Τελική Τεχνική Έκθεση

Έτος 2013

ΘΑΛΗΣ – Πολυτεχνείο Κρήτης

Πλατφόρμα προηγμένων μαθηματικών μεθόδων και λογισμικού για την επίλυση προβλημάτων πολλαπλών πεδίων (multi physics, multidomain) σε σύγχρονες υπολογιστικές αρχιτεκτονικές: Εφαρμογή σε προβλήματα Περιβαλλοντικής Μηχανικής και Ιατρικής (MATENVMED- MIS 379416)

Δράση 3.2

ΥΛΟΠΟΙΗΣΗ ΣΕ FPGAs ΚΑΙ ΠΟΛΥΠΥΡΗΝΑ ΣΥΣΤΗΜΑΤΑ (MULTICORES / MANYCORES)

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Πίνακας Περιεχομένων

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

1. Σκοπός

Ο σκοπός της Δράσης 3.2 είναι η αποδοτική υλοποίηση των αριθμητικών μεθόδων για ασυνεχή προβλήματα πολλαπλών πεδίων σε αρχιτεκτονικές επεξεργασίας γραφικών (Graphics Processor Units, GPU) καθώς και σε επαναδιατασσόμενες (reconfigurable) αρχιτεκτονικές (πχ. Field Programmable Gate Arrays, FPGAs).

Στόχος της μελέτης θα είναι να εκτιμηθεί η καταλληλόλητα του κάθε πακέτου λογισμικού (συνολικά ή κατά τμήματά του) για εκτέλεση σε καθεμιά από τις πολυπύρηνες πλατφόρμες που θα επιλεγούν ως πειραματικές πλατφόρμες στο έργο, ή εναλλακτικά για επιτάχυνση με χρήση εξειδικευμένου κατά περίπτωση αναδιατασσόμενου υλικού.

Με βάση τα αποτελέσματα της μελέτης θα μεταφερθούν και απεικονιστούν τα πακέτα λογισμικού – είτε καθολικά, είτε κατά τμήματα - στα κατάλληλα κατά περίπτωση πολυπύρηνα συστήματα. Επίσης θα κατασκευαστεί εξειδικευμένο υλικό για την επιτάχυνση συγκεκριμένων υπολογιστικών πυρήνων των εφαρμογών και το απαραίτητο λογισμικό για την επικοινωνία με τον υπολογιστή ξενιστή (Host computer).

Συγκεκριμένα για την υλοποίηση σε FPGA, ενδέχεται να χρησιμοποιηθεί λογισμικό CAD που θα αναπτυχθεί στο Πανεπιστήμιο Θεσσαλίας. Το λογισμικό αυτό μετατρέπει εφαρμογές που έχουν αναπτυχθεί σε OpenCL σε επιταχυντές υλικού (hardware accelerators). Με αυτόν τον τρόπο επιταχύνει κατά πολύ την ανάπτυξη ενός συστήματος FPGA χωρίς να χρειάζονται γνώσεις της αρχιτεκτονικής του συστήματος.

Ως ενδεικτικές πολυπύρηνες πλατφόρμες προτείνονται οι επεξεργαστές γενικού σκοπού της Intel ή της AMD καθώς και GPUs της NVIDIA ή της ATI. Ως ενδεικτική πλατφόρμα αναδιατασσόμενης λογικής προτείνονται συστήματα τα οποία περιλαμβάνουν Xilinx FPGA και μικροεπεξεργαστή σε κάρτα PCI-Express και μπορούν να τοποθετηθούν σε σύστημα αρχιτεκτονικής x86 ή συμβατό.

2. Εισαγωγή στις αρχιτεκτονικές ετερογενών συστημάτων

Σε αυτό το κεφάλαιο θα κάνουμε μία ανασκόπηση των βασικών επεξεργαστικών μονάδων που θα χρησιμοποιηθούν στην Δράση 3.2 του MATENVMED.

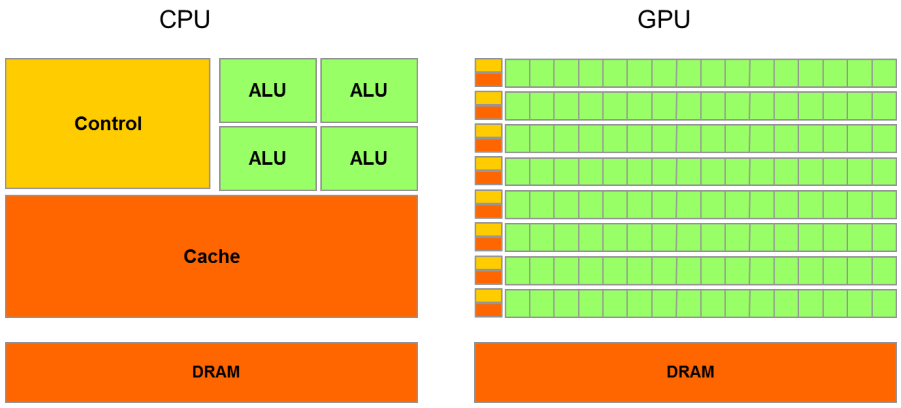
2.1. Τεχνολογία Επεξεργασίας Γραφικών (Graphics Processor Unit, GPU)

Η GPU είναι ένας σύγχρονος πολυεπεξεργαστής (multicore) με υψηλό βαθμό παραλληλίας [4]. Μία σύγχρονη GPU διαθέτει μία ενοποιημένη αρχιτεκτονική για γραφικά και υπολογιστική που χρησιμοποιείται και ως προγραμματιζόμενος

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

επεξεργαστής. Οι αρχιτεκτονικές GPU βασίζονται σε παράλληλες διατάξεις πολλών και σχετικά απλών προγραμματιζόμενων επεξεργαστών (Streaming Processors, SPs).

Σε σύγκριση με τις πολυπύρηνες CPUs (multicore CPUs), οι GPUs έχουν πολύ μεγαλύτερο αριθμό πυρήνων (cores) με διαφορετικό στυλ αρχιτεκτονικής που εστιάζει στην αποδοτική εκτέλεση πολλών νημάτων (threads). Με την χρήση πολλών απλούστερων πυρήνων και με την βελτιστοποίηση της συμπεριφοράς της παραλληλίας δεδομένων μεταξύ ομάδων νημάτων, μεγαλύτερο μέρος των τρανζίστορ του κάθε chip είναι αφιερωμένο σε υπολογισμούς, και μικρότερο σε μνήμες και σε



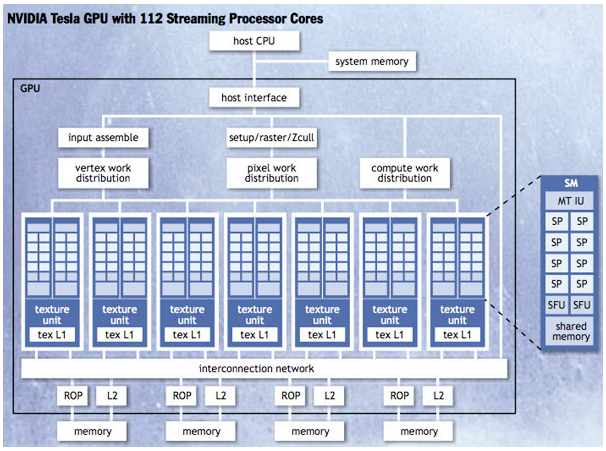
Εικόνα 1. Βασικές αρχές της αρχιτεκτονικής της CPU και της GPU

υλικό ελέγχου (Control) όπως φαίνεται στην Εικόνα 1.

Όπως φαίνεται στην Εικόνα 2, μία σύγχρονη GPU περιέχει πολλούς πυρήνες επεξεργαστών. Ο επεξεργαστής της Εικόνα 2 αποτελείται από 112 επεξεργαστές συνεχούς ροής (Streaming Processors, SPs) οι οποίοι είναι οργανωμένοι ως 14 πολυνηματικοί πολυεπεξεργαστές συνεχούς ροής (Streaming Multiprocessors, SMs). Κάθε SP εκτελεί σε κάθε χρονική στιγμή και ένα νήμα (thread) της γλώσσας CUDA που θα περιγραφεί στο επόμενο κεφάλαιο. Κάθε SM έχει οκτώ πυρήνες SP, δύο μονάδες ειδικών μαθηματικών συναρτήσεων (Special Function Units, SFUs), κρυφές μνήμες cache εντολών και σταθερών, μία Multithreaded Μονάδα εντολών και μία κοινόχρηστη μνήμη η οποία μπορεί να προσπελασθεί από όλα τα SPs που ανήκουν στο δεδομένο SM. Αυτή είναι η βασική αρχιτεκτονική Tesla που υλοποιείται στην GeForce 8800 GPU της NVIDIA. Στην ενοποιημένη αυτή αρχιτεκτονική \εκτελούνται τα παραδοσιακά προγράμματα γραφικών για σκίαση κορυφών, γεωμετρικών σχημάτων και pixels καθώς και προγράμματα γενικότερης υπολογιστικής μέσω της γλώσσας προγραμματισμού CUDA αλλά και της OpenCL.

Η Εικόνα 2 δείχνει 7 ομάδες από SMs όπου δύο SMs μοιράζονται μία μονάδα επεξεργασίας υφής (texture unit) και μία μνήμη L1 cache για δεδομένα (data). Το texture unit παραδίδει φιλτραρισμένα αποτελέσματα στον SM με δεδομένο ένα σύνολο συντεταγμένων ενός χάρτη υφής (texture map). Η μνήμη L1 cache χρησιμεύει

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης



Εικόνα 2. Βασική αρχιτεκτονική της NVIDIA Tesla GPU.

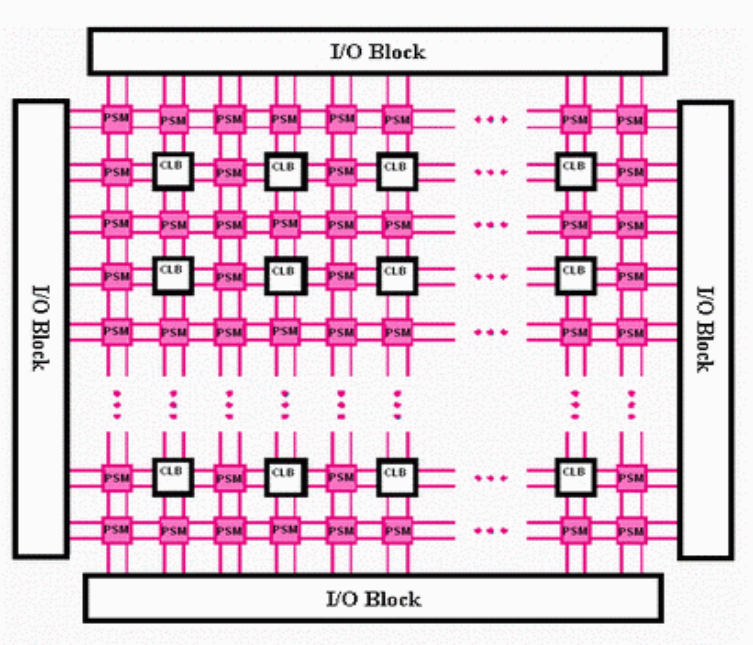
για να μειώσει τον αριθμό των προσπελάσεων από τα SPs στην κύρια μνήμη (DRAM). Ο αριθμός των επεξεργαστών και ο αριθμός των μνημών μπορούν να προσαρμοσθούν ώστε να σχεδιάζονται συστήματα GPU για διαφορετική απόδοση και διαφορετικό κόστος.

Ένα βασικό χαρακτηριστικό της αρχιτεκτονικής GPU είναι ότι όλα τα SPs ενός SM εκτελούν την ίδια εντολή κάθε χρονική στιγμή, πιθανώς με διαφορετικά δεδομένα. Υλοποιεί δηλ. η GPU ένα σύστημα Single Instruction Multiple Data (SIMD). Αυτός ο περιορισμός βοηθάει στην μείωση του instruction bandwidth και ουσιαστικά κάνει δυνατή την ταυτόχρονη εκτέλεση μεγάλου αριθμού νημάτων (threads) σε κάθε κύκλο μηχανής.

2.2. Τεχνολογία Επαναδιατασσόμενης Λογικής (Reconfigurable Logic, FPGAs)

Τα ολοκληρωμένα κυκλώματα επαναδιατασσόμενης λογικής (reconfigurable logic) στα οποία ανήκουν και οι FPGAs (Field Programmable Logic Arrays), περιλαμβάνουν συστοιχίες από υπολογιστικά κυκλώματα (CLBs) η λειτουργικότητα των οποίων μπορεί να μεταβληθεί πολλές φορές κατά τον χρόνο ζωής τους [1]. Ουσιαστικά αυτά τα υπολογιστικά κυκλώματα είναι Lookup Tables (LUTs) με N εισόδους (συνήθως το N είναι μεταξύ 4 και 6) και τα οποία μπορούν να υλοποιήσουν οποιαδήποτε συνδυαστική συνάρτηση boole με N εισόδους. Αυτά τα υπολογιστικά κυκλώματα προγραμματίζονται μέσω configuration bits και συνδέονται μεταξύ τους μέσω πόρων διασύνδεσης (interconnects) που επίσης μπορούν να προγραμματιστούν μέσω configuration bits (Εικόνα 3). Με αυτόν τον τρόπο οποιοσδήποτε αλγόριθμος μπορεί να απεικονισθεί στις συστοιχίες λογικών κυκλωμάτων με το να υπολογίσουμε πρώτα την λογική boolean συνάρτηση του αλγορίθμου, να την απεικονίσουμε στο λογικό κύκλωμα και μετά να διασυνδέσουμε τα λογικά κυκλώματα μεταξύ τους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης



Εικόνα 3. Η αρχιτεκτονική της FPGA.

Αν και η βασική αρχιτεκτονική της Εικόνα 3 παραμένει σχεδόν αναλλοίωτη σε διαδοχικές γενιές κυκλωμάτων επαναδιατασσόμενης λογικής, οι σύγχρονες FPGA περιλαμβάνουν επιπλέον διατάξεις που έχουν πολύ πιο συγκεκριμένη λειτουργικότητα. Τέτοιες διατάξεις είναι οι εξής:

- Εσωτερικές μνήμες SRAM. Κάθε FPGA περιέχει έναν μεγάλο αριθμό από μνήμες οι οποίες μπορούν να προσπελασθούν σε έναν κύκλο μηχανής. Οι μνήμες προσφέρουν πολύ μεγάλο bandwidth στην εφαρμογή γιατί γλυτώνουν το σύστημα από χρονοβόρες και κοστοβόρες προσπελάσεις σε εξωτερικές μνήμες DRAM.
- Μονάδες για Digital Signal Processing (DSP). Οι μονάδες αυτές αποτελούνται από κυκλώματα πολλαπλασιαστών και προσθετών (multiply-add) που είναι αφιερωμένα στο να επιτελούν πολύ γρήγορες πράξεις σε εφαρμογές όπως DSP filtering.
- Ολόκληροι επεξεργαστές και περιφερειακά. Οι σύγχρονες FPGAs έχουν φτάσει σε τέτοιο βαθμό ολοκλήρωσης ώστε είναι δυνατόν να περιέχουν ολόκληρα συστήματα όπως επεξεργαστές ARM και ακόμα και GPUs. Για παράδειγμα, η Zynq FPGA από την Xilinx και την ARM συνδυάζει σε ένα τσιπ έναν διπύρηνο επεξεργαστή ARM Cortex-A9 με τα παραδοσιακά υπολογιστικά κυκλώματα (CLBs) [2]. Αυτή η FPGA μπορεί να τρέξει οποιοδήποτε λειτουργικό σύστημα το οποίο έχει τρέξει στον ARM (πχ Linux) και χρησιμοποιεί τα CLBs κυρίως για την δημιουργία επιταχυντών υλικού (hardware accelerators).

