ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΜΕΣΟΛΟΓΓΙΟΥ

ΤΜΗΜΑ ΤΗΛΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ & ΔΙΚΤΥΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

"Υλοποίηση Αλγορίθμων Ψηφιακής Όρασης σε κάμερα με ενσωματωμένη FPGA"

Δεδούσης Δημήτριος

Επιβλέποντες καθηγητές: Μαριάτος Ευάγγελος, Βώρος Νικόλαος

Ναύπακτος, Απρίλιος 2013

Πρόλογος

Η παρούσα πτυχιακή εργασία εκπονήθηκε το ακαδημαϊκό έτος 2012-2013. Θα ήθελα να ευχαριστήσω όλους όσους με βοήθησαν στην εκπόνηση αυτής της πτυχιακής εργασίας. Ευχαριστώ θερμά τους επιβλέποντες καθηγητές κ. Νικόλαο Βώρο και κ. Ευάγγελο Μαριάτο, οι οποίοι μου έδωσαν την ευκαιρία να ασχοληθώ με αυτή την πτυχιακή εργασία. Επίσης θα ήθελα να ευχαριστήσω ιδιαίτερα τον κ. Ευάγγελο Μαριάτο για τη συμπαράσταση, την καθοδήγηση και το ενδιαφέρον του καθ' όλη τη διάρκεια της εκπόνησης της εργασίας αυτής. Τέλος θα ήθελα να ευχαριστήσω τους γονείς μου, Ηλία και Ελένη καθώς επίσης και τη αδερφή μου Δανάη για την συνεχή στήριξη που μου έδιναν.

Περίληψη

Για την υλοποίηση του πειράματος που έγινε σε αυτή την πτυχιακή εργασία χρησιμοποιήθηκε μια camera τεχνολογίας CMOS, η οποία έχει ενσωματωμένη μια FPGA. Κάθε χρώμα από τα τρία βασικά χρώματα (κόκκινο – πράσινο – μπλε) που “βλέπει” η κάμερα στον χώρο στέλνεται στην FPGA και η FPGA με την σειρά της μετατρέπει τα χρώματα αναμιγνύοντάς τα σε ένα νέο χρώμα το οποίο παίρνει μια συγκεκριμένη θέση, ενός pixel, μέσα στην εικόνα που φτάνει στον τελικό χρήστη. Η εικόνα από την FPGA μεταφέρεται στον υπολογιστή του τελικού χρήστη μέσω μιας θύρας Ethernet και ο υπολογιστής εμφανίζει την εικόνα στον χρήστη. Με μια δεύτερη διασύνδεση η οποία είναι σειριακή (UART) ο χρήστης μπορεί να αλλάξει τις λειτουργίες τις κάμερας όπως το άνοιγμα του κλείστρου για την ένταση της φωτεινότητας ώστε να προσαρμόσει την κάμερα στις δικές του ανάγκες.

Το περιβάλλον στο οποίο γίνεται το πείραμα αυτής της πτυχιακής εργασίας έχει ως προϋπόθεση την ύπαρξη λευκού φωτός. Η ύπαρξη του λευκού φωτός στον χώρο δίνει την δυνατότητα στην κάμερα να κάνει white balance με βάση το λευκό φώς και όχι οποιοδήποτε άλλο χρώμα.

Σκοπός αυτής της πτυχιακής εργασίας είναι η υλοποίηση αλγορίθμων σε γλώσσα προγραμματισμού C, οι οποίοι κάνουν αναγνώριση χρώματος και υπολογισμό του κέντρου του αντικειμένου που έχει αναγνωριστεί ως κόκκινο. Επιπλέον υλοποιείται αλγόριθμος ο οποίος κάνει υπολογισμό του κέντρου για πολλαπλά αντικείμενα που έχει αναγνωρίσει ως κόκκινα.

Abstract

For the implementation of the experiment that took place in this thesis, a CMOS technology camera with an embedded FPGA was used. Each of the three basic colors (red - green - blue) that the camera detects are sent to the FPGA which transforms the colors by mixing them and therefore creates a new color that is placed on a particular pixel in the picture that the end user receives. The image that the FPGA sends is transported to the end user's computer via Ethernet port and finally, it appears on the end user's screen. Furthermore and in order for the user to change the functions of the camera, such as opening and closing the shutter and managing to adjust the camera depending on the needs of the end user, a second serial connection (UART) is used.

The environment in which the experiment takes place has as a requirement the existence of white light. By choosing to work only with white light, the camera is able to function based on the white light instead of any other color-based light.

In conclusion, the purpose of this thesis is the implementation of algorithms in C programming language, which can detect colors and calculate the center of an object that has been recognized as red. Thus, an algorithm for calculating the center of multiple objects is being developed.

Περιεχόμενα

Εισαγωγή

Στις μέρες μας η μηχανική όραση (Computer Vision – CV) έχει γίνει ένα σημαντικό κομμάτι στο πεδίο της έρευνας. Έχει μεθόδους για την ανάλυση, την επεξεργασία και την κατανόηση εικόνων με σκοπό να μπορέσει να φτάσει σε ένα σημείο που θα μπορεί να μιμηθεί την ανθρώπινη όραση. Οι εφαρμογές στις οποίες λαμβάνει χώρα η μηχανική όραση ποικίλουν. Κάποιες από αυτές είναι βιομηχανικές εφαρμογές, αναγνώριση γεγονότων, αναγνώριση αντικειμένων, ανάλυση εικόνας για ιατρικούς σκοπούς κ.α..

Η υλοποίηση μηχανικής όρασης μπορεί να επιτευχθεί με πολλούς διαφορετικούς τρόπους. Οι δύο βασικές κατηγορίες στις οποίες χωρίζεται η μηχανική όραση είναι το υλικό και το λογισμικό. Αυτά τα δυο είναι τα βασικά στοιχεία ώστε να μπορέσουμε να έχουμε αξιόπιστη μηχανική όραση. Οι δύο αυτές κατηγορίες ποικίλουν, από υπολογιστικά συστήματα βιομηχανιών ή υπολογιστικά συστήματα χρηστών μέχρι ενσωματωμένα συστήματα. Ανάλογα το σύστημα που θα επιλεχτεί ως εκείνο το οποίο θα υλοποιεί μηχανική όραση, επιλέγεται και το λογισμικό το οποίο θα μπορεί να λειτουργεί στο σύστημα. Συνήθως μεγάλοι σταθμοί εργασίας λόγω χωρητικότητας της μνήμης αλλά και της επεξεργαστικής ισχύς που προσφέρουν, δίνουν και τα καλύτερα αποτελέσματα στην μηχανική όραση, χρησιμοποιώντας βιβλιοθήκες για μηχανική όραση όπως είναι η OpenCV (Open Computer Vision), η οποία έχει μεγάλες απαιτήσεις από το υλικό. Όμως το κόστος αυτόν των σταθμών εργασίας είναι αρκετά μεγάλο. Έτσι η χρήση ενσωματωμένων συστημάτων είναι αρκετά σύνηθες φαινόμενο.

Σκοπός αυτής της πτυχιακής εργασίας είναι η υλοποίηση αλγορίθμων για την επίτευξη μηχανικής όρασης και τον υπολογισμό του κέντρου τον αντικειμένων που έχουν αναγνωριστεί.

Η διάρθρωση της εργασίας χωρίζεται σε έξι κεφάλαια. Στο πρώτο κεφάλαιο γίνεται μια εισαγωγή σε βασικούς όρους των ενσωματωμένων συστημάτων, συστημάτων πραγματικού χρόνου καθώς επίσης και σε επεξεργαστές που είναι κατάλληλοι για ενσωματωμένα συστήματα. Τέλος γίνεται μια επεξήγηση των FPGA και τον ASIC κυκλωμάτων. Στο δεύτερο κεφάλαιο γίνεται περιγραφή της αρχιτεκτονικής του συστήματος που χρησιμοποιήθηκε για την υλοποίηση του πειράματος. Αρχικά γίνεται μια περιγραφή στα ολοκληρωμένα κυκλώματα, στην τεχνολογία CMOS και στον δίαυλο I^2C. Έπειτα περιγράφεται η επικοινωνία της κάμερας με την FPGA μέσω του διαύλου I^2C και τέλος γίνεται αναφορά στα χαρακτηριστικά της FPGA που χρησιμοποιείται στο σύστημα. Στο τέταρτο κεφάλαιο γίνεται περιγραφή των βασικών εργαλείων για τον προγραμματισμό της FPGA του συστήματος. Αρχικά γίνεται μια αναφορά στης λειτουργίες που έχει ένας μεταγλωττιστής και στο τί είναι η διαμεταγλώττιση. Έπειτα γίνεται αναφορά στην γλώσσα προγραμματισμού C που χρησιμοποιήθηκε, στον μεταγλωττιστή GCC και στο RTEMS το οποίο προσέφερε αυτό το εργαλείο. Τέλος γίνεται αναφορά στο τερματικό Cygwin το οποίο χρησιμοποιήθηκε και ήταν το βασικό εργαλείο για τον προγραμματισμό της FPGA. Στο πέμπτο κεφάλαιο αναλύεται το μοντέλο RGB και πώς αυτό το μοντέλο χρησιμοποιείται στο σύστημά μας και τέλος αναλύεται η τεχνική white balance η οποία είχε βασικό ρόλο στην σωστή λειτουργία των αλγορίθμων του συστήματος. Στο έκτο κεφάλαιο αναλύονται η αλγόριθμοι οι οποίοι σχεδιάστηκαν για το σύστημα. Αρχικά αναλύεται ο αλγόριθμος για την αναγνώριση του χρώματος στην εικόνα και έπειτα ο αλγόριθμος για τον υπολογισμό του κέντρου των αντικειμένων καθώς και ο αλγόριθμος Connected Component Labeling. Τέλος γίνεται μια αναφορά στα προβλήματα τα οποία αντιμετωπίστηκαν καθώς επίσης και στα συμπεράσματα τα οποία παρήχθησαν μετά το πέρας του πειράματος και στις πιθανές μελλοντικές επεκτάσεις του συστήματος.

Βασικοί όροι ενσωματωμένων συστημάτων

1.1. Εισαγωγή

Σε αυτό το κεφάλαιο γίνεται μια εισαγωγή σε βασικούς όρους των ενσωματωμένων συστημάτων, οι οποίοι χρησιμοποιούνται και περεταίρω σε αυτή την πτυχιακή εργασία, καθώς έχουν χρησιμοποιηθεί ως συστατικά στοιχεία στο σύστημα για την υλοποίηση του πειράματος. Σε αυτό το κεφάλαιο θα δούμε γενικούς όρους, όπως για παράδειγμα τι είναι τα ενσωματωμένα συστήματα και τι είναι τα συστήματα πραγματικού χρόνου, καθώς επίσης και σε πιο συγκεκριμένες έννοιες, όπως τι είναι μια προγραμματιζόμενη στο πεδίο συστοιχία πυλών (FPGA) ή τα ολοκληρωμένα κυκλώματα ειδικού σκοπού (ASIC). Τέλος γίνεται μία αναφορά σε δύο είδη αρχιτεκτονικών, οι οποίες χρησιμοποιούνται στα ενσωματωμένα συστήματα και είναι η αρχιτεκτονική von Neumann και η αρχιτεκτονική Harvard.

1.2. Τί είναι τα ενσωματωμένα συστήματα

Ένα ενσωματωμένο σύστημα είναι ένα υπολογιστικό σύστημα ειδικού σκοπού σχεδιασμένο για να κάνει συγκεκριμένες λειτουργίες και συνήθως λειτουργεί υπό περιορισμούς πραγματικού χρόνου.

Τέτοια συστήματα αποτελούν ένα κομμάτι μεγαλύτερων και ολοκληρωμένων συστημάτων, περιλαμβάνοντας υλικό ή ακόμα και μηχανικά μέρη. Σε αντίθεση ένα υπολογιστικό σύστημα γενικού σκοπού όπως για παράδειγμα ένας προσωπικός υπολογιστής, έχει σχεδιαστεί για να καλύπτει ένα μεγαλύτερο φάσμα αναγκών ανάλογο με τις ανάγκες του τελικού χρήστη. Τα ενσωματωμένα συστήματα ελέγχουν πολλές συσκευές, οι οποίες αποτελούν μέρος της καθημερινότητάς μας. Το βασικότερο μέρος ή αλλιώς η καρδιά ενός ενσωματωμένου συστήματος είναι ο πυρήνας επεξεργασίας, ο οποίος μπορεί να είναι είτε ένας μικροελεγκτής, είτε ένας επεξεργαστής ψηφιακού σήματος (DSP). Ωστόσο το σημαντικότερο χαρακτηριστικό, το οποίο θα πρέπει να θυμόμαστε για τα ενσωματωμένα συστήματα, είναι πως έχουν σχεδιαστεί για συγκεκριμένες λειτουργίες. Από τη στιγμή που έχουν σχεδιαστεί για συγκεκριμένες λειτουργίες, μπορούν να βελτιστοποιηθούν, με τέτοιο τρόπο ώστε να μειωθεί το φυσικό μέγεθος και η τιμή του εκάστοτε προϊόντος και να αυξηθεί η αξιοπιστία και η απόδοση. Το φυσικό μέγεθος ενός ενσωματωμένου συστήματος διακυμαίνεται μεταξύ φορητών συσκευών όπως για παράδειγμα ένα ψηφιακό ρολόι και σε μεγάλες σταθερές εγκαταστάσεις, υβριδικά οχήματα ή σε ελεγκτές εργοστασίων. [1]

1.2.1. Χρήση των ενσωματωμένων συστημάτων

- Σε κινητήρες οχημάτων και αυτόματα συστήματα φρένων (ABS)
- Μηχανήματα αναλήψεως (ATM)
- Οικιακές συσκευές
- Δικτυακές συσκευές όπως για παράδειγμα switches και routers
- Παιχνιδομηχανές
- Περιφερειακά υπολογιστών (εκτυπωτές, πληκτρολόγια, ποντίκια)
- Αυτοματοποιήσεις σπιτιού, συναγερμοί, θερμοστάτες
- Παγκόσμιο σύστημα εύρεσης θέσεως (GPS)
- Στρατιωτικές εφαρμογές, αυτόματα συστήματα επεξεργασίας

1.2.2. Οι επεξεργαστές των ενσωματωμένων συστημάτων

Οι ενσωματωμένοι επεξεργαστές χωρίζονται σε δυο κύριες κατηγορίες. Η πρώτη κατηγορία αποτελείται από τους συνήθεις μικροεπεξεργαστές (μP), οι οποίοι χρησιμοποιούν ξεχωριστά ολοκληρωμένα κυκλώματα για την μνήμη και ξεχωριστά για τα περιφερειακά. Η δεύτερη κατηγορία αποτελείται από τους μικροελεγκτές (μC), οι οποίοι έχουν πολύ περισσότερα περιφερειακά πάνω στο chip μειώνοντας με αυτόν τον τρόπο την κατανάλωση ενέργειας, το μέγεθος και το κόστος του ενσωματωμένου συστήματος. Σε αντίθεση με τους προσωπικούς υπολογιστές η αρχιτεκτονική των επεξεργαστών, η οποία χρησιμοποιείται στα ενσωματωμένα συστήματα, είναι ανάλογη με το λογισμικό που χρησιμοποιείται για εφαρμογές ειδικού σκοπού. Μπορεί να χρησιμοποιηθεί η αρχιτεκτονική Von Neumann ή η αρχιτεκτονική Harvard. Επιπλέον ο σχεδιασμός του επεξεργαστή μπορεί να είναι τύπου RISC ή non-RISC. Το μέγεθος των λέξεων ποικίλλει από 4 bit έως 64 bit και άνω. Συνήθως στα ενσωματωμένα συστήματα χρησιμοποιείται μέγεθος λέξης 8 bit ή 16 bit. Τέλος πρέπει να αναφερθεί πως για ενσωματωμένα συστήματα έχουν σχεδιαστεί σε μεγάλο αριθμό μικροελεγκτές, οι οποίοι σε πολλές περιπτώσεις είναι προτιμότεροι από τους μικροεπεξεργαστές, διότι οι μικροεπεξεργαστές χρειάζονται μεγαλύτερη υποστήριξη από επιπλέον κυκλώματα σε σχέση με τους μικροελεγκτές. [1]

1.2.3. Πλήρες ενσωματωμένο σύστημα

Τα πλήρη ενσωματωμένα συστήματα είναι συστήματα τα οποία έχουν διαμορφωθεί για μεγάλου όγκου λειτουργίες. Αυτά τα συστήματα ονομάζονται “συστήματα σε ένα τσιπ” (SOC) και αποτελούνται από πολλαπλούς επεξεργαστές, πολλαπλασιαστές, caches και διεπαφές σε ένα μόνο chip. Τέτοια συστήματα μπορούν να υλοποιηθούν σε ASIC κυκλώματα ή με την χρήση FPGAs. [1] (1)

1.3. Ορισμός των συστημάτων πραγματικού χρόνου

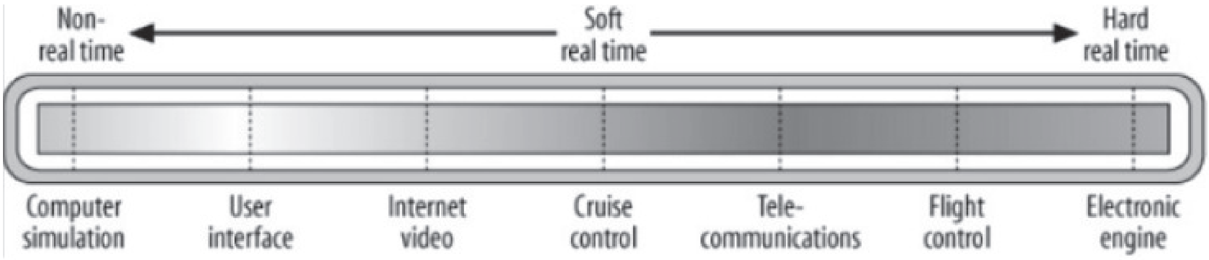
Με τον όρο συστήματα πραγματικού χρόνου εννοούμε τη μελέτη, που γίνεται σε υλικό και λογισμικό, ώστε ένα σύστημα να μπορεί να λειτουργεί υπό τους περιορισμούς που του έχουν τεθεί με βάση τον πραγματικό χρόνο. Δηλαδή είναι ο χρόνος από τη στιγμή που συμβαίνει ένα γεγονός μέχρι την ανταπόκριση του συστήματος στο συγκεκριμένο γεγονός. Ένα λογισμικό πραγματικού χρόνου πρέπει να λειτουργεί σε αυστηρούς χρονικούς περιορισμούς. Συχνά τα συστήματα πραγματικού χρόνου έχουν απόκριση σε χιλιοστά του δευτερολέπτου (milliseconds) ή ακόμα και σε μικροδευτερόλεπτα (microseconds). Σε αντίθεση ένα σύστημα το οποίο δεν είναι πραγματικού χρόνου, δεν μπορεί να εγγυηθεί τους χρόνους απόκρισής του σε ένα συγκεκριμένο γεγονός. [1][2]

1.3.1. Κριτήρια για ένα σύστημα πραγματικού χρόνου

Για να θεωρείται ένα σύστημα ότι είναι σύστημα πραγματικού χρόνου πρέπει να είναι ακριβές. Δηλαδή εκτός από την ορθότητα των αποτελεσμάτων που επιστρέφει, πρέπει να είναι ακριβές στους χρονικούς περιορισμούς που του έχουν τεθεί και να επιστρέφει τα αποτελέσματα μέσα σε αυτούς τους χρονικούς περιορισμούς. Τα συστήματα πραγματικού χρόνου κατηγοριοποιούνται ως εξής:

- Hard real-Time system. Σε αυτήν την κατηγορία αν ένα σύστημα παραβεί τους χρονικούς περιορισμούς που του έχουν τεθεί, τότε θεωρείται πως όλο το σύστημα έχει αποτύχει.
- Soft real-Time system. Σε αυτήν την κατηγορία ανήκουν τα συστήματα στα οποία οι χρονικοί περιορισμοί που τους έχουν τεθεί δεν είναι αυστηρά τηρούμενοι, με αποτέλεσμα η ποιότητα του συστήματος να είναι αρκετά χαμηλή, καθώς η χρησιμότητα των αποτελεσμάτων που επιστρέφει το σύστημα μετά τον χρονικό περιορισμό που του έχει τεθεί είναι μειωμένη.

Έτσι ο σκοπός ενός Hard real-Time συστήματος είναι η εξασφάλιση του χρονικού περιορισμού που του έχει τεθεί, ενώ ένα Soft real-Time σύστημα έχει ως σκοπό να εξασφαλίζει ένα υποσύνολο χρονικών περιορισμών βελτιστοποιώντας τα κριτήρια κάποιων συγκεκριμένων εφαρμογών. Παράδειγμα χρήσης ενός Hard real-Time συστήματος είναι ο έλεγχος της μηχανής ενός οχήματος. Η καθυστέρηση ενός σήματος μπορεί να προκαλέσει ζημιά στην μηχανή ή ακόμα να είναι απειλητική για την ανθρώπινη ζωή. Αντίθετα ένα Soft real-Time σύστημα μπορεί να θεωρηθεί η ζωντανή μετάδοση ενός βίντεο. Μπορεί να υπάρχουν καθυστερήσεις στο βίντεο με αποτέλεσμα την μειωμένη ποιότητα, αλλά αυτό δεν εμποδίζει την λειτουργικότητα του συστήματος. Στην Εικόνα 1-1 μπορούμε να δούμε κάποια παραδείγματα όπου γίνεται η διαφοροποίηση των Hard real-Time συστημάτων, των Soft real-Time συστημάτων και των συστημάτων που δεν ακλουθούν κάποιους χρονικούς περιορισμούς. [1][2]



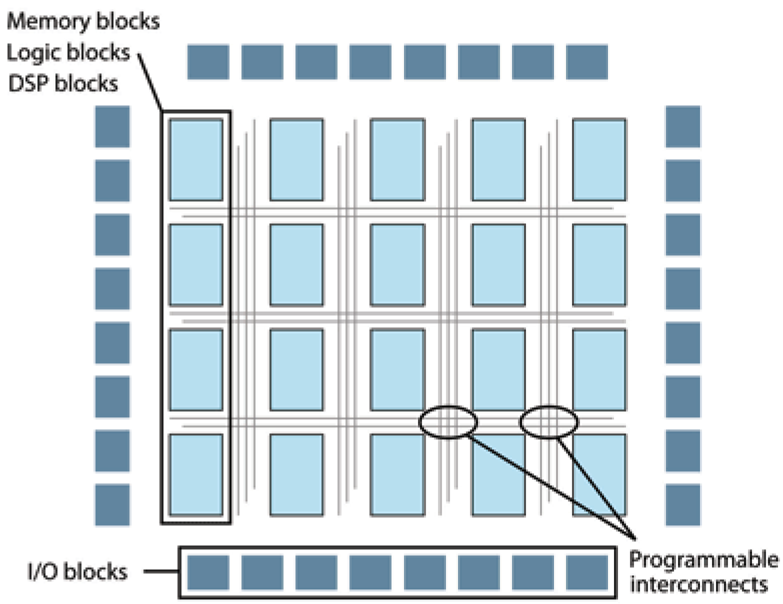
Εικόνα 1-1. Παράδειγμα χρήσης των Hard real-Time συστημάτων και των Soft real-Time συστημάτων [1]

1.4. Τί είναι μια FPGA

Μια FPGA ή προγραμματιζόμενη στο πεδίο συστοιχία πυλών είναι ένα ολοκληρωμένο κύκλωμα το οποίο είναι παραμετροποιήσημο από τον χρήστη. Η παραμετροποίηση μιας FPGA γίνεται με την χρήση μιας γλώσσας υλικού, η οποία μπορεί να είναι η VHDL ή Verilog. Τα σύγχρονα FPGAs αποτελούνται από πολλές λογικές πύλες και block μνήμης (RAM) με σκοπό να υλοποιούν πολύπλοκούς ψηφιακούς υπολογισμούς. Μια FPGA αποτελείται από λογικές πράξεις (όπως AND, OR κτλ.), από blocks εισόδου εξόδου για την συνδεσιμότητα με τα PINS την FPGA και από επαναπρογραμματιζόμενες διασυνδέσεις, ώστε όλα τα στοιχεία μέσα σε μια FPGA να είναι διασυνδεμένα. Όπως βλέπουμε και στην Εικόνα 1-2 τα λογικά blocks είναι τοποθετημένα σε έναν πίνακα δυο διαστάσεων και οι διασυνδέσεις είναι οργανωμένες ως οριζόντια και κάθετα κανάλια δρομολόγησης ανάμεσα από τα λογικά blocks. Τα κανάλια δρομολόγησης εμπεριέχουν καλώδια και επαναπρογραμματιζόμενους διακόπτες, επιτρέποντας στα λογικά blocks να διασυνδέονται με πολλούς διαφορετικούς τρόπους μεταξύ τους. Η FPGA έχει παρόμοιο πεδίο εφαρμογών με άλλα προγραμματιζόμενα ολοκληρωμένα ψηφιακά κυκλώματα, όπως τα PLD (Programmable Logic Device) και τα ASIC. Όμως τα ιδιαίτερα χαρακτηριστικά της FPGA είναι τα εξής :

- Η FPGA χάνει τον προγραμματισμό της κάθε φορά που διακόπτεται η τάση τροφοδοσίας της. Επομένως απαιτεί εξωτερικό μικροεπεξεργαστή ή μνήμη με μόνιμη συγκράτηση δεδομένων (non-volatile memory) από τα οποία θα προγραμματίζεται κάθε φορά που επανέρχεται η τάση τροφοδοσίας.

