Ελληνικό Μεσογειακό Πανεπιστήμιο

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Πτυχιακή εργασία

ΕΛΕΓΧΟΜΕΝΗ ΕΚΤΕΛΕΣΗ ΕΦΑΡΜΟΓΗΣ ΠΡΑΓΜΑΤΙΚΟΥ ΧΡΟΝΟΥ ΣΕ ΕΝΣΩΜΑΤΩΜΕΝΟ ΣΥΣΤΗΜΑ

Graduate thesis

CONTROLLED EXECUTION OF REAL TIME APPLICATION IN EMBEDDED SYSTEM

Κρητικάκης Εμμανουήλ: Α.Μ. 4416

Επιβλέπων εκπαιδευτικός : Κορνάρος Γεώργιος

Ημερομηνία παρουσίασης :23/9/20

Ευχαριστίες

Νιώθω την ανάγκη να ευχαριστήσω τον επιβλέποντα καθηγητή μου κύριο Γεώργιο Κορνάρο που μου έδωσε την ευκαιρία να ασχοληθώ με ένα θέμα που μου κέντρισε το ενδιαφέρον πάρα πολύ, καθώς και για την καθοδήγηση και την εμπιστοσύνη που μου έδειξε κατά το διάστημα εκπόνηση της πτυχιακής μου εργασίας.

Επίσης θέλω να ευχαριστήσω την οικογένεια και τους φίλους μου που με την πολύτιμη για μένα στήριξη τους έκαναν την διαδικασία εκπόνησης της πτυχιακής μου εργασίας πολύ ευκολότερη.

Σύνοψη

Τα τελευταία χρόνια τα ενσωματωμένα συστήματα έχουν ενταχθεί για τα καλά στην καθημερινότητα του σύγχρονου ανθρώπου παρόλο που αρχικά είχαν εμφανιστεί σαν συσκευές χαμηλών επιδόσεων σήμερα χρησιμοποιούνται σε αρκετά πολύπλοκες συσκευές με αυξημένες απαιτήσεις σε επιδόσεις συστήματα όμως τις περισσότερες φορές διαθέτουν περιορισμένους πόρους γι αυτό είναι απαραίτητη πολλές φορές η χρήση κάποιον πρακτικών που αυξάνουν τις επιδόσεις του συστήματος αλλά και χρησιμοποιούν όσο το δυνατό πιο αποδοτικά τους πόρους του συστήματος.

Στόχος της πτυχιακής μου είναι η ελεγχόμενη εκτέλεση μιας εφαρμογής πραγματικού χρόνου με στόχο την βέλτιστη απόδοση της εφαρμογής αλλά και του συστήματος σε σχέση με την κατανάλωση ενέργειας είναι απαραίτητη η εφαρμογή τεχνικών διαχείρισης πόρων του συστήματος. Στόχος της πτυχιακή είναι να αναπτυχθούν τεχνικές ελεγχόμενης πρόσβασης στην μνήμη ή επεξεργαστικής ισχύς.

Πίνακας περιεχομένων

Περιεχόμενα

Κατάλογος εικόνων και γραφημάτων

Abstract

In recent years, embedded systems have been integrated into the daily life of modern man, although they originally appeared as low-performance devices, today they are used in quite complex devices with increased performance requirements. But we are talking about systems with limited resources so the developers must use these resources wisely.

The aim of this undergraduate thesis is the controlled execution of a real-time application in order to optimize the performance of the application and the system in relation to energy consumption, it is necessary to apply system resource management techniques. The aim of the thesis is to develop techniques of controlled access to memory or processing power.

1. Εισαγωγή

Τα τελευταία χρόνια τα ενσωματωμένα συστήματα έχουν ενταχθεί για τα καλά στην καθημερινότητα του σύγχρονου ανθρώπου παρόλο που αρχικά είχαν εμφανιστεί σαν συσκευές χαμηλών επιδόσεων σήμερα χρησιμοποιούνται σε αρκετά πολύπλοκες συσκευές με αυξημένες απαιτήσεις σε επιδόσεις συστήματα όμως τις περισσότερες φορές διαθέτουν περιορισμένους πόρους γι αυτό είναι απαραίτητη πολλές φορές η χρήση κάποιον πρακτικών που αυξάνουν τις επιδόσεις του συστήματος αλλά και χρησιμοποιούν όσο το δυνατό πιο αποδοτικά τους πόρους του συστήματος.

Πολλές φορές όμως υπάρχει η ανάγκη το σύστημα να ανταποκρίνεται σε συγκεκριμένα χρονικά πλαίσια γι αυτό και η εκτέλεση εφαρμογών σε πραγματικό χρόνο χρησιμοποιείτε κατά κόρον στα ενσωματωμένα συστήματα και μάλιστα σε πολλούς τομείς όπως σε η αυτοκινητοβιομηχανία [1] [2], η ιατρική/γενετική [3], δίκτυα/επικοινωνίες [4] και πολλοί ακόμα τομείς τις σύγχρονης βιομηχανίας. Υπάρχουν πολλές μέθοδοι για να γίνει εκτέλεση εφαρμογών σε πραγματικό χρόνο τις οποίες και θα αναλύσουμε στην πτυχιακή αυτή(πχ real time scheduling,approximate computing,accelerators).

Στόχος της πτυχιακής είναι η ελεγχόμενη εκτέλεση μιας εφαρμογής πραγματικού χρόνου με στόχο την βέλτιστη απόδοση της εφαρμογής αλλά και του συστήματος σε σχέση με την κατανάλωση ενέργειας, είναι απαραίτητη η εφαρμογή τεχνικών διαχείρισης πόρων του συστήματος. Στόχος της πτυχιακή είναι να αναπτυχθούν τεχνικές ελεγχόμενης πρόσβασης στην μνήμη ή επεξεργαστικής ισχύς.

Ποιο συγκεκριμένα θα γίνει σύγκριση της τεχνικής fixed point arithmetic για επεξεργασία εικόνας με μειωμένη ακρίβεια με την floating point arithmetic για επεξεργασία εικόνας με πλήρη ακρίβεια. Επιπρόσθετα θα χρησιμοποιηθούν τεχνικές optimization από τον compiler της γλώσσας c. Τέλος θα παρουσιαστεί μια εφαρμογή που ακλουθώντας ντετερμινιστικά πλαίσια υλοποιεί μετατροπή εικόνας από rag σε grayscale σε συνδυασμό πλήρους και μειωμένης ακρίβειας.

Στόχος είναι να παρουσιαστούν οι βελτιστοποιήσεις στην απόδοση του συστήματος που μπορούν να επιτευχθούν με χρήση της τεχνικής fixed point arithmetic σε συνδυασμό με της αυτοματοποιημένες τεχνικές για optimization από τον compiler.

Η πλατφόρμα που χρησιμοποιούμε είναι ένα zeadboard που βασίζεται στην αρχιτεκτονική του zynq 7000 SoC με την βοήθεια του προγράμματος Vivado περιγράφουμε την αρχιτεκτονική του συστήματός μας και με το πρόγραμμα SDK της XILINX προγραμματίζουμε κατευθείαν σε baremetal(χωρίς την χρήση λειτουργικού συστήματος) σε γλωσσά προγραμματισμού C το σύστημά μας.

1.2 Αφορμή για τη διεξαγωγή της εργασίας

Στις μέρες μας τα ενσωματωμένα συστήματα και οι εφαρμογές τους σε πραγματικό χρόνο έχουν πρωταγωνιστικόο ρόλο στη ζωή του σύγχρονου ανθρώπου ,τα συστήματα αυτά σε αντίθεση με τους απλούς προσωπικούς μας υπολογιστές βρίσκονται σχεδόν παντού , από οικιακές συσκευές μέχρι αεροσκάφη.

Λογικό είναι διαρκώς οι ανάγκες τους σε απόδοση αλλά και χαμηλότερη κατανάλωση ενέργειας να μεγαλώνει γι αυτό υπάρχει η ανάγκη τεχνικών που εκμεταλλεύονται τους πόρους το συστήματος αποδοτικά.

Σε συνδυασμό της παραπάνω άποψης με το γεγονός ότι στην βιβλιογραφία και στο διαδίκτυο οι πληροφορίες σχετικά με την fixed point αριθμητική είναι περιορισμένες θεωρήθηκε ενδιαφέρον να υλοποιηθεί η εφαρμογή αυτή.

1.3 Συναφής εργασίες

Παρόμοια δουλειά με αυτή που παρουσιάζεται σε αυτή την εργασία είχε παρουσιαστεί στο διεθνές συνέδριο επεξεργασίας εικόνας τον Οκτώβριο του 2006 στην Ατλάντα των ΗΠΑ από το ινστιτούτο ηλεκτρολόγων και ηλεκτρονικών μηχανικών (IEEE) με συγγραφείς τους Christoph H. Lampert και Oliver Wirjadi οι οποίοι εφαρμόζουν αναδρομικό Γκαουσιανό φιλτράρισμα σε ισοτροπικό και ανισοτροπικό φιλτράρισμα εικόνας χρησιμοποιώντας ένα μη ορθογώνιο σχήμα διαχωρισμού του φίλτρου χρησιμοποιώντας fixed point αριθμητική [5].

Ο λόγος που το κάνουν αυτό είναι διότι κάποια συστήματα δεν διαθέτουν floating point unit(FPU) και για να αυξήσουν την απόδοση του συστήματος.

1.4 Επιδιώξεις και Στόχοι

Στόχοι αυτής της εργασίας είναι:

- Υλοποίηση τεχνικής fixed point arithmetic
- Σύγκριση του χρόνου των πράξεων μεταξύ fixed point με floating point arithmetic
- Επεξεργασία εικόνας με πλήρη και μειωμένη ακρίβεια

- Επεξεργασία εικόνας ακλουθώντας ντετερμινιστικά πλαίσια χρόνου
- Εξαγωγή χρήσιμων αποτελεσμάτων

1.5 Δομή Εργασίας

Κεφάλαιο 1:Εισαγωγή

Στο κεφάλαιο 1 γίνεται μια μικρή εισαγωγή περίληψη αυτής πτυχιακής εργασίας και αναφέρονται επιγραμματικά οι στόχοι που θέλουμε να επιτύχουμε σε αυτή την εργασία.

Κεφάλαιο 2:Εισαγωγή στα ενσωματωμένα συστήματα και στην real time επεξεργασία

Κεφάλαιο 3:Approximate calculations

Κεφάλαιο 4: Αναπαράσταση πραγματικών αριθμών και fixed point αριθμητική

Κεφάλαιο 5:Εισαγωγή στην επεξεργασία εικόνας και την μετατροπή από έγχρωμη εικόνα σε ασπρόμαυρη

Κεφάλαιο 6:Περιγραφή υλικού

Κεφάλαιο 7:Optimization

Κεφάλαιο 8:Περιγραφή λογισμικού

Κεφάλαιο:9:Αποτελέσματα – Μετρήσεις

Κεφάλαιο 10:Συμπεράσματα και μελλοντικές επεκτάσεις

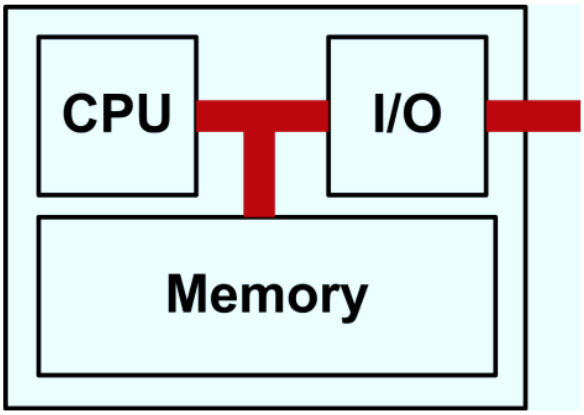
2.Εισαγωγή στα ενσωματωμένα συστήματα

[6]Συνήθως είναι πολύ εύκολο για έναν άνθρωπο να αναγνωρίσει ένα υπολογιστικό σύστημα είτε αυτό είναι ένας φορητός υπολογιστής, ένας διακομιστής, ένας επιτραπέζιος υπολογιστής. Υπάρχει όμως και ένας τύπος υπολογιστικών συστημάτων που είναι αθέατος γι αυτούς που δεν το γνωρίζουν αυτός είναι ο κόσμος των ενσωματωμένων συστημάτων κόσμος αυτός είναι πολύ μεγαλύτερος από τον κόσμο τον τυπικών υπολογιστικών συστημάτων, στην αγορά κάθε χρόνο παράγονται δισεκατομμύρια μονάδες ενσωματωμένων συστημάτων και μόνο κάποια εκατομμύρια για της τυπικές υπολογιστικές μονάδες. Τι είναι όμως τα ενσωματωμένα συστήματα;

Σύμφωνα με έναν απλό ορισμό ενσωματωμένο σύστημα είναι εκείνο που διαθέτει έναν προγραμματιζόμενο επεξεργαστή ο οποίος δεν είναι γενικού σκοπού. Υπάρχουν φυσικά και άλλοι περισσότερο αναλυτικοί ορισμοί για το τι είναι ένα ενσωματωμένο σύστημα. Είναι όμως λογικό ότι είναι δύσκολο να βρεθεί ένας ορισμός που να καλύπτει πλήρως όλα τα ενσωματωμένα συστήματα γιατί αυτά είναι πάρα πολλά και εκτελούν διαφορετικές λειτουργίες το κάθε ένα

Η αρχιτεκτονική ενός ενσωματωμένου συστήματος είναι η ίδια με εκείνη των τυπικών υπολογιστικών συστημάτων, διαθέτουν δηλαδή επεξεργαστή μνήμη και διεπαφές εισόδου

εξόδου όμως η κύρια διαφορά με τα κοινά υπολογιστικά συστήματα είναι ότι χρησιμοποιούν άλλες κατηγορίες επεξεργαστών (συνήθως με χαμηλότερες δυνατότητες) που ονομάζονται μικροεπεξεργαστές επίσης συνήθως έχουν περισσότερες διασυνδέσεις εισόδου εξόδου και περιορισμένους πόρους.Η ανάγκες των ενσωματωμένων συστημάτων οδήγησαν στην χρήση αυτής της αρχιτεκτονικής



Εικόνα 1-Αρχιτεκτονική Ενσωματωμένων

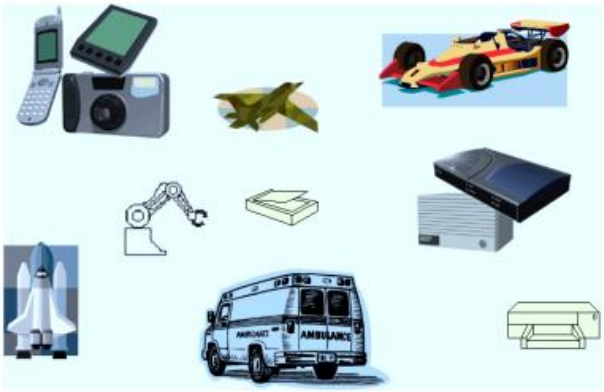
2.1 Ενσωματωμένα συστήματα στην καθημερινότητα

Το γεγονός ότι τα ενσωματωμένα συστήματα συνδυάζουν υλικό υψηλών επιδόσεων σε αρκετές περιπτώσεις μαζί με ειδικευμένο λογισμικό έχει οδηγήσει στην ραγδαία εξάπλωση της σε ολόκληρη την αγορά και είναι αξιοσημείωτο το γεγονός ότι σε ένα σπίτι σήμερα συναντάμε 30-50 ενσωματωμένα συστήματα σε ψηφιακές συσκευές και μόνο 1-2 φορητούς υπολογιστές.

Μπορούμε λοιπών να καταλάβουμε ότι τα ενσωματωμένα συστήματα είναι αναπόσπαστο μέρος της καθημερινότητας του σύγχρονου ανθρώπου, ακολουθούν παραδείγματα χρήσης των ενσωματωμένων συστημάτων στην καθημερινότητα :

- Τηλεόραση και βίντεο: Έχουν ενσωματωμένους Επεξεργαστές για να ελέγχουν την εικόνα, να ρυθμίζουν τα κανάλια, να εκτυπώνουν μηνύματα πάνω στην εικόνα, να ενεργοποιούν και να απενεργοποιούν κυκλώματα της τηλεόρασης.
- Κινητά και ασύρματα τηλέφωνα: Έχουν ενσωματωμένους Επεξεργαστές για να διεκπεραιώνουν τις διαδικασίες κωδικοποίησης αποκωδικοποίησης της φωνής, να υλοποιούν το πρωτόκολλο της επικοινωνίας, να εκτελούν πολύπλοκες διεργασίες του τηλεφώνου.
- Οχήματα : Έχουν ενσωματωμένους Επεξεργαστές είτε για βοηθητικές ενδείξεις, είτε για την ασφάλεια των επιβατών, είτε για την βελτίωση των συνθηκών οδήγησης

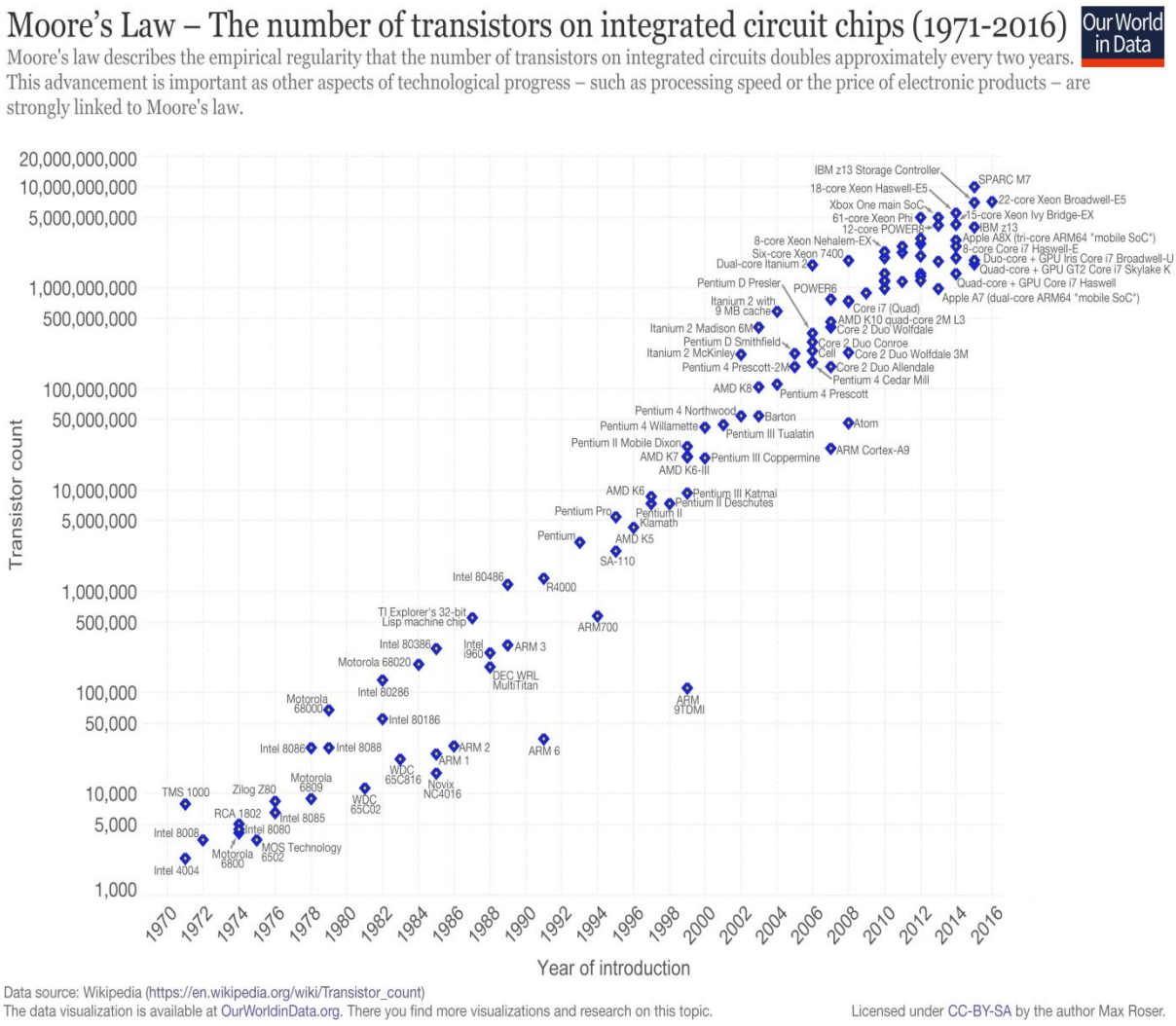
Αυτά είναι μερικά από τα πολλά παραδείγματα ενσωματωμένων συστημάτων που υπάρχουν στον κόσμο.



