Πολυτεχνειο Κρητης
Σχολη Ηλεκτρονικων Μηχανικων και Μηχανικων Υπολογιστων

"ΑΝΑΠΤΥΞΗ ΠΑΡΑΛΛΗΛΩΝ ΑΡΧΙΤΕΚΤΟΝΙΚΩΝ ΑΝΑΔΙΑΤΑΣΣΟΜΕΝΗΣ ΛΟΓΙΚΗΣ ΓΙΑ ΟΙΚΟΛΟΓΙΚΑ ΜΟΝΤΕΛΑ"

Μαραγκός Κωνσταντίνος

εξεταστικη επιτροπη

Επιβλέπων:	Καθηγητής Απόστολος Δόλλας
Μέλος Επιτροπής:	Καθηγητής Διονύσιος Πνευματικάτος
Μέλος Επιτροπής:	Αναπληρωτής Καθηγητής Ιωάννης Παπαευσταθίου

Χανιά, Κρήτη
Αύγουστος 2013

Περίληψη

Η χρήση των οικολογικών μοντέλων είναι το σημαντικότερο εργαλείο για τη μελέτη και κατανόηση της πολυπλοκότητας των φαινομένων που λαμβάνουν μέρος στα οικοσυστήματα. Τα μοντέλα αυτά χαρακτηρίζονται από μεγάλο υπολογιστικό φορτό και υψηλή κατανάλωση ισχύος, καθώς στηρίζονται στη συνεχή επίλυση συστημάτων διαφορών και απευθύνονται σε πολύ μεγάλες περιοχές μελέτης. Προκειμένου να επιτευχθεί μια ρεαλιστική προσομοίωση, τα οικολογικά μοντέλα λαμβάνουν υπόψη τους πολυάριθμους φυσικούς παράγοντες και ιδιότητες του χώρου που μελετάται. Η χρήση εξελιγμένων μοντέλων επιτρέπει στους επιστήμονες να τρέχουν μεγάλες προσομοιώσεις και να πραγματοποιούν πολλά πειράματα, τα οποία ενώ στην πραγματικότητα θα διαρκούσαν αιώνες, μπορούν γίνουν μέσα σε λίγα λεπτά μέσω ενός υπολογιστικού μοντέλου.

Με την παρούσα εργασία στοχεύουμε στην επίτευξη ενός βελτιωμένου οικολογικού μοντέλου που ασχολείται με τη μελέτη των υδάτων(*θαλασσών, ποταμών, λιμνών*). Με την δυνατότητα της αναδιάταξης που μας παρέχουν οι FPGAs μπορούμε να προσαρμόσουμε το υλικό στις ανάγκες του μοντέλου, έχοντας καλύτερη αξιοποίηση των πόρων και μείωση της κατανάλωσης ισχύος. Επίσης, αξιοποιώντας τον παραλληλισμό σε συνδιασμό με μια καλά σχεδιασμένη αρχιτεκτονική που ανταποκρίνεται στις απαιτήσεις του προβλήματος, μπορούμε να πετύχουμε σημαντική επιτάχυνση στη συνολική χρονική απόκριση.

Πιο συγκεκριμένα, αντικείμενο της διπλωματικής εργασίας, είναι η επίτευξη καλύτερης απόδοσης του μοντέλου Princeton Ocean Model(*POM*) μέσω της βελτίωσης της πιο βαριάς συνάρτησης του, με τη χρήση των FPGAs. Το POM είναι ένα ωκεανικό μοντέλο που ασχολείται με την μηχανική ρευστών σε τρισδιάστατα πλέγματα και χαρακτηρίζεται από υψηλή ακρίβεια υπολογισμών και μεγάλο φόρτο δεδομένων. Η συνάρτηση που επιλέξαμε να βελτιώσουμε καταλαμβάνει το 45% του συνολικού φόρτου του μοντέλου και μέσω της υλοποίησης της σε FPGA , αποσκοπούμε στην καλύτερη χρονική απόδοση της επεξεργασίας του χώρου και τη μείωση της κατανάλωσης ισχύος.

Τέλος, αποδείξαμε ότι με μια πλακέτα Virtex 5 (*XC5VLX330*) στα 150 MHz, μπορούμε να πετύχουμε επιτάχυνση στην εκτέλεση της συνάρτησης 3,17 φορές πιο γρήγορα σε σχέση με ένα σύστημα με επεξεργαστή 8 πυρήνων στα 2,7 GHz και 12 Mb μνήμη.

Ευχαριστίες

Πρωτίστως, θα ήθελα να ευχαριστήσω τον επιβλέπων καθηγητή μου Απόστολο Δόλλα, για την πρακτική και ηθική καθοδήγηση του καθ΄ όλη τη διάρκεια της διεκπεραίωσης αυτής της διπλωματικής εργασίας, και κυρίως για την εμπιστοσύνη που έδειξε στο πρόσωπό μου. Θα ήθελα ακόμα να τον ευχαριστήσω, γιατί μέσω της διδασκαλίας του μου μετέδωσε το μεράκι και την αγάπη για τη δουλειά που κάνει, δίνοντας μου έμπνευση και όρεξη να ασχοληθώ με το συγκεκριμένο τομέα. Επίσης, ευχαριστώ τον καθηγητή Διονύσιο Πνευματικάτο και τον αναπληρωτή καθηγητή Ιωάννη Παπαευσταθίου, που ως μέλη της εξεταστικής επιτροπής συμμετέχουν στην επιτυχή ολοκλήρωση του προπτυχιακού κύκλου σπουδών μου.

Το επόμενο πρόσωπο που θέλω να ευχαριστήσω είναι ο Δρ. Ευριπίδης Σωτηριάδης, για τις πρακτικές συμβουλές του πάνω στην εργασία αλλά και στο γεγονός ότι ήταν πάντα διαθέσιμος όποτε τον χρειαζόμουν, ανοιχτός να μοιραστεί τις σκέψεις και τους προβληματισμούς μου, να με συμβουλέψει και να με ενθαρρύνει να προχωρήσω στις επιλογές μου. Ακόμα θέλω να ευχαριστήσω όλα τα άτομα του εργαστηρίου τα οποία μου προσέφεραν ένα πολύ ευχάριστο και ευνοικό κλίμα εργασίας, κάνοντας πιο ευχάριστη την καθημερινότητα μου στον εργασιακό χώρο. Ένα πολύ μεγάλο ευχαριστώ αξίζει ο μεταπτυχιακός φοιτητής Χρήστος Ρουσόπουλος για τον πολύτιμο χρόνο που μου αφιέρωσε και την βοήθεια του στο κομμάτι διασύνδεσης της αρχιτεκτονικής με το Convey και τον έλεγχο λειτουργίας του συνολικού συστήματος. Επίσης, ευχαριστώ τον μεταπτυχιακό φοιτητή Σταύρο Αποστολάκη για τη στήριξη του και συγκεκριμένα για τη βοήθεια του στο κομμάτι των compilers.

Στην επιτυχή περάτωση όχι μόνο αυτής της εργασίας, αλλά και του κύκλου σπουδών, έπαιξαν πολύ σημαντικό ρόλο τα κοντινά μου άτομα, τα οποία με βοήθησαν να "ξεφεύγω" από τον καθημερινό φόρτο εργασίας. Θα ήθελα, λοιπόν, να ευχαριστήσω όλους τους φίλους μου στα Χανιά και στην Ελευσίνα, για όλες τις ωραίες και ξέγνοιαστες στιγμές που περάσαμε (και θα περάσουμε) μαζί. Ειδικά ένα μεγάλο ευχαριστώ θέλω να πω στην Ελισάβετ Παλογιαννίδη που πίστεψε σε μένα, με στήριζε και βρισκόταν πάντα δίπλα μου σε όλη τη διάρκεια αυτού του δύσκολου αγώνα, και για όλες τις πολύτιμες συμβουλές που με βοήθησαν να βελτιώσω τον τρόπο δουλειάς μου.

Το μεγαλύτερο ευχαριστώ όμως αξίζει η οικογένεια μου, που με στηρίζει στις επιλογές μου και βρίσκεται πάντα δίπλα μου προσφέροντας μου την ασφάλεια και την δύναμη να πραγματοποιώ τους στόχους μου. Επομένως, η συγκεκριμένη δουλειά είναι αφιερωμένη σε αυτούς για όλα όσα μου έχουν προσφέρει.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

POM	Princeton Ocean Model
CFD	Computanional Fluid Dynamics
CFL	Courant–Friedrichs–Lewy
FPGA	Field Programmable Gate Array
GPU	Graphics Processing Unit

Κεφάλαιο 1

Εισαγωγή

1.1 Κίνητρο

Κατά τη διάρκεια των τελευταίων δεκαετιών, τα οικολογικά μοντέλα ([29]) έχουν γίνει ένα σημαντικό εργαλείο για την περιβαλλοντική μελέτη και λήψη αποφάσεων. Η μελέτη του περιβάλλοντος ήταν πάντα ένα αναπόσπαστο κομμάτι έρευνας που εκτείνεται σε πολλούς επιστημονικούς τομείς. Ειδικά τα τελευταία χρόνια, η τεχνολογία έχει σημειώσει αλματώδη εξέλιξη και χρησιμοποιείται από πολλούς επιστήμονες οι οποίοι προσπαθούν να συμβάλλουν στην έγκαιρη πρόγνωση ποικίλων ανεπιθύμητων φαινομένων που απειλούν ολόκληρα οικοσυστήματα([16]).

Οι συνεχώς αυξανόμενες απαιτήσεις που παρουσιάζονται στη μελέτη της συμπεριφοράς των οικοσυστημάτων, οδηγούν στη δημιουργία όλο και πιο πολύπλοκων μοντέλων προσομοίωσης. Τα μοντέλα αυτά προκειμένου να πετύχουν με καλή ακρίβεια αξιόπιστα αποτελέσματα και να αποδώσουν μια ρεαλιστική συμπεριφορά, λαμβάνουν υπόψην τους πολυάριθμους παράγοντες, σχετικούς με τα χαρακτηριστικά του εκάστοτε συστήματος. Ο μεγάλος αριθμός των παραγόντων, καθώς και η πολύπλοκη φύση των αλγορίθμων προσομοίωσης, καθιστούν αυτά τα μοντέλα ως υπολογιστικά "βαριά". Η έγκαιρη πρόγνωση μπορεί να είναι καθοριστική για την προστασία, τόσο του ίδιου του περιβάλλοντος όσο και του ανθρώπου.

Επίσης, προσομοιώνοντας τις συμπεριφορές των οικοσυστημάτων([14]), δίνεται η δυνατότητα έρευνας και μελέτης φαινομένων που αφορούν την εξέλιξη οικοσυστημάτων στο πέρασμα του χρόνου, τα οποία για να συμβούν στην πραγματικότητα θα έπρεπε να περάσουν ολόκληρες μέρες ή ακόμα και χρόνια. Κατά συνέπεια, αναπτύσσονται συνεχώς νέες μέθοδοι προκειμένου να επιταχυνθεί η διαδικασία μελέτης των φαινομένων αυτών, διατηρώντας ωστόσο την ποιότητα των αποτελεσμάτων. Ο συνδυασμός της ταχύτητας στην πρόγνωση των φαινομένων, αλλά και της εξασφάλισης των ορθών αποτελεσμάτων, αποτελεί μείζον ζήτημα σε πολλούς επιστημονικούς κλάδους. Έτσι παρατηρούνται προσπάθειες ανάπτυξης και βελτίωσης, τόσο σε επίπεδο λογισμικού όσο και σε επίπεδο υλικού([27]). Πολλά επιστημονικά κέντρα επενδύουν υπέρογκα χρηματικά ποσά για την αγορά ειδικών μηχανημάτων ή υπερυπολογιστών προκειμένου να καλύψουν αυτές τις αυστηρές απαιτήσεις. Πέραν του υψηλού κόστους, υπάρχει και το ζήτημα της καταλάλωσης ισχύος. Μηχανήματα τόσο υψηλών προδιαγραφών καταναλώνουν υψηλή υπολογιστική ισχύ, με αποτέλεσμα όταν η λειτουργία τους γίνεται σε μόνιμη βάση(*πολύ μεγάλα χρονικά διατήματα*) να τα καθιστά μη οικονομικά συμφέρουσες λύσεις.

Συνοψίζοντας, οι περισσότερες έρευνες που γίνονται αποσκοπούν στο να βελτιώσουν αυτά τα αδύναμα σημεία. Υπάρχουν ήδη αρκετά αξιόλογα μοντέλα προσομοιώσης, με πολύ καλή

προσέγγιση αποτελεσμάτων. Για αυτό το λόγο πλέον η έρευνα δεν προχωρά τόσο στην εύρεση νέων αποδοτικότερων μοντέλων, αλλά στην βελτίωση των ήδη υπαρχόντων, επιταχύνοντας τα και μειώντας όσο είναι δυνατό την κατανάλωση της ισχύς.

1.2 Αντικείμενο Διπλωματικής

Η δουλειά που παρουσιάζεται σε αυτή τη διπλωματική εργασία, αφορά τη βελτίωση του ωκεανικού μοντέλου POM([12]) με την χρήση των FPGAs. Το συγκεκριμένο μοντέλο μελετά την συμπεριφορά των υδάτων (*θαλασσών, ποταμών, λιμνών*) και στηρίζεται στην επίλυση συστημάτων διαφορών σε τρισδιάστατα πλέγματα. Λόγω των απαιτήσεων για υψηλής ακρίβειας προσομοιώσεις, λαμβάνει υπόψην πολλούς φυσικούς παράγοντες και ιδιότητες του χώρου που μελετάται εκτελώντας πολύπλοκους αλγορίθμους, κάτι που το καθιστούν υπολογιστικά "βαρύ". Η προσπάθεια που γίνεται είναι η βελτίωση της χρονικής απόκρισης του μοντέλου σε μεγάλο φόρτο δεδομένων, σε συνδιασμό με τη μείωση της υπολογιστικής ισχύς. Για να επιτύχουμε τη βελτίωση αυτή, μελετήσαμε το μοντέλο και έπειτα στηριχτήκαμε στην υλοποίηση με FPGAs λόγω των δυνατοτήτων που μας προσφέρει η αναδιατασσόμενη λογική.

Συγκεκριμένα, σε πρώτο στάδιο παρουσιάζεται η μελέτη του μοντέλου, η οποία περιλαμβάνει την εύρεση των πιο κρίσιμων λειτουργιών του, και την επιλογή της κατάλληλότερης αριθμητικής για την ακριβή αναπαράσταση των αποτελεσμάτων. Σε δεύτερο στάδιο παρουσιάζεται η κατασκευή της αρχιτεκτονικής που περιλαμβάνει το σχεδιασμό των αριθμητικών μονάδων, τις μνήμες, τις μονάδες ελέγχου και το συγχρονισμό του συνολικού συστήματος. Τέλος, αφού γίνει ο έλεγχος και η επικύρωση της αρχιτεκτονικής, παρουσιάζεται ο διασύνδεση του υλικού με το λογισμικό φτιάχνοντας ένα υβριδικό σύστημα. Μετά την κατασκευή γίνεται η σύγκριση της απόδοσης του σε σχέση με το αρχικό μοντέλο.

1.3 Συνεισφορά

Μέσω της συγκερκιμένης διπλωματικής εργασίας παρουσιάζουμε την εντριβή των FPGAs σε ένα ακόμα επιστημονικό κλάδο, τον κλάδο των οικοσυστημάτων. Η βασική συνεισφορά αυτής της εργασίας είναι η επίτευξη της επιτάχυνσης ενός γνωστού ωκεανικού μοντέλου (*POM*) χρησιμοποιώντας FPGAs, με ταυτόχρονη μείωση της κατανάλωσης ισχύος.

Για όλους τους λόγους που αναφέραμε στην ενότητα 1.1 οι FPGAs και τα πλεονεκτήματα που παρουσιάζουν([17]), μπορούν να συμβάλλουν στην καλύτερη απόδοση των οικολογικών μοντέλων, καθώς ανταποκρίνονται πλήρως στις ανάγκες τους. Η δυναστότητα της επαναπροσαρμογής ακριβώς στις ιδιαιτερότητες του προβλήματος,η υψηλή ταχύτητα υλοποίησης, ο παραλλήλισμός και η χαμηλή κατανάλωση ισχύος είναι τα όπλα των FPGAs απέναντι σε τόσο απαιτητικές εφαρμογές, συμβάλλοντας στην ανάπτυξη και βελτίωση του κλάδου της οικολογίας.

1.4 Δομή της Διπλωματικής

Η δομή που ακολουθείται στη διπλωματική είναι η παρακάτω :

Στο κεφάλαιο 2 γίνεται μια εισαγωγή στα οικολογικά μοντέλα και στις αρχές τις οποίες στηρίζονται. Αναφέρονται κάποια βασικά χαρακτηριστικά του POM και παρουσιάζονται παρόμοιες δουλειές που έγιναν πάνω σε αυτά, με την χρήση των FPGAs και GPUs.

Το κεφάλαιο 3 αναφέρεται στην μελέτη του POM. Πρώτο στάδιο, είναι η διαδικασία του

profiling, η εύρεση του πιο υπολογιστικά βαριού τμήματος και η ανάλυση αυτού. Έπειτα ακολουθεί μια προσομοίωση του τμήματος αυτού, και τέλος γίνεται μια έρευνα σχετικά με την ακρίβεια των αποτελεσμάτων του μοντέλου και την διαδικασία επιλογής της αριθμητικής που θα χρησιμοποιήσουμε για την υλοποίηση του.

Το κεφάλαιο 4 αναφέρεται στην αποτύπωση της πιο απαιτητικής συνάρτησης του μοντέλου στο hardware. Παρουσιάζονται οι φάσεις της κατασκευής των αριθμητικών μονάδων, της κατασκευής των μνημών, της υλοποίησης των μονάδων ελέγχου και του συγχρονισμού του συνολικού συστήματος.

Το κεφάλαιο 5 αναφέρεται στον έλεγχο της σωστής λειτουργίας της αρχιτεκτονικής και την επαλήθευση των αποτελεσμάτων.

Το κεφάλαιο 6 αναφέρεται στην εγκατάσταση της αρχιτεκτονικής στο Convey, τις μεθόδους μεταφοράς δεδομένων από την μνήμη και τη σύγκριση του συνολικού χρόνου εκτέλεσης της υλοποίησης στην FPGA με το αρχικό μοντέλο σε Fortran.

Στο κεφάλαιο 7 γίνεται αναφορά στα συμπεράσματα που προέκυψαν από την περάτωση της διπλωματικής εργασίας και τη μελλοντική δουλειά με την οποία μπορεί να επεκταθεί.

Κεφάλαιο 2

Σχετική Δουλειά

2.1 Οικολογικά Μοντέλα

2.1.1 Εισαγωγή

Η ανθρώπινη δραστηριότητα και η ευημερία, εξαρτώνται και έχουν ενσωματωθεί με τη λειτουργία των οικοσυστημάτων και των υπηρεσιών που προσφέρουν. Στόχος μας, είναι να κατανοήσουμε αυτές τις βασικές λειτουργίες του οικοσυστήματος με την χρήση μαθηματικών και εννοιολογικών μοντέλων, ανάλυση συστημάτων, θερμοδυναμική, προσομοιώσεις σε υπολογιστές και την οικολογική θεωρία. Αυτές οι μέθοδοι μοντελοποίησης, μπορούν να εφαρμοστούν σε ένα ευρύ φάσμα θεμάτων που κυμαίνονται από την βασική οικολογία, στην ανθρώπινη οικολογία, στα κοινωνικο – οικολογικά συστήματα. Ένα οικολογικό μοντέλο, είναι συνήθως μια μαθηματική αναπαράσταση ενός οικολογικού συστήματος, το οποίο μελετάται με σκοπό την κατανόηση της συμπεριφοράς του πραγματικού συστήματος.

Τα οικολογικά μοντέλα([29], [4]) σχεδιάζονται συνδιάζοντας γνωστές οικολογικές σχέσεις (π.χ. η σχέση του φωτός του ήλιου και του νερού σχετικά με το ρυθμό της φωτοσύνθεσης) και δεδομένα που συλλέγονται από επιτόπιες παρατηρήσεις. Αυτά τα μοντελοποιημένα συστήματα στη συνέχεια μελετούνται, προκειμένου να γίνουν προβλέψεις σχετικά με τη δυναμική συμπεριφορά του πραγματικού συστήματος. Συχνά, η μελέτη των ανακριβειών μέσα στο μοντέλο (σε σύγκριση με εμπειρικές παρατηρήσεις) οδηγεί στη δημιουργία υποθέσεων, σχετικά με τις πιθανές οικολογικές σχέσεις που δεν είναι ακόμη γνωστές ή πλήρως κατανοητές.

Τα μοντέλα αυτά, επιτρέπουν στους ερευνητές την προσομοίωση μεγάλης κλίμακας πειραμάτων που θα ήταν πάρα πολύ δαπανηρό ή ανήθικο να εκτελεστούν σε ένα πραγματικό οικοσύστημα. Επιτρέπουν επίσης την προσομοίωση μιας διαδικασίας που θα διαρκούσε αιώνες στην πραγματικότητα, να γίνει μέσα σε λίγα λεπτά μέσω ενός υπολογιστικού μοντέλου.

Τα οικολογικά μοντέλα καλύπτουν εφαρμογές σε ένα ευρύ φάσμα επιστημονικών κλάδων, όπως είναι η διαχείριση των φυσικών πόρων, η οικοτοξικολογία και η περιβαλλοντική υγεία, η γεωργία, η διατήρηση της άγριας φύσης κ.τ.λ..

2.1.2 Τύποι Μοντέλων

Υπάρχουν δύο βασικά είδη οικολογικών μοντέλων : (1) τα αναλυτικά μοντέλα και (2) τα προσομοίωσης/υπολογιστικά μοντέλα. Τα αναλυτικά μοντέλα, είναι συχνά πιο πολύπλοκα μαθηματικά και λειτουργούν καλύτερα όταν ασχολούνται με σχετικά απλά (συχνά γραμμικά) συστήματα, ιδίως εκείνων που μπορούν να περιφραφούν με ακρίβεια από ένα σύνολο μαθημα-

τικών εξισώσεων, των οποίων η συμπεριφορά είναι γνωστή. Τα μοντέλα προσομοίωσης από την άλλη πλευρά, χρησιμοποιούν αριθμητικές τεχνικές για να λύσουν τα προβλήματα για τα οποία οι αναλυτικές λύσεις είναι ανέφικτες ή αδύνατες. Τα μοντέλα προσομοίωσης τείνουν να χρησιμοποιούνται ευρέως, και γενικά θεωρούνται πιο ρεαλιστικά, ενώ τα αναλυτικά μοντέλα εκτιμώνται για τη μαθηματική "κομψότητα" τους και την ερμηνευτική "δύναμη" τους.

2.1.3 Σχεδίαση Μοντέλων

Η διαδικασία του σχεδιασμού του μοντέλου, ξεκινά με την περιγραφή του προβλήματος που πρέπει να επιλυθεί, καθώς και τους στόχους του μοντέλου. Τα οικολογικά συστήματα αποτελούνται από έναν τεράστιο αριθμό βιοτικών(*ζωντανοί οργανισμοί*) και αβιοτικών παραγόντων(*πχ νερό, θερμοκρασία, αέρας, χωμα κλπ*) που αλληλεπιδρούν μεταξύ τους με τρόπους που είναι συχνά απρόβλεπτοι, ή τόσο περίπλοκοι ώστε είναι αδύνατο να ενσωματωθούν σε ένα υπολογιστικό μοντέλο. Λόγω αυτής της πολυπλοκότητας, τα οικολογικά μοντέλα συνήθως απλοποιούν τα συστήματα που μελετούν σε περιορισμένο αριθμό τμημάτων που είναι καλά κατανοητά, και θεωρούνται σχετικά με το πρόβλημα όπου το μοντέλο προορίζεται για την επίλυση του.

Η διαδικασία της απλοποίησης, μειώνει συνήθως ένα οικοσύστημα σε ένα μικρό αριθμό μεταβλητών καταστάσης([7]) και μαθηματικών συναρτήσεων που περιγράφουν τη φύση των σχέσεων μεταξύ τους. Ο αριθμός των τμημάτων του οικοσυστήματος που έχουν ενσωματωθεί στο μοντέλο, περιορίζεται με τη συγκέντρωση των παρόμοιων διαδικασιών που αντιμετωπίζονται ως μία λειτουργική μονάδα.

Μετά τον καθορισμό των τμημάτων που θα μοντελοποιηθούν και τις σχέσεις που συνδέονται μεταξύ τους, ένας άλλος σημαντικός παράγοντας της δομής του οικολογικού μοντέλου, είναι η αναπαράσταση του χώρου που χρησιμοποιείται. Ιστορικά, τα μοντέλα αγνοούσαν το θέμα του χώρου. Ωστόσο ο χωρικός παράγοντας αποτελεί ένα σημαντικό κομμάτι του προβλήματος, καθώς παρατηρείται ότι σε διαφορετικά χωρικά περιβάλλοντα οδηγούμαστε σε διαφορετικά αποτελέσματα. Αυστηρά χωρικά μοντέλα προσπαθούν να ενσωματώσουν ένα ετερογενή χωρικό περιβάλλον στο μοντέλο. Ένα χωρικό μοντέλο, είναι αυτό που έχει μία ή περισσότερες μεταβλητές κατάστασης, οι οποίες σχετίζονται με τον χώρο ή μπορεί να σχετίζονται με άλλες χωρικές μεταβλητές.

2.1.4 Επικύρωση Μοντέλων

Μετά την κατασκευή, τα μοντέλα επικυρώνονται για να εξασφαλιστεί ότι τα αποτελέσματα είναι αποδεκτά, σαφή και ρεαλιστικά. Μία μέθοδος επικύρωσης είναι η δοκιμή του μοντέλου με πολλαπλές σειρές δεδομένων, που είναι ανεξάρτητες από το ίδιο το σύστημα που μελετάται. Αυτό είναι σημαντικό δεδομένου ότι ορισμένες είσοδοι μπορεί να οδηγήσουν σε ένα ελλατωματικό μοντέλο με έξοδο σωστών αποτελεσμάτων.

Μια άλλη μέθοδος επικύρωσης είναι η σύγκριση των εξόδων του μοντέλου με τα δεδομένα που συλλέγονται από επιτόπιες παρατηρήσεις. Οι ερευνητές συχνά καθορίζουν εκ των προτέρων κατά πόσο μια διαφορά μπορεί να είναι αποδεκτή μεταξύ των εξόδων ενός μοντέλου και εκείνων που υπολογίζονται από τις παρατηρήσεις δεδομένων του υπό εξέταση χώρου.

2.1.5 Computanional Fluid Dynamics

Εισαγωγή

Ο κλάδος του Computanional Fluid Dynamics(*CFD*),είναι ένας κλάδος της μηχανικής των ρευστών που χρησιμοποιεί αριθμητικές μεθόδους και αλγορίθμους για την επίλυση και ανάλυση προβλημάτων που αφορούν τη μηχανική ρευστών([8]). Η χρήση του είναι ευρέως διαδεδομένη στον τομέα των οικολογικών μοντέλων. Για την εκτέλεση των υπολογισμών χρησιμοποιούνται υπολογιστές, οι οποίοι είναι απαραίτητοι για την προσομοίωση της αλληλεπίδρασης των υγρών και των αερίων σε επιφάνειες που ορίζονται από οριακές συνθήκες. Τα συστήματα αυτά καθορίζονται από τις εξισώσεις Navier-Stokes([32]) και μπορούν να λύσουν αρκετά πολύπλοκα πολυδιάστατα προβλήματα, τα οποία μπορεί να είναι γραμμικά ή μη γραμμικά. Με τη συνεχή έρευνα που γίνεται πάνω στον τομέα, εβρίσκονται συνεχώς νέες μεθόδοι βελτιώσης της ακρίβειας και της ταχύτητας ακόμα και για τις πιο περίπλοκες προσομοιώσεις.

Σε αυτό τον κλάδο στηρίζονται όλα τα οικολογικά μοντέλα που αφορούν τη μελέτη των υδάτων. Συνεπώς και τα ωκεανικά μοντέλα όπως το POM έχουν υιοθετήσει αυτές τις αρχές και ακολουθούν τους βασικούς κανόνες για την επίλυση των προβλημάτων τους.

Μεθοδολογία

Η μεθοδολογία που ακολουθείται για κάθε προσέγγιση στηρίζεται στις εξής αρχές:

1. Κατά τη διάρκεια της προεπεξεργασίας
   - Ορίζεται η γεωμετρία του προβλήματος(*φυσικά όρια*)
   - Ο όγκος που καταλαμβάνεται από το ρευστό διαιρείται σε διακριτά κελιά(*ομοιόμορφα ή ανομοιόμορφα πλέγματα*).
   - Ορίζεται η φυσική μοντελοποίηση. Για παράδειγμα οι εξισώσεις των κινήσεων, ενθαλπίας, ακτινοβολίας κ.τ.λ.
2. Ορίζονται οι οριακές συνθήκες. Αυτό περιλαμβάνει τον καθορισμό της συμπεριφοράς του ρευστού και των ιδιοτήτων του στα όρια του προβλήματος. Για παροδικά προβλήματα, επίσης ορίζονται οι οριακές συνθήκες.
3. Όταν ξεκινήσει η προσομοίωση οι εξισώσεις επιλύονται απαναληπτικά ως μια σταθερή κατάσταση ή παροδικά.
4. Τέλος ένας postprocessor χρησιμοποιείται για την ανάλυση και την οπτικοποίηση του προκύπτοντος αποτελέσματος.

Αλγόριθμοι Επίλυσης

Η διακριτοποίηση στο χώρο παράγει ένα σύστημα συνήθων διαφορικών εξισώσεων για τα ασταθή προβλήματα, και αλγεβρικών εξισώσεων για τα σταθερά προβλήματα. Για μεγάλα συστήματα και ιδιαίτερα τα τρισδιάστατα, είναι συνήθως δύσκολο να χρησιμοποιηθούν άμεσες μαθηματικές λύσεις, λόγω του μεγέθους τους. Για αυτό το λόγο, χρησιμοποιούνται επαναληπτικές μέθοδοι. Αυτές οι μέθοδοι προσπαθούν να πετύχουν αριθμητική σύγκλιση των αποτελεσμάτων από time step σε time step(*χρονικό διάστημα που μεσολαβεί μέχρι να ξεκινήσει ο επόμενος υπολογισμός*), διορθώνοντας αναλόγως το σφάλμα των μετρήσεων με βάση ορισμένα σημεία αναφοράς των πραγματικών μετρησεων.

Μια κλασική τεχνική για την επιτάχυνση της επίλυσης πολύπλοκων συστημάτων είναι η χρήση των multigrids (*πλέγματα*)([20], [31], [33]). Η βασική ιδέα των multigrids είναι η ταχύτερη σύγκλιση μιας επαναληπτικής μεθόδου με τη γενική διόρθωση από time step σε time step. Τα multigrids έχουν το πλεονέκτημα της ασυμπτωτικά βέλτιστης απόδοσης σε πολλά προβλήματα. Μπορούν να εφαρμοστούν σε συνδυασμό με οποιαδήποτε από τις κοινές τεχνικές διακριτοποίησης, και θεωρούνται γενικά από τις ταχύτερες μεθόδους. Σε αντίθεση με άλλες μεθόδους, μπορούν να διαχειριστούν πιο αποδοτικά αυθαίρετες περιοχές και οριακές συνθήκες και δεν εξαρτώνται από την διαχωρισημότητα ή άλλες ιδιότητες των εξισώσεων. Χρησιμοποιούνται ευρέως για την επίλυση περισσότερο περίπλοκων, μη συμμετρικών και μη γραμμικών συστημάτων εξισώσεων, σε δύο ή περισσότερες διαστάσεις. Μάλιστα, το μοντέλο με το οποίο ασχολείται η διπλωματική εργασία(POM) χρησιμοποιεί αυτή την τεχνική.

Σκοπιμότητα των FPGAs σε CFD Εφαρμογές

Πολλά προβλήματα τα οποία τρέχουν σε υπερυπολογιστές ή clusters απαιτούν μέρες για να ληφθεί ένας πλήρης υπολογισμός με καλά συγκλίνουσες λύσεις. Ως εκ τούτου, στη βιομηχανία υπάρχει η πρόκληση για την αναζήτηση ισχυρών υπολογιστικών πόρων, για την προσομοίωση και βελτίωση περίπλοκων εφαρμογών σε ένα εύλογο χρονικό διάστημα. Η χρήση των FPGAs είναι μια αρκετά διαδεδομένη και ελκυστική επιλογή για την επίλυση CFD εφαρμογών [30].

Σύμφωνα με το [9]η επιτάχυνση στο υλικό δίνει καλύτερα αποτελέσματα, όσον αφορά τη συνολική επιτάχυνση και το value for money, όταν εφαρμόζεται σε προβλήματα στα οποία:

- Μια μεγάλη ποσότητα του υπολογιστικού φόρτου συγκεντρώνεται σε ένα μικρό τμήμα ολόκληρου του προγράμματος και οι υπολογισμοί πρέπει να εκτελούνται αρκετές φορές για ένα τεράστιο σύνολο δεδομένων. (Συνθήκη A)

- Οι χρόνοι επικοινωνίας και κλήσεων I/O είναι μικροί σε σχέση με το συνολικό υπολογιστικό κόστος. (Συνθήκη B)

Η χρήση των FPGAs επιτρέπει το χτίσιμο ενός προσωρινού προσαρμοσμένου κυκλώματος για την επίλυση ενός προβλήματος CFD (*ή ένος μέρος του*). Δεδομένου ότι το μέγεθος, ο αριθμός και το είδος των μονάδων λειτουργίας (π.χ. προσθέτες, αφαιρέτες, πολλαπλασιαστές, διαιρέτες) καθορίζονται από τον προγραμματιστή, οι αλγόριθμοι μπορούν να παραλληλοποιηθούν σε επίπεδο εντολών, έτσι ώστε πολλές από αυτές τις εντολές να εκτελούνται στον ίδιο κύκλο ρολογιού.

Μεγάλα προβλήματα όπως π.χ. τα ωκεανικά μοντέλα, απαιτούν αρκετές εκατοντάδες υπολογισμούς, και κάθε υπολογισμός επαναλαμβάνεται εκατομμύρια φορές σε μια προσομοίωση. Για τέτοιου είδους προβλήματα, μέσω pipeline σχεδιασμών μπορεί να αξιοποιηθεί καλύτερος παραλληλισμός. Αυτός ο παραλληλισμός προέρχεται από το γεγονός ότι πολλές μονάδες υλικού μπορούν να εκτελούν διάφορες λειτουργίες ταυτόχρονα.(Συνθήκη A)

Από την άλλη μεριά, σε πολλές αριθμητικές λύσεις παρουσιάζεται το φαινόμενο του locality των δεδομένων. Αυτό σημαίνει πως ένας υπολογισμός που πρέπει να γίνει σε μια ορισμένη θέση εξαρτάται μόνο από δεδομένα τα οποία βρίσκονται σε μια μικρή γειτονιά γύρω από αυτή. Αυτό είναι ιδιαίτερα ενδιαφέρον για την υλοποίηση του παραλληλισμού, όπου ολόκληρη η υπολογιστική περιοχή αποσυντίθεται σε συγκεκριμένο αριθμό υποπεριοχών και κάθε υποπεριοχή έχει ανατεθεί σε ξεχωριστό επεξεργαστή. Εκεί η επικοινωνία μεταξύ των επεξεργαστών είναι απαραίτητη διότι οι λύσεις για τις υποπεριοχές αλληλοεξαρτούνται μεταξύ τους, αλλά μόνο στα όρια των υποπεριοχών. Αυτός είναι ο λόγος για τον οποίο το κόστος της επικοινωνίας πρέπει να είναι μικρό συγκριτικά με το συνολικό κόστος.(Συνθήκη B)

Γενικά, υπάρχει μια σειρά χαρακτηριστικών των συστημάτων βασισμένων σε FPGAs που τα καθιστούν κατάλληλα για την επιτάχυνση κρίσιμων λειτουργιών σε διαφόρου είδους προβλημάτα.

