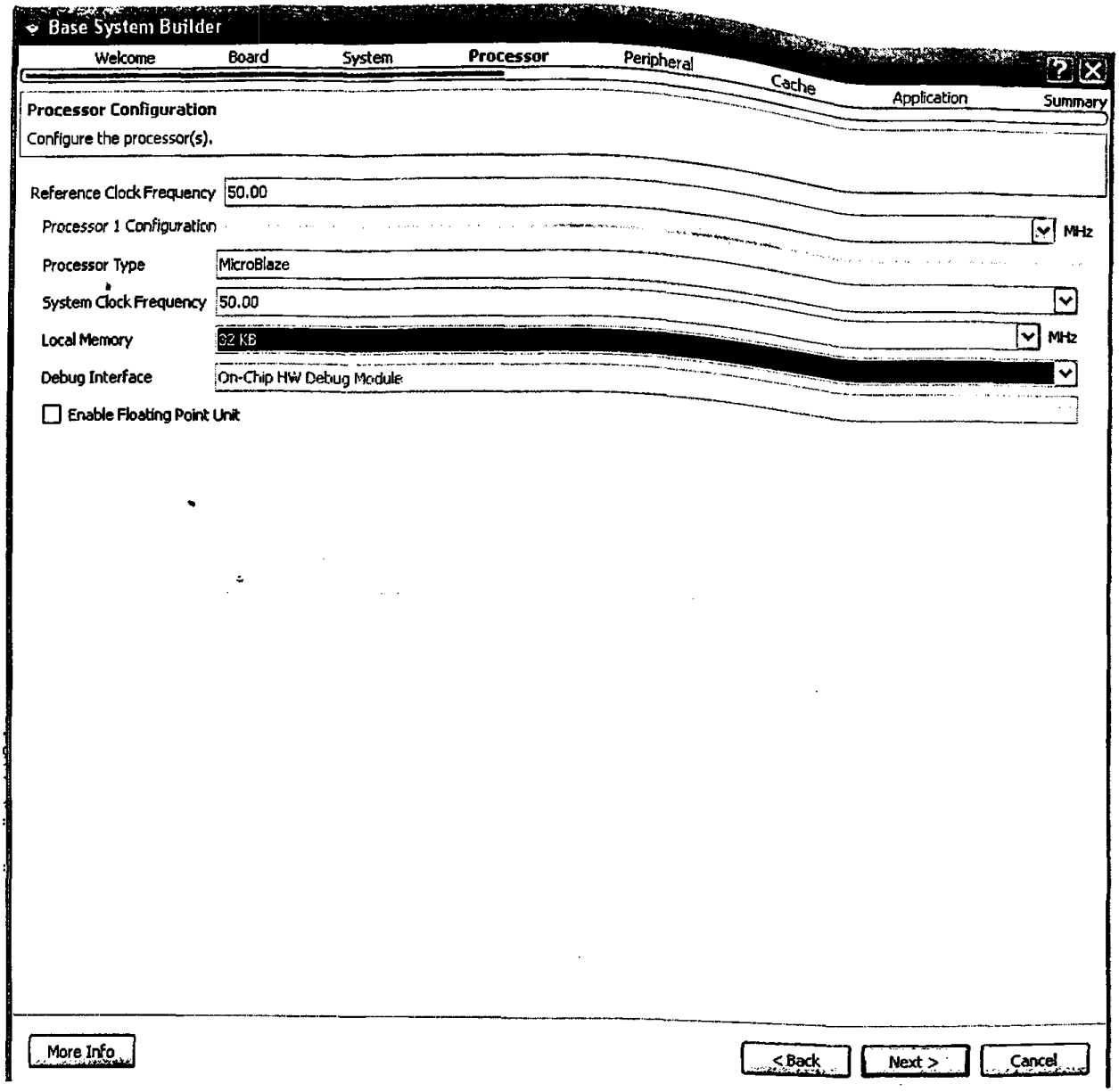




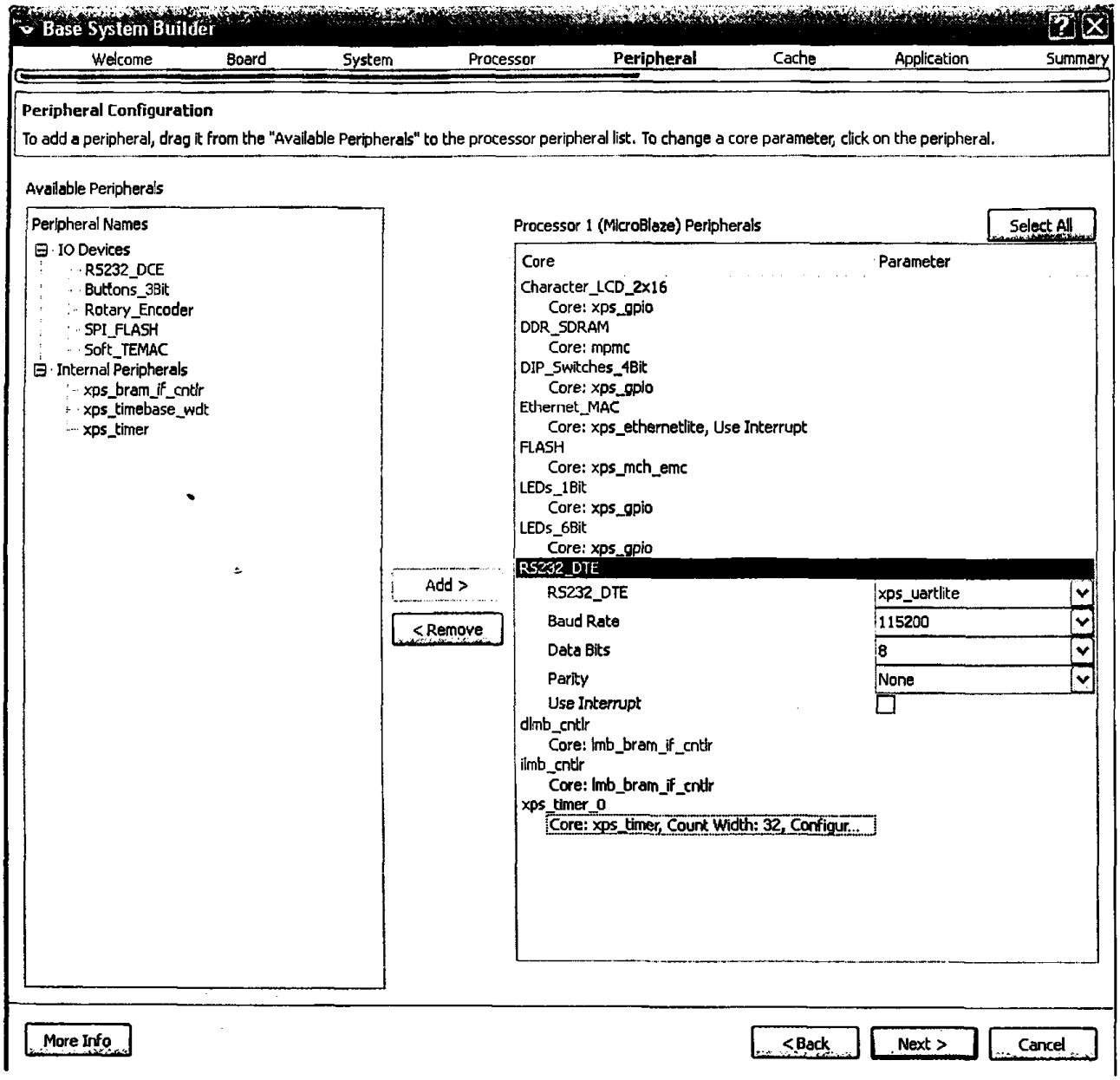
- Στην αρχική οθόνη που εμφανίζεται επιλέγουμε, την κατασκευάστρια εταιρία της πλατφόρμας, την πλατφόρμα που έχουμε επιλέξει καθώς και την κλάση της πλατφόρμας.
  - Xilinx.
  - SPARTAN-3E 1600E Microblaze Dev Board.
  - Revision D