Εικόνα 2-Παραδείγματα ενσωματωμένων συστημάτων

2.2 Ανάγκη για αυξημένη απόδοση

Μέχρι τώρα η αύξηση στην απόδοση των ενσωματωμένων αλλά και των τυπικών υπολογιστικών συστημάτων πετυχαίνονται με αύξηση του αριθμού των transistors στα κυκλώματα του επεξεργαστή αυτό έχει αρχίσει να σταματάει τα τελευταία χρόνια διαψεύδοντας τον νόμο του Moore ο οποίος είχε δηλώσει ότι τα transistors σε ένα ολοκληρωμένο κύκλωμα θα διπλασιάζονται κάθε δύο χρόνια. Πλέον η αύξηση της απόδοσης επιτυγχάνεται με την χρήση παράλληλων αρχιτεκτονικών των επεξεργαστών, την χρήση παράλληλων αλγορίθμων από τον προγραμματιστή, άλλα και με την χρήση τεχνικών που αυξάνουν την απόδοση του συστήματός μας τέτοιου είδους τεχνικές θα εφαρμόσουμε σε αυτήν την εργασία.



Εικόνα 3-Νόμος του Moore

2.3 Εκτέλεση εφαρμογών πραγματικού χρόνου σε ενσωματωμένα συστήματα(real time execution in embedded systems)

Στην επιστήμη υπολογιστών εκτέλεση σε πραγματικό χρόνο (**Real time computing** (**RTC**) **or reactive computing**)ονομάζεται ο όρος που χρησιμοποιείτε για συστήματα υλικού και λογισμικού που υπόκεινται σε "περιορισμό σε πραγματικό χρόνο" για παράδειγμα συμβάν σε απόκριση συστήματος(system response). Τα προγράμματα σε πραγματικό χρόνο πρέπει να εγγυώνται την απόκριση εντός συγκεκριμένων χρονικών περιορισμών, που συχνά αναφέρονται ως "προθεσμίες(deadlines)".

Οι αποκρίσεις(responses) σε πραγματικό χρόνο συχνά θεωρείται ότι είναι της τάξης των χιλιοστών του δευτερολέπτου και μερικές φορές των μικρο-δευτερολέπτων. Ένα σύστημα που δεν καθορίζεται ως λειτουργικό σε πραγματικό χρόνο δεν μπορεί συνήθως να *εγγυηθεί* μια απόκριση εντός οποιουδήποτε χρονικού πλαισίου, αν και μπορεί να δοθούν *τυπικοί* ή *αναμενόμενοι* χρόνοι απόκρισης. Η επεξεργασία σε πραγματικό χρόνο *αποτυγχάνει* εάν δεν ολοκληρωθεί εντός καθορισμένης προθεσμίας σε σχέση με ένα

συμβάν. οι προθεσμίες πρέπει πάντα να τηρούνται, ανεξάρτητα από το φορτίο του συστήματος . [7]

2.4 Κριτήρια για υπολογισμό σε πραγματικό χρόνο

Ένα σύστημα λέγεται ότι είναι σε *πραγματικό χρόνο* εάν η απόλυτη ορθότητα μιας λειτουργίας εξαρτάται όχι μόνο από τη λογική του ορθότητα, αλλά και από τον χρόνο στον οποίο εκτελείται. Τα συστήματα σε πραγματικό χρόνο, καθώς και οι προθεσμίες τους, ταξινομούνται ανάλογα με της συνέπειες που προκύπτουν από την έλλειψη μιας προθεσμίας σε:

- *Hard* - η έλλειψη προθεσμίας είναι μια συνολική αποτυχία του συστήματος.
- *Firm* - οι σπάνιες παραλείψεις προθεσμίας είναι ανεκτές, αλλά ενδέχεται να υποβαθμίσουν την ποιότητα εξυπηρέτησης του συστήματος. Η χρησιμότητα ενός αποτελέσματος είναι μηδέν μετά την προθεσμία του
- *Soft* - η χρησιμότητα ενός αποτελέσματος υποβαθμίζεται μετά την προθεσμία του, υποβαθμίζοντας έτσι την ποιότητα υπηρεσίας του συστήματος.

Έτσι, ο στόχος ενός hard real time συστήματος είναι να διασφαλίσει ότι τηρούνται όλες οι προθεσμίες, αλλά για τα soft real time συστήματα ο στόχος γίνεται να πληροί ένα συγκεκριμένο υποσύνολο προθεσμιών προκειμένου να βελτιστοποιηθούν ορισμένα ειδικά κριτήρια για την εφαρμογή. Τα συγκεκριμένα κριτήρια που βελτιστοποιούνται εξαρτώνται από την εφαρμογή, αλλά μερικά τυπικά παραδείγματα περιλαμβάνουν τη μεγιστοποίηση του αριθμού των προθεσμιών που πληρούνται, την ελαχιστοποίηση της καθυστέρησης των εργασιών και τη μεγιστοποίηση του αριθμού των εργασιών υψηλής προτεραιότητας που πληρούν τις προθεσμίες τους.

Ο ορισμός των firm real time συστημάτων είναι λίγο πιο νεφελώδης και ορισμένες ταξινομήσεις δεν τα περιλαμβάνουν, διακρίνοντας μόνο σκληρά και μαλακά συστήματα πραγματικού χρόνου(hard και soft real time systems).

Τα hard real time συστήματα σε χρησιμοποιούνται όταν είναι επιτακτική ανάγκη να αντιδράσει ένα συμβάν εντός αυστηρής προθεσμίας. Τέτοιες ισχυρές εγγυήσεις απαιτούνται για συστήματα για τα οποία η μη αντίδραση σε ένα ορισμένο χρονικό διάστημα θα προκαλούσε μεγάλη απώλεια με κάποιο τρόπο, ειδικά βλάπτοντας το περιβάλλον φυσικά ή απειλώντας ανθρώπινες ζωές (αν και ο αυστηρός ορισμός είναι απλώς ότι η έλλειψη της προθεσμίας συνιστά αποτυχία του συστήματος).

2.5 Χρονοδρομολόγηση(scheduling) και prioritization σε συστήματα πραγματικού χρόνου

Με τον όρο χρονοδρομολόγηση ή χρονοπρογραμματισμό (scheduling) αναφερόμαστε στον αλγόριθμο που χρησιμοποιείται για να αποφασισθεί ποια από τις διεργασίες (ή εργασίες) που είναι έτοιμες για εκτέλεση θα δεσμεύσει την ΚΜΕ για να αρχίσει να εκτελείται.

Σε συστήματα σε πραγματικό χρόνο, ο χρονοδρομολογητής (scheduler) θεωρείται ως το πιο σημαντικό στοιχείο που είναι συνήθως ένας βραχυπρόθεσμος χρονοδρομολογητής εργασιών(short-term task scheduler). Ο κύριος στόχος αυτού του χρονοδρομολογητή είναι να μειωθεί ο χρόνος απόκρισης που σχετίζεται με καθεμία από τις σχετικές διαδικασίες αντί να χειριστεί την προθεσμία [8].

Εάν χρησιμοποιείται preemptive scheduler, η εργασία σε πραγματικό χρόνο πρέπει να περιμένει έως ότου ολοκληρωθεί το αντίστοιχο χρονοδιάγραμμα εργασιών. Στην περίπτωση ενός non-preemptive scheduler, ακόμη και αν η υψηλότερη προτεραιότητα έχει εκχωρηθεί στην εργασία, πρέπει να περιμένει μέχρι την ολοκλήρωση της τρέχουσας εργασίας. Αυτή η εργασία μπορεί να είναι αργή (ή) της χαμηλότερης προτεραιότητας και μπορεί να οδηγήσει σε μεγαλύτερη αναμονή.

Μια καλύτερη προσέγγιση σχεδιάζεται συνδυάζοντας τόσο preemptive and non-preemptive scheduling. Αυτό μπορεί να γίνει με την εισαγωγή χρονικών διακοπών σε συστήματα που βασίζονται σε προτεραιότητα(priority), πράγμα που σημαίνει ότι η τρέχουσα διεργασία διακόπτεται σε χρονικό διάστημα και εάν υπάρχει μια διαδικασία υψηλότερης προτεραιότητας σε μια έτοιμη ουρά, εκτελείται προλογίζοντας την τρέχουσα διαδικασία.

Με βάση τη δυνατότητα του scheduling, την εφαρμογή (στατική ή δυναμική) και το αποτέλεσμα της ανάλυσης, ο αλγόριθμος προγραμματισμού ταξινομείται ως εξής.

1)Static table-driven approaches: Αυτοί οι αλγόριθμοι εκτελούν συνήθως μια στατική ανάλυση που σχετίζεται με την Χρονοδρομολόγηση και καταγράφουν τα ωφέλιμα χρονοδιαγράμματα. Αυτό βοηθά στην παροχή ενός προγράμματος που μπορεί να επισημάνει μια εργασία με την οποία η εκτέλεση πρέπει να ξεκινήσει κατά το χρόνο εκτέλεσης.

2) Static priority-driven preemptive approaches:
Παρόμοια με την πρώτη προσέγγιση, αυτός ο τύπος αλγορίθμων χρησιμοποιεί επίσης στατική ανάλυση της χρονοδρομολόγισης. Η διαφορά είναι ότι αντί να επιλέγει ένα συγκεκριμένο τύπο χρονοδρομολόγησης, παρέχει έναν χρήσιμο τρόπο καθορισμού προτεραιοτήτων μεταξύ διαφόρων εργασιών στον προληπτικό προγραμματισμό (preemptive scheduling).
3) Dynamic planning-based approaches:
Εδώ, τα εφικτά χρονοδιαγράμματα προσδιορίζονται δυναμικά (κατά το χρόνο εκτέλεσης). Φέρει ένα ορισμένο σταθερό χρονικό διάστημα και μια διαδικασία εκτελείται εάν και μόνο εάν ικανοποιεί το χρονικό περιορισμό.

4)Dynamic best effort approaches:
Αυτοί οι τύποι προσεγγίσεων θεωρούν προθεσμίες αντί για εφικτά χρονοδιαγράμματα. Ως εκ τούτου, η εργασία ματαιώνεται εάν συμπληρωθεί η προθεσμία. Αυτή η προσέγγιση χρησιμοποιείται ευρέως είναι τα περισσότερα από τα συστήματα σε πραγματικό χρόνο.

Στο πλαίσιο των συστημάτων πολλαπλών εργασιών(multitasking systems) η πολιτικές χρονοδρομολόγησης(scheduling polices) συνήθως καθοδηγούνται από προτεραιότητα (pre-emptive schedulers). Σε ορισμένες περιπτώσεις, αυτές μπορούν να εγγυηθούν hard real-time απόδοση(για παράδειγμα, εάν το σύνολο των εργασιών και των προτεραιοτήτων τους είναι

γνωστό εκ των προτέρων). Υπάρχουν και άλλοι hard real time schedulers, όπως ο rate monotonic που δεν είναι συνηθισμένος στα συστήματα γενικής χρήσης, καθώς απαιτεί πρόσθετες πληροφορίες για τον προγραμματισμό μιας εργασίας: δηλαδή μια δεσμευμένη ή χειρότερη εκτίμηση για το χρονικό διάστημα (worst case time execution) που πρέπει να εκτελέσει η εργασία .

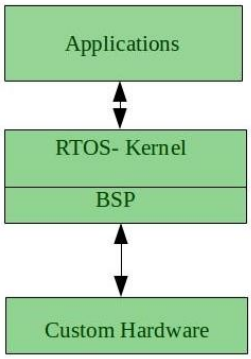
Επίσης υπάρχουν συγκεκριμένοι αλγόριθμοι για τον προγραμματισμό τέτοιων σκληρών εργασιών σε πραγματικό χρόνο, όπως ο earliest deadline first(EDF), ο οποίος, αγνοεί την γενική εναλλαγή περιεχομένου(context switching), είναι αποδοτικός για φορτία συστήματος κάτω του 100%.Νέα συστήματα χρονοδρομολόγισης επικάλυψης(overlay scheduling systems), όπως ένας adaptive partition scheduler, βοηθούν στη διαχείριση μεγάλων συστημάτων με ένα συνδυασμό εφαρμογών σκληρού πραγματικού(hard real time) και μη πραγματικού χρόνου.

Ειδικά συστήματα χρονοδρομολόγησης για ενσωματωμένες συσκευές προσπαθούν να επιτύχουν εκτέλεση σε πραγματικό χρόνο όπως στις αναφορές [9] [10]

2.6 Λειτουργικά συστήματα πραγματικού χρόνου (Real time operating systems (RTOS))

Τα συστήματα πραγματικού χρόνου πολλές φορές μπορεί να εξυπηρετούνται από ένα ειδικό είδος λειτουργικού συστήματος που λέγεται λειτουργικό σύστημα πραγματικού χρόνου (RTOS), Το χρονικό διάστημα που απαιτείται για την επεξεργασία και την απόκριση στις εισόδους είναι πολύ μικρό. Αυτό το χρονικό διάστημα ονομάζεται χρόνος απόκρισης .

Τα λειτουργικά συστήματα αυτά ανάλογα με τις απαιτήσεις χρόνου χωρίζονται στις κατηγορίες (hard,firm και soft) real time operating systems ,όπως εξηγήσαμε και στην ενότητα 2.4.



Εικόνα 4-real time operating system architecture

Πλεονεκτήματα του RTOS:

Μέγιστη κατανάλωση: Μέγιστη χρήση συσκευών και συστήματος, συνεπώς περισσότερη παραγωγή από όλους τους πόρους

Μετατόπιση εργασιών: Ο χρόνος που έχει οριστεί για την αλλαγή εργασιών σε αυτά τα συστήματα είναι πολύ μικρότερος. Για παράδειγμα, σε παλαιότερα συστήματα, χρειάζονται περίπου 10 μικρο δευτερόλεπτα για τη μετατόπιση μιας εργασίας σε άλλη και στα τελευταία συστήματα χρειάζονται 3 μικρο δευτερόλεπτα.

Εστίαση στην εφαρμογή: Εστίαση στην εκτέλεση εφαρμογών και λιγότερη σημασία σε εφαρμογές που βρίσκονται στην ουρά.

Λειτουργικό σύστημα σε πραγματικό χρόνο σε ενσωματωμένο σύστημα: Δεδομένου ότι το μέγεθος των προγραμμάτων είναι μικρό, το RTOS μπορεί επίσης να χρησιμοποιηθεί σε ενσωματωμένα συστήματα όπως στις μεταφορές και σε άλλα.

Χωρίς σφάλμα: Αυτοί οι τύποι συστημάτων είναι χωρίς σφάλματα.

Κατανομή μνήμης: Η κατανομή μνήμης διαχειρίζεται καλύτερα σε αυτόν τον τύπο συστημάτων.

Μειονεκτήματα του RTOS:

Περιορισμένες εργασίες: Πολύ λίγες εργασίες εκτελούνται ταυτόχρονα και η συγκέντρωσή τους είναι πολύ λιγότερο σε λίγες εφαρμογές για την αποφυγή σφαλμάτων.

Χρήση μεγάλων πόρων συστήματος: Μερικές φορές οι πόροι του συστήματος δεν είναι τόσο καλοί και είναι επίσης ακριβοί.

Σύνθετοι αλγόριθμοι: Οι αλγόριθμοι είναι πολύ περίπλοκοι και δύσκολο για τον σχεδιαστή να γράψει.

Πρόγραμμα οδήγησης συσκευής και σήματα διακοπής: Χρειάζεται συγκεκριμένα προγράμματα οδήγησης συσκευών και σήματα διακοπής για απόκριση νωρίτερα στις διακοπές.

Προτεραιότητα νημάτων: Δεν είναι καλό να ορίσετε προτεραιότητα νήματος, καθώς αυτά τα συστήματα είναι πολύ λιγότερο επιρρεπή σε εναλλαγή εργασιών.

Παραδείγματα λειτουργικών συστημάτων σε πραγματικό χρόνο είναι: Επιστημονικά πειράματα, συστήματα ιατρικής απεικόνισης, βιομηχανικά συστήματα ελέγχου, οπλικά συστήματα, ρομπότ, συστήματα ελέγχου εναέριας κυκλοφορίας κ.λπ.

2.7 Ασύμμετρη Πολυεπεξεργασία Asymmetric Multiprocessing

Με τον όρο ασύμμετρη πολυεπεξεργασία αναφερόμαστε σε ένα σύστημα(είτε μόνο σε επίπεδο υλικού είτε με λειτουργικό σύστημα) όπου πολλοί επεξεργαστές λειτουργούν παράλληλα αλλά χωρίς να αντιμετωπίζονται εξίσου από το σύστημα. Για παράδειγμα σε ένα σύστημα με 2 επεξεργαστές μπορεί ο ένας να έχει αναλάβει τον ρόλο να εκτελεί τις εργασίες του λειτουργικού συστήματος και ο άλλος να έχει ως αρμοδιότητα την αλληλεπίδραση με τις συσκευές εισόδου εξόδου.

Αυτή η τεχνική χρησιμοποιείτε κατά κόρον στα ενσωματωμένα συστήματα για τον λόγο ότι είναι συστήματα συνήθως ειδικού σκοπού., και έτσι προσφέρει πολλές φορές αύξηση στην απόδοση του συστήματος και πολλές φορές είναι λιγότερο δαπανηρό να υλοποιηθεί.

Ενσωματωμένα συστήματα ειδικού σκοπού με ασύμμετρη πολυεπεξεργασία έχουν σχεδιαστεί με διάφορες τεχνικές [11] [12]

2.8 Επιταχυντές (Accelerators)

Συχνά στα ενσωματωμένα συστήματα υπάρχει το πρόβλημα της κατανάλωσης ενέργειας όμως πάντα χρειαζόμαστε και μεγαλύτερες επιδόσεις και αυτό συνεπάγετε μεγαλύτερη κατανάλωση ενέργειας. Με την χρήση επιταχυντών οι οποίοι είναι υλικό υπολογιστή(συνήθως κυκλώματα) το οποί έχει σχεδιαστεί για να εκτελεί κάποιες συγκεκριμένε εργασίες πιο αποδοτικά από την CPU επιτυγχάνεται λιγότερη κατανάλωση ενέργειας [13].

Οι επιταχυντές χρησιμοποιούνται πολύ σε ενσωματωμένα συστήματα για αύξηση της απόδοσης και μικρότερης κατανάλωσης ενέργειας όμως πρέπει και οι εργασίες να μπορούν να εκτελεστούν πιο αποδοτικά στους επιταχυντές. Για παράδειγμα εργασίες που εκτελούνται παράλληλα δίνουν καλύτερα αποτελέσματα acceleration από εργασίες που εκτελούνται ακολουθιακά.

Στο zedboard δίνεται η δυνατότητα χρήσης των επιταχυντών μέσο της λογικής μονάδας logic block.

Στον σύνδεσμο [14]

2.9 Worst case execution time (WCET) και Performance Prediction

Στα ενσωματωμένα συστήματα συχνά πρέπει να γίνει υπολογισμός της απόδοσης του συστήματος (performance prediction) για να υπολογιστούν τα χρονικά όρια που παρέχει το σύστημα. Στην περίπτωση που το σύστημα είναι hard real time η μέθοδός που συνηθίζεται είναι η εξής υπολογίζεται το worst case execution time το οποίο είναι το μέγιστο χρονικό διάστημα το οποίο χρειάζεται ένα συγκεκριμένο σύστημα για να εκτελέσει μια εργασία και

σε συνδυασμό με αλγορίθμους scheduling υπολογίζεται η σκοπιμότητα(feasibility) του συστήματος. Στην περίπτωση όμως που αναφερόμαστε σε soft real time συστήματα στοχαστικές προσεγγίσεις χρησιμοποιούντα για να υπολογίσουμε την πιθανότητα να μην προλάβει μία εργασία το deadline [15].

Καταλαβαίνουμε λοιπόν ότι ο υπολογισμός του worst case execution time είναι ένα αναπόσπαστο κομμάτι στον προγραμματισμό ενσωματωμένων συστημάτων και πιο συχνά σε συστήματα πραγματικού χρόνου διότι μας βοηθάει να υπολογίσουμε την απόδοση του συστήματος .

Ενώ το WCET είναι δυνητικά εφαρμόσιμο σε πολλά συστήματα σε πραγματικό χρόνο, στην πράξη η διασφάλιση του WCET χρησιμοποιείται κυρίως από συστήματα σε πραγματικό χρόνο που σχετίζονται με υψηλή αξιοπιστία ή ασφάλεια. Αυξανόμενη χρήση λογισμικού σε συστήματα αυτοκινήτων οδηγεί επίσης την ανάγκη χρήσης ανάλυσης λογισμικού WCET.

Στο σχεδιασμό ορισμένων συστημάτων, το WCET χρησιμοποιείται συχνά ως είσοδος στην schedulability analysis , αν και μια πολύ πιο συνηθισμένη χρήση του WCET σε κρίσιμα συστήματα είναι να διασφαλιστεί ότι οι προ-κατανεμημένοι προϋπολογισμοί χρονισμού σε ένα σύστημα προγραμματισμένων διαμερισμάτων όπως το ARINC 653δ δεν παραβιάστηκαν.

3. Approximate computing (Κατά προσέγγιση υπολογισμοί)

Aapproximate computing είναι μια τεχνική υπολογισμού που επιστρέφει ένα πιθανώς ανακριβές αποτέλεσμα παρά ένα εγγυημένο ακριβές αποτέλεσμα και μπορεί να χρησιμοποιηθεί για εφαρμογές όπου ένα κατά προσέγγιση αποτέλεσμα είναι αρκετό για τον σκοπό του.