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

2.3. Ετερογενή Συστήματα (Heterogeneous Systems)

Στην σύγχρονη υπολογιστική, ο όρος Ετερογενή Συστήματα (Heterogeneous Systems) αναφέρεται σε υπολογιστικές πλατφόρμες οι οποίες αποτελούνται από ανόμοιους επεξεργαστές. Για παράδειγμα μπορεί να αποτελούνται από συμβατικές πολυπύρηνες CPUs, από GPUs και τώρα τελευταία ακόμα και από FPGAs. Η χρήση διαφορετικών αρχιτεκτονικών σε μία ετερογενή πλατφόρμα έχει επεκταθεί τα τελευταία χρόνια κυρίως λόγω της ανάγκης για αύξηση της ταχύτητας αλλά και της ενεργειακής απόδοσης ενός τέτοιου συστήματος.

Με το να χρησιμοποιούμε ακριβώς τον επεξεργαστή που χρειαζόμαστε για να επιλύσουμε ένα πρόβλημα μπορούμε να επιτύχουμε μείωση της κατανάλωσης ισχύος και ενέργειας στο ποσό που απαιτείται ακριβώς για να επιλυθεί το πρόβλημα. Από την άλλη μεριά όμως, η χρήση διαφορετικών αρχιτεκτονικών δυσκολεύει την ανάπτυξη λογισμικού για ένα τέτοιο σύστημα, μιας και η μηχανικοί λογισμικού θα πρέπει να αναπτύσσουν εφαρμογές χρησιμοποιώντας διαφορετικά εργαλεία για κάθε επεξεργαστή.

Για να επιλυθεί το πρόβλημα του προγραμματισμού εφαρμογών σε ετερογενή συστήματα, η βιομηχανία υπολογιστών δημιούργησε προγραμματιστικά μοντέλα τα οποία μπορούν να χρησιμοποιηθούν για τον προγραμματισμό διαφορετικών αρχιτεκτονικών. Για παράδειγμα, η OpenCL επεκτείνει την γλώσσα C και ορίζει ένα Application Programming Interface (API) το οποίο επιτρέπει σε προγράμματα να τρέχουν σε έναν Host επεξεργαστή (στην CPU) και να δημιουργούν παράλληλους πυρήνες (kernels). Αυτοί οι πυρήνες μπορούν να στέλνονται για εκτέλεση είτε σε μία συμβατική CPU, είτε σε GPUs είτε ακόμα και σε FPGAs. Τα προγράμματα σε OpenCL μπορούν να μεταγλωττιστούν κατά την διάρκεια της εκτέλεσης του προγράμματος (run-time compilation) γεγονός που κάνει τις εφαρμογές σε OpenCL φορητές (portable) για διαφορετικές αρχιτεκτονικές.

3. Μεθοδολογία ανάπτυξης εφαρμογών σε ετερογενή συστήματα

3.1. Προγραμματισμός εφαρμογών σε GPU (CUDA, OpenCL)

Το μοντέλο προγραμματισμού της CUDA (ή OpenCL) επεκτείνει τις γλώσσες προγραμματισμού C και C++ ώστε να εκμεταλλεύονται τους υψηλότερους βαθμούς παραλληλίας που υπάρχουν στις GPUs. Από την αρχική κυκλοφορία της CUDA το 2007, πολλοί προγραμματιστές ανέπτυξαν προγράμματα σε CUDA/OpenCL για μια ευρεία γκάμα εφαρμογών μεταξύ των οποίων multimedia, επεξεργασία σεισμικών δεδομένων, ταξινόμηση, αναζήτηση, υπολογιστική χημεία κοκ.

Ο προγραμματιστής CUDA γράφει ένα σειριακό πρόγραμμα που καλεί παράλληλους πυρήνες (kernels) οι οποίοι μπορεί να είναι από απλές συναρτήσεις μέχρι πλήρη προγράμματα. Ένας πυρήνας εκτελείται παράλληλα από ένα σύνολο

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

παράλληλων νημάτων (threads) έτσι ώστε κάθε νήμα να εκτελείται σε ένα SP σε κάθε χρονική στιγμή. Ο προγραμματιστής οργανώνει αυτά τα νήματα σε μία ιεραρχία μπλοκ νημάτων και πλέγματα από μπλόκς. Ένα τέτοιο μπλοκ νημάτων (thread block) είναι ένα σύνολο από ταυτόχρονα νήματα που μπορούν να συνεργαστούν μέσω συγχρονισμού barrier και μέσω κοινής πρόσβασης σε ένα χώρο μνήμης ιδιωτικό για το μπλοκ. Ένα πλέγμα (grid) είναι ένα σύνολο από μπλοκ νημάτων που το καθένα (μπλοκ) μπορεί να εκτελείται ανεξάρτητα και για αυτόν τον λόγο μπορούν να εκτελούνται παράλληλα.

Όταν καλεί έναν πυρήνα, ο προγραμματιστής καθορίζει τον αριθμό των νημάτων ανά μπλοκ και των αριθμό των μπλοκ που αποτελούν το πλέγμα. Κάθε νήμα λαμβάνει έναν μοναδικό αριθμό ταυτότητας νήματος (thread ID) *threadIdx* μέσα στο μπλοκ νημάτων που ανήκει με αρίθμηση `0, 1, 2, … ,blockDim-1`, και κάθε μπλοκ νημάτων λαμβάνει έναν μοναδικό αριθμό ταυτότητας μπλοκ (block ID) *blockIdx* μέσα στο πλέγμα που ανήκει. Για ευκολία, τα μπλοκ νημάτων μπορούν να έχουν μέχρι και 3 διαστάσεις και προσπελαύνονται μέσω των πεδίων .x, .y και .z.

Όλα τα παραπάνω μπορούν να φανούν με ένα απλό παράδειγμα στο οποίο θέλουμε να προσθέσουμε δύο πίνακες μίας διάστασης (Εικόνα 4). Οι πίνακες είναι κινητής υποδιαστολής απλής ακρίβειας.

Στο δεξί μέρος της Εικόνα 4 ο κώδικας CUDA είναι ο πυρήνας που εκτελεί κάθε νήμα (thread). Όταν ο προγραμματιστής καλεί τον πυρήνα `vadd` από τον κώδικα του Host, το σύστημα χρόνου εκτέλεσης (run time system) της CUDA δημιουργεί έναν αριθμό νημάτων και κάθε νήμα εκτελεί την δική του εκδοχή του πυρήνα. Σε κάθε νήμα αντιστοιχεί ένα διαφορετικό `idx` και κάθε νήμα επεξεργάζεται και διαφορετικό κομμάτι των πινάκων a, b, και c. Η εκτέλεση ενός thread ανατίθεται σε ένα συγκεκριμένο SP στην GPU.

```
void add(int* a,
     int* b,
     int* c) {
for (int idx=0;idx<sizeof(a);
idx++)
  c[idx] = a[idx] + b[idx];
}
```

```
__kernel void vadd(
     __global int* a,
     __global int* b,
     __global int* c) {
 idx = threadIDx.x +
    blockDim.x *blockIdx.x;
 c[idx] = a[idx] + b[idx];
}
```

Εικόνα 4. Κώδικας πρόσθεσης δύο διανυσμάτων σε C (αριστερά) και σε CUDA (δεξιά)

3.2. Προγραμματισμός εφαρμογών σε FPGAs

Προγραμματισμός με HDL

Η ανάπτυξη κυκλωμάτων σε FPGAs είναι μια επίπονη και χρονοβόρα διαδικασία επειδή απαιτεί από τον σχεδιαστή γνώσεις λεπτομερείς γνώσεις υλικού (hardware) και αρχιτεκτονικής. Ο σχεδιαστής θα πρέπει να αναλύσει τις προδιαγραφές του

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

προβλήματος που πρέπει να αναλύσει, να χωρίσει το πρόβλημα σε μικρότερα προβλήματα και να υλοποιήσει το κάθε ένα από αυτά με την χρήση γλωσσών περιγραφής υλικού (Hardware Description Languages, HDL) όπως η Verilog και η VHDL. Οι γλώσσες αυτές περιγράφουν την αρχιτεκτονική του συστήματος με αρκετά μεγάλη λεπτομέρεια σε επίπεδο κύκλων ρολογιού και ως εκ τούτου είναι ακατάλληλες για χρήση από προγραμματιστές λογισμικού που προτιμούν να μην εισέρχονται σε τόσο μεγάλες λεπτομέρειες και σε τόσο χαμηλό επίπεδο κοντά στο hardware. Η υλοποίηση ενός αλγορίθμου σε FPGA απαιτεί τόσο αυτόν κάθε αυτό το σχεδιασμό του αλγορίθμου σε κάποια γλώσσα περιγραφής υλικού όσο και καλή γνώση του τρόπου με τον οποίο τα δεδομένα της εφαρμογής προσπελαύνονται.

Η διαδικασία σχεδίασης ενός αλγορίθμου σε υλικό διαφέρει κατά πολύ από την ανάπτυξη εφαρμογών λογισμικού. Κατά την πλειοψηφία θεωρείται πιο δύσκολη και περισσότερο στρυφνή εφόσον δεν συγχωρεί πολλές παραβλέψεις που κάνουν οι προγραμματιστές λογισμικού. Μία πολύ σημαντική διαφορά μεταξύ λογισμικού και υλικού είναι πως το δεύτερο είναι εν γένει παράλληλο. Έτσι για να γίνει ακόμα καλύτερη εκμετάλλευση των δυνατοτήτων ενός προσαρμοσμένου hardware accelerator χρειάζεται η υλοποίηση να χαρακτηρίζεται από παραλληλία πράξεων.

Η σημασία της γνώσης σχετικά με τον τρόπο με τον οποίο τα δεδομένα μιας εφαρμογής προσπελαύνονται γίνεται εμφανής αν σκεφτεί κανείς πως η μεταφορά και διαχείριση τμημάτων μνήμης παίζει πολύ μεγάλο ρόλο στην απόδοση ενός αλγορίθμου. Στην περίπτωση των hardware accelerators υπό την μορφή FPGA το πρόβλημα γίνεται μεγαλύτερο επειδή η διαθέσιμη μνήμη είναι προδιαγεγραμμένη και σε σύγκριση με ένα τυπικό υπολογιστή κατά πολύ μικρότερη. Έτσι σε περιπτώσεις στις οποίες δεν γίνεται εκμετάλλευση του μοτίβου προσπέλασης δεδομένων, ένας αλγόριθμος μπορεί φαινομενικά να απαιτεί περισσότερη μνήμη από όση είναι διαθέσιμη σε μία δεδομένη FPGA και έτσι να μην είναι δυνατή η σύνθεσή του. Αν όμως αποδειχθεί με μαθηματικό τρόπο πως από η πραγματική απαίτηση μνήμης είναι μικρότερη από τη φαινομενική τότε γίνεται εφικτή η σύνθεση του σε μία FPGA.

Πέρα από το παραπάνω πρακτικό μέρος η γνώση που σχετίζεται με το μοτίβο προσπέλασης μνήμης μπορεί να χρησιμοποιηθεί για περαιτέρω βελτίωση της απόδοσης ενός αλγορίθμου. Εφόσον αναγνωρισθούν τα στοιχεία ενός πίνακας που διαβάζονται από κάποιο κομμάτι κώδικα υλοποιημένο ως hardware accelerator τότε μπορούμε να μειώσουμε το κόστος μεταφοράς μνήμης από τον επεξεργαστή στον hardware accelerator με το να μεταφέρουμε μόνο τα χρήσιμα στοιχεία της μνήμης που αντιστοιχεί στον δεδομένο πίνακα. Αντίστοιχη διαδικασία χρησιμοποιείται για την μεταφορά των αποτελεσμάτων από τον hardware accelerator πίσω στον επεξεργαστή, με το να αναγνωρίζονται και να μεταφέρονται μόνο κομμάτια μνήμης που αντιστοιχούν σε αποτελέσματα που παρήγαγε ο hardware accelerator κατά την εκτέλεσή του.

Προγραμματισμός με C/OpenCL – High Level Synthesis tools

Για να γίνουν οι FPGAs ελκυστικές στο επικρατούσα υπολογιστική (mainstream computing) ή και στην υπολογιστική υψηλών επιδόσεων (high performance computing) θα πρέπει να χρησιμοποιήσουν για τον προγραμματισμό τους μοντέλα

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

πλησιέστερα στα μοντέλα ανάπτυξης λογισμικού. Αυτό γίνεται επιτακτική ανάγκη λόγω της δυνατότητας των FPGAs να ενσωματώνονται σε συστήματα που αποτελούνται από CPU και GPUs και να προσφέρουν υψηλότερη απόδοση και ενεργειακή αποδοτικότητα.

Για λόγους αύξησης της αποδοτικότητας των σχεδιαστών αλλά κυρίως λόγω της επιθυμίας να επεκταθεί ο αριθμός των σχεδιαστών FPGA και σε προγραμματιστές λογισμικού, έχει δημιουργηθεί μια μεγάλη ώθηση τα τελευταία χρόνια σε εργαλεία σχεδίασης υψηλού επιπέδου (High Level Synthesis Tools). Τα εργαλεία αυτά δίνουν την ευκαιρία στον χρήστη να εκφράσει το κύκλωμα που θέλει να υλοποιήσει σε μια FPGA σε κάποια γλώσσα προγραμματισμού υψηλού επιπέδου όπως C, C++, OpenCL και να αναλάβουν την μετάφραση του προγράμματος αυτού σε γλώσσα HDL, Verilog ή VHDL. Εκτός από τον ίδιο τον αλγόριθμο σε κάποια γλώσσα σαν την C, ο σχεδιαστής θα πρέπει να καθορίσει και διάφορους περιορισμούς στο τελικό του κύκλωμα όπως χρόνο απόκρισης (latency), throughput, συχνότητα ρολογιού, αλλά και το μέγεθος του τελικού κυκλώματος σε αριθμό βασικών συστατικών στοιχείων. Αυτοί οι περιορισμοί καθοδηγούν ένα High Level Synthesis Tool στο να δημιουργήσει το κατάλληλο τελικό κύκλωμα σε HDL.

Στην συνέχεια, μια σειρά από εργαλεία CAD μετατρέπουν το HDL κύκλωμα σε λογικές πύλες (gate level netlist) και από εκεί το τοποθετούν στα υπάρχοντα υπολογιστικά κυκλώματα (CLBs) αφού το διασυνδέσουν μέσω των διαύλων διασύνδεσης (interconnects). Το τελικό αποτέλεσμα είναι ένα configuration file το οποίο περιέχει όλα τα configuration bits που διαμορφώνουν την λειτουργικότητα των CLBs και των διασυνδέσεων.

