ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ
ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΠΟΛΥΜΕΣΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΕΚΤΙΜΗΣΕΙΣ ΕΠΙΔΟΣΕΩΝ ΕΝΣΩΜΑΤΩΜΕΝΩΝ ΕΠΕΞΕΡΓΑΣΤΩΝ ΣΕ ΕΦΑΡΜΟΓΕΣ ΕΙΚΟΝΑΣ

Σπουδαστές: Συμεού Χαράλαμπος 1909
Νεοφύτου Νεόφυτος 1484

Υπεύθυνος Καθηγητής: Κορνάρος Γιώργος

Ηράκλειο Νοέμβριος 2009

ΕΚΤΙΜΗΣΕΙΣ ΕΠΙΔΟΣΕΩΝ ΕΝΣΩΜΑΤΩΜΕΝΩΝ ΕΠΕΞΕΡΓΑΣΤΩΝ ΣΕ ΕΦΑΡΜΟΓΕΣ ΕΙΚΟΝΑΣ

Πλατφόρμα Ανάπτυξης ML403 της XILINX

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1: ΓΝΩΡΙΜΙΑ ΜΕ ΤΟ ΑΝΤΙΚΕΙΜΕΝΟ ΤΩΝ FPGA

1.1 Ιστορική αναδρομή

Οι ιδρυτές της Xilinx Ross Freeman και Berrand Vanderchmint επινόησαν το πρώτο βιώσιμο και ολοκληρωμένο πακέτο αναδιατασώμενων πυλών. Το μοντέλο αυτό ήταν το XC2064 που είχε αναδιατασώμενες διασυνδεδεμένες πύλες και αποτέλεσε την απαρχή μιας νέας τεχνολογίας και αγοράς. Το μοντέλο XC2064 υποστήριζε την απλή λογική σχεδίαση με 64 αναδιατασώμενα block(CLBs) με δύο lookup πινάκων εισόδου(LUT).Ο Ross μπήκε στο Hall of fame των εθνικών ερευνητών για την εφεύρεση του.

Η δεκαετία του 1990 αποτέλεσε την απογείωση της τεχνολογικής βιομηχανίας της FPGA τόσο στην πολυπλοκότητα των υποστηριζόμενων λειτουργιών, όσο και στον όγκο της παραγωγής της. Η FPGA την δεκαετία αυτή βρήκε εφαρμογές στον τομέα των επικοινωνιών δικτύων που αργότερα χρησιμοποιήθηκε αφενός σε επίπεδο καταναλωτών και αφετέρου σε επίπεδο τεχνολογικής βιομηχανίας αφού έγινε χρήσιμο εργαλείο για τις μεγάλες βιομηχανίες.

Η σύγχρονη εξέλιξη της τάσης αυτής είναι η τάση να παίρνουν τα κακής λογικής και αρχιτεκτονικής σχεδίασης και να τα προχωρούν ένα βήμα παραπέρα συνδυάζοντας τη λογική που συνδέει τα block των παραδοσιακών FPGA με την ενσωμάτωση μικροεπεξεργαστών και σχετικών περιφερειακών ώστε να αποτελούν πλήρες σύστημα για τα προγραμματιζόμενα chip. Παραδείγματα τέτοιων υβριδικών τεχνολογιών μπορούν να βρεθούν στο Xilinx Virtex-II pro,Virtex-4 και συσκευών Virtex-5 που περιλαμβάνουν έναν ή περισσότερους επεξεργαστές PowerPC και MicroBlaze ενσωματωμένους στο πλαίσιο της λογικής σχεδίασης της FPGA καθώς και FPSLIC της Atmel η οποία χρησιμοποιεί τον AVR επεξεργαστή σε συνδυασμό με Atmel αναδιατασώμενη λογική αρχιτεκτονική.

1.2 Αρχιτεκτονική με FPGA

Η χαρακτηριστική βασική αρχιτεκτονική αποτελείται από μια σειρά αναδιατασώμενων blocks (CLBs) και καναλιών δρομολόγησης. Γενικά, όλα τα κανάλια δρομολόγησης έχουν το ίδιο πλάτος (αριθμός καλωδίων). Ένα κύκλωμα εφαρμογής πρέπει να χαρτογραφηθεί σε ένα FPGA με τους επαρκείς πόρους. Το χαρακτηριστικό block λογικής FPGA αποτελείται από έναν πίνακα συμβουλεύσεις εισαγωγής (LUT), και flip-flop, όπως παρουσιάζεται πιο κάτω.

Υπάρχει μόνο μια παραγωγή, η οποία μπορεί να είναι είτε καταχωρημένη είτε μη καταγεγραμμένη παραγωγή LUT. Το block λογικής έχει τέσσερις εισαγωγές για το LUT και μια εισαγωγή timer. Δεδομένου ότι τα σήματα ρολογιών (fanout) καθοδηγούνται κανονικά μέσω των ειδικής χρήσης αφιερωμένων δικτύων δρομολόγησης σε FPGA, αυτά και άλλα σήματα ρυθμίζονται χωριστά. Για αυτήν την αρχιτεκτονική παράδειγμα, οι θέσεις των συνδέσμων των block λογικής FPGA παρουσιάζονται πιο κάτω.

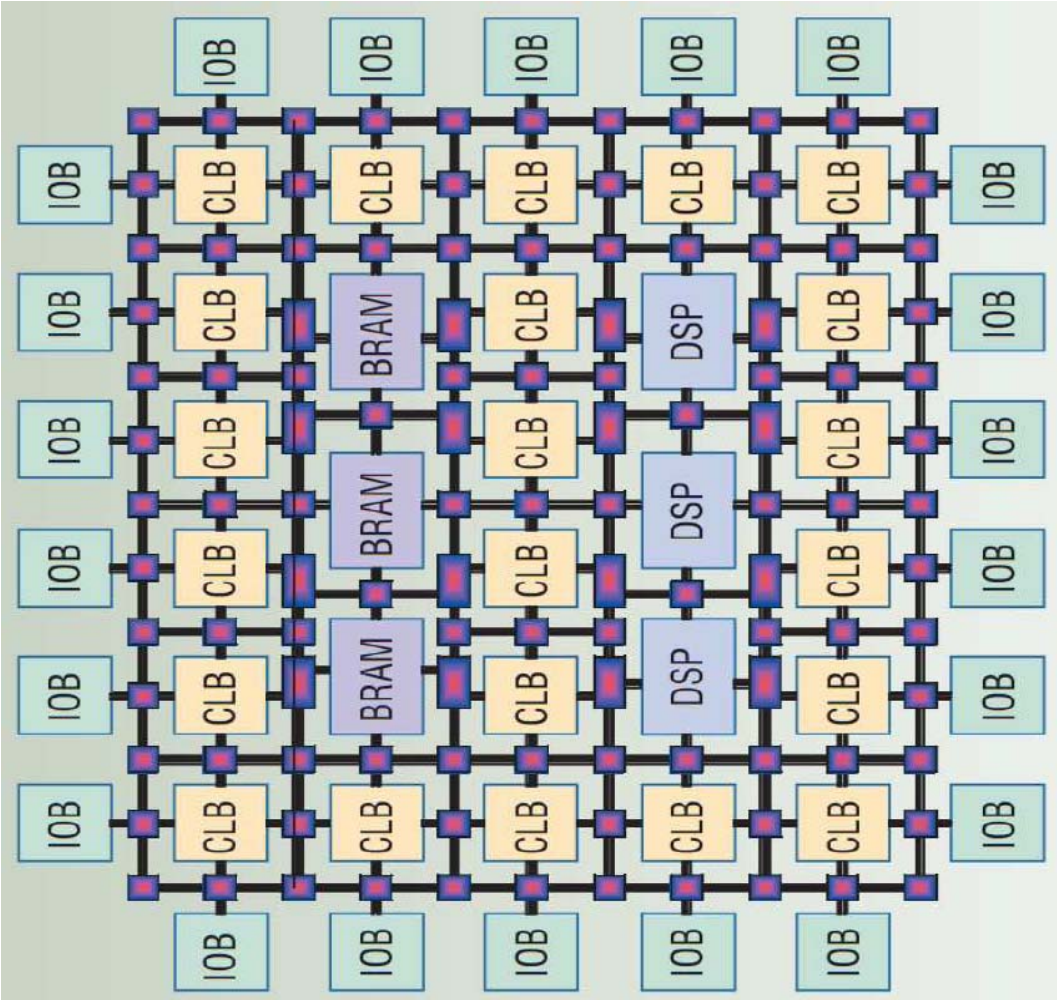
Κάθε εισαγωγή είναι προσιτή από μια πλευρά του block λογικής, ενώ ο σύνδεσμος παραγωγής μπορεί να συνδέσει με τη δρομολόγηση των καλωδίων στο κανάλι κάτω από το block λογικής. Κάθε σύνδεσμος παραγωγής block λογικής μπορεί να συνδεθεί με οποιαδήποτε από τα τμήματα καλωδίωσης στα κανάλια δίπλα σε αυτό. Ομοίως, ένα I/O block μπορεί να συνδεθεί με οποιοδήποτε από τα τμήματα καλωδίωσης στο κανάλι δίπλα σε αυτό. Παραδείγματος χάριν, ένα I/O block στην κορυφή του chip μπορεί να συνδέσει με οποιαδήποτε από τα καλώδια W (όπου το W είναι το πλάτος καναλιών) στο οριζόντιο κανάλι αμέσως κάτω από αυτό.

Γενικά, η δρομολόγηση FPGA είναι σε κάθε τμήμα καλωδίωσης που εκτείνεται μόνο σε ένα block λογικής προτού να τερματίσει σε ένα switch box. Με το να ανοίξουν μερικοί από τους αναδιατασώμενους διακόπτες μέσα σε ένα switch box, οι μακρύτερες πορείες μπορούν να κατασκευαστούν. Για την υψηλότερη ταχύτητα διασυνδέονται, μερικές αρχιτεκτονικές FPGA χρησιμοποιούν τις πιο μακροχρόνιες γραμμές δρομολόγησης που εκτείνονται στα πολλαπλά block λογικής. Όποτε σε ένα κάθετο και οριζόντιο κανάλι υπάρχει ένα switch box. Σε αυτήν την αρχιτεκτονική, όταν εισαχθεί ένα καλώδιο σε ένα switch box, υπάρχουν τρεις αναδιατασώμενους διακόπτες που επιτρέπουν σε αυτό για να συνδέσουν με τρία άλλα καλώδια στα παρακείμενα τμήματα καναλιών. Το σχέδιο, ή η τοπολογία, των διακοπτών που χρησιμοποιούνται σε αυτήν την αρχιτεκτονική είναι η επίπεδη περιοχή βασισμένη στην τοπολογία των switch box. Σε αυτήν την τοπολογία switch box, ένα καλώδιο

στη διαδρομή με αριθμό 1, συνδέεται μόνο με τα καλώδια στον αριθμό διαδρομής 1 στα παρακείμενα τμήματα καναλιών, τα καλώδια στη διαδρομή αριθμός 2 συνδέουν μόνο με άλλα καλώδια στη διαδρομή με αριθμό 2 και τα λοιπά. Ο αριθμός επεξηγείτε πιο κάτω τις συνδέσεις σε ένα switch box.

Οι σύγχρονες οικογένειες FPGA επεκτείνονται επάνω στις ανώτερες ικανότητες να συμπεριλαμβάνουν τη λειτουργία πιο υψηλού επιπέδου που καθορίζεται στο πυρίτιο. Ενσωμάτωση αυτών των κοινών λειτουργιών στο πυρίτιο μειώνει την περιοχή που απαιτείται και δίνει αυξανόμενη λειτουργική ταχύτητα έναντι στην οικοδόμηση τους από τους πρωτόγονους της. Τα παραδείγματα αυτά περιλαμβάνουν τους πολλαπλασιαστές, τα γενικά block DSP, τους ενσωματωμένους επεξεργαστές, την υψηλή λογική ταχύτητας I/O και τις ενσωματωμένες μνήμες.

Τα FPGA χρησιμοποιούνται ευρέως για την επικύρωση συστημάτων συμπεριλαμβανομένης της επικύρωσης προ-πυριτίου, την επικύρωση μετά-πυριτίου, και firmware την ανάπτυξη. Αυτό επιτρέπει στις επιχειρήσεις chip να επικυρώσει το σχέδιό τους προτού να παραχθεί το chip στο εργοστάσιο, που μειώνει το χρόνο στην αγορά.



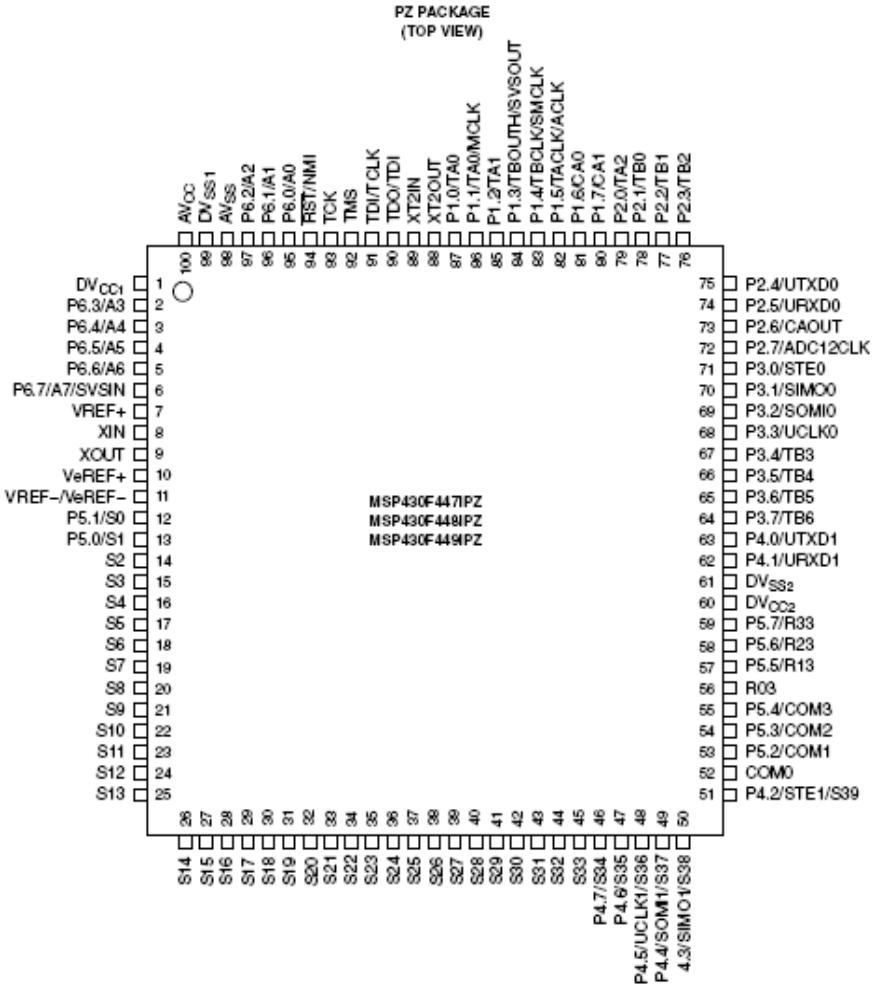
Τυπική αρχιτεκτονική FPGA

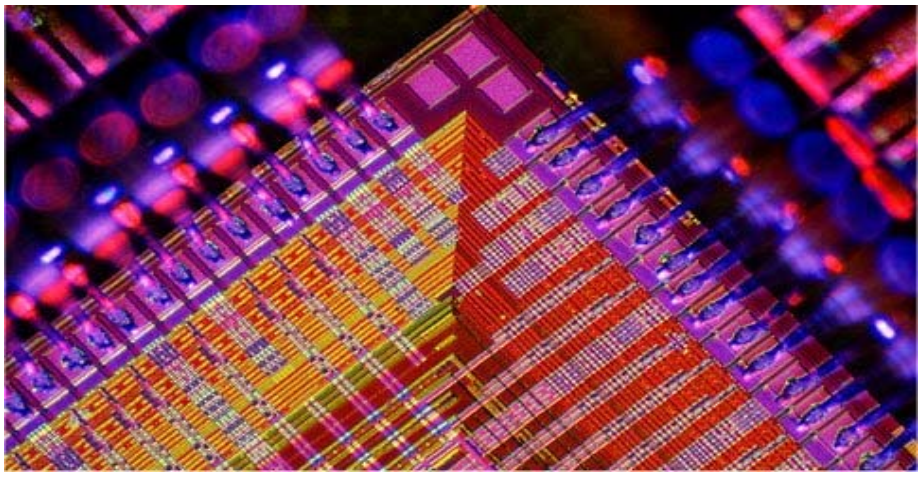
1.3 FPGA στον πυρήνα κεντρικής μονάδας επεξεργασίας

Μερικές εφαρμογές εφαρμοσμένης μηχανικής έχουν χρησιμοποιήσει μια ενιαία συσκευή FPGA για να αντικαταστήσουν τη λειτουργία ενός απλού ενσωματωμένου-μικροελεκτή. Πιο πρόσφατα, ένας πλήρης τριανταδυάμπιτος πυρήνας ΚΜΕ (κεντρικής μονάδας επεξεργασίας) μπορεί να εφαρμοστεί μέσω της αναδιατασώμενης λογικής σε μια μεγάλης χωρητικότητας FPGA. Τέτοιοι πυρήνες ΚΜΕ είναι γνωστοί ως "μαλακοί πυρήνες ΚΜΕ," παραδείγματα των οποίων είναι MicroBlaze, Nios II, και LatticeMico32 από Xilinx, Altera και το δικτυωτό πλέγμα αντίστοιχα.

Πέρα από αυτό, μερικές συσκευές FPGA περιέχουν τον αφιερωμένο πυρήνα υλικού ΚΜΕ. Τα επιλεγμένα μέρη Virtex από Xilinx, περιέχουν ένα ή περισσότερους ενσωματωμένους ΚΜΕ πυρήνες της IBM όπως PowerPC 405, εκτός από την αναδιατασώμενη λογική του FPGA. Για μια δεδομένη αρχιτεκτονική ΚΜΕ, ένας σκληρός (ενσωματωμένος) πυρήνας CPU θα ξεπεράσει έναν μαλακό-πυρήνα ΚΜΕ (δηλ., μια εφαρμογή αναδιατασώμενης-λογικής της ΚΜΕ). Η ενσωματωμένη ΚΜΕ περιέχει ακριβώς τη λογική και μόνο τις δομές λογικής που απαιτούνται για τη λειτουργία της ΚΜΕ, και η ενσωματωμένη λογική της ΚΜΕ είναι task specific βελτιστοποιημένη, ενώ ένας μαλακός πυρήνας ΚΜΕ πρέπει να ζήσει μέσα στο γενικής χρήσης πλέγμα λογικής της FPGA. Ενσωματωμένη ΚΜΕ μπορεί να είναι επίσης ευκολότερο να ενσωματωθεί σε μια FPGA εφαρμογή επειδή η καθορισμένη φύση της ενσωματωμένης ΚΜΕ κατέχει τα προβλέψιμα χαρακτηριστικά συγχρονισμού, και η πολυπλοκότητα μιας ισοδύναμης αναδιατασώμενης λογικής.

Η χρήση ενσωματωμένου CPU μπορεί να περιορίσει την επιλογή των διαθέσιμων συσκευών και των εργαλείων σχεδίου και απαιτεί υπό αυτήν τη μορφή της προσεκτικής αιτιολόγησης πέρα από τη μαλακή προσέγγιση πυρήνων ΚΜΕ.


PZ PACKAGE
(TOP VIEW)
MSP430F447IPZ
MSP430F448IPZ
MSP430F449IPZ
DVCC1
P6.3/A3
P6.4/A4
P6.5/A5
P6.6/A6
P6.7/A7/SVSIN
VREF+
XIN
XOUT
VeREF+
VREF−/VeREF−
P5.1/S0
P5.0/S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
S12
S13
S14
S15
S16
S17
S18
S19
S20
S21
S22
S23
S24
S25
S26
S27
S28
S29
S30
S31
S32
S33
P4.7/S34
P4.6/S35
P4.5/UCLK1/S36
P4.4/SOMI1/S37
4.3/SIMO1/S38
P2.4/UTXD0
P2.5/URXD0
P2.6/CAOUT
P2.7/ADC12CLK
P3.0/STE0
P3.1/SIMO0
P3.2/SOMI0
P3.3/UCLK0
P3.4/TB3
P3.5/TB4
P3.6/TB5
P3.7/TB6
P4.0/UTXD1
P4.1/URXD1
DVSS2
DVCC2
P5.7/R33
P5.6/R23
P5.5/R13
R03
P5.4/COM3
P5.3/COM2
P5.2/COM1
COM0
P4.2/STE1/S39



1.4 Πλατφόρμα Ανάπτυξης ML403

Με την νέα γενιά FPGA ML403 αναπτύχθηκαν πολλές στρατηγικές για την εκκίνηση FPGA διαμόρφωσης, σε ένα σχεδιασμό που επιτρέπει να επιλέξουμε αυτό που θέλουμε να χρησιμοποιήσουμε**.** Μπορούμε να φορτώσουμε το FPGA απευθείας από έναν κεντρικό υπολογιστή σε υπολογιστή JTAG interface. Η Virtex-4 μπορεί να εκκινήσει άμεσα flash chip. Η ML403 έχει 32MB πλατφόρμα flash, που μπορούν να προγραμματιστούν είτε με system.ace, είτε απευθείας ή μέσω JTAG.. Ορισμένα σχέδια μπορεί να χρησιμοποιήσουν ένα εξωτερικό μικροελεκτή για να φορτώσει το bit stream από μια εξωτερική συσκευή αποθήκευσης. Η προσέγγιση αυτή θα πρέπει να χρησιμοποιείται κατά κανόνα αν χρησιμοποιείτε η FPGA μόνο για την εφαρμογή περιφερειακών και ιδίως, οι επιπτώσεις είναι συνήθως ότι έχετε ήδη κάποιο άλλο μικροεπεξεργαστή με το σύστημα. Για παράδειγμα, μία προσθήκη στη PCI κάρτα θα μπορούσε να χρησιμοποιήσει αυτή την προσέγγιση για την προετοιμασία της FPGA. Η Xilinx προσφέρει μια λύση σε πακέτα που ονομάζεται system.ace, διαθέσιμα σε δύο τύπους, το ένα από αυτά χρησιμοποιεί liner flash, και το άλλο χρησιμοποιεί μια κάρτα CompactFlash. Η ML403 έχει επίσης CPLD (Complex Programmable Logic Device) onboard, το οποίο μπορεί να ρυθμίσει τις FPGA με αποστολή του bit stream και αποθήκευση στις μνήμες της FPGA.

Παρατηρούμε δύο σημαντικά σημεία. Πρώτον, στο σχεδιασμό πρέπει να ληφθεί υπόψη η ανάγκη να ρυθμιστεί η FPGA με τις συσκευές εισόδου / εξόδου και δεύτερο και σημαντικότερο πρέπει να δοθεί ιδιαίτερη προσοχή στην σχεδίαση των κυκλωμάτων αφού στα συγκεκριμένα συστήματα παίζει σημαντικό ρόλο η ακρίβεια στην σχεδίαση αν θέλουμε να πετύχουμε αποδοτικά και αξιόπιστα σχέδια.

Στο πακέτο η xilinx σχεδίασε δυο επεξεργαστές 32bit έτοιμους να τους χρησιμοποιήσουμε στις δικές μας εφαρμογές. Οι επεξεργαστές αυτοί είναι βελτιστοποιημένοι και αξιόπιστα σχεδιαστικά εργαλεία που μας προσφέρονται με το πακέτο. Οι δυο αυτοί επεξεργαστές είναι ο MicroBlaze και ο PowerPC. Οι διάφορες στους δυο αυτούς επεξεργαστές είναι στο όγκο των συστημάτων που εμπεριέχονται στο κάθε ένα και στην αποδοτικότητα τους παρόλο που έχουν περίπου την ίδια συμπεριφορά στις εφαρμογές. Ο PowerPC είναι ενσωματωμένος στον πυρήνα ενώ ο MicroBlaze είναι σχεδιασμένος πάνω στο πυρήνα.

Στο πακέτο της Xilinx έρχεται το software να συμπληρώσει την σχεδίαση των embedded system με το να προσφέρει την δυνατότητα της δόκιμης διάφορων εφαρμογών με τους C και C++ compiler. Αυτό προσφέρει στην σχεδίαση άμεση

διεπαφή με το σχεδιαστικό μοντέλο αφού μπορούμε να καθορίσουμε την συμπεριφορά των περιφερειακών που υποστηρίζει καθώς και την δημιουργία κώδικα με εφαρμογές πολυμέσων.

Ένα επιπλέον πλεονέκτημα της σχεδίασης με FPGA είναι η δυνατότητα του συστήματος αυτού να προσφέρει χωρητικότητα στην τοποθέτηση ακόμα και τριών επεξεργαστών όπως είναι ο MicroBlaze με την εφαρμογή να απογειώνεται σε δυνατότητες. Η σχεδιαστική καινοτομία αυτή προσφέρει μεγάλη υποστήριξη πολύπλοκων συστημάτων και ελέγχων με την χρήση μιας και μόνο σχεδίασης embedded system.

Η Ml403 προσφέρει ένα έτοιμο λειτουργικό σύστημα Linux με πολύ χαμηλές απαιτήσεις σε πόρους συστήματος που μπορεί να χρησιμοποιηθεί σαν ένα ολοκληρωμένο υπολογιστικό σύστημα, αφού προσφέρει και την δυνατότητα υποστήριξης όλων των περιφερειακών που χρειάζεται ένα σύγχρονο υπολογιστικό σύστημα(network controller, vga output,usb, memory hard disc drive).

ΚΕΦΑΛΑΙΟ 2 : ΕΝΣΩΜΑΤΩΜΕΝΑ ΣΥΣΤΗΜΑΤΑ(EMBEDDED SYSTEM)

2 .1 Ενσωματωμένα Συστήματα

Τα Ενσωματωμένα Συστήματα (embedded system) είναι υπολογιστικά συστήματα ειδικού σκοπού προσανατολισμένα στο να εξυπηρετήσουν τις ανάγκες των συσκευών της σύγχρονης ζωής όπως κινητά τηλέφωνα, palmtop, ελεγκτές αεροσκαφών, αυτοκινήτων κλπ. Για το λόγο αυτό συνήθως χαρακτηρίζονται από το μικρό τους μέγεθος και τα ιδιαίτερα χαρακτηριστικά τους ως προς την κατανάλωση ισχύος, την απόδοση σε συγκεκριμένες εφαρμογές και το χαμηλό τους κόστος. Ένα Ενσωματωμένο Σύστημα (Ε.Σ.) (Embedded System) αποτελεί υπολογιστική μονάδα με αρχιτεκτονική και αρχές λειτουργίας παρόμοιες με αυτές των συμβατικών υπολογιστών, η οποία ωστόσο προσαρμόζεται στις ανάγκες και απαιτήσεις της εκάστοτε εφαρμογής. Έτσι, και στην περίπτωση των Ε.Σ., βασικό δομικό στοιχείο αποτελεί ένας μικροεπεξεργαστής, ο οποίος βρίσκεται συνδεδεμένος μέσω μιας ιεραρχίας διαύλων με στοιχεία προσωρινής και μόνιμης αποθήκευσης (μνήμες RAM, EPPROM, Flash, non-Volatile). Παράλληλα, στα Ε.Σ. απαντώνται και στοιχεία εξειδικευμένου υλικού τα οποία επικοινωνούν με τα βασικά δομικά στοιχεία και καλούνται να επιτελέσουν συγκεκριμένες εργασίες ανάλογα με τις απαιτήσεις της εκάστοτε εφαρμογής σε απόδοση, κατανάλωση ισχύος, λειτουργίες I/O κ.α. Τα στοιχεία αυτά υλοποιούνται είτε σε μη επαναπρογραμματιζόμενο υλικό (VLSI, ASICs) είτε σε προγραμματιζόμενο υλικό (PLD, FPGA) και διασυνδέονται μέσω μιας ιεραρχίας (πιθανώς πολλών επιπέδων) διαύλων με τον μικροεπεξεργαστή και τη μνήμη.

Δεδομένης της επιλογής συγκεκριμένου Ε.Σ., η ζητούμενη εφαρμογή χωρίζεται σε κομμάτια που θα υλοποιηθούν στον επεξεργαστή του Ε.Σ. και κομμάτια τα οποία θα υλοποιηθούν σε υλικό. Ο τρόπος διαχωρισμού των κομματιών αυτών αποτελεί το στάδιο σχεδίασης υλικού-λογισμικού και καθορίζεται από τις απαιτήσεις (constraints) του τελικού Ε.Σ. σε απόδοση, σε μέγεθος, κατανάλωση ισχύος κ.α.

2.2 Ανάλυση Ενσωματωμένων Συστημάτων

Η xilinx παρέχει ένα πακέτο από εργαλεία μαζί με κάθε board που βοηθούν στην σχεδίαση και ανάπτυξη εφαρμογών. Το EDK (Embedded Development Kit) είναι όλα τα εργαλεία συστήματος που επιτρέπουν να σχεδιαστεί μια πλήρη εφαρμογή ενσωματωμένου επεξεργαστή σε ένα board xilinx FPGA.

Το EDK περιλαμβάνει:

- Το Xilinx Platform Studio (XPS) σύστημα που είναι μια σουίτα εργαλείων με τα οποία μπορούμε να αναπτύξουμε εφαρμογή με ενσωματωμένο επεξεργαστή.
- Το Software Development Kit (SDK), με βάση το Eclipse open-source πλαίσιο, το οποίο μπορούμε να χρησιμοποιήσουμε για την ανάπτυξη ενσωματωμένων εφαρμογών λογισμικού.

Το SDK είναι επίσης διαθέσιμο ως αυτόνομο πρόγραμμα. Το SDK παρέχει ενσωματωμένες μεθόδους επεξεργασίας περιλαμβανόμενων των προσχεδιασμένων επεξεργαστών καθορισμένων λειτουργιών και περιφερειακών από την xilinix.

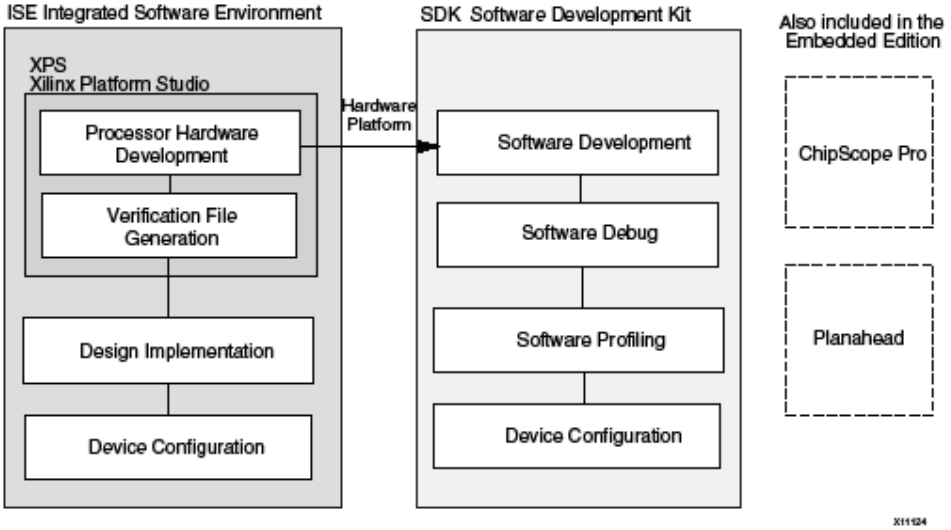


Figure 1-1: **Basic Embedded Design Process Flow**

2.3 Ανάπτυξη Υλικού

Η xilinx επιτρέπει την δημιουργία υποσυστήματος δικής μας λογικής σχεδίασης που θα είναι πιο ευέλικτο στις δικές μας ανάγκες. Η πλατφόρμα σχεδίασης της xilinx επιτρέπει την προσθήκη στο πυρήνα ενός η περισσότερων επεξεργαστών και περιφερειακών.

2.4 Ανάπτυξη Εφαρμογών

Η πλατφόρμα που είναι υπεύθυνη για το software κομμάτι της ανάπτυξης καθορίζει τους σωστούς software driver/library και δίνει την δυνατότητα να επιλέξουμε αν θέλουμε **να** ξεκινήσει από έτοιμο λογισμικό ή να χτίσουμε ένα νέο λογισμικό στις δικές μας ανάγκες.