Ένα παράδειγμα μιας τέτοιας κατάστασης είναι για μια μηχανή αναζήτησης όπου δεν υπάρχει ακριβής απάντηση για ένα συγκεκριμένο ερώτημα αναζήτησης και ως εκ τούτου, πολλές απαντήσεις μπορεί να είναι αποδεκτές. Ομοίως, η περιστασιακή απόρριψη ορισμένων καρέ σε μια εφαρμογή βίντεο μπορεί να μην εντοπιστεί λόγω των αντιληπτικών περιορισμών των ανθρώπων. Ο υπολογισμός κατά προσέγγιση βασίζεται στην παρατήρηση ότι σε πολλά σενάρια, παρόλο που η ακριβής υπολογισμός απαιτεί μεγάλη ποσότητα πόρων, επιτρέποντας την περιορισμένη προσέγγιση μπορεί να προσφέρει δυσανάλογα κέρδη στην απόδοση και την ενέργεια, ενώ παράλληλα επιτυγχάνεται αποδεκτή ακρίβεια αποτελεσμάτων.

Π.χ. Στον αλγόριθμος ομαδοποίησης k-means, επιτρέπει μόνο απώλεια 5% στην ακρίβεια ταξινόμησης και μπορεί να προσφέρει 50 φορές εξοικονόμηση ενέργειας σε σύγκριση με την ταξινόμηση πλήρους ακρίβειας .

Η βασική απαίτηση στον κατά προσέγγιση υπολογισμό είναι ότι η προσέγγιση μπορεί να εισαχθεί μόνο σε μη κρίσιμα δεδομένα, καθώς η προσέγγιση κρίσιμων δεδομένων (π.χ. λειτουργίες ελέγχου) μπορεί να οδηγήσει σε καταστροφικές συνέπειες, όπως σφάλμα προγράμματος ή εσφαλμένη έξοδο.

3.1 Στρατηγικές

Διάφορες στρατηγικές μπορούν να χρησιμοποιηθούν για την εκτέλεση υπολογισμού κατά προσέγγιση.

Κυκλώματα κατά προσέγγιση: Κατά προσέγγιση αθροιστές, πολλαπλασιαστές και άλλα λογικά κυκλώματα μπορούν να μειώσουν την επιβάρυνση του υλικού. Για παράδειγμα, ένας κατά προσέγγιση αθροιστής πολλαπλών δυαδικών ψηφίων μπορεί να αγνοήσει την αλυσίδα μεταφοράς(carry chain) και, επομένως, να επιτρέψει σε όλους τους δευτερεύοντες αθροιστές να εκτελούν παράλληλα τη λειτουργία προσθήκης.

Κατά προσέγγιση αποθηκευτικός χώρος: Αντί να αποθηκεύονται ακριβείς τιμές δεδομένων, μπορούν να αποθηκευτούν περίπου, π.χ., περικόπτοντας τα χαμηλότερα bits σε δεδομένα κινητής υποδιαστολής. Μια άλλη μέθοδος είναι η αποδοχή λιγότερο αξιόπιστης μνήμης. Για αυτό, στις DRAM [9] και eDRAM μνήμες, ο ρυθμός ανανέωσης μπορεί να μειωθεί ή να ελεγχθεί .Στις SRAM, η τάση τροφοδοσίας μπορεί να μειωθεί ή να ελεγχθεί. Γενικά, τυχόν μηχανισμοί ανίχνευσης και διόρθωσης σφαλμάτων πρέπει να απενεργοποιηθούν.

Προσέγγιση σε επίπεδο λογισμικού: Υπάρχουν διάφοροι τρόποι προσέγγισης σε επίπεδο λογισμικού. Η απομνημόνευση (memoization) μπορεί να εφαρμοστεί η οποία είναι μια τεχνική optimization . Ορισμένες επαναλήψεις βρόχων μπορούν να παραλειφθούν (που ονομάζονται διάτρηση βρόχου) για να επιτευχθεί πιο γρήγορα ένα αποτέλεσμα. Ορισμένες εργασίες μπορούν επίσης να παραλειφθούν, για παράδειγμα όταν μια συνθήκη χρόνου εκτέλεσης υποδηλώνει ότι αυτές οι εργασίες δεν θα είναι χρήσιμες (παράλειψη εργασιών). Οι αλγόριθμοι του Μόντε Κάρλο και οι τυχαιοποιημένοι αλγόριθμοι είναι η ορθότητα των συναλλαγών για εγγυήσεις χρόνου εκτέλεσης. Ο υπολογισμός μπορεί να αναδιατυπωθεί σύμφωνα με παραδείγματα που επιτρέπουν εύκολα την επιτάχυνση σε εξειδικευμένο υλικό.

Οι τεχνικές optimization και η τεχνική fixed point arithmetic που χρησιμοποιούτανε σε αυτή την εργασία ανήκουν σε αυτήν την κατηγορία στρατηγικής για approximate calculations

Κατά προσέγγιση σύστημα: Σε ένα κατά προσέγγιση σύστημα, διαφορετικά υποσυστήματα του συστήματος όπως ο επεξεργαστής, η μνήμη, ο αισθητήρας και οι λειτουργικές μονάδες προσεγγίζονται συνεργικά για να αποκτήσουν μια πολύ καλύτερη καμπύλη ανταλλαγής QE(Q-E trade-off curve) σε επίπεδο συστήματος σε σύγκριση με μεμονωμένες προσεγγίσεις σε καθένα από τα υποσυστήματα

4.1 Περιοχές εφαρμογής

Ο υπολογισμός κατά προσέγγιση έχει χρησιμοποιηθεί σε διάφορους τομείς όπου οι εφαρμογές είναι ανεκτικές σε σφάλματα, όπως επεξεργασία πολυμέσων, μηχανική εκμάθηση,

επεξεργασία σήματος, επιστημονικός υπολογισμός κ.λπ. Η Google χρησιμοποιεί αυτήν την προσέγγιση στις μονάδες επεξεργασίας Tensor (TPU, μια προσαρμοσμένη ASIC

Στις παρακάτω αναφορές μπορείτε να δείτε εργασίες για ενσωματωμένα συστήματα ειδικού σκοπού που συνδυάζουν approximate calculations με real time επεξεργασία [16] [17] [18] η και scheduling [19]

4. Αναπαράσταση πραγματικών αριθμών και fixed point αριθμητική

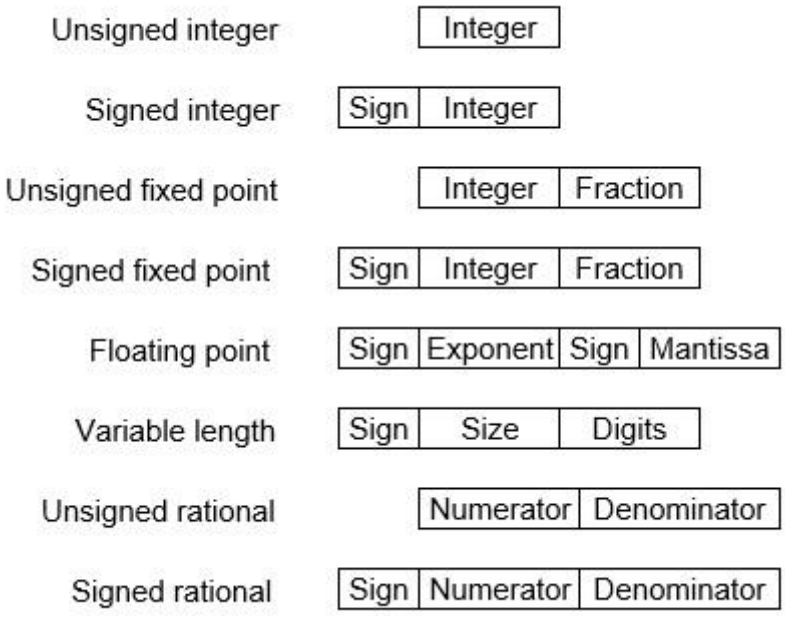
Οι ψηφιακοί υπολογιστές χρησιμοποιούν το δυαδικό σύστημα αριθμών για να αντιπροσωπεύουν όλους τους τύπους πληροφοριών μέσα στους υπολογιστές. Οι αλφαριθμητικοί χαρακτήρες αντιπροσωπεύονται χρησιμοποιώντας δυαδικά ψηφία (δηλ. 0 και 1). Οι ψηφιακές αναπαραστάσεις είναι πιο εύκολο να σχεδιαστούν, η αποθήκευση είναι εύκολη, η ακρίβεια(accuracy) και η ακρίβεια(precision) είναι μεγαλύτερες.

Υπάρχουν διάφοροι τύποι τεχνικών αναπαράστασης αριθμών για την αναπαράσταση ψηφιακών αριθμών, για παράδειγμα: Σύστημα δυαδικού αριθμού, σύστημα οκταδικών αριθμών, σύστημα δεκαδικών αριθμών και δεκαεξαδικό σύστημα κ.λπ. .

Στο κεφάλαιο αυτό θα δούμε τους δύο τρόπους με τους οποίους αποθηκεύονται οι πραγματικοί αριθμοί σε ένα υπολογιστικό σύστημα ούτος ώστε να είμαστε σε θέση στο κεφάλαιο 2.2 να εξηγήσουμε πως υλοποιείτε η τεχνική αυτή αλλά και τα θετικά και αρνητικά της.

4.1 Αποθήκευση πραγματικών αριθμών σε ένα υπολογιστικό σύστημα

Αυτές είναι δομές όπως παρακάτω



Εικόνα 5-Τύποι δεδομένων Πραγματικών αριθμών

Υπάρχουν δύο βασικές προσεγγίσεις για την αποθήκευση πραγματικών αριθμών (δηλαδή αριθμών με κλασματικό στοιχείο) στη σύγχρονη πληροφορική.

1)Η αποθήκευση τους σαν αριθμούς κινητής υποδιαστολής(floating point representation)

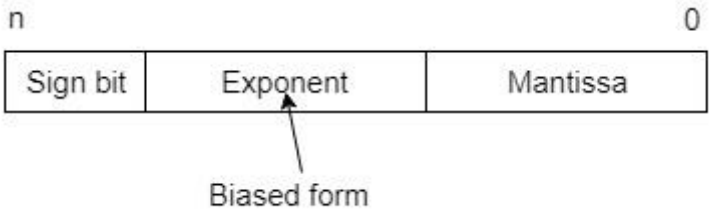
2)Η αποθήκευση τους σαν αριθμούς σταθερής υποδιαστολής(fixed point representation)

4.1.1 Αναπαράσταση αριθμών κινητής υποδιαστολής

Αυτή η αναπαράσταση δεν διατηρεί συγκεκριμένο αριθμό bit για το ακέραιο μέρος ή το κλασματικό μέρος. Αντ 'αυτού, διατηρεί έναν συγκεκριμένο αριθμό bit για τον αριθμό (που ονομάζεται mantissa και έναν ορισμένο αριθμό bit για να πει πού μέσα σε αυτόν τον αριθμό κάθεται το δεκαδικό μέρος (που ονομάζεται εκθέτης).

Η αναπαράσταση ενός αριθμού κινητής υποδιαστολής έχει δύο μέρη: το πρώτο μέρος αντιπροσωπεύει έναν προσημασμένο αριθμό σταθερής υποδιαστολής που ονομάζεται mantissa. Το δεύτερο μέρος του υποδηλώνει τη θέση του δεκαδικού (ή δυαδικού) σημείου και ονομάζεται εκθέτης. Η mantissa σταθερής υποδιαστολής μπορεί να είναι κλάσμα ή ακέραιος. Το κυμαινόμενο σημείο ερμηνεύεται πάντα για να αντιπροσωπεύει έναν αριθμό στην ακόλουθη μορφή: Mxr^{e} .

Μόνο η mantissa m και ο εκθέτης e αντιπροσωπεύονται φυσικά στους καταχωρητές (registers) (συμπεριλαμβανομένου του πρόσημου τους). Ένας δυαδικός αριθμός κινητής υποδιαστολής αντιπροσωπεύεται με παρόμοιο τρόπο εκτός από το ότι χρησιμοποιεί τη βάση 2 για τον εκθέτη. Ένας αριθμός κινητής υποδιαστολής λέγεται ότι ομαλοποιείται(normalized) εάν το πιο σημαντικό ψηφίο της mantissa είναι 1.



Εικόνα 6-Αναπαράσταση αριθμού κινητής υποδιαστολής

Έτσι, ο πραγματικός αριθμός είναι $(-1)^{s} (1 + m) x2^{(e-Bias)}$, όπου το *s* είναι το σύμβολο του για το πρόσημο , το *m* είναι η mantissa, το *e* είναι η εκθετική τιμή και το *Bias* είναι ο αριθμός για να αντισταθμιστεί ο εκθέτης.

Σημειώστε ότι οι προσημασμένοι ακέραιοι και εκθετικοί αντιπροσωπεύονται είτε από προσημασμένη(signed) αναπαράσταση, είτε από αναπαράσταση συμπληρώματος 1, είτε από αναπαράσταση συμπληρώματος δύο.

Η αναπαράσταση αριθμών κινητής υποδιαστολής είναι πιο ευέλικτη. Οποιοσδήποτε μη μηδενικός αριθμός μπορεί να αναπαριστάται σε κανονικοποιημένη μορφή $\pm (_1 b_1 b_2 b_3 ...)_2 x2^{n}$ Αυτή είναι κανονικοποιημένη μορφή αριθμού x.

4.1.2 Αναπαράσταση αριθμών σταθερής υποδιαστολής

Αυτή η αναπαράσταση έχει σταθερό αριθμό bit για ακέραιο μέρος και για κλασματικό μέρος. Για παράδειγμα, εάν η αναπαράσταση σταθερού σημείου είναι IIII.FFFF, τότε μπορείτε να αποθηκεύσετε την ελάχιστη τιμή είναι 0000.0001 και η μέγιστη τιμή είναι

9999,9999. Υπάρχουν τρία μέρη μιας αναπαράστασης αριθμού σταθερού σημείου: το πεδίο σημαδιών, το ακέραιο πεδίο και το κλασματικό πεδίο.

Εικόνα 7-Αναπαράσταση αριθμού σταθερής υποδιαστολής

Μπορούμε να αντιπροσωπεύσουμε αυτούς τους αριθμούς χρησιμοποιώντας:

- Προσημασμένη(signed) αναπαράσταση: εύρος από - $(2^{(k-1)} -1)$ έως $(2^{(k-1)} -1)$, για k bits.
- Αναπαράσταση συμπληρώματος 1: εύρος από - $(2^{(k-1)} -1)$ έως $(2^{(k-1)} -1)$, για k bits.
- Αναπαράσταση συμπληρώματος 2: από - $(2^{(k-1)})$ έως $(2^{(k-1)} -1)$, για k bits.

Η αναπαράσταση συμπληρωματικότητας 2 προτιμάται στο σύστημα υπολογιστών λόγω της σαφούς ιδιότητας και ευκολότερη για αριθμητικές λειτουργίες.

4.2 Fixed point arithmetic

Fixed point arithmetic ονομάζουμε την τεχνική που δίνει την δυνατότητα να αποθηκεύσουμε και να κάνουμε πράξεις μεταξύ κλασματικών αριθμών χωρίς την χρήση αριθμών κινητής υποδιαστολής(floating point numbers) αλλά χρησιμοποιώντας αριθμούς σταθερής υποδιαστολής (fixed point numbers).

Η τεχνική αυτή έχει πολλά θετικά αλλά και αρνητικά και θα τα αναλύσουμε στην συνέχεια όμως αξίζει να αναφερθούμε στο γεγονός ότι η τεχνική αυτή κάνει δυνατή την χρήση κλασματικών αριθμών και σε συστήματα που δεν διαθέτουν μονάδα κινητής υποδιαστολής (FPU).

4.2.1 Θετικά και αρνητικά της τεχνικής

Τα θετικά της τεχνικής σε σύγκριση με την χρήση αριθμών κινητής υποδιαστολής είναι:

1. Ευελιξία στην ακρίβεια
2. Ευελιξία στο εύρος των τιμών
3. Ευελιξία στο μέγεθος των δεδομένων

Όσο αφόρα τα 3 πρώτα θετικά αξίζει να πούμε ότι η ακρίβεια το εύρος τιμών και το μέγεθος των δεδομένων είναι προκαθορισμένα από τον οργανισμό IEEE όταν χρησιμοποιούμε floating

point arithmetic όμως στην περίπτωση της fixed point arithmetic ο προγραμματιστής μπορεί να ελέγξει αυτές τις παραμέτρους όταν γράψει το πρόγραμμα.

4. Κάποιες πράξεις είναι γρηγορότερες(smaller data size and SIMD)

Άλλο ένα πολύ μεγάλο θετικό της τεχνικής αυτής είναι ότι κάποιες πράξεις εκτελούνται γρηγορότερα αυτό παρατηρείτε αρκετά όταν το επεξεργαστικό σύστημα έχει αρχιτεκτονική SIMD(single instruction multiple data).Για παράδειγμα αν για τον τύπο των δεδομένων ο προγραμματιστής επιλέξει 1 byte(μπορεί να μην υπάρχει ανάγκη για μεγάλη ακρίβεια ή εύρος τιμών)τότε σε ένα σύστημα με αρχιτεκτονική SIMD ο επεξεργαστής θα κάνει πράξεις με 16 bytes την φορά ενώ άμα αποθηκεύαμε τους αριθμούς ως αριθμούς κινητής υποδιαστολής θα μπορούσε να κάνει πράξεις μόνο με 4 αριθμούς την φορά .

5. Είναι ευκολότερο να κάνουμε προσαρμογές τύπου(casting με ακέραιους αλλά και ευκολότερο να πάρουμε το κλασματικό κομμάτι του αριθμού
6. Η τεχνική αυτή κάνει δυνατή την χρήση κλασματικών αριθμών και σε συστήματα που δεν διαθέτουν μονάδα κινητής υποδιαστολής (FPU).

Το τελευταίο από τα θετικά είναι επίσης πολύ σημαντικό διότι δεν μπορούμε να έχουμε σε όλα τα συστήματα μονάδα κινητής υποδιαστολής (FPU) είτε για λόγους κόστους είτε για λόγους πολυπλοκότητας. Για παράδειγμα σε συστήματα για εφαρμογές επεξεργασίας σήματος (DSP) συνηθίζετε να μην υπάρχει FPU εκεί γίνεται χρήση της τεχνικής αυτής πολύ συχνά.

Τα αρνητικά της τεχνικής σε σύγκριση με την χρήση αριθμών κινητής υποδιαστολής είναι:

1. Κάποιες πράξεις είναι πιο αργές

Συνήθως ο πολλαπλασιασμός και η διαίρεση είναι λίγο πιο αργές πράξεις.

2. Η ακρίβεια και το εύρος είναι συνήθως λιγότερα
3. Συνήθως δεν υπάρχουν έτυμες από την για την υποστήριξη της τεχνικής(no inbuild instructions)
4. Ο κώδικας είναι ποιο πολύπλοκος

4.2.2 Κανόνες για fixed point arithmetic

Αναπαράσταση:

Η τιμή ενός τύπου δεδομένων σταθερής υποδιαστολής είναι ουσιαστικά ένας ακέραιος που κλιμακώνεται από έναν έμμεσο ειδικό παράγοντα που καθορίζεται από τον τύπο. Για παράδειγμα, η τιμή 1,23 μπορεί να αναπαρασταθεί ως 1230 σε τύπο δεδομένων σταθερής υποδιαστολής με συντελεστή κλιμάκωσης 1/1000 και η τιμή 1.230.000 μπορεί να αναπαρασταθεί ως 1230 με συντελεστή κλιμάκωσης 1000. Σε αντίθεση με τους τύπους δεδομένων κινητής υποδιαστολής ,ο συντελεστής κλιμάκωσης είναι ο ίδιος για όλες τις τιμές του ίδιου τύπου και δεν αλλάζει καθ 'όλη τη διάρκεια του υπολογισμού.

Ο συντελεστής κλιμάκωσης είναι συνήθως μια δύναμη του 10 (για ανθρώπινη άνεση) ή μια δύναμη του 2 (για υπολογιστική απόδοση). Ωστόσο, περιστασιακά μπορούν να χρησιμοποιηθούν και άλλοι παράγοντες κλιμάκωσης, π.χ. μια τιμή χρόνου σε ώρες μπορεί να

αναπαριστάται ως τύπος σταθερού σημείου με συντελεστή κλίμακας 1/3600 για να ληφθούν τιμές με ακρίβεια ενός δευτερολέπτου.

Η μέγιστη τιμή ενός τύπου σταθερής υποδιαστολής είναι απλά η μεγαλύτερη τιμή που μπορεί να αναπαριστάται στον υποκείμενο ακέραιο τύπο πολλαπλασιαζόμενο με τον παράγοντα κλιμάκωσης. και ομοίως για την ελάχιστη τιμή.