Βασιζόμενοι σε όλες τις προηγούμενες παρατηρήσεις, είναι σκοπός μας να εξετάσουμε την δυνατότητα χρήσης γλωσσών όπως η OpenCL για την απεικόνιση αλγορίθμων σε FPGAs. Πηγαίνοντας ένα βήμα παραπέρα, θα χρησιμοποιήσουμε τον ίδιο κώδικα OpenCL ή C για εκτέλεση σε όλες τις υπολογιστικές πλατφόρμες που θα χρησιμοποιηθούν για το project. Αυτό θα γίνει δυνατόν μέσω λογισμικού CAD που αναπτύσσεται στο Πανεπιστήμιο Θεσσαλίας και που υλοποιεί τεχνικές High Level Synthesis για την δημιουργία επιταχυντών υλικού από προγράμματα OpenCL [3]. Στα πλαίσια του project, το SOpenCL θα βελτιωθεί ώστε να παράγει βέλτιστο υλικό για τους συγκεκριμένους αλγόριθμους που θα παραχθούν στα πλαίσια του project.

Εκτός του SOpenCL θα δοκιμασθούν και εμπορικά εργαλεία High Level Synthesis όπως το Vivado HLS το οποίο αποτελεί και το στάνταρντ εργαλείο για Xilinx FPGAs. Το Vivado HLS δέχεται σαν είσοδο ένα πρόγραμμα γραμμένο σε C ή C++ καθώς και οδηγίες από τον προγραμματιστή μέσω #pragmas που υποδεικνύουν στο εργαλείο τις βελτιστοποιήσεις στην απόδοση ή στο μέγεθος του παραγόμενου υλικού που θα πρέπει να γίνουν.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

4. Υλοποίηση PDEs σε ετερογενές σύστημα CPU/GPU με την χρήση πολυεδρικού μοντέλου

4.1. Εισαγωγή

Όπως είδαμε προηγουμένως, τα ετερογενή συστήματα γίνονται ολοένα και πιο δημοφιλή στην υπολογιστική υψηλών επιδόσεων κυρίως λόγω της δυνατότητας τους να συνδυάζουν υψηλή ταχύτητα με σχετικά χαμηλή κατανάλωση ισχύος. Το σημαντικότερο πρόβλημα τους είναι ότι επιβαρύνουν τον προγραμματιστή όχι μόνο με την εύρεση και εκμετάλλευση του παραλληλισμού σε μια εφαρμογή αλλά επίσης και με την ευθύνη της μεταφοράς δεδομένων από τον χώρο διευθύνσεων ενός επεξεργαστή (πχ CPU) στον χώρο διευθύνσεων ενός διαφορετικού επεξεργαστή (πχ GPU). Ακόμα χειρότερα, η μεγάλη διαφοροποίηση μεταξύ αρχιτεκτονικών και μνημών στα ετερογενή συστήματα αναγκάζουν πολλές φορές τον προγραμματιστή να δημιουργήσει πολλές διαφορετικές εκδοχές ενός κώδικα για να μπορέσει να καλύψει όλες τις περιπτώσεις.

Για να υλοποιήσουμε τον αλγόριθμο επίλυσης PDEs μέσω τεχνικών Monte Carlo, αποφασίσαμε να χρησιμοποιήσουμε το προγραμματιστικό μοντέλο OpenCL που στοχεύει ακριβώς τα ετερογενή συστήματα. Επιπλέον δημιουργούμε ένα πλαίσιο που συνδυάζει compile-time στατική ανάλυση με run-time δυναμική ανάλυση της ροής των δεδομένων μιας εφαρμογής για να αυτοματοποιήσει την διαχείριση δεδομένων σε ετερογενή συστήματα CPU και GPU. Πέρα από την αυτόματη διαχείριση δεδομένων, το πλαίσιο αυτό βοηθάει στην κατανομή των υπολογισμών στους υπάρχοντες επεξεργαστές λαμβάνοντας υπόψιν του την υπολογιστική ισχύ και το ενεργειακό αποτύπωμα κάθε επεξεργαστή, βελτιστοποιώντας κατά αυτόν τον τρόπο την χρήση της υπολογιστικής ισχύς και των μνημών του ετερογενούς συστήματος.

Η μεθοδολογία μας εισάγει μία αμελητέα επιβάρυνση στον χρόνο εκτέλεσης κατά 1.24% σε σχέση με την περίπτωση να γίνει η διαχείριση δεδομένων μόνο από τον επεξεργαστή. Στην συνέχεια, θα εισάγουμε βασικές έννοιες του πολυεδρικού μοντέλου στην Παρ. 4.2, θα περιγράψουμε την μεθοδολογία του πλαισίου μας στην Παρ. 4.3, και θα παρουσιάσουμε τα τελικά αποτελέσματα της μεθόδου μας.

4.2. Πολυεδρικό Μοντέλο

Το πολυεδρικό μοντέλο είναι μαθηματικό πλαίσιο που χρησιμοποιείται κυρίως για ανάλυση βρόγχων (loops) για βελτιστοποίηση της μεταγλώττισης (compilation) σε ένα πρόγραμμα. Η μέθοδος του πολύεδρου θεωρεί ότι κάθε επανάληψη του loop (loop iteration) μπορεί να αναπαρασταθεί από ένα σημείο μέσα σε μαθηματικά αντικείμενα που ονομάζονται πολύεδρα (ή πολύτοπα). Μετασχηματισμοί που εφαρμόζονται πάνω σε πολύεδρα μεταφράζονται σε μετασχηματισμούς του πηγαίου κώδικα του προγράμματος μας. Η πολυεδρική ανάλυση αποτελεί ένα πολύτιμο εργαλείο για βελτιστοποίηση κώδικα μέσω του compiler λόγω της ευχέρειας να αλλάζει την μορφή των loops.

Ένα N-πολύεδρο είναι ένα γεωμετρικό αντικείμενο με επίπεδες πλευρές στον N-διάστατο χώρο. Το πεδίο ορισμού D του πολυέδρου είναι η τομή ενός πεπερασμένου

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

συνόλου από κλειστά ημίσεια διαστήματα (half spaces). Αυτή η αναπαράσταση μπορεί να γραφεί και ως ένα σύστημα από εξισώσεις και ανισώσεις:

$$D: \{x \in Q^n | Ax = b, Cx \geq D\}$$

Η ανάλυση με την μέθοδο του πολυέδρου αναπαριστά τα loops καθώς και τις εντολές μέσα στα loops ως πολύεδρα. Χρησιμοποιείται ευρέως σε δημοφιλής compilers όπως ο LLVM και ο GCC. Μέσω της δημιουργίας πολυέδρων, ένας compiler μπορεί να εντοπίσει εξαρτήσεις μεταξύ εντολών (data dependencies), να δημιουργήσει βελτιστοποιημένους χρονοπρογραμματισμούς εντολών (instruction schedules) και να ανακαλύψει παραλληλισμό μεταξύ εντολών ή επαναλήψεων ενός loop.

Παράδειγμα Ανάλυσης Πολυέδρου. Θα δείξουμε ένα απλό παράδειγμα ανάλυσης χρησιμοποιώντας την μέθοδο του πολυέδρου και αναφερόμενοι στον κώδικα της Εικόνα 5.

```
for (i=2; i <= min(M, -1+N+2), i++) {
    S1(i);
}
```

Εικόνα 5. Απλός κώδικας για το παράδειγμά μας.

Η συνθήκη που ελέγχει τον τερματισμό του loop αντιστοιχεί στον περιορισμό:

$2 \leq i \leq \min(M, -1 + N + 2)$ ο οποίος μπορεί να γραφεί και ως:

$2 \leq i \leq M$, και $2 \leq i \leq N + 1$.

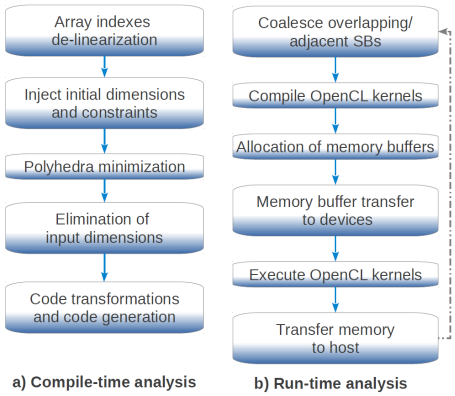
Ένα πολύεδρο μπορεί να αναπαρασταθεί από έναν πίνακα με *(1+Output+Input+Parameters+1)* στήλες και τόσες γραμμές όσος και ο αριθμός των περιορισμών που δημιουργούν το πολύεδρο. Η πρώτη στήλη δείχνει εάν η αντίστοιχη γραμμή περιγράφει ισότητα ή ανισότητα (τιμή 0 ή 1 , αντίστοιχα). Τα *Outputs* αντιστοιχούν σε δείκτες πινάκων, και στο συγκεκριμένο παράδειγμα δείχνουν ότι το εύρος τιμών της μεταβλητής *i*, που είναι το μοναδικό output είναι η ίδια η τιμή του *i*. Η τρίτη στήλη αντιστοιχεί στον μοναδικό iterator *i* του loop, ενώ οι επόμενες δύο στήλες αντιστοιχούν στις παραμέτρους. Η τελευταία στήλη αποθηκεύει το σταθερό κομμάτι των περιορισμών. Ο πίνακας που αντιπροσωπεύει το πολύεδρο του κώδικά μας είναι ο παρακάτω:

$$\begin{pmatrix} 0 & -1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & -2 \\ 1 & 0 & -1 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 1 & 1 \end{pmatrix}$$

4.3. Πλαίσιο ανάλυσης διαχείρισης δεδομένων

Σε αυτήν την παράγραφο θα εξηγήσουμε την μεθοδολογία εύρεσης του μεγέθους της μνήμης που απαιτεί ένας συγκεκριμένος υπολογισμός και τον τρόπο με τον οποίο

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης



Εικόνα 6. Η ροή του υβριδικού αλγορίθμου μας

μπορούμε να αυτοματοποιήσουμε μεταφορές δεδομένων σε μία ετερογενή πλατφόρμα. Αυτή η τεχνική μπορεί επίσης να χρησιμοποιηθεί για την παραλληλοποίηση ενός αλγορίθμου πιθανώς σε τμήματα διαφορετικού βαθμού διακριτότητας (granularity) από ότι αρχικά έχει καθορίσει ο προγραμματιστής.

Η πρώτη φάση της πολυεδρικής ανάλυσης λαμβάνει χώρα κατά την διάρκεια του compilation. Στοχεύει στο να δημιουργήσει μια σειρά από παραμετρικές εξισώσεις που να υπολογίζουν το διάστημα των θέσεων μνήμης που μπορούν να προσπελασθούν από κάθε κομμάτι του κώδικα. Κατά την διάρκεια εκτέλεσης του κώδικα, η δεύτερη φάση διαχωρίζει αυτόματα τα δεδομένα και τα μεταφέρει με έναν βέλτιστο τρόπο μεταξύ των επεξεργαστικών στοιχείων. Η Εικόνα 6 δείχνει την ροή και των δύο φάσεων του αλγορίθμου μας.

Στατική φάση (compilation).

Η ανάπτυξη της στατικής φάσης έγινε με την βοήθεια εργαλείων όπως το Polyhedral Extraction Tool (PET) [7], το ISL [6] και το PolyLib [8]. Το PET παίρνει σαν είσοδο τον πηγαίο κώδικα και δημιουργεί το πολυεδρικό μοντέλο για τα loops του κώδικα αυτού. Η ISL είναι μία βιβλιοθήκη εργαλείων για καλύτερη αναπαράσταση πολυέδρων, ενώ η PolyLib είναι μία βιβλιοθήκη που παρέχει εργαλεία για την επεξεργασία πολυέδρων.

Το αποτέλεσμα της πρώτης φάσης του αλγόριθμού μας είναι ένα σύνολο παραμετροποιημένων (parameterized) διαστημάτων για κάθε πίνακα (array) που προσπελαύνεται μέσα στο loop. Κάθε τέτοιο διάστημα περιέχει επίσης πληροφορίες για τον τύπο της προσπέλασης (read/write). Τα στάδια της φάσης αυτής που φαίνονται και στην Εικόνα 6 είναι τα εξής:

1. Array index de-linearization. Η ανάλυση μας αφορά κάθε εντολή προγράμματος μέσα σε loops που προσπελαύνει μονοδιάστατους και δισδιάστατους πίνακες εφόσον οι προσπελάσεις είναι του τύπου *array[row*width+column]*. Λόγω της παρουσίας της παραμέτρου *width,* που αναπαριστά τον αριθμό των στηλών του πίνακα, το εργαλείο PET δεν μπορεί να ολοκληρώσει την ανάλυση επειδή θεωρεί ότι η αντίστοιχη προσπέλαση δεν είναι συσχετισμένη (affine). Αντιμετωπίζουμε αυτό το πρόβλημα με το να

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

επεκτείνουμε το εργαλείο PET θεωρώντας τους δείκτες *row* και *column* σαν ανεξάρτητους δείκτες και το *width* σαν extra παράμετρο.

2. Inject initial dimensions and constraints. Ο κώδικας ενός παράλληλου πυρήνα (kernels) σε OpenCL ουσιαστικά περικλείεται από 6 loops, τα οποία αντιστοιχούν στην εκτέλεση ενός work-item της OpenCL μέσα σε ένα work-group (τα τρία εσωτερικά loops) και στην εκτέλεση ενός work-group μέσα σε ένα πλέγμα (grid) (τα τρία εξωτερικά loops). Σε αυτό το βήμα, εισάγουμε αυτά τα extra loops στο πολυεδρικό μας μοντέλο.

3. Polyhedral minimization. Ελαχιστοποίηση του μεγέθους του πολυέδρου μέσω της απαλοιφής περιορισμών που είναι πλεονάζοντες.

4. Elimination of input dimensions. Το επόμενο βήμα της στατικής φάσης είναι η μείωση της διάστασης των εισόδων (inputs) του πολυέδρου. Πολλές είσοδοι έχουν σταθερή τιμή κατά την διάρκεια της εκτέλεσης του κώδικα (πχ η παράμετρος *width*) και οι προσπελάσεις της μνήμης λόγω αυτών των εισόδων μπορούν να υπολογισθούν κατά την διάρκεια του compilation. Μετά το τέλος της φάσης αυτής, όλα τα πολύεδρα αποτελούνται από στήλες που αντιστοιχούν σε outputs και παραμέτρους.

5. Code transformations and generation. Ο τελικός στόχος αυτής της φάσης είναι να χωρίσουμε τον αρχικό υπολογισμό σε μικρότερα κομμάτια και να καθορίσουμε για κάθε κομμάτι επακριβώς τα δεδομένα εισόδου και εξόδου που απαιτούνται για να εκτελεσθεί το κομμάτι αυτό. Αυτό απαιτεί το να ξαναγράψει το εργαλείο σε αυτήν την φάση τον αρχικό πηγαίο μας κώδικα.

Ο κώδικας της Εικόνας 7 και της Εικόνας 8 δείχνουν τον κώδικα Monte Carlo για την επίλυση μερικών διαφορικών εξισώσεων (PDEs) πριν και μετά την εφαρμογή του αλγορίθμου μας, αντίστοιχα. Με κόκκινο τονίζουμε τις αλλαγές όπου υπάρχουν.