- Ο προγραμματισμός της FPGA μπορεί να αλλάζει κάθε φορά που τροποποιείται το λογισμικό του μικροεπεξεργαστή ή τα δεδομένα της μνήμης που το ελέγχει.
- Δεν υπάρχει όριο στο πόσες φορές μπορεί να επαναπρογραμματιστεί.
- Η κατανάλωση ισχύος είναι σημαντικά αυξημένη σε σχέση με τα ASIC. [3][4]



Εικόνα 1-2. Παράδειγμα αρχιτεκτονικής μιας FPGA (2)

1.5. Τί είναι τα ASIC

Η ανάπτυξη της τεχνολογίας των ολοκληρωμένων κυκλωμάτων πέραν του επιπέδου LSI οδήγησε στα κυκλώματα πολύ μεγάλης κλίμακας ολοκλήρωσης (VLSI). Σε εκείνο το στάδιο, έγινε ξεκάθαρο ότι η βιομηχανία σύντομα θα εξαντλούσε τους υπερθετικούς βαθμούς για να τους βάλει ως πρόθεμα στο “LSI”. Έτσι, ο όρος VLSI κατέληξε να αναφέρεται περισσότερο στον τρόπο με τον οποίο σχεδιάζονταν τα ολοκληρωμένα κυκλώματα παρά στον αριθμό των τρανζίστορ που περιείχαν. Ο όρος τώρα, συνήθως, σημαίνει την αναλυτική σχεδίαση σε επίπεδο κυκλώματος (circuit design) των ολοκληρωμένων κυκλωμάτων σε αντίθεση με τη σχεδίαση σε επίπεδο συστήματος (system-level design). Η εκτεταμένη διαθεσιμότητα των εργαλείων CAD για τη σχεδίαση VLSI και η ανάπτυξη της βιομηχανίας σε υπηρεσίες κατασκευής ολοκληρωμένων κυκλωμάτων έχει κάνει τώρα εφαρμόσιμη την κατασκευή ολοκληρωμένων κυκλωμάτων για ευρύ φάσμα εφαρμογών. Χρησιμοποιούμε τον όρο ολοκληρωμένο κύκλωμα ειδικού σκοπού, ή ASIC, για να αναφερθούμε σε ένα ολοκληρωμένο κύκλωμα που κατασκευάζεται για μια συγκεκριμένη εφαρμογή. Αυτό δεν είναι για να πούμε ότι ένα ASIC κατασκευάζεται απαραίτητα μόνο για έναν πελάτη ή για ένα έργο. Μάλλον, σχεδιάζεται για να ικανοποιήσει ένα συγκεκριμένο σύνολο απαιτήσεων, και έτσι περιέχει κυκλώματα που έχουν προσαρμοστεί σε αυτές τις απαιτήσεις. Μπορεί να σχεδιαστεί για ένα συγκεκριμένο τελικό προϊόν που παρέχεται από έναν κατασκευαστή, για παράδειγμα, μια φορητή συσκευή μουσικής, ένα παιχνίδι, ένα αυτοκίνητο, ένα κομμάτι ενός στρατιωτικού εξοπλισμού, ή μια βιομηχανική μηχανή. Εναλλακτικά, μπορεί να σχεδιαστεί για χρήση σε ένα εύρος προϊόντων που παρέχονται από τους κατασκευαστές για ένα συγκεκριμένο τμήμα της αγοράς. Αυτού του είδους τα ASIC μερικές φορές ονομάζονται πρότυπα προϊόντα εξειδικευμένα για εφαρμογές, ή ASSP, αφού αντιμετωπίζονται ως πρότυπα κομμάτια μέσα σε αυτό το τμήμα της αγοράς, αλλά δε χρησιμοποιούνται έξω από αυτό το τμήμα. Τέτοια παραδείγματα περιλαμβάνουν ολοκληρωμένα κυκλώματα για κινητά τηλέφωνα, τα οποία χρησιμοποιούνται από έναν αριθμό ανταγωνιστών στην κατασκευή κινητών τηλεφώνων, αλλά δεν χρησιμοποιούνται για παράδειγμα σε κυκλώματα ελέγχου οχημάτων.

Ένας από τους βασικούς λόγους για τους οποίους θα αναπτύσσαμε ένα ASIC για ένα προϊόν είναι ότι, επειδή είναι προσαρμοσμένο στις απαιτήσεις αυτής της εφαρμογής, έχει χαμηλότερο κόστος ανά ολοκληρωμένο κύκλωμα από ότι ένα προγραμματιζόμενο στοιχείο, όπως μια FPGA. Ωστόσο, για να επιτύχουμε αυτό το επίπεδο προσαρμογής στις απαιτήσεις της εφαρμογής, χρειάζεται να επενδύσουμε πολύ περισσότερη προσπάθεια για τη σχεδίαση και την επαλήθευση. Πρέπει να αποσβέσουμε το μη επαναλαμβανόμενο κόστος τεχνικής μελέτης (non-recurring engineering – NRE) σε όλα τα κομμάτια του προϊόντος που θα πουληθούν. Επομένως, έχει νόημα να χρησιμοποιήσουμε ένα ASIC μόνο εάν ο αριθμός πωλήσεων του προϊόντος μας είναι αρκετά μεγάλος. Το κόστος NRE που θα αποσβέσουμε ανά κομμάτι θα πρέπει να είναι μικρότερο από τη διαφορά του κόστους μεταξύ ενός ASIC και ενός προγραμματιζόμενου κομματιού. Αυτό, βέβαια, προϋποθέτει ότι είναι εφικτό να χρησιμοποιηθεί ένα προγραμματιζόμενο συστατικό στοιχείο. Εάν η εφαρμογή απαιτεί ένα επίπεδο απόδοσης που δεν μπορεί να επιτευχθεί με μία FPGA, τότε η μοναδική πραγματική μας επιλογή είναι ένα ASIC ή ένα ASSP, και το υψηλότερο κόστος NRE είναι ένα αναπόφευκτο τμήμα του κόστους του προϊόντος.

Υπάρχουν δυο βασικές τεχνικές σχεδίασης και κατασκευής για ASIC, που διαφοροποιούνται ως προς το βαθμό προσαρμογής στις απαιτήσεις της εφαρμογής. Πρώτον τα πλήρως προσαρμοσμένα (fully custom) ολοκληρωμένα κυκλώματα εμπεριέχουν αναλυτική σχεδίαση όλων των τρανζίστορ και των συνδέσεων σε ένα ASIC. Αυτό επιτρέπει πιο αποδοτική χρήση των πόρων του υλικού σε ένα ολοκληρωμένο κύκλωμα και έχει ως αποτέλεσμα υψηλότερη απόδοση, αλλά έχει υψηλό κόστος NRE και απαιτεί υψηλή τεχνική κατάρτιση στη σχεδίαση VLSI εντός της σχεδιαστικής ομάδας. Αυτό έχει ως επακόλουθο, τα πλήρως προσαρμοσμένα SIC να σχεδιάζονται συνήθως μόνο για προϊόντα που πωλούνται σε μεγάλες ποσότητες, όπως CPU και ολοκληρωμένα κυκλώματα για καταναλωτικές συσκευές. Δεύτερον, τα ASIC με πρότυπα κελιά (standard cell) εμπεριέχουν επιλογή βασικών κελιών, όπως πύλες και flip-flop, από μια βιβλιοθήκη για να σχηματιστεί το κύκλωμα. Τα κελιά έχουν προηγουμένως σχεδιαστεί από έναν κατασκευαστή ολοκληρωμένων κυκλωμάτων ή έναν προμηθευτή ASIC, και χρησιμοποιούνται από το εργαλείο σύνθεσης κατά τη διάρκεια της διαδικασίας σχεδίασης για να υλοποιήσουν τη σχεδίαση. Η αξία αυτής της προσέγγισης είναι ότι το κόστος NRE για κάθε σχεδίαση ASIC μειώνεται σημαντικά, αφού γίνεται απόσβεση του κόστους σχεδίασης της βιβλιοθήκης των κελιών σε έναν αριθμό σχεδιάσεων ASIC. Ο συμβιβασμός είναι ότι το ASIC μπορεί να μην είναι τόσο πυκνό ή να μην έχει την απόδοση ενός πλήρως προσαρμοσμένου ASIC. [5]

Με την πάροδο του χρόνου το μέγεθος των ASIC έχει μειωθεί σημαντικά, τα εργαλεία σχεδιασμού του ASIC έχουν βελτιστοποιηθεί και η πολυπλοκότητα των ASIC κυκλωμάτων έχει αυξηθεί από τις 5.000 πύλες στις 100 εκατομμύρια πύλες. Τα ASIC τελευταίας γενιάς εμπεριέχουν μικροεπεξεργαστές, block μνήμης όπως RAM, ROM, EEPROM και Flash. Οι σχεδιαστές των ASIC κυκλωμάτων χρησιμοποιούν γλώσσες περιγραφής υλικού (HDL) για τον σχεδιασμό και την περιγραφή λειτουργίας των ASIC, όπως είναι η VHDL ή η Verilog.

Σε αντίθεση με τις FPGA τα ASIC δεν είναι επαναπρογραμματιζόμενα. Γι’ αυτόν το λόγο ονομάζονται και κυκλώματα ειδικού σκοπού. Αλλά αυτό που προσφέρουν είναι μειωμένο μέγεθος, σημαντικά μειωμένη κατανάλωση ενέργειας, ενώ το κόστος που έχουν επιτρέπει τη μαζική παραγωγή τους.

1.6. Αναφορά στις αρχιτεκτονικές μνήμης Von Neumann και Harvard

Η αρχιτεκτονική Harvard έχει διαφορετικό δίαυλο επικοινωνίας για τα δεδομένα και διαφορετικό για τις εντολές, επιτρέποντας με αυτόν τον τρόπο την ταυτόχρονη μεταφορά των δεδομένων και των εντολών. Αντίθετα η αρχιτεκτονική Von Neumann έχει μόνο έναν δίαυλο επικοινωνίας, ο οποίος χρησιμοποιείται και για τα δεδομένα και για τις εντολές. Ως εκ τούτου η μεταφορά των δεδομένων και των εντολών στην αρχιτεκτονική Von Neumann πρέπει να είναι προγραμματισμένες, γιατί δεν μπορούν να εκτελεστούν ταυτόχρονα. (3)

Αρχιτεκτονική υλικού

2.1. Εισαγωγή

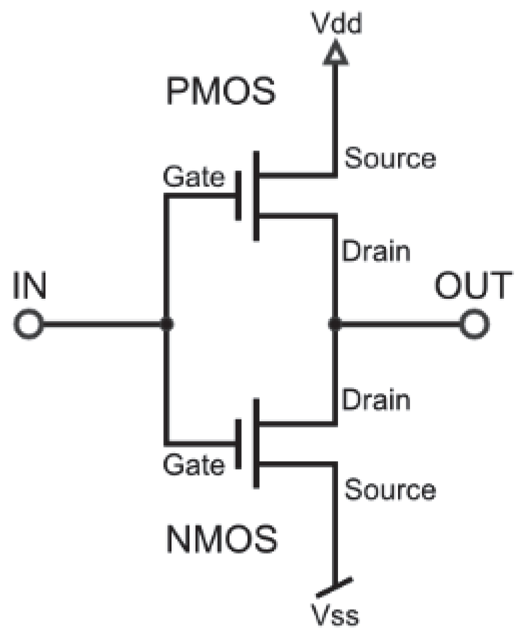
Στο προηγούμενο κεφάλαιο γίνεται μια σύντομη εισαγωγή σε βασικούς όρους που θα συναντήσουμε στη συνέχεια αυτής της πτυχιακής εργασίας. Σε αυτό το κεφάλαιο γίνεται μια περιγραφή στην αρχιτεκτονική του συστήματος που χρησιμοποιείται για το πείραμα, σε πρωτόκολλα επικοινωνίας του συστήματος (I^2C) και στον τύπο της κάμερας που χρησιμοποιείται. Επίσης γίνεται αναφορά στα χαρακτηριστικά της FPGA που χρησιμοποιείται στο σύστημα.

2.2. Ολοκληρωμένα κυκλώματα

Τα σύγχρονα ψηφιακά κυκλώματα κατασκευάζονται στην επιφάνεια ενός μικρού επίπεδου κομματιού καθαρού κρυσταλλικού πυριτίου, ως εκ τούτου και ο συνήθης όρος “chip πυριτίου” (silicon chip). Αυτά τα κυκλώματα ονομάζονται ολοκληρωμένα κυκλώματα, πολυάριθμα εξαρτήματα ολοκληρώνονται μαζί στο chip, αντί να είναι το καθένα ξεχωριστό εξάρτημα. Τα τρανζίστορ σχηματίζονται με την εναπόθεση στρωμάτων ημιαγωγικών και μονωτικών υλικών σε ορθογώνια και πολυγωνικά σχήματα στην επιφάνεια του chip. Οι αγωγοί σχηματίζονται με την εναπόθεση μετάλλου (συνήθως χαλκού) πάνω από τα τρανζίστορ, που διαχωρίζεται με μονωτικά στρώματα.

Οι φυσικές ιδιότητες του ολοκληρωμένου κυκλώματος καθορίζουν πολλά σημαντικά λειτουργικά χαρακτηριστικά, συμπεριλαμβανόμενης και της ταχύτητας εναλλαγής μεταξύ χαμηλού και υψηλού δυναμικού. Μεταξύ των πιο σημαντικών φυσικών ιδιοτήτων είναι το ελάχιστο μέγεθος κάθε στοιχείου, το ονομαζόμενο *ελάχιστο χαρακτηριστικό μέγεθος* (minimum feature size). Τα πρώτα chip είχαν ελάχιστα χαρακτηριστικά μεγέθη κάποιων δεκάδων μικρομέτρων (1 μικρόμετρο = 1micron = 1μm = 10^{-6}m). Οι βελτιώσεις στην τεχνολογία κατασκευής έχουν οδηγήσει σε μια σταθερή μείωση του χαρακτηριστικού μεγέθους από τα 10μm στις αρχές της δεκαετίας του 1970, στο 1μm στα μέσα της δεκαετίας του 1980, ενώ με τα σημερινά ολοκληρωμένα κυκλώματα να έχουν χαρακτηριστικά μεγέθη των 90nm ή 65nm. Μαζί με την επίδραση στην απόδοση του κυκλώματος, το χαρακτηριστικό μέγεθος βοηθά στον καθορισμό του αριθμού των τρανζίστορ που μπορούν να χωρέσουν σε ένα ολοκληρωμένο κύκλωμα, και συνεπώς της συνολικής πολυπλοκότητας του κυκλώματος. Ο Gordon Moore, ένας από τους πρωτοπόρους της βιομηχανίας ψηφιακών ηλεκτρονικών παρατήρησε την τάση στην αύξηση του αριθμού των τρανζίστορ και δημοσίευσε ένα άρθρο για το θέμα αυτό το 1965. Η πρόβλεψη που έκανε για μια συνεχιζόμενη τέτοια τάση συνεχίζει να ισχύει μέχρι και σήμερα και είναι γνωστή σαν Νόμος του Moore (Moore's Law). Ισχυρίζεται ότι το πλήθος των τρανζίστορ που μπορούν να τοποθετηθούν σε ένα ολοκληρωμένο κύκλωμα με ελάχιστο κόστος στοιχείου διπλασιάζεται κάθε 18 μήνες. Όταν δημοσιεύθηκε το άρθρο του Moore το πλήθος αυτό ήταν περίπου 50 τρανζίστορ, σήμερα είναι πολύ περισσότερο από ένα δισεκατομμύριο τρανζίστορ.

Μια από τις πρώτες οικογένειες ολοκληρωμένων κυκλωμάτων ψηφιακής λογικής που πέτυχε ευρύτατη χρήση ήταν η οικογένεια "transistor-transistor logic" (TTL). Οι συσκευές αυτής της οικογένειας χρησιμοποιούν τρανζίστορ διπολικής επαφής (bipolar junction transistor) συνδεδεμένα κατάλληλα για το σχηματισμό λογικών πυλών. Οι ηλεκτρικές ιδιότητες αυτών των συσκευών οδήγησαν σε ευρέως αποδεκτά σχεδιαστικά πρότυπα που ακόμη επιδρούν στις σημερινές πρακτικές λογικής σχεδίασης. Πιο πρόσφατα, οι συσκευές TTL έχουν υποσκελιστεί σε μεγάλο βαθμό από συσκευές που χρησιμοποιούν κυκλώματα "συμπληρωματικού ημιαγωγού οξειδίου μετάλλου" ("complementary metaloxide semiconductor" – CMOS) που βασίζονται σε τρανζίστορ επίδρασης πεδίου (field effect transistor – FET). Ο όρος "συμπληρωματικός" σημαίνει ότι χρησιμοποιούνται τόσο n-channel όσο και p-channel MOSFET. Η Εικόνα 2-1 δείχνει πόσα τρανζίστορ χρησιμοποιούνται σε ένα κύκλωμα CMOS για έναν αντιστροφέα. Όταν η τάση εισόδου είναι χαμηλή, το n-channel τρανζίστορ στο κάτω μέρος είναι κλειστό και το p-channel τρανζίστορ στο επάνω μέρος είναι ανοικτό τραβώντας την έξοδο στο υψηλό δυναμικό. Αντίστροφα, όταν η τάση εισόδου είναι υψηλή, το p-channel τρανζίστορ είναι κλειστό και το n-channel τρανζίστορ είναι ανοικτό τραβώντας την έξοδο στο χαμηλό δυναμικό. Τα κυκλώματα άλλων λογικών πυλών λειτουργούν παρόμοια, οδηγώντας συνδυασμούς τρανζίστορ στο ανοικτό ή κλειστό τρανζίστορ για να τραβήξουν την έξοδο στο χαμηλό ή το υψηλό δυναμικό[1] ανάλογα με την τάση στις εισόδους. [5]



Εικόνα 2-1. Κύκλωμα CMOS για έναν αντιστροφέα [5]

2.3. Τεχνολογία CMOS

Η κάμερα που χρησιμοποιήθηκε για το πείραμα είναι τεχνολογίας CMOS. Ο συμπληρωματικός ημιαγωγός οξειδίου του μετάλλου (CMOS) είναι μια τεχνολογία για την κατασκευή ολοκληρωμένων κυκλωμάτων. Η τεχνολογία CMOS χρησιμοποιείται σε μικροεπεξεργαστές, σε μικροελεγκτές σε στατικές RAM και σε άλλα ψηφιακά λογικά κυκλώματα. Επιπλέον η τεχνολογία CMOS χρησιμοποιείται σε πολλά αναλογικά κυκλώματα, όπως είναι οι αισθητήρες εικόνας[2], οι μετατροπείς δεδομένων και οι πομποδέκτες διαφόρων τύπων επικοινωνίας. Ο Frank Wanlass κατοχύρωσε ως ευρεσιτεχνία την τεχνολογία CMOS το 1967.

Η λέξη "συμπληρωματικός" στην ονομασία της συγκεκριμένης τεχνολογίας, όπως αναφέρεται στο προηγούμενο κεφάλαιο, οφείλεται στο γεγονός ότι χρησιμοποιούνται

[1] Το χαμηλό και το υψηλό δυναμικό εκφράζονται αντίστοιχα από τις λογικές τιμές "0" και "1"

[2] Η κάμερα που χρησιμοποιήθηκε στο σύστημα του πειράματος χρησιμοποιεί αισθητήρα εικόνας, διότι μετατρέπει την εικόνα σε ηλεκτρικά σήματα και τα επιστρέφει πίσω στην FPGA.

συμπληρωματικά και συμμετρικά ζευγάρια τύπου p και τύπου n (MOSFETs) για λογικές λειτουργίες. Δυο βασικά χαρακτηριστικά των συσκευών CMOS είναι η υψηλή ανοχή σε παράγοντες θορύβου και η χαμηλή κατανάλωση ενέργειας. Επίσης οι συσκευές CMOS δεν παράγουν τόση θερμότητα[3] όση παράγουν άλλου είδους λογικές συσκευές, όπως είναι η TTL ή η NMOS. Τέλος η τεχνολογία CMOS επιτρέπει να υπάρχει μεγάλο ποσοστό λογικών πυλών μέσα σε ένα chip και αυτός ήταν ο βασικός λόγος που η τεχνολογία CMOS έγινε η προτιμώμενη τεχνολογία για χρήση σε chip VLSI. [11]

2.3.1. Χαρακτηριστικά τεχνολογίας CMOS

Η τεχνολογία CMOS αποτελεί μια επιλογή του σχεδιαστή ανάμεσα από ένα σύνολο επιλογών που του προσφέρει γενικά η τεχνολογία για τον ηλεκτρονικό σχεδιασμό συστημάτων. Οι άλλες επιλογές είναι η διπολική τεχνολογία πυριτίου (silicon bipolar technology), η τεχνολογία Αρσενικούχου Γαλίου (Gallium Arsenide) και η τεχνολογία επαφής Josephson. Πιο αναλυτικά, από τις διαθέσιμες τεχνολογίες η τεχνολογία του Αρσενικούχου Γαλίου συχνά παρουσιάζει την υψηλότερη ταχύτητα ανά πύλη. Οι διπολικές τεχνολογίες δεν είναι μακριά από τη τεχνολογία του Αρσενικούχου Γαλίου, ενώ οι προηγμένες τεχνολογίες CMOS είναι ελάχιστα πίσω από τις διπολικές. Γενικά, η τεχνολογία CMOS παρουσιάζει την υψηλότερη πυκνότητα ολοκλήρωσης καθώς και την χαμηλότερη κατανάλωση ισχύος ανά πύλη. Επιπλέον, είναι διαθέσιμη και για αναλογικά κυκλώματα, αλλά τα διπολικά κυκλώματα προσφέρουν καλύτερη απόδοση. Επίσης είναι φθηνότερη για ψηφιακά κυκλώματα υψηλής πυκνότητας με μέτριες απαιτήσεις στα αναλογικά. Το κόστος σχεδίασης είναι το φθηνότερο για την τεχνολογία CMOS εξαιτίας των μεγάλων επενδύσεων που έχουν γίνει σε εργαλεία σχεδιασμού και βιβλιοθήκες κυττάρων. Ο συνδυασμός τεχνολογίας CMOS και διπολικής τεχνολογίας δίνει την εξαιρετικά ενδιαφέρουσα τεχνολογία BiCMOS, η οποία βρίσκει εφαρμογή σε ολοκληρωμένα κυκλώματα όπου υπάρχει ανάγκη για μεικτά σήματα. Για το μεγαλύτερο ποσοστό των ηλεκτρονικών συστημάτων η τεχνολογία που επιλέγεται είναι CMOS. Όμως, θα πρέπει να τονιστεί ότι δεν αποτελεί τη μόνη επιλογή και ο σχεδιαστής όταν παίρνει αποφάσεις στο επίπεδο του συστήματος θα πρέπει να είναι καλά ενημερωμένος για τα πλεονεκτήματα και τα μειονεκτήματα των άλλων τεχνολογιών.

[3] Η κάμερα που χρησιμοποιήθηκε στο σύστημα του πειράματος έχει λειτουργικότητα μεταξύ των -20 και των +60 βαθμών Κελσίου.

Τα κυριότερα χαρακτηριστικά της τεχνολογίας CMOS δίνονται σε συντομία παρακάτω:

- Στάθμες πλήρους αποκαταστάσιμης λογικής, δηλαδή η έξοδος πηγαίνει στο V_{DD} ή V_{SS}.
- Χρόνους μετάβασης – Οι χρόνοι ανόδου και καθόδου είναι της ίδιας τάξεως.
- Οι μνήμες υλοποιούνται με μεγάλη πυκνότητα ολοκλήρωσης και χαμηλή κατανάλωση ισχύος.
- Οι πύλες μετάδοσης περνούν σωστά και τις δύο λογικές στάθμες και επιτρέπουν τη χρήση αποδοτικών και ευρέως διαδεδομένων λογικών δομών, όπως πολυπλέκτες, μανδαλωτές και καταχωρητές.
- Κατανάλωση Ισχύος – Τα πλήρως συμπληρωματικά κυκλώματα έχουν σχεδόν μηδενική στατική κατανάλωση ισχύος. Κατανάλωση υπάρχει μόνο κατά τη διάρκεια των λογικών μεταβάσεων.
- Χαρακτηριστικά προφόρτισης – Τα στοιχεία n και p τύπου είναι διαθέσιμα για την προφόρτιση ενός διαύλου σε V_{DD} και V_{SS}. Οι κόμβοι μπορούν σε πολύ μικρό χρόνο να φορτιστούν πλήρως σε στάθμη V_{DD} ή εναλλακτικά σε V_{SS}.
- Τάση τροφοδοσίας – Η τάση τροφοδοσίας που απαιτείται για να αλλάξει την κατάστασή της μια πύλη είναι ένα ποσοστό του V_{DD}. Η τιμή του V_{DD} κυμαίνεται από 1.5 μέχρι 15 volts.
- Πυκνότητα πακεταρίσματος – Για n εισόδους συμπληρωματικών στατικών πυλών απαιτούνται 2n στοιχεία. Για δυναμικές πύλες και λογικές δομές με κατάλληλο λόγο διαστάσεων των τρανζίστορ απαιτούνται λιγότερες είσοδοι.
- Φυσική σχεδίαση – Η τεχνολογία CMOS δίνει κανονικές και εύκολα αυτοματοποιήσιμες φυσικές σχεδιάσεις. [11]

2.4. Ο δίαυλος επικοινωνίας I^2C

Ο δίαυλος επικοινωνίας I^2C δημιουργήθηκε στις αρχές του 1980 από την εταιρεία Phillips. Αρχικός του σκοπός ήταν να παρέχει μια εύκολη διασύνδεση μεταξύ της κεντρικής μονάδας επεξεργασίας (CPU) και των περιφερειακών chip σε μια τηλεόραση.

Οι περιφερειακές συσκευές στα ενσωματωμένα συστήματα συνήθως διασυνδέονταν με τον επεξεργαστή μέσω ενός μικροελεγκτή χρησιμοποιώντας για μεταφορά δεδομένων και

διευθύνσεων τον δίαυλο επικοινωνίας του μικροελεγκτή. Αυτό είχε σαν αποτέλεσμα την ύπαρξη μεγάλου ποσοστού καλωδιώσεων πάνω σε διάτρητες πλακέτες με σκοπό τη δρομολόγηση των δεδομένων και των διευθύνσεων.
Λόγο του αυξημένου ποσοστού των καλωδιώσεων και των επιπλέον συνδέσμων που έμπαιναν στο κύκλωμα υπήρχε αύξηση των ηλεκτρομαγνητικών παρεμβολών καθώς και της ηλεκτροστατικής αποφόρτισης. Εκτός αυτού το κόστος σε μια μαζική παραγωγή ήταν αρκετά αυξημένο λόγω των επιπλέων καλωδίων και συνδέσμων που υπήρχαν στο κύκλωμα.

Με γνώμονα τα παραπάνω προβλήματα δημιουργήθηκε ο δίαυλος I^2C από την Phillips, ο οποίος βασίζεται σε διασύνδεση ολοκληρωμένων κυκλωμάτων χρησιμοποιώντας δύο καλώδια στα οποία είναι συνδεδεμένες όλες οι περιφερειακές συσκευές. Σήμερα ο συγκεκριμένος δίαυλος έχει γίνει αποδεκτός στον κόσμο της βιομηχανίας. Επίσης έχει εφαρμοστεί και σε άλλα πεδία εκτός από την εσωτερική διασύνδεση ολοκληρωμένων κυκλωμάτων για ήχο και εικόνα όπως στις αρχές της δημιουργίας του. (4)

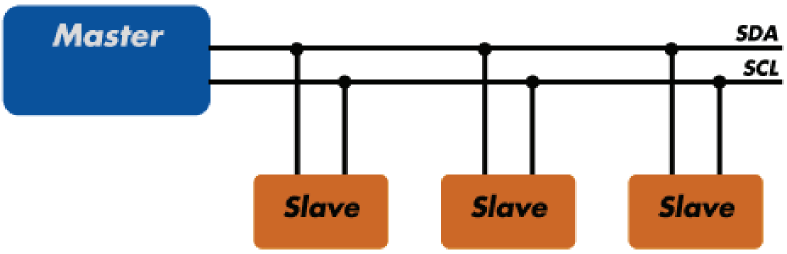
2.4.1. Τρόπος λειτουργίας του διαύλου I^2C

Όπως προαναφέρθηκε η λειτουργία του διαύλου I^2C βασίζεται στην χρήση δύο καλωδίων στα οποία είναι συνδεδεμένες οι περιφερειακές συσκευές. Σε ένα τέτοιο κύκλωμα οι περιφερειακές συσκευές συμπεριφέρονται ως slaves ενώ η κύρια συσκευή που ελέγχει τις περιφερειακές συμπεριφέρεται ως master. Το ένα από τα δύο καλώδια του διαύλου χρησιμοποιείται για σειριακή μεταφορά των δεδομένων (SDA) και το άλλο χρησιμοποιείται ως σειριακό ρολόι (SCL). Στην Εικόνα 2-2 παρουσιάζεται ένα παράδειγμα χρήσης του διαύλου I^2C.

Ο τρόπος επικοινωνίας γίνεται ως εξής:

1. Η συσκευή που συμπεριφέρεται ως master ξεκινάει μια νέα κατάσταση έναρξης. Αυτή η κατάσταση ενημερώνει όλες τις συσκευές που συμπεριφέρονται ως slave ώστε να "ακούν" το καλώδιο σειριακής μεταφοράς των δεδομένων για οδηγίες.
2. Η συσκευή master στέλνει τη διεύθυνση της εκάστοτε συσκευής slave και μια ένδειξη read/write.

3. Η συσκευή slave με την ταυτοποιημένη διεύθυνση απαντάει με ένα σήμα επιβεβαίωσης.
4. Η επικοινωνία συνεχίζεται ανάμεσα στη συσκευή master και στη συσκευή slave μέσω του καλωδίου σειριακής μεταφοράς. Και οι δυο συσκευές μπορούν να στέλνουν ή να λαμβάνουν δεδομένα ανάλογα με το αν η επικοινωνία είναι τύπου read ή write. Ο αποστολέας στέλνει 8 bit δεδομένων στον παραλήπτη και ο παραλήπτης απαντάει με 1 bit επιβεβαίωσης.
5. Όταν η επικοινωνία ολοκληρωθεί, η συσκευή master ξεκινάει μια νέα κατάσταση τερματισμού, η οποία υποδεικνύει ότι η επικοινωνία ολοκληρώθηκε. (5)



Εικόνα 2-2. Επικοινωνία συσκευών με χρήση του διαύλου I^2C (5)

2.4.2. Η λειτουργία του διαύλου I^2C στο σύστημα

Η επικοινωνία της κάμερας του συστήματος με την FPGA γίνεται μέσω του διαύλου I^2C. Συγκεκριμένα χρησιμοποιείται η πρώτη έκδοση του διαύλου I^2C η οποία εκδόθηκε το 1992 με τις εξής τροποποιήσεις σε σχέση με την αρχική υλοποίηση του I^2C:

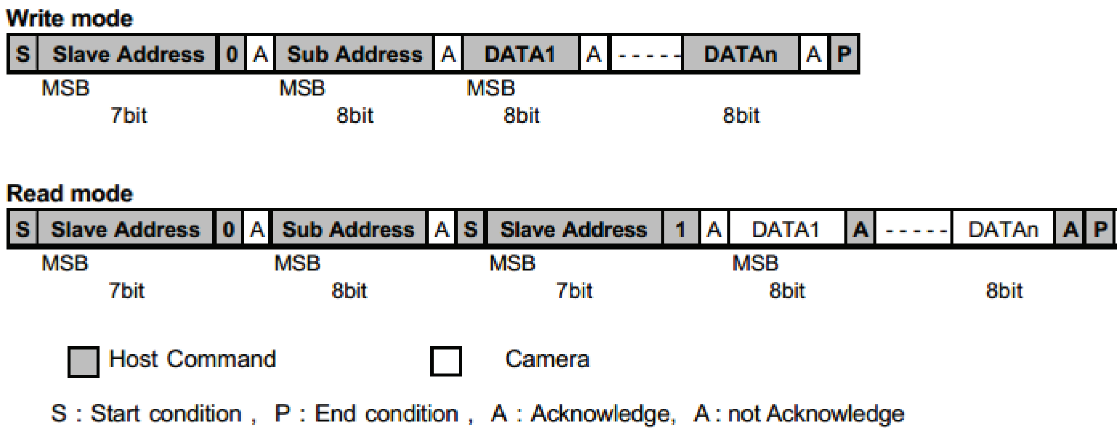
- Ο προγραμματισμός μιας slave διεύθυνσης από το λογισμικό έχει παραληφθεί.
- Η λειτουργία χαμηλής ταχύτητας έχει παραληφθεί από τα χαρακτηριστικά του διαύλου I^2C.
- Η λειτουργία υψηλής ταχύτητας έχει προστεθεί επιτρέποντας τετραπλάσια αύξηση του ρυθμού μετάδοσης στα 400 kbit/s. Περιφερειακές συσκευές που λειτουργούν σε αυτήν την ταχύτητα είναι συμβατές και με την αρχική ταχύτητα του διαύλου I^2C.
- Έχει προστεθεί διευθυνσιοδότηση της τάξης των 10 bit επιτρέποντας με αυτόν τον τρόπο να υπάρχουν επιπλέον 1024 slave διευθύνσεις.

- Οι συσκευές που λειτουργούν με βάση το συγκεκριμένο πρότυπο του I^2C δεν είναι επιρρεπείς σε ηλεκτρομαγνητικές παρεμβολές. (6)

Η κάμερα του συστήματος μπορεί να δεχτεί εντολές μέσω του συγκεκριμένου διαύλου (όπως για παράδειγμα την ενεργοποίηση κάποιων συγκεκριμένων χαρακτηριστικών που προσφέρει η κάμερα), αλλά η εικόνα που επιστρέφει γίνεται από κάποια συγκεκριμένα pins της κάμερας. Ο λόγος για τον οποίο η εικόνα επιστρέφει στο σύστημα από διαφορετικά pins και όχι από τον δίαυλο I^2C είναι γιατί τα bits της εικόνας πρέπει να επιστρέφονται παράλληλα και όχι σειριακά με βάση τα χαρακτηριστικά της κάμερας.

Η κάμερα στο σύστημά μας λειτουργεί ως slave και η FPGA ως master , των οποίων η επικοινωνία αναπαριστάται στην Εικόνα 2-3. Όπως μπορούμε να δούμε από το σχήμα η διαδικασία εγγραφής δεδομένων από την FPGA στην κάμερα γίνεται ως εξής: Η FPGA ξεκινάει μια κατάσταση έναρξης επικοινωνίας, δίνοντας τη διεύθυνση της συσκευής μεγέθους 7 bit, με την οποία θέλει να επικοινωνήσει και στην περίπτωσή μας της κάμερας. Έπειτα η κάμερα απαντάει με ένα μήνυμα επιβεβαίωσης. Εν συνεχεία η FPGA στέλνει ένα μήνυμα με την επιλεγμένη υπο-διεύθυνση της κάμερας μεγέθους 8 bit και η κάμερα απαντάει όπως και παραπάνω με ένα μήνυμα επιβεβαίωσης. Εν τέλει η FPGA αρχίζει να στέλνει δεδομένα μεγέθους 8 bit στην κάμερα και μόλις ολοκληρωθεί η επικοινωνία στέλνει ένα σήμα τερματισμού.

Η ίδια διαδικασία ισχύει και για τη μέθοδο ανάγνωσης της κάμερας από την FPGA, με την διαφορά ότι η διεύθυνση slave την πρώτη φορά στέλνεται με σκοπό ο δείκτης του διαύλου να δείχνει μια συγκεκριμένη διεύθυνση μνήμης και έπειτα ακολουθεί μια νέα κατάσταση έναρξης με ενεργοποιημένο το bit ανάγνωσης στην slave διεύθυνση, ώστε να γίνει ανάγνωση της συγκεκριμένης θέσης μνήμης της κάμερας.



Εικόνα 2-3. Τρόπος χρήσης του διαύλου I^2C στο σύστημα (7)

2.5. Οικογένεια Spartan-3 της Xilinx

Η FPGA του συστήματος του πειράματος ανήκει στην οικογένεια Spartan-3 της εταιρίας Xilinx. Η οικογένεια Spartan-3 αποτελείται από οκτώ μέλη προσφέροντας πυκνότητα λογικών πυλών, η οποία κυμαίνεται από 50.000 έως 5.000.000. Η οικογένεια Spartan-3 βασίζεται στην επιτυχία που είχε η οικογένεια Spartan-IIE αυξάνοντας το ποσοστό των λογικών πόρων, τη χωρητικότητα της εσωτερικής RAM, τον συνολικό αριθμό εισόδων/εξόδων και το συνολικό επίπεδο επίδοσης βελτιώνοντας τις λειτουργίες διαχείρισης του ρολογιού. Στον Πίνακα 2-1 αναπαρίστανται τα χαρακτηριστικά των FPGA που ανήκουν στην οικογένεια Spartan-3. (8)

Device	System Gates	Equivalent Logic Cells[1]	CLB Array (One CLB = Four Slices)			Distributed RAM Bits (K=1024)	Block RAM Bits (K=1024)	Dedicated Multipliers	DCMs	Max. User I/O	Maximum Differential I/O Pairs
			Rows	Columns	Total CLBs						
XC3S50[2]	50K	1,728	16	12	192	12K	72K	4	2	124	56
XC3S200[2]	200K	4,320	24	20	480	30K	216K	12	4	173	76
XC3S400[2]	400K	8,064	32	28	896	56K	288K	16	4	264	116
XC3S1000[2]	1M	17,280	48	40	1,920	120K	432K	24	4	391	175
XC3S1500	1.5M	29,952	64	52	3,328	208K	576K	32	4	487	221
XC3S2000	2M	46,080	80	64	5,120	320K	720K	40	4	565	270
XC3S4000	4M	62,208	96	72	6,912	432K	1,728K	96	4	633	300
XC3S5000	5M	74,880	104	80	8,320	520K	1,872K	104	4	633	300

Πίνακας 2-1. Χαρακτηριστικά των FPGA της οικογένειας Spartan-3 (8)

3.1. Εισαγωγή

Σε αυτό το κεφάλαιο γίνεται μια περιγραφή στην αρχιτεκτονική που χρησιμοποιούν οι επεξεργαστές (RISC και CISC), στους επεξεργαστές των ενσωματωμένων συστημάτων. Επιπλέον γίνεται μια ανάλυση στους Soft-Core επεξεργαστές οι οποίοι είναι σχεδιασμένοι για ενσωματωμένα συστήματα και προσαρμόζονται σε FPGA και ASIC κυκλώματα. Επίσης γίνεται μια αναφορά στον επεξεργαστή SPARC του οποίου η αρχιτεκτονική χρησιμοποιείται στο σύστημα και τέλος γίνεται μια περιγραφή του LEON 2, ο οποίος είναι και ο Soft-Core επεξεργαστής του συστήματος και είναι βασισμένος στην αρχιτεκτονική του SPARC Version 8.

3.2. Αρχιτεκτονική RISC και CISC

Στα τέλη της δεκαετίας του 1970, γίνονταν πολλοί πειραματισμοί με πολύ σύνθετες εντολές, οι οποίες ήταν εφικτές με τον ερμηνευτή. Οι σχεδιαστές προσπαθούσαν να γεφυρώσουν το “εννοιολογικό χάσμα” ανάμεσα στο τί μπορούσαν να κάνουν οι μηχανές και στο τί απαιτούσαν οι γλώσσες προγραμματισμού υψηλού επιπέδου. Σχεδόν κανείς δε σκεπτόταν το σχεδιασμό απλούστερων μηχανών, όπως ακριβώς σήμερα δε γίνεται και πολλή έρευνα για το σχεδιασμό λιγότερο ισχυρών λειτουργικών συστημάτων, δικτύων, επεξεργαστών κειμένου, κ.τ.λ.

Μια ομάδα στην IBM με επικεφαλή τον John Cocke πήγε αντίθετα στο ρεύμα, και προσπάθησε να ενσωματώσει κάποιες από τις ιδέες του Seymour Cray σε ένα μίνι υπολογιστή υψηλής απόδοσης. Η εργασία αυτή οδήγησε σε έναν πειραματικό μίνι υπολογιστή, που ονομάστηκε 801. Αν και η IBM ποτέ δεν κυκλοφόρησε στην αγορά αυτήν τη μηχανή, και τα αποτελέσματα δε δημοσιοποιήθηκαν παρά μόνο μετά από χρόνια, η ιδέα διέρρευσε, και άλλα άτομα άρχισαν να ερευνούν παρόμοιες αρχιτεκτονικές.

Το 1980, μια ομάδα στο Berkeley, με επικεφαλής τους David Patterson και Carlo Sequin, άρχισε να σχεδιάζει chip CPU βασισμένα στην τεχνολογία ολοκλήρωσης πολύ μεγάλης κλίμακας (VLSI) τα οποία δε χρησιμοποιούσαν ερμηνεία. Επινόησαν τον όρο RISC γι' αυτήν την ιδέα, και ονόμασαν RISC I το chip CPU που κατασκεύασαν, ενώ σύντομα ακολούθησε και το RISC II. Λίγο αργότερα, το 1981, στην απέναντι όχθη του κόλπου του San Francisco, στο Stanford, ο John Hennessy σχεδίασε και κατασκεύασε ένα κάπως διαφορετικό chip, που ονομάστηκε MIPS. Αυτά τα chip εξελίχθηκαν σε σημαντικά εμπορικά προϊόντα, τις μηχανές SPARC και MIPS, αντίστοιχα.

Αυτοί οι νέοι επεξεργαστές ήταν πολύ διαφορετικοί από τους επεξεργαστές που κυκλοφορούσαν στην αγορά εκείνη την εποχή. Αφού οι νέες CPU δε χρειαζόταν να είναι συμβατές προς τα πίσω με άλλα υπάρχοντα προϊόντα, οι σχεδιαστές τους ήταν ελεύθεροι να διαλέξουν νέα σύνολα εντολών, τα οποία θα μεγιστοποιούσαν τη συνολική απόδοση του συστήματος. Ενώ αρχικά η έμφαση δινόταν στις απλές εντολές που μπορούσαν να εκτελούνται γρήγορα, σύντομα διαπιστώθηκε ότι το κλειδί για την καλή απόδοση ήταν οι εντολές να μπορούν να υποβάλλονται (δηλαδή να ξεκινούν) γρήγορα. Ο χρόνος που χρειαζόταν πραγματικά η εντολή είχε μικρότερη σημασία από το πόσες εντολές μπορούσαν να ξεκινήσουν ανά δευτερόλεπτο.

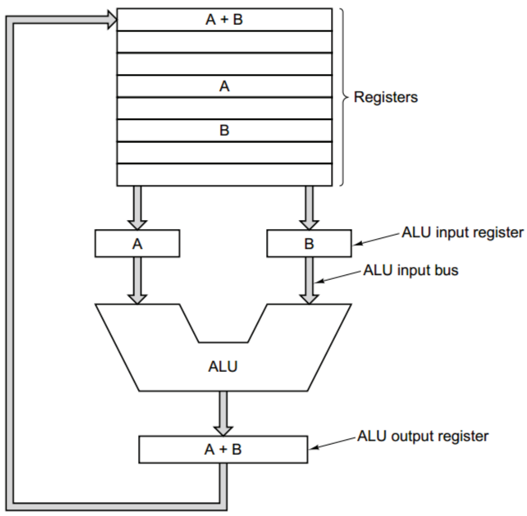
Τον καιρό που σχεδιάστηκαν για πρώτη φορά αυτοί οι απλοί επεξεργαστές, το χαρακτηριστικό που τράβηξε την προσοχή όλων ήταν ο σχετικά μικρός αριθμός εντολών που διέθεταν (κατά κανόνα γύρω στις 50). Ο αριθμός αυτός ήταν πολύ μικρότερος από τις 200 έως τις 300 εντολές των καθιερωμένων υπολογιστών, όπως ο VAX της DEC και τα κεντρικά συστήματα (mainframes) της IBM. Μάλιστα, το ακρώνυμο RISC σημαίνει Reduced Instruction Set Computer (υπολογιστής περιορισμένου συνόλου εντολών), σε αντιπαράθεση με το CISC που σημαίνει Complex Instruction Set Computer (υπολογιστής σύνθετου συνόλου εντολών – μια κάπως συγκαλυμμένη αναφορά στον VAX, που κυριαρχούσε στα τμήματα επιστήμης υπολογιστών των πανεπιστημίων εκείνη την εποχή). Σήμερα, ελάχιστοι είναι εκείνοι που θεωρούν ότι το μέγεθος του συνόλου εντολών είναι κάτι σημαντικό, αλλά το όνομα έμεινε.

Για να μην πολυλογούμε, ξεκίνησε ένας μεγάλος "θρησκευτικός πόλεμος", με τους οπαδούς του RISC να επιτίθενται στην κατεστημένη τάξη πραγμάτων (VAX, Intel, και μεγάλα κεντρικά συστήματα της IBM). Υποστήριξαν ότι ο καλύτερος τρόπος για να σχεδιαστεί ένας υπολογιστής, είναι να υπάρχει ένα μικρό σύνολο απλών εντολών που να εκτελούνται μέσα σε έναν κύκλο διαδρομής δεδομένων[4] της Εικόνας 3-1, και συγκεκριμένα, να προσκομίζονται δυο καταχωρητές, να συνδυάζονται με κάποιο τρόπο (π.χ. να γίνεται μια πρόσθεση ή μια λογική πράξη AND μεταξύ τους), και να αποθηκεύεται το αποτέλεσμα πάλι σε έναν καταχωρητή. Το επιχείρημά τους ήταν ότι, ακόμα και αν μια μηχανή RISC χρειάζεται τέσσερις ή πέντε εντολές για να κάνει κάτι που μια μηχανή CISC κάνει με μια εντολή, αν οι εντολές είναι 10 φορές πιο γρήγορες (επειδή δεν είναι ερμηνευόμενες), η μηχανή RISC είναι ο νικητής. Αξίζει επίσης να αναφέρουμε ότι, εκείνη την εποχή, η ταχύτητα της κύριας μνήμης είχε φτάσει την ταχύτητα της κύριας μνήμης ελέγχου "μόνο για ανάγνωση", γι' αυτό η επιβάρυνση λόγω της ερμηνείας είχε αυξηθεί σημαντικά, πράγμα που ευνοούσε πολύ τις μηχανές RISC.

Θα περίμενε κανείς ότι, με δεδομένα τα πλεονεκτήματα απόδοσης της τεχνολογίας RISC, οι μηχανές RISC (όπως ο επεξεργαστής Alpha της DEC) θα σάρωναν τις μηχανές CISC (όπως ο Pentium της Intel) στην αγορά. Τίποτα τέτοιο δεν έγινε. Γιατί;

Πρώτον, υπάρχει το ζήτημα της συμβατότητας προς τα πίσω, και των εκατομμυρίων δολαρίων που είχαν επενδύσει οι εταιρείες σε λογισμικό για τη σειρά επεξεργαστών της Intel. Δεύτερον, η Intel μπόρεσε κατά απροσδόκητο τρόπο να χρησιμοποιήσει τις ίδιες ιδέες ακόμα και σε μια αρχιτεκτονική CISC. Από τον επεξεργαστή 486 και μετά, οι μικροεπεξεργαστές της Intel έχουν έναν πυρήνα RISC, ο οποίος εκτελεί τις απλούστερες εντολές (που είναι κατά κανόνα οι πιο συνηθισμένες) μέσα σε έναν κύκλο της διαδρομής δεδομένων, ενώ ερμηνεύει τις πιο σύνθετες εντολές με το συνηθισμένο τρόπο μηχανών CISC. Το τελικό αποτέλεσμα είναι ότι οι πιο συνηθισμένες εντολές είναι γρήγορες, και οι λιγότερο συνηθισμένες εντολές είναι αργές. Αν και αυτή η υβριδική μέθοδος δεν είναι τόσο γρήγορη όσο θα ήταν ένας γνήσιος σχεδιασμός RISC, δίνει ανταγωνιστική συνολική απόδοση, ενώ ταυτόχρονα επιτρέπει να εκτελείται το παλιό λογισμικό χωρίς καμία τροποποίηση. [6]

[4] Διαδρομή δεδομένων σε μια τυπική CPU που χρησιμοποιεί αρχιτεκτονική Von Neumann λέγεται το τμήμα της CPU, το οποίο αποτελείται από τους καταχωρητές, την αριθμητική λογική μονάδα και πολλούς διαύλους που συνδέουν τα διάφορα μέρη.



Εικόνα 3-1. Η διαδρομή δεδομένων μιας τυπικής μηχανής von Neumann [6]

3.2.1. Χαρακτηριστικά της αρχιτεκτονικής RISC

Όπως αναφέραμε στο προηγούμενο κεφάλαιο η IBM ήταν η πρώτη εταιρία η οποία όρισε την αρχιτεκτονική RISC στα τέλη της δεκαετίας του 1970. Η έρευνα που έγινε για την συγκεκριμένη αρχιτεκτονική αναπτύχθηκε σε μεγαλύτερο επίπεδο από τα πανεπιστήμια του Berkley και του Stanford δίνοντας τα πρώτα αρχιτεκτονικά μοντέλα. Η αρχιτεκτονική RISC μπορεί να περιγραφεί σαν φιλοσοφία τριών βασικών δογμάτων:

1. Όλες οι εντολές θα εκτελούνται σε έναν κύκλο ρολογιού. Αυτό έχει ως αποτέλεσμα ότι κάθε κώδικας λειτουργίας (operational code – op code) θα πρέπει να είναι συγκεκριμένου μεγέθους και να είναι ίσος ή μικρότερος από το μέγεθος του διαύλου των δεδομένων, επιπλέων τελεστές (operands) δεν υποστηρίζονται και οι εντολές αποκωδικοποίησης πρέπει να είναι απλές ώστε να αποφεύγονται οι καθυστερήσεις.
2. Η μνήμη θα είναι προσβάσιμη μόνο από τις εντολές "φόρτωσε" (load) και "αποθήκευσε" (store). Αν μια εντολή έχει αλληλεπίδραση απευθείας με τη μνήμη αυτό έχει ως αποτέλεσμα να χρειάζονται πολλοί κύκλοι ρολογιού ώστε να εκτελεστεί η συγκεκριμένη εντολή. Αντίθετα, με έναν επεξεργαστή που χρησιμοποιεί την αρχιτεκτονική RISC τα δεδομένα είναι φορτωμένα σε έναν καταχωρητή. Οι εντολές έχουν αλληλεπίδραση μόνο με τον καταχωρητή και τέλος καταγράφονται στην κύρια μνήμη. Αυτή η ακολουθία απαιτεί το λιγότερο να εκτελεστούν τρείς εντολές. Στα

συστήματα που είναι βασισμένα σε καταχωρητές απαιτείται μεγάλο ποσοστό καταχωρητών, ώστε να διατηρηθεί η απόδοση του συστήματος.

3. Όλες οι μονάδες εκτέλεσης[5] (execution unit) θα είναι καλωδιωμένες χωρίς την υλοποίηση microcoding[6].

Τα δύο μοντέλα για την αρχιτεκτονική RISC που αναπτυχθήκαν από τα δύο πανεπιστήμια σχημάτισαν την βάση σχεδόν όλων των επεξεργαστών που χρησιμοποιούνται σήμερα. Η βασική διαφορά που έχουν είναι το σύνολο των καταχωρητών που χρησιμοποιούνται και η χρήση τους. Και τα δύο μοντέλα χρησιμοποιούν την αρχιτεκτονική Harvard χρησιμοποιώντας διαφορετικό δίαυλο για τα δεδομένα και διαφορετικό για τις εντολές όπως αναφέραμε στο πρώτο κεφάλαιο. [7]

3.3. Πυρήνες επεξεργαστών για ενσωματωμένα συστήματα

Οι CPU για τα ενσωματωμένα συστήματα διατίθενται σε ένα εύρος μεγεθών για διαφορετικές εφαρμογές. Μερικές είναι μικροεπεξεργαστές (microprocessors) ενός chip, αποτελούμενοι από μια CPU που είναι μόνη της στη συσκευασία (package). Οι περισσότερες CPU που χρησιμοποιούνται σε PC γενικής χρήσης διατίθενται επίσης και σε εκδόσεις κατάλληλες για ενσωματωμένες εφαρμογές. Παραδείγματα αποτελούν οι CPU της οικογένειας Pentium από την Intel και PowerPC από την Freescale Semiconductor. Άλλοι μικροεπεξεργαστές σχεδιάζονται ειδικά για ενσωματωμένες εφαρμογές. Και στις δύο περιπτώσεις, χρειάζεται να παρέχουμε μνήμη και ελεγκτές εισόδου/εξόδου ως ξεχωριστά chip σε μια πλακέτα τυπωμένου κυκλώματος (printed circuit board – PCB). Αντίθετα, οι μικροελεγκτές (microcontrollers) ενός chip, περιλαμβάνουν μια CPU, μνήμη εντολών και δεδομένων, και ελεγκτές εισόδου/εξόδου όλα στην ίδια συσκευασία. Πολλοί προμηθευτές μικροελεγκτών παρέχουν μια οικογένεια από chip, με την ίδια CPU, αλλά με ποικιλία στην ποσότητα της μνήμης και στην επιλογή των ελεγκτών εισόδου/εξόδου. Σε μερικές οικογένειες μικροελεγκτών, οι CPU είναι σχετικά απλές, λειτουργούν μόνο σε δεδομένα των 8 bit ή των 16 bit, με σχετικά χαμηλή απόδοση. Άλλες οικογένειες διαθέτουν περισσότερο σύνθετες CPU που λειτουργούν σε δεδομένα που φθάνουν σε μήκος μέχρι τα 32 bit και 64 bit.

[5] Μονάδες εκτέλεσης ονομάζονται τα τμήματα ενός επεξεργαστή όπου εκτελούν διάφορες λειτουργίες και υπολογισμούς.
[6] Microcode είναι οι οδηγίες που είναι αποθηκευμένες μόνιμα στην μνήμη για ανάγνωση.

Μια εναλλακτική λύση στη χρήση ενός μικροεπεξεργαστή ή ενός μικροελεγκτή προκαθορισμένης λειτουργίας είναι να συμπεριληφθεί μια CPU σε ένα συστατικό στοιχείο FPGA. Αυτό έχει το πλεονέκτημα ότι οι ελεγκτές εισόδου/εξόδου μπορούν να προσαρμοστούν στις ανάγκες μιας εφαρμογής, αλλά να βρίσκονται ακόμα στην ίδια συσκευασία με τη CPU. Η CPU στην FPGA μπορεί να υλοποιηθεί ως ένα μπλοκ προκαθορισμένης λειτουργίας ενσωματωμένο μέσα στην προγραμματιζόμενη δομή. Οι FPGA Virtex-II Pro και Virtex 4 από την Xilinx ακολουθούν αυτήν τη προσέγγιση, και περιλαμβάνουν έναν ή περισσότερους πυρήνες επεξεργαστών PowerPC. Εναλλακτικά, η CPU μπορεί να υλοποιηθεί ως ένας εύπλαστος πυρήνας (Soft-Core) χρησιμοποιώντας τους προγραμματιζόμενους πόρους της FPGA. Οι προμηθευτές FPGA παρέχουν σχεδιάσεις εύπλαστων πυρήνων επεξεργαστών που οι χρήστες μπορούν να συμπεριλάβουν ως τμήμα του συστήματός τους. Ως παράδειγμα έχουμε τους πυρήνες[7] MicroBlaze από την Xilinx, ο πυρήνας Nios-II από την Altera, και ο πυρήνας ARM από την Actel. Όλες αυτές είναι CPU σχετικά υψηλής απόδοσης που λειτουργούν σε δεδομένα με μήκος μέχρι 32 η 64 bit. Για πιο απλές σχεδιάσεις, ένας μικρότερος εύπλαστος πυρήνας που λειτουργεί σε δεδομένα 8 bit μπορεί να είναι αρκετός. Θα καταλάμβανε λιγότερους πόρους της FPGA, και θα μπορούσε να χωρέσει σε ένα μικρότερο και φθηνότερο συστατικό στοιχείο FPGA. Ο εύπλαστος πυρήνας PicoBlaze της Xilinx είναι ένα παράδειγμα, όπως και ο πυρήνας Gumnut.

Εάν η σχεδίασή μας υλοποιείται σε ένα ASIC, μπορούμε επίσης να συμπεριλάβουμε μια CPU και προσαρμοσμένη μνήμη και ελεγκτές εισόδου/εξόδου. Αρκετοί προμηθευτές παρέχουν σχεδιάσεις πυρήνων επεξεργαστών που μπορούν να συμπεριληφθούν ως μπλοκ σε ASIC. Μεταξύ των πιο ευρέως χρησιμοποιούμενων είναι οι πυρήνες ARM της ARM Ltd, οι πυρήνες PowerPC από την IBM, και οι πυρήνες MIPS από την MIPS Technologies. Με δεδομένο ότι μπορούμε να προσαρμόσουμε τη σχεδίαση σε ένα ASIC, μας δίνεται η ευκαιρία να προσαρμόσουμε και την ίδια την CPU. Η Tensilica Inc. είναι ένας προμηθευτής που παρέχει μια προσαρμοσμένη CPU που βασίζεται στις απαιτήσεις του προγράμματος που θα εκτελεστεί. Η προσέγγισή τους αφορά την ανάλυση του προγράμματος και την ένταξη μόνο των χαρακτηριστικών της CPU που απαιτούνται για την εκτέλεση αυτού του προγράμματος.

Τέλος υπάρχουν οι επεξεργαστές ψηφιακού σήματος (DSPs). Αυτοί αποτελούν εξειδικευμένα επεξεργαστικά στοιχεία που είναι βελτιστοποιημένα για τα είδη των λειτουργιών που αφορούν το χειρισμό ψηφιοποιημένων σημάτων, όπως ήχο, βίντεο ή άλλες

[7] Οι πυρήνες που αναφέρουμε σε αυτό το υποκεφάλαιο είναι εμπορικοί και όχι ανοιχτού κώδικα όπως ο πυρήνας LEON 2 που χρησιμοποιήθηκε στο σύστημα.

ροές δεδομένων από αισθητήρες. Αρκετές εφαρμογές επεξεργασίας σήματος απαιτούν οι αριθμητικές πράξεις σταθερής υποδιαστολής ή κινητής υποδιαστολής για να εκτελούνται σε υψηλό ρυθμό και σε δεδομένα μεγάλου όγκου. Μια συνηθισμένη CPU δεν θα μπορούσε να ικανοποιήσει τις απαιτήσεις σε απόδοση. Παρ' όλα αυτά, τέτοιες εφαρμογές συχνά χρειάζονται μια συμβατική CPU σε ετερογενή συστήματα πολυεπεξεργαστών (multiprocessor). Τα σύγχρονα κινητά τηλέφωνα αποτελούν καλά παραδείγματα αυτής της περίπτωσης. Η επέκταση μιας συμβατικής CPU με επιπλέον υλικό και εντολές για ψηφιακή επεξεργασία σήματος αποτελεί μια άλλη προσέγγιση για την παροχή λειτουργικότητας DSP. Μερικοί πυρήνες επεξεργαστών της Tensilica μπορούν να προσαρμοστούν στις ανάγκες της εφαρμογής με παρόμοιο τρόπο. [5]

3.4. Τί είναι οι Soft-Core επεξεργαστές

Ένας softcore επεξεργαστής είναι ένας μικροεπεξεργαστής, του οποίου η αρχιτεκτονική και η συμπεριφορά είναι εξ' ολοκλήρου σχεδιασμένες με τη χρήση μιας γλώσσας περιγραφής υλικού (HDL). Μπορεί να συντεθεί για οποιοδήποτε ολοκληρωμένο κύκλωμα ειδικού σκοπού (ASIC) ή για μια προγραμματιζόμενη στο πεδίο συστοιχία πυλών (FPGA). Η χρήση των Soft-Core επεξεργαστών έχει πολλά πλεονεκτήματα για τους σχεδιαστές ενσωματωμένων συστημάτων. Αρχικά ένας Soft-Core επεξεργαστής προσφέρει προσαρμοστικότητα και μπορεί να παραμετροποιηθεί για εφαρμογές ειδικού σκοπού με μια σχετική ευκολία. Δεύτερον από τη στιγμή που οι Soft-Core επεξεργαστές είναι ανεξάρτητης τεχνολογίας, μπορούν να συντεθούν για οποιοδήποτε κύκλωμα ASIC ή FPGA. Γι' αυτόν το λόγο έχουν μεγαλύτερη αντοχή στο χρόνο συγκριτικά με Hard-Core επεξεργαστές. Τέλος από τη στιγμή που η αρχιτεκτονική και η συμπεριφορά ενός Soft-Core επεξεργαστή περιγράφονται σε ένα υψηλότερο επίπεδο με τη χρήση κάποιας γλώσσας περιγραφής υλικού, είναι πολύ πιο εύκολη η κατανόηση του συνολικού σχεδιασμού. [8]

3.5. Ο επεξεργαστής SPARC

Στη δεκαετία του 1970, το UNIX ήταν δημοφιλές στα πανεπιστήμια, αλλά κανένας προσωπικός υπολογιστής δε μπορούσε να χρησιμοποιεί UNIX. Έτσι, οι φίλοι του UNIX ήταν υποχρεωμένοι να χρησιμοποιούν (συχνά υπερφορτωμένους) χρονομετρικούς μίνι υπολογιστές, όπως ο PDP-11 και ο VAX. Το 1981, ένας γερμανός μεταπτυχιακός φοιτητής του Stanford, ο Andy Bechtolsheim, που είχε απηυδήσει να πηγαίνει στο υπολογιστικό κέντρο για να χρησιμοποιήσει το UNIX, αποφάσισε να λύσει το πρόβλημα κατασκευάζοντας ο ίδιος έναν προσωπικό σταθμό εργασίας UNIX από έτοιμα διαθέσιμα εξαρτήματα. Τον ονόμασε Sun-1.

Σύντομα, ο Bechtolsheim προσέλκυσε το ενδιαφέρων του Vinod Khosla, ενός 27-χρονού Ινδού. Ο Khosla έπεισε τον Bechtolsheim να ιδρύσουν μια εταιρία που να κατασκευάζει και να πουλάει σταθμούς εργασίας Sun. Έπειτα, ο Khosla προσέλαβε τον Scott McNealy, έναν άλλο μεταπτυχιακό φοιτητή του Stanford, για να διευθύνει την παραγωγή. Για το γράψιμο του λογισμικού, προσέλαβαν τον Bill Joy, το βασικό σχεδιαστή του Berkeley UNIX. Αυτοί οι τέσσερις ίδρυσαν την εταιρία Sun Microsystems το 1982.

Το πρώτο προϊόν της Sun, ο υπολογιστής Sun-1, ο οποίος βασιζόταν σε ένα μικροεπεξεργαστή 68020 της Motorola, έγινε αμέσως επιτυχία. Το ίδιο συνέβη και με τις επόμενες μηχανές, Sun-2 και Sun-3, που χρησιμοποιούσαν επίσης μικροεπεξεργαστές της Motorola. Σε σύγκριση με τους άλλους προσωπικούς υπολογιστές εκείνης της εποχής, οι μηχανές αυτές ήταν πολύ πιο ισχυρές (γι’ αυτό και χαρακτηρίζονταν ως “σταθμοί εργασίας”), και ήταν από την αρχή σχεδιασμένες να δουλεύουν σε δίκτυο. Κάθε σταθμός εργασίας Sun ήταν εφοδιασμένος με μια σύνδεση δικτύου Ethernet και με λογισμικό TCP/IP για σύνδεση με το ARPANET, τον πρόδρομο του Internet.

Το 1987, η Sun, που σήμερα κάνει πωλήσεις συστημάτων μισού δισεκατομμυρίου δολαρίων το χρόνο, αποφάσισε να σχεδιάσει ένα δικό της μικροεπεξεργαστή, με βάση έναν επαναστατικό νέο σχεδιασμό (το RISC II) που προερχόταν από το Πανεπιστήμιο της California στο Berkley. Αυτός ο μικροεπεξεργαστής, που ονομάστηκε SPARC έγινε η βάση για το σταθμό εργασίας Sun-4. Σύντομα, όλα τα προϊόντα της Sun χρησιμοποιούσαν το SPARC.

Αντίθετα από πολλές άλλες εταιρείες υπολογιστών, η Sun αποφάσισε να μην κατασκευάζει η ίδια τα chip του μικροεπεξεργαστή SPARC. Έδωσε άδεια σε πολλούς διαφορετικούς κατασκευαστές ημιαγωγών να παράγουν μικροεπεξεργαστές SPARC, με την ελπίδα ότι ο ανταγωνισμός μεταξύ τους θα ωθούσε την απόδοση προς τα πάνω και τις τιμές προς τα κάτω. Οι κατασκευαστές αυτοί έθεσαν σε κυκλοφορία πολλά διαφορετικά chip, τα οποία βασίζονταν σε διαφορετικές τεχνολογίες, λειτουργούσαν σε διαφορετικές ταχύτητες ρολογιού, και είχαν διαφορετικές τιμές. Σε αυτά τα chip περιλαμβάνονταν το MicroSPARC, το HyperSPARC, το SuperSPARC, και το TurboSPARC. Αν και όλοι αυτοί οι μικροεπεξεργαστές διέφεραν σε δευτερεύοντα σημεία, ήταν όλοι συμβατοί σε επίπεδο δυαδικού κώδικα, και εκτελούσαν τα ίδια προγράμματα χρήστη χωρίς καμία τροποποίηση.

Η Sun πάντα ήθελε να έχει το SPARC ανοιχτή αρχιτεκτονική, με πολλούς προμηθευτές εξαρτημάτων και συστημάτων, ώστε να οικοδομήσει μια βιομηχανία ανταγωνιστική μέσα σε έναν κόσμο προσωπικών υπολογιστών στον οποίο είχαν ήδη κυριαρχήσει οι μικροεπεξεργαστές της Intel. Για να κερδίσει την εμπιστοσύνη των εταιριών που ενδιαφερόντουσαν για το SPARC με την προϋπόθεση ότι δεν ήθελαν να επενδύσουν σε ένα προϊόν που ελέγχεται από έναν ανταγωνιστή, η Sun δημιούργησε μια σύμπραξη εταιρειών, τη SPARC International, για να διαχειρίζεται την ανάπτυξη των μελλοντικών εκδόσεων της αρχιτεκτονικής SPARC.

Ο πρώτος SPARC ήταν μια μηχανή των 32bit, που δούλευε στα 36 MHz. Η CPU, που ονομαζόταν IU (Integer Unit – μονάδα ακεραίων), ήταν λιτή και φτωχική. Είχε μόνο τρείς κύριες μορφές εντολών, και 55 εντολές συνολικά. Μια μονάδα κινητής υποδιαστολής προσέθετε άλλες 14 εντολές. Αυτό είναι ενδιαφέρον να αντιπαρατεθεί με τη σειρά επεξεργαστών της Intel, που ξεκίνησε με chip των 8 και των 16 bit (8088, 8086, 80286), και τελικά πέρασε στα chip των 32 bit με τον 80386.

Η πρώτη ρήξη του SPARC με το παρελθόν έγινε το 1995, με την εμφάνιση τις ένατης έκδοσης (Version 9) της αρχιτεκτονικής SPARC, η οποία ήταν μια πλήρως 64 bit αρχιτεκτονική, με διευθύνσεις και καταχωρητές των 64 bit. Ο πρώτος σταθμός εργασίας Sun που υλοποίησε την αρχιτεκτονική V9 ήταν ο UltraSparc I, που κυκλοφόρησε το 1995 (Tremblay και O'Connor, 1996). Αν και ο UltraSPARC ήταν μια μηχανή των 64 bit, ήταν εντελώς συμβατός σε επίπεδο δυαδικού κώδικα με τους υπάρχοντες SPARC των 32 bit. [6] Στη συνέχεια αυτής της πτυχιακής εργασίας γίνεται αναφορά μόνο στον SPARC (Version 8) μιας και η αρχιτεκτονική του Soft-Core επεξεργαστή LEON 2 είναι βασισμένη στην αρχιτεκτονική του SPARC V8.

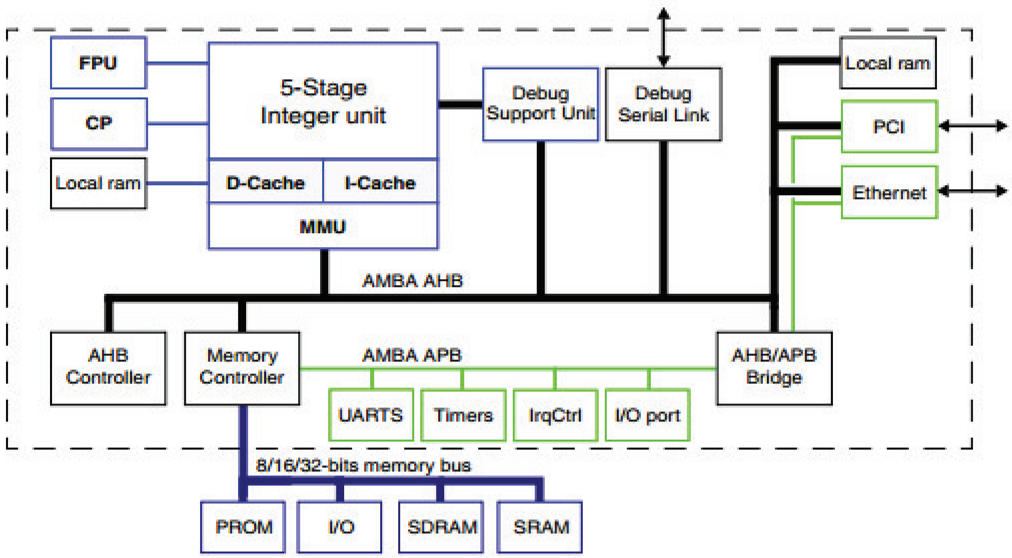
3.6. Χαρακτηριστικά του LEON 2

Στο σύστημα του πειράματος χρησιμοποιήθηκε ο Soft-Core επεξεργαστής LEON 2. Όπως προαναφέρεται παραπάνω ο LEON 2 είναι βασισμένος στην αρχιτεκτονική του επεξεργαστή SPARC Version 8. Είναι επεξεργαστής 32 bit και είναι σχεδιασμένος από τη γλώσσα υλικού VHDL. Σχεδιάστηκε για εφαρμογές, οι οποίες εκτελούνται σε ενσωματωμένα συστήματα και έχει τα εξής χαρακτηριστικά:

- Διαφορετικό δίαυλο επικοινωνίας για τα δεδομένα και διαφορετικό δίαυλο επικοινωνίας για τις εντολές ακολουθώντας με αυτόν τον τρόπο την αρχιτεκτονική Harvard.
- Διαθέτει πολλαπλασιαστές και διαιρέτες.
- Διαθέτει ελεγκτή διακοπών (Interrupt controller). Ο ελεγκτής διακοπών διαχειρίζεται συνολικά 15 διακοπές, οι οποίες μπορούν να παράγονται από εσωτερικές ή εξωτερικές πηγές. Κάθε μια από τις διακοπές μπορεί να προγραμματιστεί να λειτουργεί σε ένα από τα δυο επίπεδα προτεραιότητας. Ένας επιπλέον ελεγκτής (για 32 επιπρόσθετες διακοπές) είναι διαθέσιμος.
- Μονάδα αποσφαλμάτωσης (Debug support unit- DSU). Η μονάδα αποσφαλμάτωσης επιτρέπει την εισαγωγή breakpoints και watchpoints, καθώς και πρόσβαση στους καταχωρητές του chip από έναν απομακρυσμένο αποσφαλματωτή. Η μονάδα αποσφαλμάτωσης δεν έχει επίδραση στην απόδοση του συστήματος και είναι χαμηλής πολυπλοκότητας. Η επικοινωνία με έναν απομακρυσμένο αποσφαλματιστή γίνεται μέσω της σειριακής θύρας UART (RS232) ή μέσω ενός διαύλου υψηλής AHB (π.χ. PCI).
- Χρονομετρητές (Timers) των 24 bit, οι οποίοι μπορούν να λειτουργούν περιοδικά ή να βρίσκονται σε κατάσταση “μιας στιγμής”.
- Δυο σειριακές θύρες UART. Παρέχονται δυο σειριακές θύρες UART των 8 bit. Το baud-rate μπορεί να προγραμματιστεί ξεχωριστά και τα δεδομένα στέλνονται σε πλαίσια (frames) των 8 bit με 1 bit ισοτιμίας (parity) το οποίο υποδηλώνει τον τερματισμό της επικοινωνίας.
- Watch dog χρονομετρητής. Χρησιμοποιείται ένας Watch dog χρονομετρητής των 24 bit, ο οποίος μόλις φτάσει το 0 παράγεται ένα WDOG σήμα, το οποίο χρησιμοποιείται για επαναφορά του συστήματος (system reset).

- Διαθέτει παράλληλες θύρες εισόδου/εξόδου. 16 bit είναι διαθέσιμα σε περίπτωση που ο δίαυλος μνήμης είναι προγραμματισμένος να λειτουργεί σε 8 ή 16 bit.
- Διαθέτει μονάδα διαχείρισης μνήμης (MMU). Η μονάδα διαχείρισης μνήμης υλοποιεί την MMU που βασίζεται στον SPARC Version 8 και επιτρέπει την χρήση λειτουργικών συστημάτων όπως Linux ή Solaris.
- Διαθέτει διεπαφή μνήμης. Η διεπαφή μνήμης παρέχει άμεση διεπαφή με την προγραμματιζόμενη ROM (PROM), με τις διευθύνσεις των θυρών εισόδου/εξόδου, με την στατική RAM (SPAM) και με τη δυναμική RAM (DRAM). Η περιοχή της μνήμης μπορεί να προγραμματιστεί να χρησιμοποιεί μέγεθος δεδομένων των 8, 16 ή 32 bit.
- Ethernet MAC. Μπορεί να ενεργοποιηθεί η λειτουργία Ethernet 10/100Mbit MAC. Η MAC είναι βασισμένη στον πυρήνα Ethernet MAC από το OpenCores, με έναν επιπλέον AHB δίαυλο.

Τέλος πρέπει να αναφερθεί πως ο LEON 2 είναι πλήρως παραμετροποιήσιμος και μπορεί να υλοποιηθεί είτε σε μια FPGAκ, είτε σε ένα ASIC. Στην Εικόνα 3-2 παρουσιάζεται η αρχιτεκτονική του LEON 2. (9)



Εικόνα 3-2. Η αρχιτεκτονική του LEON 2 (9)

Εργαλεία σχεδιασμού του συστήματος

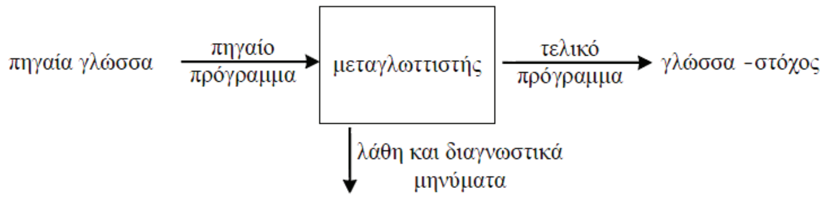
4.1. Εισαγωγή

Σε αυτό το κεφάλαιο περιγράφονται τα εργαλεία τα οποία χρησιμοποιούνται ώστε να προγραμματιστεί η FPGA του συστήματος. Αρχικά περιγράφονται οι λειτουργίες που έχει ένας μεταγλωττιστής σε ένα σύστημα. Έπειτα γίνεται μια αναφορά στην διαμεταγλώττιση και στις λειτουργίες που προσφέρει, καθώς επίσης και στον μεταγλωττιστή GCC ο οποίος χρησιμοποιείται για την διαμεταγλώττιση του πηγαίου κώδικα. Τέλος γίνεται μια αναφορά στην γλώσσα προγραμματισμού C η οποία χρησιμοποιείται από το σύστημα, και στο τερματικό Cygwin το οποίο έχει βασικό ρόλο στον προγραμματισμό του συστήματος.

4.2. Λειτουργία ενός μεταγλωττιστή (Compiler)

Μεταγλωττιστής είναι το λογισμικό, που ως σκοπό έχει τη μετάφραση ενός προγράμματος, από μία γλώσσα σε μία άλλη. Όταν ένα πρόγραμμα βρίσκεται στην αρχική του μορφή ονομάζεται πηγαίος κώδικας και είναι αναγνώσιμο από τον προγραμματιστή του. Ο μεταγλωττιστής όμως αναλαμβάνει να τον μετατρέψει τον πηγαίο κώδικα σε γλώσσα μηχανής, ώστε να μπορεί να υλοποιηθεί η λειτουργία που παρέχει το συγκεκριμένο πρόγραμμα από το υπολογιστικό σύστημα. Η κατανόηση και η υλοποίηση των τεχνικών επεξεργασίας που χρησιμοποιούν οι μεταγλωττιστές δεν είναι μία απλή υπόθεση. Οι ίδιες τεχνικές βρίσκουν εφαρμογή όχι μόνο στην κατασκευή γλωσσών προγραμματισμού, αλλά και στην ανάπτυξη προγραμμάτων αλληλεπίδρασης, όπως για παράδειγμα διερμηνευτές εντολών για την αυτοματοποίηση σεναρίων εργασιών και λογισμικό επισκόπησης κειμένου σε κάποια γλώσσα σήμανσης (π.χ. HTML). Στην Εικόνα 4-1 αναλύεται ο βασικός τρόπος λειτουργίας ενός μεταγλωττιστή.

Κάθε μεταγλωττιστής απαρτίζεται από τα τμήματα αρχικής επεξεργασίας (front end), όπου στην αρχική επεξεργασία γίνεται ανάλυση και έλεγχος του πηγαίου κώδικα για συντακτικά, σημασιολογικά κ.α. λάθη δίνοντας ως έξοδο στον προγραμματιστή ένα μήνυμα το οποίο αναφέρει το λάθος (σε περίπτωση λάθους), και την τελική επεξεργασία (back end), όπου γίνεται η μετάφραση του πηγαίου κώδικα σε γλώσσα μηχανής. Η διαδικασία της μεταγλώττισης απαρτίζεται από έναν αριθμό φάσεων επεξεργασίας. Οι φάσεις αυτές είναι η λεξική ανάλυση, η συντακτική ανάλυση, η σημασιολογική ανάλυση, η βελτιστοποίηση του πηγαίου προγράμματος σύνθεση του τελικού προγράμματος και η βελτιστοποίησή του. Κατά τη διάρκεια της επεξεργασίας κάθε φάσης υπάρχει αλληλεπίδραση με ένα ιδιαίτερα σημαντικό τμήμα του μεταγλωττιστή, που ονομάζεται πίνακας συμβόλων. Ο πίνακας συμβόλων καταγράφει πληροφορίες, που σχετίζονται με ονόματα, είτε αυτά είναι ονόματα συναρτήσεων, είτε ονόματα τύπων δεδομένων, μεταβλητών ή και σταθερών. Χρησιμοποιείται σχεδόν σε όλες τις φάσεις της μεταγλώττισης. Πιο συγκεκριμένα, κατά τη λεξική, τη συντακτική και τη σημασιολογική ανάλυση καταχωρούνται σε αυτόν ονόματα, η εκμετάλλευση των οποίων γίνεται κατά τη βελτιστοποίηση και τη σύνθεση του τελικού προγράμματος. Λειτουργίες όπως η εισαγωγή, η διαγραφή και η προσπέλαση δεδομένων στον πίνακα αυτό, χρειάζεται να εκτελούνται αποδοτικά, γι' αυτό και συνήθως γίνεται χρήση πίνακα Hash ή κάποιας δομής δένδρου. Η επεξεργασία του πηγαίου προγράμματος γίνεται με διαδοχικές σαρώσεις. Μετά την αρχική σάρωση, που έχει ως αποτέλεσμα τη δημιουργία του συντακτικού δέντρου ή κάποιας άλλης ενδιάμεσης αναπαράστασης, οι περιπτώσεις σάρωσης που ακολουθούν έχουν ως στόχο είτε να προσθέσουν πληροφορίες στην επιλεγείσα ενδιάμεση αναπαράσταση, είτε να μεταβάλλουν τη δομή της ή ακόμη και να συνθέσουν μία νέα αναπαράσταση. Σε κάθε σάρωση εκτελούνται πιθανώς περισσότερες της μιας φάσεις επεξεργασίας. Οι μεταγλωττιστές μονής σάρωσης (Single Pass Compilers) εκτελούν όλες τις φάσεις επεξεργασίας σε μία σάρωση του προγράμματος και έχουν ως αποτέλεσμα μικρούς χρόνους μετάφρασης, αλλά όχι και τόσο γρήγορο τελικό πρόγραμμα. Γλώσσες όπως η Pascal και η C χαρακτηρίζονται από μία τέτοια δομή, που καθιστά εφικτή την ανάπτυξη μεταγλωττιστών αυτού του τύπου. Οι μεταγλωττιστές όμως με δυνατότητες βελτιστοποιήσεων συνήθως χρησιμοποιούν περισσότερες από μια σάρωση, η πρώτη από τις οποίες περιλαμβάνει συνήθως τη λεξική και τη συντακτική ανάλυση του πηγαίου προγράμματος, η δεύτερη τη σημασιολογική ανάλυση και τη βελτιστοποίηση αυτού και η τελευταία τη δημιουργία του τελικού προγράμματος και τη βελτιστοποίησή του. [9][10]



Εικόνα 4-1. Αναπαράσταση της λειτουργίας ενός μεταγλωττιστή [9]

4.2.1. Διαμεταγλώττιση (Cross Compile)

Στις μέρες μας ο σχεδιασμός του λογισμικού για ένα σύστημα πραγματικού χρόνου γίνεται σε περιβάλλοντα τα οποία είναι σχεδιασμένα να μπορούν να κάνουν διαμεταγλώττιση (cross compiling). Σε ένα περιβάλλον όπου μπορεί να γίνει cross compiling τα εργαλεία τα οποία κάνουν αυτήν τη δουλεία βρίσκονται σε ένα υπολογιστικό σύστημα διαφορετικό από το ενσωματωμένο σύστημα για το οποίο προορίζεται το πρόγραμμα το οποίο παράγεται. Οι απαιτήσεις που έχει ένα ενσωματωμένο σύστημα συνήθως δεν είναι συμβατές με αυτές του υπολογιστικού συστήματος στο οποίο γίνεται το cross compiling. Επιπλέον το ενσωματωμένο σύστημα συνήθως είναι σχεδιασμένο για κάποιον ειδικό σκοπό και όχι για γενικό σκοπό όπως είναι ένα υπολογιστικό σύστημα. Αυτό έχει ως αποτέλεσμα ότι τα εργαλεία τα οποία χρησιμοποιούνται από κάποιον προγραμματιστή για cross compiling πρέπει να μπορούν να είναι παραμετροποιήσιμα ώστε ο προγραμματιστής να μπορεί να τα χρησιμοποιήσει για το συγκεκριμένο ενσωματωμένο σύστημα.

Το υπολογιστικό σύστημα πρέπει να μπορεί να παρέχει όλα όσα χρειάζονται ώστε το cross compiling να είναι επιτυχημένο, όπως γραφικό περιβάλλον για τη συρραφή του κώδικα, διαμεταγλωττιστές (cross compilers), αρκετή χωρητικότητα στον δίσκο κ.α.. Για αυτό το λόγο συνήθως τα υπολογιστικά συστήματα είναι σταθμοί εργασίας UNIX όπως αυτοί που παρέχονται από την Sun ή προσωπικοί υπολογιστές οι οποίοι έχουν κάποια έκδοση MS-Windows ή UNIX. Πολλές φορές το υπολογιστικό σύστημα στο οποίο γίνεται το cross compiling πρέπει να μπορεί να παρέχει την δυνατότητα στον προγραμματιστή να μπορεί να δημιουργήσει έγγραφα στα οποία να περιγράφει την λειτουργία του ενσωματωμένου συστήματος που δημιούργησε ή να δημιουργήσει κάποια παρουσίαση του συστήματος του. Όλα τα παραπάνω έχουν ως αποτέλεσμα το υπολογιστικό σύστημα να πρέπει να αποτελείται από έναν επεξεργαστή 32 ή 64 bit, από μνήμη RAM μεγάλης χωρητικότητας, από μια οθόνη, από ποντίκι, από πληκτρολόγιο, από σκληρούς δίσκους, από οπτικά μέσα όπως CD-ROM Drives και από μια κάρτα γραφικών.

Αντίθετα το ενσωματωμένο σύστημα έχει πολύ περιορισμένες δυνατότητες σε σχέση με ένα υπολογιστικό σύστημα. Το υλικό ενός ενσωματωμένου συστήματος είναι σχεδιασμένο για συγκεκριμένες λειτουργίες και απαιτήσεις και είναι βελτιστοποιημένο με τέτοιο τρόπο ώστε να μπορεί να εκτελέσει τις λειτουργίες που του έχουν ανατεθεί αποτελεσματικά. Αντίθετα από ένα υπολογιστικό σύστημα στο οποίο υπάρχουν σκληροί δίσκοι, πληκτρολόγια κ.α., ένα ενσωματωμένο σύστημα αποτελείται από αισθητήρες, relays και ίσως από κάποια μηχανικά μέρη όπως έχει αναφερθεί και στο πρώτο κεφάλαιο. (10)

4.2.2. Πώς λειτουργεί η διαμεταγλώττιση

Στο κεφάλαιο 4.2. αναφέραμε σε βάθος τον τρόπο λειτουργίας ενός μεταγλωττιστή. Σε αυτό το κεφάλαιο θα αναλύσουμε τα επιμέρους κομμάτια τα οποία χρειάζονται ώστε να έχουμε μια επιτυχημένη διαμεταγλώττιση.

Πολλά διαφορετικά κομμάτια λειτουργούν μαζί με σκοπό να παραχθεί η δυαδική ακολουθία για κάποιον συγκεκριμένο επεξεργαστή. Όταν παραχθεί η δυαδική ακολουθία τότε η διαμεταγλώττιση θεωρείται επιτυχημένη. Τα βασικά κομμάτια από τα οποία αποτελείται ένας μεταγλωττιστής είναι τα εξής:

- Αναλυτής (Parser): Ο αναλυτής μετατρέπει τον πηγαίο κώδικα στη γλώσσα assembly. Λόγω του ότι μετατρέπεται μια γλώσσα σε μια άλλη (από C σε assembly) ο αναλυτής πρέπει να γνωρίζει την assembly που χρησιμοποιεί ο επεξεργαστής για τον οποίο προορίζεται το μεταγλωττισμένο πρόγραμμα.
- Συμβολομεταφραστής (Assembler): Ο συμβολομεταφραστής μετατρέπει την γλώσσα assembly που έχει παραχθεί σε δυαδική ακολουθία την οποία εκτελεί ο επεξεργαστής του συστήματος για το οποίο προοριζόταν.
- Συνδέτης (Linker): Ο συνδέτης συνδυάζει ξεχωριστά αρχεία που έχουν παραχθεί από τον συμβολομεταφραστή σε μια εκτελέσιμη εφαρμογή. Διαφορετικά λειτουργικά συστήματα και διαφορετικοί επεξεργαστές χρησιμοποιούν διαφορετικούς μηχανισμούς και πρότυπα ενθυλάκωσης (encapsulation). Αυτά πρέπει να τα γνωρίζει ακριβώς ο συνδέτης.

- Η βασική βιβλιοθήκη της C (Standard C Library): Οι βασικές συναρτήσεις της C βρίσκονται σε μια βασική βιβλιοθήκη. Αυτή η βιβλιοθήκη χρησιμοποιείται σε συνδυασμό με τον συνδέτη και τον πηγαίο κώδικα για να παραχθεί η τελευταία μορφή του εκτελέσιμου αρχείου, υποθέτοντας πως οι συναρτήσεις από την βασική βιβλιοθήκη της C χρησιμοποιούνται.

Σε έναν μεταγλωττιστή, και συγκεκριμένα για τον μεταγλωττιστή GCC ο οποίος αναλύεται σε επόμενο κεφάλαιο, κάθε ένα από αυτά τα κομμάτια έχει σχεδιαστεί για να παράγει κώδικα assembly, δυαδική ακολουθία και εκτελέσιμα αρχεία για το σύστημα στο οποίο βρίσκεται. Τα κομμάτια όμως ενός διαμεταγλωττιστή όπως αναφέραμε και στο κεφάλαιο 4.2.1. παρόλο που βρίσκονται σε ένα συγκεκριμένο υπολογιστικό σύστημα έχουν σχεδιαστεί για να μπορούν να μεταγλωττίζουν ένα πρόγραμμα για ένα διαφορετικό υπολογιστικό σύστημα με διαφορετικό επεξεργαστή. Για αυτό το λόγο ο ίδιος ο μεταγλωττιστής GCC έχει μεταγλωττιστεί ώστε να μπορεί να μεταγλωττίζει προγράμματα τα οποία προορίζονται για διαφορετικό υπολογιστικό σύστημα από αυτό στο οποίο βρίσκεται. (11)

4.3. Η γλώσσα προγραμματισμού C

Η γλώσσα προγραμματισμού C δημιουργήθηκε από τον Dennis Ritchie στα Bell Labs το 1972 όταν αυτός και ο Ken Thompson ασχολούνταν με τον σχεδιασμό του λειτουργικού συστήματος UNIX. Η C ήταν μια εξέλιξη της γλώσσας B του Ken Thompson και εκείνη με τη σειρά της ήταν μια εξέλιξη της γλώσσας BCPL και δημιουργήθηκε για να μπορέσει να καλύψει κάποιες αυξημένες ανάγκες στον προγραμματισμό. Αργότερα αναπτύχθηκαν πολλές παραλλαγές της C, ενώ ήταν επόμενο να υπάρχουν ασυμφωνίες μεταξύ τους. Έτσι, δημιουργήθηκε μια επιτροπή στις αρχές του καλοκαιριού του 1983 που άρχισε να δουλεύει πάνω στη δημιουργία ενός προτύπου ANSI το οποίο και θα όριζε μια για πάντα τη γλώσσα C. Η C θεωρείται γενικά γλώσσα μέσου επιπέδου και αυτό γιατί συνδυάζει στοιχεία των γλωσσών υψηλού επιπέδου (high level languages), όπως είναι η Cobol και η Pascal και στοιχεία των γλωσσών χαμηλού επιπέδου (low level languages), όπως είναι η Assembly. Πρέπει να αναφερθεί ότι σαν γλώσσα υψηλού επιπέδου θεωρείται η γλώσσα εκείνη που είναι αρκετά περιγραφική και που είναι έτσι πιο κοντά στην ανθρώπινη γραπτή γλώσσα και σαν γλώσσα χαμηλού επιπέδου θεωρείται εκείνη που είναι πιο κοντά στη μηχανή. Αυτός ο χαρακτηρισμός δεν έχει καμία απολύτως σχέση με τις δυνατότητες της γλώσσας. Σίγουρα, η κάθε γλώσσα προγραμματισμού έχει κάποιες προτεραιότητες να εκπληρώσει. Η Pascal, για παράδειγμα, χρησιμοποιείται κυρίως για τη σωστή διδασκαλία των αρχών του προγραμματισμού, ενώ η Basic δημιουργήθηκε έτσι ώστε να δώσει τη δυνατότητα σε αρχάριους στον προγραμματισμό να κάνουν με άνεση και ευκολία τα πρώτα τους βήματα στον ιδιόμορφο αυτό χώρο. Η Clipper και η Cobol είναι καθαρά επαγγελματικές γλώσσες προγραμματισμού. Η C, όμως, μπόρεσε να φέρει τον προγραμματιστή πιο κοντά στο hardware, που με τις άλλες γνωστές γλώσσες προγραμματισμού κάτι τέτοιο θα ήταν πολύ δύσκολο να γίνει. Βέβαια, η κάθε γλώσσα προγραμματισμού κάνει και διαφορετική δουλειά και δεν θα ήταν σωστό να κάνουμε συγκρίσεις, για τον ίδιο λόγο που δεν μπορούμε να συγκρίνουμε ένα αεροπλάνο μ’ ένα ποδήλατο καθώς το καθένα είναι προορισμένο να κάνει διαφορετική δουλειά. Τα τελευταία χρόνια η C έχει καθιερωθεί ως μια από τις σημαντικότερες και δημοφιλέστερες γλώσσες προγραμματισμού. Τα σημαντικότερα πλεονεκτήματα που εξηγούν αυτήν την προτίμησή της, αναφέρονται παρακάτω.

- *Χαρακτηριστικά Σχεδίασης*, Η C έχει μοντέρνες δομές ελέγχου για να μπορούμε να κάνουμε επαναληπτικές εργασίες και για εύκολη επιλογή εναλλακτικών τρόπων δράσης. Με το πλήθος των δομών δεδομένων που διαθέτει, μπορεί να αναπαραστήσει ένα μεγάλο σύνολο από διαφορετικούς τύπους πληροφοριών. Έχει και το μεγάλο πλεονέκτημα ότι επιβάλλει τη διάσπαση του προγράμματος σε αυτοδύναμες ενότητες, τις συναρτήσεις.
- *Αποτελεσματική*, Η C είναι μια αποτελεσματική γλώσσα προγραμματισμού, που είναι τόσο συμπεριεκτική, ώστε να χρησιμοποιούμε σ' αυτήν πολύ λιγότερες λέξεις σε σχέση με άλλες γλώσσες. Έχει έναν συμπαγή και γρήγορο κώδικα.
- *Φορητή Γλώσσα*, Η C είναι μια φορητή γλώσσα, δηλ. τα προγράμματά της μπορούν να τρέξουν με λίγες ή και με καθόλου τροποποιήσεις και σε ένα άλλο σύστημα.
- *Δυναμικότητα και Ευελιξία*, Η C είναι δυναμική και ευέλικτη, δύο ιδιότητες που είναι αρκετά δημοφιλείς στους υπολογιστές. Όπως ξέρουμε, το μεγαλύτερο μέρος του δυναμικού και ευέλικτου λειτουργικού συστήματος UNIX είναι γραμμένο σε C. Αυτό ισχύει και για επεξεργαστές κειμένων, μεταγλωττιστές (compilers) και ερμηνευτές (interpreters) γλωσσών προγραμματισμού. Η C διαθέτει μερικά από τα χαρακτηριστικά ελέγχου που συνήθως τα συναντάμε στη συμβολική γλώσσα (assembly language).

- *Προσανατολισμός προς τον Προγραμματιστή*, Η C είναι προσανατολισμένη προς τις ανάγκες του προγραμματιστή, ο οποίος και έχει άμεση πρόσβαση στο υλικό. Με τη C έχουμε τη σπουδαία δυνατότητα να μπορούμε να χειριζόμαστε μεμονωμένα τα *δυαδικά ψηφία (bits)* της μνήμης. Γενικά η C είναι πολύ λιγότερο περιοριστική στο να μας αφήνει να κάνουμε ό, τι θέλουμε σε σχέση με την Pascal για παράδειγμα.

Αυτή η ελευθερία είναι και πλεονέκτημα, αλλά είναι και επικίνδυνη όπως είναι φυσικό. Στη C τα πάντα (σχεδόν) επιτρέπονται. Δεν γίνεται έλεγχος των τύπων, άρα μπορεί κανείς να ανακατέψει ό,τι δεδομένα θέλει, κάτι που είναι πολύ χρήσιμο όταν προγραμματίζουμε σε επίπεδο συστήματος. Ακόμη, η C έχει μια τεράστια βιβλιοθήκη από χρήσιμες συναρτήσεις. Η C έχει και μειονεκτήματα, γιατί όπως πολύ καλά ξέρουμε η πολύ ελευθερία βλάπτει. Για παράδειγμα, η ελευθερία έκφρασης που αναφέραμε παραπάνω ότι έχει η C, απαιτεί από τον προγραμματιστή μια αυξημένη επαγρύπνηση και υπευθυνότητα. Ακόμη, η λακωνικότητα της C σε συνδυασμό με τον πλούτο των τελεστών που έχει, έχει σαν αποτέλεσμα τη δημιουργία προγραμμάτων που είναι τόσο δυσανάγνωστα, ώστε να είναι δύσκολο να τα κατανοήσει κάποιος με την πρώτη ματιά και πολλές φορές ακόμα και αυτός που τα έγραψε. Επιπλέον, συχνά είναι πολύ δύσκολο να ανιχνευθούν και τα λογικά λάθη σ' ένα πρόγραμμα της C. Η C έχει τελικά τόσες πολλές δυνατότητες έκφρασης, ώστε να χρειαστεί πολύς καιρός για να μπορεί να πει κανείς με βεβαιότητα ότι την έμαθε καλά.

Όπως είπαμε στα προηγούμενα, η C επιβάλλει τον καταμερισμό του προγράμματος σε ενότητες, που ονομάζονται *συναρτήσεις (functions)*. Εάν είναι απαραίτητο, οι συναρτήσεις μπορούν να χωριστούν και σε μικρότερες συναρτήσεις. Επίσης, στη C το κύριο πρόγραμμα είναι κι αυτό μια συνάρτηση, που ονομάζεται *main()*. Μια μέθοδος για το γράψιμο ενός προγράμματος στη C είναι να ξεκινήσουμε γράφοντας τη συνάρτηση main(), την ενότητα του πιο πάνω επιπέδου και μετά να ασχοληθούμε με τις συναρτήσεις των πιο κάτω επιπέδων. Η διαδικασία αυτή ονομάζεται *πάνω-προς-τα-κάτω προγραμματισμός (top-down programming)*. Η αντίστροφη διαδικασία, δηλ. το να ασχοληθούμε πρώτα με τις συναρτήσεις των κατώτερων επιπέδων και μετά να ανεβαίνουμε προς τα πάνω, ονομάζεται *κάτω-προς-τα-πάνω προγραμματισμός (bottom-up programming)*. Ένα πλεονέκτημα του πάνω-προς-τα-κάτω προγραμματισμού είναι ότι μπορούμε να χαράξουμε καλύτερα τη ροή του προγράμματος, μιας και δεν ασχολούμαστε από την αρχή με τις λεπτομέρειες των επί μέρους συναρτήσεων.

Ο *μεταγλωττιστής (compiler)* της C μετατρέπει τον *πηγαίο κώδικα (source program)*, δηλ. το πρόγραμμα που γράφουμε σε C, σ' έναν *αντικειμενικό κώδικα (object program)* και το *πρόγραμμα σύνδεσης (linker)* συνδυάζει αυτόν τον κώδικα με άλλους κώδικες και δημιουργείται έτσι το *εκτελέσιμο αρχείο (executable file)*. Τα προγράμματα της C έχουν την επέκταση ***.c***. Ο ρόλος του προγράμματος σύνδεσης είναι να ενώσει τον τελικό κώδικα, τον *κώδικα εκκίνησης (start-up code)* του συστήματός μας και τον *κώδικα βιβλιοθήκης (library code)* στο εκτελέσιμο αρχείο. Ο κώδικας εκκίνησης έχει σχέση με την επικοινωνία μεταξύ του προγράμματος και του λειτουργικού συστήματος και ο κώδικας βιβλιοθήκης περιέχει τον τελικό κώδικα για πολλές συναρτήσεις. Σε μερικά συστήματα πρέπει να τρέξουμε τα προγράμματα μεταγλώττισης και σύνδεσης ξεχωριστά, ενώ σ' άλλα ο μεταγλωττιστής ενεργοποιεί το πρόγραμμα σύνδεσης αυτόματα μόνος του. (12)(13) Το σύστημα στο οποίο έγινε το πείραμα προγραμματίστηκε με βάση την γλώσσα προγραμματισμού C.

4.4. Αναφορά στο RTEMS

Το RTEMS είναι ένα λειτουργικό σύστημα πραγματικού χρόνου το οποίο έχει σχεδιαστεί ειδικά για ενσωματωμένα συστήματα. Η ανάπτυξη του RTEMS ξεκίνησε στα τέλη της δεκαετίας του 1980 με κάποιες πρόωρες εκδόσεις διαθέσιμες μέσω FTP το 1993. Η εταιρία OAR είναι αυτή η οποία διαχειρίζεται το RTEMS. Το RTEMS δεν παρέχει κανένα είδος διαχείρισης μνήμης ή διεργασιών. Υλοποιεί μια διεργασία σε πολύ-νηματικό περιβάλλον. Το RTEMS παρέχει σχεδόν όλες τις υπηρεσίες που παρέχονται από το POSIX[8], εκτός από τις ήδη υπάρχουσες που είναι για χαρτογράφηση μνήμης, δημιουργία διεργασιών και διαμοιρασμό μνήμης. (14)(15)

Οι αρχιτεκτονικές οι οποίες είναι συμβατές με το RTEMS είναι οι εξής:

- ARM
- Atmel AVR
- Blackfin
- Freescale ColdFire

[8] Το POSIX είναι ένα σύνολο από standards που χρησιμοποιούνται για να διατηρείται η συμβατότητα μεταξύ λειτουργικών συστημάτων.

- Texas Instruments C3x/C4x DSPs
- H8/300
- Intel 80386, Pentium, και αρχιτεκτονικές πάνω από την αρχιτεκτονική x86
- Lattice Mico32
- 68k
- MIPS
- Nios II
- PowerPC
- Renesas M32C
- Renesas SuperH
- SPARC
- ERC32
- LEON
- SPARC_V9

Το λειτουργικό σύστημα πραγματικού χρόνου RTEMS δεν χρησιμοποιήθηκε στο σύστημα που έγινε το πείραμα αλλά έγινε αναφορά γιατί όπως βλέπουμε και παραπάνω είναι συμβατό με τον LEON 2 που χρησιμοποιήθηκε στο σύστημα. Επομένως τα εργαλεία του RTEMS όπως για παράδειγμα ο μεταγλωττιστής, ήταν κατάλληλα για να μεταγλωττίσουν τον πηγαίο κώδικα σε μορφή κατάλληλη για τον συγκεκριμένο επεξεργαστή.

4.5. Ο μεταγλωττιστής GCC

Ο "συγγραφέας" του μεταγλωττιστή GCC είναι ο Richard Stallman, ιδρυτής του GNU Project.

Το GNU project ξεκίνησε το 1984 με σκοπό τη δημιουργία ενός δωρεάν λειτουργικού συστήματος, το οποίο θα έμοιαζε με τα UNIX για να προωθήσει την ελευθερία και τη συνεργασία ανάμεσα σε απλούς χρήστες και προγραμματιστές. Κάθε λειτουργικό σύστημα που μοιάζει με UNIX χρειάζεται ένα μεταγλωττιστή για C, και εφόσον εκείνη την εποχή δεν υπήρχε κανένας μεταγλωττιστής ο οποίος να διατίθεται δωρεάν, το GNU project έπρεπε να δημιουργήσει έναν από την αρχή. Το έργο αυτό χρηματοδοτήθηκε από δωρεές ξεχωριστών ατόμων και εταιρειών για το Free Software Foundation, μια μη κερδοσκοπική οργάνωση η οποία δημιουργήθηκε με σκοπό την υποστήριξη του GNU project.

Η πρώτη έκδοση του μεταγλωττιστή GCC έγινε το 1987. Αυτή ήταν μια σημαντική εξέλιξη, εφόσον ήταν ο πρώτος μεταγλωττιστής, ο οποίος υποστήριζε φορητότητα και βελτιστοποίηση της γλώσσας C. Από τότε ο GCC έχει γίνει ένα από τα πιο σημαντικά εργαλεία στην εξέλιξη των δωρεάν λογισμικών.

Μια σημαντική αναθεώρηση του μεταγλωττιστή ήρθε με τη σειρά 2.0 το 1992, η οποία προσέθετε την ιδιότητα να γίνεται μεταγλώττιση της γλώσσας προγραμματισμού C++. Το 1997 ένας πειραματικός κλάδος του μεταγλωττιστή (EGCS) δημιουργήθηκε, για να κάνει καλύτερη τη βελτιστοποίηση και να παρέχει υποστήριξη της γλώσσας προγραμματισμού C++. Μετά από αυτό το έργο το EGCS υιοθετήθηκε ως η νέα βασική γραμμή της ανάπτυξης του GCC, και τα χαρακτηριστικά του έγιαναν ευρέως διαθέσιμα στην έκδοση 3.0 του GCC.

Με την πάροδο του χρόνου, ο GCC επεκτάθηκε για να υποστηρίζει πολλές διαφορετικές γλώσσες προγραμματισμού, συμπεριλαμβανομένων των Fortran, ADA, Java και Objective-C. [12]

4.5.1. Τα βασικά χαρακτηριστικά του GCC

Αρχικά ο GCC είναι ένας μεταγλωττιστής, ο οποίος υποστηρίζει φορητότητα. Μπορεί να λειτουργήσει σε πολλές διαφορετικές πλατφόρμες, οι οποίες είναι διαθέσιμες σήμερα, και μπορεί να παράξει έξοδο για πολλούς τύπους επεξεργαστών. Σε αντίθεση με τους επεξεργαστές που χρησιμοποιούνται στους προσωπικούς υπολογιστές, υποστηρίζει επίσης μικροελεγκτές, DSPs και επεξεργαστές των 64 bit.

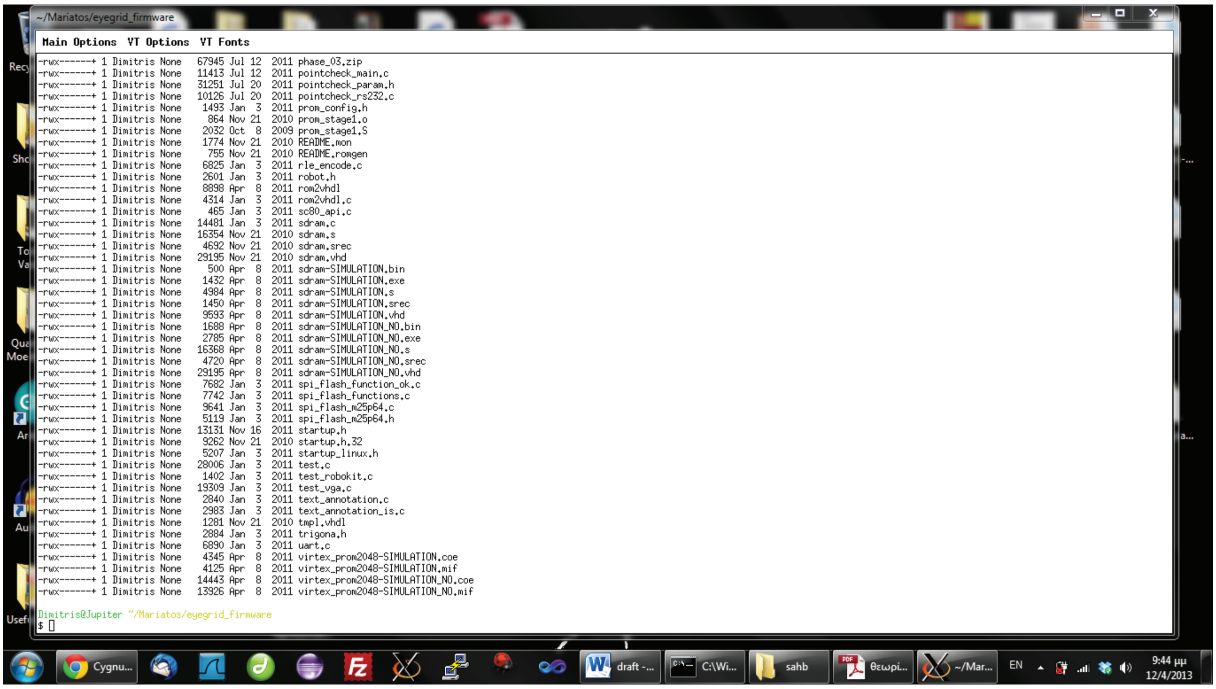
Ο GCC δεν μεταγλωττίζει προγράμματα μόνο για το υπολογιστικό σύστημα στο οποίο βρίσκεται, αλλά μπορεί να διαμεταγλωττίσει οποιοδήποτε πρόγραμμα παράγοντας εκτελέσιμα αρχεία για ένα διαφορετικό υπολογιστικό σύστημα. Αυτό επιτρέπει σε ένα πρόγραμμα να μεταγλωττιστεί για ένα ενσωματωμένο σύστημα. Ο GCC είναι γραμμένος στη γλώσσα προγραμματισμού C με ιδιαίτερη εστίαση στη φορητότητα και μπορεί να μεταγλωττίσει τον εαυτό του, ώστε να προσαρμοστεί εύκολα σε νέα συστήματα.

Ο GCC έχει τμήματα αρχικής επεξεργασίας (frontends) σε διάφορες γλώσσες για την ανάλυση διαφορετικών γλωσσών. Προγράμματα σε κάθε γλώσσα μπορούν να μεταγλωττιστούν, ή να διαμετταγλωττιστούν για οποιαδήποτε αρχιτεκτονική. Για παράδειγμα ένα πρόγραμμα γραμμένο στη γλώσσα προγραμματισμού ADA μπορεί να μεταγλωττιστεί για έναν μικροελεγκτή, ή ένα πρόγραμμα γραμμένο στη γλώσσα προγραμματισμού C μπορεί να μεταγλωττιστεί για έναν υπέρ-υπολογιστή.

Ο GCC έχει κλιμακωτό σχεδιασμό, επιτρέποντας με αυτόν τον τρόπο να προστεθούν νέες γλώσσες και νέες αρχιτεκτονικές. Προσθέτοντας μια νέα γλώσσα στο τμήμα αρχικής επεξεργασίας του GCC επιτρέπεται η χρησιμοποίησή της σε οποιαδήποτε αρχιτεκτονική, υπό την προϋπόθεση ότι παρέχονται οι απαραίτητες βιβλιοθήκες. Παρόμοια προσθέτοντας υποστήριξη για μια νέα αρχιτεκτονική γίνεται διαθέσιμη για όλες τις γλώσσες. [12]

4.6. Το τερματικό Cygwin

Το Cygwin είναι ένα περιβάλλον προσομοίωσης του UNIX/Linux στα Windows. Παρέχει στον χρήστη την δυνατότητα να δημιουργεί προγράμματα κάνοντας χρήση των βιβλιοθηκών και των υπηρεσιών του UNIX ενώ το ίδιο το εκτελέσιμο του προγράμματος τρέχει σε Windows. Εκτελώντας το Cygwin ανοίγει ένα παράθυρο στο οποίο τρέχει ο κλασικός φλοιός bash του UNIX στον οποίο μπορούν να εκτελεστούν κανονικά εντολές του UNIX bash shell, καθώς και μια πληθώρα προγραμμάτων που συναντάμε σε UNIX/Linux συστήματα (μεταξύ αυτών και ο GCC). Το Cygwin αποτελείται από δύο κομμάτια. Το πρώτο είναι μια δυναμική βιβλιοθήκη (DLL) η οποία χρησιμοποιείται σαν API παρέχοντας λειτουργίες που παρέχει ο POSIX αλλά σε περιβάλλον Windows και το δεύτερο κομμάτι είναι η συλλογή από εργαλεία λογισμικού και εφαρμογές που παρέχουν την δυνατότητα προσομοίωσης του περιβάλλοντος UNIX/Linux. (16) Το Cygwin ήταν το βασικό εργαλείο που χρησιμοποιήθηκε στο σύστημα του πειράματος γιατί προσέφερε την δυνατότητα της διαμεταγλώττισης του πηγαίου κώδικα με την χρήση του μεταγλωττιστή GCC αλλά σε περιβάλλον Windows. Στην Εικόνα 4-2 βλέπουμε το Cygwin σε περιβάλλον Windows.



Εικόνα 4-2. Το τερματικό Cygwin σε περιβάλλον Windows

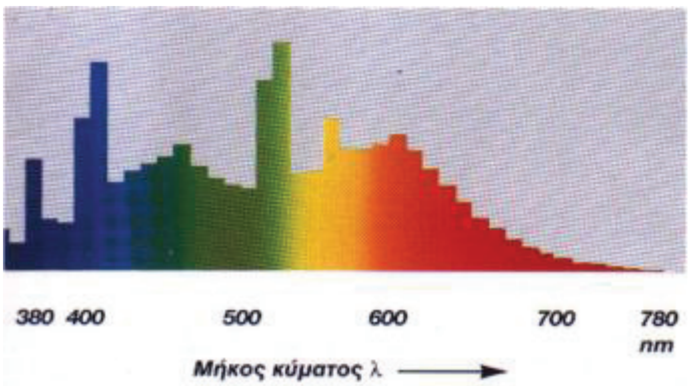
Χρωματικό μοντέλο κάμερας και White balance

5.1. Εισαγωγή

Σε αυτό το κεφάλαιο περιγράφεται το χρωματικό μοντέλο που χρησιμοποιεί η κάμερα του συστήματος για την δημιουργία χρωμάτων, και η ισοστάθμιση του λευκού φωτός (White Balance) για την σωστή αναγνώριση των χρωμάτων στην εικόνα. Αρχικά γίνεται μια αναφορά στο χρωματικό μοντέλο RGB (Red – Green – Blue), και έπειτα πως αυτό το μοντέλο εφαρμόζεται στην κάμερα. Τέλος γίνεται περιγραφή της λειτουργίας white balance που προσφέρει η κάμερα και πως βοηθάει στην καλύτερη ανάλυση της εικόνας.

5.2. Το χρωματικό μοντέλο RGB

Τα χρώματα είναι μία κωδικοποίηση του ανθρώπινου νευρικού συστήματος για να διακρίνει τα μήκη κύματος του φωτός (από 400 nm έως 700 nm) που διεγείρουν το μάτι μας, όπως φαίνεται και στην Εικόνα 5-1. Όταν στο μάτι του ανθρώπου προσπέσουν δύο ακτινοβολίες με διαφορετικά μήκη κύματος η ανθρώπινη όραση συνθέτει τα χρώματα δημιουργώντας καινούργια. Με άλλα λόγια στηριζόμενοι σε κάποια χρώματα τα οποία ονομάζουμε βασικά ή πρωτογενή μπορούμε να συνθέσουμε τα υπόλοιπα.



Εικόνα 5-1. Χρωματική απόδοση του φωτός ανάλογα με το μήκος κύματος ‘λ’ (18)

Το χρωματικό μοντέλο RGB βασίζεται στη θεωρία Young-Helmholtz για την τριχρωματική οπτική, η οποία αναπτύχθηκε από τους Thomas Young και James Hermann Helmholtz, από τις αρχές έως τα μέσα του δεκάτου ενάτου αιώνα, και στο χρωματικό τρίγωνο του James Clerk Maxwell.

Από το 17ο αιώνα, μετά τις εργασίες του Νεύτωνα αναφορικά με την Οπτική, γνωρίζουμε ότι το χρώμα δεν είναι ύλη αλλά μια αίσθηση. Με άλλα λόγια, δεν υπάρχει πάρα μόνο το φως. Χάρη στον Ισαάκ Νεύτωνα αμφισβητήθηκε ότι το ηλιακό φως είναι ένα απλό χρώμα και απέδειξε πειραματικά ότι το ηλιακό φως αποτελείται από πολλά χρώματα. Με άλλα λόγια διατύπωσε τη θεωρία ότι το φως είναι σύνθετο.

Από τα πειράματα που υλοποίησε ο Νεύτωνας, οδηγήθηκε στη μελέτη του χρωματικού φάσματος και καθιέρωσε τα επτά χρώματα του φάσματος ως "βασικά" εφόσον η ανάμειξή τους παρήγαγε λευκό φως. Επίσης, συμπέρανε ότι στο φάσμα του λευκού φωτός δεν υπάρχουν όλα τα χρώματα που βλέπουμε, όπως για παράδειγμα το καφέ χρώμα.

Ο Νεύτωνας συνέχισε τα πειράματα του με σκοπό να αναζητήσει τον ελάχιστο αριθμό βασικών χρωμάτων του φάσματος με βάση τα οποία θα πετύχαινε τη σύνθεση του λευκού φωτός. Φαίνεται ότι έψαχνε προς λάθος κατεύθυνση μιας και είχε την ιδέα ότι αυτά θα πρέπει να είναι ίδια με τα βασικά χρώματα – βαφές του ζωγράφου: κόκκινο, κίτρινο, μπλε. (17)

Στις αρχές του 19ου αιώνα ο Thomas Young διετύπωσε την Τριχρωματική Θεωρία του (Course of Lectures on Natural Philosophy, 1807) σύμφωνα με την οποία η χρωματική όραση βασίζεται σε τρεις χρωματικούς υποδοχείς του ανθρώπινου ματιού, που αντιστοιχούν σε τρία φασματικά βασικά χρώματα: Το κόκκινο, το πράσινο και το μπλε. Επίσης, υπολόγισε τα μήκη κύματος των επτά χρωμάτων – ακτινοβολιών του ορατού φάσματος του Νεύτωνα.

Στα μέσα του 19ου αιώνα και ο φυσικός Maxwell (1857) δέχθηκε ότι υπάρχει διαφορά ανάμεσα σε μια θεωρία του χρώματος ως χρωστικής ύλης στη ζωγραφική και σε μια θεωρία φωτός όπως αυτή του Young. Επίσης επεσήμανε ότι η επιλογή των τριών βασικών χρωμάτων (Red – Green – Blue, Κόκκινο – Πράσινο - Μπλε) θα μπορούσε να είναι οποιαδήποτε άλλη, αρκεί ο συνδυασμός των χρωμάτων, σε σωστές αναλογίες, να παράγει το λευκό.

Το 1965, μετά από πειράματα επιβεβαιώθηκε ότι στον αμφιβληστροειδή χιτώνα του ανθρώπινου ματιού υπάρχουν ειδικά κύτταρα, τα οποία ονομάζονται φωτοϋποδοχείς ή φωτοαισθητήρες, και είναι υπεύθυνα τόσο για την αντίληψη του αμυδρού φωτός (ραβδία) όσο για την αντίληψη των χρωμάτων (κωνία):

- S-κωνία: είναι ευαίσθητα σε φωτόνια μικρού μήκους κύματος (μπλε φως) και παρουσιάζουν μέγιστη ευαισθησία σε μήκος κύματος περίπου 420 nm.
- M-κωνία: είναι ευαίσθητα σε φωτόνια μεσαίου μήκους κύματος (πράσινο φως) και παρουσιάζουν μέγιστη ευαισθησία σε μήκος κύματος περίπου 530 nm.
- L-κωνία: είναι ευαίσθητα σε φωτόνια μεγάλου μήκους κύματος (κόκκινο φως) και παρουσιάζουν μέγιστη ευαισθησία σε μήκος κύματος περίπου 560 nm.

Τα ραβδία είναι εξειδικευμένα στην αίσθηση της έντασης του φωτός που πέφτει στο μάτι μας και ενεργοποιούνται μόνο αν φθάνουν σ' αυτό 500 φωτόνια το δευτερόλεπτο. Από την άλλη, τα κωνία ενεργοποιούνται και με 10 φωτόνια ανά δευτερόλεπτο.

Όλα τα χρώματα φωτός σύμφωνα με *CIE (Committé Internationale de l' Éclairage*) αναλύονται σε τρεις διαφορετικές χρωματικές συντεταγμένες (ηλεκτρομαγνητικές ακτινοβολίες) την Κόκκινη (R), Πράσινη(G) και Μπλε (B) (Χρωματικό Σύστημα RGB). (18) Σ' αυτό το μοντέλο κάθε χρώμα μπορεί να παρασταθεί με μία τριάδα αριθμών και τιμές από 0 έως 255. Το μοντέλο βασίζεται στο γεγονός ότι όταν μία οθόνη δεν εκπέμπει φως εμφανίζεται μαύρη. Τα υπόλοιπα χρώματα δημιουργούνται με υπέρθεση των τριών βασικών με συγκεκριμένη αναλογία. Για το λόγο αυτό, το μοντέλο RGB χαρακτηρίζεται και ως προσθετικό: Με την ανάμειξη φωτεινών ακτίνων R, G, B παράγονται άλλα χρώματα. Επομένως, το χρωματικό μοντέλο κωδικοποιεί όλα τα χρώματα που μπορούν να εμφανιστούν σε μία οθόνη (συνήθως υπολογιστή). Ο τύπος που ισχύει και χρησιμοποιείται για το RGB είναι ο παρακάτω.

$$1.0(C) = r(R) + g(G) + b(B)$$

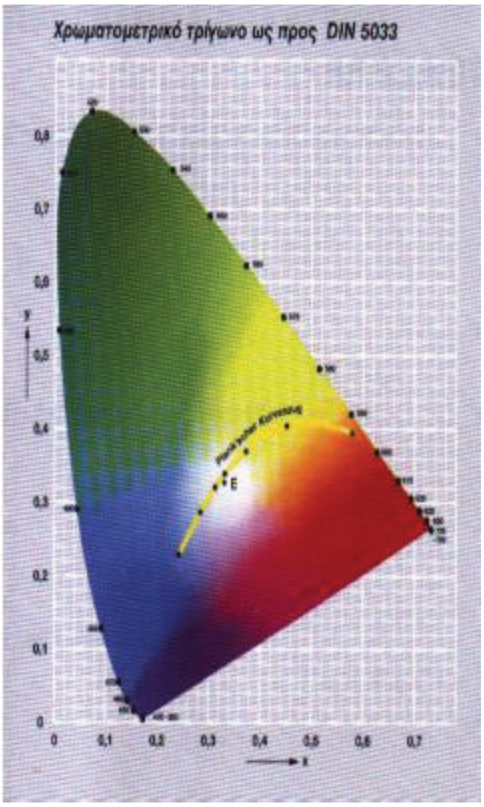
Όπου 1,0 C είναι το χρώμα που αναζητούμε, r(R) μια ποσότητα Κόκκινο, g(G) μια ποσότητα Πράσινο, b(B) μια ποσότητα Μπλε. Από τον τύπο προκύπτει ότι:

$$1 = r + g + b$$

επομένως για να προσδιοριστεί ένα χρώμα αρκεί να γνωρίζουμε δύο από τις τρεις συντεταγμένες του:

$$1 - (r + g) = b$$

Για να μην υπάρχουν όμως αρνητικές τιμές η CIE δημιούργησε το χρωματομετρικό τρίγωνο στο οποίο κάθε συντεταγμένη δίνεται κατά άξονα X και Ψ και αναπαριστάται στην Εικόνα 5-2.



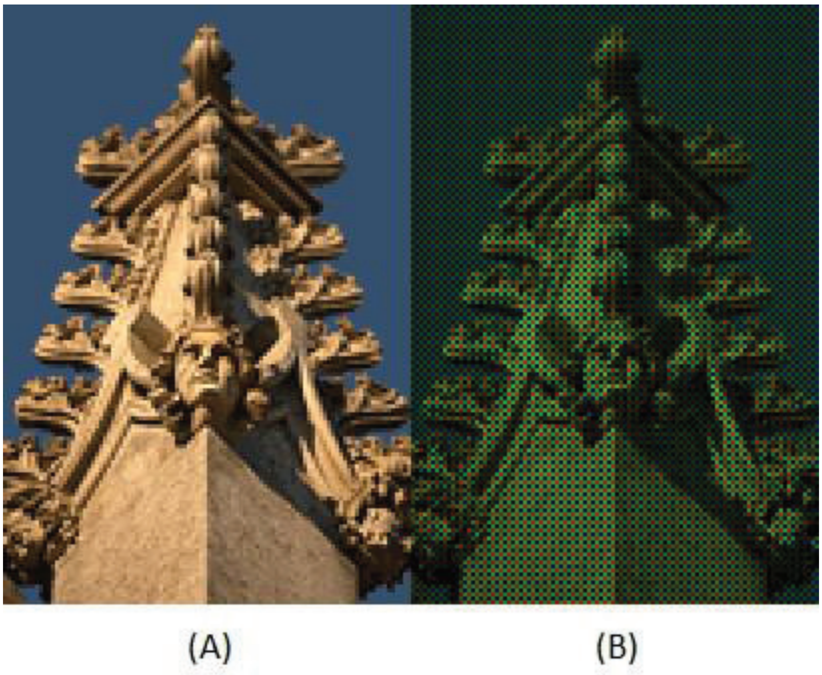
Εικόνα 5-2. Χρωματομετρικό τρίγωνο κατά CIE (18)

Εκτός του RGB υπάρχουν και άλλα χρωματικά μοντέλα όπως το CIELab, το CMY(K), το HSB και άλλα, τα οποία χρησιμοποιούνται στην επεξεργασία ψηφιακής εικόνας. Το RGB είναι όμως πιο κοντά από κάθε άλλο μοντέλο στο επίπεδο της φυσικής λειτουργίας των μέσων απεικόνισης όπως οι τηλεοράσεις, οι κάμερες και οι οθόνες των υπολογιστών. (19)

5.3. Το μοντέλο RGB στην κάμερα

Η κάμερα του συστήματος στο οποίο έγινε το πείραμα χρησιμοποιεί το μοντέλο RGB. Όπως και όλες οι ψηφιακές κάμερες που χρησιμοποιούν το μοντέλο RGB, η κάμερα του συστήματος έχει τρία φίλτρα τα οποία επιτρέπουν να περνάνε μόνο τρία συγκεκριμένα μήκη κύματος του φωτός: το κόκκινο, το πράσινο και το μπλε. Η κάμερα δεν μπορεί να ξεχωρίσει χρώματα. Το μόνο που μπορεί να "δει" στον χώρο είναι η ένταση του φωτός που υπάρχει και για αυτό το λόγο χρησιμοποιούνται αυτά τα τρία φίλτρα. Πίσω από κάθε φίλτρο βρίσκεται ένας αισθητήρας φωτός, ο οποίος λαμβάνει την ένταση του φωτός που εκπέμπεται από κάθε χρώμα. Σε μια κάμερα το σύνολο τον φίλτρων και τον αισθητήρων σχηματίζουν εκατομμύρια μικροσκοπικές εισόδους, ώστε να μπορέσει να σχηματιστεί η εικόνα που βλέπει η κάμερα εκείνη την στιγμή. Για το πράσινο χρώμα οι αισθητήρες φωτός και τα φίλτρα είναι διπλάσια από τους αισθητήρες του κόκκινου και του μπλε χρώματος. Με αυτό τον τρόπο το σύνολο των πράσινων φωτονίων είναι μεγαλύτερο από το σύνολο των κόκκινων και των μπλε και αυτό συμβαίνει γιατί το ανθρώπινο μάτι έχει μεγαλύτερη ευαισθησία στο πράσινο χρώμα σε σχέση με το κόκκινο και το μπλε.

Το κάθε χρώμα από τα τρία βασικά χρώματα, που εισέρχεται στην κάμερα την στιγμή της λήψης τοποθετείται σε έναν πίνακα Bayer. Με αυτό τον τρόπο όμως η εικόνα δεν μπορεί να αναπαρασταθεί όπως είναι στην πραγματικότητα Εικόνα 5-3. Για αυτό το λόγο ο πίνακας Bayer στέλνεται στην FPGA και η FPGA με την σειρά της διαχωρίζει τον πίνακα αυτόν σε κομμάτια των τεσσάρων pixel. Τέσσερα pixel στον πίνακα Bayer ισοδυναμούν με ένα pixel στο μοντέλο RGB. Επομένως η FPGA για κάθε τέσσερα pixel (που το κάθε pixel έχει ένα χρώμα από τα τρία βασικά χρώματα), αναμιγνύοντάς τα δημιουργεί ένα νέο pixel RGB.



Εικόνα 5-3. Στην εικόνα (A) βλέπουμε το τοπίο μετά από την λήψη και στην εικόνα (B) πως βλέπει το τοπίο η κάμερα μέσα από το φίλτρο Bayer (23)

5.4. Automatic White Balance Estimator – Perfect Reflector

Το White Balance είναι μια διαδικασία αφαίρεσης των μη ρεαλιστικών χρωμάτων, έτσι ώστε τα αντικείμενα που είναι λευκά στην πραγματικότητα, να αποδίδονται λευκά και μέσω μιας κάμερας. Για το σωστό White Balance πρέπει να ληφθεί υπ' όψη η "θερμοκρασία" χρώματος από μια πηγή φωτός. Το ανθρώπινο μάτι είναι απόλυτα ικανό να κρίνει τί είναι λευκό κάτω από διαφορετικές πηγές φωτός, αλλά οι ψηφιακές κάμερες έχουν συχνά μεγάλη δυσκολία στο αυτόματο White Balance. (22)

Μέσα στην FPGA που χρησιμοποιείται για το πείραμα του συστήματος βρίσκεται ο DIAPLOUS Automatic White Balance Estimator, ο οποίος αποτελεί μια ευέλικτη μονάδα που διαβάζει ένα ολόκληρο πλαίσιο από μια ροή από pixel δεδομένα σε μορφή RGB και δημιουργεί τις παραμέτρους για τη χρωματική ισορροπία. Από τη συγκεκριμένη μονάδα χρησιμοποιούνται τρείς αλγόριθμοι εκτίμησης: Perfect Reflector, Modified Gray World και One-Shot WB Calibration. Στο πείραμα του συστήματος χρησιμοποιείται μόνο ο Perfect Reflector.

Ο αλγόριθμος Perfect Reflector χρησιμοποιείται σε εικόνες – σκηνές, στις οποίες υπάρχει κάποιο φωτεινό σημείο, ή στην καλύτερη περίπτωση μια αντανάκλαση. Ο συγκεκριμένος αλγόριθμος βασίζεται στην υπόθεση ότι το φωτεινότερο pixel σε μια εικόνα αντιστοιχεί σε ένα σημείο μιας γυαλιστερής ή κατοπτρικής επιφάνειας, η οποία μεταφέρει ένα μεγάλο αριθμό πληροφοριών για το φωτισμό της σκηνής. Οι κατοπτρικές ή γυαλιστερές επιφάνειες αντανακλούν το πραγματικό χρώμα της φωτεινής πηγής, διότι οι λειτουργίες ανάκλασής τους είναι σταθερές για ένα ευρύ φάσμα μηκών κύματος. Ο αλγόριθμος Perfect Reflector εκμεταλλεύεται αυτή την ιδιότητα για να ρυθμίσει την εικόνα. Εντοπίζει το πιο φωτεινό pixel και το εκχωρεί ως αναφορά λευκού σημείου. Έτσι υποθέτει ότι το συγκεκριμένο σημείο θα πρέπει να είναι λευκό και υπολογίζει κατάλληλα τη θερμοκρασία του χρώματος της πηγής φωτός. Είναι ιδιαίτερα ευαίσθητος σε κορεσμένα pixels και γι' αυτόν το λόγο ο χρήστης πρέπει να έχει καλή γνώση των συνθηκών. (20)(21)

6.1. Εισαγωγή

Σε αυτό το κεφάλαιο γίνεται περιγραφή των αλγορίθμων που υλοποιήθηκαν στην FPGA του συστήματος. Αρχικά περιγράφεται ο τρόπος λειτουργίας του αλγορίθμου για την αναγνώριση του κόκκινου χρώματος. Έπειτα γίνεται περιγραφή του αλγορίθμου που υπολογίζει το κέντρο του αντικειμένου που έχει αναγνωριστεί ως κόκκινο και τέλος γίνεται περιγραφή του αλγορίθμου Connected Component Labeling (CCL), ο οποίος χρησιμοποιείται ώστε να υπολογίζεται το κέντρο πολλών διαφορετικών αντικειμένων που έχουν αναγνωριστεί ως κόκκινα στην εικόνα.

6.2. Αναγνώριση χρώματος

Η κάμερα που χρησιμοποιείται στο σύστημα, χρησιμοποιώντας το μοντέλο RGB, στέλνει τα δεδομένα της εκάστοτε εικόνας στην FPGA του συστήματος. Τα δεδομένα αυτά είναι πληροφορίες για την τιμή κάθε βασικού χρώματος (κόκκινο, πράσινο, μπλε – RGB) ανά pixel. Δηλαδή για κάθε χρώμα του χώρου γίνεται, μέσω των φίλτρων της κάμερας, χρωματική ανάλυση και διαχωρισμός του κάθε χρώματος σε κόκκινο, πράσινο και μπλε.

Τα τρία βασικά χρώματα αναπαρίστανται στην FPGA ως αριθμοί παίρνοντας τιμές από 0 έως 255. Έχοντας μια τριάδα τιμών ανά pixel και με τη χρήση των παρακάτω τύπων μπορεί να υπολογιστεί το ποσοστό έντασης για κάθε ένα από τα τρία βασικά χρώματα ανά pixel.

$$percentage = \left(\frac{r(Red)}{r(Red) + g(Green) + b(Blue)}\right) \cdot 100$$

1. Υπολογισμός για το ποσοστό του κόκκινου στην εικόνα

$$percentage = (\frac{g(Green)}{r(Red) + g(Green) + b(Blue)}) \cdot 100$$

2. Υπολογισμός για το ποσοστό του πράσινου στην εικόνα

$$percentage = (\frac{b(Blue)}{r(Red) + g(Green) + b(Blue)}) \cdot 100$$

3. Υπολογισμός για το ποσοστό του μπλε στην εικόνα

Για τον υπολογισμό των ποσοστών γίνεται η πράξη της διαίρεσης. Στη συγκεκριμένη διαδικασία παρουσιάστηκε πρόβλημα, καθώς η FPGA δεν μπορούσε να υλοποιήσει τον υπολογισμό της διαίρεσης. Επομένως δημιουργήθηκε κώδικας, ο οποίος υλοποιεί τον υπολογισμό της διαίρεσης, και δίνεται στο παράρτημα Α.

Έπειτα μέσω της FPGA γίνεται σύγκριση των τριών βασικών χρωμάτων για κάθε pixel ξεχωριστά με σκοπό την εύρεση των pixel, των οποίων το ποσοστό κόκκινου χρώματος είναι μεγαλύτερο από το ποσοστό των άλλων δυο χρωμάτων. Όταν ένα pixel αναγνωρίζεται ως κόκκινο, τότε η FPGA θέτει μια προκαθορισμένη από το χρήστη τιμή χρώματος στο συγκεκριμένο pixel, έτσι ώστε όταν η ολοκληρωμένη εικόνα φτάνει στο χρήστη, μπορεί ο ίδιος ο χρήστης να δει ποια pixel έχουν αναγνωριστεί από την FPGA ως κόκκινα. Η προκαθορισμένη τιμή που έχει οριστεί για το πείραμα μετατρέπει το κόκκινο χρώμα σε κίτρινο.

Ένα σημαντικό πρόβλημα το οποίο υπήρξε ήταν πως ο υπολογισμός του ποσοστού του κόκκινου χρώματος ήταν εσφαλμένος σε περιβάλλον όπου ο φωτισμός προερχόταν από λάμπα πυρακτώσεως. Αυτό το πρόβλημα μπόρεσε να αντιμετωπιστεί με την χρήση της τεχνικής white balance που αναφέρεται στο 5^{ο} κεφάλαιο.

Κατά την διάρκεια του υπολογισμού των pixel που έχουν κόκκινο χρώμα γίνεται και υλοποίηση ακόμα δύο αλγορίθμων, για τον υπολογισμό του κέντρου του αντικειμένου και για τον υπολογισμό του κέντρου για περισσότερα από ένα κόκκινα αντικείμενα στην εικόνα, των οποίων η ανάλυση γίνεται σε επόμενα κεφάλαια.

6.3. Υπολογισμός του κέντρου ενός αντικειμένου

Ο δεύτερος αλγόριθμος ο οποίος υλοποιήθηκε ήταν για τον εντοπισμό του κέντρου ενός αντικειμένου. Μετά την αναγνώριση του κόκκινου χρώματος στην εικόνα το κόκκινο χρώμα αντικατοπτρίζει κάποιο αντικείμενο. Ο αλγόριθμος βρίσκει στην εικόνα τα pixel που έχουν αναγνωριστεί ως κόκκινα “διαβάζοντας” ένα προς ένα τα pixel της εικόνας. Ύστερα με βάση το σχήμα που δημιουργείται στην εικόνα από το σύνολο τον κόκκινων pixel υπολογίζει το κέντρο του αντικειμένου. Ο κώδικας που υλοποιεί τον αλγόριθμο για τον υπολογισμό του κέντρου ενός αντικειμένου δίνεται στο παράρτημα Α.

Στον Πίνακα 6-1 δίνεται ένα παράδειγμα για το πώς γίνεται ο υπολογισμός του κέντρου.

X Axis

Y Axis

0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.10
1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	1.10
2.1	2.2	2.3	2.4	2.5	2.6	2.7	2.8	2.9	2.10
3.1	3.2	3.3	3.4	3.5	3.6	3.7	3.8	3.9	3.10
4.1	4.2	4.3	4.4	4.5	4.6	4.7	4.8	4.9	4.10
5.1	5.2	5.3	5.4	5.5	5.6	5.7	5.8	5.9	5.10
6.1	6.2	6.3	6.4	6.5	6.6	6.7	6.8	6.9	6.10
7.1	7.2	7.3	7.4	7.5	7.6	7.7	7.8	7.9	7.10
8.1	8.2	8.3	8.4	8.5	8.6	8.7	8.8	8.9	8.10
9.1	9.2	9.3	9.4	9.5	9.6	9.7	9.8	9.9	9.10
10.1	10.2	10.3	10.4	10.5	10.6	10.7	10.8	10.9	10.10
11.1	11.2	11.3	11.4	11.5	11.6	11.7	11.8	11.9	11.10
12.1	12.2	12.3	12.4	12.5	12.6	12.7	12.8	12.9	12.10
13.1	13.2	13.3	13.4	13.5	13.6	13.7	13.8	13.9	13.10
14.1	14.2	14.3	14.4	14.5	14.6	14.7	14.8	14.9	14.10
15.1	15.2	15.3	15.4	15.5	15.6	15.7	15.8	15.9	15.10

Πίνακας 6-1. Παράδειγμα υπολογισμού κέντρου σε έναν πίνακα μεγέθους 16x10

Τα κόκκινα σημεία είναι το αντικείμενο το οποίο έχει αναγνωριστεί από την κάμερα και το κίτρινο σημείο είναι το κέντρο του αντικειμένου. Ο αλγόριθμος ακολουθεί τον παρακάτω μαθηματικό τύπο για τον υπολογισμό του κέντρου.

Έστω πως υπάρχει ένα σύνολο σημείων:

$$X: [(y_1, x_1), (y_2, x_2), \ldots, (y_n, x_n)] \text{ όπου } n \in \mathbb{N}$$

Ο υπολογισμός για να βρεθεί το κέντρο αυτού του συνόλου γίνεται ως εξής:

$$\mathrm{M(y,x)}=(\frac{y_1+y_2+\cdots+y_n}{n}, \frac{x_1+x_2+\cdots+x_n}{n})$$

Με τον παραπάνω τρόπο γίνεται η εύρεση του σημείου το οποίο βρίσκεται στο κέντρο ενός αντικειμένου. Τέλος μετά τον υπολογισμό του κέντρου, ο αλγόριθμος χρωματίζει ένα πλήθος από pixel με κίτρινο χρώμα ώστε να μπορεί ο χρήστης να δει που ακριβώς βρίσκεται το κέντρο του αντικειμένου στην εικόνα που επιστρέφει στον υπολογιστή.