Λειτουργίες:

Μετατροπή:

Για να μετατρέψετε έναν αριθμό κινητής υποδιαστολής(floating point number) σε αριθμό σταθερής υποδιαστολής(fixed point number) αρκεί να πολλαπλασιάσουμε τον αριθμό με έναν συντελεστή κλιμάκωσης που ορίζεται από τον προγραμματιστή με βάση τις ανάγκες του προγράμματος ,στην συνέχεια να στρογγυλοποιήσουμε τον αριθμό και τέλος να κάνουμε προσαρμογή τύπου(casting) σε ακέραιο.

Για να μετατρέψετε έναν σταθερής υποδιαστολής(fixed point number) σε αριθμό κινητής υποδιαστολής(floating point number) αρκεί να διαιρέσουμε τον αριθμό με τον συντελεστή κλιμάκωσης που έχουμε ορίσει στο πρόγραμμα μας τον και στην συνέχεια να κάνουμε προσαρμογή τύπου(casting) σε αριθμό κινητής υποδιαστολής.

Πρόσθεση και αφαίρεση: Για να προσθέσετε ή να αφαιρέσετε δύο τιμές του ίδιου τύπου σταθερής υποδιαστολής, αρκεί να προσθέσετε ή να αφαιρέσετε τους υποκείμενους ακέραιους αριθμούς και να διατηρήσετε τον κοινό συντελεστή κλιμάκωσης. Το αποτέλεσμα μπορεί να αναπαρασταθεί ακριβώς στον ίδιο τύπο, αρκεί να μην υπάρξει υπερχείλιση (υπό την προϋπόθεση ότι το άθροισμα των δύο ακέραιων ταιριάζει στον υποκείμενο τύπο ακέραιου).

Όμως η για να προσθέσουμε έναν αριθμό σταθερής υποδιαστολής με έναν ακέραιο αριθμό(integer) πρέπει ο ακέραιος να πολλαπλασιαστεί με τον συντελεστή κλιμάκωσης και στην συνέχεια να γίνει η πρόσθεση.

Πολλαπλασιασμός: Για να πολλαπλασιάσετε δύο αριθμούς σταθερής υποδιαστολής πρέπει να γίνει ο πολλαπλασιασμός και στην συνέχεια να γίνει διαίρεση του αποτελέσματος με τον συντελεστή κλιμάκωσης γιατί λόγο της επιμεριστικής ιδιότητας ο συντελεστής κλιμάκωσης υπάρχει 2 φορές και πρέπει να απορροφηθεί ,από τον πολλαπλασιασμό όμως είναι πολύ εύκολο να προκύψει υπερχείλιση γι αυτό καλό θα ήταν να έχει προνοήσει ο προγραμματιστής.

Για να γίνει πολλαπλασιασμός μεταξύ ενός αριθμού σταθερής υποδιαστολής και ενός ακέραιου αριθμού δεν χρειάζεται κάποιος κανόνας, πολλαπλασιάζουμε κανονικά τους 2 αριθμούς .

Διαίρεση:

Η διαίρεση είναι σαν τον πολλαπλασιασμό αλλά ανάποδα εδώ χάνουμε τους συντελεστές κλιμάκωσης λόγο της επιμεριστικής ιδιότητα γι αυτό ο ένας αριθμός πρέπει να πολλαπλασιαστεί με τον συντελεστή κλιμάκωσης και στην συνέχεια να γίνει η διαίρεση και όπως και πριν για να μην χαθεί η ακρίβεια καλό είναι ο προγραμματιστής να κάνει προσαρμογή τύπου για να μην προκύψει υπερχείλιση.

Απόκτηση του αποτελέσματος:

Αν οποιαδήποτε στιγμή δημιουργηθεί ανάγκη για να πάρουμε το πραγματικό αποτέλεσμα των πράξεων το μόνο που κάνουμε είναι να διαιρέσουμε με τον συντελεστή κλιμάκωσης.

Δυαδικό έναντι δεκαδικού:

Οι δύο πιο κοινές κατηγορίες τύπων σταθερής υποδιαστολής είναι δεκαδικά και δυαδικά. Οι τύποι δεκαδικών σταθερών σημείων έχουν συντελεστή κλιμάκωσης που είναι δύναμη δέκα. για τους δυαδικούς τύπους σταθερής υποδιαστολής είναι δύναμη δύο.

Οι δυαδικοί τύποι σταθερής υποδιαστολής χρησιμοποιούνται συχνότερα, επειδή οι μετατροπές από fixed σε float και αντιστοίχως μπορούν να εφαρμοστούν ως γρήγορες ολισθήσεις bit . Οι δυαδικοί αριθμοί σταθερής υποδιαστολής μπορούν να αντιπροσωπεύουν κλασματικές δυνάμεις του δύο ακριβώς, αλλά, όπως οι δυαδικοί αριθμοί κινητής υποδιαστολής, δεν μπορούν να αντιπροσωπεύουν ακριβώς κλασματικές δυνάμεις του δέκα. Εάν επιθυμείτε ακριβείς κλασματικές δυνάμεις του δέκα, τότε θα πρέπει να χρησιμοποιείται δεκαδική μορφή. Για παράδειγμα, το ένα δέκατο (0,1) και το ένα εκατοστό (0,01) μπορούν να αναπαρασταθούν μόνο περίπου με δυαδικές αναπαραστάσεις σταθερής υποδιαστολής ή δυαδικής κινητής υποδιαστολής, ενώ μπορούν να αναπαρασταθούν ακριβώς σε δεκαδικές παραστάσεις σταθερής υποδιαστολής ή δεκαδικής κινητής υποδιαστολής. Αυτές οι αναπαραστάσεις μπορεί να κωδικοποιηθούν με πολλούς τρόπους, συμπεριλαμβανομένου του δυαδικού κωδικού δεκαδικού (BCD).

Συμβολισμός:

Υπάρχουν διάφοροι συμβολισμοί που χρησιμοποιούνται για να αντιπροσωπεύουν το μήκος της λέξης και το σημείο της υποδιαστολής σε έναν δυαδικό αριθμό κινητής υποδιαστολής. Στην παρακάτω λίστα, *f* αντιπροσωπεύει τον αριθμό των κλασματικών δυαδικών ψηφίων, *m* τον αριθμό μεγέθους ή ακέραιος bits, *s* τον αριθμό των bits σημείου, και *b* το συνολικό αριθμό των δυαδικών ψηφίων.

- **Q** *f* : Το πρόθεμα "Q". Για παράδειγμα, το Q15 αντιπροσωπεύει έναν αριθμό με 15 κλασματικά bit. Αυτός ο συμβολισμός είναι ασαφής δεδομένου ότι δεν καθορίζει το μήκος της λέξης, ωστόσο συνήθως θεωρείται ότι το μήκος της λέξης είναι είτε 16 είτε 32 bit ανάλογα με τον επεξεργαστή-στόχο που χρησιμοποιείται.
- **Q** *μ* **.** *στ* : Η σαφής μορφή της σημειογραφίας «Q». Δεδομένου ότι ολόκληρη η λέξη είναι ακέραιος αριθμός συμπληρώματος 2, υπονοείται ένα bit σημαδιών. Για παράδειγμα, το Q1.30 περιγράφει έναν αριθμό με 1 ακέραιο bit και 30 κλασματικά bit αποθηκευμένα ως ακέραιος αριθμός 32-bit 2.

- **fx** μ **.** β : Το πρόθεμα "fx" είναι παρόμοιο με το παραπάνω, αλλά χρησιμοποιεί το μήκος λέξης ως το δεύτερο στοιχείο στο διακεκομμένο ζεύγος. Για παράδειγμα, το fx1.16 περιγράφει έναν αριθμό με 1 bit μεγέθους και 15 κλασματικά bit σε μια λέξη 16 bit. [3]
- *s* **:** *m* **:** *f* : Ωστόσο, οι άλλοι συμβολισμοί περιλαμβάνουν ένα bit, όπως αυτό που χρησιμοποιείται στον Οδηγό χρήσης του PS2(play station 2) GS. Διαφέρει επίσης από τη συμβατική χρήση χρησιμοποιώντας κόλον αντί για περίοδο ως διαχωριστικό. Για παράδειγμα, σε αυτή τη σημειογραφία, το 0: 8: 0 αντιπροσωπεύει έναν ακέραιο 8-bit.
- **(p, q)** Χρησιμοποιείται στη γλώσσα προγραμματισμού *PL / I* , για τον καθορισμό των συνολικών ψηφίων **p** (χωρίς το σύμβολο) με το **q** μετά το σημείο ακτίνας. **Το q** μπορεί να είναι θετικό ή αρνητικό και η δυαδική ή δεκαδική ακτίνα.

5.Εισαγωγή στην επεξεργασία εικόνας και την μετατροπή από έγχρωμη εικόνα σε ασπρόμαυρη

Στο κεφάλαιο αυτό θα αναλύσουμε τις βασικές έννοιες στο κομμάτι της επεξεργασίας εικόνας που είναι απαραίτητες για να κατανοήσουμε την μετατροπή από RGB 24 bit εικόνα σε grayscale εικόνα [20].

5.1 Εισαγωγή

Η επεξεργασία σημάτων είναι τομέας στην ηλεκτρολογία και στα μαθηματικά που ασχολείται με την ανάλυση και επεξεργασία αναλογικών και ψηφιακών σημάτων και ασχολείται με την αποθήκευση, το φιλτράρισμα και άλλες λειτουργίες σε σήματα. Αυτά τα σήματα περιλαμβάνουν σήματα μετάδοσης, σήματα ήχου ή φωνής, σήματα εικόνας και άλλα σήματα κ.λπ.

Από όλα αυτά τα σήματα, το πεδίο που ασχολείται με τον τύπο των σημάτων για τα οποία η είσοδος είναι εικόνα και η έξοδος είναι επίσης μια εικόνα γίνεται στην επεξεργασία εικόνας. Όπως υποδηλώνει το όνομα, ασχολείται με την επεξεργασία των εικόνων.

Μπορεί να χωριστεί περαιτέρω σε αναλογική επεξεργασία εικόνας και ψηφιακή επεξεργασία εικόνας.

5.2 Αναλογική επεξεργασία εικόνας

Η αναλογική επεξεργασία εικόνας γίνεται σε αναλογικά σήματα. Περιλαμβάνει επεξεργασία σε δισδιάστατα αναλογικά σήματα. Σε αυτόν τον τύπο επεξεργασίας, οι εικόνες χειρίζονται με ηλεκτρικά μέσα μεταβάλλοντας το ηλεκτρικό σήμα. Το κοινό παράδειγμα είναι η τηλεοπτική εικόνα.

Η ψηφιακή επεξεργασία εικόνας έχει κυριαρχήσει στην αναλογική επεξεργασία εικόνας με το πέρασμα του χρόνου λόγω του ευρύτερου φάσματος εφαρμογών της.

5.3 Ψηφιακή επεξεργασία εικόνας

Η επεξεργασία ψηφιακής εικόνας ασχολείται με την ανάπτυξη ενός ψηφιακού συστήματος που εκτελεί λειτουργίες σε μια ψηφιακή εικόνα.

5.4 Τι είναι μια εικόνα

Μια εικόνα ορίζεται ως μια δισδιάστατη συνάρτηση, **F (x, y)** , όπου x και y είναι χωρικές συντεταγμένες, και το πλάτος του **F** σε οποιοδήποτε ζεύγος συντεταγμένων (x, y) ονομάζεται **ένταση αυτής** της εικόνας σε αυτό σημείο. Όταν οι τιμές x, y και πλάτους του **F** είναι πεπερασμένες, το ονομάζουμε **ψηφιακή εικόνα .**
Με άλλα λόγια, μια εικόνα μπορεί να οριστεί από έναν δισδιάστατο πίνακα που είναι ειδικά διατεταγμένος σε σειρές και στήλες.
Η ψηφιακή εικόνα αποτελείται από έναν πεπερασμένο αριθμό στοιχείων, καθένα από τα οποία στοιχεία έχουν μια συγκεκριμένη τιμή σε μια συγκεκριμένη θέση. Αυτά αναφέρονται ως *στοιχεία εικόνας ή pixels* .Ένα *Pixel* χρησιμοποιείται ευρύτερα για να δηλώσει τα στοιχεία μιας ψηφιακής εικόνας.

Εικόνα 8- Έγχρωμη εικόνα

Η παραπάνω εικόνα είναι ένα παράδειγμα ψηφιακής εικόνας που βλέπετε τώρα στην οθόνη του υπολογιστή σας. Αλλά στην πραγματικότητα, αυτή η εικόνα δεν είναι παρά μια δισδιάστατη σειρά αριθμών που κυμαίνονται μεταξύ 0 και 255.

128	30	123
232	123	321
123	77	89
80	255	255

Κάθε αριθμός αντιπροσωπεύει την τιμή της συνάρτησης f (x, y) σε οποιοδήποτε σημείο. Σε αυτήν την περίπτωση η τιμή 128, 230, 123 αντιπροσωπεύει το καθένα μια μεμονωμένη τιμή pixel. Οι διαστάσεις της εικόνας είναι στην πραγματικότητα οι διαστάσεις αυτού του δισδιάστατου πίνακα.

5.5 Πως σχηματίζεται μια ψηφιακή εικόνα

Δεδομένου ότι η λήψη μιας εικόνας από μια κάμερα είναι μια φυσική διαδικασία. Το φως του ήλιου χρησιμοποιείται ως πηγή ενέργειας. Ένας πίνακας αισθητήρων χρησιμοποιείται για την απόκτηση της εικόνας. Έτσι, όταν το φως του ήλιου πέφτει πάνω στο αντικείμενο, τότε η ποσότητα του φωτός που αντανακλάται από αυτό το αντικείμενο ανιχνεύεται από τους αισθητήρες και παράγεται ένα σήμα συνεχούς τάσης από την ποσότητα των δεδομένων που ανιχνεύονται. Για να δημιουργήσουμε μια ψηφιακή εικόνα, πρέπει να

μετατρέψουμε αυτά τα δεδομένα σε ψηφιακή μορφή. Αυτό περιλαμβάνει δειγματοληψία και ποσοτικοποίηση. (Συζητούνται αργότερα). Το αποτέλεσμα της δειγματοληψίας και της κβαντοποίησης καταλήγει σε ένα δισδιάστατο πίνακα ή μήτρα αριθμών που δεν είναι τίποτα άλλο από μια ψηφιακή εικόνα.

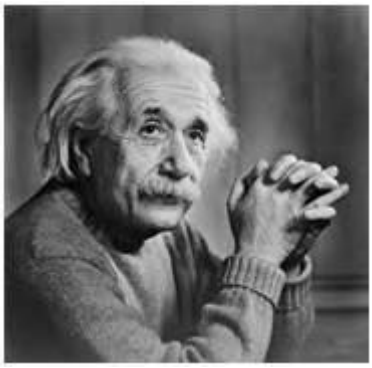
5.6 Τύποι εικόνων

Υπάρχουν διάφοροι τύποι εικόνων όμως εμείς θα χρειαστούμε μόνο 2 βασικούς τύπους για την υλοποίηση της εργασίας μας.

5.6.1 Μορφή χρώματος 8 bit (Grayscale)

Η μορφή χρώματος 8 bit είναι μια από τις πιο διάσημες μορφές εικόνας. Έχει 256 διαφορετικές αποχρώσεις χρωμάτων σε αυτό. Είναι συνήθως γνωστό ως εικόνα σε κλίμακα του γκρι(Grayscale). Το εύρος των χρωμάτων σε 8 bit κυμαίνεται από 0-255. Όπου το 0 σημαίνει μαύρο και το 255 σημαίνει λευκό και το 127 σημαίνει γκρι χρώμα. Αυτή η μορφή χρησιμοποιήθηκε αρχικά από πρώιμα μοντέλα των λειτουργικών συστημάτων UNIX και των αρχικών χρωμάτων Macintoshes.

Παρακάτω εμφανίζεται μια εικόνα σε κλίμακα του γκρι του Αϊνστάιν:



Εικόνα 9 - Άλμπερτ Αϊνστάιν σε εικόνα κλίμακας του γκρι

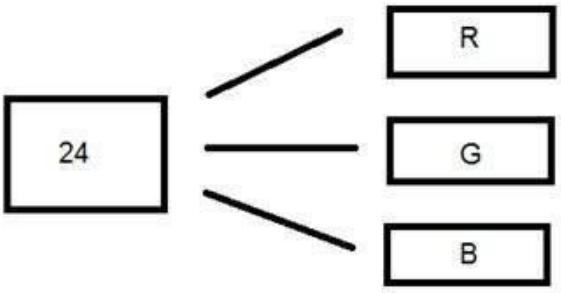
Πίσω από την εικόνα κλίμακας γκρι:

Όπως το έχουμε εξηγήσει και στο κεφάλαιο 3.4, μια εικόνα δεν είναι παρά μια δισδιάστατη συνάρτηση και μπορεί να αναπαρασταθεί από έναν δισδιάστατο πίνακα ή πίνακα. Έτσι, στην περίπτωση της εικόνας του Αϊνστάιν που φαίνεται παραπάνω, θα υπάρχει δισδιάστατος πίνακας πίσω με τιμές που κυμαίνονται μεταξύ 0 και 255.

Αλλά αυτό δεν συμβαίνει με τις έγχρωμες εικόνες.

5.6.2 Μορφή χρώματος 24 bit

Μορφή χρώματος 24 bit γνωστή και ως πραγματική μορφή χρώματος(true color). Σε μορφή χρώματος 24 bit, τα 24 bit διανέμονται σε τρεις διαφορετικές μορφές κόκκινου, πράσινου και μπλε.



Εικόνα 10-Δομή ενός pixel

Δεδομένου ότι το 24 διαιρείται εξίσου στο 8, έτσι κατανέμεται εξίσου μεταξύ τριών διαφορετικών καναλιών χρώματος.

Η διανομή τους είναι έτσι.

8 bit για R, 8 bit για G, 8 bit για B.

Εικόνα 11 - Έγχρωμη Εικόνα

Πίσω από μια εικόνα 24 bit:

Σε αντίθεση με μια εικόνα κλίμακας γκρι 8 bit, η οποία έχει έναν πίνακα πίσω από αυτήν, μια εικόνα 24 bit έχει τρεις διαφορετικούς πίνακες R, G, B.

5.7 RGB to GRAYSCALE conversion

Έχουμε ήδη καθορίσει το μοντέλο χρώματος RGB και τη μορφή γκρι κλίμακας στο κεφάλαιο για τους τύπους εικόνας. Τώρα θα μετατρέψουμε μια έγχρωμη εικόνα σε μια εικόνα σε κλίμακα του γκρι. Υπάρχουν δύο μέθοδοι για τη μετατροπή του. Και οι δύο έχουν τα δικά τους πλεονεκτήματα και μειονεκτήματα. Οι μέθοδοι είναι:

- Μέση μέθοδος
- Σταθμισμένη μέθοδος ή μέθοδος φωτεινότητας

5.7.1 Μέση μέθοδος

μέση μέθοδος είναι η πιο απλή. Απλά πρέπει να λάβετε τον μέσο όρο τριών χρωμάτων. Δεδομένου ότι είναι μια εικόνα RGB, έτσι σημαίνει ότι έχετε προσθέσει r με g με b και στη συνέχεια διαιρέστε το με 3 για να λάβετε την επιθυμητή εικόνα κλίμακας του γκρι.

Έχει γίνει με αυτόν τον τρόπο.

Κλίμακα του γκρι = (R + G + B / 3)

Για παράδειγμα:

Εικόνα 12- Έγχρωμη Εικόνα

Αν έχετε έγχρωμη εικόνα όπως η παραπάνω εικόνα και θέλετε να τη μετατρέψετε σε κλίμακα του γκρι χρησιμοποιώντας τη μέση μέθοδο. Θα εμφανιστεί το ακόλουθο αποτέλεσμα.

Εικόνα 13-Ασπρόμαυρη Εικόνα με μέση μέθοδο

Εξήγηση

Υπάρχει ένα πράγμα που πρέπει να είναι σίγουρο, ότι κάτι συμβαίνει στα πρωτότυπα έργα. Αυτό σημαίνει ότι η μέση μέθοδος μας λειτουργεί. Αλλά τα αποτελέσματα δεν ήταν όπως αναμενόταν. Θέλαμε να μετατρέψουμε την εικόνα σε κλίμακα του γκρι, αλλά αυτό αποδείχθηκε μάλλον μαύρη εικόνα.

Πρόβλημα

Αυτό το πρόβλημα προκύπτει λόγω του γεγονότος ότι παίρνουμε κατά μέσο όρο τα τρία χρώματα. Δεδομένου ότι τα τρία διαφορετικά χρώματα έχουν τρία διαφορετικά μήκη κύματος και έχουν τη δική τους συμβολή στη διαμόρφωση της εικόνας, οπότε πρέπει να

πάρουμε τον μέσο όρο σύμφωνα με τη συμβολή τους, όχι να το κάνουμε κατά μέσο όρο χρησιμοποιώντας τη μέση μέθοδο. Αυτή τη στιγμή αυτό που κάνουμε είναι αυτό,

33% του κόκκινου, 33% του πράσινου, 33% του μπλε

Παίρνουμε το 33% του καθενός, αυτό σημαίνει ότι κάθε τμήμα έχει την ίδια συνεισφορά στην εικόνα. Αλλά στην πραγματικότητα αυτό δεν συμβαίνει. Η λύση σε αυτό δόθηκε με τη μέθοδο φωτεινότητας.