Βασισμένος στην ανάλυση πολυέδρου, ο compiler δημιουργεί επίσης τον κώδικα για μία νέα συνάρτηση η οποία υπολογίζει τα διαστήματα της μνήμης που προσπελαύνονται από κάθε προσπέλαση μέσα στα κομμάτια του κώδικα που παράγονται σε αυτήν την φάση. Αυτή η συνάρτηση καλείται από το σύστημα χρόνου εκτέλεσης (RTS) όταν κληθεί ο αντίστοιχος παράλληλος πυρήνας (kernel) προς εκτέλεση.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

```
kernel void DoRandomWalks2D(
global float *D, global float *x, global float *result,
unsigned int num_walks, float btol, unsigned int nodes)
{
 private long me = get_local_id(0), us = get_local_size(0);
 private long __group_id_x = get_group_id(0);
 /* Some variable declarations and statements omitted */
 if ( __group_id_x < nodes ) {
  _x[0] = x[__group_id_x*2];
  _x[1] = x[__group_id_x*2+1];
  for ( i=0; i<num_walks; ++i ) {
    _x[0] = x[__group_id_x*2];
    _x[1] = x[__group_id_x*2+1];
    perform_random_walks(num_walks, _x, _D);
  }
  barrier(CLK_LOCAL_MEM_FENCE);
  if ( me == 0 ) {
    d = compose_d();
    result[__group_id_x] = d;
  }
}
```

Εικόνα 7. Ο αρχικός κώδικας OpenCL για την μέθοδο Monte Carlo (MC).

```
/* ocl_offsets: Holds the offset values (Dynamic mode) */
 kernel void DoRandomWalks2D( constant const int *ocl_offsets,
 global float *D, global float *x, global float *result,
 unsigned int num_walks, float btol, unsigned int nodes)
 {
 private long me = get_local_id(0), us = get_local_size(0);
 private long __group_id_x = get_global_id(0)/us;
 /* Some variable declarations and statements omitted */
 if ( __group_id_x < nodes ) {
   _x[0] = x[(__group_id_x-x0y)*x0w -x0o];
   _x[1] = x[(__group_id_x-x1y)*x1w +1 -x1o];
   for ( i=0; i<num_walks; ++i ) {
     _x[0] = x[(__group_id_x-x2y)*x2w -x2o];
     _x[1] = x[(__group_id_x-x3y)*x3w +1 -x3o];
     perform_random_walks(num_walks, _x, _D);
   }
    barrier(CLK_LOCAL_MEM_FENCE);
   if ( me == 0 ) {
    d = compose_d();
     result[__group_id_x -result2o] = d;
   }
 }
}
```

Εικόνα 8. Ο κώδικας OpenCL για την μέθοδο Monte Carlo μετά την εφαρμογή του αλγορίθμου μας

Φάση χρόνου εκτέλεσης.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Αυτή η φάση λαμβάνει χώρα κατά την εκτέλεση του κώδικα λίγο πριν το compilation του πυρήνα OpenCL που πρόκειται να εκτελεσθεί (σημ. στην OpenCL, οι πυρήνες είναι δυνατόν να γίνονται compile κατά την διάρκεια εκτέλεσης). Η συνάρτηση που παράγεται στο τέλος της προηγούμενης φάσης εκτελείται σε αυτήν την φάση με σκοπό να δημιουργήσει τις προσπελάσεις για κάθε συσκευή OpenCL. Τα στάδια της φάσης αυτής που φαίνονται και στην Εικόνα 6b είναι τα εξής:

1. Coalescing of overlapping/adjacent SBs. Επιμέρους περιοχές της μνήμης που προσπελαύνονται από κάθε κομμάτι του πυρήνα OpenCL συνενώνονται για να δημιουργήσουν μία μεγαλύτερη περιοχή με σκοπό να μειωθεί ο αριθμός των μεταφορών δεδομένων μεταξύ Host και των υπόλοιπων συσκευών (CPU ή GPU).
2. Allocation of memory buffers. Μετά την φάση της συνένωσης των επιμέρους τμημάτων μνήμης, γίνεται δυναμική δέσμευση ανάλογης ποσότητας μνήμης σε κάθε επεξεργαστικό στοιχείο (CPU ή GPU) που είναι στο σύστημα μας.
3. Memory buffer transfer to devices. Η μεταφορά δεδομένων μεταξύ του Host και των υπόλοιπων συσκευών γίνεται μόνο την πρώτη φορά ή όταν καινούργια δεδομένα έχουν δημιουργηθεί στις θέσεις μνήμης και θα πρέπει να μεταφερθούν και αυτά.

4.4. Αποτελέσματα της μεθόδου για PDEs

Χρήση Ο αλγόριθμος για υλοποίηση των elliptic PDE solvers είναι ένας εναλλακτικός τρόπος επίλυσης προβλημάτων πολλαπλών πεδίων (multi-domain, multiphysics). Ο βασικός παράλληλος πυρήνας (kernel) του αλγορίθμου αυτού (Εικόνα 3) πραγματοποιεί τυχαίους περιπάτους (random walks) από το σύνορο (boundary) ενός υπόχωρου (sub-domain) στο σύνορο του μεγαλύτερου χώρου (domain). Ο σκοπός του περιπάτου είναι ο υπολογισμός των αρχικών συνθηκών του υπόχωρου για την επίλυση των διαφορικών εξισώσεων.

Μετά από ανάλυση και profiling της εφαρμογής μέσω της σουίτας Intel(TM) Vtune αναγνωρίσαμε δύο κρίσιμα σημεία της εφαρμογής, που καταναλώνουν σχεδόν πλήρως το χρόνο εκτέλεσης της εφαρμογής. Η πρώτη μέθοδος ονομάζεται ***monte_carlo*** και είναι η συνάρτηση υπεύθυνη για την παραγωγή τυχαίων μονοπατιών. Το αμέσως επόμενο πιο χρονοβόρο τμήμα της εφαρμογής είναι η επίλυση συστημάτων μέσω της μεθόδου ***conjugate gradient*** η οποία είναι υλοποιημένη στην βιβλιοθήκη *deal.II*. Αρχικά επιλέξαμε να επικεντρωθούμε στην συνάρτηση ***monte_carlo*** μιας και αυτή αποτελεί ουσιαστικά την ουσία της εφαρμογής.

Πριν παρουσιάσουμε τον λόγο επιτάχυνσης μέσω της χρήσης OpenCL Kernels βρίσκουμε σκόπιμο να αναφέρουμε τα χαρακτηριστικά που έχει η πλατφόρμα πάνω στην οποία εκτελέσαμε τις διαφορετικές εκδόσεις της εφαρμογής.

Για την πειραματική μελέτη απόδοσης χρησιμοποιήσαμε τα εξής δύο υπολογιστικά κυκλώματα:

- Επεξεργαστής γενικού σκοπού: *Intel(R) Core(TM) i7 CPU 870 (2.93GHz)*
- Κάρτα γραφικών: *GeForce GTX 480 (1401MHz)*

Για τα πειράματα που εκτελούνται στον Intel επεξεργαστή (αρχική έκδοση του κώδικα) χρησιμοποιούμε 8 νήματα ενώ για την εκτέλεση στην κάρτα γραφικών χρησιμοποιούμε N*768 νήματα όπου N ο αριθμός των σημείων στα οποία θα εφαρμοστεί η μέθοδος ***monte_carlo***. Επειδή η συνολική επιτάχυνση του προγράμματος διαμορφώνεται σε σχέση με το μέγεθος του προβλήματος που τίθεται για επίλυση παρουσιάζουμε μερικές ενδεικτικές εκτελέσεις στον παρακάτω πίνακα:

Χρόνος εκτέλεσης CPU (Διάσταση Χώρου)	**Χρόνος εκτέλεσης GPU / Χρονοβελτίωση**	**Μέσο σφάλμα CPU**	**Μέσο Σφάλμα GPU**
100 secs ***(2D)***	*34 secs /* ***2.94x***	*0.0002041409246*	*8.811696406e-05*
241 secs ***(3D)***	*35 secs /* ***6.89x***	*0.01288890583*	*0.01290900319*
761 secs ***(2D)***	*85 secs /* ***8.95x***	*0.0002024040681*	*2.788477219e-05*
2026 secs ***(3D)***	*2004 secs /* ***1x***	*0.009886439747*	*0.009778896185*

Επιλέξαμε τυχαία 4 παραδείγματα για να παρουσιάσουμε την βελτιστοποίηση στη μέθοδο μας. Το γεγονός ότι η GPU φαίνεται να παράγει ορθότερες εκτιμήσεις είναι τυχαίο, σε γενικές γραμμές οι εκτιμήσεις των δύο υλοποιήσεων έχουν παραπλήσιες τιμές. Ιδιαίτερο ενδιαφέρον παρουσιάζει η αυξομείωση της χρονοβελτίωσης, με αποκορύφωμα το τελευταίο από τα 4 παραδείγματα. Αν και δεν οδήγησε σε ουσιαστική διαφορά στο συνολικό χρόνο εκτέλεσης, η συνάρτηση που επιλέξαμε να επιταχύνουμε είχε χρονοβελτίωση ***10.67x*** όμως η διάρκεια εκτέλεσής της ήταν *32* και *3* δευτερόλεπτα σε επεξεργαστή και κάρτα γραφικών, αντίστοιχα. Αυτό είναι ένδειξη πως μπορούμε να λάβουμε περαιτέρω χρονοβελτίωση με την βελτιστοποίηση της συνάρτησης που επιλύει τα συστήματα που προκύπτουν (*μέθοδος conjugate gradient*) μετά τη διαδικασία εκτίμησης (***monte_carlo***).

Μέσο σφάλμα

Η επιταχυμένη υλοποίηση τείνει να παράγει καλύτερες προσεγγίσεις, δηλαδή με μικρότερο μέσο σφάλμα. Αυτό οφείλεται στο γεγονός ότι τα βάρη των μονοπατιών αθροίζονται χρησιμοποιώντας 768 double precision floating point αριθμούς. Η αρχική υλοποίηση χρησιμοποιούσε 1 τέτοια μεταβλητή. Λόγω της φύσης των floating point αριθμών όταν προστίθενται μεταξύ τους αριθμοί εισάγεται ένα σφάλμα στο τελικό αποτέλεσμα, το οποίο μεγαλώνει όσο μεγαλώνει η απόσταση των αριθμών. Επειδή πλέον χρησιμοποιούνται πολλές μεταβλητές για την άθροιση των μονοπατιών είναι

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

λιγότερο πιθανό να προστεθούν αριθμοί πολύ διαφορετικοί σε σχέση με την πρώτη υλοποίηση που ήταν "απόλυτα σίγουρο" μιας και υπήρχε μόνο 1 μεταβλητή στην οποία γινόταν όλο το άθροισμα για ένα σημείο.

Μεθοδολογία

Οι αλλαγές που κάναμε έγιναν με τέτοιο τρόπο ώστε ο τρόπος υπολογισμού της μεθόδου ***monte_carlo*** να επιλέγεται κατά την εκτέλεση της εφαρμογής. Αυτό μας οδήγησε αρχικά στην αλλαγή του τρόπου δέσμευσης μνήμης για την αποθήκευση των δεδομένων εισόδου της συνάρτησης. Από μία λίστα με λίστες μεταβλητών double precision καταλήξαμε πάλι σε μία δισδιάστατη δομή της οποίας όμως τα στοιχεία είναι αποθηκευμένα σε διαδοχικές θέσεις μνήμης. Έτσι δεν χρειάστηκαν να γίνουν αλλαγές στον αρχικό κώδικα ενώ η μεταφορά δεδομένων προς την GPU έγινε στη συνέχεια με πολύ απλό τρόπο.

Το επόμενο βήμα ήταν να δημιουργηθεί μία απλή υλοποίηση του πυρήνα της ***monte_carlo***, δηλαδή ο υπολογισμός ενός μόνο μονοπατιού, σε γλώσσα C ώστε να μην έχουμε τα overheads της C++ και να μπορεί να γίνει απεσφαλμάτωση γρήγορα και εύκολα πριν καν γίνει η τελική υλοποίηση στην κάρτα γραφικών. Σε αυτό το σημείο εντοπίσαμε πως υπήρχε μια βοηθητική συνάρτηση η οποία καλείται πολλές φορές και έκανε χρήση τεσσάρων if-statements ώστε να υπολογίσει την ελάχιστη τιμή τεσσάρων παραστάσεων. Κάναμε μερικές αλλαγές στον κώδικα ώστε να μειώσουμε τους ελέγχους έχοντας ως σκοπό να καταλήξουμε σε γρηγορότερη εκτέλεση, δυστυχώς όμως η επιτάχυνση λόγω αυτής της βελτιστοποίησης ήταν μηδαμινή και στις δύο υπολογιστικές μονάδες.

Η OpenCL υλοποίηση χρησιμοποιεί πολλαπλά νήματα για τον υπολογισμό της τιμής σε ένα σημείο. Αυτό γίνεται αναθέτοντας σε καθένα από αυτά ένα κλάσμα του συνολικού αριθμού μονοπατιών που πρέπει να "περπατηθούν" ώστε να προκύψει η εκτιμώμενη τιμή του σημείου. Ενδεικτικά αναφέρουμε πως η τρέχουσα υλοποίηση όταν εκτελείται σε μία GeForce GTX 480 χρησιμοποιεί **768** τέτοια νήματα για τον υπολογισμό κάθε σημείου, όταν ο χώρος έχει 2 διαστάσεις.

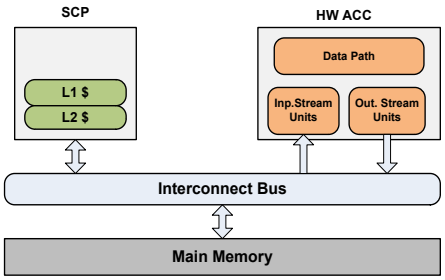
Αφού επιβεβαιώσαμε την ορθότητα του κώδικα εκτελώντας πολλαπλά πειράματα και έχοντας τις ίδιες τιμές που προκύπτουν από την αρχική υλοποίηση συνεχίσαμε με το σχεδιασμό του OpenCL Kernel. Αυτή τη φορά όμως λόγω αύξησης στον αριθμό των καταχωρητών που χρησιμοποιούνται από τον OpenCL Kernel για τα μονοπάτια σε 3D χώρο τα νήματα που χρησιμοποιούνται για την εκτίμηση ενός σημείου μειώθηκαν στα **576**.

Τέλος και στους 2 kernels χρησιμοποιήσαμε τις native υλοποιήσεις των συναρτήσεων ***cos/log/sin*** σε σημεία που δεν επηρεάζουν την ακρίβεια της εκτίμησης, μιας και εφαρμόζονται σε δεδομένα τα οποία παράγονται με ψευδοτυχαίο τρόπο. Έτσι η μείωση της ακρίβειας που υπεισέρχεται λόγω των native υλοποιήσεων εμφανίζεται ουσιαστικά ως επιπλέον τυχαιότητα, μάλιστα παρατηρήσαμε ότι σε πολλές περιπτώσεις η ακρίβεια των εκτιμήσεων βελτιώνεται.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

5. Χρήση μοντέλου παράλληλου προγραμματισμού OpenCL για σύνθεση αρχιτεκτονικών (Silicon OpenCL)

Σε αυτήν την έκθεση θα περιγράψουμε το SOpenCL, ένα νέο εργαλείο και μεθοδολογία η οποία δημιουργεί επιταχυντές υλικού (hardware accelerators) που είναι τμήματα ενός ολόκληρου συστήματος (System On Chip, SoC), όπως φαίνεται και στην Εικόνα 9.



Εικόνα 9. Το τελικό μας σύστημα που περιλαμβάνει τον επιταχυντή υλικού (HW ACC) που παράγει η OpenCL και τον κύριο επεξεργαστή (Scalar Processor, SCP).