2.5 Διαχείριση Συσκευών

Όταν οι software και hardware πλατφόρμες είναι έτοιμες τότε πρέπει να δημιουργηθεί το bit stream (δυαδική μορφή της σχεδίασης που θα τρέξει την εφαρμογή στο board) για το συγκεκριμένο board που θέλουμε να δουλέψουμε.

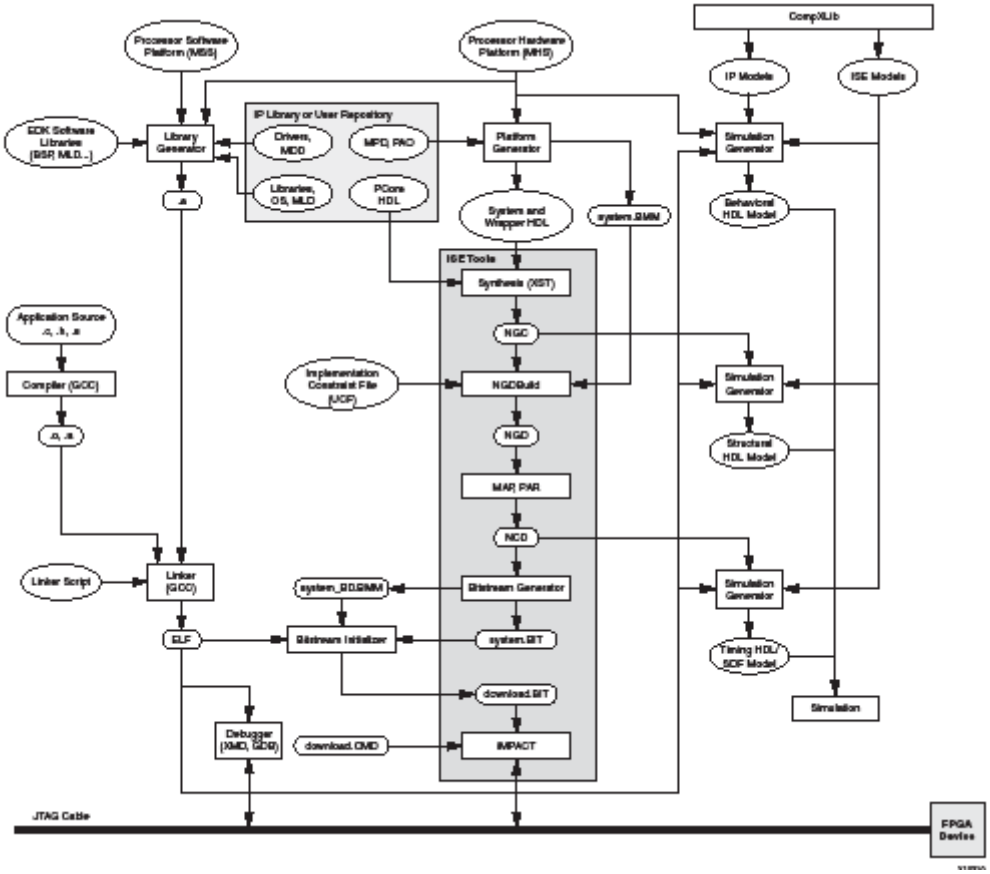
Η προτυποποίηση για το κατέβασμα του bit stream γίνετε με οποιοδήποτε πρόγραμμα, τρέχει ενσωματωμένα συστήματα και είναι συνδεδεμένο με ένα υπολογιστή

Για την παραγωγή, αποθήκευση και κατέβασμα του bit stream χρησιμοποιούνται σειριακά καλώδια, μνήμες και xilinx programmer.

2.6 Επισκόπηση EDK

Ένα ενσωματωμένο σύστημα τυπικά μπορεί να περιέχει έναν η περισσότερους πυρήνες, περιφερειακά και block μνήμης συνδεδεμένα με processor buses. Τα συστήματα περιέχουν και port εξωτερικών συνδέσεων πχ ethernet, compactflash, usb, vga, serial port. Κάθε επεξεργαστής έχει ένα αριθμό παραμέτρων που μπορούμε να τις χρησιμοποιήσουμε όπως έχουν, αλλά και να τις προσαρμόσουμε στις ανάγκες μας**.** Στο σύστημα επίσης περιέχετε και ο χάρτης με τις διευθύνσεις των διάφορων μνημών και περιφερειακών.

Η πιο κάτω εικόνα παρουσιάζει αναλυτικά την αρχιτεκτονική δομή μαζί με τα σχεδιαστικά εργαλεία που χρειάζονται για να στηθεί σωστά ένα ενσωματωμένο σύστημα.



2.7 Xilinx Platform Studio (XPS)

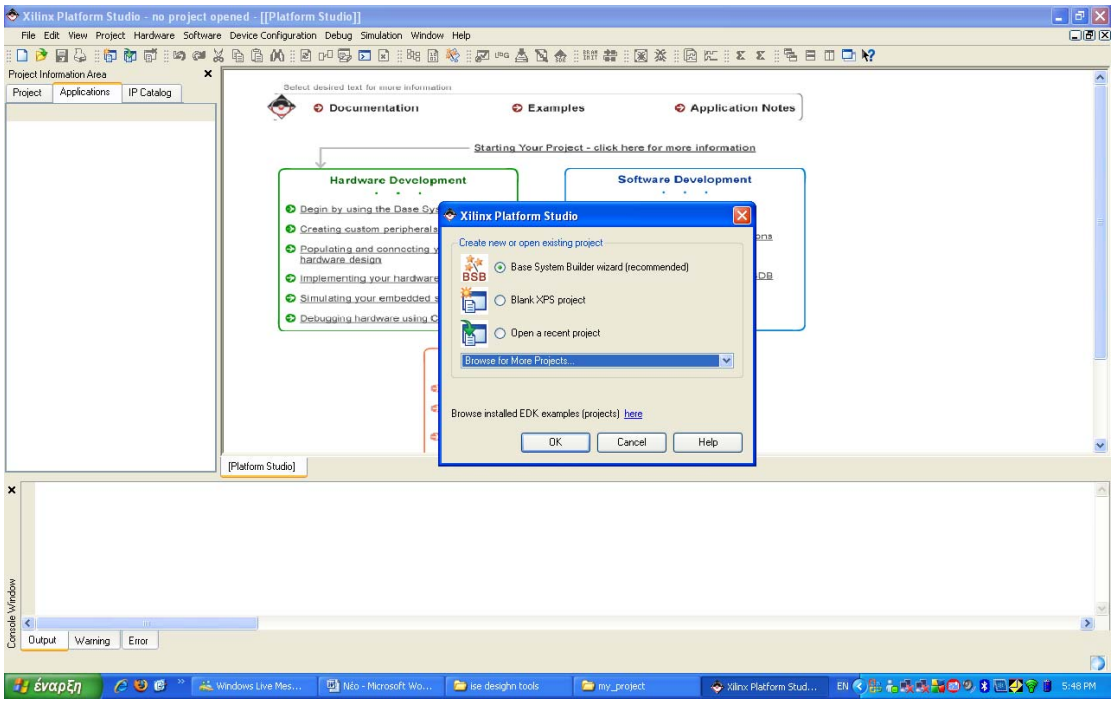
Το XPS παρέχει ένα ολοκληρωμένο περιβάλλον για τη δημιουργία ενσωματωμένων συστημάτων που βασίζονται σε δυο τύπους επεξεργαστών MicroBlaze και PowerPC Επίσης το XPS παρέχει ένα πρόγραμμα επεξεργασίας και ένα σχέδιο διαχείρισης της διεπαφής για να δημιουργήσουμε και να επεξεργαστούμε τον πηγαίο κώδικα. Το XPS προσφέρει προσαρμογή της ροής του εργαλείου με επιλογές διαμόρφωσης και παρέχει ένα γραφικό σύστημα για τη σύνδεση των επεξεργαστών, των περιφερειακών και των buses. Από XPS, μπορούμε να εκτελέσουμε όλα τα ενσωματωμένα εργαλεία συστήματος που απαιτούνται για όλα τα hardware components της εφαρμογής. Μπορούμε επίσης να εκτελέσουμε τον έλεγχο του συστήματος στο περιβάλλον XPS.

ΚΕΦΑΛΑΙΟ 3 : ΔΗΜΙΟΥΡΓΙΑ ΝΕΟΥ PROJECT ΑΠΟ ΤΟ XILINX PLATFORM STUDIO 10.1 WIZARD

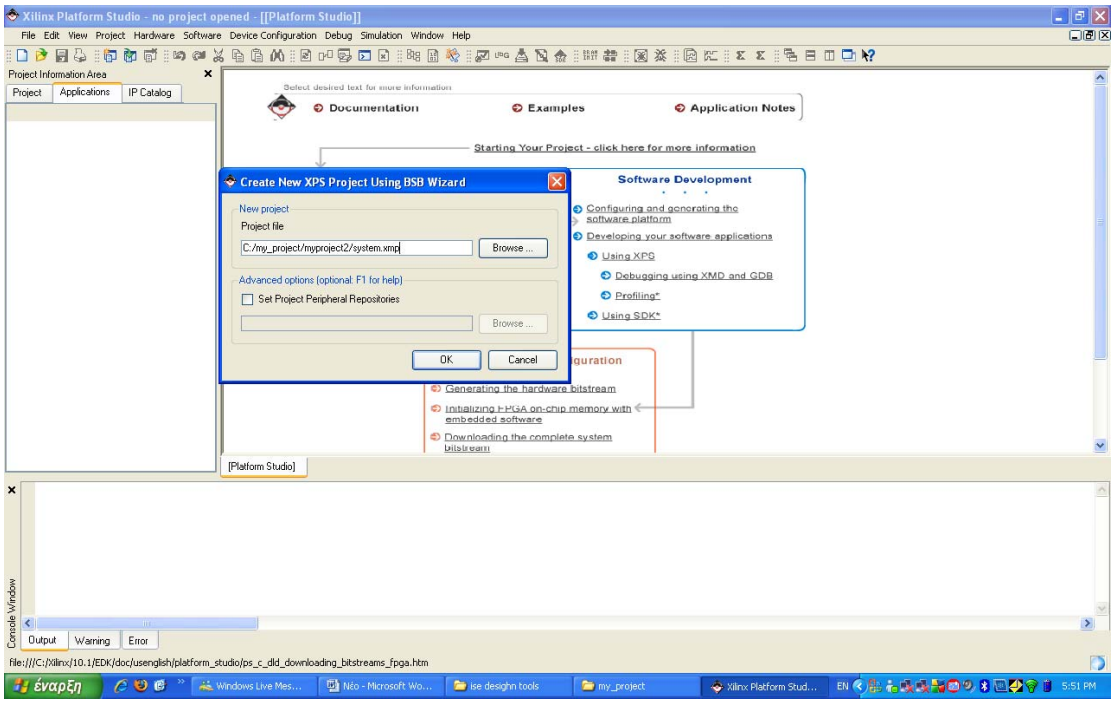
Το wizard platform studio 10.1 είναι ένας οδηγός που μας βοήθα να στήσουμε σωστά μια εφαρμογή που να δουλεύει. Για πολλές εφαρμογές δεν χρειάζεται τίποτε άλλο παρά μόνο XPS. Με βάση το board που θα επιλέξουμε, ο οδηγός μας επιτρέπει να επιλέξουμε και να ρυθμίσουμε τα βασικά στοιχεία του συστήματος όπως το είδος επεξεργαστή, debug interface, προσθήκη CACHE, τύπο μνήμης και μέγεθος, περιφερικά εισόδου / εξόδου.

3.1 Τυπική διαδικασία δημιουργίας νέου συστήματος.

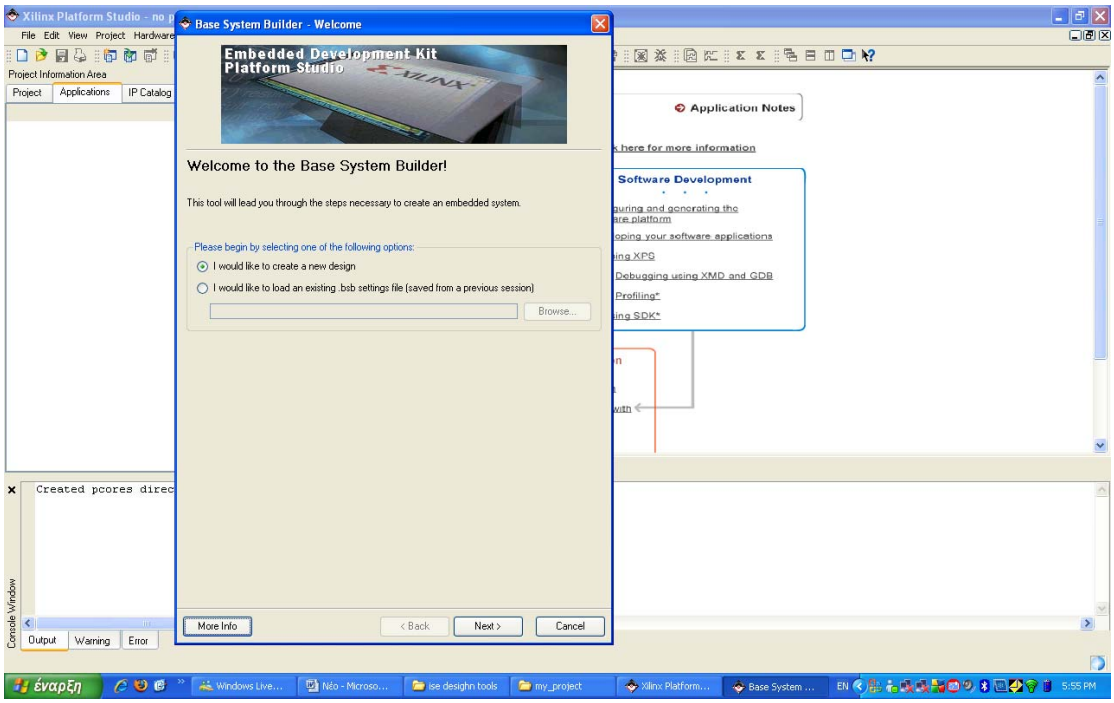
Όταν ενεργοποιήσουμε το πρόγραμμα σχεδίασης της xilinx μας εμφανίζει αυτόματα τις επιλογές, αν θέλουμε να δημιουργήσουμε νέο project με την βοήθεια του wizard.



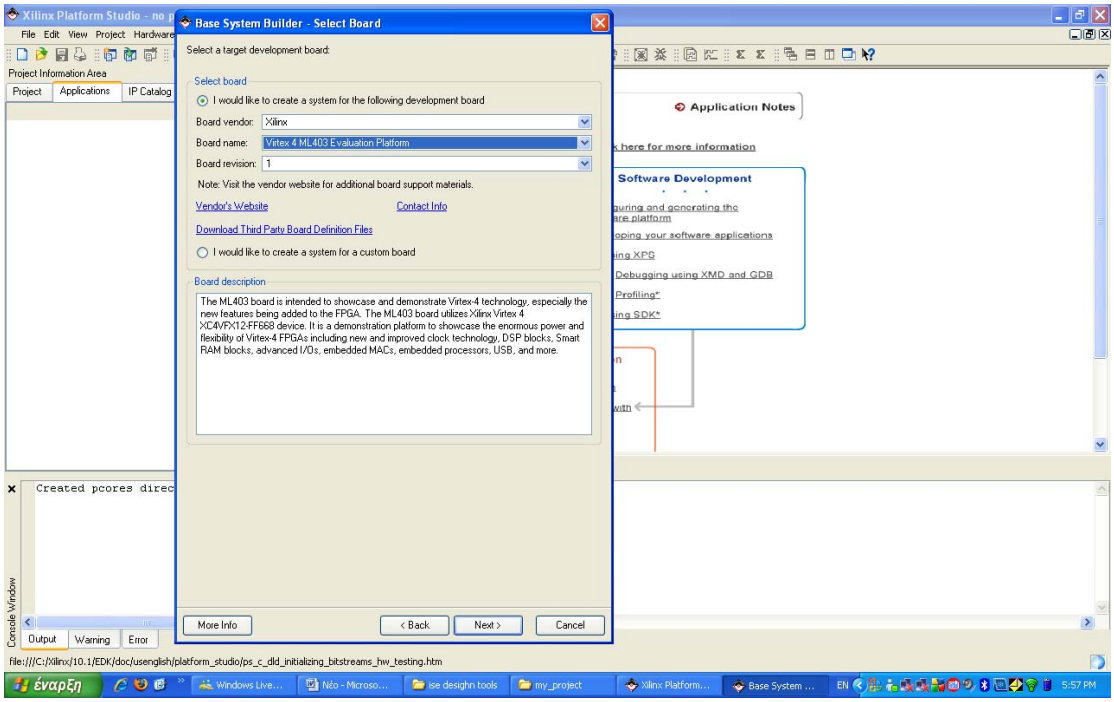
Το αμέσως επόμενο βήμα είναι να του καθορίσουμε το σημείο στο δίσκο που θα αποθηκεύσει όλα τα απαιτούμενα αρχεία της εφαρμογής.



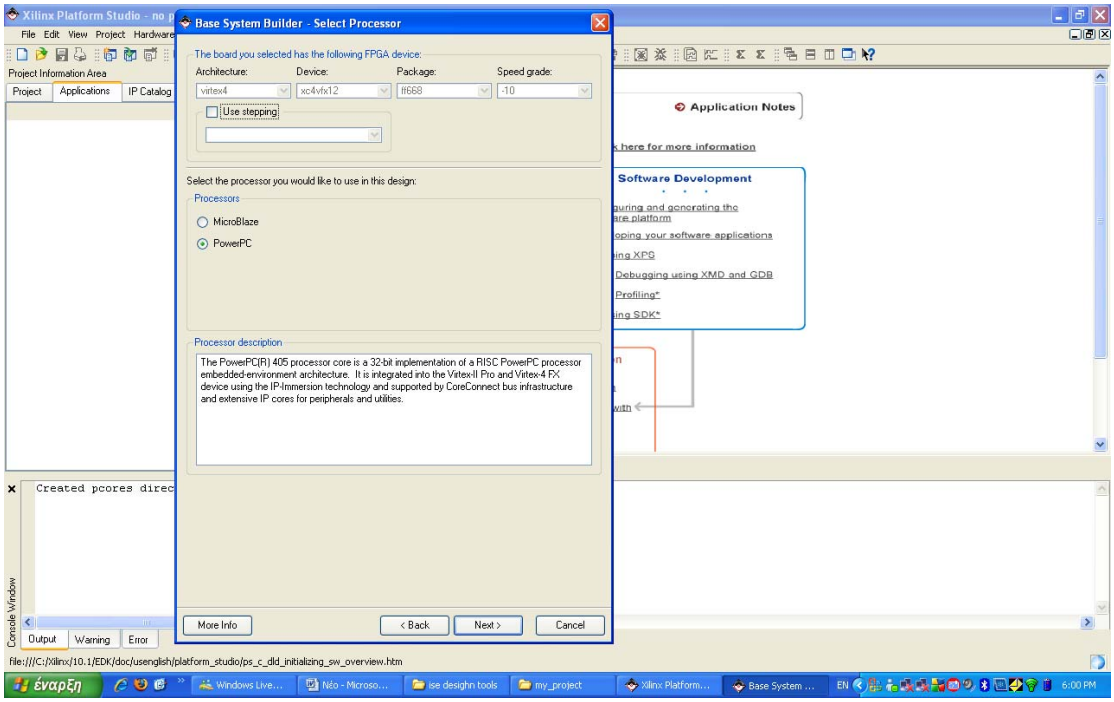
Στη συνέχεια αρχίζει η ουσιαστική δημιουργία της εφαρμογής που θα χρησιμοποιήσουμε.



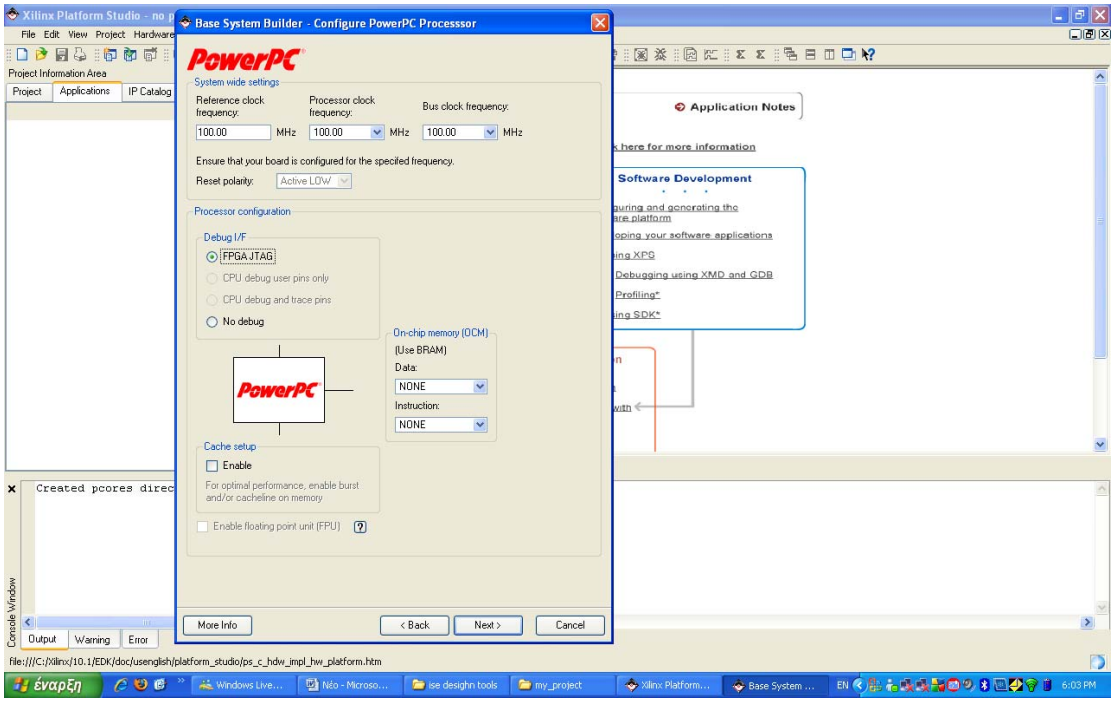
Επιλέγουμε το board που θα χρησιμοποιήσουμε**.** Στην δική μας εφαρμογή είναι το Virtex 4 ML_403.



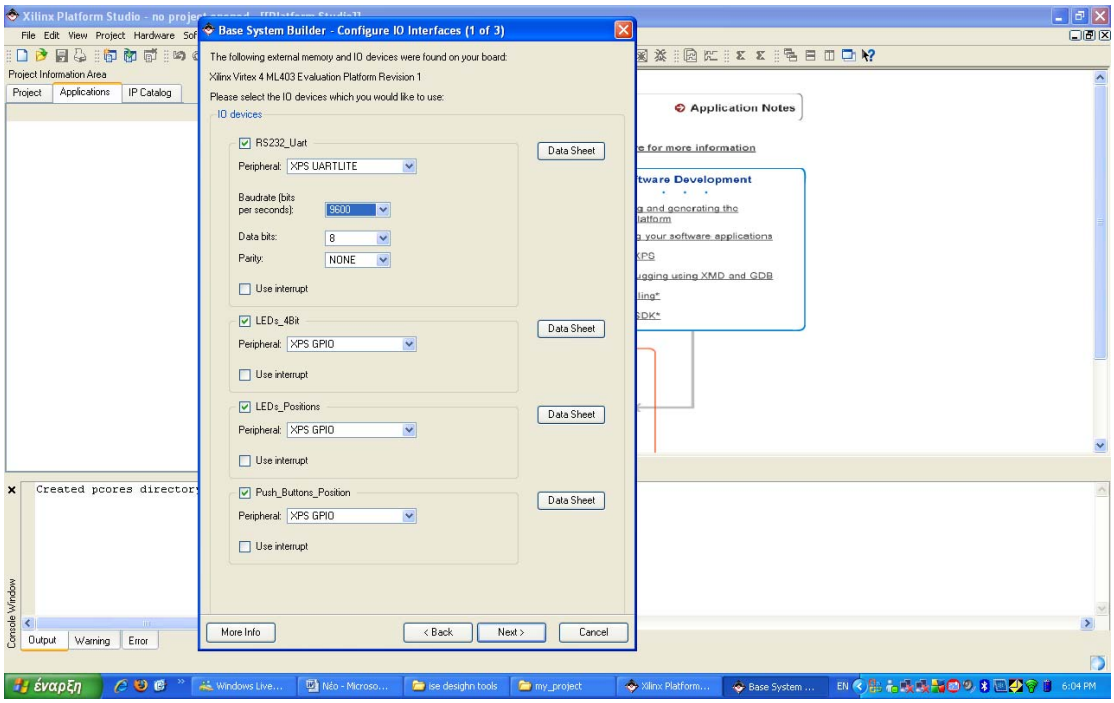
Στο επόμενο βήμα επιλέγουμε τον επεξεργαστή που θέλουμε να χρησιμοποιήσουμε. Υπάρχουν δύο έτοιμοι επεξεργαστές ο MicroBlaze και PowerPC.



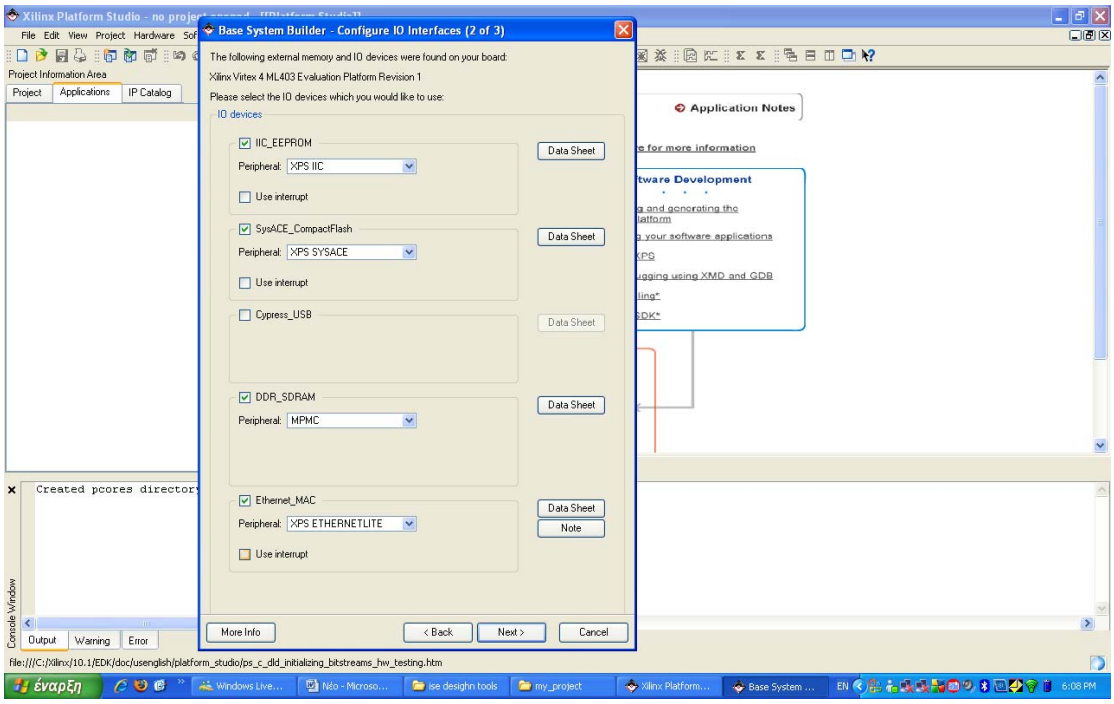
Μας εμφανίζει τα χαρακτηριστικά του επεξεργαστή που έχουμε επιλέξει, στην προκειμένη περίπτωση τον PowerPC.



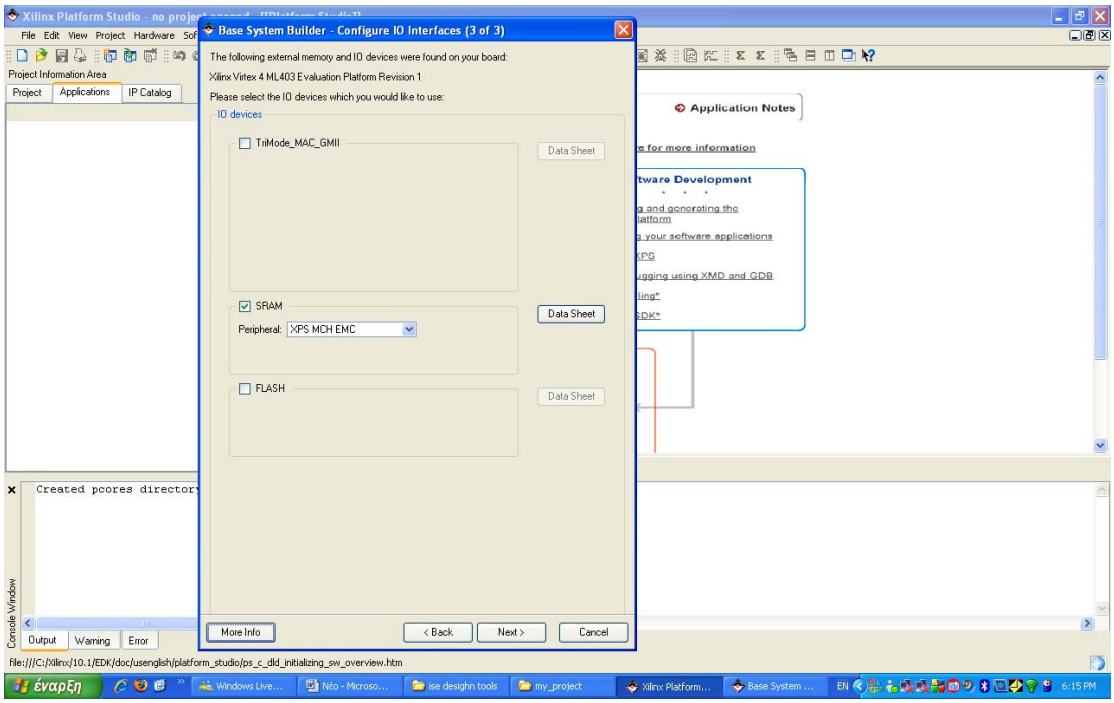
Στο επόμενο στάδιο καθορίζουμε τους τρόπους επικοινωνίας, δηλαδή τη συριακή πόρτα, τα led ,που θα μας καθοδηγούν στη σωστή λειτουργία το συστήματος καθώς και τα κουμπιά που μας δίνουν τη δυνατότητα ελέγχου του συστήματος.