1. Τα διαγράμματα ροής ελέγχου και δεδομένων για τους inner-loop υπολογισμούς μπορούν πολύ εύκολα να αντιστοιχιστούν σε pipeline υλοποιήσεις υλικού. Με το πάντρεμα της FPGA και μιας υψηλού εύρους ζώνης(high-bandwidth) εξωτερικής μνήμης είναι δυνατό να επιτευχθεί μια απόδοση αρκετών GFLOPS σε σχέση με τα εκατοντάδες MFLOPS από ένα συμβατικό επεξεργαστή.

2. Οι αριθμητικές μέθοδοι που εφαρμόζονται για την επίλυση των εξισώσεων που διέπουν τη ροή του προγράμματος, περιλαμβάνουν μόνο πολλαπλασιασμούς, προσθέσεις, διαιρέσεις και άλλες απλές λειτουργίες, ακόμα και σε περίπλοκες περιπτώσεις.

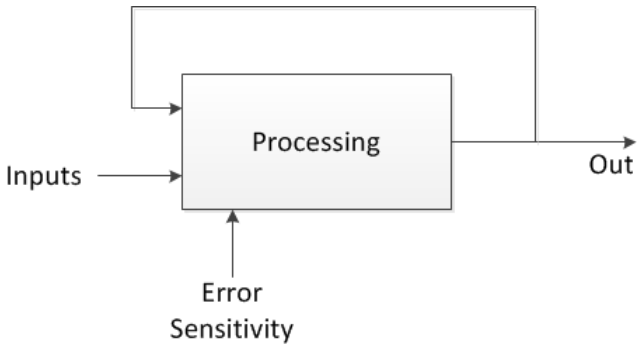
3. Η cache των δεδομένων μπορεί να σχεδιαστεί και να βελτιστοποιηθεί για να ταιριάζει με τα ιδιαίτερα χαρακτηριστικά των CFD αλγορίθμων. Η δυνατότητα συντονισμού της cache με το ταίριασμα στα συγκεκριμένα χαρακτηριστικά λειτουργίας επιτρέπει να κρατάει την pipeline σχεδίαση με τους inner-loop υπολογισμούς συνεχώς απασχολημένη.

4. Το συνολικό εύρος ζώνης της μνήμης και η ποσότητα της διαθέσιμης μνήμης μπορεί να ρυθμιστεί στο συγκεκριμένο σύστημα ώστε να ταιριάζουν οι ανάγκες του αλγορίθμου καλά με το επιθυμητό μέγεθος των πλεγμάτων.

2.2 Πειραματική Προσομοίωση Οικολογικών Μοντέλων

Για την καλύτερη κατανόηση της λειτουργίας των οικολογικών μοντέλων, χτίσαμε δικά μας εικονικά μοντέλα και πειραματιστήκαμε πάνω σε αυτά. Όλα τα πειράματα στηρίζονταν στην επίλυση συστημάτων μαθηματικών εξισώσεων πάνω σε πλέγματα. Η επίλυση γινόταν με επαναληπτικό τρόπο, διορθώνοντας τα σφάλματα των μετρήσεων κάθε φορά, μέχρι να έχουμε μια συγκλίνουσα κατάσταση. Κατά τη διάρκεια μελέτης των πειραμάτων, είχαμε τη δυνατότητα επεξεργασίας όλων των παραμέτρων που απάρτιζαν το πρόβλημα. Τέτοιοι παράμετροι ήταν οι τιμές των είσοδων, ο αριθμός των εισόδων, το μέγεθος του πλέγματος, οι συνιστώσες του χώρου που λάμβανε μέρος το πείραμα, το time step, η ευαισθησία του σφάλματος και η ακρίβεια των αποτελεσμάτων. Έγιναν πολλά πειράματα, αρχικά σε μονοδιάστατους και έπειτα σε δισδιάστατους χώρους, συνδιάζοντας όλες τις παραμέτρους που αναφέρθηκαν παραπάνω. Κάνοντας αυτή τη διαδικασία, μπορέσαμε να κατανοήσουμε τον τρόπο λειτουργίας και την πολυπλοκότητα των οικολογικών μοντέλων. Δεν θα αναφερθούμε σε πολλές λεπτομέρειες πάνω στα πειράματα που έγιναν, απλά θα γίνει μία σύντομη αναφορά σε κάποια από αυτά και θα παρουσιάσουμε τη γενική δομή τους.

Για την επίλυση και την επεξεργασία των πειραμάτων, στηριχτήκαμε στη γενική δομή που χρησιμοποιείται από κάθε οικολογικό μοντέλο. Η δομή παρουσιάζεται στο σχήμα 2.1.

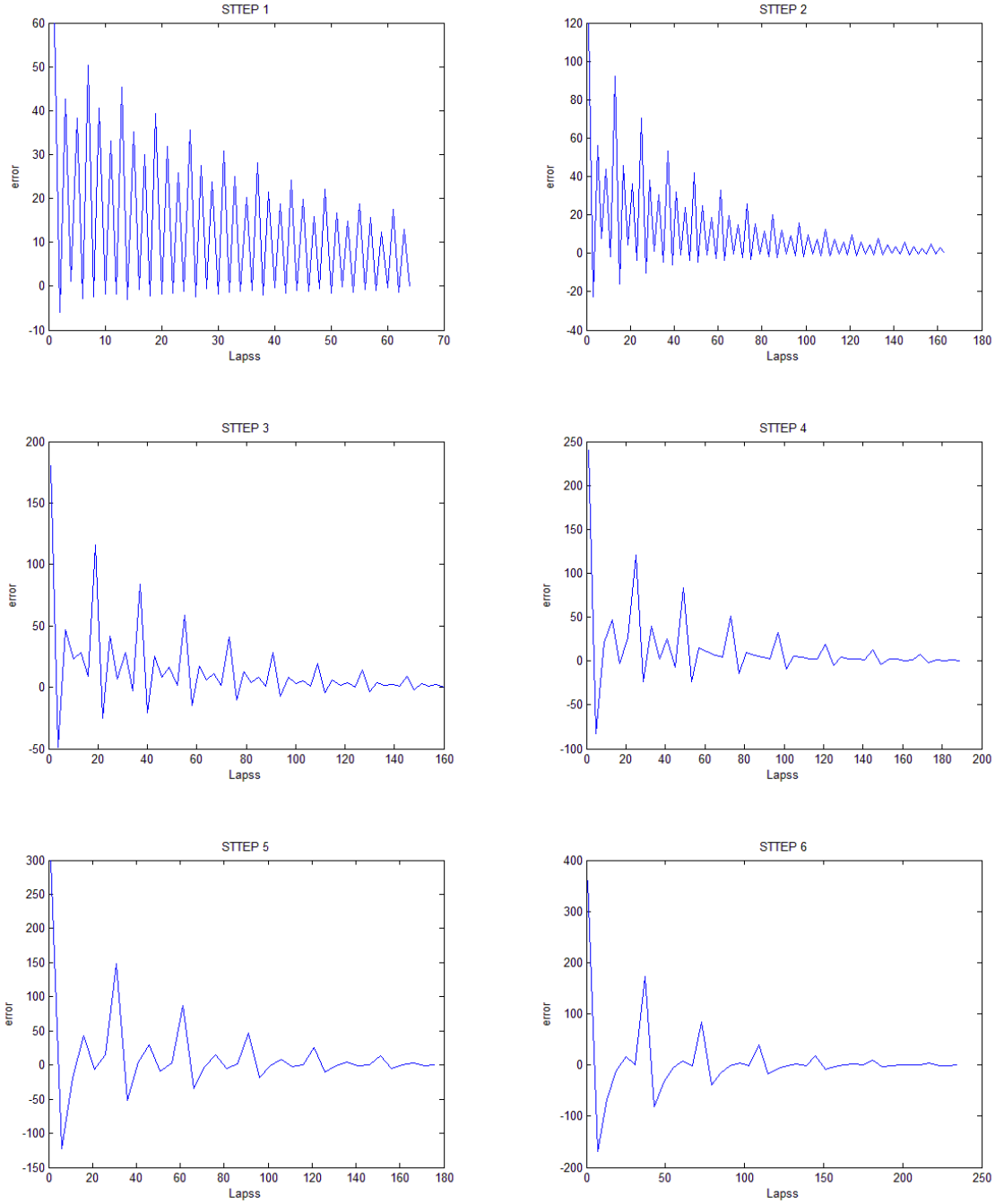


Σχήμα 2.1: Γενική δομή επεξεργασίας οικολογικών μοντέλων.

Αρχικά δίνουμε τις εισόδους του συστήματος βάση των οποίων θα γίνει η επεξεργασία και την ευαισθησία του επιτρεπτού σφάλματος. Όσο πιο αυστηρό σφάλμα βάζαμε, τόσο πιο αυστηρό κάναμε το μοντέλο μας, με αποτέλεσμα η επίλυση του να διαρκεί περισσότερο χρονικό διάστημα. Τα αποτελέσματα του όμως, διέπονταν από καλύτερη ακρίβεια σε σχέση με το να βάζαμε ένα πιο "ελαστικό" σφάλμα. Η επίλυση των διαφορικών συστημάτων, γινόταν επαναληπτικά με ρυθμό που καθοριζόταν απο το time step που ορίζαμε. Όσο καλύτερη ανάλυση έχει το πλέγμα μας (*μικρό time step*) τόσοι περισσότεροι υπολογισμοί λάμβαναν μέρος για ένα συσκεκριμένο χρονικό διάστημα. Αποτέλεσμα ήταν να έχουμε υψηλό υπολογιστικό φόρτο, αλλά η συγκλιση του αποτελέσματος γινόταν σε μικρότερο αριθμό απαναλήψεων. Αντίστοιχα, με τη χρήση μεγαλύτερου time step είχαμε λιγότερο υπολογιστικό φόρτο, αλλά χρειαζόμασταν περισσότερες επαναλήψεις για να πετύχουμε σύγκλιση. Επίσης, και το μέγεθος του πλέγματος παίζει σημαντικό ρόλο στον υπολογιστικό φόρτο και το χρόνο απόκρισης του μοντέλου, ειδικά αν συνδιαστεί με υψηλή ευαισθησία σφάλματος και μικρό time step.

Κάθε έξοδος του μοντέλου, έμπαινε ως είσοδος πάλι στο μοντέλο ετσί ώστε να μπορούμε να μελετήσουμε την απόκλιση που είχαμε σε σχέση με το πραγματικό σύστημα. Έτσι, κατά τη χρονική εξέλιξη του πειράματος, είχαμε μιας μορφής ανάδραση σύμφωνα με την οποία προσαρμόζαμε την επεξεργασία του μοντέλου στις απαιτήσεις του προβλήματος, ώστε οι επόμενες προβλέψεις να είναι πιο ακριβείς.

Στην εικόνα 2.2 βλέπουμε το ρυθμό σύγκλισης ενός απλού μοντέλου που επιλύει γραμμικές εξισώσεις σε ένα σταθερό πλέγμα, συναρτήσει του time step που έχουμε ορίσει. Όπως φένεται, για μικρό time step έχουμε ταχύτερη σύγκλιση (*όσον αφορά τον αριθμό των επαναλήψεων*), και καθώς το αυξάνουμε παρατηρούμε την ταυτόχρονη αύξηση των επαναλήψεων που χρειάζονται για τη σύγκλιση του. Γενικά, δεν σημαίνει ότι με μικρό time step θα έχουμε και ένα γρηγορότερο μοντέλο, μπορεί να καταλήξει να είναι πολύ αργό σε σχέση με την επιλογή ενός μεγαλύτερου time step. Η επιλογή του time step, καθώς και της ευαισθησίας του σφάλματος εξαρτάται από την πολυπλοκότητα και το μέγεθος του προβλήματος. Συνήθως όταν έχουμε πολύ μεγάλα προβλήματα, για να μπορέσουμε να ανταποκριθούμε στις χρονικές απαιτήσεις τους, βάζουμε πιο "χαλαρά" κριτήρια (*π.χ. μεγαλύτερο time step*) ή χρησιμοποιούμε πιο ισχυρά μηχανήματα ή και τα δύο μαζί.



Σχήμα 2.2: Η χρονική απόκριση σύγκλισης ενός πειραματικού γραμμικού μοντέλου συναρτήσει του time step. Ο οριζόντιος άξονας αναπαριστά τον αριθμό των επαναλήψεων και ο κάθετος το σφάλμα μέτρησης. Όσο μεγαλώνει το time step τόσο αυξάνονται οι επαναλήψεις προκειμένου να πετύχουμε σύγκλιση.

Πείραμα 1

Αυτό το πείραμα αφορά ένα δισδιάστατο πλέγμα, του οποίου κάθε σημείο αντιπροσωπεύει έναν κόμβο με μια τυχαία τάση. Οι τιμές που έχει στα άκρα του είναι συγκεκριμένες και

δεν αλλάζουν κατά τη διάρκεια της προσομοίωσης. Κάθε κόμβος του πλέγματος πρέπει να υπακούει στους νόμους του Kirchhoff. Αυτό που θέλουμε, είναι να φτιάξουμε μια προσομοίωση σύμφωνα με την οποία οι τιμές των τάσεων των κόμβων να συγκλίνουν, ώστε όλοι οι κόμβοι να υπακούν στους νόμους του Kirchhoff. Η διαδικασία των υπολογισμών γίνεται με βάση ένα επιτρεπτό όριο σφάλματος. Χρησιμοποιώντας την πληροφορία του κατά πόσο ξεπερνιέται αυτό το όριο, στηρίζονται οι επόμενοι υπολογισμοί ώστε στην μετέπειτα επανάληψη να προσπαθήσουν να συγκλίνουν.

Στο συγκεκριμένο πρόβλημα, πειραματιστήκαμε με την ευαισθησία του σφάλματος, το εύρος τιμών των τάσεων και το μέγεθος του πλέγματος. Για συγκεκριμένους συνδιασμούς ευαισθησίας σφάλματος, τιμών και μεγέθους πλέγματος πέρναμε μια τελείως διαφορετική απόκριση. Άλλες φορές το πείραμα μπορεί να έτρεχε για πολύ μεγάλο χρονικό διάστημα και άλλες για πολύ μικρό. Πολλές φορές μάλιστα μπορεί να πέφταμε σε ταλαντώσεις, με αποτέλεσμα τις ατέρμων επαναλήψεις δίχως κάποια σύγκλιση(*π.χ. ένα μικρό πλέγμα, με μεγάλη διακύμανση τιμών και αρκετά αυστηρό σφάλμα*).

Συμπέρασμα όλων αυτών, ήταν η βαθειά κατανόηση της μελέτης των απαιτήσεων ενός προβλήματος πριν ξεκινήσουμε να χτίσουμε κάποιο μοντέλο προσομοίωσης. Η πολυπλοκότητα των εξισώσεων, ο αριθμός των παραμέτρων, η ακρίβεια των αποτελεσμάτων, η χρονική απόκριση, το μέγεθος του υπό εξέταση χώρου κ.α., παίζουν καθοριστικό ρόλο στο πως θα προσεγγιστεί ένα πρόβλημα.

Στο σχήμα 2.3 φένεται ένα μικρό πλέγμα (*9x9*) από τα πειράματα μας, με τυχαίες ενδιάμεσες τιμές και αγκυροβολημένα άκρα. Έπειτα, στο σχήμα 2.4 φένεται το αποτέλεσμα της σύγκλισης ύστερα από ορισμένο αριθμό επαναλήψεων

```
x =

     0     2     3     2     2     4     3     3     5
     4     2     4     4     2     3     1     1     3
     4     4     1     3     2     1     2     1     4
     2     3     1     3     2     3     3     3     2
     1     1     4     1     4     3     4     1     2
     1     4     1     4     4     4     1     2     2
     4     2     4     1     2     1     1     4     3
     3     1     4     3     2     3     2     1     1
     5     1     1     1     2     1     4     4     0
```

Σχήμα 2.3: Πλέγμα Coulomb 9x9 με αγκυροβολημένες τις γωνίες του και τυχαίες τιμές τάσεων για τους υπόλοιπους κόμβους.

x =

0	0.8343	1.5820	2.4719	2.8610	3.6210	3.7295	4.0973	5.0000
0.9777	1.5450	2.1094	2.8125	3.0518	3.3528	3.7719	4.0468	4.0468
1.8539	2.3438	2.4719	2.2888	2.6367	3.2959	3.3947	3.5361	3.7719
2.5749	2.9297	2.1973	1.5625	1.6875	2.6367	2.6822	2.8968	2.8610
3.3528	2.6367	2.6367	2.4414	2.5000	1.8750	1.9531	2.4414	1.9775
3.5361	3.0899	2.8125	2.4719	2.1094	1.9531	1.0000	1.5625	2.5000
4.2433	3.9290	3.6621	2.8125	2.9297	2.1973	1.5625	1.8750	1.5820
4.3705	4.0233	3.8624	3.6621	3.5156	2.6367	2.0000	1.3733	0.8899
5.0000	4.2681	4.3165	3.3528	2.8610	2.6367	1.8539	0.9777	0

Σχήμα 2.4: Αποτέλεσμα σύγκλισης του σχήματος 2.3.

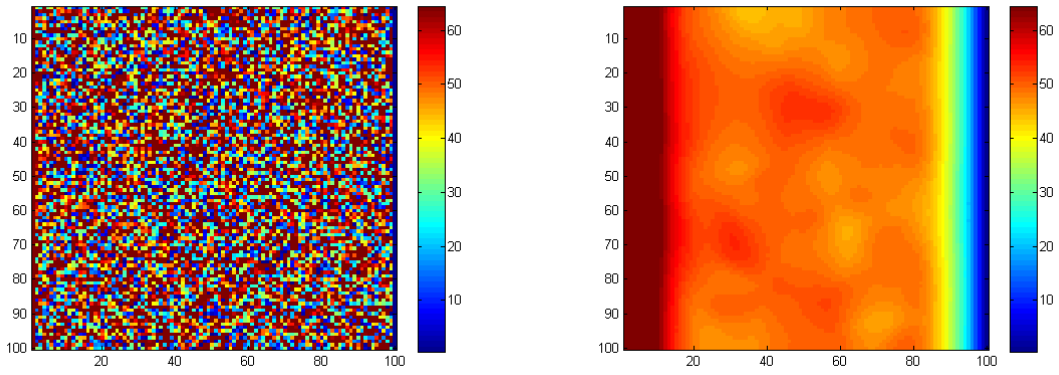
Πείραμα 2

Ένα άλλο είδος προβλήματος που ασχοληθήκαμε, αφορά δισδιάστατα πλέγματα και έχει να κάνει με τη μεταφορά θερμοκρασίας σε υγρά. Η διαδικασία της μεταφοράς θερμοκρασίας υπακούει στους νόμους του Νεύτωνα. Και εδώ ακολούθησε η ίδια μεθοδολογία και πειραματισμός με το παραπάνω πείραμα. Έχουμε κάποιες σταθερές πηγές θερμοκρασίας που εκπέμπουν σε συγκεκριμένες τιμές και με την πάροδο του χρόνου επηρεάζουν το υπόλοιπο πλέγμα. Κατά την διάρκεια των προσομοιώσεων οι πηγές παρουσιάζουν κάποιες διακυμάνσεις στις τιμές τους. Αυτό έγινε για να μπορέσουμε να μελετήσουμε ένα πιο ρεαλιστικό σύστημα, το οποίο αντιδράει αναλόγως στις μεταβολλές του περιβάλλοντος.

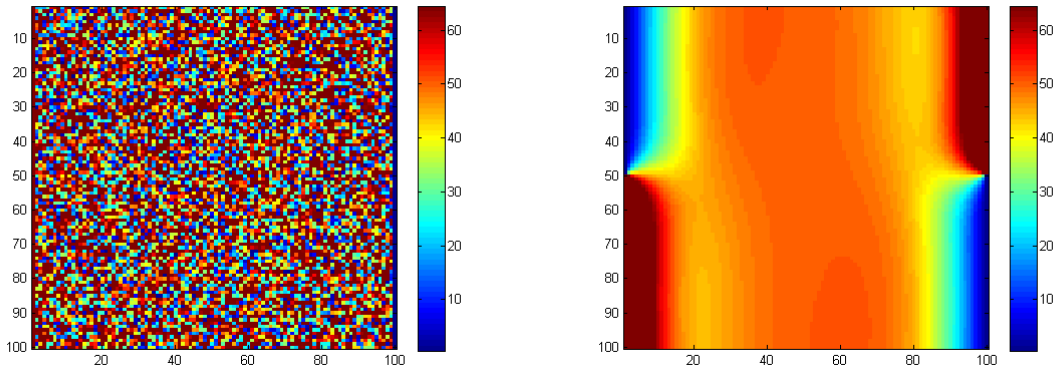
Στο πρώτο σχήμα από το 2.5 έχουμε ένα πλέγμα 100x100 με τυχαίες τιμές θερμοκρασίας, ενώ στα άκρα του έχουμε 2 σταθερές πηγές θερμότητας. Η μία εκπέμπει σε υψηλές θερμοκρασίες ενώ η άλλη σε χαμηλές. Στο δεύτερη σχήμα του 2.5 βλέπουμε το αποτέλεσμα της σύγκλισης της προσομοίωσης.

Στο πρώτο σχήμα από το 2.6 έχουμε πάλι ένα "ακατάστατο" πλέγμα 100x100 με 4 σταθερές πηγές θερμοκρασίας. Οι πηγές αυτές έχουν μεγάλες διακυμάνσεις στις τιμές τους κατά την διάρκεια της προσομοίωσης, κάτι που το καθιστά πιο πολύπλοκο σε σχέση με το προηγούμενο πείραμα. Στο δελυτερο σχήμα από το 2.6 φαίνεται το αποτέλεσμα της σύγκλισης ύστερα από πολλές επαναλήψεις.

Αυτά είναι μερικά από τα πειράματα που υλοποιήσαμε και αφορούν έναν μόνο παράγοντα, τη θερμοκρασία. Τα πραγματικά συστήματα πρόγνωσης, περιλαμβάνουν πάρα πολλούς παράγοντες όπως η πυκνότητα, η πίεση, η αλατότητα, η ταχύτητα των κυμάτων ο ατμοσφαιρικός αέρας κ.α. Με τα πειράματα αυτά καταφέραμε να πάρουμε ένα μικρό δείγμα της λειτουργίας και της πολυπλοκότητας των πραγματικών μοντέλων.



Σχήμα 2.5: Πείραμα υλοποίησης μεταφοράς θερμότητας σε πλέγμα 100x100 με 2 σταθερές πηγές στα άκρα. Αριστερα φαίνεται το τυχαία αρχικοποιημένο πλέγμα και δεξιά το αποτέλεσμα σύγκλισης ύστερα από πολλές επαναλήψεις.



Σχήμα 2.6: Πείραμα υλοποίησης μεταφοράς θερμότητας σε πλέγμα 100x100 με 4 πηγές στα άκρα. Αριστερα φαίνεται το τυχαία αρχικοποιημένο πλέγμα και δεξιά το αποτέλεσμα σύγκλισης ύστερα από πολλές επαναλήψεις.

2.3 Princeton Ocean Model

Το Princeton Ocean Model (*POM*), αποτελεί ένα ισχυρό ωκεανικό μοντέλο. Τα ωκεανικά μοντέλα προσπαθούν να προσομοιώσουν την συμπεριφορά των ωκεανών βάση ορισμένων παραμέτρων. Τέτοιες παραμέτρους συνήθως αποτελούν η θερμοκρασία, η περιεκτικότητα σε αλάτι, η πυκνότητα και τα διανύσματα κίνησης των θαλάσσιων ρευμάτων κατά τόπους αλλά και στο σύνολο τους. Στηρίζεται στην επίλυση διαφορικών εξισώσεων σε τρισδιάστατα πλέγματα και είναι βασισμένο στις αρχές του CFD. Χρησιμοποιεί το sigma-coordinate σύστημα([6]), με βάση το οποίο οι κάθετες συντεταγμένες κλιμακώνονται ανάλογα το βάθος και τη μορφολογία του βυθού, ενώ για το οριζόντιο πλέγμα χρησιμοποιεί καμπυλόγραμμες ορθοκανονικές συντεταγμένες. Υποστηρίζει τις επιφανειακές αναταράξεις και τον κυματισμό ως παράγοντες που επηρεάζουν το μοντέλο (*π.χ. επίδραση της ατμόσφαιρας*). Επίσης το μοντέλο θεωρε-

ίται ελεύθερης επιφάνειας([5]) και χρησιμοποιεί time steps για την επιλυση των συστημάτων εξισώσεων σε κάθε σημείο του πλέγματος κατά τη διάρκεια της προσομοίωσης. Χωρίζεται σε δύο τμήματα, το εξωτερικό και το εσωτερικό. Το εξωτερικό τμήμα είναι δισδιάστατο και χρησιμοποιεί ένα μικρό time step στηριζόμενο στο CFL κριτήριο([3] [22]) και την εξωτερική ταχύτητα των κυμάτων. Το εσωτερικό κομμάτι είναι τρισδιάστατο και χρησιμοποιεί ένα μεγάλο time step σύμφωνα με το CFL κριτήριο και την εσωτερική ταχύτητα των κυμάτων.

Ακόμα, εμπεριέχει διάφορες υπολογιστικές τεχνικές προκειμένου να εξομαλύνει την πολυπλοκότητα της προσομοίωσης([26]). Είναι βαρύ υπολογιστικά σε σχέση με μοντέλα άλλων κατηγοριών και τα αποτελέσματα του διέπονται από μεγάλη ακρίβεια. Προσομοιώνει ένα ευρύ φάσμα προβλημάτων όπως είναι η κυκλοφορία και η ανάμειξη των υδάτων σε εκβολές ποταμών, σε λίμνες, σε ημίκλειστες θάλασσες και ωκεανούς, και υπολογίζει τις εκάστοτε μεταβολές που συντελούνται σε επιλεγμένες παραμέτρους. Μπορεί να χρησιμοποιηθεί σε πολλές εφαρμογές γενικού σκοπού, όπως είναι η αλληλεπίδραση της θαλάσσιας κυκλοφορίας και του φυτοπλαγκτόν, η μόλυνση των υδάτων, πρόγνωση καιρικών φαινομένων κ.α. Επίσης, υπάρχει η δυνατότητα να εισάγουμε στον πυρήνα του μοντέλου δευτερεύοντα αλληλοεξαρτώμενα μοντέλα (*π.χ. σύνδεση ατμοσφαιρικού μοντέλου*).

2.4 Παρόμοιες Δουλειές

Σε αυτή την υποενότητα θα παρουσιαστούν σχετικές δουλειές που έχουν γίνει στον τομέα των CFD εφαρμογών και των οικολογικών μοντέλων σε συνδιασμό με το hardware. Αρχικά θα παρουσιαστούν κάποιες υλοποιήσεις που έγιναν με τη χρήση των FPGAs πάνω σε δίαφορες CFD εφαρμογές. Ύστερα, θα παρουσιαστούν κάποιες υλοποιήσεις οικολογικών μοντέλων με τη χρήση των FPGAs και κάποιες άλλες με τη χρήση των GPUs. Τέλος, θα γίνει μια αναφορά στη θέση της διπλωματικής και τον τρόπο με τον οποίο διαφοροποιείται από τις υπόλοιπες δουλειές.

Σξετικές Δουλειές με FPGAs πάνω σε CFD Εφαρμογές

Σε αυτή την υποενότητα παρουσιάζονται μερικές CFD εφαρμογές και η βελτίωση της απόδοσης τους χρησιμοποιώντας FPGAs.

Στο [28] αναφέρεται η βελτίωση της lattice Boltzmann μεθόδου με τη χρήση FPGA. Πρόκειται για μια μέθοδο που μοντελοποιεί την συμπεριφορά των ρευστών, μέσω πλασματικών σωματιδίων εκτελώντας διαδικασίες μεταφοράς και σύγκρουσης πάνω σε ένα διακριτό δικτυωτό πλέγμα. Στη συγκεκριμένη δουλειά παρουσιάζεται η επιτάχυνση της μεθόδου πάνω σε δισδιάστατα πλέγματα, χρησιμοποιώντας μια Virtex4 FPGA XC4VFX100 η οποία παρόλο που λειτουργεί στα 67 Mhz παρουσιάζει επιτάχυνση 1,8 φορές σε σχέση με έναν 2.2GHz Opteron επεξεργαστή.

Στο [18] παρουσιάζεται μια νέα μέθοδος υπολογισμού για το FCHC μοντέλο αερίων με χρήση FPGA. Το FCHC μοντέλο προσομοιώνει προβλήματα ρευστομηχανικής σε δισδιάστατα και τρισδιάστατα πλέγματα. Στη συγκεκριμένη εργασία βλέπουμε την επιτάχυνση του υπολογισμού χρησιμοποιώντας μια FPGA (*ADM-XRC-II με μία Virtex-II XC2V6000*) πάνω σε ένα τρισδιάστατο πλέγμα 128×128×128, με 52.7 MHz συχνότητα εκτέλεσης. Η επιτάχυνση εκτιμάται σε 200 φορές ταχύτερη επεξεργασία σε σχέση με έναν Athlon επεξεργαστή στα 1800 MHz.

Επίσης, στο [19] γίνεται αναφορά στη βελτίωση μιας άλλης μεθόδου για δικτυωτά μοντέλα αερίων με τη χρήση FPGAs. Η δουλειά που παρουσιάζεται αφορά μικρά δισδιάστατα

συστήματα με περιορισμένο εύρος μνήμης. Χρησιμοποιήθηκε μια FPGA(*ADC RC1000 με μια Virtex XCV1000*) για την επεξεργασία σε πλέγματα μεγέθους 2048×1024. Το κέρδος της επιτάχυνσης ήταν 143 φορές ταχύτερη εκτέλεση σε σχέση με έναν Pentium-III 700MHz επεξεργαστή.

Στο [15] αναφέρεται η χρήση των FPGAs και των GPUs για την βελτίωση εναέριων πολεμικών εφαρμογών που στηρίζονται σε CFD. Παρουσιάζονται διάφορα ζητήματα και ιδέες για την κατασκευή αρχιτεκτονικών προκειμένου να πετύχουν καλύτερη απόδοση σε εναέρια προβήματα. Ασχολούνται κύρίως με την κατασκυή υβριδικών συστημάτων, προσπαθώντας να βελτιώσουν τα πιο απαιτητικά τμήματα των μοντέλων χρησιμοποιώντας το υλικό.

Σχετικές Υλοποιήσεις Οικολογικών Μοντέλων Βασισμένες σε FPGAs

Σε αυτή την υποενότητα παρουσιάζονται μερικές υλοποιήσεις οικολογικών μοντέλων χρησιμοποιώντας FPGAs.

Μια σχετική δουλειά παρουσιάζεται στο [21] και αφορά την επιτάχυνση εικονικών οικολογικών μοντέλων με τη χρήση FPGAs. Αφορά την ανάλυση και επεξεργασία μοντέλων που ασχολούνται με την μελέτη του πλαγκτόν στους ωκεανούς. Το είδος της μελέτης που πρέπει να γίνει και το βάθος της πληροφορίας, είναι παράγοντες που επηρεάζουν τις απαιτήσεις για λογική, μνήμη και μεταφοράς δεδομένων, οι οποίες διαφοροποιούνται σημαντικά από μοντέλο σε μοντέλο. Με τη χρήση των FPGAs, μπορεί να προσαρμοστεί η υλοποίηση σε hardware στις συγκεκριμένες απαιτήσεις των υπό εξέταση οικολογικών συστημάτων και να επιτευχθεί σημαντική επιτάχυσνη σε σχέση με την υλοποίηση σε software. Η πλακέτα που χρησιμοποιήθηκε ήταν μια Virtex 4 FPGA της Xilinx που δούλευε στα 150 MHz και η εκτέλεση της αποδείχτηκε 39 φορές πιο γρήγορη σε σχέση με το software που τρέχει πάνω σε έναν AMD Opteron 2200 series CPU στα 1.0 GHz. Τα υπό μελέτη μοντέλα είναι μίας διάστασης και η αριθμητική που χρησιμοποιείται είναι μεταβλητής διπλής ακρίβειας.

Στο [24] παρουσιάζεται η προσπάθεια επιτάχυνσης ενός ατμοσφαιρικού μοντέλου μέσω ενός υβριδικού συστήματος CPU-FPGA χρησιμοποιώντας μικτή ακρίβεια υπολογισμών. Η ανάγκη για εξοικονόμηση πόρων, οδήγησε στη χρήση fixed point αριθμητικής στα κομμάτια του μοντέλου όπου τα αποτελέσματα το επέτρεπαν, και τη χρήση 32 bit floating point εκεί όπου η δυναμική περιοχή των υπολογισμών έπρεπε να είναι μεγαλύτερη. Η υλοποίηση έγινε σε πλέγματα 1024X1024x6 τύπου cube-sphere. Για το υβριδικό σύστημα CPU-FPGA χρησιμοποιήθηκε μια Virtex 6 FPGA και παρουσιάστηκε 100 φορές πιο γρήγορο σε σύγκριση με μια CPU 6 πυρήνων(*Intel i7 quad-core*) και 4 φορές πιο γρήγορο σε σχέση με ένα υβριδικό σύστημα με CPU 12 πυρήνων(*Intel Xeon*) και μια Fermi GPU. Ακόμα, δοκιμάστηκε μια υλοποίηση με 4 συνεργατικές FPGAs χρησιμοποιώντας το εργαλείο της Maxeler. Εκεί επιτεύχθηκε υλοποίηση 330 φορές πιο γρήγορη σε σχέση με τη CPU 6 πυρήνων και 14 φορές πιο γρήγορη σε σχέση με το υβριδικό σύστημα CPU-GPU.

Μια ακόμη δουλειά που αφορά τα θαλάσσια οικοσυστήματα αλλά δεν έχει σχέση με CFD προβλήματα παρουσιάζεται στο [10]. Σε αυτή τη δημοσίευση παρουσιάζεται η χρήση των FPGAs για την επιτάχυνση ενός αλγορίθμου ανίχνευσης ψαριών. Σκοπός είναι η προστασία του θαλάσιου οικοσυστήματος και η ρύθμιση της θαλάσσιας αλιείας. Η μελέτη των ψαριών μπορεί να οδηγήσει σε πολύ σημαντικά συμπεράσματα διαφόρων ερευνών. Οι διαδικασίες μελέτης τους όμως αποτελούν χρονοβόρες και ακριβές διαδικασίες. Έτσι, σκοπός ήταν η δημιουργία μιας αυτοματοποιημένης μεθόδου μελέτης χρησιμοποιώντας ένα σύστημα ταξινόμησης/κατάταξης με χρήση μηχανικής όρασης(*computer vision*). Αυτό το σύστημα χρησιμοποιώντας υποθαλάσσιες εικόνες βίντεο πραγματικού χρόνου, μπορεί να δώσει πληροφορίες σχετικά με το είδος

του βυθού και να μετρήσει και να κατατάξει τα ψάρια σύμφωνα με τη γνωστή μέθοδο Haar classification. Η πλακέτα που χρησιμοποιήθηκε για την πραγματικού χρονου επεξεργασία ήταν μια Virtex 5 FPGA της Xilinx. Η σύγκριση της συνεισφοράς της έγινε με ένα σύστημα που αποτελείται από έναν Intel Core Quad CPU (*2.4 GHz*) και 8 GB DDR2 SDRAM(*800MHz*). Τα αποτελέσματα έδειξαν ότι με την χρήση FPGA μπορεί να επιτευχθεί επιτάχυνση εώς 84.5 φορές για 320x240 ανάλυση εικόνων, και 37.39 φορές πιο γρήγορα για 640x480 ανάλυση.