Η OpenCL είναι ένα βιομηχανικό standard για την δημιουργία παράλληλων προγραμμάτων με σκοπό την εκτέλεση σε ετερογενή συστήματα (heterogeneous systems) [9]. Απαλλάσσει τον προγραμματιστή από το να γράψει κώδικα για κάθε πιθανή ετερογενή πλατφόρμα και με το να ασχοληθεί με τεχνικές λεπτομέρειες χαμηλού επιπέδου (low level details) δίδοντάς του την ευκαιρία να εστιάσει στον ίδιο τον αλγόριθμο και στο πρόβλημα που καλείται να επιλύσει.

Υπάρχουν κάποιες σημαντικές τεχνικές δυσκολίες στον σχεδιασμό και την υλοποίηση του SOpenCL [13]. Η OpenCL χρησιμοποιεί παράλληλους πυρήνες (kernels) για να εκφράσει υπολογισμούς σε πολύ χαμηλό επίπεδο (granularity), που είναι το επίπεδο του work-item (ή thread όπως έχουμε δει στην CUDA). Αυτό είναι ιδιαίτερα χρήσιμο γιατί φανερώνει τον παραλληλισμό της εφαρμογής στο χαμηλότερο επίπεδο και διευκολύνει την εκτέλεση του προγράμματος από CPU και GPU. Παρόλα αυτά, η δημιουργία επιταχυντών υλικού στο επίπεδο του work-item μπορεί να μην είναι ιδανική λύση λόγω του ότι κάθε τέτοιος επιταχυντής θα κάνει πολύ λίγη δουλειά κάθε φορά που καλείται.

Σαν πρώτη φάση, το εργαλείο SOpenCL περιέχει έναν μεταγλωττιστή source-to-source που μεταφράζει τον κώδικα OpenCL σε κώδικα C σε διαφορετικό όμως επίπεδο παραλληλισμού από τον κώδικα OpenCL. Στην συγκεκριμένη περίπτωση ο μεταγλωττιστής μετατρέπει τον κώδικα σε επίπεδο work-group (ή block), έτσι ώστε κάθε κλήση του επιταχυντή να εκτελεί εργασία που να αντιστοιχεί σε ένα work-group.

Η δεύτερη φάση του SOpenCL μετατρέπει τον C κώδικα σε έναν επιταχυντή υλικού σε synthesizable Verilog. Ο επιταχυντής ακολουθεί ένα συγκεκριμένο πρότυπο αρχιτεκτονικής (architectural template) το οποίο παίρνει συγκεκριμένη

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

μορφή ανάλογα με τις απαιτήσεις απόδοσης του χρήστη και την συγκεκριμένη εφαρμογή που υλοποιείται στην FPGA. Η δεύτερη φάση του SOpenCL αποτελείται από μία σειρά από βελτιστοποιήσεις του compiler όπως predication, code slicing και modulo scheduling που εφαρμόζονται διαδοχικά στον κώδικα C. Στο τελικό στάδιο το εργαλείο SOpenCL δημιουργεί synthesizable Verilog.

5.1. Πρώτη φάση του SOpenCL

Για να επιτύχουμε την αποδοτική εκτέλεση παράλληλων πυρήνων OpenCL, εφαρμόζουμε μια αλληλουχία από μετατροπές source to source για να ανεβάσουμε το επίπεδο του πυρήνα από το work-item στο work-group. Μετά τις μετατροπές αυτές στον πηγαίο κώδικα, ο νέος παράλληλος πυρήνας εκφράζει τον υπολογισμό που εκτελεί το work-group.

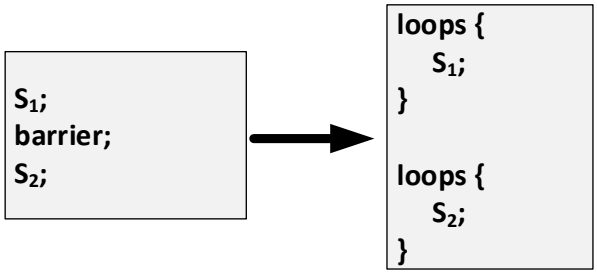
Η πρώτη φάση του SOpenCL αποτελείται από δύο βήματα:

1. Σειριοποίηση των λογικών νημάτων (Logical Thread Serialization). Αυτό το βήμα περικλείει τον κώδικα του πυρήνα με ένα τριπλό loop και με αυτόν τον τρόπο μετατρέπει τον υπολογισμό ενός work-item σε υπολογισμό ενός work-group.
2. Απαλοιφή εντολών συγχρονισμού (barriers). Η OpenCL δίνει την δυνατότητα στον προγραμματιστή να συγχρονίσει την εκτέλεση των work-items που εκτελούν έναν κώδικα OpenCL μέσω των εντολών barrier. Μία εντολή barrier αναγκάζει όλα τα work-items ενός work-group να εκτελέσουν την εντολή αυτή πριν μπορέσει οποιοδήποτε work-item να συνεχίσει με την εκτέλεση των εντολών μετά την barrier. Αντίστοιχα, η εντολή barrier μέσα σε ένα loop αναγκάζει όλα τα work-items να περιμένουν στην εντολή αυτή πριν μπορέσουν να συνεχίσουν με την επόμενη επανάληψη του loop.

   Για να εξασφαλίσουμε σωστή λειτουργικότητα κατά την μετατροπή του κώδικα από OpenCL σε C σε περίπτωση που υπάρχουν εντολές barrier, χωρίζουμε τον κώδικα σε δύο μπλόκς, ένα πριν την barrier και ένα μετά έτσι ώστε αυτά τα μπλόκς να μην περιέχουν την εντολή barrier. Μετά εφαρμόζουμε την τεχνική loop fission για να χωρίσουμε το αρχικό τριπλό loop σε δύο επιμέρους τριπλά loops για να εξασφαλίσουμε την σωστή μετατροπή του κώδικα σε C (Εικόνα 10).

Περισσότερες λεπτομέρειες για τον μετατροπές που εφαρμόζει ο compiler υπάρχουν στο [13].

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης



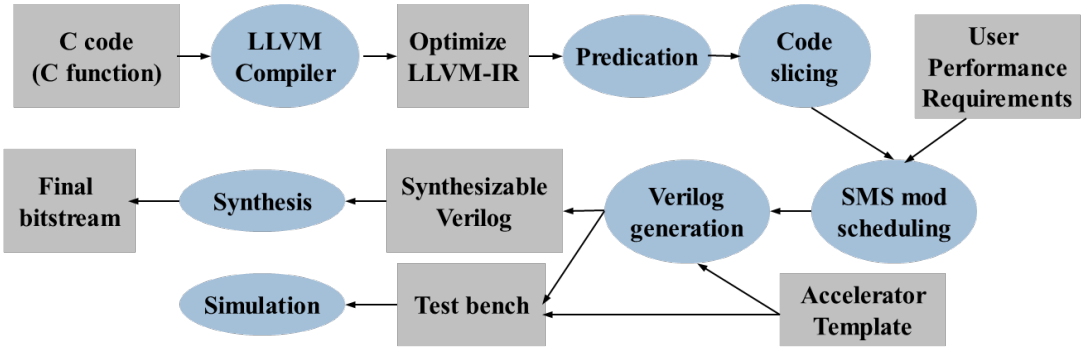
Εικόνα 10. Απαλοιφή εντολών barriers με την χρήση loop fission.

5.2. Δεύτερη φάση του SOpenCL

Μετά την πρώτη φάση, το SOpenCL μεταφράζει τον κώδικα σε Verilog σύμφωνα με την σειρά των βημάτων που φαίνονται στην Εικόνα 11. Ο αλγόριθμος αυτός χρησιμοποιεί και επεκτείνει τον LLVM compiler [11]. Ένα σημαντικό χαρακτηριστικό του SOpenCL είναι ότι χρησιμοποιεί ένα καλά σχεδιασμένο αρχιτεκτονικό πρότυπο (template) για να δημιουργήσει υλικό όπως φαίνεται στην Εικόνα 12.

Πρότυπο αρχιτεκτονικής

Η μονάδα δεδομένων της Εικόνα 12 αποτελείται από ένα δίκτυο από λειτουργικές μονάδες (Functional Units, FUs) που αναλαμβάνουν το κυρίως έργο της επεξεργασίας δεδομένων. Οι μονάδες αυτές λαμβάνουν δεδομένα από τις μονάδες Εισόδου/Εξόδου Ροής σε μορφή FIFO και παράγουν τα δεδομένα εξόδου πάλι σε μορφή FIFO. Η διασύνδεση μεταξύ των λειτουργικών μονάδων γίνεται μέσω κυκλωμάτων ουρών FIFO που αποτελούνται από πολυπλέκτες και buffers για να κατευθύνουν δεδομένα από τις μονάδες που παράγουν στις μονάδες που καταναλώνουν ενδιάμεσα αποτελέσματα.

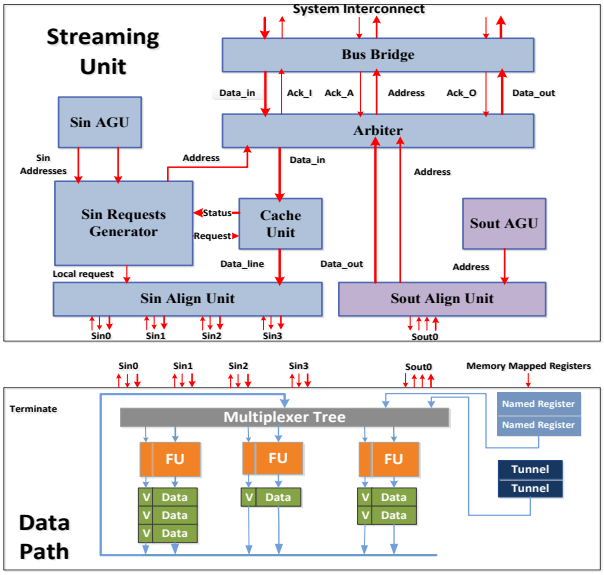


Εικόνα 11. Ο αλγόριθμος δημιουργίας επιταχυντή υλικού από κώδικα C.

Το SOpenCL διαμορφώνει τον τελικό επιταχυντή υλικού μεταβάλλοντας τον τύπο του μίγματος των FUs (δηλ. πόσους multipliers, adders, shifters, κοκ), την συγκεκριμένη λειτουργία που επιτελούν, την διασύνδεση μεταξύ των FUs καθώς και το bandwidth μεταξύ της μονάδας δεδομένων και της μονάδας Εισόδου/Εξόδου.

Η μονάδα Εισόδου/Εξόδου Ροής αναλαμβάνει όλες τις αρμοδιότητες που έχουν να κάνουν με την μεταφορά δεδομένων μεταξύ της κύριας μνήμης και του επιταχυντή.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης



Εικόνα 12. Το πρότυπο αρχιτεκτονικής που χρησιμοποιούμε περιλαμβάνει την μονάδα Δεδομένων (Data Path) και την μονάδα Εισόδου/Εξόδου Ροής (Streaming Unit)

Αυτές οι αρμοδιότητες είναι η δημιουργία των διευθύνσεων μνήμης, ευθυγράμμιση και ταξινόμηση δεδομένων από και προς τον επιταχυντή καθώς και προσαρμογή μεταξύ του επιταχυντή και του διαύλου του συστήματος (system bus). Η μονάδα Εισόδου/Εξόδου Ροής αποτελείται από μία ή περισσότερες μονάδες Εισόδου και Εξόδου. Όπως και για την μονάδα Δεδομένων, έτσι και εδώ το SOpenCL συνθέτει το κύκλωμα έτσι ώστε να ταιριάζει στα χαρακτηριστικά της εφαρμογής και στις απαιτήσεις του υπόλοιπου συστήματος.

Ροή του αλγορίθμου SOpenCL

Αναφερόμενος στην Εικόνα 11, βλέπουμε ότι το εργαλείο SOpenCL ακολουθεί μια σειρά από βήματα στον μεταγλωττιστή LLVM για να παράγει το τελικό κύκλωμα του επιταχυντή σε Verilog. Εν συντομία, τα πιο σημαντικά βήματα είναι τα εξής:

a) Compiler optimizations and Predication. Το πρώτο βήμα είναι να μετατρέψουμε τον κώδικα C σε εσωτερική μορφή του LLVM. Ο ίδιος ο LLVM εφαρμόζει μια σειρά από στάνταρντ βελτιστοποιήσεις του κώδικα όπως constant propagation, dead code elimination, strength reduction, κοκ. Το predication είναι μία τεχνική του μεταγλωττιστή η οποία μετατρέπει εξαρτήσεις ελέγχου (control dependencies) σε εξαρτήσεις δεδομένων (data dependencies). Το predication εξαλείφει όλες τις εντολές διακλάδωσης (branches) στον κώδικα C με την χρήση επιπλέον μεταβλητών του 1-bit οι οποίες ονομάζονται Boolean guards. Ο βασικός σκοπός του predication είναι η εξάλειψη όλων των διακλαδώσεων που προέρχονται από εντολές if-then-else, και η προετοιμασία του κώδικα για την επόμενη φάση του modulo scheduling.

b) Code slicing. Ο κώδικας C μετά το πρώτο βήμα εμπεριέχει εντολές Εισόδου/Εξόδου δηλ. εντολές φορτώματος και αποθήκευσης στην μνήμη καθώς και εντολές υπολογισμού. Το βήμα του code slicing διαχωρίζει τον

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

κώδικα σε 3 κομμάτια: (i) το κομμάτι κώδικα που κάνει μόνο υπολογισμούς, (ii) το κομμάτι κώδικα που περιέχει όλες τις εντολές φορτώματος (load) από την μνήμη και τις εντολές υπολογισμού της διεύθυνσης φορτώματος, και (iii) το κομμάτι κώδικα που περιέχει όλες τις εντολές αποθήκευσης (store) από την μνήμη και τις εντολές υπολογισμού της διεύθυνσης αποθήκευσης.

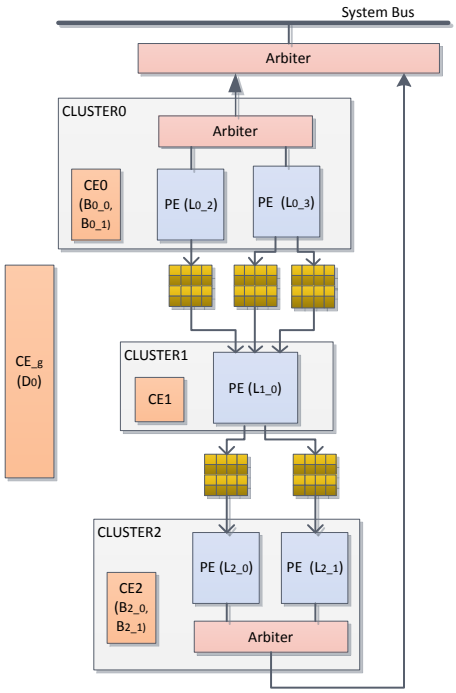
Ο λόγος που εκτελούμε αυτό το βήμα είναι επειδή κάθε ένα από τα 3 κομμάτια του κώδικα αντιστοιχεί και σε διαφορετικό κομμάτι της αρχιτεκτονικής: το υπολογιστικό κομμάτι αντιστοιχεί στην μονάδα επεξεργασίας δεδομένων, ενώ τα κομμάτια για load και store αντιστοιχούν στην μονάδα Εισόδου/Εξόδου Ροής. Με το να ξεχωρίσουμε τον κώδικα σε 3 διακριτά μέρη, μπορούμε να διαμορφώσουμε κάθε ένα τμήμα του επιταχυντή ανεξάρτητα χρησιμοποιώντας πληροφορία από το αντίστοιχο τμήμα του κώδικα.

c) Swing Modulo Scheduling (SMS). Η τεχνική SMS είναι μια τεχνική χρονοπρογραμματισμού εντολών (instruction scheduling) η οποία εκμεταλλεύεται τον παραλληλισμό μεταξύ εντολών που ανήκουν σε διαφορετικές επαναλήψεις ενός loop και τις προγραμματίζει να εκτελεσθούν παράλληλα [SMS]. Ο αλγόριθμος SMS χρησιμοποιεί ευριστικές μεθόδους για να ελαχιστοποιήσει το Iteration Interval (II), που είναι το σταθερό διάστημα αριθμού κύκλων μηχανής στο οποίο ξεκινάμε διαδοχικές επαναλήψεις του loop. Το SMS εφαρμόζεται σε κάθε loop και χωριστά σε κάθε ένα από τα τρία τμήματα του κώδικα όπως αυτά έχουν παραχθεί από το code slicing.

d) Δημιουργία υλικού. Το τελικό βήμα της δεύτερης φάσης του SOpenCL είναι η δημιουργία του επιταχυντή υλικού σε Verilog χρησιμοποιώντας την έξοδο που δημιουργεί το SMS και το αρχιτεκτονικό πρότυπο που περιεγράφηκε προηγουμένως. Εκτός από το υλικό του επιταχυντή, το SOpenCL δημιουργεί ακόμα και ένα αρχείο ελέγχου (testbench) το οποίο χρησιμοποιείται για την αυτοματοποίηση της διαδικασίας ελέγχου του παραχθέντος κυκλώματος.