- Επιλέγουμε την υλοποίηση ενός επεξεργαστή



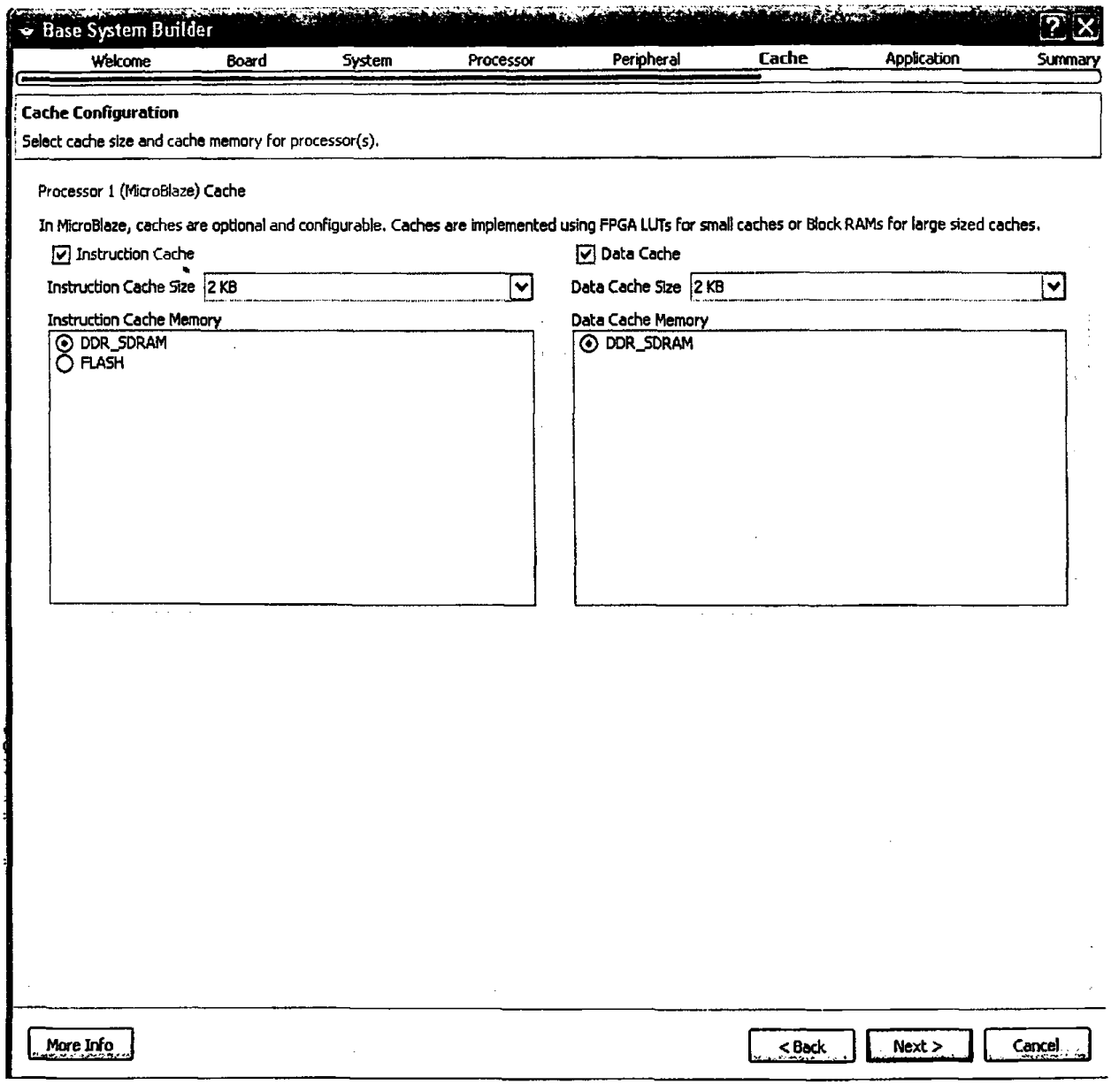
- Συνεχίζοντας στην επόμενη σελίδα επιλέγουμε ότι θα χρειαστούμε έναν επεξεργαστή ενώ στην επόμενη επιλέγουμε το χρονισμό του Hardware, το χρονισμό του επεξεργαστή καθώς και την εσωτερική μνήμη του.
  - 50.00Mhz
  - 50.00Mhz
  - 32KB