Υλοποιήσεις Βασισμένες σε GPUs

Σε αυτή την υποενότητα παρουσιάζονται μερικές υλοποιήσεις οικολογικών μοντέλων χρησιμοποιώντας GPUs.

Στο [13] παρουσιάζεται η δημιουργία ενός τρισδιάστατου ωκεανικού μοντέλου και η βελτίωση της απόδοσης του με την χρήση των GPUs. Το μοντέλο αυτό, δε χρησιμοποιεί κάποιες από τις γνωστές μεθόδους προκειμένου να μειώσει το πλήθος των υπολογισμών, παρά λύνει απευθείας και ρητά όλες τις εξισώσεις των ωκεάνιων ρευμάτων. Για προσομοιώσεις με μεγάλη ακρίβεια, η υλοποίηση του μοντέλου έγινε σε τρισδιάστατα πλέγματα βασισμένα σε sigma coordinate συντεταγμένες, με περιορισμένο time step λόγω της γρήγορης ταχύτητας των βαρομετρικών κυμάτων. Προσομοιώνει μόνο υδροδυναμική, αποκλείοντας παράγοντες όπως η αλατότητα και η θερμοκρασία. Η σύγκριση των πειραμάτων έγινε με μια CPU της Intel Core i7-920 στα 2.66 GHz και μιας GPU της NVidia Geforce GTX460 στα 775 MHz. Ο προγραμματισμός της GPU έγινε μέσω CUDA στη γλώσσα Fortran, όπου ήταν γραμμένο και το μοντέλο. Η υλοποίηση έγινε με αριθμητική μεταβλητής μονής ακρίβειας (*32 bit*). Τα αποτελέσματα των πειραμάτων έδειξαν ότι το τρισδιάστατο μοντέλο έτρεχε περίπου 20 φορές πιο γρήγορα με την χρήση της GPU.

Στο [11] παρουσιάζεται η χρήση των GPUs για την επιτάχυνση των βαροτροπικών ωκεανικών μοντέλων. Συγκεκριμένα, προσπαθεί να επιταχυνθεί η επεξεργασία μιας 2 διαστάσεων βαροτροπικής εξίσωσης στροβιλισμού, που αποτελεί μία από τις βασικές εξισώσεις των ωκεανικών μοντέλων. Η επιτάχυνση που μπορεί να επιτευχθεί καθορίζεται από το μέγεθος του πλέγματος του μοντέλου. Ο μεγάλος επιτρεπτός παραλληλισμός που παρέχουν οι GPUs μπορεί να αποδώσει εώς 50 φορές επιτάχυνση για πλέγματα ανάλυσης από 2049x2049 εώς 4097x4097. Για μικρότερα πλέγματα η βελτίωση ήταν μικρότερη ενώ για μεγαλύτερα έπεφτε σταδιακά λόγω του overhead. Τα πειράματα έλαβαν μέρος σε δισδιάστατα πλέγματα με αριθμητική μεταβλητής μονής ακρίβειας. Η σύγκριση έγινε με μια NVIDIA Tesla M1060 GPU και έναν Intel Xeon L5420 2.50 GHz CPU.

Στο [25] περιγράφεται το ωκεανικό μοντέλο Regional Ocean Modeling System (*ROMS*) και συγκεκριμένα οι προσπάθειες για την επίτευξη της επιτάχυνσής του. Το ROMS προσομοιώνει περιοχές ωκεανών σε πλέγματα λύνοντας εξισώσεις διαφορών χρησιμοποιώντας time stepping. Οι εφαρμογές του ποικίλουν και η απόδοση του εξαρτάται από την ανάλυση του πλέγματος και το μέγεθος του time step. Για την βελτίωση και παραλληλοποίηση του μοντέλου χρησιμοποιούνται εργαλεία όπως το OpenMP ή το MPI. Στη συγκεκριμένη δουλειά δοκιμάζεται η βελτίωση του μοντέλου χρησιμοποιώντας GPU μέσω CUDA πάνω σε Fortran. Η χρήση GPU μαζί με τη μικρότερη κατανάλωση ισχύος, πετυχαίνει και σημαντική επιτάχυνση του μοντέλου μέσω του μαζικού παραλληλισμού που παρέχει. Συγκεκριμένα η επιτάχυνση που επιτυγχάνεται είναι 8 φορές πιο γρήγορη από το απλό σειριακό μοντέλο, 2,5 φορές πιο γρήγορη από το OpenMP, ενώ παρόμοια εκτιμάται με το MPI. Τα πειράματα έγιναν με ρεαλιστικά δεδομένα και ο εξοπλισμός που χρησιμοποιήθηκε ήταν :
Για την απλή σειριακή υλοποίηση ένας Athlon II στα 2.9 GHz.

- Για την υλοποίηση με το εργαλείο OpenMP χρησιμοποιήθηκαν 2 Intel Xeon E5504 με 8 συνολικά πυρήνες.
- Για την υλοποίηση με το εργαλείο MPI χρησιμοποιήθηκε ένα cluster από Intel Xeon 5130 επεξεργαστές με 64 συνολικά πυρήνες.
- Για την υλοποίηση με CUDA χρησιμοποιήθηκε μία GTX 470 GPU. Αυτή η κάρτα είναι βασιμένη στην αρχιτεκτονική του Fermi, και τα χαρακτηριστικά της είναι 1280 MB μνήμη και 448 πυρήνες στα 1.22 GHz.

2.4.1 Διαφοροποίηση Της Διπλωματικής

Η δουλειά που παρουσιάζεται στην συγκεκριμένη διπλωματική εργασία αφορά ένα ωκεανικό μοντέλο, το POM. Η επεξεργασία λαμβάνει μέρος σε τρισδιάστατα πλέγματα και στηρίζεται στην επίλυση συστημάτων διαφορών. Σκοπός είναι η επιτάχυνση μιας βασικής συνάρτησης του μοντέλου αυτού, η οποία καταλαμβάνει το 45% του συνολικού φόρτου του προγράμματος. Η συνάρτηση αυτή είναι υπεύθυνη για τον υπολογισμό της ροης των ρευστών σε κάθε time step για όλα τα σημεία του πλέγματος. Χρησιμοποιείται μια μέθοδος ονομαζόμενη directional splitting, σύμφωνα με την οποία αντί να γίνεται ο υπολογισμός της πυκνότητας του ρευστού άμεσα για κάθε σημείο, σπάει αυτό τον υπολογισμό για την κάθε διάσταση ξεχωριστά, και στο τέλος συνδιάζει τα αποτελέσματα όλων των διαστάσεων προκειμένου να προκύψει το τελικό αποτέλεσμα. Πρόκειται για ένα υβριδικό μοντέλο, το οποίο τρέχει γενικά σε software(*Fortran*) και όταν έρθει η ώρα της κλήσης της συνάρτησης, η επεξεργασία μεταφέρεται στο hardware. Η υλοποίηση γίνεται σε μια Virtex 5(*XC5VLX330*) της Xilinx και η αριθμητική που χρησιμοποιείται είναι μεταβλητής μονής ακρίβειας (*32 bit*). Η εκτέλεση της συνάρτησης μέσω της FPGA με συχνότητα λειτουργίας στα 150 MHz αποδείχτηκε 3,16 φορές ταχύτερη σε σχέση με ένα σύστημα με επεξεργαστή 8 πυρήνων στα 2,7 MHz και 12 Mb μνήμη.

Κεφάλαιο 3

Μελέτη της Συνάρτησης Ενδιαφέροντος

Αυτή η ενότητα αναφέρεται στη πρώτη προσσέγιση του προβλήματος. Περιγράφεται η διαδικασία του profiling ώστε να εκτιμήσουμε τα βαριά υπολογιστικά κομμάτια του μοντέλου, η διαδικασία προσομοίωσης αυτών των κομματιών και η επιλογή της αριθμητικής για την ακρίβεια αναπαράστασης των αποτελεσμάτων.

3.1 Profiling

Η πρώτη φάση της μελέτης του μοντέλου είναι το profiling. Με αυτή τη διαδικασία μπορεί να εκτιμηθεί και να αναλυθεί ο υπολογιστικός φόρτος του μοντέλου. Πιο συγκεκριμένα μπορούμε να αποφανθούμε πως συμβάλει κάθε τμήμα του προγράμματος ξεχωριστά και έτσι να βρούμε το κρίσιμο μονοπάτι του προβλήματος. Πρόκειται για ένα μεγάλο μοντέλο το οποίο είναι γραμμένο σε γλώσσα προγραμματισμού Fortran και αποτελείται συνολικά από 107 αρχεία. Για τη διαδικασία του profiling χρησιμοποιήθηκε το εργαλείο Vtune της Intel.

Βάση του profiling εξήχθησαν τα εξής συμπεράσματα. Όπως φένεται και στην εικόνα 3.1 παρατηρήθηκε ότι το μέγιστο υπολογιστικό φόρτο κατείχε μια συνάρτηση με το όνομα advtl_lin. Η συγκεκριμένη συνάρτηση έπαιρνε συνολικά περίπου το 45% του συνολικού χρόνου εκτέλεσης. Το ποσοστό αυτό είναι αρκετά μεγάλο και ήταν μια καλή απάντηση στο ερώτημα με ποια τμηματα του προγράμματος αξίζει να ασχοληθούμε, ώστε να βελτιώσουμε την συνολική του εικόνα.

Function / Call Stack	CPU Time
+advtl_lin_	45.5%
+proft2_	10.0%
+integration_	6.6%
+nmap1to3zero_	6.0%
+set_fluxes_to_zero_	4.4%
+nmap3to1_	4.3%
+pom_step_	3.5%
+__powr8i4	2.5%
+profq_	1.5%
+atmos_forc_	1.4%
+nesting_eco_	1.4%
+dens_	1.3%
+expf.L	1.2%
+advct_	1.1%

Σχήμα 3.1: Αποτελέσματα του profiling. Η συνάρτηση ενδιαφέροντος(*advtl_lin*) αναλογεί στο 45% του συνολικού φόρτου του μοντέλου.

Οπότε με βάση τα αποτελέσματα του profiling αποφασίσαμε να ασχοληθούμε με τη μελέτη της συγκεκριμένης συνάρτησης. Σύμφωνα με το νόμο του Amdahl([1]),έχουμε ότι το τμήμα που κατέχει το 55% του συνολικού χρόνου του προγράμματος δε θα αλλάξει, ενώ το 45% είναι αυτό που θέλουμε να βελτιώσουμε. Οπότε η μέγιστη επιτάχυνση που μπορεί να επιτευχθεί είναι $1/(1 - 0.45) = 1.818$ φορές πιο γρήγορα σε σχέση με το αρχικό μοντέλο.

Πρόκειται για ένα πολύ σημαντικό κέρδος, ειδικά αν αναλογιστεί κανείς τη φύση αυτών των προβλημάτων. Είναι μοντέλα τα οποία τρέχουν πολύ μεγάλες προσομοιώσεις, με υψηλή πολυπλοκότητα, για μεγάλα χρονικά διαστήματα. Έτσι καταλήξαμε ότι άξιζε τον κόπο να ασχοληθούμε με τη συγκεκριμένη συνάρτηση, καθώς τα αποτελέσματα ήταν πολύ αισιόδοξα και θα μπορούσαμε να έχουμε ένα αξιόλογο αποτέλεσμα.

3.2 Συνάρτηση Ενδιαφέροντος

Η συνάρτηση που επιλέξαμε να ασχοληθούμε, είναι μία από τις πιο βασικές συναρτήσεις του POM και γενικότερα των ωκεανικών μοντέλων ([23]). Η χρησιμότητα της, είναι ο υπολογισμός της πυκνότητας των υδάτων, για κάθε subgrid του συνολικού πλέγματος. Αυτό γίνεται σε κάθε time step, επιλύοντας τα κατάλληλα συστήματα εξισώσεων.

Επειδή ο υπολογισμός της πυκνότητας των υδάτων σε τρισδιάστα πλέγματα είναι μια βαριά υπολογιστική και αργή διαδικασία, ειδικά όταν πρόκειται για μεγάλης διάστασης προβλήματα, το μοντέλο χρησιμοποιεί μια τεχνική λεγόμενη ως Directional Splitting. Σύμφωνα με αυτή την τεχνική, η διαδικασία της επεξεργασίας αντί να γίνει κατευθείαν μέσω ενός αριθμητικού υπολογισμού συναρτήσει όλων των διαστάσεων, σπάει ξεχωριστά για κάθε μία διάσταση. Έτσι, έχουμε 3 διαφορετικές επεξεργασίες, μία για κάθε διάσταση. Έπειτα, αφού τελειώσει η επεξεργασία όλων των διαστάσεων, ακολουθεί ο συνδιασμός των αποτελεσμάτων τους μέσω ορισμένων συστημάτων εξισώσεων, προκειμένου να γίνει η εύρεση του συνολικού αποτελέσματος. Με αυτή την τεχνική, μειώνεται η υπολογιστική πολυπλοκότητα και βελτιώνεται η χρονική απόκριση του μοντέλου.

Όσον αφορά τα τεχνικά χαρακτηριστικά, η συνάρτηση όπως και το συνολικό πρόγραμμα είναι γραμμένη σε Fortran (*300 γραμμές κώδικα*) και αποτελείται από μαθηματικά συστήματα εξισώσεων, που εφαρμόζονται σε τρισδιάστατα πλέγματα. Συγκεκριμένα, εμπεριέχει 11 συστήματα εξισώσεων και χρησιμοποιούνται 16 τριπλά loops για τη ροή των δεδομένων και την επεξεργασία τους. Επίσης, χαρακτηρίζεται από υψηλό I/O, καθώς για τον υπολογισμού του συνολικού αποτελέσματος χρειάζονται συνολικά 14 είσοδοι. Στις εισόδους συγκαταλέγονται 4 τρισδιάστατοι, 7 δισδιάστατοι, 2 μονοδιάστατοι πίνακες και 1 σταθερά. Ειδικά με τη σχεδίαση στο hardware και την επικοινωνία με τη μνήμη, αν λάβουμε υπόψην μας ότι σε μερικούς από αυτούς τους πίνακες χρειαζόμαστε ταυτόχρονα 3 διαφορετικές τιμές και όχι μία, τότε το I/O μπορεί να φτάσει μέχρι και τις 28 εισόδους σε ορισμένες περιπτώσεις.

3.3 Προσομοίωση Συνάρτησης

Ύστερα από την εύρεση και τη μελέτη της συνάρτησης με την οποία θα ασχοληθούμε, προχωρήσαμε στην προσομοίωση της. Αυτό μας έδωσε την δυνατότητα για την περεταίρω μελέτη και επεξεργασία της συνάρτησης, ανεξάρτητα από το υπόλοιπο πρόγραμμα. Η προσομοίωση έγινε σε Matlab, μια πιο οικεία γλώσσα σε σχέση με τη Fortran και με περισσότερες δυνατότητες επεξεργασίας.

Αρχικό στάδιο, πριν το στάδιο της προσομοίωσης ήταν η εξαγωγή των κατάλληλων δεδομένων που χρησιμοποιεί η συνάρτηση για να τρέξει. Χρησιμοποιώντας το πρόγραμμα της Fortran εξάγαμε και αποθηκεύσαμε όλα τα απαραίτητα data sets των εισόδων σε αρχεία, τα οποία ύστερα θα τα χρησιμοποιούσαμε ως εισόδους στο πρόγραμμα της Matlab. Έπειτα, περάσαμε στο στάδιο γραψίματος του κώδικα στη Matlab και το διάβασμα των εισόδων από τα αρχεία που είχαμε εξάγει από τη Fortran.

Τέλος, ακολούθησε ο έλεγχος και η επικύρωση των αποτελεσμάτων της προσομοίωσης. Ο έλεγχος γινόταν βήμα βήμα για κάθε σύστημα εξισώσεων ξεχωριστά και στη συνέχεια για το συνδιασμό τους. Κατά την εκτέλεση στη Fortran, για κάθε σύστημα εξισώσεων είχαμε αποθηκεύσει τις εξόδους σε αντίστοιχα αρχεία. Η αντίστοιχη δουλειά έγινε και στη Matlab, για όλες τις εξόδους των συστημάτων των εξισώσεων. Έπειτα μέσω ενός script, το οποίο έπαιρνε ως εισόδους τα αρχεία των εξόδων από τη Fortran και τη Matlab, γινόταν η σύγκριση των αποτελεσμάτων τους. Έτσι μπορούσαμε να αποφανθούμε μέσω της σύγκρισης, για την ορθότητα των αποτελεσμάτων της προσομοίωσης. Η διαδικασία σύγκρισης, έγινε για όλα τα συστήματα των εξισώσεων ένα προς ένα και μετέπειτα για τους συνδιασμούς τους, μέχρι να φτάσουμε στο συνολικό συνδιασμό όλων των εξισώσεων.

Όταν βεβαιωθήκαμε ότι φτιάξαμε μια καλή προσομοίωση, η οποία εβγάζε τα ίδια ακριβώς αποτελέσματα με τη Fortran, προχωρήσαμε στο επόμενο στάδιο επεξεργασίας, που ήταν η ανάλυση της ακρίβειας των υπολογισμών της συνάρτησης.

3.4 Μελέτη Αριθμητικής

Σε αυτή την υποενότητα θα ασχοληθούμε με τη μελέτη της αρκίβειας των υπολογισμών της συνάρτησης και τον πειραματισμό με τους διαφορους αριθμητικούς τύπους δεδομένων.

3.4.1 Κίνητρο

Το έναυσμα για την μελέτη της αριθμητικής, ήταν η βασική ανάγκη για εξοικονόμηση πόρων. Επειδή πρόκειται για ένα μοντέλο που ασχολείται με την προσομοίωση μεγάλων περιοχών σε τρισδιάστατα πλέγματα, μια τέτοια ανάγκη χρήζει επιτακτικής προτεραιότητας. Όσο περισσότερους πόρους έχουμε στη διάθεση μας, τόσο πιο εφικτή θα είναι η επεξεργασία μεγαλύτερων πλεγμάτων. Η επιλογή ενός καλού αριθμητικού τύπου δεδομένων, μπορεί να συμβάλλει σημαντικά στη μείωση των πόρων, όπως είναι η χρήση μικρότερων μνημών, μικρότερων μονάδων αριθμητικής επεξεργασίας, μικρότερων καταχωρητών κτλ.. Πέρα όμως από την εξοικονόμηση των πόρων που είναι ο βασικός στόχος μας, έχοντας μια αρχιτεκτονική που αποτελείται από μικρότερες μονάδες, αυτόματα πετυχαίνουμε και μια ταχύτερη αρχιτεκτονική. Για παράδειγμα, διαφορετική ταχύτητα επεξεργασίας δεδομένων έχουμε όταν χρησιμοποιούμε μια 64-bit αρχιτεκτονική σε σχέση με μια 32-bit. Και αυτό είναι πολύ εμφανές, ειδικά στο κομμάτι των αριθμητικών πράξεων όπου η διαφορά είναι ιδιαίτερα αισθητή.

Όλα αυτά μας οδήγησαν στην αναζήτηση ενός όσο το δυνατόν περισσότερο "φθηνού" τύπου δεδομένων, ο οποίος ταυτόχρονα έπρεπε να αποτελεί και μια αξιόπιστη λύση στα αποτελέσματα των εξόδων μας. Εννοώντας αξιόπιστη λύση, αναφέρομαι στην ακρίβεια των αποτελεσμάτων, και το σφάλμα απόκλισης το οποίο καθορίζει και την ποιότητα των αποτελεσμάτων.

3.4.2 Διαθέσιμες Επίλογές

Οι αριθητικές με τις οποίες ασχοληθήκαμε ήταν οι εξής : 64-bit floating point (*IEEE 754*), 32-bit floating point(*IEEE 754*) και fixed point με μεταβλητή υποδιαστολή.

Double Precision Floating Point

Η 64-bit floating point αριθμητική πρόκειται για μια πάρα πολύ καλή επιλογή όσον αφορά την ακρίβεια των αποτελεσμάτων, καθώς έχουμε την δυνατότητα απεικόνησης υψηλής ακρίβειας, ακόμη και των πιο απαιτητικών επιστημονικών εφαρμογών, με μεγάλη επιτυχία. Ταυτόχρονα όμως αποτελεί και μια ακριβή επιλογή καθώς έχουμε μεγάλη σπατάλη πόρων.

Single Precision Floating Point

Η 32-bit floating point αριθμητική πρόκειται για μια καλή επιλογή για την ακρίβεια των αποτελεσμάτων, αρκεί να αναφερόμαστε σε αποτελέσματα μικρότερου εύρους ακρίβειας σε σχέση με την 64-bit αριθμητική. Σε πολύ απαιτητικές περιπτώσεις ακρίβειας μπορεί να προκύψουν σημαντικά σφάλματα. Από την άλλη πλευρά όμως, έχουμε το πλεονέκτημα της εξοικονόμησης σημαντικού ποσοστού πόρων σε σχέση με την 64-bit αριθμητική και καλύτερη απόδοση στην ταχύτητα του ρολογιού.

Fixed Point

Η fixed point αριθμητική, πρόκειται για μια επιλογή με την οποία είναι στο χέρι μας να καθορίσουμε την ακρίβεια την οποία επιθυμούμε να πετύχουμε, αλλά και την σπατάλη των πόρων. Βέβαια πολλές φορές αυτά τα 2 είναι άμεσα συνδεδεμένα, γιατί συνήθως όταν θέλουμε καλύτερη ακρίβεια αυξάνουμε και τα bits της αριθμητικής. Δεν είναι όμως κανόνας αυτός, γιατί πρόκειται για μια αριθμητική την οποία προσαρμόζουμε εμείς στην ανάγκη μιας εξειδικευμένης

εφαρμογής. Έτσι, αν γνωρίζουμε το είδος και την ακρίβεια των αποτελεσμάτων που θέλουμε να πετύχουμε, είναι στο χέρι μας να προσαρμόσουμε με τον καλύτερο δυνατό τρόπο τον διαμοιρασμό των bits ανάμεσα στο ακέραιο και στο δεκαδικό μέρος. Με αυτό τον τρόπο μπορούμε να πετύχουμε ένα πάρα πολύ καλό αποτέλεσμα με την χρήση λίγων μόνων bits (*όσα είναι απαραίτητα*) και ταυτόχρονα να εξοικονομήσουμε πόρους. Είναι μια πολύ καλή επιλογή όταν τα αποτελέσματα μας είναι του ίδιου τύπου και δεν επιδέχονται μεγάλες διακυμάνσεις. Η εξειδίκευση που μας προσφέρει όμως αυτή η επιλογή πολλές φορές είναι και το μεγάλο μειονέκτημα της. Αυτό εμφανίζεται σε προβλήματα όπου τα αποτελέσματα έχουν μεγάλες διακυμάνσεις μεταξύ τους και το εύρος τιμών τους είναι εξίσου μεγάλο. Σε τέτοιες περιπτώσεις, για την αναπαράσταση τους πρέπει να χρησιμοποιηθούν πολλά bits με αποτέλεσμα την σπατάλη πορών. Τότε η επιλογή μια αριθμητικής γενικού σκοπού τύπου floating ίσως αποτελέσει αποδοτικότερη λύση.

3.4.3 Περιβάλλον Σύγκρισης

Για την επιλογή της αριθμητικής, έπρεπε να γίνει σύγκριση μεταξύ των διαφορετικών αριθμητικών τύπων, προκειμένου να αποφανθούμε ποια είναι η καταλληλότερη. Τα κριτήρια όπως προαναφέρθηκαν είναι το κόστος των απαιτούμενων πόρων, η καθυστέρηση και η ακρίβεια των αποτελεσμάτων.

Η σύγκριση είχε ως κριτήριο αναφοράς την 64-bit floating point αριθμητική. Πέρα από την εξοικονόμηση των πόρων, θέλαμε να μελετήσουμε την απόκλιση που είχαμε σε σχέση με την ακριβέστερη αριθμητική. Αν οι απαιτήσεις του μοντέλου για ακρίβεια ήταν πολύ αυστηρές και η απόκλιση της σύγκρισης μεγάλη, τότε δεν θα είχε νόημα η χρήση μιας άλλης αριθμητικής μικρότερης ακρίβειας.

Τα δεδομένα που βάζαμε ως εισόδους στο πρόγραμμα επεξεργασίας της κάθε αριθμητικής, τα πέρναμε από τα διαθέσιμα data sets του μοντέλου και όχι μόνο. Αφού μελετήσαμε το εύρος τιμών για κάθε είσοδο της συνάρτησης από το σύνολο των data sets, βάζαμε δικές μας τιμές εισόδων που αντιστοιχούσαν στο επιτρεπτό εύρος τιμών των αντίστοιχων εισόδων. Έτσι, η σύγκριση της ακρίβειας των πράξεων, έγινε με βάση πολλών πειραματισμών που ανταποκρίνονταν στις απαιτήσεις του μοντέλου που είχαμε στα χέρια μας.

Η διαδικασία της επεξεργασίας της κάθε αριθμητικής έγινε στην πλατφόρμα της Matlab. Η default αριθμητική της Matlab είναι τύπου 64 bit floating point με το format της IEEE 754. Αρχικά, μελετήσαμε το σφάλμα που είχαμε από την αριθμητική της Fortran σε σχέση με την αριθμητική της Matlab (*64-bit floating*), η οποία αποδείχτηκε απειροελάχιστη. Ύστερα, η ίδια σύγκριση έγινε και για τις υπόλοιπες αριθμητικές που εξετάζαμε πάνω στη Matlab (*32-bit floating, fixed point*). Τέλος, υπολογίσαμε την συσχέτηση του σφάλματος που αφορά τη σύγκριση της 64-bit floating αριθμητικής της Matlab με την αριθμητική της Fortran, με το σφάλμα της σύγκρισης των υπόλοιπων υπό εξέταση αριθμητικών σε σχέση με την αριθμητική της Fortran. Με αυτό τον τρόπο, μπορούσαμε να δούμε πόσο μεγάλη απόκλιση ακρίβειας είχαμε για κάθε αριθμητική και να βγάλουμε σημαντικά συμπεράσματα για τον τρόπο με τον οποίο θα κινηθούμε.

3.4.4 Χρήση Σταθερής Ακρίβειας Αριθμητική

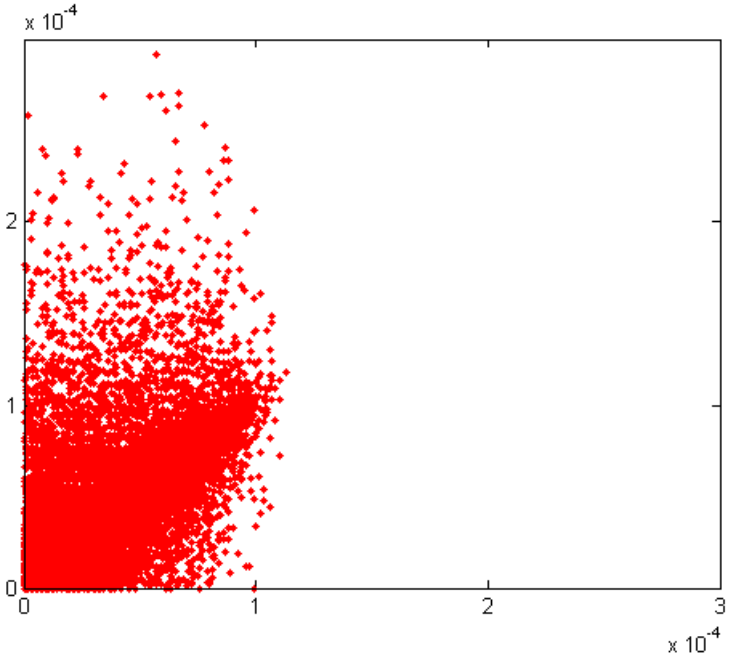
Με αυτή την αριθμητική, υπήρχε η δυνατότητα αλλαγής της θέσης της υποδιαστολλής αναλόγως την ακρίβεια που θέλαμε να πετύχουμε. Η μεταβολή της θέσης εξαρτόταν από τις εξισώσεις που έπρεπε να λυθούν σε κάθε σημείο και την ακρίβεια που απαιτούσαν. Αρχικά

προσπαθήσαμε να χωρέσουμε όλους τους υπολογισμούς σε 32 bit λέξεις. Σύντομα όμως, καταλάβαμε ότι ήταν πολύ μικρή για τις απαιτήσεις μας και έτσι έπρεπε να μεγαλώσουμε τον αριθμό των bits.

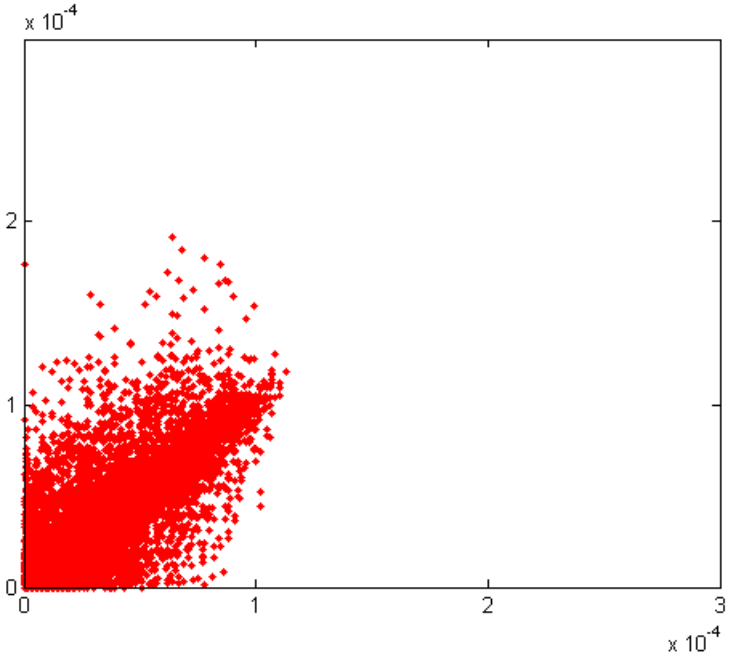
Για να έχουμε μια καλύτερη εποπτεία των διακυμάνσεων των τιμών εισόδων και εξόδων από τα συστήματα διαφορών, μελετήσαμε ξεχωριστά κάθε είσοδο και έξοδο αυτών των συστημάτων. Γνωρίζαμε ότι η συναρτηση μας συνολικά έχει 14 εισόδους. Με βάση τα διαθέσιμα data sets που είχαμε στα χέρια μας και μελετώντας το μοντέλο, βρήκαμε για κάθε είσοδο την μικρότερη και την μεγαλύτερη τιμή που μπορεί να λάβει. Με αυτό τον τρόπο, γνωρίζαμε το εύρος τιμών για κάθε είσοδο που λάμβανε μέρος στις πράξεις. Στη συνέχεια, επαναλάβαμε την ίδια διαδικασία για όλες τις εξόδους που προέκυπταν από την επίλυση κάθε εξίσωσης της συνάρτησης. Έτσι γνωρίζαμε και τις διακυμάνσεις όλων των εξόδων για κάθε τμήμα της συνάρτησης.

Όλες αυτές οι πληροφορίες ήταν πολύ σημαντικές και μας βοήθησαν στην διαμόρφωση των bits ακρίβειας των αριθμητικών πράξεων. Αρχικά, ορίζαμε τόσα bits στο ακέραιο μέρος όσα ήταν απαραίτητα ώστε να μην έχουμε έλλειψη της βασικής πληροφορίας του υπολογισμού. Έπειτα, αυτό που κάναμε ήταν να αυξάνουμε αντίστοιχα τον αριθμό των bits στο δεκαδικό μέρος, μέχρι να σιγουρευτούμε ότι τα αποτελέσματα που προέκυπταν σε σύγκριση με την 64-bit floating point αριθμητική είχαν μικρή απόκλιση. Αξίζει να αναφερθεί, ότι η σύγκριση της ακρίβειας των αποτελεσμάτων έγινε για κάθε σύστημα εξισώσεων ξεχωριστά, καθώς και για το σύνολο τους. Η θέση της υποδιαστολής δεν είναι σταθερή, μεταβάλλεται σε κάθε σύστημα εξισώσεων ανάλογα την ακρίβεια των αποτελεσμάτων που θέλουμε να πετύχουμε. Υπάρχουν συστήματα τα οποία έχουν ως έξοδο παρα πολύ μεγάλους αριθμούς, οπότε εκεί δίνουμε βάση στο ακέραιο μέρος και λιγότερο στο δεκαδικό. Αντίστοιχα, σε συστήματα τα οποία έχουν πολύ μικρές εξόδους, μπορεί να μην μας ενδιαφέρει το ακέραιο μέρος και να κατανείμουμε κατάλληλα τα bits για να πετύχουμε τη σωστή ακρίβεια στο δεκαδικό μέρος. Βέβαια δεν εκλείπουν και οι περιπτώσεις όπου τα αποτελέσματα έχουν μεγάλο εύρος τιμών, από πολύ μεγάλες εώς πολύ μικρές τιμές και εκεί πρέπει να γίνει η ανάλογη προσαρμογή. Όταν γνωρίζαμε την ακρίβεια των bits που χρειαζόμασταν για τη σωστή αναπαράσταση του ακέραιου μέρους των πράξεων, κάθε αύξηση της λέξης με επιπρόσθετα bits προοριζόταν μόνο για την ακρίβεια του δεκαδικού μέρους.

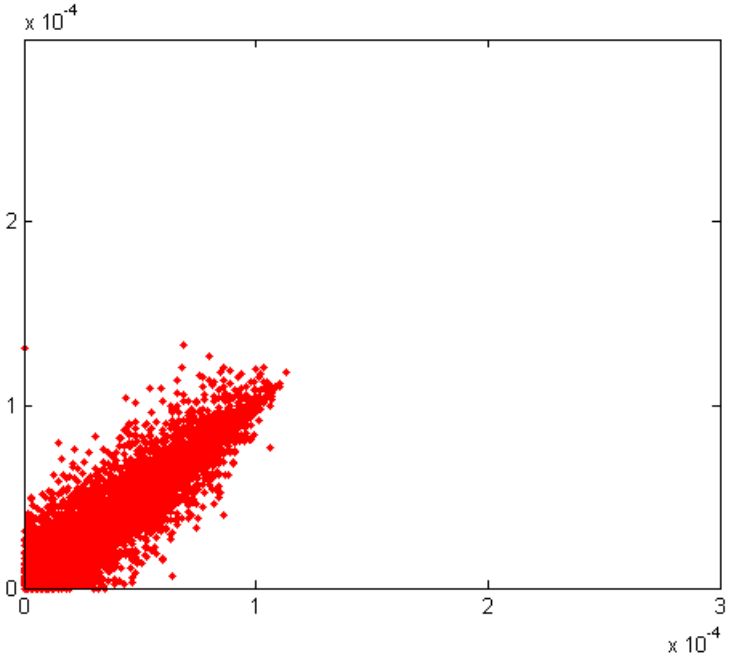
Στα σχήματα 3.2, 3.3, 3.4 φένεται η συσχέτιση του σφάλματος μεταξύ της 64-bit floating point και της fixed point αριθμητικής για διάφορετικό αριθμό bits. Οι περιπτώσεις που παρουσιάζονται στα σχήματα, έχουν όλες τον ίδιο αριθμό bits απεικόνισης του ακέραιου τμήματος. Για κάθε bit που αυξάνουμε βελτιώνουμε την ακρίβεια του δεκαδικού τμήματος. Η διαφορά από την αύξηση ενός μόνο bit είναι πολύ εμφανής στο συνολικό αποτέλεσμα. Αυτό γίνεται διότι στους τελικούς υπολογισμούς συνεισφέρουν όλα τα συστήματα εξισώσεων και το σφάλμα αθροίζεται. Τα σχήματα επίτηδες απεικονίζονται σε μεγάλη κλίμακα, για να μπορούμε να διακρίνουμε τους outliers, ειδικά στις πρώτες περιπτώσεις.