Η Εικόνα 5 δείχνει το διάγραμμα του επιταχυντή που υλοποιεί τον αλγόριθμο για LU Decomposition. Ο συγκεκριμένος αλγόριθμος αποτελείται από πολλαπλά loops τα οποία δημιουργούν και παραπάνω του ενός επιταχυντές.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης



Εικόνα 13. Το αρχιτεκτονικό διάγραμμα ενός επιταχυντή που υλοποιεί τμήμα ενός αλγορίθμου LU Decomposition.

5.3. Instruction clustering

Μία επιπλέον βελτιστοποίηση του SOpenCL είναι η ομαδοποίηση (clustering) εντολών για την δημιουργία Macro Functional Units (MFUs), δηλ. λειτουργικών μονάδες που συνδυάζουν παραπάνω από μια απλούστερες λειτουργικότητες [6]. Ο λόγος που δημιουργούμε τέτοια MFUs είναι για να μειώσουμε τον αριθμό των διασυνδέσεων μεταξύ μικρών λειτουργικών μονάδων. Με άλλα λόγια, θέλουμε να μειώσουμε την επιβάρυνση των διασυνδέσεων σε σχέση με τις λειτουργικές μονάδες. Είναι γνωστό μάλιστα ότι τις περισσότερες φορές, ή έλλειψη επαρκούς αριθμού καναλιών διασύνδεσης είναι και ο κυριότερος λόγος για το ότι ένα κύκλωμα δεν μπορεί να υλοποιηθεί σε μια FPGA.

Χρησιμοποιούμε μία μέθοδο που βασίζεται σε grammar-induction τεχνικές για να ομαδοποιήσουμε βασικές εντολές σε μακρο-εντολές (macroinstructions) οι οποίες με την σειρά τους υλοποιούνται σαν MFUs με μικρότερες απαιτήσεις σε πολυπλοκότητα διασυνδέσεων.

Η Εικόνα 14 δείχνει ένα παράδειγμα για το πώς δύο διαδοχικές προσθέσεις (μέσα στα κόκκινα κουτιά) μπορούν να συγχωνευθούν για εκτέλεση σε ένα MFU αντί να εκτελεσθούν σε ένα FU που κάνει πρόσθεση. Η βασική ιδέα είναι ότι μία μεγαλύτερη λειτουργική μονάδα μειώνει την ανάγκη για πολλές διασυνδέσεις μεταξύ μικρότερων μονάδων και συνεπώς αυξάνει την πιθανότητα να έχουμε ένα κύκλωμα το οποίο να μπορεί να πραγματοποιηθεί σε μία FPGA.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Εικόνα 14. Χρονοδρομολόγηση εντολών σε ένα Data Flow Graph (DFG) με a) βασικές και απλές εντολές και b) με μακροεντολές. Η μακροεντολή K υλοποιείται με το MFU K που είναι ένας 3-bit προσθετής με αρχιτεκτονική διοχέτευσης (pipelined).

5.1. Αποτελέσματα του SOpenCL στα μετροπρογράμματα OpenDwarfs

Το εργαλείο SOpenCL χρησιμοποιήθηκε για την υλοποίηση επιταχυντών υλικού για εφαρμογές που ανήκουν στα Dwarfs. Τα 13 Dwarfs είναι σύνολα εφαρμογών που κάθε ένα από αυτά τα σύνολα μπορούν να χαρακτηρισθούν από ένα κοινό πρότυπο (pattern) υπολογισμών και επικοινωνίας. Για παράδειγμα, ένα από τα 13 Dwarfs είναι το Dense Linear Algebra το οποίο περιλαμβάνει αλγορίθμους που επεξεργάζονται πυκνούς πίνακες (dense arrays). Ένας τέτοιος αλγόριθμος που έχουμε χρησιμοποιήσει για να αναπτύξουμε επιταχυντές υλικού είναι ο LU Decomposition [14]. Για να είμαστε πιο ακριβείς, χρησιμοποιούμε ένα σύνολο από μετροπρογράμματα, τα OpenDwarfs, που είναι open-source υλοποιήσεις σε OpenCL των Dwarfs.

Χωρίς να μπούμε σε λεπτομέρειες σχετικά με την υλοποίηση των OpenDwarfs με την χρήση του SOpenCL, τα αποτελέσματα δείχνουν ότι οι FPGAs είναι ανταγωνιστικές με τις GPUs όσον αφορά την ταχύτητα εκτέλεσης τέτοιων αλγορίθμων. Εάν λάβουμε υπόψιν μας και την κατανάλωση ενέργειας, στις οποίες οι FPGA υπερτερούν εμφανώς των GPUs, μπορούμε να βγάλουμε το συμπέρασμα ότι η χρήση εργαλείων σαν το SOpenCL έχουν την δυνατότητα να κάνουν τις FPGA πολύ ελκυστικές για την υλοποίηση παρόμοιων εφαρμογών. Περισσότερες πληροφορίες σχετικά με το SOpenCL και τις FPGAs μπορείτε να βρείτε στο [14].

6. Υλοποίηση PDEs με τεχνικές Monte Carlo σε τεχνολογία FPGA

Οι πρόσφατες εξελίξεις στην τεχνολογία των FPGA και στις μεθοδολογίες σύνθεσης υψηλού επιπέδου (High Level Synthesis - HLS) έχουν θέσει τα επαναδιατασσόμενη συστήματα στο δρόμο προς την επιστήμη ετερογενών συστημάτων υψηλών επιδόσεων (High Performance Computing - HPC). Οι FPGA επιταχυντές προσφέρουν καλύτερα χαρακτηριστικά στις επιδόσεις, στην ενέργεια και στο κόστος από ότι μία ομογενή CPU πλατφόρμα και είναι πιο αποδοτικοί ενεργειακά από ότι μια GPU πλατφόρμα.

Ωστόσο, το μεγαλύτερο εμπόδιο στην υιοθέτηση της FPGA τεχνολογίας σε HPC πλατφόρμες είναι το ότι ο προγραμματισμός των FPGA απαιτεί βαθιά γνώση χαμηλού επιπέδου σχεδίασης υλικού και μακρόχρονους στάδια ανάπτυξης. Αυτά τα

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

χαρακτηριστικά καθιστούν τις γλώσσες περιγραφής υλικού (Hardware Description Languages - HDLs) μία ακατάλληλη τεχνολογία για την δημιουργία HPC εφαρμογών στις FPGA.

Αντιθέτως, εργαλεία HLS επιτρέπουν στους σχεδιαστές να χρησιμοποιήσουν υψηλού επιπέδου γλώσσες προγραμματισμού και προγραμματιστικά μοντέλα όπως οι C/C++ και OpenCL [2][4]. Ανεβάζοντας τη διαδικασία σχεδίασης υλικού στο επίπεδο ανάπτυξης λογισμικού, η HLS επιτρέπει όχι μόνο την γρήγορη πρωτοτυπία, αλλά και την εξερεύνηση αρχιτεκτονικών. Η πλειοψηφία των εργαλείων HLS προσφέρει ένα σύνολο οδηγιών βελτιστοποίησης που στοχεύουν στην ενημέρωση του μηχανισμού HLS σύνθεσης για την επιθυμητή βελτιστοποίηση κομματιών του πηγαίου κώδικα. Ο μηχανισμός σύνθεσης δημιουργεί επιταχυντές υλικού στοχεύοντας στις επιδόσεις ή στην ελαχιστοποίηση του χώρου που καταλαμβάνει η σχεδίαση, λαμβάνοντας υπόψη τις παραπάνω οδηγίες.

Αυτή η προσέγγιση δεν εκμεταλλεύεται στο έπακρο τις δυνατότητες των μοντέρνων FPGA να ενσωματώνουν αρχιτεκτονικές που μπορεί να περιέχουν πολλαπλούς επιταχυντές υλικού, ιεραρχίες μνήμης, δομές διαύλων επικοινωνίας και διεπαφές. Αυτό που είναι πραγματικά απαραίτητο, είναι μία μεθοδολογία για τον αυτόματο εντοπισμό και αξιολόγηση αρχιτεκτονικών πολλαπλών επιταχυντών, που υλοποιούν σύνθετες εφαρμογές λογισμικού. Η πολυπλοκότητα μίας τέτοιας μεθοδολογίας έγκειται στο γεγονός ότι οι επαναδιατασσόμενες πλατφόρμες προσφέρουν πολλαπλούς βαθμούς σχεδιαστικής ελευθερίας και ενδεχομένως ένα τεράστιο αριθμό pareto-βέλτιστων υλοποιήσεων.

Στην εργασία αυτή εισάγουμε μία συστηματική αρχιτεκτονική αξιολόγηση της τοποθέτησης εφαρμογών σε FPGAs υψηλών επιδόσεων που λειτουργούν ως επιταχυντές σε ένα περιβάλλον Linux. Ο στόχος είναι να αξιοποιηθούν όλα τα ενδιαφέροντα, επιπέδου συστήματος, αρχιτεκτονικά σενάρια, ώστε να χτιστεί ένας μηχανισμός που θα μπορεί να επιλέξει αυτόματα την καταλληλότερη αρχιτεκτονική για κάθε εφαρμογή. Βασιζόμενοι σε πειραματικές αναλύσεις, θέλουμε να λάβουμε γνώση για τις βέλτιστη αντιστοίχιση εφαρμογών – αρχιτεκτονικών, ώστε αργότερα να αυτοματοποιήσουμε αυτή τη διαδικασία. Συνεπώς, συγκρίνουμε τις επιδόσεις όλων των δυνατών λύσεων σε hardware με την επίδοση ενός software κώδικα που εκτελείται σε CPU υψηλών επιδόσεων (τη μονάδα υποδοχής – Host Unit).

Ποικίλες ερευνητικές προσπάθειες μελέτησαν το πρόβλημα δημιουργίας μίας Host-FPGA διεπαφής και on-chip καναλιών επικοινωνίας. Ο Vipin et al [14], ανέπτυξε ένα πλαίσιο λειτουργίας για την χρήση PCIe επικοινωνίας μεταξύ CPU υποδοχής και FPGA. Επίσης, χρησιμοποίησε πρότυπες HLS διεπαφές διαύλου για τις on-chip επικοινωνίες. Ωστόσο, παρέλειψε την παραμετροποίηση της λειτουργίας των διαύλων, καθώς και τις διαφορετικές ιεραρχίες μνήμης. Άλλες πρόσφατες έρευνες εξέτασαν την δημιουργία καναλιών ανεπηρέαστων από την καθυστέρηση δεδομένων [15], και μνημών κοινών μεταξύ πολλαπλών FPGA επιταχυντών. Σε αυτή την έρευνα επιδιώκουμε βαθύτερα επίπεδα παραμετροποίησης του συστήματος.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

```
blur_hor:
for (i = 0; i < Height; i++)
 for (j = 0; j < Width; j++)
 tmp(i,j)=(inp(i,j-1)+inp(i,j)+inp(i,j+1)/3

blur_ver:
for (i = 0; i < Height; i++)
 for (j = 0; j < Width; j++)
 out(i,j)=(tmp(i-1,j)+tmp(i,j)+tmp(i+1,j)/3
```

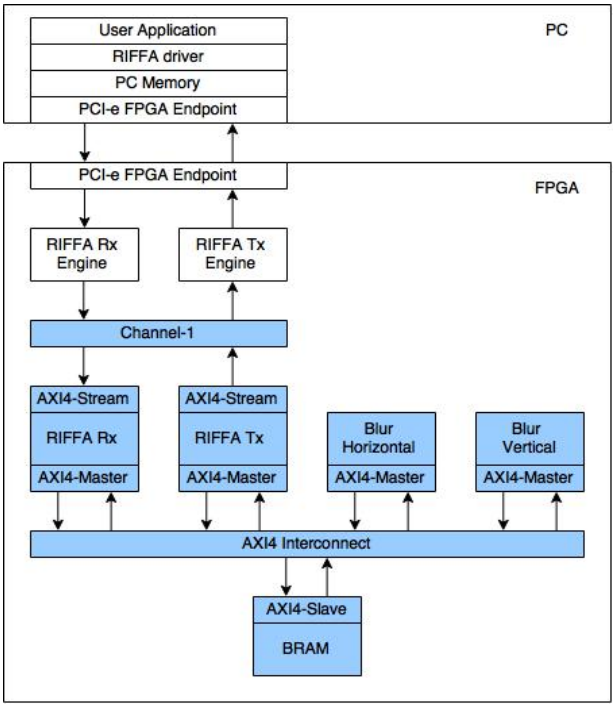
Εικόνα 15. Ο κώδικας για το οριζόντιο και το κάθετο φίλτρο Blur.

6.1. Εξερεύνηση του Χώρου Λύσεων της Σχεδίασης

Η καταχώριση 1 δείχνει τον ψευδοκώδικα του Blur φίλτρου, μία από τις εφαρμογές υπό εξέταση. Ο αλγόριθμος πρώτα εφαρμόζει ένα οριζόντιο και μετά ένα κάθετο 3-όρων βαθυπερατό φίλτρο μίας εικόνας εισόδου, αποθηκεύοντας προσωρινά το αποτέλεσμα του οριζόντιου φίλτρου στη μνήμη. Αυτός ο ψευδοκώδικας είναι βέλτιστος για μία CPU εκτέλεση, όχι για ένα σχεδιασμό στο υλικό, κάτι που επιφέρει μειονεκτήματα όταν επεξεργάζεται από HLS εργαλεία. Αυτός ο κώδικας καταλήγει σε δύο ξεχωριστούς hardware επιταχυντές, που πρέπει να επικοινωνούν μέσω μίας μεγάλης μνήμης που υλοποιείται είτε ως εξωτερική DRAM είτε ως πάνω στο chip BRAM (αν υπάρχει αρκετή BRAM στην FPGA). Με σκοπό να ενσωματωθούν οι FPGAs ως μέρος των μοντέρνων ετερογενών συστημάτων, εκμεταλλευτήκαμε τους τυποποιημένους αφαιρετικούς τρόπους επικοινωνίας που παρέχονται από υψηλού επιπέδου εργαλεία σύνθεσης όπως το Vivado HLS, ώστε να δημιουργήσουμε το προσαρμοζόμενο σύστημά μας.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Η λογική της Εικόνα 15



παράγεται από το εργαλείο Vivado HLS, με βάση τις οδηγίες του προγραμματιστή του συστήματος. Δύο επιταχυντές υλοποιούνται και συνδέονται με την BRAM μνήμη μέσω μιας AXI4-master διασύνδεσης. Η αρχική αρχιτεκτονική επεκτείνεται εξερευνώντας αυτόματα τον αριθμό και τον τύπο των διαφόρων στοιχείων. Τέτοια στοιχεία αποτελούν οι επιταχυντές από άποψη απόδοσης, μεγέθους, καθυστέρησης και αριθμού. Επίσης, περιλαμβάνονται οι δομές και ο αριθμός των διαύλων επικοινωνίας, ο αριθμός και ο τύπος των μνημών, καθώς και η διεπαφή με τη μονάδα υποδοχής.