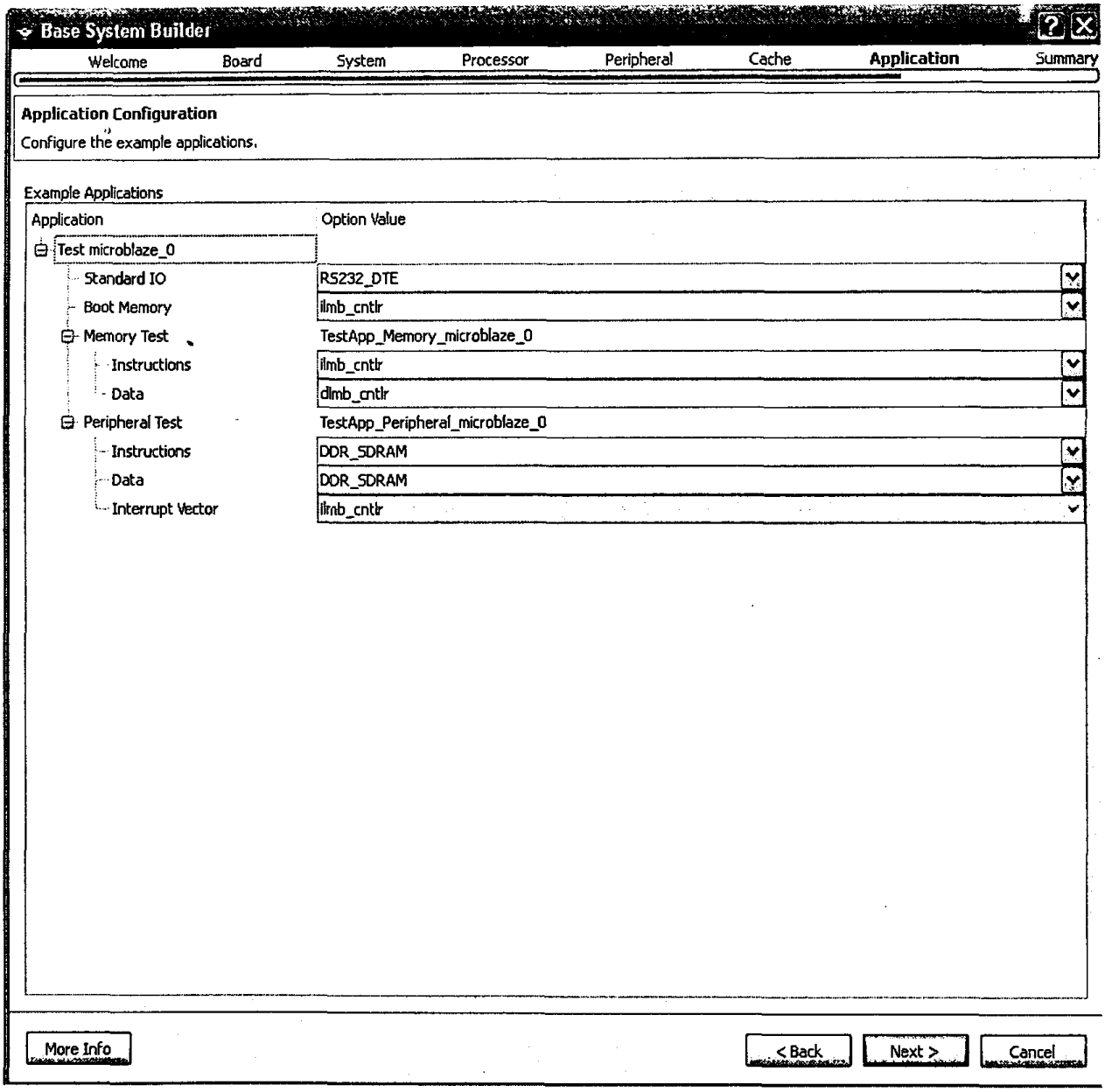
- Στην επόμενη σελίδα θα επιλέξουμε τα περιφερειακά που θα χρειαστούμε για την υλοποίηση. Επιλέγουμε τα:

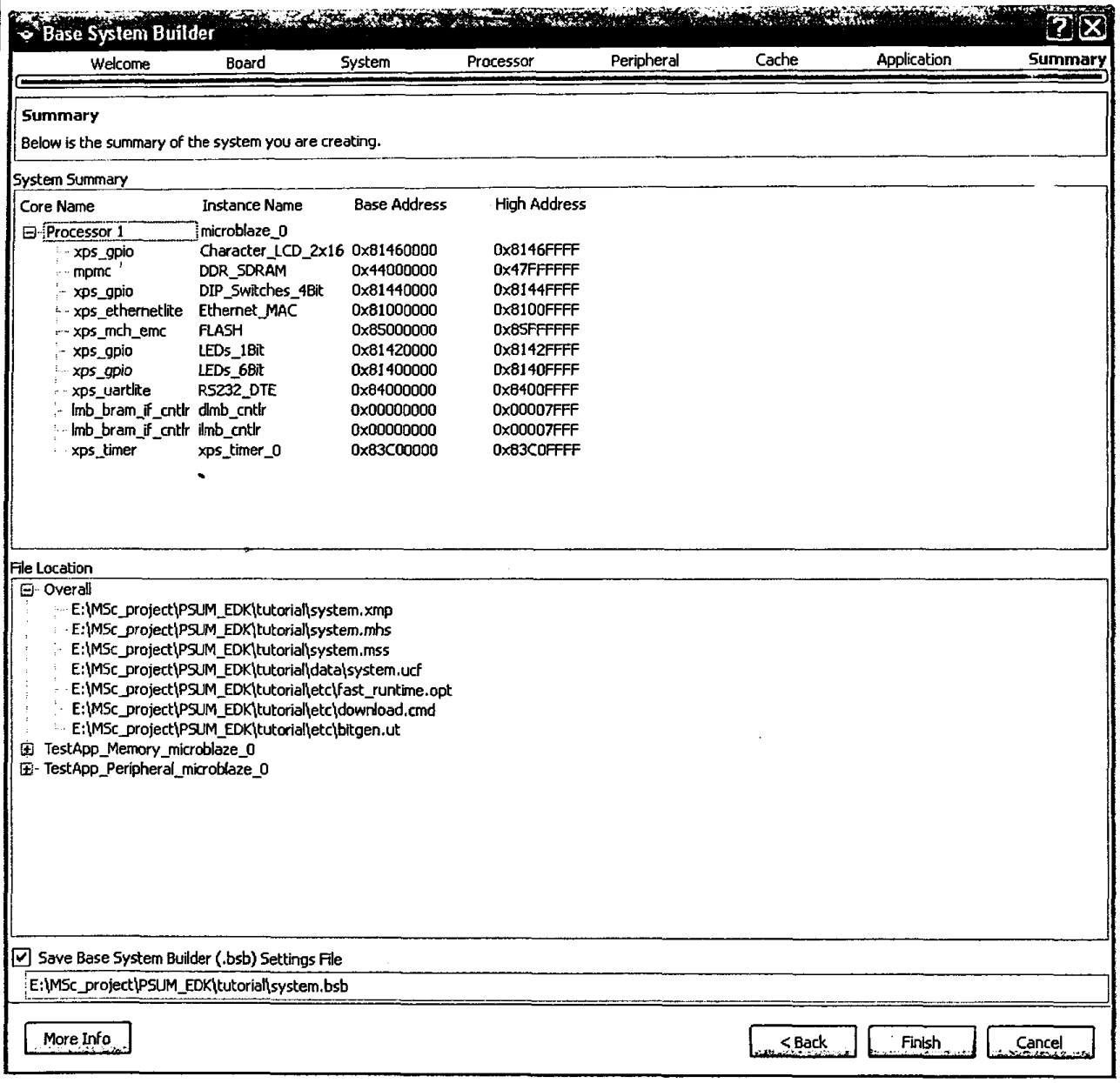
  - Character_LCD_2x16
  - DDR_SDRAM
  - DIP_SWITCHES_4Bit
  - Ethernet_MAC (with the use interrupt option)
  - FLASH
  - LED's_1bit