Σχήμα 3.2: Συσχέτιση σφάλματος 64-bit floating point με 50-bit fixed point. Με βάση τα διανύσματα σφαλμάτων η συσχέτιση εκτιμάται σε 0.507.



Σχήμα 3.3: Συσχέτιση σφάλματος 64-bit floating point με 51-bit fixed point. Με βάση τα διανύσματα σφαλμάτων η συσχέτιση εκτιμάται σε 0.647.



Σχήμα 3.4: Συσχέτιση σφάλματος 64-bit floating point με 52-bit fixed point. Με βάση τα διανύσματα σφαλμάτων η συσχέτιση εκτιμάται σε 0.716.

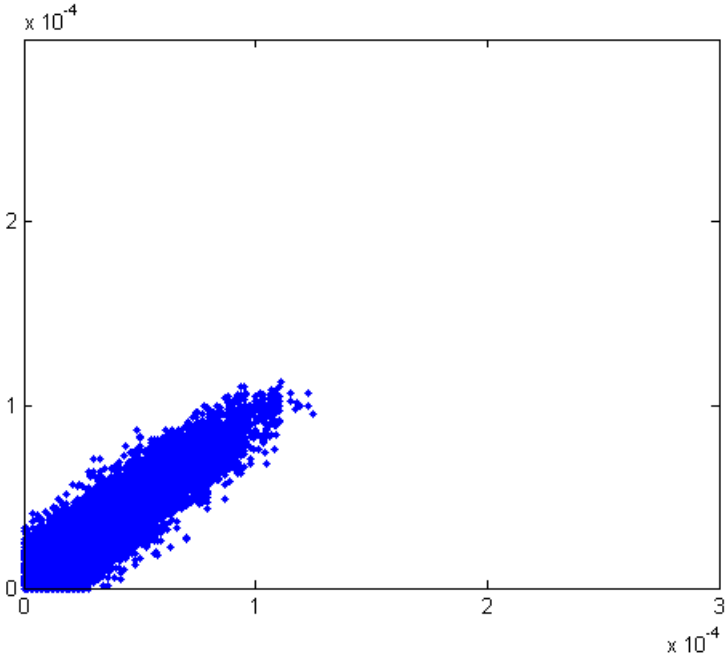
Ύστερα από αρκετές δοκιμές και πειραματισμό καταλήξαμε στο γεγονός ότι για να ανταπεξέλθουμε στις απαιτήσεις του προβλήματος για καλή ακρίβεια, έπρεπε να χρησιμοποιήσουμε τουλάχιστον 52 bits(*σχήμα 3.4*). Αυτό προέκυψε λόγω των μεγάλων διακυμάνσεων που είχαμε στα αποτελέσματα και την ανάγκη για μεγάλης ακρίβειας πράξεις. Η συνολική έξοδος της συνάρτησης εξαρτάται άμεσα από όλα τα συστήματα εξισώσεων που τη διέπουν. Αν έστω σε ένα μόνο σύστημα δεν έχουμε την επιθυμητή ακρίβεια, αρκεί ώστε να επηρεάσει αρνητικά το συνολικό αποτέλεσμα της συνάρτησης. Από 52 bits και άνω, και κατάλληλη τοποθέτηση της υποδιαστολής μπορούμε να πετύχουμε μια ακρίβεια παρόμοια με αυτή της 64-bit floaing point.

Όσο μεγαλώναμε το μέγεθος της λέξης τόσο καλύτερη ακρίβεια πετυχαίναμε. Η συνεχώς μεταβαλλόμενη θέση της υποδιαστολής προκειμένου να καλυφθούν επαρκώς οι ανάγκες για καλή ακρίβεια, αποτελεί μια χρονοβόρα διαδικασία, αλλά στο τέλος ανταμοιβόμαστε με ένα καλό αποτέλεσμα. Με την χρήση των 52 bits αποδείχθηκε ότι είχαμε ένα πάρα πολύ καλό αποτέλεσμα, όχι ακριβώς ίδιο με αυτό της μεταβλητής διπλής ακρίβειας αλλά πάρα πολύ κοντά σε αυτό.

Για τη μελέτη και επεξεργασία της συνάρτησης με την fixed point αριθμητική, χρησιμοποιήθηκε ένα ειδικό tool kit της Matlab(*Fixed-Pont Toolbox*). Με το συγκεκριμένο εργαλείο, είχαμε τη δυνατότητα να ορίζουμε το μέθεθος των bits, να μεταβάλουμε την υποδιαστολή στο σημείο που θέλαμε, καθώς και να επιλέγουμε τη μέθοδο στρογγυλοποίησης που επιθυμούσαμε. Ήταν μια μεγάλη διευκόλυνση που μας έδινε τη δυνατότητα για άπλετο πειραματισμό.

3.4.5 Χρήση Μεταβλητής Μονής Ακρίβειας Αριθμητική

Η 32-bit floating point αριθμητική αποτελείται από 32 bit λέξεις. Το format το οποίο χρησιμοποιήσαμε βασίζεται στο Standar IEEE 754. Εδώ η μελέτη ήταν πιο εύκολη σε σχέση



Η

Σχήμα 3.5: Συσχέτιση σφάλματος 64-bit floating point με 32-bit floating point. Με βάση τα διανύσματα σφαλμάτων η συσχέτιση εκτιμάται σε 0.918.

με αυτή της fixed point αριθμητικής. Έγινε σύγκριση της ακρίβειας των πράξεων για κάθε σύστημα εξισώσεων που απαρτίζει τη συνάρτηση, και συγκρίθηκαν τα αποτελέσματα με την 64-bit floating point αριθμητική. Αποδείχθηκε ότι είχαμε πολύ ικανοποιητικά αποτελέσματα με μικρό κόστος ακρίβειας, το οποίο δεν μας ανησύχησε καθώς η απόκλιση ήταν ελάχιστη.

Όπως φένεται και στο σχήμα 3.5, έχουμε μια πολύ καλή συσχέτηση σε σχέση με την 64-bit floating point αριθμητική. Είναι σταθερή, χωρίς outliers και πάρα πολύ κοντά σε αυτό που θέλουμε.

Για τη μελέτη και επεξεργασία της συνάρτησης με την 32-bit floating point, χρησιμοποιήσαμε μια ειδική συνάρτηση της matlab(*single()*), με την οποία είχαμε τη δυνατότητα να εκτελέσουμε όλους τους υπολογισμούς με βάση αυτή την αριθμητική.

3.4.6 Αποτελέσματα Σύγκρισης

Ύστερα από την μελέτη που παρουσιάστηκε στην προηγούμενη υποενότητα, καταλήξαμε σε 2 αριθμητικές με πολύ καλή ακρίβεια αποτελεσμάτων συγκρίσιμη με αυτή της 64-bit floating point. Η επιλογή της αριθμητικής όμως, δεν βασίζεται μόνο στην καλή ακρίβεια που μας παρέχουν. Πρέπει να μελετήσουμε και τι κατανάλωση πόρων έχουμε με τη χρήση της κάθε μίας, που είναι και το βασικό κριτήριο της μελέτης. Επίσης, ένα ακόμα κριτήριο είναι και η συχνότητα του ρολογιού. Μέλημα μας πέρα από τη χαμηλή κατανάλωση πόρων, είναι η κατασκευή μιας γρήγορης αρχιτεκτονικής που θα μπορεί αποδώσει ικανοποιητικά στην επεξεργασία των δεδομένων.

Η μελέτη κατανάλωσης των πόρων έγινε χρησιμοποιώντας τα έτοιμα cores της Xilinx για κάθε αριθμητική μονάδα, πάνω στην πλακέτα Virtex 5 (*XC5VLX330*), την οποία μάλιστα χρησιμοποιούμε για την κατασκευή της αρχιτεκτονικής στο επόμενο κεφάλαιο. Η σύγκριση

έγινε χρησιμοποιώντας όλες τις δυνατές επιλογές που μας παρείχε η Xilinx και αφορούν την ίδια καθυστέρηση εκτέλεσης. Παρακάτω, παρουσιάζονται σχηματικά ορισμένες συγκρίσεις αριθμητικών μονάδων για διαφορετικές καθυστερήσεις.

Adder Latency: 11 cycles	32-bit floating point (High Speed)	32-bit floating point (Low Latency)	52-bit ffixed point
Number of Slice registers	515	no	164
Number of Slice LUTs	391	no	209
Number of Occupied Slices	179	no	86
Number of LUT Flip Flop pairs used	487	no	214
Maximum Frequency(MHz)	452.284	no	589.970

Πίνακας 3.1: Σύγκριση πόρων μεταξύ 32 bit floating point και fixed point αριθμητική για αθροιστές με καθυστέρηση 11 κύκλων.

Adder Latency: 7 cycles	32-bit floating point (High Speed)	32-bit floating point (Low Latency)	52-bit ffixed point
Number of Slice registers	358	542	147
Number of Slice LUTs	375	467	185
Number of Occupied Slices	154	190	62
Number of LUT Flip Flop pairs used	436	538	190
Maximum Frequency(MHz)	332.668	411.100	589.970

Πίνακας 3.2: Σύγκριση πόρων μεταξύ 32 bit floating point και fixed point αριθμητική για αθροιστές με καθυστέρηση 7 κύκλων.

Adder Latency: 3 cycles	32-bit floating point (High Speed)	32-bit floating point (Low Latency)	52-bit ffixed point
Number of Slice registers	138	229	107
Number of Slice LUTs	396	500	126
Number of Occupied Slices	147	186	86
Number of LUT Flip Flop pairs used	402	505	139
Maximum Frequency(MHz)	239.222	267.523	584.112

Πίνακας 3.3: Σύγκριση πόρων μεταξύ 32 bit floating point και fixed point αριθμητική για αθροιστές με καθυστέρηση 3 κύκλων.

Multiplier Latency: 8 cycles	32-bit floating point	52-bit fixed point (Speed Optimized)	52-bit fixed point (Area Optimized)
Number of Slice registers	672	1.404	1.942
Number of Slice LUTs	622	1.462	1.743
Number of Occupied Slices	218	421	504
Number of LUT Flip Flop pairs used	665	1.462	1.783
Maximum Frequency(MHz)	448.029	352.237	17.657

Πίνακας 3.4: Σύγκριση πόρων μεταξύ 32 bit floating point και fixed point αριθμητική για πολλαπλασιαστές με καθυστέρηση 8 κύκλων.

Multiplier Latency: 6 cycles	32-bit floating point	52-bit fixed point (Speed Optimized)	52-bit fixed point (Area Optimized)
Number of Slice registers	587	1.378	1.942
Number of Slice LUTs	627	1.420	1.696
Number of Occupied Slices	208	413	505
Number of LUT Flip Flop pairs used	648	1.431	1.752
Maximum Frequency(MHz)	350.447	419.938	17.657

Πίνακας 3.5: Σύγκριση πόρων μεταξύ 32 bit floating point και fixed point αριθμητική για πολλαπλασιαστές με καθυστέρηση 6 κύκλων.

Multiplier Latency: 4 cycles	32-bit floating point	52-bit fixed point (Speed Optimized)	52-bit fixed point (Area Optimized)
Number of Slice registers	428	1.013	1.836
Number of Slice LUTs	584	1.388	1.687
Number of Occupied Slices	194	371	466
Number of LUT Flip Flop pairs used	616	1.394	1.687
Maximum Frequency(MHz)	293.815	293.729	17.448

Πίνακας 3.6: Σύγκριση πόρων μεταξύ 32 bit floating point και fixed point αριθμητική για πολλαπλασιαστές με καθυστέρηση 4 κύκλων.

Divider Latency: 28 cycles	32-bit floating point	52-bit fixed point
Number of Slice registers	1.320	1.048
Number of Slice LUTs	817	1.043
Number of Occupied Slices	330	447
Number of LUT Flip Flop pairs used	1.043	1.195
Number of RAMB18E1/FIFO18E1s	0	1
Number of DSP48E1s	0	18
Maximum Frequency(MHz)	486.263	181.785

Πίνακας 3.7: Σύγκριση πόρων μεταξύ 32 bit floating point και fixed point αριθμητική για διαιρέτες με καθυστέρηση 28 κύκλων.

Divider Latency: 20 cycles	32-bit floating point	52-bit fixed point
Number of Slice registers	684	862
Number of Slice LUTs	806	921
Number of Occupied Slices	317	361
Number of LUT Flip Flop pairs used	935	1.063
Number of RAMB18E1/FIFO18E1s	0	1
Number of DSP48E1s	0	18
Maximum Frequency(MHz)	276.171	162.522

Πίνακας 3.8: Σύγκριση πόρων μεταξύ 32 bit floating point και fixed point αριθμητική για διαιρέτες με καθυστέρηση 20 κύκλων.

Divider Latency: 14 cycles	32-bit floating point	52-bit fixed point
Number of Slice registers	651	445
Number of Slice LUTs	775	881
Number of Occupied Slices	296	368
Number of LUT Flip Flop pairs used	916	1.017
Number of RAMB18E1/FIFO18E1s	0	1
Number of DSP48E1s	0	18
Maximum Frequency(MHz)	276.171	57.156

Πίνακας 3.9: Σύγκριση πόρων μεταξύ 32 bit floating point και fixed point αριθμητική για διαιρέτες με καθυστέρηση 14 κύκλων.

Παρατηρούμε ότι από άποψη πόρων και συχνότητας ρολογιού η fixed point αριθμητική υπερισχύει μόνο στις μονάδες πρόσθεσης. Για πολλαπλασιασμούς και διαιρέσεις προτιμάμε την 32-bit floating point αριθμητική. Ειδικά στις μονάδες διαίρεσης, η fixed point αριθμητική είναι πολύ κατώτερη των περιστάσεων και από άποψη πόρων και ταχύτητας. Το γεγονός ότι στην αρχιτεκτονική χρησιμοποιούνται περισσότεροι πολλαπλασιαστές και διαιρέτες σε σχέση με αθροιστές είναι ένα σημαντικό πλεονέκτημα για την 32-bit floating point. Επίσης, δεν ξεχνάμε το γεγονός ότι οι 52 bit μνήμες κοστίζουν περισσότερο από τις 32 bit. Ειδικά όταν πόροι της μνήμης αποτελούν τον πιο σημαντικό παράγοντα της υλοποίησης μας, όπως θα δούμε

στο επόμενο κεφάλαιο, πρέπει να το λάβουμε καλά υπόψην μας.

Σχετικά με την ακρίβεια, παρόλο που και οι 2 αριθμητικές εμφανίζουν πολύ καλή ακρίβεια (*μην ξεχνάμε ότι η συσχέτιση των σφαλμάτων έγινε σε σχέση με την 64-bit floating point αριθμητική, η οποία παρουσιάζει πάρα πολύ μικρό σφάλμα*), αν θέλουμε όμως να είμαστε λεπτολόγοι, μπορούμε να πούμε ότι η 32 bit floating point έχει ελάχιστα καλύτερη συμπεριφορά απο την fixed point, όπως άλλωστε είδαμε και στα σχήματα παραπάνω. Βέβαια, μπορούμε να αυξήσουμε και άλλο την ακρίβεια της fixed point βάζοντας περισσότερα bits, αλλά είναι ανούσιο διότι έτσι θα αυξηθούν ταυτόχρονα οι πόροι οι οποίοι παρουσιάστηκαν ήδη αυξημένοι, και από άποψη ακρίβειας, κινούμαστε σε τόσο λεπτά επίπεδα που δεν πρόκειται να δούμε κάποια σημαντική διαφορά.

Σκεπτόμενοι όλα τα παραπάνω καταλήξαμε στην επιλογή της 32-bit floating point αριθμητικής. Μας κάλυπτε από όλες τις πλευρές, ακρίβεια, κατανάλωση πόρων και συχνότητας ρολογιού. Το αποτέλεσμα της μελέτης για την αριθμητική ακρίβεια απέδωσε καρπούς και καταφέραμε να αποφύγουμε τη χρήση της 64-bit floating point, αποκομίζοντας πολλά πλεονεκτήματα για την υλοποίηση μας.

Κεφάλαιο 4

Υλοποίηση Αρχιτεκτονικής

Αυτό το κεφάλαιο αναφέρεται στη σχεδίαση της αρχιτεκτονικής, τις διάφορες τεχνικές που χρησιμοποιήθηκαν και την γενικότερη μελέτη σχετικά με το latency, το throuput και την εξοικονόμηση πόρων.

Σκεπτόμενοι το είδος του προβλήματος το οποίο καλούμασταν να αντιμετωπίσουμε, καταλήξαμε σε σημαντικές παρατηρήσεις. Πρόκειται για ένα πρόβλημα με πολύ μεγάλο αριθμό εξόδων, οπότε το να δώσουμε βάση στο throughput ήταν πρωτίστης σημασίας. Επίσης, ο μεγάλος αριθμός φυσικών παραγόντων ως είσοδοι και η πολυπλοκότητα της επεξεργασίας όλων αυτών, το καθιστούσαν ως ενά «βαρύ» μοντέλο. Αυτό είχε σαν αποτέλεσμα και την αργή απόκριση του σε μεγάλο φόρτο δεδομένων. Οπότε πέρα από το throughput, θέλαμε να σχεδιάσουμε και μια γρήγορη αρχιτεκτονική με στόχο την ταχύτερη επεξεργασία των δεδομένων. Τέλος, πρόκειται για ένα μοντέλο το οποίο αντλεί συνέχεια δεδομένα για την επίλυση πολύ μεγάλων πλεγμάτων. Το πόσο μεγάλα πλέγματα μπορεί να επιλύσει μια FPGA ή ένας συνδιασμός από FPGAs εξαρτάται από τι πόρους έχουμε στη διάθεση μας. Άρα και η εξοικονόμηση πόρων αποτελεί πολύ σημαντικό παράγοντα, με τον οποίο πορευτήκαμε για τη σχεδίαση της αρχιτεκτονικής.

Η σχεδίαση της αρχιτεκτονικής χωρίζεται σε 3 φάσεις :

1. Στη σχεδίαση των υπολογιστικών μονάδων για την επίλυση των συστημάτων εξισώσεων.
2. Στη σχεδίαση των μνημών.
3. Στη σχεδίαση των μονάδων ελέγχου και το συχρονισμό της αρχιτεκτονικής.

4.1 Σχεδίαση Αριθμητικών Μονάδων

Σε αυτή την ενότητα επικεντρωθήκαμε στην κατασκευή των μονάδων επεξεργασίας, που είναι υπεύθυνες για την επίλυση των διαφορικών εξισώσεων. Οι μονάδες αυτές, έπρεπε να είναι σχεδιασμένες με τρόπο ώστε να έχουμε μια πολύ καλή ταχύτητα υλοποίησης. Ασχοληθήκαμε με το φτιάξουμε κάτι λειτουργικό το οποίο θα έτρεχε με μια ικανοποιητικά γρήγορη συχνότητα ρολογιού και θα απαιτούσε τους λιγότερο δυνατούς πόρους.

Για να πετύχουμε υψηλή ταχύτητα ρολογιού αποφασίσαμε να στηριχτούμε σε pipeline υλοποιήση. Όλες οι αριθμητικές μονάδες ήταν τύπου 32 bit floating point(*IEEE 754*) και για την κατασκευή τους χρησιμοποιήυθηκαν τα έτοιμα cores της Xilinx.Οι αριθμητικές μονάδες που κατασκευάστηκαν, αποτελούνταν από αθροιστές/αφαιρέτες, πολλαπλασιαστές, διαιρέτες

Αριθμητικές Μονάδες	Αθροιστές/Αφαιρέτες	Πολλαπλασιαστές	Διαιρέτες
Number of Slices	202	65	415
Number of Slice registers	547	188	1352
Number of Slice LUTs	503	103	769
Number of DSP48E1s	0	2	0

Πίνακας 4.1: Κατανάλωση πόρων από Virtex5 330 για κάθε είδος αριθμητικής μονάδας της αρχιτεκτονικής.

και συγκριτές. Για κάθε μονάδα από αυτές είχαμε την δυνατότητα να ορίζουμε τα στάδια του pipeline καθώς και το είδος των πόρων που θέλαμε να χρησιμοποήσουμε (*λογική ή DSPs*). Με κάθε αλλαγή που κάναμε, πειραματιζόμασταν με τη συχνότητα του ρολογιού και την κατανάλωση των πόρων.

Συγκεκριμένα, η κατανάλωση των πόρων για την κάθε αριθμητική μονάδα παρουσιάζεται στον παρακάτω πίνακα

Στη συνέχεια, αφού είχαμε υλοποιήσει τις αριθμητικές μονάδες με την διαδικασία που παρουσιάστηκε παραπάνω, προχωρήσαμε στην κατασκευή της αρχιτεκτονικής των μονάδων επεξεργασίας. Οι μονάδες επεξεργασίας είναι υπεύθυνες για την επίλυση των συστημάτων εξισώσεων της συνάρτησης, μέσω μαθηματικών πράξεων. Η αρχιτεκτονική τους είναι τύπου pipeline, με τη χρήση των απαραίτητων καταχωρητών για τη σωστή λειτουργία και το συγχρονισμό τους. Το throuput για κάθε μονάδα επεξεργασίας εκτιμάται σε ένα αποτέλεσμα ανά κύκλο.

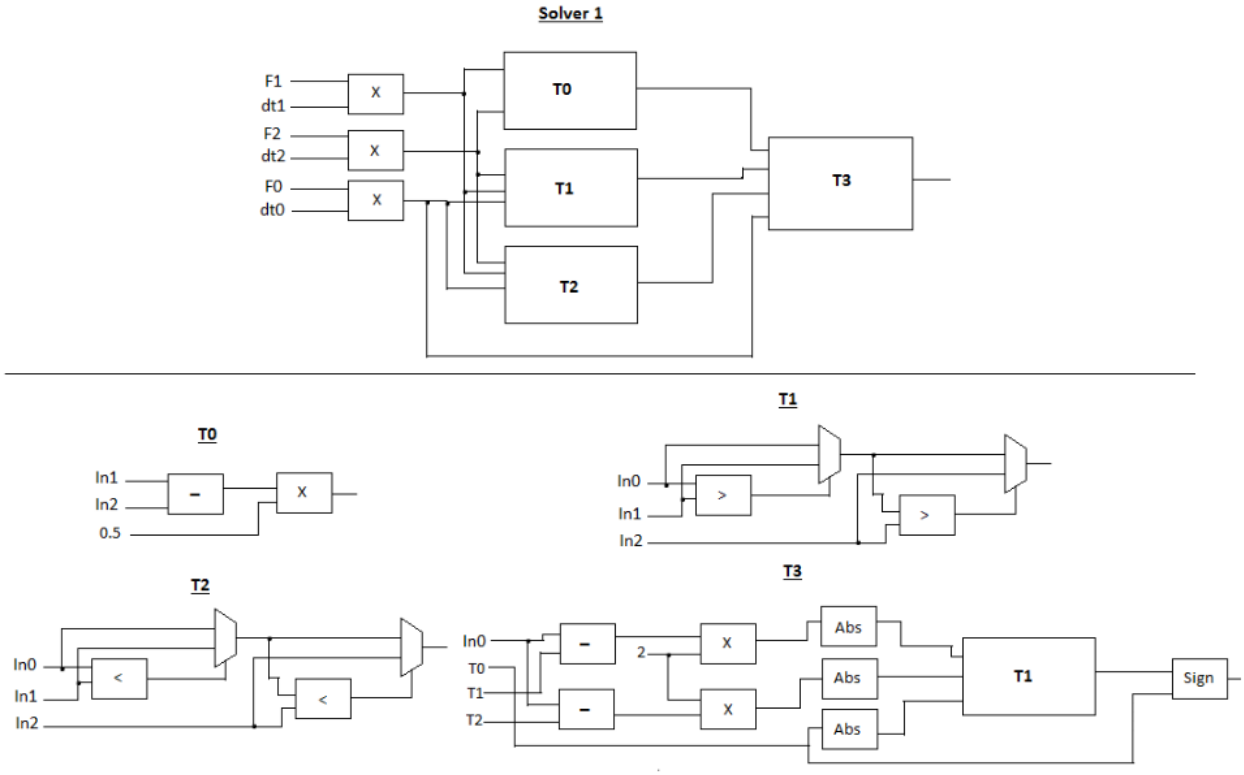
Παρακάτω παρουσιάζονται οι υλοποιήσεις των μονάδων επεξεργασίας, με βάση τις εξισώσεις των συστημάτων εξισώσεων που χρησιμοποιεί το μοντέλο.

$$delfiavg(i,j,k) = 0.5 \cdot (f(i+1,j,k) \cdot dt(i+1,j) - f(i-1,j,k) \cdot dt(i-1,j))$$

$$fimin(i,j,k) = min((f(i-1,j,k) \cdot dt(i-1,j)), (f(i,j,k) \cdot dt(i,j)), (f(i+1,j,k) \cdot dt(i+1,j)))$$

$$fimax(i,j,k) = max((f(i-1,j,k) \cdot dt(i-1,j)), (f(i,j,k) \cdot dt(i,j)), (f(i+1,j,k) \cdot dt(i+1,j)))$$

$$delfi(i,j,k) = sign(1.0, delfiavg(i,j,k)) \cdot min(|delfiavg(i,j,k)|, 2 \cdot dim((f(i,j,k) \cdot dt(i,j)), fimin(i,j,k)), 2 \cdot dim(fimax(i,j,k), (f(i,j,k) \cdot dt(i,j))))$$



Σχήμα 4.1: Υλοποίηση της Solver 1 μονάδας επεξεργασίας

if *(u(i,j,k)>=0)* **then**

$$co = 0.5 \cdot (1 - \frac{u(i,j,k) \cdot dtadv}{dx(i-1,j)}))$$

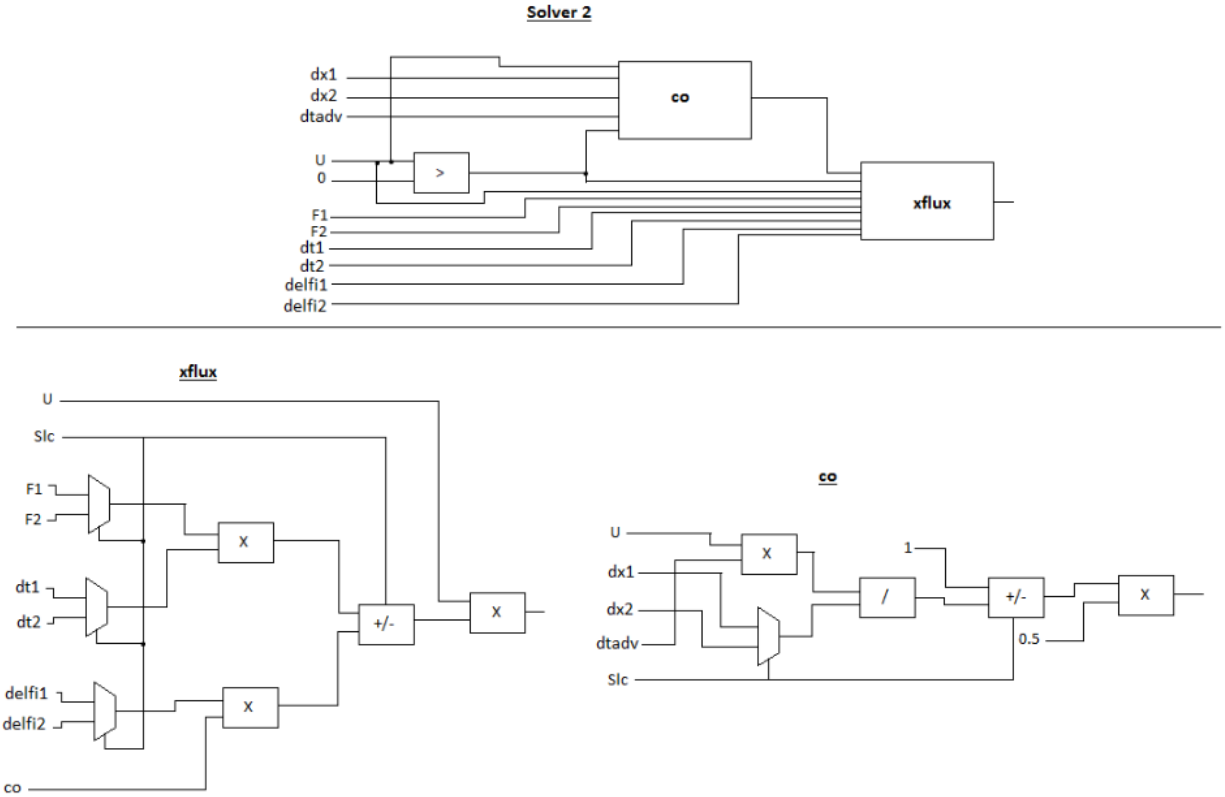
$$xflux(i,j,k) = u(i,j,k) \cdot (f(i-1,j,k) \cdot dt(i-1,j) + delfi(i-1,j,k) \cdot co))$$

else

$$co = 0.5 \cdot (1 + \frac{u(i,j,k) \cdot dtadv}{dx(i,j)}))$$

$$xflux(i,j,k) = u(i,j,k) \cdot (f(i,j,k) \cdot dt(i,j) - delfi(i,j,k) \cdot co)$$

end



Σχήμα 4.2: Υλοποίηση της Solver 2 μονάδας επεξεργασίας

if *(w(i,j,k)>=0)* **then**

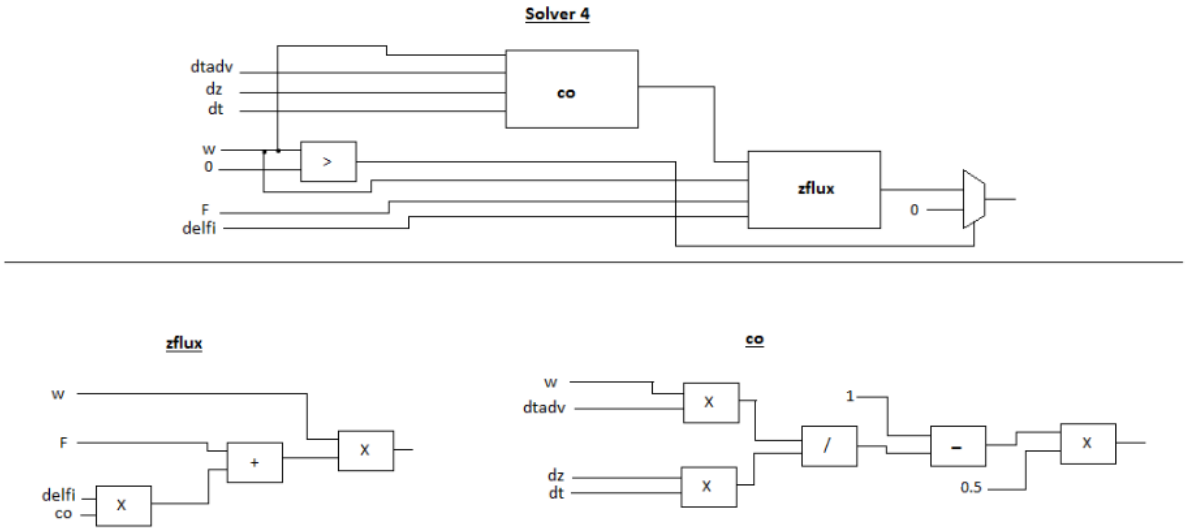
$co = 0.5 \cdot (1 - \frac{w(i,j,k) \cdot dtadv}{dz(k) \cdot dt(i,j)}))$

$zflux(i,j,k) = w(i,j,k) \cdot (f(i,j,k) + delfi(i,j,k) \cdot co))$

else

$zflux(i,j,k) = 0$

end



Σχήμα 4.3: Υλοποίηση της Solver 4 μονάδας επεξεργασίας

if *(w(i,j,k)>=0)* **then**

$$co = 0.5 \cdot (1 - \frac{w(i,j,k) \cdot dtadv}{dz(k) \cdot dt(i,j)}))$$

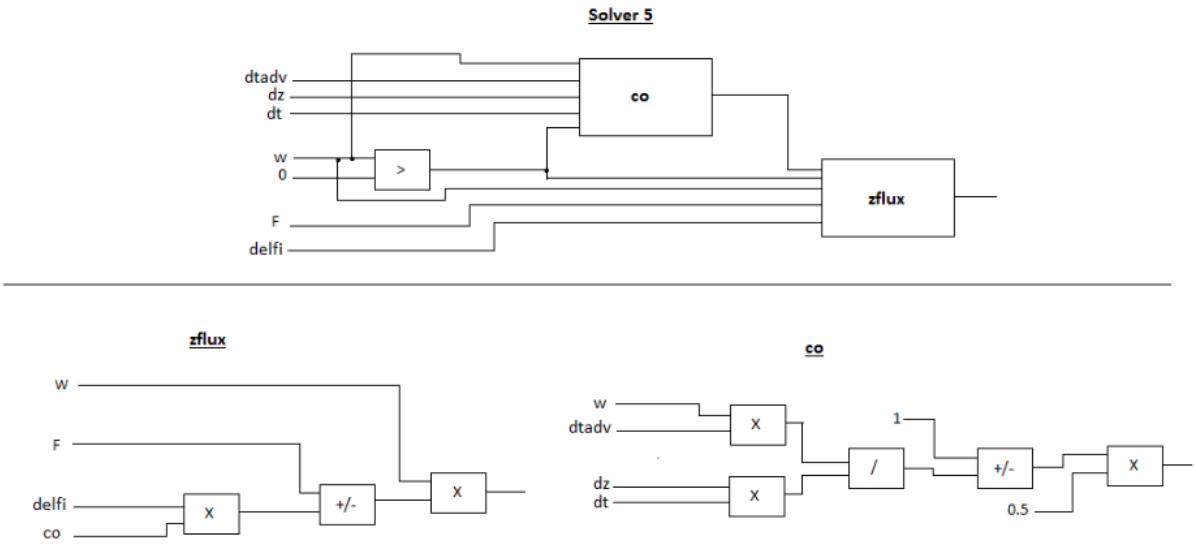
$$zflux(i,j,k) = w(i,j,k) \cdot (f(i,j,k) + delfi(i,j,k) \cdot co))$$

else

$$co = 0.5 \cdot (1 + \frac{w(i,j,k) \cdot dtadv}{dz(k-1) \cdot dt(i,j)}))$$

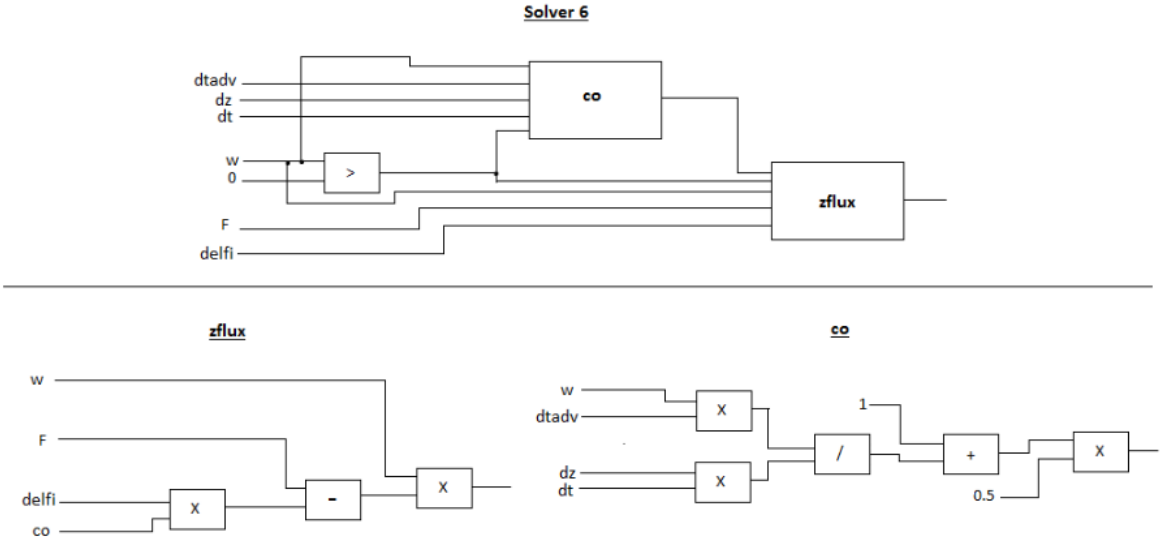
$$zflux(i,j,k) = w(i,j,k) \cdot (f(i,j,k-1) - delfi(i,j,k) \cdot co)$$

end



Σχήμα 4.4: Υλοποίηση της Solver 5 μονάδας επεξεργασίας

if *(u(i,j,k)>=0)* **then**
$zflux(i,j,k) = 0$
else
$co = 0.5 \cdot (1 + \frac{w(i,j,k) \cdot dtadv}{dz(k-1) \cdot dt(i,j)}))$
$zflux(i,j,k) = w(i,j,k) \cdot (f(i,j,k-1) - delfi(i,j,k-1) \cdot co)$
end