Επιπροσθέτως του προσαρμοζόμενου μέρους της αρχιτεκτονικής, έξτρα στοιχεία επικοινωνίας με τη μονάδα υποδοχής είναι απαραίτητα. Προς διευκόλυνση μας χρησιμοποιούμε ένα πλαίσιο επικοινωνίας ανοιχτού κώδικα, το RIFFA, που προσφέρει αφαιρετικά τη δυνατότητα στους προγραμματιστές λογισμικού να έχουν πρόσβαση στην FPGA ως μία μονάδα επιτάχυνσης συνδεδεμένη στο PCIe [20].

Το RIFFA υλοποιεί στο υλικό το πρωτόκολλο του PCIe άκρου, ώστε ο χρήστης να μην χρειάζεται να ασχοληθεί με τις λεπτομέρειες επικοινωνίας του επιταχυντή. Από την μεριά του τελευταίου συγκεκριμένα, το RIFFA προσφέρει ένα σύνολο καναλιών επικοινωνίας συνεχούς ροής που στέλνουν και λαμβάνουν δεδομένα μεταξύ της κύριας μνήμης της CPU και μίας προσαρμοσμένης FPGA λογικής. Στη μονάδα υποδοχής, η αρχιτεκτονική του RIFFA 2.0 είναι συνδυασμός ενός πυρήνα οδήγησης συσκευής και ενός συνόλου γλωσσικών δεσμεύσεων. Το RIFFA παρέχει στο χρήστη μία απλοϊκή διεπαφή προγραμματισμού εφαρμογών (API), που του επιτρέπει ξεχωριστά την πρόσβαση στα κανάλια για την μεταφορά δεδομένων στη λογική του επιταχυντή.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Εικόνα 16. Βασική αρχιτεκτονική πλατφόρμα για πειραματική αξιολόγηση. Η σκιαγραφημένη περιοχή δείχνει την παραμετροποιήσιμη λογική.

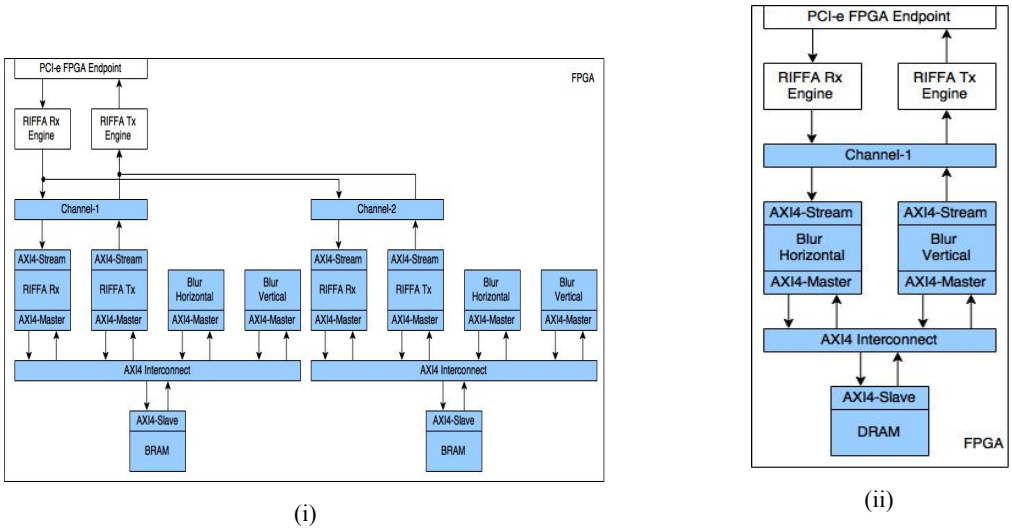
Στην Εικόνα 17 παρουσιάζονται δύο ενδεικτικά αρχιτεκτονικά σενάρια, που επεκτείνουν την αρχιτεκτονική αναφορά της Εικόνα 16. Μπορούμε αποτελεσματικά να διπλασιάσουμε την προσαρμόσιμη λογική χρησιμοποιώντας ένα επιπλέον RIFFA κανάλι επικοινωνίας (Εικόνα 17i). Ακόμα καλύτερα, η φύση της Blur εφαρμογής ως ένα συνεχής ροής μονοπάτι δεδομένων από σημείο σε σημείο, μας επιτρέπει να χρησιμοποιήσουμε το πρωτόκολλο AXI4-Stream για να μεταδώσουμε τα δεδομένα μεταξύ παραγωγού και καταναλωτή (Εικόνα 17ii). Κάποιες υλοποιήσεις μπορεί να κάνουν χρήση εξωτερικής DDR3 μνήμης (περιλαμβάνοντας έναν FPGA DDR3 ελεγκτή μνήμης), ώστε να μπορούν να φιλοξενήσουν μεγαλύτερες εικόνες με κόστος την αύξηση καθυστέρησης και χειρότερες επιδόσεις. Για να περιηγηθούμε έξυπνα στον απέραντο χώρο όλων των δυνατών σχεδιαστικών λύσεων, επινοήσαμε ένα σύνολο ευριστικών κανόνων για τη λήψη αποφάσεων:

1. Διατήρησε τα δεδομένα τοπικά και όσο το δυνατόν πιο κοντά στον επιταχυντή. Ο στόχος εδώ είναι να ελαχιστοποιηθεί η καθυστέρηση των αναγνώσεων/εγγραφών μεταξύ επιταχυντή και μνήμης δεδομένων.
2. Ελαχιστοποίησε τους κοινούς πόρους μεταξύ ανεξάρτητων επιταχυντών. Αυτή η οδηγία βοηθά στην εξάλειψη των συγκρούσεων ανάμεσα σε πολλαπλούς επιταχυντές κατά την απόκτηση πρόσβασης σε κανάλια επικοινωνίας ή θύρες προσπέλασης μνήμης.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

3. Επικάλυψε τις μεταφορές δεδομένων με το χρόνο λειτουργίας των επιταχυντών. Ο στόχος εδώ είναι να ελαχιστοποιηθεί ο χρόνος αναμονής για τη διαθεσιμότητα των δεδομένων στους επιταχυντές.

Η προσέγγιση του χώρου λύσεων της σχεδίασης ξεκινά από την βασική αρχιτεκτονική αναφοράς της Εικόνα 16. Στη συνέχεια, παίρνουμε σταδιακά κάποιες σχεδιαστικές επιλογές λαμβάνοντας υπόψη τους προαναφερθείς ευριστικούς κανόνες και αξιολογώντας την επίδραση των αποφάσεων στην επίδοση του συνολικού συστήματος.



Εικόνα 17. Δύο ενδιαφέροντα αρχιτεκτονικά σενάρια. (i) Διπλασιασμός της βασικής αρχιτεκτονικής χρησιμοποιώντας δύο RIFFA κανάλια. (ii) Χρήση AXI streaming διεπαφής μεταξύ των RIFFA καναλιών, των δύο επιταχυντών και της DRAM.

6.2. Αριθμητικά αποτελέσματα - Πειράματα

6.2.1. Μεθοδολογία

Σε αυτό το τμήμα, θα παρουσιάσουμε την έρευνα μας πάνω στην αρχιτεκτονική εξερεύνηση δύο εφαρμογών που φαίνονται στον Πίνακα 1. Για κάθε εφαρμογή, έχουμε ορίσει ένα πλήθος αρχιτεκτονικών σεναρίων που εκτείνεται σε ποικίλα σημεία του χώρου «μέγεθος εναντίον επίδοση». Ο software κώδικας βελτιστοποιείται με τη χρήση HLS οδηγιών (π.χ. διοχέτευση) με ελάχιστες αλλαγές του. Χρησιμοποιούμε το εργαλείο *Vivado HLS 2014.4* για τις hardware υλοποιήσεις μας πάνω σε μία πλακέτα *VC707* evaluation board *(XC7VX485T* FPGA). Όλα τα αποτελέσματα καταγράφονται μετά τη διαδικασία τοποθέτησης και διασύνδεσης των

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

θεμελιωδών στοιχείων του σχεδίου στην FPGA. Ο ίδιος κώδικας σε γλώσσα C εκτελείται σε έναν τετραπύρηνο επεξεργαστή *E5520 Intel Xeon* σε συχνότητα λειτουργίας *2.27 GHz* και όλα τα αποτελέσματα των επιδόσεων συγκρίνονται. Πέραν του Blur φίλτρου που περιγράψαμε παραπάνω, έχουμε αναλύσει και μία εφαρμογή Monte Carlo προσομοίωσης.

Πίνακας 1. Εφαρμογές που χρησιμοποιήθηκαν για εξερεύνηση του χώρου αρχιτεκτονικών λύσεων

App.	Description	Input Set
Monte Carlo	Monte Carlo simulations in a 2D space	120 Points, 5000 Walks per point
Blur	Blur 2D filter	4096×4096 image

Η μέθοδος Monte Carlo (MC) αναπτύχθηκε ως ένας εναλλακτικός τρόπος προσέγγισης πολυδιάστατων επιστημονικών προβλημάτων. Διασπά ένα μοναδικό πρόβλημα Μερικών Διαφορικών Εξισώσεων (Partial Differential Equation - PDE) σε ένα σύνολο ενδιάμεσων προβλημάτων. Ο κύριος πυρήνας της εφαρμογής εκτελεί τυχαίες διαδρομές από κάποια αρχικά σημεία ενός 2D πλέγματος, ώστε να εκτιμήσει τις οριακές περιπτώσεις των ενδιάμεσων προβλημάτων. Ο MC είναι ένας υπολογιστικά βαρύς αλγόριθμος με πολλές αριθμητικές πράξεις διπλής ακρίβειας και κλήσεις σε μαθηματικές βιβλιοθήκες για τους αριθμητικούς, τριγωνομετρικούς και λογαριθμικούς υπολογισμούς αριθμών κινητής υποδιαστολής (Floating Point - FP). Ωστόσο, έχει ελάχιστες απαιτήσεις επικοινωνίας.

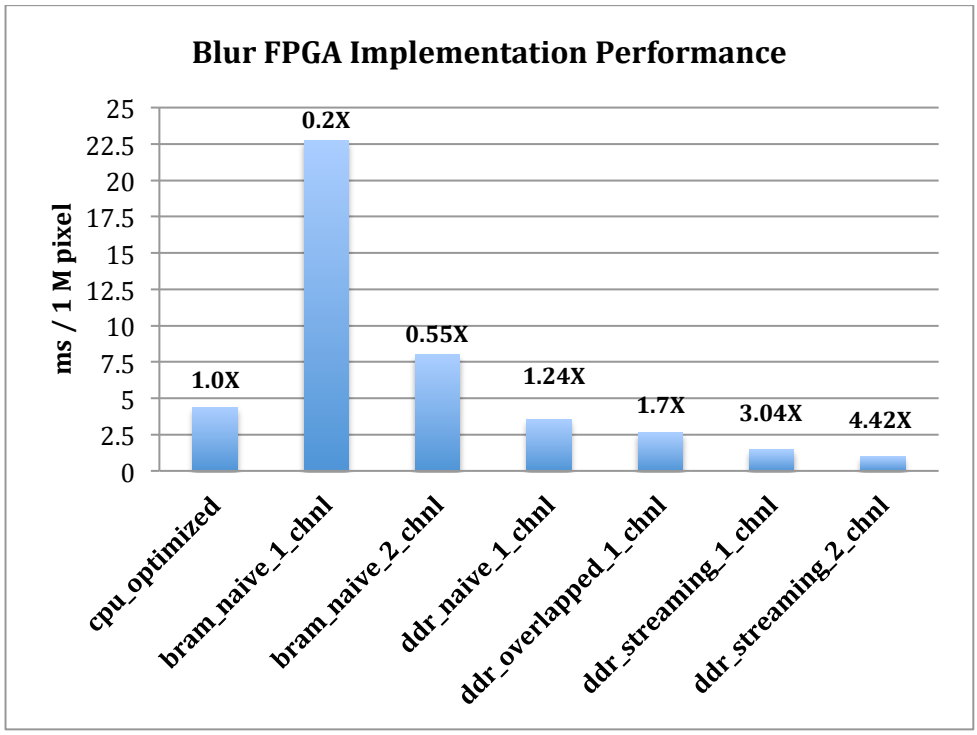
6.2.2. Αποτελέσματα

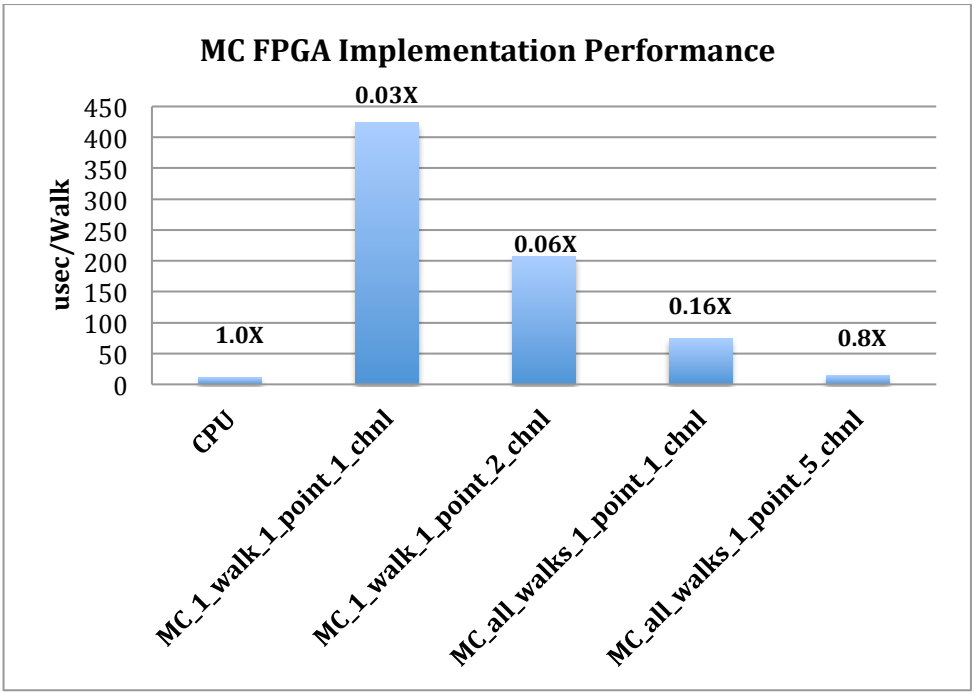
Η Εικόνα 18 παρουσιάζει τα αποτελέσματα επίδοσης των δύο περιπτώσεων υπό εξέταση σε διάφορα σενάρια υλοποίησης. Ο Πίνακας 2 δείχνει τις απαιτήσεις σε χώρο στην FPGA της κάθε πλατφόρμας.