- LED's_6bit
- RS232_DTE (baud Rate:115200, Data bits:8, Parity:None)
- dlmb_cntlr
- ilmb_cntlr
- xps_timer_0 (Count width:32, Configure Mode:One timer present, use Interrupt option)



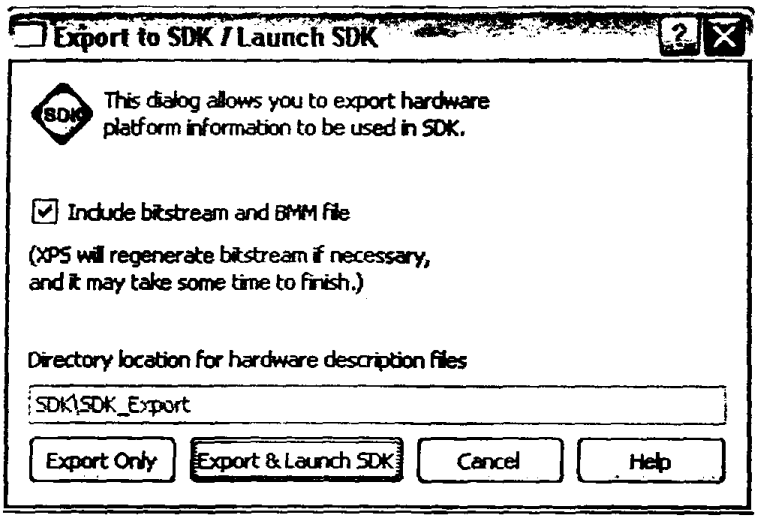
- Στην οθόνη που ακολουθεί επιλέγουμε τη μνήμη Cache του συστήματος.
  - Instruction Memory(2KB DDR SDRAM).
  - Data Memory(2KB DDR SDRAM).



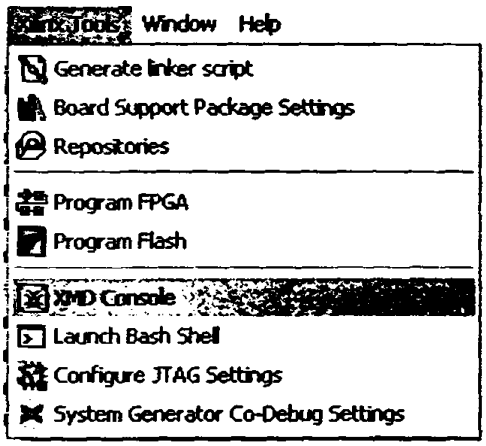


I.B Οδηγός υλοποίησης του Standalone OS στο SDK

Άνοιγμα του λογισμικού SDK μέσα από το λογισμικό EDK, επιλέγοντας το κουμπί SDK.



- επιλέγουμε Export & Launch SDK
- στο λογισμικό SDK επιλέγουμε XMD Console από την καρτέλα Xilinx tools



στο χώρο δεξιά εμφανίζεται το τερματικό XMD

- με την εντολή "mb" συνδεόμαστε με τον επεξεργαστή Microblaze.

```
                    Xilinx Microprocessor Debugger (XMD) Engine
Xilinx EDK 12.3 Build EDK_MS3.70d
Copyright (c) 1995-2009 Xilinx, Inc.  All rights reserved.

XMD%
XMD% mb
Accepted a new TCLSock connection from 127.0.0.1 on port 3629
```

```
JTAG chain configuration
--------------------------------------------------
Device   ID Code          IR Length    Part Name
 1       21c3a093            6         XC3S1600E
 2       d5046093            8         XCF04S
 3       d5046093            8         XCF04S
 4       06e5e093            8         XC2C64A

MicroBlaze Processor Configuration :
-------------------------------------
Version.............................8.00.a
Optimization........................Area
Interconnect........................PLB_v46
MMU Type............................No_MMU
No of PC Breakpoints................1
No of Read Addr/Data Watchpoints...0
No of Write Addr/Data Watchpoints..0
Instruction Cache Support..........on
Instruction Cache Base Address.....0x44000000
Instruction Cache High Address.....0x47ffffff
Data Cache Support.................on
Data Cache Base Address............0x44000000
Data Cache High Address............0x47ffffff
Exceptions  Support................off
FPU  Support.......................off
Hard Divider Support...............off
Hard Multiplier Support............on - (Mul32)
Barrel Shifter Support.............on
MSR clr/set Instruction Support....on
Compare Instruction Support........on
Data Cache Write-back Support......off

Connected to MicroBlaze "mdm" target. id = 0
Starting GDB server for "mdm" target (id = 0) at TCP port no 1234

Note:: "mbconnect" command is Deprecated. Use "connect mb" command
```

- με την εντολή "cd SDK/SDK_Workspace_35/lwip_echo_server_0/Debug" μεταφερόμαστε στον φάκελο στον οποίο είναι αποθηκευμένο το δυαδικό αρχείο .elf το οποίο προέκυψε από το compilation του κώδικα για το συγκεκριμένο hardware

- με την εντολή "dow lwip_echo_server_0.elf" φορτώνουμε το δυαδικό αρχείο στην μνήμη του επεξεργαστή Microblaze.

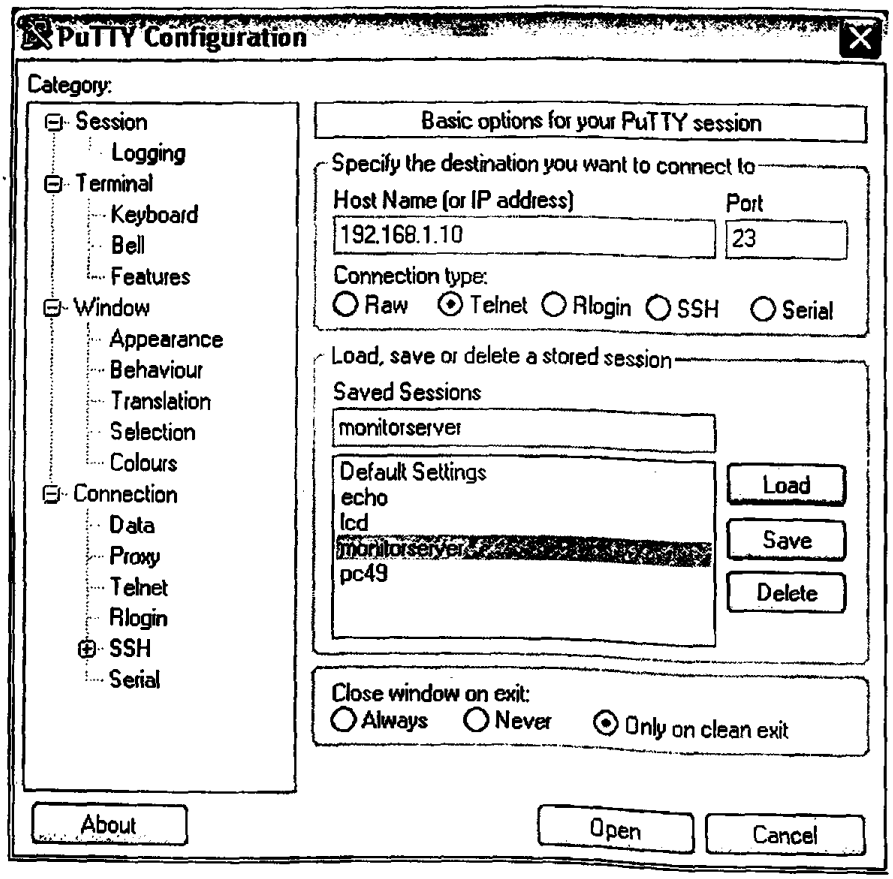
```
Downloading Program -- lwip_echo_server_0.elf
        section, .vectors.reset: 0x00000000-0x00000007
        section, .vectors.sw_exception: 0x00000008-0x0000000f
        section, .vectors.interrupt: 0x00000010-0x00000017
        section, .vectors.hw_exception: 0x00000020-0x00000027
        section, .text: 0x44000000-0x4401b717
        section, .init: 0x4401b718-0x4401b73f
        section, .fini: 0x4401b740-0x4401b75f
        section, .rodata: 0x4401b760-0x4401c537
        section, .data: 0x4401c538-0x4401cb03
        section, .ctors: 0x4401cb04-0x4401cb0b
        section, .dtors: 0x4401cb0c-0x4401cb13
        section, .eh_frame: 0x4401cb14-0x4401cb17
        section, .jcr: 0x4401cb18-0x4401cb1b
        section, .bss: 0x4401cb40-0x440b44cb
        section, .heap: 0x440b44cc-0x440be4cf
```

```
      section, .stack: 0x440be4d0-0x440c84cf
Setting PC with Program Start Address 0x00000000
System Reset .... DONE
```

- Τέλος, με την εντολή run θέτουμε σε λειτουργία τον επεξεργαστή.

```
RUNNING> XMD%
```

Χρησιμοποιώντας το λογισμικό putty συνδεόμαστε με την κεντρική μονάδα ελέγχου και αποστέλλουμε εντολές στο σύστημα.



Ι.Γ Πίνακες Διασυνδέσεών ολοκληρωμένων κυκλώματος διευθυνσιοδότησης

Διασυνδέσεις Συγκριτή 8-bit.

pin Number	Connection
1	Ground
2	Board pin #N15
3	External Switch
4	Board pin #N14
5	External Switch
6	Board pin #E15
7	External Switch
8	Board pin #V7
9	External Switch
10	Ground
11	Ground
12	Ground
13	Ground
14	Ground
15	Ground
16	Ground
17	Ground
18	Ground
19	Nand pin #1,#2
20	VCC

Συνδέσεις Πύλης Nand

pin Number	Connection
1	Comparator pin #19
2	Comparator pin #19
3	And pin #14(VCC)
4	Pc Push Button
5	Pc Push Button
6	Nor pin 3
7	Ground
8	Not Connected
9	Ground
10	Ground
11	Not Connected
12	Ground
13	Ground
14	VCC

Συνδέσεις Πύλης And Status

pin Number	Connection
1	Purple Cable
2	Purple Cable
3	Board pin R14
4	Grey Cable
5	Grey Cable
6	Board pin T14
7	Ground
8	Not Connected
9	Ground
10	Ground
11	Not Connected
12	Ground
13	Ground
14	VCC

Συνδέσεις Πύλης And Remote

pin Number	Connection
1	Board pin V6
2	Nand pin #3
3	Nor pin #2
4	Ground
5	Ground
6	Not Connected
7	Ground
8	Not Connected
9	Ground
10	Ground
11	Not Connected
12	Ground
13	Ground
14	VCC

Συνδέσεις Πύλης Nor

pin Number	Connection
1	Motherboard Switch
2	And Remote pin3
3	Nand pin 6
4	Not Connected
5	Ground
6	Ground
7	Ground
8	Ground
9	Ground
10	Not Connected
11	Ground
12	Ground-
13	Not Connected
14	VCC