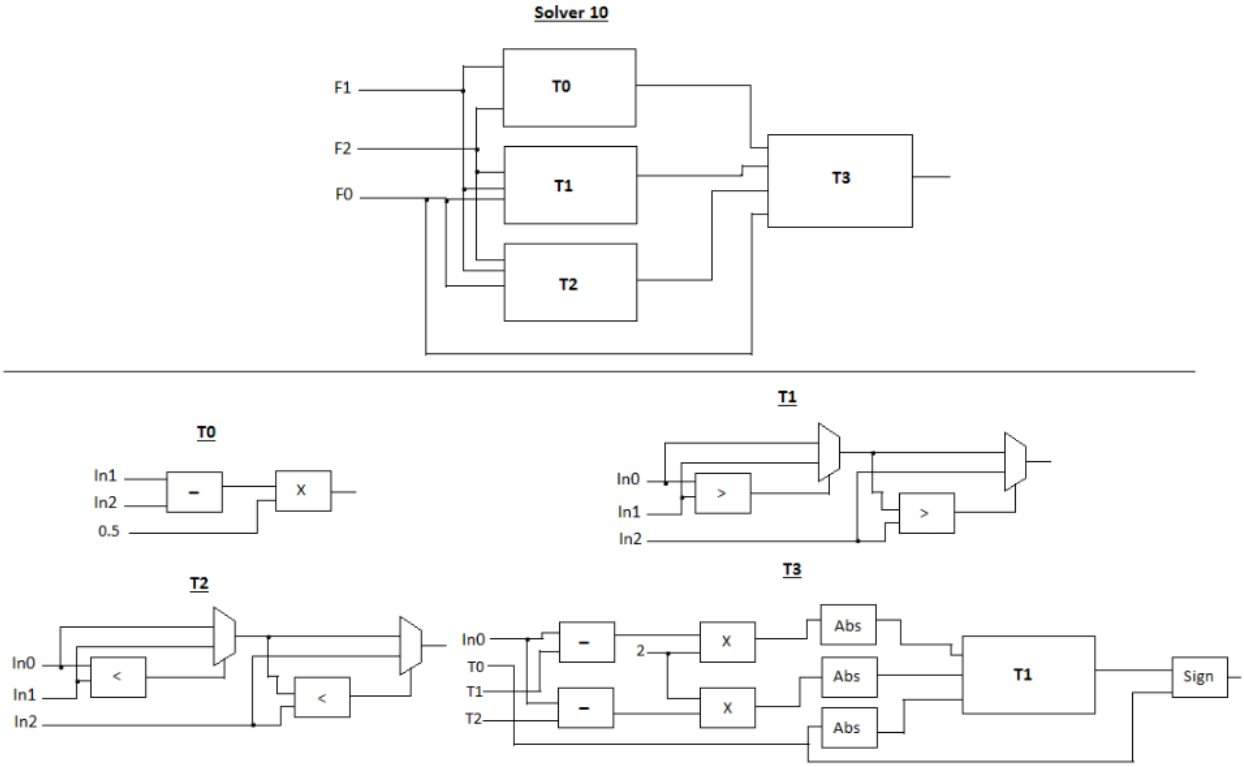
Σχήμα 4.5: Υλοποίηση της Solver 6 μονάδας επεξεργασίας

$$delfiavg(i,j,k) = 0.5 \cdot (f(i+1,j,k) - f(i-1,j,k))$$

$$fimin(i,j,k) = min(f(i-1,j,k), f(i,j,k), f(i+1,j,k))$$

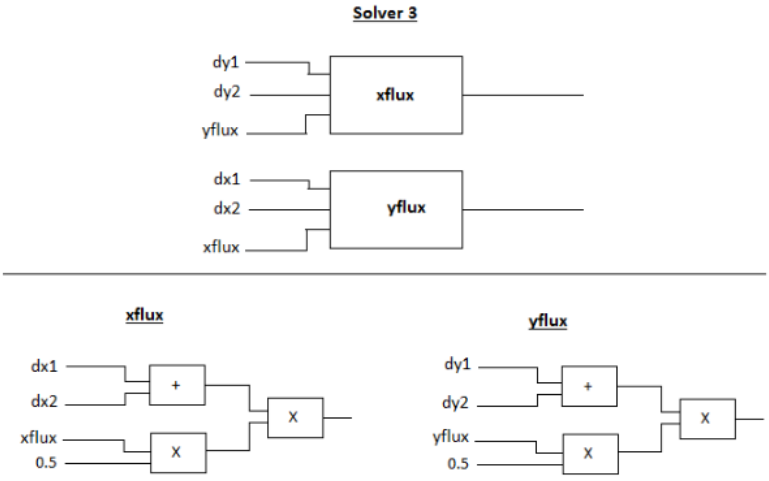
$$fimax(i,j,k) = max(f(i-1,j,k), f(i,j,k), f(i+1,j,k))$$

$$delfi(i,j,k) = sign(1.0, delfiavg(i,j,k)) \cdot min(|delfiavg(i,j,k)|,$$
$$2 \cdot dim(f(i,j,k), fimin(i,j,k)), 2 \cdot dim(fimax(i,j,k), f(i,j,k)))$$



Σχήμα 4.6: Υλοποίηση της Solver 10 μονάδας επεξεργασίας

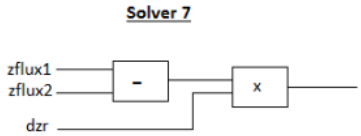
$$xflux(i,j,k) = 0.5 \cdot (dy(i,j) + dy(i-1,j)) \cdot xflux(i,j,k)$$
$$yflux(i,j,k) = 0.5 \cdot (dx(i,j) + dx(i,j-1)) \cdot yflux(i,j,k)$$



Σχήμα 4.7: Υλοποίηση της Solver 3 μονάδας επεξεργασίας

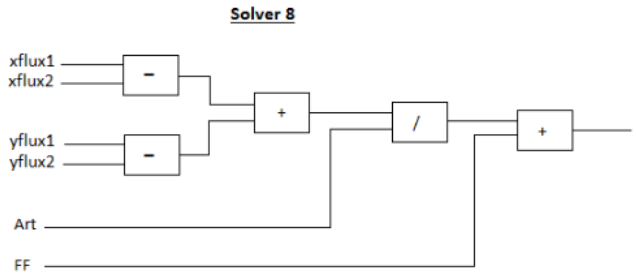
.

$$ff(i,j,k) = dzr(k) \cdot (zflux(i,j,k) - zflux(i,j,k+1))$$



Σχήμα 4.8: Υλοποίηση της Solver 7 μονάδας επεξεργασίας

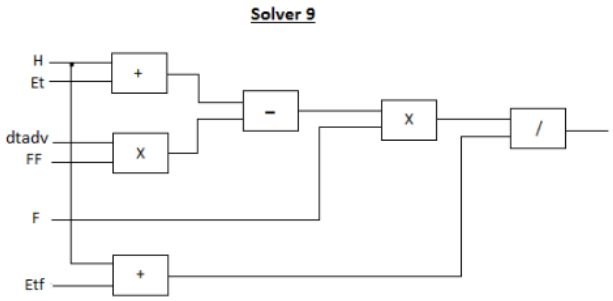
$$ff(i,j,k) = ff(i,j,k) + \frac{xflux(i+1,j,k) - xflux(i,j,k) + yflux(i,j+1,k) - yflux(i,j,k)}{art(i,j)}$$



Σχήμα 4.9: Υλοποίηση της Solver 8 μονάδας επεξεργασίας

.

$$ff(i,j,k) = \frac{f(i,j,k) \cdot (h(i,j) + et(i,j)) - dtadv \cdot ff(i,j,k)}{h(i,j) + rtf(i,j)}$$



Σχήμα 4.10: Υλοποίηση της Solver 9 μονάδας επεξεργασίας

4.2 Σχεδίαση Μνημών

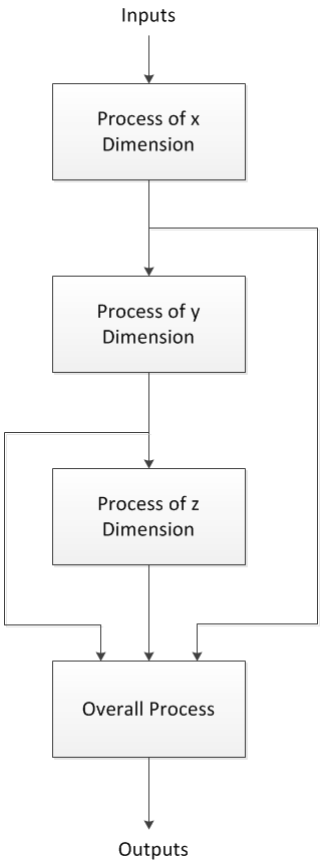
Λόγω του γεγονότος της χρήσης πολλών δεδομένων εισόδων, έπρεπε να έχουμε στη διάθεση μας πολλές μνήμες για την αποθηκευση και την επεξεργασία τους. Στην Fortran αυτά τα δεδομένα είναι αποθηκευμένα σε πίνακες 3 διαστάσεων, 2 διαστάσεων και 1 διάστασης. Εμείς καλούμαστε να υλοποιήσουμε την αποθήκευση τους σε μνήμες με τον πιο αποδοτικό τρόπο και την χρήση τους με τον πιο γρήγορο.

4.2.1 Τεχνικές Κατασκευής Μνημών

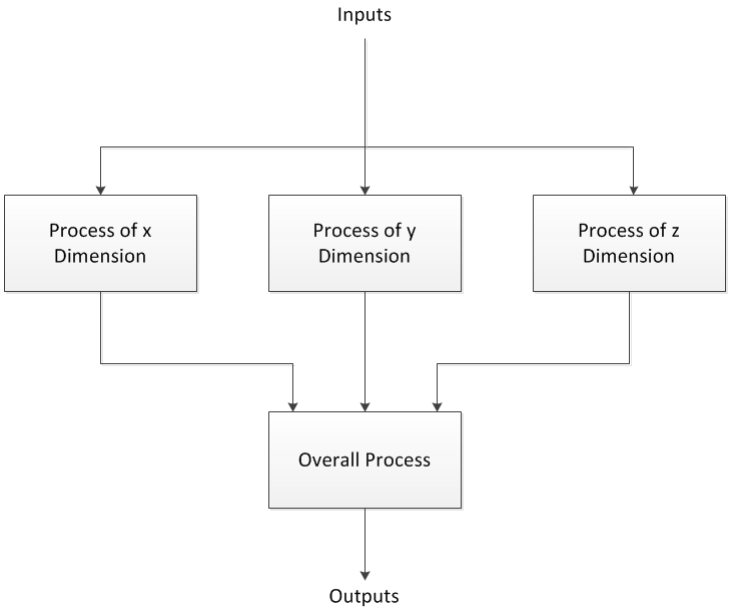
Αρχικά η υλοποίηση τους είχε γίνει μόνο με μνήμες βασισμένες σε block rams. Αλλά λόγω του μεγάλου φόρτου δεδομένων που είχαμε, παρατηρήθηκε πολύ μεγάλη σπατάλη των πόρων αυτού του είδους. Βλέποντας την σπατάλη αυτή και σε συνδιασμό με την ελάχιστη χρήση πόρων λογικής, αποφασίσαμε πολλές από αυτές να τις υλοποιήσουμε με τη χρήση λογικής. Έτσι οι μεγαλύτερες μνήμες υλοποιήθηκαν με block rams και οι μικρότερες με λογική. Με αυτό τον τρόπο καταφέραμε να μειώσουμε τον αριθμό των block rams κατά ένα σημαντικό ποσοστό. Η ανάγκη για διατήρηση ενός καλού throughput (*1 αποτέλεσμα ανά κύκλο*) και ενός χαμηλού latency μας οδήγησε στην ανάγκη ανάπτυξης διαφόρων τεχνικών.

Σύμφωνα με το πρόβλημα, είχαμε τον υπολογισμός της ροής του ρευστού στις 3 διαστάσεις (x,y,z). Η απαίτηση για μια γρήγορη αρχιτεκτονική, μας οδήγησε στην εκμετάλευση παραλληλισμού όπου υπήρχε η δυνατότητα. Παρατηρήθηκε ότι ο υπολογισμός της ροής ύδατος για κάθε διάσταση είναι μια διαδικασία ανεξάρτητη από τις υπόλοιπες διαστάσεις. Συνεπώς, καταλήξαμε στο γεγονός της παράλληλης και ανεξάρτητης επεξεργασίας των 3 διαστάσεων ταυτόχρονα (*σχήμα 4.11, 4.12*). Με αυτό τον τρόπο, καταφέραμε να αποφύγουμε πολλούς κύκλους καθυστέρησης σε σχέση με την αρχική σειριακή υλοποίηση.

Εκτός από την ταχύτητα υλοποίησης, επωφελούμαστε και σε άλλους τομείς. Ένας από αυτούς είναι η στην αποφυγή χρήσης buffers για τα δεδομένα που χρησιμοποιήθηκαν από την επεξεργασία της μίας διάστασης και θέλουμε να τα ξαναχρησιμοποιήσουμε για την επεξεργασία και των υπόλοιπων διαστάσεων,(*που βρίσκονται πιο κάτω στην ιεραρχία σύμφωνα με τη ροή του προγράμματος*) αν ακολουθούσαμε το σειριακό μοντέλο. Επίσης, δεν χρησιμοποιούμε buffers για την αποθήκευση των αποτελεσμάτων κάθε διάστασης, μέχρι να υπολογιστούν και τα αποτελέσματα των υπόλοιπων διαστάσεων, τα οποία στη συνέχεια επεξεργάζονται όλα μαζί σε μετέπειτα μονάδες επεξεργασίας. Έτσι καταφέραμε να εξοικονομήσουμε αρκετούς πόρους και η πολυπλοκότητα επεξεργασίας των δεδομένων απλοποιήθηκε σε σημαντικό βαθμό.



Σχήμα 4.11: Σειριακή επεξεργασία των διαστάσεων



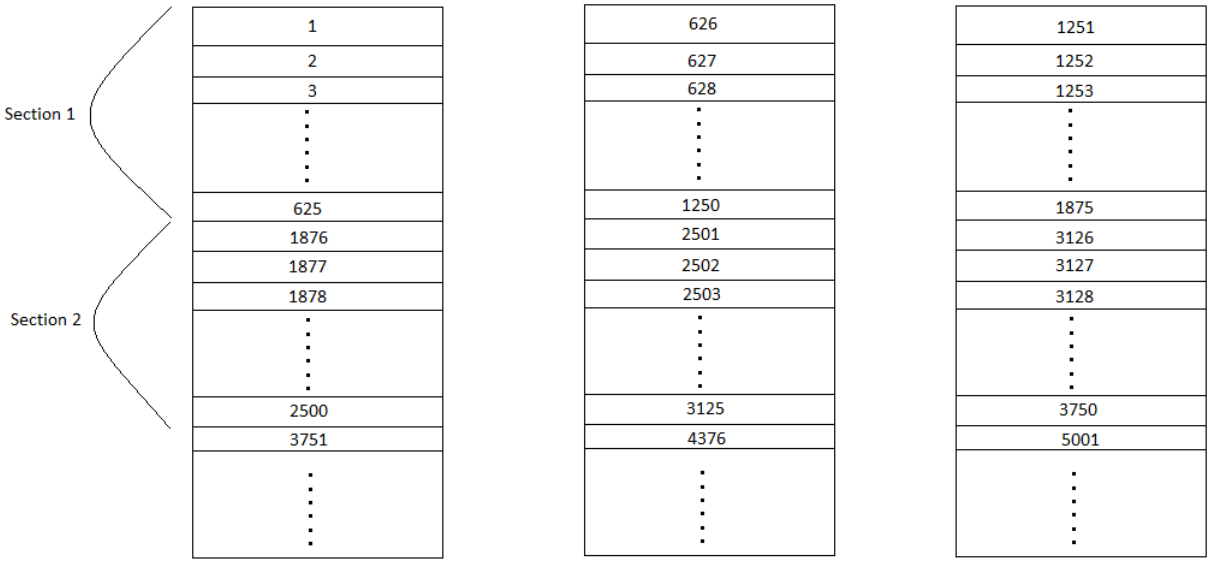
Σχήμα 4.12: Παράλληλη επεξεργασία των διαστάσεων

Επειδή για κάθε διάσταση γίνεται διαφορετική διάσχιση των πινάκων δεδομένων (*άρα και των*

μνημών μας) την ίδια χρονική στιγμή, για τους κοινούς προσπελάσιμους πίνακες φτιάχνουμε 3 αντίγραφα, ένα για κάθε διάσταση. Είναι επιτακτική η ανάγκη των αντιγράφων, διότι για τον ίδιο πίνακα δεν διασχίζουμε τα ίδια δεδομένα σε κάθε χρονική στιγμή. Πολλά από τα δεδομένα μάλιστα, που πέρνουμε για την επεξεργασία της μίας διάστασης δεν χρησιμοποιούνται για την επεξεργασία των υπόλοιπων διαστάσεων και αντίστροφα. Οπότε για να μπορέσουμε να πετύχουμε την ταυτόχρονη επεξεργασία όλων των διαστάσεων, μας διευκολύνει πάρα πολύ η δημιουργία αντιγράφων. Διαφορετικά, θα είχαμε υψηλή πολυπλοκότητα για τον έλεγχο τους, πολλές ανεπιθύμητες καθυστερήσεις και σπατάλη πόρων για το συγχρονισμό τους.

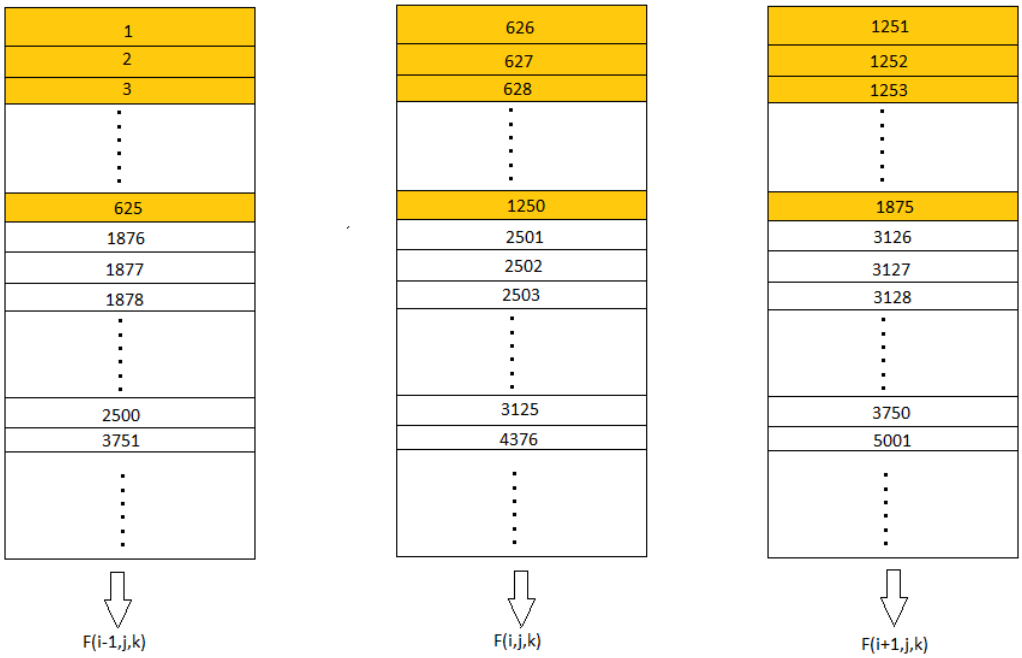
Μια άλλη μια τεχνική που χρησιμοποιήθηκε, είναι το σπάσιμο των μνημών σε 3 μικότερες συνολικού μεγέθους (*και των 3*) ίσου με τη αρχική. Αυτό έγινε λόγω του γεγονότος ότι στα συστήματα εξισώσεων που επιλύονται, χρειάζονται ως είσοδο ταυτόχρονα 3 διαφορετικές τιμές του ίδιου πίνακα (*π.χ. F(i-1,j,k) , F(i,j,k), F(i+1,j,k)*). Αν δεν γίνει αυτό, τότε δεν θα μπορούμε να έχουμε ένα αποτέλεσμα ανά κύκλο, αλλά ένα αποτέλεσμα ανά 3 κύκλους, ή ανά 2 αν χρησιμοποιήσουμε dual port μνήμες. Το σπάσιμο της μνήμης σε 3 μικρότερες έγινε με την εξής τεχνική :

Για την x διάσταση :



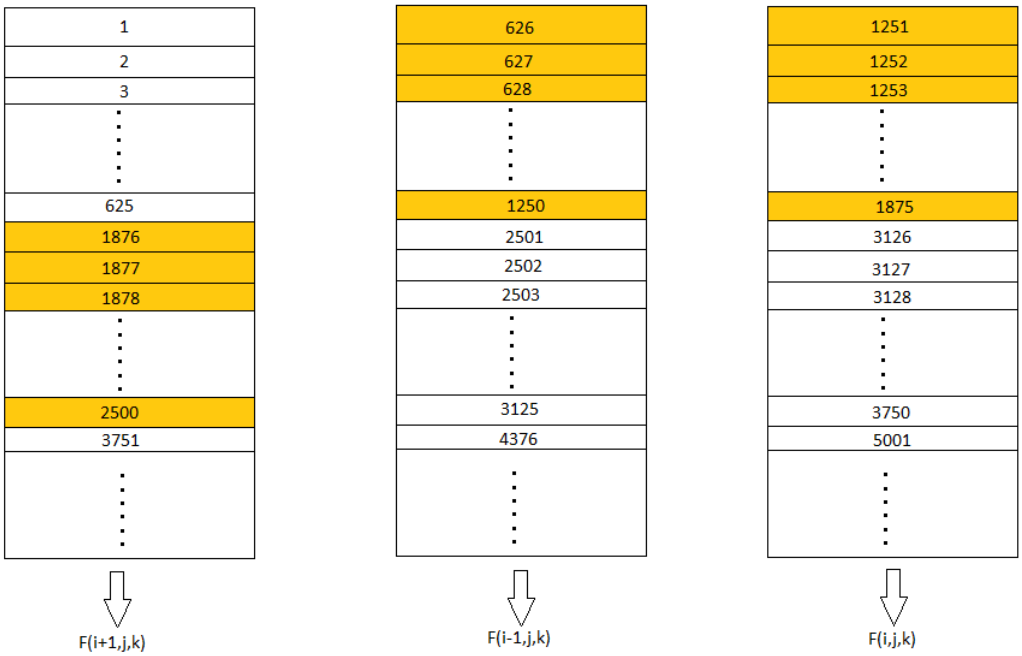
Σχήμα 4.13: Δομή σπάσμένης μνήμης που αναφέρεται σε κάποιο τρισδιάστατο πίνακα, για διάσχιση 3 ταυτόχρονα τιμών του ως προς την x διάσταση.

Θα παρουσιάσουμε τη διάσχιση των μνημών για την ταυτόχρονη προσπέλαση τους με ένα σχηματικό παράδειγμα.Έστω ότι έχουμε τον τρισδιάστατο πίνακα F και θέλουμε να πάρουμε ταυτόχρονα τις τιμές F(i,j,k), F(i+1,j,k) και F(i-1,j,k)



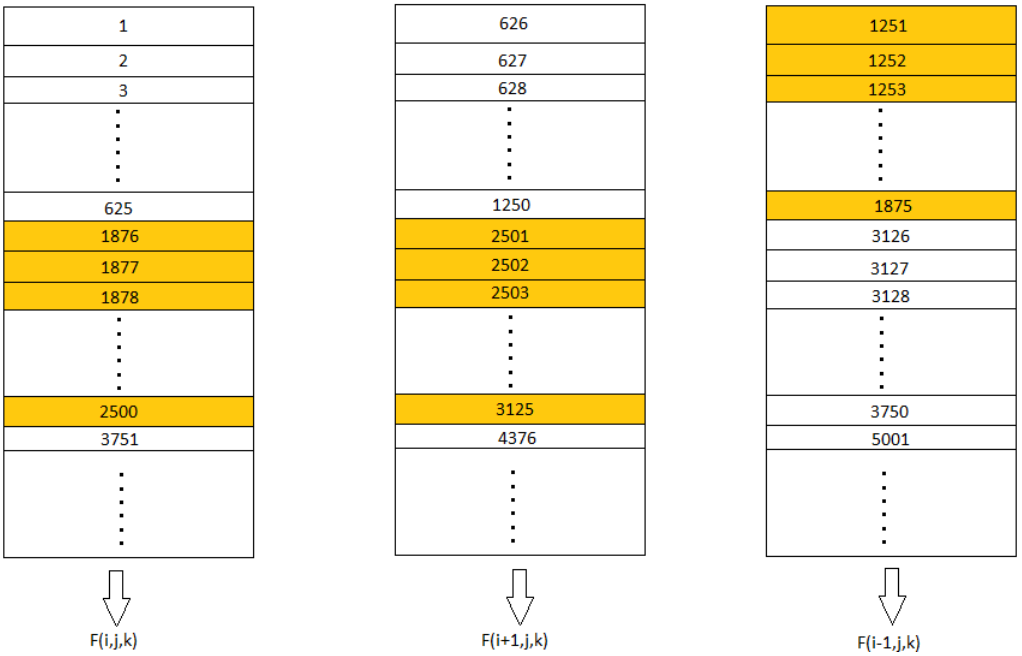
Σχήμα 4.14: Με πορτοκαλί χρώμα είναι το σύνολο των κελιών που αποτελούν ένα section. Αυτό το τμήμα χρησιμοποιείται για την προσπέλαση της κάθε μνήμης. Με άσπρο είναι το αχρησιμοποίητο τμήμα.

Όπως φαίνεται στο σχήμα 4.14 το σύνολο των σκιαγραφημένων με πορτοκαλί κελιών για κάθε μνήμη αποτελεί το section ,που είναι το τμήμα που χρησιμοποιείται από κάθε μνήμη, και με άσπρο χρώμα το σύνολο των αχρησιμοποίητων κελιών. Για τους πρώτους 625 κύκλους (*όσο το μέγεθος του section*) οι τιμές των F(i,j,k), F(i+1,j,k) και F(i-1,j,k) θα βγαίνουν από τις αντίστοιχες μνήμες όπως φένεται στο σχήμα 4.14. Σε κάθε κύκλο γίνεται προσπέλαση μίας μίας θέσης από τα sections μέχρι να φτάσει στην τελευτάια θέση όπου και γίνεται η αλλαγή των sections όπως φένεται παρακάτω σχήμα 4.15.



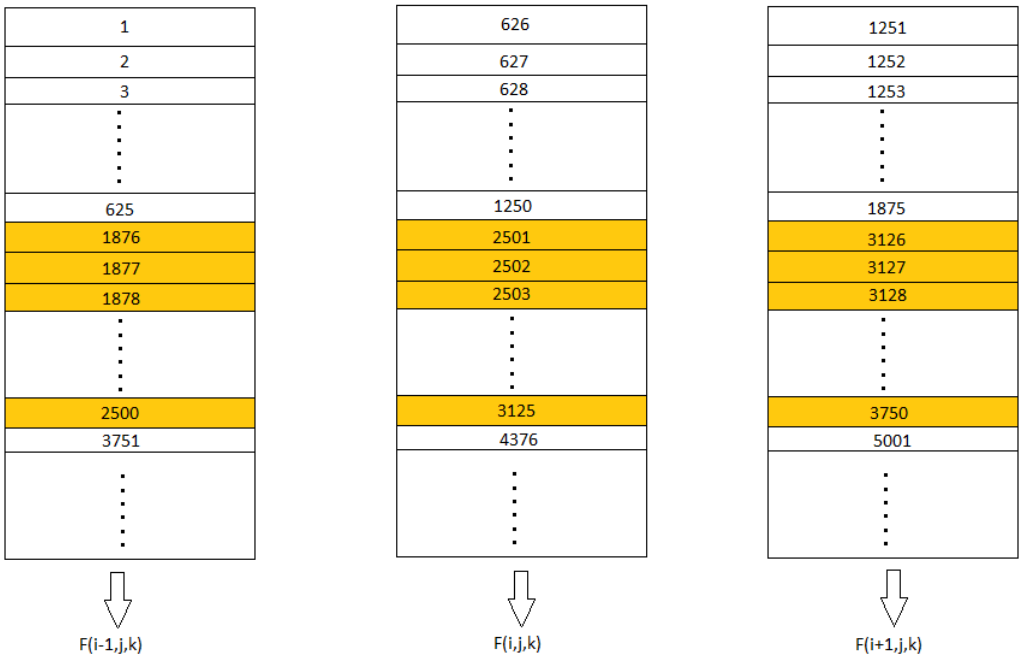
Σχήμα 4.15: Ύστερα από την πρώτη αλλαγή των sections.

Μετά την αλλαγή των sections προκύπτει το παραπάνω σχήμα. Για τους επόμενους 625 κύκλους παρατηρούμε ότι οι τιμές F(i,j,k), F(i+1,j,k) και F(i-1,j,k) βγαίνουν από διαφορετικά sections αυτή τη φορά.



Σχήμα 4.16: Ύστερα από την δεύτερη αλλαγή των sections.

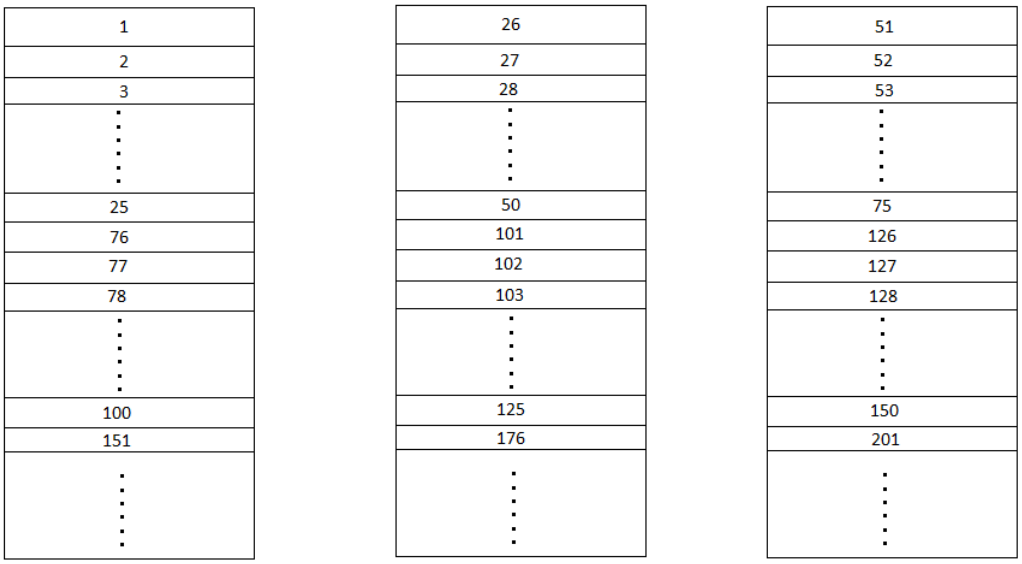
Ύστερα από ακόμη μια αλλαγή sections, αλλάζουν πάλι οι τιμές που παίρνουμε από κάθε μνήμη



Σχήμα 4.17: Ύστερα από την τρίτη αλλαγή των sections.

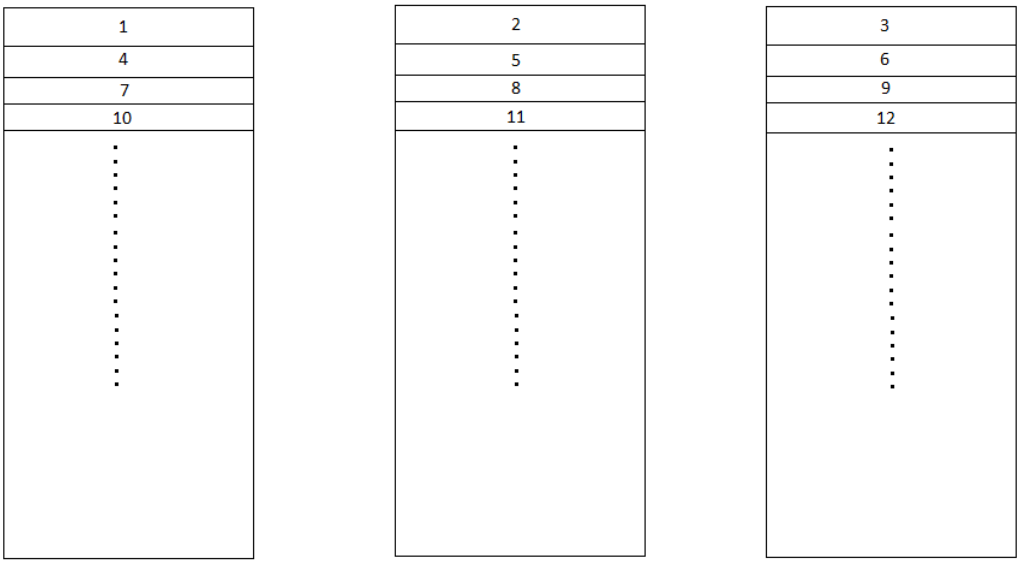
Τέλος, ξαναεπιστρέφουμε στην αρχική περίπτωση και η διαδικασία συνεχίζεται με τον τρόπο που αναλύσαμε. Συνολικά έχουμε 3 διαφορετικές καταστάσεις, κατά της οποίες λαμβάνει μέρος η διαδικασία της προσπέλασης των μνημών. Η διαδικασία τελειώνει όταν προσπελαστούν όλα τα δεδομένα των μνημών.

Ομοίως για την y διάσταση :



Σχήμα 4.18: Δομή σπάσμένης μνήμης που αναφέρεται σε κάποιο τρισδιάστατο πίνακα, για διάσχιση 3 ταυτόχρονα τιμών του ως προς την y διάσταση.

Ομοίως για την z διάσταση :

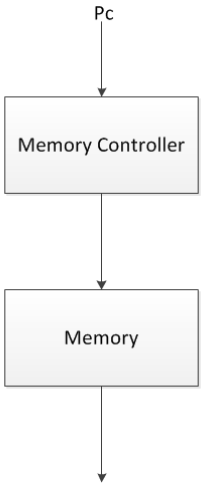


Σχήμα 4.19: Δομή σπάσμένης μνήμης που αναφέρεται σε κάποιο τρισδιάστατο πίνακα, για διάσχιση 3 ταυτόχρονα τιμών του ως προς την z διάσταση.

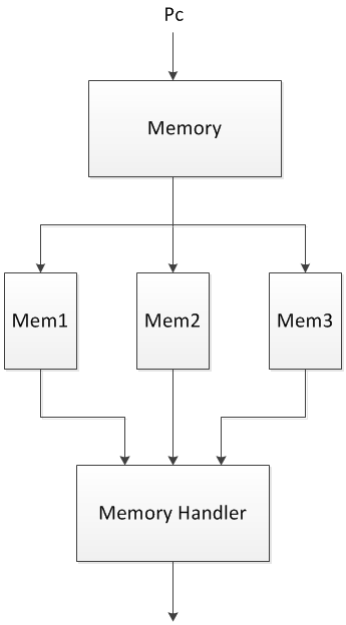
Το μέγεθος του κάθε section με το οποίο χωρίζουμε τις μνήμες καθορίζεται από την απόσταση του σημείου F(i,j,k) από το F(i+1,j,k) και F(i-1,j,k) για την x διάσταση, από το F(i,j+1,k) και F(i,j-1,k) για την y διάσταση, από το F(i,j,k+1) και F(i,j,k-1) για την z διάσταση. Έτσι το μέγεθος των sections αλλάζει από διάσταση σε διάσταση αναλόγως τον τρόπο με τον οποίο έχουμε αποθηκεύσει τα δεδομένα στις μνήμες.

Για την υλοποίηση της τεχνικής αυτής, χρησιμοποιήθηκε ένας memory controller ο οποίος είναι υπεύθυνος για την διευθυνσιοδότηση και την αλλαγή καταστάσεων των 3 μικρότερων μνημών ταυτόχρονα, σύμφωνα με τον αλγόριθμο που δείξαμε παραπάνω. Επίσης, χρειάστηκε και ένας memory handler ο οποίος είναι υπεύθυνος για τον διαμοιρασμό των εξόδων των μνημών προς τις υπόλοιπες μονάδες. Λόγω του αλγορίθμου, τα στοιχεία για κάθε πίνακα δεν είναι προκαθορισμένο από ποια μνήμη τα παίρνουμε(*έχουμε πλέον 3 διαφορετικές μνήμες για κάθε πίνακα*).

Ακόμα, αξίζει να αναφερθεί το γεγονός ότι με την ταυτόχρονη διάσχιση των 3 μικρότερων μνημών σε σχέση με την ενιαία μεγαλύτερη, πετυχαίνουμε υψηλότερους ρυθμούς ρολογιού.



Σχήμα 4.20: Χρήση μιας ενιαίας μνήμης με τον αντίστοιχο memory controller για την διευθυνσιοδότηση της.



Σχήμα 4.21: Χρήση 3 μικρότρων παράλληλων μνημών με τον αντίστοιχο memory controller για την διευθυνσιοδότηση τους και τον memory handler για τον διαμοιρασμό των δεδομένων τους.

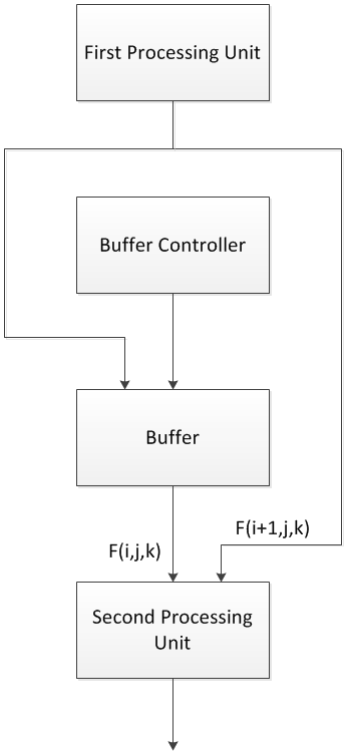
4.2.2 Χρησιμότητα των Buffers

Κατά την διάρκεια εκτέλεσης του μοντέλου παρουσιάστηκε η επιτακτική ανάγκη κατασκευής κάποιων ενδιάμεσων buffers. Αυτό έγινε για τον λόγο ότι σε κάποια αποτελέσματα μιας μονάδας επεξεργασίας όπου οι έξοδοι της οδηγούνται σε κάποια άλλη μονάδα επεξεργασίας, υπήρχε η ανάγκη για την ταυτόχρονη είσοδο 2 εξόδων της. Δηλαδή, έστω ότι μια μονάδα επεξεργασίας βγάζει κάθε φορά το αποτέλεσμα X(i,j,k) και η αμέσως επόμενη βαθμίδα θέλει

ως εισόδους της τις τιμές X(i,j,k) και X(i+1,j,k). Αναγκαστικά τότε επειδή δεν μπορούμε να της δώσουμε ταυτόχρονα αυτές τις 2 τιμές, χρησιμοποιούμε μία FIFO(*buffer*) της οποίας η δομή θα αναλυθεί παρακάτω.

Οι buffers υλοποιήθηκαν με την εξής αρχή: Αποτελούν μνήμες τύπου dual port Ram, write first after read. Σε κάθε κύκλο η μονάδα που βγάζει το αποτέλεσμα (*έστω F(i+1,j,k)*) το περνάει κατευθείαν στην επόμενη μονάδα επεξεργασίας, ενώ ταυτόχρονα το γράφει και στον buffer από το ένα port του, έστω στη θέση n. Την προηγούμενη τιμή F(i,j,k) την παίρνουμε από το άλλο port του buffer από την θέση n+1, στην οποία είχε αποθηκευτεί από κάποιο προηγούμενο κύκλο(*συγκεκριμένα πριν από τόσους κύκλους όσο είναι η απόσταση των στοιχείων F(i,j,k) και F(i+1,j,k)*). Αυτό που γίνεται δηλαδή, είναι κάθε φορά να αποθηκεύουμε την τιμή F(i+1,j,k) του buffer, την οποία θα την χρησιμοποιήσουμε σε κάποιο επόμενο κύκλο ως την τιμή F(i,j,k). Με αυτο τον τρόπο μπορούμε να πάρουμε ταυτόχρονα και τις 2 τιμές, την F(i+1,j,k) κατευθείαν από την έξοδο της μονάδας επεξεργασίας και την F(i,j,k) από τον buffer. Στον ίδιο κύκλο γίνεται και το διάβασμα της παλιάς τιμής του αποτελέσματος (*F(i,j,k)*)) και η εγγραφή του καινούργιου (*F(i+1,j,k)*)), το οποίο θα χρησιμοποιηθεί σε κάποιο επόμενο κύκλο(*τόσους κύκλους όσο η απόσταση των στοιχείων F(i,j,k) και F(i+1,j,k)*).

Αυτή η ανάγκη εμφανίζεται κατά την επεξεργασία και των 3 διαστάσεων. Η τεχνική που χρησιμοποιούμε είναι η ίδια, αλλά αλλάζει το μέγεθος του buffer (*FIFO*) κάθε φορά.



Σχήμα 4.22: Δομή χρήσης buffer για την ταυτόχρονη είσοδο 2 αποτελεσμάτων από την πρώτη μονάδα επεξεργασίας στην δεύτερη

Οι τρισδιάστατοι πίνακες είναι αποθηκευμένοι στις μνήμες μας με τη μορφή :

for i=1 to L1
    for j=1 to L2

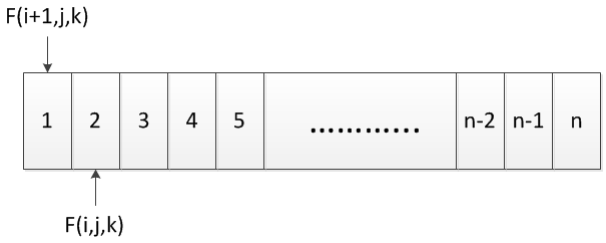
for k=1 to L3

Αυτό σημαίνει πως αν έχουμε τον τρισδιάστατο πίνακα F, η προσπέλαση της θέσης του F(i+1,j,k) απαιτεί μετάβαση στη μνήμη μας κατά L2*L3 θέσεις κάθε φορά. Η προσπέλαση της F(i,j+1,k) απαιτεί μετάβαση στη μνήμη κατά L2 θέσεις, ενώ αντίστοιχα η προσπέλαση της F(i,j,k+1) απαιτεί μετάβαση στη μνήμη κατά 1 μόνο θέση. Εκμεταλευόμενοι αυτή τη σημαντική πληροφορία, μπορούμε να εξοικονομήσουμε πόρους φτιάχνοντας buffers τόσο μεγάλους όσο είναι το μέγεθος της μετάβασης για κάθε διάσταση και όχι όσο είναι το μέγεθος ολόκληρου του πίνακα.

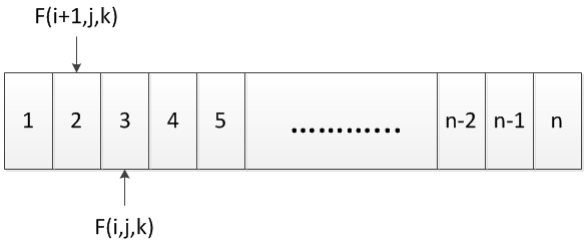
Έτσι για την x διάσταση έγινε χρήση ενός L2*L3 μεγέθους buffer, για την y διάσταση έγινε χρήση ενός L2 μεγέθους buffer, ενώ για την z διάσταση επειδή η μετάβαση απαιτεί 1 μόνο θέση απλά προσθέσαμε εναν καταχωρητή, ο οποίος κρατάει την προηγούμενη τιμή του αποτελέσματος κάθε φορά χωρίς να χρησιμοποιήσουμε κανέναν buffer.

Έστω ότι ο buffer για την x διάσταση έχει L2*L3 θέσεις, μόλις φτάσουμε στην θέση L2*L3 και διαβάσουμε την παλιά τιμή, η επόμενη θέση που θα διαβαστεί η επόμενη παλιά τιμή θα είναι η θέση 0. Αυτό που κάνουμε δηλαδή, είναι κύκλους στη μνήμη προκειμένου να πετάξουμε τις παλιές τιμές που δεν θα ξαναχρησιμοποιηθούν και να εγγράψουμε τις νέες (*σχήμα 4.23, 4.24, 4.25*). Με αυτό τον τρόπο εξοικονομούμε πολλλούς πόρους μνήμης.

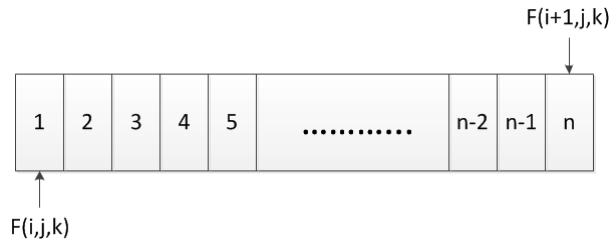
Επειδή αυτές οι διαδικασίες έχουν μια μκρή πολυπλοκότητα, πάνω από κάθε buffer έχουμε έναν controller ο οποίος είναι υπεύθυνος για τον έλεγχο της σωστής λειτουργίας του. Ο controller διευθυνδιοδοτεί και τα 2 ports του buffer (*σχήμα 4.22*).



Σχήμα 4.23: Παράδειγμα λειτουργίας buffer



Σχήμα 4.24: Λειτουργία buffer για τον επόμενο κύκλο λειτουργίας του παραπάνω σχήματος



Σχήμα 4.25: Λειτουργία buffer ύστερα από αρκετούς κύκλους λειτουργίας των παραπάνω σχημάτων

4.3 Σχεδίαση Μονάδων Ελέγχου και Συγχρονισμός

Σε αυτή την ενότητα έγινε ο συνδιασμός των 2 προηγούμενων ενοτήτων με κατάλληλες μονάδες ελέγχου και ο συγχρονισμός όλου του συστήματος.

4.3.1 Τεχνικά Συμπεράσματα Μελέτης Μοντέλου

Για να αρχίσουμε την υλοποίηση αυτού του σταδίου, έπρεπε πρώτα να έχουμε μια συνολική εποπτεία όλου του προγράμματος από την αρχή μέχρι το τέλος. Έπρεπε να ξεφύγουμε από τον σειριακό μοτίβο του προβλήματος και να το αποτυπώσουμε σε μορφή κυκλώματος, κάνοντας όποιες δυνατές βελτιστοποιήσεις μπορούσαμε. Πολλά από τα δεδομένα εισόδου χρησιμοποιούνταν από την αρχή μέχρι το τέλος της συνάρτησης σε διάφορα κομμάτια της, όπως παρομοίως γινόταν και με τις εξόδους κάποιων μονάδων επεξεργασίας. Έτσι η τοποθέτηση των μονάδων επεξεργασίας και των μονάδων ελέγχου, έπρεπε να γίνει με τέτοιο τρόπο ώστε η αρχιτεκτονική μας να έχει μια συνεχόμενη ροή εκτέλεσης χωρίς περιττούς ενδιάμεσους buffers και dublication διαφόρων μοναδών.

Με μια ποιοτική μελέτη του προγράμματος, αρχικά είχαμε εντοπίσει κάποια σημεία τα οποία ήταν περιττά κατά την εκτέλεση του, και το μόνο που μας πρόσθεταν ήταν κύκλοι καθυστέρησης. Αυτά τα σημεία θέλαμε να τα αποφύγουμε και με κατάλληλο έλεγχο να αποτυπώσουμε μια πιο "έξυπνη" και γρήγορη αρχιτεκτονική. Ένα ενδεικτικό παράδειγμα, είναι η μη προσπέλαση όλων των στοιχείων των πινάκων (*πολλές φορές τα άκρα δεν συνυπολογίζονται στην διαδικασία επεξεργασίας*), κάτι το οποίο το εκμεταλευτήκαμε και μπορέσαμε να εξοικονομήσουμε εκατοντάδες κύκλους περιττής επεξεργασίας.

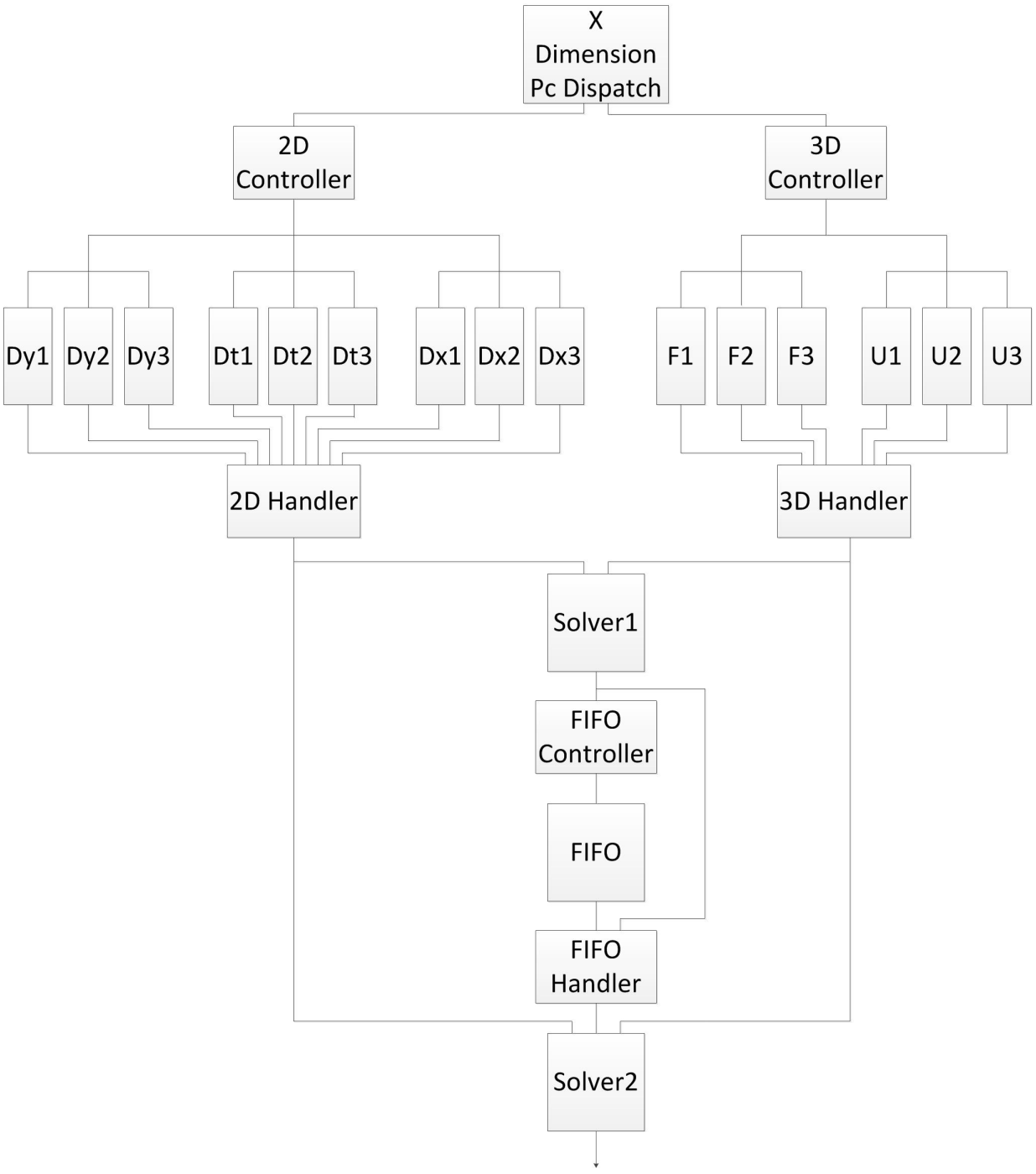
Άλλη μια μέθοδος η οποία προήλθε από την μελέτη του προγράμματος, ήταν η συγχώνευση πολλών διαδικασιών επεξεργασίας σε μία. Σε πολλά σημεία της συνάρτησης παρατηρήθηκε η σπατάλη πολλών κύκλων για επεξεργασία κάποιων δεδομένων και ύστερα να ακολουθεί μια άλλη επεξεργασια στην οποία λάμβαναν μέρος οι έξοδοι της προηγούμενης. Πολλά τέτοια σημεία του κώδικα τα ενοποιήσαμε σε μία μόνο επεξεργασία η οποία έκανε την ταυτόχρονη υλοποίηση των 2 προηγούμενων ξεχωριστών διαδικασιών.

Με τέτοιου είδους παρατηρήσεις, μπορέσαμε να αποτυπώσουμε μια λειτουργική αρχιτεκτονική, εξοικονομώντας όσο το δυνατόν περισσότερους πόρους και αποφεύγοντας άσκοπους επεξεργαστικούς κύκλους.

4.3.2 Ανάλυση των Διαστάσεων

Αφού είχαμε σχεδιάσει το πλάνο της υλοποίησης προχωρήσαμε στο στάδιο της αποτύπωσης του, φτιάχνωντας τις μονάδες ελέγχου για τη σωστή ροή των δεδομένων και το συγχρονισμό τους. Αρχικό στάδιο της υλοποίησης ήταν τα ανεξάρτητα datapath για την επεξεργασία της κάθε διάστασης.

Για την x διάσταση :



Σχήμα 4.26: Αρχιτεκτονική υλοποίησης για την x διάσταση.

Στο συγκεκριμένο datapath αρχίζουμε με μια μονάδα την Pc_Dispatch, η οποία είναι υπεύθυνη για τη σωστή ροή του προγράμματος. Αυτό που κάνει είναι να γνωστοποιεί στις υπόλοιπες μονάδες σε ποιο σημείο της υλοποίησης βρισκόμαστε κάθε φορά. Με βάση αυτή την πληροφορία γίνονται όλοι οι απαραίτητοι έλεγχοι συγχρονισμού και οι προσπέλασεις των μνημών.

Ύστερα ακολουθούν οι μονάδες ελέγχου διευθυνσιοδότησης των μνημών. Έχουμε 2

τέτοιες μονάδες ελέγχου, μία για την επίβλεψη των 3 διαστάσεων πινάκων και μία για των 2 διαστάσεων. Οι δύο αυτές μονάδες είναι διαφορετικές μεταξύ τους καθώς οι διευθυνσιοδοτήσεις γίνονται με διαφορετικό τρόπο. Κατά την εκτέλεση του προγράμματος οι τιμές των τρισδιάστατων μνημών χρησιμοποιούνται μία φορά, ενώ των δισδιάστατων η κάθε τιμή χρησιμοποιείται περισσότερες φορες. Έτσι μέσα στον έλεγχο έχουν ενσωματωθεί και οι απαραίτητοι μετρητές, με βάση τους οποίους γίνεται η προσπέλαση των επόμενων θέσεων.

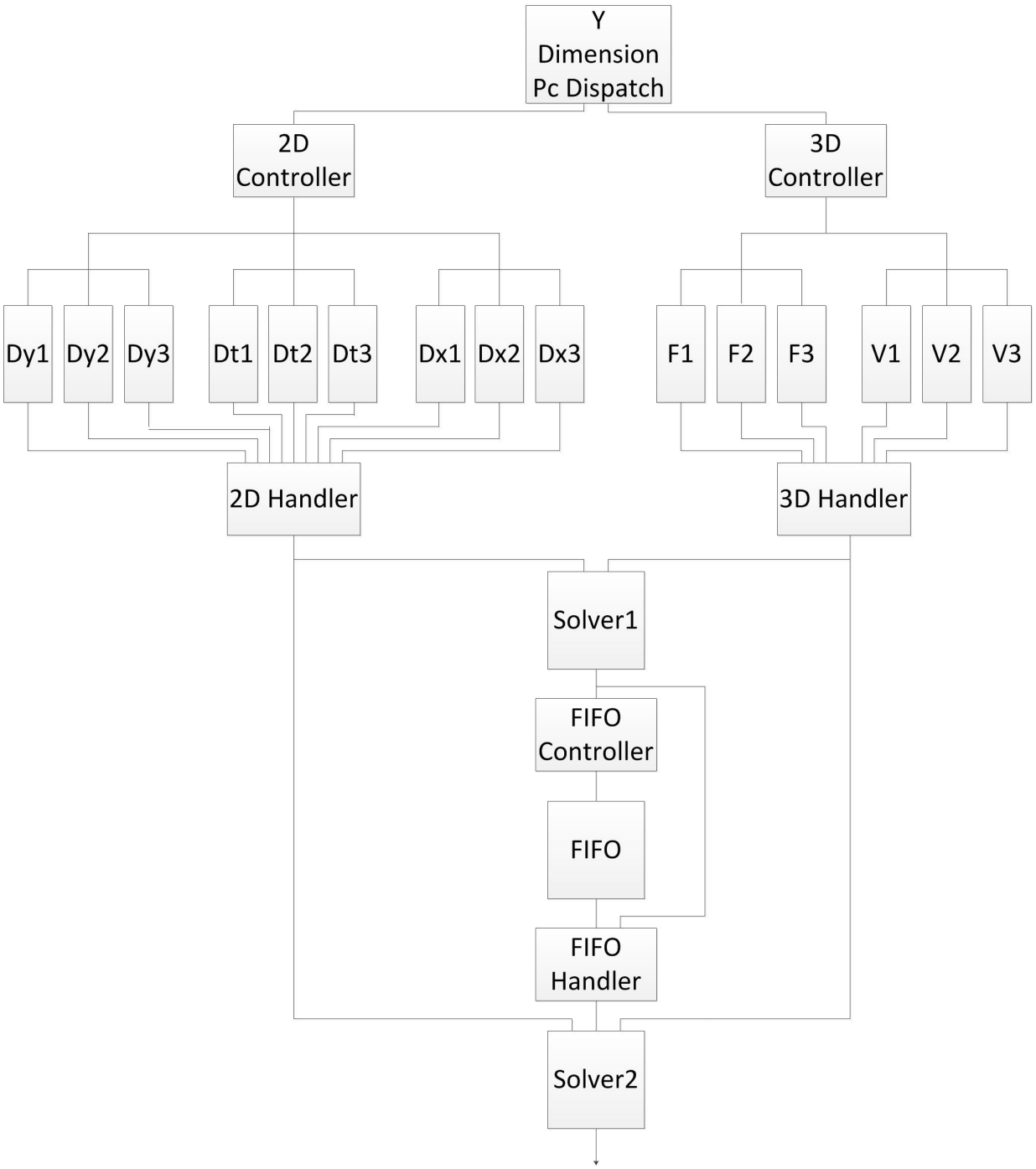
Επόμενο στάδιο της ιεραρχίας είναι οι μνήμες, οι οποίες είναι χωρισμένες σε 3 κομμάτια η κάθε μία, για τους λόγους που αναφέραμε στην προηγούμενη φάση υλοποίησης.

Ύστερα, κάτω από τις μνήμες βρίσκονατι οι αντίστοιχοι memory handlers για τις τρισδιάστατες και τις δισδιάστατες μνήμες. Αυτοί είναι υπεύθυνοι για τον διαμοιρασμό των εξόδων από τις μνήμες στις υπόλοιπες μονάδες επεξεργασίας.

Έπειτα ακολουθεί μια μονάδα επεξεργασίας, η οποία παίρνει τις εισόδους της από τους memory handlers. Ακολουθεί μία διάταξη buffer με τον τρόπο που έχουμε αναλύσει στην προηγούμενη φάση (*με τις αντίστοιχες μονάδες ελέγχου και διαχείρησης*), η οποία είναι υπεύθυνη για τον διαμοιρασμό της τωρινής εξόδου της 1ης μονάδας επεξεργασίας και την αποθήκευση της, αλλά και της προήγουμενης τιμής εξόδου(*από τον προηγούμενο κύκλο*).Τις τιμές αυτές, τις περνάει ως εισόδους στην 2η μονάδα επεξεργασίας.

Τέλος, ακολουθεί η 2η μονάδα επεξεργασίας που παίρνει τις εισόδους της και από τους memory handlers και από την προηγούμενη διάταξη του buffer.

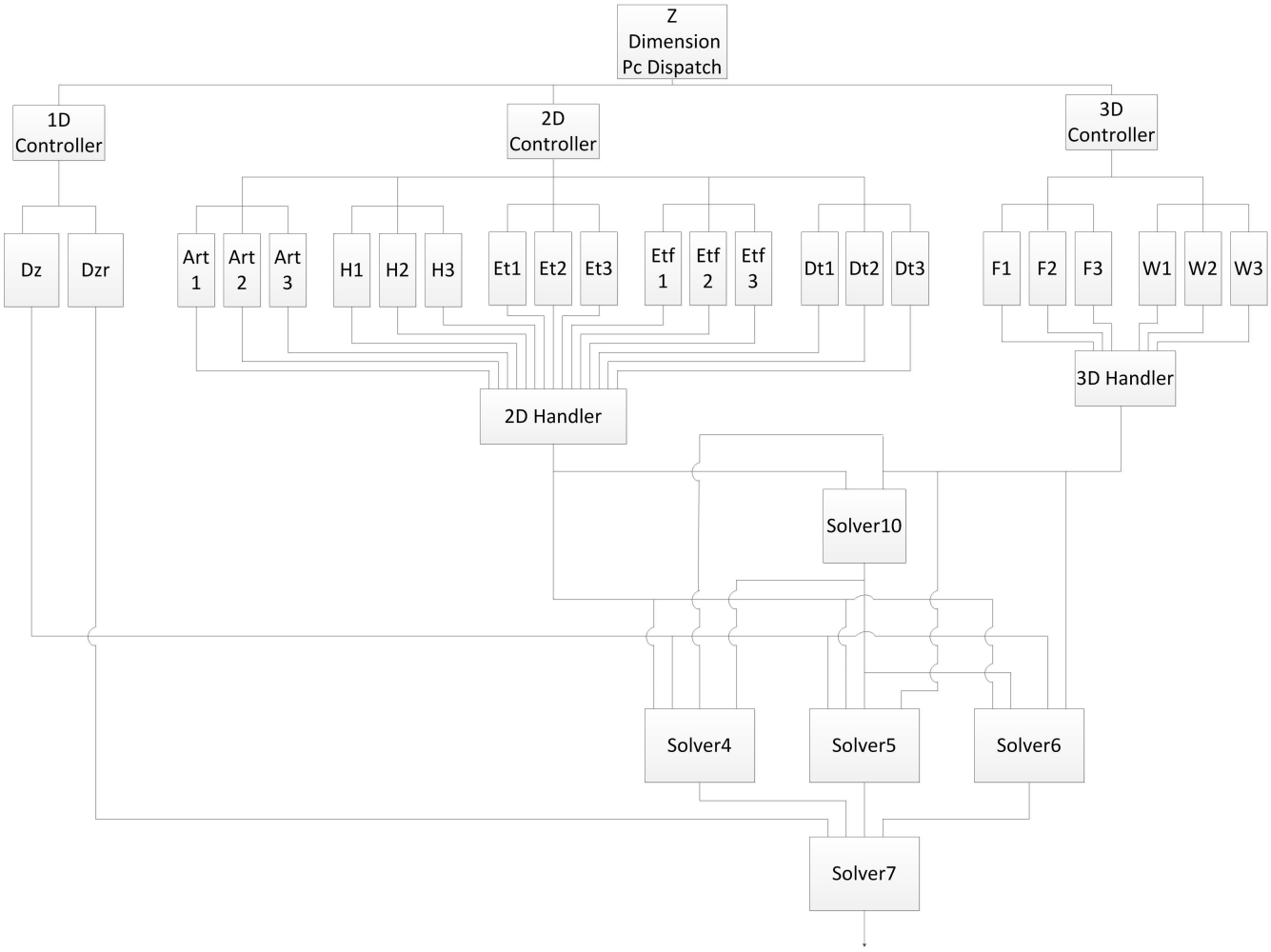
Για την y διάσταση :



Σχήμα 4.27: Αρχιτεκτονική υλοποίησης για την y διάσταση.

Το συγκεκριμένο datapath είναι ίδιο με αυτο της x διάστασης και στηρίζεται στις ίδιες αρχές. Οι διαφορές που έχουν, βρίσκονται σε όλες τις μονάδες ελέγχου και διαχείρησης των μνημών. Στη y διάσταση είναι διαφορετικά αποθηκευμένα τα δεδομένα στις μνήμες σε σχέση με την διάσταση. Μπορεί συνολικά να έχουμε το ίδιο μέγεθος δεδομένων με τις υπόλοιπες διαστάσεις, αλλά οι μνήμες είναι χωρισμένες διαφορετικά, και άρα το μέγεθος της κάθε μίας διαφέρει με το αντίστοιχο των άλλων διαστάσεων. Το ίδιο συμβαίνει και με την διάταξη του buffer.

Για την z διάσταση :



Σχήμα 4.28: Αρχιτεκτονική υλοποίησης για την z διάσταση.

Το datapath αυτής της διάστασης στηρίζεται στις ίδιες αρχές με τις προηγούμενες διαστάσεις με κάποιες σημαντικές διαφορές. Οι ιδιαιτερότητες που παρουσιάζει είναι :

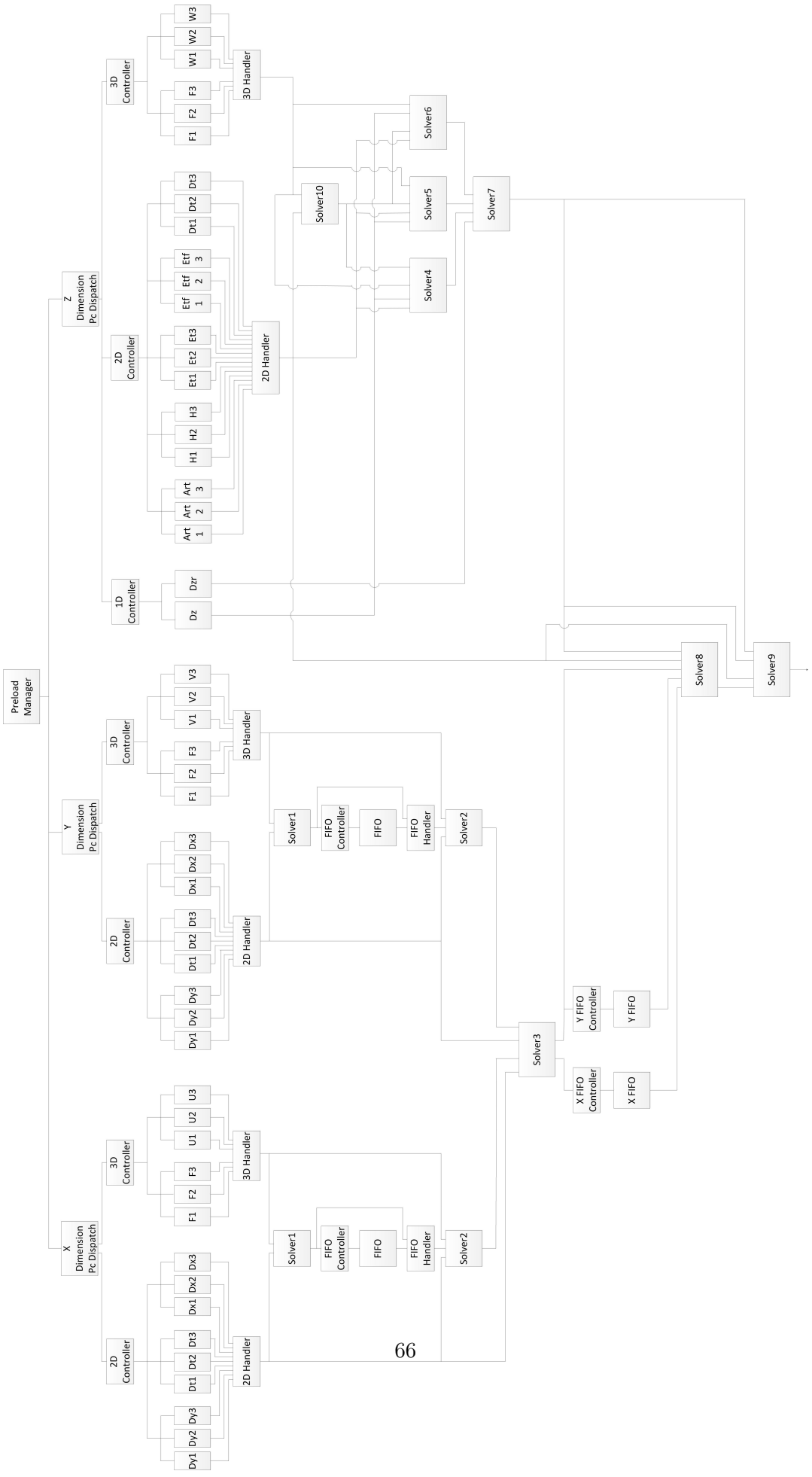
- Εδώ χρησιμοποιούμε και μονοδιάστατους πίνακες. Για αυτό το λόγο έχουμε και μια επιπλέον μονάδα ελέγχου υπεύθυνη για την διευθυνσιοδότηση τους. Αυτοί οι πίνακες είναι μικροί και δεν είναι χωρισμένοι σε μικρότερους, γιατί δεν υπάρχει το πρόβλημα ταυτόχρονης προσπέλασης 3 στοιχείων τους.
- Έχουμε χρήση περισσότερων δισδιάστατων πινάκων με αποτέλεσμα περισσότερη πολυπλοκότητα στον έλεγχο και διαμοιρασμό των δεδομένων.
- Έχουμε περισσότερες μονάδες επεξεργασίας που επικοινωνούν η μία με την άλλη.
- Δεν χρησιμοποιούμε κάποιον buffer καθώς διαδοχικές τιμές δεδομένων έχουν απόσταση μίας μόνο θέσης μεταξύ τους σε αυτή τη διάσταση. Χρησιμοποιούμε έναν καταχωρητή που κρατάει την προηγούμενη τιμή του αποτελέσματος όπου χρειάζεται.

- Οι μονάδες ελέγχου και διαχείρησης των μνημών διαφέρουν από τις υπόλοιπες διαστάσεις και αυτό οφείλεται πάλι στον τρόπο με τον οποίο είναι αποθηκευμένα τα δεδομένα στις μνήμες.

4.3.3 Ολοκλήρωση και Λειτουργία Αρχιτεκτονικής

Έπειτα από την υλοποίηση και έλεγχο των datapaths για κάθε διάσταση, προχωρήσαμε στην ολοκλήρωση της συνολικής αρχιτεκτονικής ενοποιώντας όλα τα προηγούμενα datapaths.

Το συνολικό datapath της αρχιτεκτονικής παρουσιάζεται παρακάτω :


Preload Manager
X Dimension Pc Dispatch
Y Dimension Pc Dispatch
Z Dimension Pc Dispatch
3D Controller
2D Controller
1D Controller
3D Handler
2D Handler
F1
F2
F3
U1
U2
U3
V1
V2
V3
W1
W2
W3
Dx1
Dx2
Dx3
Dy1
Dy2
Dy3
Dz
Dzr
Dt1
Dt2
Dt3
Et1
Et2
Et3
Etf 1
Etf 2
Etf 3
Art 1
Art 2
Art 3
H1
H2
H3
Solver1
Solver2
Solver3
Solver4
Solver5
Solver6
Solver7
Solver8
Solver9
Solver10
FIFO Controller
FIFO
FIFO Handler
X FIFO Controller
X FIFO
Y FIFO Controller
Y FIFO

Μια πιο αφηρημένη εικόνα του datapath παρουσιάζεται στο σχήμα 4.30. Αρχίζει με μια μονάδα ελέγχου την Preload Manager, που είναι υπεύθυνη για το συνολικό έλεγχο, την έναρξη λειτουργίας και το συγχρονισμό της αρχιτεκτονικής. Η δομή της αποτελείται από 3 μονάδες ελέγχου που σχετίζονται με τον έλεγχο όλων των μνημών στο στάδιο προφόρτωσης των δεδομένων. Η μία από αυτές χρησιμοποιείται για τη διαχείρηση της διαδικασίας φόρτωσης των δεδομένων σε όλους τους τρισδιάστατους πίνακες, και αντίστοιχα οι άλλες 2 είναι υπεύθυνες για την διαχείρηση των δισδιάστατων και μονοδιάστατων πινάκων. Αυτές οι μονάδες δεν έχουν καμία σχέση με τις μονάδες ελέγχου προσπέλασης των μνημών. Η χρησιμότητα τους είναι μόνο η εγγραφή των μνημών στο αρχικό στάδιο και όχι η μετέπειτα ανάγνωση τους.

Έπειτα, ακολουθούν οι διατάξεις επεξεργασίας των διαφορετικών διαστάσεων που αναλύθηκαν παραπάνω. Τέλος, γίνεται ο συνδιασμός των εξόδων όλων των διαστάσεων μέσω των κατάλληλων μονάδων επεξεργασίας όπως φένεται και στο σχήμα 4.29. Για τον σωστό συγχρονισμό όλων των διατάξεων που αποτελούν την αρχιτεκτονική, χρησιμοποιήθηκαν οι απαραίτητοι καταχωρητές προκειμένου να επικαλυφθούν τυχόν καθυστερήσεις.

Λειτουργία Αρχιτεκτονικής

Η λειτουργία της αρχιτεκρονικής χωρίζεται σε δύο στάδια. Το στάδιο προφόρτωσης των δεδομένων εισόδου(*όλους τους πίνακες*) και το στάδιο επεξεργασίας των εισόδων.

Το στάδιο της προφόρτωσης των δεδομένων εξαρτάται αποκλειστικά από τη μονάδα Preload Manager. Οπως αναφέρθηκε παραπάνω, η μονάδα αυτή αποτελείται από 3 μικρότερες μονάδες ελέγχου, μία για τους τρισδιάστατους, μία για τους δισδιάστατους και μία για τους μονδιάστατους πίνακες. Κάθε φορά που έρχεται ένα έγκυρο δεδομένο από τη DRAM, αναλαμβάνουν οι μονάδες αυτές να το προφορτώσουν στην αντίστοιχη μνήμη/μνήμες χρησιμοποιώντας κατάλληλα σήματα ελέγχου. Ταυτόχρονα μπορούμε να έχουμε πολλά δεδομένα εισόδων, μέχρι 14 όσο είναι και το I/O της συνάρτησης. Σε τέτοιες περιπτώσεις γίνεται η ταυτόχρονη προφόρτωση όλων των δεδομένων ενεργοποιώντας περισσότερα σήματα ελέγχου. Σε περίπτωση που δεν έχουμε κάποιο δεδομένο ή αργήσει να έρθει, σταματάει αυτόματα η διαδικασία προφόρτωσης μέχρι να υπάρξει κάποια νέα είσοδος. Η διαδικασία της προφόρτωσης δεν γίνεται συγχρονισμένα για όλες τις εισόδους, κάθε είσοδος δρα ανεξάρτητα από τις υπόλοιπες. Με αυτό τον τρόπο έχουμε μια γρήγορη διαδικασία μεταφοράς δεδομένων από την Dram στην αρχιτεκτονική μας. Οι μέθοδοι μεταφοράς δεδομένων που δοκιμάστηκαν θα αναλυθούν με περισσότερες λεπτομέρεις στο κεφάλαιο 6. Όταν προφορτωθούν όλα τα δεδομένα στις μνήμες τότε η Main Control δίνει το σήμα έναρξης της επεξεργασίας των δεδομένων.

Το στάδιο της επεξεργασίας των δεδομένων είναι πιο straightforward από τη στιγμή που φορτώσαμε τις μνήμες. Έχοντας το πλεονέκτημα των προφορτωμένων εισόδων μπορούμε να έχουμε 100% utilization της pipeline σχεδίασης. Δεν υπάρχει καμία καθυστέρηση μεταφοράς και συγχρονισμού δεδομένων, και σε κάθε κύκλο μπορούμε να επεξεργαζόμαστε νέα δεδομένα. Σε συνδιασμό με όλες τις τεχνικές κατασκευής των μνημών που παρουσιάστηκαν στην ενότητα 4.2, υπήρχε η δυνατότητα ταυτόχρονης προσπέλασης 3 στοιχείων από τον κάθε πίνακα. Αυτό βοήθησε πολύ στο έργο μας να πετύχουμε ένα ικανοποιητικό throughput 1 αποτέλεσμα ανά κύκλο.

4.4 Χαρακτηριστικά Αρχιτεκτονικής

Σε αυτή την ενότητα παρουσιάζονται συνοπτικά όλα τα χαρακτηριστικά της αρχιτεκτονικής:

· Είναι πλήρως pipeline.

· Τρέχει με συχνότητα 150 Mhz.

· Το throughput είναι 1 αποτέλεσμα ανά κύκλο.

· Η αριθμητική που χρησιμοποιεί είναι 32-bit floating point.

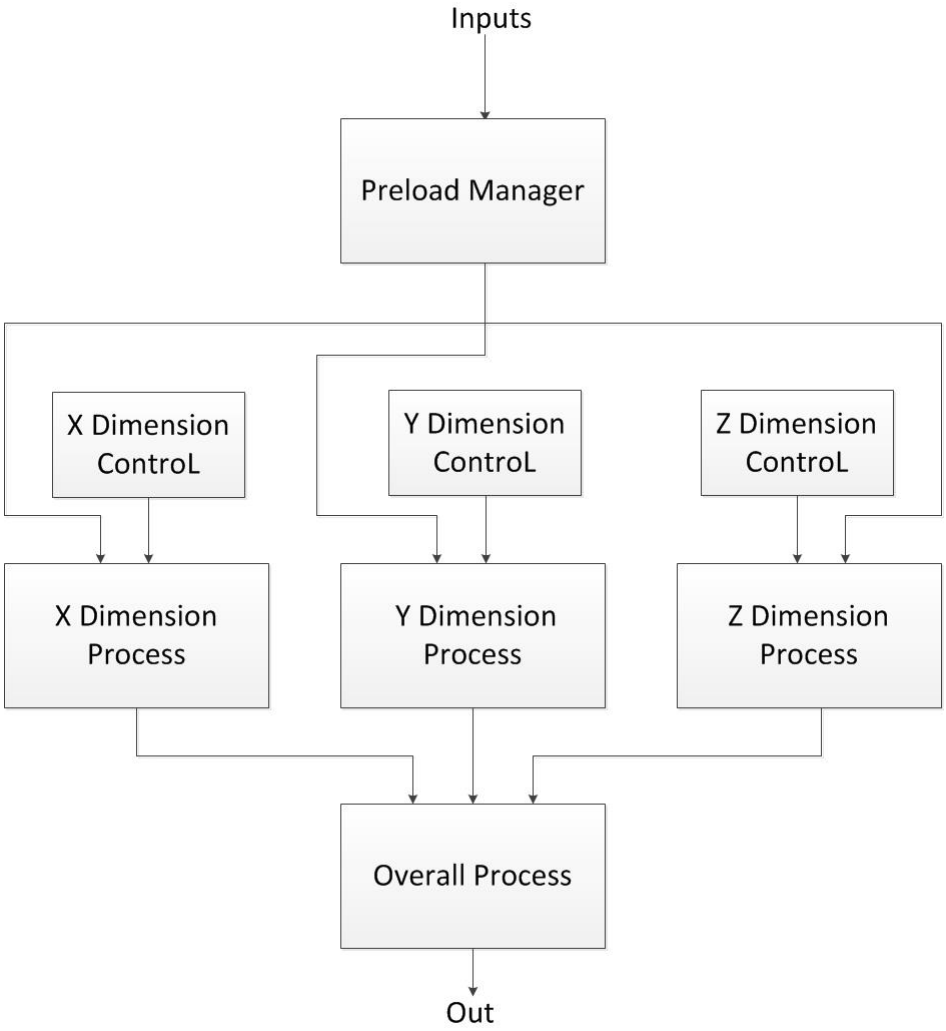
· Οι συνολικές αριθμητικές μονάδες για την επίλυση των εξισώσεων είναι τύπου floating. Συγκεκριμένα περιέχει 27 αθροιστές, 42 πολλαπλασιαστές, 6 διαιρέτες και 22 συγκριτές.

· Η λειτουργία της χωρίζεται σε 2 στάδια. Το στάδιο της προφόρτωσης των δεδομένων και το στάδιο της επεξεργασίας.

· Οι κατανάλωση των συνολικών πόρων παρουσιάζεται στον πίνακα 4.2.

Resources of Virtex 5 VLX330	% Used
Number of Slice Registers	21
Number of Slice LUTs	23
Number of Block RAM/FIFO	60
Number of DSP48Es	48

Πίνακας 4.2: Συνολικοί πόροι αρχιτεκτονικής



Σχήμα 4.30: Μια πιο αφηρημένη απεικόνιση της αρχιτεκτονικής.

Κεφάλαιο 5

Έλεγχος και Επικύρωση Αρχιτεκτονικής

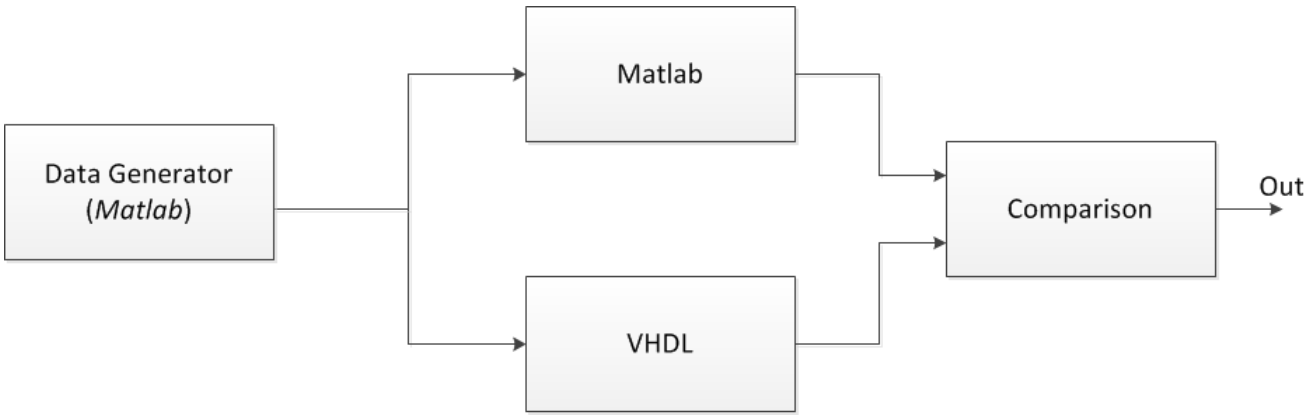
Αυτό το κεφάλαιο αναφέρεται στον έλεγχο της λειτουργίας της αρχιτεκτονικής και την επικύρωση της. Παρουσιάζεται ο έλεγχος κάθε τμήματος της αρχιτεκτονικής ξεχωριστά και έπειτα ο έλεγχος της συνολικής αρχιτεκτονικής για διάφορα σετ δεδομένων. Τέλος, ακολουθεί η επαλήθευση των αποτελεσμάτων του συνολικού μοντέλου.

5.1 Διαδικασία Ελέγχου

Για κάθε σχεδίαση που υλοποιούμε στο hardware, εκτιμούμε την ορθότητα των εξόδων της συγκρίνοντας τες με τις εξόδους αντίστοιχων προσομοιώσεων σε software. Όπως αναφέρουμε στο κεφάλαιο 2 έχουμε φτιάξει μια προσομοίωση σε Matlab για τη συνάρτηση ενδιαφέροντος και έχουμε ελέγξει τη εγκυρότητα της. Με βάση αυτή την προσομοίωση, μπορούμε να ελέγξουμε τη σχεδίαση μας στο hardware συγκρίνοντας τα αποτελέσματα της Matlab με αυτά της VHDL.

Τα αποτελέσματα από την προσομοίωση της VHDL εγγράφονται σε ένα αρχείο. Έχουμε δημιουργήσει ένα σύστημα, σύμφωνα με το οποίο για κάθε κύκλο που παίρνουμε κάποιο αποτέλεσμα από τη σχεσίαση, εγγράφεται αυτόματα σε ένα συγκεκριμένο αρχείο. Το ίδιο γίνεται με την προσομοίωση στη Matlab, γράφοντας τα αποτελέσμτα του προγράμματος(*ισοδύναμου με τη σχεδίαση στο hardware*) σε ένα άλλο αρχείο. Με αυτό τον τρόπο κάθε φορά που θέλουμε να ελέγξουμε μια σχεδίαση στο hardware, έχουμε 2 αρχεία, ένα με τα αποτελέσματα της VHDL και ένα με τα αποτελέσματα της Matlab. Μέσω ενός script που παίρνει ως εισόδους αυτά τα 2 αρχεία συγκρίνουμε τα αποτελέσματα από τις 2 προσομοιώσεις και ελέγχουμε την ταύτιση τους.

Η διαδικασία που περιγράψαμε παραπάνω φένεται συνοπτικά στο σχήμα 5.1 όπου δατα είναι τα δεδομένα εισόδου που είναι κοινά και για τις 2 προσομοιώσεις, και εαλυατιον η μονάδα σύγκρισης των εξόδων των 2 προσομοιώσεων. Η έξοδος που παίρνουμε (*ουτ*) είναι ένα αρχείο με τα αποτελέσματα της σύγκρισης.



Σχήμα 5.1: Δομή συστήματος ελέγχου για κάθε σχεδίαση στο hardware.

Οι είσοδοι που παίρνουν τα προγράμματα της Matlab και της VHDL είναι οι ίδιοι. Απλά οι είσοδοι της Matlab είναι σε δεκαδική μορφή, με αποτέλεσμα να πρέπει να τους μετατρέπουμε σε δυαδική μορφή, προκειμένου να τους χρησιμοποιήσουμε και στην VHDL. Το format της μετατροπής αντιστοιχεί στην αριθμητική την οποία έχουμε επιλέξει για την κατασκευή της αρχιτεκτονική μας, 32-bit single precision της IEEE 754. Στο τέλος της επεξεργασίας προκειμένου να συγκρίνουμε τις εξόδους της VHDL και της Matlab, πρέπει πάλι να αποκωδικοποιήσουμε τις εξόδους της VHDL από δυαδική σε δεκαδική αναπαράσταση, για να μπορέσουμε να τις συγκρίνουμε με την Matlab. Τέλος, πρέπει να αναφερθεί ότι όλες οι πράξεις που γίνονται στη Matlab είναι τύπου 32-bit single precision όπως και στο hardware , ωστέ να έχουμε εκριβείς συγκρίσεις.

Η εγγραφή των αποτελεσμάτων από το hardware σε αρχείο είναι μια πολύ εύχρηστη και γρήγορη μέθοδος αποσφαλμάτωσης. Μπορούμε να δώσουμε πολλές χιλιάδες εισόδους και ο έλεγχος των αποτελεσμάτων να γίνει αυτόματα μέσω του script που περιγράφτηκε στην προηγούμενη παράγραφο. Με αυτή τη διαδικασία έχουμε τη δυνατότητα εφαρμογής ενός αυστηρού ελέγχου και την παρατήρηση της συμπεριφοράς της σχεδίασης για μεγάλο αριθμό διαφορετικών εισόδων.

5.2 Έλεγχος Μονάδων Επεξεργασίας

Όπως είδαμε στο κεφάλαιο 4, μια μονάδα επεξεργασίας αποτελείται από πολλές αριθμητικές μονάδες(*αθροιστές, πολλαπλασιαστές, διαιρέτες, συγκριτές*). Για τον έλεγχο κάθε μονάδας επεξεργασίας, χρησιμοποιούμε την bottom up μέθοδο. Σύμφωνα με τη αυτή τη μέθοδο, για κάθε τμήμα που κατασκευάζουμε και αποτελεί κομμάτι της δομής μιας μονάδας επεξεργασίας, γίνεται ο έλεγχος της λειτουργίας και ακρίβειας των αποτελεσμάτων του. Η διαδικασία αυτή γίνεται για κάθε μονάδα που προστίθεται στην αρχιτεκτονική της μονάδας επεξεργασίας. Μόλις ολοκληρωθεί μια μονάδα επεξεργασίας τότε γίνεται ο έλεγχος ως προς το σύνολο της. Με αυτό τον τρόπο έγινε η κατασκευή όλων των μονάδων επεξεργασίας.

Όταν τελειώσουμε με την κατασκευή όλων των μονάδων επεξεργασίας, τότε συνεχίζουμε με την εξέταση μεγαλύτερων τμημάτων της συνάρτησης που αποτελούν και συνδιάζουν αυτές τις μονάδες επεξεργασίας. Για κάθε μονάδα επεξεργασίας που προσθέτουμε και μεγαλώνει η αρχιτεκτονική γίνεται ταυτόχρονα ο αυστηρός έλεγχος της. Η διαδικασία συνεχίζεται μέχρι να ενσωματωθούν στον έλεγχο όλες οι μονάδες επεξεργασίας και να ολοκληρωθεί η τελική μορφή της συνάρτησης.

Τον έλεγχο της συνάρτησης τον χτίσαμε σιγά σιγά από κάτω προς τα πάνω, ξεκινώντας από τα πιο μικρά κομμάτια, φτιάχνοντας στη συνέχεια άλλα μεγαλύτερα, με τη σειρά τους και αυτά τα χρησιμοποιήσαμε για την κατασκευή ακόμα πιο μεγάλων και συνεχίσαμε μέχρι να ολοκληρωθεί η συνολική δομή της συνάρτησης. Για τον έλεγχο της κάθε σχεδίασης από την μικρότερη μέχρι τη μεγαλύτερη, χρησιμοποιούμε πολλές χιλιάδες εισόδους για να εγγυηθούμε την ορθή λειτουργίας της. Επίσης, όπως εξηγήσαμε και στην ενότητα 5.1 για την αξιολόγηση του κάθε κομματιού έχουμε και τον αντίστοιχο κώδικα σε Matlab. Κάθε φορά συγκρίνουμε τα αποτελέσματα της Matlab με τα αποτελέσματα της VHDL για κοινές εισόδους, και εξακριβώνουμε την σωστή λειτουργία της σχεδίασης.

5.3 Επικύρωση και Επαλήθευση Αρχιτεκτονικής

Όταν περατώθηκε η κατασκευή της συνολικής αρχιτεκτονικής της συνάρτησης και ελέγχθηκαν όλα τα στάδια επεξεργασίας της, περάσαμε στο στάδιο της επικύρωσης της. Για να εξακριβώσουμε ότι η αρχιτεκτονική βγάζει σωστά αποτελέσματα και η λειτουργία της ανταποκρίνεται στις απαιτήσεις του μοντέλου που είχαμε στα χέρια μας έπρεπε να την ελέγξουμε με πραγματικά δεδομένα και να συγκρίνουμε τα αποτελέσματα της με αυτά της Fortran.

Αρχικά, ελέγχουμε τη συνάρτηση ανεξάρτητα από το υπόλοιπο πρόγραμμα. Οι είσοδοι που χρησιμοποιούμε είναι πραγματικά δεδομένα τα οποία εξάγαμε από το μοντέλο, και εξακιβώνουμε τη σωστή απόκριση της με βάση αυτά. Στη διάθεση μας έχουμε συνολικά 2 διαφορετικά σετ δεδομένων, τα οποία χρησιμοποιούμε για την επικύρωση της αρχιτεκτονικής. Έπειτα, προκειμένου να γίνει ένας πιο αυστηρός έλεγχος βάζουμε και δικά μας σετ δεδομένων που στηρίζονται στη φύση των πραγματικών δεδομένων. Για κάθε είσοδο που χρησιμοποιεί η συνάρτηση από τα πραγματικά σετ δεδομένων έχουμε μελετήσει το εύρος των τιμών της κατά τη διάρκεια εκτέλεσης του μοντέλου. Με αυτό τον τρόπο χρησιμοποιώντας την Matlab φτιάχνουμε καινούργια σετ δεδομένων εντός των προδιαγραφών των πραγματικών δεδομένων και τα χρησιμοποιούμε για τον περεταίρω έλεγχο της αρχιτεκτονικής. Και με αυτά τα σύνολα δεδομένων, επαληθεύουμε τη σωστή λειτουργία της αρχιτεκτονική μας.

Η επικύρωση της συνάρτησης γίνεται αρχικά σε επίπεδο προσομοίωσης και έπειτα εξακριβώνεται και σε πραγματική εκτέλεση πάνω στην FPGA. Η διαδικασία ελέγχου των εξόδων παραμένει η ίδια με αυτήν σε επίπεδο προσομοίωσης. Τα αποτελέσματα από την εκτέλεση στην FPGA εγγράφονται σε αρχείο(*μέσω του περιβάλλοντος του Convey*) και συγκρίνονται με τα αντίστοιχα αποτελέσματα της Matlab. Η επικύρωση γίνεται για όλα τα διαφορετικά σετ δεδομένων που δοκιμάζουμε.

Κεφάλαιο 6

Εγκατάσταση Αρχιτεκτονικής στο Convey και Αποτελέσματα

Το κεφάλαιο αυτό αναφέρεται στην εγκατάσταση της αρχιτεκτονικής στο εργαλείο της Convey και τις μεθόδους που χρησιμοποιήσαμε για την μεταφορά των δεδομένων από τη μνήμη. Επίσης παρουσιάζονται τα τελικά αποτελέσματα σύγκρισης του συνολικού χρόνου απόκρισης της αρχιτεκτονικής μας, σε σχέση με το πρόγραμμα της Fortran.

6.1 Εγκατάσταση Αρχιτεκτονικής στο Convey

Για τον έλεγχο της αρχιτεκτονικής σε πραγματική εκτέλεση πάνω στην FPGA και της σύνδεσης της με το υπόλοιπο πρόγραμμα, αποφασίσαμε να χρησιμοποιήσουμε το εργαλείο της Convey. Ο βασικός λόγος ήταν το εύκολο προγραμματιστικό περιβάλλον και το ευέλικτο interface που μας προσφέρει. Μέσω του Convey, όπως αναφέρεται πιο αναλυτικά και στο παράρτημα Α΄, κάθε FPGA συνδέεται με 8 memory controllers οι οποίοι βλέπουν σε όλα τα bank της μνήμης. Για κάθε memory controller έχουμε 2 διαύλους μεταφοράς δεδομένων, άρα συνολικά μπορούμε να διαχειριστούμε 16 διαφορετικές θύρες εισόδου. Κάτι τέτοιο αποδείχτηκε πολύ χρήσιμο, καθώς το I/O της σχεδίασης μας χαρακτηρίζεται από 14 εισόδους.

Για τον έλεγχο της λειτουργίας της σχεδίασης πάνω στην FPGA, δημιουργήσαμε ένα σύστημα με το προγραμματιστικό περιβάλλον που μας παρέχει το Convey σύμφωνα με το οποίο :

1. Αρχικά φορτώνουμε τα δεδομένα εισόδων στη μνήμη του host συστήματος.
2. Με την εκκίνηση της λειτουργίας της αρχιτεκτονικής τα δεδομένα αυτά μεταφέρονται στη μνήμη του Convey (*co-processor*)
3. Από τη μνήμη του Convey γίνεται η μεταφορά των δεδομένων στις εισόδους της αρχιτεκτονική μας.
4. Μόλις τελειώσει η επεξεργασία της αρχιτεκτονικής τα αποτελέσματα εγγράφονται στη μνήμη του Convey.
5. Έπειτα από τη μνήμη του Convey μεταφέρονται πάλι στη μνήμη του host, για να μπορέσουν να χρησιμοποιηθούν από το υπόλοιπο πρόγραμμα.

Για την εκτέλεση όλων αυτών των βημάτων χρησιμοποιήθηκε το προγραμματιστικό περιβάλλον της C που μας παρέχει το Convey, μέσω του οποίου προγραμματίσαμε την επικοινωνία της σχεδίασης μας με το Convey και την επικοινωνία του Convey με τον Host.

Η επικοινωνία και ο συγχρονισμός για την υλοποίηση όλων των παραπάνω βημάτων όμως, είχε και το αντίστοιχο κόστος. Η υλοποίηση των μέθοδων μεταφοράς δεδομένων από τη μνήμη, ο συγχρονισμός τους και η επικοινωνία της FPGA με όλο το σύστημα απαιτούσαν την ανάλογη κατανάλωση πόρων από την FPGA. Στους παρακάτω πίνακες φαίνεται η κατανάλωση των πόρων της FPGA πριν και μετά την εγκατάσταση της στο Convey . Όπως παρατηρούμε παρουσιάζεται μια σημαντική αύξηση της κατανάλωσης τους.

Resources of Virtex 5 VLX330	% Used
Number of Slice Registers	21
Number of Slice LUTs	23
Number of Block RAM/FIFO	60
Number of DSP48Es	48

Πίνακας 6.1: Συνολικοί πόροι σχεδίασης πριν την εγκατάσταση στο Convey

Resources of Virtex 5 VLX330	% Used
Number of Slice Registers	63
Number of Slice LUTs	60
Number of Block RAM/FIFO	87
Number of DSP48Es	48

Πίνακας 6.2: Συνολικοί πόροι σχεδίασης μετά την εγκατάσταση στο Convey

6.2 Μέθοδοι Μεταφοράς Δεδομένων

Σε αυτή την ενότητα παρουσιάζουμε τις μεθόδους μεταφοράς δεδομένων από την μνήμη προς την αρχιτεκτονική για την επεξεργασία τους, και τις αντίστοιχες συγκρίσεις απόδοσης τους σε σχέση με την εκτέλεση στη Fortran.

Αρχικό στάδιο πριν εφαρμόσουμε οποιαδήποτε μέθοδο μεταφοράς δεδομένων, είναι η μέτρηση του χρόνου απόκρισης της συνάρτησης από το πρόγραμμα της Fortran ανεξάρτητα από το υπόλοιπο μοντέλο. Η μέτρηση αυτή έγινε με το εργαλείο Vtune της Intel.

Επόμενο στάδιο είναι η μεταφορά δεδομένων από τη μνήμη, η μέτρηση της απόδοσης της σχεδίασης μας και η σύγκριση της με τη Fortran. Όπως αναφέραμε και στο κεφάλαιο 4 στην ενότητα 4.3.3 στη λειτουργία της αρχιτεκρονικής διακρίνουμε 2 στάδια. Το στάδιο της προφόρτωσης των δεδομένων και το στάδιο της επεξεργασίας των δεδομένων, όπου η ολοκλήρωση του πρώτου σταδίου είναι απαραίτητη για την έναρξη του δεύτερου. Σε όλες τις μεθόδους που χρησιμοποιούμε ο χρόνος απόκρισης του σταδίου της επεξεργασίας μένει σταθερός, ενώ αυτό που αλλάζει είναι ο χρόνος μεταφοράς δεδομένων μέχρι να προφορτωθούν οι είσοδοι στην αρχιτεκτονική.

6.2.1 Μεταφορά Δεδομένων με Χρήση Block Rams

Η πρώτη μέθοδος μεταφοράς δεδομένων γίνεται με τη χρήση Block Rams on chip, όπου όλα τα δεδομένα προφορτώνονται στις Block Rams. Ο λόγος που πήραμε μετρήσεις με την χρήση προφορτωμένων Block Rams ήταν για να έχουμε μια εικόνα της βέλτιστης απόκρισης της σχεδίασης μας σε συνθήκες, όπου για κάθε κύκλο ρολογιού έχουμε δεδομένα προς επεξεργασία. Με αυτό τον τρόπο, μπορούμε να αποφανθούμε πόση καθυστέρηση μας προσθέτει η μεταφορά των δεδομένων από τη DRAM σε σχέση με την ιδανική περίπτωση.

Με την έναρξη της λειτουργίας της σχεδίασης μας, κατά το στάδιο της προφόρτωσης των δεδομένων, η μεταφορά των εισόδων γίνεται από τις προφορτωμένες Block Rams. Συνολικά έχουμε 14 εισόδους και σε κάθε κύκλο ρολογιού μεταφέρουμε 14 δεδομένα. Μόλις τελειώσει το στάδιο της προφόρτωσης, αρχίζει το στάδιο της επεξεργασίας των δεδομένων. Ο συνολικός χρόνος εκτέλεσης και των 2 σταδίων της σχεδίασης στο hardware προέκυψε 4 φορές πιο μικρός σε σχέση με την εκτέλεση της συνάρτησης στη Fortan.

6.2.2 Συγχρονισμένη Μεταφορά Δεδομένων

Η δεύτερη μέθοδος μεταφοράς δεδομένων είναι η συγχρονισμένη μεταφορά από τη DRAM. Όπως αναφέραμε έχουμε 14 διαφορετικές εισόδους (*πίνακες*) που θέλουμε να προφορτώσουμε στην αρχιτεκτονική. Αρχικά στέλνουμε ένα αίτημα στη μνήμη για την ανάκτηση όλων των δεδομένων και όταν έχουμε ένα έτοιμο δεδομένο και για τις 14 εισόδους τότε μόνο να πραγματοποιείται η μεταφορά τους. Στη συνέχεια στέλνουμε ένα καινούργιο αίτημα ανάκτησης, περιμένουμε να συχρονιστούν όλες οι είσοδοι και προφορτώνουμε με αυτές τις μνήμες της αρχιτεκτονικής. Η διαδικασία συνεχίζεται με τον ίδιο τρόπο μέχρι να προφορτωθούν όλα τα δεδομένα.

Η μέτρηση της απόδοσης αυτής της μεθόδου αποδείχτηκε 14,53 φορές πιο αργή σε σχέση με την εκτέλεση σε Fortran. Αυτό το κακό αποτέλεσμα οφείλεται στο γεγονός του συγχρονισμού όλων των εισόδων προτού προφορτωθούν στις μνήμες της αρχιτεκτονικής. Η πολύ αργή ανάκτηση κάποιων δεδομένων επηρεάζει όλη τη διαδικασία, διότι ο χρόνος μεταφοράς εξαρτάται από το μέγιστο χρόνο ανάκτησης του πιο αργού δεδομένου από τη μνήμη.

6.2.3 Ασύγχρονη Μεταφορά Δεδομένων

Η τρίτη μέθοδος μεταφοράς δεδομένων αποτελεί μια βελτιωμένη έκδοση της δεύτερης, όπου η μεταφορά των δεδομένων από τη DRAM γίνεται με ασύγχρονο τρόπο. Σε αυτή τη μέθοδο κάθε θύρα εισόδου αντιμετωπίζεται ανεξάρτητα από τις υπόλοιπες εισόδους. Λόγω του γεγονότος ότι έχουμε 14 εισόδους, στέλνουμε 14 ανεξάρτητες αιτήσεις προς την μνήμη για την ανάκτηση των αντίστοιχων δεδομένων. Κάθε φορά που ανακτάμε κάποιο/κάποια δεδομένα τα προφορτώνουμε κατευθείαν στην/στις αντίστοιχες μνήμες της αρχιτεκτονικής και ταυτόχρονα στέλνουμε νέα αιτήματα προς τη DRAM για την ανάκτηση των επόμενων στοιχείων. Μόλις προφορτωθεί πλήρως μια μνήμη που αντιστοιχεί σε μια είσοδο γρηγορότερα από τις υπόλοιπες μνήμες, τότε σταματάμε να στέλνουμε νέα αιτήματα στη DRAM για αυτή την είσοδο και απλά περιμένουμε να τελειώσουν την προφόρτωση τους και οι υπόλοιπες μνήμες. Μόλις προφορτωθούν όλες οι είσοδοι τότε αρχίζει το στάδιο της επεξεργασίας των δεδομένων.

Η μέτρηση της απόδοσης αυτής της μεθόδου παρουσιάστηκε αισθητά καλύτερη από την συγχρονισμένη μέθοδο. Πετύχαμε 3,17 φορές πιο γρήγορη εκτέλεση σε σχέση με την εκτέλεση στη Fortran. Επίσης πρόκειται για ένα αποτέλεσμα το οποίο δεν απέχει πάρα πολύ από

την ιδανική περίπτωση με την χρήση των προφορτωμένων Block Rams.

Στο παρακάτω πίνακα παρουσιάζονται συνοπτικά όλες οι εκτελέσεις που έγιναν πάνω σε FPGA για τις διάφορες μεθόδους μεταφοράς δεδομένων, με τους αντίστοιχους συνολικούς χρόνους εκτέλεσης της συνάρτησης και φαίνεται η απόδοση τους σε σχέση με την εκτέλεση σε Fortran. Η μέτρηση του χρόνου εκτέλεσης της συνάρτησης από Fortran έγινε με το εργαλείο Vtune της Intel ενώ για την εκτέλεση σε FPGA μέσω του Convey.

Η σύγκριση των αποτελεσμάτων έγινε μεταξύ ενός συστήματος με επεξεργαστή 8 πυρήνων στα 2,7 GHz και 12 Mb μνήμη για την εκτέλεση σε Fortran, και μιας πλακέτας Virtex 5 (*XC5VLX330*) στα 150 MHz για τις εκτελέσεις σε FPGA.