Blur. Έξι σενάρια υλοποίησης μελετήθηκαν για την εφαρμογή Blur. Το πρώτο σενάριο απεικονίζει την βασική αρχιτεκτονική αναφοράς της Εικόνα 16 (*bram_naive_1_chnl*). Η CPU υποδοχής στέλνει μία μοναδική γραμμή στον πυρήνα του οριζόντιου φίλτρου blur για επεξεργασία και περιμένει την έξοδο του επιταχυντή πριν στείλει την επόμενη γραμμή, έως ότου προσπελαστεί ολόκληρη η εικόνα. Η ίδια διαδικασία ακολουθείται και για το κάθετο blur φίλτρο, με τη διαφορά ότι χρειάζονται 3 γραμμές πριν ξεκινήσει η επεξεργασία. Το σενάριο αυτό έχει εφαρμογή και είναι επιθυμητό όταν δεν υπάρχει αρκετή εσωτερική ή εξωτερική του chip μνήμη για να αποθηκευτεί ολόκληρη η εικόνα, με αποτέλεσμα να τη διαμερίζουμε σε μικρότερα κομμάτια ώστε να χωρέσει στους διαθέσιμους πόρους μνήμης. Μία άλλη εκδοχή αυτού του σεναρίου (*bram_naive_2_chnl*) αναπαράγει το

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

υλικό του ενός RIFFA καναλιού σε 2 κανάλια, για να εκμεταλλευτεί τον παραλληλισμό της Blur εφαρμογής. Παρόλο που το δεύτερο σενάριο βελτιώνει την επίδοση της απλοϊκής υλοποίησης με BRAM, παραμένει χειρότερη από τη βελτιστοποιημένη υλοποίηση σε CPU. Η αποστολή μικρών πακέτων δεδομένων μέσω του PCIe δεν είναι αποδοτική, καθώς πληρώνουμε συχνά το κόστος εκκίνησης των μεταφορών ανάγνωσης και εγγραφής. Ως αποτέλεσμα, οι μεταφορές πάνω στο PCIe καταλαμβάνουν τα 2/3 του συνολικού χρόνου εκτέλεσης.





Εικόνα 18. Πειραματικά αποτελέσματα επίδοσης των εφαρμογών Blur και Monte-Carlo. Οι αριθμοί πάνω από τις στήλες υποδεικνύουν την σύγκριση σε σχέση με την βέλτιστη CPU υλοποίηση.

Το τρίτο σενάριο του Blur (*ddr_naive_1_chnl*) χρησιμοποιεί την off-chip DDR μνήμη για να αποθηκεύσει το σύνολο της εικόνας αντί να την διαμερίζει σε πολλαπλά τμήματα. Η χρήση της DDR προσφέρει κάποια οφέλη· Οι PCIe μεταφορές ανάγνωσης/εγγραφής καταναλώνουν λιγότερο χρόνο σε σύγκριση με τα πρώτα σενάρια, καθώς εξαλείφουμε το κόστος εκκίνησής τους. Το δεύτερο προνόμιο της DDR είναι ότι δεν χρειάζεται να στείλουμε την ενδιάμεση έξοδο, του οριζόντιου φίλτρου, πίσω στη CPU, αλλά την κρατάμε στη DDR για να την επεξεργαστεί το κάθετο blur φίλτρο και στη συνέχεια να γράψουμε το αποτέλεσμα πίσω στη CPU υποδοχής. Με αυτό τον τρόπο, το τρίτο σενάριο πετυχαίνει μία βελτίωση ως προς το βέλτιστο χρόνο της CPU.

Για να βελτιώσουμε ακόμα περισσότερο την επίδοση του σεναρίου #3, επιτρέπουμε το οριζόντιο blur να ξεκινήσει αμέσως μόλις η πρώτη γραμμή της εικόνας αποθηκευτεί στη DDR αντί να περιμένουμε τη φόρτωση ολόκληρης της εικόνας. Επίσης, επιτρέπουμε την εγγραφή των δεδομένων πίσω στην CPU πριν ο επιταχυντής του κάθετου blur ολοκληρώσει την εκτέλεση του. Τα παραπάνω παρουσιάζονται στο τέταρτο σενάριο (*ddr_overlapped_1_chnl*). Η επικάλυψη της εκτέλεσης των επιταχυντών με τις μεταφορές δεδομένων μεταξύ FPGA και Host είναι δυνατή λόγο του σταθερού τρόπου προσπέλασης δεδομένων που παρουσιάζουν οι πυρήνες του blur. Παρότι αυτή η υλοποίηση εξαλείφει το κόστος των

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

περισσότερων μεταφορών, η μετακίνηση δεδομένων μεταξύ DDR και επιταχυντών εισάγει ένα μη αμελητέο κόστος.

Πίνακας 3. Αποτελέσματα χωρικών απαιτήσεων για το σύνολο των διαφορετικών υλοποιήσεων με βάση τη χρήση των υλικών πόρων της XC7VX485T FPGA

Benchmark	LUT	FF	BRAM	DSP48
BLUR				
bram_naive_1_chnl	15%	10%	5%	11%
bram_naive_2_chnl	28%	20%	10%	23%
ddr_naive_1_chnl	20%	15%	6%	5%
ddr_overlapped_1_chnl	20%	15%	6%	5%
ddr_streaming_1_chnl	13%	9%	4%	3%
ddr_streaming_2_chnl	21%	14%	6%	6%
Monte-Carlo (MC)				
MC_1_walk_1_point_1_chnl	8%	5%	2%	5%
MC_1_walk_1_point_2_chnl	13%	9%	3%	9%
MC_all_walks_1_point_1_chnl	9%	7%	2%	5%
MC_all_walks_1_point_2_chnl	16%	12%	3%	9%

Με σκοπό την περαιτέρω βελτίωση του σχεδίου και την ελαχιστοποίηση του κόστους επικοινωνίας DDR και επιταχυντή, χρησιμοποιούμε συνεχούς ροής AXI-stream διεπαφές για την ανάγνωση δεδομένων από τις FIFOs των καναλιών και για την εγγραφή του αποτελέσματος στην DDR. Το κάθετο blur φίλτρο θα διαβάσει δεδομένα από την DDR, θα τα επεξεργαστεί και θα στείλει την έξοδο κατευθείαν στο κανάλι του RIFFA μέσω μίας συνεχούς ροής AXI-stream διεπαφής. Σε αυτό το σενάριο, εξαλείφουμε το 60% των μετακινήσεων δεδομένων μεταξύ DDR και επιταχυντή. Αυτό είναι εφικτό λόγω της streaming φύσης των πυρήνων του blur. Αυτή η αρχιτεκτονική πετυχαίνει την καλύτερη επίδοση σε σύγκριση με την CPU υλοποίηση. Επιπλέον, καταλαμβάνει μικρότερη περιοχή (βλ. Πίνακα 3) από την απλοϊκή DDR και τις επικαλυπτόμενες υλοποιήσεις, κάτι που επιτρέπει την αναπαραγωγή περισσότερων αντιγράφων του επιταχυντή, για την εκμετάλλευση του παραλληλισμού και την βελτιστοποίηση των επιδόσεων όπως στην περίπτωση του *ddr_streaming_2_chnl*.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Monte-Carlo προσομοίωση. Ο πυρήνας του MC αλγορίθμου είναι υπολογιστικά απιτητικός και με ελάχιστες προσπελάσεις μνήμης. Για αυτό το λόγο, τα διάφορα σενάρια υλοποίησης προκύπτουν από ποικίλες παραμετροποιήσεις των ίδιων των επιταχυντών και από πολλαπλές αναπαραγωγές της εμφάνισης τους. Στο βασικό αρχιτεκτονικό σενάριο, ο επιταχυντής είναι κατασκευασμένος να πραγματοποιεί ένα μοναδικό μονοπάτι από ένα σημείο του πλέγματος σε κάθε χρήση του και μόνο ένας μοναδικός επιταχυντής έχει υλοποιηθεί στην FPGA (*MC_1_walk_1_point_1_chnl*). Αυτό το σενάριο αποδίδει πολύ χειρότερα από την CPU υλοποίηση. Οι πράξεις διπλής ακρίβειας έχουν μεγαλύτερη καθυστέρηση στις FPGA από ότι σε μία CPU. Επίσης, οι βιβλιοθήκες του Vivado HLS για τριγωνομετρικές πράξεις (sin, cos) δεν έχουν τη δυνατότητα διοχέτευσης και είναι λιγότερο αποδοτικές από τις αντίστοιχες της CPU. Η υπεροχή των FPGA βασίζεται στην εκτέλεση πολλών παράλληλων υπολογισμών. Δυστυχώς, οι υπολογισμοί του πυρήνα του MC, πάνω σε ένα μοναδικό μονοπάτι είναι ακολουθιακοί και δεν μπορούν να παραλληλοποιηθούν. Για αυτό το λόγο το βασικό αρχιτεκτονικό σενάριο αποδίδει ανεπαρκώς πάνω στην FPGA. Για να βελτιώσουμε την απόδοση πρέπει να εκμεταλλευτούμε υψηλότερου επιπέδου παραλληλισμό ανάμεσα στα πολλαπλά σημεία και μονοπάτια που εκτελούνται. Το δεύτερο σενάριο περιλαμβάνει δύο εμφανίσεις του επιταχυντή, ώστε να παραλληλοποιήσει τους υπολογισμούς των διαδρομών από ένα μοναδικό μονοπάτι (*MC_1_walk_1_point_2_chnl*). Αποτέλεσμα αυτού είναι να μειωθεί ο χρόνος εκτέλεσης στο μισό, παραμένοντας όμως χειρότερος από τη CPU. Για να πετύχουμε παρόμοιες επιδόσεις με τη CPU πρέπει να αναπαράγουμε γύρω στα 40 αντίγραφα αυτού του επιταχυντή.

Ένα άλλο χαρακτηριστικό προς βελτιστοποίηση είναι η ελαχιστοποίηση του χρονικού κόστους λόγο κλήσεων/χρήσεων του επιταχυντή, αυξάνοντας το επίπεδο υπολογισμών για κάθε μοναδικό επιταχυντή. Αυτό επιτρέπει την διοχέτευση υπολογισμών για πολλαπλά μονοπάτια, κάτι που θα έχει ισχυρή επίδραση στις επιδόσεις. Το τρίτο σενάριο λοιπόν, ελαχιστοποιεί τις κλήσεις του επιταχυντή και το κόστος τους, ρυθμίζοντας τον επιταχυντή να πραγματοποιεί όλα τα μονοπάτια από ένα σημείο ανά κλήση (*MC_all_walks_1_point_1_chnl*). Το τέταρτο σενάριο αναπαράγει πέντε επιταχυντές για να παραλληλοποιήσει τους υπολογισμούς πολλαπλών σημείων (*MC_all_walks_1_point_5_chnl*). Η τελευταία υλοποίηση επιφέρει κορεσμό των πόρων της FPGA και σχεδόν πετυχαίνει την επίδοση της CPU.

7. Βιβλιογραφία

[1] Katherine Compton and Scott Hauck. Reconfigurable computing: a survey of systems and software. *ACM Computing Surveys.* Vol. 34, no. 2. June 2002.

[2] L.H. Crockett, R.A. Elliot. The Zynq Book: Embedded Processing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable SoC. 2014. Strathclyde Academic Media. *Available at www.zynqbook.com*.

[3] Muhsen Owaida, Nikolaos Bellas, Christos Antonopoulos, Konstantis Daloukas, Charalambos Antoniadis. Massively Parallel Programing Models Used as Hardware Description Languages: The OpenCL Case. *International Conference on Computer-Aided Design (ICCAD),* November 6-10, 2011, San Jose, CA

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

[4] David A. Patterson and John L. Hennessy, Computer Organization and Design - The Hardware / Software Interface, (Revised 4th Edition). *The Morgan Kaufmann Series in Computer Architecture and Design.* Academic Press, 2012

[5] V. Vassiliadis, C.D. Antonopoulos, G. Zindros. Automating data management in heterogeneous systems using polyhedral analysis. *Proceedings of the 19th Panhellenic Conference on Informatics (PCI 2015).* Pages 317-322. October 2015. Athens, Greece.

[6] S. Verdoolaege. ISL: An integer set library for the polyhedral model. *In Mathematical Software (ICMS 2010),* pages 299-302. Springer, 2010.

[7] S. Verdoolaege and T. Grosser. Polyhedral extraction tool. In Proceedings of Second International Workshop on Polyhedral Compilation Techniques (IMPACT'12), 2012.

[8] D. K. Wilde. A library for doing polyhedral operations. *Parallel Algorithms and Application*, 15(3-4):137-166, 2000.

[9] Khronos OpenCL Working Group. Editor: A. Munshi, "The OpenCL Specification", Version: 1.1 Document Revision: June 11, 2010.

[10] Konstantinos Krommydas, Wu-Chun Feng, Muhsen Owaida, Christos D. Antonopoulos and Nikolaos Bellas. On the Portability of OpenCL Dwarfs on Fixed and Reconfigurable Parallel Platforms. *25th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP).* June 18-20, 2014, Zurich, Switzerland.

[11] C. Lattner and V. Adve. "LLVM: A Compilation Framework for Lifelong Program Analysis Transformation, *"Proceedings of the International Symposium on Code Generation and Optimization (CGO'04)*, March 2004, pp. 75-86, Palo Alto, CA.

[12] J. Llosa, A. Gonzalez, E. Ayguade, and M. Valero. "Swing Modulo Scheduling: A Lifetime-Sensitive Approach, *"Proceedings of the 1996 Conference on Parallel Architectures and Compilation Techniques (PACT)*, October, 1996, pp. 80-86, Boston, MA.

[13] Muhsen Owaida, Nikolaos Bellas, Konstantis Daloukas, Christos D Antonopoulos. Synthesis of Platform Architectures from OpenCL Programs. *IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, May 1-3, 2011, Salt Lake City, UT

[14] Muhsen Owaida, Christos D. Antonopoulos, Nikolaos Bellas. A Grammar Induction Method for Clustering of Operations in Complex FPGA Designs. *IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM), Regular paper.* May 11-13, 2014. Boston, MA

[15] K. Vipin, S. Shreejith, D. Gunasekera, S. *A. Fahmy, N. Kapre*. System-Level FPGA Device Driver with High-Level Synthesis Support. *International Conference on Field Programmable Technology (FPT),* Kyoto, Japan, December 9-11, 2013.

[16] K. Fleming, H. J. Yang, M. Adler, J. Emer. The LEAP FPGA Operating System. *24th International Conference on Field Programmable Logic and Applications (FPL).* Munich, Germany, September 2-4, 2014.

[17] H. J. Yang, K. Fleming, M. Adler, J. Emer. LEAP Shared Memories: Automating the Construction of FPGA Coherent Memories. *22nd International Symposium on Field Programmable Custom Computing Machines (FCCM).* Boston, USA, May 11-13, 2014.

[18] Vivado Design Suite User Guide: High Level Synthesis. *Online at www.xilinx.com*.

[19] Altera OpenCL SDK Programming Guide. *Online at www.altera.com.*

[20] M. Jacobsen, R. Kastner. RIFFA 2.0: A reusable integration framework for FPGA accelerators. *23rd International Conference on Field programmable Logic and Applications (FPL).* Porto, Portugal, September 2-4, 2013.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης