ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ, ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

Υλοποίηση και Βελτιστοποίηση Αλγορίθμου Διόρθωσης Παραμόρφωσης Εικόνας από Ευρυγώνιους Φακούς στον Επεξεργαστή Cell

Implementation and Optimization of a Wide-angle Lens Distortion Correction Algorithm on the Cell Broadband Engine

Διπλωματική Εργασία

Κωνσταντής Α. Νταλούκας

Επιβλέποντες Καθηγητές : Νικόλαος Μπέλλας
Αναπληρωτής Καθηγητής

Χρήστος Δ. Αντωνόπουλος
Συμβασιούχος Διδάσκων ΠΔ 407/80

Βόλος, Ιούνιος 2009

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ,
ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

Υλοποίηση και Βελτιστοποίηση Αλγορίθμου Διόρθωσης Παραμόρφωσης Εικόνας από Ευρυγώνιους Φακούς στον Επεξεργαστή Cell

Διπλωματική Εργασία

Κωνσταντής Α. Νταλούκας

Επιβλέποντες : Νικόλαος Μπέλλας
Αναπληρωτής Καθηγητής

Χρήστος Δ. Αντωνόπουλος
Συμβασιούχος Διδάσκων ΠΔ 407/80

Εγκρίθηκε από την διμελή εξεταστική επιτροπή την 16η Ιουνίου 2009

.....	.....
Ν. Μπέλλας	Χ. Δ. Αντωνόπουλος
Αναπληρωτής Καθηγητής	Συμβασιούχος Διδάσκων
	ΠΔ 407/80

Διπλωματική Εργασία για την απόκτηση του Διπλώματος του Μηχανικού Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων του Πανεπιστημίου Θεσσαλίας, στα πλαίσια του Προγράμματος Προπτυχιακών Σπουδών του Τμήματος Μηχανικών Η/Υ, Τηλεπικοινωνιών και Δικτύων του Πανεπιστημίου Θεσσαλίας.

.....

Κωνσταντής Νταλούκας

Διπλωματούχος Μηχανικός Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων Πανεπιστημίου Θεσσαλίας

Στην οικογένειά μου και στους φίλους μου

Ευχαριστίες

Με την περάτωση της παρούσας εργασίας, θα ήθελα να ευχαριστήσω θερμά τους επιβλέποντες της διπλωματικής εργασίας κ. Νικόλαο Μπέλλα και κ. Χρήστο Αντωνόπουλο για την εμπιστοσύνη που επέδειξαν στο πρόσωπό μου, την άριστη συνεργασία, την συνεχή καθοδήγηση και τις ουσιώδεις υποδείξεις και παρεμβάσεις, που διευκόλυναν την εκπόνηση της πτυχιακής εργασίας.

Επίσης, θα ήθελα να ευχαριστήσω τους φίλους και συνεργάτες του Εργαστηρίου Ε5 για την υποστήριξη και την δημιουργία ενός ευχάριστου και δημιουργικού κλίματος.

Τέλος, οφείλω ένα μεγάλο ευχαριστώ στην οικογένειά μου και στους φίλους μου για την αμέριστη υποστήριξη και την ανεκτίμητη βοήθεια που μου παρείχαν τόσο κατά την διάρκεια των σπουδών μου όσο και κατά την εκπόνηση της διπλωματικής εργασίας.