Μέθοδοι Μεταφοράς Δεδομένων	Σύγκριση με την εκτέλεση σε Fortran (4,70 sec)
Χρήση προφορτομένων Block Rams	4 φορές ταχύτερη (1.17 sec)
Σύγχρονη μεταφορά	14,53 φορές πιο αργή (68.30 sec)
Ασύγχρονη μεταφορά	3,17 φορές ταχύτερη (1.48 sec)

Πίνακας 6.3: Η απόδοση της σχεδίασης με τη χρήση διαφορετικών μεθόδων μεταφοράς δεδομένων.

6.3 Αποτελέσματα

Όπως παρατηρούμε και στον πίνακα 6.3, η ασύγχρονη μεταφορά δεδομένων οδηγεί στην καλύτερη απόδοση, γι αυτό και επιλέξαμε αυτή τη μέθοδο για την υλοποίηση του συστήματος. Πετύχαμε 3,16 φορές καλύτερη απόδοση της συνάρτησης στο hardware, σε σχέση με το αρχικό πρόγραμμα εκτέλεσης στη Fortan.

Σύμφωνα με το νόμο του Amdahl([1]) έχουμε :

$$\frac{1}{(1-P)+\frac{P}{S}} = \frac{1}{(1-0.45)+\frac{0.45}{3.17}} = 1.44$$

όπου P είναι το τμήμα (*45%*) από το συνολικό μοντέλο το οποίο βελτιώσαμε και S είναι η επιτάχυνση (*3.16*) που πετύχαμε πάνω στο τμήμα αυτό. Το αποτέλεσμα μας δείχνει ότι η συνολική επιτάχυνση του υβριδικού μοντέλου είναι 1.44. Δηλαδή έχουμε ένα νέο σύστημα το οποίο έχει 1.44 καλύτερη απόδοση σε σχέση με το αρχικό σύστημα που τρέχει μόνο σε Fortran. Το συγκεκριμένο αποτέλεσμα αναφέρεται στην επίλυση πλεγμάτων μεγέθους 46x25x25 με τη χρήση μίας FPGA.

Κεφάλαιο 7

Συμπεράσματα και Μελλοντική Δουλειά

7.1 Ανασκόπηση

Στη συγκεκριμένη διπλωματική εργασία ασχοληθήκαμε με την βελτίωση του ωκεανικού μοντέλου POM με τη χρήση FPGA.

Αρχικά ασχοληθήκαμε με τη γενική μελέτη και κατανόηση των οικολογικών μοντέλων μέσω της διαθέσιμης βιβλιογραφίας, αλλά και με την δημιουργία δικών μας πειραμάτων. Στη συνέχεια, επικεντρωθήκαμε στη μελέτη του POM. Μέσω της διαδικασίας του profiling αναλύσαμε το μοντέλο και εντοπίσαμε τα κρίσιμα σημεία της λειτουργίας του. Εντοπίσαμε μια βασική συνάρτηση που καταλάμβανε το 45% του συνολικού φόρτου του μοντέλου και αποφασίσαμε να ασχοληθούμε με τη βελτίωση της, προκειμένου να αναβαθμίσουμε τη συνολική απόδοση του μοντέλου. Σκοπός ήταν η δημιουργία ενός υβριδικού μοντέλου, κατά το οποίο η εκτέλεση της συγκεκριμένης συνάρτησης θα γίνεται στο hardware, ενώ το υπόλοιπο πρόγραμμα θα συνεχίζει να εκτελείται στο software. Μέσω της προσομοίωσης σε Matlab μπορέσαμε και μελετήσαμε τη δομή της συνάρτησης και τα μαθηματικά που την διέπουν. Μεγάλο μέρος της μελέτης αποτέλεσε το κομμάτι της αριθμητικής ακρίβειας των αποτελεσμάτων. Ύστερα από μια μεγάλη μελέτη και πολλούς πειραματισμούς με διάφορες αριθμητικές, καταλήξαμε στη χρήση της 32-bit floating point αριθμητικής για την ακρίβεια των πράξεων της συνάρτησης, η οποία είναι μια αριθμητική η απoία ανταποκρίνεται στις απαιτήσεις ακρίβειας του μοντέλου, και παρουσιάζει μικρότερη κατανάλωση πόρων στην FPGA σε σχέση με την 64-bit floating point.

Επόμενο βήμα, ήταν η κατασκευή της αρχιτεκτονικής της συνάρτησης στο hardware, κατά την οποία χρησιμοποιήσαμε την bottom up μέθοδο. Σύμφωνα με αυτή τη μέθοδο, χτίσαμε την αρχιτεκτονική μας βήμα βήμα από τα πιο μικρά κομμάτια ανεβαίνοντας στα πιο μεγάλα και έπειτα στα ακόμη μεγαλύτερα μέχρι την ολοκλήρωση όλης της συνάρτησης. Για κάθε κατσκευή ενός τμήματος της αρχιτεκτονικής γινόταν και ο αντίστοιχος έλεγχος της λειτουργίας του με ένα πολύ μεγάλο αριθμό διαφορετικών εισόδων. Με τη βοήθεια της προσομοίωσης μέσω Matlab είχαμε τη δυνατότητα να εξακριβώνουμε την ορθότητα των εξόδων από το hardware, συγκρίνοντας τες με τις αντίστοιχες που προέκυψαν από την προσομοίωση.

Το βήμα που ακολούθησε την κατασκευή της αρχιτεκτονικής της συνάρτησης και τον

έλεγχο της, ήταν η επαλήθευση των αποτελεσμάτων. Η επαλήθευση των εξόδων της αρχιτεκτονικής έγινε με βάση τα αποτελέσματα της Fortan. Για όλα τα διαθέσιμα σύνολα δεδομένων που είχαμε στα χέρια μας, μπορέσαμε και επαληθεύσαμε τη σωστή λειτουργία της σχεδίασης μας. Επίσης για τον περεταίρω έλεγχο των εξόδων της, φτιάξαμε και δικά μας δεδομένα που ανταποκρίνονταν στις προδιαγραφές των πραγματικών δεδομένων και επαληθεύσαμε την ορθή λειτουργία της.

Επόμενο στάδιο της διπλωματικής, ήταν η εγκατάσταση της σχεδίασης μας στο εργαλείο της Convey. Το ευέλικτο προγραμματιστικό περιβιβάλλον και το εξειδικευμένο εύχρηστο inteface που μας παρείχε, έκαναν τον έλεγχο της λειτουργίας της αρχιτεκτονικής πάνω σε μια FPGA και την διασύνδεση της με το υπόλοιπο πρόγραμμα ευκολότερη. Έτσι, μπορέσαμε και ελέγξαμε πάλι τη λειτουργία της σχεδίασης μας, αυτή τη φορά, όμως, σε πραγματική εκτέλεση πάνω στην FPGA. Μέρος της εγκατάστασης της αρχιτεκτονικής στο Convey, ήταν και η επικοινωνία της με την μνήμη για την μεταφορά των δεδομένων. Παρόλο που το Convey μας παρέχει την δυνατότητα διασύνδεσης 4 παράλληλων συνεργατικών FPGA, εμείς χρησιμοποιήσαμε μόνο τη μία.

Τελευταίο στάδιο της εργασίας ήταν η μέτρηση της απόδοσης της συνάρτησης με εκτέλεση στο hardware σε σχέση με την εκτέλεση στο software. Η υλοποίηση στο hardware έγινε σε μια πλακέτα Virtex 5 (*XC5VLX330*) στα 150 MHz, ενώ στο software με έναν επεξεργαστή 8 πυρήνων στα 2,7 GHz . Η μέτρηση της απόδοσης έγινε με την χρήση διαφόρων μεθόδων μεταφοράς δεδομένων από τη μνήμη. Η τελική υλοποίηση και ταυτόχρονα η πιο γρήγορη πάνω στην FPGA, αποδείχτηκε 3,26 φορές ταχύτερη από την εκτέλεση της συνάρτησης στη Fortran. Σύμφωνα με τον νόμο του Amdahl η επιτάχυνση αυτή μας προσέφερε στο συνολικό μοντέλο 1,44 φορές καλύτερη απόδοση.

7.2 Συμπεράσματα

Μέσω αυτής της διπλωματικής εργασίας μπορέσαμε να κατανοήσουμε τη λειτουργία και την πολυπλοκότητα των οικολογικών μοντέλων και να επιβεβαιώσαμε τη συνεισφορά των FPGAs σε ένα ακόμα επιστημονικό κλάδο. Με την αναδιάταξη του υλικού μπορέσαμε και χτίσαμε μια σχεδίαση που ανταποκρινόταν πλήρως στις απαιτήσεις του κρίσιμου τμήματος του μοντέλου, και έτσι καταφέραμε να βελτιώσουμε την απόδοση του. Η συνολική επιτάχυνση του μοντέλου αποδείχτηκε 1,44 φορές καλύτερη σε σχέση με το αρχικό μοντέλο, και μπορεί σαν αριθμός να μην φαίνεται αρκετά μεγάλος, αλλά η συνεισφορά του σε τέτοιου είδους προβλήματα είναι πολύ μεγάλη. Τα οικολογικά μοντέλα όπως αναφέρουμε και στο κεφάλαιο 1 πρόκειται για αρκετά πολύπλοκα μοντέλα όπου η διάρκεια των προσομοιώσεων τους μπορεί να διαρκέσει πολύ μεγάλα χρονικά διαστήματα. Έστω και αυτή η επιτάχυνση όταν έχουμε να κάνουμε με τόσο χρονοβόρες προσομοιώσεις, μπορεί να αποδειχτεί τεράστιο πλεονέκτημα. Εκτός της συνεισφοράς στην εξοικονόμηση του χρόνου εκτέλεσης των προσομοιώσεων, με αυτή την επιτάχυνση οι επιστήμονες μπορούν να επενδύσουν και στην βελτιωμένη ποιότητα της ακρίβειας των αποτελεσμάτων. Στον ίδιο χρόνο που έτρεχε μια προσομοίωση πριν την βελτίωση, τώρα μπορεί να τρέχει μια ανανεωμένη έκδοση η οποία είτε θα λαμβάνει υπόψην της κάποιο επιπλέον φυσικό παράγοντα, είτε θα τρέχει την υπάρχουσα προσομοίωση με μικρότερο time stem. Με αυτό τον τρόπο μπορούμε να βελτιώσουμε την ποιότητα των αποτελεσμάτων που παίρνουμε στο ίδιο χρονικό διάστημα σε σχέση με τα αποτελέσμτα που παίρναμε πριν τη βελτίωση του μοντέλου. Επίσης, ένα ακόμα βασικό πλεονέκτημα που κερδίζουμε με την χρήση των FPGAs είναι η μειωμένη κατανάλωση της ισχύος. Όπως αναφέραμε και στο κεφάλαιο 1 είναι ένα σημαντικό πρόβλημα που αντιμετωπίζουν τα οικολογικά μοντέλα, λόγω των πολύ μεγάλων

προσομοιώσεων που εκτελούν.

Έτσι και εμείς μέσω αυτής της εργασίας μπορέσαμε να βάλουμε ένα μικρό λιθαράκι, στην προσπάθεια που γίνεται για την βελτίωση της απόδοσης αυτών των απαιτητικών μοντέλων. Πέρα από τα καθαρά πλεονεκτήματα που είχαμε από την χρήση της FPGA, τη σημαντική διαφορά στην απόδοση της σχεδίασης μας, την πετύχαμε με τη χρήση του Convey. Μπορεί να μην χρησιμοποιήσαμε τον παράλληλο συνδιασμό των 4 FPGAs, αλλά το inteface της FPGA με την επικοινωνία της μνήμης ήταν το κλειδί της επιτυχίας. Η συνάρτηση που σχεδιάσαμε χαρακτηρίζεται από υψηλό I/O, αλλά η δυνατότητα που μας παρέχει το Convey με την ταυτόχρονη σύνδεση της FPGA με τους 8 memory controller που βλέπουν σε όλα τα bank της μνήμης, μας έδωσε τη λύση. Μπορούσαμε σε κάθε κύκλο να χειριζόμαστε μέχρι 16 θύρες εισόδου προς τη σχεδίαση μας, κάτι που μας έδινε μεγάλη ευλυγισία στη μεταφορά των δεδομένων, μειώνοντας αισθητά το χρόνο καθυστέρησης και τις συγκρούσεις των δεδομένων.

7.3 Μελλοντική Δουλειά

Σε αυτή την ενότητα αναφερόμαστε στο πλάνο της μελλοντικής δουλειάς που μπορεί να γίνει προκειμένου να βελτιωθεί η απόδοση του μοντελου σε μεγαλύτερο βαθμό και να γίνει ο έλεγχος της απόκρισης του σε μεγαλύτερης κλίμακας προβλήματα.

Η πιο σημαντική βελτίωση που μπορεί να γίνει στο σύστημα που φτιάξαμε είναι η αξιοποίηση και των 4 FPGAs του Convey. Με αυτό τον τρόπο μπορούμε να αυξήσουμε το throughput του συνολικού συστήματος εφόσον οι καθυστερήσεις και οι συγκρούσεις των δεδομένων που μεταφέρονται από την μνήμη δεν είναι μεγάλες. Επίσης, μπορεί να βελτιωθεί ο χρόνος απόκρισης του μοντέλου, αφού πλέον κάθε FPGA θα εκτελεί το 1/4 του συνολικού φόρτου του προβλήματος παράλληλα με τις υπόλοιπες FPGAs. Όμως για την ταυτόχρονη υλοποίηση με 4 FPGAs, απαιτείται τμηματοποίηση του προβλήματος σε 4 κομμάτια τα οποία να μπορούν παραλληλοποιηθούν και η σχεδίαση τους να μην υπερβαίνει τους διαθέσιμους πόρους της κάθε FPGA. Ακόμη, λόγω του μεγάλου I/O που θα προκύψει χρησιμοποιώντας και τις 4 FPGAs, θα πρέπει να σχεδιαστεί ένας κατάλληλος controller για την μνήμη που θα ελαχιστοποιεί τις συγκρούσεις των δεδομένων κατά τη μεταφορά τους. Το I/O για το συγκεκριμένο μοντέλο είναι το πιο σημαντικό και απαιτητικό κομμάτι, στο οποίο θα πρέπει να δωθεί ιδιαίτερη έμφαση.

Επίσης, σαν μελλοντική δουλειά μπορεί να θεωρηθεί ο έλεγχος επίλυσης μεγαλύτερων πλεγμάτων από το μοντέλο. Σε αυτή τη διπλωματική εργασία επιλύουμε πλέγματα τα οποία χωράνε σε μία FPGA (*στη συγκεκριμένη 46x25x25*). Ενδιαφέρον θα παρουσίαζε η απόκριση της σχεδίασης μας στην επίλυση μεγαλύτερων πλεγμάτων τα οποία δεν χωράνε στην FPGA. Αυτό μπορεί να γίνει καλώντας επαναληπτικά τη σχεδίαση στη μία FPGA ή χρησιμοποιώντας και τις 4 FPGAs από το Convey.

Ακόμα ένα σημαντικό κομμάτι που μπορεί να γίνει, είναι η μελέτη για την αποφυγή του σταδίου της προφόρτωσης των δεδομένων της αρχιτεκτονικής. Το κομμάτι της προφόρτωσης καταναλώνει περίπου το μισό χρόνο από τη συνολική εκτέλεση της συνάρτησης. Βέβαια θα αυξηθεί αρκετά το I/O χωρίς αυτό το στάδιο, αλλά με κατάλληλη διαχείριση της μεταφοράς των δεδομένων ίσως μπορέσει να βελτιωθεί η απόδοση σε μεγαλύτερο βαθμό.

Επίσης, μία μελλοντική δουλειά που μπορεί να γίνει για περαιτέρω εξοικονόμηση πόρων, είναι να επανασχεδιαστούν οι μονάδες ελέγχου για τη μεταφορά των δεδομένων από τη μνήμη. Σε αυτή την εργασία τα δεδομένα τα προφορτώνουμε με τη σειρά, με αποτέλεσμα να χρησιμοποιούμε κάποιες ΦΙΦΟ για αυτό το σκοπό, οι οποίες όμως καταναλώνουν σημαντικούς πόρους από βλοςκ ραμς. Αυτό που μπορεί να γίνει είναι να εξαλειφθούν αυτές οι ΦΙΦΟ και τα δεδομένα να προφορτώνονται εκτός σειράς όπως καταφθάνουν. Οι χρήση των βλοςκ ραμς αποτελεί

το πιο κρίσιμο κομμάτι των πόρων οπότε μια τέτοια βελτίωση χρήζεται πολύ σημαντική.

Παράρτημα Α΄

Βασικά Χαρακτηριστικά του Convey

A.1 Convey

At this section we present some basic features of Convey system as described in [2].

The Convey coprocessor is a programmable hardware solution to increase application performance beyond what is typically possible in a standard x86 system. Because of its programmable nature, the hardware allows the architecture to be reconfigured to meet the needs of the application. These reconfigurable instruction sets are called personalities. Convey provides some personalities, such as the single-precision and double-precision vector personalities, that can be used to accelerate certain applications. Some applications, however, require specialized functionality that is not provided by these personalities. As a result, Convey has designed a framework to enable the development of Custom Application Engine Personalities, including instruction set extensions to allow execution of custom instructions.

A.1.1 Coprocessor Architecture

The Convey Coprocessor is made up of three major sets of components: The Application Engine Hub (AEH), the Memory Controllers (MCs) and the Application Engines (AEs). Custom instructions developed for the Convey coprocessor are implemented in the Application Engines FPGAs, and the AEs are the only FPGAs that are reconfigured for different personalities. The AEs contain 4 major interfaces to the rest of the system: the dispatch interface, memory controller interface, CSR/debug interface and AE-to-AE interface.

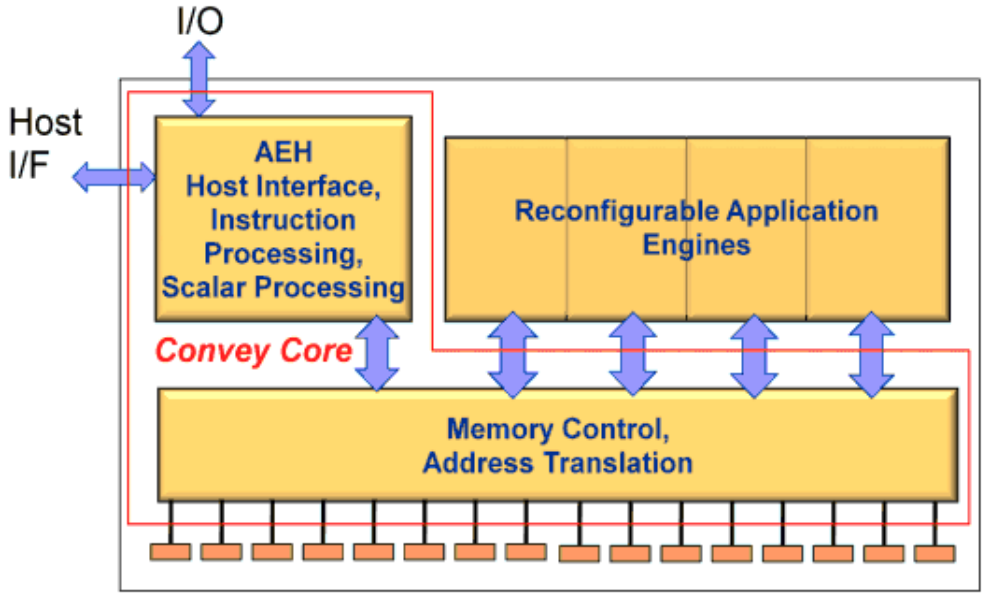


Figure A.1: Convey Coprocessor Block Diagram

A.1.2 Developing a Custom Personality

This subsection contains step-by-step instructions for the process of developing a custom personality for the Convey Coprocessor.

Analyze Application

The first step of personality development is to completely understand the problem to be solved. How does the current application perform on existing hardware? What bottlenecks are limiting the performance? What data structures are involved? How parallelizable is the application? Answers to these questions provide the first insight into how the application can be accelerated in hardware.

Evaluate Hardware Options

With a detailed knowledge of the application and its performance limitations, the second step is to evaluate options for implementing the application in hardware. This requires a good understanding of the hardware architecture and the FPGA resources available to the custom personality. Once a concept for hardware design is completed, the performance of the hardware can be compared relative to the existing application performance.

Define Custom Instructions

With a hardware concept in place, the functions implemented by the hardware design can then be mapped to custom instructions. These are a set of instructions in the Convey Instruction Set Architecture reserved for custom personalities. Instruction sets designed to be used across a wide variety of applications typically have a large number of instructions that perform relatively simple operations. But because a custom personality is designed to

improve the performance of a single application, it might implement very few instructions with much more complex behavior.

Develop Software Model of Custom Personality

Convey provides an architecture simulation environment to allow rapid prototyping of both the hardware and software components of a custom personality. This environment is written in C++ to emulate the rest of the system. It includes hardware models of instruction dispatch, register state and the memory subsystem. With a hardware design concept in place, and a definition of custom instructions to interface to that hardware, a software model can be developed to emulate the hardware. The hardware model can then be simulated with the rest of the system to prove the concept before detailed design begins.

Modify Application to Use Coprocessor

Until a later release of the PDK, the application must be modified in order to dispatch instructions to the coprocessor. The application calls functions that explicitly define the custom instructions to be called.

Simulate Application with Convey Architecture Simulator

Once the AE software model is in place and the appropriate changes to the application have been made, the application can be run against the Convey architecture simulator. This step allows the application and the custom instruction set to be debugged before the hardware is designed.

Develop FPGA Hardware

With an instruction set architecture defined, the hardware implementation can begin.

Simulate Hardware in Convey Simulation Environment

As an extension of the Convey architecture simulator, Convey provides a hardware simulation environment with bus-functional models for all hardware interfaces to the Application Engine (AE) FPGA. Using a standard VPI interface (Verilog Procedural Interface) the architecture simulator can be used to provide stimulus to the HDL simulation.

Integrate with Convey Hardware

The final step is to run the application on the Convey Coprocessor hardware.

A.1.3 Memory Controller Interface

The Memory Controller (MC) Interface gives the AEs direct access to coprocessor memory. Each of the 4 AEs is connected to each of the 8 MCs (Memory Controllers) through a 333MHz DDR interface. The MC interface inside the AE FPGAs is provided by Convey. Each of 8 MC interfaces in the AE FPGA is directly connected to a single Memory Controller, and each MC physically connects to 1/8 of the coprocessor memory.

The diagram A.2 below shows the AE-to-MC connectivity on the coprocessor.

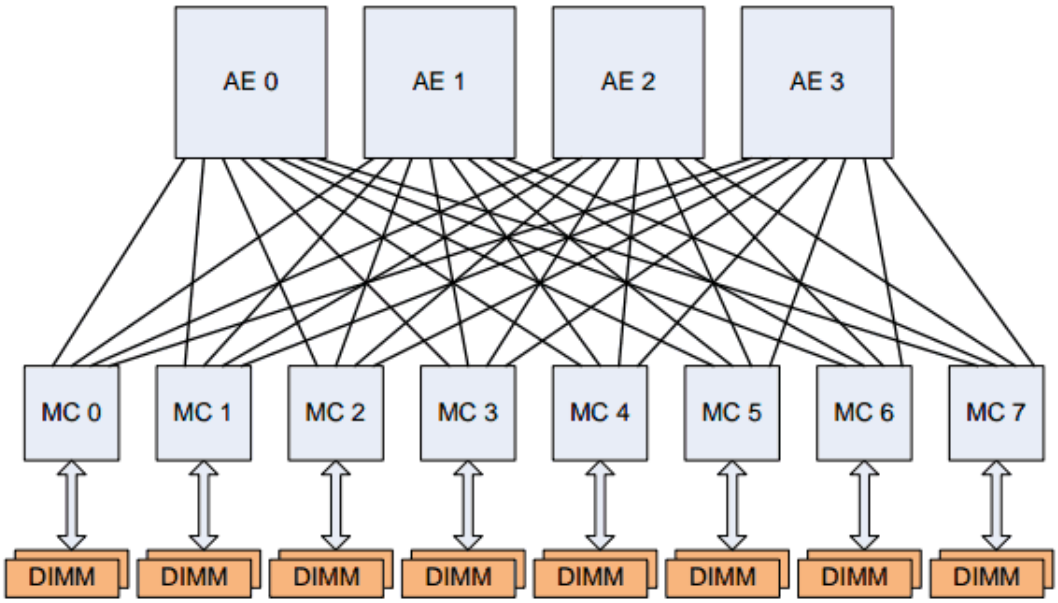


Figure A.2: Coprocessor AE Memory Connections

The 8 MC interfaces are physically located on the left and right sides of the AE FPGA, as shown in Figure A.3 below. Each Memory Controller is connected to 2 DIMMs. The AE personality is responsible for decoding the virtual memory address so that only requests intended for a particular MC's attached memory are sent to that MC. The diagram A.3 below shows the MC interface connections to the Memory Controllers on the coprocessor.

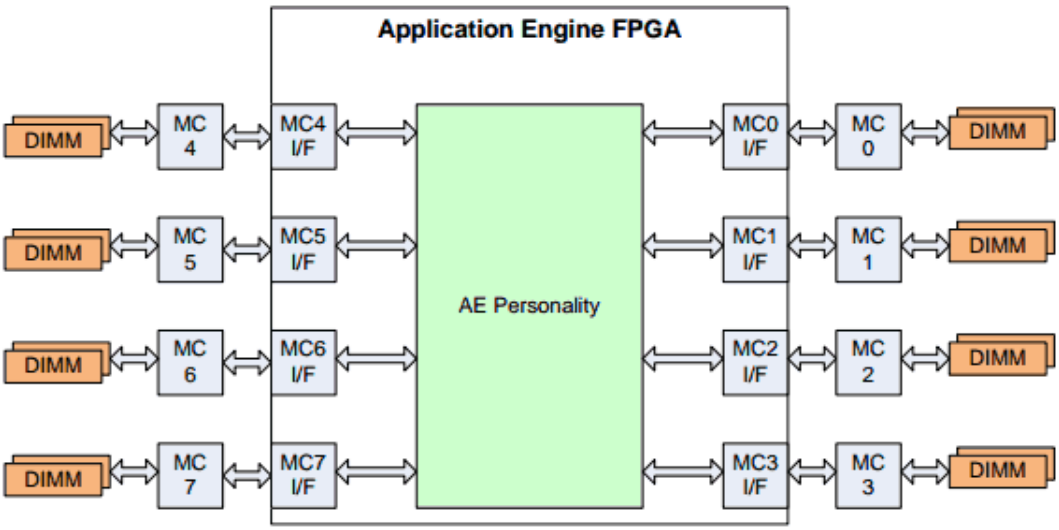


Figure A.3: Coprocessor AE Memory Connections

Each Memory Controller is connected to 2 DIMMs. The AE personality must decode the virtual memory address so that only requests intended for a particular MC's attached

memory are sent to that MC.

Memory System

The Convey memory system using Scatter/Gather DIMMs has 1024 memory banks. The banks are spread across eight memory controllers (MCs). Each memory controller has two 64-bit busses, and each bus in accessed as eight sub busses (8-bits per sub bus). Finally, each sub bus has eight banks. The 1024 banks is the product of 8 MCs * 2 DIMMs/MC * 8 sub bus/DIMM * 8 bank/sub bus. The diagram below shows the coprocessor memory hierarchy.

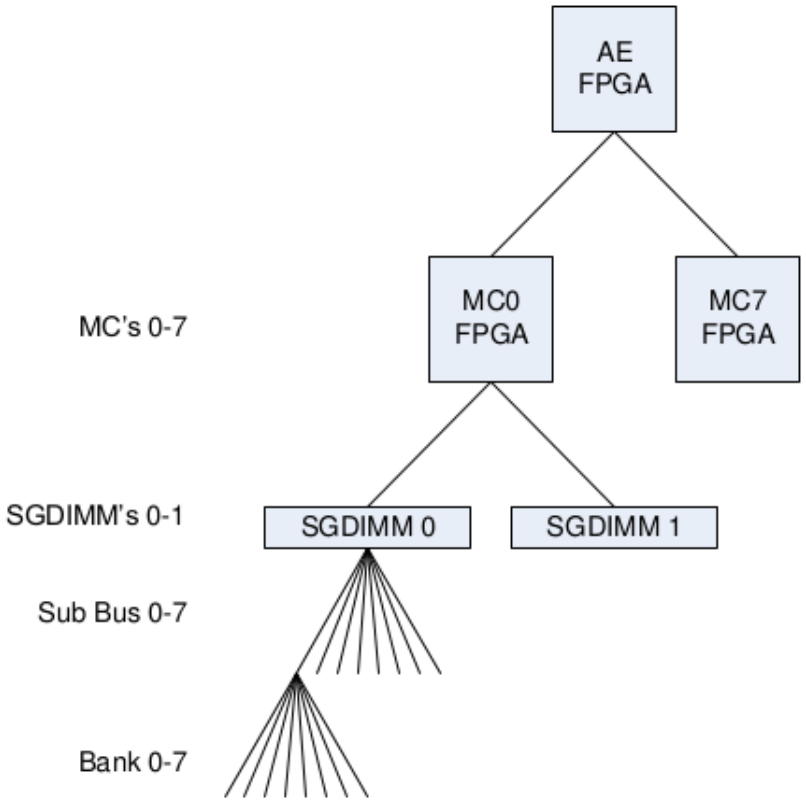


Figure A.4: Memory Hierarchy

Bibliography

[1] amdahl's law. http://en.wikipedia.org/wiki/Amdahl%27s_law.

[2] convey personality development kit reference manual. http://www.google.com/url?sa=t&rct=j&q=convey%20pdk%20manual&source=web&cd=1&cad=rja&ved=0CCsQFjAA&url=http%3A%2F%2Fwikis.ece.iastate.edu%2Fcpre584%2Fimages%2F6%2F64%2FConveyPDKReferenceManual.pdf&ei=xqr7Ufgrg9uzBtadgKAI&usg=AFQjCNEN5MdXYFxqmfYTyuEsKhL0-Itufw&bvm=bv.50165853,d.Yms.

[3] courant–friedrichs–lewy condition. http://en.wikipedia.org/wiki/Courant%E2%80%93Friedrichs%E2%80%93Lewy_condition.

[4] ecosystem model. https://en.wikipedia.org/wiki/Ecosystem_model.

[5] free surface. http://en.wikipedia.org/wiki/Free_surface.

[6] sigma coordinate system. http://en.wikipedia.org/wiki/Sigma_coordinate_system.

[7] state variable. https://en.wikipedia.org/wiki/State_variable.

[8] Anderson, J. D., et al. *Computational fluid dynamics*, vol. 206. McGraw-Hill New York, 1995.

[9] Andrés, E., Carreras, C., Caffarena, G., del Carmen Molina, M., Nieto-Taladriz, O., and Palacios, F. a methodology for cfd acceleration through reconfigurable hardware. In *46th AIAA Aerospace Sciences Meeting and Exhibit* (2008), pp. 7–10.

[10] Benson, B., Cho, J., Goshorn, D., and Kastner, R. field programmable gate array (fpga) based fish detection using haar classifiers. *American Academy of Underwater Sciences* (2009).

[11] Bleichrodt, F., Bisseling, R. H., and Dijkstra, H. A. accelerating a barotropic ocean model using a gpu. *Ocean Modelling 41* (2012), 16–21.

[12] Blumberg, A. F., and Mellor, G. L. a description of a three-dimensional coastal ocean circulation model. *Coastal and estuarine sciences 4* (1987), 1–16.

[13] Chen, B., Zhu, J., and Li, L. accelerating 3d ocean model development by using gpu computing. In *Future Control and Automation*. Springer, 2012, pp. 37–43.

[14] Davidson, R., and Clymer, A. the desirability and applicability of simulating ecosystems. *Annals of the New York Academy of Sciences 128*, 3 (1966), 790–794.

[15] Dongarra, J., Peterson, G., Tomov, S., Allred, J., Natoli, V., and Richie, D. exploring new architectures in accelerating cfd for air force applications. In *DoD HPCMP Users Group Conference, 2008. DOD HPCMP UGC* (2008), IEEE, pp. 472–478.

[16] Higashi, M., and Burns, T. P. *Theoretical studies of ecosystems: the network perspective*, vol. 15. Cambridge University Press Cambridge, MA, 1991.

[17] Hill, T. the benefits of fpga coprocessing. *Xcell Journal. v58* (2006), 29–31.

[18] Kobori, T., and Maruyama, T. a high speed computation system for 3d fchc lattice gas model with fpga. In *Field Programmable Logic and Application*. Springer, 2003, pp. 755–765.

[19] Kobori, T., Maruyama, T., and Hoshino, T. high speed computation of lattice gas automata with fpga. In *Field-Programmable Logic and Applications: The Roadmap to Reconfigurable Computing*. Springer, 2000, pp. 801–804.

[20] Koren, B., Hemker, P., and Everaars, C. multiple semi-coarsened multigrid for 3d cfd. *AIAA paper 97* (1997), 892–902.

[21] Lamoureux, J., Field, T., and Luk, W. accelerating a virtual ecology model with fpgas. In *Application-specific Systems, Architectures and Processors, 2009. ASAP 2009. 20th IEEE International Conference on* (2009), IEEE, pp. 67–74.

[22] Lax, P. D. hyperbolic difference equations: A review of the courant-friedrichs-lewy paper in the light of recent developments. *IBM Journal of Research and Development 11*, 2 (1967), 235–238.

[23] Lin, S.-J., Chao, W. C., Sud, Y., and Walker, G. a class of the van leer-type transport schemes and its application to the moisture transport in a general circulation model. *Monthly Weather Review 122*, 7 (1994), 1575–1593.

[24] Lin Gan, Haohuan Fu, W. L. C. Y. W. X. G. Y. accelerating solvers for global atmospheric equations through mixed-precision data flow engine. IEEE.

[25] Mak, J., Choboter, P., and Lupo, C. numerical ocean modeling and simulation with cuda. In *OCEANS 2011* (2011), IEEE, pp. 1–6.

[26] Mellor, G. L. *Users guide for a three dimensional, primitive equation, numerical ocean model*. Program in Atmospheric and Oceanic Sciences, Princeton University Princeton, NJ 08544-0710, 1998.

[27] Mitsch, W. J., and Jørgensen, S. E. ecological engineering: an introduction to ecotechnology.

[28] Sano, K., Mencer, O., and Luk, W. fpga-based acceleration of the lattice boltzmann method. In *Proceedings of the International Conference on Parallel Computational Fluid Dynamics* (2007).

[29] Shi, P.-J., Men, X.-Y., Sandhu, H. S., Chakraborty, A., Li, B.-L., Ou-Yang, F., Sun, Y.-C., and Ge, F. the "general" ontogenetic growth model is inapplicable to crop growth. *Ecological Modelling 266* (2013), 1–9.

[30] Smith, W. D., and Schnore, A. R. towards an rcc-based accelerator for computational fluid dynamics applications. *The journal of Supercomputing 30*, 3 (2004), 239–261.

[31] Tai, C., Zhao, Y., and Liew, K. parallel computation of unsteady three-dimensional incompressible viscous flow using an unstructured multigrid method. *Computers & structures 82*, 28 (2004), 2425–2436.

[32] Temam, R. *Navier–Stokes Equations*. American Mathematical Soc., 1984.

[33] Wright, N., and Gaskell, P. an efficient multigrid approach to solving highly recirculating flows. *Computers & fluids 24*, 1 (1995), 63–79.