Παράρτημα ΙΙ

ΙΙ.Α Κώδικας Λογισμικού Κεντρικής Μονάδας ελέγχου.

```
#include <stdio.h>

#include "xparameters.h"

#include "netif/xadapter.h"
#include "platform.h"
#include "platform_config.h"

/* defined by each RAW mode application */
void print_app_header();
int start_application();
int transfer_data();

/* missing declaration in lwIP */
void lwip_init();

void
print_ip(char *msg, struct ip_addr *ip)
{
        print(msg);
        xil_printf("%d.%d.%d.%d\n\r", ip4_addr1(ip), ip4_addr2(ip),
                        ip4_addr3(ip), ip4_addr4(ip));
}

void
print_ip_settings(struct ip_addr *ip, struct ip_addr *mask, struct ip_addr *gw)
{
        print_ip("Board IP: ", ip);
        print_ip("Netmask : ", mask);
        print_ip("Gateway : ", gw);
}

int main()
{
        struct netif *netif, server_netif;
        struct ip_addr ipaddr, netmask, gw;

        /* the mac address of the board. this should be unique per board */
        unsigned char mac_ethernet_address[] = { 0x00, 0x0a, 0x35, 0x00, 0x01, 0x02 };

        netif = &server_netif;

        init_platform();

        /* initliaze IP addresses to be used */
        IP4_ADDR(&ipaddr,  192, 168,   1, 10);
        IP4_ADDR(&netmask, 255, 255, 255,  0);
        IP4_ADDR(&gw,      192, 168,   1,  1);

        print_app_header();
        print_ip_settings(&ipaddr, &netmask, &gw);

        lwip_init();

        /* Add network interface to the netif_list, and set it as default */
        if (!xemac_add(netif, &ipaddr, &netmask, &gw, mac_ethernet_address, PLATFORM_EMAC_BASEADDR)) {
                xil_printf("Error adding N/W interface\n\r");
                return -1;
        }
        netif_set_default(netif);
```

```
	/* Create a new DHCP client for this interface.
	 * Note: you must call dhcp_fine_tmr() and dhcp_coarse_tmr() at
	 * the predefined regular intervals after starting the client.
	 */
	/* dhcp_start(netif); */

	/* now enable interrupts */
	platform_enable_interrupts();

	/* specify that the network if is up */
	netif_set_up(netif);

	/* start the application (web server, rxtest, txtest, etc..) */
	start_application();

	/* receive and process packets */
	while (1) {
		xemacif_input(netif);
		transfer_data();
	}

	/* never reached */
	cleanup_platform();

	return 0;
}



/*Libraries*/
#include <stdio.h>
#include <string.h>
#include "xparameters.h"
#include "xgpio.h"
#include "xstatus.h"
#include "lwip/err.h"
#include "lwip/tcp.h"
/*Global variables*/
#define LED_DELAY  1000000
#define LED_DELAY2		3000000
#define LED_CHANNEL 1
/*Functions*/
int getstatus(int pcnumber);
int cpc(int idc);
int openpc(int ido);
void GpioDriverHandler(void *CallBackRef);
/*Drivers*/
XGpio GpioOutput; /* The driver instance for GPIO Device configured as O/P */
XGpio GpioInput; /* The driver instance for GPIO Device configured as I/P */

int transfer_data() {
	return 0;
}

void print_app_header()
{
	xil_printf("\n\r\n\r-----lwIP TCP Control & Monitor Server ------\n\r");
	xil_printf("This Server Monitors and Controls the GriD Site\n\r");
}

err_t recv_callback(void *arg, struct tcp_pcb *tpcb,
		struct pbuf *p, err_t err)
{
	/*variables*/
	int i;
	int status[0x1e];
	/* do not read the packet if we are not in ESTABLISHED state */
	if (!p) {
		tcp_close(tpcb);
		tcp_recv(tpcb, NULL);
		return ERR_OK;
	}

	/*create a variable to store the payload*/
	char *data = (char *)p->payload;
	xil_printf("\n\r\n\r%s\n\r",data);
	/* indicate that the packet has been received */
```

```
tcp_recved(tpcb, p->len);


/*Check the received command */
if(!strncmp(data,"status",6)){
            char * pch;
            int gn;
            pch = strtok (data,"status.");
            pbuf_free(p->payload);
            gn=atoi(pch);
            i=gn;
            status[i]=getstatus(i);
            if(status[i]==0x0){
            strcpy(data,"Pc has no Power\n\r");
            tcp_write(tpcb, p->payload,17, 1);
            pbuf_free(p->payload);
            }
            else if(status[i]==0x1){
            strcpy(data,"Only Power\n\r");
            tcp_write(tpcb, p->payload,12, 1);
            pbuf_free(p->payload);
            }
            else if(status[i]==0x3){
            strcpy(data,"Pc Open\n\r");
            tcp_write(tpcb, p->payload,9, 1);
            pbuf_free(p->payload);
            }
            else{
            strcpy(data,"ERROR!!!\n\r");
            tcp_write(tpcb, p->payload,10, 1);
            pbuf_free(p->payload);
            }
}
else if(!strncmp(data,"close",5)){
            char * pch;
            int gn;
            pch = strtok (data,"status.");
            pbuf_free(p->payload);
            gn=atoi(pch);
            i=gn;
            status[i]=getstatus(i);
            if(status[i]==0x0){
            strcpy(data,"Pc has no Power\n\r");
            tcp_write(tpcb, p->payload,17, 1);
            pbuf_free(p->payload);
            }
            else if(status[i]==0x1){
            strcpy(data,"Pc Already Closed\n\r");
            tcp_write(tpcb, p->payload,19, 1);
            pbuf_free(p->payload);
            }
            else if(status[i]==0x3){
            cpc(i);
            strcpy(data,"Pc Closed\n\r");
            tcp_write(tpcb, p->payload,11, 1);
            pbuf_free(p->payload);
            }
            else{
            strcpy(data,"ERROR!!!\n\r");
            tcp_write(tpcb, p->payload,10, 1);
            pbuf_free(p->payload);
            }
}
else if(!strncmp(data,"open",4)){
            char * pch;
            int gn;
            pch = strtok (data,"status.");
            pbuf_free(p->payload);
            gn=atoi(pch);
            i=gn;
            status[i]=getstatus(i);
            if(status[i]==0x0){
            strcpy(data,"Pc has no Power\n\r");
            tcp_write(tpcb, p->payload,18, 1);
            pbuf_free(p->payload);
            }
            else if(status[i]==0x1){
            openpc(i);
            strcpy(data,"Pc opened\n\r");
```

```
                tcp_write(tpcb, p->payload,12, 1);
                pbuf_free(p->payload);
                                                }
                else if(status[i]==0x3){
                strcpy(data,"Pc already open\n\r");
                tcp_write(tpcb, p->payload,18, 1);
                pbuf_free(p->payload);
                }
                else{
                strcpy(data,"ERROR!!!\n\r");
                tcp_write(tpcb, p->payload,10, 1);
                pbuf_free(p->payload);
                }
}
else if(!strncmp(data,"help",4)){
pbuf_free(p->payload);
strcpy(data,"For status type:status.<pcnumber or all>\n\r");
tcp_write(tpcb, p->payload,42, 1);
pbuf_free(p->payload);
strcpy(data,"To close a PC type:close.<pcnumber or all>\n\r");
tcp_write(tpcb, p->payload,44, 1);
pbuf_free(p->payload);
strcpy(data,"To open a PC type:open.<pcnumber or all>\n\r");
tcp_write(tpcb, p->payload,42, 1);
pbuf_free(p);
}
else if(!strncmp(data,"exit",4)){
                tcp_close(tpcb);
}
else if(!strncmp(data,"paul",4)){
                pbuf_free(p->payload);
