ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΣΤΗ ΜΙΚΡΟΗΛΕΚΤΡΟΝΙΚΗ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σχεδιασμός και υλοποίηση αναδιαμορφούμενου ενσωματωμένου συστήματος μέτρησης αιθητήρων χωρητικότητας σε FPGA

Ιωάννης Ε. Ζαφειράκης

***Επιβλέπων*: Χατζανδρούλης Σταύρος,** Ερευνητής, ΕΚΕΦΕ “ΔΗΜΟΚΡΙΤΟΣ”

ΑΘΗΝΑ

ΟΚΤΩΒΡΙΟΣ 2016

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σχεδιασμός και υλοποίηση αναδιαμορφούμενου ενσωματωμένου συστήματος μέτρησης αιθητήρων χωρητικότητας σε FPGA

Ιωάννης Ε. Ζαφειράκης

Α.Μ.: ΜΜ250

ΕΠΙΒΛΕΠΩΝ: **Χατζανδρούλης Σταύρος,** Ερευνητής, ΕΚΕΦΕ "ΔΗΜΟΚΡΙΤΟΣ"

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ: **Κυριάκης-Μπιτζάρος Ευστάθιος,** Καθηγητής, ΤΕΙ ΠΕΙΡΑΙΑ
Αραπογιάννη Αγγελική, Καθηγήτρια, ΕΚΠΑ

ΟΚΤΩΒΡΙΟΣ 2016

ΠΕΡΙΛΗΨΗ

Οι έξυπνοι αισθητήρες χρησιμοποιούνται όλο και πιο πολύ στις μέρες μας για την παρακολούθηση του περιβάλλοντος. Ουσιαστικά ένας έξυπνος αισθητήρας είναι ένα σύστημα το οποίο περιλαμβάνει εκτός από τα αισθητήρια και έναν επεξεργαστή ο οποίος παρέχει τη δυνατότητα τοπικής επεξεργασίας των μετρήσεων. Στο πλαίσιο της διπλωματικής εργασίας υλοποιήθηκε ένα έξυπνο και ευέλικτο ενσωματωμένο σύστημα για την μέτρηση αισθητήρων χωρητικότητας. Στο σύστημα εκτός από το κύκλωμα διεπαφής των αισθητήρων έχει ενσωματωθεί ένας επεξεργαστής LEON3 με τα απαραίτητα περιφερειακά και έχει εγκατασταθεί λειτουργικό σύστημα Linux. Η υλοποίηση έγινε σε FPGA της σειράς CYCLON της ALTERA στο αναπτυξιακό σύστημα της Terasic DE2-115.

Για την μέτρηση των αισθητήρων έχει υλοποιηθεί ένα κύκλωμα διεπαφής το οποίο μετατρέπει της μεταβολές της χωρητικότητας σε μεταβολές συχνότητας με χρήση ενός ειδικά διαμορφωμένου ταλαντωτή δακτυλίου. Ακολούθως, χρησιμοποιείται ένας προγραμματιζόμενος μετρητής συχνότητας ο οποίος διαθέτει μεταβλητό χρονικό παράθυρο μέτρησης ώστε να παρέχει ευελιξία ως προς το χρόνο μέτρησης, την ακρίβεια και το εύρος των μετρούμενων συχνοτήτων. Το κύκλωμα διεπαφής έχει συνδεθεί στον εσωτερικό δίαυλο δεδομένων (AMBA bus) του επεξεργαστή LEON3 ώστε να συμπεριφέρεται ως ένα τυπικό περιφερειακό του επεξεργαστή και να επιτυγχάνεται εύκολη και αποδοτική διαχείρισή του από το λογισμικό της εφαρμογής. Στο δίαυλο AMBA έχουν συνδεθεί και άλλα περιφερειακά όπως για παράδειγμα μια SVGA οθόνη επαφής (touch screen), μια μονάδα δικτύου (ETHERNET) και ένα πληκτρολόγιο τα οποία προσδίδουν στο συνολικό σύστημα επιπλέον δυνατότητες και ευελιξία. Με το ενσωματωμένο λειτουργικό σύστημα Linux ο χρήστης του συστήματος μπορεί να χρησιμοποιεί ένα καθιερωμένο περιβάλλον για την επεξεργασία των μετρήσεων και την επικοινωνία με τους αισθητήρες. Ο χρήστης μπορεί να παρατηρεί τα αποτελέσματα στην SVGA οθόνη ή να εισάγει εντολές επεξεργασίας από το πληκτρολόγιο. Ταυτόχρονα υπάρχει η δυνατότητα απομακρυσμένης σύνδεσης με το σύστημα και μεταφοράς των αποτελεσμάτων σε ένα απομακρυσμένο υπολογιστή. Έχουν υλοποιηθεί προγράμματα σε γλώσσα C για την επεξεργασία των μετρήσεων και για τον έλεγχο του κυκλώματος διεπαφής. Ένα πρόγραμμα το οποίο υλοποιήθηκε αξιοποιεί την “touch screen” λειτουργία της οθόνης, ώστε να μην χρειάζεται απαραίτητα συνδεδεμένο πληκτρολόγιο στο σύστημα. Με το πρόγραμμα αυτό μπορεί να γίνει βαθμονόμηση του αισθητήρα και να υπολογίζεται η μέση τιμή και η διασπορά των μετρήσεων. Για να επαληθευθεί η ορθή λειτουργία του συστήματος ελήφθησαν μετρήσεις με χωρητικούς αισθητήρες αερίων οι οποίοι αποτελούνται από διαπλεκόμενα (interdigitated) ηλεκτρόδια και ένα στρώμα πολυμερούς, του οποίου οι ιδιότητες μεταβάλλονται με την απορρόφηση συγκεκριμένων αερίων. Η απόκριση του προτεινόμενου συστήματος για διάφορες συγκεντρώσεις αναλυτών συγκρίθηκε με τις μετρήσεις των ίδιων αισθητήρων με σύστημα γέφυρας και προέκυψε ικανοποιητική σύμπτωση. Η ευαισθησία του συστήματος είναι, επίσης, ικανοποιητική γιατί δίνει τη δυνατότητα μέτρησης πολύ μικρών μεταβολών της χωρητικότητας οι οποίες αντιστοιχούν σε μεταβολή μερικών δεκάδων Hz στη συχνότητα ταλάντωσης.

Το σύστημα που υλοποιήθηκε μπορεί να χρησιμοποιηθεί σε πλήθος εφαρμογών και να προσαρμοστεί σε διαφορετικά περιβάλλοντα. Με την ενσωμάτωση του επεξεργαστή LEON3 ο οποίος είναι ευέλικτος και παραμετροποιήσιμος μπορούν να γίνουν εύκολα προσθήκες υλικού (hardware) και να προσαρμοστεί κατάλληλα το λογισμικό ώστε να προστεθούν επιπλέον λειτουργίες. Συνολικά, στην διπλωματική εργασία παρουσιάζεται ένα ευέλικτο, αυτόνομο, εύχρηστο και αποδοτικό “έξυπνο” σύστημα για την μέτρηση

αισθητήρων χωρητικότητας, το οποίο περιλαμβάνει πολλαπλές λειτουργίες επεξεργασίας και επικοινωνίας και έχει πάρα πολλές δυνατότητες εξέλιξης. Η παρούσα υλοποίηση μπορεί να αποτελέσει οδηγό για παρόμοιες υλοποιήσεις στο μέλλον και παρουσιάζει τις προοπτικές των έξυπνων συστημάτων σε συνδυασμό με αισθητήρες.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: έξυπνα συστήματα αισθητήρων, ενσωματωμένα συστήματα

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: LEON3, FPGA, VHDL, EMBEDDED LINUX, “έξυπνοι” αισθητήρες

ABSTRACT

Smart sensors are used increasingly nowadays for environmental monitoring. A smart sensor is a system which includes apart from the sensor element a processor which enables local processing of the measurements. Within the framework of this master thesis a smart and flexible embedded system for measuring capacitance sensors has been designed and implemented. The system developed comprises of the sensor interface circuit, a LEON3 processor equipped with the necessary peripherals and a Linux operating system. The system implementation is done using a CYCLON series ALTERA FPGA, the Terasic DE2-115 development board and a custom board hosting the sensor elements.

For the measurement of the sensors an interface circuit which converts the capacitance changes in frequency changes by using a specially designed ring oscillator is implemented. Subsequently, a programmable frequency counter featuring a variable measurement time window is used in order to provide flexibility to the measurement time, the accuracy and the range of elaborated frequencies. The interface circuit is connected to the internal data bus (AMBA bus) of LEON3 processor in order to behave as a typical peripheral of the processor and to achieve easy and efficient management from the application software. On the AMBA bus are also connected the standard peripherals of the system, namely, a SVGA touch screen display, a network module (ETHERNET) and a keyboard which give to the system additional capabilities and flexibility. With the embedded Linux operating system the system user can use a standard environment for the processing of the measurements and the communication with the sensors. The user can observe the results on the SVGA display or introduce editing commands from the keyboard. At the same time there is the possibility of a remote connection to the system and the transfer of the results to a remote computer. Programs written in the C programming language are developed for the processing of the measurements and the control of the interface circuit. A stand-allone system has been demonstrated exploiting the capabilities of the "touch screen" display mode and eliminating the need for the presence of a keyboard. The developed software module has the ability to perform sensor calibration and calculation of the mean value and the deviation of the measurements for all channels. To verify the system operation an array of four gas sensors has been used. The capacitive gas sensors consist of interdigitated electrodes and a polymer layer, whose electrical properties vary with the absorption of certain gases (analytes). The response of the proposed system compared to the capacitance measurements using a bridge for various concentrations of analytes has showed satisfactory agreement. The sensitivity of the system is also satisfactory as it can measure very small changes of capacitance resulting to a change of few tens of Hz in oscillation frequency.

The system that was implemented can be used in numerous applications and can be adapted to various environments. By integrating the LEON3 processor which is flexible and configurable, hardware alterations can be easily made and the software can be adapted in order to add extra functionality. Overall, this thesis presents a flexible, stand-alone, easy-to-use and efficient smart system for the measurement of capacitive sensors, which includes multiple processing and communication functions and has many development possibilities. This implementation can be used as a guide for similar implementations in the future and shows the prospects of smart sensor systems.

SUBJECT AREA: smart sensor systems, embedded systems

KEYWORDS: LEON3, FPGA, VHDL, EMBEDDED LINUX, smart sensors

ΕΥΧΑΡΙΣΤΙΕΣ

Αρχικά, θα ήθελα να ευχαριστήσω όλους τους καθηγητές του “ΔΙΑΤΜΗΜΑΤΙΚΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΣΤΗ ΜΙΚΡΟΗΛΕΚΤΡΟΝΙΚΗ” για τις γνώσεις που μου παρείχαν.

Θα ήθελα να ευχαριστήσω ξεχωριστά τον επιβλέποντα της διπλωματικής κ. Σταύρο Χατζανδρούλη, ερευνητή στο Ινστιτούτο Νανοεπιστήμης και Νανοτεχνολογίας του ΕΚΕΦΕ “ΔΗΜΟΚΡΙΤΟΣ”, για την δυνατότητα που μου έδωσε να ασχοληθώ με το αντικείμενο αυτής της διπλωματικής εργασίας και για την βοήθεια που μου προσέφερε καθ’όλη την διάρκεια εκπόνησής της. Ευχαριστώ, επίσης, τον κ. Ευστάθιο Κυριάκη-Μπιτζάρο, καθηγητή του Τμήματος Ηλεκτρονικών Μηχανικών του ΤΕΙ Πειραιά, για την καθοδήγηση και την άμεση και ουσιαστική βοήθεια που μου προσέφερε. Ακόμα, ευχαριστώ την κ. Μυρτώ-Κυριακή Φιλιππίδου, υποψήφια διδάκτορα, για την βοήθειά της στις μετρήσεις που ελήφθησαν στο ΕΚΕΦΕ “ΔΗΜΟΚΡΙΤΟΣ”.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου και ειδικότερα την μητέρα μου που με παρότρυνε για την ολοκλήρωση του μεταπτυχιακού και για την υλική αλλά και ψυχολογική υποστήριξη που μου παρείχε κατά την διάρκεια αυτών των δύο χρόνων.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

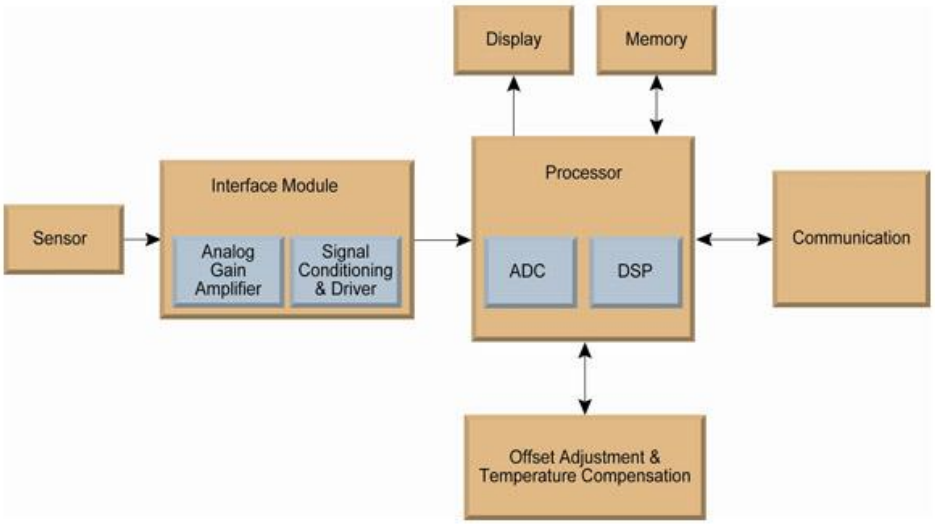
ΠΡΟΛΟΓΟΣ

Η διπλωματική υλοποιήθηκε σε συνεργασία με το ΤΕΙ Πειραιά και το ερευνητικό κέντρο ΕΚΕΦΕ “Δημόκριτος”. Η υλοποίηση έγινε, στο μεγαλύτερο μέρος της, στο τμήμα Ηλεκτρονικών Μηχανικών ΤΕ της Σχολής Τεχνολογικών Εφαρμογών του ΤΕΙ Πειραιά και οι πειραματικές μετρήσεις ελήφθησαν σε εργαστήριο του Ινστιτούτου Νανοεπιστήμης και Νανοτεχνολογίας του ΕΚΕΦΕ “ΔΗΜΟΚΡΙΤΟΣ”. Η διπλωματική αυτή μπορεί να θεωρηθεί ως συνέχεια της πτυχιακής εργασίας του κ.Παύλου Σπυρίδωνα, "Σχεδίαση και Υλοποίηση σε FPGA/CPLD ηλεκτρονικού κυκλώματος για τη μέτρηση αισθητήρων τύπου χωρητικότητας" [31].

1. ΕΙΣΑΓΩΓΗ

1.1 Εισαγωγή στα Έξυπνα συστήματα αισθητήρων

Οι αισθητήρες και τα συστήματα αισθητήρων παίζουν σημαντικό ρόλο στο να μπορούμε να αντιληφθούμε το περιβάλλον. Υπάρχει στις μέρες μας μια ανάπτυξη και στροφή του τομέα της τεχνολογίας των αισθητήρων προς τα “έξυπνα” συστήματα αισθητήρων. Ο ορισμός ενός “έξυπνου” συστήματος ποικίλλει. Σε γενικές γραμμές ένας “έξυπνος” αισθητήρας είναι ο συνδυασμός ενός στοιχείου αισθητήρα με δυνατότητες επεξεργασίας, οι οποίες παρέχονται από έναν μικροεπεξεργαστή. Ουσιαστικά, οι “έξυπνοι” αισθητήρες είναι στοιχεία αισθητήρων (sensing elements) με ενσωματωμένη “νοημοσύνη”. Το σήμα του αισθητήρα “τροφοδοτεί” τον μικροεπεξεργαστή, ο οποίος επεξεργάζεται τα δεδομένα και παρέχει στον εξωτερικό χρήστη το αποτέλεσμα της μέτρησης. Στο παρακάτω σχήμα φαίνεται ένα τυπικό διάγραμμα βαθμίδων ενός “έξυπνου” αισθητήρα με μικροεπεξεργαστή (σχήμα 1.1).



Σχήμα 1.1: Σχηματικό διάγραμμα “έξυπνου” αισθητήρα με μικροεπεξεργαστή [1]

Η θεμελιώδης ιδέα ενός “έξυπνου” συστήματος περιλαμβάνει την ενσωμάτωση μικροεπεξεργαστών με την τεχνολογία αισθητήρων, την επεξεργασία των δεδομένων, την έξοδο των αποτελεσμάτων στον χρήστη, την σημαντική βελτίωση της απόδοσης και την διεύρυνση των λειτουργιών του συστήματος αισθητήρων. Ο “έξυπνος” αισθητήρας έχει διάφορα επίπεδα επεξεργασίας, όπως: η συλλογή του σήματος από τα στοιχεία αισθητήρων, η επιβεβαίωση και η ερμηνεία των δεδομένων, η μεταφορά των επεξεργασμένων σημάτων και η εμφάνισή τους. Κατ’ αρχάς, το σήμα των αισθητήρων μετατρέπεται από αναλογικό σε ψηφιακό. Η ενσωματωμένη “νοημοσύνη” παρακολουθεί συνεχώς τα διακριτά στοιχεία αισθητήρων, εξετάζει τα δεδομένα που παρέχονται και κάνει τις απαραίτητες επεξεργασίες σε αυτά, ακόμα μπορεί να ελέγχει περιοδικά τη σωστή λειτουργία των αισθητήρων. Τα επεξεργασμένα δεδομένα, τελικά, γίνονται πληροφορία η οποία μπορεί να μεταφερθεί στον εξωτερικό χρήστη. Ο χρήστης, τέλος, μπορεί να επιλέξει την πολυπλοκότητα των δεδομένων που θα λαμβάνει [2].

Ένα σημαντικό χαρακτηριστικό των “έξυπνων” αισθητήρων είναι ότι τα δεδομένα που παρέχονται στον χρήστη έχουν υψηλή αξιοπιστία. Επίσης, μπορούν να προστεθούν “έξυπνα” χαρακτηριστικά όπως: αυτό-βαθμονόμηση (self-calibration), αυτόματος έλεγχος της υγείας του συστήματος (self-health) και πρόσθετες μετρήσεις (αυτόματος μηδενισμός (auto-zero), βαθμονόμηση (calibration) των αισθητήρων, έλεγχος της θερμοκρασίας και της πίεσης, διορθώσεις στην σχετική υγρασία κ.α.). Η ικανότητα του “έξυπνου” αισθητήρα να πραγματοποιεί εσωτερική επεξεργασία επιτρέπει στο σύστημα,

όχι μόνο να παρέχει στον χρήστη επεξεργασμένα δεδομένα, αλλά και να έχει επίγνωση της κατάστασης, καθώς και να ελέγχει την αξιοπιστία των δεδομένων που παράγονται. Επίσης, ένα “έξυπνο” σύστημα αισθητήρων μπορεί να βελτιστοποιεί την απόδοση των επιμέρους αισθητήρων και να οδηγεί σε καλύτερη κατανόηση των αποτελεσμάτων, της μέτρησης και τελικά του περιβάλλοντος στο οποίο λαμβάνεται η μέτρηση. Συνολικά, ο συνδυασμός του μικροεπεξεργαστή με τον αισθητήρα επιτρέπει τη σχεδίαση ενός συστήματος το οποίο προσαρμόζεται σε ένα μεταβαλλόμενο περιβάλλον για μια εφαρμογή ή επιτρέπει την μετατροπή (modification) του συστήματος ώστε να καλύπτει τις ανάγκες ευρέως φάσματος εφαρμογών [2].

Μια σημαντική εξέλιξη των έξυπνων αισθητήρων είναι η νέα γενιά έξυπνων αισθητήρων οι οποίοι μπορούν να δικτυωθούν. Αυτά τα δίκτυα αισθητήρων έχουν την ικανότητα επιμέρους αυτοπροσδιορισμού (self-identification) και επιτρέπουν τον επανα-προγραμματισμό του συστήματος, αν είναι απαραίτητο. Η επικοινωνία μεταξύ των διαφόρων αισθητήρων μπορεί να επιτευχθεί είτε ανάμεσα σε ένα “έξυπνο” αισθητήρα και ένα κόμβο επικοινωνίας (communication hub), είτε ανάμεσα σε διαφορετικούς έξυπνους αισθητήρες. Αυτές οι λειτουργίες παρέχουν στο σύστημα μεγαλύτερη αξιοπιστία και δυνατότητες. Επιπλέον, η πληροφορία μπορεί να διαμοιραστεί πιο γρήγορα, αξιόπιστα και αποδοτικά [2].

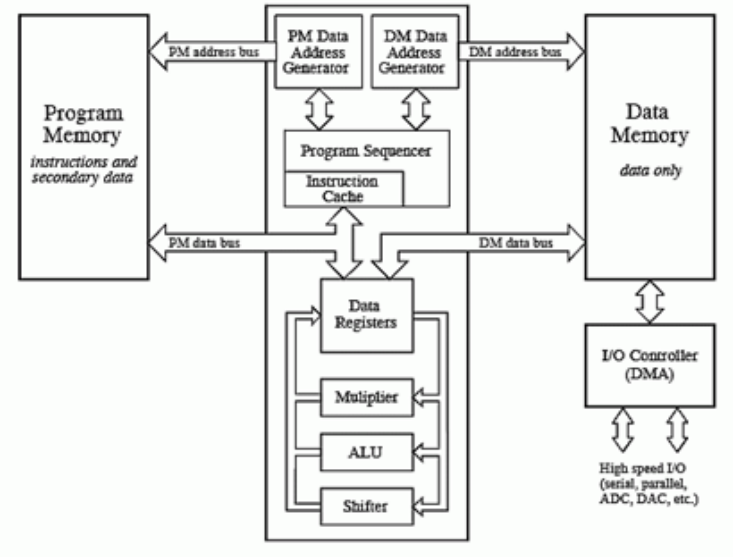
Ένας στόχος στην ανάπτυξη συστημάτων “έξυπνων” αισθητήρων είναι η υλοποίηση ευέλικτων συστημάτων στα οποία οι πληροφορίες παρέχονται στον χρήστη όποτε αυτός τις χρειάζεται, όπως και σε οποιαδήποτε μορφή αυτός επιθυμεί, ανάλογα με την εφαρμογή. Ο στόχος της έρευνας πάνω στους “έξυπνους” αισθητήρες είναι η ανάπτυξη συστημάτων αισθητήρων τα οποία παρέχουν στον χρήστη την πληροφορία/δεδομένα που απαιτούνται ώστε να πάρει τις σωστές αποφάσεις [2].

1.2 Συστήματα αισθητήρων σε FPGA

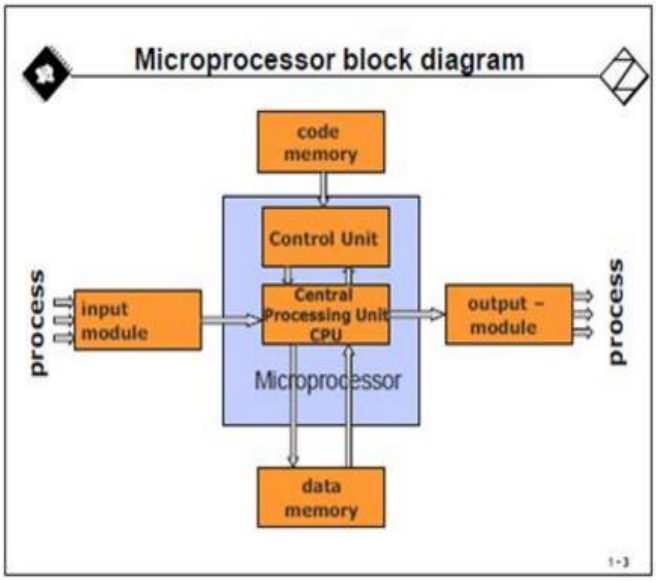
Η σύγχρονη τάση στην εξέλιξη των συστημάτων αισθητήρων έγκειται στην αναζήτηση μεθόδων για περισσότερη ακρίβεια και ανάλυση, ενώ την ίδια στιγμή να μειώνεται το μέγεθος και η κατανάλωση ενέργειας [3]. Η χρήση των συσκευών FPGAs παρέχει την δυνατότητα επανα-προγραμματιζόμενου υλικού (reprogrammable hardware) , το οποίο μπορεί να βοηθήσει στην υλοποίηση ενός αναδιαρθρώσιμου (reconfigurable) συστήματος αισθητήρων. Αυτή η δυνατότητα προσαρμογής επιτρέπει την υλοποίηση πολύπλοκων εφαρμογών, χρησιμοποιώντας μερική αναδιάρθρωση, με χαμηλή κατανάλωση ισχύος. Για εφαρμογές υψηλών απαιτήσεων οι συσκευές FPGAs προτιμώνται λόγω της υψηλής αποδοτικότητας που παρέχουν ως αποτέλεσμα της αρχιτεκτονικής ευελιξίας τους (παραλληλισμό, μνήμη πάνω στο τσιπ (on-chip memory) κ.τ.λ.), της δυνατότητας αναδιάρθρωσης τους και της πολύ υψηλής απόδοσης τους στην υλοποίηση αλγορίθμων. Συνολικά οι συσκευές FPGAs έχουν βελτιώσει την απόδοση των συστημάτων αισθητήρων και έχουν οδηγήσει σε μια σημαντική αύξηση της χρήσης αυτών των συστημάτων στο πεδίο των εφαρμογών [3][4].

Η ικανότητα επεξεργασίας σε συστήματα αισθητήρων κατά κανόνα βασίζεται σε επεξεργαστές ψηφιακού σήματος (DSP) ή προγραμματιζόμενους μικροεπεξεργαστές. Ο σκοπός του μικροεπεξεργαστή είναι να δέχεται δεδομένα ως είσοδο, να τα επεξεργάζεται και να εμφανίζει τα αποτελέσματα στην έξοδο. Ο μικροεπεξεργαστής είναι βελτιστοποιημένος ως προς την χρήση μνήμης, την γενική εκτέλεση εντολών, το μειωμένο κόστος και την χαμηλή κατανάλωση ενέργειας. Από την άλλη ο DSP επεξεργαστής είναι ένα ειδικό είδος μικροεπεξεργαστή ο οποίος είναι ειδικά σχεδιασμένος να εκτελεί πολύπλοκες αριθμητικές πράξεις με μεγάλους αριθμούς, γρήγορα και αποδοτικά. Οι DSP επεξεργαστές χρησιμοποιούνται συνήθως σε εφαρμογές επεξεργασίας εικόνας, αναγνώρισης φωνής και στις τηλεπικοινωνίες. Γενικά οι DSP επεξεργαστές είναι πιο αποδοτικοί στην εκτέλεση αριθμητικών πράξεων και

ειδικά στο πολλαπλασιασμό. Στο σχήμα 1.2 και 1.3 φαίνεται το τυπικό διάγραμμα βαθμίδων ενός DSP επεξεργαστή και ενός μικροεπεξεργαστή αντίστοιχα [5][6]



Σχήμα 1.2: Τυπική αρχιτεκτονική επεξεργαστή ψηφιακού σήματος [5]



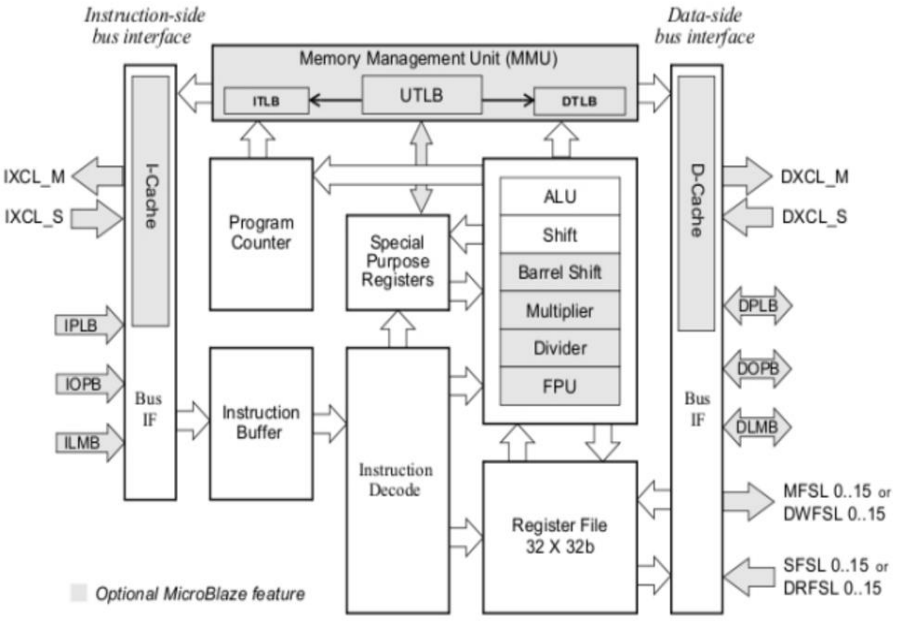
Σχήμα 1.3: Τυπική αρχιτεκτονική ενός μικροεπεξεργαστή [6]

Η χρήση FPGA παρέχει συγκεκριμένη τεχνολογία υλικού, η οποία επιτρέπει μέσω του επαναπρογραμματισμού την υλοποίηση αναδιαρθρώσιμου συστήματος αισθητήρων. Η μερική αναδιάρθρωση είναι η διαδικασία μετατροπής ορισμένων κομματιών “λογικής” τα οποία υλοποιούνται σε ένα FPGA. Με αυτό τον τρόπο το κύκλωμα μπορεί να μετατραπεί ώστε να προσαρμοστεί στις ανάγκες διαφορετικών διεργασιών. Αυτή η ικανότητα προσαρμογής επιτρέπει την υλοποίηση πολύπλοκων εφαρμογών με πολύ χαμηλή κατανάλωση ενέργειας.

Οι σημερινές δυνατότητες των αρχιτεκτονικών των συσκευών FPGA επιτρέπουν όχι μόνο την υλοποίηση απλών ακολουθιακών και συνδυαστικών κυκλωμάτων αλλά ακόμα και την ενσωμάτωση υψηλού επιπέδου επεξεργαστών μέσω λογισμικού (soft processors). Η χρήση αυτών των ενσωματωμένων επεξεργαστών προσφέρει σημαντικά πλεονεκτήματα στον σχεδιαστή, συμπεριλαμβανομένης της παραμετροποίησης (customization), του μετριασμού της απαξίωσης, της μείωσης του κόστους και της “επιτάχυνσης” του υλικού (hardware acceleration) [7]. Οι ενσωματωμένοι επεξεργαστές

για FPGA χρησιμοποιούν τα λογικά στοιχεία του FPGA για τη δημιουργία εσωτερικών στοιχείων μνήμης, διαύλους δεδομένων και ελέγχου (data και control buses), εσωτερικών και εξωτερικών περιφερειακών και ελεγκτών μνήμης (memory controllers).

Η Altera και η Xilinx (οι 2 μεγαλύτερες εταιρίες στην βιομηχανία των συσκευών FPGA) παρέχουν FPGA συσκευές οι οποίες έχουν ενσωματωμένους επεξεργαστές μέσα στο τσιπ του FPGA. Αυτού του είδους οι επεξεργαστές υλοποιούνται απευθείας στο υλικό (hard processors). Ένα παράδειγμα είναι η περίπτωση του Dual ARM Cortex-A9 επεξεργαστή στην συσκευή Zynq-7000 της Xilinx και ο ARM Cortex-A9 MPCore στις συσκευές CYCLONE V της Altera. Από την άλλη μεριά η αρχιτεκτονική των μικροεπεξεργαστών μέσω λογισμικού “κτίζεται” εξ’ ολοκλήρου χρησιμοποιώντας γλώσσα περιγραφής υλικού (HDL). Οι πιο γνωστοί επεξεργαστές μέσω λογισμικού είναι ο LEON3 από την Aeroflex Gaisler, ο οποίος έχει υλοποιηθεί με τη γλώσσα VHDL, είναι 32-μπιτ και βασίζεται στην αρχιτεκτονική SPARC V8, επίσης ο Nios II, ο οποίος είναι επίσης 32-μπιτ και είναι ειδικά σχεδιασμένος για χρήση σε συσκευές της Altera (firm επεξεργαστής) και ο Microblaze της Xilinx. Ο τελευταίος είναι ένας 32-μπιτ RISC Harvard αρχιτεκτονικής με πολύ πλούσιο σύνολο εντολών (instruction set), το οποίο είναι βελτιστοποιημένο για ενσωματωμένες εφαρμογές [8].



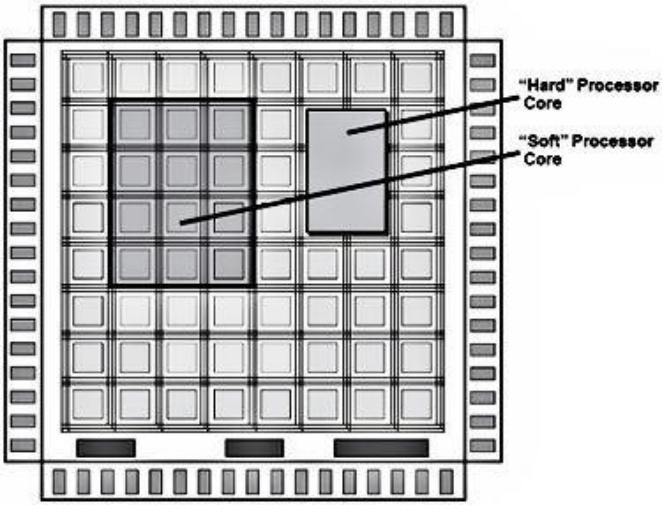
Σχήμα 1.4: Διάγραμμα βαθμίδων Microblaze επεξεργαστή [8]

Οι πόροι σε υλικό που υλοποιούνται σε ένα FPGA διαφέρουν σημαντικά ανάλογα με τον κατασκευαστή και τη συσκευή FPGA. Παρόλα αυτά, ένας μεγάλος αριθμός συσκευών περιλαμβάνουν δομικά στοιχεία (components) τα οποία είναι κατάλληλα για εφαρμογή σε συστήματα αισθητήρων. Οι περισσότερες συσκευές FPGA μπορούν να προσφέρουν αναδιαρθρώσιμα πρότυπα εισόδου/εξόδου (I/O standards) έτσι ώστε να επιτρέπεται σε ένα μεγάλο εύρος συσκευών να συνδέεται και να λειτουργεί σε διαφορετικά επίπεδα τάσης χωρίς την χρήση προσαρμογέα (adapter) ή μετατροπέα τάσης (voltage converter). Με αυτόν το τρόπο απλοποιείται σημαντικά η σχεδίαση και μειώνεται το κόστος. Για παράδειγμα, οι συσκευές Virtex-7 από την Xilinx παρέχουν διάφορα πρότυπα εισόδου/εξόδου όπως LVDS, HT, RSDS, BLVDS, διαφορικό SSTL, και διαφορικό HSTL τα οποία μπορούν να λειτουργούν σε διαφορετικά επίπεδα τάσης από 1.2-3.3V [9]. Κάποιες συσκευές FPGA περιλαμβάνουν ένα μεγάλο αριθμό αριθμητικών μπλοκ, τα οποία μπορεί να είναι είτε χαμηλής πολυπλοκότητας, όπως απλοί πολλαπλασιαστές, είτε πιο πολύπλοκα, όπως μονάδες DSP. Οι μονάδες DSP αποτελούνται από συνδυασμούς διαφόρων δομικών στοιχείων όπως πολλαπλασιαστές,

διαιρέτες, αθροιστές, συσσωρευτές (accumulators), καταχωρητές ολίσθησης (shift registers) κ.α. Μια μονάδα DSP βελτιώνει σημαντικά την απόδοση ενός FPGA και επιτρέπει την επίτευξη μεγαλύτερης παραγωγικότητας και ευελιξίας, όπως και την μείωση του κόστους και της κατανάλωσης ενέργειας. Για παράδειγμα, οι συσκευές της οικογένειας Spartan-7(από την Xilinx) έχουν μέχρι και 160 DSP μπλοκ, τα οποία υλοποιούν αποδοτικά λειτουργίες πολλαπλασιασμού και πολλαπλασιασμού-άθροισης. Κάθε μπλοκ περιλαμβάνει 25 x 18 πολλαπλασιαστές, έναν 48-μπιτ συσσωρευτή και έναν αθροιστή. Επιπλέον, οι εσωτερικές μνήμες παρέχουν πολύ υψηλές ταχύτητες σε σχέση με τις εξωτερικές μνήμες. Οι σημερινές συσκευές FPGA περιλαμβάνουν μεγάλα μεγέθη εσωτερικών μπλοκ μνήμης, για παράδειγμα, έως και 32,832kb εσωτερικής RAM στις συσκευές Virtex-6 της Xilinx[10]. Επίσης, μπορούν να υπάρχουν και άλλα είδη μνήμης, όπως ROM και καταχωρητές. Επιπλέον, ο σχεδιαστής μπορεί να σχεδιάσει και άλλες μνήμες όπως FIFO. Συνολικά, η μείωση του κόστους των συσκευών FPGA, οι αυξημένες δυνατότητές τους και η δυνατότητα βελτίωσης της απόδοσης των συστημάτων αισθητήρων έχουν οδηγήσει στην αύξηση της χρήση τους σε νέα πεδία εφαρμογών σχετιζόμενα με αισθητήρες. [3][4]

1.3 Ενσωματωμένοι επεξεργαστές σε FPGAs

Οι επεξεργαστές σε ένα FPGA είναι IPs και μπορούν να κατηγοριοποιηθούν στους εξής 3 τύπους ανάλογα με τον τρόπο υλοποίησης τους [7]: μέσω λογισμικού ανοικτού τύπου (soft), μέσω λογισμικού κλειστού τύπου (firm) και αυτοί που υλοποιούνται απευθείας στο υλικό (hard). Οι επεξεργαστές μέσω λογισμικού ανοικτού τύπου είναι υλοποιήσεις επεξεργαστών σε γλώσσα περιγραφής υλικού (HDL) και δεν απαιτείται για την υλοποίησή τους συγκεκριμένη αρχιτεκτονική FPGA. Αυτοί επεξεργαστές συνήθως έχουν χαμηλότερη απόδοση και είναι λιγότερο αποδοτικοί ως προς τους πόρους που καταναλώνουν. Οι επεξεργαστές μέσω λογισμικού κλειστού τύπου είναι επίσης HDL υλοποιήσεις αλλά έχουν βελτιστοποιηθεί για μια συγκεκριμένη FPGA αρχιτεκτονική. Ο Nios II της Altera και ο Microblaze της Xilinx είναι παραδείγματα κλειστού τύπου επεξεργαστών. Οι επεξεργαστές οι οποίοι υλοποιούνται απευθείας στο υλικό είναι προσαρμοσμένοι πάνω στο FPGA και είναι IP επιπέδου πύλης. Ο Virtex-II Pro της XILINX και ο Virtex-4 405 PowerPC core είναι παραδείγματα τέτοιων επεξεργαστών.



Σχήμα 1.5: Επεξεργαστής μέσω λογισμικού και απευθείας στο υλικό σε ένα FPGA [7]

1.3.1 Πλεονεκτήματα και μειονεκτήματα επεξεργαστών μέσω λογισμικού-ανοικτοί

Πλεονεκτήματα

- Υψηλό επίπεδο φορητότητας (portability)
- Πιο οικονομικοί

- Χαμηλού κόστους/ελεύθερες πηγές (open source) λόγω ευκολότερης υλοποίησης
- Προσαρμόζονται σχετικά εύκολα σε συγκεκριμένες αρχιτεκτονικές
- Εύκολη τροποποίηση

Μειονεκτήματα

- Χαμηλότερο επίπεδο βελτιστοποίησης με αποτέλεσμα μειωμένη απόδοση και υψηλότερη κατανάλωση πόρων
- Συχνά απαιτείται περισσότερος σχεδιαστικός φόρτος
- Λιγότερες οδηγίες χρήσης (documentation) για συγκεκριμένες αρχιτεκτονικές
- Διαφορές στα σχεδιαστικά εργαλεία μπορούν να επηρεάσουν τα αποτελέσματα και τα οποία δεν είναι πάντα προβλέψιμα

1.3.2 Πλεονεκτήματα και μειονεκτήματα επεξεργαστών μέσω λογισμικού-κλειστοί

Πλεονεκτήματα

- Πολύ καλή βελτιστοποίηση για συγκεκριμένες αρχιτεκτονικές
- Σχετικά εύκολη παραμετροποίηση
- Η απόδοση, η κατανάλωση πόρων και ισχύος είναι αρκετά καλά προσδιορισμένη.
- Υψηλό επίπεδο αξιοπιστίας στην απόδοση και στην λειτουργικότητα
- Η σχεδίαση έχει επαληθευτεί εκ των προτέρων για την λειτουργικότητά της
- Εύκολη επαλήθευση (testing) στο συγκεκριμένο περιβάλλον
- Εύκολη πρόσβαση σε πηγές προσομοίωσης (testbenches)
- Σχετικά καλές οδηγίες χρήσης
- Δυνατότητα πρόσβασης στην τεχνογνωσία σχεδιασμού

Μειονεκτήματα

- Υπάρχουν περιορισμένα κίνητρα για τον κατασκευαστή ώστε οι επεξεργαστές τους να είναι φορητοί (portable) σε άλλες αρχιτεκτονικές
- Η πρόσβαση σε βοήθεια για τον σχεδιασμό σε πολλές περιπτώσεις είναι έναντι αμοιβής

1.3.3 Πλεονεκτήματα και μειονεκτήματα επεξεργαστών απευθείας στο υλικό

Πλεονεκτήματα

- Πολύ καλές οδηγίες χρήσης, υψηλή βελτιστοποίηση, υψηλή απόδοση και αξιόπιστη σταθερή υλοποίηση
- Ευκολία στη απόκτησή του, όπως το να αγοράζεις ένα συνηθισμένο κύκλωμα
- Καμία καθυστέρηση, άμεση πρόσβαση στη λειτουργικότητά του
- Υψηλό επίπεδο αξιοπιστίας στη λειτουργικότητα και γνωστά λάθη
- Μετρημένος και πολύ καλά χαρακτηρισμένος
- Χαμηλή κατανάλωση ισχύος

Μειονεκτήματα

- Πολύ υψηλή βελτιστοποίηση σε συγκεκριμένες αρχιτεκτονικές με αποτέλεσμα να είναι πολύ δύσκολη η μεταφορά σε άλλη αρχιτεκτονική χωρίς την αισθητή μείωση της απόδοσης ή με αρκετό κόστος
- Υψηλό κίνητρο για τον κατασκευαστή ώστε ο επεξεργαστής να είναι άμεσα συνδεδεμένος με την αρχιτεκτονική του.
- Σταθερή υλοποίηση, χωρίς την δυνατότητα τροποποίησης ή πρόσθεσης νέων στοιχείων αν απαιτούνται.

1.4 Πλεονεκτήματα Χρήσης CPU σε FPGA

Ένας λόγος για την επιλογή ενός ενσωματωμένου επεξεργαστή σε FPGA είναι η ανάγκη για επίτευξη καλύτερης ολοκλήρωσης του υλικού (hardware) και του λογισμικού (software) με την ταυτόχρονη σχεδίαση (hardware software co-design). Η υλοποίηση ενός επεξεργαστή σε συσκευή FPGA μπορεί να μειώσει το κόστος του συστήματος και να βελτιώσει την απόδοσή του [7]. Ένα ακόμα πλεονέκτημα της χρήσης επεξεργαστή βασισμένου σε FPGA είναι η μείωση της “απαρχαίωσης” του συστήματος. Οι παραδοσιακοί επεξεργαστές έχουν ένα περιθώριο ζωής (end-of-life), το οποίο τελειώνει όταν η τεχνολογία αναπτύσσεται ή όταν οι πωλήσεις πέφτουν κάτω από ένα όριο. Χρησιμοποιώντας όμως ένα επεξεργαστή μέσω λογισμικού είναι εύκολο να μεταφέρεις το σύστημα σε νέες γενιές FPGA. Υπάρχουν φυσικά κάποιοι περιορισμοί και κάποιες συνθήκες που πρέπει να ικανοποιούνται όπως η πρόσβαση στον HDL κώδικα του επεξεργαστή και οι άδειες για την χρήση των εργαλείων.

Παρακάτω φαίνονται επιγραμματικά κάποια από τα πλεονεκτήματα χρήσης επεξεργαστή σε FPGA [7].

- Η δυνατότητα υλοποίησης όλων ή των περισσότερων λειτουργιών του συστήματος σε μια μόνο συσκευή.
- Η δυνατότητα υλοποίησης υψηλά προσαρμοσμένης ενσωματωμένης επεξεργαστικής λύσης.
- Η δυνατότητα υλοποίησης μόνο της συγκεκριμένης λειτουργίας η οποία απαιτείται.
- Η δυνατότητα υλοποίησης επεκτάσιμης (scalable) επεξεργαστικής λύσης.
- Η δυνατότητα βελτίωσης της απόδοσης του συστήματος.
- Δυνατότητα τροποποίησης του συστήματος σε μετέπειτα σχεδιαστικό κύκλο.
- Βελτιστοποίηση των διασυνδέσεων μεταξύ του επεξεργαστή και των περιφερειακών.
- Βελτιστοποίηση της συνεργασίας υλικού και λογισμικού (co-design).
- Πιο αποτελεσματική διασύνδεση του συστήματος (chip-to-chip interface).
- Δυνατότητα χρησιμοποίησης του ίδιου υλικού σε πολλές εφαρμογές.
- Δυνατότητα μείωσης του κόστους υλοποίησης.
- Δυνατότητα υλοποίησης κατά προτίμηση (custom) επεξεργαστών.
- Δυνατότητα υλοποίησης πολύ-επεξεργαστικού συστήματος(multi-processor).

1.5 Μικροεπεξεργαστές μέσω λογισμικού

Οι επεξεργαστές μέσω λογισμικού χρησιμοποιούνται όλο και πιο πολύ. Διάφορες εξελίξεις διευκολύνουν τη χρήση αυτών επεξεργαστών, για παράδειγμα τα σταθερά και

πιο αξιόπιστα εργαλεία σύνθεσης, τα ASIC και FPGA μεγαλύτερης χωρητικότητας, τα νέα εμπορικά εργαλεία για εξειδικευμένους επεξεργαστές (ASIP), όπως και η ολοένα αυξανόμενη απαίτηση για υψηλές αποδόσεις και ενσωματωμένη επεξεργασία (embedded processing) χαμηλής κατανάλωσης [11].

Ενώ οι παραδοσιακοί επεξεργαστές πρέπει να βελτιστοποιηθούν για καλή απόδοση σε ένα ολόκληρο τομέα εφαρμογών, οι επεξεργαστές μέσω λογισμικού μπορούν να προσαρμοστούν στις συγκεκριμένες εφαρμογές που εκτελούν. Για παράδειγμα, μια συγκεκριμένη εφαρμογή μπορεί να εκτελεστεί καλύτερα σε έναν επεξεργαστή με μεγάλη κρυφή μνήμη (cache) και με μονάδα κινητής υποδιαστολής (floating-point unit), ενώ μια άλλη εφαρμογή, μπορεί να απαιτεί έναν πολλαπλασιαστή υλικού (hardware multiplier) και να μην χρειάζεται καθόλου κρυφή μνήμη. Σχεδόν όλοι οι επεξεργαστές μέσω λογισμικού είναι παραμετροποιήσιμοι και μπορούν να μεταβληθούν οι παράμετροι τους από τον χρήστη ανάλογα με τις ανάγκες του. Κοινές παράμετροι για παράδειγμα είναι οι μονάδες συνεπεξεργαστή (co-processor units) , όπως η μονάδα υλικού για αριθμούς κινητής υποδιαστολής (hardware floating-point), ο πολλαπλασιαστής (multiplier), ο διαιρέτης (divider) και μονάδες ολισθητών (barrel shifter units). Επίσης, μπορεί να ρυθμιστεί η αρχιτεκτονική της κρυφής μνήμης (π.χ. χρήση ενός ή δύο επιπέδων κρυφής μνήμης, ξεχωριστή ή ενωμένη (unified) κρυφή μνήμη, το μέγεθος της κρυφής μνήμης, το μέγεθος γραμμής (line size), η συσχέτιση (associativity), και η πολιτική αποθήκευσης (write back policy)). Ακόμα υπάρχει η δυνατότητα να παραμετροποιηθούν τα χαρακτηριστικά της μικρο-αρχιτεκτονικής του επεξεργαστή (το βάθος της διασωλήνωσης (depth of pipeline), η λογική παράκαμψης (bypass), το μέγεθος του αρχείου καταχωρητών (register-file) κ.τ.λ) [11]. Η πλειονότητα των επεξεργαστών μέσω λογισμικού περιλαμβάνει πολλά παραμετροποιήσιμα χαρακτηριστικά και η τάση στις καινούργιες εκδόσεις είναι να αυξάνεται συνέχεια ο αριθμός αυτών των παραμέτρων. [7]

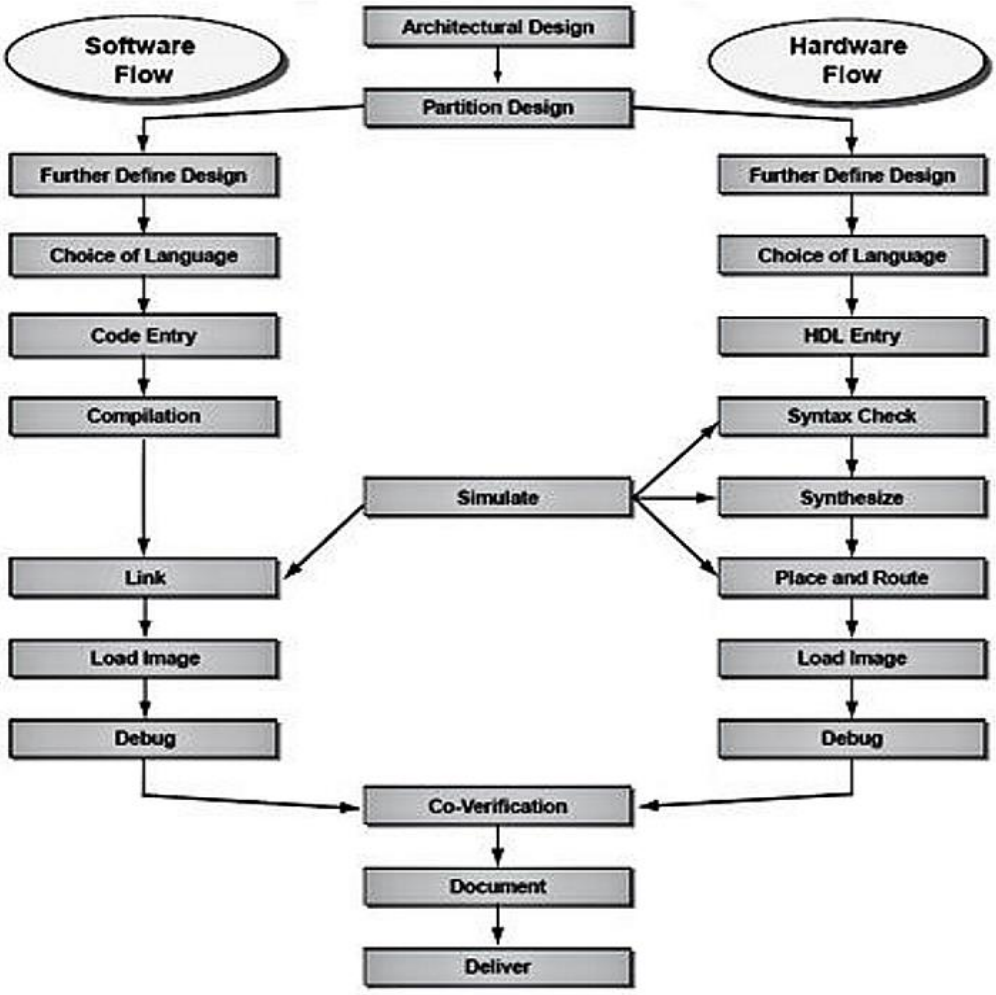
1.6 Χρήση επεξεργαστών μέσω λογισμικού και συν-σχεδίαση (co-design)

Ο επεξεργαστής είναι ένα από τα πιο ευέλικτα συστατικά στην εργαλειοθήκη του σχεδιαστή ενσωματωμένων συστημάτων. Η ευελιξία στη σχεδίαση ενός επεξεργαστή έχει εξελιχθεί εξαιτίας της τεχνολογικής προόδου στον τομέα του υλικού και του λογισμικού. Η ευελιξία των συσκευών FPGA μπορεί να ενισχυθεί από ενσωματωμένους επεξεργαστές μέσα στο FPGA. Η υλοποίηση ενός ενσωματωμένου επεξεργαστή μέσα σε ένα FPGA απαιτεί τις ίδιες αποφάσεις και συμβιβασμούς (trade-offs) που απαιτούνται για την σχεδίαση ενός αυτόνομου επεξεργαστή. Μερικοί από τους παράγοντες οι οποίοι επηρεάζουν την υλοποίηση ενός ενσωματωμένου επεξεργαστή είναι οι σαφείς και συγκεκριμένες απαιτήσεις του συστήματος, η καλή μεθοδολογία σχεδιασμού, ο αποτελεσματικός συν-σχεδιασμός και ο κατάλληλος διαμερισμός του σχεδιασμού. Υπάρχουν πολλοί συμβιβασμοί υλικού και λογισμικού οι οποίοι πρέπει να ληφθούν υπόψη ώστε να υλοποιηθεί ο επεξεργαστής στο FPGA. Μερικές σχεδιαστικές επιλογές περιλαμβάνουν την επιλογή του επεξεργαστή, την επιλογή των περιφερειακών μπλοκ και IP, την αρχιτεκτονική της μνήμης του επεξεργαστή και τον τρόπο διασύνδεσης των διαφόρων στοιχείων [7].

Μερικές επιλογές λογισμικού περιλαμβάνουν την ενημερωμένη κωδικοποίηση (informed coding), την επιλογή και χρήση λειτουργικών συστημάτων πραγματικού χρόνου (RTOS) και την ανάπτυξη οδηγών συσκευών (device driver). Τα εργαλεία λογισμικού και υλικού παίζουν σημαντικό ρόλο και πρέπει να γίνει προσπάθεια ώστε να επιλεχθούν τα καλύτερα διαθέσιμα.

Ως προς τις αποφάσεις για το συν-σχεδιασμό πρέπει να ληφθεί υπόψη ότι η ανάπτυξη ενσωματωμένου λογισμικού μπορεί να καταναλώνει ως και το 50% ή και περισσότερο από την επεξεργαστική ισχύ του ενσωματωμένου επεξεργαστή [7]. Είναι πολύ σημαντικό να ακολουθείται μια συνεκτική ροή ως προς την ανάπτυξη του υλικού και του

λογισμικού για την ταχύτερη ολοκλήρωση ενός συστήματος. Αυτή η συνεργασία μεταξύ του υλικού και του λογισμικού μπορεί να βοηθήσει στην παράλληλη ανάπτυξη του συστήματος. Το σύστημα σχεδίασης και τα εργαλεία που χρησιμοποιούνται είναι σημαντικά για τον αποδοτικό συν-σχεδιασμό. Τα εργαλεία σχεδίασης (toolchain) είναι η συλλογή εργαλείων για το υλικό και το λογισμικό τα οποία χρησιμοποιούνται για την σχεδίαση, την προσομοίωση, την ρύθμιση των παραμέτρων και την αποσφαλμάτωση (debugging). Μια αποδοτική εργαλειοθήκη σχεδίασης μπορεί να παρέχει υψηλό επίπεδο αλληλεπίδρασης και συνεργασίας μεταξύ του υλικού και του λογισμικού. Στο σχήμα 1.6 φαίνεται η σχέση και η αλληλεπίδραση μεταξύ των εργαλείων σχεδίασης [7]



Σχήμα 1.6: Διάγραμμα ροής των σταδίων ανάπτυξης υλικού και λογισμικού [7]

1.7 Τελευταίες εξελίξεις στην τεχνολογία

Υπάρχουν πάρα πολλές υλοποιήσεις στις μέρες μας οι οποίες χρησιμοποιούν διεπαφές αισθητήρων σε FPGAs και “έξυπνα” χαρακτηριστικά σχετιζόμενα με αισθητήρες. Αυτές οι υλοποιήσεις περιλαμβάνουν συστήματα τα οποία έχουν διάφορους επεξεργαστές είτε μέσω λογισμικού είτε όχι. Κάθε υλοποίηση αφορά διαφορετικούς αισθητήρες και εφαρμογές. Οι εφαρμογές αυτές μπορεί να αφορούν την επεξεργασία εικόνας, την παροχή ασφάλειας, τον περιβαλλοντικό και βιομηχανικό ελέγχο και πολλά άλλα ανάλογα με τους αισθητήρες που είναι συνδεδεμένοι στο σύστημα και τα περιφερειακά που περιλαμβάνουν. Τελευταία υπάρχει ραγδαία ανάπτυξη σε συστήματα ασύρματων κόμβων αισθητήρων (WSN) και η έρευνα κατευθύνεται προς αυτήν την κατεύθυνση για την ανάπτυξη αποδοτικών WSN συστημάτων. Η απόδοση ενός τέτοιου συστήματος αξιολογείται κυρίως από τους πόρους που καταναλώνει. Στη συνέχεια παρουσιάζονται κάποια αντιπροσωπευτικά συστήματα τα οποία αφορούν ένα εύρος εφαρμογών. Τα

συστήματα που παρουσιάζονται βασίζονται σε επεξεργαστές μέσω λογισμικού σε συσκευές FPGA.

Στο [12] παρουσιάζεται ένας κόμβος αισθητήρων βασισμένος σε ένα σύστημα SoC DE2-70. Το σύστημα περιλαμβάνει έναν επεξεργαστή OpenRisc 1200, ένα λειτουργικό σύστημα μ/OS II RTOS και ένα δίαυλο Wishbone. Το SoC αποτελείται από τρία βασικά δομικά στοιχεία : μια μονάδα απόκτησης των δεδομένων η οποία τροφοδοτείται από ένα ψηφιακό θερμόμετρο, από τον επεξεργαστή που επεξεργάζεται τα δεδομένα και ένα λογικό μπλοκ, το οποίο διασυνδέει ένα δέκτη ZigBee βασισμένο στο πρωτόκολλο επικοινωνίας RS-232. Η συσκευή FPGA είναι ένα Altera Cyclone II (EP2C70). Στο [13] παρουσιάζεται μια παρόμοια υλοποίηση όπου ο κόμβος των αισθητήρων βασίζεται σε μια συσκευή Altera Cyclone I (EP1C6) και η πληροφορίες μεταδίδονται μέσω Bluetooth, WLAN ή Infrared . Το σύστημα περιλαμβάνει έναν NIOS II επεξεργαστή. Στα [14][15] οι συγγραφείς παρουσιάζουν ένα αυτόνομο σύστημα βασισμένο σε συσκευή FPGA. Χρησιμοποιείται μια συσκευή Zefant XS3-2000 και ένα FPGA Spartan-3 (XC3S2000). Στο σύστημα είναι επίσης συνδεδεμένο ένα CPLD και μια FLASH 128Mb, όπως και ένας πομποδέκτης Xemics. Η υλοποίηση βασίζεται σε έναν LEON2 32-bit επεξεργαστή. Το σύστημα έχει πολλές επεξεργαστικές δυνατότητες και επιτρέπει την διασύνδεση διαφορετικών ειδών αισθητήρων. Στο [16] παρουσιάζεται ένα αυτόνομο σύστημα για επεξεργασία εικόνας βασισμένο στη συσκευή της Altera Cyclone II FPGA (EP2C3) . Το σύστημα περιλαμβάνει έναν πομποδέκτη 2.4 GHz NRF24L01 και έναν αισθητήρα εικόνας CMOS. Στο σύστημα υπάρχει ένας επεξεργαστής Nios II. Στο [17] παρουσιάζεται ένα σύστημα φωνητικής αναγνώρισης (“bird call identification”) βασισμένο στο FPGA Spartan 3E, XC3S1600E. Το σύστημα περιλαμβάνει έναν Microblaze επεξεργαστή. Στο [18] οι συγγραφείς παρουσιάζουν ένα κόμβο αισθητήρων για την ανίχνευση φωτιάς σε δάσος. Η υλοποίηση βασίζεται σε ένα Altera Cyclone II (EP2C70) FPGA και χρησιμοποιείται ένας Nios II σε συνδυασμό με ένα αισθητήρα θερμοκρασίας και ένα πομποδέκτη Bluetooth PTR4500. Τέλος στο [19] παρουσιάζεται ένα σύστημα βασισμένο σε ένα FPGA Altera Cyclone II (EP2C5) το οποίο αφορά την παρακολούθηση της γραμμής παραγωγής των αεροσκαφών. Ο κόμβος αισθητήρων περιλαμβάνει έναν πομποδέκτη NRF2401 transceiver (2.4 GHz) και ένα επεξεργαστή NIOS II.

Στη συνέχεια παρουσιάζονται κάποια συστήματα που έχουν υλοποιηθεί και σχετίζονται είτε με ταλαντωτές δακτυλίου είτε με μετρητές συχνότητας. Στο [20] χρησιμοποιείται ένα FPGA με ένα θερμικό αισθητήρα για την μέτρηση της συμπεριφοράς ταλαντωτών δακτυλίου σε διαφορετικά επίπεδα τάσης. Στο σύστημα χρησιμοποιείται ένας μικροεπεξεργαστής Microblaze. Στο [21] παρουσιάζεται ένα σύστημα το οποίο επιτρέπει την μέτρηση ορισμένων ηλεκτρικών παραμέτρων (κυρίως αντίσταση και χωρητικότητα) συνδέοντας απευθείας στο FPGA την υπό μέτρηση μονάδα. Δεν απαιτείται κάποιος μετατροπέας από αναλογικό σε ψηφιακό σήμα ή κάποιο εξωτερικό κύκλωμα. Στο σύστημα χρησιμοποιείται ένας Picoblaze επεξεργαστής. Στο [22] παρουσιάζεται μια ανάλυση των διεπαφών χωρητικών αισθητήρων με μικροεπεξεργαστές. Η ανάλυση αυτή αφορά κυρίως σε διεπαφές οι οποίες μετράνε τον χρόνο φόρτισης και αποφόρτισης των RC κυκλωμάτων. Περιγράφονται επίσης τρόποι βαθμονόμησης τέτοιων συστημάτων. Στο [23] περιγράφεται ένα σύστημα υλοποιημένο σε FPGA το οποίο μετράει τον θόρυβο σε ταλαντωτές Quartz κρυστάλλου. (Quartz crystal oscillator). Με αυτό το σύστημα μπορεί να μετρηθεί σε πραγματικό χρόνο ο θόρυβος συχνότητας και η ανάλυση των QCM αισθητήρων. Η υλοποίηση έχει γίνει σε ένα Virtex-4 FPGA της Xilinx και περιλαμβάνει έναν Microblaze επεξεργαστή. Στο [24] περιγράφεται ένα σύστημα το οποίο μετράει χρονικά διαστήματα. Η απόδοση ενός GPS εξαρτάται από την ακρίβεια και την σταθερότητα αυτών των χρονικών διαστημάτων. Το σύστημα βασίζεται σε ένα FPGA Spartan-3 της XILINX και ο έλεγχος γίνεται με έναν Picoblaze

επεξεργαστή. Ο Picoblaze λαμβάνει τα δεδομένα των μετρήσεων και τις πληροφορίες για τον χρόνο και τις επεξεργάζεται. Ταυτόχρονα ελέγχει την κατάσταση του συστήματος και είναι αρμόδιος για την αποδοτική λειτουργία του.

1.8 Προτεινόμενο "έξυπνο" σύστημα αισθητήρων

Στα πλαίσια της διπλωματικής εργασίας, η οποία αφορά στην ανάπτυξη ενός συστήματος "έξυπνων" αισθητήρων με τη χρήση ενσωματωμένου επεξεργαστή σε συσκευή FPGA, επιλέχτηκε η χρήση του επεξεργαστή LEON3 της Aeroflex Gaisler. Ο σκοπός ήταν να υλοποιηθεί ένα σύστημα το οποίο να είναι παραμετροποιήσιμο, να μεταφέρεται εύκολα σε επόμενες γενιές FPGA και να προσαρμόζεται στις εκάστοτε ανάγκες των μετρήσεων και των αισθητήρων. Ο LEON3 παρέχει όλες αυτές τις δυνατότητες, προσφέρει υψηλό επίπεδο παραμετροποίησης, είναι δωρεάν, δεν είναι υλοποιήσιμος μόνο για συγκεκριμένες αρχιτεκτονικές αλλά μπορεί να μεταφερθεί σχεδόν σε οποιοδήποτε συσκευή FPGA, έχει πολύ καλές οδηγίες για τη χρήση του (documentation) και υποστήριξη. Ακόμα η χρήση του επεξεργαστή και η εγκατάσταση του είναι εύκολη και προσφέρεται μαζί με αυτόν μια πληθώρα από διαθέσιμα δομικά στοιχεία.

Επίσης, στόχος της διπλωματικής ήταν το "έξυπνο" σύστημα αισθητήρων να περιλαμβάνει και ένα λειτουργικό σύστημα, π.χ. Linux, ώστε να γίνει αυτόνομο, να μπορεί να χρησιμοποιηθεί εύκολα και να προσφέρει λειτουργίες που μόνο με τη χρήση ενός λειτουργικού συστήματος μπορούν να αποδοθούν ικανοποιητικά. Τέτοιες λειτουργίες για παράδειγμα είναι η ασφαλής μεταφορά των δεδομένων των αισθητήρων μέσω ασφαλούς σύνδεσης (SSH ή SFTP) και η εύκολη διαχείριση των αποτελεσμάτων χρησιμοποιώντας SVGA οθόνη και πληκτρολόγιο. Ο επεξεργαστής LEON3 υποστηρίζει το λειτουργικό σύστημα Linux και διαθέτει πολλούς έτοιμους οδηγούς (drivers) οι οποίοι χρησιμοποιήθηκαν για την επίτευξη των στόχων αυτών.

Τέλος, με τον LEON3 μπορούν να χρησιμοποιηθούν σχεδόν όλα τα γνωστά εργαλεία σχεδίασης για υλικό (π.χ. QUARTUS της Altera και ISE της Xilinx) και για λογισμικό (π.χ. ECLIPSE) και μπορεί να υπάρξει αποδοτική συνεργασία μεταξύ των δύο (co-design). Αυτό ήταν σημαντικό στην επιλογή του συγκεκριμένου επεξεργαστή και βοήθησε κατά την ανάπτυξη του συστήματος, όπως βοήθησε και η ύπαρξη ενός λειτουργικού εργαλείου αποσφαλμάτωσης, του GRMON.

Στο επόμενο κεφάλαιο θα υπάρξει αναλυτική παρουσίαση της αρχιτεκτονικής και της λειτουργικότητας του επεξεργαστή LEON3 και των εργαλείων σχεδίασής του. Στο 3ο κεφάλαιο θα γίνει ανάλυση του κυκλώματος των αισθητήρων. Το κύκλωμα των αισθητήρων, είναι το κύκλωμα διασύνδεσης (interface) μεταξύ των αισθητήρων και του επεξεργαστή. Θα αναλυθεί πώς μπορεί να μετρηθεί η συχνότητα των αισθητήρων από το συγκεκριμένο κύκλωμα και θα αναπτυχθεί η θεωρία στην οποία βασίζεται. Στο 4ο κεφάλαιο θα παρουσιαστεί το υλικό μέρος της υλοποίησης, δηλαδή ποια δομικά στοιχεία χρησιμοποιήθηκαν, ο τρόπος διασύνδεσης τους αλλά και ο τρόπος λειτουργίας τους. Θα παρουσιαστεί η ενσωμάτωση του κυκλώματος της μέτρησης των αισθητήρων στο σύστημα και οι αλλαγές που έγιναν σε αυτό ώστε να γίνει πιο αποδοτικό και η διασύνδεσή του με τον επεξεργαστή. Το 5ο κεφάλαιο περιλαμβάνει το λογισμικό (software) μέρος της υλοποίησης. Παρουσιάζεται η χρήση του λειτουργικού συστήματος Linux με τον LEON3 και αναλύεται το Linux image που χρησιμοποιήθηκε για την υλοποίηση. Ακόμα υπάρχει επεξήγηση του κώδικα C, του προγράμματος για τις μετρήσεις, ο οποίος "τρέχει" στον επεξεργαστή. Στο 6ο κεφάλαιο υπάρχει η συνολική παρουσίαση του κυκλώματος και οι μετρήσεις που λήφθηκαν με αυτό. Οι μελλοντικές βελτιώσεις και τα συμπεράσματα κλείνουν το κεφάλαιο. Τέλος στα παραρτήματα υπάρχουν οδηγίες χρήσης για την εγκατάσταση του LEON3 και του Linux, όπως και οι κώδικες C με σχόλια.

2. Ο ΕΠΕΞΕΡΓΑΣΤΗΣ LEON3

2.1 Εισαγωγή

Ο LEON3 είναι ένας 32-μπιτ μικροεπεξεργαστής, ο οποίος βασίζεται στην αρχιτεκτονική SPARC-V8 RISC, όπως και το σύνολο των εντολών του. Αρχικά, είχε σχεδιαστεί από το Ευρωπαϊκό Κέντρο Διαστημικής Έρευνας και Τεχνολογίας (ESTEC), που είναι κομμάτι του Ευρωπαϊκού Οργανισμού Διαστήματος (ESA) και στη συνέχεια από την Gaisler Research. Η περιγραφή του είναι σε γλώσσα VHDL. Ο LEON3 έχει ένα διπλό μοντέλο αδειών το LGPL/GPL , σύμφωνα με το οποίο μπορεί να χρησιμοποιηθεί χωρίς χρηματικό αντίτιμο και χρειάζεται ειδική άδεια για χρήση του σε εμπορικές εφαρμογές. Ο επεξεργαστής είναι παραμετροποιήσιμος μέσα από τις παραμέτρους της γλώσσας VHDL (VHDL generics) και χρησιμοποιείται στα SoCs και στην έρευνα αλλά και σε εμπορικές εφαρμογές [25].

Παρακάτω παρουσιάζονται τα βασικά πλεονεκτήματα και μειονεκτήματα του LEON3 σε σχέση με άλλους επεξεργαστές μέσω λογισμικού [7]:

Πλεονεκτήματα

- Δεν απαιτείται άδεια για εκπαιδευτική και ερευνητική χρήση
- Είναι διαθέσιμος ολόκληρος ο RTL πηγαίος κώδικα του επεξεργαστή
- Γρήγορη υποστήριξη
- Είναι συμβατός με το λειτουργικό σύστημα Linux και υποστηρίζει RTOS

Μειονεκτήματα

- Δεν υποστηρίζονται όλα τα FPGAs, αν και υποστηρίζεται αρκετά μεγάλος αριθμός.
- Δεν έχει πολύ διαδεδομένη χρήση σε σχέση με άλλους επεξεργαστές, όπως ο Microblaze και ο NiosII

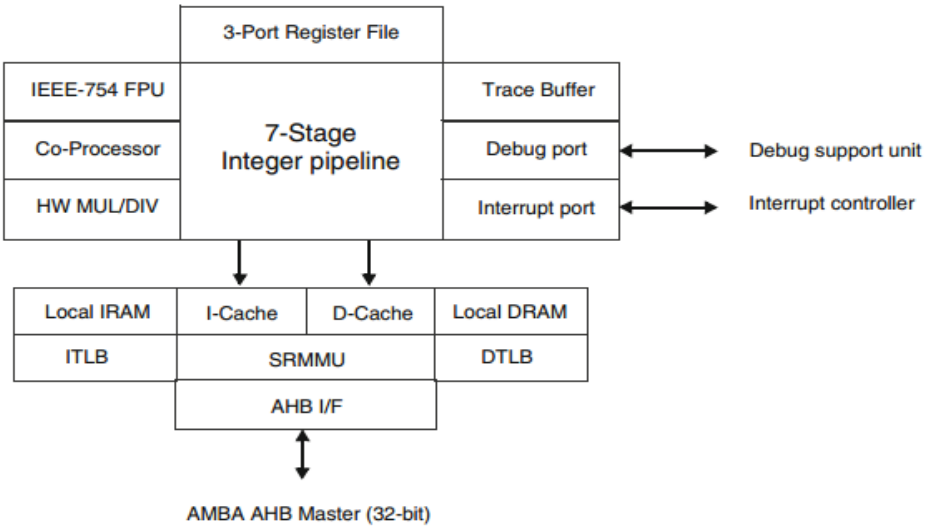
2.2 Αρχιτεκτονική - Χαρακτηριστικά

Ο LEON3 είναι ένα 32-bit επεξεργαστής σύμφωνος με την αρχιτεκτονική IEEE-1754 (SPARC V8). Είναι σχεδιασμένος για ενσωματωμένες εφαρμογές και συνδυάζει υψηλή απόδοση με χαμηλή πολυπλοκότητα και χαμηλή κατανάλωση ισχύος. Ο LEON3 έχει τα εξής βασικά χαρακτηριστικά: 7 επίπεδα διασωλήνωσης (pipeline) με Harvard αρχιτεκτονική, ξεχωριστή κρυφή μνήμη για δεδομένα (data) και για εντολές (Instruction), μονάδα διαχείρισης μνήμης (memory management unit), πολλαπλασιαστή και διαιρέτη υλικού, αποσφαλμάτωση πάνω στο τσιπ και υποστήριξη πολύ-επεξεργαστικής επέκτασης. Ο επεξεργαστής LEON3 μπορεί να ενισχυθεί με ανοχή σε σφάλματα (fault-tolerance) εναντίων SEU λαθών (LEON3FT). Η ανοχή σε σφάλματα είναι επικεντρωμένη στην προστασία των RAM μπλοκ πάνω στο τσιπ, τα οποία χρησιμοποιούνται για την υλοποίηση αρχείων καταχωρητών και της κρυφής μνήμης. Ρυθμίζοντας τον επεξεργαστή ώστε να έχει ανοχή σε σφάλματα προστίθενται επιπλέον εσωτερικοί καταχωρητές, πεδία καταχωρητών και αλλάζει ο τρόπος διευθυνσιοδότησης και σύνδεσης των συσκευών με τον επεξεργαστή [25].

Συνοπτικά τα χαρακτηριστικά του επεξεργαστή LEON3 είναι τα εξής [25]:

- Σύνολο εντολών SPARC V8 με επεκτάσεις V8e
- Προηγμένη διασωλήνωση 7 επιπέδων
- Πολλαπλασιασμός και διαίρεση υλικού και μονάδες MAC
- Υψηλής απόδοσης πλήρως διασωληνωμένο IEEE-754 FPU
- Ξεχωριστή κρυφή μνήμη για εντολές και δεδομένα με τη τεχνική ‘snooping’
- Ρυθμιζόμενη κρυφή μνήμη 1 - 4 τρόπων, 1 - 256 Kbytes/τρόπο. Τυχαία (Random), με LRU
- Τοπική RAM, 1 - 512 Kbytes

- SPARC Reference MMU (SRMMU) με ρυθμιζόμενο TLB
- AMBA-2.0 με AHB διεπαφή διαύλου (bus interface)
- Προηγμένη αποσφαλμάτωση πάνω στο τσιπ (Advanced on-chip debug support)
- Υποστήριξη συμμετρικού πολύ-επεξεργαστικού συστήματος (SMP)
- Λειτουργία χαμηλής κατανάλωσης (Power-down mode) και τεχνική 'clock gating'
- Δυναμικός (Robust) και πλήρως συνχρονισμένος μονής κορυφής (single-edge) σχεδιασμός ρολογιού
- Ταχύτητες ως και 125 MHz σε FPGA και 400 MHz σε 0.13 um ASIC
- Έκδοση με ανοχή σε σφάλματα (Fault-tolerant) για διαστημικές εφαρμογές
- Υψηλή παραμετροποίηση
- Μεγάλο εύρος εργαλείων λογισμικού: μεταγλωττιστές (compilers), πυρήνες (kernels), προσομοιωτές (simulators) και εργαλεία αποσφαλμάτωσης
- Υψηλή απόδοση: 1.4 DMIPS/MHz, 1.8 CoreMark/MHz (gcc -4.1.2)



Σχήμα 2.1: Διάγραμμα βαθμίδων του επεξεργαστή LEON3 [26]

Ο LEON3 διανέμεται ως μέρος της GRLIB IP βιβλιοθήκης και έτσι επιτρέπεται η εύκολη ενσωμάτωσή του σε πολύπλοκα SoCs. Με την GRLIB μπορεί να σχεδιαστεί ένα τροποποιήσιμο LEON3 πολυεπεξεργαστικό (multi-processor) σύστημα το οποίο να έχει μέχρι και 4 επεξεργαστές και μεγάλο εύρος περιφερειακών [25]. Στην συνέχεια παρουσιάζεται πιο αναλυτικά η λειτουργία του κάθε δομικού στοιχείου του LEON3.

2.2.1 Μονάδα ακεραίων (Integer unit)

Η μονάδα ακεραίων του LEON3 υλοποιεί το πλήρες σύνολο εντολών της SPARC V8 αρχιτεκτονικής, το οποίο περιλαμβάνει εντολές πολλαπλασιασμού και διαίρεσης. Ο αριθμός των καταχωρητών ρυθμίζεται μέσα σε ένα όριο (2-32), με την προεπιλεγμένη ρύθμιση να είναι στο 8. Η διασωλήνωση αποτελείται από 7 στάδια με ξεχωριστή διασύνδεση για τις εντολές και τα δεδομένα (Harvard αρχιτεκτονική) [26].

2.2.2 Κρυφή μνήμη

Ο LEON3 έχει ένα υψηλά παραμετροποιήσιμο σύστημα κρυφής μνήμης, το οποίο αποτελείται από μια ξεχωριστή μνήμη για τις εντολές και τα δεδομένα. Και οι δύο μνήμες μπορούν να ρυθμιστούν με 1-4 τρόπους, 1-256 KB/τρόπο, 16 ή 32 bytes/γραμμή. Η μνήμη για τις εντολές περιλαμβάνει ένα έγκυρο (valid) μπιτ για κάθε 32-μπιτ λέξη και χρησιμοποιεί συνεχή ροή (streaming) κατά την διάρκεια του γεμίσματος των γραμμών (line-refill) ώστε να ελαχιστοποιηθεί η καθυστέρηση του γεμίσματος. Η μνήμη για τα

δεδομένα έχει ένα έγκυρο μπιτ (valid bit) σε κάθε γραμμή, χρησιμοποιεί την πολιτική 'write-through' και υλοποιεί μια διπλής λέξης ενδιάμεση μνήμη (buffer) - εγγραφής. Μπορεί να χρησιμοποιηθεί η τεχνική 'snooping' στον δίαυλο AHB ώστε να διατηρηθεί η συνεκτικότητα της κρυφής μνήμης. Μπορεί να προστεθεί μια τοπική RAM είτε στην μνήμη δεδομένων είτε στη μνήμη εντολών ώστε να επιτρέπεται πρόσβαση σε '0-waitstates' χωρίς την πρόσβαση στον AHB δίαυλο [26].

2.2.3 Μονάδα κινητής υποδιαστολής (FPU)

Ο LEON3 περιλαμβάνει διασυνδέσεις για σύνδεση με μονάδες κινητής υποδιαστολής και για κατά προτίμηση συνεπεξεργαστή (custom co-processor). Δύο ελεγκτές FPU είναι διαθέσιμοι, ένας για τον GRFPU ο οποίος έχει υψηλή απόδοση και ένας για τον GRFPU-Lite. Οι επεξεργαστές κινητής υποδιαστολής και ο συνεπεξεργαστής λειτουργούν παράλληλα με τη μονάδα ακεραίων και δεν μπλοκάρουν την λειτουργία της εκτός και αν υπάρχει κάποια εξάρτηση στα δεδομένα ή στους πόρους μεταξύ τους [26].

2.2.4 Μονάδα διαχείρισης μνήμης (Memory management unit)

Υπάρχει η επιλογή να επιλεχθεί για μονάδα διαχείρισης μνήμης μια μονάδα SPARC V8 Reference Memory Management Unit (SRMMU). Η μονάδα SRMMU υλοποιεί το πλήρες μοντέλο του SPARC V8 MMU και παρέχει αντιστοίχιση (mapping) μεταξύ 32-μπιτ εικονικών (virtual) διευθύνσεων και της φυσικής μνήμης. Η MMU μπορεί να ρυθμιστεί να έχει ως και 64 TLB 'entries' για κάθε υλοποιήσιμο TLB [26].