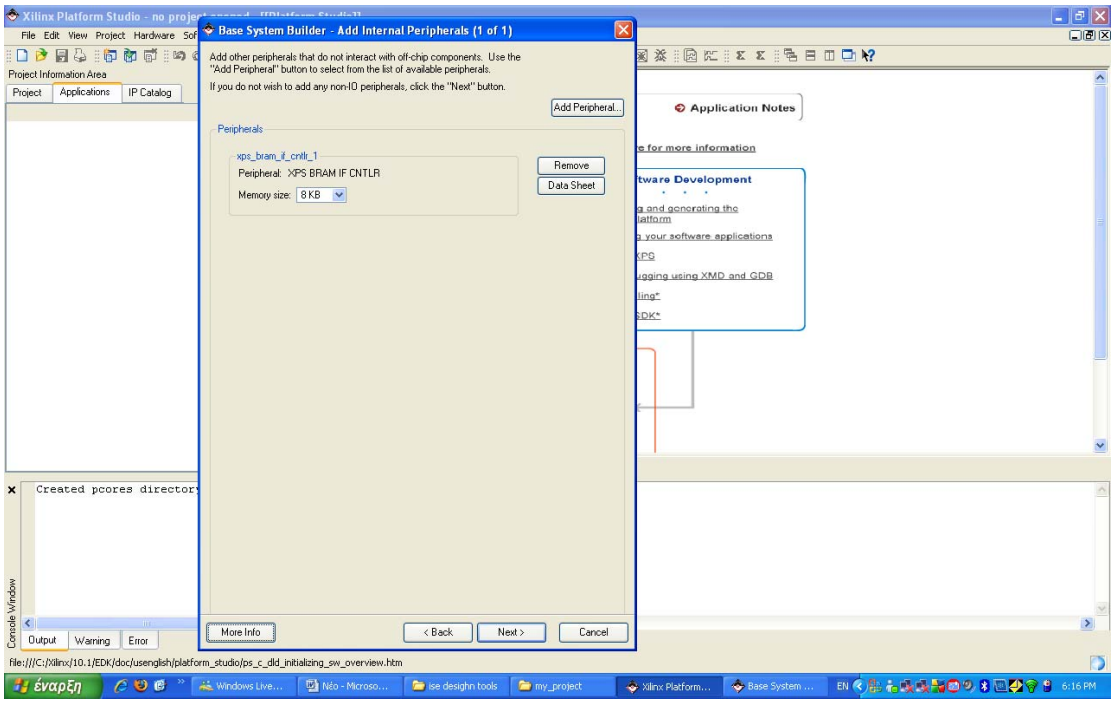


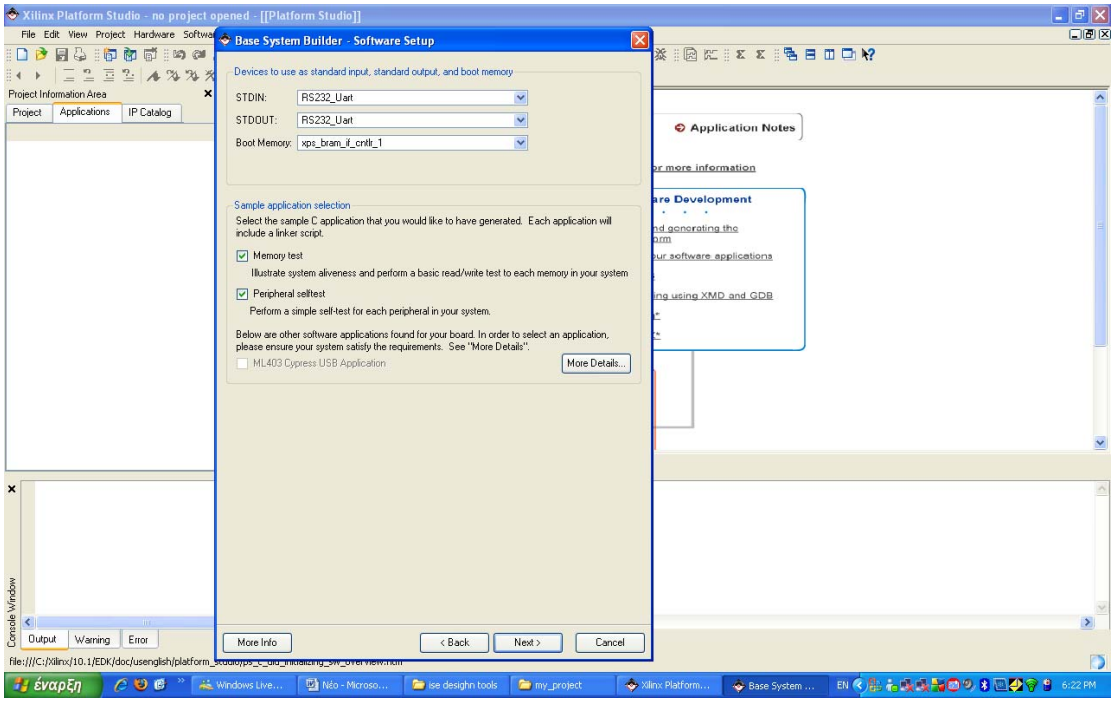
Φτάνουμε στο σημείο που καθορίζουμε τις διάφορες μνήμες που θα χρησιμοποιεί η εφαρμογή μας καθώς επίσης και τις συσκευές εισόδου / εξόδου όπως ethernet και compactflash memory.

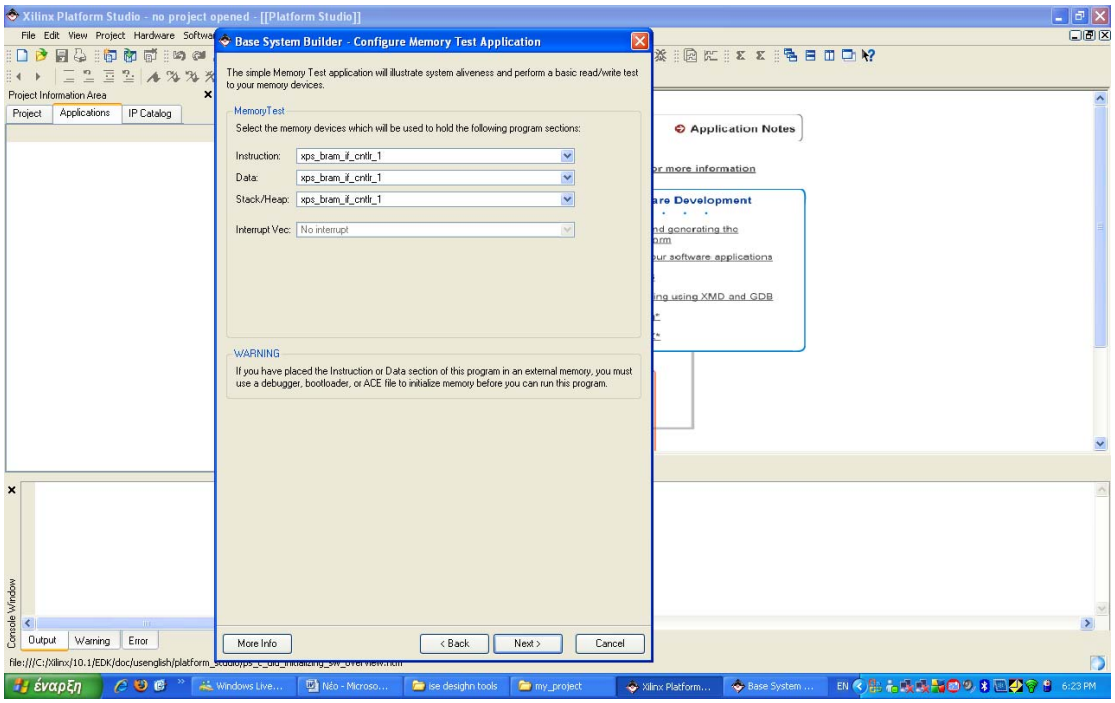


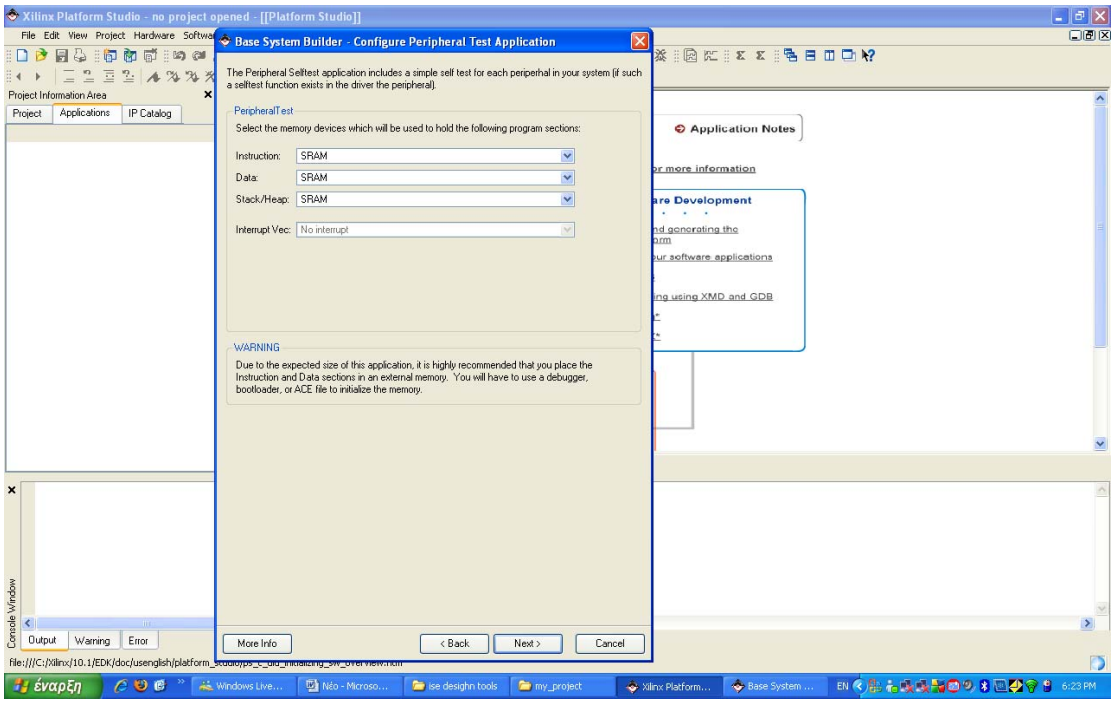
Τα επόμενα τρία βήματα καθορίζουν τις μνήμες που θα χρησιμοποιηθούν, με ποία δεδομένα θα έχουν σχέση και πού θα στέλνει τα αποτελέσματα.



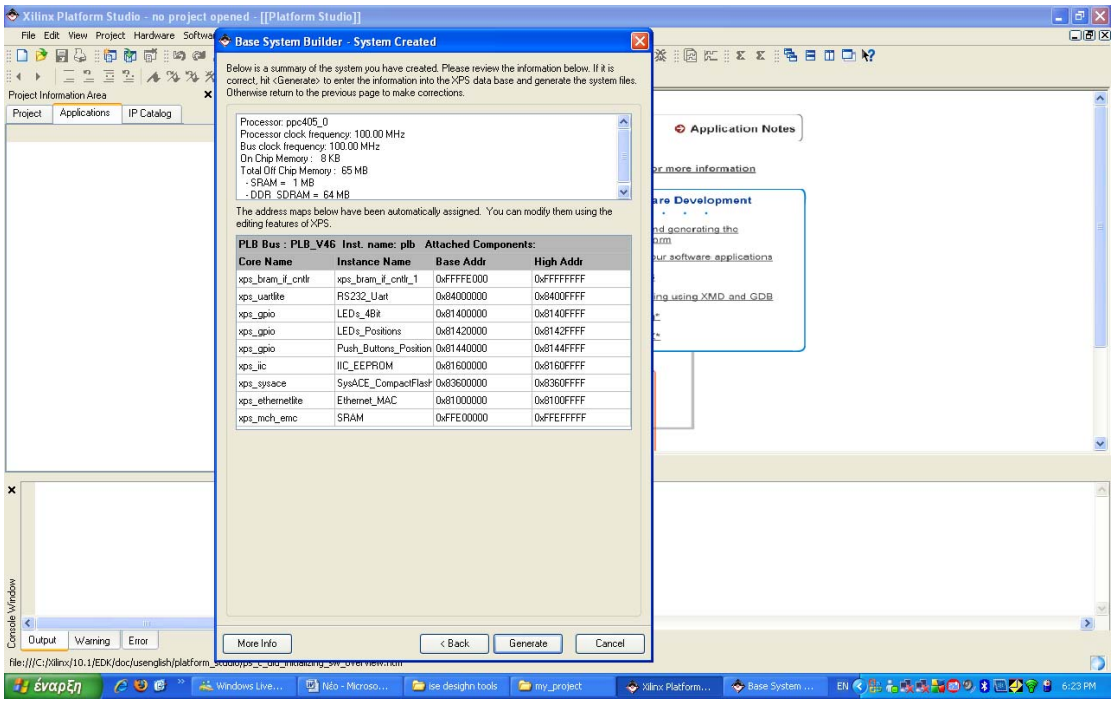

Base System Builder - Add Internal Peripherals (1 of 1)
Add other peripherals that do not interact with off-chip components. Use the "Add Peripheral" button to select from the list of available peripherals.
If you do not wish to add any non-IO peripherals, click the "Next" button.
Add Peripheral...
Peripherals
xps_bram_if_cntlr_1
Peripheral: XPS BRAM IF CNTLR
Memory size: 8 KB
Remove
Data Sheet
More Info
< Back
Next >
Cancel
Application Notes
Created pcores director
Output
Warning
Error
Console Window


Base System Builder - Software Setup
Devices to use as standard input, standard output, and boot memory
STDIN: RS232_Uart
STDOUT: RS232_Uart
Boot Memory: xps_bram_if_cntlr_1
Sample application selection
Select the sample C application that you would like to have generated. Each application will include a linker script.
Memory test
Illustrate system aliveness and perform a basic read/write test to each memory in your system
Peripheral selftest
Perform a simple self-test for each peripheral in your system.
Below are other software applications found for your board. In order to select an application, please ensure your system satisfy the requirements. See "More Details".
ML403 Cypress USB Application
More Details...
More Info
< Back
Next >
Cancel
Application Notes
Output
Warning
Error
Console Window

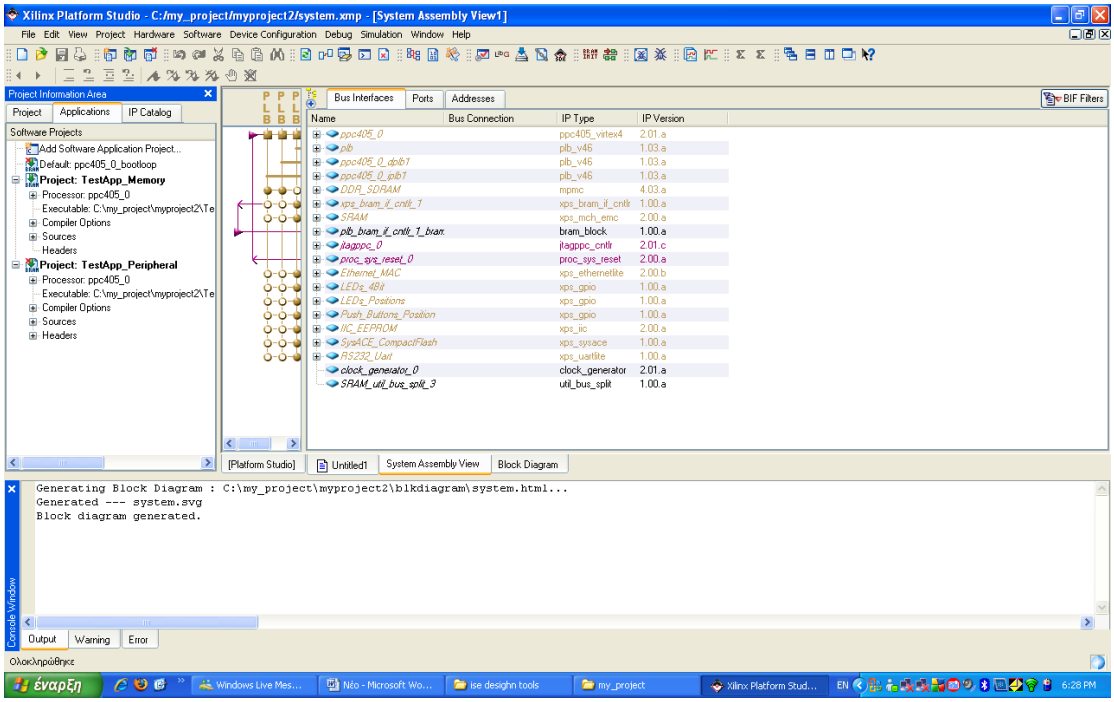

Xilinx Platform Studio - no project opened - [[Platform Studio]]
Base System Builder - Configure Memory Test Application
The simple Memory Test application will illustrate system aliveness and perform a basic read/write test to your memory devices.
MemoryTest
Select the memory devices which will be used to hold the following program sections:
Instruction: xps_bram_if_cntlr_1
Data: xps_bram_if_cntlr_1
Stack/Heap: xps_bram_if_cntlr_1
Interrupt Vec: No interrupt
WARNING
If you have placed the Instruction or Data section of this program in an external memory, you must use a debugger, bootloader, or ACE file to initialize memory before you can run this program.
More Info
< Back
Next >
Cancel
Application Notes

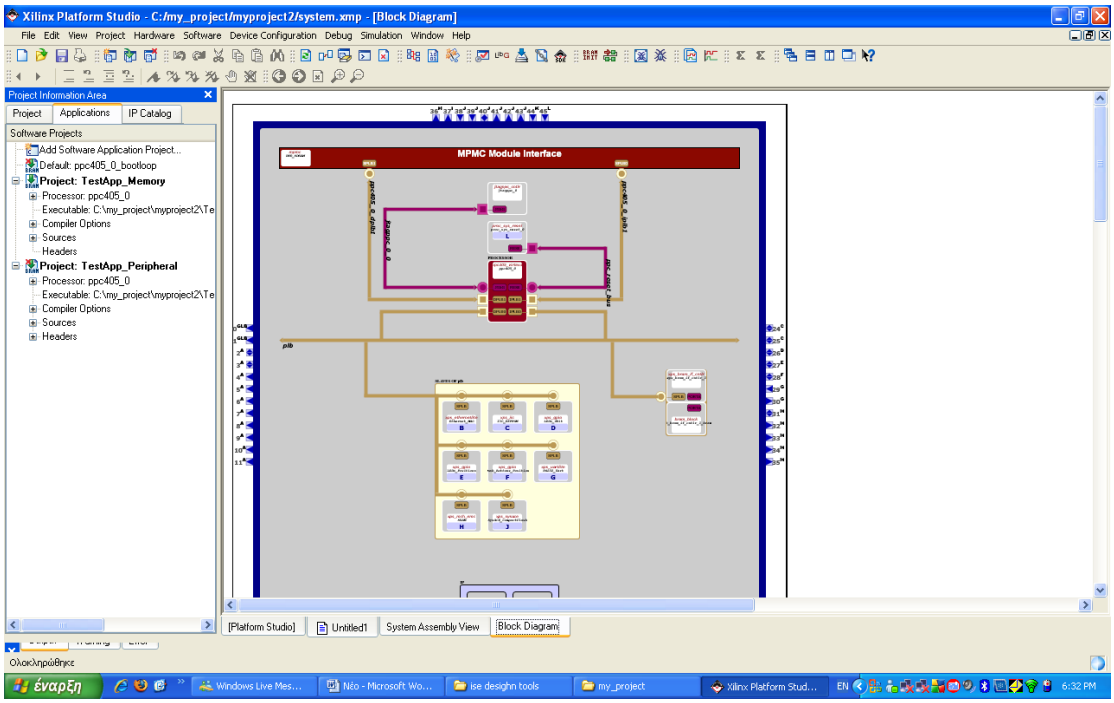

Xilinx Platform Studio - no project opened - [[Platform Studio]]
Base System Builder - Configure Peripheral Test Application
The Peripheral Selftest application includes a simple self test for each peripehal in your system (if such a selftest function exists in the driver the peripheral).
PeripheralTest
Select the memory devices which will be used to hold the following program sections:
Instruction: SRAM
Data: SRAM
Stack/Heap: SRAM
Interrupt Vec: No interrupt
WARNING
Due to the expected size of this application, it is highly recommended that you place the Instruction and Data sections in an external memory. You will have to use a debugger, bootloader, or ACE file to initialize the memory.
More Info
< Back
Next >
Cancel
Application Notes

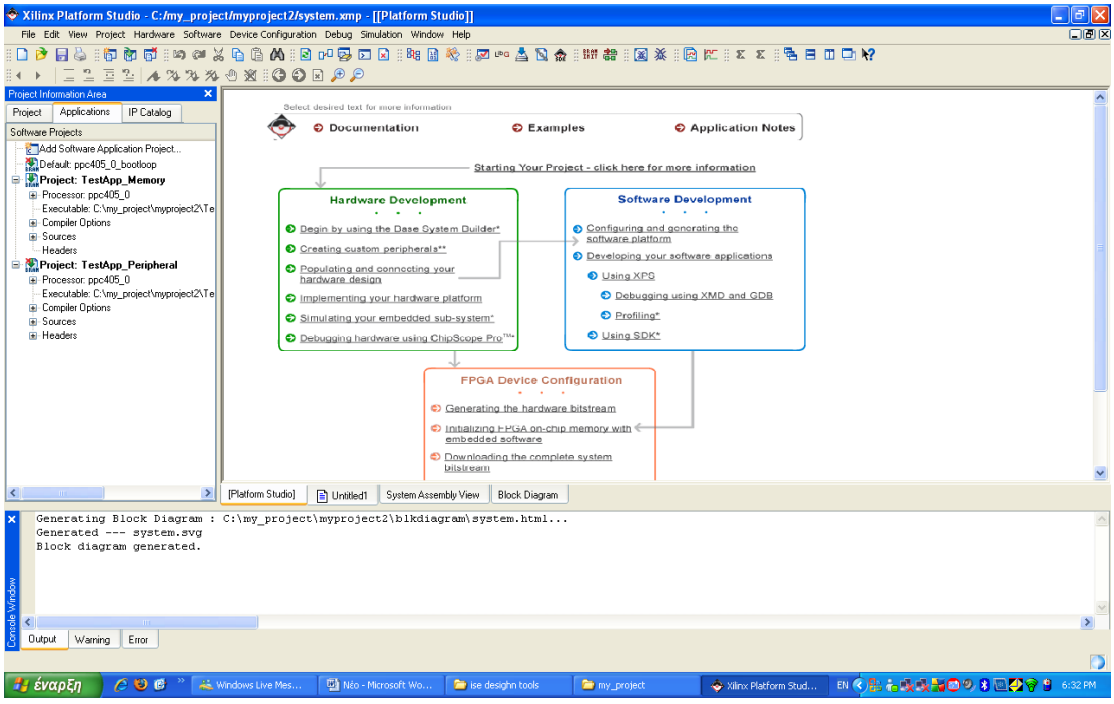
Στο τελευταίο στάδιο όσο αφορά τις μνήμες και συσκευές εισόδου / εξόδου, μας εμφανίζει ένα πίνακα με τα μεγέθη, από πού αρχίζουν και πού τελειώνουν, αφού έχει μεγάλο ρόλο όπως θα δούμε στα επόμενα στάδια.



Με την επιλογή Generate το σύστημα αρχίζει να δημιουργείτε και μας εμφανίζει όλα τα απαραίτητα σχεδιαγράμματα και πληροφορίες έτσι ώστε να γίνει πιο κατανοητό προς εμάς το τελικό ενσωματωμένο σύστημα που έχει δημιουργηθεί.




Xilinx Platform Studio - C:/my_project/myproject2/system.xmp - [Block Diagram]
File Edit View Project Hardware Software Device Configuration Debug Simulation Window Help
Project Information Area
Project Applications IP Catalog
Software Projects
Add Software Application Project...
Default: ppc405_0_bootloop
Project: TestApp_Memory
Processor: ppc405_0
Compiler Options
Sources
Headers
Project: TestApp_Peripheral
Processor: ppc405_0
Compiler Options
Sources
Headers
MPMC Module Interface
[Platform Studio]
Untitled1
System Assembly View
Block Diagram
έναρξη
Windows Live Mes...
Νέο - Microsoft Wo...
ise designh tools
my_project
Xilinx Platform Stud...
EN
6:32 PM


Xilinx Platform Studio - C:/my_project/myproject2/system.xmp - [[Platform Studio]]
File Edit View Project Hardware Software Device Configuration Debug Simulation Window Help
Project Information Area
Project Applications IP Catalog
Software Projects
Add Software Application Project...
Default: ppc405_0_bootloop
Project: TestApp_Memory
Processor: ppc405_0
Compiler Options
Sources
Headers
Project: TestApp_Peripheral
Processor: ppc405_0
Compiler Options
Sources
Headers
Select desired text for more information
Documentation
Examples
Application Notes
Starting Your Project - click here for more information
Hardware Development
Begin by using the Base System Builder*
Creating custom peripherals**
Populating and connecting your hardware design
Implementing your hardware platform
Simulating your embedded sub-system*
Debugging hardware using ChipScope Pro™
Software Development
Configuring and generating the software platform
Developing your software applications
Using XPS
Debugging using XMD and GDB
Profiling*
Using SDK*
FPGA Device Configuration
Generating the hardware bitstream
Initializing FPGA on-chip memory with embedded software
Downloading the complete system bitstream
[Platform Studio]
Untitled1
System Assembly View
Block Diagram
Generating Block Diagram : C:\my_project\myproject2\blkdiagram\system.html...
Generated --- system.svg
Block diagram generated.
Console Window
Output
Warning
Error
έναρξη
Windows Live Mes...
Νέο - Microsoft Wo...
ise designh tools
my_project
Xilinx Platform Stud...
EN
6:32 PM

3.2 Οδηγός Ανάπτυξης Εφαρμογών

Το SDK παρέχει εν περιβάλλων υλοποίησης software εφαρμογών.

- Μπορεί να χρησιμοποιηθεί το ISE και XPS σε μικρής χωρητικότητας δίσκους.
- Υποστηρίζει δημιουργία συστημάτων με πολυεπεξεργαστές.
- Μπορεί να βελτιστοποιεί τα κυκλώματα σχεδιασμού με αποτέλεσμα περισσότερη αποδοτικότητα και σταθερότητα.
- Υποστηρίζει εφαρμογές μόνο με πηγαίο κώδικα.
- Ελέγχει το σύστημα σε τελικό στάδιο όσο αφορά software και hardware.
- Παρέχει C/C++ editor αλλά και compilation environment.

3.3 Δημιουργία Βιβλιοθηκών

Το Libgen δημιουργεί τις απαραίτητες βιβλιοθήκες όσο αφορά τα driver της συσκευής, το file system και την διασύνδεση του ενσωματωμένου επεξεργαστή στο σύστημα. Η πλατφόρμα λογισμικού ορίζει για κάθε επεξεργαστή, τα driver που σχετίζονται με τις συσκευές που περιλαμβάνονται στη δική του πλατφόρμα υλικού, επιλεγμένες βιβλιοθήκες, συσκευές εισόδου και εξόδου, ρουτίνες χειρισμού, και άλλα συναφή χαρακτηριστικά λογισμικού. Παίρνοντας τις βιβλιοθήκες και τα driver από την εγκατάσταση, μαζί με βιβλιοθήκες και driver για κάθε περιφερειακό που παρέχεται, το SDK είναι σε θέση να δημιουργήσει, συμπεριλαμβανομένων των βιβλιοθηκών και των driver, ένα executable format (.ELF) που θα είναι έτοιμο να τρέχει στην πλατφόρμα του επεξεργαστή.

3.4 Bitstream Initialize

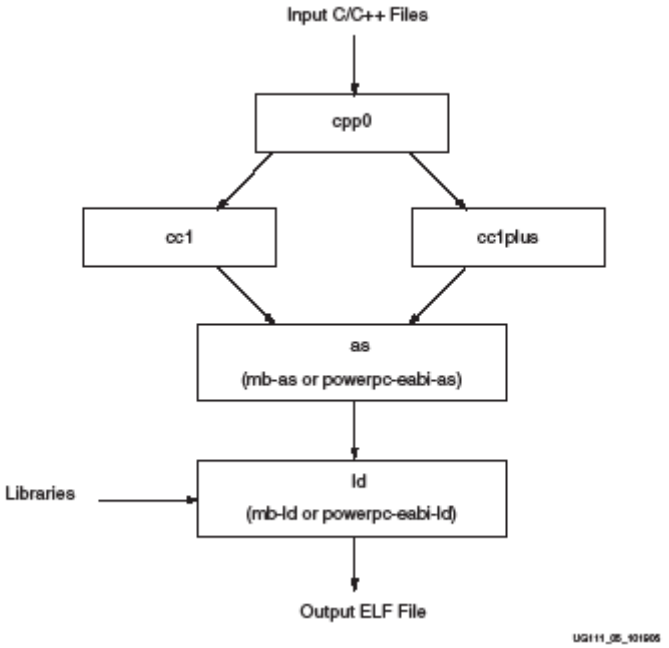
Το bitstream initialize είναι ένα εργαλείο που μας βοήθα να διασύνδεουμε τις διάφορες μνήμες του συστήματος με τον επεξεργαστή και πώς θα συμπεριφέρεται ο επεξεργαστής μέσα από το τις πληροφορίες που του έχουμε δώσει από το software.Το bitstream μπορεί να διαβαστεί μόνο από το hardware αφού είναι καθαρά σε γλώσσα μηχανής (Binary system) ,να υλοποιεί όλες τις λειτουργίες που έχουμε καθορίσει μέσα από το ISE και να ενσωματώσει όλες τις εφαρμογές του ELF αρχείου για κάθε επεξεργαστή ξεχωριστά

3.5Δημιουργία System ACE Αρχείου (GenACE)

Το XPS Generates Xilinx System Ace δημιουργεί ένα αρχείο από το FPGA bit stream ,ELF και data file. Το αρχείο αυτό κάνει την ίδια δουλειά με το bit stream με την διάφορα όμως ότι το system.ace δημιουργείται για να ξεκινά την εφαρμογή από την εξωτερική κάρτα μνήμης του board compact flash.Το system.ace μπορεί να δημιουργηθεί για οποιοδήποτε τύπο επεξεργαστή, για οποιοδήποτε περιφερειακό υποστηρίζει το board με την μόνη προϋπόθεση το board να υποστηρίζει εκκίνηση συστήματος από compactflash κάρτα.

3.6 Gnu Compiler

Ο Gnu Compiler περιλαμβάνετε στο EDK και χρησιμοποιεί το binary system για εξαγωγή εκτελέσιμου αρχείου. Τα χαρακτηριστικά της διαδικασίας που χρειάζεται για να γίνει compile ένας κώδικας όσο αφορά το PowerPC και τον MicroBlaze φαίνονται στο πιο κάτω σχήμα.



ΚΕΦΑΛΑΙΟ 4 :ΠΕΙΡΑΜΑΤΙΚΟ ΣΤΑΔΙΟ

4.1 Εφαρμογές Εικόνων

Στο στάδιο της δημιουργίας κώδικα θα τρέξουμε σε διαφορετικούς επεξεργαστές και με διαφορετικές μνήμες επικοινωνίας τις εφαρμογές για να δούμε την απόδοση των επεξεργαστών. Δημιουργήσαμε δυο κώδικες οι οποίοι στην ουσία έχουν κύριο σκοπό το φιλτράρισμα εικόνων τύπου PGM. Για να μπορέσουμε να περάσουμε την εικόνα μέσα στον κώδικα της εφαρμογής τρέξαμε κάποια εικόνα τύπου .pgm σε ένα άλλο κώδικα που μετατρέπει την εικόνα σε ένα αρχείο .txt και στα περιεχόμενα του αρχείου αυτού μετατρέπει την εικόνα σε ένα δυδιάστατο μη προσημασμένο πίνακα χαρακτήρων. Ο πίνακας αυτός πρέπει να είναι οπωσδήποτε στατικός όταν χρησιμοποιηθεί από κάποιον κώδικα για να μην έχουμε αλλαγές και ανακριβή αποτελέσματα στις μετρήσεις μας.

Ας εστιάσουμε όμως στους δυο κύριους κώδικες που θα γίνουν τα πειράματα για να εκτιμήσουμε τις επιδώσεις των επεξεργαστών με διαφορετικά χαρακτηριστικά κάθε φορά. Τα κύρια χαρακτηριστικά και των δυο εφαρμογών που θα υλοποιήσουμε σε κώδικα C είναι να διαβάζουμε μια εικόνα από ένα στατικό πίνακα τον οποίο θα τον περάσουμε από κάποιες συναρτήσεις για να βγάλουμε αποτελέσματα. Τα αποτελέσματα από την εκτέλεση των εφαρμογών θα στέλνονται μέσω μιας συριακής θύρας σε ένα υπολογιστή ο οποίος θα κάνει καταγράφει τα δεδομένα σε ένα αρχείο. Ο πρώτος κώδικας θα ασχολείται με την αλλαγή των αντιθέσεων των χρωμάτων σε οριζόντια επίπεδα και αφού αναγνωρίζει μέσα από τις συναρτήσεις τις αντιθέσεις, έπειτα θα δημιουργεί ένα δεύτερο πίνακα ο οποίος θα τυπώνει με αχνές γραμμές μόνο τις εναλλαγές σε οριζόντιο επίπεδο. Ο πίνακας αυτός θα στέλνεται όπως προαναφέραμε μέσω συριακής θύρας σε ένα υπολογιστή που θα καταγράφει τα δεδομένα σε ένα αρχείο .txt το οποίο θα το ανοίξουμε στη συνεχεία με ένα image editor για να δούμε την φιλτραρισμένη εικόνα(πχ ένα σκάκι προσπαθεί να βρει τις οριζόντιες γραμμές ανάμεσα στα τετράγωνα και να τις βάζει στο νέο πίνακα που δημιουργεί). Ο κώδικας αυτός λέγεται gradient βρίσκεται στο **Παραρτημα1**.Ο δεύτερος κώδικας θα ασχολείται πάλι με την αλλαγή των χρωμάτων, όμως αυτή τη φορά δεν θα είναι σε οριζόντιο επίπεδο αλλά και σε κατακόρυφο με σκοπό να προσπαθεί να αναγνωρίσει τα περιγράμματα από την εικόνα(πχ ένα σκάκι προσπαθεί να βρει τα περιγράμματα από τα τετράγωνα και τα πιόνια). Η δουλειά αυτή θα γίνεται πάλι με συναρτήσεις και με τον ίδιο τρόπο όπως στον πρώτο κώδικα

θα στέλνονται και θα καταγράφονται σε ένα υπολογιστή. Ο δεύτερος κώδικας λέγεται canny και βρίσκεται στο **Παραρτημα2**.