Κωνσταντής Νταλούκας
Βόλος, 2009

Περιεχόμενα

Κατάλογος Πινάκων

Κατάλογος Σχημάτων

Κατάλογος Συντομογραφιών

API Application Programming Interface

ALF Accelerated Library Framework

ASIC Application Specific Instruction Set

BEI Broadband Engine Interface

BIC Broadband Interface Controller

BRU Branch Unit

CAD Computer Aided Design

CBE Cell Broadband Engine

CBEA Cell Broadband Engine Architecture

CESOF CBE Embedded SPE Object Format

CPI Cycles Per Instruction

CPU Central Processing Unit

DLP Data Level Parallelism

DMA Direct Memory Access

DSP Digital Signal Processing

EIB Element Interconnect Bus

ESL Electronic System Level

FoV Field of View

FPGA Field Programmable Gate Array

fps Frames per Second

FPU Floating Point Unit

FSB Front-Side Bus

FXU Fixed-Point Unit

GFlops Giga Floating-Point operations per second

HDL Hardware Description Languages

IIC Internal Interrupt Controller

ILP Instruction Level Parallelism

IOC I/O Controller

LS Local Store

LSU Load and Store Unit

IU Instruction Unit

MFC Memory Flow Controller

MIC Memory Interface Controller

MIMD Multiple Instruction Multiple Data

MMU Memory Management Unit

PPE Power Processing Element

PPSS PowerPC processor Storage Subsystem

PPU PowerPC Processing Unit

PSNR Peak Signal to Noise Ratio

RISC Reduced Instruction Set Computer

ROI Region Of Interest

sDFG Streaming Data Flow Graph

SDK Software Development Kit

SIMD Single Instruction Multiple Data

SMP Symmetric Multiprocessing

SMT Simultaneous Multithreading

SoC System on Chip

SPE Synergistic Processing Element

SPU Synergistic Processing Unit

SRAMs Static Random Access Memories

SSE Streaming SIMD Extensions

TLP Thread Level Parallelism

VLSI Very Large Scale Integration

VSU Vector/Scalar Unit

VXU Vector/SIMD Multimedia Extension Unit

XDR Rambus Extreme Data Rate

Περίληψη

Οι ευρυγώνιοι φακοί, *wide-angle* ή *fisheye lenses*, χρησιμοποιούνται σε επιστημονικές εφαρμογές ή εφαρμογές εικονικής πραγματικότητας (*virtual reality*) ώστε να επεκτείνουν το πεδίο θέασης, Field of View (FoV), των συμβατικών φωτογραφικών μηχανών. Λόγω του μεγαλύτερου πεδίου θέασης, οι εικόνες που λαμβάνονται από τέτοιου είδους φακούς εμφανίζουν κάποιου είδους παραμόρφωση. Η διόρθωση παραμόρφωσης που προκαλείται από την χρήση ευρυγώνιων φακών είναι μία εφαρμογή επεξεργασίας εικόνας (*image warping application*), η οποία χρησιμοποιείται ώστε να μετατρέψει την παραμορφωμένη εικόνα από το μοντέλο των ευρυγώνιων φακών στο μοντέλο της κεντρικής προοπτικής προβολής (*central perspective space*). Η εφαρμογή χαρακτηρίζεται από ένα μη γραμμικό, συνεχούς ροής, πρότυπο προσπέλασης της μνήμης το οποίο καθιστά το εύρος ζώνης της κύριας μνήμης έναν παράγοντα ο οποίος επηρεάζει σημαντικά την απόδοση της εφαρμογής.

Ο επεξεργαστής *Cell*, Cell Broadband Engine (CBE), είναι ένας μη συμβατικός και ετερογενής πολύ-επεξεργαστής. Η αρχιτεκτονική του επεξεργαστή *Cell* επιτρέπει την επιτάχυνση εφαρμογών με μεγάλο βαθμό παραλληλισμού τόσο σε επίπεδο νημάτων (*thread-level parallelism*) όσο και σε επίπεδο δεδομένων (*data-level parallelism*).

Η παρούσα διπλωματική εργασία παρουσιάζει την υλοποίηση, βελτιστοποίηση και αξιολόγηση ενός αλγορίθμου για την διόρθωση παραμόρφωσης εικόνας, που προκαλείται από την χρήση ευρυγώνιων φακών, στον επεξεργαστή *Cell* και σε πραγματικό χρόνο. Για την επίλυση των προβλημάτων του συστήματος μνήμης εφαρμόζουμε βελτιστοποιήσεις σε επίπεδο πηγαίου κώδικα όπως η τεχνική του *tiling* για την καλύτερη εκμετάλλευση της *on-chip* μνήμης των Synergistic Processing Elements (SPEs) και για την μεγιστοποίηση της επαναχρησιμοποίησης των δεδομένων εντός του *frame*. Οι προτεινόμενες βελτιστοποιήσεις καθιστούν την διόρθωση παραμόρφωσης εικόνας στον επεξεργαστή CBE μία διαδικασία που μπορεί να πραγματοποιηθεί σε πραγματικό χρόνο ενώ επιτυγχάνεται επιτάχυνση (*speedup*) ίση με $7.27x$, σε σύγκριση με έναν επεξεργαστή *Intel Core2 Duo*.

Λέξεις Κλειδιά:
Παραμόρφωση Εικόνας, Πραγματικός Χρόνος, Παράλληλος Προγραμματισμός,
Υπολογισμός Stencil, Cell B.E., Ετερογενείς Επεξεργαστές Πολλαπλών Πυρήνων

Abstract

Wide-angle lenses are often used in scientific or virtual reality applications to enlarge the field of view of a conventional camera. Wide-angle lens distortion correction is an *image warping application* which transforms the distorted images back to the natural-looking, *central perspective space*. This application is characterized by a non-linear streaming memory access pattern that makes main memory bandwidth a key performance limiter.

Cell is a heterogeneous multi-core processor that has been architected to accelerate applications with large degree of thread- and data-level parallelism.

This thesis presents the implementation, optimization and evaluation of a wide-angle lens distortion correction algorithm on the Cell Broadband Engine. We alleviate the 'memory wall' inefficiencies by applying source level optimizations such as *tiling* to better utilize the on-chip memory of the SPE, and maximize data reuse within a frame of pixel data. Using these transformations on the Cell processor, we are able to achieve a $7.27x$ speedup compared to a *Intel Core2 Duo* processor, and enable potential applications such as real-time correction for video teleconferencing using cameras with wide-angle lenses.

Keywords:
Image Warping, Real-Time, Parallel Programming, Stencil Computation, Cell B.E., Heterogeneous Multi-Core Processors

Κεφάλαιο 1

Εισαγωγή

1.1 Περιγραφή του Προβλήματος και Συμβολή της Εργασίας

Οι ευρυγώνιοι φακοί επιτρέπουν την απεικόνιση ενός μεγάλου τομέα του περιβάλλοντος χώρου σε μία μόνο φωτογραφία. Ενώ οι συμβατικοί ευθύγραμμοι φακοί απεικονίζουν τις εισερχόμενες ακτίνες του φωτός σε μία επίπεδη, φωτο-ευαίσθητη επιφάνεια, οι ευρυγώνιοι φακοί απεικονίζουν τις ακτίνες του φωτός σε μία σφαιρική επιφάνεια επιτρέποντας έτσι ένα μεγαλύτερο πεδίο θέασης (FoV). Είναι σύνηθες φαινόμενο αυτού του είδους οι φακοί να επιτρέπουν ένα πεδίο θέασης ίσο με 180^o. Οι ημισφαιρικές εικόνες που λαμβάνονται από τους ευρυγώνιους φακούς χρησιμοποιούνται σε εξειδικευμένες εφαρμογές όπως εφαρμογές παρακολούθησης [42] και πλοήγησης *robot* [36], εφαρμογές εικονικής πραγματικότητας [73], φωτογραφία [75], αστρονομία, εφαρμογές για μέτρηση της ακτινοβολίας του ηλίου σε διάφορα οικοσυστήματα [31][67], κ.α.

Επιπλέον, η χρήση των ευρυγώνιων φακών σε επικρατούσες εφαρμογές όπως ψηφιακή φωτογραφία, λήψη *video* ή ακόμη και τηλε-διάσκεψη (*video-conferencing*) σε πραγματικό χρόνο συνεχώς αυξάνεται. Με την λήψη ενός μεγαλύτερου τμήματος του περιβάλλοντος χώρου, μία *camera* με ευρυγώνιο φακό επιτυγχάνει μία μεγαλύτερη οριζόντια και κάθετη γωνία θέασης.

Το πρόβλημα που δημιουργείται από την απεικόνιση σε μία σφαιρική επιφάνεια είναι η δημιουργία παραμόρφωσης στην παραγόμενη εικόνα. Επομένως, για να είναι δυνατή η χρήση των ευρυγώνιων φακών η εικόνα από τον ευρυγώνιο φακό θα πρέπει να διορθωθεί και να μετατραπεί στο μοντέλο *κεντρικής προοπτικής προβολής* σε πραγματικό χρόνο πριν την παρουσίαση στον τελικό χρήστη της εφαρμογής. Όπως παρουσιάζεται στο Κεφάλαιο 4, η διόρθωση της παραμόρφωσης που προκαλείται από ευρυγώνιους φακούς σε πραγματικό χρόνο δεν είναι δυνατή με χρήση των σύγχρονων επεξεργαστών γενικού σκοπού. Αυτά τα αποτελέσματα μας οδηγούν στην χρήση ενός μη συμβατικού, υψηλής απόδοσης, επεξεργαστή όπως είναι ο επεξεργαστής CBE.

Ο επεξεργαστής *Cell* (*STI Cell Broadband Engine*) αποτελείται από έναν επεξεργαστή *PowerPC* (PPE) και οχτώ Συνεργατικά Στοιχεία Επεξεργασίας (SPEs). Ο *PowerPC* είναι ένας επεξεργαστής Simultaneous Multithreading (SMT), δύο δρόμων, με συχνότητα λειτουργίας ίση με $3.2\ GHz$. Η αρχιτεκτονική των SPEs υποστηρίζει την εκτέλεση πράξεων Single Instruction Multiple Data (SIMD), με εύρος ίσο με $128\ bits$, και ενσωματώνει μία τοπική μνήμη, Local Store (LS), μεγέθους $256\ KB$. Η τοπική μνήμη αποτελεί ενοποιημένη μνήμη εντολών και δεδομένων και ελέγχεται μέσω του λογισμικού ενώ η πρόσβαση στην κύρια μνήμη πραγματοποιείται μέσω ασύγχρονων εντολών Direct Memory Access (DMA). Τέλος, το *dual-issue pipeline* σε κάθε SPE επιτρέπει την εκτέλεση δύο SIMD εντολών σε κάθε κύκλο, επιτυγχάνοντας έτσι μέγιστη θεωρητική απόδοση της τάξης των 204.8 GFlops για πράξεις αριθμών κινητής υποδιαστολής απλής ακριβείας.

1.2 Αποτελέσματα της Διπλωματικής Εργασίας

Η παρούσα διπλωματική εργασία παρουσιάζει μία αρκετά πολύπλοκη εφαρμογή επεξεργασίας εικόνας και περιγράφει τις βελτιστοποιήσεις που απαιτούνται στο επίπεδο του πηγαίου κώδικα για την εκμετάλλευση των υπολογιστικών δυνατοτήτων της ετερογενούς αρχιτεκτονικής του επεξεργαστή CBE ώστε να είναι δυνατή η επίτευξη διόρθωσης παραμόρφωσης σε πραγματικό χρόνο και η μετέπειτα χρήση του αλγορίθμου σε εφαρμογές που θέτουν περιορισμούς πραγματικού χρόνου.

Επίσης, αποτιμάται η απόδοση της εφαρμογής για διάφορα σενάρια υλοποίησης της εφαρμογής. Οι λεπτομερείς βελτιστοποιήσεις που εφαρμόζονται στον κώδικα της εφαρμογής είναι εξαιρετικά απίθανο να πραγματοποιηθούν από κάποιον μεταγλωττιστή βελτιστοποιήσεων, κυρίως λόγω της εφαρμογής μετασχηματισμών υψηλού επιπέδου στον αρχικό, σειριακό κώδικα της εφαρμογής. Συν τοις άλλοις, προσδιορίζονται κάποιες βελτιστοποιήσεις που αντιβαίνουν στην κοινή λογική, οι οποίες όμως ελαχιστοποιούν την επίδραση των μη ευθυγραμμισμένων προσβάσεων στην τοπική μνήμη των SPEs.

Τέλος, αναδεικνύονται κάποιες αδυναμίες των εμπορικά διαθέσιμων μεταγλωττιστών για τον επεξεργαστή CBE και υπογραμμίζεται η ανάγκη για εξελιγμένες δυνατότητες που θα πρέπει να προσφέρουν οι μεταγλωττιστές ώστε να διευκολυνθεί η διαδικασία προγραμματισμού εφαρμογών σε πολύπλοκες, ετερογενείς αρχιτεκτονικές πολλαπλών πυρήνων όπως ο επεξεργαστής CBE.

1.3 Διάρθρωση της Διπλωματικής Εργασίας

Στο Κεφάλαιο 2 παρουσιάζεται το πρόβλημα της παραμόρφωσης που εισάγεται από την χρήση ευρυγώνιων φακών και ο προτεινόμενος αλγόριθμος. Αρχικά, γίνεται μία εισαγωγή και αναλύονται τα μοντέλα προοπτικής προβολής και προβολής στον ευρυγώνιο χώρο. Έπειτα, παρουσιάζεται και αναλύεται ο προτεινόμενος αλγόριθμος για την διόρθωση της παραμόρφωσης που εισάγεται από ευρυγώνιους φακούς.