                strcpy(data,"\n");
                tcp_write(tpcb, p->payload,2, 1);
}
else if(!strncmp(data,"killall",7)){
                for(i=0;i<6;i++){
                                status[i]=getstatus(i);
                                                if(status[i]==0x0){
                                                strcpy(data,"Pc has no Power\n\r");
                                                tcp_write(tpcb, p->payload,17, 1);
                                                pbuf_free(p->payload);
                                                }
                                                else if(status[i]==0x1){
                                                strcpy(data,"Pc Already Closed\n\r");
                                                tcp_write(tpcb, p->payload,19, 1);
                                                pbuf_free(p->payload);
                                                }
                                                else if(status[i]==0x3){
                                                cpc(i);
                                                strcpy(data,"Pc Closed\n\r");
                                                tcp_write(tpcb, p->payload,11, 1);
                                                pbuf_free(p->payload);
                                                }
                                                else{
                                                strcpy(data,"ERROR!!!\n\r");
                                                tcp_write(tpcb, p->payload,10, 1);
                                                pbuf_free(p->payload);
                                                }
                }
                }
else if(!strncmp(data,"wakeall",7)){
                for(i=0;i<6;i++){
                                status[i]=getstatus(i);
                                                if(status[i]==0x0){
                                                strcpy(data,"Pc has no Power\n\r");
                                                tcp_write(tpcb, p->payload,18, 1);
                                                pbuf_free(p->payload);
                                                }
                                                else if(status[i]==0x1){
                                                openpc(i);
                                                strcpy(data,"Pc opened\n\r");
                                                tcp_write(tpcb, p->payload,12, 1);
                                                pbuf_free(p->payload);
                                                                                }
                                                else if(status[i]==0x3){
                                                strcpy(data,"Pc already open\n\r");
                                                tcp_write(tpcb, p->payload,18, 1);
                                                pbuf_free(p->payload);
                                                }
```

```
                                                    else{
                                                    strcpy(data,"ERROR!!!\n\r");
                                                    tcp_write(tpcb, p->payload,10, 1);
                                                    pbuf_free(p->payload);
                                                    }                       }
        }
        else{
                pbuf_free(p->payload);
                strcpy(data,"Invalid Command\n\r");
                tcp_write(tpcb, p->payload,17, 1);
                pbuf_free(p);
        }

        strcpy(data,"[User]: ");
         ''     pbuf_free(p->payload);
                tcp_write(tpcb, p->payload,8, 1);
                pbuf_free(p);

        return 0;
}

err_t accept_callback(void *arg, struct tcp_pcb *newpcb, err_t err)
{
        static int connection = 1;

        //xil_printf("Connection (%d) Accepted\n\r", connection);

        /* set the receive callback for this connection */
        tcp_recv(newpcb, recv_callback);

        /* just use an integer number indicating the connection id as the
           callback argument */
        tcp_arg(newpcb, (void*)connection);

        /* increment for subsequent accepted connections */
        connection++;

        return ERR_OK;
}


int start_application()
{
        struct tcp_pcb *pcb;
        err_t err;
        unsigned port = 23;

        /* create new TCP PCB structure */
        pcb = tcp_new();
        if (!pcb) {
                xil_printf("Error creating PCB. Out of Memory\n\r");
                return -1;
        }

        /* bind to specified @port */
        err = tcp_bind(pcb, IP_ADDR_ANY, port);
        if (err != ERR_OK) {
                xil_printf("Unable to bind to port %d: err = %d\n\r", port, err);
                return -2;
        }

        /* we do not need any arguments to callback functions */
        tcp_arg(pcb, NULL);

        /* listen for connections */
        pcb = tcp_listen(pcb);
        if (!pcb) {
                xil_printf("Out of memory while tcp_listen\n\r");
                return -3;
        }

        /* specify callback to use for incoming connections */
        tcp_accept(pcb, accept_callback);

        xil_printf("Server started @ port %d\n\r", port);

        return 0;
}
```

```
int getstatus(int pcnumber){
int pcstatus;
volatile int Delay;
XGpio_Initialize(&GpioOutput,XPAR_LEDS_6BIT_DEVICE_ID);
XGpio_SetDataDirection(&GpioOutput, LED_CHANNEL, 0x0);
XGpio_DiscreteWrite(&GpioOutput, LED_CHANNEL, 0x0);
XGpio_DiscreteWrite(&GpioOutput, LED_CHANNEL,pcnumber);
for (Delay = 0; Delay < LED_DELAY; Delay++);
        XGpio_Initialize(&GpioInput,XPAR_DIP_SWITCHES_4BIT_DEVICE_ID);
        XGpio_SetDataDirection(&GpioInput, LED_CHANNEL, 0xFFFFFFFF);
        pcstatus = XGpio_DiscreteRead(&GpioInput, LED_CHANNEL);
        XGpio_DiscreteClear(&GpioOutput, LED_CHANNEL,pcnumber);
        return pcstatus;
        }
int cpc(int clpseid){
        xil_printf("\n\r\n\r-----lwIP TCP Control & Monitor Server ------\n\r");
        volatile int Delay;
        XGpio_Initialize(&GpioOutput,XPAR_LEDS_6BIT_DEVICE_ID);
        XGpio_SetDataDirection(&GpioOutput, LED_CHANNEL, 0x0);
        XGpio_DiscreteWrite(&GpioOutput, LED_CHANNEL, 0x0);
        XGpio_DiscreteWrite(&GpioOutput, LED_CHANNEL,closeid);
        for (Delay = 0; Delay < LED_DELAY2; Delay++);
        XGpio_DiscreteClear(&GpioOutput, LED_CHANNEL,closeid);

return 0;
}
int openpc(int openid){
        xil_printf("\n\r\n\r-----lwIP TCP Control & Monitor Server ------\n\r");
        volatile int Delay;
        XGpio_Initialize(&GpioOutput,XPAR_LEDS_6BIT_DEVICE_ID);
        XGpio_SetDataDirection(&GpioOutput, LED_CHANNEL, 0x0);
        XGpio_DiscreteWrite(&GpioOutput, LED_CHANNEL, 0x0);
        XGpio_DiscreteWrite(&GpioOutput, LED_CHANNEL,openid);
        for (Delay = 0; Delay < LED_DELAY; Delay++);
        XGpio_DiscreteClear(&GpioOutput, LED_CHANNEL,openid);

return 0;
}
```

ΙΙ.Β Κώδικας υλοποίησης ιστοσελίδας διαχείρισης.

```
<?php
$us=$_POST["text1"];
$ps=$_POST["text2"];
if($us=="admin"&& $ps=="1010"){
$expire=time()+3600;
setcookie("admin", "Administrator", $expire);
echo "Login success.";
echo "<html>
<head><meta http-equiv='refresh' content='1;url=../php_files/ckcookie.php'></head></html>";
}
else{
echo "Login error: <font color='red'>user name</font> or <font color='red'>password</font> unknown.";
echo "<html>
<head><meta http-equiv='refresh' content='1;url=../php_files/ckcookie.php'></head></html>";
}
?>
```

```
<?php
echo "<link href='../html_files/styles.css' rel='stylesheet' type='text/css'/> ";
if(isset($_COOKIE["admin"])){
echo "<div id='login' class='boxed' >
                        <h2 class='title'>Welcome,". $_COOKIE["admin"]." </h2>
                        <div class='content2'></div></div>";
}
else{
echo "<div id='login' class='boxed' >
                        <h2 class='title'>Administrator Login </h2>
                        <div class='content2'>
                                <form id='form1' method='post' action='../php_files/login.php'>
                                        <fieldset>
                                        <legend>Login</legend>
                                        <label for='text1'>Login Name:</label>
                                        <input id='text1' type='text' name='text1' value='' />
                                        <label for='text2'>Password:</label>
                                        <input id='text2' type='password' name='text2' value='' />
                                        <input id='submit1' type='submit' name='submit1' value='Login' />
                                        </fieldset>
                                </form>
                        </div>
                </div>";
                }
?>
```

```
        <?php
setcookie ("admin", "", time() - 3700);
echo"Log out success";
echo"<html>
<head><meta http-equiv='refresh' content='1;url=ckcookie.php'></head></html>";
?>
```

```
        <html>
<head>
<meta http-equiv="refresh" content="300;url=../html_files/loading.html">
<style>
table {border:solid 1px; background-color: white; opacity:0.7;}
td {border:solid 1px; height:70px; width:70px; text-align:center;}
tr{border:solid 1px;}
input{height:70px; width:70px;}
.button{background:url('../images/button.png') no-repeat top;}
</style>
<script type="text/javascript">
</script>
<?php
function checkstatus($status,$pcnum)
{
if(!strncmp($status,"OK!",3)){
ok($pcnum);
}
```

```
elseif(!strncmp($status,"Only Power",10)){
onlypower($pcnum);
}
elseif(!strncmp($status,"No Power",8)){
nopower();
}
else
error();

}
function ok($idclose){
echo "<td bgcolor='#0FF00'><img src='../images/check4.gif' alt='Ok' /></td>";
echo "<td bgcolor='#0FF00'><img src='../images/check4.gif' alt='Ok' /></td>";
echo "<td><form method='post' action='close.php'>"."<input type='hidden' value='".$idclose."' name='pcclose'/>"."<input type='Submit'
value='Close'/></form></td>";
}
function onlypower($idopen){
echo "<td bgcolor='#0FF00'><img src='../images/check4.gif' alt='Ok' /></td>";
echo "<td bgcolor='#FF0000'><img src='../images/funcyred2.png' alt='Closed' /></td>";
echo "<td><form method='post' action='open.php'>"."<input type='hidden' value='".$idopen."' name='pcopen'/>"."<input type='Submit'
value='Open'/></form></td>";
}
function nopower(){
echo "<td bgcolor='#FF0000'><img src='../images/funcyred2.png' alt='No Power' /></td>";
echo "<td bgcolor='#FF0000'><img src='../images/funcyred2.png' alt='No Power' /></td>";
echo "<td><input type='submit' value='No Power'/></td>";
}
function error(){
echo "<td><img src='../images/error.png' alt='Error' /></td>";
echo "<td><img src='../images/error.png' alt='Error' /></td>";
echo "<td><input type='submit' value='Error'/></td>";
}
 ?>
</head>
<body>
<?php
if (isset($_COOKIE["admin"])){
$str = exec("ping -n 1 -w 1 192.168.1.10", $input, $result);
if ($result == 0){
require_once "PHPTelnet.php";
$telnet = new PHPTelnet();
$result = $telnet->Connect('192.168.1.10','paul','paul');
echo "<center>
<table>
<tr>
<td><strong>Pc:num</strong></td>
<td><strong>Power</strong></td>
<td><strong>Status</strong></td>
<td><strong>Control</strong></td>
</tr>";
if ($result == 0) {
$i=0;
for($i=1;$i<6;$i++){
$istring = (string)$i;
$command="status.";
$command.=$istring;
$telnet->DoCommand($command, $result);
echo "<tr><td>PC:"."$i"."</td>";
checkstatus($result,$istring);
echo "</tr>";
}
$telnet->Disconnect();
}
}else{
echo "</br></br><center> <b>Board has No Power</b> or <b>Network cable unplugged</b> click below for troubleshooting.</br>";
echo "<a href='../html_files/troubleshooting.html'target='theframe'>Troubleshooting</a></center>";
}
}
else{echo"Please Login.";}
?>
</body>
</html>
```

```
<html>
<head><meta http-equiv="refresh" content="1;url=mysitebeta2.php"></head>
<body>

<?php
$i=$_POST["pcopen"];
$istring = (string)$i;
$command="open.";
$command.=$istring;
require_once "PHPTelnet.php";
$telnet = new PHPTelnet();
$result = $telnet->Connect('192.168.1.10','paul','paul');
if ($result == 0) {
$telnet->DoCommand($command, $result);
}
$telnet->Disconnect();

echo "</br></br></br></br><center><img src='../images/loading.gif'/></center>";
?>
</body>
</html>


<html>
<head><meta http-equiv="refresh" content="1;url=mysitebeta2.php"></head>
<body>

<?php
$i=$_POST["pcclose"];
$istring = (string)$i;
$command="close.";
$command.=$istring;
require_once "PHPTelnet.php";
$telnet = new PHPTelnet();
$result = $telnet->Connect('192.168.1.10','paul','paul');
if ($result == 0) {
$telnet->DoCommand($command, $result);
}
$telnet->Disconnect();

echo "</br></br></br></br><center><img src='../images/loading.gif'/></center>";
?>
</body>
</html>

<html>
<head>
<?php
if (isset($_COOKIE["admin"])){
echo"Type the number of the Pc you want to close.";
echo "<form action='../php_files/close.php' method='post' ><input type='text' maxlength='2' size='4' name='pcclose'/><input
type='Submit' value='Close'/></form>";
echo"Type the number of the Pc you want to open.";
echo "<form action='../php_files/open.php' method='post' ><input type='text' maxlength='2' size='4'name='pcopen'/><input
type='Submit' value='Open'/></form>";
echo"Close all Pcs.";
echo "<form action='../php_files/close.php' method='post' ><input type='hidden' value='all' name='pcclose'/><input type='Submit'
value='Close All'/></form>";
echo"Open all Pcs.";
echo "<form action='../php_files/open.php' method='post' ><input type='hidden' value='all' name='pcopen'/><input type='Submit'
value='Open All'/></form>";
}
else{
echo "Please Login.";
}
?>
</head>
<body>

</body>
</html>
```