2.2.5 Αποσφαλμάτωση

Η διασωλήνωση του LEON3 περιλαμβάνει την δυνατότητα να επιτρέπεται μη-παρεμβατική (non-intrusive) αποσφαλμάτωση στο υλικό. Για την διευκόλυνση της αποσφαλμάτωσης λογισμικού, μπορούν να ενεργοποιηθούν ως και 4 καταχωρητές-παρατηρητές (watchpoint registers). Ο κάθε καταχωρητής μπορεί να προκαλέσει μια διακοπή (breakpoint trap) σε κάποια εντολή ή σε κάποια διεύθυνση δεδομένων. Όταν έχει επιλεχτεί η χρήση της μονάδας αποσφαλμάτωσης, οι καταχωρητές-παρατηρητές μπορούν να χρησιμοποιηθούν ώστε να γίνει είσοδος στη λειτουργία αποσφαλμάτωσης. Μέσω της διασύνδεσης της μονάδας αποσφαλμάτωσης, υπάρχει πλήρης πρόσβαση στους καταχωρητές του επεξεργαστή και της κρυφής μνήμης. Η διασύνδεση της μονάδας αποσφαλμάτωσης επιτρέπει επίσης τις λειτουργίες 'single stepping', 'instruction tracing' και 'hardware breakpoint/watchpoint control'. Ένας εσωτερικός ανιχνευτής-ενδιάμεσης μνήμης (trace buffer) μπορεί να παρακολουθεί και να αποθηκεύει τις "εκτελεσθείσες" εντολές και μπορεί ύστερα να διαβαστεί μέσω της διασύνδεσης της μονάδας αποσφαλμάτωσης [26].

2.2.6 Διασύνδεση διακοπών (interrupt interface)

Ο LEON3 υποστηρίζει το μοντέλο διακοπών SPARC V8 με συνολικά 15 ασύγχρονες διακοπές (Interrupts). Η διασύνδεση των διακοπών προσφέρει την δυνατότητα και να παραχθούν διακοπές αλλά και να αναγνωρισθούν [26].

2.2.7 Λειτουργία χαμηλής κατανάλωσης (Power-down mode)

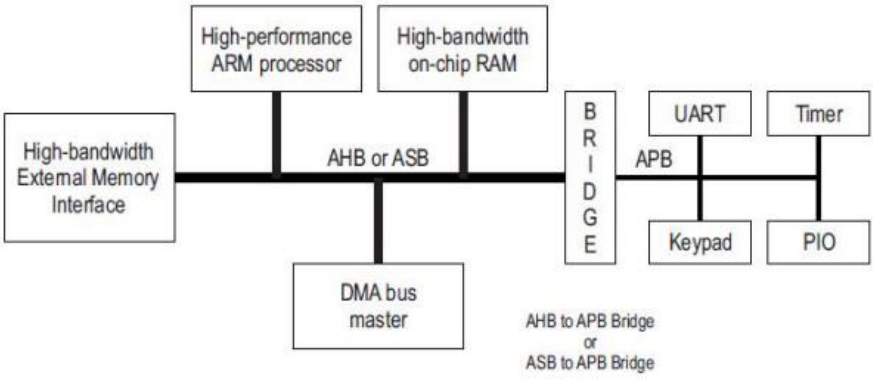
Ο LEON3 επεξεργαστής μπορεί να μπει σε λειτουργία χαμηλής κατανάλωσης, η οποία σταματάει τη διασωλήνωση και τις κρυφές μνήμες μέχρι την επόμενη διακοπή. Ο επεξεργαστής υποστηρίζει την δυνατότητα επιλογής της τεχνικής 'clock gating' κατά την διάρκεια της περιόδου χαμηλής κατανάλωσης, παρέχοντας ένα σήμα ενεργοποίησης το οποίο είναι συνδεδεμένο με ένα εξωτερικό ρολόι. Την ίδια στιγμή παρέχεται ένα ξεχωριστό ρολόι στο μικρό κομμάτι του επεξεργαστή το οποίο χρειάζεται να "τρέχει" κατά τη λειτουργία χαμηλής κατανάλωσης, ώστε να ελέγχει για εντολή "ξυπνήματος" (wake-up calls) και να διατηρεί την συνεκτικότητα της κρυφής μνήμης [26].

2.2.8 Υποστήριξη πολυεπεξεργαστικού συστήματος (Multi-processor support)

Ο LEON3 έχει σχεδιαστεί ώστε να χρησιμοποιείται σε πολύ-επεξεργαστικά συστήματα. Κάθε επεξεργαστής έχει το δικό του μοναδικό κωδικό. Οι 'write-through' κρυφές μνήμες και ο μηχανισμός του 'snooping' διασφαλίζουν την συνεκτικότητα της κρυφής μνήμης σε συστήματα με κοινές μνήμες [26].

2.3 Αρχιτεκτονική του διαύλου επικοινωνίας AMBA (AMBA BUS)

Το όνομα AMBA προέρχεται από το Advanced Microcontroller Bus Architecture. Είναι ένα πρότυπο διαύλου (bus standard) το οποίο αναπτύχθηκε από την ARM. Το σύστημα AMBA χρησιμοποιείται στον LEON3 για την διασύνδεση των διαφόρων περιφερειακών μεταξύ τους αλλά και με τον επεξεργαστή. Το AMBA μπορεί να χαρακτηριστεί σαν ένα πρότυπο επικοινωνίας πάνω στο τσιπ (on-chip communication standard) για τον σχεδιασμό υψηλής απόδοσης ενσωματωμένων μικροεπεξεργαστών. Το τυπικό AMBA σύστημα φαίνεται στο σχήμα 2.2. Υπάρχουν δύο δίαυλοι επικοινωνίας, ο ένας λειτουργεί με υψηλές αποδόσεις για τα στοιχεία υψηλής ταχύτητας (AHB), όπως η εσωτερική μνήμη του τσιπ και τη μονάδα DMA και το άλλο (APB) στο οποίο συνδέονται περιφερειακά τα οποία δεν απαιτούν υψηλό εύρος ζώνης (bandwidth) [27].



Σχήμα 2.2: Δίαυλος επικοινωνίας AMBA [27]

2.3.1 Υψηλής απόδοσης δίαυλος επικοινωνίας AHB

Το AMBA AHB είναι ο δίαυλος υψηλών επιδόσεων. Απευθύνεται σε μονάδες (modules) υψηλής απόδοσης και υψηλών συχνοτήτων. Υποστηρίζει την αποδοτική σύνδεση των επεξεργαστών, των μνημών πάνω στο τσιπ αλλά και των εξωτερικών μνημών. Το AHB με την ευκολία χρήσης του βοηθάει στη αποδοτική ροή της σχεδίασης μέσω σύνθεσης και αυτοματοποιημένων τεχνικών ελέγχου [27].

2.3.2 Προηγμένος δίαυλος επικοινωνίας ASB

Το AMBA ASB είναι ένα εναλλακτικός δίαυλος ιδανικός για χρήση όταν τα υψηλής αποδόσεως χαρακτηριστικά του AHB διαύλου δεν είναι απαραίτητα. Το ASB επίσης υποστηρίζει την αποδοτική σύνδεση μεταξύ των επεξεργαστών των μνημών πάνω στο τσιπ αλλά και των εξωτερικών μνημών [27].

2.3.3 Προηγμένος περιφερειακός δίαυλος επικοινωνίας APB

Το AMBA APB είναι βελτιστοποιημένο ως προς την κατανάλωση ενέργειας και ως προς τη μειωμένη πολυπλοκότητα στη διασύνδεση του με τις περιφερειακές μονάδες. Το APB απευθύνεται σε περιφερειακά χαμηλής κατανάλωσης. Ο δίαυλος APB μπορεί να χρησιμοποιηθεί σε συνδυασμό και με το δίαυλο AHB αλλά και με το δίαυλο ASB [27].

2.4 Η βιβλιοθήκη GRLIB

Η GRLIB IP βιβλιοθήκη είναι ένα ενσωματωμένο σετ από επαναχρησιμοποιούμενους IP πυρήνες (cores), σχεδιασμένο για την ανάπτυξη συστημάτων SoC. Έχει αναπτυχθεί από την AeroFlex Gaisler. Οι IP πυρήνες της βιβλιοθήκης είναι επικεντρωμένοι γύρω από το κοινό δίαυλο πάνω στο τσιπ και χρησιμοποιείται μια συνεκτική μέθοδος για σύνθεση και προσομοίωση. Η βιβλιοθήκη παρέχεται με την άδεια χρήσης GNU/GPL. Επίσης η βιβλιοθήκη δεν συνδέεται με κάποιον προμηθευτή (vendor independent) και υποστηρίζει διάφορα CAD εργαλεία και τεχνολογίες. Το ιδιαίτερο χαρακτηριστικό της GRLIB IP βιβλιοθήκης είναι η “σύνδεσε και τρέξε” (plug&play) μέθοδος για την ρύθμιση και την σύνδεση των IP πυρήνων [28].

2.4.1 Διαθέσιμοι πυρήνες στην βιβλιοθήκη GRLIB

Η βιβλιοθήκη παρέχει διάφορους πυρήνες όπως: μονάδα ελέγχου των διαύλων AMBA AHB/APB, τον επεξεργαστή Leon3 SPARC, ελεγκτή (controller) 32-bit PC133 SDRAM, γέφυρα 32-bit PCI με DMA, 10/100/1000 Mbit Ethernet MAC, 8/16/32-bit PROM και ελεγκτή SRAM, ελεγκτές για 16/32/64-bit DDR/DDR2, USB-2.0 host, ελεγκτή CAN, ελεγκτή TAP, SPI, I^2C, ATA, UART με FIFO, μονάδα χρονισμού (timer), ελεγκτή διακοπών, και πύλη 32-bit GPIO [28].

2.4.2 Οργάνωση της βιβλιοθήκης

Η οργάνωση της GRLIB IP βιβλιοθήκης είναι με τέτοιο τρόπο ώστε κάθε VHDL βιβλιοθήκη να περιλαμβάνει ένα αριθμό πακέτων, τα οποία δηλώνουν τους πυρήνες IP και τον τύπο διασύνδεσής τους. Τα αρχεία (scripts) για την σύνθεση και την προσομοίωση δημιουργούνται αυτόματα από ένα αρχείο “makefile”. Η πρόσθεση και αφαίρεση βιβλιοθηκών και πακέτων μπορεί να γίνει χωρίς την τροποποίηση ολόκληρης της βιβλιοθήκης. Με αυτό τον τρόπο διασφαλίζεται ότι η μια βιβλιοθήκη δεν επηρεάζει την άλλη. Υπάρχουν κάποιες γενικές βιβλιοθήκες για τον καθορισμό της δομής των δεδομένων και των λειτουργιών των εργαλείων [28].

2.4.3 Τρόπος σχεδίασης

Στην GRLIB IP βιβλιοθήκη όλοι οι GRLIB πυρήνες χρησιμοποιούν την ίδια δομή για την σύνδεση με τον AMBA δίαυλο και έτσι μπορούν να συνδεθούν εύκολα μεταξύ τους. Υπάρχει ένας ελεγκτής για τον δίαυλο AHB και μία γέφυρα AHB/APB, τα οποία επιτρέπουν την εύκολη δημιουργία ενός AHB/APB συστήματος [28].

2.4.4 Η δυνατότητα “σύνδεσε και τρέξε”

Οι ρυθμίσεις για το υλικό του συστήματος μπορούν να εντοπιστούν με τη χρήση λογισμικού χρησιμοποιώντας την δυνατότητα “σύνδεσε και τρέξε” της GRLIB. Αυτή η δυνατότητα επιτρέπει σε εφαρμογές λογισμικού ή λειτουργικά συστήματα να ρυθμίζονται ώστε να ταιριάζουν με το εκάστοτε υλικό. Με αυτό τον τρόπο η ανάπτυξη του λογισμικού απλοποιείται κατά πολύ, καθώς δεν χρειάζεται να ρυθμίζεται κάθε φορά ώστε να ταιριάζει με το εκάστοτε υλικό. Στην GRLIB , η “σύνδεσε και τρέξε” λειτουργία αποτελείται από τρεις πληροφορίες [28]:

- Ένα μοναδικό αριθμό αναγνώρισης για κάθε πυρήνα IP
- Η αντιστοίχιση με την μνήμη (memory mapping) των διαύλων AHB/APB
- Το χρησιμοποιούμενο διάνυσμα διακοπών (interrupt vector).

Αυτές οι πληροφορίες στέλνονται στον αποκωδικοποιητή του διαύλου και αποθηκεύονται σε μια μικρή περιοχή στη αρχή του χώρου διευθύνσεων (address space). Οποιοσδήποτε AHB Master μπορεί να διαβάσει τις ρυθμίσεις του συστήματος χρησιμοποιώντας το δίαυλο επικοινωνίας. Για την παροχή της πληροφορίας “σύνδεσε

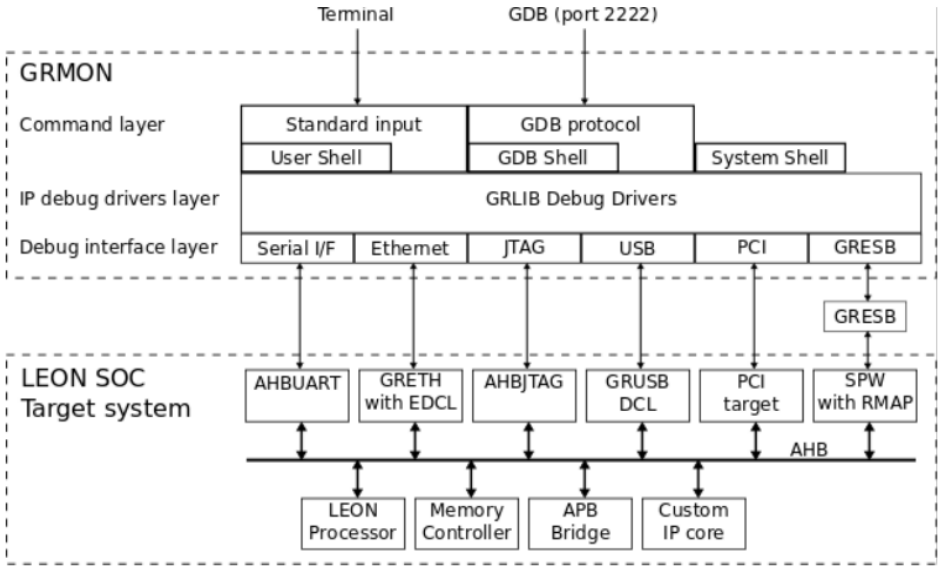
και τρέξε" από τα AMBA στοιχεία με έναν αρμονικό τρόπο, υπάρχει ένα αρχείο στο οποίο κρατούνται οι ρυθμίσεις από τις AMBA συσκευές. Αυτό το αρχείο με τις ρυθμίσεις αποτελείται από 8 32-μπιτ λέξεις, από τις οποίες οι 4 έχουν πληροφορία για τον τύπο του πυρήνα και τη δρομολόγηση των διακοπών (interrupt routing) και οι άλλες 4 περιέχουν πληροφορίες για την αντιστοίχιση στη μνήμη. Για κάθε συσκευή περιλαμβάνονται πληροφορίες για την ταυτότητα του κατασκευαστή, τη ταυτότητα της συσκευής, τον αριθμό έκδοσης. Στις πληροφορίες περιλαμβάνονται η αρχική διεύθυνση της περιοχής της συσκευής, μια μάσκα η οποία καθορίζει το μέγεθος της περιοχής της συσκευής, πληροφορίες για το αν η περιοχή είναι 'cacheable' ή 'pre-fetchable' και πληροφορία για το αν η συγκεκριμένη περιοχή αναγνωρίζεται ως AHB 'memory bank', AHB 'I/O bank' ή APB 'I/O bank' [28].

2.5 GRMON – Πρόγραμμα αποσφαλμάτωσης του LEON3

Το GRMON είναι ένα γενικό σύστημα αποσφαλμάτωσης για τον επεξεργαστή LEON3 και για SoCs τα οποία βασίζονται στην GRLIB IP βιβλιοθήκη. Το GRMON περιλαμβάνει τις εξής λειτουργίες [29]:

- Γράψιμο/Διάβασμα σε όλους τους καταχωρητές της μνήμης και του συστήματος
- Ενσωματωμένο μεταγλωττιστή γλώσσας μηχανής (disassembler)
- Κατέβασμα και εκτέλεση εφαρμογών του LEON3
- Διαχείριση σημείων διακοπής και ελέγχου (Breakpoint/watchpoint management)
- Απομακρυσμένη σύνδεση στον GNU αποσφαλματωτή (GDB)
- Υποστήριξη διεπαφών USB, JTAG, RS232, PCI, Ethernet και SpaceWire
- Tcl

Το σύστημα επικοινωνίας του GRMON συνδέεται με τη διεπαφή του αποσφαλματωτή η οποία βρίσκεται πάνω στο υλικό, και διαμέσου αυτής μπορεί να διαβάσει και να γράψει στον δίαυλο επικοινωνίας του τσιπ (AHB). Η διεπαφή του αποσφαλματωτή μπορεί να έχει διάφορους τύπους. Ο επεξεργαστής LEON3 υποστηρίζει αποσφαλμάτωση με σειριακή UART, με 32-μπιτ PCI, με JTAG, με Ethernet και με SpaceWire. Πάνω στο σύστημα όλα τα είδη διεπαφών αντιμετωπίζονται ως AHB MASTERS και το πρωτόκολλο αποσφαλμάτωσης υλοποιείται στο υλικό. Δεν χρειάζεται να υπάρχει λογισμικό για να γίνει αποσφαλμάτωση σε ένα LEON3 σύστημα και το σύστημα δεν χρειάζεται καν να έχει επεξεργαστή [29].



Σχήμα 2.3 Διάγραμμα λειτουργίας του GRMON [29]

Υπάρχουν δύο τρόποι λειτουργίας του GRMON, μέσω εντολών και με τη χρήση του προγράμματος GDB. Με τον πρώτο τρόπο οι εντολές δίνονται χειροκίνητα μέσω ενός τερματικού παραθύρου. Με τον δεύτερο τρόπο, το GRMON λειτουργεί σαν GDB πύλη (gateway). Το GRMON έχει υλοποιηθεί χρησιμοποιώντας τρία λειτουργικά επίπεδα: το επίπεδο εντολών, το επίπεδο αποσφαλμάτωσης των οδηγών (debug driver), και το επίπεδο αποσφαλμάτωσης της διασύνδεσης (debug interface). Το επίπεδο εντολών παίρνει εντολές από τον χρήστη και τις μεταφέρει σε ένα “κέλυφος” (Shell) Tcl. Υπάρχει επίσης η δυνατότητα να ξεκινήσει μια υπηρεσία GDB ‘server’, η οποία θα έχει το δικό της κέλυφος. Το κάθε κέλυφος έχει τις δικές του εντολές και μεταβλητές. Πολλές εντολές εξαρτώνται από οδηγούς και αποτυγχάνουν αν ο πυρήνας δεν είναι παρών στο σύστημα. Το επίπεδο αποσφαλμάτωσης των οδηγών ανιχνεύει και αρχικοποιεί τους πυρήνες. Το επίπεδο αποσφαλμάτωσης της διασύνδεσης υλοποιεί το πρωτόκολλο αποσφαλμάτωσης για κάθε υποστηριζόμενη διεπαφή. Ποια διεπαφή θα χρησιμοποιηθεί επιλέγεται μέσα από την εντολή εκκίνησης του GRMON [29].

Όταν το GRMON συνδέεται αρχικά με το σύστημα, το σαρώνει ώστε να εντοπίσει ποιοι πυρήνες IP είναι συνδεδεμένοι σε αυτό. Αυτό γίνεται διαβάζοντας την πληροφορία “σύνδεσε και τρέξε”, η οποία βρίσκεται συνήθως στη διεύθυνση 0xfffff000 του διαύλου AHB. Ο αποσφαλματωτής αναγνωρίζει τα IP που είναι συνδεδεμένα και υλοποιεί αν απαιτείται κάποιες λειτουργίες αρχικοποίησης σε αυτά. Για παράδειγμα σε ένα ελεγκτή μνήμης, η λειτουργία αρχικοποίησης περιλαμβάνει την ανίχνευση του μεγέθους της RAM μνήμης που είναι προσαρμοσμένη στο σύστημα. Για την αρχικοποίηση του UART απαιτείται η ρύθμιση του ρυθμού λειτουργίας (baud rate) και των FIFOs. Μετά την ολοκλήρωση της αρχικοποίησης, εμφανίζονται οι ρυθμίσεις του συστήματος [29].

```
GRMON2 LEON debug monitor v2.0.15 professional version

Copyright (C) 2012 Aeroflex Gaisler - All rights reserved.
For latest updates, go to http://www.gaisler.com/
Comments or bug-reports to support@gaisler.com


GRLIB build version: 4111
Detected frequency:  40 MHz

Component                            Vendor
LEON3 SPARC V8 Processor             Aeroflex Gaisler
AHB Debug UART                       Aeroflex Gaisler
JTAG Debug Link                      Aeroflex Gaisler
GRSPW2 SpaceWire Serial Link         Aeroflex Gaisler
LEON2 Memory Controller              European Space Agency
AHB/APB Bridge                       Aeroflex Gaisler
LEON3 Debug Support Unit             Aeroflex Gaisler
Generic UART                         Aeroflex Gaisler
Multi-processor Interrupt Ctrl.      Aeroflex Gaisler
Modular Timer Unit                   Aeroflex Gaisler
General Purpose I/O port             Aeroflex Gaisler

Use command 'info sys' to print a detailed report of attached cores

grmon2>
```

Εικόνα 2.1 GRMON Εμφάνιση ρυθμίσεων [29]

Πιο λεπτομερείς πληροφορίες μπορούν να εμφανιστούν χρησιμοποιώντας την εντολή ‘info sys’. Η λεπτομερής ανάλυση του συστήματος περιλαμβάνει πληροφορίες για την αντιστοίχιση των διευθύνσεων, την αναγνώριση των διακοπών του συστήματος και τις ρυθμίσεις για κάθε πυρήνα. Πληροφορίες για το σε ποιον δίαυλο επικοινωνίας AMBA (AHB ή APB) είναι συνδεδεμένοι οι πυρήνες μπορούν να εμφανιστούν χρησιμοποιώντας την επιλογή ‘-v’. Το GRMON αποδίδει ένα μοναδικό όνομα σε κάθε πυρήνα, το οποίο εμφανίζεται στα αριστερά [29].

```
grmon2> info sys
  cpu0      Aeroflex Gaisler  LEON3 SPARC V8 Processor
            AHB Master 0
  ahbuart0  Aeroflex Gaisler  AHB Debug UART
            AHB Master 1
            APB: 80000700 - 80000800
            Baudrate 115200, AHB frequency 40000000.00
  ahbjtag0  Aeroflex Gaisler  JTAG Debug Link
            AHB Master 2
  grspw0    Aeroflex Gaisler  GRSPW2 SpaceWire Serial Link
            AHB Master 3
            APB: 80000A00 - 80000B00
            IRQ: 10
            Number of ports: 1
  mctrl0    European Space Agency  LEON2 Memory Controller
            AHB: 00000000 - 20000000
            AHB: 20000000 - 40000000
            AHB: 40000000 - 80000000
            APB: 80000000 - 80000100
            8-bit prom @ 0x00000000
            32-bit sdram: 1 * 64 Mbyte @ 0x40000000
            col 9, cas 2, ref 7.8 us
  apbmst0   Aeroflex Gaisler  AHB/APB Bridge
            AHB: 80000000 - 80100000
  dsu0      Aeroflex Gaisler  LEON3 Debug Support Unit
            AHB: 90000000 - A0000000
            AHB trace: 128 lines, 32-bit bus
            CPU0:  win 8, hwbp 2, itrace 128, V8 mul/div, srmmu, lddel 1
                   stack pointer 0x43fffff0
                   icache 2 * 4096 kB, 32 B/line lru
                   dcache 1 * 4096 kB, 16 B/line
  uart0     Aeroflex Gaisler  Generic UART
            APB: 80000100 - 80000200
            IRQ: 2
            Baudrate 38461
  irqmp0    Aeroflex Gaisler  Multi-processor Interrupt Ctrl.
            APB: 80000200 - 80000300
  gptimer0  Aeroflex Gaisler  Modular Timer Unit
            APB: 80000300 - 80000400
            IRQ: 8
            8-bit scalar, 2 * 32-bit timers, divisor 40

  grgpio0   Aeroflex Gaisler  General Purpose I/O port
            APB: 80000800 - 80000900
```

Εικόνα 2.2 Χρησιμοποίηση της εντολής ‘info sys’ στο GRMON [29]

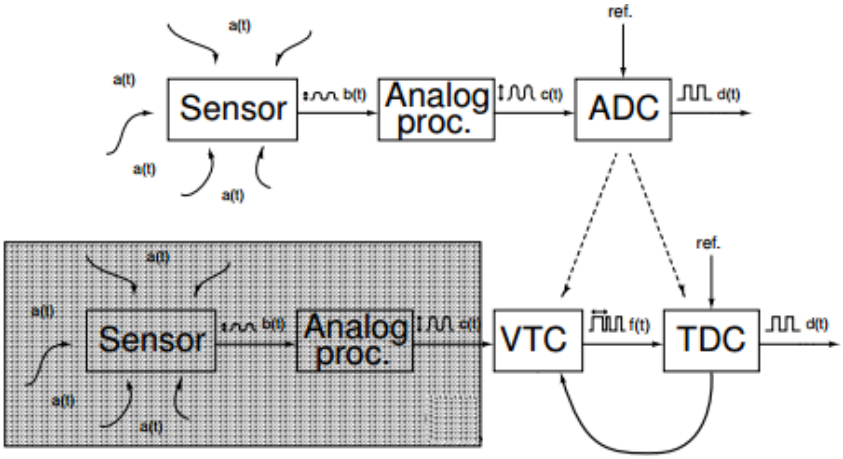
Πιο λεπτομερείς οδηγίες για την εγκατάσταση και λειτουργία του GRMON, την σύνθεση και την προσομοίωση βρίσκονται στο παράρτημα Α.

3. ΔΙΕΠΑΦΕΣ ΑΙΣΘΗΤΗΡΩΝ

3.1 Εισαγωγή

Η χρήση των αισθητήρων συνεχώς αυξάνεται στο πεδία των εφαρμογών (αυτοκινητοβιομηχανία, ασύρματα δίκτυα, υγεία, καταναλωτικά ηλεκτρονικά κ.α.). Τα σήματα των αισθητήρων πρέπει να προ-επεξεργαστούν πριν την τροφοδότηση τους στο υψηλότερο επίπεδο του συστήματος. Αυτή η προ-επεξεργασία γίνεται με τη χρήση των διεπαφών του αισθητήρα (sensor interfaces).

Παραδοσιακά μια διεπαφή αισθητήρα αποτελείται από ένα αισθητήρα, ο οποίος μετατρέπει μια φυσική ποσότητα a(t) σε ένα ηλεκτρικό αναλογικό σήμα b(t). Στη συνέχεια μια σειρά αναλογικών επεξεργασιών μετατρέπει το σήμα σε ένα πιο εύχρηστο μεγαλύτερο σήμα c(t) και στη συνέχεια το σήμα ψηφιοποιείται σε ένα σήμα d(t) (σχήμα 3.1).



Σχήμα 3.1: Διεπαφές αισθητήρα [30]

Αυτές οι αρχιτεκτονικές έχουν διάφορα προβλήματα, όπως αρκετό θόρυβο. Όταν όμως το σήμα μεταφέρεται στο πεδίο της συχνότητας πολλά από αυτά τα προβλήματα παύουν να υπάρχουν. Οι μετατροπείς βασισμένοι στο χρόνο (time-based) μπορούν να αντικαταστήσουν τους παραδοσιακούς μετατροπείς από αναλογικό-σε-ψηφιακό σήμα (ADC), όπως φαίνεται και στο σχήμα 3.1. Ο μετατροπέας τάσης-σε-χρόνο (VTC) μετατρέπει το αναλογικό σήμα c(t) σε πληροφορίες χρόνου ή συχνότητας και ο μετατροπέας χρόνου-σε-ψηφιακό σήμα (TDC) ψηφιοποιεί την πληροφορία, με την βοήθεια μιας συχνότητας αναφοράς.

Στα πλαίσια της διπλωματικής έχει υλοποιηθεί μια αντίστοιχη διεπαφή αισθητήρων με την οποία μετατρέπονται οι αναλογικές τιμές ενός αισθητήρα σε ψηφιακές. Για τον σκοπό αυτό οι πληροφορίες από τον αισθητήρα μετατρέπονται από ένα αναλογικό σήμα (χωρητικότητα) σε ένα άλλο (συχνότητα) χρησιμοποιώντας έναν μετατροπέα. Μετέπειτα αυτή η πληροφορία μπορεί να επεξεργαστεί με την χρήση του επεξεργαστικού συστήματος που έχει ενσωματωθεί.

3.2 Το κύκλωμα της διεπαφής αισθητήρων

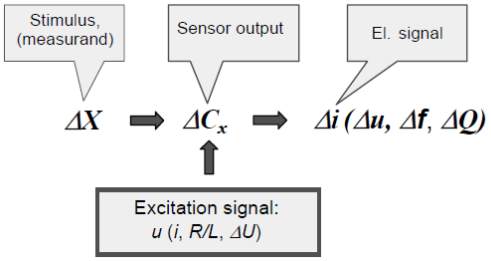
Ο σκοπός του κυκλώματος που θα αναλυθεί παρακάτω, το οποίο ουσιαστικά είναι μια διεπαφή αισθητήρων, είναι η διασύνδεση των αισθητήρων με το συνολικό κύκλωμα και με τον επεξεργαστή. Το κύκλωμα διεπαφής αφορά συστοιχίες αισθητήρων τύπου χωρητικότητας. Για την επεξεργασία των σημάτων η χωρητικότητα μετατρέπεται σε συχνότητα. Αυτή η μετατροπή υλοποιείται στο παρόν κύκλωμα χρησιμοποιώντας μια παραλλαγή ενός ταλαντωτή δακτυλίου (ring oscillator) και έναν μετρητή (counter). Το κύκλωμα έχει υλοποιηθεί μέσα σε μια συσκευή FPGA χρησιμοποιώντας τη γλώσσα VHDL και είναι πλήρως ψηφιακό. Ακόμα για την διασύνδεση των αισθητήρων πρέπει να

χρησιμοποιηθεί και ένα κύκλωμα Schmitt trigger. Αυτό το κύκλωμα βοηθάει στο να υπάρχει σταθερή συχνότητα και να μπορούν να μετρηθούν πολύ μικρές χωρητικότητες [31]. Σε κάποια FPGAs το Schmitt trigger κύκλωμα υπάρχει ενσωματωμένο, όπως π.χ. στο MAXV και στο MAX10. Σε άλλα πρέπει να προστεθεί με εξωτερικό κύκλωμα. Το κύκλωμα είναι παραμετροποιήσιμο και δεν λειτουργεί με συγκεκριμένο αριθμό αισθητήρων αλλά μπορούν να προστεθούν και να αφαιρεθούν αισθητήρες με κάποιες μικρές αλλαγές. Ο χρήστης μπορεί να ορίζει πότε να ξεκινήσει μια μέτρηση, να ελέγχει την σειρά των αισθητήρων από τους οποίους θα γίνεται η μέτρηση και να παίρνει τα αποτελέσματα όποτε το επιθυμεί. Ουσιαστικά το κύκλωμα της διεπαφής των αισθητήρων είναι η καρδιά του συστήματος, η οποία συνεργάζεται με το “μυαλό” που είναι ο επεξεργαστής, και μαζί επιτρέπουν την άμεση επαφή του χρήστη με τους αισθητήρες και την επεξεργασία των μετρήσεων με “έξυπνο” και αποδοτικό τρόπο.

Στη συνέχεια αναλύονται κάποιες έννοιες και παρουσιάζονται θεωρητικά στοιχεία τα οποία είναι σημαντικά για την κατανόηση του κυκλώματος.

3.3 Αισθητήρες χωρητικότητας

Η αισθητήρες χωρητικότητας χρησιμοποιούνται πάρα πολύ στη βιομηχανία, καθώς έχουν απλή κατασκευή και καλή απόδοση. Η τιμή της χωρητικότητας εξαρτάται από την τιμή της μεταβλητής εισόδου (του μετρούμενου φυσικού μεγέθους). Φυσιολογικά, η κύρια λειτουργία της διεπαφής του αισθητήρα χωρητικότητας είναι η μέτρηση της χωρητικότητας. Αυτό μπορεί να γίνει με διάφορους τρόπους όπως φαίνεται και στο σχήμα 3.2 [32].



Σχήμα 3.2: Διασύνδεση του αισθητήρα χωρητικότητας με διαφορετικά σήματα διέγερσης αναφοράς [32]

Με αρμονικά σήματα διεγέρσεως (u ή i) μπορεί να μετρηθεί η ηλεκτρική επαγωγική αντίσταση του άγνωστου πυκνωτή. Γνωρίζοντας την συχνότητα, την διέγερση και τις τιμές των μετρούμενων σημάτων μπορεί να βρεθεί η τιμή της άγνωστης χωρητικότητας. Ένας άλλος τρόπος για να μετρηθεί η χωρητικότητα είναι να γίνει κομμάτι ενός κυκλώματος ταλαντωτή που βρίσκει τη συχνότητα, μαζί με ένα ακόμα παθητικό στοιχείο. Για ένα πρώτης τάξης ταλαντωτή συνήθως αυτό είναι μια αντίσταση, ενώ για ένα δεύτερης τάξης ταλαντωτή αυτό μπορεί να είναι ένα πηνίο. Σε αυτήν την περίπτωση η πληροφορία για την υπό μέτρηση χωρητικότητα είναι η τιμή της συχνότητας/περιόδου του σήματος. Εναλλακτικά, η χωρητικότητα του αισθητήρα C_X μπορεί να εξαχθεί μετρώντας το φορτίο $Q_X=C_X.U_{REF}$, που είναι αποθηκευμένο [32].

Υπάρχουν τρεις βασικές παράμετροι οι οποίες μπορούν να χρησιμοποιηθούν για την αξιολόγηση της διεπαφής ενός αισθητήρα χωρητικότητας: (1) η ανάλυση (resolution), (2) ο χρόνος που χρειάζεται για να πραγματοποιηθεί μια μέτρηση, και (3) η σταθερότητα. Σε άλλες εφαρμογές και άλλες παράμετροι μπορεί να είναι σημαντικές, όπως η κατανάλωση ενέργειας, η γραμμικότητα και το δυναμικό εύρος. Η επιθυμητή ακρίβεια πετυχαίνετε με βαθμονόμηση σε σχέση με μια ακριβή αναφορά (calibration).

Η χωρητικότητα των αισθητήρων (πυκνωτών) εξαρτάται από τρείς παράγοντες [31]:

1. Από το μέγεθος της επιφάνειας των οπλισμών τους,

2. Από την απόσταση μεταξύ των οπλισμών τους και
3. Από το υλικό που παρεμβάλλεται μεταξύ των οπλισμών, δηλαδή από το διηλεκτρικό τους.

Η γενική σχέση που συνδυάζει τους τρείς προηγούμενους παράγοντες είναι η εξής:

$$C = \frac{\text{Επιφάνεια} * \text{Διηλεκτρική\_σταθερά}}{\text{Απόσταση}}$$

3.4 Ταλαντωτής δακτυλίου

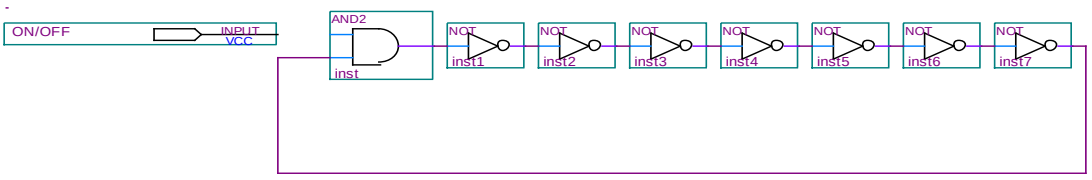
Ο ταλαντωτής δακτυλίου είναι ένα κύκλωμα που συνδυάζει διάφορα στάδια καθυστέρησης, τα οποία είναι συνδεδεμένα σε ένα κλειστό βρόχο. Οι ταλαντωτές δακτυλίου έχουν μεγάλο ενδιαφέρον καθώς, έχουν κάποια πολύ χρήσιμα χαρακτηριστικά. Κάποια από αυτά τα χαρακτηριστικά είναι [33]: 1) μπορούν να σχεδιασθούν πολύ εύκολα, χρησιμοποιώντας την τελευταία τεχνολογία ολοκληρωμένων κυκλωμάτων (CMOS,BiCMOS), 2) μπορούν να επιτευχθούν ταλαντώσεις με χαμηλή τάση, 3) μπορούν να επιτευχθούν υψηλές συχνότητες ταλάντωσης με μικρή ισχύ, 4) μπορεί να συντονιστεί ηλεκτρικά (tuned), 5) παρέχει μεγάλο εύρος συντονισμού, 6) μπορεί να παρέχει πολυφασικές εξόδους, λόγω της δομής του. Αυτές οι έξοδοι μπορούν να συνδυαστούν για την δημιουργία πολυφασικών ρολογιών, κάτι το οποίο είναι πολύ χρήσιμο σε πλήθος τηλεπικοινωνιακών εφαρμογών.

Η συχνότητα ταλάντωσης του ταλαντωτή δακτυλίου εξαρτάται από την καθυστέρηση διάδοσης τ_d κάθε σταδίου και τον αριθμό των σταδίων. Για την επίτευξη αυτοσυντηρούμενης ταλάντωσης, ο δακτύλιος πρέπει να παρέχει μια διαφορά φάσης 2π. Σε έναν ταλαντωτή δακτυλίου m σταδίων, κάθε στάδιο παρέχει διαφορά φάσης π/m και η μεταφορά στο DC παρέχει την υπόλοιπη διαφορά φάσης π. Για το λόγο αυτό, το σήμα ταλάντωσης πρέπει να περάσει από κάθε στάδιο m καθυστέρησης μια φορά ώστε να παραχθεί η διαφορά φάσης π σε χρόνο $m\tau_d$ και πρέπει να περάσει από κάθε στάδιο μια δεύτερη φορά για να παραχθεί η υπόλοιπη διαφορά φάσης π, σε μια περίοδο χρόνου $2m\tau_d$. Έτσι η συχνότητα της ταλάντωσης δίνεται από τον τύπο [33]:

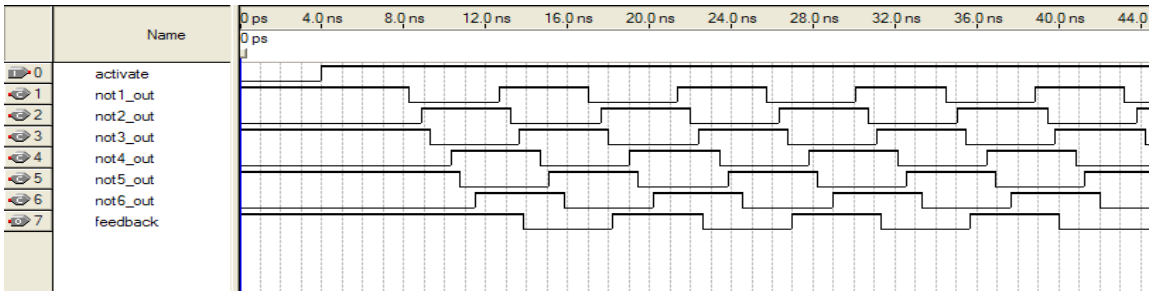
$$f_o = \frac{1}{2m\tau_d} \quad (1)$$

Η συχνότητα ταλάντωσης ενός ταλαντωτή δακτυλίου μπορεί να καθοριστεί από το τ_d, το οποίο εξαρτάται από παραμέτρους του κυκλώματος. Η μεγαλύτερη δυσκολία στον καθορισμό του τ_d προέρχεται από τις μη-γραμμικότητες και τα παρασιτικά του κυκλώματος.

Από την παραπάνω εξίσωση φαίνεται ότι ο αριθμός των αντιστροφέων ο οποίος χρησιμοποιείται σε ένα ταλαντωτή δακτυλίου και η καθυστέρηση διάδοσης του κάθε σταδίου περιορίζουν τη συχνότητα ταλάντωσης. Στα σχήματα 3.3 και 3.4 που ακολουθούν φαίνεται ένας ταλαντωτής δακτυλίου και η προσομοίωση του [31]:



Σχήμα 3.3: Ταλαντωτής δακτυλίου με πύλη AND στην είσοδο, η οποία αποτελεί διακόπτη ενεργοποίησης του ταλαντωτή [31]

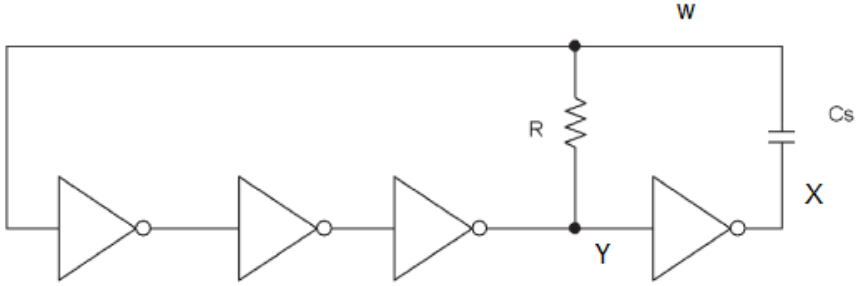


Σχήμα 3.4: Ενδιάμεσα σήματα αντιστροφέων ταλαντωτή δακτυλίου σχήματος 3.3 [31]

3.5 Μετατροπή του ταλαντωτή δακτυλίου, για την υλοποίηση της διεπαφής αισθητήρων.

Στα FPGA οι λογικές συναρτήσεις υλοποιούνται με ρυθμιζόμενα λογικά μπλοκ (CLB). Αυτά περιέχουν πίνακες αναφοράς (LUT) και καταχωρητές. Η καθυστέρηση ενός LUT εξαρτάται από την τελική δυαδική λογική που υλοποιεί. Τα λογικά μπλοκ είναι συνδεδεμένα μεταξύ τους με πίνακες διακοπτών (switch matrices). Για την σύνδεση μεταξύ των CLBs χρησιμοποιούνται δύο ή περισσότεροι πίνακες διακοπτών, αυτή η σύνδεση λέγεται δρομολόγηση (routing). Ωστόσο, υπάρχουν δύο βασικά προβλήματα για την υλοποίηση ενός ταλαντωτή δακτυλίου σε συσκευές FPGA, η ευαισθησία στη θερμοκρασία και η σύζευξη (crosstalk) μεταξύ των σημάτων.

Η θερμοκρασία του κυκλώματος επηρεάζει τη συχνότητα ταλάντωσης ενός ταλαντωτή δακτυλίου μέσω διαφόρων παραμέτρων. Η τάση κατωφλίου των τρανζίστορ CMOS μειώνεται 2-4mV/kelvin [36]. Τα ακριβή νούμερα εξαρτώνται από το επίπεδο προσμείξεων του ημιαγωγού. Αυτή η αλλαγή στη τάση κατωφλίου οδηγεί σε υψηλότερα ρεύματα διαρροής σε υψηλότερες θερμοκρασίες και έτσι το κύκλωμα γίνεται πιο γρήγορο. Επίσης, η κινητικότητα των φορέων (μ) στο πυρίτιο εξαρτάται από τη θερμοκρασία. Η επαγωγική σύζευξη εμφανίζεται σε πολύ υψηλές συχνότητες οι οποίες δεν υπάρχουν στα σημερινά FPGA. Το πρόβλημα είναι τα φαινόμενα που δημιουργούνται λόγω των χωρητικών συζεύξεων. Ένα σήμα παρουσιάζει χωρητικότητα ως προς τη γη και τα γειτονικά σήματα. Αν το επίπεδο της τάσης γειτονικών σημάτων αλλάζει, μπορεί να παρατηρηθεί μια διαφορά στη χωρητικότητα σύζευξης. Το πόσο επηρεάζει αυτό την καθυστέρηση εξαρτάται από τις χωρητικότητες μεταξύ των σημάτων (λόγω σύζευξης) και της γης. Ακόμα ένα πρόβλημα για την υλοποίηση ενός ταλαντωτή δακτυλίου σε FPGA είναι ότι οι ακροδέκτες (pins) οδηγούνται από ενδιάμεσα κυκλώματα οδήγησης (buffers), οι οποίες δίνουν μεγάλο ρεύμα. Άρα εάν συνδεθεί οποιαδήποτε χωρητικότητα, δεν θα μεταβληθεί η συχνότητα του κυκλικού ταλαντωτή καθώς η χωρητικότητα αυτή φορτίζεται μέσω των ενδιάμεσων οδηγών πολύ γρήγορα [37]. Για αυτούς τους λόγους υλοποιήθηκε μια παραλλαγή του κυκλώματος του ταλαντωτή δακτυλίου, η οποία φαίνεται στο σχήμα 3.5. Το νέο κύκλωμα αποτελείται από έναν ταλαντωτή δακτυλίου και μία αντίσταση (εξωτερική ή εσωτερική ανάλογα το ολοκληρωμένο κύκλωμα).



Σχήμα 3.5: Κύκλωμα ταλαντωτή. Όπου C_s η άγνωστη χωρητικότητα [37]

Το κύκλωμα αποτελείται από τέσσερις αντιστροφείς, μία αντίσταση R και το χωρητικό αισθητήρα. Η ταλάντωση παράγεται από αυτοδιέγερση, χωρίς να εφαρμοστούν εξωτερικά σήματα, εκτός από την πηγή τροφοδοσίας. Η περίοδος του ταλαντωτή του σχήματος 3.5 εξαρτάται κυρίως από την σταθερά χρόνου RC_s, καθώς θεωρείται ότι είναι αρκετά μεγαλύτερη από τις καθυστερήσεις των αντιστροφέων. Γενικά, όταν το λογικό κατώφλι των αντιστροφέων είναι ορισμένο στο $V_{DD}/2$, το κύκλωμα παρέχει μια συμμετρική κυματομορφή εξόδου με περίοδο [37]:

$$T = T_{\text{ch}} + T_{\text{dis}} = 2RC_s \ln 3 \tag{2}$$

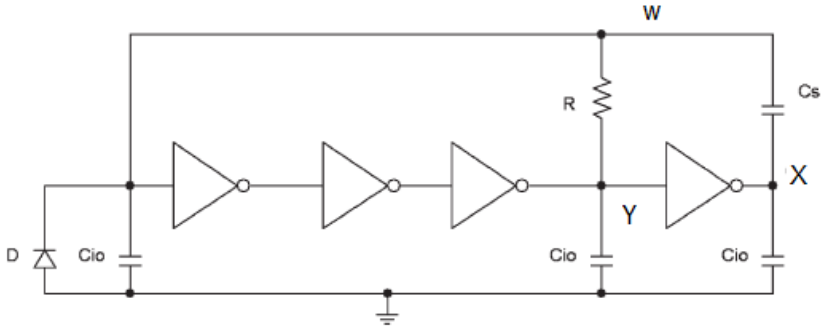
,όπου το T_{ch} και το T_{dis} είναι ο χρόνος φόρτισης και αποφόρτισης του πυκνωτή αντίστοιχα

Η μεταβολή της τάσης στο σημείο W του κυκλώματος, όταν η τάση τροφοδοσίας είναι V_{DD} είναι:

$$V_{\text{W max}} = 3V_{DD}/2 \tag{3}$$

$$V_{\text{W min}} = -V_{DD}/2 \tag{4}$$

Η αντίσταση και ο πυκνωτής στο σύστημα που υλοποιήθηκε βρίσκονται εξωτερικά του τσιπ έτσι οι ακροδέκτες εισόδου/εξόδου του τσιπ επηρεάζουν την απόδοση του κυκλώματος [37]. Τα στοιχεία που προστίθενται στο κύκλωμα τροποποιούν την μεταβολή της τάσης στο σημείο W, όπως και την έξοδο του ταλαντωτή. Στο σχήμα 3.6 φαίνεται το πραγματικό κύκλωμα του ταλαντωτή με τις παρασιτικές χωρητικότητες και τη δίοδο προστασίας που βρίσκεται στις θύρες εισόδου/εξόδου του ολοκληρωμένου κυκλώματος.



Σχήμα 3.6: Πραγματικό κύκλωμα ταλαντωτή , με παρασιτικά στοιχεία και δίοδο προστασίας εισόδου/εξόδου [37]

Για την βελτίωση της απόδοσης του κυκλώματος και για την εξασφάλιση ταλαντώσεων ακόμα και για μικρές χωρητικότητες, ο 1ος αντιστροφέας του ταλαντωτή δακτυλίου έχει αντικατασταθεί με ένα Schmitt Trigger. Οι παρακάτω τύποι έχουν εξαχθεί για την φόρτιση και την αποφόρτιση του κυκλώματος όπως και για την μεταβολή της τάσης στο σημείο W [37]:

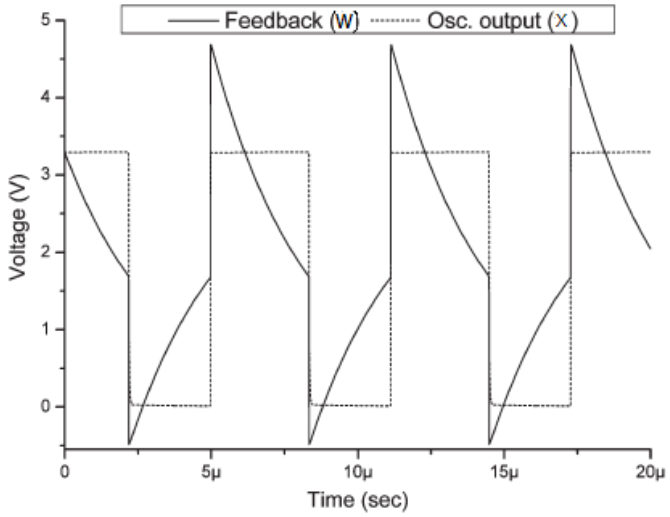
$$T_{\text{dis}} = R(C_s + C_{\text{io}}) \ln \frac{V_{\text{W max}}}{V_{\text{ltl}}} \tag{5}$$

$$T_{\mathrm{ch}} = R(C_s + C_{\mathrm{io}}) \ln \frac{V_{DD} - V_{\mathrm{W\,min}}}{V_{DD} - V_{\mathrm{lth}}} \tag{6}$$

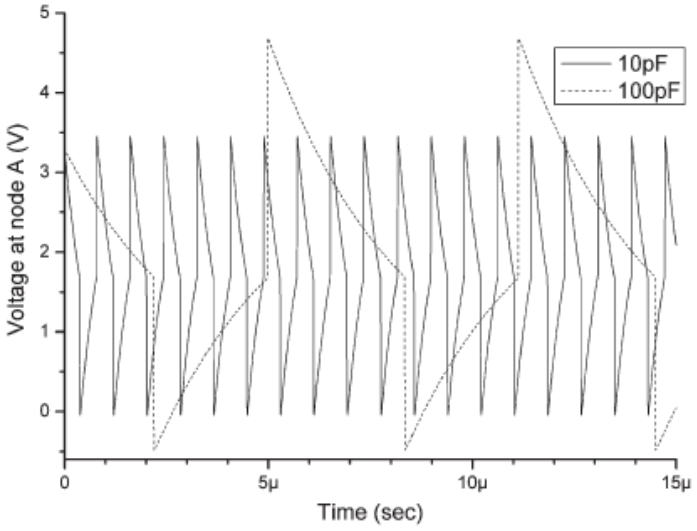
$$V_{\mathrm{W\,max}} = V_{DD} \frac{C_s}{C_s + C_{\mathrm{io}}} + V_{\mathrm{lth}} \tag{7}$$

$$V_{\mathrm{W\,min}} = \max\left(-V_{\mathrm{diode}}, V_{DD} - \left(V_{DD} \frac{2C_s + C_{\mathrm{io}}}{C_s + C_{\mathrm{io}}} - V_{\mathrm{ltl}}\right)\right) \tag{8}$$

,όπου το V_{ltl} και το V_{lth} είναι το κατώφλι μεταβολής του Schmitt trigger από υψηλό σε χαμηλό δυναμικό και από χαμηλό σε υψηλό δυναμικό αντίστοιχα. Το C_{io} είναι η παρασιτική χωρητικότητα εισόδου/εξόδου. Στο σχήμα 3.7 φαίνεται η προσομοίωση για το σημείο W και X του κυκλώματος με χωρητικότητα C_S=100pf και στο σχήμα 3.8 η προσομοίωση του σημείου W για χωρητικότητες 10pf και 100pf.



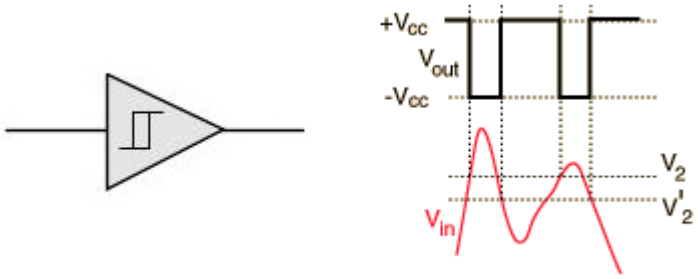
Σχήμα 3.7: Προσομοίωση ταλαντωτή. Σημείο W και X [37]



Σχήμα 3.8: Προσομοίωση ταλαντωτή στο σημείο W για χωρητικότητες 10pf και 100pf [37]

3.5.1 Schmitt trigger

Το κύκλωμα Schmitt trigger ουσιαστικά είναι ένας συγκριτής ο οποίος κάνει την έξοδο αρνητική όταν η είσοδος περνάει ένα θετικό κατώφλι τάσης. Στη συνέχεια χρησιμοποιεί θετική ανάδραση αρνητικής τάσης για την αποφυγή της επιστροφής στην προηγούμενη κατάσταση, μέχρι το σήμα εισόδου να περάσει ένα χαμηλό κατώφλι δυναμικού. Με αυτό τον τρόπο σταθεροποιεί την μεταγωγή (switching) από την ταχεία ενεργοποίηση από τον θόρυβο όταν περνάει το σημείο ενεργοποίησης. Δηλαδή παρέχει ανάδραση, η οποία δεν είναι αντεστραμμένη κατά φάση. Στην περίπτωση που το σήμα το οποίο ανατροφοδοτείται είναι ένα αρνητικό σήμα κρατάει την έξοδο σταθερή σε αρνητική τάση μέχρι η τάση να πέσει κάτω από ένα κατώφλι.



Σχήμα 3.9: Σύμβολο Schmitt trigger και Vin/Vout διάγραμμα [38]

3.6 Μετρητές συχνότητας

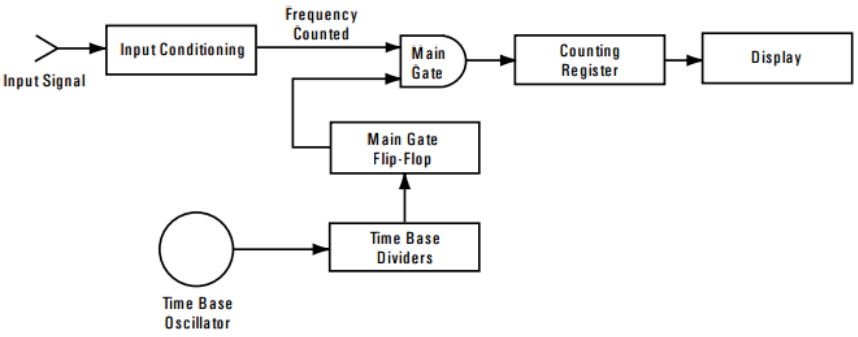
3.6.1 Τυπικός μετρητής

Ένας τυπικός μετρητής συχνότητας είναι μια ψηφιακή συσκευή η οποία μετράει τη συχνότητα ενός σήματος στη είσοδό της. Η συχνότητα f, μπορεί να καθοριστεί από τον αριθμό των κύκλων του σήματος σε μια προκαθορισμένη περίοδο χρόνου. Αυτό μπορεί να αναπαρασταθεί από την παρακάτω εξίσωση [39]:

$$f = n / t \qquad (9)$$

,όπου *n* είναι ο αριθμός των κύκλων που παράγει το σήμα εισόδου και *t* το χρονικό παράθυρο στο οποίο συμβαίνουν. Όταν το *t=1s*, τότε η συχνότητα μετριέται σε n κύκλους ανά δευτερόλεπτο ή σε *n* Hz.

Όπως φαίνεται από την εξίσωση 9, η συχνότητα *f* ενός σήματος με τον τυπικό μετρητή, μετριέται υπολογίζοντας τον αριθμό των κύκλων, *n* και διαιρώντας με το παράθυρο χρόνου *t*. Το διάγραμμα βαθμίδων ενός τέτοιου μετρητή φαίνεται παρακάτω (εικόνα 3.10).



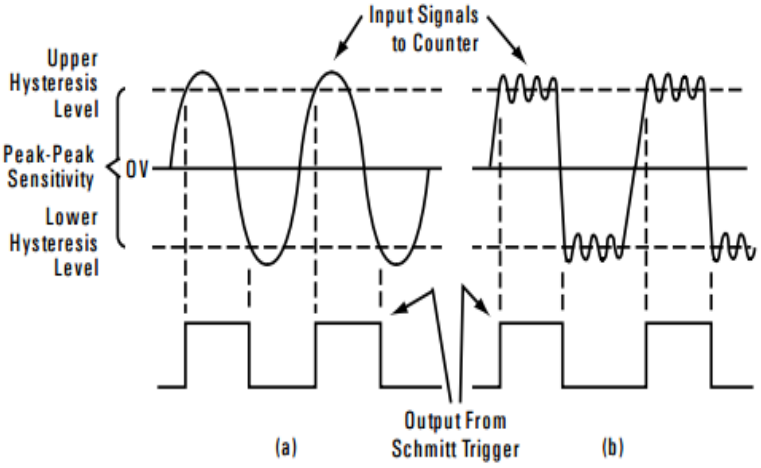
Σχήμα 3.10: Διάγραμμα βαθμίδων ενός τυπικού μετρητή συχνότητας [39]

Το σήμα εισόδου, αρχικά, προσαρμόζεται σε μια μορφή η οποία είναι συμβατή με το εσωτερικό κύκλωμα. Το σήμα που εμφανίζεται στην είσοδο είναι μια σειρά παλμών, όπου κάθε παλμός αναπαριστά ένα κύκλο ή ένα γεγονός. Όταν η κεντρική πύλη (main

gate) είναι ανοικτή, οι παλμοί επιτρέπεται να περνάνε και να μετρούνται από τον μετρητή. Το άνοιγμα και το κλείσιμο της κεντρικής πύλης ελέγχεται από τη βάση χρόνου *t* (time base). Από την εξίσωση 9, είναι φανερό ότι η ακρίβεια της μέτρησης συχνότητας εξαρτάται από την ακρίβεια του *t*. Ο διαιρέτης βάσης χρόνου (Time Base Divider) παίρνει το σήμα από τον ταλαντωτή βάσης χρόνου (time base oscillator) ως είσοδο και παρέχει στην έξοδο μια σειρά παλμών. Ο χρόνος *t*, καθορίζεται από τη περίοδο αυτής της σειράς παλμών. Ο συνολικός αριθμός παλμών που μετριέται από τον καταχωρητή του μετρητή (counting register) για τον επιλεγμένο χρόνο είναι η συχνότητα του σήματος εισόδου. Για παράδειγμα, αν ο συνολικός αριθμός που μετράει ο μετρητής είναι 50000 και ο επιλεγμένος χρόνος 1sec, τότε η συχνότητα του σήματος εισόδου είναι 50000Hz [39].

Η ευαισθησία του μετρητή καθορίζεται από την ελάχιστη τιμή του σήματος εισόδου που μπορεί να μετρηθεί. Συνήθως η ευαισθησία μετριέται σε τιμή RMS μιας ημιτονοειδούς εισόδου. Το κέρδος του ενισχυτή και η διαφορά τάσης μεταξύ των επιπέδων υστέρησης του κυκλώματος Schmitt trigger καθορίζουν την ευαισθησία του μετρητή. Αρχικά, μπορεί να φαίνεται ότι όσο μεγαλύτερη ευαισθησία έχει ο μετρητής τόσο το καλύτερο, αλλά αυτό δεν συμβαίνει καθώς στον τυπικό μετρητή ο θόρυβος μπορεί να προκαλέσει λάθος σκανδαλισμό (triggering). Η βέλτιστη ευαισθησία εξαρτάται από την αντίσταση της εισόδου, καθώς όσο μεγαλύτερη είναι η αντίσταση τόσο καλύτερη ανοχή υπάρχει στο θόρυβο [39].

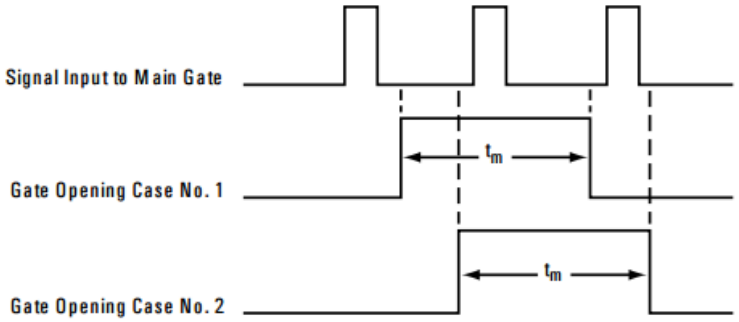
Για να αυξηθεί ο μετρητής κατά μια τιμή θα πρέπει η είσοδος να περάσει και το υψηλό και το χαμηλό επίπεδο υστέρησης του κυκλώματος Schmitt trigger. Για αυτό στη (b) περίπτωση, όπως φαίνεται στο σχήμα 3.11, το “κουδούνισμα” της εισόδου δεν οδηγεί σε αύξηση του μετρητή.



Σχήμα 3.11: “Κουδούνισμα” εισόδου μετρητή [39]

Στους μετρητές συχνότητας υπάρχουν δύο ειδών λάθη που μπορούν να μειώσουν τη ακρίβεια του κυκλώματος, το “±1 λάθος της μέτρησης” και το “λάθος του συνολικού χρόνου βάσης”.

Όταν ένας ηλεκτρονικός μετρητής παίρνει μια μέτρηση, μπορεί να υπάρχει ένα λάθος ±1 στο λιγότερο σημαντικό ψηφίο. Αυτό μπορεί να συμβεί λόγω του μη συγχρονισμού μεταξύ του εσωτερικού ρολογιού του μετρητή και του σήματος εισόδου. Στο σχήμα 3.12 φαίνεται ότι η κεντρική πύλη είναι ανοικτή και στις δύο περιπτώσεις κατά το ίδιο διάστημα αλλά ο μη συγχρονισμός μεταξύ του εσωτερικού ρολογιού και του σήματος εισόδου προκαλεί διαφορετική τιμή στον μετρητή.



Σχήμα 3.12: ±1 λάθος του μετρητή [39]

Το λάθος χρόνου βάσης είναι το αποτέλεσμα της διαφοράς μεταξύ της πραγματικής τιμής της συχνότητας του ταλαντωτή χρόνου βάσης και της κανονικοποιημένης συχνότητας.

Συνολικά για την ακρίβεια του μετρητή συχνότητας ισχύει ότι το συνολικό λάθος της μέτρησης καθορίζεται ως το άθροισμα του ±1 λάθους της μέτρησης και του λάθους του συνολικού χρόνου βάσης. Το λάθος της συχνότητας λόγω του ±1 μπορεί να οριστεί ως [39]:

$\Delta f/f = \pm 1/f_{in}$, όπου το f_{in} είναι η συχνότητα εισόδου

Φαίνεται πως όσο μεγαλύτερη είναι η συχνότητα, τόσο μικρότερο είναι το λάθος λόγω του ±1 μετρήματος. Το λάθος στη συχνότητα λόγω του χρόνου βάσης, είναι ένας παράγοντας που συνήθως μετριέται σε μέρη ανά εκατομμύριο (ppm). Αν, για παράδειγμα, το λάθος του χρόνου βάσης είναι 1 στο εκατομμύριο (1 × 10^–6), το λάθος σε ένα σήμα 10 MHz είναι:

(1 × 10^–6) × 10^7 Hz ή 10 Hz.

Έτσι το λάθος στο συγκεκριμένο παράδειγμα λόγω του χρόνου βάσης είναι ±1 × 10^–6 και το λάθος λόγω του ±1 είναι ±1/10^7. Φαίνεται, λοιπόν, ότι για συχνότητες μικρότερες του 1 MHz το λάθος ±1 κυριαρχεί, ενώ σε μεγαλύτερες υποσκελίζεται από το λάθος χρόνου βάσης.

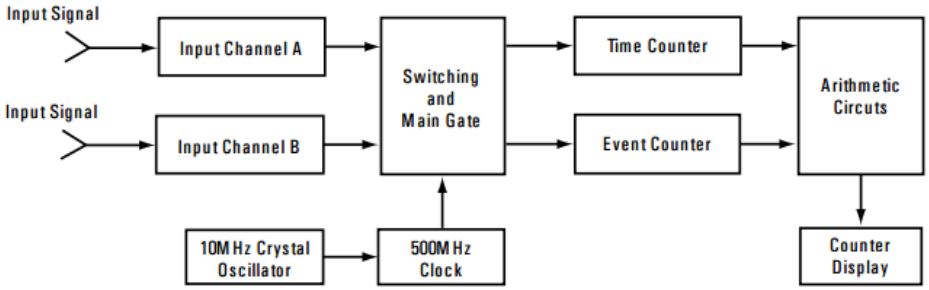
3.6.2 “ Αντίστροφος” μετρητής

Άλλο ένα είδος μετρητών συχνότητας είναι ο “ αντίστροφος” (Reciprocal) μετρητής. Ο αντίστροφος μετρητής μετράει πάντα την περίοδο του σήματος εισόδου. Για τον υπολογισμό της συχνότητας απαιτείται η αντιστροφή της τιμής της περιόδου. Αυτός ο μετρητής χρησιμοποιείται όλο και πιο πολύ καθώς προσφέρει δύο σημαντικά χαρακτηριστικά [39]:

- Το ±1 λάθος είναι ανεξάρτητο της συχνότητας της εισόδου.
- Το χαρακτηριστικό του υπολογισμού της περιόδου προσφέρει τη δυνατότητα ελέγχου της κεντρικής πύλης σε πραγματικό χρόνο.

Στο σχήμα 3.13 φαίνεται ένα παράδειγμα διαγράμματος βαθμίδων ενός αντίστροφου μετρητή. Ουσιαστικά, είναι το ίδιο με αυτό ενός τυπικού μετρητή, με τη διαφορά ότι το μέτρημα γίνεται με δύο διαφορετικούς καταχωρητές για το χρόνο και για τα γεγονότα. Ο μετρητής γεγονότων (Event Counter) δέχεται το μέτρημα των παλμών από το σήμα εισόδου και την ίδια στιγμή ο μετρητής χρόνου (time counter) δέχεται τον αριθμό των παλμών του εσωτερικού ρολογιού, για όσο χρονικό διάστημα η κεντρική πύλη είναι ανοικτή. Σε μια περίοδο μέτρησης, η κεντρική πύλη είναι ανοικτή για ακριβώς μια περίοδο του σήματος εισόδου. Κατά τη διάρκεια αυτού του χρονικού παραθύρου, ο μετρητής γεγονότων θα έχει μόνο ένα μέτρημα, ενώ ο μετρητής χρόνου θα έχει έναν αριθμό παλμών ρολογιού. Ο αριθμός των παλμών που έχουν μετρηθεί

πολλαπλασιάζεται με τη περίοδο του ρολογιού και έχει σαν αποτέλεσμα τη περίοδο του σήματος εισόδου [39].

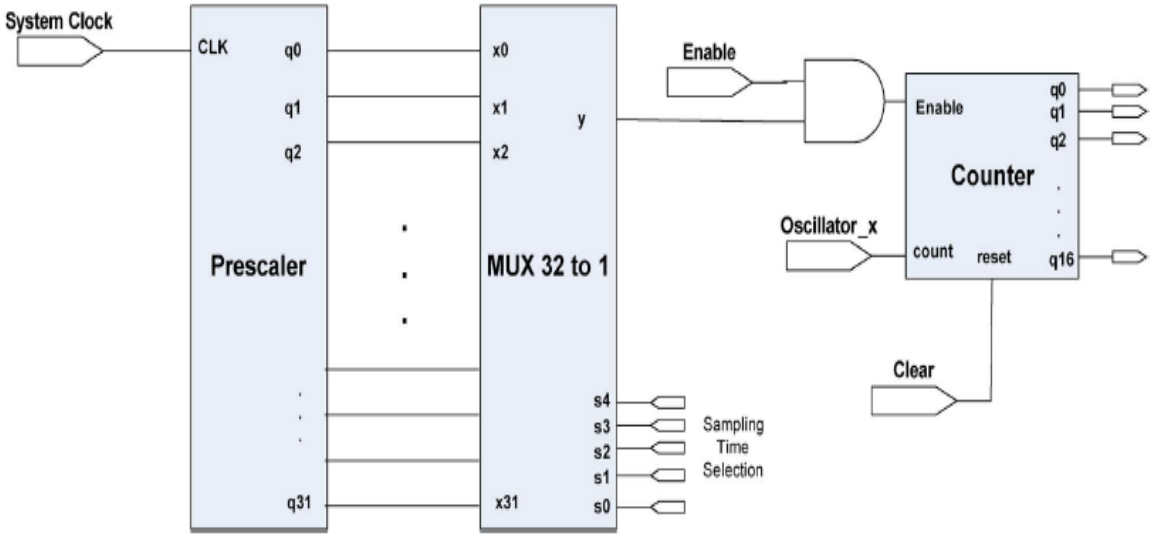


Σχήμα 3.13: Διάγραμμα βαθμίδων ενός αντίστροφου μετρητή [39]

3.6.3 Μετρητής συχνότητας της διπλωματικής

Στη διεπαφή των αισθητήρων της διπλωματικής η συχνότητα μετριέται με έναν μετρητή ειδικά διαμορφωμένο έτσι ώστε να μην απαιτείται χρήση μνήμης για την αποθήκευση των δεδομένων. Το κύκλωμα αναγνώρισης της συχνότητας φαίνεται στο σχήμα 3.14 [37]. Ο μετρητής ξεκινάει να λειτουργεί όταν δοθεί η κατάλληλη εντολή. Επίσης δίνεται ως εντολή το χρονικό παράθυρο της μέτρησης. Ο μετρητής μετράει τον αριθμό των παλμών κατά τη διάρκεια αυτού του χρονικού παραθύρου. Στο τέλος της μέτρησης μπορεί να υπολογιστεί η συχνότητα με τον εξής τρόπο.

Συχνότητα = (τιμή μετρητή) / (Περίοδος εσωτερικού ρολογιού * χρονικό παράθυρο)



Σχήμα 3.14: Κύκλωμα αναγνώρισης της συχνότητας [37]

Το διάστημα μέτρησης T_C παράγεται από έναν 32-μπιτ 'prescaler', ο οποίος διαιρεί το ρολόι του συστήματος σε δυνάμεις του 2, δημιουργώντας έτσι 32 διαφορετικά χρονικά παράθυρα. Τα χρονικά παράθυρα κυμαίνονται από *T*sys ως 2^31*T*sys, όπου το *T*sys είναι η περίοδος του ρολογιού του συστήματος [37]. Το T_C πρέπει να επιλεχτεί έτσι ώστε να μην γίνεται υπερχείλιση στον μετρητή και επίσης να μην βγάζει μηδενικές τιμές ο μετρητής. Ένας πολυπλέκτης 32-σε-1 χρησιμοποιείται για την επιλογή της κατάλληλης εξόδου του 'prescaler'. Η έξοδος του πολυπλέκτη συνδέεται με το σήμα ενεργοποίησης του 16-μπιτ μετρητή, ο οποίος χρησιμοποιεί την έξοδο του ταλαντωτή ως ρολόι. Έτσι η 16-μπιτ τιμή η οποία παράγεται από την έξοδο του μετρητή για ένα γνωστό χρονικό παράθυρο είναι απευθείας ανάλογη με τη συχνότητα ταλάντωσης του ταλαντωτή.

3.7 Πρωτόκολλο επικοινωνίας – Συνολικό κύκλωμα

Το σύστημα που είχε υλοποιηθεί στο [31] και στο οποίο στηρίζεται η διεπαφή της παρούσας διπλωματικής περιελάμβανε τον μετρητή που μόλις περιεγράφηκε και 16 ταλαντωτές. Το σύστημα επικοινωνεί με ένα υπολογιστή χρησιμοποιώντας μια σειριακή θύρα. Για να ελεγχθούν οι λειτουργίες του συστήματος έχει δημιουργηθεί ένα πρωτόκολλο επικοινωνίας. Στη συνέχεια γίνεται μια παρουσίαση αυτού του πρωτοκόλλου καθώς χρησιμοποιήθηκε παρόμοιο πρωτόκολλο και για την υλοποίηση της διπλωματικής.

Για την υλοποίηση του πρωτοκόλλου επικοινωνίας υπάρχουν τρείς καταχωρητές των 5-μπιτ. Με τον πρώτο ελέγχεται η έναρξη και διακοπή της λειτουργίας οποιουδήποτε από τους 16 ταλαντωτές, με το δεύτερο ελέγχεται ο χρόνος δειγματοληψίας και τέλος με τον τρίτο ελέγχονται τρία διαφορετικά σήματα. Τα σήματα αυτά είναι, η έναρξη και λήξη του παλμού δειγματοληψίας, το σήμα μετάδοσης δεδομένων από το FPGA προς τον υπολογιστή και το σήμα επιλογής μεταξύ χαμηλού (LOW) και υψηλού (HIGH) Byte του αποτελέσματος προς μετάδοση [31].

ΚΑΤΑΧΩΡΗΤΗΣ ΕΛΕΓΧΟΥ ΤΟΥ FPGA - FPGA_CONTROL

Bit:	7	6	5	4	3	2	1	0
	write	**S1**	**S2**	**B4**	**B3**	**B2**	**B1**	**B0**

write: Χρησιμοποιείται πάντα για να αποθηκευτούν νέες εντολές

S1-S0: Χρησιμοποιούνται για να επιλεχθεί ένας από τους τέσσερις καταχωρητές.

B4-B0: Δεδομένα προς καταχωρητή

Πίνακας 1: Καταχωρητής ελέγχου [31]

S1	S0	Select register
0	0	OSCILLATOR_CONTROL
0	1	SAMPLING_TIME_CONTROL
1	0	SIGNAL_CONTROL
1	1	-----

ΚΑΤΑΧΩΡΗΤΗΣ ΕΛΕΓΧΟΥ ΤΑΛΑΝΤΩΤΗ - OSCILLATOR_CONTROL

Bit	7	6	5	4	3	2	1	0
	write	**0**	**0**	**B4**	**B3**	**B2**	**B1**	**B0**

B4=NC

B3-B0: Επιλέγεται ποιος από τους 16 ταλαντωτές θα ενεργοποιηθεί

ΚΑΤΑΧΩΡΗΤΗΣ ΕΛΕΓΧΟΥ ΧΡΟΝΟΥ ΔΕΙΓΜΑΤΟΛΗΨΙΑΣ - SAMPLING_TIME_CONTROL

Bit:	7	6	5	4	3	2	1	0
	write	**0**	**1**	**B4**	**B3**	**B2**	**B1**	**B0**

B4-B0: Επιλέγεται ο διαιρέτης του ρολογιού, έτσι ώστε να προκύψει ο επιθυμητός χρόνος δειγματοληψίας.

Για παράδειγμα εάν διαιρεθεί ένα ρολόι 25MHz με το 2, η συχνότητα που προκύπτει θα είναι F=25.000.000/2=12.500.000Hz και η περίοδος T=1/12.500.000 Hz =80ns. Άρα ο χρόνος δειγματοληψίας θα είναι το μισό της περιόδου αυτής, δηλαδή 40ns. Αυτό συμβαίνει επειδή υπάρχει δειγματοληψία μόνο στη θετική παρυφή του παλμού που προκύπτει από τη διαιρεμένη συχνότητα[18].

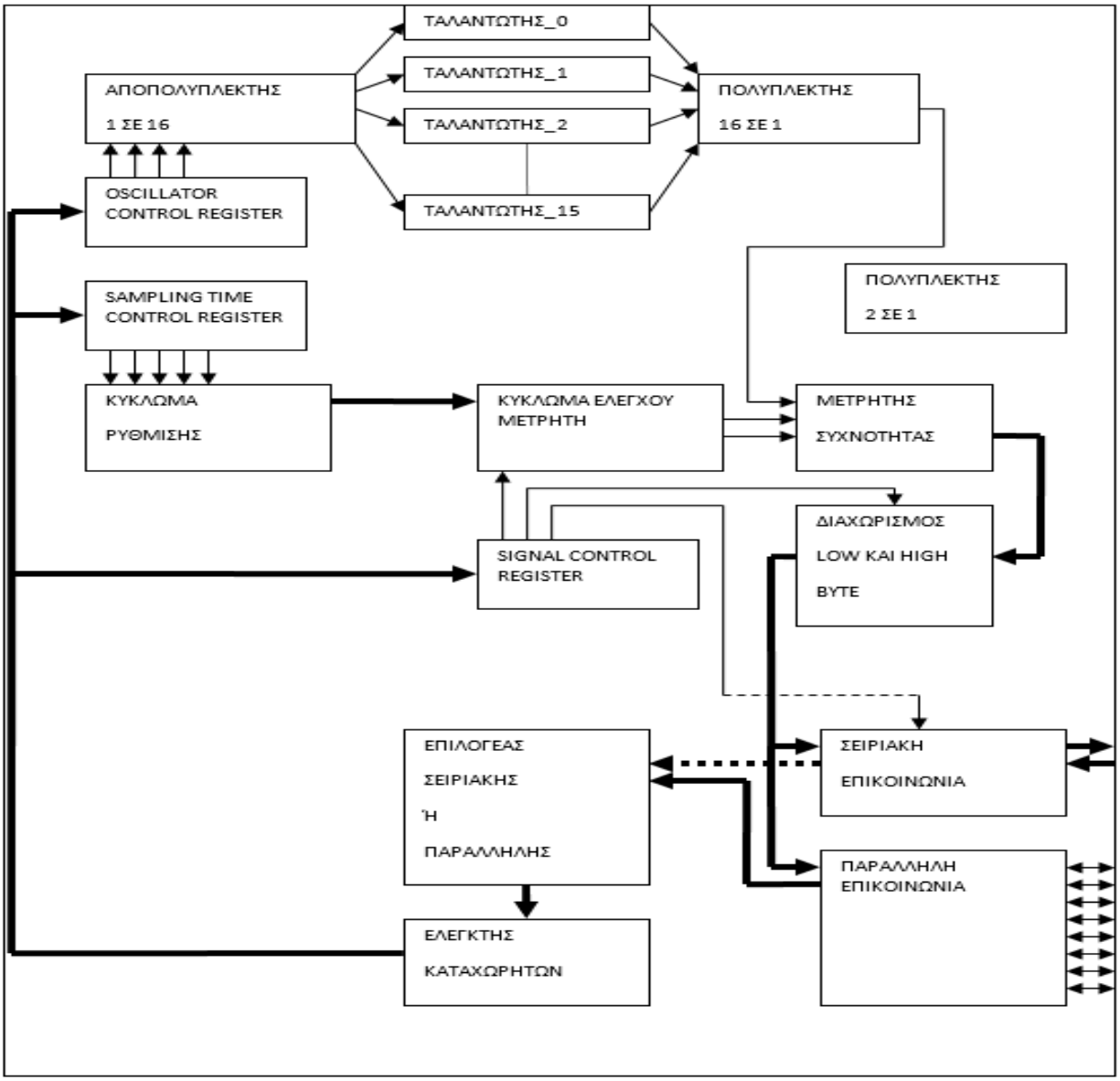
ΚΑΤΑΧΩΡΗΤΗΣ ΣΗΜΑΤΩΝ ΕΛΕΓΧΟΥ - SIGNAL_CONTROL

Bit:	7	6	5	4	3	2	1	0
	write	**1**	**0**	**X**	**X**	**B2**	**B1**	**B0**

X=NC
B2: Εκκίνηση δειγματοληψίας. Όταν έχει τη τιμή ‘1’ ξεκινάει ενώ με την τιμή ‘0’ τερματίζει.
B1: Μετάδοση δεδομένων στον υπολογιστή. Με την τιμή ‘1’ μεταδίδονται τα δεδομένα από το FPGA στον υπολογιστή (μόνο για χρήση σειριακής θύρας).
B0: Επιλογή μεταξύ χαμηλού και υψηλού Byte του αποτελέσματος της δειγματοληψίας. χαμηλό byte με τιμή ‘1’ και υψηλό byte με τιμή ‘0’.