Το Κεφάλαιο 3 αποτελεί μία σύντομη εισαγωγή στον επεξεργαστή *Cell* (Cell Broadband Engine Architecture (CBEA)). Σε αυτό το κεφάλαιο παρουσιάζεται η αρχιτεκτονική των επιμέρους τμημάτων του επεξεργαστή και τα εργαλεία που προσφέρονται για την ανάπτυξη εφαρμογών στον επεξεργαστή.

Στο Κεφάλαιο 4 παρουσιάζεται η διαδικασία υλοποίησης και βελτιστοποίησης του αλγορίθμου για την εκτέλεση στον επεξεργαστή *Cell* και αναλύονται τα διάφορα αποτελέσματα που προέκυψαν από την εκτέλεση και τις διάφορες προσομοιώσεις, ενώ στο Κεφάλαιο 5 εξετάζεται η ευαισθησία της εφαρμογής σε διάφορες παραμέτρους και παρουσιάζονται μερικά από τα διάφορα *trade-offs* που υπήρξαν κατά την διάρκεια της υλοποίησης του αλγορίθμου.

Έπειτα, στο Κεφάλαιο 6 πραγματοποιείται μία πιο λεπτομερής ανάλυση, χρησιμοποιώντας και μετρικές χαμηλού επιπέδου, της εκτέλεσης της εφαρμογής στον επεξεργαστή *Cell*, σε έναν επεξεργαστή πολλαπλών πυρήνων της *Intel* και σε μία Field Programmable Gate Array (FPGA) που υλοποιεί έναν *streaming accelerator* ο οποίος έχει παραχθεί αυτόματα με την χρήση ενός εργαλείου Electronic System Level (ESL).

Τέλος, στο Κεφάλαιο 7 παρουσιάζονται τα κύρια συμπεράσματα που προέκυψαν από την παρούσα εργασία.

Κεφάλαιο 2

Διόρθωση Παραμόρφωσης Εικόνας από Ευρυγώνιους Φακούς

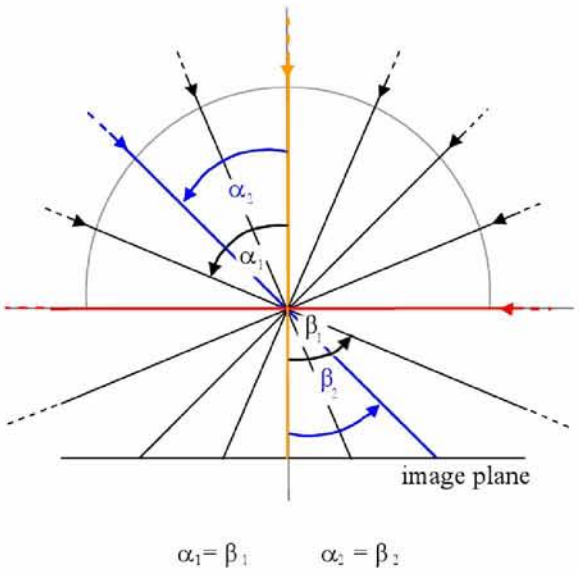
Σε αυτό το κεφάλαιο γίνεται μία εισαγωγή στην παραμόρφωση που προκαλείται από την χρήση ευρυγώνιων φακών, αναλύονται τα μοντέλα προοπτικής προβολής και προβολής ευρυγώνιων φακών και παρουσιάζεται ο προτεινόμενος αλγόριθμος διόρθωσης.

2.1 Εισαγωγή

Όπως αναφέρθηκε στην ενότητα 1.1, οι ευρυγώνιοι φακοί επιτρέπουν την λήψη εικόνων με μεγαλύτερο FoV σε σχέση με τους συμβατικούς φακούς. Αυτό επιτυγχάνεται με την απεικόνιση σε μία σφαιρική, και όχι σε μία επίπεδη, επιφάνεια. Όμως, η απεικόνιση σε μία σφαιρική επιφάνεια έχει ως αποτέλεσμα την παραμόρφωση της παραγόμενης εικόνας. Αυτού του είδους η παραμόρφωση είναι γνωστή και ως *barrel distortion* [69], όπου η μεγέθυνση της εικόνας μειώνεται καθώς αυξάνεται η απόσταση από τον οπτικό άξονα. Ως αποτέλεσμα, οι ευθείες γραμμές κυρτώνονται και οι αποστάσεις μεταξύ των αντικειμένων υφίστανται στρέβλωση. Οι ευρυγώνιοι φακοί εκμεταλλεύονται αυτή την παραμόρφωση για την απεικόνιση μίας ευρείας επιφάνειας του αντικειμένου στην πεπερασμένη επιφάνεια της εικόνας.

Το σχήμα 2.1 παρουσιάζει μία εικόνα η οποία έχει ληφθεί με χρήση ευρυγώνιου φακού. Όπως παρατηρούμε, η παραμόρφωση της εικόνας, υπό την μορφή ενός είδους κύρτωσης, είναι εμφανής

Σχήμα 2.1: Εικόνα που έχει ληφθεί με χρήση ευρυγώνιου φακού.



Σχήμα 2.2: Μοντέλο Κεντρικής Προοπτικής Προβολής.

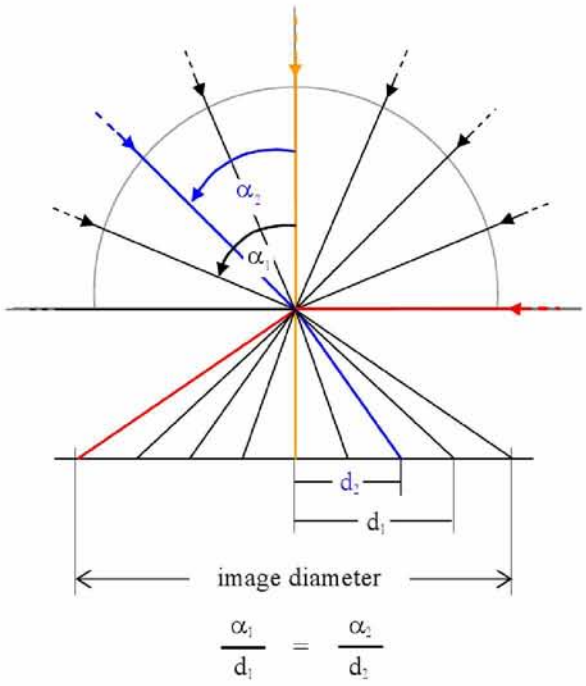
και ιδιαιτέρως έντονη στις γωνίες αυτής. Λόγω αυτής της παραμόρφωσης, οι ευρυγώνιοι φακοί δεν μπορούν να χρησιμοποιηθούν σε εφαρμογές εάν δεν ληφθεί ειδική μέριμνα. Επομένως, πριν την προβολή της εικόνας στον τελικό χρήστη, θα πρέπει να μεσολαβεί κάποιο στάδιο διόρθωσης, όπου θα αφαιρείται η παραμόρφωση και η εικόνα θα μετασχηματίζεται στο μοντέλο της κεντρικής προοπτικής προβολής.

2.2 Κεντρική Προοπτική Προβολή και Μοντέλο Προβολής Ευρυγώνιων Φακών

Το μαθηματικό μοντέλο της *κεντρικής προοπτικής προβολής* βασίζεται στην υπόθεση ότι η γωνία πρόσπτωσης μίας ακτίνας από ένα σημείο είναι ίση με την γωνία μεταξύ της ακτίνας και του οπτικού άξονα (ο άξονας απεικονίζεται με πορτοκαλί χρώμα) στο επίπεδο της εικόνας. Το σχήμα 2.2, [48], παρουσιάζει το μοντέλο προβολής για δύο ακτίνες, με γωνίες πρόσπτωσης α_1 και α_2. Από το σχήμα, γίνεται εμφανές ότι σημεία του αντικειμένου με γωνία πρόσπτωσης που προσεγγίζει τις 90^o προβάλλονται σε άπειρη απόσταση από το πρωτεύον σημείο[1] (*principal point*). Η ακτίνα με το κόκκινο χρώμα αποτελεί ένα παράδειγμα ακτίνας με γωνία πρόσπτωσης ίση με 90^o, η οποία δεν προβάλλεται στο επίπεδο της εικόνας. Αυτό έχει ως αποτέλεσμα τον περιορισμό των τιμών του FoV σε γωνίες οι οποίες δεν αποκλίνουν πολύ από τον οπτικό άξονα.