4.2 Επιλογή Επεξεργαστή

Το στάδιο αυτό είναι το κύριο πειραματικό μέρος της εφαρμογής αφού εδώ θα πάρουμε όλες τις μετρήσεις για να εκτιμήσουμε την αποδοτικότητα του κάθε επεξεργαστή σε συνδυασμό με κάποια περιφερειακά και με διαφορετικές παραμέτρους κάθε φορά. Όπως αναφέραμε και σε προηγούμενο κεφαλαίο όλα σχεδιάζονται από το XPS(Xilinx Platform Studio) το οποίο είναι το κύριο εργαλείο για της επεξεργασία όλων των λειτουργιών αλλά και τον καθορισμό όλων των περιφερειακών που θα συνδυαστούν με τον επεξεργαστή και τελικά θα τρέξουν την εφαρμογή.

Οι επεξεργαστές που μπορούμε να δοκιμάσουμε στην εφαρμογή μας είναι δυο και δίνονται έτοιμοι από το XPS για να χρησιμοποιηθούν απευθείας σε εφαρμογή. Οι δυο αυτοί επεξεργαστές δεν διαφέρουν σε χαρακτηριστικά (πχ 32bit) αλλά διαφέρουν στη αρχιτεκτονική τους σχεδίαση με αποτέλεσμα να συμπεριφέρονται διαφορετικά στις ίδιες εφαρμογές. Οι επεξεργαστές είναι ο PowerPC και MicroBlaze. Και σους δυο αυτούς επεξεργαστές μπορούμε να εφαρμόσουμε σχεδόν όλα τα εργαλεία περιφερειακά και παραμέτρους που είναι διαθέσιμα από το board έτσι μπορούμε με την ίδια εφαρμογή να πάρουμε αποτελέσματα και τελικά να συγκρίνουμε τα επίπεδα αποδοτικότητας του κάθε επεξεργαστή.

Ο PowerPC από τις οδηγίες που δίνει ο κατασκευαστής είναι σχετικά πιο αποδοτικός από τον MicroBlaze όμως αυτό θα το δούμε μέσα από τις πειραματικές μας μετρήσεις. Αυτό που μπορούμε αρχικά να πούμε είναι ότι στον PowerPC δεν μπορούμε να εφαρμόσουμε CACHE μνήμη απευθείας στον επεξεργαστή και αν τελικά πρέπει να εφαρμόσουμε την CACHE μνήμη πρέπει να παραμετροποιήσουμε την DDR μνήμη έτσι ώστε να έχει την δική της CACHE μνήμη. Επιπλέον όλες τις μετρήσεις θα τις πάρουμε από timer που θα εφαρμόσουμε στο σύστημα. Στον PowerPC μπορούμε να εφαρμόσουμε και software και hardware timer ενώ στον MicroBlaze μόνο software timer.

4.3 Διαθέσιμες μνήμες

Το επόμενο σημαντικό ζήτημα στην σχεδίαση και υλοποίηση κάποιας εφαρμογής είναι η επιλογή μνήμης που θα επιλέξουμε να φορτώσει και να τρέξει την εφαρμογή μας. Όπως ξέρουμε γενικά από τα συστήματα υπολογιστών το μέγεθος της μνήμης και η συχνότητα λειτουργίας της δίνουν μια καλή απόδοση στο σύστημα σε συνδυασμό με την επεξεργαστική ισχύ. Αν ένα σύστημα με πολύ καλό επεξεργαστή υστερεί σε μια ικανοποιητικού μεγέθους και συχνότητα λειτουργίας μνήμη τότε το σύστημα αυτό δεν θα δίνει σε καμία περίπτωση το 100% των προδιαγραφών του.

Ας εξετάσουμε λοιπόν ποιες μνήμες διαθέτει το σύστημα μας και τί μπορούμε να κάνουμε με αυτές για να πετύχουμε όσο το δυνατόν περισσότερη απόδοση. Το FPGA Ml403 board της xilinx διαθέτει τρεις τύπους μνήμης που μπορείς άμεσα να τους χρησιμοποιήσεις σε συνδυασμό με τον επεξεργαστή. Οι τρεις αυτοί τύποι μνήμης είναι η XPS_BRAM, η SRAM_C και η DDR_SDRAM. Σε κάθε περίπτωση η κάθε μνήμη έχει τα πλεονεκτήματα και τα μειονεκτήματα της σε σχέση με τις άλλες δυο. Ας δούμε λοιπόν πιο αναλυτικά τα χαρακτηριστικά της κάθε μιας ξεχωριστά για να κατανοήσουμε σε τι μπορούμε να χρησιμοποιήσουμε την κάθε μια.

Η XPS_BRAM είναι η πιο μικρή μνήμη από τις τρεις, η οποία είναι μεγέθους 8Kbytes και βρίσκεται μέσα στο chip που είναι σχεδιασμένος ο επεξεργαστής σε block μνήμης ανάμεσα στα CLB block. Όπως μπορούμε να καταλάβουμε αυτή η μνήμη θα είναι σε κάθε περίπτωση η πιο αποδοτική αφού δεν χρειάζεται εξωτερικά processor buses να επικοινωνήσει με τον επεξεργαστή έτσι μειώνονται οι αποστάσεις και μειώνεται σημαντικά και ο χρόνος αλληλεπίδρασης. Τελικά όμως όσο και να ακούγεται ιδανικό δεν είναι καθόλου λειτουργικός αυτός ο τύπος μνήμης με τα 8Kbytes που διαθέτει, δεν μπορεί να τρέξει μεγάλες εφαρμογές και μάλλον έχει σχεδιαστεί για δόκιμες σε θέματα βελτιστοποίησης.

Ο επόμενος τύπος μνήμης είναι SRAM_C η οποία έχει 1024Kbytes μέγεθος, βρίσκεται εξωτερικά από το chip και επικοινωνεί με τον επεξεργαστή μέσω κάποιων processor buses που είναι σχεδιασμένα πάνω στο board. Η μνήμη αυτή είναι η αμέσως επόμενη πιο γρηγορότερη μνήμη του συστήματος με μέγεθος ικανοποιητικό για υλοποίηση μεσαίου τύπου εφαρμογές και σε καλό επίπεδο απόδοσης επεξεργαστή-μνήμης.

Η τελευταία μνήμη που διαθέτει το board είναι η DDR_SDRAM η οποία έχει μέγεθος 65536Kbytes(65MB), βρίσκεται έξω από το chip και επικοινωνεί με τον επεξεργαστή μέσω processor buses τα οποία έχουν σχεδιαστεί πάνω στο board.

Ο τύπος μνήμης αυτός είναι ο πιο αργός σε σχέση με τους άλλους δυο όμως έχει μέγεθος αρκετό το οποίο μπορεί να ικανοποιήσει θεωρητικά για οποιαδήποτε εφαρμογή θέλουμε. Η DDR_SDRAM είναι ο τύπος μνήμης που χρησιμοποιείται σε πιο ευρύ φάσμα σε σχέση με τους άλλους δυο αφού μπορεί να συνδυαστεί και με CACHE μνήμη η οποία μπορεί πετύχει πολύ καλό ποσοστό απόδοσης και επιπλέον το μέγεθος της μπορεί να δώσει στον σχεδιαστή κάποιας εφαρμογής τη δυνατότητα να υλοποιήσει ένα αρχικό μοντέλο μεγάλου μεγέθους να το ελέγξει για τυχόν σχεδιαστικά λάθη και στη συνεχεία να το βελτιστοποιήσει.

Στο τελευταίο σημείο που πρέπει να δώσουμε έμφαση είναι πως μπορούμε να διαχειριστούμε την CACHE μνήμη. Την CACHE μνήμη δεν μπορούμε να την χρησιμοποιήσουμε άμεσα σε μια εφαρμογή όπως ξέρουμε, αλλά μπορούμε να πετύχουμε συνδυάζοντας την με άλλους τύπους μνήμης εντυπωσιακά αποτελέσματα. Στην CACHE μνήμη υπάρχουν αρκετοί περιορισμοί όπως είναι στον τύπο επεξεργαστή που μπορούμε να χρησιμοποιήσουμε αφού στον PowerPC δεν μπορούμε να την χρησιμοποιήσουμε άμεσα σε συνδυασμό με τον επεξεργαστή αλλά μέσω της DDR. Στον MicroBlaze η CACHE μνήμη δεν μας έχει πολλούς περιορισμούς και έτσι αν θέλουμε ένα σύστημα καλό με CACHE μνήμη θα πρέπει να χρησιμοποιήσουμε MicroBlaze επεξεργαστή.

Γενικά το κεφαλαίο μνήμης πρέπει να το προσέχουμε σε κάθε περίπτωση αφού στην ουσία όλα τα δεδομένα περνούν από τουλάχιστον ένα τύπο μνήμης τυχαίας προσπέλασης. Στο πειραματικό επίπεδο της δικής μας εφαρμογής όλες οι μετρήσεις θα παρθούν από τον συνδυασμό του επεξεργαστή με την μνήμη. Στο κεφαλαίο αυτό θα επανέλθουμε πιο αναλυτικά σε επόμενο κομμάτι της πειραματικής διαδικασίας με περισσότερες λεπτομέρειες και εκτιμήσεις.

4.4 Μετρητές

Στις διάφορες δόκιμες που θα κάνουμε με τα διάφορα χαρακτηριστικά για να καταγράψουμε το ποσοστό της απόδοσης ενός συστήματος θα πρέπει να θέσουμε ένα μέτρο σύγκρισης για να μπορέσουμε να δούμε τις διάφορες που έχουν μεταξύ τους. Ο πιο εύκολος τρόπος να συγκρίνουμε τα διάφορα συστήματα είναι να προσθέσουμε timer στην εφαρμογή μας ούτως ώστε να μετρούμε το χρόνο που κάνει ένα σύστημα από την ώρα που θα ξεκινήσει μέχρι την ώρα που θα τελειώσει και να μας τυπώνει το αποτέλεσμα στην έξοδο. Έπειτα θα περνούμε αυτό το χρόνο και θα τον συγκρίνουμε με την αντίστοιχη εφαρμογή αλλά σε διαφορετικό σύστημα.

Στο board που θα χρησιμοποιήσουμε υπάρχουν δυο τρόποι να προσθέσουμε timer ο πρώτος τρόπος είναι ο hardware timer ο οποίος βρίσκεται πάνω στο board και επικοινωνεί με τον επεξεργαστή με buses και ο δεύτερος τρόπος είναι ο software timer ο οποίος γράφεται μέσα στον κώδικα της εφαρμογής γίνεται compile και τρέχει σαν πρόγραμμα της εφαρμογής. Ο software timer δηλώνεται μέσα στην main του κύριος προγράμματος και ξεκινά, σταματά και τυπώνει με τις κατάλληλες εντολές όπου εμείς θέλουμε. Ο hardware timer είναι ελάχιστα πιο σωστός από τον software timer για το λόγο ότι δεν περιμένει να φορτωθεί η εφαρμογή στη μνήμη για να ξεκινήσει. Οι δυο αυτοί timer έχουν περιορισμούς ως προς το τύπο επεξεργαστή που χρησιμοποιούμε. Για παράδειγμα ένα σύστημα σχεδιασμένο και παραμετροποιημένο να δουλεύει με PowerPC μπορούμε να χρησιμοποιήσουμε και τους δυο timer ενώ σε ένα σύστημα σχεδιασμένο και παραμετροποιημένο να δουλεύει με MicroBlaze δεν μπορεί να δουλέψει με τον hardware timer και έτσι περνούμε μετρήσεις εφαρμόζοντας τον software timer. Σε κάθε περίπτωση η διάφορα στους δυο είναι πολύ μικρή σχεδόν αμελητέα. Οι δυο timer μετρούν κύκλους ρολογιού δηλαδή Hertz και τα αποτελέσματα τους είναι σε δεκαεξαδική μορφή. Το αποτέλεσμα σε second τελικά βγαίνει από το ότι το board δουλεύει σε συχνότητα 100MHertz έτσι το μετατρέπουμε σε δεκαδικό και το διαιρούμε με 1000000000 για να βρούμε το αποτέλεσμα.

4.5 Πειραματική διαδικασία

Όπως προαναφέραμε οι επιδόσεις θα υπολογιστούν από διαφορετικά συστήματα τα οποία σε κάθε περίπτωση θα τρέχουν την ίδια εφαρμογή αλλά με διαφορετικό συνδυασμό επεξεργαστή-μνήμης. Οι δυο εφαρμογές μας είναι ο canny και gradient οι οποίοι ασχολούνται κυρίως με φιλτράρισμα κάποιων εικόνων και θα τους χρησιμοποιήσουμε και τους δυο στα ίδια συστήματα για να μετρήσουμε τις επιδώσεις επεξεργαστή μνήμης αλλά και την απόδοση του συστήματος με τις δυο αυτές εφαρμογές. Παρακάτω θα αναφέρουμε αναλυτικά τα έξι διαφορετικά συστήματα που θα χρησιμοποιήσουμε με τις δυο αυτές εφαρμογές:

- Σύστημα βασισμένο σε PowerPC με μνήμη SRAM, hardware/software timer και θα τρέξει τις εφαρμογές gradient/canny.
- Σύστημα βασισμένο σε PowerPC με μνήμη DDR, hardware/software timer και θα τρέξει τις εφαρμογές gradient/canny.
- Σύστημα βασισμένο σε PowerPC με μνήμη DDR και CACHE, hardware/software timer και θα τρέξει τις εφαρμογές gradient/canny.

- Σύστημα βασισμένο σε MicroBlaze με μνήμη SRAM, software timer και θα τρέξει τις εφαρμογές gradient/canny.
- Σύστημα βασισμένο σε MicroBlaze με μνήμη DDR, software timer και θα τρέξει τις εφαρμογές gradient/canny.
- Σύστημα βασισμένο σε MicroBlaze με μνήμη DDR και CACHE, software timer και θα τρέξει τις εφαρμογές gradient/canny.

Η πιο πάνω διαδικασία θα γίνετε σε ξεχωριστά project τα οποία όταν σχεδιαστούν και παραμετροποιηθούν τότε με την βοήθεια μιας εντολής θα δημιουργείται το τελικό αρχείο system.ace το οποίο θα τρέξει την εφαρμογή μας.

Η εντολή αυτή είναι:

xmd –tcl genace.tcl –jprog –board ml403 –target ppc_hw -hw implementation/download.bit –elf TestApp_Memory/executable.elf –ace system.ace –start_address 0x00000000

(η κύρια εντολή για την δημιουργία του system.ace είναι αυτή, όμως σε μερικές περιπτώσεις μπορεί να αλλάξει ανάλογα με τον επεξεργαστή ή την μνήμη που χρησιμοποιούμε).

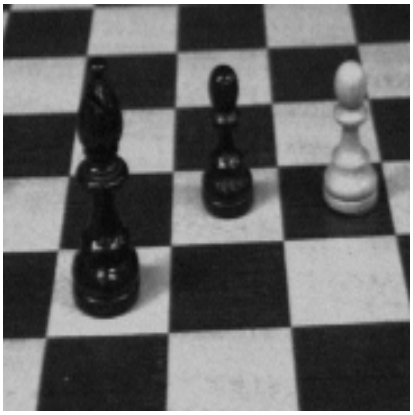
Οι μετρήσεις όλες έγιναν σε εργαστηριακό επίπεδο με εικόνα μεγέθους 90KB και μέγεθος πίνακα 181x146.Τα αποτελέσματα βρίσκονται στο πιο κάτω πίνακα με τις αντίστοιχες εφαρμογές και τα αντίστοιχα συστήματα. Τα αποτελέσματα όλα είναι σε δευτερόλεπτα.

Πίνακας 1: Αποτελέσματα μετρήσεων

		PowerPC	MicroBlaze
Gradient	SRAM	0,47s	2,24s
	DDR	0,56s	3,38s
	DDR + CACHE	0,04s	0,16s
Canny	SRAM	5,31s	14,6s
	DDR	6,44s	26,8s
	DDR + CACHE	0,57s	3,29s

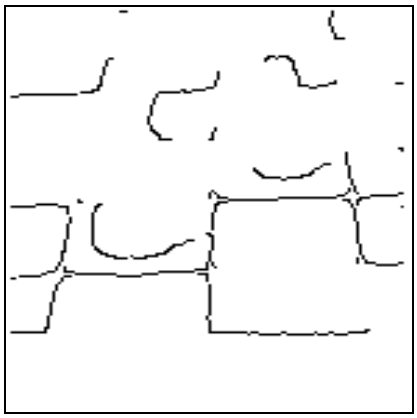
.

Τα αποτελέσματα πριν και μετά το φιλτράρισμα της εικόνας:



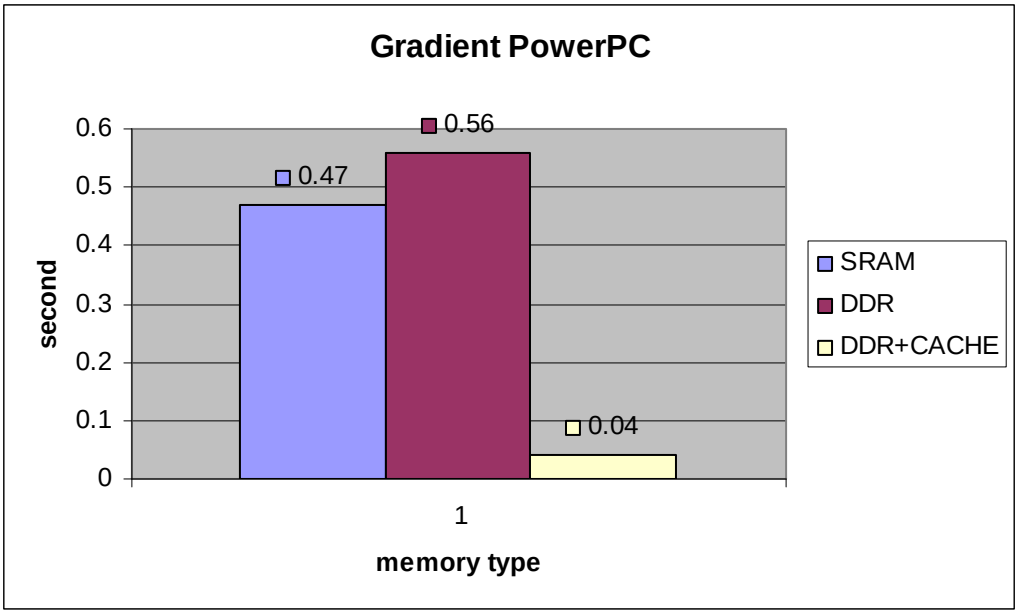
Εικόνα πριν από το φιλτράρισμα

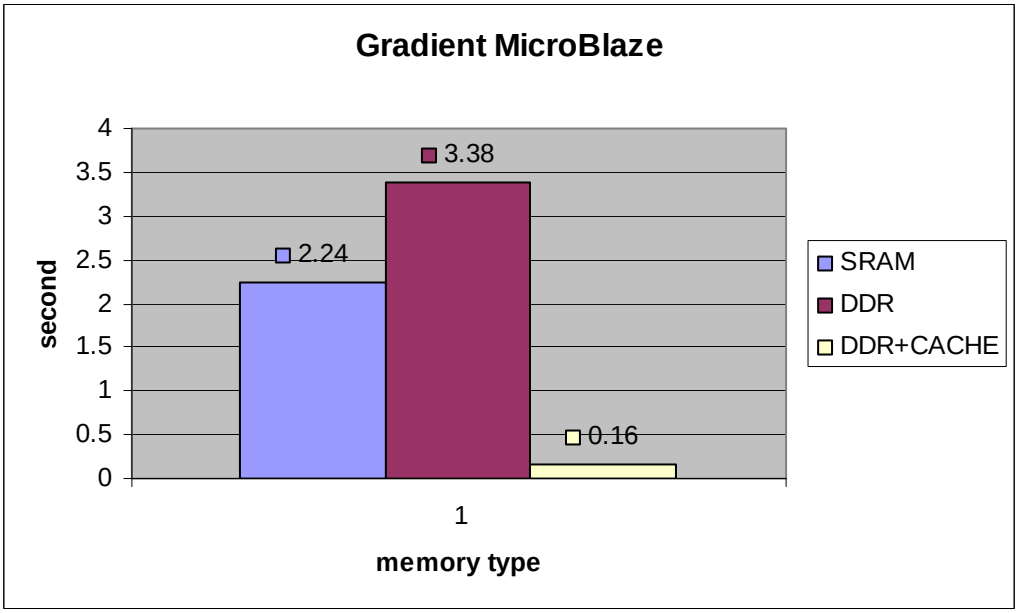
Gradient application



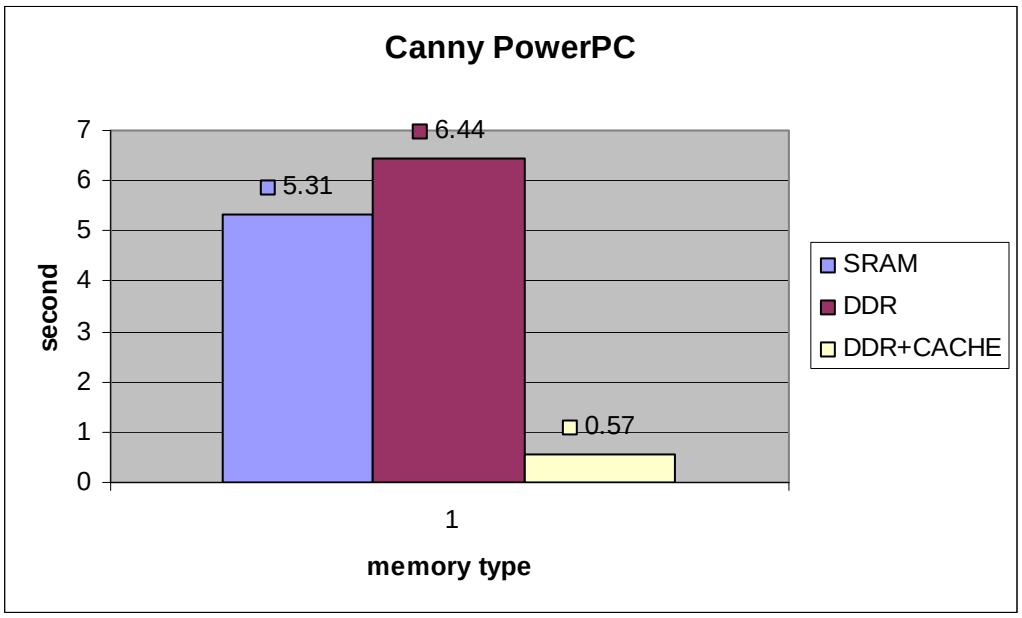
Canny application

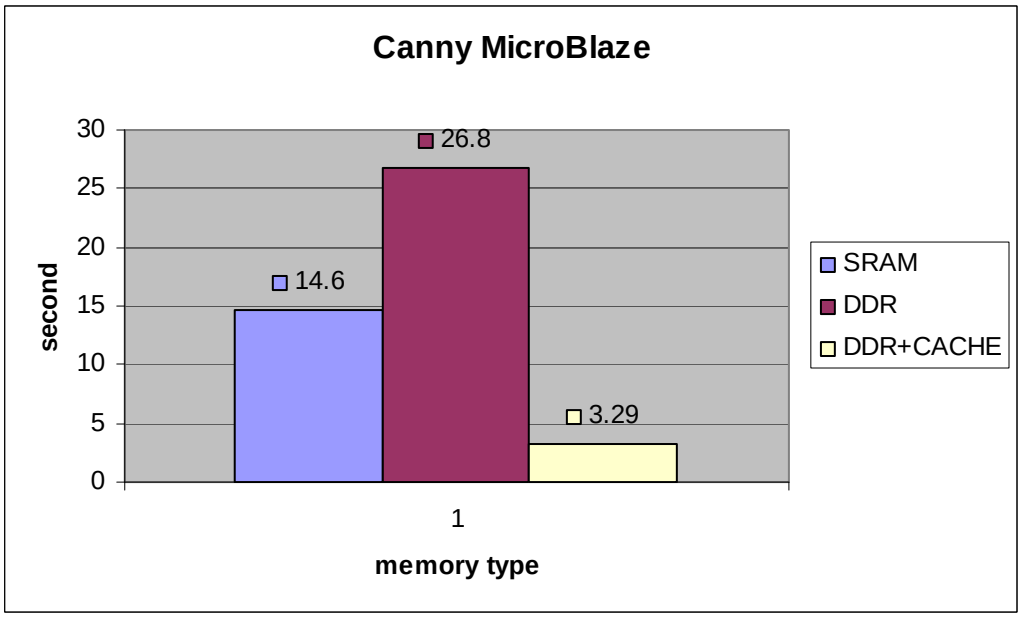
Γραφικές παραστάσεις της Gradient εφαρμογής με PowerPC και MicroBlaze συνάρτηση του τύπου μνήμης.





Γραφικές παραστάσεις της Canny εφαρμογής με PowerPC και MicroBlaze συνάρτηση του τύπου μνήμης.





4.6 Ανάλυση αποτελεσμάτων

Ας εξετάσουμε τα αποτελέσματα της πειραματικής διαδικασίας για να καταλήξουμε στο ποιο από τα συστήματα που έχουμε υλοποιήσει είναι πιο αποδοτικά σωστό, σε τί υστερεί και σε τί υπερτερεί έναντι των άλλων. Η πειραματική διαδικασία χωρίζεται σε δυο μέρη, την εφαρμογή gradient και την εφαρμογή canny. Έχουμε σχεδιάσει έξι συστήματα τα οποία αναφέραμε σε προηγούμενο στάδιο και θα τα εφαρμόσουμε όλα και στις δυο εφαρμογές. Οι δυο εφαρμογές διαφέρουν μεταξύ τους έτσι δεν μπορούμε να τις συγκρίνουμε άμεσα, για το λόγο ότι η gradient εφαρμογή τρέχει μόνο μια κύρια συνάρτηση για φιλτράρισμα, ενώ η canny εφαρμογή τρέχει δυο συναρτήσεις από δυο φορές την κάθε μια έτσι κατά περίπου αναλογία η gradient είναι το ένα τέταρτο της canny. Για τον πιο πάνω λόγο οι συγκρίσεις μας θα γίνουν σε σχέση επεξεργαστή-μνήμης και όχι τόσο σε επίπεδο εφαρμογών.

Όπως έχουμε αναφέρει και πιο πάνω η κυριότερη διαδικασία είναι ο συνδυασμός επεξεργαστή-μνήμης. Οι τρόποι συνδυασμού επεξεργαστή- μνήμης είναι τρεις και τους εφαρμόζουμε σε κάθε επεξεργαστή σε κάθε εφαρμογή. Στην πρώτη εφαρμογή τρεις τρόποι συνδυασμού μνήμης με τον PowerPC μας έδειξε τα εξής αποτελέσματα, με την SRAM έτρεξε την εφαρμογή σε 0,47 δευτερόλεπτα με την DDR σε 0,56 δευτερόλεπτα και ο συνδυασμός PowerPC DDR+CACHE σε 0,04 δευτερόλεπτα. Από το πρώτο στάδιο της διαδικασίας βλέπουμε την τεραστία διάφορα στην μείωση χρόνου προσθέτοντας CACHE μνήμη στο σύστημα. Με την ίδια εφαρμογή ο συνδυασμός των τριών τύπων μνήμης με τον MicroBlaze μας έδειξε τα εξής αποτελέσματα SRAM 2,24 δευτερόλεπτα, DDR 3,38 δευτερόλεπτα και DDR+CACHE 0,16 δευτερόλεπτα. Τα αποτελέσματα του συνδυασμού MicroBlaze και τριών τύπων μνήμης δείχνει την διάφορα στο επιδώσεις PowerPC MicroBlaze και ακόμα μια φορά το πόσο σημαντικό είναι σε ένα σύστημα να υπάρχει CACHE μνήμη.

Την ίδια διαδικασία θα ακολουθήσουμε και με την δεύτερη εφαρμογή για να ξεκαθαρίσουμε τον συνδυασμό στον οποίο έχουμε την μεγαλύτερη απόδοση του συστήματος σε χρόνο. Η canny εφαρμογή είναι πιο πολύπλοκη σε θέματα συναρτήσεων που χρησιμοποιεί και είναι αναμενόμενο να πάρει περισσότερο χρόνο για να τρέξει στο σύστημα. Ας δούμε λοιπόν πως έχουν τα αποτελέσματα σε αυτή την διαδικασία. Ο συνδυασμός PowerPC και των τριών τύπων μνήμης έδειξε τα αποτελέσματα SRAM 5,31 δευτερόλεπτα , DDR 6,44 δευτερόλεπτα και DDR+CACHE 0,57 δευτερόλεπτα. Ακόμα μια φορά βλέπουμε την συμπεριφορά του

συστήματος με τον συνδυασμό μνήμης DDR+CACHE να έχει τεράστια διάφορα απόδοσης από τις άλλους δυο, ανεξάρτητος επεξεργαστή και εφαρμογής. Τα αποτελέσματα του συνδυασμού MicroBlaze και των τριών τύπων μνήμης ήταν SRAM 14,6 δευτερόλεπτα , DDR 26,8 δευτερόλεπτα και DDR+CACHE 3,29 δευτερόλεπτα. Ο συνδυασμός αυτός επαλήθευσε μια ακόμα φορά τις πιο πάνω εκτιμήσεις μας και ξεκαθάρισε το τοπίο στο ποίος συνδυασμός είναι πιο αποδοτικός.

Στην πειραματική μας διαδικασία εφαρμόσαμε σε κάθε περίπτωση δυο επεξεργαστές και τρεις συνδυασμούς μνήμης. Το πρώτο που θέλαμε να δούμε στην πειραματική διαδικασία είναι ποίος από τους δυο επεξεργαστές είναι πιο αποδοτικός και δεύτερο τον συνδυασμό μνήμης που βελτιστοποιεί τις επιδώσεις του συστήματος. Ο επεξεργαστής που είχε την καλύτερη απόδοση σε όλα τα σταδία είναι ο PowerPC με σημαντική διάφορα σε επίπεδο εφαρμογών αλλά και σε επίπεδο συστήματος. Αυτό που όμως έπαιξε καθοριστικό ρόλο στην απόδοση συστήματος- εφαρμογών είναι ο συνδυασμός μνήμης DDR+CACHE. Ο συνδυασμός αυτός μείωσε το χρόνο της όλης διαδικασίας σημαντικά και έτσι δεν άφησε καμία αμφιβολία στο ποιος είναι ο ιδανικός τρόπος να δημιουργήσουμε ένα σύστημα που να έχει την καλύτερη απόδοση συνδυασμού συστήματος-εφαρμογών.