Επειδή τα δεδομένα που στέλνονται είναι αποτελέσματα από μετρήσεις, δεν πρέπει να αλλοιωθούν. Για το λόγο αυτό αν στέλνονται εντολές διαδοχικά τότε μεταξύ τους πρέπει να υπάρχει μια χρονική καθυστέρηση [31]. Έτσι ώστε να μη πέσει η επόμενη εντολή πάνω στην προηγούμενη. Η καθυστέρηση αυτή εξαρτάται από την προηγούμενη εντολή που στάλθηκε. Επίσης, το μέγεθος του χρόνου δειγματοληψίας πρέπει να επιλεχθεί με προσοχή, καθώς αν είναι πολύ μεγάλο σε σχέση με τη συχνότητα της ταλάντωσης θα υπάρξει υπερχείλιση (overflow). Ο μετρητής μπορεί να μετρήσει μέχρι 65536 τιμές. Υπάρχει ένα σήμα στο κύκλωμα που ενημερώνει για τυχόν υπερχείλιση. Όταν υπάρχει υπερχείλιση θα πρέπει να μειωθεί ο χρόνος δειγματοληψίας για να ληφθεί σωστή μέτρηση.

Στο σχήμα 3.15 φαίνεται το συνολικό διάγραμμα της διεπαφής των αισθητήρων που παρουσιάζεται στο [31]. Το κύκλωμα περιλαμβάνει δυνατότητα παράλληλης και σειριακής επικοινωνίας. Στο σχήμα φαίνονται οι καταχωρητές του κυκλώματος. Ο καταχωρητής “oscillator control” για την επιλογή του ταλαντωτή, ο καταχωρητής “sampling time control” για την επιλογή του χρόνου δειγματοληψίας και ο καταχωρητής “signal control” για τον έλεγχο του μετρητή. Οι αντιστάσεις και οι χωρητικότητες του τροποποιημένου ταλαντωτή δακτυλίου συνδέονται εξωτερικά του FPGA. Για αυτό το σκοπό μπορεί να χρησιμοποιηθεί μια πλακέτα CPLD ή στη περίπτωση της παρούσας διπλωματικής χρησιμοποιώντας μια υποδοχή HSMC και ένα εξωτερικό κύκλωμα.



Σχήμα 3.15: Συνολικό διάγραμμα βαθμίδων διεπαφής αισθητήρων [31]

3.8 Συμπεράσματα

Το κύκλωμα που μόλις περιγράφηκε υλοποιήθηκε, στα πλαίσια παλαιότερης πτυχιακής εργασίας, στο αναπτυξιακό σύστημα GFEC MAX V Starter Kit το οποίο περιλαμβάνει ένα CPLD MAXV 5M1270ZT144C5N της Altera και για την ανάγνωση και διαχείριση των αποτελεσμάτων χρησιμοποιήθηκε προσωπικός υπολογιστής με σύνδεση μέσω της σειριακής θύρας UART RS232. Στην υλοποίηση αυτή δεν περιλαμβάνονταν κάποιος επεξεργαστής στο σύστημα και δεν υπήρχε δυνατότητα για επεξεργασία των μετρήσεων κατά τη λήψη τους. Στην παρούσα πτυχιακή ενσωματώθηκε αυτό το κύκλωμα σε ένα συνολικό επεξεργαστικό σύστημα, το οποίο είναι αυτόνομο και δεν χρειάζεται σύνδεση με κάποιον υπολογιστή για την αποτύπωση των αποτελεσμάτων. Στο σύστημα περιέχεται ο LEON3 επεξεργαστής και μία κατάλληλη έκδοση του λειτουργικού συστήματος Linux. Τα αποτελέσματα μπορούν να εμφανιστούν σε μια SVGA οθόνη και ο χρήστης να περιεργαστεί τα δεδομένα με πληκτρολόγιο. Για τον σκοπό αυτό το παραπάνω κύκλωμα διεπαφής των αισθητήρων έπρεπε να προσαρμοστεί στα νέα δεδομένα. Στο επόμενο κεφάλαιο θα γίνει αναλυτική περιγραφή των αλλαγών και βελτιώσεων που έγιναν στο κύκλωμα και όλων των συστατικών που χρησιμοποιήθηκαν.

4. ΥΛΙΚΟ ΜΕΡΟΣ ΥΛΟΠΟΙΗΣΗΣ

4.1 Εισαγωγικά

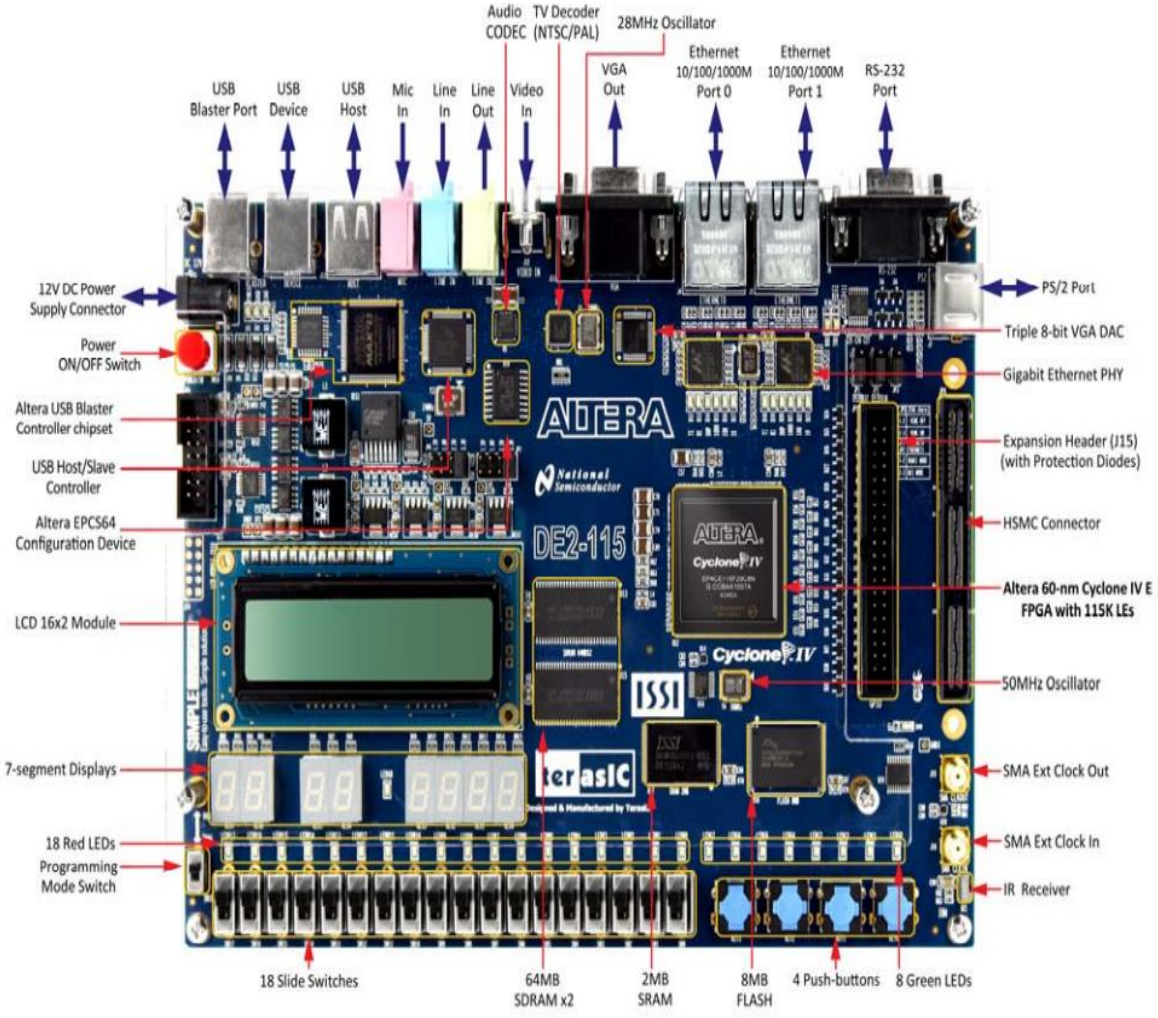
Σκοπός της παρούσας διπλωματικής ήταν να υλοποιηθεί ένα σύστημα αυτόνομο και λειτουργικό. Για το λόγο αυτό έπρεπε να χρησιμοποιηθούν αρκετά δομικά στοιχεία και περιφερειακά. Τα δομικά αυτά στοιχεία επιτρέπουν στο χρήστη να έχει διαδραστικό ρόλο με το σύστημα. Ο χρήστης μπορεί να έχει πρόσβαση στο λειτουργικό σύστημα και να το επηρεάζει, μπορεί να βλέπει τα αποτελέσματα των μετρήσεων και να τα επεξεργάζεται, να έχει απομακρυσμένη σύνδεση με το σύστημα και πολλές άλλες δυνατότητες. Για να επιτευχθούν όλες αυτές οι λειτουργίες πρέπει να υπάρχει μια συνεργασία μεταξύ των διαφόρων δομικών στοιχείων του συστήματος. Η χρησιμοποίηση του LEON3 επεξεργαστή και της βιβλιοθήκης GRLIB που παρουσιάστηκε στο κεφάλαιο 2 επιτρέπουν αυτήν τη συνεργασία με αποδοτικό τρόπο. Επίσης, τα υπόλοιπα εργαλεία της Aeroflex Gaisler όπως το Linuxbuild και το GRMON ήταν πολύ σημαντικά για τη δημιουργία του λειτουργικού συστήματος και την αποσφαλμάτωση αντίστοιχα. Σε αυτό το κεφάλαιο θα παρουσιαστεί το υλικό μέρος της υλοποίησης. Αρχικά, παρουσιάζεται η συσκευή στην οποία έγινε η υλοποίηση και τα χαρακτηριστικά της. Στη συνέχεια παρουσιάζονται όλα τα δομικά στοιχεία του συστήματος που υλοποιήθηκε, τα οποία αφορούν το υλικό μέρος, δηλαδή τα περιφερειακά που είναι συνδεδεμένα με τον επεξεργαστή LEON3.Έπειτα παρουσιάζεται η μετατροπή του κυκλώματος της διεπαφής αισθητήρων του κεφαλαίου 3 ώστε να μπορεί να συνδεθεί με το υπόλοιπο επεξεργαστικό σύστημα. Τέλος, παρουσιάζεται το εξωτερικό κύκλωμα αισθητήρων και ο τρόπος σύνδεσής του με το υπόλοιπο σύστημα.

4.2 Συστατικά μέρη του συστήματος

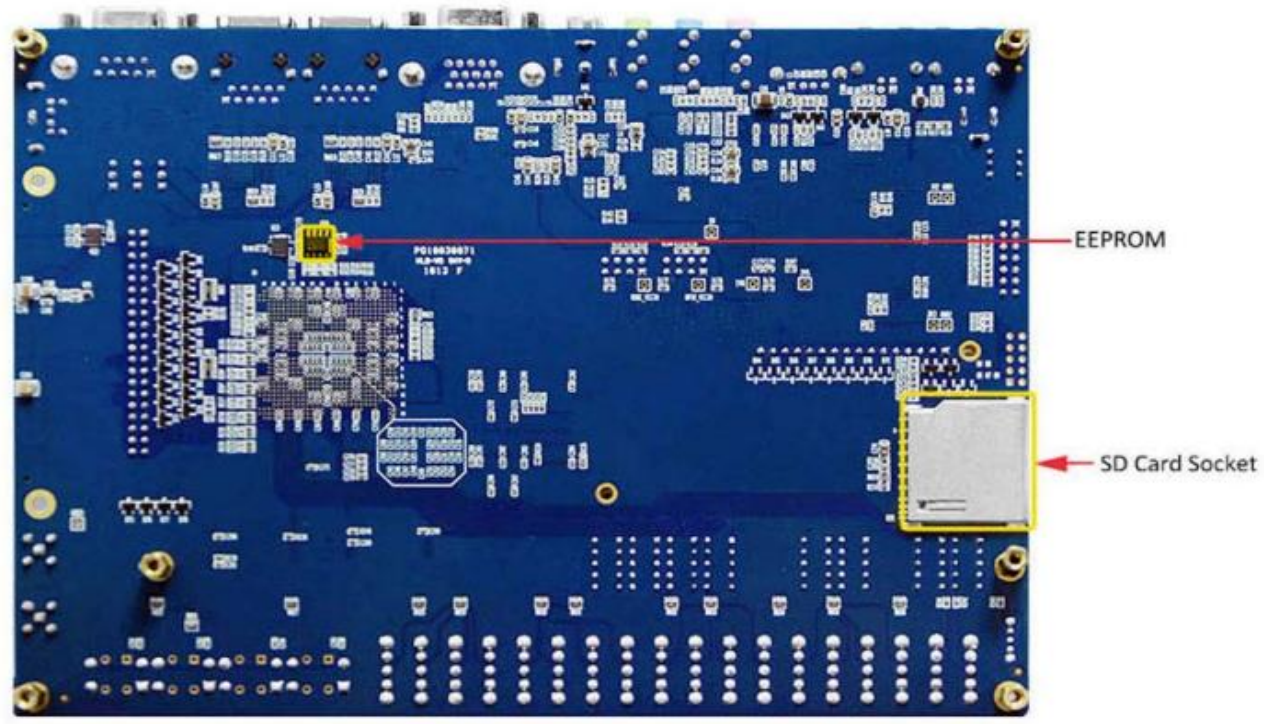
Το σύστημα που υλοποιήθηκε βασίζεται στον LEON3 επεξεργαστή και τα περιφερειακά που διατίθενται στη βιβλιοθήκη GRLIB. Η υλοποίηση έχει γίνει σε μια συσκευή DE2-115 της Terasic, η οποία περιλαμβάνει ένα FPGA Cyclone IV EP4CE115 της Altera. Η συγκεκριμένη συσκευή παρέχει όλες τις δυνατότητες για τη δημιουργία ενός αυτόνομου αποδοτικού συστήματος και επίσης είναι συμβατή με τον LEON3 επεξεργαστή. Παρακάτω παρουσιάζονται τα βασικά χαρακτηριστικά της [40].

- Συσκευή FPGA Altera Cyclone IV 4CE115
- Μνήμη προγραμματισμού (Serial Configuration device) της Altera– EPCS64
- USB Blaster πάνω στη συσκευή για τον προγραμματισμό της. Υποστηρίζεται προγραμματισμός και με JTAG και με λειτουργία Active Serial (AS).
- 2MB SRAM
- Δύο 64MB SDRAM
- 8MB μνήμη FLASH
- Υποδοχή κάρτας SD
- 4 πιεζόμενα κουμπιά (Push buttons)
- 18 διακόπτες (Slide switches)
- 18 Κόκκινα LEDs
- 9 Πράσινα LEDs
- Ταλαντωτής 50MHz για πηγές ρολογιού
- 24 μπιτ CODEC για ήχο ποιότητας CD με γραμμή εισόδου και εξόδου, και υποδοχή για μικρόφωνο

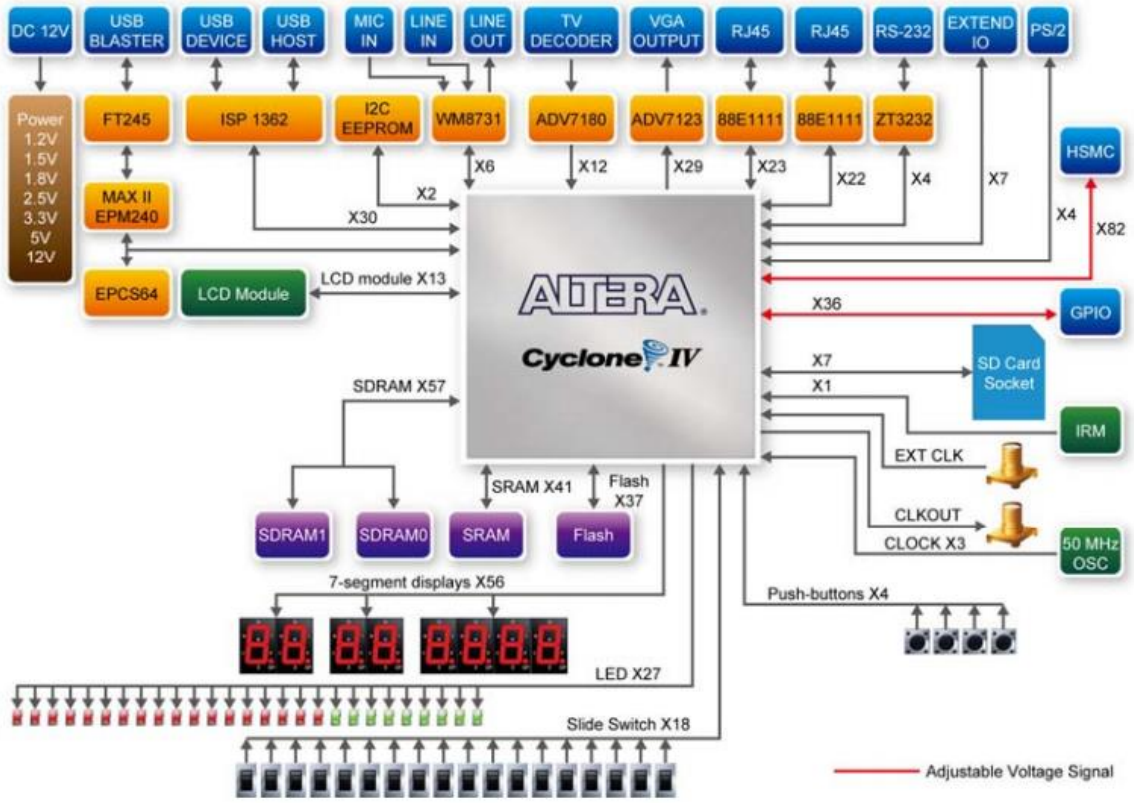
- VGA DAC με βύσμα (connector) VGA-out
- Αποκωδικοποιητής TV (NTSC/PAL/SECAM) και βύσμα TV-in
- 2 Gigabit Ethernet PHY με RJ45 βύσματα
- Ελεγκτής USB Host/Slave με βύσματα USB τύπου A και τύπου B
- Σειριακή θύρα RS-232 και βύσμα 9-ακροδεκτών
- Βύσμα για ποντίκι/πληκτρολόγιο PS/2
- Δέκτης IR
- 2 Βύσματα SMA για εξωτερικό ρολόι εισόδου/εξόδου
- Ένα βύσμα επέκτασης (Expansion Header) 40-ακροδεκτών με προστασία διόδου
- Ένα βύσμα HSMC
- Μια μονάδα 16x2 LCD



Εικόνα 4.1: Συσκευή DE2-115 από την Terasic [40]



Εικόνα 4.2: Πίσω μέρος της συσκευής DE2-115 [40]



Εικόνα 4.3: Διάγραμμα βαθμίδων της συσκευής DE2-115 [40]

Όπως φαίνεται η συσκευή DE2-115 υποστηρίζει πολλά περιφερειακά τα οποία μπορούν να χρησιμοποιηθούν για να υλοποιηθεί ένα πλήρες σύστημα. Στη συνέχεια παρουσιάζονται τα δομικά στοιχεία που χρησιμοποιήθηκαν από το σύστημα της παρούσας διπλωματικής. Τα δομικά στοιχεία αυτά είναι συνδεδεμένα με τον LEON3

επεξεργαστή. Για τη σύνδεση με τον επεξεργαστή χρησιμοποιείται και ο δίαυλος επικοινωνίας AHB για υψηλής απόδοσης περιφερειακά και ο δίαυλος APB για περιφερειακά χαμηλής απόδοσης.

Βασικά δομικά στοιχεία συστήματος:

- Επεξεργαστής LEON3 SPARC V8
- ETHERNET
- PS2 Πληκτρολόγιο
- SVGA οθόνη με λειτουργία “touch screen”
- Γενικού σκοπού θύρες εισόδου/εξόδου (GPIO)
- I^2C δίαυλος επικοινωνίας
- Διεπαφή αισθητήρων
- Ελεγκτής μνήμης
- Ελεγκτής διακοπών
- HSMC

Στη εικόνα 4.4 φαίνονται τα δομικά στοιχεία που αναγνωρίζει ο αποσφαλματωτής του συστήματος, GRMON.

```
 This eval version will expire on 07/10/2016

JTAG chain (1): EP3C120/EP4CE115
 GRLIB build version: 4156
 Detected frequency:  50 MHz

 Component                            Vendor
 LEON3 SPARC V8 Processor             Cobham Gaisler
 AHB Debug UART                       Cobham Gaisler
 JTAG Debug Link                      Cobham Gaisler
 GR Ethernet MAC                      Cobham Gaisler
 SVGA frame buffer                    Cobham Gaisler
 LEON2 Memory Controller              European Space Agency
 AHB/APB Bridge                       Cobham Gaisler
 LEON3 Debug Support Unit             Cobham Gaisler
 Generic UART                         Cobham Gaisler
 Multi-processor Interrupt Ctrl.      Cobham Gaisler
 Modular Timer Unit                   Cobham Gaisler
 PS2 interface                        Cobham Gaisler
 PS2 interface                        Cobham Gaisler
 AMBA Wrapper for OC I2C-master       Cobham Gaisler
 General Purpose I/O port             Cobham Gaisler
 Unknown device                       OpenCores

 Use command 'info sys' to print a detailed report of attached cores

grmon2>
```

Εικόνα 4.4 Αναγνώριση περιφερειακών και επεξεργαστή LEON3 από το GRMON

Το GRMON αναγνωρίζει και τη γέφυρα AHB/APB που είναι απαραίτητη για τη σύνδεση των χαμηλής αποδόσεως περιφερειακών, όπως το πληκτρολόγιο, τις θύρες GPIO και την διεπαφή των αισθητήρων (‘unknown device’), με τον δίαυλο AHB και τον επεξεργαστή. Στην εικόνα 4.5 φαίνεται σε τι είδους δίαυλο είναι συνδεδεμένο το κάθε περιφερειακό και άλλες χρήσιμες πληροφορίες. Επίσης υπάρχει η διεπαφή για την αποσφαλμάτωση μέσω JTAG (JTAG Debug Link), η γενική διεπαφή για την αποσφαλμάτωση (LEON3 Debug Support Unit), η διεπαφή για την UART σύνδεση με

το σύστημα (Generic Uart) και η μονάδα για τη λειτουργία και την αποσφαλμάτωση μέσω ETHERNET. Ακόμα υπάρχει μια μονάδα χρονισμού που είναι απαραίτητη για τον σωστό χρονισμό των σημάτων σε λειτουργικά συστήματα (Linux), μια μονάδα ελεγκτή μνήμης, ένας δίαυλος I^2C και ένας ελεγκτής διακοπών. Τέλος, στο σύστημα είναι συνδεδεμένη και η ενδιάμεση μνήμη εικόνας (SVGA frame buffer) για τη χρήση της SVGA οθόνης.

```
grmon2> info sys
  cpu0      Cobham Gaisler  LEON3 SPARC V8 Processor
            AHB Master 0
  ahbuart0  Cobham Gaisler  AHB Debug UART
            AHB Master 1
            APB: 80000700 - 80000800
            Baudrate 115200, AHB frequency 50.00 MHz
  ahbjtag0  Cobham Gaisler  JTAG Debug Link
            AHB Master 2
  greth0    Cobham Gaisler  GR Ethernet MAC
            AHB Master 3
            APB: 80000E00 - 80000F00
            IRQ: 12
            edcl ip 192.168.0.51, buffer 2 kbyte
  svga0     Cobham Gaisler  SVGA frame buffer
            AHB Master 6
            APB: 80000600 - 80000700
            clk0: 33.00 MHz  clk1: 65.00 MHz  clk2: 50.00 MHz  clk3: 25.00 MHz
  mctrl0    European Space Agency  LEON2 Memory Controller
            AHB: 00000000 - 20000000
            AHB: 40000000 - 80000000
            APB: 80000000 - 80000100
            8-bit prom @ 0x00000000
            32-bit sdram: 1 * 128 Mbyte @ 0x40000000
            col 10, cas 2, ref 7.8 us
  apbmst0   Cobham Gaisler  AHB/APB Bridge
            AHB: 80000000 - 80100000
  dsu0      Cobham Gaisler  LEON3 Debug Support Unit
            AHB: 90000000 - A0000000
            AHB trace: 128 lines, 32-bit bus
            CPU0:  win 8, hwbp 2, itrace 128, V8 mul/div, srmmu, lddel 1
                   stack pointer 0x47fffff0
                   icache 4 * 4 kB, 32 B/line, lru
                   dcache 4 * 4 kB, 16 B/line, lru, snoop tags
  uart0     Cobham Gaisler  Generic UART
            APB: 80000100 - 80000200
            IRQ: 2
            Baudrate 38343, FIFO debug mode
  irqmp0    Cobham Gaisler  Multi-processor Interrupt Ctrl.
            APB: 80000200 - 80000300
  gptimer0  Cobham Gaisler  Modular Timer Unit
            APB: 80000300 - 80000400
            IRQ: 8
            16-bit scalar, 2 * 32-bit timers, divisor 50
  ps2ifc0   Cobham Gaisler  PS2 interface
            APB: 80000400 - 80000500
            IRQ: 4
  ps2ifc1   Cobham Gaisler  PS2 interface
            APB: 80000500 - 80000600
            IRQ: 5
  i2cmst0   Cobham Gaisler  AMBA Wrapper for OC I2C-master
            APB: 80000800 - 80000900
            IRQ: 3
  gpio0     Cobham Gaisler  General Purpose I/O port
            APB: 80000900 - 80000A00

  adev15    OpenCores  Unknown device
            APB: 80000C00 - 80000D00

grmon2>
```

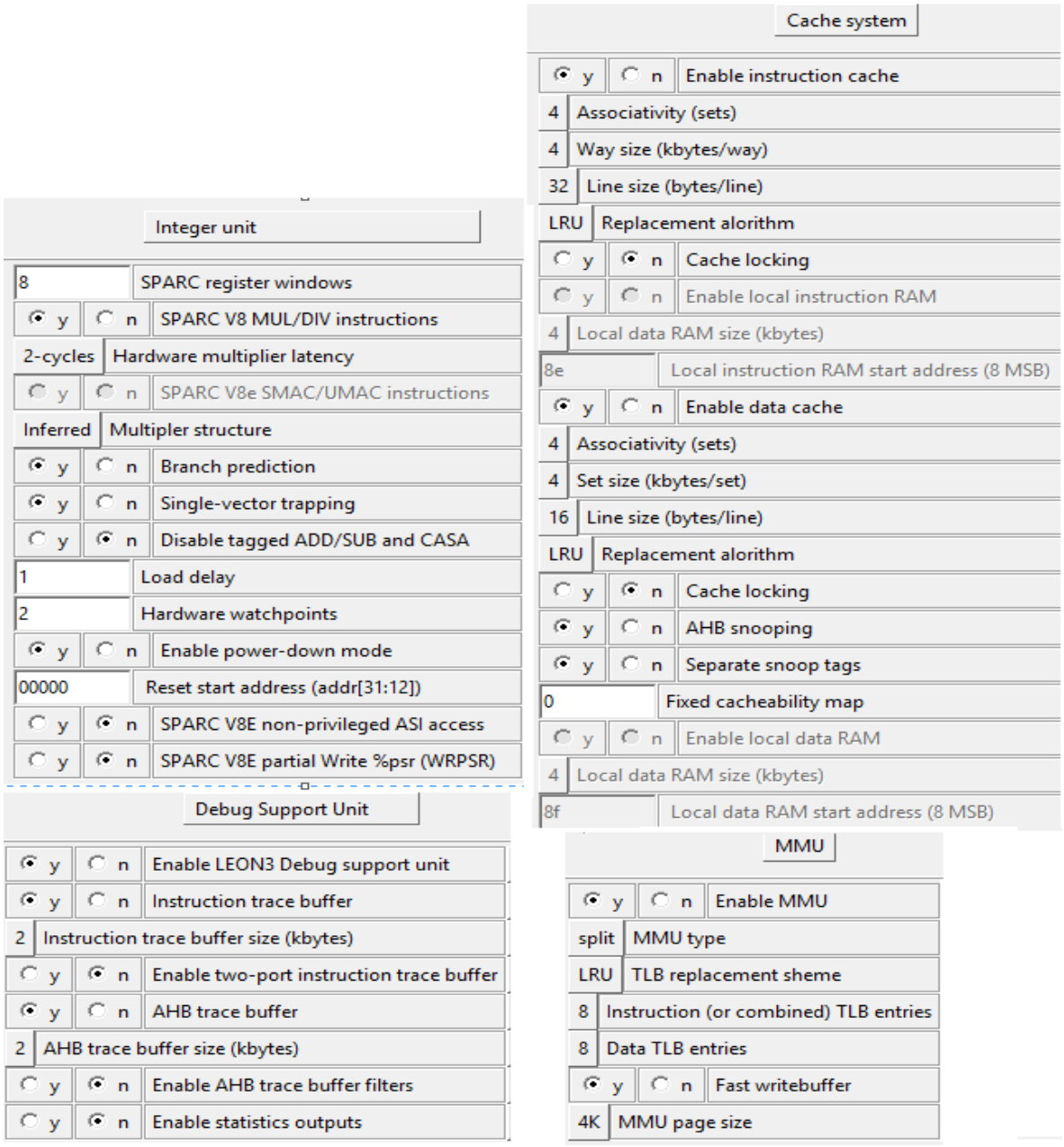
Εικόνα 4.5: Πιο λεπτομερής ανασκόπηση των συσκευών του συστήματος χρησιμοποιώντας την εντολή 'info sys' του GRMON

Στη συνέχεια αναλύεται κάθε δομικό στοιχείο και περιφερειακό ξεχωριστά για την καλύτερη κατανόηση του συστήματος.

4.2.1 Επεξεργαστής LEON3 SPARC V8

Όπως ειπώθηκε στο κεφάλαιο 1 και 2 ο LEON3 είναι ένας επεξεργαστής ο οποίος υλοποιείται μέσω λογισμικού. Είναι παραμετροποιήσιμος και διατίθεται μια πληθώρα εναλλακτικών επιλογών για την υλοποίησή του. Με τη χρησιμοποίηση της βιβλιοθήκης

GRLIB μπορούν να επιλεχθούν πολύ εύκολα οι παράμετροι του επεξεργαστή. Στη εικόνα 4.6 φαίνονται οι επιλογές του επεξεργαστή LEON3 για το σύστημα της διπλωματικής, χρησιμοποιώντας το εργαλείο του Linux λειτουργικού συστήματος, “xconfig”.

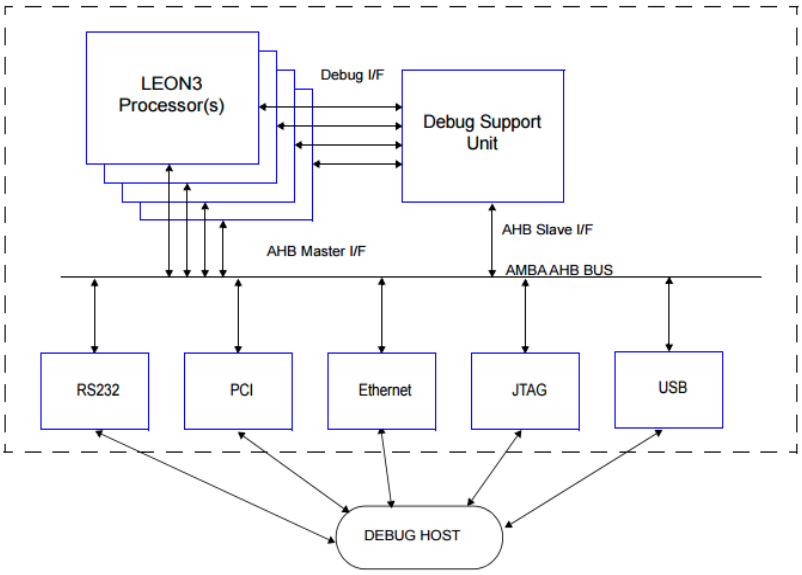


Εικόνα 4.6: Παραμετροποίηση του επεξεργαστή χρησιμοποιώντας το ‘xconfig’

Στο σύστημα επιλέχθηκε να υπάρχει ένας επεξεργαστής ο οποίος να περιλαμβάνει όλες τις βασικές λειτουργίες και να μπορεί να υποστηρίξει το λειτουργικό σύστημα Linux. Δεν έχει επιλεχτεί η χρήση μονάδας κινητής υποδιαστολής. Για τη κρυφή μνήμη έχουν επιλεγεί οι προκαθορισμένες τιμές με LRU, την τεχνική ‘snooping’ και ξεχωριστή κρυφή μνήμη για δεδομένα και εντολές. Για τη χρήση Linux λειτουργικού είναι απαραίτητη μια μονάδα διαχείρισης μνήμης (MMU). Επίσης έχει επιλεχτεί η μονάδα αποσφαλμάτωσης (Debug Support Unit) και η αρχική διεύθυνση του επεξεργαστή έχει καθοριστεί στη θέση 00000, καθώς σε αυτό το σημείο υπάρχει η μνήμη PROM που είναι αποθηκευμένη η εικόνα Linux. Έτσι ο επεξεργαστής κατά την εκκίνηση θα φορτώνει το λειτουργικό σύστημα. Οι υπόλοιπες ρυθμίσεις φαίνονται στην εικόνα 4.6 και είναι οι προκαθορισμένες (default).

4.2.2 Μονάδα αποσφαλμάτωσης LEON3 (LEON3 Debug Support Unit)

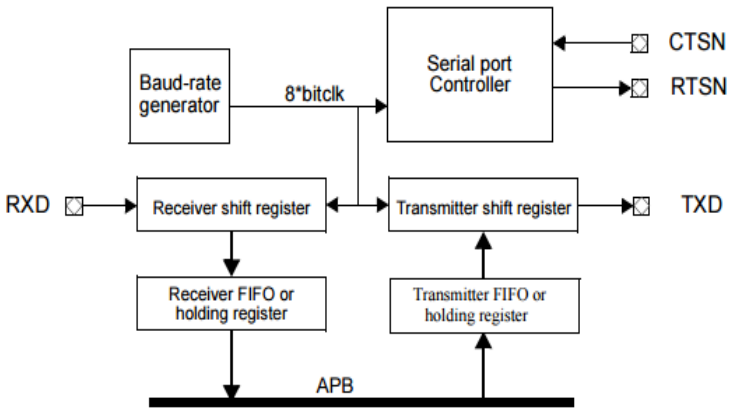
Για την απλοποίηση της αποσφαλμάτωσης πάνω στο υλικό, ο LEON3 επεξεργαστής μπαίνει σε μια λειτουργία αποσφαλμάτωσης κατά την οποία η διασωλήνωση είναι ανενεργή (idle) και ο επεξεργαστής ελέγχεται από μια ειδική διεπαφή αποσφαλμάτωσης. Η μονάδα αποσφαλμάτωσης του LEON3 (DSU) χρησιμοποιείται για τον έλεγχο του επεξεργαστή κατά τη λειτουργία της αποσφαλμάτωσης. Το DSU δρα ως ένας AHB υποτελής (slave) και μπορεί να έχει πρόσβαση σε αυτό κάθε κύριος(master)-AHB. 'Ενας εξωτερικός εξυπηρετητής (host) αποσφαλμάτωσης μπορεί να έχει πρόσβαση στο DSU μέσω αρκετών διεπαφών. Τέτοια διεπαφή μπορεί να είναι η σειριακή UART (RS232), η JTAG, η PCI, η USB ή η ETHERNET. Η μονάδα DSU υποστηρίζει πολύ-επεξεργαστικά συστήματα και μπορεί να διαχειριστεί ως και 16 επεξεργαστές [26].



Σχήμα 4.1: Σύνδεση LEON3/DSU[26]

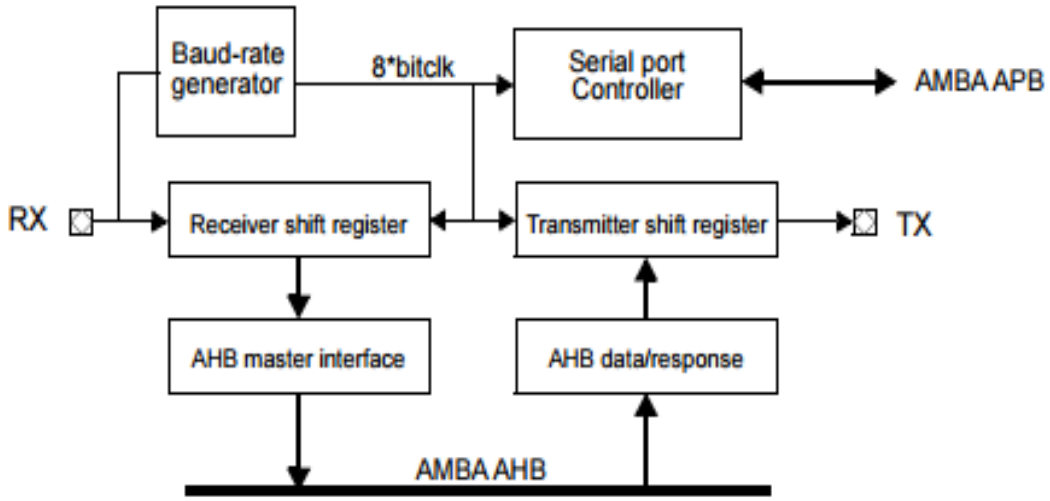
4.2.3 Σειριακή διεπαφή AMBA APB–UART (Generic Uart) και αποσφαλμάτωση μέσω UART (AHB DEBUG UART)

Η διεπαφή αυτή παρέχεται για σειριακή επικοινωνία. Το UART υποστηρίζει παράθυρο δεδομένων (data frame) των 8 μπιτ, ένα προαιρετικό μπιτ ισοτιμίας (parity), και ένα ή δύο μπιτ σταματήματος (stop bit). Για τη δημιουργία του ρυθμού μετάδοσης των μπιτ (bit-rate), κάθε UART έχει ένα προγραμματιζόμενο 12-μπιτ διαιρέτη ρολογιού [26]



Σχήμα 4.2: Διάγραμμα βαθμίδων διεπαφής UART [26]

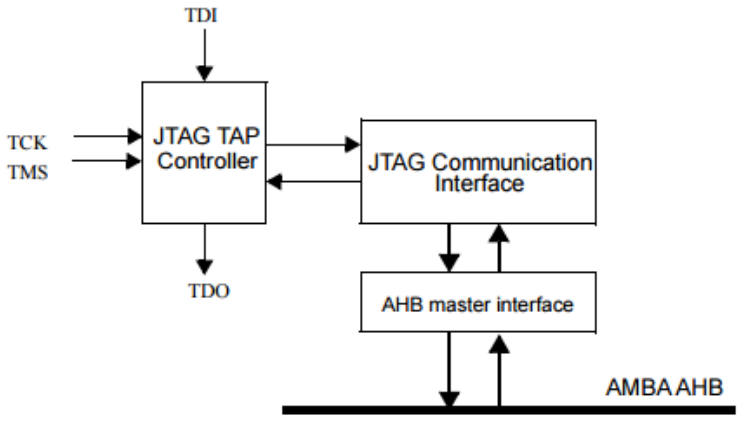
Σε κάποια στάδια κατά την ανάπτυξη του συστήματος έγινε αποσφαλμάτωση με το UART κύκλωμα, αν και κυρίως χρησιμοποιήθηκε το δίκτυο (ETHERNET) και το JTAG. Γενικά είναι χρήσιμο να υπάρχει μια θύρα UART σε ένα αυτόνομο σύστημα. Η διεπαφή για την αποσφαλμάτωση μέσω UART, αποτελείται από μια UART συνδεδεμένη στο δίαυλο επικοινωνίας AHB ως κύριος-AHB. Για την μεταφορά παραμέτρων πρόσβασης και δεδομένων υποστηρίζεται ένα απλό πρωτόκολλο επικοινωνίας. Μέσω του συνδέσμου επικοινωνίας (communication link), μπορεί να δημιουργηθεί μια εντολή γραψίματος ή διαβάσματος σε οποιαδήποτε διεύθυνση στον δίαυλο AMBA AHB [26].



Σχήμα 4.3: Διάγραμμα βαθμίδων αποσφαλμάτωσης μέσω UART [26]

4.2.4 Αποσφαλμάτωση μέσω JTAG (JTAG DEBUG LINK)

Η διεπαφή αποσφαλμάτωσης JTAG προσφέρει πρόσβαση στο δίαυλο επικοινωνίας AMBA AHB πάνω στο τσιπ, μέσω του JTAG. Η διεπαφή υλοποιεί ένα απλό πρωτόκολλο, το οποίο μεταφράζει τις εντολές του JTAG σε μεταγωγές (transfers) του διαύλου AHB. Μέσω του συνδέσμου επικοινωνίας μπορεί να δημιουργηθεί μια εντολή γραψίματος ή διαβάσματος σε οποιαδήποτε διεύθυνση στον δίαυλο AMBA AHB [26].



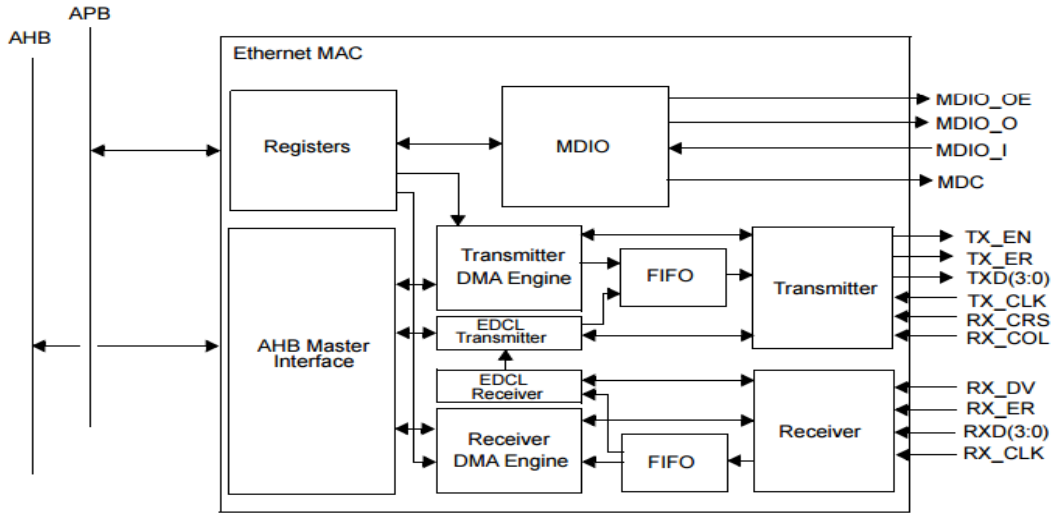
Σχήμα 4.4: Διάγραμμα βαθμίδων αποσφαλμάτωσης με JTAG [26]

4.2.5 Διεπαφή Δικτύου ETHERNET (GR ETHERNET MAC ή GRETH)

Ο ελεγκτής ETHERNET του Cobham Gaisler (GRETH) , παρέχει την δυνατότητα διασύνδεσης μεταξύ του διαύλου AMBA-AHB και ενός ETHERNET δικτύου. Υποστηρίζει ταχύτητες 10/100Mbit. Η διεπαφή του AMBA αποτελείται από μια διεπαφή διαύλου APB για τις ρυθμίσεις και τον έλεγχο, και μια διεπαφή κύριου-AHB διαύλου η οποία ελέγχει τη ροή των δεδομένων. Η ροή των δεδομένων ελέγχεται από κανάλια

άμεσης πρόσβασης μνήμης (DMA). Υπάρχει μια μηχανή DMA(DMA engine) για τον πομπό και άλλη μία για τον δέκτη. Και οι 2 μοιράζονται την ίδια διεπαφή κύριου-AHB. Η διεπαφή δικτύου ETHERNET υποστηρίζει και την MII και την RMII διεπαφή [26].

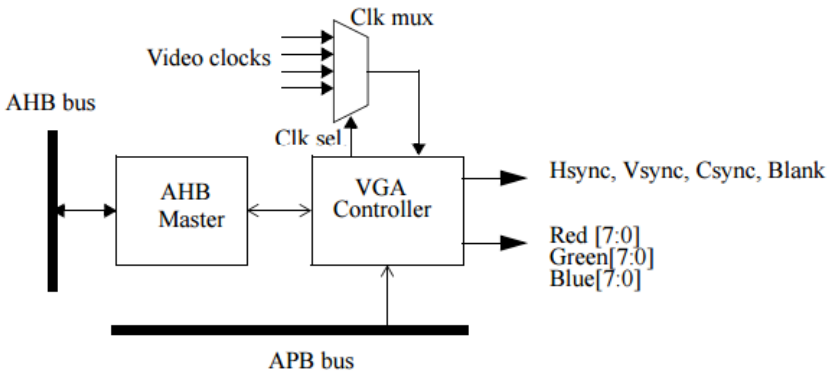
Η διεπαφή δικτύου υποστηρίζει την επιλογή αποσφαλμάτωσης μέσω ETHERNET χρησιμοποιώντας το πρωτόκολλο EDCL. Το πρωτόκολλο αυτό είναι βασισμένο στα UDP/IP και χρησιμοποιείται για απομακρυσμένη αποσφαλμάτωση.



Σχήμα 4.5 Διάγραμμα βαθμίδων της εσωτερικής δομής του κυκλώματος GRETH [26]

4.2.6 Η ενδιάμεση μνήμη εικόνας SVGA (SVGA frame buffer)

Αυτός ο πυρήνας είναι ένας βασισμένος σε εικονοστοιχεία (pixel) ελεγκτής βίντεο, με δυνατότητες εμφάνισης τυπικών και κατά προτίμηση αναλύσεων οι οποίες έχουν μεταβλητό βάθος σε μπιτ (bit depth) και ρυθμό ανανέωσης (refresh rate). Ο ελεγκτής βίντεο αποτελείται από μια μονάδα συγχρονισμού, μια κεντρική μονάδα ελέγχου, μια μονάδα FIFO και μια διεπαφή κύριου-AHB όπως φαίνεται στη παρακάτω εικόνα [26].



Σχήμα 4.6 Διάγραμμα βαθμίδων του ελεγκτή VGA [26]

Ο πυρήνας χρησιμοποιεί μια εξωτερική ενδιάμεση μνήμη για τα καρέ η οποία βρίσκεται στο χώρο διευθύνσεων του AHB διαύλου. Κάθε καρέ δημιουργείται παίρνοντας τα εικονοστοιχεία από τη μνήμη και στέλνοντάς τα στην οθόνη μέσω ενός εξωτερικού DAC, χρησιμοποιώντας τρία 8-μπιτ διανύσματα. Για τη μείωση της καθυστέρησης λόγω του διαύλου AHB, τα εικονοστοιχεία αποθηκεύονται σε μια μνήμη FIFO μέσα στον πυρήνα. Η αρχική διεύθυνση της ενδιάμεσης μνήμης καρέ ορίζεται σε ένα καταχωρητή και μπορεί να είναι οποιαδήποτε στο χώρο διευθύνσεων του AHB διαύλου. Εκτός από τα

διανύσματα για τα χρώματα, ο ελεγκτής βίντεο διαθέτει και τα σήματα ελέγχου HSYNC, VSYNC, CSYNC και BLANK [26].

Ο χρονισμός του βίντεο μπορεί να προγραμματιστεί διαμέσου των παραμέτρων 'Video Length', 'Front Porch', 'Sync Length' και 'Line Length', οι οποίες αποθηκεύονται σε καταχωρητές (πίνακας 2). Η επιλογή του βάθους των μπιτ και η ενεργοποίηση του ελεγκτή γίνεται μέσω του καταχωρητή καταστάσεως (status register). Αυτές οι παράμετροι επιτρέπουν την απεικόνιση μεγάλου εύρους αναλύσεων και ρυθμών ανανέωσης.

Η συχνότητα λειτουργίας του ρολογιού των εικονοστοιχείων υπολογίζεται από:

Μήκος οριζόντιας γραμμής* μήκος κάθετης γραμμής * ρυθμός ανανέωσης

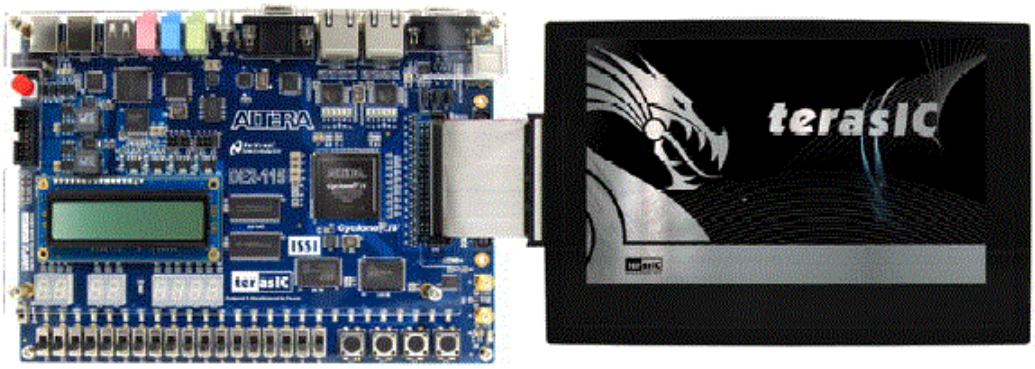
Ο πυρήνας μπορεί να χρησιμοποιήσει βάθος μπιτ 8, 16 και 32 μπιτ. Όταν χρησιμοποιείται 32 μπιτ, χρησιμοποιούνται τα μπιτ [23:0] με 8 μπιτ για κάθε χρώμα, όταν είναι 16, τα χρώματα είναι σε μορφή [5,6,5] μπιτ και όταν χρησιμοποιείται 8 βάθος μπιτ, χρησιμοποιείται ένα CLUT (color lookup table). Το CLUT έχει 256 θέσεις, η καθεμιά με 24 μπιτ. Οι τιμές των 8 μπιτ που διαβάζονται από τη μνήμη χρησιμοποιούνται ως δείκτης στο CLUT, ώστε να επιλεχτεί το κατάλληλο χρώμα [26].

Πίνακας 2: Καταχωρητές του ελεγκτή VGA [26]

APB address offset	Register
0x00	Status register
0x04	Video length register
0x08	Front Porch register
0x0C	Sync Length register
0x10	Line Length register
0x14	Framebuffer Memory Position register
0x18	Dynamic Clock 0 register
0x1C	Dynamic Clock 1 register
0x20	Dynamic Clock 2 register
0x24	Dynamic Clock 3 register
0x28	CLUT Access register

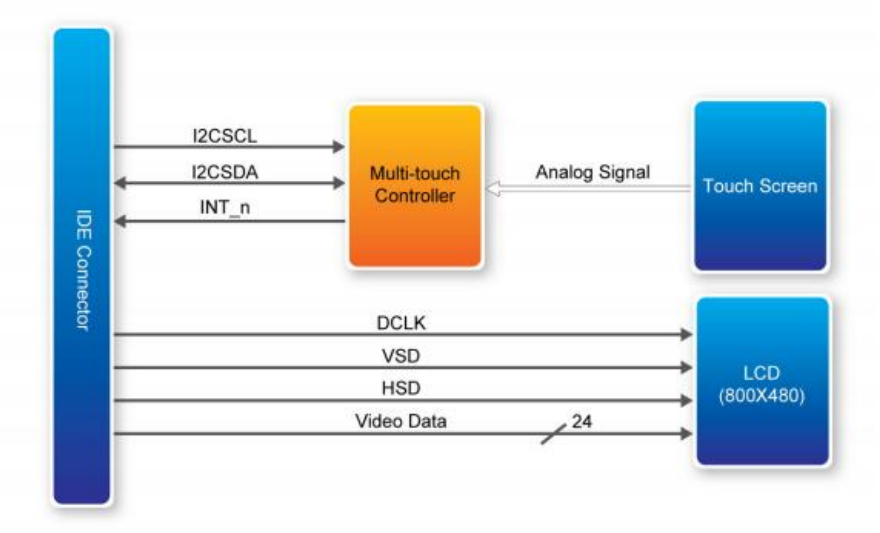
4.2.7 Οθόνη που χρησιμοποιήθηκε στο σύστημα

Για την υλοποίηση χρησιμοποιήθηκε η οθόνη της Terasic, Multi-touch LCD Module (MTL). Η συγκεκριμένη οθόνη είναι μια γενικού σκοπού οθόνη επαφής (touch screen) χωρητικοτήτων, κατάλληλη για εφαρμογές σε συσκευές FPGA. Προσφέρει τη δυνατότητα κινήσεων πολλαπλής επαφής (multi-touch gesture) καθώς και της απλής επαφής. Χρησιμοποιείται ένα καλώδιο IDE με ένα προσαρμογέα IDE σε GPIO για την διασύνδεση με διάφορα FPGA, και η σύνδεση γίνεται διαμέσου της διεπαφής των 2x20 GPIO θυρών πάνω στην οθόνη.

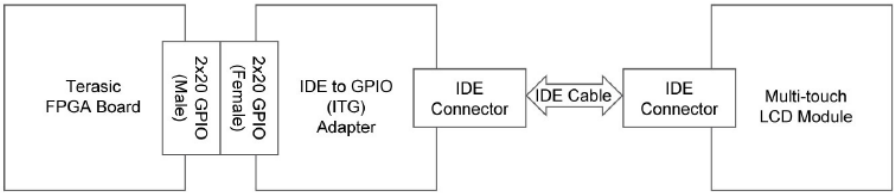


Εικόνα 4.7 Οθόνη MTL συνδεδεμένη με ένα αναπτυξιακό DE2-115 [41]

Στο σχήμα 4.7 φαίνεται το διάγραμμα βαθμίδων της MTL οθόνης. Το βύσμα IDE περιέχει τα καλώδια από τις περιφερειακές διεπαφές. Για την διασύνδεση με τη λειτουργία επαφής χρησιμοποιείται ένας δίαυλος I²C, με τον οποίο διαβάζονται οι συντεταγμένες του σημείου της οθόνης που πατήθηκε [41].

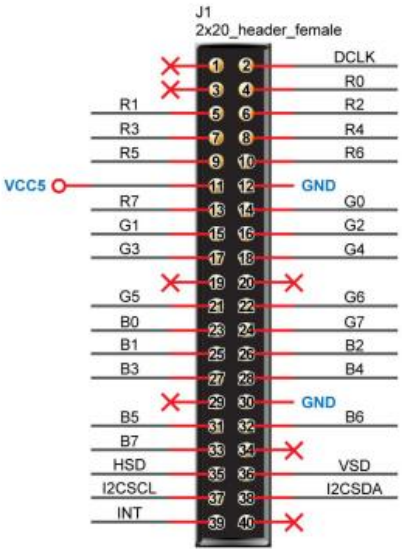


Σχήμα 4.7: Διάγραμμα βαθμίδων της οθόνης MTL [41]



Σχήμα 4.8: Διάγραμμα βαθμίδων σύνδεσης της οθόνης με μια συσκευή FPGA [41]

Στο σχήμα 4.8 φαίνεται ο τρόπος σύνδεσης του IDE βύσματος με τις θύρες GPIO της συσκευής FPGA. Υπάρχει ένας προσαρμογέας ITG για τη σύνδεση με τις GPIO θύρες του FPGA. Η θηλυκή (female) πλευρά του GPIO έχει τη διασύνδεση της εικόνας 4.8 [41]. Το R,B,G (π.χ. στο R1 σήμα) συμβολίζουν τα χρώματα κόκκινο, μπλε και πράσινο αντίστοιχα και ο αριθμός το αντίστοιχο μπιτ



Εικόνα 4.8: Θηλυκή πλευρά της διασύνδεσης με θύρες GPIO [41]

Η οθόνη LCD έχει ανάλυση 800x480 και τρέχει με ρυθμό εικονοστοιχείου 33MHz. Στον πίνακα 3 φαίνονται τα χαρακτηριστικά της LCD οθόνης που χρησιμοποιήθηκε.

Πίνακας 3: Χαρακτηριστικά LCD οθόνης [41]

Item	*Typical Value*	*Unit*
Pixel Rate	33	MHz
Horizontal Period	1056	Pixel
Horizontal Pulse Width	30	Pixel
Horizontal Back Porch	16	Pixel
Horizontal Front Porch	210	Pixel
Horizontal Valid	800	Pixel
Vertical Period	525	Line
Vertical Pulse Width	13	Line
Vertical Back Porch	10	Line
Vertical Front Porch	22	Line
Vertical Valid	480	Line

Για την σύνδεση της οθόνης με το υπόλοιπο σύστημα χρησιμοποιήθηκαν οι θύρες GPIO τις συσκευής DE2-115 και ένα PLL που δημιουργεί τη συχνότητα των 33MHz. Επίσης χρησιμοποιήθηκε ο ελεγκτής SVGA που περιεγράφηκε στο κεφάλαιο 4.2.6 , που είναι υπεύθυνος για την αποστολή των σημάτων των χρωμάτων και των σημάτων λειτουργίας (HSD, VSD, CLK). Η διακοπή για το πάτημα της οθόνης έχει συνδεθεί με ένα 'GPIO pad' το οποίο είναι εύκολα διαχειρίσιμο και μπορεί να χρησιμοποιηθεί ως διακοπή, όπως αναφέρεται και στο κεφάλαιο 4.2.10. Τέλος για την σύνδεση των I²C σημάτων (I2CSCL, I2CSDA) χρησιμοποιήθηκε ο πυρήνας I2CMST ο οποίος παρέχεται από την βιβλιοθήκη GRLIB και περιγράφεται στη συνέχεια. Στην εικόνα 4.9 φαίνεται σε γλώσσα VHDL η σύνδεση των περιφερειακών μεταξύ τους αλλά και με τον δίαυλο AMBA του LEON3 επεξεργαστή, για την σωστή λειτουργία της οθόνης.

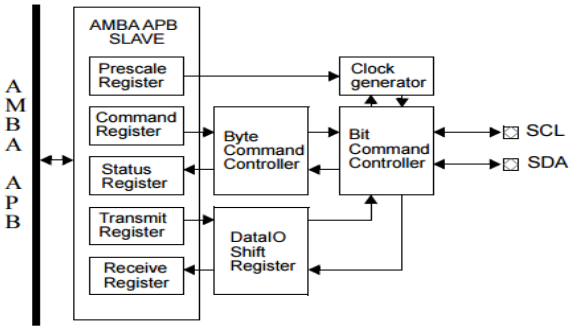
```
vgaclkgen : altera_eek_clkgen --PLL clock 33MHz
generic map (clk0_mul => 2, clk0_div => 3, clk1_mul  => 4, clk1_div  => 5,
clk_freq => BOARD_FREQ)
 port map (clock_50, clkmvga, clkmvga3x, clk_sel, clkmvga_lck);

vga0 : svgactrl --SVGA controller
generic map(memtech => memtech, pindex => 6, paddr => 6, hindex => 6,
clk0 => 30303, clk1 => 15385, clk2 => 20000, clk3 => 40000)
 port map (rstn, clkm, clkmvga, apbi, apbo(6), vgao, ahbmi, ahbmo(6),
clk_sel);

vgaclk_pad : outpad generic map (tech => padtech) --clk
       port map (gpio(1), clkmvga);
hsync_pad : outpad generic map (tech => padtech)
       port map (gpio(30), vgao.hsync);-- hsync
vsync_pad : outpad generic map (tech => padtech)
       port map (gpio(31), vgao.vsync);  --vsync
vga_red_pads : for i in 0 to 7  generate --τοποθέτηση χρωμάτων στις κατάλληλες θύρες
        red_pad : outpad generic map (tech => padtech) --κόκκινο
       port map (gpio(i+3), vgao.video_out_r(i));
     end generate;
vga_green_pads : for i in 0 to 4  generate --πράσινο
        green_pad : outpad generic map (tech => padtech)
       port map (gpio(i+11), vgao.video_out_g(i));
     end generate;
        green_pad5 : outpad generic map (tech => padtech)
       port map (gpio(18), vgao.video_out_g(5));
        green_pad6 : outpad generic map (tech => padtech)
       port map (gpio(19), vgao.video_out_g(6));
        green_pad7 : outpad generic map (tech => padtech)
       port map (gpio(21), vgao.video_out_g(7));
        blue_pad1 : outpad generic map (tech => padtech)
       port map (gpio(20), vgao.video_out_b(0));
vga_blue_pads1_4 : for i in 0 to 3  generate --μπλε
        blue_pad : outpad generic map (tech => padtech)
       port map (gpio(i+22), vgao.video_out_b(i+1));
     end generate;
vga_blue_pads5_7 : for i in 0 to 2  generate
        blue_pad : outpad generic map (tech => padtech)
        port map (gpio(i+26), vgao.video_out_b(i+5));
     end generate;
```

Εικόνα 4.9: Σύνδεση οθόνης με τον LEON3 σε γλώσσα VHDL

4.2.8 I²C δίαυλος επικοινωνίας

4.2.8.1 Ο πυρήνας στο υλικό (I2CMST - I²C - master)

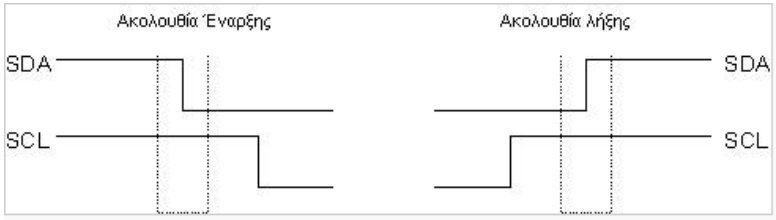
Ο πυρήνας I²C-master είναι μια τροποποιημένη έκδοση του OpenCores I²C -Master και περιλαμβάνει διεπαφή για σύνδεση με τον δίαυλο APB. Ο πυρήνας είναι συμβατός με το πρότυπο Philips I²C και υποστηρίζει 7 και 10 μπιτ διευθυνσιοδότησης. Υποστηρίζεται κανονική ταχύτητα λειτουργίας 100 kb/s και γρήγορη ταχύτητα 400 kb/s. Στο σχήμα 4.9 φαίνεται το διάγραμμα βαθμίδων του πυρήνα [26].



Σχήμα 4.9: Διάγραμμα βαθμίδων I2CMST [26]

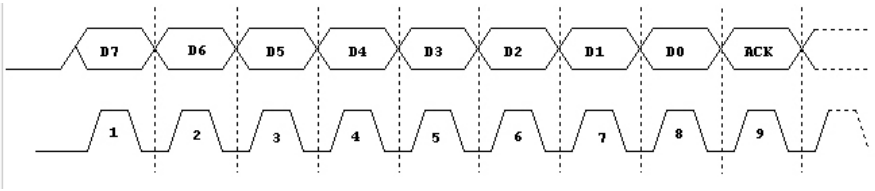
4.2.8.2 Πρωτόκολλο επικοινωνίας με I²C

Οι συσκευές στον δίαυλο I²C είναι είτε κύριες είτε υποτελείς. Η κύρια συσκευή είναι αυτή που ελέγχει και οδηγεί τη γραμμή ρολογιού SCL (η οποία παράγει τους παλμούς ρολογιού). Οι υποτελείς συσκευές είναι αυτές που ανταποκρίνονται στις κύριες συσκευές. Μία υποτελής συσκευή δεν μπορεί να ξεκινήσει μία μεταφορά πάνω στο δίαυλο, μόνο μία κύρια συσκευή μπορεί. Σε έναν δίαυλο μπορεί να είναι συνδεμένες πολλές κύριες και πολλές υποτελείς συσκευές. Και οι κύριες και οι υποτελείς συσκευές μπορούν να μεταφέρουν δεδομένα στον δίαυλο, αλλά μόνο οι κύριες συσκευές ελέγχουν την μεταφορά [42].



Σχήμα 4.10: Ακολουθίες Έναρξης Λήξης στο δίαυλο I²C [42]

Όταν μία κύρια συσκευή επιθυμεί να επικοινωνήσει με μία υποτελή συσκευή, ξεκινά στέλνοντας στον δίαυλο μία ακολουθία έναρξης (start sequence). Η ακολουθία έναρξης, είναι μία από τις δύο ειδικές ακολουθίες που ορίζονται στο δίαυλο I²C, ή άλλη είναι η ακολουθία λήξης (stop sequence). Οι ακολουθίες έναρξης και λήξης διαφέρουν στο ότι είναι οι μοναδικές θέσεις στις οποίες επιτρέπεται να αλλάζει η γραμμή δεδομένων (SDA), ενόσω η γραμμή ρολογιού είναι σε κατάσταση λογικού 1 (high). Όταν μεταφέρονται δεδομένα, η γραμμή SDA πρέπει να παραμένει σταθερή και να μην αλλάζει όσο η γραμμή ρολογιού είναι στο λογικό 1. Οι ακολουθίες έναρξης και λήξης σημαδεύουν την έναρξη και τη λήξη μιας μεταφοράς με μία υποτελή συσκευή. Ο δίαυλος θεωρείται ότι είναι αποσχολημένος, μετά από μία ακολουθία έναρξης και ελεύθερος λίγο χρόνο μετά την ακολουθία λήξης.

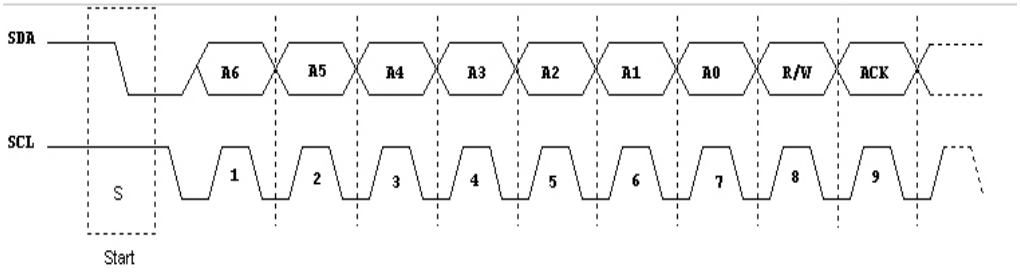


Σχήμα 4.11: Αποστολή byte στο δίαυλο I²C [42]

Σε όλες τις υποτελείς συσκευές που συνδέονται στον δίαυλο, έχει αποδοθεί ένας αριθμός σαν διεύθυνση. Οι κύριες συσκευές δεν είναι απαραίτητο να έχουν διεύθυνση, εκτός εάν υπάρχουν πολλές κύριες συσκευές στον δίαυλο (περιβάλλον 'Multi-master'). Οι κύριες συσκευές μπορούν να διαλέξουν αυθαίρετα μία από τις συνδεμένες υποτελείς συσκευές, για επικοινωνία, χρησιμοποιώντας τη διεύθυνσή της. Οι διευθύνσεις των συσκευών του I²C διαύλου είναι είτε 7 μπιτ (θεωρητικά έως 128 συσκευές στο δίαυλο), είτε 10 μπιτ (θεωρητικά έως 1024 συσκευές στο δίαυλο). Για την περίπτωση που η διευθυνσιοδότηση είναι 7-μπιτ μόνο 112 διευθύνσεις είναι διαθέσιμες στη πραγματικότητα. Ορισμένες διευθύνσεις χρησιμοποιούνται για ειδικούς σκοπούς π.χ η I²C διεύθυνση 0, είναι γενική κλήση προς όλες τις συσκευές. Στον παρακάτω πίνακα φαίνονται οι διευθύνσεις που είναι δεσμευμένες για ειδικές χρήσεις [42].

Πίνακας 4: Διευθύνσεις που είναι δεσμευμένες για ειδικές χρήσεις στον δίαυλο I2C [42]

Διεύθυνση Slave	R/W bit	Περιγραφή
0000 000	0	Δ/νση Γενική Κλήσης
0000 000	1	START Byte
0000 001	x	Δ/νση CBUS
0000 010	x	Δεσμευμένο για διαφορετικές μορφές διαύλου
0000 011	x	Δεσμευμένο για μελλοντικούς σκοπούς
0000 1xx	x	High Speed mode master code
1111 1xx	x	Δεσμευμένο για μελλοντικούς σκοπούς
1111 0xx	x	Διευθυνσιοδότηση slave 10 bit

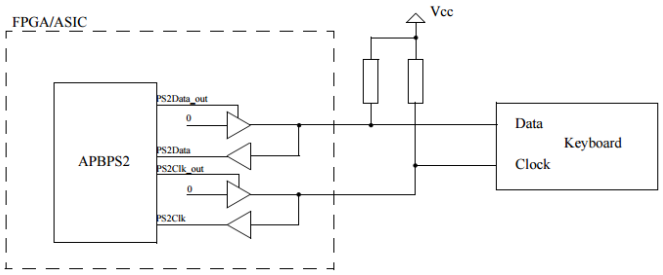


Σχήμα 4.12: Αποστολή Διεύθυνσης 7 μπιτ [42]

Στο σχήμα 4.12 στέλνεται από μια κύρια συσκευή η διεύθυνση 'A6A5A4A3A2A1A0' των 7–μπιτ της συσκευής με την οποία επιθυμείται επικοινωνία. Το επιπλέον (R/W) μπιτ χρησιμεύει να πληροφορήσει την υποτελή συσκευή, εάν η κύρια συσκευή πρόκειται να γράψει ή να διαβάσει από αυτήν. Εάν το μπιτ είναι 0 η κύρια συσκευή πρόκειται να γράψει. Εάν είναι 1 πρόκειται να διαβάσει από την υποτελή συσκευή. Τα 7 μπιτ της διεύθυνσης τοποθετούνται στα 7 πάνω μπιτ (MSB) του byte, ενώ το μπιτ ανάγνωσης/εγγραφής (R/W) στο λιγότερο σημαντικό μπιτ (LSB) [42].

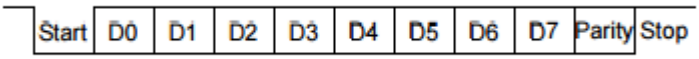
4.2.9 PS/2 Πληκτρολόγιο (APBPS2)

Η διεπαφή PS/2 είναι ένας διπλής κατεύθυνσης (bidirectional) σειριακός δίαυλος, ο οποίος χρησιμοποιείται κυρίως για την επικοινωνία πληκτρολογίου και ποντικιού (mouse). Ο πυρήνας APBPS2 υλοποιεί το πρωτόκολλο PS2 χρησιμοποιώντας τον δίαυλο επικοινωνίας APB. Στο σχήμα 4.13 φαίνεται μια μονάδα APBPS2 και η ηλεκτρική διεπαφή της (electrical interface) [26].



Σχήμα 4.13: Ηλεκτρική διεπαφή του πυρήνα APBPS2 [26]

Τα δεδομένα του πυρήνα PS/2 στέλνονται σε πλαίσια των 11μπιτ. Το πρώτο μπιτ είναι μπιτ εκκίνησης, στην συνέχεια στέλνονται τα 8 μπιτ δεδομένων, μετά ένα μπιτ μονής ισοτιμίας (odd parity bit) και τέλος ένα μπιτ σταματήματος [26].



Σχήμα 4.14: Πλαίσιο 11 μπιτ για την επικοινωνία του APBPS2 [26]

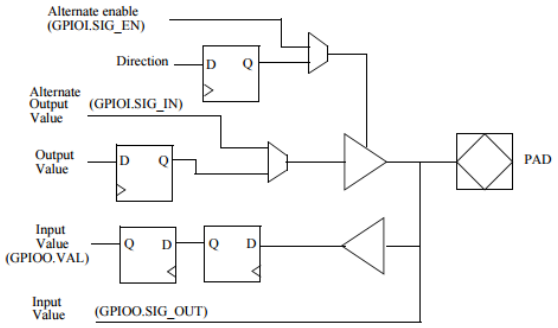
```
kbd : apbps2
 generic map(pindex => 4, paddr => 4, pirq => 4)
     port map(rstn, clkm, apbi, apbo(4), moui, mouo);

kbd2 : apbps2
  generic map(pindex => 5, paddr => 5, pirq => 5)
     port map(rstn, clkm, apbi, apbo(5), kbdi, kbdo);

 kbdclk_pad : iopad generic map (tech => padtech)
     port map (PS2_CLK,kbdo.ps2_clk_o, kbdo.ps2_clk_oe, kbdi.ps2_clk_i);
 kbdata_pad : iopad generic map (tech => padtech)
       port map (PS2_DAT, kbdo.ps2_data_o, kbdo.ps2_data_oe, kbdi.ps2_data_i);
 mouclk_pad : iopad generic map (tech => padtech)
     port map (PS2_CLK2,mouo.ps2_clk_o, mouo.ps2_clk_oe, moui.ps2_clk_i);
 mouata_pad : iopad generic map (tech => padtech)
    port map (PS2_DAT2, mouo.ps2_data_o, mouo.ps2_data_oe, moui.ps2_data_i);
```

Εικόνα 4.10: Σύνδεση πληκτρολογίου και ποντικιού με τον επεξεργαστή σε γλώσσα VHDL

4.2.10 Γενικού σκοπού θύρες εισόδου/εξόδου (GPIO)

Ο GPIO (General Purpose Input/Output, γενικού σκοπού είσοδος/έξοδος) είναι ένας ακροδέκτης ενός ολοκληρωμένου κυκλώματος που η συμπεριφορά του (ακόμα και το αν θα λειτουργεί σαν είσοδος η έξοδος) ρυθμίζεται από το χρήστη μέσω λογισμικού. Οι ακροδέκτες GPIO από προεπιλογή είναι απενεργοποιημένοι και δεν έχουν κάποια προκαθορισμένη χρήση. Η ιδέα χρήσης τους είναι ότι κατά την ανάπτυξη ενός συστήματος μπορεί να χρειαστούν επιπλέον ψηφιακές γραμμές ελέγχου. Έχοντας τις θύρες GPIO στο τσιπ δεν χρειάζεται να υλοποιηθούν επιπλέον κυκλώματα για την παροχή αυτών των επιπλέον γραμμών.

Ο πυρήνας GPIO του συστήματος είναι επεκτάσιμος και παρέχει την επιλογή υποστήριξης διακοπών. Το εύρος των θυρών μπορεί να καθοριστεί από 2-32 μπιτ, μέσω των VHDL παραμέτρων. Η δημιουργία της διακοπής είναι διαθέσιμη μόνο στις γραμμές, όπου το αντίστοιχο μπιτ της VHDL παραμέτρου (‘imask’) έχει τη τιμή 1. Κάθε μπιτ της θύρας GPIO μπορεί να οριστεί σαν σήμα εισόδου ή εξόδου και μπορεί να δημιουργήσει διακοπή. Για τη δημιουργία της διακοπής πρέπει να καθοριστεί η πολικότητα (polarity) τους σήματος και το αν θα είναι διακοπή επιπέδου/κορυφής (level/edge). Ακόμα υπάρχει η δυνατότητα της χρήσης των GPIO ακροδεκτών από άλλα σήματα. Σε αυτή την περίπτωση ο καταχωρητής εξόδου μπορεί να παρακαμφθεί από τον καταχωρητή παράκαμψης (‘bypass register’) [26].



Σχήμα 4.15 Διάγραμμα βαθμίδων ενός GPIO pad [26]

Στο σχήμα 4.15 φαίνεται το διάγραμμα βαθμίδων μια γραμμής εισόδου/εξόδου GPIO (‘GPIO pad’) και στον πίνακα 5 οι καταχωρητές του πυρήνα GPIO. Αλλάζοντας τις τιμές των καταχωρητών μπορεί να μεταβληθεί η λειτουργία της κάθε GPIO θύρας [26].

Πίνακας 5: Καταχωρητές πυρήνα GPIO [26]

APB address offset	Register
0x00	I/O port data register
0x04	I/O port output register
0x08	I/O port direction register
0x0C	Interrupt mask register
0x10	Interrupt polarity register
0x14	Interrupt edge register
0x18	Bypass register
0x1C	Capability register
0x20 - 0x3C	Interrupt map register(s). Address 0x20 + 4*n contains interrupt map registers for IO[4*n : 3+4+n], if implemented.
0x40	Interrupt available register, if implemented
0x44	Interrupt flag register, if implemented
0x48	Input enable register, if implemented
0x4C	Pulse register, if implemented
0x50	Input enable register, if implemented, logical-OR
0x54	I/O port output register, logical-OR
0x58	I/O port direction register, logical-OR
0x5C	Interrupt mask register, logical-OR
0x60	Input enable register, if implemented, logical-AND
0x64	I/O port output register, logical-AND
0x68	I/O port direction register, logical-AND
0x6C	Interrupt mask register, logical-AND
0x70	Input enable register, if implemented, logical-XOR
0x74	I/O port output register, logical-XOR
0x78	I/O port direction register, logical-XOR
0x7C	Interrupt mask register, logical-XOR

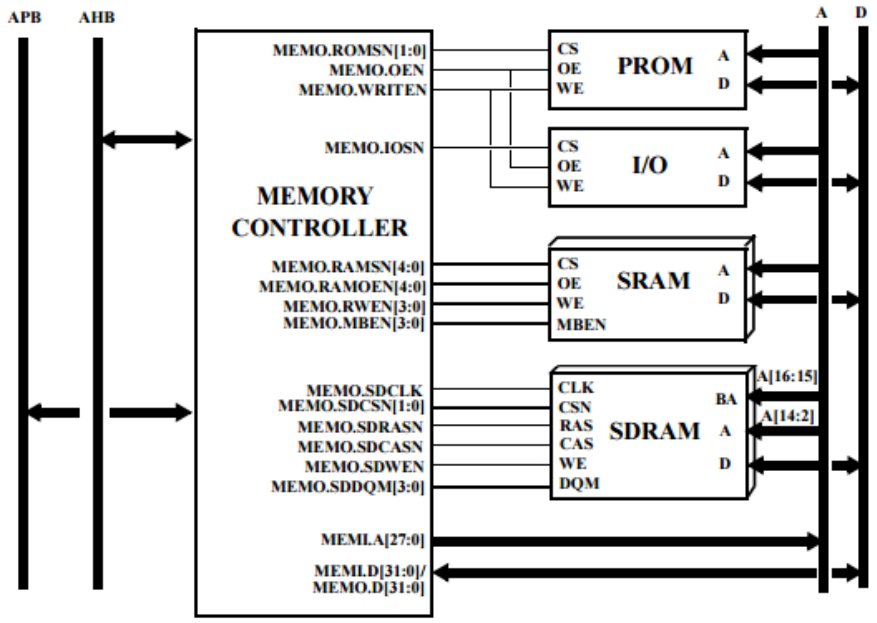
4.2.11 Ελεγκτής μνήμης (LEON2 Memory Controller)

Η κύρια αποστολή ενός ελεγκτή μνήμης είναι το διάβασμα, το γράψιμο και η ανανέωση (refresh) της μνήμης τυχαίας προσπέλασης (RAM). Εάν δεν υπάρχει αυτή η ανανέωση μπορούν να χαθούν δεδομένα από τη μνήμη. Για αυτό το λόγο η χρήση ενός ελεγκτή μνήμης είναι πολύ σημαντική. Ο ελεγκτής μνήμης γράφει και διαβάζει τη μνήμη RAM μέσω πολυπλεκτών και από-πολυπλεκτών (demultiplexers). Επιλέγει τη κατάλληλη γραμμή, στήλη και θέση μνήμης για το γράψιμο/διάβασμα των δεδομένων.

Ο ελεγκτής μνήμης του συστήματος της διπλωματικής χειρίζεται διαύλους μνήμης που συνδέονται με μια μνήμη PROM, με συσκευές εισόδου/εξόδου (memory mapped I/O devices), με μια ασύγχρονη στατική RAM (SRAM) και μια σύγχρονη δυναμική RAM

(SDRAM). Ο ελεγκτής δρα ως υποτελής στον δίαυλο AHB. Η λειτουργία του ελεγκτή μνήμης προγραμματίζεται μέσω των καταχωρητών ρύθμισης 1, 2, 3 (MCFG1, MCFG2 & MCFG3), διαμέσω του διαύλου APB. Ο δίαυλος μνήμης μπορεί να καθοριστεί ώστε να έχει λειτουργία 8 ή 16 μπιτ για εφαρμογές με χαμηλή μνήμη και με απαιτήσεις επιδόσεων.

Η αποκωδικοποίηση για την επιλογή του τσιπ (chip-select decoding) γίνεται για δύο τράπεζες μνήμης PROM (memory banks), μια τράπεζα μνήμης εισόδου/εξόδου, πέντε τράπεζες μνήμης SRAM και δύο τράπεζες μνήμης SDRAM. Ο ελεγκτής αποκωδικοποιεί τρία διαστήματα διευθύνσεων (address space) (PROM, I/O και RAM), των οποίων η αντιστοίχιση γίνεται μέσω των VHDL παραμέτρων [26].

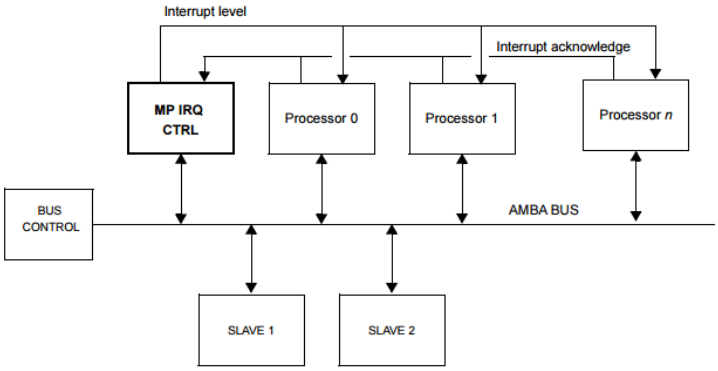


Σχήμα 4.16: Ο ελεγκτής μνήμης συνδεδεμένος στον AMBA δίαυλο και σε διαφορετικά είδη συσκευών μνήμης [26]

4.2.12 Ελεγκτής διακοπών για πολύ-επεξεργαστικό σύστημα (Multi-processor interrupt controller)

Ένας προγραμματιζόμενος ελεγκτής διακοπών (PIC) είναι μια συσκευή που χρησιμοποιείται για να επιτευχθεί ο συνδυασμός διαφόρων διακοπών σε ένα σύστημα με ένα ή περισσότερους επεξεργαστές. Ο ελεγκτής διακοπών εισάγει επίπεδα προτεραιότητας στις διακοπές. Όταν η συσκευή πρέπει να διαχειριστεί πολλές διακοπές, τις τοποθετεί σε μια σειρά προτεραιότητας.

Το σύστημα AMBA της GRLIB παρέχει ένα σύστημα διακοπών, όπου οι γραμμές των διακοπών δρομολογούνται μαζί με τα υπόλοιπα σήματα των APB/AHB διαύλων, δημιουργώντας ένα δίαυλο διακοπών. Οι διακοπές από τις μονάδες που είναι συνδεδεμένες στους διαύλους AHB και APB δρομολογούνται διαμέσου του διαύλου, συνδυάζονται και διαδίδονται πίσω στις μονάδες. Ο ελεγκτής διακοπών είναι προσαρμοσμένος πάνω στον δίαυλο APB ως ένας υποτελής και παρακολουθεί τις διακοπές. Ο ελεγκτής διακοπών βάζει προτεραιότητες, βάζει μάσκα και προωθεί τη διακοπή με την μεγαλύτερη προτεραιότητα στον επεξεργαστή. Σε πολύ-επεξεργαστικά συστήματα η διακοπή προωθείται σε όλους τους επεξεργαστές [26].

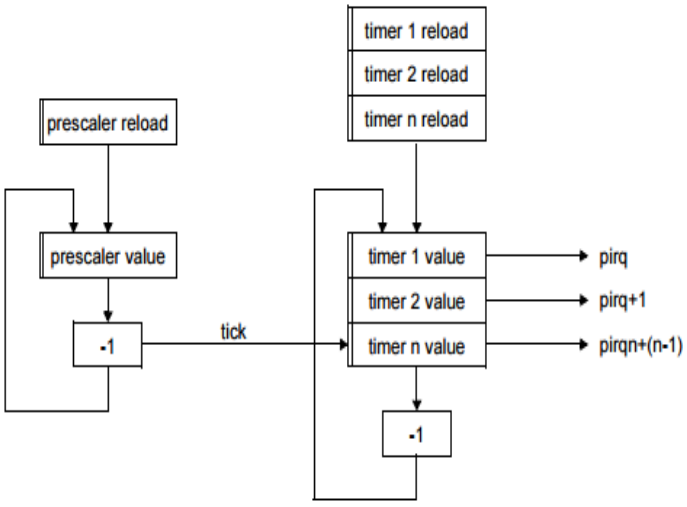


Σχήμα 4.17 Πολυεπεξεργαστικό σύστημα LEON με ελεγκτή διακοπών [26]

4.2.13 Γενικού σκοπού μονάδα χρονισμού (GPTIMER)

Η μονάδα χρονισμού είναι πολύ σημαντική στα ενσωματωμένα συστήματα. Μπορεί να μετράει τους κύκλους ρολογιού του επεξεργαστή ή να μετράει γεγονότα που συμβαίνουν. Πολλά λειτουργικά συστήματα χρειάζονται αυτή τη μονάδα για να λειτουργήσουν, καθώς συμβάλλει στη δημιουργία των διακοπών λογισμικού και τη λειτουργία των οδηγητών (drivers).

Η γενικού σκοπού μονάδα χρονισμού του LEON3 επεξεργαστή παρέχει έναν κοινό 'prescaler' και μειούμενους (decrementing) χρονομετρητές. Ο αριθμός και το πλάτος των χρονομετρητών είναι παραμετροποιήσιμος μέσω των VHDL παραμέτρων. Η μονάδα χρονισμού συμπεριφέρεται ως υποτελής στο δίαυλο APB. Ακόμα η μονάδα είναι ικανή να παρέχει διακοπές όταν περάσει κάποιο κατώφλι. Η διακοπή που δημιουργείται μπορεί να ρυθμιστεί να είναι κοινή για ολόκληρη τη μονάδα χρονισμού ή ξεχωριστή για κάθε χρονομετρητή [26].

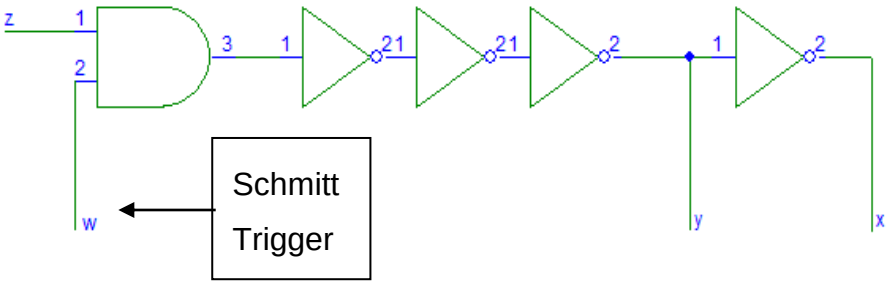


Σχήμα 4.18: Διάγραμμα βαθμίδων μονάδας χρονισμού [26]

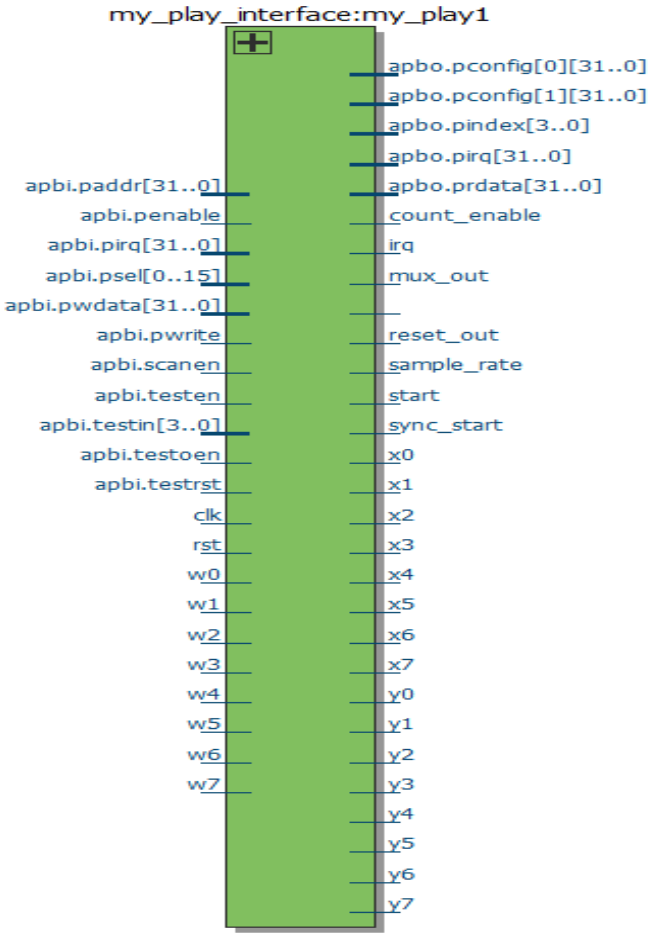
4.3 Κύκλωμα διεπαφής αισθητήρων

Στο κεφάλαιο 3 έγινε μια γενική παρουσίαση του κυκλώματος μέτρησης της συχνότητας και της διεπαφής των αισθητήρων. Σε αυτό το σημείο γίνεται μια πιο αναλυτική παρουσίαση του κυκλώματος. Παρουσιάζονται οι αλλαγές που έγιναν για την σύνδεση του με τον APB δίαυλο και ο τρόπος επικοινωνίας με τον επεξεργαστή. Στο σχήμα 4.20 φαίνονται τα σήματα εισόδου και εξόδου του κυκλώματος αισθητήρων, το οποίο μπορεί να μετρήσει τη συχνότητα μέχρι και σε 8 συνδεδεμένους αισθητήρες. Τα σήματα w0,x0,y0,w1,x1,y1 κ.τ.λ. αντιστοιχούν στα σημεία του ταλαντωτή δακτυλίου που

περιγράφηκε στο κεφάλαιο 3 (σχήμα 4.19) και ο αντίστοιχος αριθμός που ακολουθεί τα γράμματα w,x,y, δείχνει τον αριθμό του ταλαντωτή.



Σχήμα 4.19: Ταλαντωτής δακτυλίου με Schmitt Trigger [31]



Σχήμα 4.20: Σήματα εισόδου/εξόδου και σύνδεσης με APB δίαυλο της διεπαφής αισθητήρων

4.3.1 Περιγραφή σημάτων της διεπαφής αισθητήρων

Τα σήματα ‘count_enable’, ‘mux_out’, ‘reset_out_pin’, ‘sample_rate’, ‘start’ και ‘sync_start’ χρησιμοποιούνται για την αποσφαλμάτωση του κυκλώματος. Το ‘count_enable‘ δείχνει πότε ο μετρητής λειτουργεί, το ‘mux_out είναι η έξοδος του ταλαντωτή που χρησιμοποιείται για έλεγχο του κυκλώματος, το ‘reset_out_pin’ δείχνει πότε γίνεται η επανεκκίνηση του μετρητή, το ‘sample_rate’ δείχνει το ρυθμό δειγματοληψίας, το σήμα ‘start’ δείχνει πότε έρχεται εντολή εκκίνησης και το ‘sync_start’ πότε ξεκινά η δειγματοληψία. Το σήμα ‘irq’ δείχνει ότι τελείωσε η μέτρηση και ουσιαστικά είναι η διακοπή που στέλνεται στον επεξεργαστή. Όλα τα σήματα που περιεγράφηκαν μέχρι τώρα (εκτός του ‘irq’) οδηγούνται εξωτερικά της συσκευής, χρησιμοποιώντας ένα βύσμα HSMC το οποίο θα περιγραφεί στη συνέχεια.

Τα υπόλοιπα σήματα αφορούν τη σύνδεση του κυκλώματος με τον δίαυλο APB για την επικοινωνία με τον επεξεργαστή [26]:

apbi.paddr[31:0] (APB address bus): Ο δίαυλος διευθύνσεων του APB. Μπορεί να έχει μέγεθος ως και 32 μπιτ.

apbi.penable (APB strobe): Το σήμα αυτό χρησιμοποιείται για τον συγχρονισμό όλων των προσβάσεων (διάβασμα/γράψιμο) στον δίαυλο. Το σήμα ενεργοποίησης χρησιμοποιείται για να δείξει ότι η μεταφορά (transfer) βρίσκεται στον δεύτερο κύκλο. Η ανερχόμενη παρυφή του 'penable' συμβαίνει στη μέση μιας μεταφοράς του APB διαύλου.

apbi.pirq: Ο ελεγκτής διακοπών ελέγχει τις διακοπές 1-15 του διαύλου διακοπών (APBI.PIRQ[15:1]). Όταν κάποια από τις γραμμές ενεργοποιηθεί, τότε ενεργοποιείται και το αντίστοιχο μπιτ στον καταχωρητή διακοπών. Το μπιτ αυτό θα μείνει στο λογικό 1 ακόμα και αν η PIRQ γραμμή απενεργοποιηθεί. Απενεργοποιείται είτε με εντολή λογισμικού είτε από σήμα αναγνώρισης της διακοπής από τον επεξεργαστή. Ο συγκεκριμένος δίαυλος δεν χρησιμοποιείται στο κύκλωμα.

apbi.psel (σήμα επιλογής APB) : Είναι ένα σήμα από τον αποκωδικοποιητή της περιφερειακής μονάδας γέφυρας του διαύλου και συνδέεται με κάθε περιφερειακό υποτελή του διαύλου. Το σήμα δείχνει ότι η υποτελής συσκευή έχει επιλεχτεί και απαιτείται μια μεταφορά δεδομένων.

apbi.pwdata (APB δίαυλος εγγραφής):Ο δίαυλος εγγραφής δεδομένων οδηγείται από την περιφερειακή μονάδα γέφυρας του διαύλου κατά τη διάρκεια των κύκλων εγγραφής (όταν το PWRITE είναι στο λογικό 1). Ο δίαυλος εγγραφής μπορεί να έχει μήκος ως και 32 μπιτ.

apbi.pwrite (κατεύθυνση μεταφοράς στο APB): Όταν είναι στο λογικό 1 το σήμα δείχνει ότι συμβαίνει μια εγγραφή στον δίαυλο APB και όταν είναι στο 0 δείχνει ότι γίνεται διάβασμα.

apbo.prdata (Δίαυλος διαβάσματος): Ο δίαυλος διαβάσματος οδηγείται από τον επιλεγμένο υποτελή κατά την διάρκεια των κύκλων διαβάσματος (όταν το PWRITE είναι στο 0). Ο δίαυλος διαβάσματος μπορεί να έχει ως και 32 μπιτ μέγεθος.

apbi.scanen, apbi.testen, apbi.testin, apbi.testoen apbi.testrst: Σήματα για αποσφαλμάτωση. Δεν χρησιμοποιούνται στο κύκλωμα.

Apbo.pconfig[0] [1]: Δίαυλοι ρυθμίσεων των περιφερειακών. Περιέχουν πληροφορίες για κάθε περιφερειακό.

apbo.pindex: αριθμός αναγνώρισης του κυκλώματος

Για την σύνδεση της διεπαφής με τον δίαυλο APB πρέπει να συνδεθούν τα σήματα του διαύλου κατάλληλα με τους καταχωρητές του κυκλώματος διεπαφής και πρέπει να ακολουθηθεί το πρωτόκολλο επικοινωνίας του διαύλου. Στην εικόνα 4.11 φαίνεται σε γλώσσα VHDL ο τρόπος σύνδεσης της διεπαφής αισθητήρων με τον δίαυλο APB. Στην εικόνα φαίνονται οι καταχωρητές 'oscillator_control', 'sampling time_control', 'signal_control' και 'wr' που χρησιμοποιούνται από το κύκλωμα διεπαφής όπως περιεγράφηκε στο κεφάλαιο 3.7. Αρχικά, έχουν τιμή μηδέν. Κατά την ενεργοποίηση του σήματος επανεκκίνησης (rst) μηδενίζονται ξανά. Το 'rdata' είναι ένας καταχωρητής που χρησιμοποιείται για την αποστολή των αποτελεσμάτων των μετρήσεων στον δίαυλο APB. Σε αυτόν τον καταχωρητή αποθηκεύεται το αποτέλεσμα κατά την διάρκεια του διαβάσματος και στη συνέχεια στέλνεται στον δίαυλο 'apbo.prdata'.

```
reg : process(rst, clk)
begin
  if (rst = '0') then --Ενεργοποίηση επανεκκίνησης
  rdata <= (others => '0');
  oscilator_control  <= (others => '0');
  sampling_time_control  <= (others => '0');
  signal_control  <= (others => '0');
  wr <='0';
elsif rising_edge(clk) then
  rdata <= (others => '0'); --αρχικοποίηση σημάτων--
  wr <='0';
if (apbi.psel(pindex)) = '1' then --διάβασμα αποτελεσμάτων
 case apbi.paddr(4 downto 2) is
  when "001" =>
     rdata(15 downto 0) <= result;
     rdata(31 downto 16) <=(others => '0');
   when others => null;
 end case;
end if;
  --γράψιμο στους καταχωρητές του κυκλώματος
 if (apbi.psel(pindex) and apbi.penable and apbi.pwrite) = '1' then
  case apbi.paddr(4 downto 2) is
   when "000" =>  --όλοι οι καταχωρητές γράφονται με μια εντολή γραψίματος
    oscilator_control <= apbi.pwdata(4 downto 0);
    sampling_time_control <= apbi.pwdata(9 downto 5);
    signal_control <= apbi.pwdata(14 downto 10);
    wr <= apbi.pwdata(15);
    reset <= apbi.pwdata(16);
   when others => null;
  end case;
 end if;
 end if;
 end process;
   apbo.prdata <= rdata;
```

Εικόνα 4.11: Περιγραφή σε γλώσσα VHDL της σύνδεσης της διεπαφής αισθητήρων με τον δίαυλο AMBA.

Για την διασύνδεση της διεπαφής με το λογισμικό χρησιμοποιούνται 2 καταχωρητές, ένας για διάβασμα και ένας για γράψιμο. Όταν μέσω του λογισμικού έρθει μια εντολή για διάβασμα ή γράψιμο των καταχωρητών το σήμα ‘apbi.psel’ θα γίνει ‘1’. Για το γράψιμο θα πρέπει επίσης το ‘apbi.penable’ και το ‘apbi.write’ να γίνει ‘1’. Αυτά τα σήματα θα ενεργοποιηθούν όποτε ο επεξεργαστής είναι διαθέσιμος.

Στη συνέχεια ελέγχεται ποιος από τους δύο καταχωρητές έχει ζητηθεί. Αυτό εντοπίζεται από το σήμα ‘apbi.paddr’. Αν το σήμα είναι ‘0’ (apbi.paddr(4 downto 2)=”000”) τότε σημαίνει ότι τα δεδομένα που βρίσκονται στον δίαυλο ‘apbi.pwdata’ πρέπει να γραφτούν στους καταχωρητές όπως φαίνεται στην εικόνα 4.11. Αν το σήμα ‘apbi.paddr’ είναι ‘4’ (apbi.paddr(4 downto 2)=”001”) τότε θα πρέπει να διαβαστεί ο καταχωρητής ‘results’ της διεπαφής και η τιμή του να περάσει στον καταχωρητή ‘rdata’. Αν χρειαζόντουσαν και άλλοι καταχωρητές για την επικοινωνία με το λογισμικό μπορούν να προστεθούν και άλλες συνθήκες για το ‘apbi.paddr’ με την εντολή ‘case’ της VHDL. Ο κάθε καταχωρητής που χρησιμοποιείται καταλαμβάνει 4 θέσεις στο ‘apbi.paddr’. Το ‘case’ για έναν ακόμα καταχωρητή θα ήταν: case apbi.paddr(4 downto 2) when =>”010”.

Στο λογισμικό θα πρέπει να δημιουργηθεί η διασύνδεση με τους καταχωρητές με τον εξής τρόπο:

```
struct my_play_regs_t  * my_play_regs = (struct my_play_regs_t  *)pu;

struct my_play_regs_t {

volatile int write;

volatile int read;

};
```

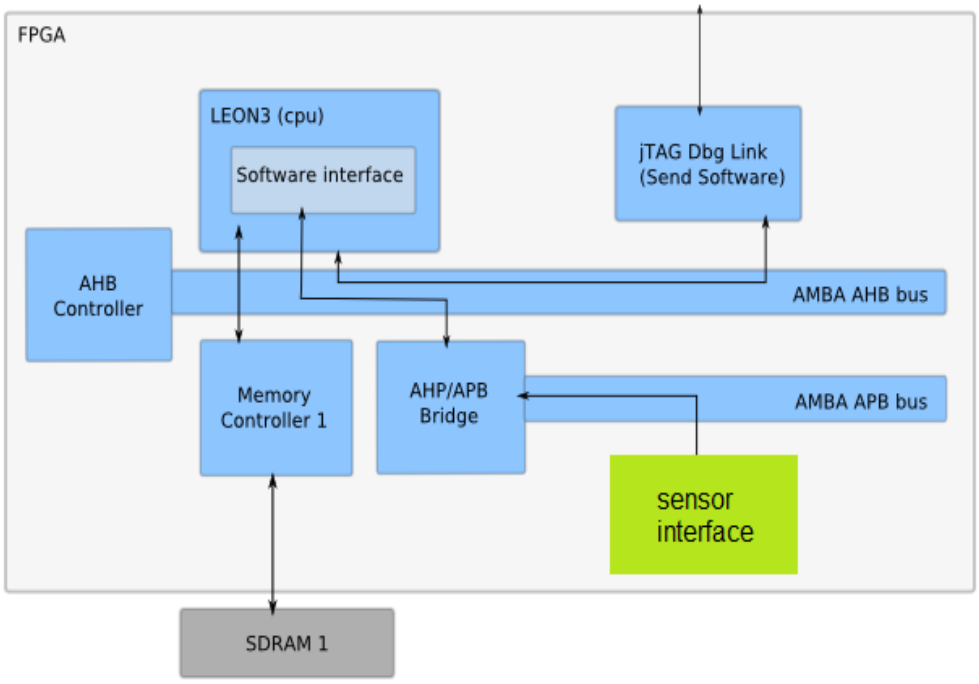
Με αυτές τις εντολές ορίζεται ο τρόπος διασύνδεσης μεταξύ του προγράμματος λογισμικού και του περιφερειακού του διαύλου APB. Ορίζονται οι 2 καταχωρητές που χρησιμοποιούνται ‘write’, ‘read’, με πρώτο τον ‘write’ όπως ειπώθηκε. Το ‘pu’ αντιστοιχεί στην φυσική διεύθυνση του περιφερειακού που είναι συνδεδεμένο στον δίαυλο APB. ‘Ετσι όταν στο λογισμικό γραφτεί η εντολή:

```
my_play_regs->write=x;
```

Θα σταλεί μια εντολή διαβάσματος προς το περιφερειακό που έχει οριστεί. Θα ενεργοποιηθούν τα σήματα που περιεγράφηκαν (‘apbi.psel’, ‘apbi.penable’, ‘apbi.write’) και η τιμή της μεταβλητής ‘x’ θα περάσει στους καταχωρητές (‘oscillator_control’, ‘sampling time_control’, ‘signal_control’ και ‘wr’). Όταν γραφτεί η εντολή:

```
y=my_play_regs->read;
```

Θα σταλεί μια εντολή διαβάσματος στο περιφερειακό. Η τιμή του καταχωρητή ‘result’ θα περάσει στην μεταβλητή ‘y’. Στο σχήμα 4.21 φαίνεται ένα σχηματικό διάγραμμα με τον τρόπο που είναι συνδεδεμένο το περιφερειακό με τον δίαυλο και τον επεξεργαστή.



Σχήμα 4.21: Σύνδεση διεπαφής αισθητήρων στο επεξεργαστικό σύστημα Leon3

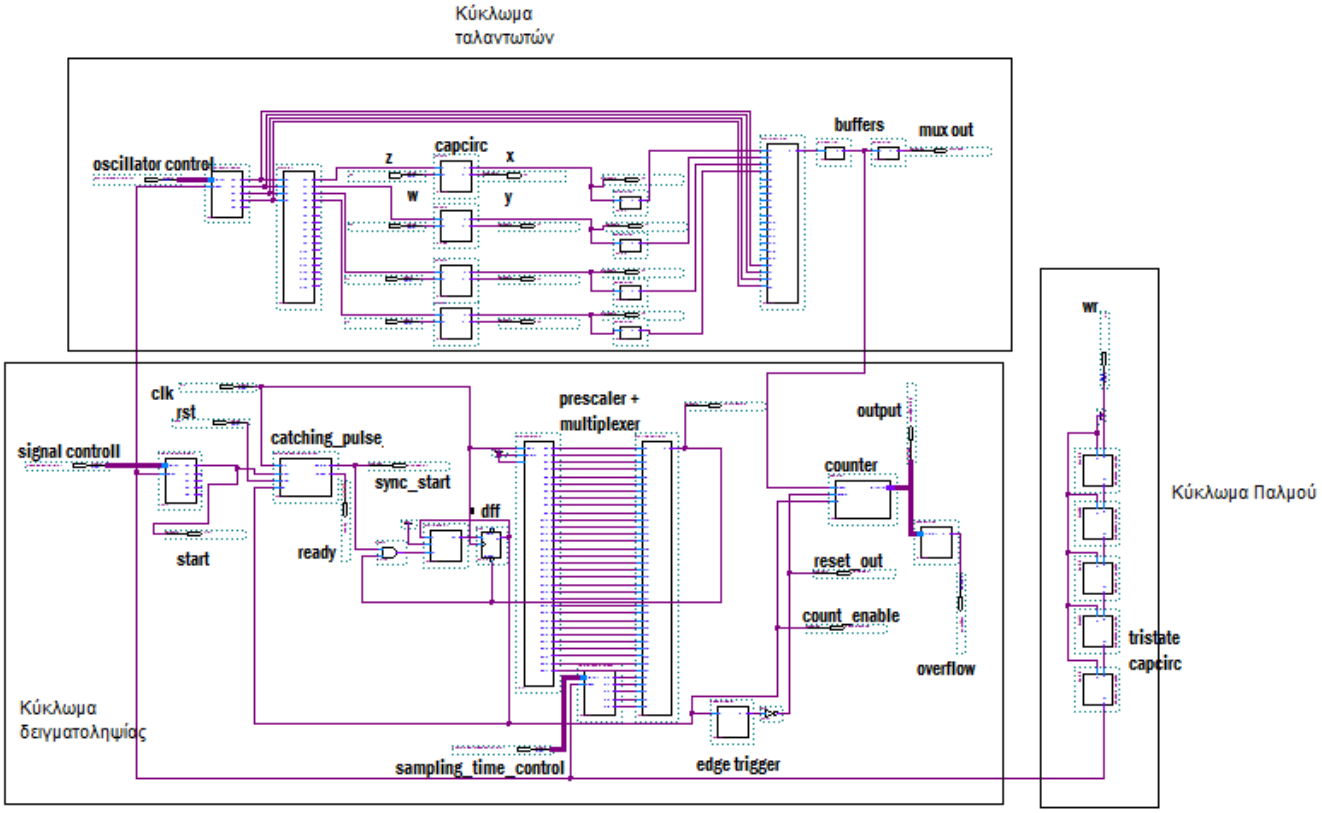
Στην εικόνα 4.12 φαίνεται η σύνδεση της διεπαφής των αισθητήρων στο επίπεδο διασύνδεσης του διαύλου APB και ο καθορισμός των καταχωρητών για την επικοινωνία.

```
signal wr,ready: std_logic;
signal reset :std_logic:='0';
signal oscilator_control,sampling_time_control,signal_control :std_logic_vector(4 downto 0);
signal result :  STD_LOGIC_VECTOR(15 DOWNTO 0);

--constant REVISION       : amba_version_type := 0;
constant pconfig       : apb_config_type := (
                       0 => ahb_device_reg ( VENDOR_OPENCORES, OPENCORES_my_play, 0, 0, 0),
                       1 => apb_iobar(paddr, pmask));

  --signal r, rin : std_logic;
   signal rdata : std_logic_vector(31 downto 0); --καταχωρητής rdata
begin

  irq <= ready; --η διακοπή του κυκλώματος η οποία στη συνέχεια συνδέεται με το gpio pad
  my_play2 : my_play_top --port map της διεπαφής αισθητήρων στο επίπεδο επικοινωνίας με τον δίαυλο APB
  port map
   (
      clk => clk , w0 => w0 ,  w1 => w1, w2 => w2 ,w3 => w3 ,w4 => w4 ,w5 => w5 ,
      w6 => w6 ,w7 => w7 ,w8 => w8 ,w9 => w9 ,w10 => w10 ,w11 => w11 ,w12 => w12 ,
      w13 => w13,w14 => w14 , w15 => w15 ,wr => wr ,reset => reset ,oscilator_control => oscilator_control  ,
      sampling_time_control => sampling_time_control  ,signal_control => signal_control ,
      x0 =>  x0,x1 => x1 ,x2 => x2 ,x3 => x3 ,x4 => x4 ,x5 =>  x5,x6 => x6 ,x7 => x7,x8 => x8 ,x9 => x9,
      x10 => x10 ,x11 => x11 ,x12 => x12 ,x13 => x13 ,x14 => x14 ,x15 => x15 ,y0 => y0 ,y1 => y1 ,
      y2 => y2 ,y3 => y3 ,y4 => y4 ,y5 => y5 ,y6 => y6 ,y7 => y7 ,y8 => y8 ,y9 => y9 ,
      y10 => y10 ,y11 => y11 ,y12 => y12 ,y13 => y13 ,y14 => y14 ,y15 => y15 ,overflow => overflow ,
      mux_out => mux_out ,ready => ready,result => result
   );
```

Εικόνα 4.12: Σύνδεση διεπαφής αισθητήρων με VHDL και δήλωση καταχωρητών

4.3.2 Κύκλωμα διεπαφής - τρόπος λειτουργίας και αλλαγές

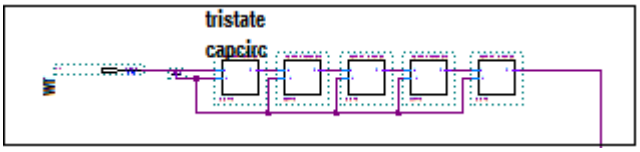
Όπως περιγράφηκε στο κεφάλαιο 3 το κύκλωμα μετράει τη συχνότητα των αισθητήρων που είναι συνδεδεμένοι σε αυτό. Το κύκλωμα αποτελείται ουσιαστικά από 3 κομμάτια, το κύκλωμα δειγματοληψίας, το κύκλωμα παλμού και το κύκλωμα ταλαντωτών, όπως φαίνεται και στο σχήμα 4.22. Σε σχέση με το αρχικό κύκλωμα που παρουσιάστηκε έχουν γίνει κάποιες αλλαγές. Καταρχήν υπάρχουν 3 εσωτερικοί καταχωρητές των 5 μπιτ ο καθένας. Αυτοί είναι ο ‘oscillator control’, ο ‘signal control’ και ο ‘sampling time control’. Ο καταχωρητής ελέγχου (‘FPGA control’) δεν υπάρχει καθώς και οι 3 καταχωρητές ανανεώνονται με μια εντολή. Επίσης υπάρχει το ‘wr’ που είναι 1 μπιτ και είναι το σήμα που επιτρέπει την εγγραφή των δεδομένων στους καταχωρητές. Ακόμα έχει προστεθεί ένα νέο κύκλωμα, το ‘catching pulse’, το οποίο επιτρέπει την έναρξη της δειγματοληψίας με ένα μόνο παλμό του σήματος ‘wr’ και είναι υπεύθυνο για τον σωστό συγχρονισμό της μέτρησης. Το ‘catching pulse’ κύκλωμα είναι υπεύθυνο και για την διακοπή (‘Interrupt’) που στέλνεται στον επεξεργαστή όταν έχει ολοκληρωθεί η μέτρηση. Έχει προστεθεί μία ακόμα διακοπή στο κύκλωμα σε περίπτωση που γίνει κάποια υπερχείλιση στον μετρητή. Τέλος, η τελική υλοποίηση περιλαμβάνει 4 ταλαντωτές, αριθμός ο οποίος μπορεί να αλλάξει. Στη συνέχεια παρουσιάζεται το κάθε μέρος του κυκλώματος ξεχωριστά.



Σχήμα 4.22: Συνολικό κύκλωμα μέτρησης συχνοτήτων

4.3.2.1 Κύκλωμα παλμού

Το κύκλωμα παλμού παίρνει ως είσοδο το σήμα ‘wr’, δηλαδή το σήμα εγγραφής των καταχωρητών. Στην έξοδο βγαίνει ένα καθυστερημένο σήμα ως προς το ‘wr’. Αυτή η καθυστέρηση δημιουργείται με στάδια πυλών ‘and’ και ‘not’. Το κάθε κύκλωμα ‘tristate_capcirc’ περιλαμβάνει ένα στάδιο ‘and’ και τρία στάδια ‘not’. Αυτή η καθυστέρηση επιτρέπει στο σήμα ‘wr’ να χρησιμοποιηθεί ως ρολόι στα κυκλώματα εγγραφής των καταχωρητών. Όταν το ‘wr’ αλλάζει πολύ γρήγορα η αποθήκευση των τιμών των καταχωρητών μπορεί να μην γίνει σωστά. Ουσιαστικά το κύκλωμα παλμού έχει προστεθεί για λόγους αξιοπιστίας και σταθερότητας του συστήματος. Το κύκλωμα παλμού συνολικά επιτρέπει την σωστή ενημέρωση των καταχωρητών του συστήματος.

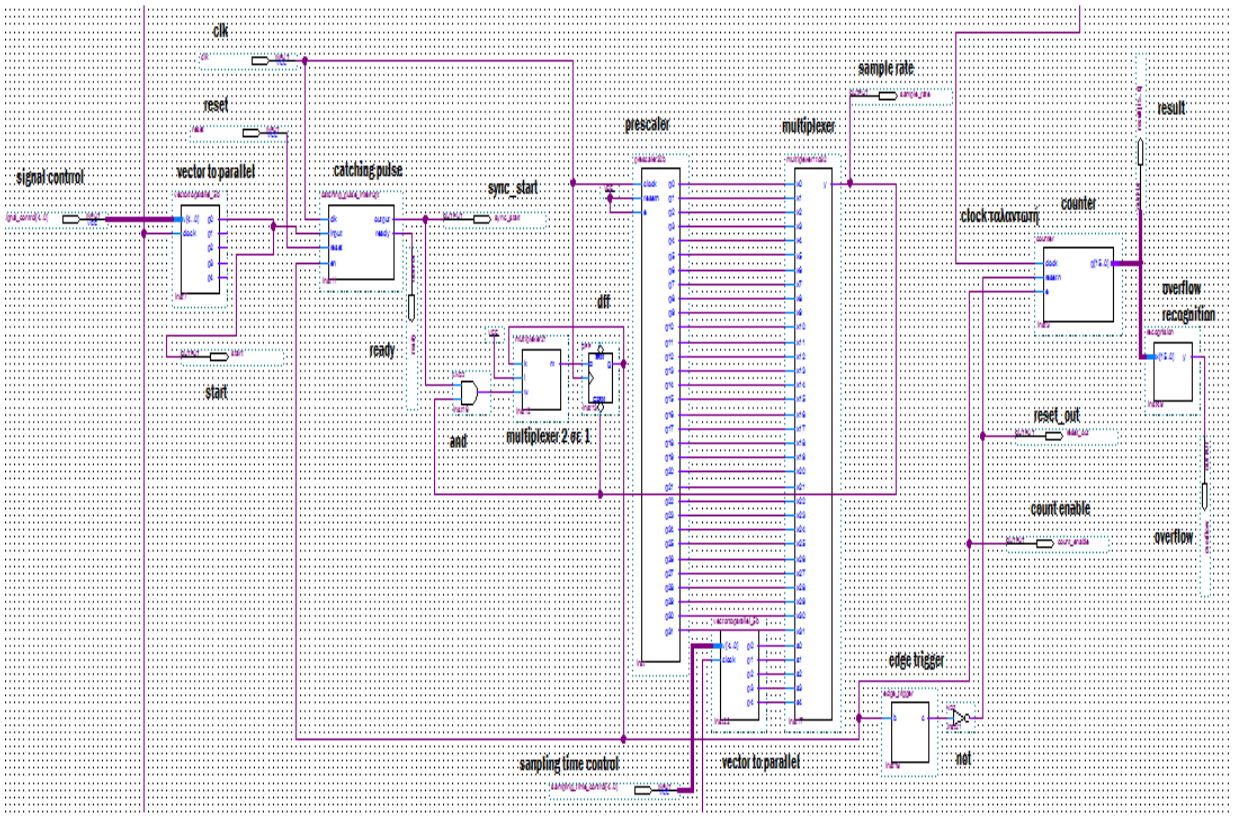


Σχήμα 4.23: Κύκλωμα παλμού

4.3.2.2 Κύκλωμα δειγματοληψίας

Το κύκλωμα δειγματοληψίας είναι υπεύθυνο για τη ρύθμιση του χρονικού παραθύρου, την δημιουργία των διακοπών, την λειτουργία του μετρητή και την έξοδο των αποτελεσμάτων. Όταν έρχεται μια εντολή εγγραφής σε καταχωρητές, δηλαδή όταν το σήμα από το κύκλωμα παλμού γίνει ‘1’, τα κυκλώματα ‘vector to parallel’ παίρνουν τα μπιτ των καταχωρητών και τα βγάζουν στην έξοδό τους παράλληλα. Το κύκλωμα ‘vector to parallel’ που οδηγείται από τον καταχωρητή ‘signal control’ ουσιαστικά βγάζει στην έξοδο μόνο το 1ο μπιτ. Αυτό το μπιτ θα έχει την τιμή ‘1’ όταν σταλεί η εντολή για γράψιμο καθώς όλες οι λειτουργίες σε αυτό το τροποποιημένο κύκλωμα γίνονται με μια εντολή. Με μία εντολή γραψίματος ενημερώνεται ταυτόχρονα ο ‘signal control’, ο ‘sampling time control’ και ο oscillator control’ καταχωρητής. Με την καθυστέρηση από

το κύκλωμα παλμού στο σήμα ‘wr’ είναι σίγουρο ότι οι τιμές θα έχουν περαστεί στους καταχωρητές σωστά. Όταν λοιπόν σταλεί η εντολή γραψίματος, η είσοδος του κυκλώματος ‘catching pulse’ θα γίνει ‘1’. Το κύκλωμα αυτό ουσιαστικά είναι ένα FSM. Αρχικά, περιμένει ένα σήμα γραψίματος, όταν αυτό έρθει, οδηγεί το σήμα ‘sync_start’ στην τιμή ‘1’ όπως και τη μία είσοδο της πύλης ‘and’. Η περιγραφή του πολυπλέκτη και του ‘prescaler’ έγινε στο κεφάλαιο 3. Υπενθυμίζεται εδώ ότι η έξοδος του πολυπλέκτη είναι μια υποδιαίρεση του ρολογιού του συστήματος. Αυτή η υποδιαίρεση καθορίζεται από την τιμή του καταχωρητή ‘sampling time control’.



Σχήμα 4.24: Κύκλωμα δειγματοληψίας

Στην επόμενη φάση το’ catching pulse’ κύκλωμα “κοιτάζει” αν το σήμα ‘count_enable’ έχει την τιμή ‘1’. Το ‘count enable’ είναι η έξοδος από το D flip flop . Το ‘catching pulse’ κύκλωμα θα οδηγήσει το σήμα ‘ready’ στην τιμή ‘1’, όταν το σήμα ‘count enable’ γίνει δύο φορές μηδέν. Αυτό γίνεται γιατί την στιγμή που δίνεται η εντολή γραψίματος μπορεί να αλλάξει και η συχνότητα δειγματοληψίας της μέτρησης και έτσι η πρώτη τιμή του μετρητή να μην είναι η σωστή. Για να είναι σίγουρα σωστή η μέτρηση θα πρέπει το σήμα ‘count enable’ να γίνει δύο φορές μηδέν. Όταν συμβεί αυτό σημαίνει ότι η μέτρηση έχει ολοκληρωθεί. Το σήμα ‘ready’ είναι η διακοπή του κυκλώματος, το σήμα που ενημερώνει τον επεξεργαστή ότι η μέτρηση έχει ολοκληρωθεί. Το αποτέλεσμα της μέτρησης είναι η έξοδος του μετρητή και αποθηκεύεται στον καταχωρητή ‘result’. Ο καταχωρητής ‘result’ μπορεί να διαβαστεί με μια εντολή διαβάσματος μέσω του διαύλου APB.

Το κύκλωμα του μετρητή αυξάνει κατά ένα όταν αναγνωρίζεται μια θετική κορυφή από το κύκλωμα των ταλαντωτών. Στο τέλος της δειγματοληψίας θα έχει μετρηθεί ο αριθμός των κορυφών στο χρονικό παράθυρο που έχει οριστεί. Ο μετρητής μπορεί να μετρήσει μέχρι την τιμή 65536. Αν οι κορυφές του ταλαντωτή υπερβούν αυτό τον αριθμό τότε δημιουργείται υπερχείλιση. Το κύκλωμα ‘recognition’ ελέγχει συνέχεια αν ο μετρητής έχει την τιμή 65536. Σε περίπτωση υπερχείλισης ενημερώνεται ο επεξεργαστής με μια διακοπή.

Οι διακοπές του κυκλώματος (ready, overflow) έχουν συνδεθεί με ‘GPIO pads’, τα οποία επιτρέπουν την εύκολη διαχείρισή τους. Στο κεφάλαιο 5 θα παρουσιαστεί ο τρόπος ενεργοποίησής τους με το λειτουργικό σύστημα Linux

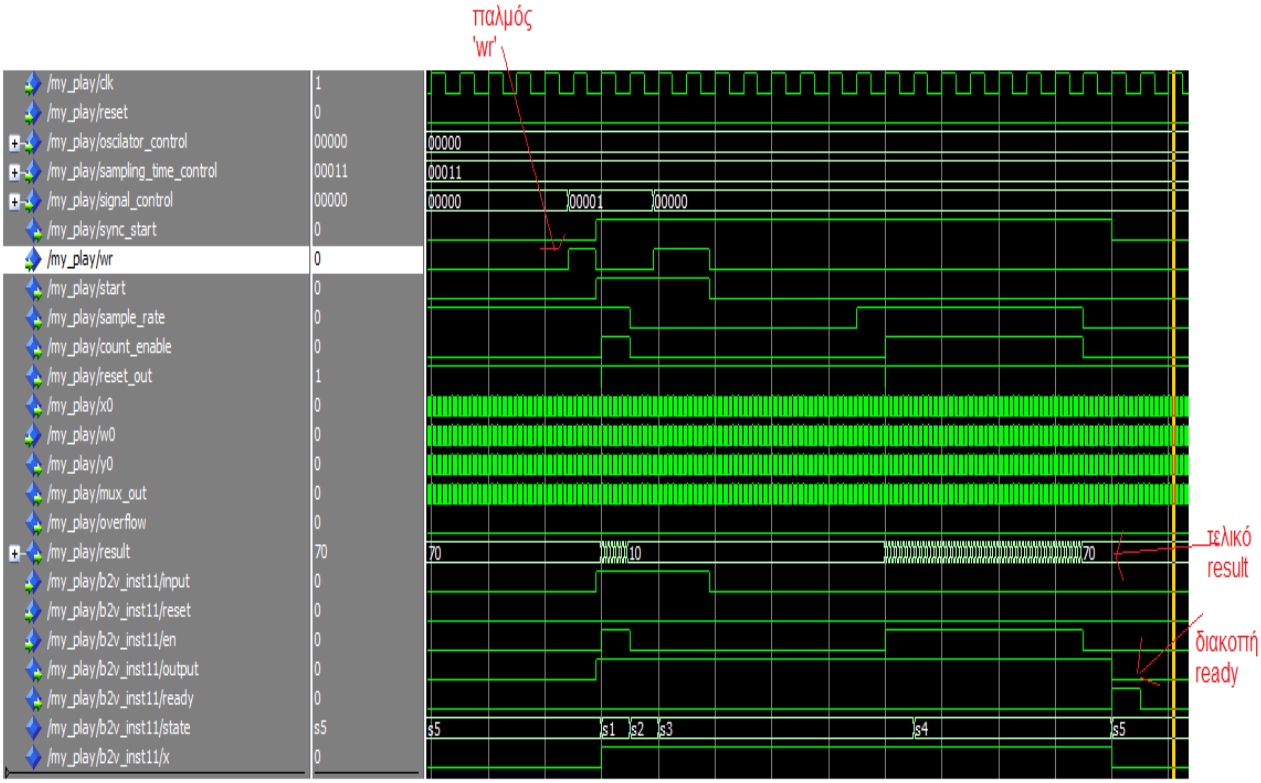
```
gpio_irq : if CFG_GRGPIO_ENABLE /= 0 generate     -- GR GPIO unit
  grgpio0: grgpio
    generic map( pindex => 9, paddr => 9, imask => CFG_GRGPIO_IMASK, nbits => CFG_GRGPIO_WIDTH)
    port map( rstn, clkm, apbi, apbo(9), gpioi, gpioo);

    i2c_irq_pad: inpad generic map (tech => padtech) ---διακοπής οθόνης
        port map (lcd_irq, gpioi.din(1));

    my_play_irq_pad: inpad generic map (tech => padtech) --διακοπή ολοκλήρωσης της μέτρησης
        port map (my_play_irq, gpioi.din(2));

     overflow_irq_pad: inpad generic map (tech => padtech)-- διακοπή υπερχείλισης
        port map (overflow, gpioi.din(3));

 end generate;
```

Εικόνα 4.13: Σύνδεση διακοπών με GPIO pad

Στην εικόνα 4.14 φαίνεται η προσομοίωση του κυκλώματος. Με ρολόι περιόδου 100ps και ένα σήμα που ταλαντώνει με 10ps στέλνεται η εντολή:

wr=1,reset=0,oscillator_control=00000,sampling_time_control=3,signal control=00001

Το ‘wr’ “κάνει” ένα παλμό ώστε να ενημερωθούν οι καταχωρητές. Το ‘sample rate’ είναι το χρονικό παράθυρο, το οποίο σε αυτή την περίπτωση είναι (2^3)*100ps=800ps. Το ‘count enable’ είναι το διάστημα που ο μετρητής μετράει τις θετικές κορυφές του σήματος ‘x’. Το ‘start’ δείχνει ότι έχει ενημερωθεί ο καταχωρητής ‘signal control’. Το ‘reset out’ είναι το σήμα επανεκκίνησης του μετρητή. Τα x0,y0,z0 είναι τα σημεία του ταλαντωτή δακτυλίου. Το ‘overflow’ είναι το σήμα υπερχείλισης και το ‘results’ ο καταχωρητής με το αποτέλεσμα του μετρητή. Το ‘mux_out’ χρησιμοποιείται για την αποσφαλμάτωση.

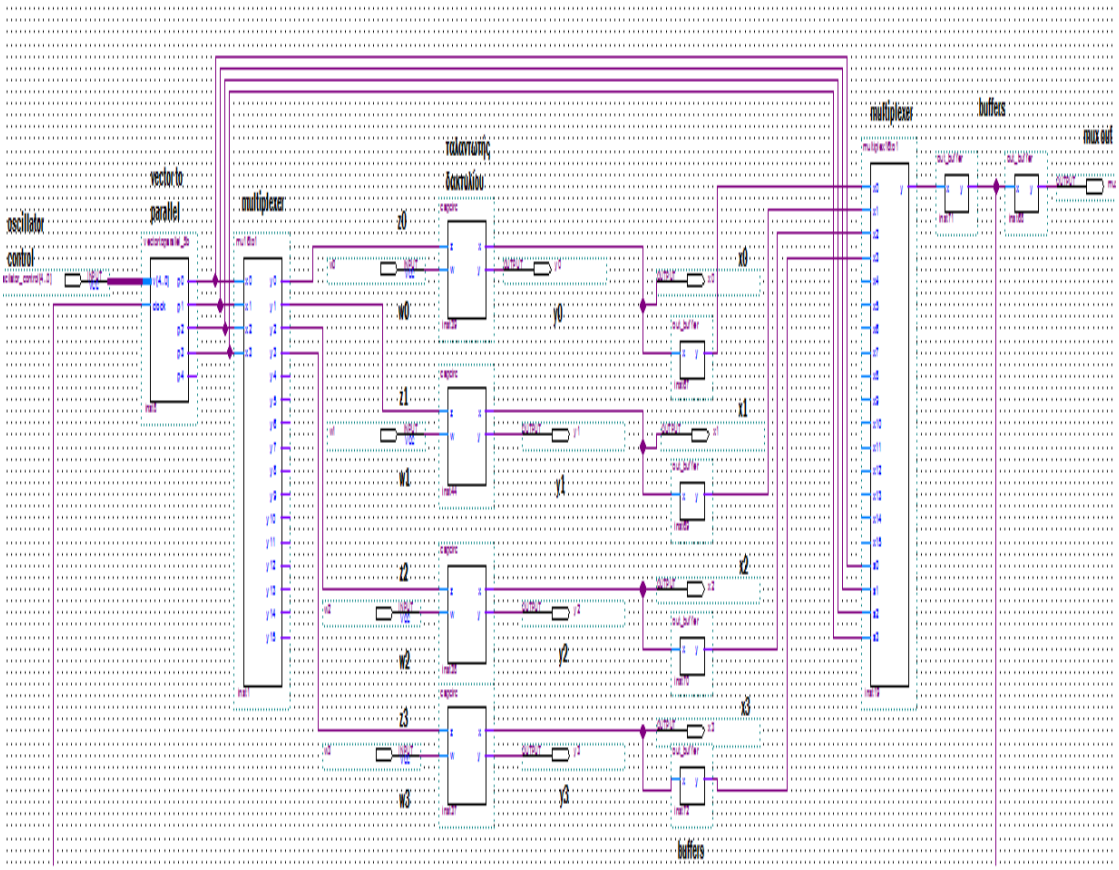


Εικόνα 4.14: Προσομοίωση κυκλώματος δειγματοληψίας

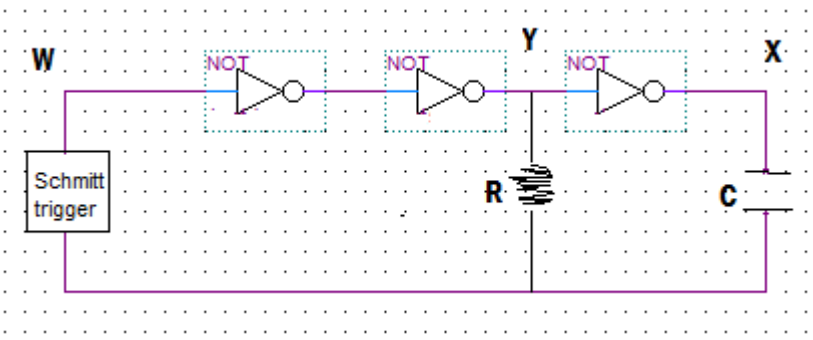
Τα σήμα που έχουν μπροστά το “/my_play/b2v-inst11” είναι τα σήματα του κυκλώματος “catching pulse”. Αρχικά, το κύκλωμα είναι στην αρχική κατάσταση (s5) και όταν το ‘start’ πάρει την τιμή ‘1’, αλλάζει κατάσταση (s1). Στη συνέχεια αλλάζει κατάσταση, όταν το σήμα ‘count enable’ αλλάζει τιμή. Όταν το ‘count enable’ πάει από ‘1’ σε ‘0’ δύο φορές, ενεργοποιείται το σήμα ‘ready’. Σε αυτή την προσομοίωση φαίνεται ο λόγος που χρειάζεται να γίνει αυτό. Την στιγμή που πρέπει να ξεκινήσει η μέτρηση το ‘sample rate’ έχει την τιμή ‘1’ και έτσι ο μετρητής θα μετρήσει μέχρι το ‘sample rate’ να γίνει ‘0’. Αυτή η τιμή του μετρητή είναι λάθος γιατί δεν έχει μετρηθεί όλο το χρονικό παράθυρο. Το τελικό αποτέλεσμα του μετρητή είναι: 70. Το κύκλωμα έχει υπολογίσει την περίοδο (8*100ps)/70=11.42ps αντί 10ps. Αυτό οφείλεται στο πολύ μικρό χρονικό παράθυρο που έχει επιλεχτεί.

4.3.2.3 Κύκλωμα ταλαντωτών

Το κύκλωμα ταλαντωτών περιλαμβάνει τους ταλαντωτές δακτυλίου, τις διεπαφές για την σύνδεση με τους αισθητήρες, το κύκλωμα επιλογής ταλαντωτή και την αποστολή της αναγνωρισμένης παρυφής από τον ταλαντωτή στον μετρητή. Όταν σταλεί ένα σήμα εγγραφής, επιλέγεται και ο ταλαντωτής από τον οποίο θα ληφθεί η μέτρηση συχνότητας. Αυτή η επιλογή γίνεται με τον καταχωρητή ‘oscillator control’. Το κύκλωμα ‘vector to parallel’ στέλνει τα μπιτ του καταχωρητή στους πολυπλέκτες. Ο 1ος πολυπλέκτης (αριστερά στο σχήμα 4.25) ενεργοποιεί τον αντίστοιχο ταλαντωτή ανάλογα με την τιμή των μπιτ που θα λάβει. Οι υπόλοιποι ταλαντωτές μένουν ανενεργοί. Αν έχει συνδεθεί μια χωρητικότητα στο κύκλωμα του ταλαντωτή, ο ταλαντωτής δακτυλίου θα αρχίσει να ταλαντώνει. Αυτή η ταλάντωση δεν σταματάει, παρά μόνο αν γίνει αλλαγή ταλαντωτή ή αν υπάρξει κάποιο εξωτερικό πρόβλημα. Ο 2ος πολυπλέκτης κοιτάζει συνεχώς την έξοδο ‘x’ του ταλαντωτή. Ο ταλαντωτής μετά τη σύνδεση της αντίστασης και του πυκνωτή φαίνεται στο σχήμα 4.26. Η τιμή του σημείου ΄x’ στέλνεται στον μετρητή ο οποίος αυξάνει κατά ένα, όταν δει μια θετική παρυφή αυτής της τιμής.



Σχήμα 4.25: Κύκλωμα ταλαντωτή



Σχήμα 4.26: Κύκλωμα ταλαντωτή δακτυλίου μετά την σύνδεση πυκνωτή και αντίστασης

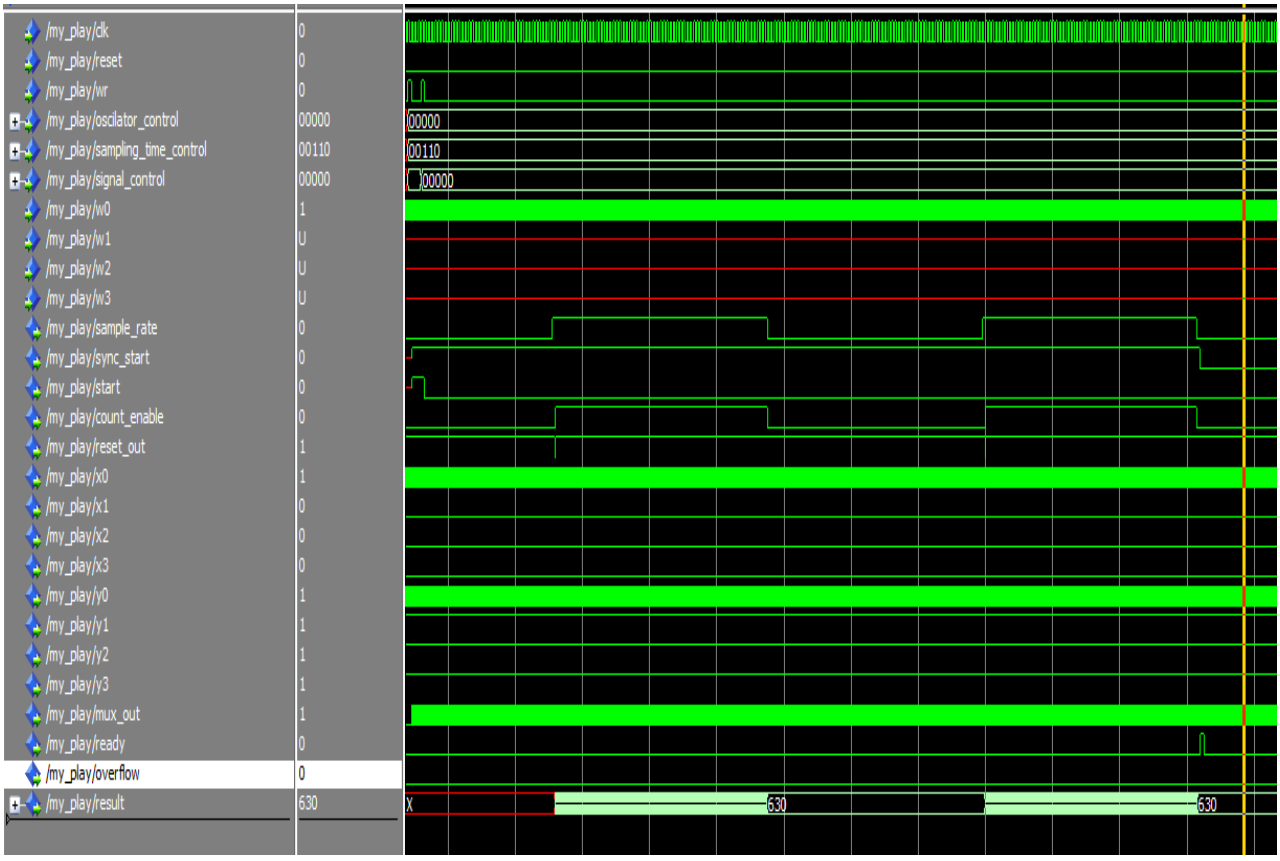
Ο Schmitt trigger είναι συνδεδεμένος στο σημείο 'w' εξωτερικά της συσκευής FPGA (χρησιμοποιείται ένα ΟΚ 74HCHCxxx), καθώς η συγκεκριμένη συσκευή δεν διαθέτει εσωτερικά.

4.3.2.4 Προσομοιώσεις

Στην εικόνα 4.15 φαίνεται η προσομοίωση του κυκλώματος. Το ρολόι (clk) έχει οριστεί να έχει περίοδο 100ps και υπάρχει ένα σήμα που ταλαντώνει με περίοδο 10ps. Αρχικά, στέλνεται η εντολή:

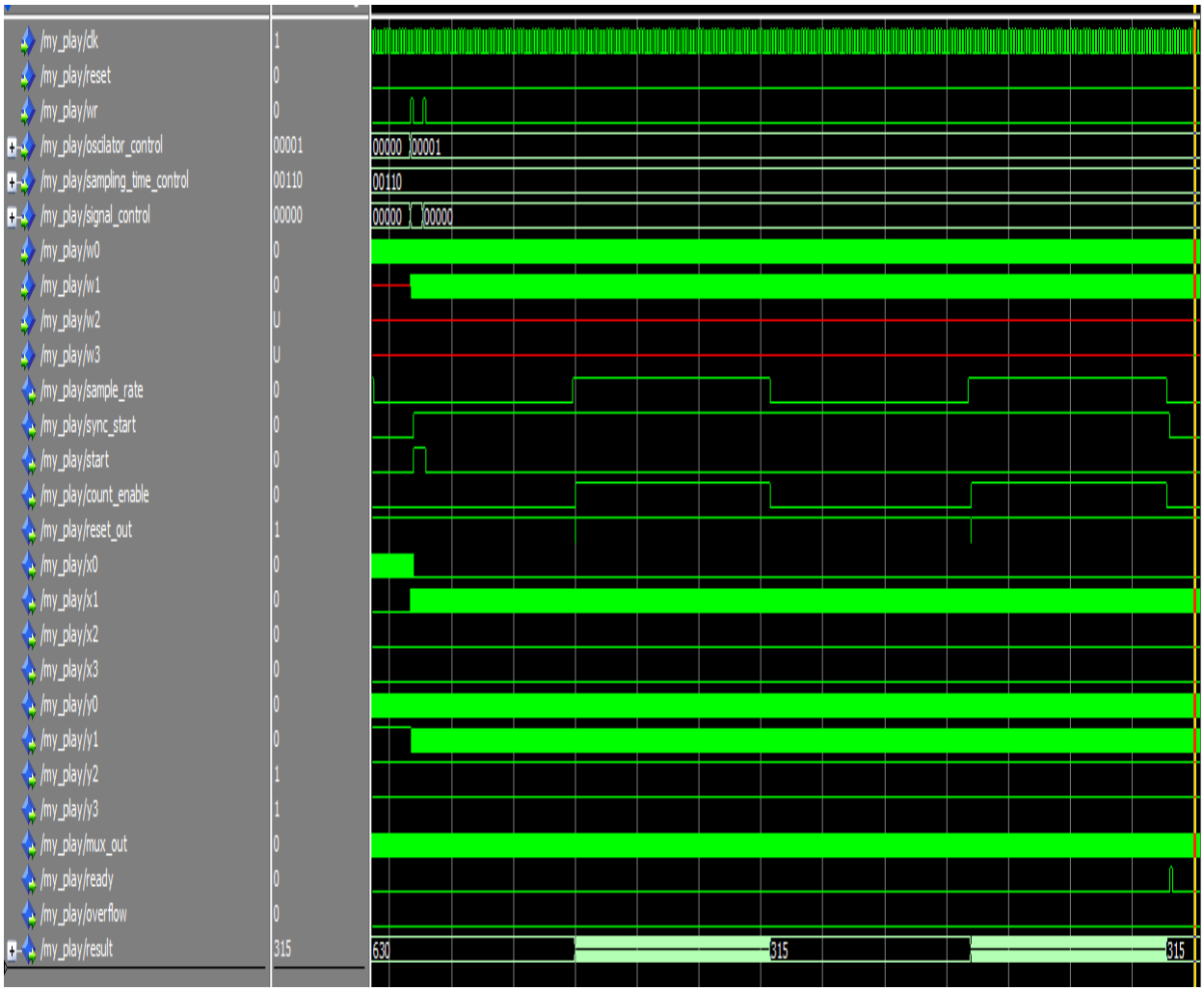
wr=1,reset=0,oscillator_control=00000,sampling_time_control=6,signal control=00001

Στη συνέχεια στέλνεται άλλη μια εντολή η οποία μηδενίζει το 'signal control'. Στην εικόνα φαίνεται το χρονικό παράθυρο το οποίο είναι (2^6)*100ps=6400ps. Το αποτέλεσμα του μετρητή είναι 630, έτσι η περίοδος του σήματος που ταλαντώνει υπολογιζεται: 6400/630=10.1587ps



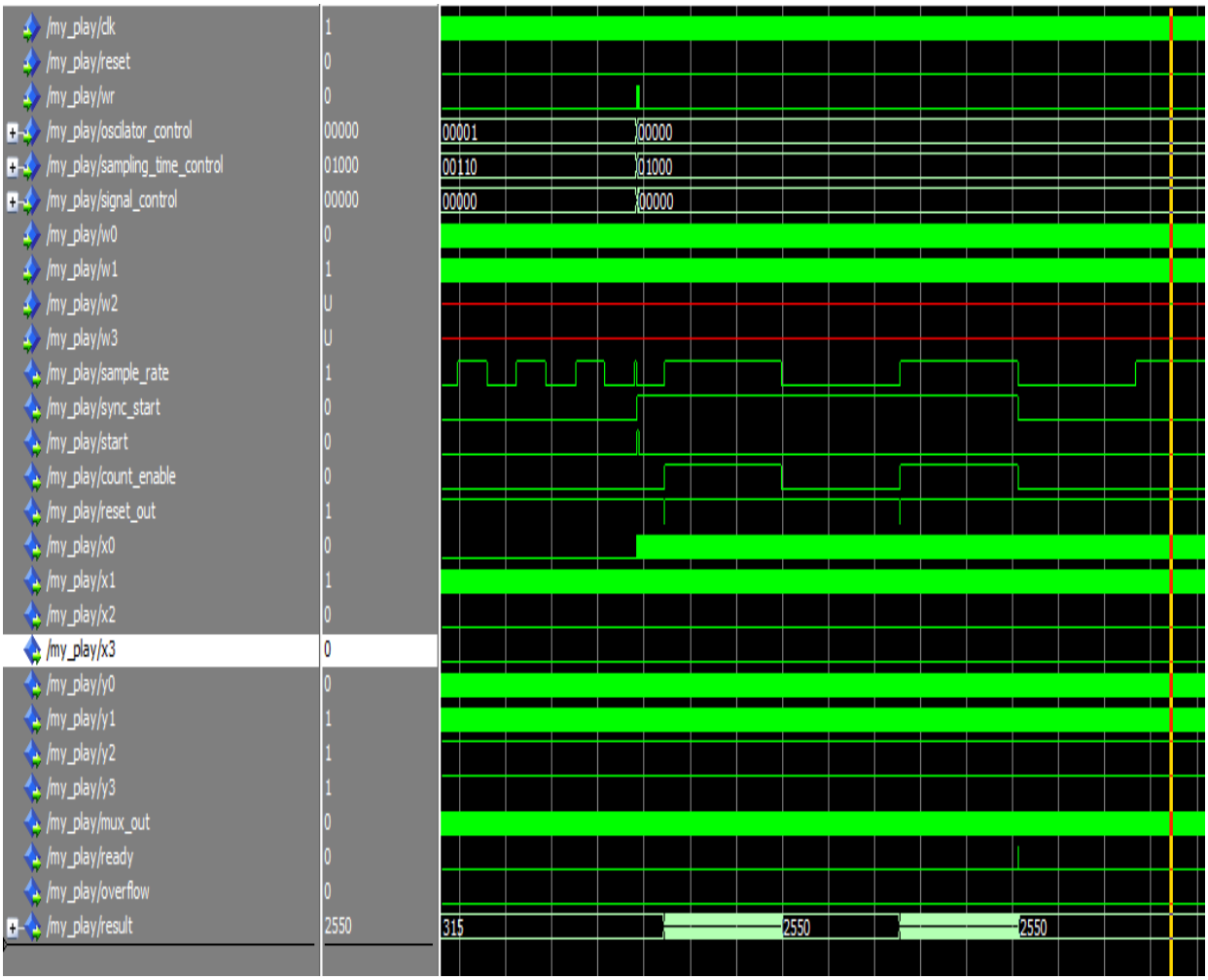
Εικόνα 4.15: Προσομοίωση με ρόλοι 100ps, ταλαντωτή 10ps και παράμετρο χρονικού παραθύρου=6

Στην εικόνα 4.16 φαίνεται η προσομοίωση ενός σήματος που ταλαντώνει με 20ps. Το αποτέλεσμα του μετρητή είναι 315, έτσι η περίοδος που έχει υπολογιστεί είναι 6400/315=20.317ps. Ουσιαστικά, η τιμή του μετρητή είναι η μισή σε σχέση με την τιμή των 10ps του προηγούμενου παραδείγματος.



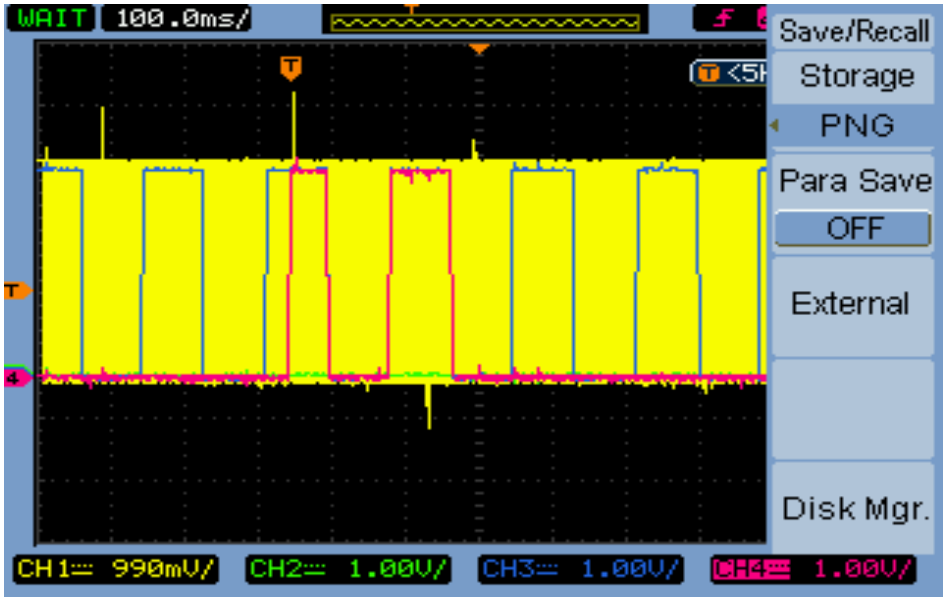
Εικόνα 4.16: Προσομοίωση με ρόλοι 100ps, ταλαντωτή 20ps και παράμετρο χρονικού παραθύρου=6

Στην εικόνα 4.17 ταλαντώνει ένα σήμα με περίοδο 10ps αλλά έχει αλλάξει το χρονικό παράθυρο της μέτρησης. Η παράμετρος είναι τώρα 8. Έτσι το χρονικό παράθυρο τετραπλασιάζεται, όπως φαίνεται και στην εικόνα (“sample rate”). Η τιμή του μετρητή είναι 2550, έτσι η περίοδος υπολογίζεται ως 256*100ps/2550=10.039ps. Βλέπουμε ότι έχει βελτιωθεί η ακρίβεια. Όσο μεγαλώνει το χρονικό παράθυρο τόσο καλύτερη η ακρίβεια της μέτρησης.



Εικόνα 4.17: Προσομοίωση με ρόλοι 100ps, ταλαντωτή 10ps και παράμετρο χρονικού παραθύρου=8

Στην εικόνα 4.18 φαίνεται μια μέτρηση, όπως φαίνεται από τον παλμογράφο, χρησιμοποιώντας τα σήματα αποσφαλμάτωσης. Το κίτρινο σήμα(ch1) είναι το σήμα το οποίο ταλαντώνει με τον ταλαντωτή δακτυλίου. Το μπλε (ch3) είναι το “sample rate” και το κόκκινο (ch4) είναι το “count enable”.

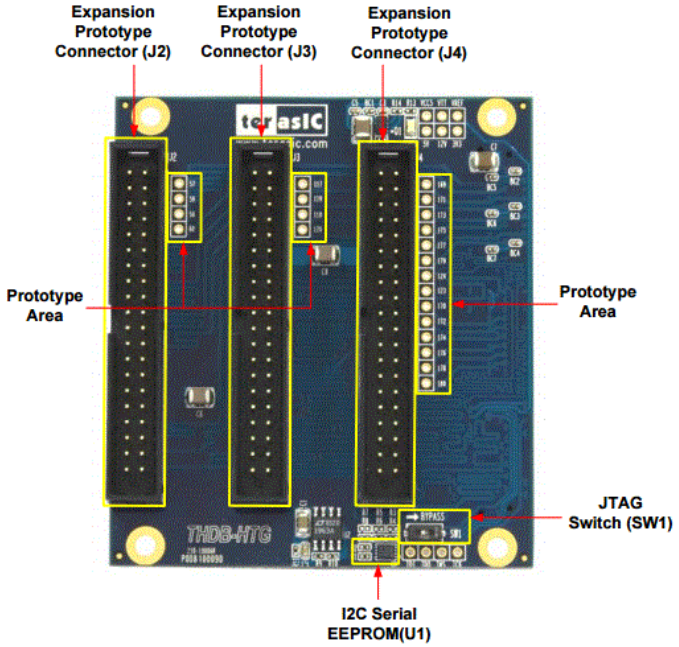


Εικόνα 4.18: Εμφάνιση μέτρησης στον παλμογράφο

4.4 Εξωτερικό Κύκλωμα

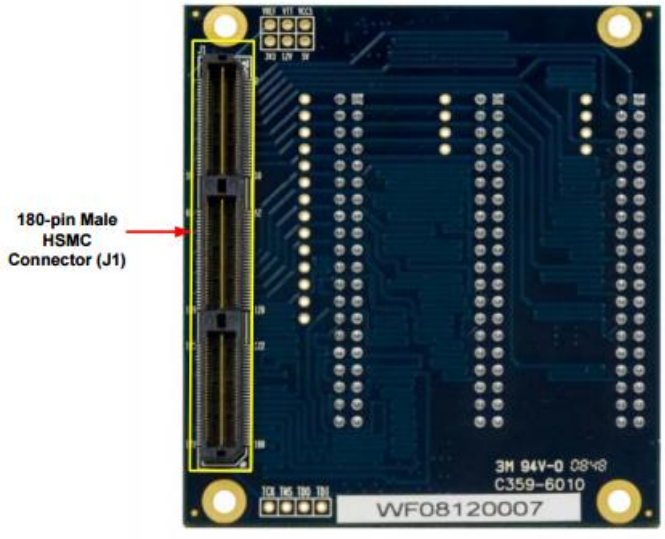
4.4.1 Κύκλωμα THDB-HTG

Το εξωτερικό κύκλωμα στο οποίο συνδέονται οι αντιστάσεις και οι αισθητήρες χωρητικοτήτων έχει συνδεθεί με το υπόλοιπο σύστημα χρησιμοποιώντας την θύρα HSMC της συσκευής. Για αυτόν τον σκοπό χρησιμοποιήθηκε ένα βύσμα HSMC που είναι συμβατό με τη συσκευή, το THDB-HTG. Το βύσμα 'THDB-HTG' είναι σχεδιασμένο να μετατρέπει τους ακροδέκτες εισόδου/εξόδου ενός HSMC σε τρεις ομάδες 40 ακροδεκτών GPIO. Στην εικόνα 4.19 φαίνεται η πάνω πλευρά αυτού του κυκλώματος. Στην εικόνα φαίνονται οι 3 ομάδες GPIO θυρών (J2, J3, J4) . Η κάθε ομάδα διαθέτει 36 ακροδέκτες εισόδου/εξόδου και ακροδέκτες παροχής τροφοδοσίας 3.3/5 volts [43].



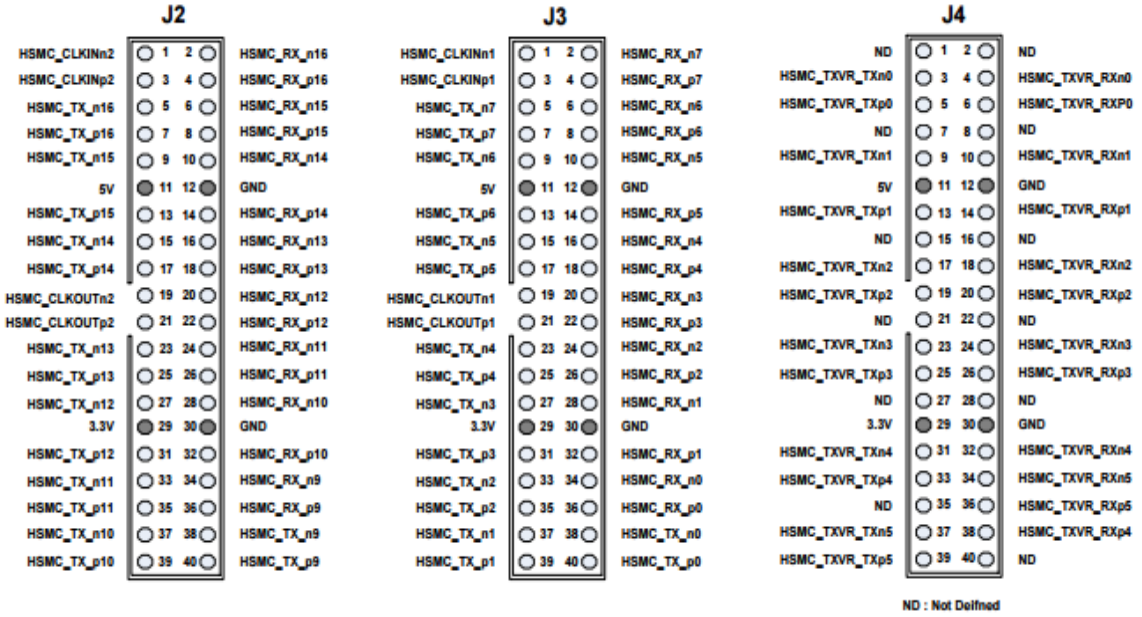
Εικόνα 4.19: Πάνω πλευρά της συσκευής THDB-HTG [43]

Στην εικόνα 4.20 φαίνεται η πίσω πλευρά του κυκλώματος. Το βύσμα J1 συνδέεται με τη θύρα HSMC της συσκευής DE2-115.



Εικόνα 4.20: Πίσω πλευρά της συσκευής THDB-HTG [43]

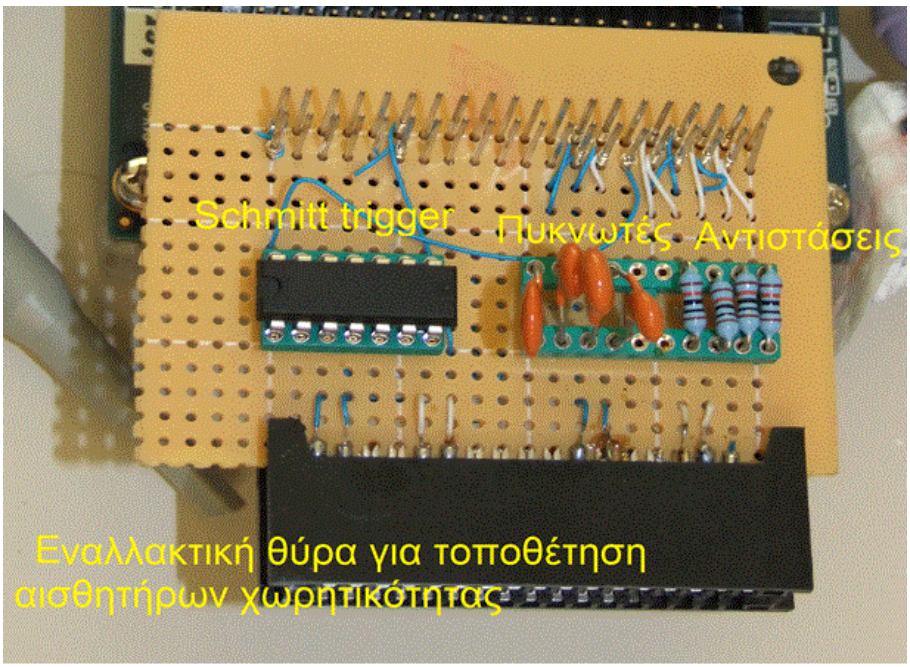
Με την χρήση αυτού του κυκλώματος γίνεται εύκολη η σύνδεση των σημάτων του κυκλώματος μετρήσεων με το εξωτερικό κύκλωμα. Κάθε ακροδέκτης HSMC της συσκευής DE2-115 μπορεί να αντιστοιχηθεί με ένα ακροδέκτη του κυκλώματος THDB-HTG. Η αντιστοίχιση των σημάτων φαίνεται στην εικόνα 4.21 [43].



Εικόνα 4.21: Πίνακας αντιστοίχισης HSMC ακροδεκτών με τις ομάδες J2,J3,J4 της συσκευής THDB-HTG [43]

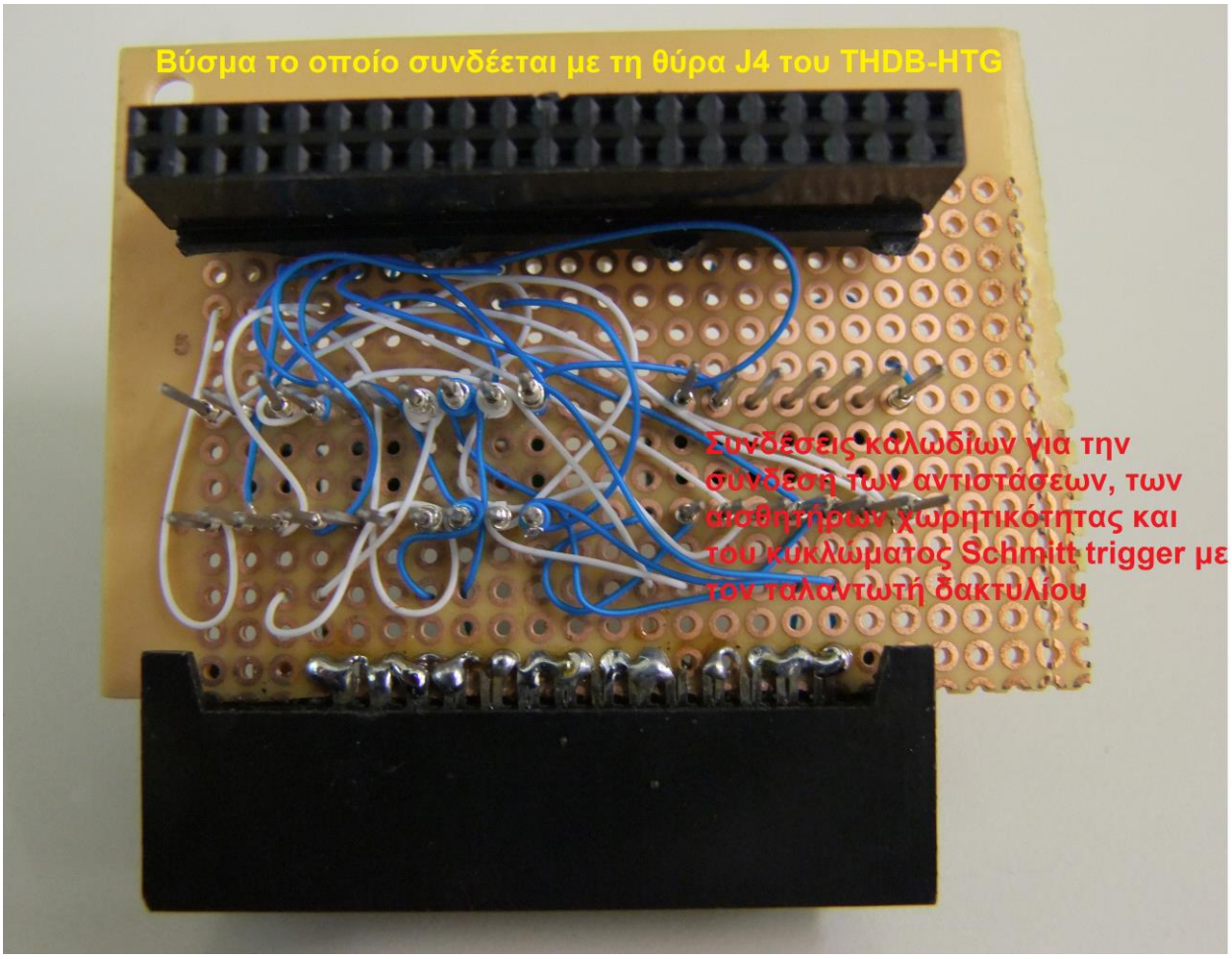
4.4.2 Κύκλωμα αισθητήρων

Στο κύκλωμα αισθητήρων έχουν συνδεθεί τα κομμάτια του ταλαντωτή δακτυλίου που λείπουν. Σε αυτό το κύκλωμα οδηγούνται τα σήματα x_n,y_n και w_n του κυκλώματος ταλαντωτών. Επίσης, αυτό το κύκλωμα περιλαμβάνει και τα κυκλώματα Schmitt Trigger, τα οποία βελτιώνουν την σταθερότητα των ταλαντωτών δακτυλίου και την αξιοπιστία-ακρίβεια των μετρήσεων. Στην εικόνα 4.22 φαίνεται το πάνω μέρος του κυκλώματος αισθητήρων.



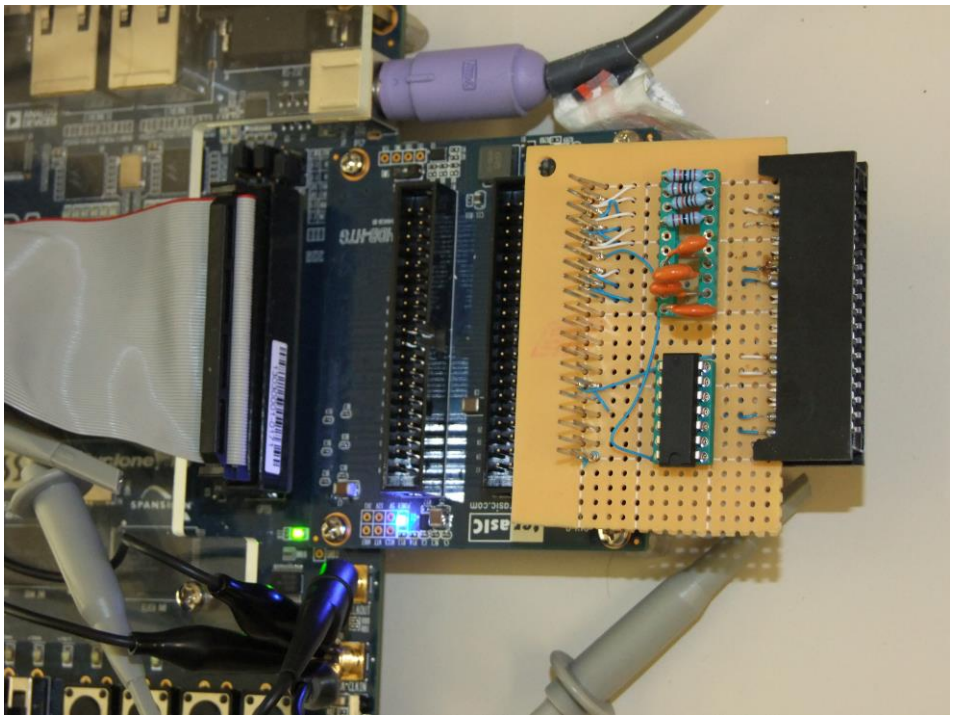
Εικόνα 4.22: Πάνω όψη κυκλώματος αισθητήρων

Υπάρχουν δύο τρόποι σύνδεσης αισθητήρων χωρητικότητας. Ο 1ος είναι με την απευθείας ή μέσω καλωδίων σύνδεση στους ακροδέκτες της βάσης δίπλα από τις αντιστάσεις και ο 2ος χρησιμοποιώντας την θύρα που βρίσκεται στο μπροστινό μέρος της κατασκευής. Στο σχήμα 4.26 φαίνεται ο τρόπος συνδεσμολογίας για έναν ταλαντωτή. Η αντίσταση, ο Schmitt trigger και η χωρητικότητα βρίσκονται στο εξωτερικό κύκλωμα, τα υπόλοιπα στοιχεία βρίσκονται στο κύκλωμα ταλαντωτών εσωτερικά του FPGA.



Εικόνα 4.23: Πίσω όψη κυκλώματος αισθητήρων

Στην εικόνα 4.23 φαίνεται η πίσω όψη του κυκλώματος των αισθητήρων. Φαίνεται το βύσμα με το οποίο συνδέεται το κύκλωμα με την μονάδα THDB-HTG, στη θύρα J4. Η σύνδεση των στοιχείων του ταλαντωτή έχει γίνει με απλά καλώδια, όπως φαίνεται στο σχήμα. Το κύκλωμα περιλαμβάνει 4 ταλαντωτές, δηλαδή μπορούν να μετρηθούν μέχρι και 4 αισθητήρες ταυτόχρονα.



Εικόνα 4.24: Σύνδεση κυκλώματος αισθητήρων με το υπόλοιπο σύστημα χρησιμοποιώντας την μονάδα THDB-HTG

Στην εικόνα 4.24 φαίνεται η σύνδεση της μονάδας THDB-HTG με το εξωτερικό κύκλωμα. Το τμήμα J3 της μονάδας THDB-HTG χρησιμοποιήθηκε για τα σήματα αποσφαλμάτωσης.

4.5 Απαιτήσεις συστήματος

Παρακάτω φαίνονται τα συνολικά αποτελέσματα της σύνθεσης του συστήματος από το πρόγραμμα QUARTUS.

Κύκλωμα διεπαφής

Total logic elements 138 / 21,280 (< 1 %)
Total combinational functions 135 / 21,280 (< 1 %)
Dedicated logic registers 67 / 21,280 (< 1 %)
Total pins 54 / 167 (32 %)

Συνολικό κύκλωμα με επεξεργαστή

Total logic elements 25,218 / 114,480 (22 %)
Total combinational functions 22,264 / 114,480 (19 %)
Dedicated logic registers 9,931 / 114,480 (9 %)
Total pins 186 / 529 (35 %)
Total memory bits 424,416 / 3,981,312 (11 %)
Embedded Multiplier 9-bit elements 8 / 532 (2 %)
Total PLLs 2 / 4 (50 %)

Το αρχείο το οποίο δημιουργείται από την σύνθεση στο QUARTUS (αρχείο .pof) έχει αποθηκευτεί στη συσκευή του αναπτυξιακού EPCS64 για Active Serial programming ώστε όταν ανοίγει το αναπτυξιακό, η συσκευή FPGA να φορτώνεται αυτόματα με το υλοποιημένο κύκλωμα. Η εικόνα Linux που δημιουργήθηκε και θα περιγραφεί στο κεφάλαιο 5 χρειάζεται 7.5 MB μαζί με τα προγράμματα μέτρησης και κάποιες εικόνες και έχει αποθηκευτεί στη μνήμη FLASH του αναπτυξιακού, ώστε να φορτώνεται κατά την εκκίνηση του συστήματος (μετά την φόρτωση του κυκλώματος στο FPGA). Η μνήμη FLASH του αναπτυξιακού στην οποία αποθηκεύεται η εικόνα είναι 8MB.

5 . ΛΟΓΙΣΜΙΚΟ ΜΕΡΟΣ ΥΛΟΠΟΙΗΣΗΣ

5.1 Εισαγωγικά

Σε αυτό το κεφάλαιο παρουσιάζεται το λογισμικό μέρος της υλοποίησης του έξυπνου συστήματος μέτρησης αισθητήρων. Στο προηγούμενο κεφάλαιο έγινε η περιγραφή του υλικού μέρους της υλοποίησης. Για την αξιοποίηση του υλικού μέρους χρειάστηκε να υλοποιηθεί ένα λογισμικό που να εκμεταλλεύεται τις δυνατότητες των περιφερειακών που συνδέθηκαν με το σύστημα.

Επιλέχθηκε και ενσωματώθηκε στο σύστημα μία έκδοση του λειτουργικού συστήματος Linux. Η χρήση του λειτουργικού συστήματος Linux σε ενσωματωμένες συσκευές είναι δημοφιλής, για παράδειγμα, σε καταναλωτικές συσκευές (gadgets), σε δρομολογητές, σε εφαρμογές σε αυτοκίνητα κ.α. Εξαιτίας της σπονδυλωτής δομής (modular) του Linux είναι εύκολη η αφαίρεση κομματιών από το λειτουργικό σύστημα τα οποία δεν χρειάζονται στα ενσωματωμένα συστήματα. Το Linux μπορεί να τρέξει σε ένα ενσωματωμένο σύστημα χρησιμοποιώντας έναν τελείως “γυμνό” πυρήνα (kernel) που κάνει μόνο τις βασικές λειτουργίες. Αυτή η ιδιότητα είναι πολύ χρήσιμη σε συστήματα που έχουν μικρές επεξεργαστικές δυνατότητες και μικρή χωρητικότητα μνήμης. Ένα από τα σημαντικότερα πλεονεκτήματα του Linux είναι ότι είναι ένα πλήρως λειτουργικό σύστημα με υποστήριξη υπηρεσιών διαδικτύου. Επίσης, σε ένα σύστημα Linux μπορούν να προστεθούν και να αφαιρεθούν τμήματα (modules) από τον πυρήνα κατά τη λειτουργία του, κάτι που κάνει το σύστημα ευέλικτο. Άλλα πλεονεκτήματα είναι ότι ο κώδικας του Linux είναι διαθέσιμος σε όλους και δωρεάν, είναι φορητός σε όλους τους επεξεργαστές, είναι τροποποιήσιμος και αξιόπιστος.

Το λειτουργικό σύστημα Linux υποστηρίζεται από την Aeroflex Gaisler και τον επεξεργαστή LEON3. Έχει δημιουργηθεί ένα πρόγραμμα για την εύκολη ενσωμάτωση του λειτουργικού συστήματος Linux σε ένα σύστημα LEON3, το LINUXBUILD. Το LINUXBUILD συνδυάζει το περιβάλλον BUILDROOT με διάφορα προγράμματα της Aeroflex Gaisler που ήδη υπάρχουν, όπως το ‘MKLINUXIMG’ και το ‘MKPROM’. Το Buildroot είναι ένα σύνολο από ‘makefiles και ‘patches’ τα οποία απλοποιούν και αυτοματοποιούν την διαδικασία κτισίματος ενός ολοκληρωμένου και λειτουργικού περιβάλλοντος Linux, για ένα ενσωματωμένο σύστημα. Το ‘MKLINUXIMG’ συνδέει τον πυρήνα του Linux με την κεντρική μνήμη του εκάστοτε συστήματος και η εφαρμογή ‘MKPROM’ χρησιμοποιείται για τη δημιουργία FLASH/PROM εικόνων εκκίνησης (bootable images).

Στη συνέχεια υλοποιήθηκαν προγράμματα λογισμικού σε γλώσσα C τα οποία χρησιμοποιούνται για τις μετρήσεις του κυκλώματος της διεπαφής των αισθητήρων. Αυτά τα προγράμματα τρέχουν στο ‘user-space’ επίπεδο του Linux και χρησιμοποιούν διάφορες λειτουργίες και εφαρμογές του λειτουργικού συστήματος που υλοποιήθηκε.

Σε αυτό το κεφάλαιο αρχικά, θα παρουσιαστεί το λειτουργικό σύστημα που χρησιμοποιήθηκε και ο τρόπος ενσωμάτωσής του με το υφιστάμενο υλικό που περιγράφηκε στο κεφάλαιο 4. Στην συνέχεια, θα παρουσιαστούν τα προγράμματα που υλοποιήθηκαν για τις μετρήσεις των αισθητήρων και ο τρόπος εγκατάστασης και αποσφαλμάτωσής τους. Τέλος, θα παρουσιαστούν ο τρόπος αλληλεπίδρασης των προγραμμάτων λογισμικού με το υλικό μέρος της υλοποίησης και θα εξηγηθούν κομμάτια του κώδικα.

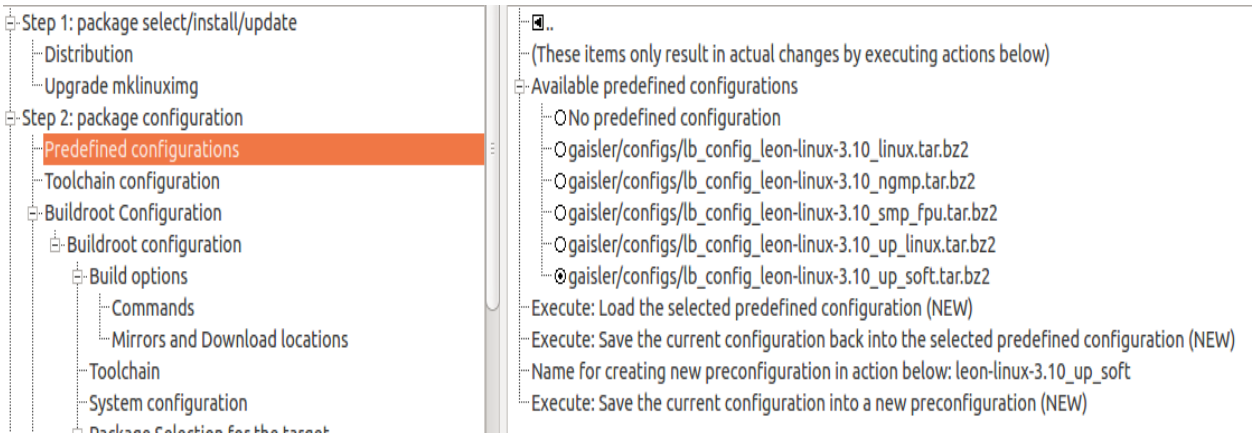
5.2 Δημιουργία εικόνας Linux με LINUXBUILD

Σημείωση: Στα επόμενα 'host' σύστημα θεωρείται το σύστημα στο οποίο γίνεται η αποσφαλμάτωση, δηλαδή η συσκευή DE2-115 και 'client' σύστημα είναι ο υπολογιστής που έχει τα προγράμματα για την υλοποίηση και αποσφαλμάτωση.

Για την δημιουργία της εικόνας Linux χρησιμοποιήθηκε το πρόγραμμα LINUXBUILD της Aeroflex Gaisler. Με το LINUXBUILD επιλέγονται τα προγράμματα που θα τρέχουν στην εικόνα Linux και οι οδηγητές που χρειάζονται για την σύνδεση με το υλικό. Κατά την εγκατάσταση του LINUXBUILD υπάρχουν κάποιες προεπιλεγμένες ρυθμίσεις για την υποστήριξη του επεξεργαστή LEON3.

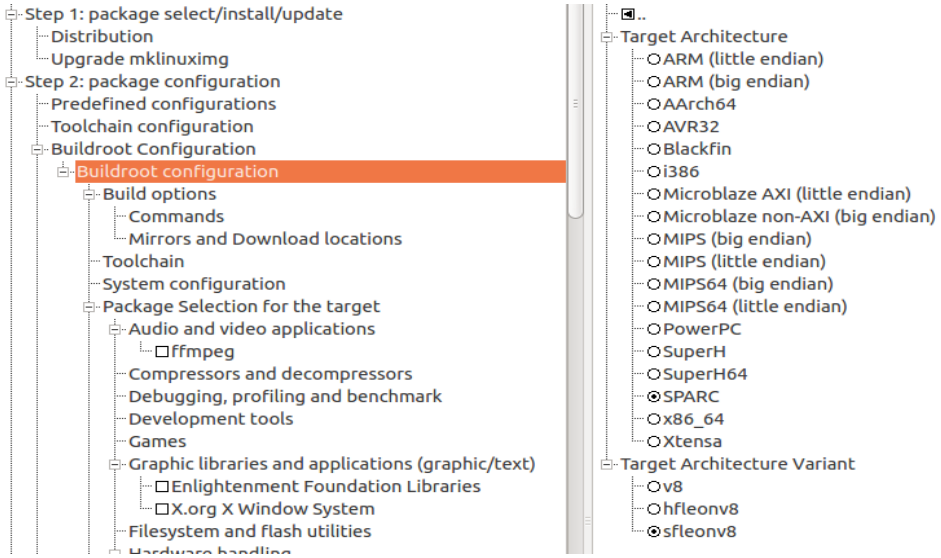
Στη συνέχεια παρουσιάζονται τα βασικά βήματα που ακολουθήθηκαν για την δημιουργία της Linux εικόνας και οι βασικές παράμετροι που χρησιμοποιήθηκαν στο πρόγραμμα LINUXBUILD. Το LINUXBUILD τρέχει στον 'client' και το PC 'client' πρέπει να έχει λειτουργικό σύστημα Linux για να τρέξει το LINUXBUILD. Οδηγίες για την εγκατάσταση του Linuxbuild υπάρχουν στο παράρτημα.

Στη εικόνα 5.1 φαίνεται η προεπιλεγμένη ρύθμιση που χρησιμοποιήθηκε για το σύστημα της διπλωματικής. Όταν επιλεχτεί αυτή η ρύθμιση φορτώνονται αυτόματα όλες οι επιλογές για την ενσωμάτωση της εικόνας σε ένα LEON3 σύστημα.

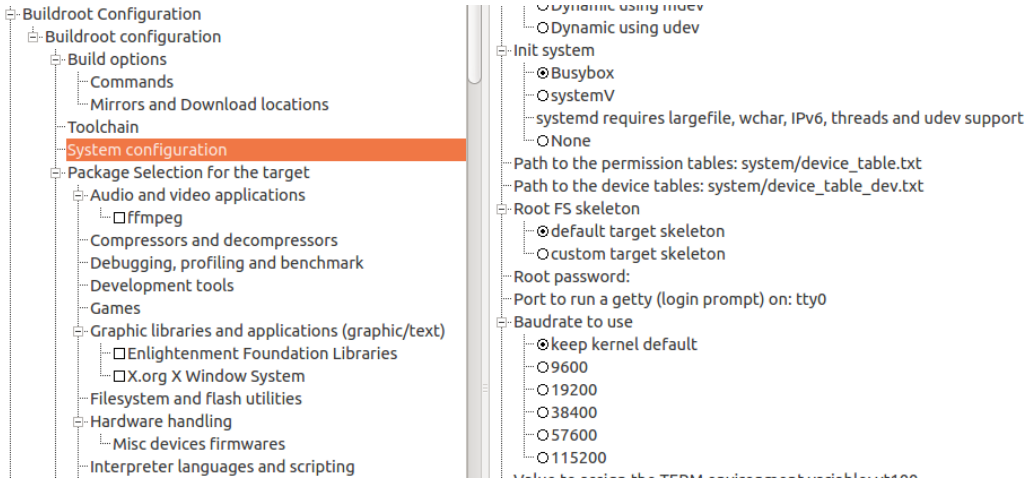


Εικόνα 5.1: Με την επιλογή 'gaisler/configs/lb_config_leon-linux-3.10_up_soft_tar.bz2', η εικόνα Linux που δημιουργείται είναι συμβατή με ένα επεξεργαστή LEON3.

Στην εικόνα 5.2 φαίνεται το σημείο στο οποίο επιλέγεται η αρχιτεκτονική του επεξεργαστή και ο τύπος('sfleon' ή 'hfleon' για το αν χρησιμοποιείτε ή όχι μονάδα FPU).

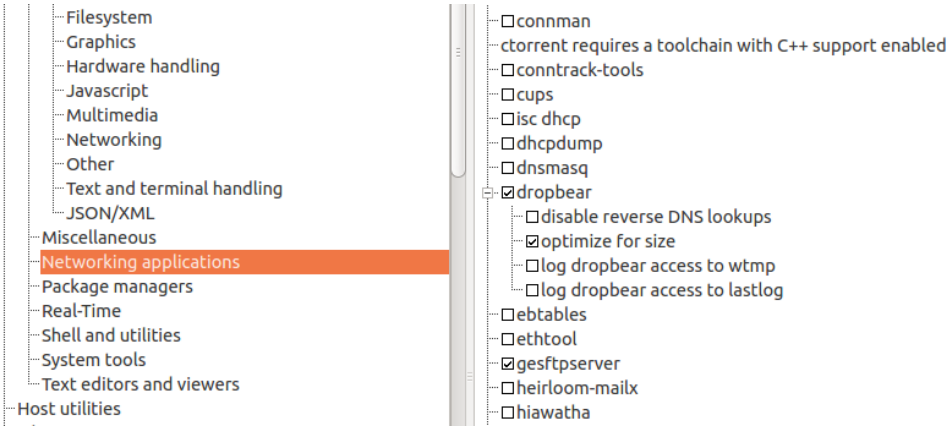


Εικόνα 5.2: Επιλογή αρχιτεκτονικής στο LINUXBUILD



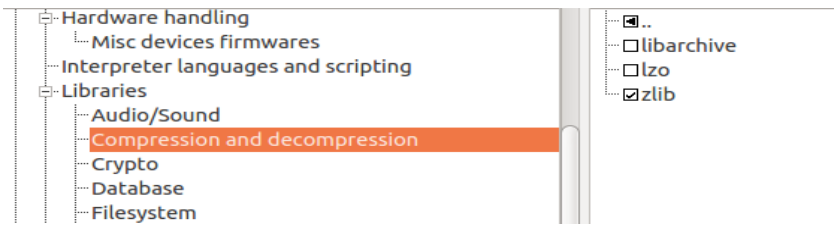
Εικόνα 5.3: Βασικές ρυθμίσεις συστήματος

Στην εικόνα 5.3 φαίνεται η επιλογή των βασικών παραμέτρων του συστήματος. Έχει επιλεχθεί το πρόγραμμα ‘busybox’. Το πρόγραμμα αυτό περιλαμβάνει τις βασικές εντολές του Linux και η απόδοση του είναι βελτιστοποιημένη έτσι ώστε να χρησιμοποιείται σε ενσωματωμένα συστήματα. Σε αυτό το σημείο μπορεί να επιλεχθεί από το πεδίο “Root password” και ο κωδικός για την είσοδο σαν “υπερ-χρήστης” (super-user ή root login). Για τη δημιουργία του κωδικού χρειάζεται να είναι εγκατεστημένο στο σύστημα που δημιουργείται η εικόνα (client) το ‘mkpasswd’. Για την λειτουργία της οθόνης ως τερματικού η επιλογή “port to run a getty” πρέπει να είναι στο ‘tty0’.



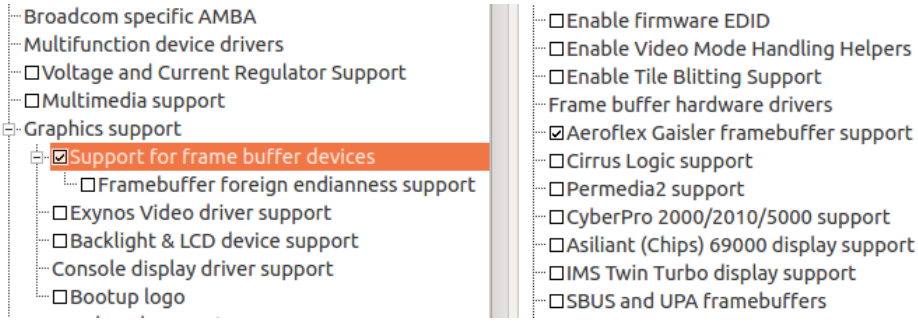
Εικόνα 5.4: Επιλογή ‘dropbear’ και ‘sftp server’ στο LINUXBUILD

Στην εικόνα 5.4 φαίνεται η επιλογή του προγράμματος ‘dropbear’ και ενός ‘sftp server’. Αυτές οι επιλογές προστέθηκαν ώστε το σύστημα να μπορεί να δημιουργήσει ασφαλείς συνδέσεις SSH, για τη μεταφορά δεδομένων από και προς το σύστημα, αλλά και για την αποσφαλμάτωση. Το dropbear είναι ένα πρόγραμμα το οποίο παρέχει έναν συμβατό με το SSH πρωτόκολλο εξυπηρετητή (server) και πελάτη (client). Για την λειτουργία του πρωτοκόλλου SFTP χρειάζεται ο ‘gesftpserver’, καθώς το ‘dropbear’ δεν υλοποιεί το SFTP πρωτόκολλο.



Εικόνα 5.5: Επιλογή βιβλιοθήκης συμπίεσης ‘zlib’

Για την συμπίεση των αρχείων και των εικόνων χρησιμοποιήθηκε η βιβλιοθήκη ‘zlib’. Στο σύστημα έχουν προ-αποθηκευτεί κάποιες εικόνες σε μορφή .raw, οι οποίες σε κανονική μορφή πιάνουν αρκετό χώρο στο σύστημα. Για αυτό τον λόγο έχουν συμπιεστεί σε μορφή ‘gzip’.



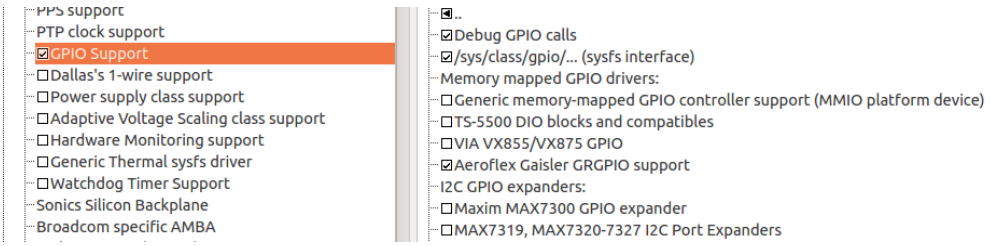
Εικόνα 5.6: Επιλογή οδηγητή για την ενδιάμεση μνήμη εικόνας

Στην συνέχεια επιλέχτηκαν οι οδηγοί του συστήματος για την υποστήριξη του υλικού. Στην εικόνα 5.6 φαίνεται η επιλογή για υποστήριξη της ενδιάμεσης μνήμης καρέ της Gaisler Aeroflex. Σε αυτό το σημείο πρέπει να επισημανθεί ότι έγινε μια τροποποίηση του οδηγητή της οθόνης, καθώς οι προεπιλεγμένες επιλογές του οδηγού δεν αντιστοιχούσαν στις παραμέτρους της οθόνης που χρησιμοποιήθηκε. Στο φάκελο του LINUXBUILD ‘\linux\linux-src\drivers\video’ βρίσκεται ο οδηγητής της οθόνης ‘grvga’. Εκεί πρέπει να προστεθούν τα χαρακτηριστικά της οθόνης που χρησιμοποιείται, όπως φαίνεται στην εικόνα 5.7. Τα χαρακτηριστικά της οθόνης έχουν παρουσιαστεί στον πίνακα 3 (βλ. κεφάλαιο 4).

```
static const struct fb_videomode grvga_modedb[] = {
    {
     /* 640x480 @ 60 Hz */
     NULL, 60, 800, 480, 30303, 16, 210, 10, 22, 30, 13,
     0, FB_VMODE_NONINTERLACED
    }, {
     /* 800x480 @ 60 Hz */
     NULL, 60, 800, 480, 30303, 16, 210, 10, 22, 30, 13,
     0, FB_VMODE_NONINTERLACED
    }, {
     /* 800x600 @ 72 Hz */
     NULL, 72, 800, 600, 20000, 64, 56, 23, 37, 120, 6,
     0, FB_VMODE_NONINTERLACED
    }, {
     /* 1024x768 @ 60 Hz */
     NULL, 60, 1024, 768, 15385, 160, 24, 29, 3, 136, 6,
     0, FB_VMODE_NONINTERLACED
    }
};
```

Εικόνα 5.7: Τροποποίηση του οδηγητή ‘grvga’

Για την δημιουργία των διακοπών ,όπως ειπώθηκε και στο κεφάλαιο 4, τα σήματα διακοπών έχουν συνδεθεί σε ‘GPIO pads’. Για την ενεργοποίηση τους χρειάζεται ο οδηγητής για τον πυρήνα της Aeroflex Gaisler GRGPIO. Επίσης για την δημιουργία των διακοπών χρησιμοποιείται το ‘user-space’ επίπεδο του λειτουργικού συστήματος Linux και η διεπαφή ‘sysfs’ (sysfs interface). Με αυτή την διεπαφή μπορεί ο χρήστης του συστήματος να ενεργοποιήσει διακοπές που είναι συνδεδεμένες στο GPIO. Μπορεί να καθορίσει την πολικότητα και το επίπεδο ενεργοποίησης της διακοπής, όπως και αν η θύρα GPIO θα είναι θύρα εισόδου ή εξόδου.



Εικόνα 5.8: Υποστήριξη GPIO

Το ‘sysfs interface’ λειτουργεί ως εξής: Για την ενεργοποίηση μιας GPIO θύρας ο χρήστης πρέπει να πάει στο σημείο /sys/class/gpio/ του συστήματος (host). Σε αυτό το σημείο υπάρχει η συσκευή που έχει αναγνωριστεί από το λειτουργικό σύστημα (Εικόνα 5.9 – gpiochip224).

```
# cd /sys/class/gpio
# ls
export       gpiochip224  unexport
#
```

Εικόνα 5.9: GPIO sysfs interface

Σε αυτό το σημείο υπάρχει επίσης η επιλογή ενεργοποίησης (export) και απενεργοποίησης (unexport) θυρών. Ως θύρα GPIO θεωρείται ο ακροδέκτης GPIO. Στη συσκευή της εικόνας 5.9 η ενεργοποίηση της πρώτης θύρας GPIO μπορεί να γίνει με την εντολή *echo 225 > export* (το 224 συμβολίζει τον ακροδέκτη gpio(0)). Στην εικόνα 5.10 φαίνεται ο τρόπος ενεργοποίησης της θύρας-ακροδέκτη. Για την χρησιμοποίηση της θύρας ως διακοπή πρέπει να καθοριστεί η κατεύθυνση (direction) και η ακμή (edge), δηλαδή αν η διακοπή ενεργοποιείται στην ανερχόμενη ή κατερχόμενη παρυφή του σήματος. Αυτό μπορεί να γίνει με την εκτέλεση των εντολών:

echo “in” > /sys/class/gpio/gpio225/direction

echo “falling” > /sys/class/gpio/gpio225/edge

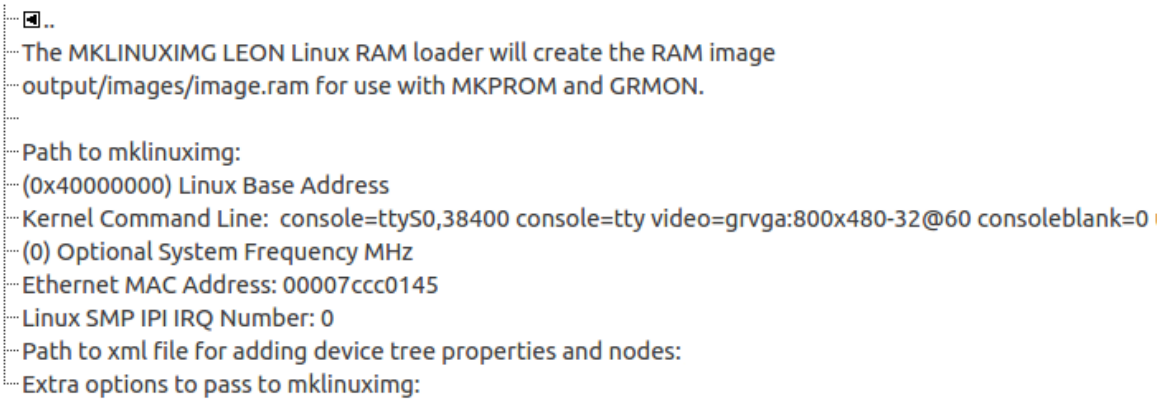
```
# cd /sys/class/gpio
# ls
export       gpiochip224  unexport
# echo 225 > export
# ls
export       gpio225      gpiochip224  unexport
# cd gpio225
# ls
active_low  device      direction   edge        subsystem   uevent      value
#
```

Εικόνα 5.10: Ενεργοποίηση GPIO θύρας

Μπορεί να εντοπιστεί η τιμή της θύρας GPIO ανά πάσα στιγμή διαβάζοντας το αρχείο “value” με την εντολή “cat value”. Στο λογισμικό που δημιουργήθηκε χρησιμοποιήθηκε αυτός ο τρόπος για την ανίχνευση των διακοπών. Η ακριβής περιγραφή του τρόπου λειτουργίας γίνεται στο κεφάλαιο περιγραφής του κώδικα.

Όταν έχουν καθοριστεί όλες οι παράμετροι του συστήματος και τα προγράμματα που χρειάζονται στο λειτουργικό σύστημα μπορεί να αρχίσει η δημιουργία της εικόνας Linux. Για αυτόν τον σκοπό χρησιμοποιείται το εργαλείο ‘MKLINUXIMG’ το οποίο είναι ενσωματωμένο στο ‘LINUXBUILD’. Στο σημείο του ‘LINUXBUILD’ το οποίο αφορά το ‘MKLINUXIMG’ μπορούν να καθοριστούν διάφοροι παράμετροι του συστήματος για το οποίο δημιουργείται η εικόνα Linux. Στην εικόνα 5.11 φαίνονται οι παράμετροι που χρησιμοποιήθηκαν για το σύστημα της διπλωματικής. Αυτό το παράθυρο βρίσκεται στο μενού του Linuxbuild “LEON Linux Loader configuration” Η διεύθυνση “Linux base address” δείχνει το σημείο της RAM στο οποίο φορτώνεται η Linux εικόνα. Το “kernel command line” είναι οι πρώτες εντολές που εκτελούνται στον πυρήνα κατά την φόρτωση του λειτουργικού συστήματος. Σε αυτό το σημείο έχει καθοριστεί να περνάει ο έλεγχος της κονσόλας στην οθόνη SVGA που είναι συνδεδεμένη με το σύστημα. Αυτό γίνεται με την εντολή:

console=tty video=grvga:800x480-32@60



Εικόνα 5.11: Παράμετροι MKLINUXIMG ('LEON Linux Loader configuration')

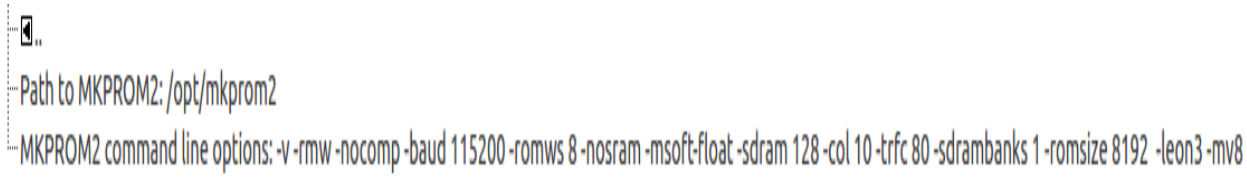
Η παράμετρος 'video' έχει την παρακάτω μορφή

video=<conn>:<xres>x<yres>[M][R][-<bpp>][@<refresh>][i][m][eDd]

όπου,
<conn>: συσκευή
<xres> x <yres>: ανάλυση
M: λειτουργία CVT
R: 'reduced blanking'
<bpp>: βάθος χρώματος
@<refresh>: ρυθμός ανανέωσης
i: 'interlaced (non-CVT mode)'
m: όρια (margins)
e: αναγκαστικά ανοικτή έξοδος
d: αναγκαστικά κλειστή έξοδος
D: ψηφιακή έξοδος αναγκαστικά ανοικτή

Η παράμετρος 'consoleblank' αφορά τη λειτουργία 'screen saver' της οθόνης. Η οθόνη με τις προκαθορισμένες τιμές μπαίνει σε λειτουργία εξοικονόμησης ενέργειας μετά από κάποιο διάστημα αδράνειας του συστήματος. Με την επιλογή 'consoleblank=0' αυτή η λειτουργία απενεργοποιείται. Μπορεί να ενεργοποιηθεί ξανά όποτε ο χρήστης το επιθυμεί μετά την φόρτωση του λειτουργικού συστήματος αλλάζοντας τιμή στο αρχείο '/sys/module/kernel/parameters/consoleblank'.

Τέλος για την δημιουργία μιας εικόνας η οποία μπορεί να αποθηκευτεί σε μια μνήμη FLASH/PROM χρησιμοποιείται η εφαρμογή MKPROM, η οποία επίσης είναι ενσωματωμένη στο LINUXBUILD (στο μενού 'Create PROM image with MKPROM2'). Στο σημείο αυτό του LINUXBUILD πρέπει να καθοριστούν με ακρίβεια οι παράμετροι των μνημών του συστήματος και το είδος του συστήματος. Οι παράμετροι που μπορούν να χρησιμοποιηθούν υπάρχουν στις οδηγίες χρήσης του MKPROM. Για το σύστημα της διπλωματικής χρησιμοποιήθηκαν οι παράμετροι της εικόνας 5.12.



Εικόνα 5.12: Παράμετροι MKPROM (μενού 'Create PROM image with MKPROM2')

Όταν καθοριστούν και οι παράμετροι του MKLINUXIMG και του MKPROM μπορεί να δημιουργηθούν οι εικόνες Linux ('build' μενού). Οι εικόνες που δημιουργούνται αποθηκεύονται στον φάκελο /output/images του LINUXBUILD. Το αρχείο Image.prom

είναι η εικόνα Linux η οποία μπορεί να φορτωθεί στη FLASH μέσω του GRMON με τις εντολές:

Flash erase all (για το καθάρισμα της flash)
flash load …/output/images/image.prom (φόρτωση της εικόνας στη flash)

```
grmon2> flash

  AMD-style 8-bit flash

  Manuf.        : AMD
  Device        : MX29LV128MB

  1 x 8 Mbytes = 8 Mbytes total @ 0x00000000

  CFI information
  Flash family  : 2
  Flash size    : 64 Mbit
  Erase regions : 2
  Erase blocks  : 135
  Write buffer  : 32 bytes (limited to 32)
  Lock-down     : Supported
  Region  0     : 8 blocks of 8 kB
  Region  1     : 127 blocks of 64 kB

grmon2>
```

Εικόνα 5.13: Εντοπισμός FLASH μνήμης στο GRMON

Υπάρχουν διάφοροι τρόποι για την εισαγωγή αρχείων και προγραμμάτων τα οποία δεν υπάρχουν στο LINUXBUILD. Στην υλοποίηση της διπλωματικής χρησιμοποιήθηκε ο φάκελος στο “client” PC ’\dist\buildroot\build-br\target’, όπου είναι εγκατεστημένο το LINUXBUILD. Σε αυτό τον φάκελο μπορούν να γίνουν τροποποιήσεις της εικόνας Linux και να προστεθούν αρχεία που θα χρησιμοποιηθούν κατά την φόρτωση του λειτουργικού συστήματος. Για παράδειγμα στο φάκελο ‘/opt’ αποθηκεύτηκαν οι συμπιεσμένες εικόνες που χρησιμοποιούνται από το πρόγραμμα μετρήσεων. Στον φάκελο ‘/bin’ έχουν αποθηκευτεί τα προγράμματα μετρήσεων. Με αυτό τον τρόπο τα προγράμματα μπορούν να χρησιμοποιηθούν σαν εντολές του συστήματος και να αρχίσουν να εκτελούνται όταν πατηθεί το όνομά τους στο τερματικό.

```
# ls /opt
calculator.raw.gz              main_menu.raw.gz
calibration_menu.raw.gz        welcome.raw.gz
hardware_configuration.raw.gz
# ls /bin
ash             dumpkmap        login           pipe_progress   stty
busybox         echo            ls              printenv        su
cat             egrep           lsattr          ps              sync
catv            false           mkdir           pwd             tar
chattr          fdflush         mknod           rm              touch
chgrp           fgrep           mktemp          rmdir           true
chmod           getopt          more            run-parts       umount
chown           grep            mount           sed             uname
cp              gunzip          mountpoint      setarch         usleep
cpio            gzip            mt              setserial       vi
date            hostname        mv              sh              watch
dd              kill            netstat         sleep           zcat
df              linux32         nice            smetrics
dmesg           linux64         pidof           smetrics4
dnsdomainname   ln              ping            smetrics_test
#
```

Εικόνα 5.14: Εικόνες και προγράμματα (smetrics,smetrics4,smetrics_test) αποθηκευμένα στην Linux εικόνα

Τέλος μπορούν να γίνουν τροποποιήσεις στα ήδη υπάρχοντα αρχεία. Για παράδειγμα το αρχείο ‘/etc/network/interfaces’ έχει τροποποιηθεί ώστε να δημιουργείται IP

διεύθυνση αυτόματα κατά την διάρκεια φόρτωσης του λειτουργικού συστήματος. Έχουν προστεθεί οι γραμμές:

 # Do DHCP for ETH0
auto eth0
iface eth0 inet dhcp

Μετά την τροποποίηση αυτών των φακέλων πρέπει να ξανατρέξει η δημιουργία εικόνας από το LINUXBUILD (MKLINUXIMG και MKPROM) ώστε οι αλλαγές να οριστικοποιηθούν στην εικόνα LINUX.

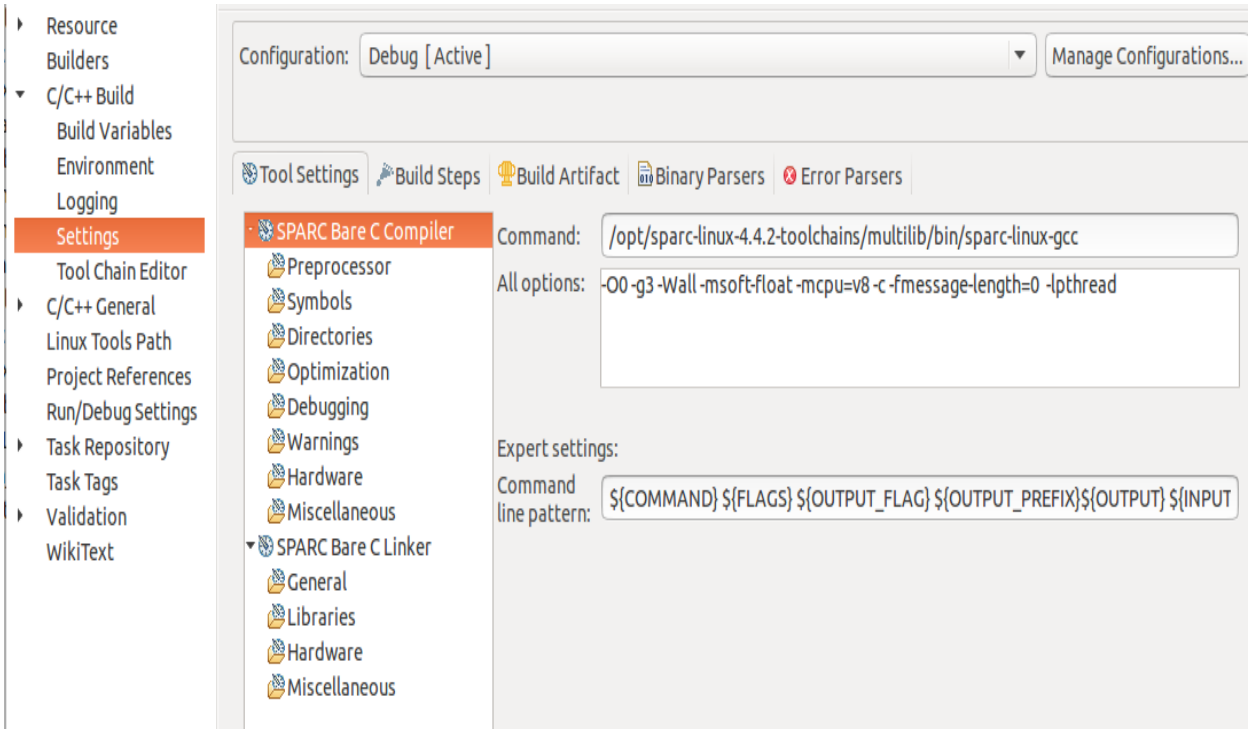
5.3 Αποσφαλμάτωση και μεταγλώττιση προγραμμάτων

Για τις μετρήσεις των αισθητήρων δημιουργήθηκαν κάποια προγράμματα σε γλώσσα C, τα οποία εκτελούνται στο “user-space” επίπεδο του Linux. Αυτά τα προγράμματα εκμεταλλεύονται τις δυνατότητες του λειτουργικού συστήματος και αξιοποιούν τα περιφερειακά που είναι συνδεδεμένα στον επεξεργαστή LEON3. Όπως ειπώθηκε προηγουμένως τα προγράμματα έχουν αποθηκευτεί στον φάκελο /bin κατά την διάρκεια δημιουργίας της εικόνας Linux. Η αποσφαλμάτωση και η ανάπτυξη των προγραμμάτων έγινε με το πρόγραμμα ECLIPSE, χρησιμοποιώντας την δυνατότητα απομακρυσμένης αποσφαλμάτωσης του συστήματος μέσω SSH σύνδεσης. Η μεταγλώττιση (compile) των προγραμμάτων γίνεται με τα εργαλεία σχεδίασης τα οποία διατίθενται από την ιστοσελίδα της Aeroflex Gaisler. Ακολουθούν κάποιες οδηγίες για την σωστή χρήση των εργαλείων.

Το εργαλείο σχεδίασης μπορεί να κατεβεί από την εξής ιστοσελίδα:

http://www.gaisler.com/anonftp/linux/linux-2.6/toolchains/

Στην εικόνα 5.15 και 5.16 φαίνεται η σύνδεση του εργαλείου σχεδίασης με το ECLIPSE και οι παράμετροι που χρησιμοποιήθηκαν για την μεταγλώττιση των προγραμμάτων.



Εικόνα 5.15: Παράμετροι για την μεταγλώττιση στο ECLIPSE

Η παράμετρος ‘-msoft-float’ αφορά επεξεργαστή χωρίς μονάδα FPU και η παράμετρος ‘–lpthread’ χρειάζεται όταν χρησιμοποιούνται παράλληλα νήματα (thread) στο

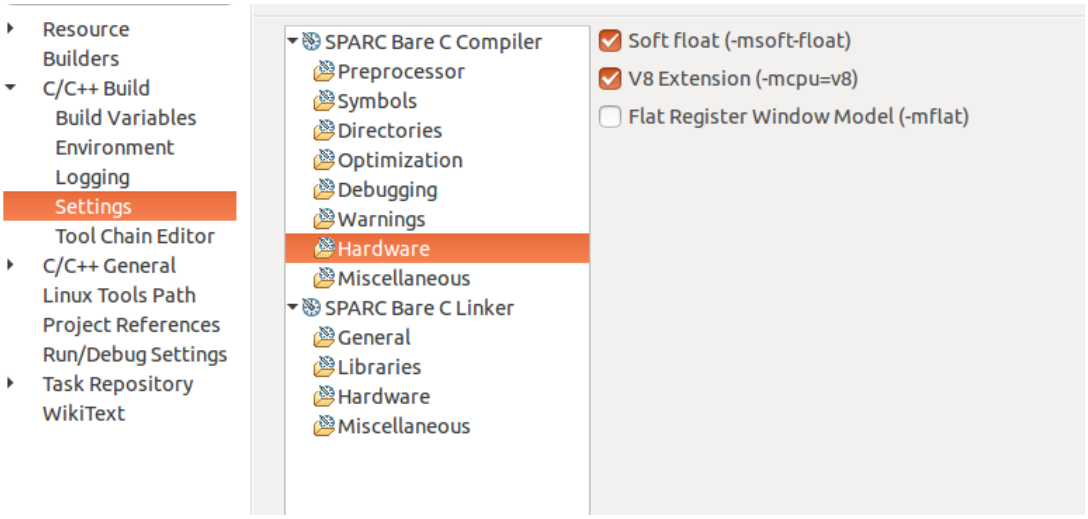
πρόγραμμα. Ακόμα για την αποσφαλμάτωση μέσω GRMON θα πρέπει να εγκατασταθεί μια επέκταση του ECLIPSE, το LEON IDE. Αυτό μπορεί να γίνει ως εξής:

1.Επιλέξτε στο ECLIPSE περιβάλλον Help->Install New Software

2.Πατήστε Add

3.Στο πεδίο 'name' εισάγετε 'LIDE' και στο 'Location' http://gaisler.com/eclipse/lide_kepler/ και πατήστε OK

4.Επιλέξτε το LEON IDE και πατήστε Next. Ακολουθεί η εγκατάσταση της επέκτασης και μετά από μια επανεκκίνηση του ECLIPSE η επέκταση θα μπορεί να χρησιμοποιηθεί.



Εικόνα 5.16: Επιλογή 'soft float' και 'V8 extension' στο ECLIPSE

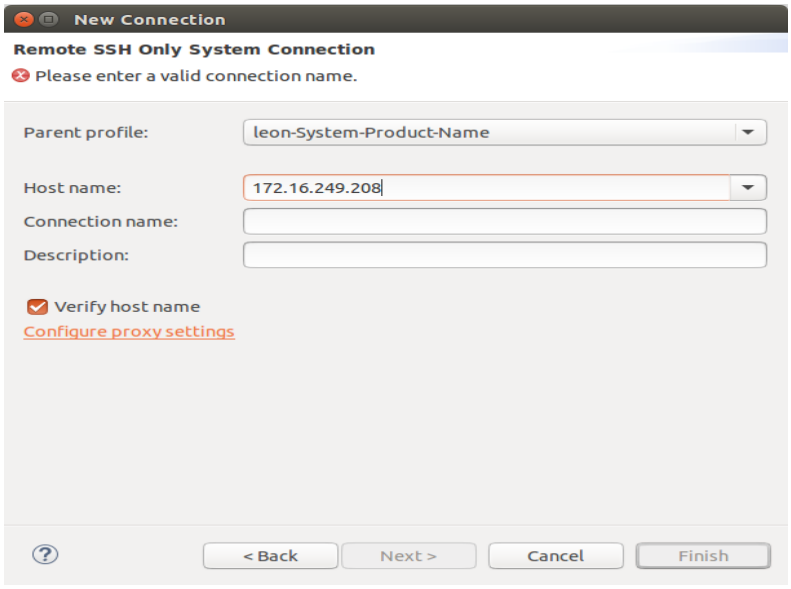
Η αποσφαλμάτωση του συστήματος μπορεί να γίνει μέσω SSH σύνδεσης. Αρχικά, θα πρέπει να έχει φορτωθεί στο 'host' σύστημα η εικόνα Linux, κατά προτίμηση χρησιμοποιώντας την εικόνα Linux για την FLASH (image.prom). Το λειτουργικό σύστημα θα πρέπει να διαθέτει δυνατότητες SSH σύνδεσης και να έχει συνδεθεί ένα καλώδιο ETHERNET ώστε να υπάρχει IP διεύθυνση στο σύστημα. Για αυτόματη απόκτηση της IP διεύθυνσης θα πρέπει το καλώδιο να είναι συνδεδεμένο κατά την φόρτωση του λειτουργικού συστήματος και να έχει γίνει η τροποποίηση στο αρχείο 'interfaces', που αναφέρθηκε προηγουμένως . Για τον εντοπισμό της IP διεύθυνσης στο 'host' σύστημα μπορεί να χρησιμοποιηθεί η εντολή 'ifconfig' (εικόνα 5.17). Τέλος θα πρέπει να υπάρχει κάποιος κωδικός για την είσοδο σαν υπέρ-χρήστης στο σύστημα.

```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:00:7C:CC:01:45
          inet addr:192.168.2.6  Bcast:192.168.2.255  Mask:255.255.255.0
          inet6 addr: fe80::200:7cff:fecc:145/64 Scope:Link
          UP BROADCAST RUNNING  MTU:1500  Metric:1
          RX packets:2601 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1462 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:219566 (214.4 KiB)  TX bytes:185468 (181.1 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

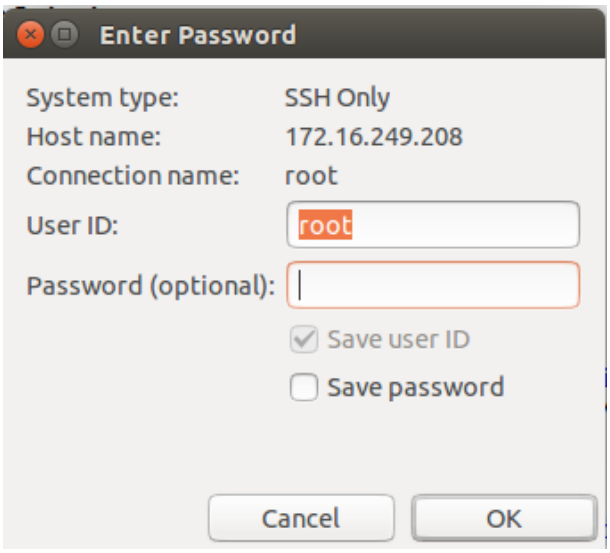
Εικόνα 5.17: Εντοπισμός IP διεύθυνσης σε Linux σύστημα

Στο ‘client’ σύστημα θα πρέπει να υπάρχει επίσης IP διεύθυνση και θα πρέπει και τα 2 συστήματα (host-client) να είναι στο ίδιο υπό-δίκτυο (sub-net). Αφού έχει γίνει η σύνδεση των συστημάτων στο δίκτυο μπορεί να αρχίσει η αποσφαλμάτωση μέσω του ECLIPSE. Για την διπλωματική χρησιμοποιήθηκαν διάφοροι τρόποι αποσφαλμάτωσης. Είτε χρησιμοποιήθηκε η κονσόλα του ‘client’ υπολογιστή είτε η SVGA οθόνη του ‘host’ συστήματος. Για την σύνδεση με το ‘host’ σύστημα μέσω ECLIPSE ακολουθούνται τα παρακάτω βήματα:

1. Window->open perspective->other->remote system explorer
2. Δημιουργία νέας σύνδεσης- δεξί κλικ-> new-> connection
3. Επιλογή ‘SSH only’
4. Στο host name τοποθετείται η IP διεύθυνση του host συστήματος(εικόνα 5.18)
5. Στο user id προτείνεται η προκαθορισμένη τιμή ‘root’ και password ο κωδικός που έχει επιλεχτεί για τον υπερ-χρήστη (εικόνα 5.19)
6. Αν βγει ένα μήνυμα με ‘warning’ πατήστε ‘yes’

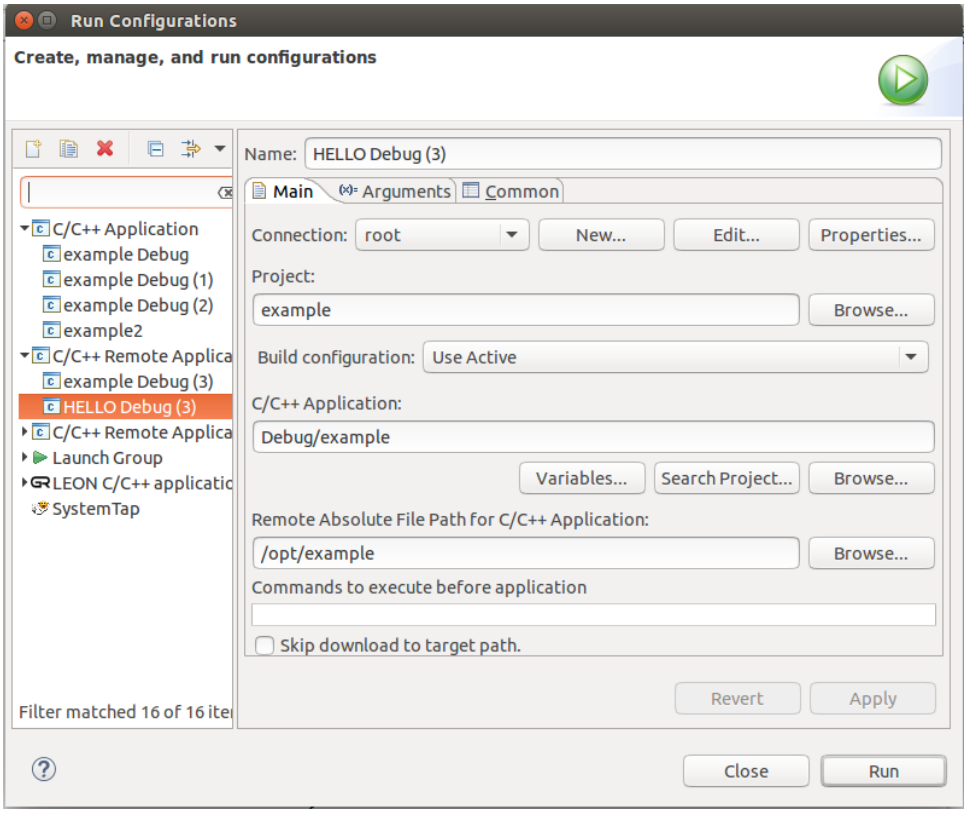


Εικόνα 5.18: SSH σύνδεση μέσω ECLIPSE



Εικόνα 5.19: SSH σύνδεση μέσω ECLIPSE ID και Password

Όταν επιτευχθεί η σύνδεση μπορεί να γίνει περιήγηση στους φακέλους του ‘host’ συστήματος. Μπορούν να μεταφερθούν αρχεία στους φακέλους και να ληφθούν αρχεία από το ‘host’ σύστημα. Οι αλλαγές αυτές στο παρόν σύστημα χάνονται μετά το κλείσιμο του. Για την αποσφαλμάτωση ενός προγράμματος στο οποίο έχει γίνει μεταγλώττιση, μπορεί είτε να μεταφερθεί το πρόγραμμα σε κάποιον φάκελο του ‘host’ συστήματος είτε να τρέξει στον ‘client’ μέσω του ECLIPSE κατευθείαν και τα μηνύματα λάθους να εμφανιστούν στην κονσόλα του ECLIPSE. Με τον πρώτο τρόπο πρέπει να μεταφερθεί το εκτελέσιμο αρχείο του προγράμματος σε κάποιο φάκελο του host συστήματος, π.χ με ‘drag and drop’. Μετά θα πρέπει στο host σύστημα στο φάκελο που είναι το πρόγραμμα να πατηθεί η εντολή “./name_of_application”. Τα αποτελέσματα του προγράμματος θα εμφανιστούν στην κονσόλα του συστήματος, για την διπλωματική αυτή ήταν η SVGA οθόνη. Ο δεύτερος τρόπος είναι η απευθείας εκτέλεση του προγράμματος μέσω του ECLIPSE. Για να γίνει αυτό θα πρέπει το “perspective” του ECLIPSE να είναι στη προκαθορισμένη επιλογή “C/C++”. Στο πρόγραμμα που επιθυμείται η αποσφαλμάτωση επιλέγεται με δεξί κλικ το μενού ‘run configurations’ (Εικόνα 5.20).



Εικόνα 5.20: Run configurations για απομακρυσμένη αποσφαλμάτωση

Στην επιλογή ‘C/C++ Remote applications’, τοποθετείται στο πεδίο ‘project’ το όνομα του προγράμματος, στο πεδίο ‘C/C++’ το σημείο που είναι το εκτελέσιμο αρχείο και στο πεδίο ‘Remote absolute file path for ‘C/C++ Applications’ το σημείο που το πρόγραμμα θα τρέξει στο ‘host’ σύστημα. Όταν πατηθεί η επιλογή ‘run’ το πρόγραμμα θα τρέξει στο host σύστημα αλλά τα μηνύματα λάθους και οι εντολές ‘printf’ θα εμφανιστούν στην κονσόλα του ECLIPSE. Το εκτελέσιμο αρχείο θα έχει μεταφερθεί στο ‘host’ σύστημα και μπορεί να εκτελεστεί και αποκεί αργότερα. Γενικά μπορούν να χρησιμοποιηθούν αρκετοί τρόποι για την αποσφαλμάτωση αλλά οι δύο τρόποι που περιγράφηκαν χρησιμοποιήθηκαν στην υλοποίηση της διπλωματικής.

5.4 Προγράμματα μετρήσεων

Για την υλοποίηση χρησιμοποιήθηκαν διάφορα προγράμματα τα οποία είτε χρησιμοποιούν την ‘touch screen’ λειτουργία της οθόνης είτε όχι. Για τις πειραματικές

μετρήσεις των αισθητήρων χρησιμοποιήθηκε ένα πρόγραμμα το οποίο δέχεται ως είσοδο το χρονικό παράθυρο των μετρήσεων, τον αριθμό των μετρήσεων και το χρόνο μεταξύ των μετρήσεων. Οι μετρήσεις λαμβάνονται από 4 αισθητήρες χωρητικότητας που είναι συνδεδεμένοι στο σύστημα μέσω του εξωτερικού κυκλώματος. Η συχνότητα του κάθε αισθητήρα μετριέται σειριακά, δηλαδή πρώτα ο 1ος αισθητήρας, μετά ο 2ος , 3ος , 4ος ,1ος κ.τ.λ. Τα αποτελέσματα, δηλαδή οι συχνότητες λειτουργίας του κάθε αισθητήρα, αποθηκεύονται σε αρχεία τα οποία μπορούν να διαβαστούν στην SVGA οθόνη ή να μεταφερθούν μέσω SSH σύνδεσης σε ένα απομακρυσμένο υπολογιστή.

Στην εικόνα 5.21 φαίνεται το αρχικό κομμάτι του προγράμματος. Η εντολή *FILE *ptr_file* δημιουργεί μια μεταβλητή αρχείου, η οποία μπορεί να χρησιμοποιηθεί για την δημιουργία αρχείων και την αποθήκευση δεδομένων σε αυτό. Στην εικόνα 5.21 φαίνεται ο τρόπος δημιουργίας των αρχείων. Κάθε κανάλι έχει το δικό του αρχείο.

Για την επικοινωνία με τους καταχωρητές του κυκλώματος μετρήσεων δεν μπορούν να χρησιμοποιηθούν οι φυσικές διευθύνσεις στις οποίες είναι συνδεδεμένο το περιφερειακό. Για παράδειγμα στο GRMON το κύκλωμα μέτρησης φαίνεται να είναι συνδεδεμένο στις διευθύνσεις ‘80000c00-80000d00’. Αυτή η διεύθυνση αφορά στο περιφερειακό, όταν δεν υπάρχει κάποιο λειτουργικό σύστημα, είναι δηλαδή η φυσική διεύθυνση. Στο ‘user-space’ επίπεδο του Linux λειτουργικού συστήματος όλες αυτές οι διευθύνσεις δεν μπορούν να χρησιμοποιηθούν κατευθείαν και ο χρήστης δεν μπορεί να έχει απευθείας πρόσβαση στις φυσικές διευθύνσεις. Στο ‘user-space’ επίπεδο χρησιμοποιούνται εικονικές (virtual) διευθύνσεις. Για την πρόσβαση στο περιφερειακό έπρεπε να δημιουργηθεί ένας οδηγός ή να χρησιμοποιηθεί η συνάρτηση *mmap*. Η συνάρτηση αυτή, αν χρησιμοποιηθεί κατάλληλα, επιτρέπει την πρόσβαση σε φυσικές διευθύνσεις από το ‘user-space’ επίπεδο. Η *mmap* συνάρτηση μπορεί να αντιστοιχίσει μια φυσική διεύθυνση με μια εικονική. Με αυτόν τον τρόπο μπορεί η νέα διεύθυνση (η οποία έχει αντιστοιχηθεί με την φυσική διεύθυνση) να χρησιμοποιηθεί στο “user-space” επίπεδο του λειτουργικού συστήματος. Η συνάρτηση *mmap* έχει την παρακάτω μορφή:

#include <sys/mman.h>

*void *mmap(void *addr, size_t length, int prot, int flags, int fd, off_t offset);*

,όπου το *addr* είναι η αρχική διεύθυνση για την νέα αντιστοίχιση, αν είναι μηδέν τότε ο πυρήνας επιλέγει την διεύθυνση, τ*o length* αφορά στο μέγεθος της αντιστοίχισης, το πεδίο *prot* αφορά στο τρόπο προστασίας της μνήμης κατά την αντιστοίχιση*,* στο πεδίο *flags* αποφασίζεται, αν η ανανέωση στην αντιστοίχιση είναι ορατή και στις άλλες διεργασίες οι οποίες αντιστοιχίζονται στην ίδια περιοχή, το *fd* δηλαδή *file descriptor* είναι το αρχείο στο οποίο επιθυμείται η αντιστοίχιση και το *offset* καθορίζει το αρχικό σημείο το οποίο θα αντιστοιχηθεί από τον *fd*.

Όλες οι φυσικές διευθύνσεις του συστήματος είναι αποθηκευμένες στο αρχείο */dev/mem*. Έτσι χρησιμοποιείται αυτό το αρχείο ως *fd*, όπως φαίνεται και στην εικόνα 5.21. Το μέγεθος της αντιστοίχισης *mmap_size* έχει καθοριστεί στο κώδικα με την εντολή *#define MAP_SIZE 8192UL.* Οι σελίδες που δημιουργούνται μπορούν και να διαβαστούν και να γραφτούν *(PROT_READ/PROT_WRITE).* Το *flag* έχει την τιμή *MAP_SHARED.* Το *memfd* αφορά στον φάκελο */dev/mem* και ως *offset* έχει οριστεί η φυσική διεύθυνση του κυκλώματος *0x80000c00 (addr_textvga*) με μια μάσκα που έχει την τιμή *MAP_MASK= (MAP_SIZE - 1).*

```
    FILE *ptr_file;/* δημιουργία μεταβλητής αρχείου */
    memfd = open("/dev/mem", O_RDWR | O_SYNC); /*καθορισμός μεταβλητής memfd και άνοιγμα του fd /dev/mem */
    mapped_base = mmap(0, MAP_SIZE, PROT_READ|PROT_WRITE, MAP_SHARED, memfd, addr_textvga & ~MAP_MASK);/*mmap συνάρτηση */
    if (mapped_base == MAP_FAILED) /*μήνυμα σε περίπτωση λάθους */
    {
      errx(1, "mmap failure");/*περίπτωση λάθους */
    }
      mapped_dev_base = mapped_base + (addr_textvga & MAP_MASK);
      pu = mapped_dev_base; /*αποθήκευση της νέας virtual διεύθυνσης σε μια μεταβλητή */
    struct my_play_regs_t  * my_play_regs = (struct my_play_regs_t  *)pu; //καταχωρητές κυκλώματος μέτρησης
    struct my_play_regs_t {
    volatile int write;   // εγγραφή στους καταχωρητές
    volatile int read; // διάβασμα αποτελέσματος
    };
    status=pthread_create(&thread_id,NULL,thread_routine2,NULL);//δημιουργία νήματος για διακοπη1
    status=pthread_create(&thread_id,NULL,thread_routine3,NULL);//δημιουργία νήματος για διακοπη2

    ptr_file =fopen("/opt/channel1.txt", "w"); //δημιουργία αρχείων για το γράψιμο των αποτελεσμάτων
    fclose(ptr_file);                          //κάθε κανάλι έχει τον δικό του φάκελο
    ptr_file =fopen("/opt/channel2.txt", "w");
    fclose(ptr_file);
    ptr_file =fopen("/opt/channel3.txt", "w");
    fclose(ptr_file);
    ptr_file =fopen("/opt/channel4.txt", "w");
    fclose(ptr_file);
system("clear");
```

Εικόνα 5.21: Αρχικό κομμάτι προγράμματος μέτρησης

Μετά την αντιστοίχιση της φυσικής διεύθυνσης σε μια εικονική διεύθυνση, η εικονική διεύθυνση μπορεί να χρησιμοποιηθεί για το διάβασμα και το γράψιμο στους καταχωρητές του κυκλώματος. Στην εικόνα 5.21 φαίνεται ο καθορισμός των καταχωρητών. Η διεύθυνση από την οποία διαβάζονται οι καταχωρητές είναι η *pu*, η εικονική διεύθυνση της αντιστοίχισης. Οι καταχωρητές είναι δύο, ένας για το γράψιμο, ο οποίος ενημερώνει τους τρεις 5-μπιτ καταχωρητές του κυκλώματος και το σήμα 'wr' (βλ. κεφάλαιο 4) και ένας για το διάβασμα του αποτελέσματος της μέτρησης. Στην συνέχεια, δημιουργούνται τα νήματα διακοπών τα οποία τρέχουν παράλληλα με το κεντρικό πρόγραμμα (εικόνα 5.22).

Το νήμα διακοπής εκτελείται συνέχεια κατά την διάρκεια εκτέλεσης του προγράμματος και ενημερώνει το κυρίως πρόγραμμα αν έχει υπάρξει κάποια διακοπή. Στο κεφάλαιο 5.2 παρουσιάστηκε ο τρόπος δημιουργίας των διακοπών με τον οδηγητή GPIO. Αρχικά, πρέπει να γίνει η εξαγωγή της θύρας-ακροδέκτη GPIO που αντιστοιχεί στην επιθυμητή διακοπή (βλ. κεφάλαιο 4 - εικόνα 4.13). Στη συνέχεια, ορίζεται η θύρα ως είσοδος και η διακοπή να δημιουργείται στην ανερχόμενη παρυφή. Για να εντοπιστεί όσο πιο γρήγορα γίνεται η διακοπή χρησιμοποιήθηκε η συνάρτηση *poll*, η οποία περιμένει να αλλάξει η τιμή της θύρας για να συνεχιστεί η εκτέλεση του προγράμματος. Όταν γίνει κάποια διακοπή, μια καθολική μεταβλητή *z* παίρνει την τιμή 1. Αυτή την μεταβλητή μπορεί να την δει και το κυρίως πρόγραμμα και έτσι να ενημερωθεί για την διακοπή. Στη συνέχεια, απενεργοποιείται η θύρα και το νήμα ξεκινάει από την αρχή. Έχει δημιουργηθεί ένα τέτοιο νήμα για κάθε διακοπή του συστήματος. Κάθε νήμα ελέγχει και μια διαφορετική καθολική μεταβλητή για την ενημέρωση του κυρίως προγράμματος.

```
start1:
    /* export */
      fn = "/sys/class/gpio/export"; //εξαγωγή gpio226 το οποίο αντιστοιχεί στο gpio(2)
      fd = open(fn, O_WRONLY); if(fd < 0)  ERREXIT("open export")
      rc = write(fd, "226", 3); if(rc != 3) ERREXIT("write export")
      close(fd);
      /* direction */
      fn = "/sys/class/gpio/gpio226/direction";//ορισμός της θύρας ως είσοδος
      fd = open(fn, O_RDWR);  if(fd < 0)  ERREXIT("open direction")
      rc = write(fd, "in", 3);if(rc != 3) ERREXIT("write direction")
      close(fd);
      /* edge */
      fn = "/sys/class/gpio/gpio226/edge";//διακοπή στην ανερχόμενη κορυφή
      fd = open(fn, O_RDWR);        if(fd < 0)  ERREXIT("open edge")
      rc = write(fd, "rising", 8); if(rc != 8) ERREXIT("write edge")
      rc = lseek(fd, 0, SEEK_SET);  if(rc < 0)  ERREXIT("lseek edge")
      rc = read(fd, buf, 10);       if(rc <= 0) ERREXIT("read edge")
      buf[rc] = '\0';
      close(fd);
      /* open the 'value' file */
      fn = "/sys/class/gpio/gpio226/value"; //άνοιγμα του φακέλου που περιέχει την τιμή της θύρας
      fd = open(fn, O_RDWR);  if(fd < 0)  ERREXIT("open value")
      rc = read(fd, buf, 2);       if (rc != 2) ERREXIT("read value")//πρέπει να γίνει ένα διάβασμα πριν την χρήση της poll
      xfds[0].fd      = fd;// poll()
      xfds[0].events  = POLLPRI | POLLERR;
      xfds[0].revents  = 0;
      rc = poll(xfds, 1, -1);//περιμένωντας για την διακοπή
      z=1;
      close(fd);
      if(rc == -1) ERREXIT("poll value")
      fn = "/sys/class/gpio/unexport";//απενεργοποίηση της θύρας gpio
      fd = open(fn, O_WRONLY); if(fd < 0)  ERREXIT("open unexport")
      rc = write(fd, "226", 3); if(rc != 3) ERREXIT("write unexport")
      close(fd);
     goto start1; // επιστροφή στην αρχή.
      return 0;
```

Εικόνα 5.22: Νήμα διακοπής

Στην εικόνα 5.23 φαίνεται ο κώδικας μέτρησης της συχνότητας ενός καναλιού. Αρχικά, ζητείται ο αριθμός των μετρήσεων (μεταβλητή points) για κάθε κανάλι και η καθυστέρηση μεταξύ των μετρήσεων(delay). Το κάθε κανάλι μετριέται μετά το άλλο, σειριακά. Μετά την μέτρηση του τελευταίου καναλιού υπάρχει η δυνατότητα μιας καθυστέρησης π.χ 5s. Μετά την καθυστέρηση αρχίζει πάλι η μέτρηση από το 1ο κανάλι. Αυτό θα γίνει για όσες φορές έχει καθοριστεί από την μεταβλητή 'points'.

Στην συνέχεια επιλέγεται το μέγεθος του χρονικού παραθύρου(μεταβλητή 'window'). Η μεταβλητή μπορεί να πάρει τιμές από 0-31, για αυτό χρησιμοποιείται και η μάσκα. Η τιμή αυτή αναπαριστά το 'n' στον τύπο 2^n. Το 'loop' που υπολογίζει το x3 αναπαριστά αυτόν τον τύπο. Έτσι αν το window=1 , τότε το x3=4 κ.τ.λ. Η μεταβλητή x2 είναι ίση με την περίοδο του ρολογιού 50Mhz, αφού $1/(50*10^9)$ =0.00000002. Ακολουθούν οι μετρήσεις. Αρχικά, προσαρμόζονται οι μεταβλητές set1,y2 σε μορφή που είναι συμβατή με τους καταχωρητές του κυκλώματος μέτρησης (βλ. κεφάλαιο 4). Οι καταχωρητές του κυκλώματος μέτρησης είναι τρεις των 5 μπιτ και δύο του 1 μπιτ (wr,reset). Το πρωτόκολλο σύνδεσης με τον AMBA δίαυλο έχει παρουσιαστεί στο κεφάλαιο 4. Οι καταχωρητές είναι συνδεδεμένοι στον δίαυλο APB, όπως φαίνεται στην εικόνα 5.24. Ο δίαυλος επικοινωνίας είναι 32 μπιτ αλλά χρησιμοποιούνται μόνο τα 17 μπιτ. Τα πρώτα 5 καθορίζουν ποιος ταλαντωτής θα μετρηθεί (μεταβλητή y4 στον κώδικα, για το 1ο κανάλι είναι μηδέν). Τα επόμενα 5 είναι το χρονικό παράθυρο (y3), τα επόμενα 5 είναι ο

καταχωρητής εκκίνησης της δειγματοληψίας και τα επόμενα 2 είναι καταχωρητής εγγραφής(wr) και το σήμα επανεκκίνησης.