6.4. Connected Component Labeling (CCL)

Το πρόβλημα το οποίο είχε δημιουργηθεί χρησιμοποιώντας τον αλγόριθμο για τον υπολογισμό του κέντρου όπως αναφέραμε στο κεφάλαιο 6.2. είναι πως στην περίπτωση που υπήρχαν δύο αντικείμενα στην εικόνα, εκ των οποίων και τα δύο ήταν χρώματος κόκκινου τότε ο αλγόριθμος για τον υπολογισμό του κέντρου, υπολόγιζε το κέντρο μεταξύ αυτών των δύο αντικειμένων και όχι το κέντρο κάθε ενός αντικειμένου ξεχωριστά. Αυτό είχε ως αποτέλεσμα το κέντρο το οποίο υπολογιζόταν να είναι λάθος και στο σύνολο του πειράματος σαν αποτέλεσμα ήταν άχρηστο. Αυτό το οποίο υλοποιήθηκε για να αντιμετωπιστεί αυτό το πρόβλημα ήταν ο αλγόριθμος Connected Component Labeling και ο κώδικας του συγκεκριμένου αλγορίθμου δίνεται στο παράρτημα Α.

Ο αλγόριθμος Connected Component Labeling σκανάρει όλη την εικόνα και κατηγοριοποιεί τα pixel σε συγκεκριμένες ομάδες δίνοντάς τους μια συγκεκριμένη ετικέτα. Στον Πίνακα 6-2 βλέπουμε πώς είναι διαχωρισμένα τα pixel σε μια εικόνα βάσει του αλγορίθμου Connected Component Labeling. Στα pixel τα οποία έχουν εντοπιστεί ως κόκκινα δίνεται ο αριθμός 1 και στα υπόλοιπα pixel δίνεται ο αριθμός 0. Έπειτα γίνεται ακόμα ένα σκανάρισμα της εικόνας δίνοντας ετικέτες στις διαφορετικές ομάδες από άσσους που υπήρχαν.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
0	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0
0	0	0	1	1	1	1	0	0	0	1	1	1	1	0	0	0
0	0	1	1	1	1	0	0	0	1	1	1	0	0	1	1	0
0	1	1	1	0	0	1	1	0	0	0	1	1	1	0	0	0
0	0	1	1	0	0	0	0	0	1	1	0	0	0	1	1	0
0	0	0	0	0	0	1	1	1	1	0	0	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Πίνακας 6-2. Κατηγοριοποίηση και διαχωρισμός των κόκκινων pixel από τα υπόλοιπα χρώματα.

Ο συγκεκριμένος αλγόριθμος υλοποιήθηκε στο σύστημα με σκοπό να μπορεί να δίνει ετικέτες σε κάθε ένα σύνολο από κόκκινα pixel που αναπαριστούν κάποιο αντικείμενο. Το κάθε σύνολο έχει μια μοναδική ετικέτα και με αυτό τον τρόπο μπορεί να διαφοροποιηθεί στην εικόνα και να υπολογιστεί το κέντρο του.

Η λειτουργία του αλγορίθμου ακολουθεί τα εξής βήματα:

- Σκανάρισμα της εικόνας
- Εύρεση του πρώτου κόκκινου pixel
- Σύγκριση του pixel που βρέθηκε με το προηγούμενο pixel στον άξονα x και σύγκριση με το προηγούμενο pixel στον άξονα y
- Στην περίπτωση που το pixel στο οποίο βρίσκεται είναι κόκκινο και το προηγούμενο pixel για τον άξονα x και το προηγούμενο για τον άξονα y αντίστοιχα έχει άλλο χρώμα τότε δίνεται μια νέα ετικέτα.
- Στην περίπτωση που το προηγούμενο pixel στον άξονα x ή το προηγούμενο στον άξονα y αντίστοιχα είναι κόκκινο τότε στο pixel στο οποίο βρίσκεται, δίνεται η ίδια ετικέτα με αυτήν που είχε το προηγούμενο pixel.

Με αυτόν τον τρόπο το σύστημα μπορεί να διαχωρίσει τα αντικείμενα που έχουν κόκκινο χρώμα και έπειτα με βάση την ετικέτα που είχε δοθεί σε κάθε ένα από τα αντικείμενα να υπολογίσει το κέντρο τους.

Συμπεράσματα και προβλήματα που αντιμετωπίστηκαν

Με την χρήση των αλγορίθμων που αναφέρθηκαν στο προηγούμενο κεφάλαιο πραγματοποιείται το πείραμα και γίνεται αναγνώριση χρώματος και υπολογισμός του κέντρου του αντικειμένου. Αυτό το οποίο μπορεί να είναι ως επακόλουθο με την χρήση των παραπάνω αλγορίθμων είναι η αναγνώριση του χρώματος του ανθρώπινου δέρματος με σκοπό να γίνει αναγνώριση προσώπου (Face Recognition). Το πλεονέκτημα είναι πως ο αλγόριθμος που υλοποιεί την αναγνώριση του χρώματος καταναλώνει λιγότερους υπολογιστικούς πόρους με αποτέλεσμα και λιγότερη υπολογιστική ισχύ. Στην περίπτωση που αναγνωρίζει στον χώρο το χρώμα του δέρματος τότε μπορεί να ξεκινήσει η διαδικασία αναγνώρισης προσώπου.

Το πρόβλημα που μπορεί να παρουσιαστεί με την χρήση του αλγορίθμου αναγνώρισης χρώματος είναι η περίπτωση του να κάνει λάθος εκτίμηση του χρώματος λόγω εσφαλμένου φωτισμού. Ο φωτισμός είναι σημαντικός παράγοντας στο να μπορεί να γίνει σωστή αναγνώριση χρώματος. Στο πείραμα το πρόβλημα που αντιμετωπίστηκε ήταν η δυσλειτουργία του αλγορίθμου σε περιβάλλον που πηγή φωτός ήταν λάμπες πυρακτώσεως. Οι λάμπες πυρακτώσεως δεν παράγουν λευκό φως, αλλά μια απόχρωση του κόκκινου με αποτέλεσμα το white balance της κάμερας να είναι προσαρμοσμένο σε αυτό. Αυτό είχε ως συνέπεια η αναγνώριση του κόκκινου χρώματος στον χώρο να είναι λανθασμένη γιατί οι ανακλάσεις από τα αντικείμενα είχαν μια απόχρωση του κόκκινου χρώματος και έτσι η κάμερα αναγνώριζε κόκκινο χρώμα σε όλο τον χώρο.

Ο αλγόριθμος για τον υπολογισμό του κέντρου σε συνδυασμό με τον αλγόριθμο Connected Component Labeling μπορεί να χρησιμοποιηθεί με τέτοιο τρόπο ώστε η κάμερα να γνωρίζει σε ποίο σημείο της εικόνας βρίσκεται ένα αντικείμενο που έχει αναγνωριστεί ως κόκκινο. Το κέντρο χρησιμοποιείται ως σημείο αναφοράς για την κάμερα ώστε στην περίπτωση που η κάμερα έχει την ικανότητα να κινηθεί είτε βρισκόμενη πάνω σε ένα ρομπότ, είτε βρισκόμενη πάνω σε έναν ρομποτικό βραχίονα να μπορεί να πλησιάζει αντικείμενα ή ακόμα και να τα πιάνει.

Προτάσεις για περαιτέρω έρευνα

Ο αλγόριθμος Connected Component Labeling ο οποίος κάνει διαχωρισμό τον αντικειμένων που έχουν αναγνωριστεί ως κόκκινα στην εικόνα χρειάζεται να κάνει πολλές επαναλήψεις σκαναρίσματος στην εικόνα ώστε να μπορεί να διαχωρίζει όλα τα αντικείμενα. Αυτό έχει ως αποτέλεσμα την δημιουργία καθυστέρησης στην επεξεργασία της εικόνας και την αποστολή της στον τελικό χρήστη. Μια βελτιστοποίηση αυτού του αλγορίθμου ώστε να μειωθούν στο σύνολο τους οι επαναλήψεις θα επέτρεπε την ταχύτερη επεξεργασία και κατ' επέκταση την λιγότερη χρήση υπολογιστικών πόρων. Έχοντας λιγότερη χρήση υπολογιστικών πόρων το σύστημα αποκτάει χαμηλότερη κατανάλωση και θα μπορούσε η τροφοδοσία του να αποτελείται από μια μπαταρία.

Όπως έχει αναφερθεί η επιστροφή της εικόνας από την κάμερα στον υπολογιστή γίνεται μέσω μιας θύρας Ethernet. Αυτό έχει ως αποτέλεσμα την ανάγκη για επιπλέον καλωδιώσεις στον χώρο στον οποίο βρίσκεται η κάμερα. Επίσης δημιουργούνται περιορισμοί στην τοποθέτηση της κάμερας εφόσον πρέπει να έχει επικοινωνία με τον υπολογιστή μέσω καλωδίου. Αυτό το οποίο μπορεί να γίνει είναι η ενσωμάτωση ενός chip το οποίο να υλοποιεί το πρωτόκολλο ZigBee (IEEE 802.15.4). Με αυτό τον τρόπο θα μπορούν να αποστέλλονται οι συντεταγμένες του κέντρου ενός αντικειμένου με ασύρματη μετάδοση στον υπολογιστή και όχι με την χρήση καλωδίου, επιτρέποντας στον χρήστη να γνωρίζει την τοποθεσία του αντικειμένου, που έχει αναγνωριστεί ως κόκκινο στην εικόνα.

Πίνακας συντομογραφιών

1) ABS: Anti-lock Breaking System
2) AHB: Advanced High-performance Bus
3) API: Application Programming Interface
4) ASIC: Application Specific Integrated Circuit
5) ASSP: Application Specific Standard Product
6) ATM: Automate Teller Machine
7) BiCMOS: Bipolar Complementary Metal Oxide Semiconductor
8) BIT: Binary Digit
9) CAD: Computer Aided Design
10) CCL: Connected Component Labeling
11) CISC: Complex Instruction Set Computer
12) CMOS: Complementary Metal Oxide Semiconductor
13) CPU: Central Processing Unit
14) DEC: Digital Equipment Corporation
15) DLL: Dynamic-Link Library
16) DRAM: Dynamic Random Access Memory
17) DSP: Digital Signal Processing
18) EEPROM: Electrically Erasable Programmable Read Only Memory
19) EGCS: Experimental/Enhanced GNU Compiler System
20) FET: Field Effect Transistor
21) FPGA: Field Programmable Gate Array
22) FTP: File Transfer Protocol
23) GCC: GNU Compiler Collection
24) GNU: GNU's Not UNIX
25) GPS: Global Positioning System
26) HDL: Hardware Description Language
27) HTML: HyperText Markup Language
28) I^2C: Inter Integrated Circuit

29) IBM: International Business Machines Corporation
30) IP: Internet Protocol address
31) LSI: Large Scale Integration
32) MAC: Media Access Control address
33) MIPS: Microprocessor without Interlocked Pipeline Stages
34) MMU: Memory Management Unit
35) MOSFET: Metal Oxide Semiconductor Field Effect Transistor
36) μC: Microcontroller
37) μP: Microprocessor
38) NMOS: N-type Metal Oxide Semiconductor Logic
39) OAR: On-Line Application Research
40) PCB: Printed Circuit Board
41) PCI: Peripheral Component Interconnect
42) PLD: Programmable Logic Device
43) POSIX: Portable Operating System Interface
44) RAM: Random Access Memory
45) RISK: Reduced Instruction Set Computer
46) ROM: Read Only Memory
47) RTEMS: Real Time Executive for Multiprocessor Systems
48) SCL: Serial Clock
49) SDA: Serial Data
50) SOC: System on Chip
51) SPARC: Scalable Processor Architecture
52) SRAM: Static Random Access Memory
53) SUN: Stanford University Network
54) TCP: Transmission Control Protocol
55) TTL: Transistor Transistor Logic
56) UART: Universal Asynchronous Receiver/Transmitter
57) VAX: Virtual Address eXtension
58) VLSI: Very Large Scale Integration

Βιβλιογραφία

1. Michael Barr, Anthony Massa (October 01, 2006). "*Programming Embedded Systems*". O'Reilly.
2. Edward L. Lamie (2005). "*Real-Time Embedded Multithreading: Using ThreadX and ARM*". CMP Books.
3. Christian Baumann, University of Innsbruck (January 13, 2010). "*Field Programmable Gate Array (FPGA)*". Summary paper for the seminar "Embedded System Architecture".
4. Stephen Brown, Zvonko Vranesic (2003). "*Fundamentals of Digital Logic with Verilog design*". McGraw-Hill Companies, Inc.
5. Peter J. Ashenden (2010). "Ψηφιακή σχεδίαση. Ενσωματωμένα συστήματα με VHDL". Elsevier.
6. Andrew S. Tanenbaum (2007). "Η αρχιτεκτονιή των υπολογιστών. Μια δομημένη προσέγγιση". Κλειδάριθμος.
7. Steave Heath (1995). "*Microprocessor Architecture: RISC, CISC and DSP 2^nd^ Edition*". Butterworth-Heinemann Ltd.
8. Jason G. Tong, Ian D. L. Anderson and Mohammed A. S. Khalid (2006). "Soft-Core Processors for Embedded Systems". The 18th International Confernece on Microelectronics (ICM)
9. Κ. Λάζος, Π. Κατσαρός, Ζ. Καραϊσκος (2004). "*Μεταγλωττιστές γλωσσών προγραμματισμού: θεωρία και πράξη*". Θεσσαλονίκη.
10. Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman (2007). "*Compilers: Principles, Techniques, & Tools 2nd Edition*". Pearson Education, Inc.
11. Neil Weste, Kamran Eshraghian (1996). "*Σχεδίαση Ολοκληρωμένων Κυκλωμάτων CMOS VLSI*". Α. Παπασωτηρίου & ΣΙΑ Ο.Ε..
12. Brian Gough Forward by Richard M. Stallman (March 2004). "An introduction to GCC for the GNU compilers gcc and g++". Network Theory Ltd.

Ηλεκτρονικές πηγές

(1) http://www.xilinx.com/fpga/asic.htm (Προσπελάστηκε στις 29 Μαρτίου 2013)

(2) http://www.electronicproducts.com/Digital_ICs/Standard_and_Programmable_Logic/The_evolution_of_FPGA_coprocessing.aspx (Προσπελάστηκε στις 29 Μαρτίου 2013)

(3) http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.faqs/ka3839.html (Προσπελάστηκε στις 4 Απριλίου 2013)

(4) http://www.esacademy.com/en/library/technical-articles-and-documents/miscellaneous/i2c-bus/general-introduction/history-of-the-i2c-bus.html (Προσπελάστηκε στις 4 Απριλίου 2013)

(5) http://www.totalphase.com/support/kb/10037/ (Προσπελάστηκε στις 8 Απριλίου 2013)

(6) http://www.classic.nxp.com/acrobat_download2/literature/9398/39340011.pdf (Προσπελάστηκε στις 8 Απριλίου 2013)

(7) https://www.sparkfun.com/datasheets/Sensors/Imaging/TCM8240MD.pdf (Προσπελάστηκε στις 8 Απριλίου 2013)

(8) http://www.xilinx.com/support/documentation/data_sheets/ds099.pdf (Προσπελάστηκε στις 10 Απριλίου 2013)

(9) http://www.weblearn.hs-bremen.de/risse/RST/WS04/Leon2/Quellen/leon2-1.0.24-xst.pdf(Προσπελάστηκε στις 10 Απριλίου 2013)

(10) http://www.rtems.org/onlinedocs/releases/rtemsdocs-4.6.5/share/rtems/pdf/started.pdf(Προσπελάστηκε στις 11 Απριλίου 2013)

(11) http://sourceware.org/ml/crossgcc/2005-08/msg00114/l-cross-ltr.pdf(Προσπελάστηκε στις 11 Απριλίου 2013)

(12) http://dide.flo.sch.gr/Plinet/Tutorials/Tutorials-C-Part-1.html (Προσπελάστηκε στις 13 Απριλίου 2013)

(13) http://www.ti.com/lit/an/snoa099/snoa099.pdf (Προσπελάστηκε στις 13 Απριλίου 2013)

(14) http://wiki.rtems.org/wiki/index.php/Main_Page (Προσπελάστηκε στις 13 Απριλίου 2013)

(15) http://www.rtems.org/ (Προσπελάστηκε στις 15 Απριλίου 2013)

(16) http://seg.ece.upatras.gr/Courses/bpl/cygwinUsersGuide.pdf (Προσπελάστηκε στις 15 Απριλίου 2013)

(17) http://makolas.blogspot.gr/2010/10/to-rgb-scratch.html (Προσπελάστηκε στις 15 Απριλίου 2013)

(18) http://www.teiath.gr/userfiles/eadsa_web_admin/lessons/c_semester/2012-13-arxitektonikiFotismosXoroiErgasias-theoryFotismos-Klonizakis.pdf (Προσπελάστηκε στις 16 Απριλίου 2013)

(19) http://www.cold.org.gr/library/Downloads/docs/%CE%9C%CE%95%CE%A4%CE%A1%CE%97%CE%A3%CE%97%20%CE%A7%CE%A1%CE%A9%CE%9C%CE%91%CE%A4%CE%9F%CE%A3%20-%20%CE%A7%CE%A1%CE%A9%CE%9C%CE%91%CE%A4%CE%9F%CE%9C%CE%95%CE%A4%CE%A1%CE%99%CE%91.pdf (Προσπελάστηκε στις 16 Απριλίου 2013)

(20) http://www.diaplous.com/sitebuildercontent/sitebuilderfiles/dpls_ipb05_wbest_01_00.pdf (Προσπελάστηκε στις 16 Απριλίου 2013)

(21) http://www.csie.ntu.edu.tw/~fuh/personal/ANovelAutomaticWhiteBalanceMethodforDigital.pdf (Προσπελάστηκε στις 16 Απριλίου 2013)

(22) http://www.cambridgeincolour.com/tutorials/white-balance.htm (Προσπελάστηκε στις 17 Απριλίου 2013)

(23) http://www.cambridgeincolour.com/tutorials/camera-sensors.htm (Προσπελάστηκε στις 17 Απριλίου 2013)

Παράρτημα A

Στις ακόλουθες σελίδες περιέχεται ο κώδικας σε C για την υλοποίηση των τριών αλγορίθμων: του αλγορίθμου για την αναγνώριση χρώματος, του αλγορίθμου που υπολογίζει το κέντρο των αντικειμένων και του αλγορίθμου Connected Component Labeling για τον υπολογισμό του κέντρου σε περισσότερα από ένα αντικείμενα. Επίσης περιέχεται ο κώδικας που υλοποιεί την πράξη της διαίρεσης στο σύστημα. Ο κώδικας αναγνώρισης κόκκινου χρώματος και Connected Component Labeling είναι μέρος από τον συνολικό κώδικα που εκτελείται στην FPGA.

Κώδικας αναγνώρισης κόκκινου χρώματος και Connected Component Labeling

```
void make_smooth() {
    int x;
            int y;
            unsigned long int pos2;
            unsigned long int wf, wt, wf2, wf3;
    unsigned long int array[255][255];
    int gr,gg,gb;
            unsigned long int count_x = 0;
            unsigned long int count_y = 0;
            unsigned long int sumup = 1;
            unsigned long int mean_x = 0;
            unsigned long int mean_y = 0;
            unsigned long int mean_x_1 = 0;
            unsigned long int mean_y_1 = 0;
            int i = 0;
            int j = 0;
            int percentageRed = 0;
            int percentageGreen = 0;
            int percentageBlue = 0;
            int foreground = 1;
            int background = 0;
            int NewLabel = 5; //starting from 5 for security reasons
            int lx = 0;
            int back = 0; //one pixel back
            int up = 0;//one pixel up
            int back_diagonal_up = 0;
```

```
            int back_diagonal_down = 0;
            int down = 0;
            int front = 0;
            int front_diagonal_up = 0;
            int front_diagonal_down = 0;
            unsigned long int count_x_sec = 0;
            unsigned long int count_y_sec = 0;
            unsigned long int sumup_sec = 0;
            unsigned long int mean_x_sec = 0;
            unsigned long int mean_y_sec = 0;
            unsigned long int mean_y_1_2 = 0;
            unsigned long int mean_x_1_2 = 0;
            int k;

/* -------------------- Color identification --------------- */
    for (y=0; y<255;y++)
    {
            pos2 = FRAME_POS | (  y     <<AL_IMG_WIDTH_PWR);
        for (x=0;x<255;x++)
          {
        wf2  = IMG(pos2);
            gr = B3(wf2);   gg = B2(wf2);  gb = B0(wf2);
            int sum = gr + gg + gb;
            int red = gr*100;
            int green = gg*100;
            int blue = gb*100;
            percentageRed = (int) division(red, sum);
```

```
percentageGreen = (int) division(green, sum);
percentageBlue = (int) division(blue, sum);
if (percentageRed > percentageGreen && percentageRed > percentageBlue) {
gr=0xff;
gg=0xff;
gb=0x00;
array[x][y] = 1;
/* front = 1;
front_diagonal_up = 1;
front_diagonal_down = 1; */
}       else {
array[x][y] = 0;
}

IMG(pos2) = (gr<<24) | (gg<<16) | (gb);
pos2+=4;
} //x for loop for the image
} // y for loop for the image

/* ------------------- Conected component labeling -------------------- */
for (y=1; y<255;y++)
{
for (x=1;x<255;x++)
{
if (array[x][y] == foreground)
{
back = array[x-1][y]; //labeled rows
```

```
up = array[x][y-1]; //labeled columns
if(back == background && up == background)
{
        NewLabel++;
        lx = NewLabel;
}
else if((back != up)&&(back != background)&&(up != background))
{
        lx = up;
}
else if(up != background)
{
        lx = up;
}
else if(back != background)
{
        lx = back;
}
array[x][y] = lx;
}
        } // x for loop for the matrix
        } // y for loop for the matrix
for ( k = 1; k <= NewLabel-6; k++ ) // -6 is used because we start counting
NewLabel variable from 5
        {
        for ( y = 1; y < 255; y++ )
                {
```

```
            for ( x = 1; x < 255; x++ )
                    {
        if( array[x][y] == k)
        {
        count_x = count_x + x;
        count_y = count_y + y;
        sumup++;
        }
                    }
            }
            calculate(mean_x, mean_y, sumup);
        }

}
```

Κώδικας για τον υπολογισμό του κέντρου

```
/* calculating the center of the object (for loop is used to magnify
the targeted area */
#include "DivisionDedousis.c"

void calculate(int count_x, int count_y, int sumup){
                unsigned long int mean_x = 0;
                unsigned long int mean_y = 0;
                unsigned long int wf;
                unsigned long int mean_x_1 = 0;
                unsigned long int mean_y_1 = 0;
                int i = 0;
                int j = 0;
                unsigned long int pos2;

        mean_x = (int) division(count_x, sumup);
        mean_y = (int) division(count_y, sumup);
        wf = 0xFFFF0000;
        for (i = 1; i<=10; i++){
        mean_y_1 = mean_y + i;
        for (j = 1; j<=10; j++){
        mean_x_1 =  mean_x + j;
        pos2 = FRAME_POS | ( mean_y_1<<AL_IMG_WIDTH_PWR) | (mean_x_1<<2);
        IMG(pos2) = wf;
        }
        }
        }
```

Κώδικας για την υλοποίηση της διαίρεσης

```
int dividend, divisor, remainder;
int division(int tempdividend, int tempdivisor) {

  int quotient = 1;

    if (tempdivisor == 0)
    return 0;

  while (tempdivisor <= tempdividend) {
    /* Here divisor < dividend, therefore left shift (multiply by 2)
      divisor and quotient */
    tempdivisor = tempdivisor << 1;
    quotient = quotient << 1;
  }

  /* We have reached the point where divisor > dividend,
  therefore divide divisor and quotient by two */
  tempdivisor = tempdivisor >> 1;
  quotient = quotient >> 1;

  /* Call division recursively for the difference to get the
  exact quotient */
  quotient = quotient + division(tempdividend - tempdivisor, divisor);
```

```
    return quotient;
}
```