Οι πιο πάνω πειραματική διαδικασία μας έδωσε να καταλάβουμε ποια είναι τα κύρια χαρακτηριστικά που καθορίζουν την συμπεριφορά ενός συστήματος και πως μπορούμε να βελτιστοποιήσουμε ένα σύστημα. Αν πρέπει να πούμε ποιο ήταν το σύστημα με την μεγαλύτερη απόδοση σίγουρα ο συνδυασμός PowerPC με DDR+CACHE όμως η πιο σωστή εκτίμηση είναι εκτός από την καλή αρχιτεκτονική σχεδίαση ενός επεξεργαστή που εννοείται παίζει σημαντικό ρόλο στην απόδοση ενός συστήματος, ο συνδυασμός μνήμης με CACHE πετυχαίνει τελικά να προσφέρει πιο ψηλά επίπεδα απόδοσης σε ένα σύστημα.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Από όλη την πειραματική διαδικασία μας δόθηκε η δυνατότητα να καταλάβουμε τις απεριόριστες δυνατότητες που προσφέρει ο τομέας αυτός, στην υλοποίηση ενός ενσωματωμένου συστήματος. Το μεγαλύτερο μέρος της παραγωγής ηλεκτρονικών πλακετών και διάφορων τύπων επεξεργαστών χρησιμοποιούν τα board αυτά σε συνδυασμό με το XPS για να υλοποιήσουν, να δοκιμάσουν και να στείλουν τελικά στην παραγωγή τα μοντέλα τους. Όλοι εμείς που ασχολούμαστε με την επιστήμη της Πληροφορικής πρέπει να δούμε το θέμα πιο σοβαρά αφού υπάρχει μέλλον και ανάπτυξη στο τομέα αυτό. Επιπλέον δεν πρέπει να παραλείψουμε το γεγονός ότι αν κάποιος ασχοληθεί με τον τομέα αυτό τότε του δίνεται η δυνατότητα να καταλάβει ακριβώς πώς δουλεύει ένας υπολογιστής, πώς σχεδιάζεται ένα σύστημα και τί απόδοση μπορεί να έχει με τα χαρακτηριστικά που διαθέτει, από κινητά τηλεφωνά, υπολογιστές και γενικά ότι έχει σχέση με ολοκληρωμένα κυκλώματα.

Επίσης , μας δίνετε η δυνατότητα να κατανοήσουμε λειτουργίες των υπολογιστών που αν και τις γνωρίζαμε σε θεωρητικό επίπεδο δεν μπορούσαμε να καταλάβουμε τί ρόλο παίζουν σε ένα σύστημα και πόσο μπορούν να προσφέρουν στη απόδοση του. Μέσα από την διαδικασία αυτή μπορέσαμε να καταλάβουμε κάποια επιπλέον χαρακτηριστικά που παρέχουν οι εταιρίες παραγωγής συστημάτων με αποτέλεσμα να μπορούμε να εκτιμήσουμε ένα σύστημα πόσο μπορεί να αποδώσει στις ανάγκες που το χρειαζόμαστε..

Όσο αφορά την δική μας πειραματική διαδικασία μέσα από τις μετρήσεις που πήραμε μπορέσαμε να κατανοήσουμε πως κάποιες λεπτομέρειες μπορούν να παίξουν σημαντικό ρόλο στην απόδοση ενός συστήματος και στη συμπεριφορά ενός επεξεργαστή. Είδαμε τελικά πως μια πολύ μικρού μεγέθους μνήμη όπως είναι η CACHE μπορεί να απογειώσει τις δυνατότητες ενός συστήματος. Αν σκεφτούμε λοιπόν την όλη διαδικασία, καταλήγουμε στο συμπέρασμα ότι ένα σύστημα μπορεί να γίνει καλύτερο προσθέτοντας τους κάποια μικρά χαρακτηριστικά που όμως παίζουν μεγάλο ρόλο στα επίπεδα απόδοσης του.

ΠΡΟΒΛΗΜΑΤΑ ΚΑΙ ΛΥΣΕΙΣ

Στο πειραματικό στάδιο της όλης διαδικασίας συναντήσαμε κάποια προβλήματα που όπως φάνηκε παίζουν σημαντικό ρόλο στη σωστή λειτουργία κάποιου συστήματος και μίας εφαρμογής. Τα προβλήματα αυτά θα τα αναφέρουμε πιο κάτω:

- Ένα σημαντικό πρόβλημα που συναντήσαμε αρχικά σε επίπεδο εφαρμογής είναι κάποιες συναρτήσεις κώδικα C έπρεπε να μετατραπούν σε αντίστοιχες συναρτήσεις του compiler που υποστηρίζει το XPS, για να τρέξει η εφαρμογή χωρίς προβλήματα. Οι συναρτήσεις αυτές βρίσκονται σε κάποιες βιβλιοθήκες για παράδειγμα #include< xparameters.h>, #include<xutil.h> και #include<xtime.h>.
- Στο επόμενο σημείο που βρήκαμε δυσκολία είναι στο ότι καθώς το bitstream προσπαθούσε να φορτώσει την εφαρμογή λόγο των μεγέθους του κώδικα και των εφαρμογών υπερχείλιζαν, οι ουρές και οι στοίβες αφού αρχική τιμή είναι μόλις στα 256byte. Για να αντιμετωπίσουμε το προβλημα αυτό αυξήσαμε την τιμή του Heap και Stack σε A0000 και 1000 αντίστοιχα.
- Το τελευταίο προβλημα που αξίζει να αναφέρουμε είναι σε επίπεδο μετρητή, αφού ο hardware timer έχει μέγεθος 32bit έτσι κυρίως στη δεύτερη εφαρμογή που ήταν πιο πολύπλοκη σε θέμα συναρτήσεων, ο μετρητής γέμιζε και μηδενιζόταν. Για το λόγο αυτό δεν μπορούσαμε να πάρουμε μετρήσεις. Η λύση σε αυτό το πρόβλημα ήταν στην ίδια εφαρμογή να κόψουμε την εικόνα στη μέση, να πάρουμε μετρήσεις για την μισή εικόνα και στη συνεχεία τα αποτελέσματα να τα πολλαπλασιάσουμε με το δυο για να δούμε τελικά πόσο χρόνο θα έκανε με την ολόκληρη εικόνα.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Xilinx Documentation
   http://www.xilinx.com/publications/index.htm
2. History of Programmable Gates
   http://en.wikipedia.org/wiki/Field-programmable_gate_array
3. PowerPC Datasheet
   http://www.xilinx.com/support/documentation/ip_documentation/ppc440mc_ddr2.pdf
4. MicroBlaze Datasheet
   http://www.xilinx.com/support/documentation/sw_manuals/mb_ref_guide.pdf
5. ML403 Evaluation Platform
   http://www.xilinx.com/support/documentation/boards_and_kits/ug080.pdf
6. Pgm Image Format
   http://en.wikipedia.org/wiki/Netpbm_format
7. Xilinx Platform Studio
   http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/edk_ctt.pdf
8. FPGA Architecture
   http://klabs.org/richcontent/Tutorial/MiniCourses/architecture_logic_mapld2001/Architecture_Section/04_FPGA_Architecture.PDF
9. Embedded system
   http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/est_rm.pdf
   http://en.wikipedia.org/wiki/Embedded_system
10. Canny, J., A Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986

ΠΑΡΑΡΤΗΜΑ 1

Η εφαρμογή αυτή ασχολείται με την αλλαγή των αντιθέσεων των χρωμάτων σε οριζόντια επίπεδα και αφού αναγνωρίζει μέσα από τις συναρτήσεις, τις αντιθέσεις θα δημιουργεί ένα δεύτερο πίνακα ο οποίος θα τυπώσει με αχνές γραμμές μόνο τις εναλλαγές σε οριζόντιο επίπεδο. Ο πίνακας αυτός θα στέλνεται μέσω συριακής θύρας σε ένα υπολογιστή ο οποίος θα καταγράφει τα δεδομένα σε ένα αρχείο .txt το οποίο θα το ανοίξουμε στη συνεχεία με ένα image editor για να δούμε την φιλτραρισμένη εικόνα(πχ σε ένα σκάκι που προσπαθεί να βρει τις οριζόντιες γραμμές ανάμεσα στα τετράγωνα και να τις βάζει στο νέο πίνακα που δημιουργεί).

```
/* Gradient edge detector - no templates */
/* G1 */

#include <stdio.h>
#include <stdlib.h>
#include <stdlib.h>
#include <malloc.h>
#include <string.h>
#include <math.h>

#include "xparameters.h"
#include "xtmrctr.h"
#include "xtime_l.h"

#define MAX 255
#define ROWS 181
#define COLUMNS 35
#define CLASS 5
#define PI 3.14


unsigned char imgdata[181][35] = {
{   30,   32,   40,   31,   30,   30,   39,   31,   29,   32,   28,
34,   26,   30,   31,   32,   28,   27,   34,   38,   96,  174,  192,
194,  194,  195,  191,  186,  184,  191,  191,  193,  190,  183,
171,  147,  120,   89,   58,   40,   30,   26,   26,   27,   28,
22,   24,   19,   26,   25,   33,   48,   76,  109,  146,  178,  184,
188,  195,  196,  194,  196,  199,  193,  193,  190,  188,  188,
190,  193,  196,  196,  194,  150,   71,   43,   39,   34,   33,
36,   31,   32,   34,   37,   33,   35,   31,   40,   31,   32,   32,
35,   34,   28,   34,   29,   37,   31,   33,   29,   32,   36,   38,
31,   35,   29,   35,   30,   35,   33,   39,   36,   30,   27,   36,
36,   37,   37,   35,   40,   43,   36,   30,   29,   34,   31,   39,
91,  174,  184,  191,  197,  192,  193,  195,  188,  195,  192,  186,
191,  193,  192,  188,  191,  192,  194,  192,  195,  189,  184,
194,  196,  192,  191,  195,  190,  191,  198,  195,  193,  189,
193,  195,  187,  183,  190,  190,  189,  190,  187,  186,  189,
190,  192,  186,  188,  187,  193,  186,  182,  180,},
{   31,   25,   29,   30,   32,   29,   27,   34,   31,   26,   30,
30,   30,   34,   32,   29,   33,   31,   35,   54,  120,  183,  189,
197,  195,  194,  195,  192,  197,  199,  191,  188,  186,  189,
192,  191,  191,  183,  164,  133,  100,   78,   63,   55,   49,
48,   52,   58,   74,  105,  140,  170,  183,  187,  196,  193,  192,
194,  193,  194,  193,  198,  198,  193,  200,  195,  195,  194,
```

```
194,  192,  195,  197,  196,  144,   72,   43,   37,   39,   34,
34,   37,   37,   38,   35,   38,   38,   38,   34,   34,   38,   32,
38,   34,   35,   36,   30,   32,   36,   38,   44,   36,   34,   41,
35,   34,   33,   30,   31,   37,   36,   37,   34,   31,   40,   38,
37,   40,   41,   39,   41,   36,   41,   38,   37,   38,   38,   34,
86,  168,  187,  192,  194,  194,  193,  192,  191,  189,  190,  187,
193,  191,  190,  192,  196,  189,  193,  189,  196,  192,  185,
195,  195,  191,  192,  188,  190,  193,  187,  189,  192,  182,
192,  190,  190,  187,  187,  185,  183,  190,  192,  188,  192,
192,  191,  186,  183,  189,  183,  188,  181,  179,},
{   34,   31,   36,   27,   36,   31,   31,   36,   31,   33,   35,
33,   28,   31,   32,   37,   31,   35,   36,   50,  139,  185,  189,
191,  194,  196,  194,  196,  192,  192,  193,  194,  189,  191,
191,  192,  188,  195,  190,  192,  183,  181,  181,  178,  177,
180,  185,  189,  192,  193,  189,  197,  193,  196,  186,  192,
197,  190,  197,  200,  194,  192,  191,  193,  195,  193,  192,
192,  200,  194,  190,  190,  192,  146,   68,   39,   35,   30,
38,   33,   35,   33,   31,   38,   38,   34,   31,   27,   34,   32,
32,   38,   28,   33,   33,   29,   29,   30,   25,   31,   32,   33,
33,   37,   41,   39,   37,   28,   34,   34,   34,   32,   33,   35,
35,   35,   34,   41,   37,   34,   32,   36,   35,   37,   31,   37,
37,   80,  167,  187,  190,  190,  186,  188,  188,  181,  189,  185,
193,  187,  192,  187,  190,  193,  191,  190,  196,  192,  194,
193,  198,  192,  194,  191,  191,  193,  187,  182,  184,  190,
187,  184,  190,  183,  182,  183,  173,  174,  174,  167,  152,
159,  153,  163,  158,  151,  154,  145,  142,  139,  134,},
{   30,   35,   34,   36,   39,   33,   39,   39,   34,   34,   32,
30,   32,   33,   31,   32,   30,   30,   36,   59,  153,  183,  191,
193,  185,  191,  192,  193,  190,  190,  191,  189,  193,  193,
193,  193,  192,  196,  192,  196,  196,  193,  197,  191,  192,
195,  191,  191,  194,  193,  190,  191,  192,  195,  199,  199,
190,  192,  196,  194,  198,  193,  195,  193,  195,  194,  194,
194,  192,  195,  193,  191,  192,  141,   61,   37,   34,   30,
33,   33,   35,   34,   37,   40,   36,   36,   35,   26,   36,   31,
37,   34,   36,   36,   30,   35,   33,   31,   36,   34,   30,   32,
42,   35,   32,   31,   28,   36,   27,   32,   35,   36,   38,   37,
42,   40,   34,   32,   33,   35,   31,   33,   42,   37,   33,   31,
38,   65,  160,  191,  188,  185,  192,  194,  190,  195,  195,  187,
187,  194,  187,  182,  186,  178,  180,  185,  181,  181,  176,
178,  182,  168,  163,  157,  155,  149,  138,  134,  127,  114,
110,  106,  102,   99,   95,   91,   78,   73,   67,   65,   58,
54,   54,   52,   56,   51,   49,   39,   52,   44,   46,},
{   38,   37,   33,   35,   35,   27,   28,   34,   37,   40,   34,
32,   35,   32,   40,   32,   34,   34,   36,   76,  166,  188,  191,
192,  193,  195,  191,  193,  195,  192,  188,  191,  190,  201,
192,  194,  196,  192,  197,  198,  192,  191,  196,  191,  196,
193,  190,  195,  189,  196,  191,  197,  195,  191,  190,  194,
192,  198,  194,  188,  191,  190,  196,  189,  191,  190,  197,
193,  194,  195,  192,  192,  190,  136,   63,   34,   31,   31,
31,   34,   32,   32,   35,   32,   32,   31,   36,   37,   34,   24,
25,   28,   28,   33,   30,   27,   27,   30,   35,   31,   29,   33,
29,   32,   28,   26,   30,   37,   34,   31,   37,   41,   32,   33,
31,   29,   30,   38,   30,   38,   35,   36,   45,   42,   42,   42,
45,   56,  129,  145,  143,  136,  127,  128,  118,  107,  106,   95,
92,   94,   87,   79,   75,   66,   68,   63,   56,   64,   58,   50,
49,   49,   50,   46,   46,   47,   48,   41,   50,   42,   44,   44,
42,   46,   43,   44,   47,   49,   43,   41,   39,   38,   42,   34,
40,   39,   37,   34,   36,   31,   36,},
{   32,   31,   30,   31,   41,   29,   36,   35,   38,   33,   36,
33,   35,   34,   34,   39,   34,   39,   46,   99,  174,  187,  195,
193,  192,  193,  190,  194,  197,  197,  192,  196,  192,  189,
```

```
190,  190,  195,  197,  193,  195,  196,  196,  195,  194,  191,
194,  198,  192,  199,  194,  195,  195,  193,  197,  198,  195,
193,  195,  199,  199,  193,  193,  194,  197,  193,  191,  198,
190,  197,  194,  196,  190,  194,  129,   60,   38,   37,   29,
29,   32,   33,   38,   35,   36,   34,   34,   35,   32,   34,   37,
36,   37,   35,   38,   41,   38,   42,   41,   41,   41,   39,   47,
48,   53,   59,   47,   61,   68,   74,   74,   71,   65,   84,   96,
91,   94,   93,   94,  111,  116,  124,  127,  132,  146,  155,  156,
155,  151,   93,   60,   46,   45,   49,   43,   45,   48,   49,
49,   41,   38,   42,   44,   40,   46,   38,   43,   42,   39,   38,
38,   39,   42,   41,   38,   37,   42,   37,   34,   42,   44,   37,
36,   36,   33,   37,   38,   35,   36,   32,   33,   34,   41,   36,
33,   33,   30,   32,   22,   28,   34,   31,},
{   38,   32,   36,   34,   33,   30,   33,   31,   29,   32,   32,
31,   40,   33,   36,   32,   32,   33,   40,  111,  179,  196,  194,
194,  192,  197,  194,  193,  194,  194,  190,  201,  193,  192,
190,  199,  198,  191,  196,  193,  196,  193,  192,  197,  195,
195,  193,  194,  194,  194,  194,  193,  193,  196,  192,  196,
196,  200,  195,  200,  193,  189,  192,  187,  193,  192,  190,
186,  191,  193,  187,  190,  186,  128,   64,   55,   58,   60,
63,   67,   75,   74,   80,   88,   92,   97,   98,  103,  116,  120,
115,  122,  125,  137,  135,  138,  139,  151,  151,  149,  153,
162,  167,  165,  167,  171,  168,  166,  183,  184,  163,  162,
182,  182,  174,  175,  171,  169,  169,  176,  177,  181,  186,
181,  187,  189,  185,  166,   93,   54,   41,   37,   37,   31,
36,   40,   36,   34,   37,   30,   38,   35,   34,   46,   42,   44,
42,   45,   46,   42,   40,   36,   35,   37,   37,   34,   41,   38,
33,   35,   38,   31,   39,   34,   36,   33,   35,   32,   30,   33,
35,   30,   28,   30,   26,   28,   26,   28,   31,   34,   25,},
{   34,   37,   34,   38,   36,   29,   28,   34,   28,   36,   40,
30,   34,   37,   34,   33,   40,   39,   55,  130,  183,  192,  198,
189,  192,  192,  195,  200,  191,  195,  195,  194,  196,  193,
184,  188,  187,  193,  192,  196,  185,  186,  180,  182,  190,
183,  182,  175,  177,  172,  172,  167,  161,  152,  147,  146,
152,  143,  129,  122,  115,  107,  109,  106,  105,   97,   88,
83,   76,   82,   82,   84,   82,  105,  149,  157,  151,  161,  172,
163,  156,  162,  159,  173,  172,  170,  166,  165,  171,  170,
176,  175,  165,  164,  166,  168,  171,  179,  173,  179,  176,
179,  185,  182,  184,  177,  177,  176,  188,  187,  177,  170,
180,  184,  180,  178,  174,  174,  174,  184,  185,  182,  182,
186,  189,  183,  179,  170,  101,   60,   46,   43,   35,   31,
29,   33,   31,   39,   36,   36,   33,   29,   33,   36,   37,   39,
39,   34,   36,   41,   36,   39,   41,   33,   34,   34,   37,   40,
32,   35,   36,   32,   33,   32,   25,   26,   35,   30,   31,   40,
33,   30,   33,   30,   32,   30,   28,   26,   31,   37,   30,},
{   40,   38,   40,   38,   37,   40,   41,   39,   47,   42,   42,
49,   54,   50,   50,   57,   47,   51,   70,  108,  121,  122,  121,
111,  106,  105,   94,   91,   88,   84,   93,   90,   88,   84,
74,   71,   70,   73,   69,   66,   68,   61,   59,   58,   57,   56,
54,   55,   51,   49,   53,   51,   53,   46,   47,   52,   44,   48,
43,   39,   42,   49,   42,   46,   45,   44,   42,   40,   43,   47,
48,   45,   42,   95,  168,  178,  172,  182,  181,  178,  173,  178,
182,  186,  178,  175,  173,  175,  172,  171,  174,  177,  179,
168,  173,  178,  169,  165,  182,  181,  185,  183,  176,  181,
181,  175,  179,  178,  180,  185,  178,  172,  180,  185,  182,
185,  177,  174,  177,  176,  179,  187,  185,  191,  190,  184,
180,  174,  115,   56,   40,   37,   33,   29,   37,   29,   43,
32,   40,   37,   33,   31,   43,   42,   37,   33,   33,   33,   32,
34,   36,   36,   31,   33,   33,   36,   33,   35,   37,   39,   35,
28,   36,   27,   29,   30,   32,   32,   32,   34,   31,   25,   30,
26,   29,   29,   35,   29,   32,   31,   36,},
```

```
{  117,  120,  125,  107,  120,  121,  122,  125,  123,  121,  146,
139,  153,  158,  156,  150,  151,  162,  155,   95,   60,   49,
48,   42,   42,   38,   38,   43,   49,   45,   41,   36,   45,   40,
42,   38,   40,   45,   42,   41,   37,   43,   40,   38,   37,   36,
47,   42,   37,   38,   39,   37,   38,   33,   32,   35,   40,   36,
33,   37,   39,   42,   37,   31,   35,   36,   35,   35,   33,   33,
31,   35,   39,  100,  173,  184,  195,  190,  185,  181,  185,  186,
187,  183,  179,  183,  182,  176,  181,  183,  182,  181,  187,
180,  183,  183,  181,  178,  187,  187,  183,  186,  181,  183,
184,  183,  184,  189,  185,  181,  183,  182,  183,  186,  187,
193,  176,  182,  182,  184,  187,  187,  187,  182,  187,  186,
193,  171,  128,   54,   42,   39,   36,   33,   29,   32,   30,
33,   36,   36,   36,   31,   32,   33,   37,   38,   36,   34,   35,
32,   36,   37,   28,   33,   36,   38,   38,   44,   32,   33,   32,
38,   37,   28,   32,   32,   31,   28,   33,   31,   39,   34,   33,
36,   29,   30,   29,   28,   30,   29,   32,},
{  165,  174,  168,  156,  153,  156,  159,  164,  157,  161,  172,
178,  177,  176,  178,  170,  174,  166,  139,   79,   47,   37,
37,   41,   35,   35,   38,   37,   32,   38,   36,   36,   37,   30,
30,   32,   37,   37,   37,   36,   38,   37,   34,   38,   35,   46,
40,   34,   41,   39,   34,   32,   39,   32,   35,   37,   40,   35,
35,   34,   35,   35,   34,   32,   29,   27,   31,   34,   33,   41,
34,   30,   35,  100,  168,  187,  191,  191,  187,  180,  187,  188,
184,  189,  190,  191,  195,  188,  194,  185,  186,  186,  188,
193,  188,  190,  193,  188,  194,  194,  186,  186,  190,  187,
182,  184,  184,  184,  181,  185,  184,  188,  184,  189,  185,
190,  186,  185,  187,  186,  189,  188,  192,  188,  189,  188,
189,  186,  132,   60,   42,   32,   41,   33,   35,   38,   39,
42,   32,   36,   31,   29,   35,   29,   34,   30,   25,   33,   38,
36,   35,   33,   37,   42,   37,   35,   41,   41,   36,   37,   34,
41,   40,   34,   35,   34,   28,   31,   32,   39,   34,   31,   31,
31,   33,   29,   26,   28,   28,   27,   31,},
{  181,  175,  169,  169,  171,  176,  177,  174,  177,  172,  179,
180,  180,  182,  178,  178,  171,  168,  133,   66,   39,   35,
28,   27,   28,   30,   31,   36,   34,   35,   33,   34,   32,   30,
27,   32,   33,   39,   38,   41,   40,   36,   39,   38,   39,   38,
38,   35,   37,   33,   32,   37,   38,   32,   33,   32,   34,   35,
31,   33,   32,   29,   34,   27,   32,   26,   28,   35,   32,   35,
32,   32,   34,  110,  180,  184,  192,  185,  190,  190,  191,  195,
193,  184,  181,  187,  191,  182,  181,  186,  194,  192,  187,
188,  198,  191,  186,  193,  189,  191,  185,  189,  190,  190,
188,  189,  192,  182,  189,  184,  188,  195,  191,  192,  191,
187,  192,  193,  187,  186,  192,  196,  198,  192,  191,  192,
195,  185,  146,   74,   42,   35,   36,   36,   35,   31,   33,
38,   35,   34,   31,   31,   28,   34,   37,   39,   37,   34,   33,
31,   34,   35,   33,   34,   36,   38,   41,   37,   33,   45,   35,
35,   33,   33,   32,   29,   35,   27,   30,   31,   29,   32,   28,
28,   28,   29,   36,   25,   25,   30,   28,},
{  181,  172,  175,  175,  175,  176,  182,  175,  177,  179,  186,
176,  179,  186,  185,  181,  176,  165,  108,   53,   41,   35,
36,   35,   36,   31,   33,   36,   34,   35,   29,   33,   33,   32,
29,   30,   28,   34,   29,   34,   39,   38,   37,   45,   44,   39,
39,   38,   34,   36,   39,   35,   34,   33,   36,   26,   28,   32,
32,   29,   29,   35,   28,   30,   31,   31,   31,   39,   38,   40,
41,   43,   43,  114,  176,  190,  190,  192,  191,  186,  194,  200,
191,  191,  189,  188,  189,  190,  186,  181,  184,  191,  191,
186,  187,  191,  187,  184,  190,  191,  183,  181,  189,  188,
183,  190,  188,  190,  182,  195,  189,  190,  187,  188,  183,
185,  193,  190,  192,  179,  189,  188,  194,  190,  188,  188,
185,  189,  146,   64,   41,   36,   33,   35,   29,   31,   30,
33,   33,   29,   25,   39,   33,   39,   36,   35,   30,   30,   35,
```

```
33,   35,   29,   30,   30,   27,   35,   34,   35,   34,   34,   35,
35,   29,   34,   30,   32,   32,   30,   30,   28,   33,   33,   32,
30,   27,   34,   28,   31,   27,   31,   33,},
{  187,  189,  182,  186,  180,  185,  175,  179,  184,  185,  187,
187,  186,  192,  182,  177,  181,  165,  104,   51,   38,   34,
33,   38,   31,   32,   37,   33,   36,   33,   31,   30,   35,   33,
31,   28,   30,   30,   30,   33,   39,   36,   34,   34,   40,   39,
40,   35,   33,   37,   37,   38,   32,   35,   31,   32,   32,   34,
34,   32,   32,   33,   27,   32,   30,   38,   30,   31,   29,   37,
38,   35,   39,  118,  177,  188,  190,  191,  192,  190,  196,  191,
190,  190,  190,  187,  189,  191,  189,  188,  188,  194,  191,
191,  188,  191,  191,  193,  192,  192,  192,  189,  189,  191,
188,  185,  187,  187,  185,  185,  191,  190,  187,  187,  187,
190,  189,  192,  190,  188,  191,  187,  188,  190,  191,  189,
185,  180,  152,   82,   54,   42,   35,   35,   33,   33,   35,
38,   30,   36,   32,   34,   41,   29,   34,   35,   35,   36,   29,
32,   31,   32,   37,   34,   34,   31,   31,   27,   34,   35,   31,
32,   31,   28,   32,   25,   32,   28,   31,   28,   29,   31,   33,
30,   25,   33,   29,   28,   24,   30,   27,},
{  189,  183,  187,  192,  190,  190,  189,  184,  192,  195,  184,
181,  189,  186,  192,  189,  184,  150,   87,   40,   32,   34,
35,   37,   36,   31,   30,   29,   36,   26,   31,   31,   32,   30,
32,   32,   31,   26,   34,   27,   36,   38,   39,   43,   43,   38,
37,   37,   33,   38,   33,   34,   31,   35,   37,   30,   38,   38,
33,   35,   32,   32,   31,   33,   31,   25,   36,   33,   40,   44,
43,   42,   42,  127,  181,  192,  196,  191,  195,  187,  191,  188,
195,  191,  185,  186,  199,  192,  193,  190,  186,  193,  194,
191,  190,  190,  192,  190,  186,  188,  192,  191,  191,  195,
190,  194,  187,  190,  189,  187,  184,  186,  194,  192,  193,
191,  193,  187,  192,  194,  192,  185,  187,  189,  191,  192,
188,  186,  167,   84,   44,   40,   33,   36,   33,   29,   33,
25,   36,   30,   30,   30,   28,   26,   35,   32,   33,   34,   34,
39,   30,   33,   32,   31,   31,   33,   40,   38,   33,   30,   31,
29,   32,   29,   28,   29,   28,   32,   31,   30,   32,   33,   29,
32,   29,   34,   25,   25,   29,   34,   32,},
{  178,  187,  183,  180,  188,  186,  180,  185,  187,  185,  189,
184,  187,  188,  187,  188,  180,  146,   75,   41,   33,   30,
30,   26,   36,   35,   32,   33,   31,   33,   30,   31,   38,   35,
43,   35,   35,   31,   38,   38,   31,   32,   41,   33,   39,   38,
34,   33,   34,   39,   34,   34,   34,   31,   27,   38,   34,   30,
38,   35,   27,   30,   26,   30,   31,   31,   30,   33,   30,   31,
33,   28,   40,  130,  183,  191,  190,  191,  190,  190,  188,  195,
194,  194,  191,  189,  191,  191,  195,  191,  191,  193,  192,
188,  195,  192,  192,  194,  192,  190,  184,  187,  193,  189,
191,  191,  186,  189,  186,  190,  189,  189,  191,  192,  190,
190,  192,  189,  190,  193,  189,  188,  194,  195,  192,  191,
192,  189,  174,   97,   48,   37,   36,   36,   38,   33,   34,
40,   31,   31,   30,   38,   33,   33,   31,   40,   35,   29,   29,
34,   33,   33,   32,   34,   32,   37,   36,   35,   39,   37,   32,
29,   26,   27,   28,   26,   27,   24,   26,   26,   30,   23,   29,
27,   27,   28,   27,   21,   24,   29,   27,},
{  183,  179,  178,  179,  187,  189,  182,  185,  187,  190,  183,
186,  189,  182,  193,  182,  180,  130,   67,   41,   39,   30,
35,   29,   31,   34,   30,   30,   34,   32,   34,   33,   33,   35,
35,   39,   37,   31,   35,   33,   33,   41,   38,   37,   43,   42,
39,   38,   28,   33,   35,   36,   33,   33,   33,   37,   38,   31,
31,   33,   31,   31,   28,   27,   26,   29,   28,   31,   32,   40,
39,   41,   49,  130,  180,  191,  191,  191,  189,  194,  189,  186,
191,  192,  193,  193,  192,  198,  194,  195,  191,  185,  187,
194,  192,  192,  193,  189,  189,  180,  185,  183,  188,  190,
189,  188,  187,  185,  189,  194,  187,  185,  194,  187,  191,
```

```
194,  189,  195,  188,  186,  192,  187,  195,  193,  192,  193,
197,  188,  177,  112,   48,   34,   32,   31,   31,   28,   32,
40,   38,   25,   33,   30,   34,   34,   30,   31,   29,   32,   29,
29,   32,   30,   33,   30,   34,   37,   29,   32,   29,   34,   34,
32,   25,   26,   22,   30,   30,   26,   27,   29,   31,   28,   26,
29,   25,   31,   32,   28,   31,   30,   31,},
{  185,  181,  184,  182,  180,  183,  179,  179,  179,  187,  189,
190,  181,  187,  186,  182,  175,  121,   51,   37,   34,   34,
24,   28,   36,   33,   32,   37,   36,   33,   33,   31,   35,   30,
31,   38,   31,   29,   33,   27,   32,   33,   34,   35,   33,   35,
34,   34,   33,   31,   29,   37,   34,   34,   30,   32,   34,   30,
26,   30,   27,   34,   33,   35,   36,   34,   26,   30,   43,   54,
50,   36,   36,  132,  183,  193,  196,  193,  195,  191,  190,  192,
190,  191,  193,  200,  193,  193,  193,  193,  189,  189,  190,
185,  192,  186,  186,  191,  192,  189,  192,  191,  191,  182,
188,  191,  194,  189,  190,  188,  189,  185,  191,  193,  189,
190,  192,  195,  189,  191,  190,  190,  196,  182,  189,  193,
192,  192,  181,  130,   58,   36,   33,   35,   30,   34,   35,
32,   30,   30,   33,   34,   31,   30,   29,   35,   32,   28,   31,
31,   31,   32,   34,   30,   32,   32,   28,   28,   35,   29,   30,
33,   25,   30,   24,   29,   25,   33,   35,   29,   30,   30,   28,
30,   30,   28,   28,   36,   32,   39,   31,},
{  176,  173,  185,  186,  178,  184,  187,  185,  181,  188,  191,
190,  185,  185,  189,  184,  170,  111,   46,   36,   37,   32,
34,   36,   31,   32,   39,   31,   28,   34,   34,   33,   28,   29,
32,   38,   37,   30,   36,   30,   33,   32,   33,   35,   32,   29,
32,   35,   35,   32,   36,   36,   37,   42,   34,   29,   27,   27,
34,   28,   30,   30,   23,   30,   27,   27,   30,   32,   32,   43,
41,   35,   45,  141,  183,  192,  194,  194,  186,  192,  191,  191,
192,  191,  193,  195,  194,  195,  190,  196,  188,  194,  198,
192,  197,  193,  193,  188,  190,  189,  193,  191,  197,  192,
193,  191,  190,  191,  188,  190,  196,  189,  190,  187,  189,
191,  191,  186,  192,  192,  189,  195,  194,  187,  185,  190,
191,  192,  189,  132,   64,   40,   34,   29,   34,   33,   33,
37,   28,   31,   31,   31,   33,   34,   32,   31,   34,   41,   31,
29,   34,   31,   33,   42,   33,   34,   34,   30,   32,   34,   27,
28,   28,   34,   29,   28,   33,   27,   32,   28,   27,   28,   28,
31,   31,   38,   37,   34,   29,   33,   34,},
{  185,  192,  184,  189,  179,  188,  187,  185,  183,  194,  192,
187,  186,  190,  188,  189,  162,   96,   42,   39,   31,   30,
36,   33,   32,   32,   32,   30,   37,   33,   35,   31,   26,   31,
30,   33,   31,   31,   31,   27,   30,   33,   29,   31,   29,   40,
32,   31,   30,   30,   28,   25,   31,   29,   31,   32,   34,   32,
29,   28,   33,   29,   35,   26,   31,   24,   28,   29,   31,   39,
39,   37,   45,  137,  186,  192,  196,  194,  187,  194,  199,  189,
190,  194,  193,  191,  191,  192,  193,  196,  195,  192,  192,
192,  192,  194,  191,  195,  190,  189,  189,  188,  191,  190,
190,  186,  195,  193,  194,  190,  194,  194,  197,  186,  194,
190,  194,  190,  192,  192,  191,  191,  190,  197,  192,  189,
191,  193,  191,  135,   67,   41,   36,   38,   35,   36,   36,
37,   40,   34,   34,   36,   37,   31,   30,   31,   30,   29,   30,
33,   35,   36,   34,   35,   31,   35,   36,   34,   37,   34,   35,
31,   32,   32,   27,   23,   28,   30,   30,   29,   26,   21,   29,
27,   27,   30,   37,   42,   30,   33,   30,},
{  188,  188,  187,  188,  182,  183,  187,  187,  184,  184,  189,
184,  191,  198,  192,  185,  164,   88,   39,   39,   27,   31,
36,   29,   35,   37,   32,   32,   31,   31,   33,   33,   36,   31,
26,   34,   32,   31,   31,   31,   29,   29,   38,   31,   36,   34,
32,   36,   37,   32,   33,   32,   29,   31,   28,   30,   29,   31,
29,   35,   30,   30,   34,   37,   23,   30,   26,   27,   29,   29,
26,   33,   44,  143,  186,  195,  194,  191,  190,  194,  186,  189,
```

```
192,  193,  191,  195,  182,  192,  199,  191,  197,  193,  190,
185,  188,  195,  196,  194,  194,  191,  190,  194,  191,  191,
191,  185,  185,  190,  190,  194,  193,  200,  190,  194,  196,
195,  188,  189,  190,  192,  191,  196,  193,  192,  191,  187,
192,  194,  192,  146,   69,   39,   35,   35,   30,   37,   33,
35,   31,   33,   31,   39,   37,   34,   30,   25,   29,   27,   32,
27,   29,   30,   33,   36,   37,   30,   32,   22,   25,   33,   30,
27,   30,   35,   28,   29,   25,   26,   26,   29,   34,   30,   28,
26,   30,   33,   33,   37,   34,   33,   33,},
{  191,  189,  193,  191,  190,  188,  190,  189,  194,  186,  184,
187,  192,  193,  193,  190,  145,   75,   42,   35,   39,   31,
30,   33,   34,   32,   30,   28,   38,   31,   36,   31,   29,   28,
27,   26,   32,   25,   29,   31,   27,   34,   29,   33,   30,   27,
28,   33,   30,   30,   26,   32,   32,   28,   27,   30,   32,   30,
28,   30,   28,   32,   29,   31,   30,   27,   28,   30,   34,   32,
38,   37,   53,  141,  182,  193,  187,  194,  196,  187,  185,  187,
190,  195,  193,  191,  187,  186,  190,  187,  192,  190,  192,
194,  192,  195,  189,  190,  190,  189,  190,  198,  191,  189,
187,  185,  189,  188,  188,  188,  186,  188,  192,  193,  194,
197,  194,  190,  189,  192,  192,  192,  193,  193,  193,  191,
192,  188,  194,  151,   78,   39,   39,   34,   32,   32,   34,
31,   35,   30,   31,   29,   27,   34,   34,   28,   29,   29,   31,
28,   33,   33,   28,   40,   37,   35,   32,   26,   25,   35,   27,
27,   36,   33,   29,   36,   33,   33,   28,   29,   32,   28,   25,
25,   29,   26,   27,   35,   26,   34,   33,},
{  190,  190,  188,  194,  191,  195,  185,  190,  194,  190,  186,
187,  191,  190,  189,  181,  127,   63,   35,   31,   32,   33,
29,   36,   33,   29,   31,   31,   38,   30,   29,   25,   27,   27,
27,   28,   26,   30,   25,   30,   28,   32,   32,   28,   31,   29,
28,   30,   29,   27,   25,   33,   29,   32,   33,   30,   26,   29,
33,   35,   29,   30,   28,   34,   32,   29,   43,   39,   41,   41,
42,   41,   55,  141,  182,  190,  188,  188,  192,  192,  188,  193,
194,  189,  187,  185,  187,  183,  192,  194,  185,  189,  188,
190,  190,  182,  187,  181,  186,  185,  188,  189,  186,  191,
185,  177,  187,  194,  191,  193,  189,  187,  184,  191,  191,
192,  194,  186,  177,  183,  192,  195,  191,  186,  192,  193,
196,  189,  195,  164,   83,   48,   38,   34,   37,   30,   30,
29,   29,   31,   35,   39,   36,   34,   33,   35,   28,   29,   28,
26,   28,   30,   27,   27,   30,   29,   32,   26,   28,   30,   28,
25,   26,   25,   23,   27,   26,   29,   19,   23,   27,   35,   26,
22,   23,   25,   23,   30,   25,   38,   25,},
{  187,  190,  189,  191,  198,  190,  189,  190,  190,  186,  187,
192,  186,  193,  190,  182,  116,   61,   37,   35,   32,   34,
32,   35,   32,   35,   35,   63,   77,   40,   36,   22,   31,   26,
31,   34,   34,   24,   29,   31,   32,   27,   30,   31,   29,   32,
28,   26,   28,   26,   30,   30,   27,   31,   31,   31,   26,   29,
28,   31,   37,   35,   31,   32,   25,   30,   34,   39,   44,   36,
44,   43,   53,  143,  186,  192,  192,  189,  185,  186,  187,  184,
186,  187,  188,  189,  190,  188,  195,  184,  178,  179,  186,
187,  183,  186,  185,  188,  188,  191,  185,  185,  183,  183,
185,  182,  181,  193,  190,  193,  195,  187,  180,  181,  186,
186,  193,  192,  186,  188,  188,  186,  188,  188,  193,  189,
189,  187,  189,  162,   86,   41,   31,   33,   25,   25,   23,
27,   22,   25,   25,   26,   31,   29,   25,   25,   30,   30,   26,
26,   25,   27,   25,   26,   26,   30,   30,   29,   29,   27,   23,
30,   29,   30,   27,   26,   28,   23,   32,   28,   22,   29,   29,
28,   28,   28,   31,   25,   32,   28,   32,},
{  189,  191,  196,  199,  192,  187,  193,  195,  190,  190,  189,
193,  187,  194,  193,  178,  109,   52,   34,   29,   31,   23,
35,   29,   38,   33,   35,   37,   38,   32,   27,   29,   26,   29,
24,   27,   33,   26,   28,   24,   28,   30,   29,   33,   30,   39,
```

```
35,   36,   29,   27,   32,   29,   27,   29,   33,   26,   28,   34,
29,   28,   32,   38,   32,   27,   28,   30,   33,   31,   32,   31,
35,   37,   61,  147,  182,  190,  192,  190,  191,  192,  182,  186,
187,  191,  190,  192,  194,  194,  195,  195,  184,  188,  175,
173,  179,  190,  188,  186,  191,  190,  182,  175,  177,  173,
170,  175,  171,  186,  190,  188,  195,  186,  175,  177,  183,
188,  187,  189,  187,  183,  191,  184,  190,  189,  190,  188,
186,  192,  188,  175,   99,   47,   41,   27,   29,   29,   33,
30,   27,   37,   33,   29,   33,   30,   42,   31,   31,   33,   34,
30,   32,   30,   33,   30,   27,   31,   29,   27,   24,   28,   25,
30,   29,   30,   35,   37,   20,   27,   37,   32,   28,   29,   31,
34,   28,   31,   31,   30,   33,   37,   28,},
{  192,  191,  191,  195,  193,  193,  199,  194,  188,  193,  193,
189,  195,  196,  192,  165,   97,   45,   35,   30,   30,   28,
29,   29,   32,   34,   28,   38,   34,   32,   33,   32,   34,   28,
32,   26,   22,   27,   27,   25,   29,   29,   29,   27,   35,   32,
33,   29,   34,   34,   29,   36,   29,   28,   23,   22,   28,   29,
29,   28,   35,   32,   34,   29,   29,   32,   35,   33,   26,   27,
30,   32,   47,  143,  177,  189,  193,  196,  192,  192,  192,  192,
187,  192,  191,  195,  195,  193,  193,  196,  193,  194,  193,
191,  183,  183,  196,  191,  189,  195,  191,  182,  172,  176,
177,  177,  180,  185,  192,  188,  190,  193,  186,  184,  188,
192,  191,  188,  187,  183,  173,  181,  178,  183,  188,  194,
188,  191,  189,  178,  108,   48,   42,   33,   37,   31,   31,
30,   28,   35,   34,   33,   27,   26,   31,   33,   27,   27,   34,
29,   31,   29,   35,   30,   32,   28,   29,   25,   33,   32,   26,
27,   30,   27,   34,   30,   31,   28,   29,   29,   30,   31,   33,
28,   32,   32,   34,   26,   29,   30,   29,},
{  191,  191,  194,  193,  194,  189,  186,  190,  192,  192,  189,
191,  191,  196,  192,  156,   90,   48,   32,   28,   28,   32,
26,   32,   32,   35,   32,   30,   30,   33,   32,   29,   32,   30,
32,   28,   25,   27,   29,   31,   34,   31,   36,   34,   29,   37,
38,   39,   33,   33,   30,   28,   33,   30,   28,   28,   28,   26,
29,   33,   36,   36,   31,   29,   36,   27,   35,   37,   35,   31,
33,   25,   54,  153,  191,  188,  194,  192,  192,  193,  191,  192,
193,  195,  193,  192,  193,  192,  192,  196,  196,  193,  182,
189,  183,  189,  187,  183,  188,  193,  193,  193,  188,  189,
182,  177,  174,  181,  189,  193,  192,  192,  182,  176,  192,
189,  192,  189,  174,  169,  176,  186,  187,  193,  188,  184,
189,  188,  188,  177,  114,   58,   43,   31,   33,   33,   29,
34,   32,   28,   35,   30,   30,   30,   31,   30,   33,   27,   33,
26,   36,   29,   34,   28,   32,   35,   33,   24,   30,   32,   31,
32,   32,   30,   31,   34,   30,   26,   29,   31,   26,   27,   30,
30,   30,   28,   29,   28,   28,   27,   29,},
{  187,  188,  187,  188,  191,  190,  193,  196,  194,  189,  192,
191,  191,  187,  182,  148,   75,   35,   35,   34,   28,   23,
29,   28,   35,   33,   29,   25,   31,   31,   33,   29,   29,   26,
26,   29,   31,   29,   29,   23,   27,   32,   28,   30,   33,   29,
29,   32,   21,   29,   28,   31,   32,   27,   30,   26,   28,   32,
28,   30,   26,   30,   30,   33,   35,   34,   37,   32,   26,   33,
30,   34,   63,  159,  191,  191,  193,  194,  195,  191,  192,  193,
195,  193,  191,  198,  194,  189,  191,  190,  190,  191,  193,
192,  184,  183,  187,  184,  188,  193,  189,  188,  178,  179,
188,  181,  187,  196,  188,  189,  190,  193,  184,  186,  191,
188,  189,  188,  188,  187,  184,  186,  183,  187,  193,  189,
191,  189,  194,  185,  125,   63,   39,   36,   32,   29,   32,
22,   29,   32,   31,   32,   25,   28,   32,   32,   35,   34,   37,
33,   32,   32,   28,   27,   30,   27,   31,   36,   31,   28,   27,
28,   30,   33,   34,   27,   24,   30,   33,   32,   34,   32,   26,
29,   27,   29,   27,   30,   23,   34,   31,},
```

```
{  192,  186,  174,  169,  176,  180,  192,  196,  193,  196,  194,
190,  188,  179,  170,  125,   61,   36,   34,   32,   23,   22,
24,   30,   26,   33,   28,   31,   25,   28,   25,   24,   29,   33,
32,   26,   24,   23,   27,   29,   25,   29,   24,   29,   25,   34,
33,   30,   27,   32,   29,   26,   26,   29,   25,   30,   30,   34,
33,   28,   26,   29,   32,   35,   29,   32,   35,   31,   36,   31,
40,   37,   71,  165,  187,  195,  194,  192,  194,  192,  192,  192,
196,  191,  188,  190,  195,  193,  197,  189,  192,  189,  186,
188,  191,  194,  190,  189,  186,  191,  192,  190,  191,  186,
181,  178,  185,  188,  180,  186,  189,  190,  182,  183,  184,
186,  181,  193,  185,  186,  179,  176,  184,  189,  198,  190,
190,  180,  179,  175,  140,   66,   41,   34,   31,   32,   34,
27,   35,   32,   31,   33,   29,   32,   23,   29,   28,   36,   25,
33,   33,   31,   29,   32,   28,   25,   31,   30,   27,   30,   39,
35,   32,   30,   33,   36,   31,   29,   29,   30,   31,   28,   27,
31,   33,   31,   28,   29,   31,   28,   30,},
{  193,  184,  178,  185,  188,  193,  195,  197,  192,  190,  193,
189,  183,  181,  167,  122,   53,   38,   26,   27,   28,   27,
23,   31,   31,   30,   30,   29,   27,   32,   30,   29,   31,   29,
30,   31,   21,   30,   29,   32,   34,   30,   31,   27,   32,   33,
34,   44,   39,   32,   31,   33,   36,   29,   31,   30,   36,   30,
31,   29,   34,   34,   34,   31,   32,   38,   34,   32,   32,   32,
31,   36,   71,  162,  189,  196,  196,  192,  197,  192,  192,  195,
194,  199,  194,  188,  188,  193,  193,  195,  196,  194,  193,
195,  194,  185,  188,  190,  191,  195,  188,  193,  188,  195,
189,  188,  182,  187,  187,  184,  189,  190,  185,  189,  189,
184,  187,  195,  181,  176,  184,  193,  194,  196,  194,  188,
182,  178,  184,  185,  148,   77,   46,   34,   25,   32,   32,
30,   31,   32,   29,   29,   35,   30,   27,   29,   33,   30,   33,
32,   35,   36,   33,   29,   31,   31,   35,   37,   35,   30,   30,
27,   28,   34,   26,   32,   28,   29,   28,   28,   32,   28,   24,
27,   28,   28,   22,   29,   27,   31,   31,},
{  191,  187,  194,  189,  189,  185,  191,  197,  193,  191,  193,
188,  187,  183,  167,  112,   56,   43,   36,   29,   30,   30,
29,   34,   27,   32,   36,   35,   29,   31,   32,   31,   35,   30,
31,   33,   30,   31,   32,   26,   27,   29,   29,   27,   30,   27,
37,   37,   34,   40,   32,   31,   31,   31,   37,   30,   32,   33,
32,   35,   35,   31,   38,   36,   37,   40,   35,   28,   33,   37,
43,   35,   70,  168,  191,  191,  193,  191,  190,  194,  195,  192,
192,  194,  194,  189,  194,  188,  192,  201,  190,  195,  196,
193,  193,  190,  191,  193,  190,  194,  191,  187,  186,  190,
187,  184,  181,  184,  179,  184,  191,  185,  191,  194,  188,
189,  188,  187,  193,  182,  181,  189,  193,  190,  193,  184,
187,  196,  184,  184,  158,   70,   48,   32,   28,   29,   28,
34,   33,   25,   25,   26,   28,   31,   29,   28,   28,   32,   28,
34,   33,   30,   29,   31,   34,   24,   28,   34,   27,   29,   32,
30,   30,   28,   26,   31,   35,   34,   29,   30,   28,   27,   30,
30,   24,   30,   31,   34,   34,   42,   37,},
{  192,  191,  189,  187,  190,  194,  194,  192,  191,  190,  188,
189,  193,  193,  180,  108,   52,   36,   39,   35,   33,   33,
31,   33,   32,   36,   35,   34,   32,   32,   34,   33,   30,   30,
34,   31,   30,   31,   33,   30,   30,   29,   25,   30,   29,   24,
36,   40,   34,   33,   44,   33,   34,   23,   32,   32,   27,   30,
31,   31,   29,   31,   32,   35,   29,   32,   33,   34,   39,   32,
36,   44,   87,  172,  189,  193,  191,  193,  195,  196,  191,  194,
189,  193,  191,  190,  197,  192,  197,  194,  190,  194,  190,
193,  199,  190,  192,  194,  195,  193,  193,  184,  190,  187,
195,  193,  186,  189,  183,  184,  189,  190,  190,  189,  185,
186,  191,  186,  188,  192,  189,  188,  191,  192,  185,  187,
194,  192,  190,  187,  165,   78,   40,   38,   30,   30,   33,
23,   34,   33,   32,   33,   32,   31,   30,   28,   26,   30,   32,
```

```
31,   30,   31,   30,   27,   38,   29,   30,   31,   28,   28,   30,
29,   25,   26,   28,   23,   32,   32,   31,   32,   36,   32,   33,
30,   33,   34,   34,   31,   29,   30,   29,},
{  195,  185,  186,  190,  192,  194,  192,  182,  188,  185,  192,
193,  194,  191,  173,   94,   57,   37,   32,   32,   33,   32,
30,   31,   30,   33,   32,   32,   32,   31,   32,   34,   34,   26,
37,   31,   27,   29,   32,   31,   31,   31,   29,   30,   32,   32,
33,   31,   32,   29,   29,   29,   28,   35,   29,   28,   25,   29,
26,   34,   33,   33,   33,   43,   38,   44,   39,   37,   41,   36,
32,   37,   79,  170,  188,  186,  187,  193,  198,  199,  197,  193,
197,  190,  192,  186,  189,  192,  197,  191,  194,  193,  193,
191,  185,  190,  190,  191,  197,  196,  195,  193,  192,  198,
192,  190,  189,  190,  195,  192,  194,  192,  189,  192,  190,
189,  188,  187,  188,  186,  199,  190,  188,  187,  187,  190,
187,  190,  188,  187,  163,   89,   47,   40,   33,   34,   30,
41,   28,   33,   33,   34,   36,   27,   28,   34,   33,   36,   25,
32,   37,   31,   29,   30,   33,   29,   24,   29,   29,   26,   27,
25,   25,   29,   33,   29,   33,   31,   34,   32,   31,   33,   29,
36,   35,   34,   29,   33,   40,   39,   37,},
{  193,  190,  193,  192,  195,  191,  190,  187,  190,  194,  187,
190,  191,  189,  155,   74,   42,   36,   35,   30,   30,   30,
29,   30,   29,   31,   32,   29,   30,   29,   29,   37,   31,   31,
30,   28,   31,   31,   29,   28,   26,   30,   26,   35,   30,   35,
31,   30,   26,   31,   36,   25,   29,   31,   33,   31,   32,   32,
30,   33,   28,   30,   32,   32,   30,   30,   29,   31,   37,   32,
34,   42,   96,  172,  188,  186,  196,  192,  190,  192,  196,  193,
192,  187,  187,  189,  187,  194,  192,  190,  194,  199,  193,
190,  192,  194,  193,  199,  195,  189,  187,  199,  192,  192,
189,  186,  194,  188,  190,  191,  190,  192,  191,  193,  195,
191,  185,  188,  184,  189,  188,  190,  192,  192,  190,  194,
194,  184,  187,  189,  171,  101,   49,   36,   31,   32,   32,
36,   27,   29,   28,   32,   32,   28,   37,   32,   33,   36,   33,
37,   31,   33,   37,   32,   24,   32,   30,   29,   24,   34,   29,
34,   32,   25,   33,   33,   30,   33,   31,   26,   33,   26,   27,
33,   30,   34,   29,   30,   34,   33,   34,},
{  193,  192,  191,  193,  192,  191,  187,  191,  192,  188,  191,
192,  192,  187,  144,   72,   40,   35,   29,   32,   30,   31,
30,   26,   30,   29,   27,   30,   37,   27,   33,   31,   39,   35,
29,   31,   30,   28,   23,   27,   31,   33,   31,   31,   31,   30,
26,   33,   28,   33,   28,   30,   31,   35,   38,   32,   29,   27,
37,   31,   31,   29,   29,   28,   25,   27,   30,   31,   27,   33,
37,   46,   89,  166,  175,  186,  188,  192,  196,  192,  188,  184,
180,  191,  188,  196,  197,  195,  188,  182,  180,  189,  191,
193,  194,  189,  188,  184,  187,  190,  190,  190,  188,  185,
185,  192,  195,  194,  193,  189,  190,  188,  196,  192,  188,
188,  186,  187,  191,  195,  193,  190,  187,  187,  191,  193,
195,  197,  191,  187,  176,  116,   62,   38,   32,   33,   30,
37,   34,   35,   40,   37,   32,   34,   36,   41,   32,   31,   31,
31,   33,   30,   32,   29,   31,   32,   34,   35,   28,   35,   31,
32,   26,   31,   36,   33,   34,   33,   33,   35,   35,   38,   29,
33,   29,   29,   34,   33,   32,   34,   34,}
};


struct header {
      int nr, nc;             /* Rows and columns in the image */
      int oi, oj;             /* Origin */
};

/*      The IMAGE data structure        */
```

```
struct image {
            struct header *info;            /* Pointer to header */
            unsigned char **data;           /* Pixel values */
};
typedef struct image * IMAGE;

int    PBM_SE_ORIGIN_COL=0, PBM_SE_ORIGIN_ROW=0;

struct image  *newimage (int nr, int nc)
{
        struct image  *x;               /* New image */
        unsigned char *ptr;             /* new pixel array */
        int i;

        if (nr < 0 || nc < 0) {
                printf ("Error: Bad image size (%d,%d)\n", nr, nc);
                return 0;
        }

/*      Allocate the image structure    */
        x = (struct image  *) malloc( sizeof (struct image) );
        if (!x) {
                printf ("Out of storage in NEWIMAGE.\n");
                return 0;
        }

/*      Allocate and initialize the header      */

        x->info = (struct header *)malloc( sizeof(struct header) );
        if (!(x->info)) {
                printf ("Out of storage in NEWIMAGE.\n");
                return 0;
        }
        x->info->nr = nr;       x->info->nc = nc;
        x->info->oi = x->info->oj = 0;

/*      Allocate the pixel array        */

        x->data = (unsigned char **)malloc(sizeof(unsigned char
*)*nr);


/* Pointers to rows */
        if (!(x->data)) {
                printf ("Out of storage in NEWIMAGE.\n");
                return 0;
        }

        for (i=0; i<nr; i++)
        {
          x->data[i] = (unsigned char *)malloc(nc*sizeof(unsigned
char));
          if (x->data[i]==0)
          {
                printf ("Out of storage. Newimage - row %d\n", i);
                exit(1);
          }
        }
        return x;
}
```

```
/*function prototypes*/
void grad1 (IMAGE x);
int range (IMAGE im, int i, int j);
float ** f2d (int nr, int nc);
void freeimage (struct image  *z);
void sys_abort (int val, char *mess);
void copy (IMAGE *a, IMAGE b);
void CopyVarImage (IMAGE *a, IMAGE *b);
IMAGE Output_PBM (IMAGE image, char *filename);


main ()
{
    XTmrCtr XPS_Timer ;
    //typedef unsigned long long XTime;
    XTime start ;
    XTime end ;
    unsigned int diff ;
    //void XTime_SetTime(XTime Xtime);
    //void XTime_GetTime(XTime *Xtime);



      int i, j;
      IMAGE x;

      printf ("\nPGM file class %d size %d columns X %d rows
Max=%d\n", CLASS, COLUMNS, ROWS, MAX);

      x = (IMAGE)newimage (ROWS, COLUMNS);
      x->info->oi = PBM_SE_ORIGIN_ROW;
      x->info->oj = PBM_SE_ORIGIN_COL;
      PBM_SE_ORIGIN_ROW = 0;
      PBM_SE_ORIGIN_COL = 0;
      for (i=0; i<ROWS; i++)
          for (j=0; j<COLUMNS; j++)
            x->data[i][j] = imgdata[i][j];


    XTmrCtr_Initialize(&XPS_Timer,XPAR_XPS_TIMER_0_DEVICE_ID) ;
    XTmrCtr_SetResetValue(&XPS_Timer,0,0x00000000) ;
    XTmrCtr_Reset(&XPS_Timer,0) ;
    XTmrCtr_Start(&XPS_Timer,0) ;


    start=0;
    XTime_SetTime(start) ;
    XTime_GetTime(&start) ;

      grad1 (x);

    XTime_GetTime(&end) ;

    XTmrCtr_Stop(&XPS_Timer,0) ;
    i = XTmrCtr_GetValue(&XPS_Timer,0) ;
    putnum(i) ;
    print("\r\n");

    printf ("Time : %L\n",end);
    putnum( end ) ;
```

```
    print("\r\n");
    i = (unsigned int) end-start ;
    putnum( i ) ;
    print("\r\n");
    i = (unsigned int) (end>>32) - (start>>32)  ;
    putnum( i ) ;
    print("\r\n");

    //delay :
    for (i=0; i<8000000; ++i) ;







      Output_PBM (x, "grad1.pgm");
      printf ("Output is in file 'grad1.pgm'\n");
}



void grad1 (IMAGE x)
{
      int i,j,k;
      double z, dx, dy;
      int pmax=0, pmin=255, thresh = 128;

      for (i=x->info->nr-1; i>= 1; i--)
        for (j=x->info->nc-1; j>=1; j--)
        {
          dx = (x->data[i][j]-x->data[i-1][j]);
          dy = (x->data[i][j]-x->data[i][j-1]);
          z = sqrt(dx*dx + dy*dy);

          if (z > 255.0) z = 255.0;
           else if (z < 0.0) z = 0.0;
          x->data[i][j] = (unsigned char)z;
          if (pmax < (unsigned char)z) pmax = (unsigned char)z;
          if (pmin > (unsigned char)z) pmin = (unsigned char)z;
        }
      thresh = (pmax-pmin)/2;

      for (i=0; i<x->info->nr; i++)
      {
        x->data[i][0] = 0;
        x->data[i][x->info->nc-1] = 0;
      }
      for (j=0; j<x->info->nc; j++)
      {
        x->data[0][j] = 0;
        x->data[x->info->nr-1][j] = 0;
      }

/*    Threshold   */
      for (i=0; i<x->info->nr; i++)
        for (j=0; j<x->info->nc; j++)
          if (x->data[i][j] > thresh) x->data[i][j] = 0;
           else x->data[i][j] = 255;
}
```

```
int range (IMAGE im, int i, int j)
{
        if ((i<0) || (i>=im->info->nr)) return 0;
        if ((j<0) || (j>=im->info->nc)) return 0;
        return 1;
}



float ** f2d (int nr, int nc)
{
        float **x, *y;
        int i;

        x = (float **)calloc ( nr, sizeof (float *) );
        if (x == 0)
        {
          printf ("Out of storage: F2D.\n");
          exit (1);
        }

        for (i=0; i<nr; i++)
        {
          x[i] = (float *) calloc ( nc, sizeof (float)  );
          if (x[i] == 0)
          {
            printf ("Out of storage: F2D %d.\n", i);
            exit (1);
          }
        }
        return x;
}

void freeimage (struct image  *z)
{
/*      Free the storage associated with the image Z    */
        int i;

        if (z != 0)
        {
          for (i=0; i<z->info->nr; i++)
              free (z->data[i]);
           free (z->info);
           free (z->data);
           free (z);
        }
}

void sys_abort (int val, char *mess)
{
        printf ("**** System library ABORT %d: %s ****\n",
                        val, mess);
        exit (2);
}

void copy (IMAGE *a, IMAGE b)
{
        CopyVarImage (a, &b);
}
```

```
void CopyVarImage (IMAGE *a, IMAGE *b)
{
        int i,j;

        if (a == b) return;
        if (*a) freeimage (*a);
        *a = newimage ((*b)->info->nr, (*b)->info->nc);
        if (*a == 0) sys_abort (0, "No more storage.\n");

        for (i=0; i<(*b)->info->nr; i++)
          for (j=0; j< (*b)->info->nc; j++)
            (*a)->data[i][j] = (*b)->data[i][j];
        (*a)->info->oi = (*b)->info->oi;
        (*a)->info->oj = (*b)->info->oj;
}


IMAGE Output_PBM (IMAGE image, char *filename)
{
      FILE *f;
      int i,j,k, perline;
      char buf1[64];

//strcpy (buf1, filename);
      if (image->info->nc > 20) perline = 20;
       else perline = image->info->nc-1;
//f = fopen (buf1, "w");
//if (f == 0) sys_abort (0, "Can't open output file.");

      printf ("P2\n#origin %d %d\n",image->info->oj,image->info->oi);
      printf ("%d %d %d\n", image->info->nc, image->info->nr, 255);
      k = 0;
      for (i=0; i<image->info->nr; i++)
        for (j=0; j<image->info->nc; j++)
        {
            printf ( "%d ", image->data[i][j]);
            k++;
            if (k > perline)
            {
              printf ("\n");
              k = 0;
            }
        }
      printf ("\n");
//fclose (f);
      return image;
}
```

ΠΑΡΑΡΤΗΜΑ 2

Η εφαρμογή θα ασχολείται πάλι με την αλλαγή των χρωμάτων όμως αυτή τη φορά δεν θα είναι σε οριζόντιο επίπεδο αλλά και σε κατακόρυφο με σκοπό να προσπαθεί να αναγνωρίσει τα περιγράμματα από την εικόνα(π.χ σε ένα σκάκι που προσπαθεί να βρει τα περιγράμματα από τα τετράγωνα και τα πιόνια). Ο πίνακας αυτός θα στέλνεται μέσω συριακής θύρας σε ένα υπολογιστή ο οποίος θα καταγράφει τα δεδομένα σε ένα αρχείο .txt το οποίο θα το ανοίξουμε στη συνεχεία με ένα image editor για να δούμε την φιλτραρισμένη εικόνα.

/*Canny*/

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <string.h>
#include <math.h>

#define MAX 255
#define ROWS 181
#define COLUMNS 35
#define CLASS 5
#define PI 3.14
//#include "lib.h"
#include "xparameters.h"
#include "xtmrctr.h"
#include "xtime_l.h"
/* Scale floating point magnitudes and angles to 8 bits */
#define ORI_SCALE 40.0
#define MAG_SCALE 20.0

/* Biggest possible filter mask */
#define MAX_MASK_SIZE 20


unsigned char imgdata[181][35] = {
{   30,   32,   40,   31,   30,   30,   39,   31,   29,   32,   28,
34,   26,   30,   31,   32,   28,   27,   34,   38,   96,  174,  192,
194,  194,  195,  191,  186,  184,  191,  191,  193,  190,  183,
171,  147,  120,   89,   58,   40,   30,   26,   26,   27,   28,
22,   24,   19,   26,   25,   33,   48,   76,  109,  146,  178,  184,
188,  195,  196,  194,  196,  199,  193,  193,  190,  188,  188,
190,  193,  196,  196,  194,  150,   71,   43,   39,   34,   33,
36,   31,   32,   34,   37,   33,   35,   31,   40,   31,   32,   32,
35,   34,   28,   34,   29,   37,   31,   33,   29,   32,   36,   38,
31,   35,   29,   35,   30,   35,   33,   39,   36,   30,   27,   36,
36,   37,   37,   35,   40,   43,   36,   30,   29,   34,   31,   39,
91,  174,  184,  191,  197,  192,  193,  195,  188,  195,  192,  186,
191,  193,  192,  188,  191,  192,  194,  192,  195,  189,  184,
194,  196,  192,  191,  195,  190,  191,  198,  195,  193,  189,
193,  195,  187,  183,  190,  190,  189,  190,  187,  186,  189,
190,  192,  186,  188,  187,  193,  186,  182,  180,},
{   31,   25,   29,   30,   32,   29,   27,   34,   31,   26,   30,
30,   30,   34,   32,   29,   33,   31,   35,   54,  120,  183,  189,
197,  195,  194,  195,  192,  197,  199,  191,  188,  186,  189,
192,  191,  191,  183,  164,  133,  100,   78,   63,   55,   49,
```

```
48,   52,   58,   74,  105,  140,  170,  183,  187,  196,  193,  192,
194,  193,  194,  193,  198,  198,  193,  200,  195,  195,  194,
194,  192,  195,  197,  196,  144,   72,   43,   37,   39,   34,
34,   37,   37,   38,   35,   38,   38,   38,   34,   34,   38,   32,
38,   34,   35,   36,   30,   32,   36,   38,   44,   36,   34,   41,
35,   34,   33,   30,   31,   37,   36,   37,   34,   31,   40,   38,
37,   40,   41,   39,   41,   36,   41,   38,   37,   38,   38,   34,
86,  168,  187,  192,  194,  194,  193,  192,  191,  189,  190,  187,
193,  191,  190,  192,  196,  189,  193,  189,  196,  192,  185,
195,  195,  191,  192,  188,  190,  193,  187,  189,  192,  182,
192,  190,  190,  187,  187,  185,  183,  190,  192,  188,  192,
192,  191,  186,  183,  189,  183,  188,  181,  179,},
{   34,   31,   36,   27,   36,   31,   31,   36,   31,   33,   35,
33,   28,   31,   32,   37,   31,   35,   36,   50,  139,  185,  189,
191,  194,  196,  194,  196,  192,  192,  193,  194,  189,  191,
191,  192,  188,  195,  190,  192,  183,  181,  181,  178,  177,
180,  185,  189,  192,  193,  189,  197,  193,  196,  186,  192,
197,  190,  197,  200,  194,  192,  191,  193,  195,  193,  192,
192,  200,  194,  190,  190,  192,  146,   68,   39,   35,   30,
38,   33,   35,   33,   31,   38,   38,   34,   31,   27,   34,   32,
32,   38,   28,   33,   33,   29,   29,   30,   25,   31,   32,   33,
33,   37,   41,   39,   37,   28,   34,   34,   34,   32,   33,   35,
35,   35,   34,   41,   37,   34,   32,   36,   35,   37,   31,   37,
37,   80,  167,  187,  190,  190,  186,  188,  188,  181,  189,  185,
193,  187,  192,  187,  190,  193,  191,  190,  196,  192,  194,
193,  198,  192,  194,  191,  191,  193,  187,  182,  184,  190,
187,  184,  190,  183,  182,  183,  173,  174,  174,  167,  152,
159,  153,  163,  158,  151,  154,  145,  142,  139,  134,},
{   30,   35,   34,   36,   39,   33,   39,   39,   34,   34,   32,
30,   32,   33,   31,   32,   30,   30,   36,   59,  153,  183,  191,
193,  185,  191,  192,  193,  190,  190,  191,  189,  193,  193,
193,  193,  192,  196,  192,  196,  196,  193,  197,  191,  192,
195,  191,  191,  194,  193,  190,  191,  192,  195,  199,  199,
190,  192,  196,  194,  198,  193,  195,  193,  195,  194,  194,
194,  192,  195,  193,  191,  192,  141,   61,   37,   34,   30,
33,   33,   35,   34,   37,   40,   36,   36,   35,   26,   36,   31,
37,   34,   36,   36,   30,   35,   33,   31,   36,   34,   30,   32,
42,   35,   32,   31,   28,   36,   27,   32,   35,   36,   38,   37,
42,   40,   34,   32,   33,   35,   31,   33,   42,   37,   33,   31,
38,   65,  160,  191,  188,  185,  192,  194,  190,  195,  195,  187,
187,  194,  187,  182,  186,  178,  180,  185,  181,  181,  176,
178,  182,  168,  163,  157,  155,  149,  138,  134,  127,  114,
110,  106,  102,   99,   95,   91,   78,   73,   67,   65,   58,
54,   54,   52,   56,   51,   49,   39,   52,   44,   46,},
{   38,   37,   33,   35,   35,   27,   28,   34,   37,   40,   34,
32,   35,   32,   40,   32,   34,   34,   36,   76,  166,  188,  191,
192,  193,  195,  191,  193,  195,  192,  188,  191,  190,  201,
192,  194,  196,  192,  197,  198,  192,  191,  196,  191,  196,
193,  190,  195,  189,  196,  191,  197,  195,  191,  190,  194,
192,  198,  194,  188,  191,  190,  196,  189,  191,  190,  197,
193,  194,  195,  192,  192,  190,  136,   63,   34,   31,   31,
31,   34,   32,   32,   35,   32,   32,   31,   36,   37,   34,   24,
25,   28,   28,   33,   30,   27,   27,   30,   35,   31,   29,   33,
29,   32,   28,   26,   30,   37,   34,   31,   37,   41,   32,   33,
31,   29,   30,   38,   30,   38,   35,   36,   45,   42,   42,   42,
45,   56,  129,  145,  143,  136,  127,  128,  118,  107,  106,   95,
92,   94,   87,   79,   75,   66,   68,   63,   56,   64,   58,   50,
49,   49,   50,   46,   46,   47,   48,   41,   50,   42,   44,   44,
42,   46,   43,   44,   47,   49,   43,   41,   39,   38,   42,   34,
40,   39,   37,   34,   36,   31,   36,},
```

```
{   32,   31,   30,   31,   41,   29,   36,   35,   38,   33,   36,
33,   35,   34,   34,   39,   34,   39,   46,   99,  174,  187,  195,
193,  192,  193,  190,  194,  197,  197,  192,  196,  192,  189,
190,  190,  195,  197,  193,  195,  196,  196,  195,  194,  191,
194,  198,  192,  199,  194,  195,  195,  193,  197,  198,  195,
193,  195,  199,  199,  193,  193,  194,  197,  193,  191,  198,
190,  197,  194,  196,  190,  194,  129,   60,   38,   37,   29,
29,   32,   33,   38,   35,   36,   34,   34,   35,   32,   34,   37,
36,   37,   35,   38,   41,   38,   42,   41,   41,   41,   39,   47,
48,   53,   59,   47,   61,   68,   74,   74,   71,   65,   84,   96,
91,   94,   93,   94,  111,  116,  124,  127,  132,  146,  155,  156,
155,  151,   93,   60,   46,   45,   49,   43,   45,   48,   49,
49,   41,   38,   42,   44,   40,   46,   38,   43,   42,   39,   38,
38,   39,   42,   41,   38,   37,   42,   37,   34,   42,   44,   37,
36,   36,   33,   37,   38,   35,   36,   32,   33,   34,   41,   36,
33,   33,   30,   32,   22,   28,   34,   31,},
{   38,   32,   36,   34,   33,   30,   33,   31,   29,   32,   32,
31,   40,   33,   36,   32,   32,   33,   40,  111,  179,  196,  194,
194,  192,  197,  194,  193,  194,  194,  190,  201,  193,  192,
190,  199,  198,  191,  196,  193,  196,  193,  192,  197,  195,
195,  193,  194,  194,  194,  194,  193,  193,  196,  192,  196,
196,  200,  195,  200,  193,  189,  192,  187,  193,  192,  190,
186,  191,  193,  187,  190,  186,  128,   64,   55,   58,   60,
63,   67,   75,   74,   80,   88,   92,   97,   98,  103,  116,  120,
115,  122,  125,  137,  135,  138,  139,  151,  151,  149,  153,
162,  167,  165,  167,  171,  168,  166,  183,  184,  163,  162,
182,  182,  174,  175,  171,  169,  169,  176,  177,  181,  186,
181,  187,  189,  185,  166,   93,   54,   41,   37,   37,   31,
36,   40,   36,   34,   37,   30,   38,   35,   34,   46,   42,   44,
42,   45,   46,   42,   40,   36,   35,   37,   37,   34,   41,   38,
33,   35,   38,   31,   39,   34,   36,   33,   35,   32,   30,   33,
35,   30,   28,   30,   26,   28,   26,   28,   31,   34,   25,},
{   34,   37,   34,   38,   36,   29,   28,   34,   28,   36,   40,
30,   34,   37,   34,   33,   40,   39,   55,  130,  183,  192,  198,
189,  192,  192,  195,  200,  191,  195,  195,  194,  196,  193,
184,  188,  187,  193,  192,  196,  185,  186,  180,  182,  190,
183,  182,  175,  177,  172,  172,  167,  161,  152,  147,  146,
152,  143,  129,  122,  115,  107,  109,  106,  105,   97,   88,
83,   76,   82,   82,   84,   82,  105,  149,  157,  151,  161,  172,
163,  156,  162,  159,  173,  172,  170,  166,  165,  171,  170,
176,  175,  165,  164,  166,  168,  171,  179,  173,  179,  176,
179,  185,  182,  184,  177,  177,  176,  188,  187,  177,  170,
180,  184,  180,  178,  174,  174,  174,  184,  185,  182,  182,
186,  189,  183,  179,  170,  101,   60,   46,   43,   35,   31,
29,   33,   31,   39,   36,   36,   33,   29,   33,   36,   37,   39,
39,   34,   36,   41,   36,   39,   41,   33,   34,   34,   37,   40,
32,   35,   36,   32,   33,   32,   25,   26,   35,   30,   31,   40,
33,   30,   33,   30,   32,   30,   28,   26,   31,   37,   30,},
{   40,   38,   40,   38,   37,   40,   41,   39,   47,   42,   42,
49,   54,   50,   50,   57,   47,   51,   70,  108,  121,  122,  121,
111,  106,  105,   94,   91,   88,   84,   93,   90,   88,   84,
74,   71,   70,   73,   69,   66,   68,   61,   59,   58,   57,   56,
54,   55,   51,   49,   53,   51,   53,   46,   47,   52,   44,   48,
43,   39,   42,   49,   42,   46,   45,   44,   42,   40,   43,   47,
48,   45,   42,   95,  168,  178,  172,  182,  181,  178,  173,  178,
182,  186,  178,  175,  173,  175,  172,  171,  174,  177,  179,
168,  173,  178,  169,  165,  182,  181,  185,  183,  176,  181,
181,  175,  179,  178,  180,  185,  178,  172,  180,  185,  182,
185,  177,  174,  177,  176,  179,  187,  185,  191,  190,  184,
180,  174,  115,   56,   40,   37,   33,   29,   37,   29,   43,
32,   40,   37,   33,   31,   43,   42,   37,   33,   33,   33,   32,
```

```
34,   36,   36,   31,   33,   33,   36,   33,   35,   37,   39,   35,
28,   36,   27,   29,   30,   32,   32,   32,   34,   31,   25,   30,
26,   29,   29,   35,   29,   32,   31,   36,},
{  117,  120,  125,  107,  120,  121,  122,  125,  123,  121,  146,
139,  153,  158,  156,  150,  151,  162,  155,   95,   60,   49,
48,   42,   42,   38,   38,   43,   49,   45,   41,   36,   45,   40,
42,   38,   40,   45,   42,   41,   37,   43,   40,   38,   37,   36,
47,   42,   37,   38,   39,   37,   38,   33,   32,   35,   40,   36,
33,   37,   39,   42,   37,   31,   35,   36,   35,   35,   33,   33,
31,   35,   39,  100,  173,  184,  195,  190,  185,  181,  185,  186,
187,  183,  179,  183,  182,  176,  181,  183,  182,  181,  187,
180,  183,  183,  181,  178,  187,  187,  183,  186,  181,  183,
184,  183,  184,  189,  185,  181,  183,  182,  183,  186,  187,
193,  176,  182,  182,  184,  187,  187,  187,  182,  187,  186,
193,  171,  128,   54,   42,   39,   36,   33,   29,   32,   30,
33,   36,   36,   36,   31,   32,   33,   37,   38,   36,   34,   35,
32,   36,   37,   28,   33,   36,   38,   38,   44,   32,   33,   32,
38,   37,   28,   32,   32,   31,   28,   33,   31,   39,   34,   33,
36,   29,   30,   29,   28,   30,   29,   32,},
{  165,  174,  168,  156,  153,  156,  159,  164,  157,  161,  172,
178,  177,  176,  178,  170,  174,  166,  139,   79,   47,   37,
37,   41,   35,   35,   38,   37,   32,   38,   36,   36,   37,   30,
30,   32,   37,   37,   37,   36,   38,   37,   34,   38,   35,   46,
40,   34,   41,   39,   34,   32,   39,   32,   35,   37,   40,   35,
35,   34,   35,   35,   34,   32,   29,   27,   31,   34,   33,   41,
34,   30,   35,  100,  168,  187,  191,  191,  187,  180,  187,  188,
184,  189,  190,  191,  195,  188,  194,  185,  186,  186,  188,
193,  188,  190,  193,  188,  194,  194,  186,  186,  190,  187,
182,  184,  184,  184,  181,  185,  184,  188,  184,  189,  185,
190,  186,  185,  187,  186,  189,  188,  192,  188,  189,  188,
189,  186,  132,   60,   42,   32,   41,   33,   35,   38,   39,
42,   32,   36,   31,   29,   35,   29,   34,   30,   25,   33,   38,
36,   35,   33,   37,   42,   37,   35,   41,   41,   36,   37,   34,
41,   40,   34,   35,   34,   28,   31,   32,   39,   34,   31,   31,
31,   33,   29,   26,   28,   28,   27,   31,},
{  181,  175,  169,  169,  171,  176,  177,  174,  177,  172,  179,
180,  180,  182,  178,  178,  171,  168,  133,   66,   39,   35,
28,   27,   28,   30,   31,   36,   34,   35,   33,   34,   32,   30,
27,   32,   33,   39,   38,   41,   40,   36,   39,   38,   39,   38,
38,   35,   37,   33,   32,   37,   38,   32,   33,   32,   34,   35,
31,   33,   32,   29,   34,   27,   32,   26,   28,   35,   32,   35,
32,   32,   34,  110,  180,  184,  192,  185,  190,  190,  191,  195,
193,  184,  181,  187,  191,  182,  181,  186,  194,  192,  187,
188,  198,  191,  186,  193,  189,  191,  185,  189,  190,  190,
188,  189,  192,  182,  189,  184,  188,  195,  191,  192,  191,
187,  192,  193,  187,  186,  192,  196,  198,  192,  191,  192,
195,  185,  146,   74,   42,   35,   36,   36,   35,   31,   33,
38,   35,   34,   31,   31,   28,   34,   37,   39,   37,   34,   33,
31,   34,   35,   33,   34,   36,   38,   41,   37,   33,   45,   35,
35,   33,   33,   32,   29,   35,   27,   30,   31,   29,   32,   28,
28,   28,   29,   36,   25,   25,   30,   28,},
{  181,  172,  175,  175,  175,  176,  182,  175,  177,  179,  186,
176,  179,  186,  185,  181,  176,  165,  108,   53,   41,   35,
36,   35,   36,   31,   33,   36,   34,   35,   29,   33,   33,   32,
29,   30,   28,   34,   29,   34,   39,   38,   37,   45,   44,   39,
39,   38,   34,   36,   39,   35,   34,   33,   36,   26,   28,   32,
32,   29,   29,   35,   28,   30,   31,   31,   31,   39,   38,   40,
41,   43,   43,  114,  176,  190,  190,  192,  191,  186,  194,  200,
191,  191,  189,  188,  189,  190,  186,  181,  184,  191,  191,
186,  187,  191,  187,  184,  190,  191,  183,  181,  189,  188,
183,  190,  188,  190,  182,  195,  189,  190,  187,  188,  183,
```

```
185,  193,  190,  192,  179,  189,  188,  194,  190,  188,  188,
185,  189,  146,   64,   41,   36,   33,   35,   29,   31,   30,
33,   33,   29,   25,   39,   33,   39,   36,   35,   30,   30,   35,
33,   35,   29,   30,   30,   27,   35,   34,   35,   34,   34,   35,
35,   29,   34,   30,   32,   32,   30,   30,   28,   33,   33,   32,
30,   27,   34,   28,   31,   27,   31,   33,},
{  187,  189,  182,  186,  180,  185,  175,  179,  184,  185,  187,
187,  186,  192,  182,  177,  181,  165,  104,   51,   38,   34,
33,   38,   31,   32,   37,   33,   36,   33,   31,   30,   35,   33,
31,   28,   30,   30,   30,   33,   39,   36,   34,   34,   40,   39,
40,   35,   33,   37,   37,   38,   32,   35,   31,   32,   32,   34,
34,   32,   32,   33,   27,   32,   30,   38,   30,   31,   29,   37,
38,   35,   39,  118,  177,  188,  190,  191,  192,  190,  196,  191,
190,  190,  190,  187,  189,  191,  189,  188,  188,  194,  191,
191,  188,  191,  191,  193,  192,  192,  192,  189,  189,  191,
188,  185,  187,  187,  185,  185,  191,  190,  187,  187,  187,
190,  189,  192,  190,  188,  191,  187,  188,  190,  191,  189,
185,  180,  152,   82,   54,   42,   35,   35,   33,   33,   35,
38,   30,   36,   32,   34,   41,   29,   34,   35,   35,   36,   29,
32,   31,   32,   37,   34,   34,   31,   31,   27,   34,   35,   31,
32,   31,   28,   32,   25,   32,   28,   31,   28,   29,   31,   33,
30,   25,   33,   29,   28,   24,   30,   27,},
{  189,  183,  187,  192,  190,  190,  189,  184,  192,  195,  184,
181,  189,  186,  192,  189,  184,  150,   87,   40,   32,   34,
35,   37,   36,   31,   30,   29,   36,   26,   31,   31,   32,   30,
32,   32,   31,   26,   34,   27,   36,   38,   39,   43,   43,   38,
37,   37,   33,   38,   33,   34,   31,   35,   37,   30,   38,   38,
33,   35,   32,   32,   31,   33,   31,   25,   36,   33,   40,   44,
43,   42,   42,  127,  181,  192,  196,  191,  195,  187,  191,  188,
195,  191,  185,  186,  199,  192,  193,  190,  186,  193,  194,
191,  190,  190,  192,  190,  186,  188,  192,  191,  191,  195,
190,  194,  187,  190,  189,  187,  184,  186,  194,  192,  193,
191,  193,  187,  192,  194,  192,  185,  187,  189,  191,  192,
188,  186,  167,   84,   44,   40,   33,   36,   33,   29,   33,
25,   36,   30,   30,   30,   28,   26,   35,   32,   33,   34,   34,
39,   30,   33,   32,   31,   31,   33,   40,   38,   33,   30,   31,
29,   32,   29,   28,   29,   28,   32,   31,   30,   32,   33,   29,
32,   29,   34,   25,   25,   29,   34,   32,},
{  178,  187,  183,  180,  188,  186,  180,  185,  187,  185,  189,
184,  187,  188,  187,  188,  180,  146,   75,   41,   33,   30,
30,   26,   36,   35,   32,   33,   31,   33,   30,   31,   38,   35,
43,   35,   35,   31,   38,   38,   31,   32,   41,   33,   39,   38,
34,   33,   34,   39,   34,   34,   34,   31,   27,   38,   34,   30,
38,   35,   27,   30,   26,   30,   31,   31,   30,   33,   30,   31,
33,   28,   40,  130,  183,  191,  190,  191,  190,  190,  188,  195,
194,  194,  191,  189,  191,  191,  195,  191,  191,  193,  192,
188,  195,  192,  192,  194,  192,  190,  184,  187,  193,  189,
191,  191,  186,  189,  186,  190,  189,  189,  191,  192,  190,
190,  192,  189,  190,  193,  189,  188,  194,  195,  192,  191,
192,  189,  174,   97,   48,   37,   36,   36,   38,   33,   34,
40,   31,   31,   30,   38,   33,   33,   31,   40,   35,   29,   29,
34,   33,   33,   32,   34,   32,   37,   36,   35,   39,   37,   32,
29,   26,   27,   28,   26,   27,   24,   26,   26,   30,   23,   29,
27,   27,   28,   27,   21,   24,   29,   27,},
{  183,  179,  178,  179,  187,  189,  182,  185,  187,  190,  183,
186,  189,  182,  193,  182,  180,  130,   67,   41,   39,   30,
35,   29,   31,   34,   30,   30,   34,   32,   34,   33,   33,   35,
35,   39,   37,   31,   35,   33,   33,   41,   38,   37,   43,   42,
39,   38,   28,   33,   35,   36,   33,   33,   33,   37,   38,   31,
31,   33,   31,   31,   28,   27,   26,   29,   28,   31,   32,   40,
39,   41,   49,  130,  180,  191,  191,  191,  189,  194,  189,  186,
```

```
191,  192,  193,  193,  192,  198,  194,  195,  191,  185,  187,
194,  192,  192,  193,  189,  189,  180,  185,  183,  188,  190,
189,  188,  187,  185,  189,  194,  187,  185,  194,  187,  191,
194,  189,  195,  188,  186,  192,  187,  195,  193,  192,  193,
197,  188,  177,  112,   48,   34,   32,   31,   31,   28,   32,
40,   38,   25,   33,   30,   34,   34,   30,   31,   29,   32,   29,
29,   32,   30,   33,   30,   34,   37,   29,   32,   29,   34,   34,
32,   25,   26,   22,   30,   30,   26,   27,   29,   31,   28,   26,
29,   25,   31,   32,   28,   31,   30,   31,},
{  185,  181,  184,  182,  180,  183,  179,  179,  179,  187,  189,
190,  181,  187,  186,  182,  175,  121,   51,   37,   34,   34,
24,   28,   36,   33,   32,   37,   36,   33,   33,   31,   35,   30,
31,   38,   31,   29,   33,   27,   32,   33,   34,   35,   33,   35,
34,   34,   33,   31,   29,   37,   34,   34,   30,   32,   34,   30,
26,   30,   27,   34,   33,   35,   36,   34,   26,   30,   43,   54,
50,   36,   36,  132,  183,  193,  196,  193,  195,  191,  190,  192,
190,  191,  193,  200,  193,  193,  193,  193,  189,  189,  190,
185,  192,  186,  186,  191,  192,  189,  192,  191,  191,  182,
188,  191,  194,  189,  190,  188,  189,  185,  191,  193,  189,
190,  192,  195,  189,  191,  190,  190,  196,  182,  189,  193,
192,  192,  181,  130,   58,   36,   33,   35,   30,   34,   35,
32,   30,   30,   33,   34,   31,   30,   29,   35,   32,   28,   31,
31,   31,   32,   34,   30,   32,   32,   28,   28,   35,   29,   30,
33,   25,   30,   24,   29,   25,   33,   35,   29,   30,   30,   28,
30,   30,   28,   28,   36,   32,   39,   31,},
{  176,  173,  185,  186,  178,  184,  187,  185,  181,  188,  191,
190,  185,  185,  189,  184,  170,  111,   46,   36,   37,   32,
34,   36,   31,   32,   39,   31,   28,   34,   34,   33,   28,   29,
32,   38,   37,   30,   36,   30,   33,   32,   33,   35,   32,   29,
32,   35,   35,   32,   36,   36,   37,   42,   34,   29,   27,   27,
34,   28,   30,   30,   23,   30,   27,   27,   30,   32,   32,   43,
41,   35,   45,  141,  183,  192,  194,  194,  186,  192,  191,  191,
192,  191,  193,  195,  194,  195,  190,  196,  188,  194,  198,
192,  197,  193,  193,  188,  190,  189,  193,  191,  197,  192,
193,  191,  190,  191,  188,  190,  196,  189,  190,  187,  189,
191,  191,  186,  192,  192,  189,  195,  194,  187,  185,  190,
191,  192,  189,  132,   64,   40,   34,   29,   34,   33,   33,
37,   28,   31,   31,   31,   33,   34,   32,   31,   34,   41,   31,
29,   34,   31,   33,   42,   33,   34,   34,   30,   32,   34,   27,
28,   28,   34,   29,   28,   33,   27,   32,   28,   27,   28,   28,
31,   31,   38,   37,   34,   29,   33,   34,},
{  185,  192,  184,  189,  179,  188,  187,  185,  183,  194,  192,
187,  186,  190,  188,  189,  162,   96,   42,   39,   31,   30,
36,   33,   32,   32,   32,   30,   37,   33,   35,   31,   26,   31,
30,   33,   31,   31,   31,   27,   30,   33,   29,   31,   29,   40,
32,   31,   30,   30,   28,   25,   31,   29,   31,   32,   34,   32,
29,   28,   33,   29,   35,   26,   31,   24,   28,   29,   31,   39,
39,   37,   45,  137,  186,  192,  196,  194,  187,  194,  199,  189,
190,  194,  193,  191,  191,  192,  193,  196,  195,  192,  192,
192,  192,  194,  191,  195,  190,  189,  189,  188,  191,  190,
190,  186,  195,  193,  194,  190,  194,  194,  197,  186,  194,
190,  194,  190,  192,  192,  191,  191,  190,  197,  192,  189,
191,  193,  191,  135,   67,   41,   36,   38,   35,   36,   36,
37,   40,   34,   34,   36,   37,   31,   30,   31,   30,   29,   30,
33,   35,   36,   34,   35,   31,   35,   36,   34,   37,   34,   35,
31,   32,   32,   27,   23,   28,   30,   30,   29,   26,   21,   29,
27,   27,   30,   37,   42,   30,   33,   30,},
{  188,  188,  187,  188,  182,  183,  187,  187,  184,  184,  189,
184,  191,  198,  192,  185,  164,   88,   39,   39,   27,   31,
36,   29,   35,   37,   32,   32,   31,   31,   33,   33,   36,   31,
26,   34,   32,   31,   31,   31,   29,   29,   38,   31,   36,   34,
```

```
32,   36,   37,   32,   33,   32,   29,   31,   28,   30,   29,   31,
29,   35,   30,   30,   34,   37,   23,   30,   26,   27,   29,   29,
26,   33,   44,  143,  186,  195,  194,  191,  190,  194,  186,  189,
192,  193,  191,  195,  182,  192,  199,  191,  197,  193,  190,
185,  188,  195,  196,  194,  194,  191,  190,  194,  191,  191,
191,  185,  185,  190,  190,  194,  193,  200,  190,  194,  196,
195,  188,  189,  190,  192,  191,  196,  193,  192,  191,  187,
192,  194,  192,  146,   69,   39,   35,   35,   30,   37,   33,
35,   31,   33,   31,   39,   37,   34,   30,   25,   29,   27,   32,
27,   29,   30,   33,   36,   37,   30,   32,   22,   25,   33,   30,
27,   30,   35,   28,   29,   25,   26,   26,   29,   34,   30,   28,
26,   30,   33,   33,   37,   34,   33,   33,},
{  191,  189,  193,  191,  190,  188,  190,  189,  194,  186,  184,
187,  192,  193,  193,  190,  145,   75,   42,   35,   39,   31,
30,   33,   34,   32,   30,   28,   38,   31,   36,   31,   29,   28,
27,   26,   32,   25,   29,   31,   27,   34,   29,   33,   30,   27,
28,   33,   30,   30,   26,   32,   32,   28,   27,   30,   32,   30,
28,   30,   28,   32,   29,   31,   30,   27,   28,   30,   34,   32,
38,   37,   53,  141,  182,  193,  187,  194,  196,  187,  185,  187,
190,  195,  193,  191,  187,  186,  190,  187,  192,  190,  192,
194,  192,  195,  189,  190,  190,  189,  190,  198,  191,  189,
187,  185,  189,  188,  188,  188,  186,  188,  192,  193,  194,
197,  194,  190,  189,  192,  192,  192,  193,  193,  193,  191,
192,  188,  194,  151,   78,   39,   39,   34,   32,   32,   34,
31,   35,   30,   31,   29,   27,   34,   34,   28,   29,   29,   31,
28,   33,   33,   28,   40,   37,   35,   32,   26,   25,   35,   27,
27,   36,   33,   29,   36,   33,   33,   28,   29,   32,   28,   25,
25,   29,   26,   27,   35,   26,   34,   33,},
{  190,  190,  188,  194,  191,  195,  185,  190,  194,  190,  186,
187,  191,  190,  189,  181,  127,   63,   35,   31,   32,   33,
29,   36,   33,   29,   31,   31,   38,   30,   29,   25,   27,   27,
27,   28,   26,   30,   25,   30,   28,   32,   32,   28,   31,   29,
28,   30,   29,   27,   25,   33,   29,   32,   33,   30,   26,   29,
33,   35,   29,   30,   28,   34,   32,   29,   43,   39,   41,   41,
42,   41,   55,  141,  182,  190,  188,  188,  192,  192,  188,  193,
194,  189,  187,  185,  187,  183,  192,  194,  185,  189,  188,
190,  190,  182,  187,  181,  186,  185,  188,  189,  186,  191,
185,  177,  187,  194,  191,  193,  189,  187,  184,  191,  191,
192,  194,  186,  177,  183,  192,  195,  191,  186,  192,  193,
196,  189,  195,  164,   83,   48,   38,   34,   37,   30,   30,
29,   29,   31,   35,   39,   36,   34,   33,   35,   28,   29,   28,
26,   28,   30,   27,   27,   30,   29,   32,   26,   28,   30,   28,
25,   26,   25,   23,   27,   26,   29,   19,   23,   27,   35,   26,
22,   23,   25,   23,   30,   25,   38,   25,},
{  187,  190,  189,  191,  198,  190,  189,  190,  190,  186,  187,
192,  186,  193,  190,  182,  116,   61,   37,   35,   32,   34,
32,   35,   32,   35,   35,   63,   77,   40,   36,   22,   31,   26,
31,   34,   34,   24,   29,   31,   32,   27,   30,   31,   29,   32,
28,   26,   28,   26,   30,   30,   27,   31,   31,   31,   26,   29,
28,   31,   37,   35,   31,   32,   25,   30,   34,   39,   44,   36,
44,   43,   53,  143,  186,  192,  192,  189,  185,  186,  187,  184,
186,  187,  188,  189,  190,  188,  195,  184,  178,  179,  186,
187,  183,  186,  185,  188,  188,  191,  185,  185,  183,  183,
185,  182,  181,  193,  190,  193,  195,  187,  180,  181,  186,
186,  193,  192,  186,  188,  188,  186,  188,  188,  193,  189,
189,  187,  189,  162,   86,   41,   31,   33,   25,   25,   23,
27,   22,   25,   25,   26,   31,   29,   25,   25,   30,   30,   26,
26,   25,   27,   25,   26,   26,   30,   30,   29,   29,   27,   23,
30,   29,   30,   27,   26,   28,   23,   32,   28,   22,   29,   29,
28,   28,   28,   31,   25,   32,   28,   32,},
```

```
{  189,  191,  196,  199,  192,  187,  193,  195,  190,  190,  189,
193,  187,  194,  193,  178,  109,   52,   34,   29,   31,   23,
35,   29,   38,   33,   35,   37,   38,   32,   27,   29,   26,   29,
24,   27,   33,   26,   28,   24,   28,   30,   29,   33,   30,   39,
35,   36,   29,   27,   32,   29,   27,   29,   33,   26,   28,   34,
29,   28,   32,   38,   32,   27,   28,   30,   33,   31,   32,   31,
35,   37,   61,  147,  182,  190,  192,  190,  191,  192,  182,  186,
187,  191,  190,  192,  194,  194,  195,  195,  184,  188,  175,
173,  179,  190,  188,  186,  191,  190,  182,  175,  177,  173,
170,  175,  171,  186,  190,  188,  195,  186,  175,  177,  183,
188,  187,  189,  187,  183,  191,  184,  190,  189,  190,  188,
186,  192,  188,  175,   99,   47,   41,   27,   29,   29,   33,
30,   27,   37,   33,   29,   33,   30,   42,   31,   31,   33,   34,
30,   32,   30,   33,   30,   27,   31,   29,   27,   24,   28,   25,
30,   29,   30,   35,   37,   20,   27,   37,   32,   28,   29,   31,
34,   28,   31,   31,   30,   33,   37,   28,},
{  192,  191,  191,  195,  193,  193,  199,  194,  188,  193,  193,
189,  195,  196,  192,  165,   97,   45,   35,   30,   30,   28,
29,   29,   32,   34,   28,   38,   34,   32,   33,   32,   34,   28,
32,   26,   22,   27,   27,   25,   29,   29,   29,   27,   35,   32,
33,   29,   34,   34,   29,   36,   29,   28,   23,   22,   28,   29,
29,   28,   35,   32,   34,   29,   29,   32,   35,   33,   26,   27,
30,   32,   47,  143,  177,  189,  193,  196,  192,  192,  192,  192,
187,  192,  191,  195,  195,  193,  193,  196,  193,  194,  193,
191,  183,  183,  196,  191,  189,  195,  191,  182,  172,  176,
177,  177,  180,  185,  192,  188,  190,  193,  186,  184,  188,
192,  191,  188,  187,  183,  173,  181,  178,  183,  188,  194,
188,  191,  189,  178,  108,   48,   42,   33,   37,   31,   31,
30,   28,   35,   34,   33,   27,   26,   31,   33,   27,   27,   34,
29,   31,   29,   35,   30,   32,   28,   29,   25,   33,   32,   26,
27,   30,   27,   34,   30,   31,   28,   29,   29,   30,   31,   33,
28,   32,   32,   34,   26,   29,   30,   29,},
{  191,  191,  194,  193,  194,  189,  186,  190,  192,  192,  189,
191,  191,  196,  192,  156,   90,   48,   32,   28,   28,   32,
26,   32,   32,   35,   32,   30,   30,   33,   32,   29,   32,   30,
32,   28,   25,   27,   29,   31,   34,   31,   36,   34,   29,   37,
38,   39,   33,   33,   30,   28,   33,   30,   28,   28,   28,   26,
29,   33,   36,   36,   31,   29,   36,   27,   35,   37,   35,   31,
33,   25,   54,  153,  191,  188,  194,  192,  192,  193,  191,  192,
193,  195,  193,  192,  193,  192,  192,  196,  196,  193,  182,
189,  183,  189,  187,  183,  188,  193,  193,  193,  188,  189,
182,  177,  174,  181,  189,  193,  192,  192,  182,  176,  192,
189,  192,  189,  174,  169,  176,  186,  187,  193,  188,  184,
189,  188,  188,  177,  114,   58,   43,   31,   33,   33,   29,
34,   32,   28,   35,   30,   30,   30,   31,   30,   33,   27,   33,
26,   36,   29,   34,   28,   32,   35,   33,   24,   30,   32,   31,
32,   32,   30,   31,   34,   30,   26,   29,   31,   26,   27,   30,
30,   30,   28,   29,   28,   28,   27,   29,},
{  187,  188,  187,  188,  191,  190,  193,  196,  194,  189,  192,
191,  191,  187,  182,  148,   75,   35,   35,   34,   28,   23,
29,   28,   35,   33,   29,   25,   31,   31,   33,   29,   29,   26,
26,   29,   31,   29,   29,   23,   27,   32,   28,   30,   33,   29,
29,   32,   21,   29,   28,   31,   32,   27,   30,   26,   28,   32,
28,   30,   26,   30,   30,   33,   35,   34,   37,   32,   26,   33,
30,   34,   63,  159,  191,  191,  193,  194,  195,  191,  192,  193,
195,  193,  191,  198,  194,  189,  191,  190,  190,  191,  193,
192,  184,  183,  187,  184,  188,  193,  189,  188,  178,  179,
188,  181,  187,  196,  188,  189,  190,  193,  184,  186,  191,
188,  189,  188,  188,  187,  184,  186,  183,  187,  193,  189,
191,  189,  194,  185,  125,   63,   39,   36,   32,   29,   32,
22,   29,   32,   31,   32,   25,   28,   32,   32,   35,   34,   37,
```

```
33,   32,   32,   28,   27,   30,   27,   31,   36,   31,   28,   27,
28,   30,   33,   34,   27,   24,   30,   33,   32,   34,   32,   26,
29,   27,   29,   27,   30,   23,   34,   31,},
{  192,  186,  174,  169,  176,  180,  192,  196,  193,  196,  194,
190,  188,  179,  170,  125,   61,   36,   34,   32,   23,   22,
24,   30,   26,   33,   28,   31,   25,   28,   25,   24,   29,   33,
32,   26,   24,   23,   27,   29,   25,   29,   24,   29,   25,   34,
33,   30,   27,   32,   29,   26,   26,   29,   25,   30,   30,   34,
33,   28,   26,   29,   32,   35,   29,   32,   35,   31,   36,   31,
40,   37,   71,  165,  187,  195,  194,  192,  194,  192,  192,  192,
196,  191,  188,  190,  195,  193,  197,  189,  192,  189,  186,
188,  191,  194,  190,  189,  186,  191,  192,  190,  191,  186,
181,  178,  185,  188,  180,  186,  189,  190,  182,  183,  184,
186,  181,  193,  185,  186,  179,  176,  184,  189,  198,  190,
190,  180,  179,  175,  140,   66,   41,   34,   31,   32,   34,
27,   35,   32,   31,   33,   29,   32,   23,   29,   28,   36,   25,
33,   33,   31,   29,   32,   28,   25,   31,   30,   27,   30,   39,
35,   32,   30,   33,   36,   31,   29,   29,   30,   31,   28,   27,
31,   33,   31,   28,   29,   31,   28,   30,},
{  193,  184,  178,  185,  188,  193,  195,  197,  192,  190,  193,
189,  183,  181,  167,  122,   53,   38,   26,   27,   28,   27,
23,   31,   31,   30,   30,   29,   27,   32,   30,   29,   31,   29,
30,   31,   21,   30,   29,   32,   34,   30,   31,   27,   32,   33,
34,   44,   39,   32,   31,   33,   36,   29,   31,   30,   36,   30,
31,   29,   34,   34,   34,   31,   32,   38,   34,   32,   32,   32,
31,   36,   71,  162,  189,  196,  196,  192,  197,  192,  192,  195,
194,  199,  194,  188,  188,  193,  193,  195,  196,  194,  193,
195,  194,  185,  188,  190,  191,  195,  188,  193,  188,  195,
189,  188,  182,  187,  187,  184,  189,  190,  185,  189,  189,
184,  187,  195,  181,  176,  184,  193,  194,  196,  194,  188,
182,  178,  184,  185,  148,   77,   46,   34,   25,   32,   32,
30,   31,   32,   29,   29,   35,   30,   27,   29,   33,   30,   33,
32,   35,   36,   33,   29,   31,   31,   35,   37,   35,   30,   30,
27,   28,   34,   26,   32,   28,   29,   28,   28,   32,   28,   24,
27,   28,   28,   22,   29,   27,   31,   31,},
{  191,  187,  194,  189,  189,  185,  191,  197,  193,  191,  193,
188,  187,  183,  167,  112,   56,   43,   36,   29,   30,   30,
29,   34,   27,   32,   36,   35,   29,   31,   32,   31,   35,   30,
31,   33,   30,   31,   32,   26,   27,   29,   29,   27,   30,   27,
37,   37,   34,   40,   32,   31,   31,   31,   37,   30,   32,   33,
32,   35,   35,   31,   38,   36,   37,   40,   35,   28,   33,   37,
43,   35,   70,  168,  191,  191,  193,  191,  190,  194,  195,  192,
192,  194,  194,  189,  194,  188,  192,  201,  190,  195,  196,
193,  193,  190,  191,  193,  190,  194,  191,  187,  186,  190,
187,  184,  181,  184,  179,  184,  191,  185,  191,  194,  188,
189,  188,  187,  193,  182,  181,  189,  193,  190,  193,  184,
187,  196,  184,  184,  158,   70,   48,   32,   28,   29,   28,
34,   33,   25,   25,   26,   28,   31,   29,   28,   28,   32,   28,
34,   33,   30,   29,   31,   34,   24,   28,   34,   27,   29,   32,
30,   30,   28,   26,   31,   35,   34,   29,   30,   28,   27,   30,
30,   24,   30,   31,   34,   34,   42,   37,},
{  192,  191,  189,  187,  190,  194,  194,  192,  191,  190,  188,
189,  193,  193,  180,  108,   52,   36,   39,   35,   33,   33,
31,   33,   32,   36,   35,   34,   32,   32,   34,   33,   30,   30,
34,   31,   30,   31,   33,   30,   30,   29,   25,   30,   29,   24,
36,   40,   34,   33,   44,   33,   34,   23,   32,   32,   27,   30,
31,   31,   29,   31,   32,   35,   29,   32,   33,   34,   39,   32,
36,   44,   87,  172,  189,  193,  191,  193,  195,  196,  191,  194,
189,  193,  191,  190,  197,  192,  197,  194,  190,  194,  190,
193,  199,  190,  192,  194,  195,  193,  193,  184,  190,  187,
195,  193,  186,  189,  183,  184,  189,  190,  190,  189,  185,
```

```
186,  191,  186,  188,  192,  189,  188,  191,  192,  185,  187,
194,  192,  190,  187,  165,   78,   40,   38,   30,   30,   33,
23,   34,   33,   32,   33,   32,   31,   30,   28,   26,   30,   32,
31,   30,   31,   30,   27,   38,   29,   30,   31,   28,   28,   30,
29,   25,   26,   28,   23,   32,   32,   31,   32,   36,   32,   33,
30,   33,   34,   34,   31,   29,   30,   29,},
{  195,  185,  186,  190,  192,  194,  192,  182,  188,  185,  192,
193,  194,  191,  173,   94,   57,   37,   32,   32,   33,   32,
30,   31,   30,   33,   32,   32,   32,   31,   32,   34,   34,   26,
37,   31,   27,   29,   32,   31,   31,   31,   29,   30,   32,   32,
33,   31,   32,   29,   29,   29,   28,   35,   29,   28,   25,   29,
26,   34,   33,   33,   33,   43,   38,   44,   39,   37,   41,   36,
32,   37,   79,  170,  188,  186,  187,  193,  198,  199,  197,  193,
197,  190,  192,  186,  189,  192,  197,  191,  194,  193,  193,
191,  185,  190,  190,  191,  197,  196,  195,  193,  192,  198,
192,  190,  189,  190,  195,  192,  194,  192,  189,  192,  190,
189,  188,  187,  188,  186,  199,  190,  188,  187,  187,  190,
187,  190,  188,  187,  163,   89,   47,   40,   33,   34,   30,
41,   28,   33,   33,   34,   36,   27,   28,   34,   33,   36,   25,
32,   37,   31,   29,   30,   33,   29,   24,   29,   29,   26,   27,
25,   25,   29,   33,   29,   33,   31,   34,   32,   31,   33,   29,
36,   35,   34,   29,   33,   40,   39,   37,},
{  193,  190,  193,  192,  195,  191,  190,  187,  190,  194,  187,
190,  191,  189,  155,   74,   42,   36,   35,   30,   30,   30,
29,   30,   29,   31,   32,   29,   30,   29,   29,   37,   31,   31,
30,   28,   31,   31,   29,   28,   26,   30,   26,   35,   30,   35,
31,   30,   26,   31,   36,   25,   29,   31,   33,   31,   32,   32,
30,   33,   28,   30,   32,   32,   30,   30,   29,   31,   37,   32,
34,   42,   96,  172,  188,  186,  196,  192,  190,  192,  196,  193,
192,  187,  187,  189,  187,  194,  192,  190,  194,  199,  193,
190,  192,  194,  193,  199,  195,  189,  187,  199,  192,  192,
189,  186,  194,  188,  190,  191,  190,  192,  191,  193,  195,
191,  185,  188,  184,  189,  188,  190,  192,  192,  190,  194,
194,  184,  187,  189,  171,  101,   49,   36,   31,   32,   32,
36,   27,   29,   28,   32,   32,   28,   37,   32,   33,   36,   33,
37,   31,   33,   37,   32,   24,   32,   30,   29,   24,   34,   29,
34,   32,   25,   33,   33,   30,   33,   31,   26,   33,   26,   27,
33,   30,   34,   29,   30,   34,   33,   34,},
{  193,  192,  191,  193,  192,  191,  187,  191,  192,  188,  191,
192,  192,  187,  144,   72,   40,   35,   29,   32,   30,   31,
30,   26,   30,   29,   27,   30,   37,   27,   33,   31,   39,   35,
29,   31,   30,   28,   23,   27,   31,   33,   31,   31,   31,   30,
26,   33,   28,   33,   28,   30,   31,   35,   38,   32,   29,   27,
37,   31,   31,   29,   29,   28,   25,   27,   30,   31,   27,   33,
37,   46,   89,  166,  175,  186,  188,  192,  196,  192,  188,  184,
180,  191,  188,  196,  197,  195,  188,  182,  180,  189,  191,
193,  194,  189,  188,  184,  187,  190,  190,  190,  188,  185,
185,  192,  195,  194,  193,  189,  190,  188,  196,  192,  188,
188,  186,  187,  191,  195,  193,  190,  187,  187,  191,  193,
195,  197,  191,  187,  176,  116,   62,   38,   32,   33,   30,
37,   34,   35,   40,   37,   32,   34,   36,   41,   32,   31,   31,
31,   33,   30,   32,   29,   31,   32,   34,   35,   28,   35,   31,
32,   26,   31,   36,   33,   34,   33,   33,   35,   35,   38,   29,
33,   29,   29,   34,   33,   32,   34,   34,}
};



struct header {
      int nr, nc;             /* Rows and columns in the image */
      int oi, oj;             /* Origin */
```

```
};

/*      The IMAGE data structure        */
struct image {
            struct header *info;            /* Pointer to header */
            unsigned char **data;           /* Pixel values */
};
typedef struct image * IMAGE;


struct image  *newimage (int nr, int nc)
{
        struct image  *x;               /* New image */
        unsigned char *ptr;             /* new pixel array */
        int i;

        if (nr < 0 || nc < 0) {
                printf ("Error: Bad image size (%d,%d)\n", nr, nc);
                return 0;
        }

/*      Allocate the image structure    */
        x = (struct image  *) malloc( sizeof (struct image) );
        if (!x) {
                printf ("Out of storage in NEWIMAGE.\n");
                return 0;
        }

/*      Allocate and initialize the header      */

        x->info = (struct header *)malloc( sizeof(struct header) );
        if (!(x->info)) {
                printf ("Out of storage in NEWIMAGE.\n");
                return 0;
        }
        x->info->nr = nr;
      x->info->nc = nc;
        x->info->oi = 0;
      x->info->oj = 0;

/*      Allocate the pixel array        */

        x->data = (unsigned char **)malloc(sizeof(unsigned char
*)*nr);


/* Pointers to rows */
        if (!(x->data)) {
                printf ("Out of storage in NEWIMAGE.\n");
                return 0;
        }

        for (i=0; i<nr; i++)
        {
          x->data[i] = (unsigned char *)malloc(nc*sizeof(unsigned
char));
          if (x->data[i]==0)
          {
                printf ("Out of storage. Newimage - row %d\n", i);
                exit(1);
          }
```

```
        }
        return x;
}

/* Fraction of pixels that should be above the HIGH threshold */
float ratio = 0.1;
int WIDTH = 0;
int    PBM_SE_ORIGIN_COL=0, PBM_SE_ORIGIN_ROW=0;


/*function prototypes*/
int trace (int i, int j, int low, IMAGE im,IMAGE mag, IMAGE ori);
float gauss(float x, float sigma);
float dGauss (float x, float sigma);
float meanGauss (float x, float sigma);
void hysteresis (int high, int low, IMAGE im, IMAGE mag, IMAGE
oriim);
void canny (float s, IMAGE im, IMAGE mag, IMAGE ori);
void seperable_convolution (IMAGE im, float *gau, int width, float
**smx, float **smy);
void dxy_seperable_convolution (float** im, int nr, int nc, float
*gau, int width, float **sm, int which);
void nonmax_suppress (float **dx, float **dy, int nr, int nc, IMAGE
mag, IMAGE ori);
void estimate_thresh (IMAGE mag, int *low, int *hi);

int range (IMAGE im, int i, int j);
float ** f2d (int nr, int nc);
void freeimage (struct image  *z);
void sys_abort (int val, char *mess);
void copy (IMAGE *a, IMAGE b);
void CopyVarImage (IMAGE *a, IMAGE *b);
IMAGE Output_PBM (IMAGE image, char *filename);


int main ()
{
    XTmrCtr XPS_Timer ;
    //typedef unsigned long long XTime;
    XTime start ;
    XTime end ;
    unsigned int diff ;
    //void XTime_SetTime(XTime Xtime);
    //void XTime_GetTime(XTime *Xtime);

      int i,j,k=CLASS;
      float s=4.0;
      int low=5,high=3;
      IMAGE im, magim, oriim;


      printf ("Parameters: HIGH: %d LOW %d Sigma %f\n", high, low,
s);
      printf ("\nPBM file class %d size %d columns X %d rows
Max=%d\n", CLASS, COLUMNS, ROWS, MAX);

       im = (IMAGE)newimage (ROWS, COLUMNS);
       im->info->oi = PBM_SE_ORIGIN_ROW;
       im->info->oj = PBM_SE_ORIGIN_COL;
       PBM_SE_ORIGIN_ROW = 0;
       PBM_SE_ORIGIN_COL = 0;
```

```
       for (i=0; i<ROWS; i++)
          for (j=0; j<COLUMNS; j++)
            im->data[i][j] = imgdata[i][j];

 XTmrCtr_Initialize(&XPS_Timer,XPAR_XPS_TIMER_0_DEVICE_ID) ;


    start=0;
    XTime_SetTime(start) ;
    XTime_GetTime(&start) ;


    XTime_GetTime(&end) ;



    printf ("Time : %L\n",end);
    putnum( end ) ;
    print("\r\n");
    i = (unsigned int) end-start ;
    putnum( i ) ;
    print("\r\n");
    i = (unsigned int) (end>>32) - (start>>32)  ;
    putnum( i ) ;
    print("\r\n");

    //delay :
    for (i=0; i<8000000; ++i) ;

/* Create local image space */
      magim = newimage (im->info->nr, im->info->nc);
      if (magim == NULL)
      {
        printf ("Out of storage: Magnitude\n");
        exit (1);
      }

      oriim = newimage (im->info->nr, im->info->nc);
      if (oriim == NULL)
      {
        printf ("Out of storage: Orientation\n");
        exit (1);
      }
    XTmrCtr_SetResetValue(&XPS_Timer,0,0x00000000) ;
    XTmrCtr_Reset(&XPS_Timer,0) ;
    XTmrCtr_Start(&XPS_Timer,0) ;

/* Apply the filter */
      canny (s, im, magim, oriim);
XTmrCtr_Stop(&XPS_Timer,0) ;
    i = XTmrCtr_GetValue(&XPS_Timer,0) ;
    putnum(i) ;
    print("\r\n");

       XTmrCtr_SetResetValue(&XPS_Timer,0,0x00000000) ;
    XTmrCtr_Reset(&XPS_Timer,0) ;
    XTmrCtr_Start(&XPS_Timer,0) ;
// Hysteresis thresholding of edge pixels
      hysteresis (high, low, im, magim, oriim);
XTmrCtr_Stop(&XPS_Timer,0) ;
    i = XTmrCtr_GetValue(&XPS_Timer,0) ;
```

```
    putnum(i) ;
    print("\r\n");
      for (i=0; i<WIDTH; i++)
        for (j=0; j<im->info->nc; j++)
          im->data[i][j] = 255;

      for (i=im->info->nr-1; i>im->info->nr-1-WIDTH; i--)
        for (j=0; j<im->info->nc; j++)
          im->data[i][j] = 255;

      for (i=0; i<im->info->nr; i++)
        for (j=0; j<WIDTH; j++)
          im->data[i][j] = 255;

      for (i=0; i<im->info->nr; i++)
        for (j=im->info->nc-WIDTH-1; j<im->info->nc; j++)
          im->data[i][j] = 255;


      Output_PBM (im, "canny.pgm");

      printf ("Output file is:\n");
      printf ("  canny.pgm - edge-only image\n");
}

float norm (float x, float y)
{
      return (float) sqrt ( (double)(x*x + y*y) );
}

void canny (float s, IMAGE im, IMAGE mag, IMAGE ori)
{
      int width;
      float **smx,**smy;
      float **dx,**dy;
      int i,j,k,n;
      float gau[MAX_MASK_SIZE], dgau[MAX_MASK_SIZE], z;

/* Create a Gaussian and a derivative of Gaussian filter mask */
      for(i=0; i<MAX_MASK_SIZE; i++)
      {
        gau[i] = meanGauss ((float)i, s);
        if (gau[i] < 0.005)
        {
            width = i;
            break;
        }
        dgau[i] = dGauss ((float)i, s);
      }

      n = width+width + 1;
      WIDTH = width/2;
      printf ("Smoothing with a Gaussian (width = %d) ...\n", n);

      smx = f2d (im->info->nr, im->info->nc);
      smy = f2d (im->info->nr, im->info->nc);

/* Convolution of source image with a Gaussian in X and Y directions
*/
      seperable_convolution (im, gau, width, smx, smy);
```

```
/* Now convolve smoothed data with a derivative */
      printf ("Convolution with the derivative of a Gaussian...\n");
      dx = f2d (im->info->nr, im->info->nc);
      dxy_seperable_convolution (smx, im->info->nr, im->info->nc,
             dgau, width, dx, 1);
      free(smx[0]); free(smx);

      dy = f2d (im->info->nr, im->info->nc);
      dxy_seperable_convolution (smy, im->info->nr, im->info->nc,
             dgau, width, dy, 0);
      free(smy[0]); free(smy);

/* Create an image of the norm of dx,dy */
      for (i=0; i<im->info->nr; i++)
        for (j=0; j<im->info->nc; j++)
        {
            z = norm (dx[i][j], dy[i][j]);
            mag->data[i][j] = (unsigned char)(z*MAG_SCALE);
        }

/* Non-maximum suppression - edge pixels should be a local max */

      nonmax_suppress (dx, dy, (int)im->info->nr, (int)im->info->nc,
mag, ori);

      free(dx[0]); free(dx);
      free(dy[0]); free(dy);
}

/*      Gaussian        */
float gauss(float x, float sigma)
{
    float xx;

    if (sigma == 0) return 0.0;
    xx = (float)exp((double) ((-x*x)/(2*sigma*sigma)));
    return xx;
}

float meanGauss (float x, float sigma)
{
      float z;

      z = (gauss(x,sigma)+gauss(x+0.5,sigma)+gauss(x-0.5,sigma))/3.0;
      z = z/(PI*2.0*sigma*sigma);
      return z;
}

/*      First derivative of Gaussian    */
float dGauss (float x, float sigma)
{
      return -x/(sigma*sigma) * gauss(x, sigma);
}

/*      HYSTERESIS thersholding of edge pixels. Starting at pixels
with a
      value greater than the HIGH threshold, trace a connected
sequence
      of pixels that have a value greater than the LOW threhsold.
*/
```

```
void hysteresis (int high, int low, IMAGE im, IMAGE mag, IMAGE oriim)
{
      int i,j,k;

      printf ("Beginning hysteresis thresholding...\n");
      for (i=0; i<im->info->nr; i++)
        for (j=0; j<im->info->nc; j++)
          im->data[i][j] = 0;

      if (high<low)
      {
        estimate_thresh (mag, &high, &low);
        printf ("Hysteresis thresholds (from image): HI %d LOW %D\n",
high, low);
      }
/* For each edge with a magnitude above the high threshold, begin
   tracing edge pixels that are above the low threshold.
*/

      for (i=0; i<im->info->nr; i++)
        for (j=0; j<im->info->nc; j++)
          if (mag->data[i][j] >= high)
            trace (i, j, low, im, mag, oriim);

/* Make the edge black (to be the same as the other methods) */
      for (i=0; i<im->info->nr; i++)
        for (j=0; j<im->info->nc; j++)
          if (im->data[i][j] == 0) im->data[i][j] = 255;
          else im->data[i][j] = 0;
}

/*      TRACE - recursively trace edge pixels that have a threshold >
the low
      edge threshold, continuing from the pixel at (i,j).
*/

int trace (int i, int j, int low, IMAGE im,IMAGE mag, IMAGE ori)
{
      int n,m;
      char flag = 0;

      if (im->data[i][j] == 0)
      {
        im->data[i][j] = 255;
        flag=0;
        for (n= -1; n<=1; n++)
        {
          for(m= -1; m<=1; m++)
          {
            if (i==0 && m==0) continue;
            if (range(mag, i+n, j+m) && mag->data[i+n][j+m] >= low)
            if (trace(i+n, j+m, low, im, mag, ori))
            {
                flag=1;
                break;
            }
          }
          if (flag) break;
        }
        return(1);
      }
```

```
      return(0);
}

void seperable_convolution (IMAGE im, float *gau, int width,
            float **smx, float **smy)
{
      int i,j,k, I1, I2, nr, nc;
      float x, y;

      nr = im->info->nr;
      nc = im->info->nc;

      for (i=0; i<nr; i++)
        for (j=0; j<nc; j++)
        {
          x = gau[0] * im->data[i][j]; y = gau[0] * im->data[i][j];
          for (k=1; k<width; k++)
          {
            I1 = (i+k)%nr; I2 = (i-k+nr)%nr;
            y += gau[k]*im->data[I1][j] + gau[k]*im->data[I2][j];
            I1 = (j+k)%nc; I2 = (j-k+nc)%nc;
            x += gau[k]*im->data[i][I1] + gau[k]*im->data[i][I2];
          }
          smx[i][j] = x; smy[i][j] = y;
        }
}

void dxy_seperable_convolution (float** im, int nr, int nc,  float
*gau,
            int width, float **sm, int which)
{
      int i,j,k, I1, I2;
      float x;

      for (i=0; i<nr; i++)
        for (j=0; j<nc; j++)
        {
          x = 0.0;
          for (k=1; k<width; k++)
          {
            if (which == 0)
            {
            I1 = (i+k)%nr; I2 = (i-k+nr)%nr;
            x += -gau[k]*im[I1][j] + gau[k]*im[I2][j];
            }
            else
            {
            I1 = (j+k)%nc; I2 = (j-k+nc)%nc;
            x += -gau[k]*im[i][I1] + gau[k]*im[i][I2];
            }
          }
          sm[i][j] = x;
        }
}

void nonmax_suppress (float **dx, float **dy, int nr, int nc,
            IMAGE mag, IMAGE ori)
{
      int i,j,k,n,m;
      int top, bottom, left, right;
      float xx, yy, g2, g1, g3, g4, g, xc, yc;
```

```
      for (i=1; i<mag->info->nr-1; i++)
      {
        for (j=1; j<mag->info->nc-1; j++)
        {
          mag->data[i][j] = 0;

/* Treat the x and y derivatives as components of a vector */
          xc = dx[i][j];
          yc = dy[i][j];
          if (fabs(xc)<0.01 && fabs(yc)<0.01) continue;

          g  = norm (xc, yc);

/* Follow the gradient direction, as indicated by the direction of
   the vector (xc, yc); retain pixels that are a local maximum. */

          if (fabs(yc) > fabs(xc))
          {

/* The Y component is biggest, so gradient direction is basically
UP/DOWN */
            xx = fabs(xc)/fabs(yc);
            yy = 1.0;

            g2 = norm (dx[i-1][j], dy[i-1][j]);
            g4 = norm (dx[i+1][j], dy[i+1][j]);
            if (xc*yc > 0.0)
            {
            g3 = norm (dx[i+1][j+1], dy[i+1][j+1]);
            g1 = norm (dx[i-1][j-1], dy[i-1][j-1]);
            } else
            {
            g3 = norm (dx[i+1][j-1], dy[i+1][j-1]);
            g1 = norm (dx[i-1][j+1], dy[i-1][j+1]);
            }

          } else
          {

/* The X component is biggest, so gradient direction is basically
LEFT/RIGHT */
            xx = fabs(yc)/fabs(xc);
            yy = 1.0;

            g2 = norm (dx[i][j+1], dy[i][j+1]);
            g4 = norm (dx[i][j-1], dy[i][j-1]);
            if (xc*yc > 0.0)
            {
            g3 = norm (dx[i-1][j-1], dy[i-1][j-1]);
            g1 = norm (dx[i+1][j+1], dy[i+1][j+1]);
            }
            else
            {
            g1 = norm (dx[i-1][j+1], dy[i-1][j+1]);
            g3 = norm (dx[i+1][j-1], dy[i+1][j-1]);
            }
          }

/* Compute the interpolated value of the gradient magnitude */
          if ( (g > (xx*g1 + (yy-xx)*g2)) &&
```

```
             (g > (xx*g3 + (yy-xx)*g4)) )
          {
            if (g*MAG_SCALE <= 255)
            mag->data[i][j] = (unsigned char)(g*MAG_SCALE);
            else
            mag->data[i][j] = 255;
            ori->data[i][j] = atan2 (yc, xc) * ORI_SCALE;
          } else
          {
            mag->data[i][j] = 0;
            ori->data[i][j] = 0;
          }

        }
      }
}

void estimate_thresh (IMAGE mag, int *hi, int *low)
{
      int i,j,k, hist[256], count;

/* Build a histogram of the magnitude image. */
      for (k=0; k<256; k++) hist[k] = 0;

      for (i=WIDTH; i<mag->info->nr-WIDTH; i++)
        for (j=WIDTH; j<mag->info->nc-WIDTH; j++)
          hist[mag->data[i][j]]++;

/* The high threshold should be > 80 or 90% of the pixels
      j = (int)(ratio*mag->info->nr*mag->info->nc);
*/
      j = mag->info->nr;
      if (j<mag->info->nc) j = mag->info->nc;
      j = (int)(0.9*j);
      k = 255;

      count = hist[255];
      while (count < j)
      {
        k--;
        if (k<0) break;
        count += hist[k];
      }
      *hi = k;

      i=0;
      while (hist[i]==0) i++;

      *low = (*hi+i)/2.0;
}

/*      Check that a pixel index is in range. Return TRUE(1) if so.
*/

int range (IMAGE im, int i, int j)
{
        if ((i<0) || (i>=im->info->nr)) return 0;
        if ((j<0) || (j>=im->info->nc)) return 0;
        return 1;
}
```

```
float ** f2d (int nr, int nc)
{
        float **x, *y;
        int i;

        x = (float **)calloc ( nr, sizeof (float *) );
        if (x == 0)
        {
          printf (stderr, "Out of storage: F2D.\n");
          exit (1);
        }

        for (i=0; i<nr; i++)
        {
          x[i] = (float *) calloc ( nc, sizeof (float)  );
          if (x[i] == 0)
          {
            printf (stderr, "Out of storage: F2D %d.\n", i);
            exit (1);
          }
        }
        return x;
}


void freeimage (struct image  *z)
{
/*      Free the storage associated with the image Z    */
        int i;

        if (z != 0)
        {
          for (i=0; i<z->info->nr; i++)
              free (z->data[i]);
           free (z->info);
           free (z->data);
           free (z);
        }
}

void sys_abort (int val, char *mess)
{
        printf (stderr, "**** System library ABORT %d: %s ****\n",
                        val, mess);
        exit (2);
}

void copy (IMAGE *a, IMAGE b)
{
        CopyVarImage (a, &b);
}


void CopyVarImage (IMAGE *a, IMAGE *b)
{
        int i,j;

        if (a == b) return;
        if (*a) freeimage (*a);
```

```
        *a = newimage ((*b)->info->nr, (*b)->info->nc);
        if (*a == 0) sys_abort (0, "No more storage.\n");

        for (i=0; i<(*b)->info->nr; i++)
          for (j=0; j< (*b)->info->nc; j++)
            (*a)->data[i][j] = (*b)->data[i][j];
        (*a)->info->oi = (*b)->info->oi;
        (*a)->info->oj = (*b)->info->oj;
}

IMAGE Output_PBM (IMAGE image, char *filename)
{
      FILE *f;
      int i,j,k, perline;
      char buf1[64];

//    strcpy (buf1, filename);
      if (image->info->nc > 20) perline = 20;
       else perline = image->info->nc-1;
      //f = sysace_fopen( buf1, "w" );
//    f = fopen (buf1, "w");
//    if (f == 0) sys_abort (0, "Can't open output file.");

      printf ("P2\n#origin %d %d\n",image->info->oj,image->info->oi);
      printf ("%d %d %d\n", image->info->nc, image->info->nr, 255);
      k = 0;
      for (i=0; i<image->info->nr; i++)
        for (j=0; j<image->info->nc; j++)
        {
            printf ("%d ", image->data[i][j]);
            k++;
            if (k > perline)
            {
              printf (f, "\n");
              k = 0;
            }
        }
      printf ("\n");
      //fclose (f);
      return image;
}
```