```
loop:
printf (ANSI_COLOR_MAGENTA"Choose number of points\n"); //εισαγωγή αριθμού σημείων
scanf("%d",&points);
printf (ANSI_COLOR_MAGENTA"Choose time delay between measurements (microseconds 10^-6 s)\n");
scanf("%d",&delay);// εισαγωγή καθυστέρησης μεταξύ των μετρήσεων
printf (ANSI_COLOR_MAGENTA"Choose time window\n"); //καθορισμός χρονικού παραθύρου
scanf("%d",&window);
y3=(window)&mask; //mask=0x1f
x3=1;
for ( i=0; i < window; i++)
       {
    x3=x3*2; //καθορισμός υποδιαίρεσης ρολογιού
       }
x2=0.00000002; // περίοδος ρολογιού 50Mhz
for (t = 0; t < points; t++)
     {
    y4=0&mask;
    set1=((00001<<15) | (00001<<10) | (y3<<5) | y4); //τοποθέτηση παραμέτρων στην εντολή γραψίματος
    y2=((00001<<15) | (00000<<10) | (y3<<5) | y4); //το μπιτ 10 κάνει ένα παλμό
    my_play_regs->write=set1; //εγγραφή καταχωρητή write
    my_play_regs->write=y2;
    while (z==0){ //περιμένει το z να γίνει 1, από το νήμα διακοπής, που σηματοδοτεί την ολοκλήρωση της μέτρησης
    }
    z=0;
    if (overflow==1){ //αν έχει γίνει διακοπή υπερχείλισης, από το νήμα διακοπής που αντιστοιχεί στο gpio(3)
    printf (ANSI_COLOR_RED"    OVERFLOW HAPPENED TO CHANNEL 1 REDUCE TIME WINDOW!!!\n"ANSI_COLOR_RESET );
    goto end;
    }
    b=my_play_regs->read; //αν όχι διάβασμα του καταχωρητή αποτελέσματος
    b=b&0x0000ffff; //μόνο τα 16 μπιτ είναι χρήσιμα
    x4=x3*x2; //χρονικό παράθυρο*περίοδος ρολογιού
    x4=x4/b; //(χρονικό παράθυρο*περίοδος ρολογιού)/αριθμός παλμών=περίοδος ταλάντωσης
    x4=1/x4; // 1/περίοδος ταλάντωσης= συχνότητα ταλάντωσης
    x5=x4;
    ptr_file =fopen("/opt/channel1.txt", "a"); //άνοιγμα φακέλου αποτελεσμάτων
    fprintf(ptr_file," %d  \n",x5); //γράψιμο της συχνότητας στον φάκελο
    fclose(ptr_file);//κλείσιμο φακέλου
```

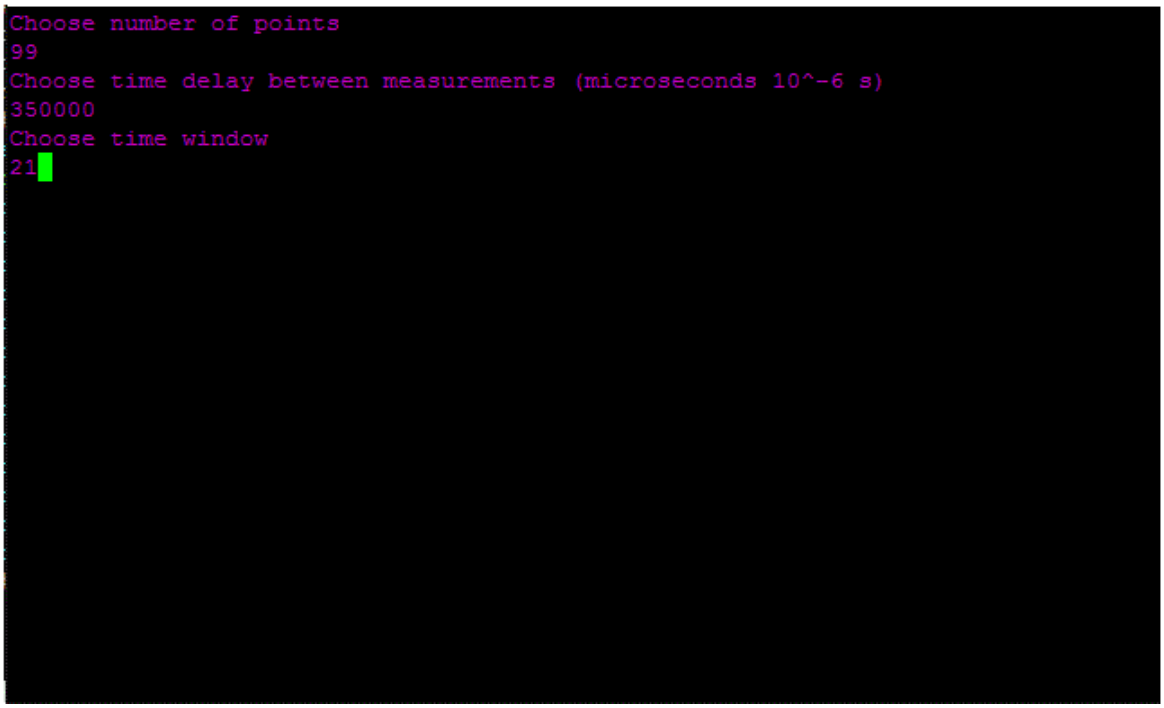
Εικόνα 5.23: Κώδικας μέτρησης συχνότητας

```
oscilator_control <= apbi.pwdata(4 downto 0);
sampling_time_control <= apbi.pwdata(9 downto 5);
signal_control <= apbi.pwdata(14 downto 10);
wr <= apbi.pwdata(15);
reset <= apbi.pwdata(16);
```

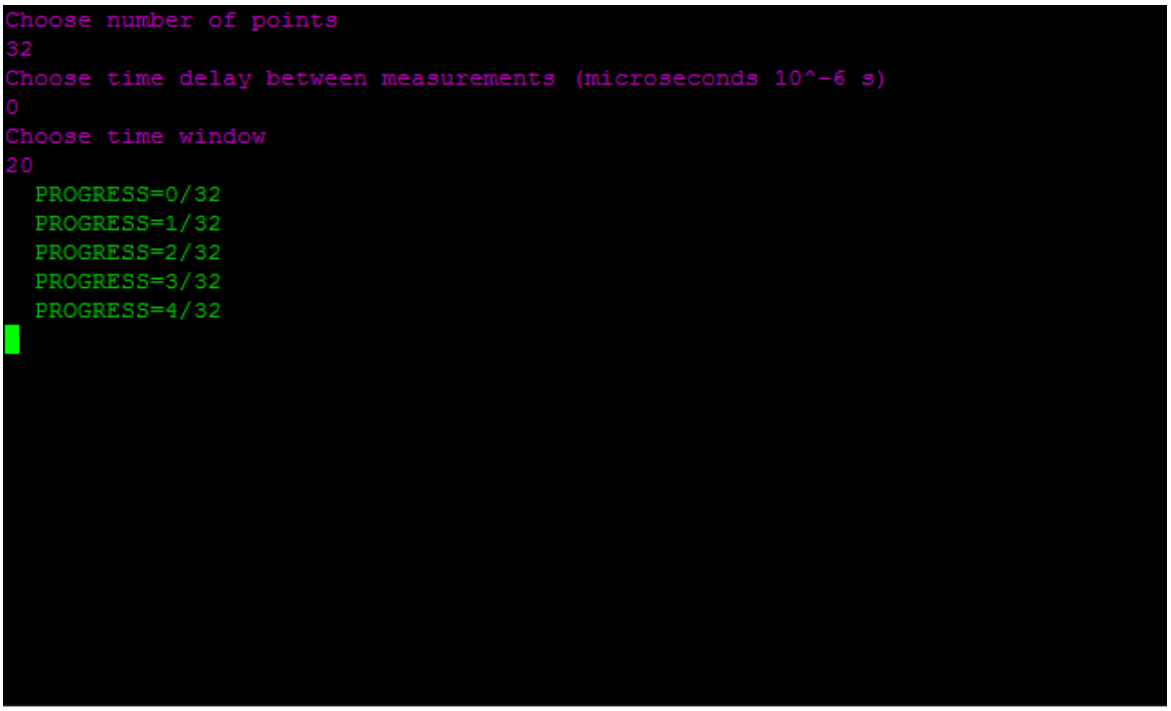
Εικόνα 5.24: Σύνδεση καταχωρητών στον δίαυλο APB κατά το διάβασμα

Έτσι για την εκκίνηση της μέτρησης στέλνονται δύο τιμές στους καταχωρητές η 1^{η} με το μπιτ εκκίνησης δειγματοληψίας να έχει την τιμή ένα και η 2^{η} με την τιμή μηδέν. Στην εικόνα 5.23 φαίνεται ο κώδικας που χρησιμοποιήθηκε. Στη συνέχεια ο κώδικας περιμένει μέχρι να γίνει η καθολική μεταβλητή *z* ένα. Αυτό θα γίνει, όταν στο νήμα διακοπής για την ολοκλήρωση της μέτρησης αναγνωριστεί μια διακοπή. Αφού ολοκληρωθεί η μέτρηση ελέγχεται, αν έχει γίνει υπερχείλιση στον μετρητή. Το νήμα διακοπής για την υπερχείλιση θα έχει κάνει την τιμή o*verflow* 1 σε περίπτωση υπερχείλισης. Αν δεν υπάρχει υπερχείλιση, μπορεί να διαβαστεί το αποτέλεσμα με μια εντολή διαβάσματος. Το αποτέλεσμα αποθηκεύεται σε μια μεταβλητή ‘*b*’. Από το αποτέλεσμα χρειαζόμαστε μόνο τα 16 μπιτ αφού ο μετρητής μετράει μέχρι το 65535.

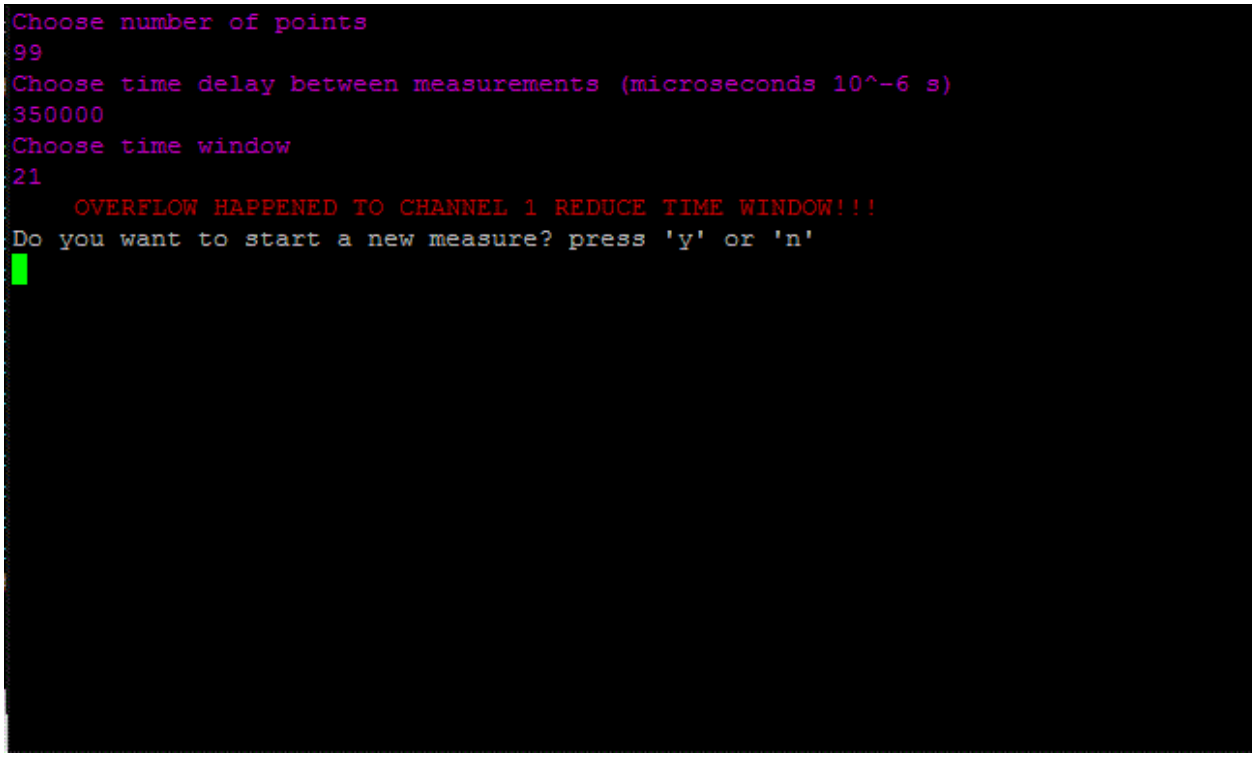
Ακολουθεί ο υπολογισμός της συχνότητας ταλάντωσης με βάση το αποτέλεσμα ‘b’ και το χρονικό παράθυρο που έχει επιλεχθεί. Το αποτέλεσμα αποθηκεύεται στο αρχείο του καναλιού, όπως φαίνεται στην εικόνα 5.23, και το πρόγραμμα προχωράει στη μέτρηση του επόμενου καναλιού. Στις εικόνες 5.25,5.26,5.27 φαίνεται η εκτέλεση του προγράμματος. Στην εικόνα 5.25 φαίνεται η είσοδος των παραμέτρων του συστήματος. Η χρονική καθυστέρηση πρέπει να εισαχθεί σε μs. Στην εικόνα 5.26 φαίνεται η εκτέλεση κατά την διάρκεια των μετρήσεων. Εμφανίζεται η πρόοδος των μετρήσεων. Στην εικόνα 5.27 έχει γίνει υπερχείλιση σε κάποιο από τα κανάλια και η μέτρηση έχει σταματήσει.



Εικόνα 5.25: Εκτέλεση προγράμματος μέτρησης- Είσοδος παραμέτρων

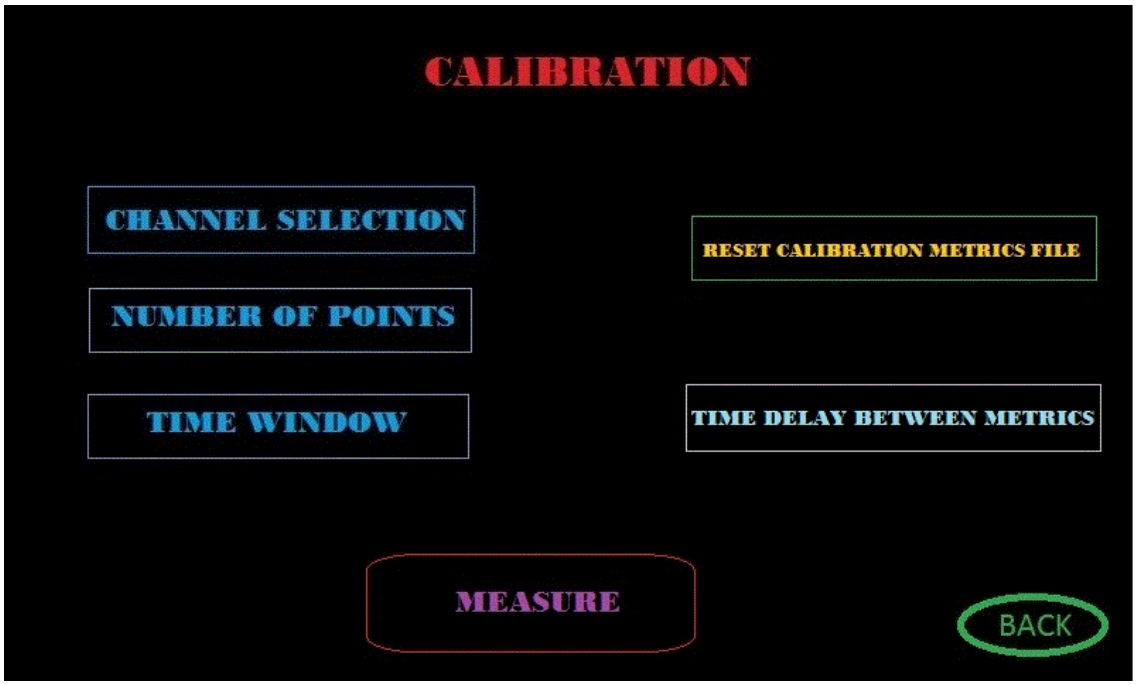


Εικόνα 5.26: Εκτέλεση προγράμματος μέτρησης- Διαδικασία μέτρησης

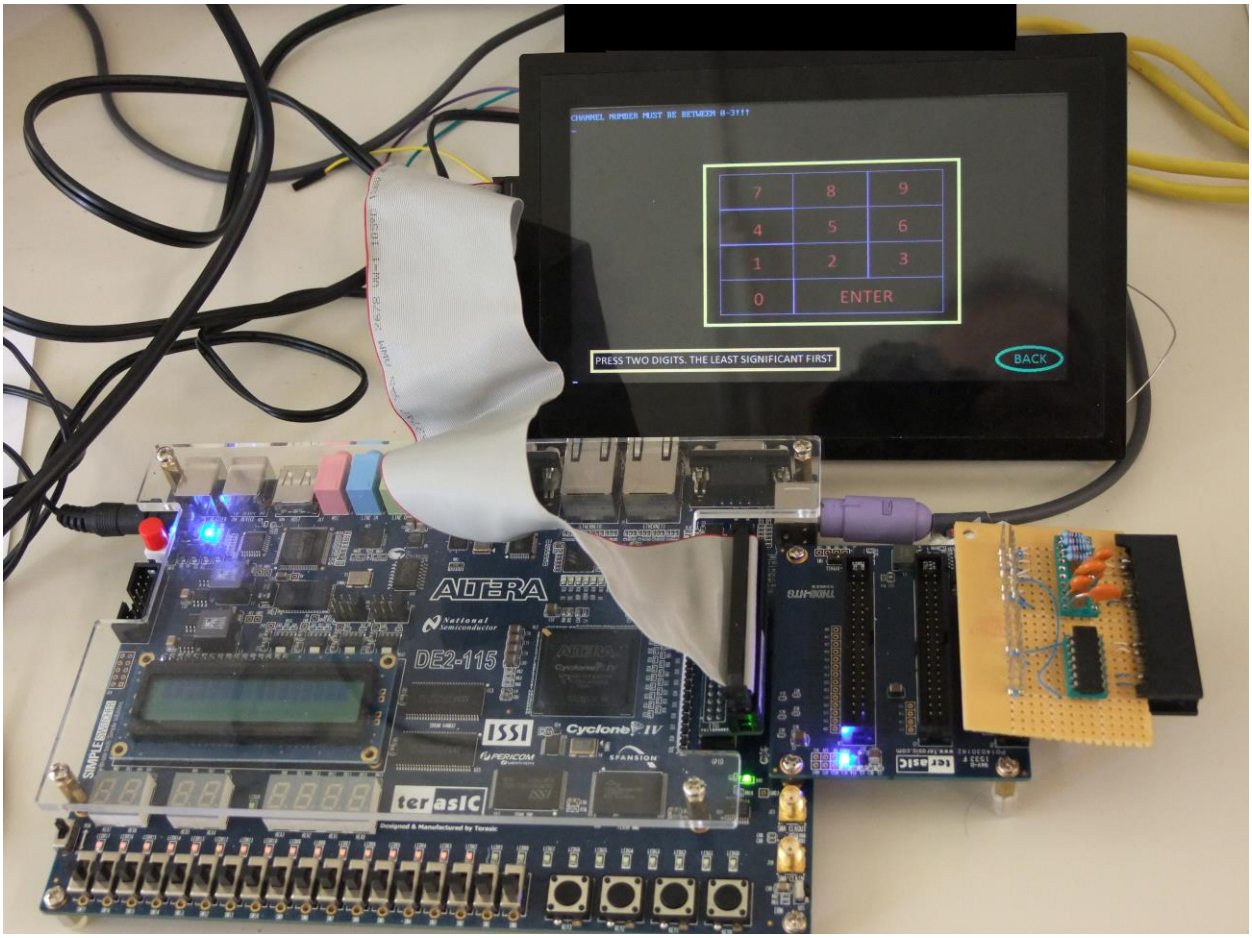


Εικόνα 5.27: Εκτέλεση προγράμματος μέτρησης- Υπερχείλιση

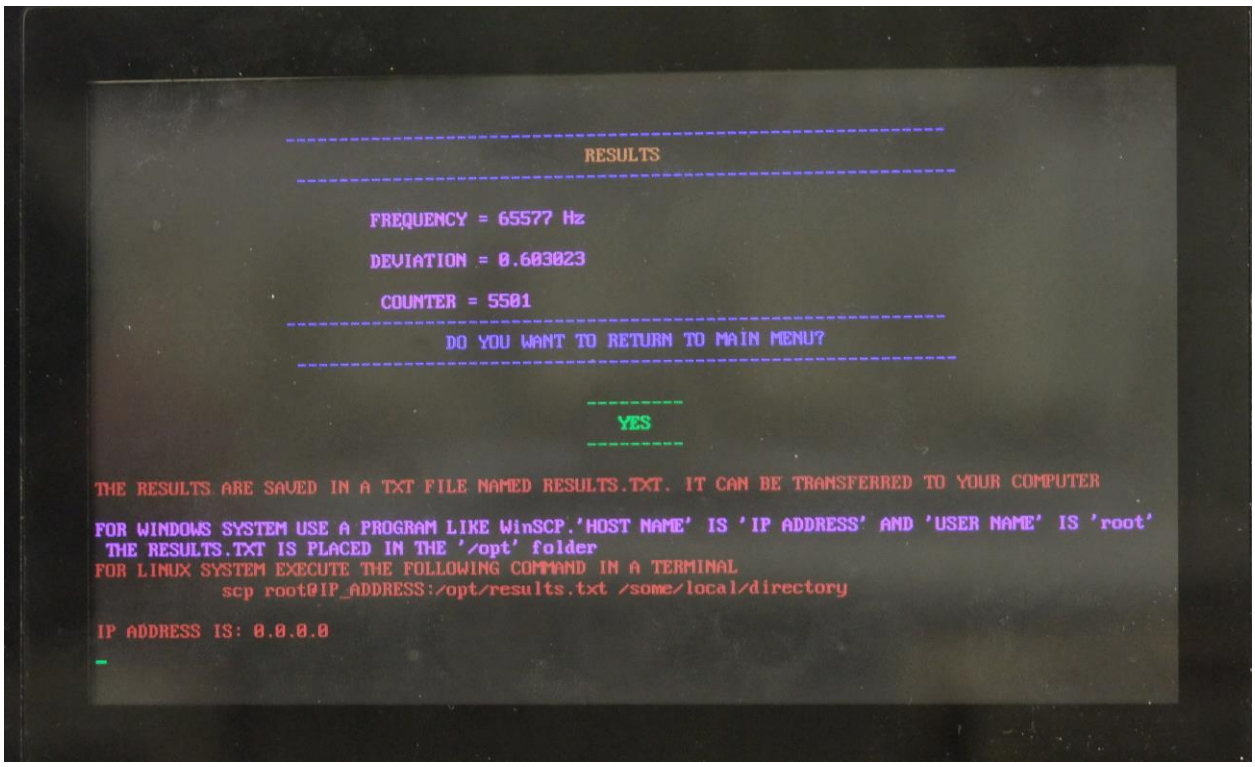
Έχουν δημιουργηθεί και άλλα προγράμματα για τις μετρήσεις. Ένα από αυτά χρησιμοποιεί την λειτουργία 'touch screen της' οθόνης στην περίπτωση που δεν υπάρχει πληκτρολόγιο στο σύστημα. Ο κώδικας αυτού του προγράμματος υπάρχει στο παράρτημα. Στις εικόνες 5.28,5.29 φαίνονται κάποιες λειτουργίες του προγράμματος. Στο μενού βαθμονόμησης επιλέγεται μόνο ένα κανάλι για μετρήσεις. Μπορεί να επιλεχθεί το χρονικό παράθυρο ο αριθμός των σημείων και η χρονική καθυστέρηση μεταξύ των μετρήσεων. Στο τέλος των μετρήσεων εμφανίζεται η μέση τιμή και η διασπορά τη συχνότητας του καναλιού για την διευκόλυνση της βαθμονόμησης (εικόνα 5.30). Το μενού πληκτρολογίου χρησιμεύει για την είσοδο των παραμέτρων χρησιμοποιώντας την 'touch screen' λειτουργία της οθόνης.



Εικόνα 5.28: Πρόγραμμα με 'touch screen' – Μενού βαθμονόμησης



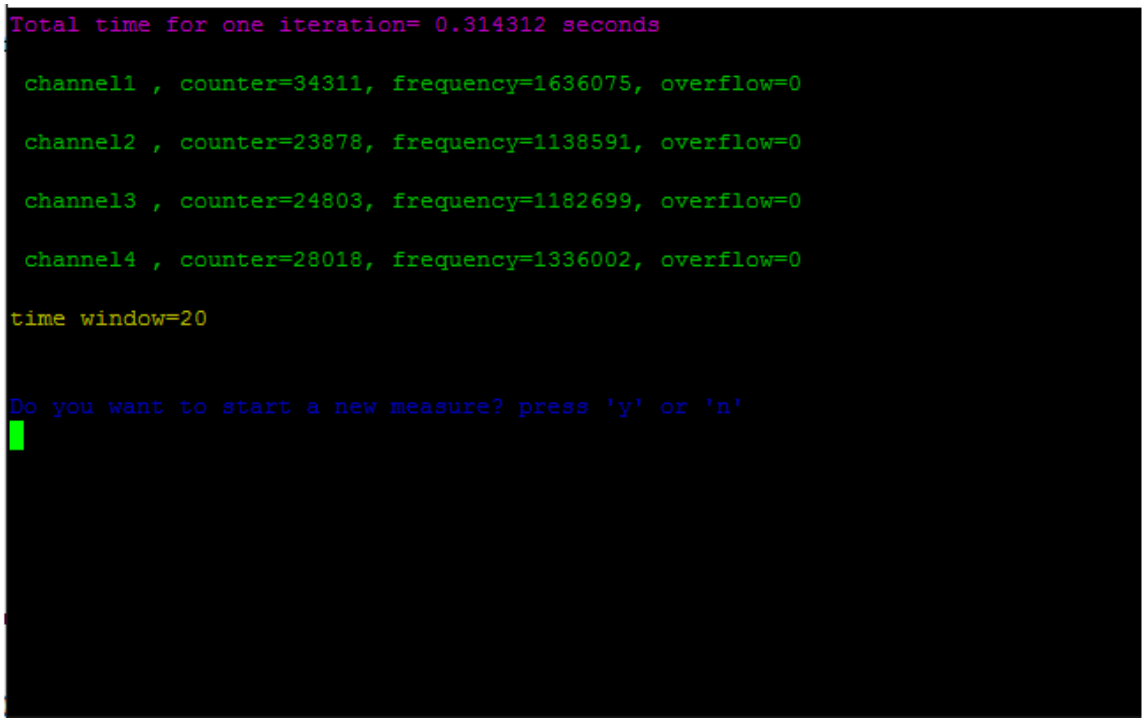
Εικόνα 5.29: Πρόγραμμα με 'touch screen' – Μενού πληκτρολογίου στην SVGA οθόνη



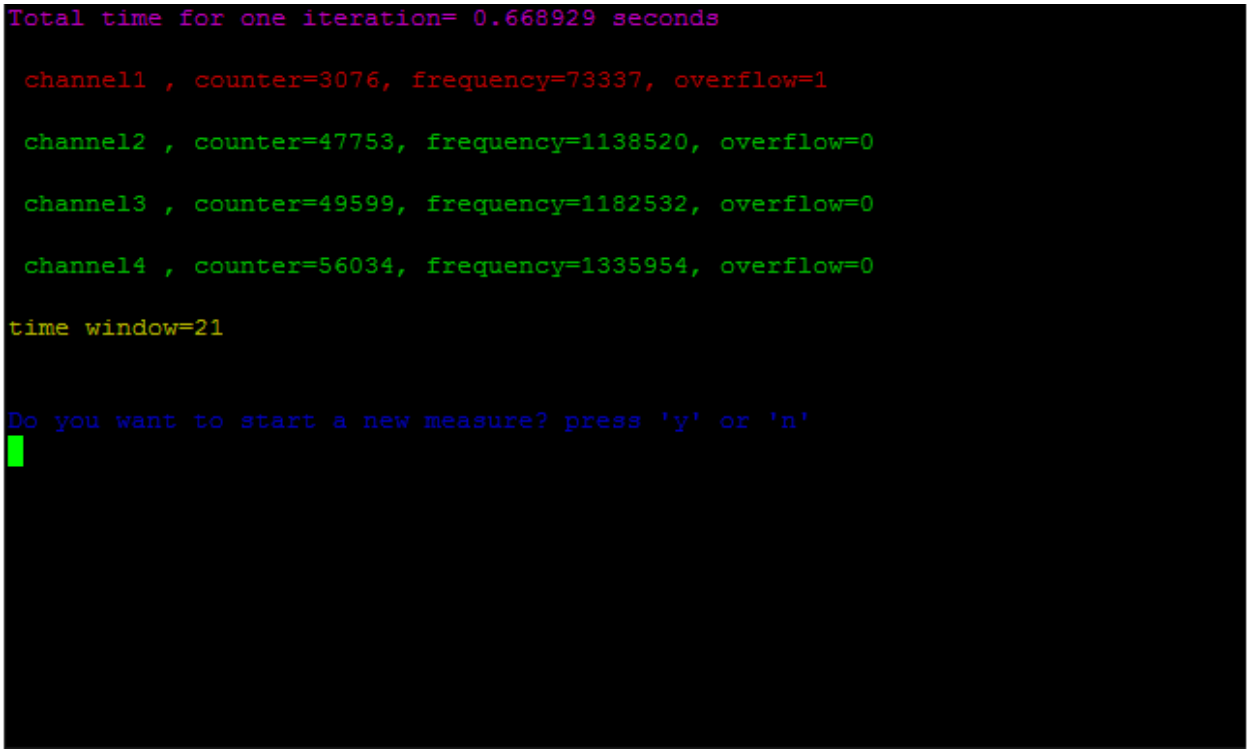
Εικόνα 5.30: Πρόγραμμα με 'touch screen' – Εμφάνιση αποτελέσματος μέτρησης

Ένα ακόμα πρόγραμμα που υλοποιήθηκε χρησιμοποιείται για τον γρήγορο έλεγχο των καναλιών. Στο πρόγραμμα αυτό, επιλέγοντας μόνο το χρονικό παράθυρο, φαίνεται, αν υπάρχει υπερχείλιση σε κάποιο κανάλι και πόσο είναι οι τιμές του μετρητή και της

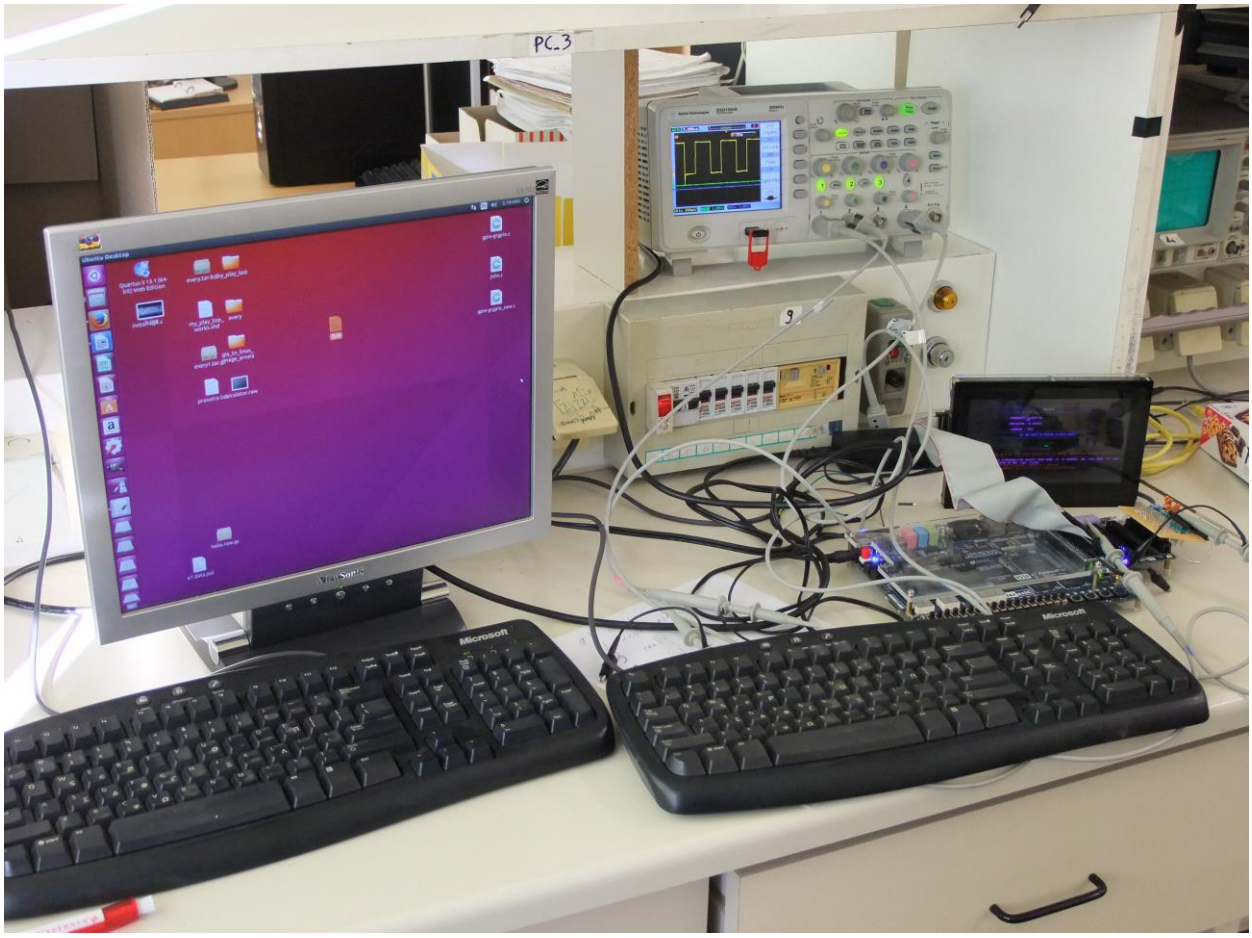
συχνότητας σε μια μόνο μέτρηση. Στην εικόνα 5.31 φαίνεται μια μέτρηση και στα 4 κανάλια με παράμετρο χρονικού παραθύρου 20. Στην οθόνη εμφανίζεται η τιμή του μετρητή, η συχνότητα, αν υπάρχει υπερχείλιση και ο συνολικός χρόνος για το πέρασμα και των τεσσάρων καναλιών. Στην εικόνα 5.32 αυξάνεται κατά ένα η παράμετρος του χρονικού παραθύρου με αποτέλεσμα να διπλασιάζονται οι τιμές των μετρητών. Στο κανάλι 1 δημιουργείται υπερχείλιση.



Εικόνα 5.31: Πρόγραμμα για έλεγχο καναλιών



Εικόνα 5.32: Πρόγραμμα για έλεγχο καναλιών - Υπερχείλιση



Εικόνα 5.33: Περιβάλλον ανάπτυξης του συστήματος

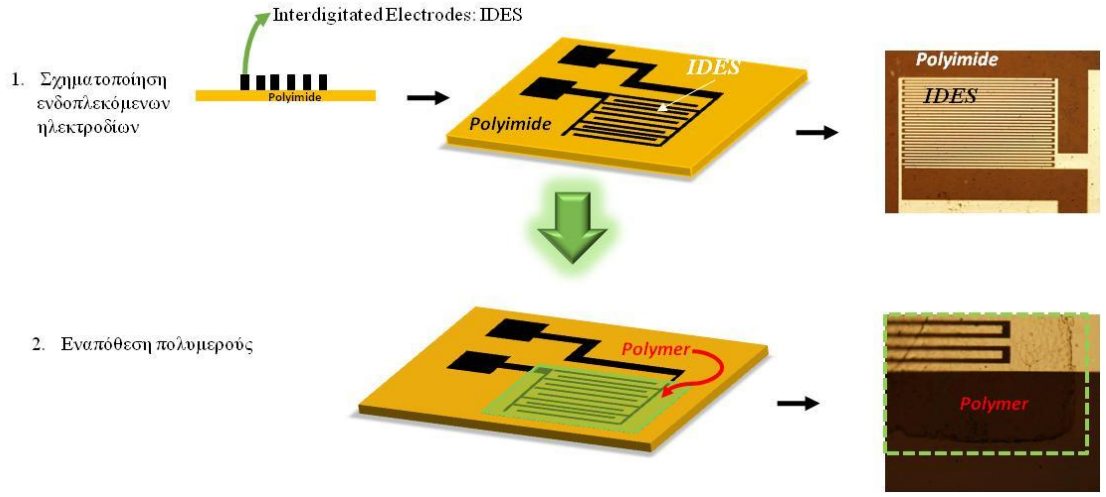
6. ΜΕΤΡΗΣΕΙΣ, ΜΕΛΛΟΝΤΙΚΕΣ ΒΕΛΤΙΩΣΕΙΣ ΚΑΙ ΣΥΜΠΕΡΑΣΜΑΤΑ

6.1 Εισαγωγικά

Στο κεφάλαιο αυτό παρουσιάζεται συνολικά το σύστημα και κάποιες μετρήσεις που ελήφθησαν με αυτό. Οι μετρήσεις που ελήφθησαν είναι χαρακτηριστικές για το πού μπορεί να χρησιμοποιηθεί το σύστημα και τι δυνατότητες έχει γενικά. Το σύστημα είναι ευέλικτο και μπορεί να χρησιμοποιηθεί σε πλήθος εφαρμογών, οι οποίες σχετίζονται με χωρητικούς αισθητήρες και μέτρηση συχνότητας. Μπορούν, φυσικά, να γίνουν κάποιες βελτιώσεις ανάλογα με την εφαρμογή και το σκοπό για τον οποίο χρησιμοποιείται. Αυτές οι βελτιώσεις παρουσιάζονται στη συνέχεια. Στο τέλος του κεφαλαίου παρουσιάζονται τα συμπεράσματα της υλοποίησης της διπλωματικής.

6.2 Μετρήσεις

Στο πλαίσιο της διπλωματικής το σύστημα που υλοποιήθηκε χρησιμοποιήθηκε για την μέτρηση αισθητήρων χωρητικότητας ενδοπλεκόμενων ηλεκτροδίων. Οι αισθητήρες αυτοί έχουν υλοποιηθεί όπως φαίνεται στην εικόνα 6.1.



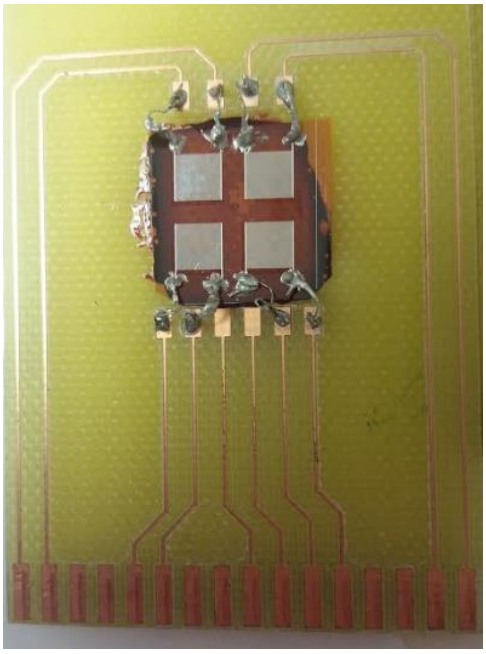
Εικόνα 6.1: Κατασκευή χωρητικών αισθητήρων με ενδοπλεκόμενα ηλεκτρόδια

Αρχικά, γίνεται η σχηματοποίηση των ενδοπλεκόμενων ηλεκτρονίων σε μια πολυϊμιδική επιφάνεια και στη συνέχεια εναποτίθεται ένα στρώμα πολυμερούς. Τα πολυμερή, όταν είναι τοποθετημένα σε ένα χωρητικό αισθητήρα με ενδοπλεκόμενα ηλεκτρόδια, μπορούν να χρησιμοποιηθούν για την ανίχνευση VOCs (αναλυτών) στον αέρα. Οι αναλύτες απορροφώνται από το στρώμα του πολυμερούς με αποτέλεσμα να αλλάζουν οι φυσικές ιδιότητες του στρώματος και να αλλάζουν τα παρακάτω χαρακτηριστικά της χωρητικότητας του αισθητήρα:

1) Όγκος

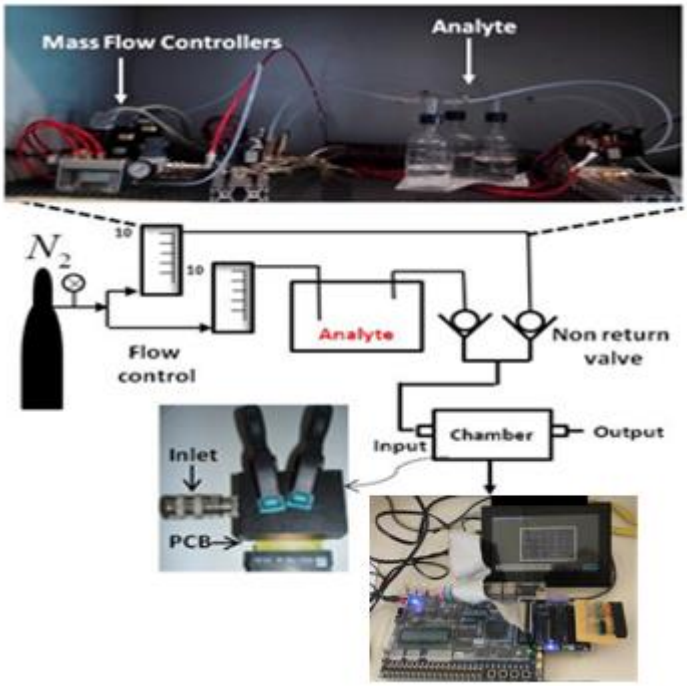
2) Διηλεκτρική σταθερά

Για την μέτρηση χρησιμοποιήθηκαν 4 αισθητήρες όπου ο καθένας περιέχει διαφορετικό πολυμερές. Τα πολυμερή που χρησιμοποιήθηκαν είναι τα PMMA (Poly methyl methacrylate) , EPR (Ethylene propylene rubber), PHS (Polymer Hermetic Seal) και pHEMA (Polyhydroxyethylmethacrylate). Στην εικόνα 6.2 φαίνονται οι 4 αισθητήρες τοποθετημένοι σε ένα PCB.



Εικόνα 6.2: PCB με τους αισθητήρες που χρησιμοποιήθηκαν για την μέτρηση

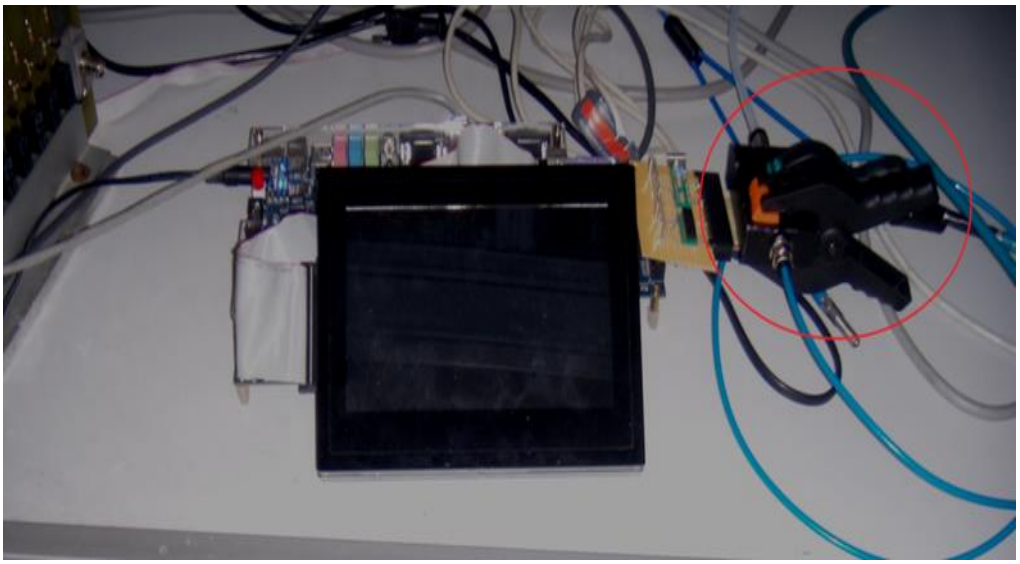
Η διάταξη (setup) της μέτρησης φαίνεται στην εικόνα 6.3. Η διάταξη ουσιαστικά είναι μια μονάδα διανομής αερίων η οποία αποτελείται από τις φιάλες, μέσα στις οποίες περιέχονται οι αναλύτες, δύο ροόμετρα με τα οποία ρυθμίζεται η συγκέντρωση των ατμών των αναλυτών και του ξηρού αζώτου, και ένα μεταλλικό θάλαμο στον οποίο αναμιγνύεται το μίγμα αζώτου και πτητικής ένωσης. Κατά την διάρκεια της μέτρησης οι αισθητήρες βρίσκονται μέσα στο μεταλλικό θάλαμο και το PCB με τους αισθητήρες είναι συνδεδεμένο με το εξωτερικό κύκλωμα του συστήματος μέτρησης.



Εικόνα 6.3: Μετρητική διάταξη

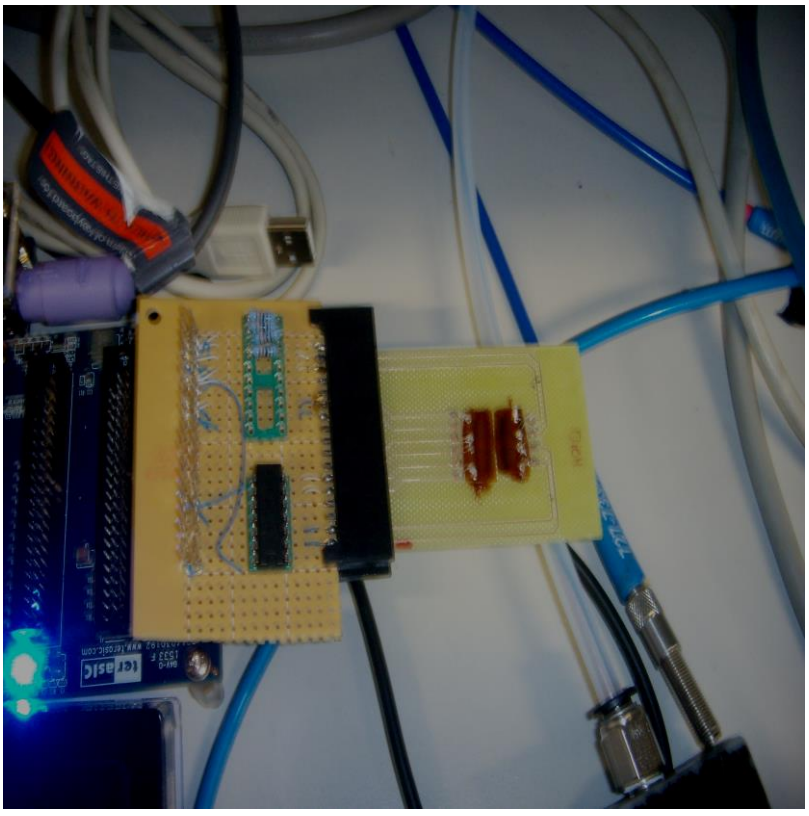
Η μέτρηση γίνεται ως εξής: το άζωτο ρέει καθόλη την διάρκεια της μέτρησης και χωρίζεται σε δύο μέρη. Το ένα μέρος περνά από τις φιάλες με τους αναλύτες και από ένα υπολογιστή μπορεί να ρυθμιστεί σε ποια φιάλη με συγκεκριμένο αναλύτη θα δημιουργηθούν φυσαλίδες ώστε να δημιουργηθούν ατμοί. Όταν το άζωτο περάσει πάνω

από τις φιάλες, θα παρασύρει τα μόρια του αναλύτη και στην συνέχεια θα ενωθεί με την δεύτερη ροή αζώτου για να επιτευχθεί η επιθυμητή συγκέντρωση. Το μίγμα στο τέλος εισέρχεται στο μεταλλικό θάλαμο με τους αισθητήρες. Στην εικόνα 6.4 φαίνεται πώς έχει συνδεθεί το σύστημα της διπλωματικής με την μετρητική διάταξη.

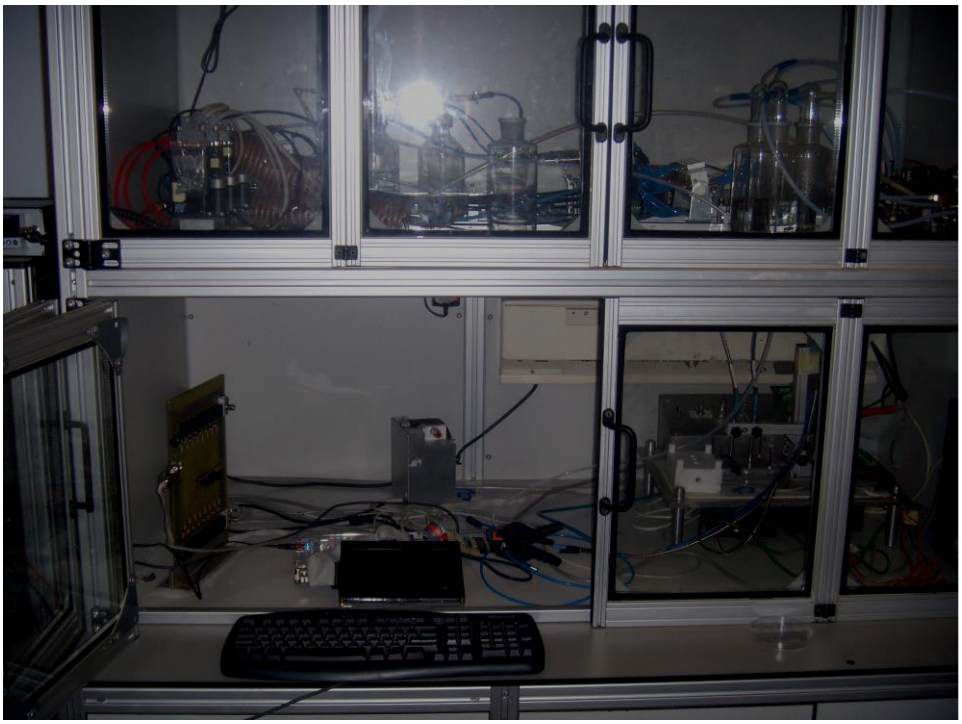


Εικόνα 6.4: Το σύστημα της διπλωματικής συνδεδεμένο με την μετρητική διάταξη

Στην εικόνα 6.5 φαίνεται η σύνδεση του PCB με τους αισθητήρες με το εξωτερικό κύκλωμα του συστήματος και στην εικόνα 6.6 φαίνεται μια γενική άποψη του χώρου μετρήσεων.

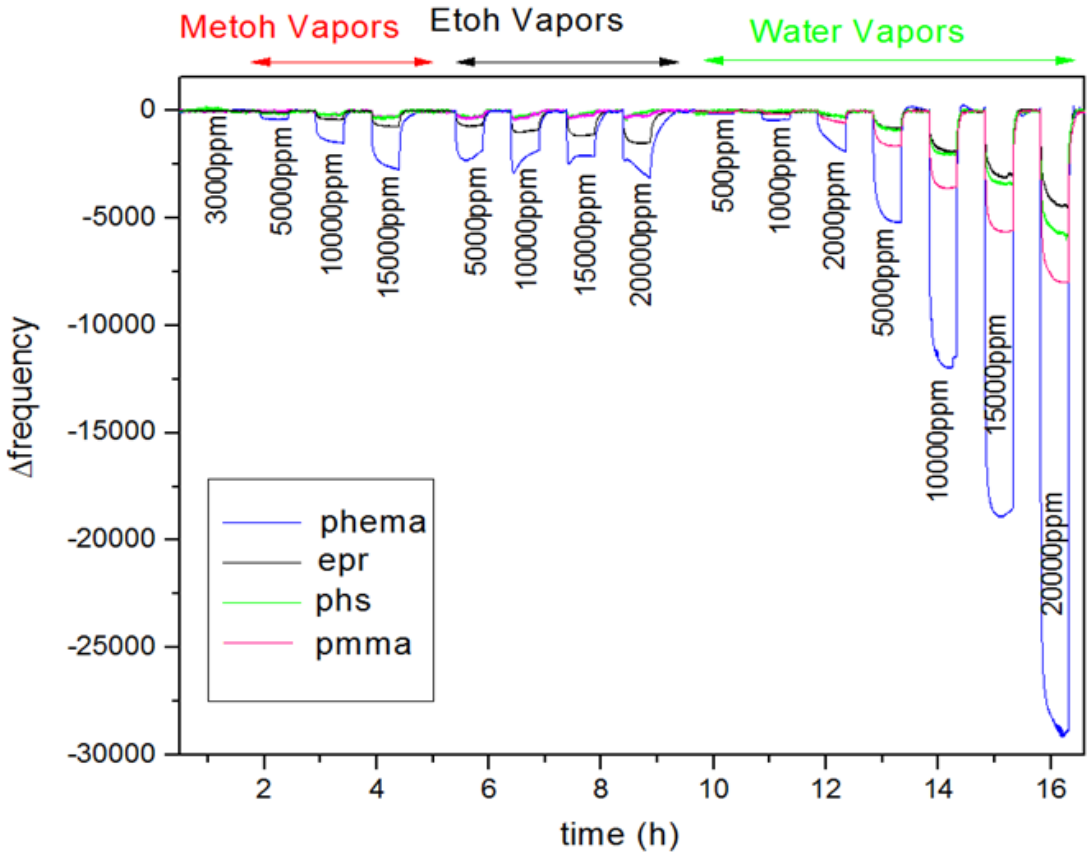


Εικόνα 6.5: Σύνδεση του PCB με τους αισθητήρες με το σύστημα της διπλωματικής



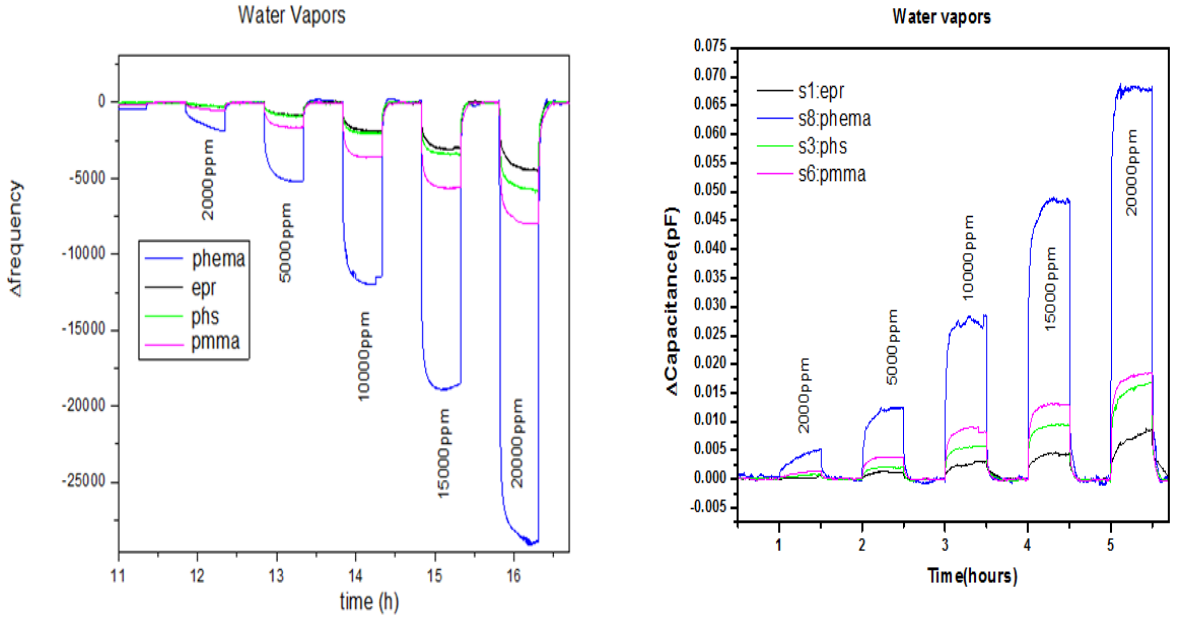
Εικόνα 6.6: Γενική άποψη του χώρου μετρήσεων

Για την μέτρηση που λήφθηκε χρησιμοποιήθηκαν κατά σειρά οι αναλύτες μεθανόλη (metoh), αιθανόλη (etoh) και νερό (H_2O). Η μέτρηση διήρκεσε 17 ώρες και χρησιμοποιήθηκαν διάφορες συγκεντρώσεις από κάθε αέριο για να φανεί πώς αντιδράει ο κάθε αισθητήρας-πολυμερές. Στην εικόνα 6.7 φαίνεται το αποτέλεσμα της μέτρησης χρησιμοποιώντας το σύστημα της διπλωματικής.



Εικόνα 6.7: Μεταβολές της συχνότητας της μέτρησης (Δf/h)

Στην εικόνα 6.7 φαίνονται οι μεταβολές με αρχική συχνότητα των αισθητήρων, όταν ρέει μόνο άζωτο. Η συχνότητα αυτή εξαρτάται από την αρχική χωρητικότητα των αισθητήρων και από τις παρασιτικές χωρητικότητες του εξωτερικού κυκλώματος. Φαίνεται ότι το 'phemma' αντιδράει και με τους 3 αναλύτες, το 'pmma' και το 'phs' αντιδρά κυρίως με το νερό και το 'epr' αντιδρά και στους 3 αναλύτες αλλά με μικρότερη ένταση από το 'phemma'. Επίσης, παρατηρείται ότι το σύστημα μπορεί να εντοπίσει τις μεταβολές ακόμα και στις μικρές συγκετρώσεις, οι οποίες μεταβάλλουν την συχνότητα σε μερικές δεκάδες Hz. Στην εικόνα 6.8 φαίνονται οι μεταβολές της συχνότητας και της χωρητικότητας με αναλύτη το νερό. Οι χωρητικές μεταβολές (δεξιό διάγραμμα εικόνας 6.8) έχουν ληφθεί από ένα σύστημα με γέφυρα το οποίο χρησιμοποιήθηκε για την εξακρίβωση της σωστής λειτουργίας του συστήματος της διπλωματικής.



Εικόνα 6.8: Μεταβολές της συχνότητας και της χωρητικότητας με αναλύτη το νερό

Από την εικόνα 6.8 μπορεί να φανεί η σωστή λειτουργία του συστήματος. Οι μεταβολές της συχνότητας ακολουθούν τις προβλεπόμενες μεταβολές της χωρητικότητας. Το διάγραμμα των συχνοτήτων είναι αντιστραμμένο καθώς, όταν αυξάνει η χωρητικότητα, μικραίνει η συχνότητα. Οι κυματομορφές της συχνότητας και της χωρητικότητας έχουν παρόμοια μορφή και οι σχετικές μεταβολές μεταξύ των αισθητήρων επιβεβαιώνονται. Το 'phemma' αντιδρά περισσότερο στο νερό, μετά το 'pmma', μετά το 'phs' και μετά το 'epr'. Για τον ακριβή προσδιορισμό των μεταβολών και την αντιστοίχιση της μεταβολής της συχνότητας σε πόση μεταβολή χωρητικότητας αντιστοιχεί, πρέπει να γίνει βαθμονόμηση του συστήματος.

Το σύστημα της διπλωματικής είναι ευέλικτο και μπορεί να χρησιμοποιηθεί σε αρκετά είδη μετρήσεων. Μπορούν να συνδεθούν όλα τα είδη χωρητικών αισθητήρων και να εντοπιστεί η συχνότητα λειτουργία τους. Δεν είναι ανάγκη να είναι συνδεδεμένοι 4 αισθητήρες στο σύστημα. Χρησιμοποιώντας ένα αισθητήρα κάθε φορά μπορεί να γίνει βαθμονόμηση με γνωστές χωρητικότητες, δηλαδή να εντοπιστεί η χωρητικότητα σε ποια συχνότητα αντιστοιχεί. Επίσης, μπορεί να γίνει απευθείας βαθμονόμηση του συστήματος για γνωστές τιμές του μετρούμενου μεγέθους ώστε να γίνεται εφικτή η απευθείας αντιστοίχιση της συχνότητας ταλάντωσης με την τιμή του μετρούμενου μεγέθους.

Όπως έχει ειπωθεί και στο κεφάλαιο 5 τα αποτελέσματα των μετρήσεων αποθηκεύονται σε αρχεία. Αυτά τα αρχεία μπορούν να διαβαστούν κατά την διάρκεια των μετρήσεων

και έτσι να υπάρχει άμεση ανασκόπηση των αποτελεσμάτων. Ταυτόχρονα τα αρχεία των αποτελεσμάτων μπορούν να μεταφερθούν μέσω SSH ή SFTP σύνδεσης σε ένα άλλο σύστημα. Αυτές οι λειτουργίες διευκολύνουν την επεξεργασία των δεδομένων και επιτρέπουν την άμεση επαφή του χρήστη με το σύστημα και τους αισθητήρες. Έτσι το σύστημα δεν είναι απλά ένα εργαλείο μέτρησης συχνοτήτων αλλά ένα σύστημα το οποίο διαθέτει ενσωματωμένη νοημοσύνη ικανή να αντιλαμβάνεται το περιβάλλον και να αλληλεπιδρά με τον χρήστη σε ‘σχεδόν’ πραγματικό χρόνο.

6.3 Μελλοντικές βελτιώσεις

Το σύστημα που έχει υλοποιηθεί μπορεί να υποστεί πολλές βελτιώσεις. Καθώς έχει χρησιμοποιηθεί ένας επεξεργαστής μέσω λογισμικού ανοικτού τύπου αυτές οι βελτιώσεις μπορούν να γίνουν εύκολα και γρήγορα. Αρχικά, μπορεί να χρησιμοποιηθεί ένα άλλο FPGA το οποίο έχει περισσότερους πόρους σε μνήμη. Κάτι τέτοιο επιτρέπει την καλύτερη συνολική λειτουργία του συστήματος. Επίσης, μπορεί να χρησιμοποιηθεί μια συσκευή FPGA η οποία διαθέτει εσωτερικά Schmitt trigger και να μην χρειάζεται εξωτερικό κύκλωμα. Ως προς το λειτουργικό σύστημα μπορεί να χρησιμοποιηθεί ένα λειτουργικό σύστημα πραγματικού χρόνου (RTOS). Ένα τέτοιο σύστημα επιτρέπει την πιο γρήγορη αλληλεπίδραση του χρήστη με το σύστημα σε πραγματικό χρόνο και οι λειτουργίες μέτρησης μπορούν να γίνουν πιο γρήγορα. Στο [44] παρουσιάζεται η διαφορά στους χρόνους μεταξύ ενός RTOS και ενός απλού Linux πυρήνα (το RTOS Linux είναι ως και 3 φορές πιο γρήγορο στον εντοπισμό διακοπών). Εναλλακτικά μπορεί να χρησιμοποιηθεί ένα πιο πλούσιο λειτουργικό σύστημα Linux (ο LEON3 υποστηρίζει μόνο Linux συστήματα προς το παρόν) με περισσότερες λειτουργίες ακόμα και με γραφικό περιβάλλον για πιο εύκολη χρήση. Για την μείωση του θορύβου και των παρασιτικών χωρητικοτήτων μπορεί να χρησιμοποιηθεί ένα PCB. Ακόμα για τις διακοπές του κυκλώματος μέτρησης και την σύνδεση του με το επίπεδο του πυρήνα του Linux μπορούν να υλοποιηθούν οδηγοί και να μην χρειάζεται η συνάρτηση ‘’mmap’’ και τα παράλληλα νήματα (βλ. κεφάλαιο 5). Στο επίπεδο του πυρήνα θα μπορούσε να επιτευχθεί πιο γρήγορη ανταπόκριση του συστήματος με το κύκλωμα μετρήσεων και πιο γρήγορη εκτέλεση των μετρήσεων. Επίσης, ως προς την ανάλυση του συστήματος και των μετρήσεων μπορεί να χρησιμοποιηθεί μεγαλύτερος μετρητής ως και 32-μπιτ για τον συγκεκριμένο επεξεργαστή, αν και για μεγάλα χρονικά παράθυρα θα απαιτείται αρκετός χρόνος για κάθε μέτρηση. Παράλληλα μπορούν να προστεθούν και άλλοι επεξεργαστές στο σύστημα για να γίνει πιο αποδοτικό. Για παράδειγμα θα μπορούσε κάθε κανάλι να έχει τον δικό του επεξεργαστή και οι μετρήσεις να γίνονται παράλληλα ή το κάθε νήμα διακοπής να τρέχει σε ξεχωριστό επεξεργαστή για καλύτερη απόδοση. Τέλος θα μπορούσε να προστεθούν αρκετές λειτουργίες στο σύστημα όπως λειτουργία ‘wifi’, διεπαφή USB αλλά και το σύστημα να γίνει πλήρως αυτόνομο χρησιμοποιώντας για τροφοδοσία μπαταρία ή άλλες μορφές ενέργειας.

6.4 Συμπεράσματα

Όπως φαίνεται από τις υλοποιήσεις του κεφάλαιου 1.7 οι συσκευές FPGA και οι επεξεργαστές που υλοποιούνται μέσω λογισμικού χρησιμοποιούνται πάρα πολύ στην έρευνα σε εφαρμογές με αισθητήρες. Υπάρχουν, επίσης, κάποιες υλοποιήσεις που συνδυάζουν και λειτουργικά συστήματα αλλά προς το παρόν δεν υπάρχει ευρεία χρήση σε συνδυασμό με επεξεργαστές και αισθητήρες, κυρίως λόγω απόδοσης και κατανάλωσης πόρων ενός τέτοιου συστήματος. Προτιμάται η σύνδεση αισθητήρων απευθείας στον επεξεργαστή κυρίως σε WSN συστήματα. Παρόλα αυτά, όπως ειπώθηκε και σε προηγούμενα κεφάλαια, το λειτουργικό σύστημα μπορεί να προσφέρει πάρα πολλές δυνατότητες και η ενσωμάτωση του στο επεξεργαστικό σύστημα μπορεί να γίνει πλέον πολύ εύκολα με τα σημερινά εργαλεία. Γενικά, υπάρχει συμβιβασμός(trade-off) μεταξύ της κατανάλωσης πόρων και το πλήθος των λειτουργιών

που περιλαμβάνει το εκάστοτε σύστημα. Στη συγκεκριμένη υλοποίηση κρίθηκε ότι ένα λειτουργικό σύστημα θα είναι πολύ χρήσιμο για τον σκοπό της εφαρμογής. Η SVGA, το πληκτρολόγιο, το περιβάλλον Linux και οι λειτουργίες δικτύου προσδίδουν στο σύστημα ένα είδος αυτονομίας και επιτρέπουν την εύκολη διαχείριση των αισθητήρων. Το σύστημα ούτως η άλλως έχει παροχή ρεύματος μέσω καλωδίου και δεν υπάρχει πρόβλημα ως προς την κατανάλωση πόρων.

Στο [37] είχε υλοποιηθεί η προηγούμενη έκδοση του συστήματος. Ο σκοπός της διπλωματικής ήταν αυτό το σύστημα να συνδεθεί με έναν επεξεργαστή και ένα λειτουργικό σύστημα ώστε να μπορεί ο χρήστης να έχει πιο εύκολη και αποδοτική συνεργασία με τους χωρητικούς αισθητήρες. Αυτός ο στόχος επετεύχθη με την παρούσα διπλωματική. Υλοποιήθηκε ένα ευέλικτο, αυτόνομο και αποδοτικό σύστημα το οποίο μπορεί να προσαρμοστεί σε διαφορετικές συνθήκες και περιβάλλοντα. Φυσικά στο σύστημα μπορούν να γίνουν ακόμα πολλές βελτιώσεις, όπως ειπώθηκε στο κεφάλαιο 6.3 και να προστεθούν αρκετές λειτουργίες οι οποίες πλέον θα μπορούν να ενσωματωθούν πολύ εύκολα στο παρόν σύστημα. Στην παρούσα διπλωματική εξερευνήθηκαν οι δυνατότητες και προοπτικές εξέλιξης του συστήματος και φάνηκε ότι υπάρχουν ακόμα πολλά περιθώρια βελτίωσης. Ο σκοπός της διπλωματικής δεν ήταν να υλοποιηθεί ένα τέλειο σύστημα για μια συγκεκριμένη εφαρμογή αλλά ένα ευέλικτο και εύκολα προσαρμοζόμενο σύστημα το οποίο θα μπορεί να χρησιμοποιηθεί σε πολλές εφαρμογές.

Επιπλέον, στη διπλωματική εξερευνήθηκαν αρκετές έξυπνες λειτουργίες οι οποίες μπορούν να συνδυαστούν με τους αισθητήρες. Με την χρήση των αισθητήρων και του επεξεργαστή μπορεί να γίνει βαθμονόμηση του συστήματος, υπολογίζοντας τον μέσο όρο, την διασπορά των μετρήσεων κ.τ.λ., να ενημερώνεται ο χρήστης στην ολοκλήρωση της μέτρησης και σε περίπτωση υπερχείλισης, να μεταφέρονται αυτόματα τα αποτελέσματα μέσω SSH σύνδεσης κ.α. Η συνεργασία του επεξεργαστή με το λειτουργικό σύστημα επιτρέπει την ενσωμάτωση πολλών ακόμα έξυπνων λειτουργιών στο σύστημα. Ανάλογα με την εκάστοτε εφαρμογή, μπορούν να ενσωματωθούν και οι αντίστοιχες έξυπνες λειτουργίες που απαιτούνται.

Ως προς την ποιότητα και την ευκρίνεια των μετρήσεων και αυτή είναι ευέλικτη. Καθορίζεται από τον χρήστη πόση ανάλυση και ευκρίνεια χρειάζεται αλλάζοντας το χρονικό παράθυρο. Αν χρειάζεται υψηλή ευκρίνεια, επιλέγεται μεγαλύτερο χρονικό παράθυρο με κόστος τον χρόνο στις μετρήσεις. Η ευελιξία στο χρονικό παράθυρο επιτρέπει την μέτρηση μεγάλου εύρους συχνοτήτων από μερικά kHz ως και μερικά MHz χωρίς να δημιουργείται υπερχείλιση στον μετρητή. Επίσης, οι μετρήσεις είναι σταθερές και αξιόπιστες με την χρήση του Schmitt trigger.

Ως συμπέρασμα της διπλωματικής μπορεί να εξαχθεί. επίσης, ότι η χρήση του επεξεργαστή LEON3 παρέχει σε ένα σύστημα πάρα πολλές δυνατότητες. Η βιβλιοθήκη GRLIB και τα προγράμματα LINUXBUILD και GRMON επιτρέπουν την δημιουργία ενός πλήρους συστήματος εύκολα και με αποδοτικό συν-σχεδιασμό. Υπάρχουν πολλοί επεξεργαστές που μπορούν να επιλεχθούν για την χρήση ενός τέτοιου συστήματος σαν της διπλωματικής και οι οποίοι μπορούν να παρέχουν περισσότερη επεξεργαστική ισχύ και δυνατότητες αλλά αυτοί είτε υλοποιούνται απευθείας στο υλικό είτε είναι κλειστού τύπου (Microblaze, Nios ii) και έτσι δεν μπορούν να παρέχουν την ίδια ευελιξία με τον LEON3. Ο δίαυλος AMBA που χρησιμοποιείται με τον LEON3 είναι, επίσης, αποδοτικός, απλός και επιτρέπει την εύκολη διασύνδεση περιφερειακών με τον επεξεργαστή.

Τέλος, ως προς τους μετρητές συχνότητας σε συνδυασμό με την διεπαφή χωρητικών αισθητήρων διαπιστώνεται ότι αποτελούν μια αξιόπιστη λύση για την μετατροπή της χωρητικότητας σε συχνότητα. Αυτή η μετατροπή επιτρέπει την παρακολούθηση της

μεταβολής στη χωρητικότητα ενός αισθητήρα και σε συνδυασμό με άλλες έξυπνες λειτουργίες μπορεί να καθοριστεί η ακριβής τιμή της χωρητικότητας του αισθητήρα. Η διεπαφή που υλοποιήθηκε προσφέρει υψηλή ευελιξία ως προς τους αισθητήρες που μπορούν να χρησιμοποιηθούν και τις μετρήσεις που μπορούν να ληφθούν.

Συνολικά, το συμπέρασμα της διπλωματικής είναι ότι συνδυασμός διεπαφών αισθητήρων με ένα επεξεργαστή και ένα λειτουργικό σύστημα οδηγεί σε συστήματα τα οποία μπορούν να βοηθήσουν στη καλύτερη κατανόηση του περιβάλλοντος και η έρευνα θα πρέπει να συνεχιστεί σε αυτόν τον τομέα. Υπάρχουν πολλές δυνατότητες βελτίωσης αυτών των συστημάτων όσο η τεχνολογία εξελίσσεται και βελτιώνονται οι αισθητήρες, η τεχνολογία των συσκευών αλλά και τα σχεδιαστικά εργαλεία. Οι “έξυπνοι“ αισθητήρες έχουν παρόν αλλά, όπως φαίνεται, θα έχουν και πολύ μέλλον.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
accumulator	συσσωρευτής
adapter	προσαρμογέας
address space	χώρος διευθύνσεων
advanced on-chip debug support	προηγμένη αποσφαλμάτωση πάνω στο τσιπ
amplifier	ενισχυτής
associativity	συσχέτιση
attenuator	εξασθενητής
auto zero	αυτόματος μηδενισμός
bandwidth	εύρος ζώνης
barrel shifter	ολισθητής
baud rate	ρυθμός λειτουργίας
bidirectional	διπλής κατεύθυνσης
bit depth	βάθος σε μπιτ
bit-rate	ρυθμός μετάδοσης των μπιτ
breakpoint trap	παγίδα διακοπής
breakpoint/watchpoint management	διαχείριση σημείων διακοπής και ελέγχου
buffer	ενδιάμεση μνήμη
bus interface	διεπαφή διαύλου
bypass	παράκαμψη
cache	κρυφή μνήμη
calibration	βαθμονόμηση
chip	τσιπ
chip-to-chip interface	διασύνδεση του συστήματος
clock design	σχεδιασμός ρολογιού
co-design	συν-σχεδιασμός
communication hub	κόμβος επικοινωνίας
communication link	σύνδεσμος επικοινωνίας
compiler	μεταγλωττιστής
component	δομικό στοιχείο
connector	βύσμα
controller	ελεγκτής
co-processor unit	μονάδα συνεπεξεργαστή
core	πυρήνας
counter	μετρητής
counting register	καταχωρητής του μετρητή
coupling	σύζευξη
crosstalk	σύζευξη
custom	κατά προτίμηση
custom co-processor	κατά προτίμηση συνεπεξεργαστής
data	δεδομένα
data frame	παράθυρο δεδομένων
debug monitors	εργαλεία αποσφαλμάτωσης
debug Support Unit	μονάδα αποσφαλμάτωσης
decrementing	μειούμενος
default	προκαθορισμένη τιμή
depth of pipeline	βάθος διασωλήνωσης

design flow	ροή σχεδίασης
device driver	οδηγός συσκευής
digital signal processing	επεξεργασία ψηφιακού σήματος
divider	διαιρέτης
documentation	οδηγίες χρήσης
double-word write-buffer	διπλής λέξης buffer-εγγραφής
drain current	ρεύμα διαρροής
driver	οδηγητής
electrical interface	ηλεκτρική διεπαφή
embedded processing	ενσωματωμένη επεξεργασία
end-of-life	περιθώριο ζωής
event Counter	μετρητής γεγονότων
expansion Header	βύσμα επέκτασης
fault-tolerance	ανοχή σε σφάλματα
female	θηλυκή
firm processor	Επεξεργαστής μέσω λογισμικού κλειστού τύπου
floating-point unit	μονάδα κινητής υποδιαστολής
fully synchronous	πλήρως συνχρονισμένος
gadget	καταναλωτική συσκευή
gate	πύλη
gateway	πύλη
hard processor	Επεξεργαστής απευθείας στο υλικό
hardware	υλικό
hardware multiplier	πολλαπλασιαστής υλικού
host	εξυπηρετητής
idle	ανενεργός
informed coding	ενημερωμένη κωδικοποίηση
Instruction	εντολή
instruction set	σύνολο εντολών
integer unit	μονάδα ακεραίων
interface	κύκλωμα διασύνδεσης
interrupt	διακοπή
interrupt interface	διεπαφή διακοπών
interrupt routing	δρομολόγηση των διακοπών
interrupt vector	διάνυσμα διακοπών
kernel	πυρήνας
level/edge	επιπέδου/κορυφής
line size	μέγεθος γραμμής
line-refill	γέμισμα γραμμών
logic synthesis	λογική σύνθεση
main gate	κεντρική πύλη
mapping	αντιστοίχιση
margin	όρια
master	κύριος
matching	συμβατότητα
modification	μετατροπή
modular	σπονδυλωτή δομή
module	τμήμα
mouse	ποντίκι υπολογιστή

multiplier	πολλαπλασιαστής
multi-processor	πολύ-επεξεργαστής
multi-touch gesture	κινήσεων πολλαπλής επαφής
non-intrusive	μη παρεμβατική
odd parity	μονή ισοτιμία
off	κλείσιμο
on	άνοιγμα
on-chip	πάνω στο τσιπ
overflow	υπερχείλιση
parity	ισοτιμία
part	κομμάτι
pin	ακροδέκτης
pixel	εικονοστοιχείο
polarity	πολικότητα
portability	φορητότητα
positive edge-triggering	διέγερση θετικού μετώπου
power-down mode	λειτουργία χαμηλής κατανάλωσης
programmable logic	προγραμματιστική λογική
push button	πιεζόμενο κουμπί
ramp	τριγωνικό σήμα εισόδου
reciprocal	αντίστροφος
reconfigurable	αναδιαρθρώσιμο
refresh rate	ρυθμός ανανέωσης
register-file	φάκελος καταχωρητή
reprogrammable hardware	επανα-προγραμματιζόμενου υλικού
resolution	ανάλυση
ring oscillator	ταλαντωτής δακτυλίου
robust	δυναμικός
routing	δρομολόγηση
scalable	επεκτάσιμος
self-calibration	αυτόματη βαθμονόμηση
Self-health	αυτό-διάγνωση της υγείας
self-identification	αυτοπροσδιορισμού
semiconductor devices	ημι-αγωγικές συσκευές
sensing elements	στοιχεία αισθητήρων
serial Configuration device	συσκευή σειριακής ρύθμισης
setup	διάταξη
shell	κέλυφος
shift registers	καταχωρητές μετατόπισης
simulator	προσομοιωτής
single ended	απλής απολήξεως
single-edge	μονής κορυφής
slave	Υποτελής
Slide switch	διακόπτης
smart sensor	“Έξυπνος” αισθητήρας
soft processor	Επεξεργαστής μέσω λογισμικού ανοικτού τύπου
software	λογισμικό
software debugging	αποσφαλμάτωση λογισμικού
source	πηγή

standard	πρότυπο
state of the art	τελευταίες εξελίξεις στη τεχνολογία
status register	καταχωρητής καταστάσεως
stop bit	μπιτ σταματήματος
streaming	συνεχής ροή
sub-net	υπό-δίκτυο
switch matrice	πίνακας διακοπής
switching	μεταγωγή
testbench	πηγή προσομοίωσης
testing	επαλήθευση
thread	νήμα
threshold voltage	τάση κατωφλίου
time base	βάση χρόνου
time Base Divider	διαιρέτης βάσης χρόνου
time base oscillator	ταλαντωτής βάσης χρόνου
time counter	μετρητής χρόνου
timer	χρονιστής
toolchain	εργαλείο σχεδίασης
touch screen	οθόνη επαφής
trace buffer management	διαχειριστής των ενδιάμεσων μνημών
trade-off	συμβιβασμός
transactions	συναλλαγές
transconductance	διαγωγιμότητα
transfer	μεταγωγή
triggering	σκανδαλισμό
tuned	ηλεκτρικός συντονισμός
unified	ενωμένη
valid	έγκυρο
vendor independent	δεν συνδέεται με κάποιον προμηθευτή
VHDL generics	παράμετροι της γλώσσας VHDL
virtual	εικονικός
voltage converter	μετατροπέας τάσης
wake-up calls	εντολή ξυπνήματος
watchpoint registers	καταχωρητές-παρατηρητές
write back policy	πολιτική αποθήκευσης
I/O standards	πρότυπα εισόδου/εξόδου

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

AHB	Advanced High Performance Bus
AMBA	Advance Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
AS	Active Serial
ASB	Advanced System Bus
ASIC	Application-specific Integrated circuit
ASIP	application-specific instruction set
ATA	Advanced Technology Attachment
BiCMOS	Bipolar Complementary Metal Oxide Semiconductor
BLVDS	Bus LVDS
CAD	Computer-aided design
CD	Compact Disk
CLB	configurable logic blocks
CLUT	color lookup table
CPLD	Complex programmable logic device
CPU	Central Processing Unit
DAC	Digital-to-analog converter
DC	Direct Current
DE2	Development and Education board VERSION 2
DMA	Direct Memory Access
DMIPS	Dhrystone MIPS
DSP	Digital signal processing
DSU	Debug Support Unit
EDCL	Ethernet Debug Communication Link
ESA	European Space Agency
ESTEC	European Space Research and Technology Centre
FIFO	First in first out
FPGA	Field Programmable Gate Array
FPU	Floating Point Unit
GCC	GNU Compiler Collection
GDB	GNU debugger
GNU	GNU is not Unix

GPIO	General-purpose input/output
GTL	GenStat for Teaching and Learning
HDL	Hardware Description Language
HSMC	High Speed Mezzanine Card
HSTL	High-speed transceiver logic
HT	HyperTransport Protocol
I^2C	Inter-integrated Circuit
IDE	Integrated development environment
IEEE	Institute of Electrical and Electronics Engineers
IP	Intellectual property
IR	Infrared
ITG	IDE to GPIO
JTAG	Joint Test Action Group
LCD	liquid-crystal display (
LDT	Lighting Data Transport
LED	Light Emitting Diode
LGPL	Lesser General Public License
LRU	Least Recently Used
LUT	lookup tables
LVCMOS	Low Voltage Complementary Metal Oxide Semiconductor
LVDS	Low Voltage Differential Signaling
LVPECL	Low Voltage positive emmiter-coupled logic
LVTTL	Low Voltage Tranzistor- Tranzistor Logic
MAC	Multiplier Accumulator
Mb	Mega-byte
MHz	Mega Hertz
MMU	Memory management unit
NC	Not care
NMOS	N-type metal-oxide-semiconductor
NTSC	National Television System Committee
PAL	Phase Alternating Line
PCB	Printed circuit board
PCI	Peripheral Component Interconnect
PHY	Physical layer of the OSI model

PMOS	P-type metal-oxide-semiconductor
PPM	Part per million
PROM	Programmable read-only memory
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
RMS	Root Mean Square
ROM	Read Only Memory
RS-232	Recommended Standard 232
RSDS	Reduced swing differential signalling
RTL	Register Transfer Level
RTOS	Real Time Operating Systems
SD	Secure Digital
SDRAM	Synchronous dynamic random-access-memory
SECAM	Sequential Color with Memory
SEU	Single event upset
SFPT	SSH File Transfer Protocol
SMA	SubMiniature version
SMP	Symmetric Multi-processor support
SOC	System on Chip
SPARC	Scalable Processor Architecture
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
SRMMU	SPARC Reference Memory Management Unit
SSH	Secure Shell
SSTL	Stub Series Terminated Logic
SVGA	Super Video Graphics Array
TAP	Test Access Port
TLB	Translation lookaside buffer
TV	Television
UART	Universal asynchronous receiver/transmitter
USB	Universal serial bus
VGA	Video Graphics Array
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit

VOC	Volatile organic compounds
WSN	Wireless Sensor Network
I/O	Input/output

ΠΑΡΑΡΤΗΜΑ Ι

Α) Εγκατάσταση GRLIB και "τρέξιμο" προγράμματος στο GRMON

Η βιβλιοθήκη GRLIB διανέμεται σαν αρχείο 'gzipped tar-file' και μπορεί να εγκατασταθεί σε οποιοδήποτε μέρος του λειτουργικού συστήματος με τις εντολές:

gunzip -c grlib-com-1.5.0-bxxxx.tar.gz | tar xf – ή tar xvf grlib-com-1.5.0-bxxxx.tar.gz

Η βιβλιοθήκη GRLIB έχει την παρακάτω ιεραρχία:

bin	αρχεία 'scripts' και αρχεία υποστήριξης των εργαλείων
boards	αρχεία υποστήριξης για τις συσκευές
designs	πρότυπα σχέδια ('template designs')
doc	οδηγίες χρήσης
lib	VHDL βιβλιοθήκες
netlists	'netlists'
software	εργαλεία λογισμικού και αποσφαλμάτωσης
verification	πρόγραμματα αποσφαλμάτωσης ('test benches')

Η GRLIB χρησιμοποιεί το εργαλείο GNU 'make' για την δημιουργία των προγραμμάτων 'scripts' , για την μεταγλώττιση και την σύνθεση των σχεδίων. Για αυτό θα πρέπει να εγκατασταθεί σε ένα 'unix' ή 'unix-like' περιβάλλον.

Οργάνωση των φακέλων

Η GRLIB είναι οργανωμένη γύρω από βιβλιοθήκες VHDL, στις οποίες κάθε δομικό στοιχείο 'IP vendor' έχει το δικό του μοναδικό όνομα. Κάθε στοιχείο 'vendor' έχει και τον δικό του υπό-φάκελο κάτω από το φάκελο 'grlib/lib', στον οποίο φυλάσσονται όλα τα αρχεία 'scripts'.

Οι βασικοί φάκελοι που υπάρχουν στο GRLIB είναι:

grlib	*πακέτα με κοινό τύπο δεδομένων και λειτουργιών*
gaisler	*δομικά στοιχεία του Cobham Gaisler και εργαλεία*
*tech/**	*βιβλιοθήκες για συγκεκριμένες τεχνολογίες και για προσομοίωση σε επίπεδο πύλης*
techmap	*'wrappers' για την αντιστοίχιση των 'macro cells' (RAM, pads)*
work	*δομικά στοιχεία και πακέτα για την βιβλιοθήκη 'work' της VHDL*

Εγκατάσταση των βιβλιοθηκών για προσομοίωση

Οι βιβλιοθήκες προσομοίωσης αντιγράφονται από το φάκελο εγκατάστασης του εκάστοτε εργαλείου σύνθεσης (π.χ. QUARTUS) και στη συνέχεια μπορούν να "τρέξουν" με οποιοδήποτε εργαλείο προσομοίωσης. Σε κάποια σχέδια (designs) απαιτούνται έτοιμες βιβλιοθήκες, σε αυτή την περίπτωση θα υπάρχει σημείωση στο 'README.txt' αρχείο.

Οι επόμενες οδηγίες αφορούν την εγκατάσταση των βιβλιοθηκών σε όλη την βιβλιοθήκη GRLIB. Τα βήματα πρέπει να γίνουν μόνο μια φορά και αφορούν όλα τα σχέδια. Οι εντολές μπορούν να εκτελεστούν από τον αρχικό φάκελο της GRLIB αν έχει οριστεί η μεταβλητή περιβάλλοντος $GRLIB , για παράδειγμα με την εντολή:

export GRLIB=/home/user/grlib-com-1.5.0-b4161

Οι εντολές, επίσης, μπορούν να εκτελεστούν από οποιοδήποτε πρότυπο σχέδιο στον φάκελο 'designs/'.

Εγκατάσταση βιβλιοθηκών Altera

Οι βιβλιοθήκες της Altera αντιγράφονται από τον φάκελο εγκατάστασης του Quartus II. Πρέπει να έχει οριστεί η μεταβλητή περιβάλλοντος *$QUARTUS_ROOTDIR*. Για παράδειγμα με την εντολή:

export QUARTUS_ROOTDIR=/usr/local/altera/quartus13.1/quartus/

Οι βιβλιοθήκες της Altera εγκαθιστώντε στη συνέχεια με την εντολή:

make install-altera

Προσομοίωση

Το πρότυπο σχέδιο (template design) μπορεί να προσομοιωθεί στη συνέχεια με ένα πρόγραμμα αποσφαλμάτωσης. Το πρόγραμμα αποσφαλμάτωσης περιλαμβάνει μια εξωτερική μνήμη PROM και μια SDRAM, οι οποίες προ-φορτώνονται. Το πρόγραμμα αποσφαλμάτωσης θα τρέξει στον LEON3 επεξεργαστή και θα ελέγξει την λειτουργικότητα του σχεδίου. Επίσης, θα εμφανίζει διαγνωστικά στοιχεία στην κονσόλα του προσομοιωτή κατά την εκτέλεση του.

Οι ακόλουθες εντολές θα πρέπει να εκτελεστούν ώστε να γίνει μεταγλώττιση και προσομοίωση του πρότυπου σχεδίου και του πρόγραμματος αποσφαλμάτωσης στο πρόγραμμα Mentor ModelSim/QuestaSim ή στο Aldec Riviera-PRO (ο προσομοιωτής επιλέγεται από την μεταβλητή περιβάλλοντος GRLIB_SIMULATOR, η προκαθορισμένη τιμή είναι το Modelsim):

Make sim
make sim-launch

Το πρόγραμμα αποσφαλμάτωσης αποτελείται από δύο κομμάτια, ένα φορτωτή PROM (PROM boot loader - prom.S) και το πρόγραμμα προς έλεγχο (systest.c). Και τα δυο κομμάτια μπορούν να μεταγλωττιστούν χρησιμοποιώντας την εντολή ‘make soft’. Αυτό απαιτεί την εγκατάσταση του εργαλείου ‘BCC tool-chain’ στον υπολογιστή. Το εργαλείο ‘BCC tool-chain’ περιλαμβάνει τις λειτουργίες σάρωσης AMBA “σύνδεσε και τρέξε” και μπορεί να σαρώσει τον δίαυλο AHB.

Ο φορτωτής PROM περιλαμβάνει τον κώδικα για την αρχικοποίηση του επεξεργαστή, του ελεγκτή μνήμης και των άλλων περιφερειακών. Αν λείπει το αρχείο ‘prom.S’ τότε χρησιμοποιείται ένα έτοιμο αρχείο που βρίσκεται στον φάκελο ‘software/leon3/prom.S’. Οι σταθερές των ρυθμίσεων που χρησιμοποιούνται από το αρχείο ‘prom.S’ βρίσκονται στο αρχείο ‘prom.h’ . Αν έχουν γίνει αλλαγές στον ελεγκτή μνήμης ή στη βασική διεύθυνση της κύριας μνήμης, τότε πιθανόν να χρειάζεται ανανέωση του ‘prom.h’ και του ‘prom.S’. Αν τροποποιηθούν το ‘prom.h’ ή το ‘prom.S’ τότε χρειάζεται η εντολή ‘make soft’ ώστε οι αλλαγές να οριστικοποιηθούν.

Σύνθεση

Στο πρότυπο σχέδιο μπορεί να γίνει σύνθεση είτε με το Synplify, Precision,ISE/XST ή το Quartus. Για την απευθείας σύνθεση μέσω του QUARTUS, χρησιμοποιείται η εντολή:

make quartus

Και για να ανοίξει το εργαλείο QUARTUS:

make quartus-launch

Τρέξιμο εφαρμογών

Για το κατέβασμα και την αποσφαλμάτωση εφαρμογών στη συσκευή, χρησιμοποιείται το GRMON. Το GRMON μπορεί να συνδεθεί με το σύστημα προς έλεγχο με την χρήση RS232, JTAG, ethernet, USB, PCI ή SpaceWire.

Για το κατέβασμα της εφαρμογής χρησιμοποιείται η εντολή 'load'. Και για την εκτέλεση η εντολή 'run'. Το αποτέλεσμα της εξόδου εμφανίζεται στο παράθυρο του GRMON, αν το GRMON έχει ξεκινήσει με την παράμετρο '-u', αλλιώς τα αποτελέσματα θα εμφανίζονται στη θύρα RS232 της συσκευής.

Προγραμματισμός της μνήμης Flash/PROM

Μια εικόνα PROM δημιουργείται με το εργαλείο MKPROM2, το οποίο μπορεί να κατεβεί από την ιστοσελίδα www.gaisler.com.

Όταν δημιουργηθεί η εικόνα PROM, η μνήμη FLASH/PROM που βρίσκεται στη συσκευή μπορεί να προγραμματιστεί μέσω του GRMON με τις εντολές:

flash erase all (για το άδειασμα της flash)

flash load prom.out

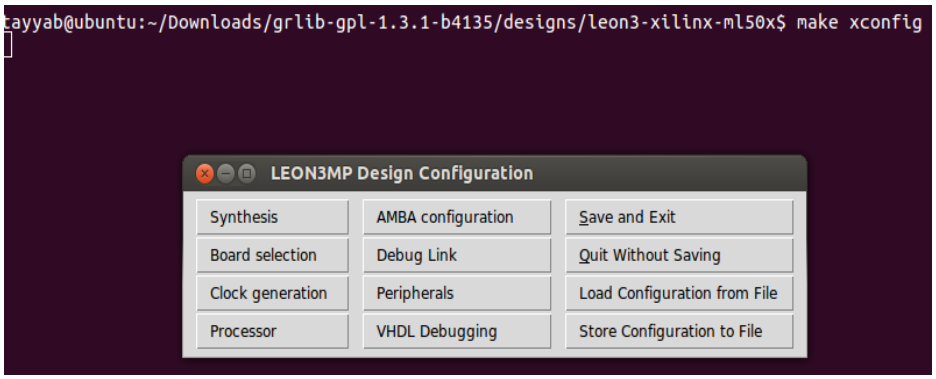
Παρακάτω παρουσιάζονται τα βήματα για την φόρτωση και το τρέξιμο ενός προγράμματος σε μια συσκευή που περιλαμβάνει επεξεργαστή LEON3. Τα βήματα αφορούν σε λειτουργικό σύστημα LINUX

1 Εγκατάσταση του εργαλείου σχεδίασης (toolchain) το οποίο είναι διαθέσιμο στην ιστοσελίδα *gaisler.com.*

Εικόνα Α.1: Κατέβασμα GRLIB από την ιστοσελίδα της GAISLER (gaisler.com)

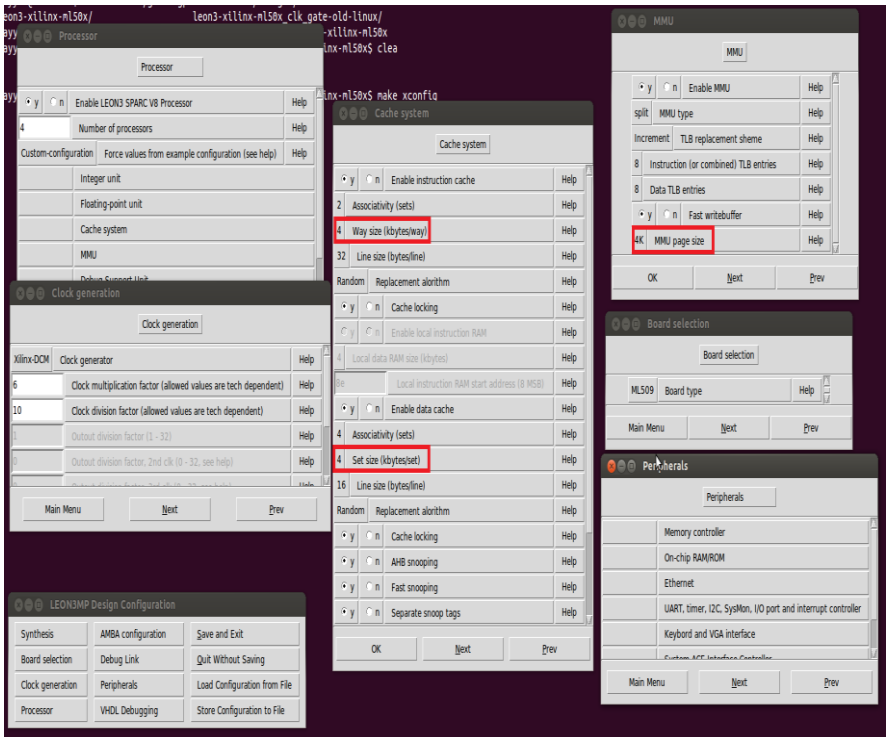
Η βιβλιοθήκη μπορεί να εξαχθεί ('extract') σε οποιονδήποτε φάκελο. Προτιμάται η εγκατάσταση να γίνει στον φάκελο /opt.

2. Είσοδος στο φάκελο της GRLIB η οποία αντιστοιχεί στην συσκευή που θέλετε να συνδέσετε τον επεξεργαστή. Για παράδειγμα, για την συσκευή 'ml50x' της Xilinx πρέπει να πάτε στον φάκελο του "\..\designs\leon3-xilinx-ml50x". Στον φάκελο αυτόν πατάτε την εντολή 'make xconfig'. Το εργαλείο 'make xconfig' χρειάζεται την τελευταία έκδοση του 'tcl/tk'.



Εικόνα Α.2: Γραφικό περιβάλλον ρύθμισης για τον Leon3

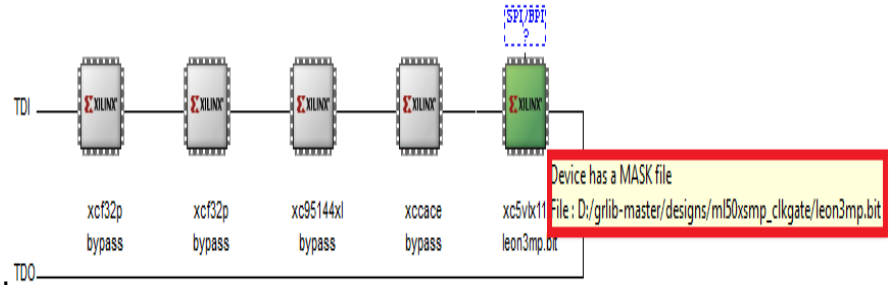
3. Εξερευνήστε τις διαφορετικές επιλογές, οι οποίες είναι σε μορφή ‘Push buttons’ και επιλέξτε τις επιθυμητές ρυθμίσεις.



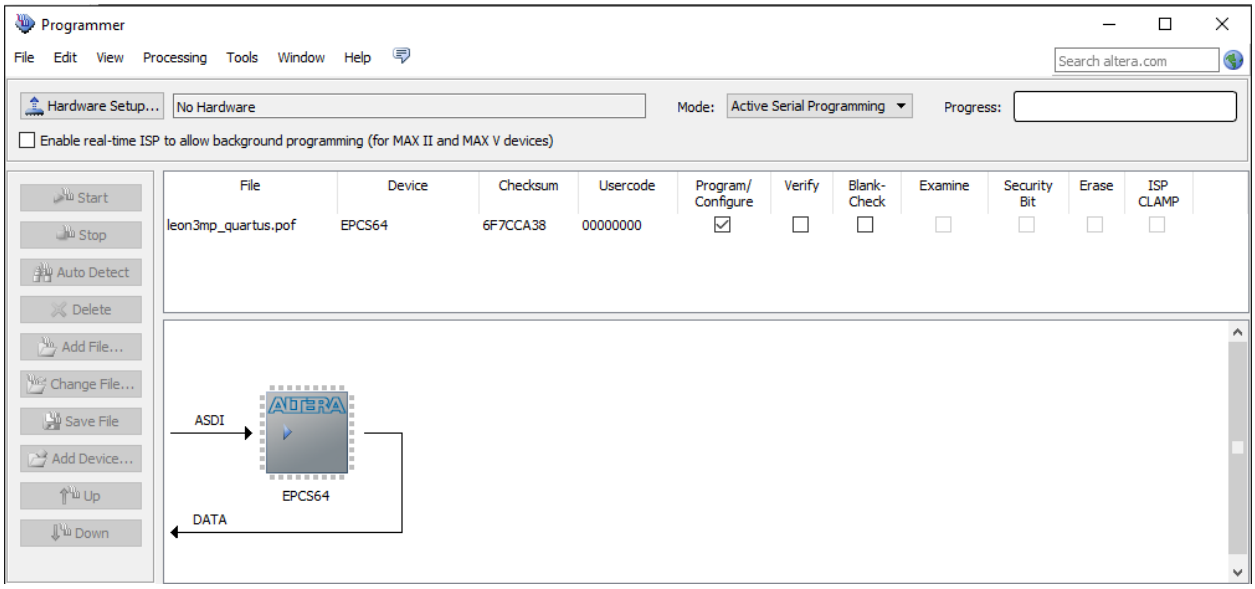
Εικόνα Α.3: Επιλογές ρυθμίσεων για τον Leon3

4. Μετά την ρύθμιση του επεξεργαστή μπορεί να αρχίσει η σύνθεση, με την εντολή 'make ise' αν η συσκευή είναι της Xilinx ή ‘make quartus’ για Altera. Εναλλακτικά, η σύνθεση μπορεί να γίνει ανοίγοντας το αντίστοιχο σχεδιαστικό εργαλείο, για παράδειγμα για το QUARTUS με την εντολή ‘make quartus-launch’. Θα πρέπει να έχουν καθοριστεί οι μεταβλητές περιβάλλοντος του κάθε σχεδιαστικού εργαλείου.

5. Φόρτωση του ‘bitstream’ που δημιουργείται από την σύνθεση στην συσκευή.

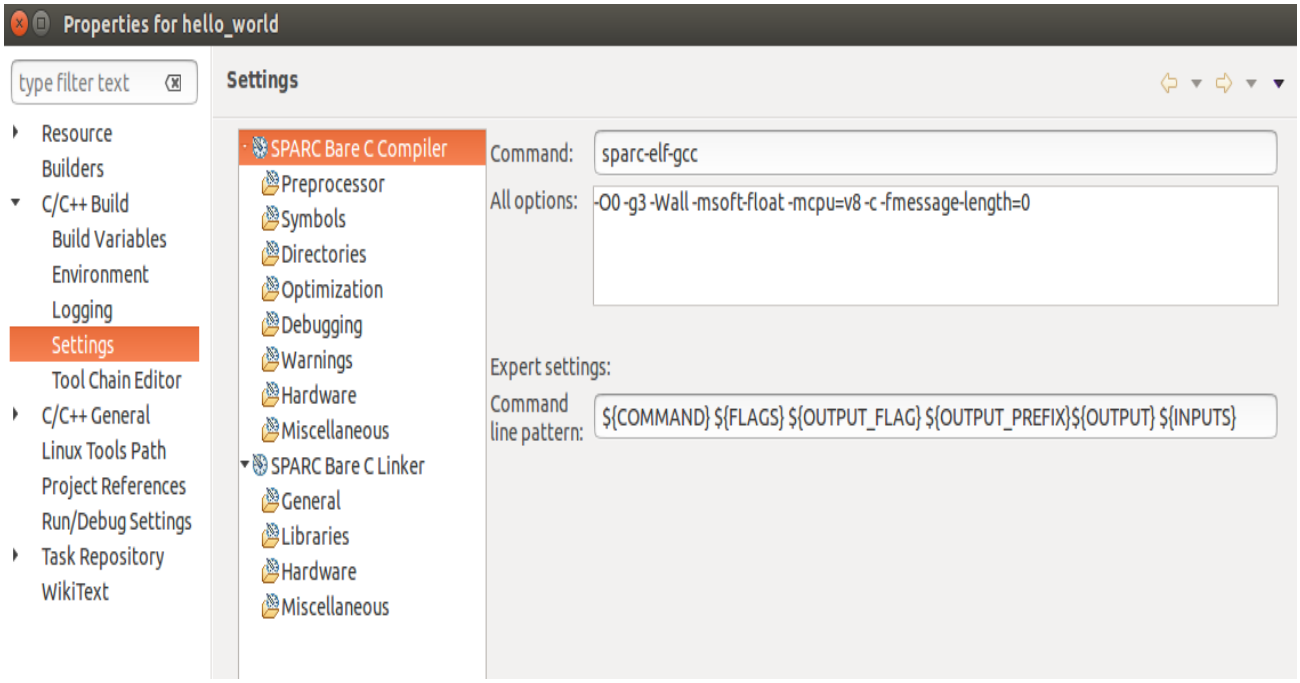


Εικόνα Α.4: Αποθήκευση του ‘Bitstream’ στη συσκευή ML509 με το πρόγραμμα ISE

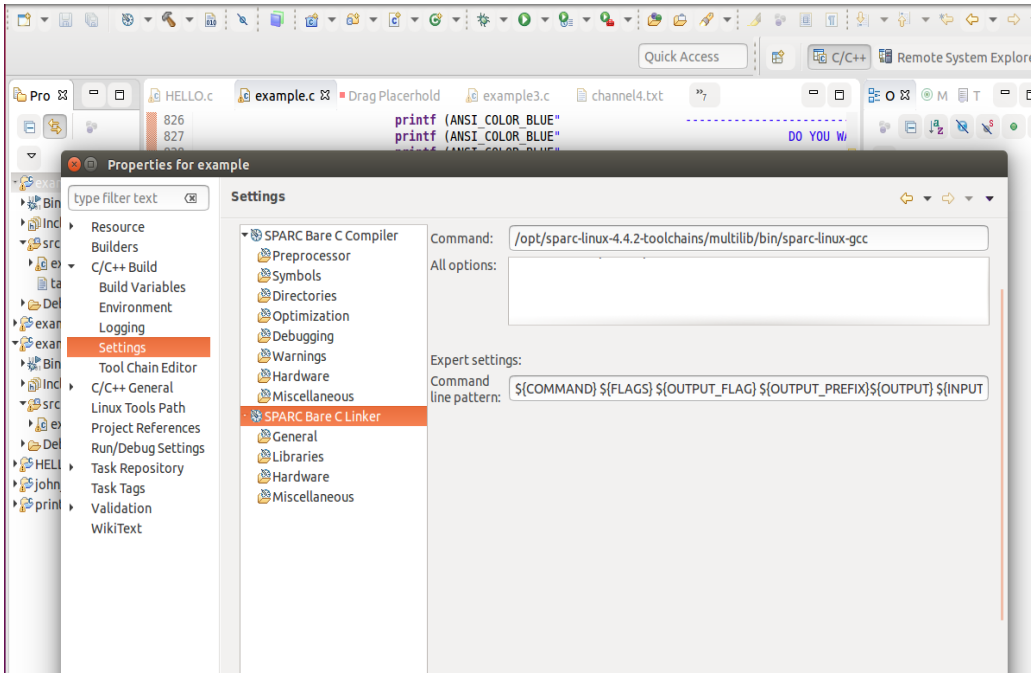


Εικόνα Α.5: Αποθήκευση του 'Bitstream' στην μνήμη FLASH της συσκευής DE2-115 με 'active serial programming' στο πρόγραμμα QUARTUS

6. Για το τρέξιμο ενός προγράμματος και την φόρτωση του στο σύστημα μπορεί να χρησιμοποιηθεί το ECLIPSE και το GRMON. Η μεταγλώττιση του προγράμματος γίνεται στο ECLIPSE αφού πρώτα έχει συνδεθεί το εργαλείο σχεδίασης της SPARC με αυτό.

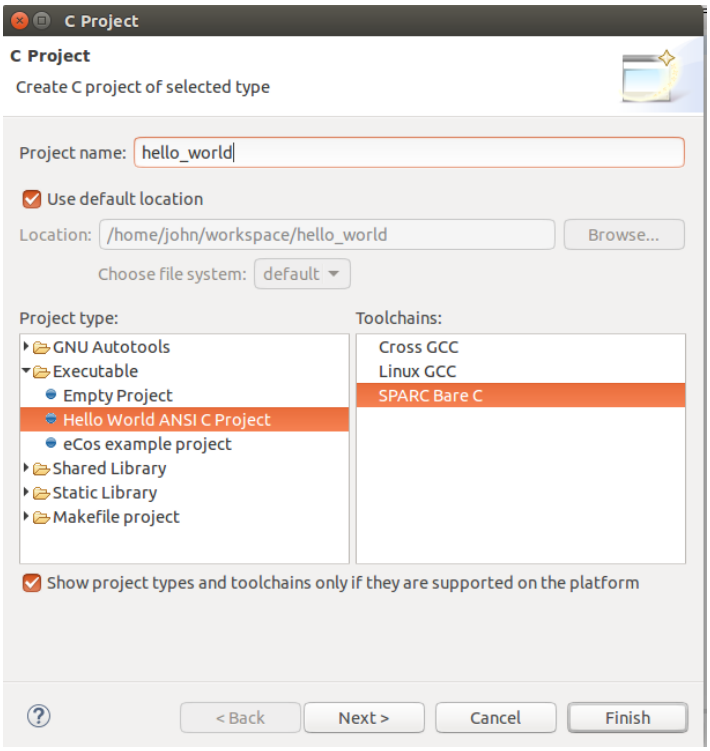


Εικόνα Α.6: Σύνδεση του εργαλείου σχεδίασης της SPARC με το ECLIPSE (compiler)

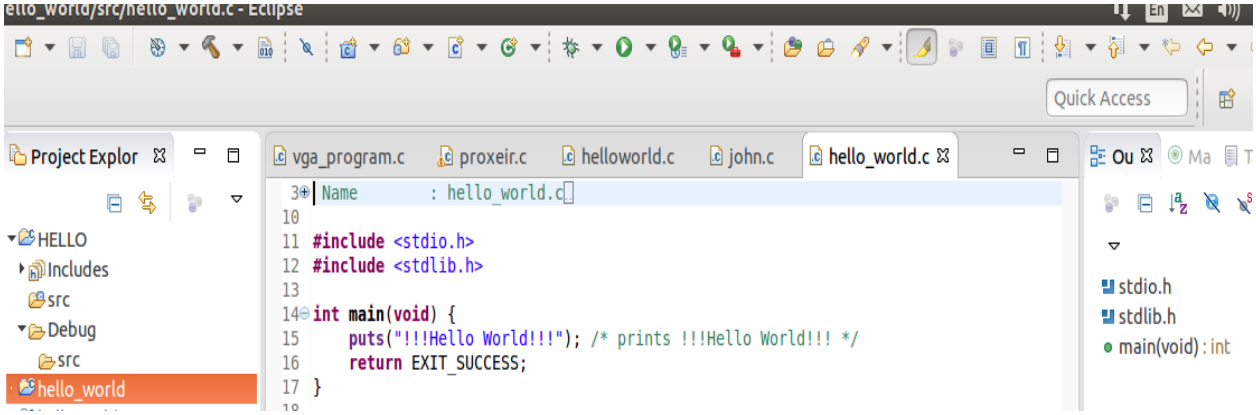


Εικόνα Α.7: Σύνδεση του εργαλείου σχεδίασης της SPARC με το ECLIPSE (linker)

Το εργαλείο σχεδίασης της SPARC έχει αποθηκευτεί στο φάκελο '/opt'. Στην εικόνα Α.6 έχει γίνει η σύνδεση για την μεταγλώττιση και στην Α.7 για την σύνθεση (link) του προγράμματος. Στην εικόνα Α.8 φαίνεται η δημιουργία ενός προγράμματος στο ECLIPSE. Θα πρέπει να έχει γίνει και η εγκατάσταση του LEON IDE όπως περιγράφεται στο κεφάλαιο 5.3. Στην εικόνα Α.9 φαίνεται ο κώδικας του προγράμματος.



Εικόνα Α.8: Δημιουργία προγράμματος στο ECLIPSE

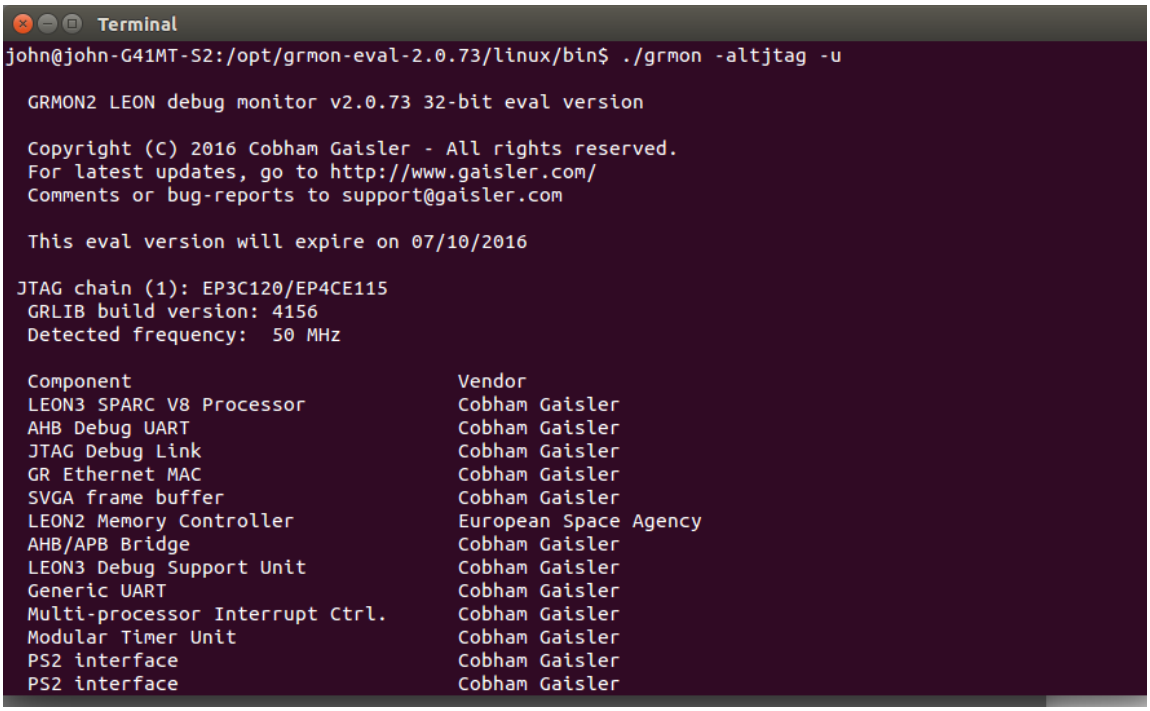


Εικόνα Α.9: Πρόγραμμα 'Hello World' στο ECLIPSE

7. Μετά την μεταγλώττιση, το πρόγραμμα μπορεί να φορτωθεί μέσω του GRMON στη συσκευή που τρέχει ο επεξεργαστής LEON3. Το GRMON μπορεί να φορτωθεί με την παρακάτω εντολή για συσκευή της Altera με JTAG (εικόνα Α.10):

grmon -altjtag -u
('-u' για εμφάνιση αποτελεσμάτων στο GRMON)

Για φόρτωση εικόνας Linux στη μνήμη RAM πρέπει να χρησιμοποιηθεί και η εντολή '-nb' και για σύνδεση μέσω ETHERNET η παράμετρος '*-eth'* και '*-ip'* αντί για *-altjtag*.



Εικόνα Α.10: Άνοιγμα GRMON

8. Αφού έχει ανοίξει το GRMON και έχουν ανιχνευθεί τα περιφερειακά του συστήματος μπορεί να φορτωθεί το πρόγραμμα. Το εκτελέσιμο αρχείο του προγράμματος βρίσκεται στο φάκελο εργασίας ('workspace') του ECLIPSE στον υπο-φάκελο 'Debug'. Για την φόρτωση του προγράμματος πατήστε τις παρακάτω εντολές

load /.../hello
run