Έχετε δει το πρόβλημα που εμφανίζεται στη μέση μέθοδο.

5.7.2 Η Σταθμισμένη μέθοδος ή μέθοδος φωτεινότητας

Η σταθμισμένη μέθοδος έχει μια λύση σε αυτό το πρόβλημα. Δεδομένου ότι το κόκκινο χρώμα έχει μεγαλύτερο μήκος κύματος και των τριών χρωμάτων, και το πράσινο είναι το χρώμα που όχι μόνο έχει μικρότερο μήκος κύματος από το κόκκινο χρώμα, αλλά και το πράσινο είναι το χρώμα που δίνει πιο χαλαρωτική επίδραση στα μάτια.

Αυτό σημαίνει ότι πρέπει να μειώσουμε τη συνεισφορά του κόκκινου χρώματος και να αυξήσουμε τη συνεισφορά του πράσινου χρώματος και να βάλουμε τη συνεισφορά μπλε χρώματος μεταξύ αυτών των δύο.

Έτσι, η νέα εξίσωση που σχηματίζεται είναι:

Νέα εικόνα κλίμακας του γκρι = ((0,3 * R) + (0,59 * G) + (0,11 * B)).

Σύμφωνα με αυτήν την εξίσωση, το κόκκινο έχει συνεισφέρει 30%, το πράσινο έχει συνεισφέρει 59% που είναι μεγαλύτερο και στα τρία χρώματα και το μπλε έχει συνεισφέρει 11%.

Εφαρμόζοντας αυτήν την εξίσωση στην εικόνα, το παίρνουμε

Αρχική εικόνα:

Εικόνα 14- Έγχρωμη Εικόνα

Εικόνα κλίμακας του γκρι:

Εικόνα 15-Ασπρόμαυρη εικόνα με σταθμισμένη μέθοδο

Εξήγηση

Όπως μπορείτε να δείτε εδώ, ότι η εικόνα έχει πλέον μετατραπεί σωστά σε κλίμακα του γκρι χρησιμοποιώντας σταθμισμένη μέθοδο. Σε σύγκριση με το αποτέλεσμα της μέσης μεθόδου, αυτή η εικόνα είναι πιο φωτεινή.

6 Κεφάλαιο περιγραφή υλικού

Σε αυτό το κεφάλαιο θα αναφερθούμε στο το υλικό που χρησιμοποιήσαμε για την υλοποίηση της εφαρμογής μας και επίσης θα περιγράψουμε την αρχιτεκτονική του συστήματος με περισσότερη έμφαση στα δομικά στοιχεία που χρησιμοποιήθηκαν άμεσα.

6.1 Zedboard

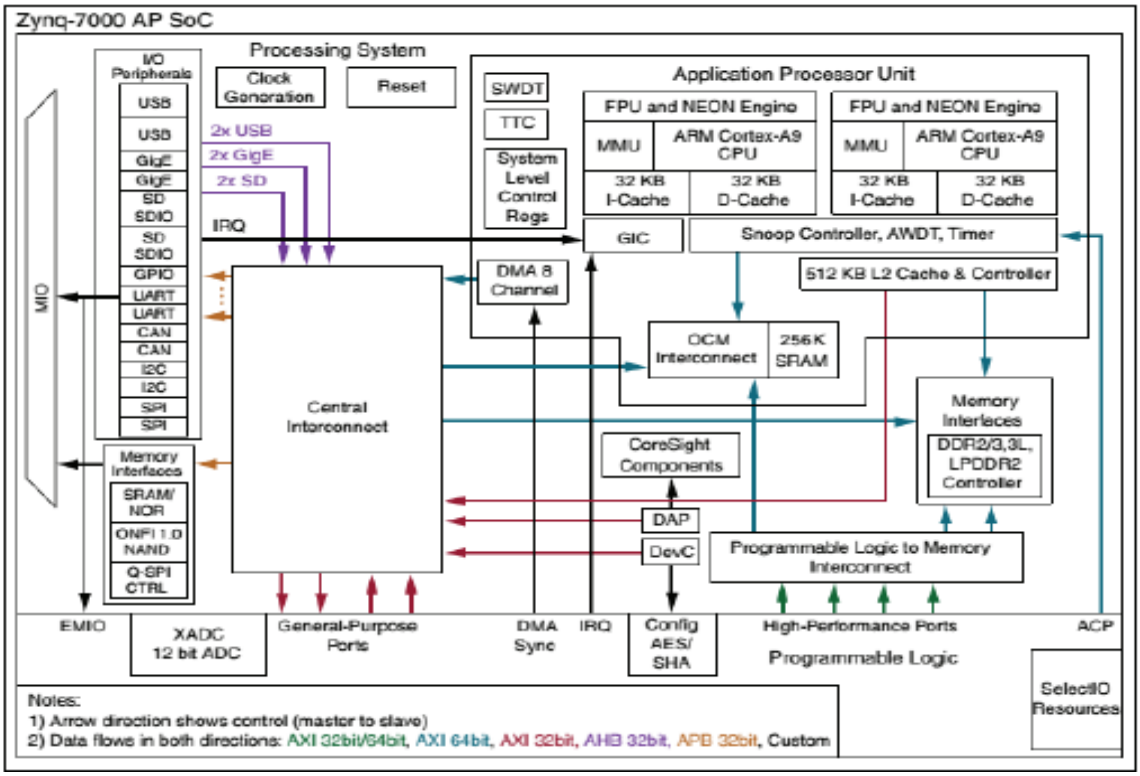
Η πλατφόρμα που χρησιμοποιήσαμε για την υλοποίηση της εφαρμογής μας είναι το zedboard,το οποίο είναι ένα πλήρες πακέτο ανάπτυξης εφαρμογών ενσωματωμένων συστημάτων οποία βασίζεται στο Zynq®-7000 SoC [21].Το zed board περιέχει όλα τα απαραίτητα στοιχεία και διεπαφές για την δημιουργία μιας ευρείας γκάμας εφαρμογών όπως για παράδειγμα εφαρμογές επεξεργασίας video και εικόνας, ανάπτυξη λειτουργικών

συστημάτων, εφαρμογές ενσωματωμένων συστημάτων κλπ.

Εικόνα 16-Zedboard

6.2 Zynq®-7000 Soc

Το Zynq®-7000 SoC είναι βασισμένο στην αρχιτεκτονική Xilinx® SoC και ενσωματώνει έναν διπύρηνο επεξεργαστή ARM® CortexTM-A9 μαζί με programmable logic (PL) σε μία μόνο πλακέτα η οποία είναι σύγχρονη , προσφέρει υψηλή απόδοση, χαμηλή κατανάλωση ενέργειας αλλά και ευελιξία. Η καρδιά του συστήματος επεξεργασίας είναι οι πυρήνες του επεξεργαστή ARM Cortex-A9 αλλά το σύστημα επεξεργασίας εμπεριέχει και on-chip μνήμη , διεπαφή για εξωτερική μνήμη αλλά και ένα πλούσιο σετ με περιφερειακά εισόδου εξόδου I/O.



Εικόνα 17-Αρχιτεκτονική Zynq-7000 SoC

6.3 Το επεξεργαστικό σύστημα (PS)

[21]Το επεξεργαστικό απαρτίζεται από 4 βασικά κομμάτια τα οποία και θα αναλύσουμε στην συνέχεια. Αυτά είναι τα ακόλουθα:

- Μονάδα επεξεργασίας εφαρμογών(Application Processor Unit (APU))
- Διεπαφές μνήμης(memory interfaces)
- Περιφερικά εισόδου εξόδου(I/O peripherals)
- Διασυνδέσεις (interconnect)

6.3.1 Μονάδα επεξεργασίας εφαρμογών (APU)

Η μονάδας επεξεργασίας(APU) απαρτίζεται από έναν διπύρηνο επεξεργαστή ARM Cortex-A9 οπού ο κάθε πυρήνας έχει συχνότητα επεξεργασίας στα 667 MHz, υποστηρίζει επεξεργασία αριθμών κινητής υποδιαστολής για μονή και διπλή ακρίβεια, έχει την δυνατότητα υποστήριξης αρχιτεκτονικής SIMD(single instruction multiple data)μέσω της μονάδας NEON που διαθέτει κάθε επεξεργαστικός πυρήνας, υποστηρίζει συμπίεση κώδικα μέσω του Thumb®-2 support,επίσης ο κάθε επεξεργαστής διαθέτει μνήμες Level 1 caches(ξεχωριστά εντολές και δεδομένα 32kb η κάθε μνήμη),επιπρόσθετα διαθέτει και memory management unit(MMU),τέλος αξίζει να αναφέρουμε ότι παρέχεται η δυνατότητα λειτουργίας ως single processor, symmetric dual processor, και asymmetric dual processor αλλά και δυνατότητα ασφαλούς λειτουργίας μέσω της τεχνολογίας TrustZone®.

Εκτός από τις δυνατότητες που έχει ο κάθε πυρήνας ξεχωριστά όπως αναφέραμε παραπάνω υπάρχουν και κάποια ακόμα χαρακτηριστικά που διαθέτει η μονάδα επεξεργασίας(APU,

διαθέτει κοινή level 2 cache (512 kb) για τους 2 πυρήνες, διαθέτει μια πολύ γρήγορη on chip RAM μνήμη(256kb) η οποία μπορεί να προσπελαστεί από την CPU και την programmable logic(PL) και είναι ιδικά σχεδιασμένη για να μην προκύπτουν μεγάλες καθυστερήσεις όταν την προσπελαύνει η CPU.Επίσης περιέχει και General interrupt controller (GIC) ,τρείς watch dog timers (WDT),και δύο τριπλούς timers/counters (TTC).

6.3.2 Διεπαφές μνήμης(memory interfaces)

Η μονάδα διεπαφής μνήμης περιλαμβάνει έναν δυναμικό ελεγκτή μνήμης και modules διεπαφής στατικής μνήμης Ο ελεγκτής δυναμικής μνήμης υποστηρίζει μνήμες DDR3, DDR3L, DDR2 και LPDDR2. Οι ελεγκτές στατικής μνήμης υποστηρίζουν μία NAND διεπαφή flash, διεπαφή flash Quad-SPI, παράλληλος δίαυλος δεδομένων και παράλληλη διεπαφή φλας NOR.

Δυναμικές διεπαφές μνήμης(Dynamic Memory Interfaces)

Ο ελεγκτής μνήμης DDR πολλαπλών πρωτοκόλλων(DDR memory controller) μπορεί να ρυθμιστεί ώστε να παρέχει προσβάσεις 16-bit ή 32-bit σε εύρος διευθύνσεων 1 GB χρησιμοποιώντας μια διαμόρφωση μονής κατάταξης με μνήμες DRAM 8-bit, 16-bit ή 32-bit. Το ECC υποστηρίζεται με πρόσβαση σε δίαυλο(bus) 16 bit . Το PS ενσωματώνει τόσο τον ελεγκτή DDR όσο και το σχετικό PHY, συμπεριλαμβανομένου του δικού του αποκλειστικού σετ εισόδου/εξόδου. Ταχύτητα υποστηρίζεται έως 1333 Mb / s για DDR3

Ο ελεγκτής μνήμης DDR έχει πολλές θύρες και επιτρέπει στο σύστημα επεξεργασίας(PS) και στην προγραμματιζόμενη λογικής(PL)πρόσβαση σε μια κοινή μνήμη. Ο ελεγκτής DDR διαθέτει τέσσερις θυρίδες AXI slave για αυτόν τον σκοπό:

• Μια θήρα 64-bit είναι αφιερωμένη για τον επεξεργαστή ARM μέσω του ελεγκτή προσωρινής μνήμης cache L2 και μπορεί να διαμορφωθεί για χαμηλές καθυστερήσεις.

• Δύο 64-bit θήρες προορίζονται για πρόσβαση από την PL

• Μία θύρα AXI 64-bit μοιράζεται από όλους τους άλλους Masters AXI μέσω της κεντρικής διασύνδεσης.

Στατικές διεπαφές μνήμης(Static Memory Interfaces)

Οι διεπαφές στατικής μνήμης υποστηρίζουν εξωτερικές στατικές μνήμες:

• Δίαυλος δεδομένων SRAM 8-bit που υποστηρίζει έως 64 MB

• 8-bit παράλληλο flash NOR που υποστηρίζει έως 64 MB

• Υποστήριξη flash ONFi 1.0 NAND με 1-bit ECC • 1-bit SPI, 2-bit SPI, 4-bit SPI (quad-SPI) ή two quad-SPI (8-bit) serial NOR flash

6.3.3 Περιφερικά εισόδου εξόδου (I/O peripherals (IOP)

Τα περιφερικά εισόδου εξόδου είναι μια συλλογή από διεπαφές βιομηχανικού προτύπου που προορίζονται για εξωτερική επικοινωνία. Κάποια από τα περιφερικά αυτά είναι 192 general purpose input output(GPIO)signals για επικοινωνία του PS με το PL μέσω μιας μονάδας που ονομάζεται EMIO ,και 54 GPIO signals για πινάκια πάνω την συσκευή, επίσης δύο gigabit Ethernet controllers,usb controllers,δύο SPI controllers(master-slave),δύο SD/SDIO controllers,δυο can ,δύο uart και δύο i2c controllers,και τέλος PS MIO I/O.

6.3.4 Διασυνδέσεις (interconnect)

Η Μονάδα επεξεργασίας εφαρμογών (APU) η μονάδα διεπαφής μνήμης και τα περιφερικά εισόδου εξόδου είναι όλα συνδεδεμένα μεταξύ τους και με το PL μέσω μίας πολυεπίπεδης ARM διασύνδεσης AMBA AXI. Η διασύνδεση δεν αποκλείει και υποστηρίζει πολλαπλές ταυτόχρονες συναλλαγές master-slave.

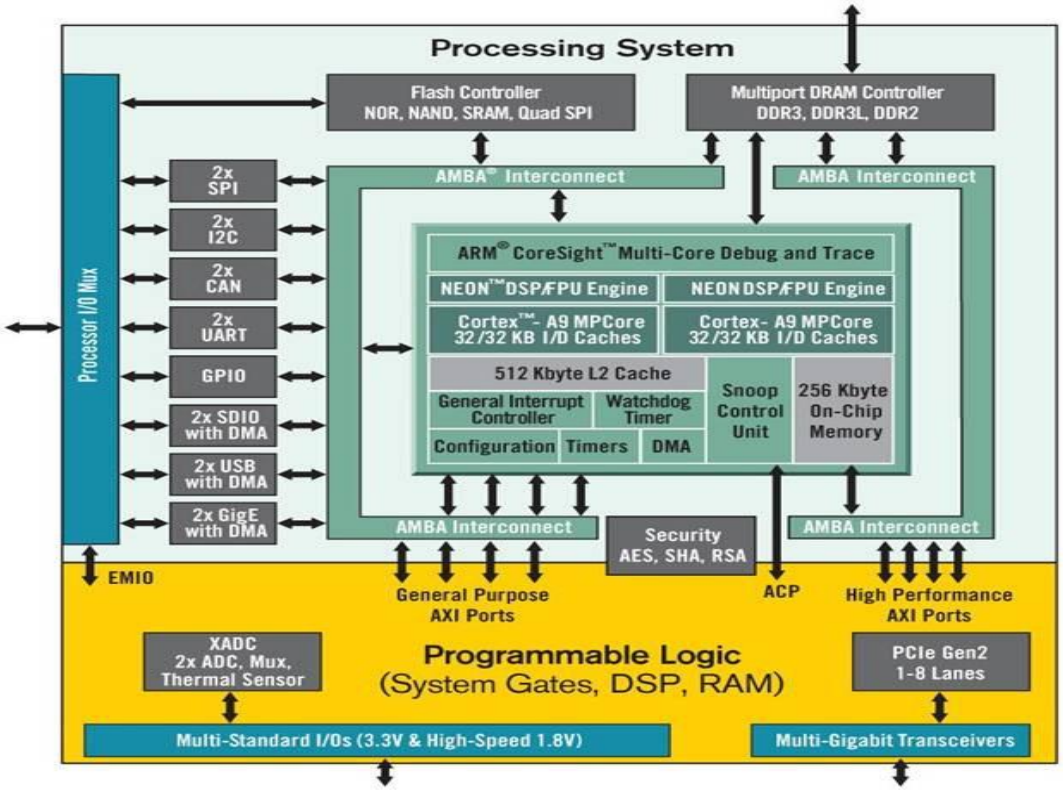
Η διασύνδεση έχει σχεδιαστεί έτσι ώστε οι συνδέσεις μεταξύ τον master και τον slaves είναι όσο το δυνατό πιο αποδοτική από άποψη καθυστερήσεων όσο και από άποψη bandwidth , επίσης και η CPU ARM, να έχει τις μικρότερες διαδρομές προς τη μνήμη.

Η κυκλοφορία(traffic) μέσω της διασύνδεσης μπορεί να ρυθμιστεί μέσω του μπλοκ Quality of Service (QoS) στη διασύνδεση. Η λειτουργία QoS χρησιμοποιείται για τη ρύθμιση της κίνησης που δημιουργείται από τον επεξεργαστή, τον ελεγκτή DMA και μια συνδυασμένη οντότητα που αντιπροσωπεύει τους masters στις διεπαφές εισόδου εξόδου.

6.4 Programmable logic (PL)

Ως προγραμματιζόμενη λογική (PL) αναφέρεται εκείνο το κομμάτι του zynq 7000 το οποίο προσφέρει στον χρήστη μια πλούσια αρχιτεκτονική για να την διαμορφώσει ο ίδιος όπως ταιριάζει στις ανάγκες του.

Η PL προσφέρει διαμορφώσιμα λογικά μπλοκ (configurable logic blocks(CLB) τα οποία δίνουν και την δυνατότητα χρήσης look up tables τα οποία έχουν και δυνατότητα μνήμης, επίσης προσφέρει 36kb block ram μνήμη ,διαχείριση ρολογιών ,μονάδα για ψηφιακή επεξεργασία σήματος DSP48E1,διαμορφόσημα input/output και πομποδέκτες gigabit χαμηλής ισχύος.

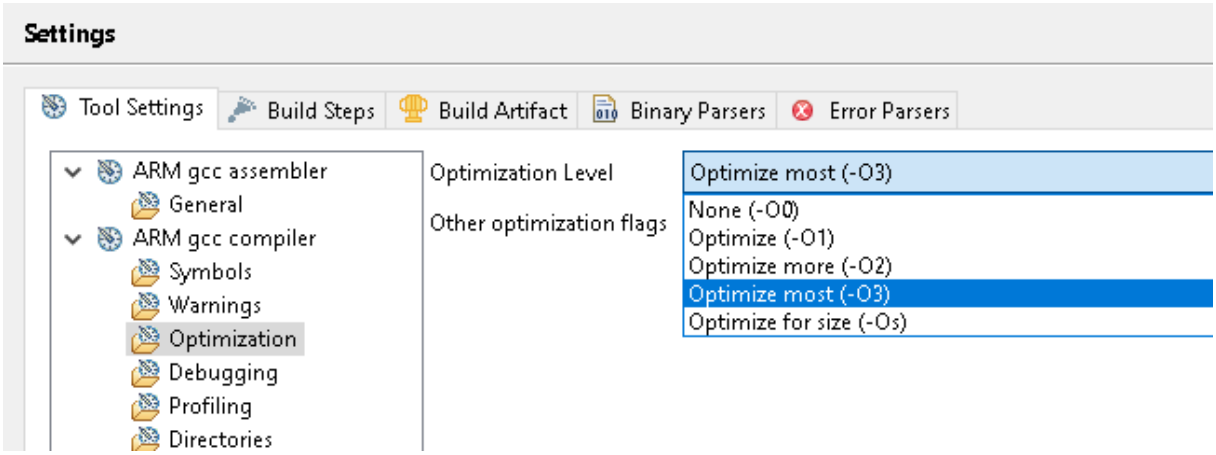


Εικόνα 18 - Processing System and Programmable logic

7. Optimization

7.1 Code optimization

Το λογισμικό παράχθηκε γι αυτήν την εργασία στο πρόγραμμα SDK της Xilinx το οποίο έδινε επιλογές optimization στον compiler για βελτιστοποίηση κώδικα. Συνδυαστικά λοιπόν με την τεχνική fixed point arithmetic έγινε χρήση του optimization του compiler σε level 3 για να πετύχουμε την βέλτιστη απόδοση.



Εικόνα 19- optimization options

7.2 Optimization flags for FPU

Επίσης το SDK δίνει την δυνατότητα να διαχειριστούμε τους αριθμούς κινητής υποδιαστολής με τρείς διαφορετικές επιλογές αναλόγως με το αν γνωρίζουμε την ύπαρξη FPU στο σύστημά μας [22].