```
grmon2> load /home/john/workspace/john/Debug/john
  40000000 .text                    23.6kB /  23.6kB   [===============>] 100%
  40005E80 .data                     2.7kB /   2.7kB   [===============>] 100%
  Total size: 26.30kB (2.29Mbit/s)
  Entry point 0x40000000
  Image /home/john/workspace/john/Debug/john loaded

grmon2> run
!!!Hello World!!!

  Program exited normally.

grmon2>
```

Εικόνα Α.11: Φόρτωση και τρέξιμο προγράμματος στον Leon3

Περισσότερες πληροφορίες υπάρχουν στις οδηγίες χρήσης της GRLIB [28] και του GRMON [29] , όπως και στον οδηγό ανάπτυξης και ρύθμισης του LEON3 επεξεργαστή [45].

Β) Οδηγίες χρήσης LINUXBUILD

Η Aeroflex Gaisler έχει αναπτύξει ένα πρόγραμμα ,το LINUXBUILD, το οποίο συνδυάζει το BUILDROOT περιβάλλον με διάφορα προγράμματα της Gaisler που ήδη υπάρχουν, όπως το MKLINUXIMG και το MKPROM. Συνδυάζει διάφορα προγράμματα τα οποία έχουν δημιουργηθεί ώστε να μπορούν να λειτουργούν αυτόνομα. Παρακάτω φαίνονται συνοπτικά όλα τα προγράμματα που χρησιμοποιούνται από το LINUXBUILD:

• εργαλεία σχεδίασης GNU Toolchain - GCC, BINUTILS, GLIBC

• πυρήνα Linux για επεξεργαστή LEON (Linux Kernel + LEON Linux patches)

• φορτωτή εικόνας στη RAM (LEON Linux RAM loader - MKLINUXIMG)

• Buildroot + LEON patches

• MKPROM2

Για τη λειτουργία του LINUXBUILD είναι απαραίτητα τα παρακάτω:

• σύστημα Linux

• εργαλείο σχεδίασης “SPARC/LEON Linux Toolchain”

• MKPROM2 – για τη δημιουργία των PROM/FLASH images

• wget

• git

• Πρόσβαση στο Internet

Επίσης, ίσως χρειαστούν και κάποια άλλα εργαλεία για τη λειτουργία του Buildroot όπως τα ‘bison’, ‘flex’, ‘msgfmt’, ‘makeinfo’ κ.τ.λ. Κατά τη δημιουργία των εικόνων Linux θα εμφανιστούν λάθη αν λείπουν κάποιες βιβλιοθήκες ή εργαλεία από το σύστημα.

Διευθύνσεις για το κατέβασμα των απαραίτητων αρχείων:

LINUXBUILD: http://gaisler.com/anonftp/linux/linux-2.6/linuxbuild/linuxbuild-x.y.z.tar.bz2
Linux toolchain: http://gaisler.com/anonftp/linux/linux-2.6/toolchains/sparc-linux-4.4.2/
MKPROM2: http://gaisler.com/anonftp/mkprom2/linux/

Συνοπτικές οδηγίες για την δημιουργία εικόνας Linux:

1. Κατέβασμα και εγκατάσταση του εργαλείου σχεδίασης Linux (Linux toolchain)

2. Κατέβασμα και ξεπακετάρισμα (unpack) του LINUXBUILD

3. Είσοδος στο φάκελο του LINUXBUILD που δημιουργήθηκε

4. Τρέξιμο εργαλείου ‘make xconfig’ (εδώ ίσως χρειαστούν και κάποιες βιβλιοθήκες ανάλογα με το σύστημα)

5. Πάτημα της επιλογής “Install Linux”
6. Διπλό κλικ στην επιλογή “Execute Linux installation of latest stable leon-linux”

7. Πάτημα της επιλογής ‘yes’ στο μήνυμα που θα ανοίξει και περιμένουμε να τελειώσει η εγκατάσταση.

8. Ελέγχουμε ότι έχει τελειώσει η εγκατάσταση χωρίς λάθη και πατάμε την επιλογή “return to close window”.

9. Πάμε στο τέλος του μενού στην επιλογή “Step3: build”

10. Διπλό κλικ στην επιλογή “Execute make build”

11.Πατάμε ‘yes’ στο μήνυμα που θα εμφανιστεί.

12.Ελέγχουμε ότι το κτίσιμο έγινε χωρίς λάθη και πατάμε την επιλογή “return to close window”.

13.Κλείνουμε το μενού ‘xconfig’

Αν όλα έχουν πάει καλά οι εικόνες θα έχουν δημιουργηθεί στον φάκελο του LINUXBUILD “output/images”:

• image.ram - RAM image (για φόρτωση και τρέξιμο στη RAM)

• image - virtual address image (για αποσφαλμάτωση)

Παρακάτω γίνεται μια πιο αναλυτική αναφορά στην εγκατάσταση και ανάπτυξη του LINUXBUILD περιβάλλοντος:

Εγκατάσταση LINUXBUILD:

Μετά το κατέβασμα του LINUXBUILD κάνουμε εξαγωγή (extract) του πακέτου χρησιμοποιώντας την εντολή “tar -xf”:

$ tar -xf linuxbuild-x.y.z.tar.bz2

$ cd linuxbuild-x.y.z

Εργαλείο σχεδίασης-TOOLCHAIN

Πριν την ανάπτυξη στο LINUXBUILD θα πρέπει να έχει προηγηθεί η εγκατάσταση του εργαλείου σχεδίασης “SPARC/LEON Linux toolchain”. Το εργαλείο σχεδίασης “gcc-4.4.2 multilib based toolchain” μπορεί να κατέβει από το site της Aeroflex Gaisler “http://gaisler.com/anonftp/linux/linux-2.6/toolchains/sparc-linux-4.4.2/”. Το εργαλείο σχεδίασης θα πρέπει να εγκατασταθεί στον φάκελο “/opt” και έτσι να δημιουργηθεί ο φάκελος “/opt/sparc-linux-x.y.z-toolchains/multilib”. Ο ”bin” φάκελος ο οποίος περιέχει το εργαλείο ‘sparc-linux-gcc’ θα πρέπει να οριστεί σαν μεταβλητή περιβάλλοντος(PATH VARIABLE)

$ export PATH=/opt/sparc-linux-4.4.2-toolchains/multilib/bin:$PATH

$ which sparc-linux-gcc

/opt/sparc-linux-4.4.2-toolchains/multilib/bin/sparc-linux-gcc

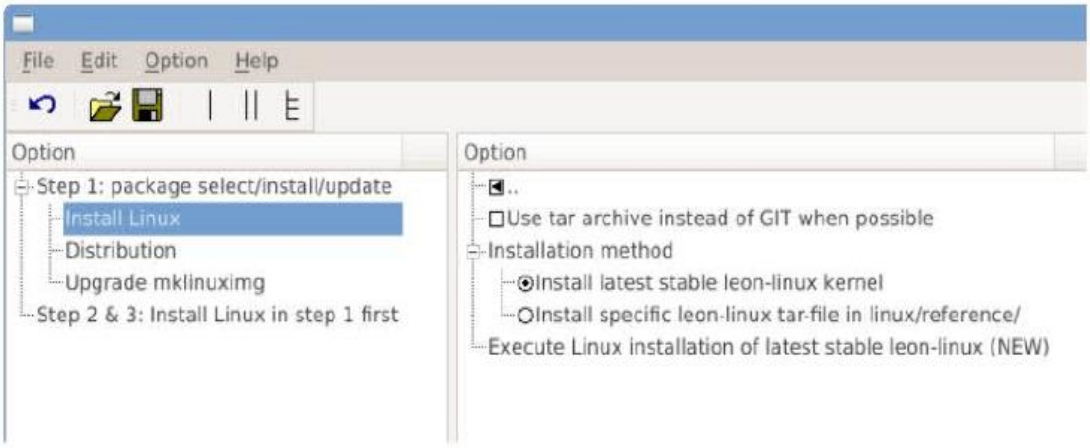
σημείωση: Η εγκατάσταση του εργαλείου σχεδίασης θα πρέπει να γίνει απαραίτητα σε αυτό το φάκελο (/opt).

Εγκατάσταση πυρήνα LINUX

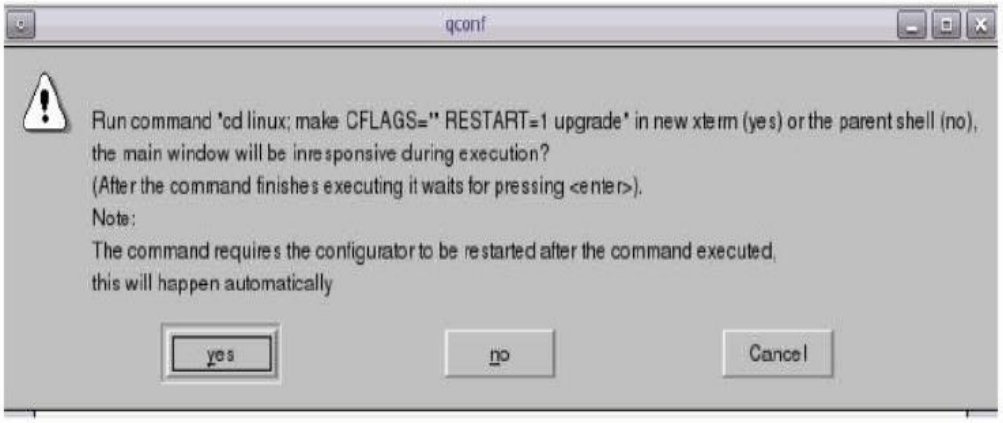
Αυτόματα το Linux εγκαθιστάτε δημιουργώντας ένα αντίγραφο από την επίσημη ιστοσελίδα για Linux (“Linux GIT repository”) και εφαρμόζοντας ‘patches’ που χρειάζονται από εκεί. Εάν επιλεχτεί η επιλογή "Use tar archive instead of GIT when possible", τότε κατεβαίνουν ‘tar’ αρχεία από την ιστοσελίδα “kernel.org” και χρησιμοποιούνται ως βάση (‘patch-base’) (στο πλαίσιο της διπλωματικής δεν χρειάστηκε αυτή η επιλογή).

Η τελευταία σταθερή έκδοση (stable version) που είναι συμβατή με τον επεξεργαστή LEON εγκαθίσταται αυτόματα. Για την εγκατάσταση διαφορετικής έκδοσης πρέπει να κατέβει και να τοποθετηθεί στον φάκελο “linux/reference” του LINUXBUILD η έκδοση πριν το άνοιγμα του GUI (graphical user interface). Στη συνέχεια πατάμε την επιλογή "Install specific leon-linux tar-file".

Η εγκατάσταση του πυρήνα εκτελείται όταν πατηθεί από το GUI η επιλογή "Execute Linux installation". Ένα μενού διαλόγου θα εμφανιστεί το οποίο ρωτά αν θέλουμε να γίνει η εγκατάσταση σε ένα καινούργιο τερματικό (‘xterm’) ή στο ήδη υπάρχον. (δεν έχει κάποια ουσιαστική διαφορά η συγκεκριμένη επιλογή).



Εικόνα B1: Μενού επιλογής δομικών στοιχείων Linux για εγκατάσταση/αναβάθμιση



Εικόνα B2: Μήνυμα διαλόγου

Buildroot

Το Buildroot εγκαθίσταται αυτόματα. Στο βήμα 1 του GUI υπάρχει η επιλογή να μην χρησιμοποιηθεί κάποιο ‘distribution’ π.χ. για τη δημιουργία Linux με έτοιμο σύστημα φακέλων (file system image). Αυτή η επιλογή είναι χρήσιμη, όταν χρησιμοποιούνται αρχεία “custom scripts” για την δημιουργία ενός συστήματος φακέλων και όταν ο

πυρήνας χρησιμοποιεί σύστημα φακέλων μέσω ενός NFS ή το σύστημα φακέλων βρίσκεται στη FLASH/PROM. (Για τη παρούσα διπλωματική χρησιμοποιήθηκε το Buildroot).

MKLINUXIMG Linux RAM loader

Η τελευταία έκδοση του MKLINUXIMG τη στιγμή έκδοσης του LINUXBUILD είναι ήδη προεγκαταστημένη. Υπάρχει η επιλογή για αναβάθμιση στο βήμα 1 του GUI.

MKPROM2

Η εφαρμογή MKPROM χρησιμοποιείται για τη δημιουργία εικόνων προς φόρτωση από την μνήμη FLASH/PROM ("FLASH/PROM bootable"). Πρέπει να κατέβει και να εγκατασταθεί από το χρήστη ανεξάρτητα από το LINUXBUILD. Το MKPROM είναι ένα ξεχωριστό εργαλείο και μπορεί να χρησιμοποιηθεί και με άλλα εργαλεία του LEON και λειτουργικά συστήματα ανεξάρτητα από το LINUXBUILD. Για την χρησιμοποίησή του με το LINUXBUILD θα πρέπει να τοποθετηθεί στο μονοπάτι (PATH) του συστήματος.

Ρύθμιση του LINUXBUILD

Μετά την εγκατάσταση των επιμέρους εργαλείων μπορεί να αρχίσει η παραμετροποίηση (configuration) τους χρησιμοποιώντας το GUI του LINUXBUILD.

Υπάρχουν διάφορα γραφικά εργαλεία που μπορούν να χρησιμοποιηθούν:

Η χρησιμοποίηση του εργαλείου "xconfig" είναι η προτεινόμενη λειτουργία:

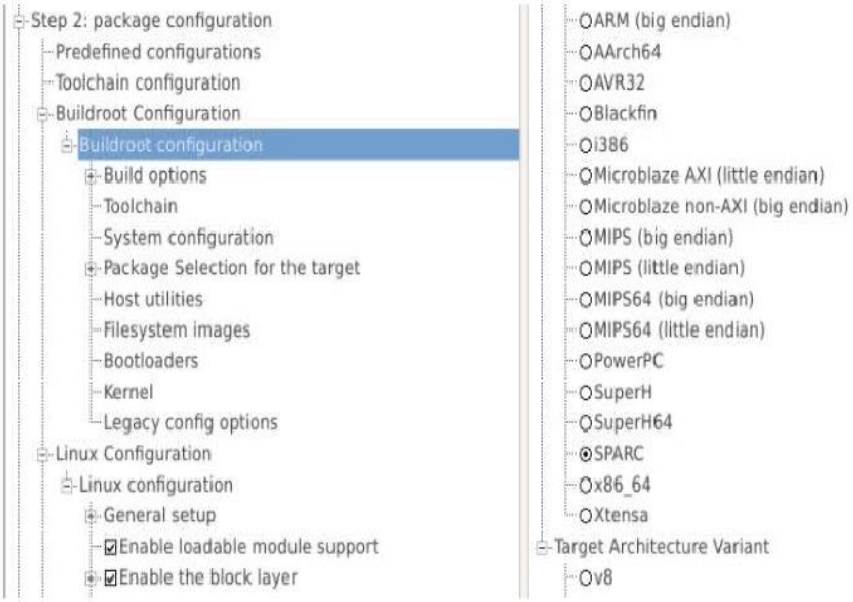
• make xconfig - Qt based GUI (για αυτό το εργαλείο χρειάζονται οι βιβλιοθήκες Qt-3/4)

Δεν προτείνονται αλλά παρέχονται και τα παρακάτω εργαλεία

• make gconfig - GTK based GUI

• make menuconfig - ncurses based

Παρακάτω φαίνονται τα μενού ρύθμισης του Buildroot και του Linux, στο κεντρικό μενού του LINUXBUILD περιβάλλοντος:



Εικόνα Β3: Μενού ρύθμισης Buildroot

Εικόνα Β4: Μενού ρύθμισης Linux

Προκαθορισμένες ρυθμίσεις

Υπάρχουν κάποιες προκαθορισμένες LEON ρυθμίσεις (cofigurations) που μπορούν να χρησιμοποιηθούν για ευκολία. Αυτές οι ρυθμίσεις βρίσκονται στον φάκελο “gaisler/configs” και μπορούν να φορτωθούν στο LINUXBUILD με την επιλογή "Predefined configurations" στο GUI. Επίσης, μπορεί να δημιουργηθεί μια νέα ρύθμιση και να αποθηκευτεί. Για τη φόρτωση προκαθορισμένων ρυθμίσεων πρέπει να επιλεχθεί αυτή που επιθυμείτε και να γίνει διπλό κλικ στην επιλογή "Load the selected configuration". Το GUI θα επανεκκινήσει και θα φορτωθούν οι νέες ρυθμίσεις. Για την αποθήκευση μιας αλλαγής σε μια προεπιλεγμένη ρύθμιση επιλέγεται η επιλογή "Save the current configuration back…". Για την αποθήκευση των αλλαγών σε ένα καινούργιο αρχείο πρέπει να επιλεχτεί ένα όνομα και να πατηθεί η επιλογή “Save the current configuration into a new..."

Ρυθμίσεις του εργαλείου σχεδίσης (Toolchain)

Εδώ υπάρχει η επιλογή να χρησιμοποιηθεί ένα εξωτερικό εργαλείο (“external toolchain”) που βρίσκεται στο μονοπάτι (προκαθορισμένη επιλογή), ένα εργαλείο που εισάγεται από τον χρήστη ή να χρησιμοποιηθεί το εργαλείο σχεδίασης του Buildroot. Η επιλογή θα πρέπει να συμβαδίζει με την επιλογή του εργαλείου σχεδίασης στο μενού Buildroot. Για την διπλωματική χρησιμοποιήθηκε το εξωτερικό εργαλείο το οποίο βρίσκεται στο “μονοπάτι (sparc-linux).

Ρυθμίσεις Buildroot

Το Buildroot χρησιμοποιείται για την ανάπτυξη των ”εφαρμογών οι οποίες θα τρέχουν στην εικόνα Linux. Επιλέγονται όλα τα προγράμματα που είναι απαραίτητα για το σύστημα. Επίσης, στο Buildroot επιλέγετε αν το σύστημα που θα τρέχει το λειτουργικό σύστημα Linux, υποστηρίζει μονάδα FPU(floating-point unit) ή όχι.

Ρυθμίσεις Linux

Σε αυτό το κομμάτι καθορίζονται οι βασικές λειτουργίες του πυρήνα Linux που θα δημιουργηθεί, π.χ. ποιοι οδηγοί θα εγκατασταθούν ώστε να τρέχει το απαραίτητο υλικό.

Ρυθμίσεις MKLINUXIMG

Σε αυτό το σημείο καθορίζονται οι παράμετροι με τις οποίες θα λειτουργήσει το πρόγραμμα MKLINUXIMG. Ουσιαστικά, το MKLINUXIMG συνδέει τον πυρήνα Linux με την κεντρική μνήμη. Ο φορτωτής “LEON Linux RAM loader” είναι υπεύθυνος για τον καθορισμό του χαμηλού επιπέδου (“low-level”) περιβάλλοντος το οποίον απαιτείται από τον πυρήνα Linux. Υπάρχουν διάφορες επιλογές, όπως από πιο σημείο της μνήμης RAM θα αρχίσει να εκτελείτε το πρόγραμμα και ο καθορισμός της MAC διεύθυνσης. Επίσης, μπορεί να καθοριστεί η “γραμμή εντολών του πυρήνα” (kernel command line), που είναι οι πρώτες εντολές που τρέχουν στον πυρήνα Linux. Με αυτό τον τρόπο μπορεί να επιλεχτεί και η κονσόλα που θα εμφανίζεται το τερματικό π.χ. σειριακή θύρα.

Ρυθμίσεις MKPROM2

Σε αυτό το κομμάτι καθορίζεται πως, θα ρυθμιστεί και θα χρησιμοποιηθεί το πρόγραμμα MKPROM2. Εάν επιλεγεί, θα πρέπει να καθοριστεί και το μονοπάτι στο οποίο είναι εγκατεστημένο το MKPROM2. Θα πρέπει να καθοριστούν επακριβώς οι παράμετροι του συστήματος στο οποίο θα τρέχει η εικόνα ώστε να είναι λειτουργική. Η εικόνα PROM που δημιουργείται μπορεί να φορτωθεί σε μια μνήμη FLASH χρησιμοποιώντας τις εντολές του GRMON (flash erase,flash load).

Κτίσιμο Εικόνας

Μετά την εγκατάσταση του Linuxbuild, την ρύθμιση των παραμέτρων του Linuxbuild και όλων των επιμέρους εργαλείων, η δημιουργία της εικόνας Linux πραγματοποιείται κάνοντας διπλό κλικ στην επιλογή “execute make build”. Οι παραχθείσες εικόνες τοποθετούνται στο φάκελο “output/directory”. Η εικόνα “image.ram” μπορεί να τρέξει μέσω του GRMON απευθείας στη μνήμη RAM. Αν έχει χρησιμοποιηθεί το MKPROM, θα υπάρχει το “image.prom” στον ίδιο φάκελο.

Μετά από σημαντικές αλλαγές (π.χ. αλλαγές στο εργαλείο σχεδίασης, μονάδα FPU ή όχι) ή αφαίρεση προγραμμάτων του Buildroot απαιτείται η δημιουργία της εικόνας από την αρχή. Αυτό μπορεί να γίνει κάνοντας διπλό κλικ στην επιλογή “execute make clean” και μετά τις αλλαγές επιλέγοντας “execute make build”. Αν υπάρξουν λάθη κατά την δημιουργία των εικόνων, ίσως απαιτείται η εγκατάσταση κάποιων εργαλείων.

ΠΑΡΑΡΤΗΜΑ ΙΙ

Α) Κώδικας μετρήσεων

```
#include <stdio.h> //βιβλιοθήκες//
#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <pthread.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/poll.h>
#include <signal.h>
#include <sys/ioctl.h>
#include <sys/mman.h>
#include <stdbool.h>
#include <err.h>
#include <math.h>
#include <malloc.h>

volatile int b,set1,y2,loop=0;
volatile int z=0;
volatile int overflow=0;
#define ERREXIT(str) {printf("err %s, %s\n", str, strerror(errno)); return -1;}
#define ANSI_COLOR_RED     "\x1b[31m" //χρώματα
#define ANSI_COLOR_GREEN   "\x1b[32m"
#define ANSI_COLOR_YELLOW  "\x1b[33m"
#define ANSI_COLOR_BLUE    "\x1b[34m"
#define ANSI_COLOR_MAGENTA "\x1b[35m"
#define ANSI_COLOR_CYAN    "\x1b[36m"
#define ANSI_COLOR_RESET   "\x1b[0m"
void*thread_routine3()   //νήμα για διακοπή υπερχείλισης
{
        struct pollfd xfds[1];
         const char *fn;
         char buf[4];
         int rc;
         int fd;
start3:
       /* export */
         fn = "/sys/class/gpio/export";
         fd = open(fn, O_WRONLY); if(fd < 0)  ERREXIT("open export")
         rc = write(fd, "227", 3); if(rc != 3) ERREXIT("write export")
         close(fd);
         /* direction */
         fn = "/sys/class/gpio/gpio227/direction";
         fd = open(fn, O_RDWR);  if(fd < 0)  ERREXIT("open direction")
         rc = write(fd, "in", 3);if(rc != 3) ERREXIT("write direction")
         close(fd);
         /* edge */
         fn = "/sys/class/gpio/gpio227/edge";
         fd = open(fn, O_RDWR);       if(fd < 0)  ERREXIT("open edge")
```

```
        rc = write(fd, "rising", 8); if(rc != 8) ERREXIT("write edge")
        rc = lseek(fd, 0, SEEK_SET);  if(rc < 0)  ERREXIT("lseek edge")
        rc = read(fd, buf, 10);       if(rc <= 0) ERREXIT("read edge")
        buf[rc] = '\0';
        close(fd);
        fn = "/sys/class/gpio/gpio227/value";
        fd = open(fn, O_RDWR);  if(fd < 0)  ERREXIT("open value")
        rc = read(fd, buf, 2);       if (rc != 2) ERREXIT("read value")
        xfds[0].fd      = fd;
        xfds[0].events   = POLLPRI | POLLERR;
        xfds[0].revents  = 0;
        rc = poll(xfds, 1, -1);
        overflow=1;
        close(fd);
        if(rc == -1) ERREXIT("poll value")
        fn = "/sys/class/gpio/unexport";
        fd = open(fn, O_WRONLY); if(fd < 0)  ERREXIT("open unexport")
        rc = write(fd, "227", 3); if(rc != 3) ERREXIT("write unexport")
        close(fd);
        goto start3;
  return 0;
}
void*thread_routine2()  //νήμα για διακοπή κυκλώματος μέτρησης
{
       struct pollfd xfds[1];
        const char *fn;
        char buf[4];
        int rc;
        int fd;
start1:
        fn = "/sys/class/gpio/export";
        fd = open(fn, O_WRONLY); if(fd < 0)  ERREXIT("open export")
        rc = write(fd, "226", 3); if(rc != 3) ERREXIT("write export")
        close(fd);
        fn = "/sys/class/gpio/gpio226/direction";
        fd = open(fn, O_RDWR);  if(fd < 0)  ERREXIT("open direction")
        rc = write(fd, "in", 3);if(rc != 3) ERREXIT("write direction")
        close(fd);
        fn = "/sys/class/gpio/gpio226/edge";
        fd = open(fn, O_RDWR);       if(fd < 0)  ERREXIT("open edge")
        rc = write(fd, "rising", 8); if(rc != 8) ERREXIT("write edge")
        rc = lseek(fd, 0, SEEK_SET);  if(rc < 0)  ERREXIT("lseek edge")
        rc = read(fd, buf, 10);       if(rc <= 0) ERREXIT("read edge")
        buf[rc] = '\0';
        close(fd);
        fn = "/sys/class/gpio/gpio226/value";
        fd = open(fn, O_RDWR);  if(fd < 0)  ERREXIT("open value")
        rc = read(fd, buf, 2);       if (rc != 2) ERREXIT("read value")
        xfds[0].fd      = fd;
        xfds[0].events   = POLLPRI | POLLERR;
        xfds[0].revents  = 0;
        rc = poll(xfds, 1, -1);
```

```
        z=1;
        close(fd);
        if(rc == -1) ERREXIT("poll value")
        fn = "/sys/class/gpio/unexport";
        fd = open(fn, O_WRONLY); if(fd < 0)  ERREXIT("open unexport")
        rc = write(fd, "226", 3); if(rc != 3) ERREXIT("write unexport")
        close(fd);
       goto start1;
       return 0;
}
#define MAP_SIZE 8192UL // μέγεθος σελίδας για την 'mmap' συνάρτηση
#define MAP_MASK (MAP_SIZE - 1)
volatile int *pu; //καθολικές μεταβλητές
volatile int b,set1,y2;
main ()
{
int  x;
x = 0;
int        addr_textvga = 0x80000c00; //διεύθυνση κυκλώματος μετρήσεων
void      *mapped_base;
void      *mapped_dev_base;
int i=0;
int  volatile *pcid;
int  volatile  cid;
int memfd;
char mask=0x1f;
int y3=0;
int y4=0;
int t;
int points=0;
int delay=0;
int window=0;
int x5;
double x2,x4;
double x3=1;
char stop;
pthread_t thread_id;
int status;
FILE *ptr_file;
memfd = open("/dev/mem", O_RDWR | O_SYNC); //'mmap' συνάρτηση
mapped_base = mmap(0, MAP_SIZE, PROT_READ|PROT_WRITE, MAP_SHARED,
memfd, addr_textvga & ~MAP_MASK);
if (mapped_base == MAP_FAILED)
errx(1, "mmap failure");
mapped_dev_base = mapped_base + (addr_textvga & MAP_MASK);
pu = mapped_dev_base;
pcid = (int *) (((void *) pu) + 0xf0704);
struct my_play_regs_t  * my_play_regs = (struct my_play_regs_t  *)pu;
struct my_play_regs_t {
volatile int write;
volatile int read;
};
```

```
status=pthread_create(&thread_id,NULL,thread_routine2,NULL); //δημιουργία νημάτων
status=pthread_create(&thread_id,NULL,thread_routine3,NULL);
cid = *pcid;
ptr_file =fopen("/opt/channel1.txt", "w"); //άνοιγμα φακέλων
fclose(ptr_file);
ptr_file =fopen("/opt/channel2.txt", "w");
fclose(ptr_file);
ptr_file =fopen("/opt/channel3.txt", "w");
fclose(ptr_file);
ptr_file =fopen("/opt/channel4.txt", "w");
fclose(ptr_file);
system("clear");
loop:
printf (ANSI_COLOR_MAGENTA"Choose number of points\n");
scanf("%d",&points);
printf (ANSI_COLOR_MAGENTA"Choose time delay between measurements
(microseconds 10^-6 s)\n");
scanf("%d",&delay);
printf (ANSI_COLOR_MAGENTA"Choose time window\n");
scanf("%d",&window);
y3=(window)&mask;
x3=1;
for ( i=0; i < window; i++)
     {
   x3=x3*2;
     }
x2=0.00000002;
for (t = 0; t < points; t++)
        {
//κανάλι 0//
y4=0&mask;
set1=((00001<<15) | (00001<<10) | (y3<<5) | y4); //αποστολή εντολών στο κύκλωμα
y2=((00001<<15) | (00000<<10) | (y3<<5) | y4);
my_play_regs->write=set1;
my_play_regs->write=y2;
while (z==0){
         }
z=0;
if (overflow==1){
printf (ANSI_COLOR_RED"    OVERFLOW HAPPENED TO CHANNEL 1 REDUCE
TIME WINDOW!!!\n"ANSI_COLOR_RESET );
        goto end;
             }
b=my_play_regs->read;
b=b&0x0000ffff;
x4=x3*x2;
x4=x4/b;
x4=1/x4;
x5=x4; //υπολογισμός συχνότητας
ptr_file =fopen("/opt/channel1.txt", "a"); //αποθήκευση στον φάκελο
fprintf(ptr_file," %d  \n",x5);
fclose(ptr_file);
```

```
//κανάλι 1//
y4=1&mask;
set1=((00001<<15) | (00001<<10) | (y3<<5) | y4);
y2=((00001<<15) | (00000<<10) | (y3<<5) | y4);
my_play_regs->write=set1;
my_play_regs->write=y2;
while (z==0){  //μέχρι να ολοκληρωθεί η μέτρηση
        }
z=0;
if (overflow==1) //περίπτωση υπερχείλισης
            {
             printf (ANSI_COLOR_RED"    OVERFLOW HAPPENED TO CHANNEL 2
REDUCE TIME WINDOW!!!\n" );
            goto end;
            }
b=my_play_regs->read;
b=b&0x0000ffff;
x4=x3*x2;
x4=x4/b;
x4=1/x4;
x5=x4;
ptr_file =fopen("/opt/channel2.txt", "a");
fprintf(ptr_file," %d \n",x5);
fclose(ptr_file);
//κανάλι 2//
y4=2&mask;
set1=((00001<<15) | (00001<<10) | (y3<<5) | y4);
y2=((00001<<15) | (00000<<10) | (y3<<5) | y4);
my_play_regs->write=set1;
my_play_regs->write=y2;
while (z==0){
       }
      z=0;
   if (overflow==1){
  printf (ANSI_COLOR_RED"    OVERFLOW HAPPENED TO CHANNEL 3 REDUCE
TIME WINDOW!!!\n" );
     goto end;
         }
b=my_play_regs->read;
b=b&0x0000ffff;
x4=x3*x2;
x4=x4/b;
x4=1/x4;
x5=x4;
ptr_file =fopen("/opt/channel3.txt", "a");
fprintf(ptr_file," %d\n",x5);
fclose(ptr_file);
//κανάλι 3//
y4=3&mask;
set1=((00001<<15) | (00001<<10) | (y3<<5) | y4);
y2=((00001<<15) | (00000<<10) | (y3<<5) | y4);
my_play_regs->write=set1;
```

```
my_play_regs->write=y2;
while (z==0){
            }
          z=0;
   if (overflow==1){
           printf (ANSI_COLOR_RED"    OVERFLOW HAPPENED TO CHANNEL 4
REDUCE TIME WINDOW!!!\n" );
        goto end;
            }
b=my_play_regs->read;
b=b&0x0000ffff;
x4=x3*x2;
x4=x4/b;
x4=1/x4;
x5=x4;
ptr_file =fopen("/opt/channel4.txt", "a");
fprintf(ptr_file," %d\n",x5);
fclose(ptr_file);
printf (ANSI_COLOR_GREEN"  PROGRESS=%d/%d\n",t,points);
usleep(delay);
               }
system("clear"); //καθάρισμα οθόνης
printf ("end of measurements\n");
end:
printf ("Do you want to start a new measure? press 'y' or 'n'\n");
scanf("   %c",&stop);
if (stop =='y'){
   system("clear");
   goto loop;
      }
system("echo 226 > /sys/class/gpio/unexport"); //απενεργοποίηση διακοπών
system("echo 227 > /sys/class/gpio/unexport");
printf ("\n"ANSI_COLOR_RESET);
system("clear");
}
```

Β) Κώδικας για τον έλεγχο των καναλιών

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <pthread.h>
#include <unistd.h>      /* For read(), close() */
#include <fcntl.h>       /* For open() */
#include <errno.h>       /* For errno */
#include <sys/poll.h>    /* For poll() */
#include <signal.h>
#include <sys/ioctl.h>
#include <sys/mman.h>
#include <stdbool.h>
#include <err.h>
```

```
#include <math.h>
#include <malloc.h>
#include <time.h>
volatile int b1,b2,b3,b4,set1,y2,loop=0;
volatile int exit_=0;
volatile int z=0;
volatile int overflow=0;
#define ERREXIT(str) {printf("err %s, %s\n", str, strerror(errno)); return -1;}
#define ANSI_COLOR_RED     "\x1b[31m"
#define ANSI_COLOR_GREEN   "\x1b[32m"
#define ANSI_COLOR_YELLOW  "\x1b[33m"
#define ANSI_COLOR_BLUE    "\x1b[34m"
#define ANSI_COLOR_MAGENTA "\x1b[35m"
#define ANSI_COLOR_CYAN    "\x1b[36m"
#define ANSI_COLOR_RESET   "\x1b[0m"
//για την υπερχειλιση//
void*thread_routine3()
{
        struct pollfd xfds[1];
         const char *fn;
         char buf[4];
         int rc;
         int fd;
start3:
         fn = "/sys/class/gpio/export";
         fd = open(fn, O_WRONLY); if(fd < 0)  ERREXIT("open export")
         rc = write(fd, "227", 3); if(rc != 3) ERREXIT("write export")
         close(fd);
         fn = "/sys/class/gpio/gpio227/direction";
         fd = open(fn, O_RDWR);  if(fd < 0)  ERREXIT("open direction")
         rc = write(fd, "in", 3);if(rc != 3) ERREXIT("write direction")
         close(fd);
         fn = "/sys/class/gpio/gpio227/edge";
         fd = open(fn, O_RDWR);        if(fd < 0)  ERREXIT("open edge")
         rc = write(fd, "rising", 8); if(rc != 8) ERREXIT("write edge")
         rc = lseek(fd, 0, SEEK_SET);  if(rc < 0)  ERREXIT("lseek edge")
         rc = read(fd, buf, 10);       if(rc <= 0) ERREXIT("read edge")
         buf[rc] = '\0';
         close(fd);
         fn = "/sys/class/gpio/gpio227/value";
         fd = open(fn, O_RDWR);  if(fd < 0)  ERREXIT("open value")
         rc = read(fd, buf, 2);       if (rc != 2) ERREXIT("read value")
         xfds[0].fd       = fd;
         xfds[0].events   = POLLPRI | POLLERR;
         xfds[0].revents  = 0;
             rc = poll(xfds, 1, -1);
         overflow=1;
         close(fd);
         if(rc == -1) ERREXIT("poll value")
         fn = "/sys/class/gpio/unexport";
         fd = open(fn, O_WRONLY); if(fd < 0)  ERREXIT("open unexport")
         rc = write(fd, "227", 3); if(rc != 3) ERREXIT("write unexport")
```

```
        close(fd);
        goto start3;
           return 0;
            }
void*thread_routine2() //για την διακοπή του κυκλώματος μέτρησης//
{       struct pollfd xfds[1];
        const char *fn;
        char buf[4];
        int rc;
        int fd;
start1:
        fn = "/sys/class/gpio/export";
        fd = open(fn, O_WRONLY); if(fd < 0)  ERREXIT("open export")
        rc = write(fd, "226", 3); if(rc != 3) ERREXIT("write export")
        close(fd);
        fn = "/sys/class/gpio/gpio226/direction";
        fd = open(fn, O_RDWR);  if(fd < 0)  ERREXIT("open direction")
        rc = write(fd, "in", 3);if(rc != 3) ERREXIT("write direction")
        close(fd);
        fn = "/sys/class/gpio/gpio226/edge";
        fd = open(fn, O_RDWR);       if(fd < 0)  ERREXIT("open edge")
        rc = write(fd, "rising", 8); if(rc != 8) ERREXIT("write edge")
        rc = lseek(fd, 0, SEEK_SET);  if(rc < 0)  ERREXIT("lseek edge")
        rc = read(fd, buf, 10);       if(rc <= 0) ERREXIT("read edge")
        buf[rc] = '\0';
        close(fd);
        fn = "/sys/class/gpio/gpio226/value";
        fd = open(fn, O_RDWR);  if(fd < 0)  ERREXIT("open value")
        rc = read(fd, buf, 2);       if (rc != 2) ERREXIT("read value")
        xfds[0].fd       = fd;
        xfds[0].events   = POLLPRI | POLLERR;
        xfds[0].revents  = 0;
        rc = poll(xfds, 1, -1);
        z=1;
        close(fd);
        if(rc == -1) ERREXIT("poll value")
        fn = "/sys/class/gpio/unexport";
        fd = open(fn, O_WRONLY); if(fd < 0)  ERREXIT("open unexport")
        rc = write(fd, "226", 3); if(rc != 3) ERREXIT("write unexport")
        close(fd);
       goto start1;;
        return 0;
}
volatile unsigned char coordinate[4];
volatile char screen_pressed;
volatile int b,ready,sum,set1,y2,epan;
#define MAP_SIZE 8192UL
#define MAP_MASK (MAP_SIZE - 1)
#define ICLEAR 3
#define IMASK  16
#define IFORCE 2
#define ILEVEL 0
```

```
volatile int *pu;
main ()
{
int  x;
x = 0;
int       addr_textvga = 0x80000c00;
void          *mapped_base;
void          *mapped_dev_base;
int i=0;
int  volatile *pcid;
int  volatile  cid;
int           memfd;
char mask=0x1f;
int y3=0;
int y4=0;
int t;
int points=0;
int delay=0;
int window=0;
int x5_1=0;
int x5_2=0;
int x5_3=0;
int x5_4=0;
double x2,x4;
double x3=1;
char stop;
pthread_t thread_id;
int status;
int overflow1=0;
int overflow2=0;
int overflow3=0;
int overflow4=0;
FILE *ptr_file;
memfd = open("/dev/mem", O_RDWR | O_SYNC);
mapped_base = mmap(0, MAP_SIZE, PROT_READ|PROT_WRITE, MAP_SHARED,
memfd, addr_textvga & ~MAP_MASK);
if (mapped_base == MAP_FAILED)
     errx(1, "mmap failure");
     mapped_dev_base = mapped_base + (addr_textvga & MAP_MASK);
     pu = mapped_dev_base;
     pcid = (int *) (((void *) pu) + 0xf0704);
struct my_play_regs_t  * my_play_regs = (struct my_play_regs_t  *)pu;
struct my_play_regs_t {
volatile int write;
volatile int read;
};
status=pthread_create(&thread_id,NULL,thread_routine2,NULL);
status=pthread_create(&thread_id,NULL,thread_routine3,NULL);
cid = *pcid;
system("clear");
loop:
printf (ANSI_COLOR_MAGENTA"Choose time window (19-32)\n");
```

```
scanf("%d",&window);
y3=(window)&mask;
x3=1;
for ( i=0; i < window; i++)
            {
        x3=x3*2;
              }
x2=0.00000002;
struct timeval  tv1, tv2;
gettimeofday(&tv1, NULL);
//κανάλι 0//
y4=0&mask;
set1=((00001<<15) | (00001<<10) | (y3<<5) | y4);
y2=((00001<<15) | (00000<<10) | (y3<<5) | y4);
my_play_regs->write=set1;
my_play_regs->write=y2;
while (z==0){
}
z=0;
if (overflow==1){
    overflow1=1;
    overflow=0;
     }
b1=my_play_regs->read;
b1=b1&0x0000ffff;
x4=x3*x2;
x4=x4/b1;
x4=1/x4;
x5_1=x4;
ptr_file =fopen("/opt/test1.txt", "a");
fprintf(ptr_file," %d   %d \n",t,x5_1);
fclose(ptr_file);
//κανάλι 1//
y4=1&mask;
set1=((00001<<15) | (00001<<10) | (y3<<5) | y4);
y2=((00001<<15) | (00000<<10) | (y3<<5) | y4);
my_play_regs->write=set1;
my_play_regs->write=y2;
while (z==0){
}
z=0;
if (overflow==1){
     overflow2=1;
     overflow=0;
      }
b2=my_play_regs->read;
b2=b2&0x0000ffff;
x4=x3*x2;
x4=x4/b2;
x4=1/x4;
x5_2=x4;
ptr_file =fopen("/opt/test2.txt", "a");
```

```
fprintf(ptr_file," %d   %d \n",t,x5_2);
fclose(ptr_file);
//κανάλι 2//
y4=2&mask;
set1=((00001<<15) | (00001<<10) | (y3<<5) | y4);
y2=((00001<<15) | (00000<<10) | (y3<<5) | y4);
my_play_regs->write=set1;
my_play_regs->write=y2;
while (z==0){
}
z=0;
if (overflow==1){
        overflow3=1;
        overflow=0;
                }
b3=my_play_regs->read;
b3=b3&0x0000ffff;
x4=x3*x2;
x4=x4/b3;
x4=1/x4;
x5_3=x4;
ptr_file =fopen("/opt/test3.txt", "a");
fprintf(ptr_file," %d   %d \n",t,x5_3);
fclose(ptr_file);
//κανάλι 3//
y4=3&mask;
set1=((00001<<15) | (00001<<10) | (y3<<5) | y4);
y2=((00001<<15) | (00000<<10) | (y3<<5) | y4);
my_play_regs->write=set1;
my_play_regs->write=y2;
while (z==0){
}
z=0;
if (overflow==1){
      overflow4=1;
      overflow=0;
}
b4=my_play_regs->read;
b4=b4&0x0000ffff;
x4=x3*x2;
x4=x4/b4;
x4=1/x4;
x5_4=x4;
ptr_file =fopen("/opt/test4.txt", "a");
fprintf(ptr_file," %d   %d \n",t,x5_4);
fclose(ptr_file);
printf("end of measure\n");
gettimeofday(&tv2, NULL);
system("clear");
printf (ANSI_COLOR_MAGENTA"Total time for one iteration= %f seconds\n",
(double) (tv2.tv_usec - tv1.tv_usec) / 1000000 + (double) (tv2.tv_sec - tv1.tv_sec));
printf("\n");
```

```
if(overflow1==0){
printf (ANSI_COLOR_GREEN" channel1 , counter=%d, frequency=%d,
overflow=%d\n", b1,x5_1,overflow1) ;
printf("\n");
        }
else if(overflow1==1){
printf (ANSI_COLOR_RED" channel1 , counter=%d, frequency=%d, overflow=%d\n",
b1,x5_1,overflow1);
printf("\n");
if(overflow2==0){
printf (ANSI_COLOR_GREEN" channel2 , counter=%d, frequency=%d,
overflow=%d\n", b2,x5_2,overflow2) ;
printf("\n");
}
else if(overflow2==1){
printf (ANSI_COLOR_RED" channel2 , counter=%d, frequency=%d, overflow=%d\n",
b2,x5_2,overflow2);
printf("\n");
    }
if(overflow3==0){
printf (ANSI_COLOR_GREEN" channel3 , counter=%d, frequency=%d,
overflow=%d\n", b3,x5_3,overflow3) ;
printf("\n");
        }
else if(overflow3==1){
printf (ANSI_COLOR_RED" channel3 , counter=%d, frequency=%d, overflow=%d\n",
b3,x5_3,overflow3);
printf("\n");
}
if(overflow4==0){
printf (ANSI_COLOR_GREEN" channel4 , counter=%d, frequency=%d,
overflow=%d\n", b4,x5_4,overflow4) ;
printf("\n");
}
else if(overflow4==1){
printf (ANSI_COLOR_RED" channel4 , counter=%d, frequency=%d, overflow=%d\n",
b4,x5_4,overflow4);
printf("\n");
        }
printf(ANSI_COLOR_YELLOW"time window=%d\n",window);
end:
printf("\n");
printf("\n");
overflow1=0;
overflow2=0;
overflow3=0;
overflow4=0;
printf (ANSI_COLOR_BLUE"Do you want to start a new measure? press 'y' or 'n'\n");
scanf("   %c",&stop);
if (stop =='y'){
      system("clear");
       goto loop;
```

```
 }
system("echo 226 > /sys/class/gpio/unexport");
system("echo 227 > /sys/class/gpio/unexport");
printf (ANSI_COLOR_RESET"\n");
system("clear");
}
```

Γ) Κώδικας με οθόνη “touch screen”

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <pthread.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/poll.h>
#include <signal.h>
#include <sys/ioctl.h>
#include <sys/mman.h>
#include <stdbool.h>
#include <err.h>
#include <math.h>
#include <malloc.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <net/if.h>
#include <unistd.h>
#include <arpa/inet.h>
#define MAP_SIZE 8192UL
#define MAP_MASK (MAP_SIZE - 1)
#define ICLEAR 3
#define IMASK  16
#define IFORCE 2
#define ILEVEL 0
#define ANSI_COLOR_RED     "\x1b[31m"/* χρώματα */
#define ANSI_COLOR_GREEN   "\x1b[32m"
#define ANSI_COLOR_YELLOW  "\x1b[33m"
#define ANSI_COLOR_BLUE    "\x1b[34m"
#define ANSI_COLOR_MAGENTA "\x1b[35m"
#define ANSI_COLOR_CYAN    "\x1b[36m"
#define ANSI_COLOR_RESET   "\x1b[0m"
#define ERREXIT(str) {printf("err %s, %s\n", str, strerror(errno)); return -1;}
/* Register fields */
/* Control register */
#define CTR_EN  (1 << 7)  /* Enable core */
#define CTR_IEN (1 << 6)  /* Interrupt enable */
/* Command register */
#define CR_STA (1 << 7)   /* Generate start condition */
#define CR_STO (1 << 6)   /* Generate stop condition */
#define CR_RD  (1 << 5)   /* Read from slave */
#define CR_WR  (1 << 4)   /* Write to slave */
#define CR_ACK (1 << 3)   /* ACK, when a receiver send ACK (ACK = 0)
                             or NACK (ACK = 1) */
#define CR_IACK (1 << 0)  /* Interrupt acknowledge */
/* Status register */
#define SR_RXACK (1 << 7) /* Receibed acknowledge from slave */
#define SR_BUSY  (1 << 6) /* I2C bus busy */
#define SR_AL    (1 << 5) /* Arbitration lost */
#define SR_TIP   (1 << 1) /* Transfer in progress */
#define SR_IF    (1 << 0) /* Interrupt flag */
/* Reset values */
#define PRER_RESVAL 0xffff
#define CTR_RESVAL  0
#define RXR_RESVAL  0
#define SR_RESVAL   0
#define PRESCALER   0x0063
#define I2C_MTL_ADDR 0x08
struct i2cmstregs {
volatile unsigned int prer;
volatile unsigned int ctr;
volatile unsigned int xr;
```

```
volatile unsigned int csr;
};
///Συνάρτηση για τον εντοπισμό των συντεταγμένων της οθόνης
int i2cmst_tes (int addr, unsigned char *coordinate)
{
  int i;
  struct i2cmstregs *regs;
  regs = (struct i2cmstregs *) addr;
  regs->prer = PRESCALER;
  regs->ctr = CTR_EN;
  for (i=0; i<4; i++) {
      /* Address multi touch controller i2c slave for write */
     regs->xr = I2C_MTL_ADDR << 1;
     regs->csr = CR_STA | CR_WR;
     while (regs->csr & SR_TIP)   //wait until transfer complete
       ;
     /* Select register 3 */
     regs->xr = 0x03 + i;
     regs->csr = CR_WR;
     while (regs->csr & SR_TIP)   //wait until transfer complete
       ;
     /* Address multi touch controller i2c slave for reading */
     regs->xr = (I2C_MTL_ADDR << 1) | 1;
     regs->csr = CR_STA | CR_WR;

     while (regs->csr & SR_TIP)   //wait until transfer complete
       ;
     regs->csr = CR_RD | CR_STO | CR_ACK;
     while (regs->csr & SR_TIP)   //wait until transfer complete
       ;
     coordinate[i] = (unsigned char)regs->xr;
  }
return 0;
}
//καθολικές μεταβλητές
volatile int *pu;
volatile unsigned char coordinate[4];
volatile int z,z1,overflow=0;
//νήμα διακοπής οθόνης
void*thread_routine(void*arg)
{
            struct pollfd xfds[1];
             const char *fn;
             char buf[4];
             int rc;
             int fd;
start1:
             fn = "/sys/class/gpio/export";
             fd = open(fn, O_WRONLY); if(fd < 0)  ERREXIT("open export")
             rc = write(fd, "225", 3); if(rc != 3) ERREXIT("write export")
             close(fd);
             fn = "/sys/class/gpio/gpio225/direction";
             fd = open(fn, O_RDWR);  if(fd < 0)  ERREXIT("open direction")
             rc = write(fd, "in", 3);if(rc != 3) ERREXIT("write direction")
             close(fd);
             fn = "/sys/class/gpio/gpio225/edge";
             fd = open(fn, O_RDWR);        if(fd < 0)  ERREXIT("open edge")
             rc = write(fd, "falling", 8); if(rc != 8) ERREXIT("write edge")
             rc = lseek(fd, 0, SEEK_SET);  if(rc < 0)  ERREXIT("lseek edge")
             rc = read(fd, buf, 10);       if(rc <= 0) ERREXIT("read edge")
             buf[rc] = '\0';
             close(fd);
             fn = "/sys/class/gpio/gpio225/value";
             fd = open(fn, O_RDWR);  if(fd < 0)  ERREXIT("open value")
             rc = read(fd, buf, 2);       if (rc != 2) ERREXIT("read value")
             xfds[0].fd       = fd;
             xfds[0].events   = POLLPRI | POLLERR;
             xfds[0].revents  = 0;
             rc = poll(xfds, 1, -1);
             i2cmst_tes (pu, coordinate);
             z=1;
             close(fd);
             if(rc == -1) ERREXIT("poll value")
             fn = "/sys/class/gpio/unexport";
             fd = open(fn, O_WRONLY); if(fd < 0)  ERREXIT("open unexport")
             rc = write(fd, "225", 3); if(rc != 3) ERREXIT("write unexport")
             close(fd);
             rc = poll(0, 1, 1000);
```

```
            goto start1;

            return 0;
}
//νήμα υπερχείλισης
void*thread_routine3(void*arg)
{
           struct pollfd xfds[1];
            const char *fn;
            char buf[4];
            int rc;
            int fd;
start3:
           /* export */
            fn = "/sys/class/gpio/export";
            fd = open(fn, O_WRONLY); if(fd < 0)  ERREXIT("open export")
            rc = write(fd, "227", 3); if(rc != 3) ERREXIT("write export")
            close(fd);
            fn = "/sys/class/gpio/gpio227/direction";
            fd = open(fn, O_RDWR);  if(fd < 0)  ERREXIT("open direction")
            rc = write(fd, "in", 3);if(rc != 3) ERREXIT("write direction")
            close(fd);
            fn = "/sys/class/gpio/gpio227/edge";
            fd = open(fn, O_RDWR);        if(fd < 0)  ERREXIT("open edge")
            rc = write(fd, "rising", 8); if(rc != 8) ERREXIT("write edge")
            rc = lseek(fd, 0, SEEK_SET);  if(rc < 0)  ERREXIT("lseek edge")
            rc = read(fd, buf, 10);       if(rc <= 0) ERREXIT("read edge")
            buf[rc] = '\0';
            close(fd);
            fn = "/sys/class/gpio/gpio227/value";
            fd = open(fn, O_RDWR);  if(fd < 0)  ERREXIT("open value")
            rc = read(fd, buf, 2);       if (rc != 2) ERREXIT("read value")
            xfds[0].fd       = fd;
            xfds[0].events   = POLLPRI | POLLERR;
            xfds[0].revents  = 0;
            rc = poll(xfds, 1, -1);
            overflow=1;
            close(fd);
            if(rc == -1) ERREXIT("poll value")
            fn = "/sys/class/gpio/unexport";
            fd = open(fn, O_WRONLY); if(fd < 0)  ERREXIT("open unexport")
            rc = write(fd, "227", 3); if(rc != 3) ERREXIT("write unexport")
            close(fd);
            goto start3;
  return 0;
}

///νήμα για διακοπή κυκλώματος μέτρησης
void*thread_routine2(void*arg)
{
           struct pollfd xfds[1];
            const char *fn;
            char buf[4];
            int rc;
            int fd;
start2:
            fn = "/sys/class/gpio/export";
            fd = open(fn, O_WRONLY); if(fd < 0)  ERREXIT("open export")
            rc = write(fd, "226", 3); if(rc != 3) ERREXIT("write export")
            close(fd);
            fn = "/sys/class/gpio/gpio226/direction";
            fd = open(fn, O_RDWR);  if(fd < 0)  ERREXIT("open direction")
            rc = write(fd, "in", 3);if(rc != 3) ERREXIT("write direction")
            close(fd);
            fn = "/sys/class/gpio/gpio226/edge";
            fd = open(fn, O_RDWR);        if(fd < 0)  ERREXIT("open edge")
            rc = write(fd, "rising", 8); if(rc != 8) ERREXIT("write edge")
            rc = lseek(fd, 0, SEEK_SET);  if(rc < 0)  ERREXIT("lseek edge")
            rc = read(fd, buf, 10);       if(rc <= 0) ERREXIT("read edge")
            buf[rc] = '\0';
            close(fd);
            fn = "/sys/class/gpio/gpio226/value";
            fd = open(fn, O_RDWR);  if(fd < 0)  ERREXIT("open value")
            rc = read(fd, buf, 2);       if (rc != 2) ERREXIT("read value")
            xfds[0].fd       = fd;
            xfds[0].events   = POLLPRI | POLLERR;
            xfds[0].revents  = 0;
            rc = poll(xfds, 1, -1);
```

```
            z1=1;
            close(fd);
            if(rc == -1) ERREXIT("poll value")
            fn = "/sys/class/gpio/unexport";
            fd = open(fn, O_WRONLY); if(fd < 0)  ERREXIT("open unexport")
            rc = write(fd, "226", 3); if(rc != 3) ERREXIT("write unexport")
            close(fd);
           goto start2;
            return 0;
}
#define MAP_SIZE 8192UL  //παράμετροι και σταθερές
#define MAP_MASK (MAP_SIZE - 1)
#define ICLEAR 3
#define IMASK  16
#define IFORCE 2
#define ILEVEL 0

main (int argc,char*argv[])
{
pthread_t thread_id; //δήλωση μεταβλητών
int status;
int  x;
int hardware_config=0;
int calib_menu=0;
int start_menu=0;
int channel_sel=0;
int channel1=0;
int channel2=0;
int point1=0;
int point2=0;
int point3=0;
int point4=0;
int point5=0;
int number_of_points=0;
int time_window=0;
int decision=0;
 x = 0;
int   addr_textvga3 = 0x80000800;
int       addr_textvga = 0x80000c00;
void          *mapped_base;
void          *mapped_dev_base;
int  volatile *pcid;
int  volatile  cid;
int           memfd;
int choice=0;
double sampling_time=0;
int time1=0;
int time2=0;
int time_total=0;
int i = 0;
int calculate=0;
int y2=0;
int y3=0;
int y4=0;
int set1=0;
int t=0;
int loop=0;
int b=0;
char mask=0x1f;
  //μεταβλητές για τον χρόνο καθυστέρησης
int delay=0;
int time_delay1=0;
int time_delay2=0;
int time_delay3=0;
int time_delay_sel=0;
///μεταβλητές για τις πράξεις
int sum=0;
int metrisi=0;
int store_points[99999],store_time[99999],store_channel[99999];
int x5,x5_endiameso;
double x2,x4,x4_endiameso;
double x3=1;
float sum_float[99999],diaspora[99999];
int sum_final[99999];
int calculate_decision=0;
int res[99999];
//απόκτηση ip διεύθυνσης
int fd;
```

```
struct ifreq ifr;
char iface[] = "eth0";
fd = socket(AF_INET, SOCK_DGRAM, 0);
        //Type of address to retrieve - IPv4 IP address
ifr.ifr_addr.sa_family = AF_INET;
        //Copy the interface name in the ifreq structure
strncpy(ifr.ifr_name , iface , IFNAMSIZ-1);
ioctl(fd, SIOCGIFADDR, &ifr);
close(fd);

memfd = open("/dev/mem", O_RDWR | O_SYNC);  //mmap συνάρτηση
mapped_base = mmap(0, MAP_SIZE, PROT_READ|PROT_WRITE, MAP_SHARED, memfd, addr_textvga & ~MAP_MASK);
if (mapped_base == MAP_FAILED)
   errx(1, "mmap failure");
   mapped_dev_base = mapped_base + (addr_textvga & MAP_MASK);
   pu = mapped_dev_base;
   pcid = (int *) (((void *) pu) + 0xf0704);
struct my_play_regs_t  * my_play_regs = (struct my_play_regs_t  *)pu; //καθορισμός καταχωρητών για επικοινωνία με το κύκλωμα
struct my_play_regs_t {
volatile int write;   // write the registers
volatile int read; // read the result
};
system("echo 0 > /sys/class/graphics/fbcon/cursor_blink");
system("dd if=/dev/zero of=/dev/fb0 >& /dev/null");
memfd = open("/dev/mem", O_RDWR | O_SYNC);
mapped_base = mmap(0, MAP_SIZE, PROT_READ|PROT_WRITE, MAP_SHARED, memfd, addr_textvga3 & ~MAP_MASK);
if (mapped_base == MAP_FAILED)
     errx(1, "mmap failure");
     mapped_dev_base = mapped_base + (addr_textvga3 & MAP_MASK);
     pu = mapped_dev_base;
     pcid = (int *) (((void *) pu) + 0xf0704);
     cid = *pcid;
     overflow=0;
status=pthread_create(&thread_id,NULL,thread_routine,(void*)argv[0]);
status=pthread_create(&thread_id,NULL,thread_routine2,(void*)argv[0]);
status=pthread_create(&thread_id,NULL,thread_routine3,(void*)argv[0]);
system ("clear");
system ("printf loading....\n");
system ("gunzip /opt/calibration_menu.raw.gz");
system ("gunzip /opt/calculator.raw.gz");
system ("gunzip /opt/hardware_configuration.raw.gz");
system ("gunzip /opt/welcome.raw.gz");
system ("gunzip /opt/main_menu.raw.gz");
system(" cat /opt/welcome.raw > /dev/fb0");
FILE *ptr_file;
while(z==0){
        }
z=0;
z1=0;
system(" cat /opt/main_menu.raw > /dev/fb0");
main_menu:
system(" cat /opt/main_menu.raw > /dev/fb0");
/*****************main menu ********************/ //κεντρικό μενού
while((hardware_config==0)&&(calib_menu==0))
{
if (z == 1) {
        z = 0;
           if( (coordinate[0]*256+coordinate[1]>580) && (coordinate[0]*256+coordinate[1]<800) &&
(coordinate[2]*256+coordinate[3]>380) && (coordinate[2]*256+coordinate[3]<480))
             {
           goto end;
             }
         else if( (coordinate[0]*256+coordinate[1]>180) && (coordinate[0]*256+coordinate[1]<580) &&
(coordinate[2]*256+coordinate[3]>110) && (coordinate[2]*256+coordinate[3]<175))
             {
            system(" cat /opt/hardware_configuration.raw > /dev/fb0");
           hardware_config=1;
             }
            else if( (coordinate[0]*256+coordinate[1]>180) && (coordinate[0]*256+coordinate[1]<580) &&
(coordinate[2]*256+coordinate[3]>180) && (coordinate[2]*256+coordinate[3]<230))
             {
            system(" clear");
            system(" cat /opt/calibration_menu.raw > /dev/fb0");
            calib_menu=1;
             }
        }
}
```

```
/*****************main menu ********************/
/*****************hardware_config menu ********************/
if (hardware_config==1)
{
      while(1)
            {
          if (z==1)
                {
                   z=0;
                if( (coordinate[0]*256+coordinate[1]>580) && (coordinate[0]*256+coordinate[1]<800) &&
(coordinate[2]*256+coordinate[3]>380) && (coordinate[2]*256+coordinate[3]<480)) /*14400 baud rate */
                   {
                   hardware_config=0;
                   goto main_menu;
                   }
                }
          }
}
/*****************hardware_config menu ********************/
/*****************calibration menu ********************/
else if (calib_menu==1)
{
     while(1)
           {
           if ((z==1) && (calculate==0) && (decision==0) && (channel_sel==0)&& (number_of_points==0)&& (time_window==0) &&
(time_delay_sel==0))
                {
                z=0;
/*********calibration kentriko menu ******/
                   if( (coordinate[0]*256+coordinate[1]>580) && (coordinate[0]*256+coordinate[1]<800) &&
(coordinate[2]*256+coordinate[3]>380) && (coordinate[2]*256+coordinate[3]<480))
                   {
                   calib_menu=0;
                   goto main_menu;
                   }
              else if( (coordinate[0]*256+coordinate[1]>470) && (coordinate[0]*256+coordinate[1]<780) &&
(coordinate[2]*256+coordinate[3]>130) && (coordinate[2]*256+coordinate[3]<215))
                   {
                   system("clear");
                   system(" cat /opt/calibration_menu.raw > /dev/fb0");
                   printf ("\n" );
                   printf ("\n" );
                   printf ("\n" );
                   printf ("\n" );
                   printf ("\n" );
                   printf ("\n" );
                   printf ("\n" );
                   printf ("\n" );
                   printf ("\n" );
                   printf ("\n" );
                   printf ("\n" );
                   printf ("\n" );
                   printf ("\n" );
                   printf ("\n" );
                   printf ("\n" );
                   printf ("\n" );
                   printf ("\n" );
                   ptr_file =fopen("/opt/results.txt", "w");
                   fclose(ptr_file);
                   printf (ANSI_COLOR_GREEN"CALIBRATION FILE HAS BEEN RESET SUCCESSFULLY!\n");
                   }
          else if( (coordinate[0]*256+coordinate[1]>470) && (coordinate[0]*256+coordinate[1]<780) &&
(coordinate[2]*256+coordinate[3]>265) && (coordinate[2]*256+coordinate[3]<330))
                   {
                   time_delay_sel=1;
                   system("clear");
                   system(" cat /opt/calculator.raw > /dev/fb0");
                   printf (ANSI_COLOR_GREEN"Digits pressed for time delay selection ( Digit 3 = MSB and Digit 1 =  LSB):\n" );
                   printf ("\n" );
                   printf (ANSI_COLOR_GREEN"SUPPORTED TIME DELAY: 0-999 seconds:\n" );
                   printf ("\n" );
                   }
          else if( (coordinate[0]*256+coordinate[1]>10) && (coordinate[0]*256+coordinate[1]<340) &&
(coordinate[2]*256+coordinate[3]>120) && (coordinate[2]*256+coordinate[3]<173))
                   {
                   channel_sel=1;
                   system("clear");
                   system(" cat /opt/calculator.raw > /dev/fb0");
                   printf (ANSI_COLOR_GREEN"Digits pressed for channel selection ( Digit 2 = MSB and Digit 1 =  LSB):\n" );
```

```
                printf ("\n" );
                printf (ANSI_COLOR_GREEN"SUPPORTED CHANNELS: 0-3:\n" );
                printf ("\n" );
                }
        else if( (coordinate[0]*256+coordinate[1]>10) && (coordinate[0]*256+coordinate[1]<340) &&
(coordinate[2]*256+coordinate[3]>202) && (coordinate[2]*256+coordinate[3]<245))
                {
                number_of_points=1;
                system("clear");
                system(" cat /opt/calculator.raw > /dev/fb0");
                printf (ANSI_COLOR_GREEN"Digits pressed for number of points ( Digit 5 = MSB and Digit 1 =  LSB):\n" );
                printf ("\n" );
                printf (ANSI_COLOR_GREEN"SUPPORTED POINTS: 1-99999:\n" );
                printf ("\n" );
                }
        else if( (coordinate[0]*256+coordinate[1]>10) && (coordinate[0]*256+coordinate[1]<340) &&
(coordinate[2]*256+coordinate[3]>265) && (coordinate[2]*256+coordinate[3]<330))
                {
                time_window=1;
                system("clear");
                system(" cat /opt/calculator.raw > /dev/fb0");
                printf (ANSI_COLOR_GREEN"Digits pressed for time window selection ( Digit 2 = MSB and Digit 1 =  LSB):\n");
                printf ("\n" );
                printf (ANSI_COLOR_GREEN"SUPPORTED TIME WINDOW NUMBERS: 18-31:\n" );
                printf ("\n" );
                }
        else if( (coordinate[0]*256+coordinate[1]>270) && (coordinate[0]*256+coordinate[1]<480) &&
(coordinate[2]*256+coordinate[3]>385) && (coordinate[2]*256+coordinate[3]<460))
                {
                if ((point5*10000+point4*1000+point3*100+point2*10 + point1)==0)
                        {
                        system("clear");
                      system(" cat /opt/calibration_menu.raw > /dev/fb0");
                      printf (ANSI_COLOR_GREEN"CHOOSE NUMBER OF POINTS!!!\n" );
                      }
                else if ((time2*10 + time1)<=17 || (time2*10 + time1)>31 )
                      {
                      system("clear");
                      system(" cat /opt/calibration_menu.raw > /dev/fb0");
                      printf (ANSI_COLOR_GREEN"CHOOSE TIME WINDOW BETWEEN 18-31!!!\n" );
                      }
                else
                          {
                      calculate=1;
                      system("clear");
                      printf ("\n" );
                      printf ("\n" );
                      printf (ANSI_COLOR_CYAN"          ----------------------------------------------------------\n");
                      printf (ANSI_COLOR_CYAN" | THE CHOSEN CHANNEL IS  %d%d            |\n",channel2,channel1 );
                      printf (ANSI_COLOR_CYAN"                  ----------------------------------------------------------\n");
                      printf ("\n" );
                      printf (ANSI_COLOR_CYAN"                  ----------------------------------------------------------\n");
                      printf (ANSI_COLOR_CYAN"                  |  THE CHOSEN POINTS ARE  %d%d%d%d%d
|\n",point5,point4,point3,point2,point1 );
                      printf (ANSI_COLOR_CYAN"          ----------------------------------------------------------\n" );
                      printf ("\n" );
                      printf (ANSI_COLOR_CYAN"          ----------------------------------------------------------\n");
                      printf (ANSI_COLOR_CYAN"    | THE CHOSEN TIME WINDOW IS  %d%d          |\n",time2,time1 );
                      printf (ANSI_COLOR_CYAN"          ----------------------------------------------------------\n");
                      printf ("\n" );
                      printf (ANSI_COLOR_MAGENTA"   SAMPLING TIME FREQUENCY:  %e Hz \n",sampling_time);
                      printf ("\n" );
                      printf (ANSI_COLOR_MAGENTA"           SAMPLING TIME         :  %e s                  TIME DELAY:%d \n
",(1/sampling_time), (time_delay3*100+time_delay2*10+time_delay1));
                      printf ("\n" );
                      printf (ANSI_COLOR_YELLOW"  TOTAL TIME FOR MEASUREMENTSâ‰ˆ%e s
\n",(((time_delay3*100+time_delay2*10+time_delay1)+(1/sampling_time))*(point5*10000+point4*1000+point3*100+point2*10+point
1)));
                      printf ("\n" );
                      printf (ANSI_COLOR_BLUE"          ----------------------------------------------------------\n");
                            printf (ANSI_COLOR_BLUE"        DO YOU WANT TO START THE MEASUREMENT?        \n ");
                      printf (ANSI_COLOR_BLUE"          ----------------------------------------------------------\n");
                      printf ("\n" );
                      printf ("\n" );
                      printf (ANSI_COLOR_GREEN"       ---------"ANSI_COLOR_RED"               ---------\n" );
                      printf (ANSI_COLOR_GREEN"      YES   "ANSI_COLOR_RED"                  NO   \n" );
                      printf (ANSI_COLOR_GREEN"    ---------"ANSI_COLOR_RED"    ---------\n" ANSI_COLOR_RESET);
                      }
```

```
                    }
                  }
        else if  ((z==1) && (calculate==1))
                 {
                   z=0;
                 if  (calculate_decision==0){
                   if( (coordinate[0]*256+coordinate[1]>240) && (coordinate[0]*256+coordinate[1]<340) &&
(coordinate[2]*256+coordinate[3]>395) && (coordinate[2]*256+coordinate[3]<480))
                    {
                     system("clear");
                     printf (ANSI_COLOR_BLUE" PLEASE WAIT SENSOR-METRICS TAKES METRICS...        \n
"ANSI_COLOR_RESET);
                     loop=point5*10000+point4*1000+point3*100+point2*10+point1;
                     store_points[metrisi]=loop;
                     y3=(time2*10+time1)&mask;
                     store_time[metrisi]=y3;
                      y4=(channel2*10+channel1)&mask;
                      store_channel[metrisi]=y4;
                      delay=(time_delay3*100+time_delay2*10+time_delay1);
                      set1=((00001<<15) | (00001<<10) | (y3<<5) | y4);
                      y2=((00001<<15) | (00000<<10) | (y3<<5) | y4);
                       for ( i=0; i<store_time[metrisi]; i++)
                                {
                              x3=x3*2;
                               }
                      x2=0.00000002;
                      ptr_file =fopen("/opt/results.txt", "a");
                      fprintf(ptr_file," MEASUREMENT     CHANNEL      POINTS      TIME-WINDOW      COUNTER-POINTS
FREQUENCY         DEVIATION(COUNTER)\n");
                      fprintf(ptr_file," ----------------------------------------------------------------------------------------------------------\n");
                      fprintf(ptr_file,"MEASURE: %d  TIME DELAY: %d s\n",metrisi,delay);
                      fprintf(ptr_file,"                %d          %d           %d                                        \n", channel1,
(point5*10000+point4*1000+point3*100+point2*10 + point1), (time2*10 + time1));
                      fclose(ptr_file);
                      for (t = 0; t < loop; t++)
                              {
                              my_play_regs->write=set1;
                              my_play_regs->write=y2;
                              while (z1==0)
                                      {
                                    }
                              z1=0;
                              b=my_play_regs->read;
                              b=b&0x0000ffff;
                              res[t]=b;
                              sum=sum+b;
                                //αποθήκευση σε φακέλους//
                              x4_endiameso=x3*x2;
                              x4_endiameso=x4_endiameso/(res[t]);
                              x4_endiameso=1/x4_endiameso;
                              x5_endiameso=x4_endiameso;
                              ptr_file =fopen("/opt/results.txt", "a");
                              fprintf(ptr_file,"    %d                    %d          %d              \n",t,res[t],x5_endiameso);
                              fclose(ptr_file);
                              system("clear");
                              printf (ANSI_COLOR_GREEN"counter = %d , METRIC=%d \n", b, t );
                              sleep(delay);
                              }
                              printf ("\n" );
                              printf (ANSI_COLOR_GREEN"SAVING RESULTS\n" );
                                   if (overflow==0){
                              sum_final[metrisi]=sum;
                              sum_float[metrisi]=sum/(store_points[metrisi]);
                              sum_final[metrisi]=sum_final[metrisi]/(store_points[metrisi]);
                              diaspora[metrisi]=0;
                              x2=0.00000002;
                              x4=x3*x2;
                              x4=x4/(sum_final[metrisi]);
                              x4=1/x4;
                              x5=x4;
                              for ( i=0; i<store_points[metrisi]; i++)
                                       {
                                     diaspora[metrisi]= diaspora[metrisi] + ((res[i]-sum_final[metrisi])*(res[i]-sum_final[metrisi]));
                                      }
                              diaspora[metrisi]=sqrt(diaspora[metrisi]/store_points[metrisi]);
                              ptr_file =fopen("/opt/results.txt", "a");
                            fprintf(ptr_file,"----------------------------------------------------------------------------------------------------------\n ");
                            fprintf(ptr_file,"FINAL RESULTS: \n ");
```

```
                                fprintf(ptr_file," MEASUREMENT     CHANNEL     POINTS     TIME-WINDOW     COUNTER-POINTS
FREQUENCY        DEVIATION(COUNTER)\n");
                                fprintf(ptr_file,"-------------------------------------------------------------------------------------------------------\n");
                                fprintf(ptr_file,"                %d          %d          %d              %d              %d              %f
\n",channel1, (point5*10000 + point4*1000 + point3*100 + point2*10 + point1), (time2*10 + time1),(sum/store_points[metrisi])
,x5,diaspora[metrisi]);
                            fprintf(ptr_file,"-------------------------------------------------------------------------------------------------------\n");
                            fclose(ptr_file);
                               system("clear");
                            printf ("\n" );
                            printf ("\n" );
                            printf (ANSI_COLOR_BLUE"              ------------------------------------------------------------\n");
                            printf (ANSI_COLOR_YELLOW"                                    RESULTS                      \n ");
                            printf (ANSI_COLOR_BLUE"              ------------------------------------------------------------\n");
                            printf ("\n" );
                            printf (ANSI_COLOR_MAGENTA"                    FREQUENCY = %d Hz\n",x5) ;
                                 printf ("\n" );
                               printf (ANSI_COLOR_MAGENTA"                    DEVIATION = %f\n",diaspora[metrisi]) ;
                            printf ("\n" );
                               printf (ANSI_COLOR_MAGENTA"                    COUNTER = %d\n",(sum/store_points[metrisi]));
                            printf (ANSI_COLOR_BLUE"              ------------------------------------------------------------\n");
                            printf (ANSI_COLOR_BLUE"                    DO YOU WANT TO RETURN TO MAIN MENU?  \n ");
                            printf (ANSI_COLOR_BLUE"              ------------------------------------------------------------\n");
                            printf ("\n" );
                            printf (ANSI_COLOR_GREEN"                              ---------"ANSI_COLOR_RED"             \n" );
                            printf (ANSI_COLOR_GREEN"                               YES   "ANSI_COLOR_RED"             \n" );
                            printf (ANSI_COLOR_GREEN ---------"ANSI_COLOR_RED"             \n" ANSI_COLOR_RESET);
                            printf ("\n" );
                            printf (ANSI_COLOR_RED"THE RESULTS ARE SAVED IN A TXT FILE NAMED RESULTS.TXT. IT
CAN BE TRANSFERRED TO YOUR COMPUTER \n ");
                            printf ("\n" );
                            printf (ANSI_COLOR_MAGENTA"FOR WINDOWS SYSTEM USE A PROGRAM LIKE WinSCP.'HOST
NAME' IS 'IP ADDRESS' AND 'USER NAME' IS 'root'  \n ");
                            printf (ANSI_COLOR_MAGENTA"THE RESULTS.TXT IS PLACED IN THE '/opt' folder\n" );
                            printf (ANSI_COLOR_RED"FOR LINUX SYSTEM EXECUTE THE FOLLOWING COMMAND IN A
TERMINAL  \n ");
                            printf (ANSI_COLOR_RED"          scp root@IP_ADDRESS:/opt/results.txt /some/local/directory\n");
                            printf ("\n" );
                            printf ("IP ADDRESS IS: %s\n",inet_ntoa(( (struct sockaddr_in *)&ifr.ifr_addr )->sin_addr) );
                            sum=0;
                            x3=1;
                            x4=0;
                            x5=0;
                            metrisi=metrisi+1;
                            calculate_decision=1;
                                    }
                        else if (overflow==1)
                            {
                        system("clear");
                        printf ("\n" );
                        printf ("\n" );
                        printf (ANSI_COLOR_RED"        OVERFLOW HAPPENED REDUCE TIME WINDOW!!!\n" );
                        printf ("\n" );
                        printf ("\n" );
                        printf ("\n" );
                        printf ("\n" );
                        printf ("\n" );
                        printf ("\n" );
                        printf (ANSI_COLOR_BLUE"              ------------------------------------------------------------\n");
                        printf (ANSI_COLOR_BLUE"                                    RETURN TO MAIN MENU?              \n ");
                        printf (ANSI_COLOR_BLUE"              ------------------------------------------------------------\n");
                        printf ("\n" );
                        printf ("\n" );
                        printf ("\n" );
                        printf (ANSI_COLOR_GREEN"                              ---------"ANSI_COLOR_RED"             \n" );
                        printf (ANSI_COLOR_GREEN"                               YES   "ANSI_COLOR_RED"             \n" );
                        printf (ANSI_COLOR_GREEN"              ---------"ANSI_COLOR_RED"             \n" ANSI_COLOR_RESET);
                        overflow=0;
                        sum=0;
                        calculate_decision=1;
                             }
                    }
    else if( (coordinate[0]*256+coordinate[1]>450) && (coordinate[0]*256+coordinate[1]<550) &&
(coordinate[2]*256+coordinate[3]>395) && (coordinate[2]*256+coordinate[3]<480))
                        {
                                calculate=0;
                    system("clear");
                    system(" cat /opt/calibration_menu.raw > /dev/fb0");
```

```
                        }
                    }
    else if  (calculate_decision==1)
                    {
                  if( (coordinate[0]*256+coordinate[1]>340) && (coordinate[0]*256+coordinate[1]<440) &&
(coordinate[2]*256+coordinate[3]>230) && (coordinate[2]*256+coordinate[3]<330))
                    {
                    calculate=0;
                    calculate_decision=0;
                    system("clear");
                    system(" cat /opt/calibration_menu.raw > /dev/fb0");
                        choice=0;
                    }
                  }
            }
    else if  ((z==1) && (calculate==0) && (decision==0) && (channel_sel==1) && (number_of_points==0)&& (time_window==0) &&
(time_delay_sel==0))
          {
          z=0;
          if( (coordinate[0]*256+coordinate[1]>580) && (coordinate[0]*256+coordinate[1]<800) &&
(coordinate[2]*256+coordinate[3]>380) && (coordinate[2]*256+coordinate[3]<480))
            {
            channel_sel=0;
            system(" cat /opt/calibration_menu.raw > /dev/fb0");
            }
    else if( (coordinate[0]*256+coordinate[1]>235) && (coordinate[0]*256+coordinate[1]<362) &&
(coordinate[2]*256+coordinate[3]>300) && (coordinate[2]*256+coordinate[3]<340)) /* 0 */
            {
            if (choice==0)
                {
                  channel1=0;
                choice=1;
                  printf (ANSI_COLOR_BLUE"Digit 1 = 0\n" );
                }
            else if (choice==1)
                {
                    channel2=0;
                  choice=0;
                printf (ANSI_COLOR_MAGENTA"Digit 2 = 0\n" );
                }
                }
   else if( (coordinate[0]*256+coordinate[1]>235) && (coordinate[0]*256+coordinate[1]<362) &&
(coordinate[2]*256+coordinate[3]>238) && (coordinate[2]*256+coordinate[3]<280)) /* 1 */
              {
            if (choice==0)
                {
                channel1=1;
                choice=1;
                  printf (ANSI_COLOR_BLUE"Digit 1 = 1\n" );
                }
            else if (choice==1)
                {
                channel2=1;
                choice=0;
                printf (ANSI_COLOR_MAGENTA"Digit 2 = 1\n" );
                }
            }
  else if( (coordinate[0]*256+coordinate[1]>370) && (coordinate[0]*256+coordinate[1]<470) &&
(coordinate[2]*256+coordinate[3]>238) && (coordinate[2]*256+coordinate[3]<280)) /* 2 */
            {
            if (choice==0)
                  {
                  channel1=2;
                choice=1;
                printf (ANSI_COLOR_BLUE"Digit 1 = 2\n" );
                  }
             else if (choice==1)
                  {
                channel2=2;
                choice=0;
                printf (ANSI_COLOR_MAGENTA"Digit 2 = 2\n" );
                  }
             }
  else if( (coordinate[0]*256+coordinate[1]>490) && (coordinate[0]*256+coordinate[1]<605) &&
(coordinate[2]*256+coordinate[3]>238) && (coordinate[2]*256+coordinate[3]<280)) /* 3 */
             {
             if (choice==0)
                {
```

```
                    channel1=3;
                    choice=1;
                    printf (ANSI_COLOR_BLUE"Digit 1 = 3\n" );
                    }
                  else if (choice==1)
                    {
                    channel2=3;
                    choice=0;
                    printf (ANSI_COLOR_MAGENTA"Digit 2 = 3\n" );
                    }
                  }
   else if( (coordinate[0]*256+coordinate[1]>235) && (coordinate[0]*256+coordinate[1]<360) &&
(coordinate[2]*256+coordinate[3]>175) && (coordinate[2]*256+coordinate[3]<225)) /* 4 */
                  {
                  if (choice==0)
                    {
                      channel1=4;
                    choice=1;
                    printf (ANSI_COLOR_BLUE"Digit 1 = 4\n" );
                    }
                  else if (choice==1)
                    {
                    channel2=4;
                    choice=0;
                    printf (ANSI_COLOR_MAGENTA"Digit 2 = 4\n" );
                    }
                  }
   else if( (coordinate[0]*256+coordinate[1]>370) && (coordinate[0]*256+coordinate[1]<470) &&
(coordinate[2]*256+coordinate[3]>175) && (coordinate[2]*256+coordinate[3]<225)) /* 5 */
                  {
                  if (choice==0)
                    {
                    channel1=5;
                    choice=1;
                    printf (ANSI_COLOR_BLUE"Digit 1 = 5\n" );
                     }
                  else if (choice==1)
                    {
                    channel2=5;
                    choice=0;
                    printf (ANSI_COLOR_MAGENTA"Digit 2 = 5\n" );
                    }
                  }
   else if( (coordinate[0]*256+coordinate[1]>490) && (coordinate[0]*256+coordinate[1]<605) &&
(coordinate[2]*256+coordinate[3]>175) && (coordinate[2]*256+coordinate[3]<225)) /* 6 */
                  {
                  if (choice==0)
                    {
                    channel1=6;
                    choice=1;
                    printf (ANSI_COLOR_BLUE"Digit 1 = 6\n" );
                    }
                  else if (choice==1)
                    {
                    channel2=6;
                    choice=0;
                    printf (ANSI_COLOR_MAGENTA"Digit 2 = 6\n" );
                    }
                  }
  else if( (coordinate[0]*256+coordinate[1]>235) && (coordinate[0]*256+coordinate[1]<360) &&
(coordinate[2]*256+coordinate[3]>100) && (coordinate[2]*256+coordinate[3]<170)) /* 7 */
                 {
                   if (choice==0)
                    {
                    channel1=7;
                    choice=1;
                    printf (ANSI_COLOR_BLUE"Digit 1 = 7\n" );
                    }
                 else if (choice==1)
                    {
                    channel2=7;
                    choice=0;
                    printf (ANSI_COLOR_MAGENTA"Digit 2 = 7\n" );
                    }
                  }
  else if( (coordinate[0]*256+coordinate[1]>370) && (coordinate[0]*256+coordinate[1]<470) &&
(coordinate[2]*256+coordinate[3]>100) && (coordinate[2]*256+coordinate[3]<170)) /* 8 */
                 {
```

```
                if (choice==0)
                    {
                    channel1=8;
                    choice=1;
                    printf (ANSI_COLOR_BLUE"Digit 1 = 8\n" );
                    }
                else if (choice==1)
                    {
                    channel2=8;
                    choice=0;
                    printf (ANSI_COLOR_MAGENTA"Digit 2 = 8\n" );
                    }
                }
  else if( (coordinate[0]*256+coordinate[1]>490) && (coordinate[0]*256+coordinate[1]<605) &&
(coordinate[2]*256+coordinate[3]>100) && (coordinate[2]*256+coordinate[3]<170)) /* 9 */
                {
                if (choice==0)
                    {
                    channel1=9;
                    choice=1;
                    printf (ANSI_COLOR_BLUE"Digit 1 = 9\n" );
                    }
                else if (choice==1)
                    {
                    channel2=9;
                    choice=0;
                     (ANSI_COLOR_MAGENTA"Digit 2 = 9\n" );
                    }
                }
else  if( (coordinate[0]*256+coordinate[1]>370) && (coordinate[0]*256+coordinate[1]<620) &&
(coordinate[2]*256+coordinate[3]>300) && (coordinate[2]*256+coordinate[3]<340)) /*enter */
                    {
                      if (channel2*10+channel1>=0 && channel2*10+channel1<=3){
                            system("clear");
                            printf ("\n" );
                            printf ("\n" );
                            printf (ANSI_COLOR_CYAN"         THE CHOSEN CHANNEL IS  %d%d      \n ",channel2,channel1 );
                            printf ("\n" );
                            printf (ANSI_COLOR_BLUE"               ----------------------------------------------------------------\n");
                            printf (ANSI_COLOR_BLUE"                                DO YOU WANT TO CONTINUE?         \n ");
                            printf (ANSI_COLOR_BLUE"               ----------------------------------------------------------------\n");
                            printf ("\n" );
                            printf ("\n" );
                            printf (ANSI_COLOR_GREEN"                              ---------"ANSI_COLOR_RED"               ---------\n" );
                            printf (ANSI_COLOR_GREEN"                            YES   "ANSI_COLOR_RED"                 NO   \n" );
                            printf (ANSI_COLOR_GREEN"    ---------"ANSI_COLOR_RED"    ---------\n" ANSI_COLOR_RESET);
                            decision=1;
                                        }
                       else
                            {
                            system("clear");
                            system(" cat /opt/calculator.raw > /dev/fb0");
                            printf (ANSI_COLOR_CYAN"CHANNEL NUMBER MUST BE BETWEEN 0-3!!!  \n ");
                             }
                    }
          }
/*gia time delay*/
  else if  ((z==1) && (calculate==0) && (decision==0) && (channel_sel==0) && (number_of_points==0)&& (time_window==0)&&
(time_delay_sel==1))
        {
        z=0;
        if( (coordinate[0]*256+coordinate[1]>580) && (coordinate[0]*256+coordinate[1]<800) &&
(coordinate[2]*256+coordinate[3]>380) && (coordinate[2]*256+coordinate[3]<480))
          {
          time_delay_sel=0;
          system(" cat /opt/calibration_menu.raw > /dev/fb0");
          }
    else if( (coordinate[0]*256+coordinate[1]>235) && (coordinate[0]*256+coordinate[1]<362) &&
(coordinate[2]*256+coordinate[3]>300) && (coordinate[2]*256+coordinate[3]<340)) /* 0 */
                    {
                if (choice==0)
                        {
                        time_delay1=0;
                        choice=1;
                        printf (ANSI_COLOR_BLUE"Digit 1 = 0\n" );
                        }
                else if (choice==1)
                        {
```

```
                        time_delay2=0;
                        choice=2;
                        printf (ANSI_COLOR_MAGENTA"Digit 2 = 0\n" );
                        }
                else if (choice==2)
                        {
                        time_delay3=0;
                        choice=0;
                        printf (ANSI_COLOR_MAGENTA"Digit 3 = 0\n" );
                        }
                }
   else if( (coordinate[0]*256+coordinate[1]>235) && (coordinate[0]*256+coordinate[1]<362) &&
(coordinate[2]*256+coordinate[3]>238) && (coordinate[2]*256+coordinate[3]<280)) /* 1 */
                {
                if (choice==0)
                        {
                        time_delay1=1;
                        choice=1;
                        printf (ANSI_COLOR_BLUE"Digit 1 = 1\n" );
                        }
                else if (choice==1)
                        {
                        time_delay2=1;
                        choice=2;
                        printf (ANSI_COLOR_MAGENTA"Digit 2 = 1\n" );
                        }
                else if (choice==2)
                        {
                        time_delay3=1;
                        choice=0;
                        printf (ANSI_COLOR_MAGENTA"Digit 3 = 1\n" );
                        }
                }
  else if( (coordinate[0]*256+coordinate[1]>370) && (coordinate[0]*256+coordinate[1]<470) &&
(coordinate[2]*256+coordinate[3]>238) && (coordinate[2]*256+coordinate[3]<280)) /* 2 */
                {
                if (choice==0)
                          {
                        time_delay1=2;
                        choice=1;
                        printf (ANSI_COLOR_BLUE"Digit 1 = 2\n" );
                        }
                else if (choice==1)
                        {
                        time_delay2=2;
                        choice=2;
                        printf (ANSI_COLOR_MAGENTA"Digit 2 = 2\n" );
                          }
                else if (choice==2)
                        {
                        time_delay3=3;
                        choice=0;
                        printf (ANSI_COLOR_MAGENTA"Digit 3 = 2\n" );
                        }
                }
  else if( (coordinate[0]*256+coordinate[1]>490) && (coordinate[0]*256+coordinate[1]<605) &&
(coordinate[2]*256+coordinate[3]>238) && (coordinate[2]*256+coordinate[3]<280)) /* 3 */
                {
                if (choice==0)
                        {
                        time_delay1=3;
                        choice=1;
                        printf (ANSI_COLOR_BLUE"Digit 1 = 3\n" );
                        }
                else if (choice==1)
                        {
                        time_delay2=3;
                        choice=2;
                        printf (ANSI_COLOR_MAGENTA"Digit 2 = 3\n" );
                        }
                else if (choice==2)
                          {
                        time_delay3=4;
                        choice=0;
                        printf (ANSI_COLOR_MAGENTA"Digit 3 = 3\n" );
                        }
                }
```

```
    else if( (coordinate[0]*256+coordinate[1]>235) && (coordinate[0]*256+coordinate[1]<360) &&
(coordinate[2]*256+coordinate[3]>175) && (coordinate[2]*256+coordinate[3]<225)) /* 4 */
                {
                if (choice==0)
                        {
                        time_delay1=4;
                        choice=1;
                        printf (ANSI_COLOR_BLUE"Digit 1 = 4\n" );
                        }
                else if (choice==1)
                        {
                        time_delay2=4;
                        choice=2;
                        printf (ANSI_COLOR_MAGENTA"Digit 2 = 4\n" );
                        }
                    else if (choice==2)
                        {
                        time_delay3=4;
                        choice=0;
                        printf (ANSI_COLOR_MAGENTA"Digit 3 = 4\n" );
                        }
                }
else if( (coordinate[0]*256+coordinate[1]>370) && (coordinate[0]*256+coordinate[1]<470) &&
(coordinate[2]*256+coordinate[3]>175) && (coordinate[2]*256+coordinate[3]<225)) /* 5 */
                {
                if (choice==0)
                        {
                        time_delay1=5;
                        choice=1;
                        printf (ANSI_COLOR_BLUE"Digit 1 = 5\n" );
                        }
                else if (choice==1)
                        {
                        time_delay2=5;
                        choice=2;
                        printf (ANSI_COLOR_MAGENTA"Digit 2 = 5\n" );
                        }
                else if (choice==2)
                        {
                        time_delay3=5;
                        choice=0;
                        printf (ANSI_COLOR_MAGENTA"Digit 3 = 5\n" );
                            }
                }
else if( (coordinate[0]*256+coordinate[1]>490) && (coordinate[0]*256+coordinate[1]<605) &&
(coordinate[2]*256+coordinate[3]>175) && (coordinate[2]*256+coordinate[3]<225)) /* 6 */
                {
                if (choice==0)
                        {
                        time_delay1=6;
                        choice=1;
                        printf (ANSI_COLOR_BLUE"Digit 1 = 6\n" );
                        }
                else if (choice==1)
                        {
                        time_delay2=6;
                        choice=2;
                        printf (ANSI_COLOR_MAGENTA"Digit 2 = 6\n" );
                        }
                else if (choice==2)
                        {
                        time_delay3=6;
                        choice=0;
                        printf (ANSI_COLOR_MAGENTA"Digit 3 = 6\n" );
                        }
                }
   else if( (coordinate[0]*256+coordinate[1]>235) && (coordinate[0]*256+coordinate[1]<360) &&
(coordinate[2]*256+coordinate[3]>100) && (coordinate[2]*256+coordinate[3]<170)) /* 7 */
                {
                if (choice==0)
                        {
                        time_delay1=7;
                        choice=1;
                        printf (ANSI_COLOR_BLUE"Digit 1 = 7\n" );
                        }
                else if (choice==1)
                        {
                        time_delay2=7;
```

```
                        choice=2;
                        printf (ANSI_COLOR_MAGENTA"Digit 2 = 7\n" );
                        }
                else if (choice==2)
                        {
                        time_delay3=7;
                        choice=0;
                        printf (ANSI_COLOR_MAGENTA"Digit 3 = 7\n" );
                        }
                }
else if( (coordinate[0]*256+coordinate[1]>370) && (coordinate[0]*256+coordinate[1]<470) &&
(coordinate[2]*256+coordinate[3]>100) && (coordinate[2]*256+coordinate[3]<170)) /* 8 */
                {
                if (choice==0)
                        {
                        time_delay1=8;
                        choice=1;
                        printf (ANSI_COLOR_BLUE"Digit 1 = 8\n" );
                         }
                  else if (choice==1)
                        {
                        time_delay2=8;
                        choice=2;
                        printf (ANSI_COLOR_MAGENTA"Digit 2 = 8\n" );
                        }
        else if (choice==2)
                           {
                        time_delay3=8;
                        choice=0;
                        printf (ANSI_COLOR_MAGENTA"Digit 3 = 8\n" );
                           }
                           }
else if( (coordinate[0]*256+coordinate[1]>490) && (coordinate[0]*256+coordinate[1]<605) &&
(coordinate[2]*256+coordinate[3]>100) && (coordinate[2]*256+coordinate[3]<170)) /* 9 */
        {
        if (choice==0)
                {
                time_delay1=9;
                choice=1;
                printf (ANSI_COLOR_BLUE"Digit 1 = 9\n" );
                }
        else if (choice==1)
                {
                time_delay2=9;
                choice=2;
                printf (ANSI_COLOR_MAGENTA"Digit 2 = 9\n" );
                }
        else if (choice==2)
                {
                time_delay3=9;
                choice=0;
                printf (ANSI_COLOR_MAGENTA"Digit 3 = 9\n" );
                }
        }
else  if( (coordinate[0]*256+coordinate[1]>370) && (coordinate[0]*256+coordinate[1]<620) &&
(coordinate[2]*256+coordinate[3]>300) && (coordinate[2]*256+coordinate[3]<340)) /*enter */
        {
        if (channel2*10+channel1>=0 && channel2*10+channel1<=3)
{
                system("clear");
                printf ("\n" );
                printf ("\n" );
                printf (ANSI_COLOR_CYAN"                THE CHOSEN TIME DELAY IS  %d%d%d             \n
",time_delay3,time_delay2,time_delay1 );
                printf ("\n" );
                printf (ANSI_COLOR_BLUE"              ----------------------------------------------------------------\n");
                printf (ANSI_COLOR_BLUE"                        DO YOU WANT TO CONTINUE?        \n ");
                printf (ANSI_COLOR_BLUE"              ----------------------------------------------------------------\n");
                printf ("\n" );
                printf ("\n" );
                printf (ANSI_COLOR_GREEN"                       ---------"ANSI_COLOR_RED"               ---------\n" );
                printf (ANSI_COLOR_GREEN"                     YES   "ANSI_COLOR_RED"                NO   \n" );
                printf (ANSI_COLOR_GREEN"  -------"ANSI_COLOR_RED"        ---------\n" ANSI_COLOR_RESET);
                decision=1;
                }
        }
}
/*********gia time delay ******/
```

```
/*********channel selection menu ******/
//gia decision//
 else if  ((z==1) && (decision==1) )
          {
          z=0;
          if( (coordinate[0]*256+coordinate[1]>240) && (coordinate[0]*256+coordinate[1]<340) &&
(coordinate[2]*256+coordinate[3]>110) && (coordinate[2]*256+coordinate[3]<200))
                    {
                    channel_sel=0;
                    number_of_points=0;
                    time_window=0;
                    decision=0;
                    time_delay_sel=0;
                    system("clear");
                    system(" cat /opt/calibration_menu.raw > /dev/fb0");
                         choice=0;
                    }
          else if( (coordinate[0]*256+coordinate[1]>450) && (coordinate[0]*256+coordinate[1]<550) &&
(coordinate[2]*256+coordinate[3]>110) && (coordinate[2]*256+coordinate[3]<200))
                    {
                    decision=0;
                    choice=0;
                    system("clear");
                    system(" cat /opt/calculator.raw > /dev/fb0");
                    if ( channel_sel==1)
                           {
                        printf (ANSI_COLOR_GREEN"Digits pressed for channel selection ( Digit 2 = MSB and Digit 1 =  LSB):\n" );
                        printf ("\n" );
                        printf (ANSI_COLOR_GREEN"SUPPORTED CHANNELS: 0-3\n" );
                        printf ("\n" );
                          }
                    else if ( time_window==1)
                       {
                            printf (ANSI_COLOR_GREEN"Digits pressed for time window selection ( Digit 2 = MSB and Digit 1 =
LSB):\n" );
                        printf ("\n" );
                          printf (ANSI_COLOR_GREEN"SUPPORTED TIME WINDOW NUMBERS: 18-99\n" );
                        printf ("\n" );
                        }
                    else if ( number_of_points==1)
                       {
                        printf (ANSI_COLOR_GREEN"Digits pressed for number of points ( Digit 2 = MSB and Digit 1 =  LSB):\n" );
                        printf ("\n" );
                        printf (ANSI_COLOR_GREEN"SUPPORTED POINTS: 1-99999\n" );
                        printf ("\n" );
                        }
                    else
                        {
                        printf (ANSI_COLOR_GREEN"Digits pressed for time delay ( Digit 3 = MSB and Digit 1 =  LSB):\n" );
                        printf ("\n" );
                        printf (ANSI_COLOR_GREEN"SUPPORTED POINTS: 0-999\n" );
                        printf ("\n" );
                        }
                    }
          }
          //decision//
          //number of points menu//
else if  ((z==1) && (calculate==0)&& (decision==0) && (channel_sel==0) && (number_of_points==1) && (time_window==0) &&
(time_delay_sel==0))
       {
       z=0;
       if( (coordinate[0]*256+coordinate[1]>580) && (coordinate[0]*256+coordinate[1]<800) &&
(coordinate[2]*256+coordinate[3]>380) && (coordinate[2]*256+coordinate[3]<480))
                    {
                    number_of_points=0;
                    system(" cat /opt/calibration_menu.raw > /dev/fb0");
                    }
       else if( (coordinate[0]*256+coordinate[1]>235) && (coordinate[0]*256+coordinate[1]<362) &&
(coordinate[2]*256+coordinate[3]>300) && (coordinate[2]*256+coordinate[3]<340)) /* 0 */
       {
       if (choice==0)
                    {
                    point1=0;
                    choice=1;
                    printf (ANSI_COLOR_BLUE"Digit 1 = 0\n" );
                    }
       else if (choice==1)
                    {
```

```
                point2=0;
                choice=2;
                printf (ANSI_COLOR_MAGENTA"Digit 2 = 0\n" );
                }
        else if (choice==2)
                {
                point3=0;
                choice=3;
                printf (ANSI_COLOR_MAGENTA"Digit 3 = 0\n" );
                }
            else if (choice==3)
                  {
                point4=0;
                choice=4;
                printf (ANSI_COLOR_MAGENTA"Digit 4 = 0\n" );
                }
            else if (choice==4)
                {
                point5=0;
                choice=0;
                printf (ANSI_COLOR_MAGENTA"Digit 5 = 0\n" );
                }
            }
else if( (coordinate[0]*256+coordinate[1]>235) && (coordinate[0]*256+coordinate[1]<362) &&
(coordinate[2]*256+coordinate[3]>238) && (coordinate[2]*256+coordinate[3]<280)) /* 1 */
            {
            if (choice==0)
                {
                point1=1;
                choice=1;
                printf (ANSI_COLOR_BLUE"Digit 1 = 1\n" );
                }
            else if (choice==1)
                {
                point2=1;
                choice=2;
                printf (ANSI_COLOR_MAGENTA"Digit 2 = 1\n" );
                }
            else if (choice==2)
                {
                point3=1;
                choice=3;
                printf (ANSI_COLOR_MAGENTA"Digit 3 = 1\n" );
                }
            else if (choice==3)
                {
                point4=1;
                choice=4;
                printf (ANSI_COLOR_MAGENTA"Digit 4 = 1\n" );
                }
            else if (choice==4)
                {
                point5=1;
                choice=0;
                printf (ANSI_COLOR_MAGENTA"Digit 5 = 1\n" );
                }
            }
else if( (coordinate[0]*256+coordinate[1]>370) && (coordinate[0]*256+coordinate[1]<470) &&
(coordinate[2]*256+coordinate[3]>238) && (coordinate[2]*256+coordinate[3]<280)) /* 2 */
            {
            if (choice==0)
                {
                point1=2;
                choice=1;
                printf (ANSI_COLOR_BLUE"Digit 1 = 2\n" );
                }
            else if (choice==1)
                {
                point2=2;
                choice=2;
                printf (ANSI_COLOR_MAGENTA"Digit 2 = 2\n" );
                }
            else if (choice==2)
                {
                point3=2;
                choice=3;
                printf (ANSI_COLOR_MAGENTA"Digit 3 = 2\n" );
                }
```

```
        else if (choice==3)
                {
                point4=2;
                choice=4;
                printf (ANSI_COLOR_MAGENTA"Digit 4 = 2\n" );
                }
        else if (choice==4)
                {
                point5=2;
                choice=0;
                printf (ANSI_COLOR_MAGENTA"Digit 5 = 2\n" );
                }
        }
 else if( (coordinate[0]*256+coordinate[1]>490) && (coordinate[0]*256+coordinate[1]<605) &&
(coordinate[2]*256+coordinate[3]>238) && (coordinate[2]*256+coordinate[3]<280)) /* 3 */
        {
        if (choice==0)
                {
                point1=3;
                choice=1;
                printf (ANSI_COLOR_BLUE"Digit 1 = 3\n" );
                }
        else if (choice==1)
                {
                point2=3;
                choice=2;
                printf (ANSI_COLOR_MAGENTA"Digit 2 = 3\n" );
                }
        else if (choice==2)
                {
                point3=3;
                choice=3;
                printf (ANSI_COLOR_MAGENTA"Digit 3 = 3\n" );
                }
        else if (choice==3)
                {
                point4=3;
                choice=4;
                   printf (ANSI_COLOR_MAGENTA"Digit 4 = 3\n" );
                }
        else if (choice==4)
                {
                point5=3;
                choice=0;
                printf (ANSI_COLOR_MAGENTA"Digit 5 = 3\n" );
                }
        }
else if( (coordinate[0]*256+coordinate[1]>235) && (coordinate[0]*256+coordinate[1]<360) &&
(coordinate[2]*256+coordinate[3]>175) && (coordinate[2]*256+coordinate[3]<225)) /* 4 */
        {
        if (choice==0)
                {
                point1=4;
                choice=1;
                printf (ANSI_COLOR_BLUE"Digit 1 = 4\n" );
                }
        else if (choice==1)
                {
                point2=4;
                choice=2;
                printf (ANSI_COLOR_MAGENTA"Digit 2 = 4\n" );
                }
        else if (choice==2)
                {
                point3=4;
                choice=3;
                 (ANSI_COLOR_MAGENTA"Digit 3 = 4\n" );
                }
        else if (choice==3)
                {
                point4=4;
                choice=4;
                printf (ANSI_COLOR_MAGENTA"Digit 4 = 4\n" );
                }
        else if (choice==4)
                {
                point5=4;
                choice=0;
```

```
                printf (ANSI_COLOR_MAGENTA"Digit 5 = 4\n" );
                }
        }
else if( (coordinate[0]*256+coordinate[1]>370) && (coordinate[0]*256+coordinate[1]<470) &&
(coordinate[2]*256+coordinate[3]>175) && (coordinate[2]*256+coordinate[3]<225)) /* 5 */
        {
        if (choice==0)
                {
                point1=5;
                choice=1;
                printf (ANSI_COLOR_BLUE"Digit 1 = 5\n" );
                }
        else if (choice==1)
                {
                point2=5;
                choice=2;
                printf (ANSI_COLOR_MAGENTA"Digit 2 = 5\n" );
                }
        else if (choice==2)
                {
                point3=5;
                choice=3;
                printf (ANSI_COLOR_MAGENTA"Digit 3 = 5\n" );
                }
        else if (choice==3)
                {
                point4=5;
                choice=4;
                printf (ANSI_COLOR_MAGENTA"Digit 4 = 5\n" );
                }
        else if (choice==4)
                {
                point5=5;
                choice=0;
                printf (ANSI_COLOR_MAGENTA"Digit 5 = 5\n" );
                }
        }
else if( (coordinate[0]*256+coordinate[1]>490) && (coordinate[0]*256+coordinate[1]<605) &&
(coordinate[2]*256+coordinate[3]>175) && (coordinate[2]*256+coordinate[3]<225)) /* 6 */
        {
        if (choice==0)
                {
                point1=6;
                choice=1;
                printf (ANSI_COLOR_BLUE"Digit 1 = 6\n" );
                }
        else if (choice==1)
                {
                point2=6;
                choice=2;
                printf (ANSI_COLOR_MAGENTA"Digit 2 = 6\n" );
                }
        else if (choice==2)
                {
                point3=6;
                choice=3;
                printf (ANSI_COLOR_MAGENTA"Digit 3 = 6\n" );
                }
        else if (choice==3)
                {
                point4=6;
                choice=4;
                printf (ANSI_COLOR_MAGENTA"Digit 4 = 6\n" );
                }
        else if (choice==4)
                {
                point5=6;
                choice=0;
                printf (ANSI_COLOR_MAGENTA"Digit 5 = 6\n" );
                }
        }
else if( (coordinate[0]*256+coordinate[1]>235) && (coordinate[0]*256+coordinate[1]<360) &&
(coordinate[2]*256+coordinate[3]>100) && (coordinate[2]*256+coordinate[3]<170)) /* 7 */
        {
        if (choice==0)
                {
                point1=7;
                choice=1;
```

```
                printf (ANSI_COLOR_BLUE"Digit 1 = 7\n" );
                }
        else if (choice==1)
                {
                point2=7;
                choice=2;
                printf (ANSI_COLOR_MAGENTA"Digit 2 = 7\n" );
                }
        else if (choice==2)
                {
                point3=7;
                choice=3;
                printf (ANSI_COLOR_MAGENTA"Digit 3 = 7\n" );
                }
        else if (choice==3)
                {
                point4=7;
                choice=4;
                printf (ANSI_COLOR_MAGENTA"Digit 4 = 7\n" );
                }
        else if (choice==4)
                {
                point5=7;
                choice=0;
                printf (ANSI_COLOR_MAGENTA"Digit 5 = 7\n" );
                }
        }
if( (coordinate[0]*256+coordinate[1]>370) && (coordinate[0]*256+coordinate[1]<470) &&  (coordinate[2]*256+coordinate[3]>100) &&
(coordinate[2]*256+coordinate[3]<170)) /* 8 */
        {
        if (choice==0)
                {
                point1=8;
                choice=1;
                printf (ANSI_COLOR_BLUE"Digit 1 = 8\n" );
                }
        else if (choice==1)
                {
                point2=8;
                choice=2;
                printf (ANSI_COLOR_MAGENTA"Digit 2 = 8\n" );
                }
        if (choice==2)
                {
                point3=8;
                choice=3;
                printf (ANSI_COLOR_MAGENTA"Digit 3 = 8\n" );
                }
        else if (choice==3)
                {
                point4=8;
                choice=4;
                printf (ANSI_COLOR_MAGENTA"Digit 4 = 8\n" );
                }
        else if (choice==4)
                {
                point5=8;
                choice=0;
                printf (ANSI_COLOR_MAGENTA"Digit 5 = 8\n" );
                }
        }
else if( (coordinate[0]*256+coordinate[1]>490) && (coordinate[0]*256+coordinate[1]<605) &&
(coordinate[2]*256+coordinate[3]>100) && (coordinate[2]*256+coordinate[3]<170)) /* 9 */
                {
        if (choice==0)
                {
                point1=9;
                choice=1;
                printf (ANSI_COLOR_BLUE"Digit 1 = 9\n" );
                }
        else if (choice==1)
                {
                point2=9;
                choice=2;
                printf (ANSI_COLOR_MAGENTA"Digit 2 = 9\n" );
                }
        else if (choice==2)
                {
```

```
                point3=9;
                choice=3;
                printf (ANSI_COLOR_MAGENTA"Digit 3 = 9\n" );
                }
        else if (choice==3)
                {
                point4=9;
                choice=4;
                printf (ANSI_COLOR_MAGENTA"Digit 4 = 9\n" );
                }
        else if (choice==4)
                {
                point5=9;
                choice=0;
                printf (ANSI_COLOR_MAGENTA"Digit 5 = 9\n" );
                }
        }
else  if( (coordinate[0]*256+coordinate[1]>370) && (coordinate[0]*256+coordinate[1]<620) &&
(coordinate[2]*256+coordinate[3]>300) && (coordinate[2]*256+coordinate[3]<340)) /*enter */
        {
        if ((point5*10000+point4*1000+point3*100+point2*10+point1)!=0)
            {
            system("clear");
           printf ("\n" );
           printf ("\n" );                                                                                                  printf
(ANSI_COLOR_CYAN" THE CHOSEN POINTS ARE  %d%d%d%d%d  \n ",point5,point4,point3,point2,point1 );
           printf ("\n" );
           printf (ANSI_COLOR_BLUE"              ------------------------------------------------------------\n");
           printf (ANSI_COLOR_BLUE"                        DO YOU WANT TO CONTINUE?        \n ");
           printf (ANSI_COLOR_BLUE"              ------------------------------------------------------------\n");
           printf ("\n" );
           printf ("\n" );
           printf (ANSI_COLOR_GREEN"                      ---------"ANSI_COLOR_RED"              ---------\n" );
           printf (ANSI_COLOR_GREEN"                        YES   "ANSI_COLOR_RED"                NO   \n" );
           printf (ANSI_COLOR_GREEN"      ---------"ANSI_COLOR_RED"              ---------\n"             ANSI_COLOR_RESET);
           decision=1;
           }
        else
            {
           system("clear");
           system(" cat /opt/calculator.raw > /dev/fb0");
           printf (ANSI_COLOR_CYAN"POINTS NUMBER MUST BE GREATER THAN 0!!!             \n ");
           }
        }
}
                        //number of points menu//
                        //time window menu//
else if  ((z==1) && (calculate==0) && (decision==0) && (channel_sel==0)&& (number_of_points==0)&& (time_window==1) &&
(time_delay_sel==0))
        {
        z=0;
        if( (coordinate[0]*256+coordinate[1]>580) && (coordinate[0]*256+coordinate[1]<800) &&
(coordinate[2]*256+coordinate[3]>380) && (coordinate[2]*256+coordinate[3]<480))
                {
                time_window=0;
                system(" cat /opt/calibration_menu.raw > /dev/fb0");                                                                    }
        else if( (coordinate[0]*256+coordinate[1]>235) && (coordinate[0]*256+coordinate[1]<362) &&
(coordinate[2]*256+coordinate[3]>300) && (coordinate[2]*256+coordinate[3]<340)) /* 0 */
        {               if (choice==0)
                        {
                           time1=0;
                              choice=1;
                           printf (ANSI_COLOR_BLUE"Digit 1 = 0\n" );
                        }
                else if (choice==1)
                        {
                           time2=0;
                              choice=0;
                           printf (ANSI_COLOR_MAGENTA"Digit 2 = 0\n" );
                        }

                  }
        else if( (coordinate[0]*256+coordinate[1]>235) && (coordinate[0]*256+coordinate[1]<362) &&
(coordinate[2]*256+coordinate[3]>238) && (coordinate[2]*256+coordinate[3]<280)) /* 1 */
                {
                if (choice==0)
                        {
                        time1=1;
```

```
                        choice=1;
                        printf (ANSI_COLOR_BLUE"Digit 1 = 1\n" );
                        }
                else if (choice==1)
                        {
                        time2=1;
                        choice=0;
                        printf (ANSI_COLOR_MAGENTA"Digit 2 = 1\n" );
                        }
                }
        else if( (coordinate[0]*256+coordinate[1]>370) && (coordinate[0]*256+coordinate[1]<470) &&
(coordinate[2]*256+coordinate[3]>238) && (coordinate[2]*256+coordinate[3]<280)) /* 2 */
                {
                if (choice==0)
                        {
                        time1=2;
                        choice=1;
                        printf (ANSI_COLOR_BLUE"Digit 1 = 2\n" );
                        }
                else if (choice==1)
                          {
                        time2=2;
                        choice=0;
                        printf (ANSI_COLOR_MAGENTA"Digit 2 = 2\n" );
                        }
                }
        else if( (coordinate[0]*256+coordinate[1]>490) && (coordinate[0]*256+coordinate[1]<605) &&
(coordinate[2]*256+coordinate[3]>238) && (coordinate[2]*256+coordinate[3]<280)) /* 3 */
                {
                if (choice==0)
                        {
                        time1=3;
                        choice=1;
                        printf (ANSI_COLOR_BLUE"Digit 1 = 3\n" );
                        }
                else if (choice==1)
                        {
                        time2=3;
                        choice=0;
                        printf (ANSI_COLOR_MAGENTA"Digit 2 = 3\n" );
                        }
                }
        else if( (coordinate[0]*256+coordinate[1]>235) && (coordinate[0]*256+coordinate[1]<360) &&
(coordinate[2]*256+coordinate[3]>175) && (coordinate[2]*256+coordinate[3]<225)) /* 4 */
                {
                if (choice==0)
                        {
                        time1=4;
                        choice=1;
                        printf (ANSI_COLOR_BLUE"Digit 1 = 4\n" );
                        }
                else if (choice==1)
                        {
                        time2=4;
                        choice=0;
                        printf (ANSI_COLOR_MAGENTA"Digit 2 = 4\n" );
                        }
                }
        else if( (coordinate[0]*256+coordinate[1]>370) && (coordinate[0]*256+coordinate[1]<470) &&
(coordinate[2]*256+coordinate[3]>175) && (coordinate[2]*256+coordinate[3]<225)) /* 5 */
                {
                if (choice==0)
                        {
                        time1=5;
                        choice=1;
                        printf (ANSI_COLOR_BLUE"Digit 1 = 5\n" );
                        }
                else if (choice==1)
                        {
                        time2=5;
                        choice=0;
                        printf (ANSI_COLOR_MAGENTA"Digit 2 = 5\n" );
                        }
                }
        else if( (coordinate[0]*256+coordinate[1]>490) && (coordinate[0]*256+coordinate[1]<605) &&
(coordinate[2]*256+coordinate[3]>175) && (coordinate[2]*256+coordinate[3]<225)) /* 6 */
                {
                if (choice==0)
```

```
                        {
                        time1=6;
                        choice=1;
                        printf (ANSI_COLOR_BLUE"Digit 1 = 6\n" );
                          }
                else if (choice==1)
                        {
                        time2=6;
                        choice=0;
                        printf (ANSI_COLOR_MAGENTA"Digit 2 = 6\n" );
                        }
                }
        else if( (coordinate[0]*256+coordinate[1]>235) && (coordinate[0]*256+coordinate[1]<360) &&
(coordinate[2]*256+coordinate[3]>100) && (coordinate[2]*256+coordinate[3]<170)) /* 7 */
                {
                if (choice==0)
                        {
                        time1=7;
                        choice=1;
                        printf (ANSI_COLOR_BLUE"Digit 1 = 7\n" );
                        }
                else if (choice==1)
                        {
                        time2=7;
                        choice=0;
                        printf (ANSI_COLOR_MAGENTA"Digit 2 = 7\n" );
                        }
                }
        else if( (coordinate[0]*256+coordinate[1]>370) && (coordinate[0]*256+coordinate[1]<470) &&
(coordinate[2]*256+coordinate[3]>100) && (coordinate[2]*256+coordinate[3]<170)) /* 8 */
                {
                if (choice==0)
                        {
                        time1=8;
                        choice=1;
                        printf (ANSI_COLOR_BLUE"Digit 1 = 8\n" );
                        }
                else if (choice==1)
                        {
                        time2=8;
                        choice=0;
                        printf ("Digit 1 = 8\n" );
                          }
                }
        else if( (coordinate[0]*256+coordinate[1]>490) && (coordinate[0]*256+coordinate[1]<605) &&
(coordinate[2]*256+coordinate[3]>100) && (coordinate[2]*256+coordinate[3]<170)) /* 9 */
                {
                if (choice==0)
                        {
                        time1=9;
                        choice=1;
                        printf (ANSI_COLOR_BLUE"Digit 1 = 9\n" );
                        }
                else if (choice==1)
                        {
                        time2=9;
                        choice=0;
                        printf (ANSI_COLOR_MAGENTA"Digit 2 = 9\n" );
                        }
                }
        else  if( (coordinate[0]*256+coordinate[1]>370) && (coordinate[0]*256+coordinate[1]<620) &&
(coordinate[2]*256+coordinate[3]>300) && (coordinate[2]*256+coordinate[3]<340)) /*enter */
                {
                if ((time2*10+time1)>=18 && (time2*10+time1)<=31)
                        {
                        system("clear");
                        printf ("\n" );
                        printf ("\n" );
                        printf (ANSI_COLOR_CYAN"                THE CHOSEN TIME WINDOW IS  %d%d              \n
",time2,time1 );
                        printf ("\n" );
                        printf (ANSI_COLOR_BLUE"              ------------------------------------------------------------\n");
                        printf (ANSI_COLOR_BLUE"                        DO YOU WANT TO CONTINUE?         \n ");
                        printf (ANSI_COLOR_BLUE"              ------------------------------------------------------------\n");
                        printf ("\n" );
                        printf ("\n" );
                        printf (ANSI_COLOR_GREEN"                    ---------"ANSI_COLOR_RED"              ---------\n" );
                        printf (ANSI_COLOR_GREEN"                   YES   "ANSI_COLOR_RED"                NO   \n" );
```

```
                        printf (ANSI_COLOR_GREEN"    ---------"ANSI_COLOR_RED"      ---------\n" ANSI_COLOR_RESET);
                        time_total=10*time2 + time1;
                        sampling_time=1;
                        for( i = 0; i < time_total; i++ ){
                                sampling_time=2*sampling_time;
                                }
                        sampling_time=50000000/sampling_time;
                        printf ("\n" );
                        printf ("\n" );
                        printf ("\n" );
                        printf ("\n" );
                        printf (ANSI_COLOR_MAGENTA"     SAMPLING TIME FREQUENCY:  %e Hz \n",sampling_time);
                        printf ("\n" );
                        printf ("\n" );
                        printf (ANSI_COLOR_MAGENTA"SAMPLING TIME:  %e s\n "ANSI_COLOR_RESET,(1/sampling_time));
                        decision=1;
                        }
                else
                                {
                           system("clear");
                        system(" cat /opt/calculator.raw > /dev/fb0");
                        printf (ANSI_COLOR_CYAN"TIME WINDOW MUST BE BETWEEN 18-31!!!   \n " );
                           }
                 }
              }
           }
      }
/***************** τέλος calibration menu ********************/
 end:
printf ("\n" ANSI_COLOR_RESET);
system("dd if=/dev/zero of=/dev/fb0 >& /dev/null");
system("echo 1 > /sys/class/graphics/fbcon/cursor_blink");
system("echo 225 > /sys/class/gpio/unexport");
system("echo 226 > /sys/class/gpio/unexport");
system("echo 227 > /sys/class/gpio/unexport");
system("clear");
return 0;
}
```

ΑΝΑΦΟΡΕΣ

[1] https://www.ecnmag.com/article/2010/05/create-multifunctional-flexible-low-cost-smart-sensing-designs, Sudhir Bommena,*Create Multifunctional, Flexible, Low Cost, Smart-Sensing Designs*

[2] Gary W. Hunter, Joseph R. Stetter, Peter J. Hesketh, Chung-Chiun Liu, *Smart Sensor Systems,* The Electrochemical Society Interface, 2010

[3] Gabriel J. García , Carlos A. Jara, Jorge Pomares, Aiman Alabdo, Lucas M. Poggi and Fernando Torres, *A Survey on FPGA-Based Sensor Systems: Towards Intelligent and Reconfigurable Low-Power Sensors for Computer Vision, Control and Signal Processing,* Special Issue State-of-the-Art Sensors Technology in Spain 2013

[4] Antonio de la Piedra, An Braeken and Abdellah Touhafi, *Sensor Systems Based on FPGAs and Their Applications: A Survey,* in Sensors 12(9):12235-64 · December 2012

[5] Steven W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*

[6] http://archive.cnx.org/contents/28f325b1-a7b5-4024-be16-95f0fc06c47e@2/introduction-to-tms320f28335

[7] R.C. Cofer and Ben Harding, *Basics of core-based FPGA design*, http://www.embedded.com/

[8] http://svenand.blogdrive.com/comments?id=99 , *FPGA design from scratch*. Part 50, Thursday, October 09, 2008

[9] *Xilinx 7 Series FPGAs Overview* DS180 (v2.0) September 27, 2016

[10] *Xilinx Virtex-6 Family Overview*, DS150 (v2.5) August 20, 2015

[11] David Sheldon, Frank Vahid, Stefano Lonardi, *Soft-core Processor Customization using the Design of Experiments Paradigm*, Design, Automation & Test in Europe Conference & Exhibition, 2007

[12] Wei, J.; Wang, L.; Wu, F.; Chen, Y.; Ju, L. *Design and Implementation of Wireless Sensor Node Based on Open Core.* In Proceedings of the IEEE Youth Conference on Information, Computing and Telecommunication, Beijing, China, 20–21 September 2009; pp. 102–105.

[13] Gayathri Chalivendra Renuka Srinivasan N.S. Murthy, *FPGA Based Re-Configurable Wireless Sensor Network Protocol*. In Proceedings of the International Conference on Electronic Design, Penang, Malaysia, 1–3 December 2008; pp. 1–4.

[14] Hinkelmann, H.; Reinhardt, A.; Varyani, S.; Glesner, M. *A Reconfigurable Prototyping Platform for Smart Sensor Networks*. In Proceedings of the 2008 4th Southern Conference on Programmable Logic, Bariloche, Argentina, 26–28 March 2008; pp. 125–130.

[15] Hinkelmann, H.; Zipf, P.; Glesner, M. *A Domain-Specific Dynamically Reconfigurable Hardware Platform for Wireless Sensor Networks*. In Proceedings of the International Conference on Field-Programmable Technology, Kitakyushu, Japan, 12–14 December 2007; pp. 313–316.

[16] Chao, H.Z.; Pan, L.Y.; Zeng, Z.; Meng, M.H. *A Novel FPGA-Based Wireless Vision Sensor Node.* In Proceedings of the IEEE International Conference on Automation and Logistics, Shenyang, China, 5–7 August 2009; pp. 841–846.

[17] Liu, H.; Bergmann, N. *An FPGA Soft-Core Based Implementation of a Bird Call Recognition System for Sensor Networks*. In Proceedings of the 2010 Conference on Design and Architectures for Signal and Image Processing (DASIP), Edinburgh, UK, 26–28 October 2010; pp. 1–6.

[18] Muralidhar, P.; Rao, C. *Reconfigurable Wireless Sensor Network Node Based on Nios Core*. In Proceedings of the Fourth International Conference on Wireless Communication and Sensor Networks, Indore, India, 27–29 December 2008; pp. 67–72.

[19] Zhang, G. Aviation *Manufacturing Equipment Based WSN Security Monitoring System*. In Proceedings of the 2011 9th International Conference on Reliability, Maintainability and Safety (ICRMS), Guiyang, China, 12–15 June 2011; pp. 499–503.

[20] León-Franco, J.J.; Boemo, E.; Castillo, E.; Parrilla, L. *Ring oscillators as thermal sensors in FPGAs: Experiments in low voltage.* In Proceedings of the VI Southern Programmable Logic Conference (SPL), Ipojuca, Brazil, 24–26 March 2010; pp. 133–137

[21] Ares, L.; Rodríguez-andina, J.J.; Fariña, J. *FPGA-Based Direct Resistance and Capacitance Measurements.* In Proceedings of the 35th IEEE Annual Conference in Industrial Electronics, IECON'09, Porto, Portugal, 3–5 November 2009; pp. 2837–2841.

[22] F. Reverter, M. Gasulla, R. Pallàs-Areny, *"A low-cost microcontroller interface for low-value capacitive sensors"*, Proc. 2004 IEEE Instrumentation and Measurement Technology Conference, pp. 1771-1775, May 2004.

[23] M.J. Moure, P. Rodiz, M.D. Valdes, L. Rodriguez and J. Farina, *Development of a FPGA-based SoC Instrument for the characterization of High Sensitivity QCM Oscillator Sensors*, on IEEE Industrial Electronics, IECON 2006 - 32nd Annual Conference

[24] Lu Xianghong, Chen Rujun, He Zhanxiang, Qiu Kailin, *A time interval measurement system based on FPGA used for frequency calibration*, on 2010 IEEE Instrumentation and Measurement Technology Conference (I2MTC)

[25] http://www.gaisler.com/cms/index.php?option=com_content&task=view&id=13&Itemid=53

[26] Jiri Gaisler, "*GRLIB IP Core User's Manual Version 1.0.22*", January 2010 Copyright Aeroflex"
[27] ARM Limited, "*AMBA Specification (Rev 2.0)",* 1999, ARM IHI 0011
[28] Jiri Gaisler, Sandi Habinc, "*GRLIB IP Library User's Manual Version 1.0.22*" Copyright Aeroflex Gaisler, 2010
[29] Jiri Gaisler, "*GRMON Users Manual*", AEROFLEX GAISLER AB, version 1.1.45, March 2009
[30] Hans Danneels, Kristof Coddens and Georges Gielen, *A Fully-Digital, 0.3V, 270 nW Capacitive Sensor Interface Without External References, 2011 Proceedings of the* ESSCIRC
[31] Παύλου Σπυρίδων, *Σχεδίαση και Υλοποίηση σε FPGA/CPLD ηλεκτρονικού κυκλώματος για τη μέτρηση αισθητήρων τύπου χωρητικότητας*, Πτυχιακή εργασία, Σχολή Τεχνολογικών Εφαρμογών, ΤΕΙ ΠΕΙΡΑΙΑ
[32] S. Nihtianov,Stoyan, *High Performance Capacitive Sensors Electronics Interfaces for Displacement Measurement in Industrial Applications,* Proceedings SENSOR 2009, Volume I, p. 281-286
[33] M. K. Mandal, B. C. Sarkar, *Ring Oscillator: Characteristics and applications, Indian Journal of Pure & Applied Physics Vol.48, February, pp.136-145*
[34] Alioto M & Palumbo G, *IEEE Trans on Circuits and Syst I,* 48 (2001) 210
[35] Docking S & Sachdev M*, IEEE J Solid State Circuits*, 39 (2004) 533
[36] Martin Gag, Tim Wegner, Ansgar Waschki, Dirk Timmermann, *Temperature and On-chip Crosstalk Measurement using Ring Oscillators in FPGA,* 2012 IEEE 15th International Symposium on Design and Diagnostics of Electronic Circuit and Systems (DDECS)
[37] Efstathios D. Kyriakis-Bitzaros, Nikolaos A. Stathopoulos, Spyridon Pavlos, Dimitrios Goustouridis, and Stavros Chatzandroulis*, A Reconfigurable Multichannel Capacitive Sensor Array Interface,* IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT, VOL. 60, No. 9, September 2011
[38] http://hyperphysics.phy-astr.gsu.edu/hbase/electronic/schmitt.html
[39] Application Note 200 Electronic Counter Series, *Fundamentals of the Electronic Counters*
[40] *Altera, DE2-115 User manual*
[41] *Altera, Multi-touch LCD Module Second Edition - User Manual*
[42] https://el.wikipedia.org/wiki/I%C2%B2C
[43] *THDB-HTG User Manual*, Document Version 1.0 DEC. 23, 2008 by Terasic
[44] Szabolcs Hajdu , Sándor-Tihamér Brassai, *Implementation of Embedded Linux Systems on FPGA Based Circuits for Real Time Process Control ,* MACRo 2015- 5 th International Conference on Recent Achievements in Mechatronics, Automation, Computer Science and Robotics
[45] Jiri Gaisler, *"Configuration and Development Guide"* Jan 2016, Version 1.5.0
[46] Randy Frank, *Understanding Smart sensors, 3rd Edition*
[47] M. Danek, Leos Kafka, Lukas Kohout, Jaroslav Sykora, Roman Bartosinski,*UTLEON3: Exploring Fine-Grain Multi-Threading in FPGAs, Springer*
[48] Jawwad Raza Syed, *Leon3 NoC System Generator*, Master thesis KTH Royal Institute of Technology ICT/Electronics, September 2010