Η εντολή που πρέπει να δώσουμε στο πεδίο other optimization flags είναι η εξής

-mfloat-abi=name

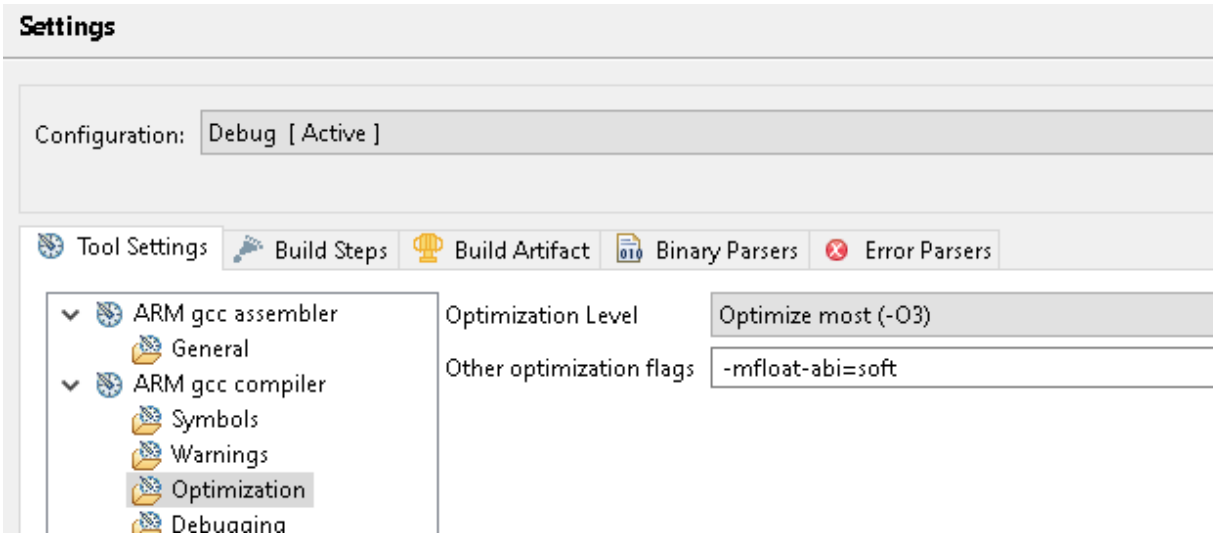
Όπου η μεταβλητή name μπορεί να πάρει 3 τιμές soft,softfp,hard.

soft: Υποθέτει ότι το σύστημα δεν διαθέτει μονάδα FPU και όλες οι πράξεις μεταξύ πραγματικών αριθμών γίνονται μέσω έτοιμων συναρτήσεων, οι καταχωρητές που χρησιμοποιούνται είναι εκείνοι της CPU η μέθοδος αυτή είναι αρκετά χρονοβόρα αλλά απαραίτητη σε πλατφόρμες που δεν διαθέτουν FPU.

softfp: Ελέγχει αν το σύστημα διαθέτει FPU. Οι αριθμοί φορτώνονται στους καταχωρητές της CPU.Στην περίπτωση που το σύστημα διαθέτει FPU μεταφέρει τους αριθμούς κινητής υποδιαστολής στους καταχωρητές της FPU και την χρησιμοποίει για τις πράξεις των αριθμών κινητής υποδιαστολής. Η μέθοδος αυτή προσφέρει περισσότερη ευελιξία στην εφαρμογή μας

hard: Υποθέτει ότι το σύστημα διαθέτει FPU και φορτώνει κατευθείαν τους αριθμούς κινητής υποδιαστολής στους καταχωρητές της FPU και την χρησιμοποίει για τις πράξεις. Οι επιλογές softfp hard χρησιμοποιούν και οι 2 την FPU άμα υπάρχει αλλά στην επιλογή hard αποφεύγεται η μεταφορά από τους καταχωρητές της CPU σε αυτούς της FPU.Συνεπώς η τρίτη επιλογή δίνει καλύτερα αποτελέσματα όταν είμαστε σίγουροι ότι υπάρχει FPU στο σύστημα.

Ακολουθεί παράδειγμα χρήσης του soft.



8. Περιγραφή λογισμικού

8.1 Λογισμικό για fixed point arithmetic

Για να υλοποιηθεί σε λογισμικό η τεχνική fixed point arithmetic όπως την περιγράψαμε στο κεφάλαιο 2 στην αρχή του προγράμματος να δηλώθηκε μια σταθερά που ορίζει πόσα από τα bits του fixed point αριθμού μας θα χρησιμοποιηθούν για το κλασματικό μέρος της αναπαράστασης

```
//define how many bits do i need for fractional part
#define FIXED_POINT_FRACTIONAL_BITS 8
```

Στην συνέχεια περιγράψαμε τον τύπο fixed_point_t

```
/// Fixed-point Format (16-bit)
typedef unsigned short int fixed_point_t;
```

Στην συνέχεια φτιάχτηκαν οι συναρτήσεις για μετατροπή από fixed point σε float

```
inline double fixed_to_float(fixed_point_t input)
{
    return ((float)input / (float)(1 << FIXED_POINT_FRACTIONAL_BITS));
}
```

και το αντίστροφο

```
inline fixed_point_t float_to_fixed(float input)
{
    return (fixed_point_t)(round(input * (1 << FIXED_POINT_FRACTIONAL_BITS)));
}
```

Και τέλος οι συναρτήσεις για να γίνονται οι πράξεις μεταξύ fixed point αριθμών

Πρόσθεση

```
fixed_point_t fixed_add(fixed_point_t x, fixed_point_t y)
{
    return x+y;
}
```

Αφαίρεση

```
fixed_point_t fixed_sub(fixed_point_t x, fixed_point_t y)
{
    return x-y;
}
```

Πολλαπλασιασμός

```
fixed_point_t fixed_mul(fixed_point_t x, fixed_point_t y)
{
    return (((int32_t)x * (int32_t)y) / (1 << FIXED_POINT_FRACTIONAL_BITS));
}
```

Διαίρεση

```
fixed_point_t fixed_div(fixed_point_t x, fixed_point_t y)
{
    return ((int32_t)x * (1 << FIXED_POINT_FRACTIONAL_BITS)) / y;
}
```

8.2 Περιγραφή λογισμικού RGB to Grayscale

Στην εφαρμογή αυτή οι τιμές των έγχρωμων εικόνων μας δίνονται έτοιμες σε έναν μονοδιάστατο πίνακα που βρίσκεται σε ένα ξεχωριστό αρχείο γλώσσας C.

code.c | lscript.ld | image_5x5.c

```
* image

typedef struct {
#include <stdint.h>


static const uint8_t image[75] = {
    0x7c, 0x7e, 0x71, 0x83, 0x85, 0x78, 0x8e, 0x90, 0x83, 0x96, 0x98, 0x8b, 0x9a, 0x9c, 0x8f,
    0x7d, 0x7f, 0x72, 0x84, 0x86, 0x79, 0x8e, 0x90, 0x83, 0x96, 0x98, 0x8b, 0x9a, 0x9c, 0x8f,
    0x7d, 0x7f, 0x72, 0x84, 0x86, 0x79, 0x8e, 0x90, 0x83, 0x96, 0x98, 0x8b, 0x9a, 0x9c, 0x8f,
    0x7f, 0x81, 0x74, 0x85, 0x87, 0x7a, 0x8f, 0x91, 0x84, 0x96, 0x98, 0x8b, 0x9a, 0x9c, 0x8f,
    0x80, 0x82, 0x75, 0x86, 0x88, 0x7b, 0x8f, 0x91, 0x84, 0x96, 0x98, 0x8b, 0x99, 0x9b, 0x8e
};
```

Εικόνα 20-Κώδικας για αποθήκευση στοιχείων έγχρωμης εικόνας σε πίνακα

Ο πίνακας αυτός περιέχει τις τιμές ενός pixel της εικόνας ανά 3 θέσεις του πίνακα δηλαδή μια τιμή 8 bit για το κόκκινο μια για το πράσινο και μία για το μπλε, γι αυτό και χρησιμοποιείτε το ο τύπος δεδομένων uint8_t.

Επίσης δηλώνουμε και τον πίνακα gray_image στον οποίο θα βάλουμε τις τιμές της γκρίζας εικόνας που θα παράγουμε.

```
// matrix for the gray scale result
uint8_t gray_image[line*column];
```

Εικόνα 21-Κώδικας για δημιουργία πίνακα που θα αποθηκεύσω τις τιμές τις ασπρόμαυρης εικόνας

Στην συνέχεια εμείς παίρνουμε την τιμή του κάθε χρώματος για το συγκεκριμένο pixel την πολλαπλασιάζουμε με τον κατάλληλο συντελεστή για το κάθε χρώμα. Οι συντελεστές φαίνονται στην παρά κάτω εικόνα.

```
//coefficients for RGB to grayscale
float R=0.298936,G=0.587043,B=0.114021;
fixed_point_t fixed_R,fixed_G,fixed_B;
```

Εικόνα 22-Κώδικας για Συντελεστές RGB to Grayscale

Στην συνέχεια αθροίζουμε τις τιμές αυτές για το κάθε pixel και έτσι προκύπτουν οι τιμές της grayscale εικόνας μας. Δηλαδή κάθε 3 τιμές του πίνακα image θα μας δίνουν και μία τιμή για τον πίνακα gray_image..

Η διαδικασία αυτή υλοποιείτε με τον παρά κάτω κώδικα και επιπρόσθετα υπάρχουν εντολές για τον hardware timer που χρησιμοποιούμε για τις μετρήσεις μας.

```
// Start the timer runnin (it counts down)
temp= XScuTimer_GetCounterValue(&my_Timer);
XScuTimer_Start(&my_Timer);
for(i=0;i<line*column*chanel;i+=3)
{
        gray_image[j]=round(R*image[i]+G*image[i+1]+B*image[i+2]);
        //gray_image[j]=fixed_mul(fixed_R,image[i])+fixed_mul(fixed_G,image[i+1])+fixed_mul(fixed_B,image[i+2]);
        j+=1;
}
// Read the value of the timer
XScuTimer_Stop(&my_Timer);
timer_value= XScuTimer_GetCounterValue(&my_Timer);
```

Εικόνα 23-Κώδικας για RGB to Grayscale μετατροπή

Η ίδια διαδικασία γίνεται και όταν οι συντελεστές μας για τα χρώματα μετατρέπονται σε fixed point αριθμούς απλός υλοποιείτε η εντολή που υπάρξει σε σχόλιο από πάνω.

```
fixed_R=float_to_fixed(R);
fixed_G=float_to_fixed(G);
fixed_B=float_to_fixed(B);
//printf("fixed_red=%d\nfixed_green=%d\nfixed_blue=%d\n",fixed_R,fixed_G,fixed_B);
```

Στόχος είναι να μετρήσουμε πόσους κύκλους ρολογιού του hardware timer χρειάζεται η κάθε μέθοδος για να κάνει την μετατροπή.

Εικόνα 24-Κώδικας για μετατροπή συντελεστών σε fixed point αναπαράσταση

8.3 Περιγραφή λογισμικού για real time επεξεργασία εικόνας.

Τέλος δημιουργήθηκε η εφαρμογή που ανάλογα με το κατώφλι χρόνου πού ορίζεται στο πρόγραμμα γίνεται επεξεργασία εικόνας συνδυαστικά με πλήρη και μειωμένη ακρίβεια για να προλάβει να ολοκληρώσει την επεξεργασία μέσα στα χρονικά όρια.

Οι μεταβλητές που χρησιμοποιήσαμε για να γίνει αυτό είναι οι ακόλουθες

```
// variables for time limits
float plhrhs=900,meiwmenh=300,threshold=600,euros,timi;
euros=plhrhs-meiwmenh;
timi=threshold-meiwmenh;
float pososto_plhrhs;
```

Ξέραμε από τις μετρήσεις μας ότι για να γίνει επεξεργασία 5χ5 εικόνας με πλήρη ακρίβεια χρειάστηκαν 900 cc και για να γίνει η επεξεργασία με μειωμένη ακρίβεια χρειάζονταν 300 cc

Έτσι ανάλογα με το κατώφλι (threshold) που δίνουμε υπολογίζεται πόσο ποσοστό της εικόνας πρέπει να υπολογιστεί με πλήρη και πόσο με μειωμένη ακρίβεια έτσι ώστε να γίνει η επεξεργασία μέσα στο όριο χρόνου που έχουμε δηλώσει στο threshold.

Αυτό γίνεται στον κώδικα που ακολουθεί

```
//FULL AND REDUCED PRECISION
            }else if(threshold>meiwmenh && threshold<plhrhs)
            {
                pososto_plhrhs=timi/euros*100;
                pososto_plhrhs=floor(pososto_plhrhs*100)/100;
                float posa_stoixia=floor(pososto_plhrhs/100*line*column);
                float p,k;
                p=posa_stoixia*chanel;
                k=(line*column*chanel)-p;
                for(i=0;i<posa_stoixia*chanel;i+=3)
                {
                    gray_image[j]=round(R*image[i]+G*image[i+1]+B*image[i+2]);
                    printf("%d ",gray_image[j]);
                    j++;
                }
                for(i=p;i<p+k;i+=3)
                {
                    gray_image[j]=fixed_mul(fixed_R,image[i])+fixed_mul(fixed_G,image[i+1])+fixed_mul(fixed_B,image[i+2]);
                    printf("%d ",gray_image[j]);
                    j++;
                }

            }
```

Στις περιπτώσεις που το threshold είναι μικρότερο από τα όρια που έχουμε δηλώσει γίνεται μετατροπή με πλήρη ή μειωμένη ακρίβεια αντίστοιχος ή εμφανίζεται μήνυμα ότι ο χρόνος δεν αρκεί.

9. Αποτελέσματα-Μετρήσεις

Στο κεφάλαιο αυτό θα γίνει παρουσίαση των αποτελεσμάτων των και το μετρήσεων που πήραμε από τις παραπάνω εφαρμογές.

Για την μέτρηση του χρόνου χρησιμοποιήθηκε ο ScuTimer ο οποίος και είχε αρχικοποιηθεί με την μισή συχνότητα του Cortex-A9 (666 MHz είναι η συχνότητα του A9)

Ο κώδικας καθώς και τα δεδομένα ήταν αποθηκευμένα στην μνήμη DDR του συστήματος και η μνήμη cache ήταν ενεργοποιημένη.

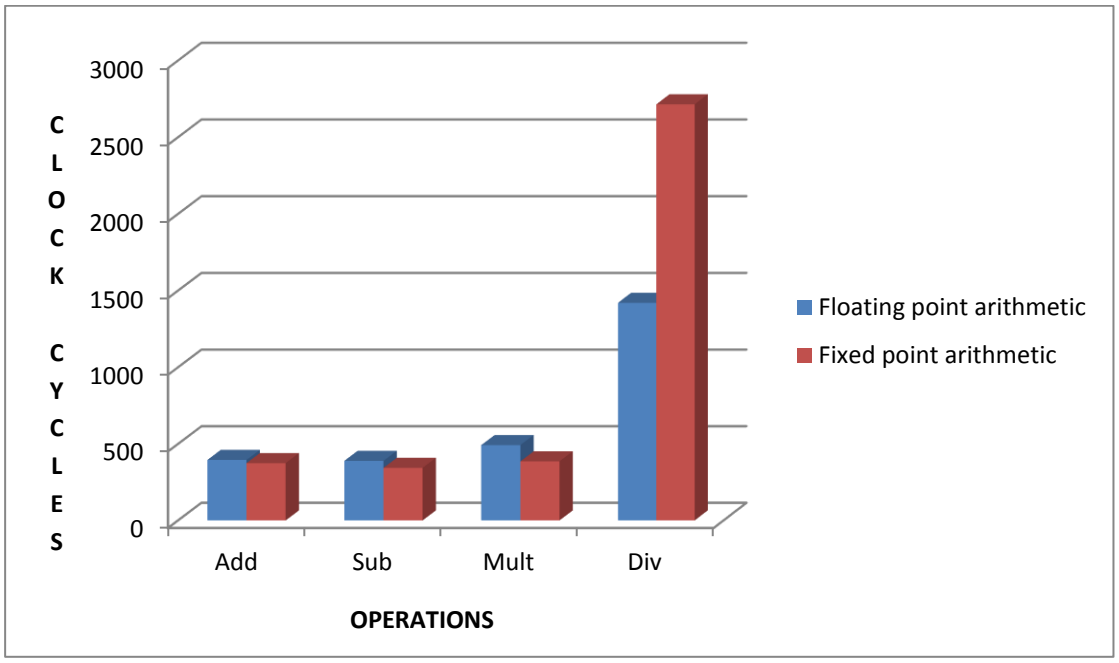
9.1 Πράξεις μεταξύ πινάκων

Σε αυτό το παράδειγμα μετρήσαμε τους χρόνους σε κύκλους ρολογιού cc του ScuTimer που χρειάζονται για να υλοποιηθούν οι 4 βασικές πράξεις μεταξύ δύο πινάκων που οι τιμές τους είναι δεκαδικοί αριθμοί

A) B) Για πίνακες 1χ100 με ενεργοποιημένη την μνήμη cache και **level 3 optimization.** Η
B) Μνήμη στην οποία ήταν αποθηκευμένα τα δεδομένα ήταν η DDR

Πράξεις μεταξύ πινάκων	
	Clock cycles

	Floating point arithmetic	Fixed point arithmetic
Add	395	374
Sub	389	343
Mult	492	386
div	1420	2714



Εικόνα 25-fixed vs float πράξεις σε 1x100 πίνακες

9.2.RGB TO GRAYSCALE IMAGE CONVERSION

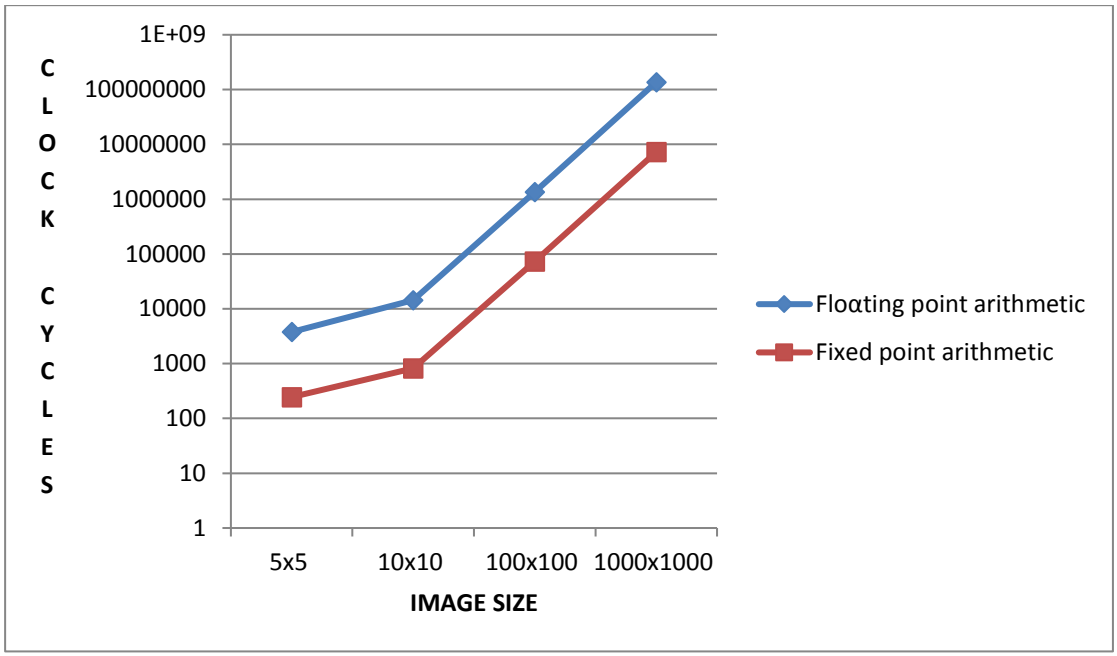
Σε αυτό το παράδειγμα μετρήσαμε τον χρόνο σε κύκλους ρολογιού του cc του ScuTimer που χρειάστηκε για να ολοκληρωθεί η μετατροπή εικόνας από έγχρωμη εικόνα με βάθος χρώματος 24 bit με πλήρη και μειωμένη ακρίβεια.

Η μνήμη στην οποία ήταν αποθηκευμένα τα δεδομένα ήταν η DDR και χρησιμοποιήσαμε **optimization lvl 3.**

Με επιλογή soft

Μετατροπή από RGB σε Grayscale με soft		
	Clock cycles	
	Floating point arithmetic	*Fixed point arithmetic*
5x5 image	*3777*	*245*
10x10 image	*14327*	*821*

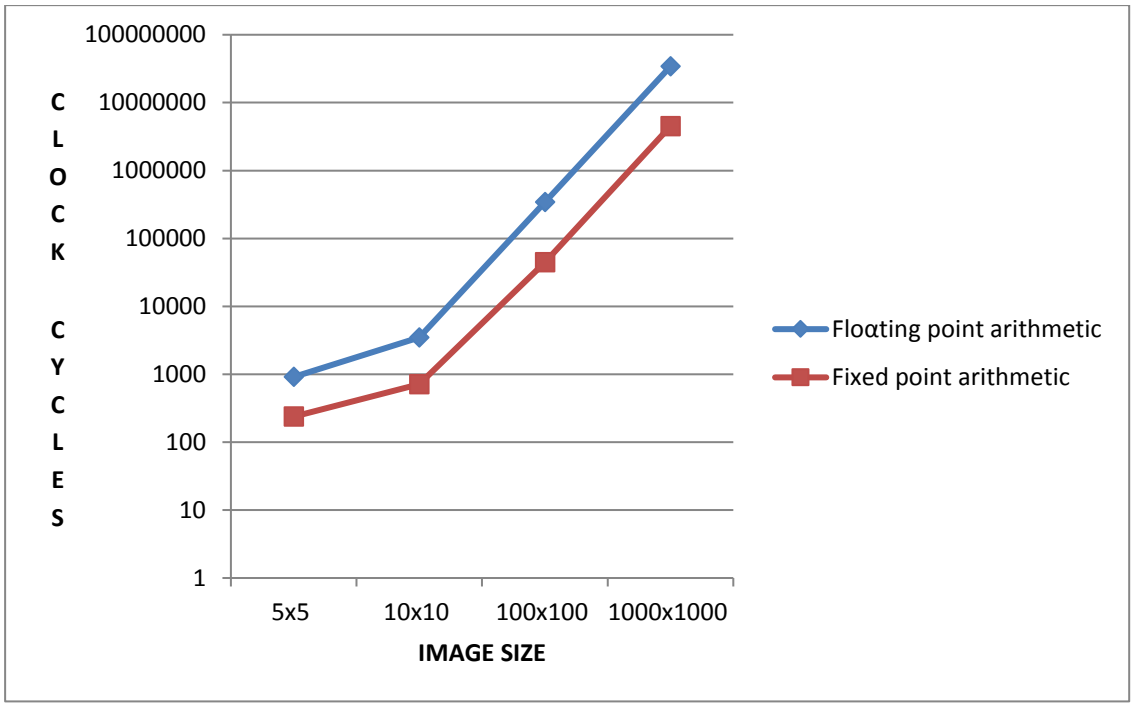
100x100 image	*1360678*	*72722*
1000x1000 image	*137048235*	*7274021*



Εικόνα 26-Αυξηση χρόνου ανά μέγεθος με επιλογή soft

Με επιλογή softfp

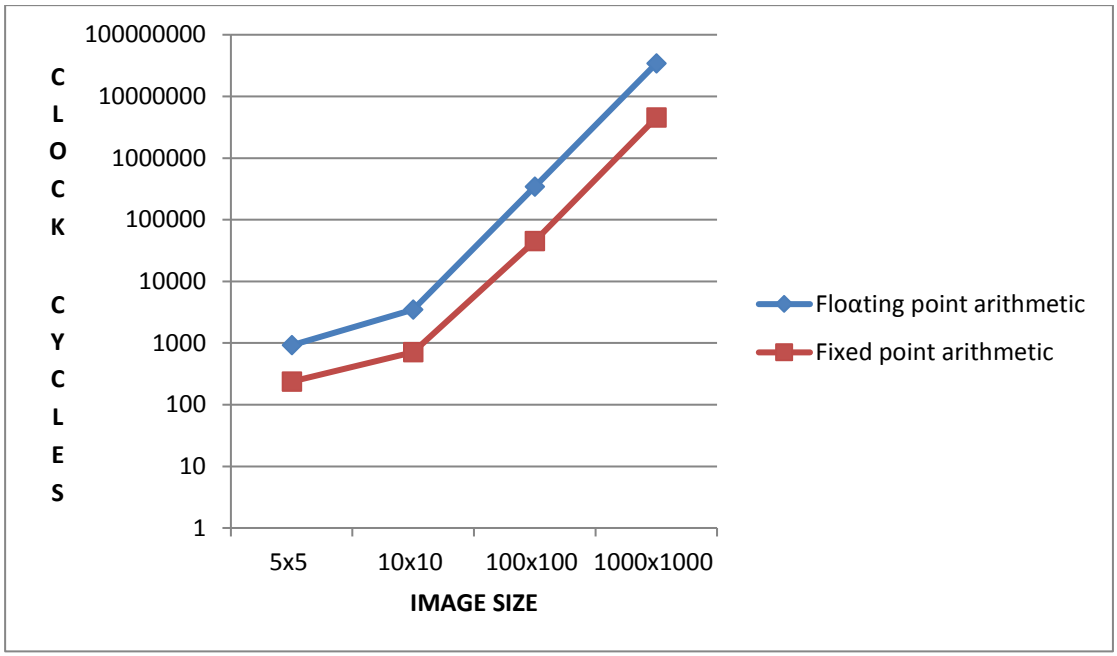
Μετατροπή από RGB σε Grayscale με softfp		
	Clock cycles	
	Floating point arithmetic	*Fixed point* arithmetic
5x5 image	*913*	*238*
10x10 image	*3481*	*711*
100x100 image	*342091*	*44895*
1000x1000 image	*34213648*	*4514491*



Εικόνα 27-Αυξηση χρόνου ανά μέγεθος με επιλογή softfp

Με επιλογή hard

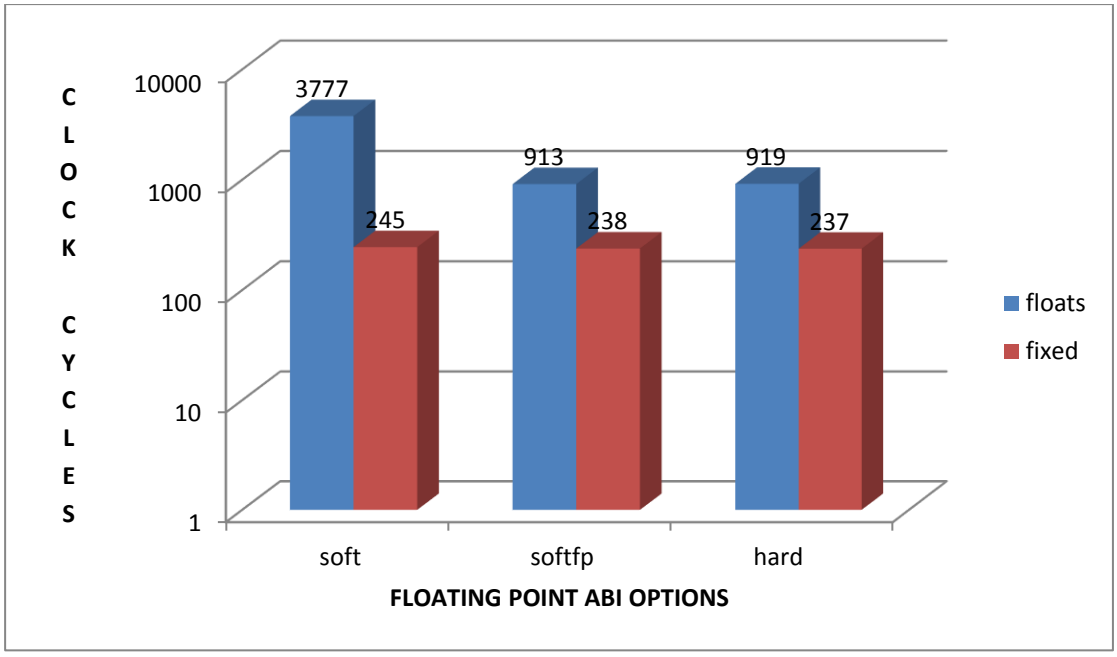
Μετατροπή από RGB σε Grayscale με hard		
	Clock cycles	
	Floating point arithmetic	*Fixed point* arithmetic
5x5 image	*919*	*237*
10x10 image	*3484*	*710*
100x100 image	*342152*	*44870*
1000x1000 image	*34213654*	*4514827*



Εικόνα 28-Αυξηση χρόνου ανά μέγεθος με επιλογή hard

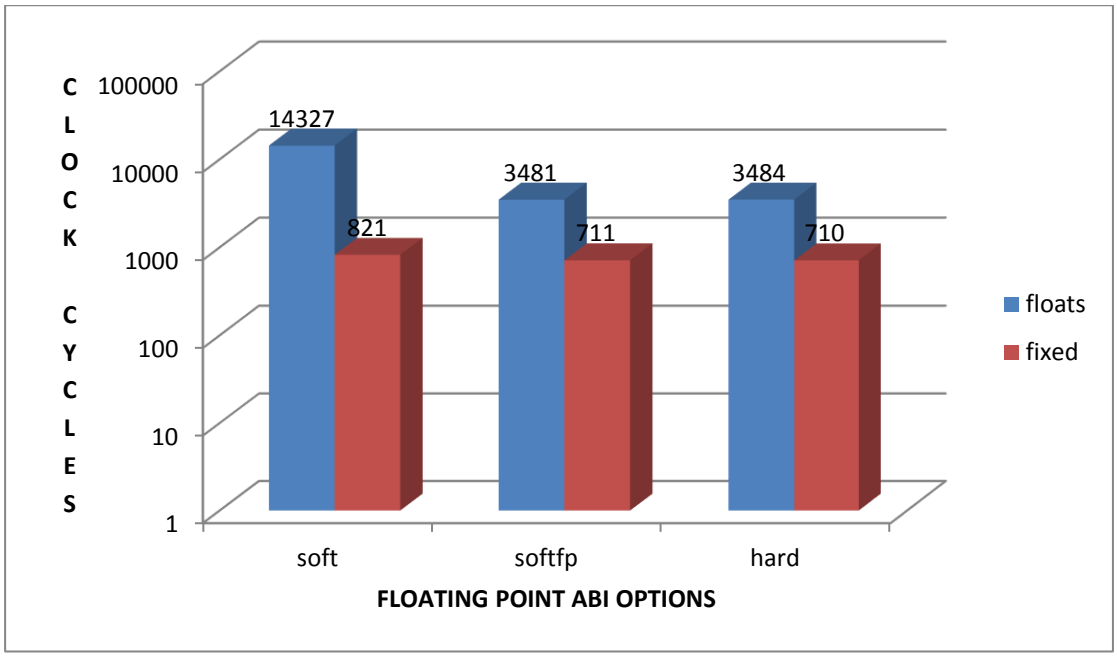
Σύγκριση μεταξύ soft,softfp,hard ανά μέγεθος εικόνας

Για 5χ5 εικόνα



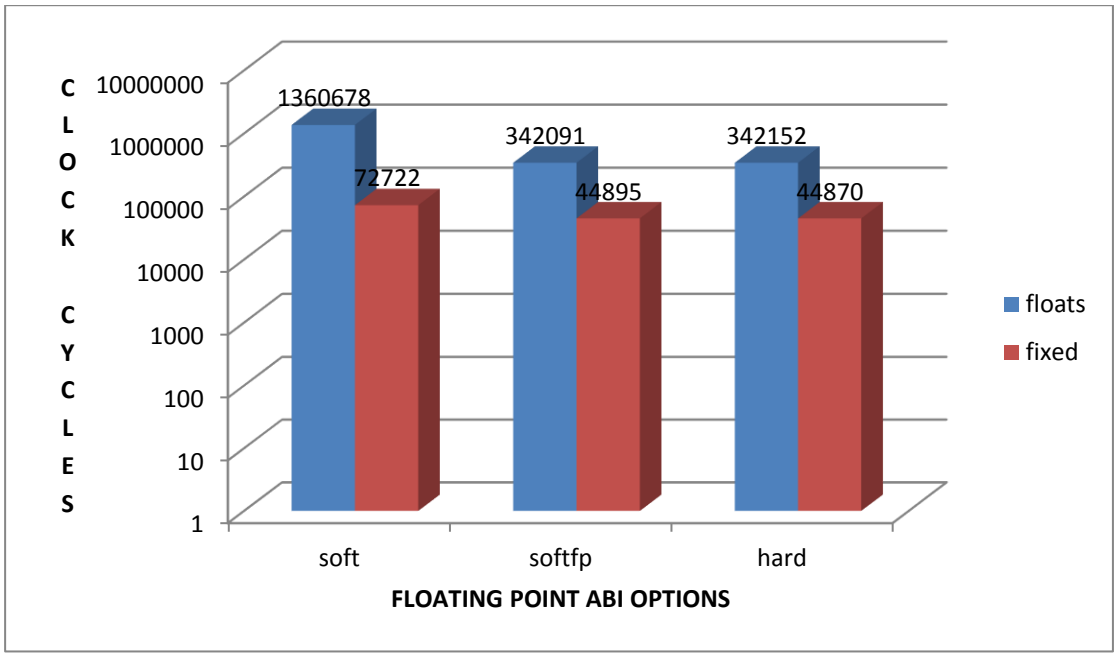
Εικόνα 29-soft vs softfp vs hard χρόνος μετατροπής σε 5χ5 εικόνα

Για 10χ10 εικόνα



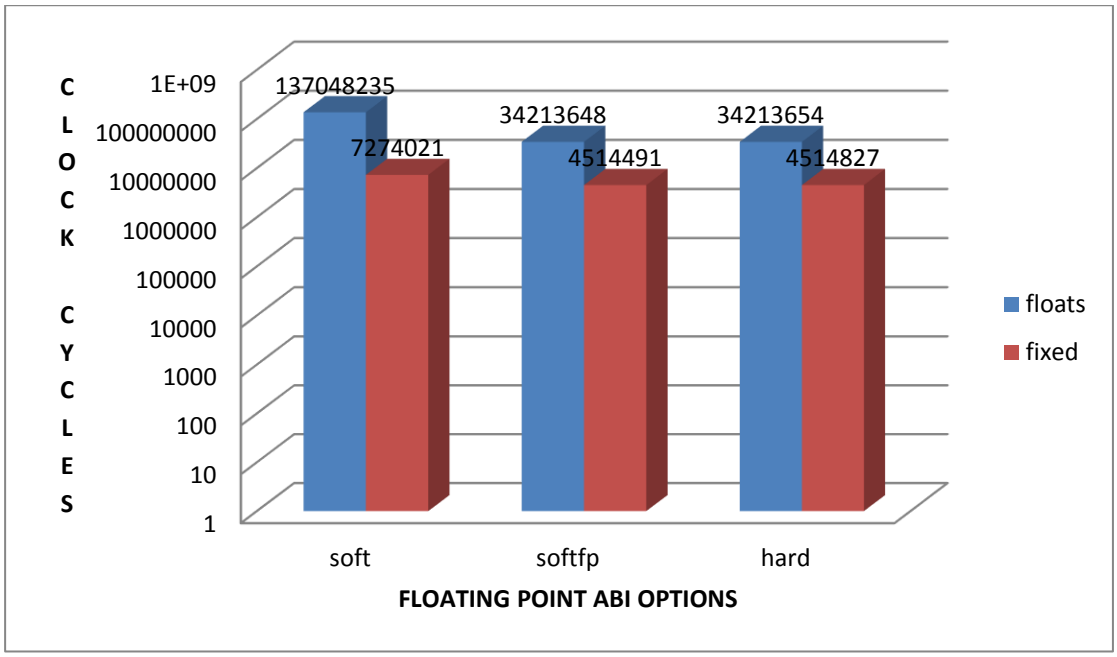
Εικόνα 30-soft vs softfp vs hard χρόνος μετατροπής σε 10χ10 εικόνα

Για 100χ100 εικόνα



Εικόνα 31-soft vs softfp vs hard χρόνος μετατροπής σε 100χ100 εικόνα

Για 1000χ1000 εικόνα



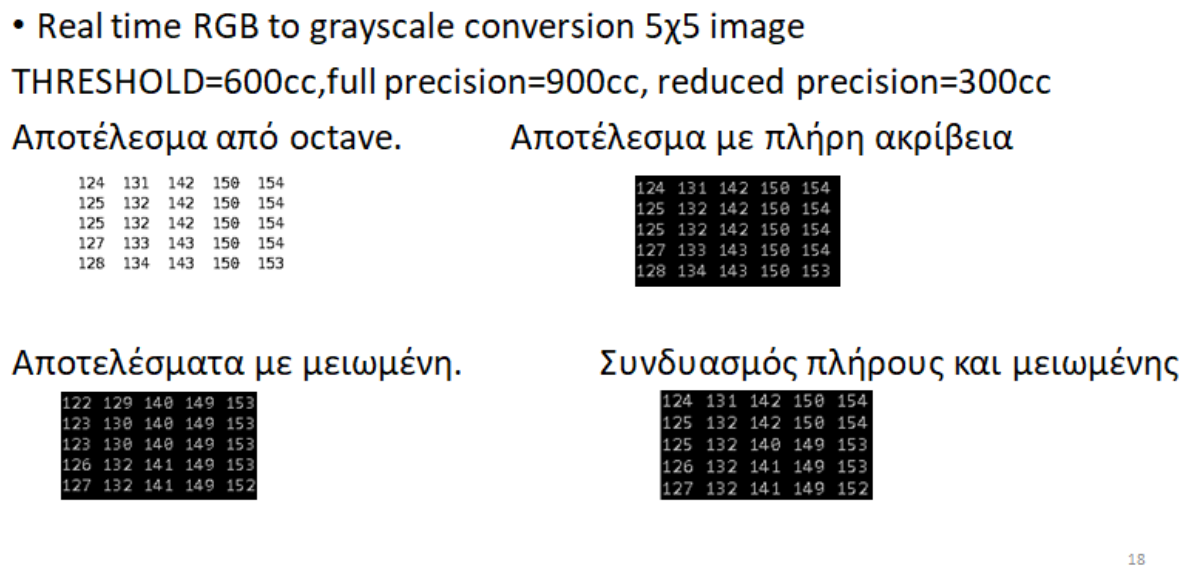
Εικόνα 32-soft vs softfp vs hard χρόνος μετατροπής σε 1000χ1000 εικόνα

Είναι εμφανές ότι με την τεχνική της μειωμένης ακρίβειας ο χρόνος της επεξεργασίας βελτιώνεται πάρα πολύ αλλά υπάρχει μια μικρή απόκλιση στις τιμές. Όμως στην συγκεκριμένη περίπτωση οι μικρές αυτές αλλαγές δεν είναι *δ*ιακριτές με το ανθρώπινο μάτι.

Επίσης ο χρόνος εκτέλεσης μεταβάλλεται ανάλογα με τα δεδομένα εισόδου.

9.3 Αποτελέσματα real time επεξεργασίας.

Στο συγκεκριμένο παράδειγμα γίνεται μετατροπή από έγχρωμη εικόνα σε ασπρόμαυρη με την χρήση της fixed point arithmetic για μειωμένη ακρίβεια άλλα και floating point arithmetic για πλήρη ακρίβεια σκοπός της εφαρμογής αυτής είναι να συνδυάσει τις 2 τεχνικές για να γίνει η μετατροπή της εικόνας μέσα σε αποδεκτά όρια χρόνο που ορίζονται ως κύκλοι στην μεταβλητή threshold.



Εικόνα 33 - αποτελέσματα από την μετατροπή εικόνας 5χ5

Στο παρά πάνω παράδειγμα γνωρίζουμε από τις μετρήσεις ότι χρειάζονται 900 κύκλοι ρολογιού περίπου για να μετατρέψουμε μια εικόνα από έγχρωμη σε ασπρόμαυρη και 300 κύκλοι για να το κάνουμε με μειωμένη ακρίβεια .Όμως εμείς αντισταθμίζουμε τον χρόνο με την ακρίβεια γνωρίζοντας ότι έχουμε στην διάθεση μας 600 κύκλους ρολογιού (μεταβλητή threshold) και έτσι γίνεται συνδυασμένη υπολογισμός με πλήρη ακρίβεια και μειωμένη για να προλάβουμε αυτό το deadline. Θα μπορούσε να χαρακτηριστεί σαν soft real time εκτέλεση εφαρμογής.

10. Συμπεράσματα και μελλοντικές επεκτάσεις

Είναι εμφανές ότι η χρήση της τεχνικής μειωμένης ακρίβειας fixed point arithmetic έχει πολλά οφέλη που αφορούν την απόδοση του συστήματος αλλά και τον χώρο αποθήκευσης όμως αυτό δεν είναι δεδομένο για όλες τις εφαρμογές που τρέχουν σε ένα σύστημα.

Το συμπέρασμα είναι ότι πριν χρησιμοποιηθούν οι τεχνικές υπολογισμού κατά προσέγγιση πρέπει ο προγραμματιστής να έχει ζυγίσει τα θετικά και τα αρνητικά τους και να εντοπίσει τα κομμάτια της εφαρμογής του που θα μπορούσε να έχει χρήση και όφελος κάτι τέτοιο.

10.1 Μελλοντικές Επεκτάσεις

Μπορούν να ελεγχτούν πάρα πολλές επεκτάσεις συμφώνα με την μεθοδολογία που αναπτύξαμε σε αυτή την πτυχιακή εργασία, για παράδειγμα μπορούν να δημιουργηθούν hardware blocks τα οποία θα αναλάβουν τις πράξεις μεταξύ fixed point αριθμών με αποτέλεσμα καλύτερη απόδοση ή να χρησιμοποιήσουμε μεγαλύτερο φάσμα δεδομένων εισόδου για καλύτερα αποτελέσματα ή ακόμα και να συγκριθούν άλλες τεχνικές για approximate computing με την τεχνική fixed point arithmetic.Επίσης θα μπορούσε να γίνει παράλληλη επεξεργασία χρησιμοποιώντας και τους 2 πυρήνες του zedboard για μεγαλύτερη απόδοση.

Βιβλιογραφία

[1] G. Kornaros, O. Tomoutzoglou, D. Mbakoyiannis, N. Karadimitriou και M. Coppola, «Towards holistic secure networking in connected vehicles through securing CAN-bus communication and firmware-over-the-air updating».*Journal of Systems Architecture (2020), vol. 109, pp. 101761.*

[2] G. Trouli και G. Kornaros, «Automotive Virtual In-sensor Analytics for Securing Vehicular Communication».*IEEE Design & Test, vol.37, issue 3, pp. 91-98, print ISSN: 2168-2356, online ISSN: 2168-2364, June 2020, DOI: 10.1109/MDAT.2020.2974914.*

[3] G. Kornaros, «A Soft Multi-core Architecture for Edge Detection and Data,» *J. Syst. Architect. JSA (2009), Volume 56, Issue 1,* Pages 48-62 January 2010.

[4] A. Nikologiannis, G. Kornaros, I. Papaefstathiou και C. Kachris, «An FPGA-based Queue Management System for High Speed Networking Devices,» *Elsevier Journal: Microprocessors and Microsystems, Volume 28, Issues 5-6,* Pages 223-236 , 2 August 2004.

[5] C. H. Lampert και O. Wirjadi, «Anisotropic Gaussian Filtering using Fixed Point Arithmetic,» Atlanta USA, 2006.

[6] Μ. Δασυγένης και Δ. Σουντρής, Ενσωματωμένα Συστήματα ο αθέατος ψηφιακός κόσμος, 2015.

[7] K. G. Shin και P. Ramanathan, «Real-Time Computing: A new discipline of,» *Proceedings of the IEEE,vol 82, no 1,* 1994.

[8] geeksforgeeks, «Types of Operating Systems,» https://www.geeksforgeeks.org/types-of-operating-systems/?ref=lbp, 2019.

[9] G. Kornaros και M. Coppola, «Enabling Efficient Job Dispatching in Accelerator-extended Heterogeneous Systems with Unified Address Space,» σε *Procs of 30th International Symposium on Computer Architecture and High Performance Computing, SBAC-PAD 2018*, 2018.

[10] G. Kornaros and M. Pratikakis, "VWQS: A dispatching mechanism of variable-size tasks in heterogeneous systems", International Conference on High Performance Computing & Simulation, HPCS 2016, Innsbruck, Austria, July 18-22, 2016, pp. 196-203, DOI: 10.1109/HPCSim.2016.7568335.

[11] O. Tomoutzoglou, D. Mbakoyiannis, G. Kornaros και M. Coppola, «Efficient Job Offloading in Heterogeneous Systems through Hardware-assisted Packet-based Dispatching and User-level Runtime Infrastructure,» *IEEE Transactions on Computer-Aided Design of Integrated Circuits*

and Systems, vol. 39, issue 5, pp. 1017-1030, print ISSN: 0278-0070, online ISSN: 1937-4151, May 2020.

[12] G. Kornaros, «RSMCC: Enabling Ring-based Software Managed Cache-Coherent Embedded SoCs,» *Proceedings of the 28th Euromicro International Conference on Parallel, Distributed and Network Based Processing (PDP 2020),* 11-13 March, pp 131-135, 2020.

[13] ALTERA, «Adding Hardware Accelerators to Reduce Power in Embedded,» 2009.

[14] G. Kornaros και A. Motakis, *On Scaling Speedup with Coarse-Grain Coprocessor Accelerators on Reconfigurable Platforms,* Digital Systems Design, Euromicro Symposium on, pp. 355-362, 2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools, 2010.

[15] J. Happe, «Performance Prediction for Embedded Systems,» https://sdqweb.ipd.kit.edu/publications/pdfs/happe2005b.pdf, Germany, 2005.

[16] J. Zhou, J. Yan, T. Wei, M. Chen και X. S. Hu, «Energy-adaptive scheduling of imprecise computation tasks for QoS optimization in real-Time MPSoC systems».*Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017,.*

[17] W. Baek και T. M. Chilimbi, «Green: a framework for supporting energy-conscious programming using controlled approximation,» n Proceedings of the 31st ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '10). Association for Computing Machinery, New York, NY, USA, 198–209. DOI:https://doi.org/10.1145/1806596.1806620.

[18] G. R. Wiedenhoft και A. A. Fröhlich, Using Imprecise Computation Techniques for Power Management in Real-Time Embedded Systems,in book "Distributed Embedded Systems: Design, Middleware and Resources",, Boston: Springer US, 2008.

[19] J. W. S. Liu, K.-J. Lin, W.-K. Shih, A. C.-s. Yu, J.-Y. Chung και W. Zhao, Algorithms for Scheduling Imprecise Computations, University of Illinois at Urbana-Champaign, 1991.

[20] https://www.tutorialspoint.com/dip/image_processing_introduction.htm, «Digital Image Processing Introduction,» Tutorials Point.

[21] Zynq-7000 SoC Technical Reference Manual, https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf.

[22] H. Qin, Boost Software Performance on Zynq-7000, Xilinx, June 12, 2014

[23] A. Motakis, G. Kornaros, M. Coppola, “Dynamic Resource Management in Modern Multicore SoCs by Exposing NoC Services”, IEEE, 6th International Workshop on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC’2011), Montpellier, France, Jun. 20-21, 2011

Παράρτημα Α

Ελεγχόμενη εκτέλεση εφαρμογής πραγματικού χρόνου σε ενσωματωμένο σύστημα

Κρητικάκης Εμμανουήλ: Α.Μ. 4416
Επιβλέπων εκπαιδευτικός : Κορνάρος Γεώργιος

Περίγραμμα

- Εισαγωγή
- Περιγραφή υλικού του συστήματος
- Περιγραφή λογισμικού
- Αποτελέσματα και μετρήσεις
- Συμπεράσματα και μελλοντικές επεκτάσεις

Εισαγωγή

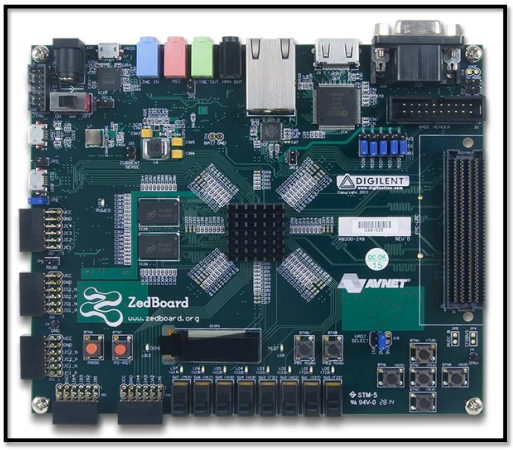
Τι ακριβώς υλοποιήσαμε;

- Τεχνική Fixed point arithmetic για πράξεις με μειωμένη ακρίβεια.
- Σύγκριση μεταξύ fixed point arithmetic και floating point arithmetic στις βασικές πράξεις μεταξύ πινάκων.
- Μετατροπή έγχρωμης εικόνας σε ασπρόμαυρη με πλήρη και μειωμένη ακρίβεια.
- Σύγκριση των αποτελεσμάτων με και χωρίς την χρήση FPU.
- Real time επεξεργασία εικόνας ακλουθώντας ένα κατώφλι χρόνου.

2

Περιγραφή υλικού του συστήματος

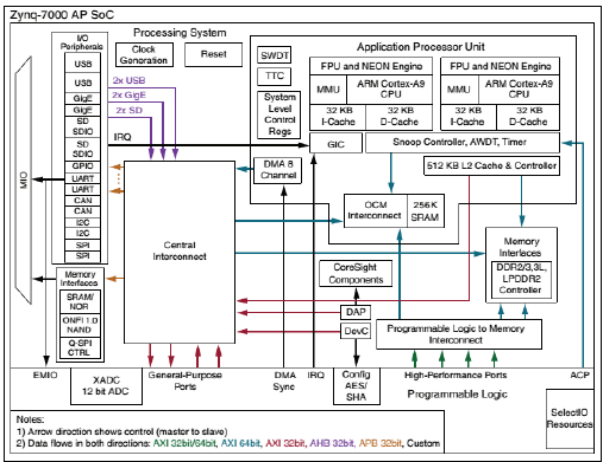
- Ποια θα είναι η πλατφόρμα του συστήματος;
- Γιατί επιλέξαμε το zedboard;



3

Περιγραφή υλικού του συστήματος

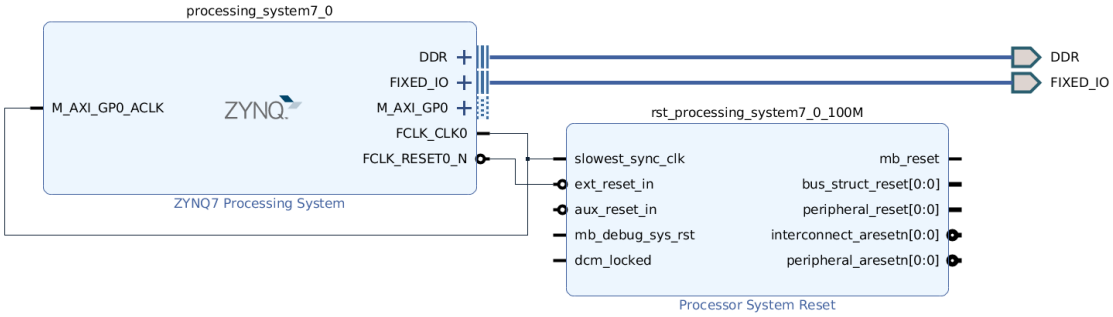
- Ποια είναι η αρχιτεκτονική του συστήματος;



4

Περιγραφή υλικού του συστήματος

- Περιγραφή συστήματος στο vivado



5

Περιγραφή λογισμικού του συστήματος

- Μετατροπή από κινητής υποδιαστολής σε fixed

```
inline fixed_point_t float_to_fixed(double input)
{
   return (fixed_point_t)(round(input * (1 << FIXED_POINT_FRACTIONAL_BITS)));
}
```

- Μετατροπή από fixed σε κινητής υποδιαστολής

```
inline double fixed_to_float(fixed_point_t input)
{
   return ((double)input / (double)(1 << FIXED_POINT_FRACTIONAL_BITS));
}
```

6

Περιγραφή λογισμικού του συστήματος

- Πρόσθεση

```
fixed_point_t fixed_add(fixed_point_t x, fixed_point_t y)
{
    return x+y;
}
```

- Αφαίρεση

```
fixed_point_t fixed_sub(fixed_point_t x, fixed_point_t y)
{
   return x-y;
}
```

7

Περιγραφή λογισμικού του συστήματος

- Πολλαπλασιασμός

```
fixed_point_t fixed_mul(fixed_point_t x, fixed_point_t y)
{
   return ((int32_t)x * (int32_t)y) / (1 << FIXED_POINT_FRACTIONAL_BITS);
}
```

- Διαίρεση

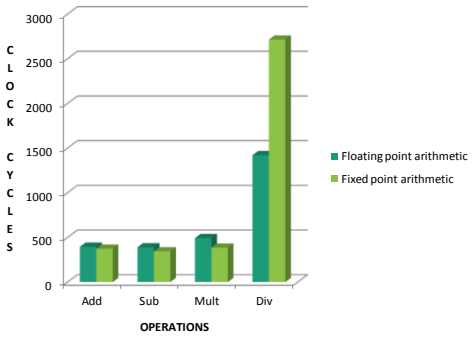
```
fixed_point_t fixed_mul(fixed_point_t x, fixed_point_t y)
{
   return ((int32_t)x * (int32_t)y) / (1 << FIXED_POINT_FRACTIONAL_BITS);
}
```

8

Αποτελέσματα και Μετρήσεις

- Πράξεις μεταξύ πινάκων 1χ100

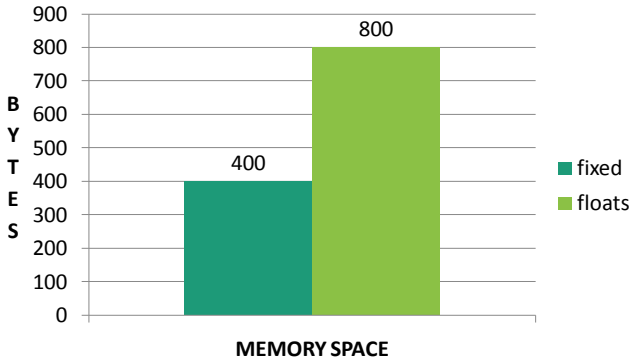
	Floating point arithmetic	Fixed point arithmetic
Add	395	374
Sub	389	343
Mult	492	386
div	1420	2714



9

Αποτελέσματα και Μετρήσεις

- Χώρος στην DDR για δυο πίνακες 1χ100

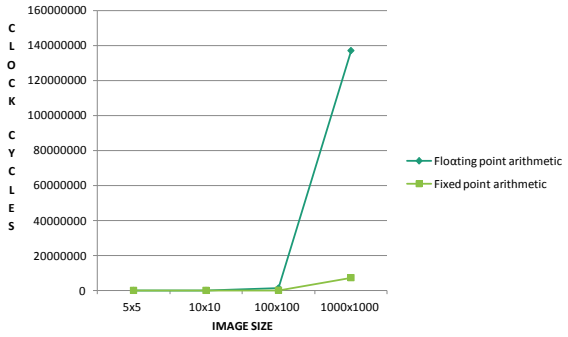


10

Αποτελέσματα και συμπεράσματα

- Rgb image to grayscale με επιλογή soft (Χωρίς FPU)

	Floating point arithmetic	*Fixed point* arithmetic
5x5 image	*3777*	*245*
10x10 image	*14327*	*821*
100x100 image	*1360678*	*72722*
1000x1000 image	*137048235*	*7274021*

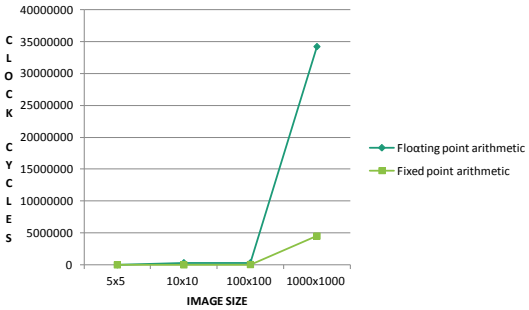


11

Αποτελέσματα και συμπεράσματα

- Rgb image to grayscale με επιλογή softfp(Υποθέτει ότι υπάρχει FPU)

	Floating point arithmetic	*Fixed point* arithmetic
5x5 image	*919*	*237*
10x10 image	*3484*	*710*
100x100 image	*342152*	*44870*
1000x1000 image	*34213654*	*4514827*

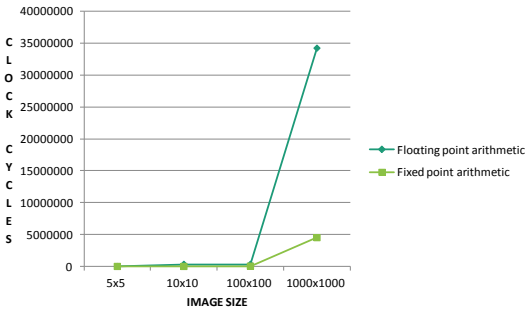


13

Αποτελέσματα και συμπεράσματα

- Rgb image to grayscale με επιλογή hardΥποθέτει ότι υπάρχει FPU)

	Floating point arithmetic	*Fixed point* arithmetic
5x5 image	*919*	*237*
10x10 image	*3484*	*710*
100x100 image	*342152*	*44870*
1000x1000 image	*34213654*	*4514827*

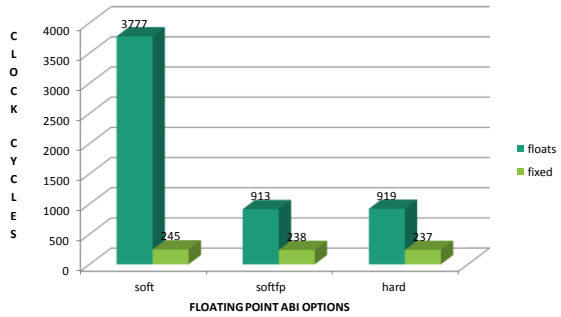


13

Αποτελέσματα και συμπεράσματα

- Rgb image to grayscale Soft vs Softfp vs Hard

5X5 Image

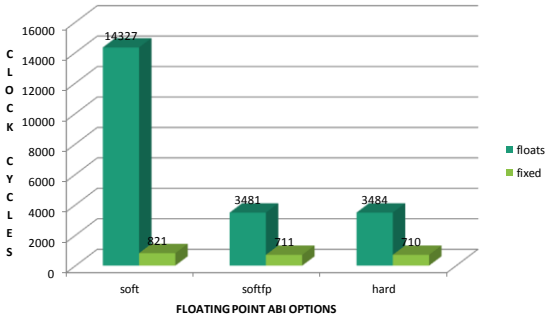


14

Αποτελέσματα και συμπεράσματα

- Rgb image to grayscale Soft vs Softfp vs Hard

10X10 Image

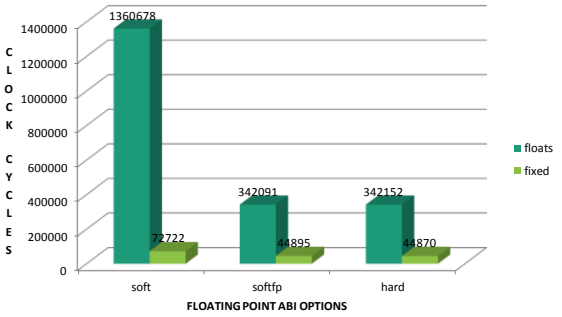


15

Αποτελέσματα και συμπεράσματα

- Rgb image to grayscale Soft vs Softfp vs Hard

100X100 Image

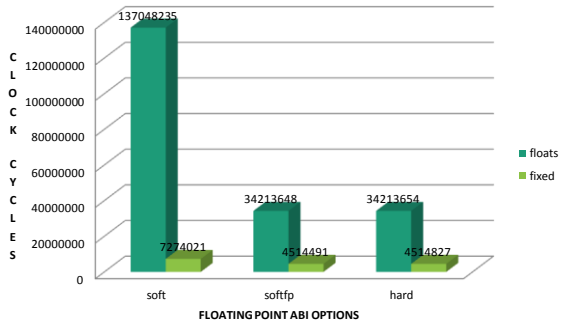


16

Αποτελέσματα και συμπεράσματα

- Rgb image to grayscale Soft vs Softfp vs Hard

1000X1000 Image



17

Αποτελέσματα και Μετρήσεις

- Real time RGB to grayscale conversion 5χ5 image

THRESHOLD=600cc,full precision=900cc, reduced precision=300cc

Αποτέλεσμα από octave.

```
124  131  142  150  154
125  132  142  150  154
125  132  142  150  154
127  133  143  150  154
128  134  143  150  153
```

Αποτέλεσμα με πλήρη ακρίβεια

```
124 131 142 150 154
125 132 142 150 154
125 132 142 150 154
127 133 143 150 154
128 134 143 150 153
```

Αποτελέσματα με μειωμένη.

```
122 129 140 149 153
123 130 140 149 153
123 130 140 149 153
126 132 141 149 153
127 132 141 149 152
```

Συνδυασμός πλήρους και μειωμένης

```
124 131 142 150 154
125 132 142 150 154
125 132 140 149 153
126 132 141 149 153
127 132 141 149 152
```

18

Συμπεράσματα

- Είναι εμφανές ότι η χρήση της τεχνικής μειωμένης ακρίβειας fixed point arithmetic έχει πολλά οφέλη που αφορούν την απόδοση του συστήματος αλλά και τον χώρο αποθήκευσης όμως αυτό δεν σημάνει ότι μπορεί να χρησιμοποιηθεί σε άλλες τις εφαρμογές.
- Το συμπέρασμα είναι ότι πριν χρησιμοποιηθούν οι τεχνικές υπολογισμού κατά προσέγγιση πρέπει ο προγραμματιστής να έχει ζυγίσει τα θετικά και τα αρνητικά τους και να εντοπίσει τα κομμάτια της εφαρμογής του που θα μπορούσε να έχει χρήση και όφελος κάτι τέτοιο.

19

Μελλοντικές Επεκτάσεις

- Δημιουργία hardware blocks για να αναλάβουν τις πράξεις μεταξύ fixed point.
- Παράλληλη επεξεργασία των δεδομένων και από τους 2 πυρήνες.
- Σύγκριση τεχνικών approximate computing με fixed point arithmetic
- Χρήση μεγαλύτερου φάσματος δεδομένων εισόδου εξαγωγή περισσότερων αποτελεσμάτων

20

Ευχαριστώ!!

21