Όπως και στο μοντέλο των πανοραμικών εικόνων [47], η γεωμετρία των εικόνων που λαμβάνονται με χρήση ευρυγώνιων φακών δεν ακολουθεί το μοντέλο της κεντρικής προοπτικής προβολής. Επομένως, οι μαθηματικές ιδιότητες αυτού του μοντέλου δεν μπορούν να εφαρμοσθούν στο μοντέλο των ευρυγώνιων φακών. Για να επεκταθεί το εύρος των τιμών που μπορεί να λάβει

[1]Το πρωτεύον σημείο είναι το σημείο από το οποίο διέρχονται όλες οι ακτίνες.



Σχήμα 2.3: Μοντέλο Προβολής Ευρυγώνιων Φακών.

η γωνία θέασης ενός φακού και για την πλήρη προβολή ημισφαιρίων στο επίπεδο της εικόνας, η ύπαρξη ενός νέου μοντέλου προβολής κρίνεται αναγκαία.

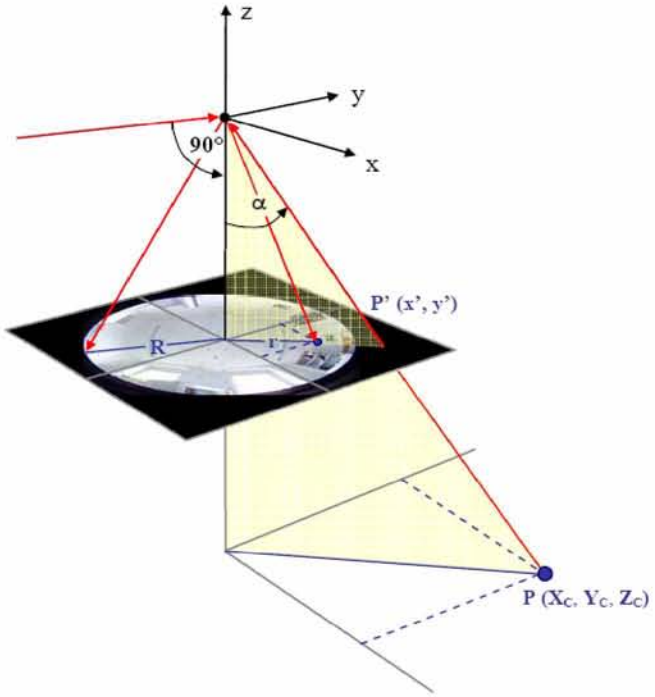
Το μοντέλο προβολής ευρυγώνιων φακών[2] βασίζεται στην παραδοχή ότι, στην ιδανική περίπτωση, η γωνία πρόσπτωσης είναι ανάλογη της απόστασης μεταξύ του σημείου της εικόνας και του πρωτεύοντος σημείου. Το μοντέλο αυτό παρουσιάζεται στο σχήμα 2.3, όπου ισχύει η σχέση $\frac{d_1}{d_2} = \frac{\alpha_1}{\alpha_2}$. Χρησιμοποιώντας αυτό το μοντέλο, οι εισερχόμενες ακτίνες διαθλώνται πιο κοντά στον οπτικό άξονα, επεκτείνοντας έτσι το FoV των φακών.

Προκειμένου να είναι δυνατή η ανακατασκευή της προβολής του σημείου ενός αντικειμένου στο επίπεδο της ημισφαιρικής εικόνας, θα πρέπει οι συντεταγμένες του αντικειμένου και της εικόνας να αναφέρονται στο ίδιο σύστημα συντεταγμένων. Για την αντιστοίχιση των συντεταγμένων (i, j) ενός σημείου από τον δισδιάστατο χώρο της κεντρικής προοπτικής προβολής στις συντεταγμένες (x, y) του σημείου στον τρισδιάστατο, ευρυγώνιο, χώρο, θα πρέπει πρώτα να υπολογισθούν οι συντεταγμένες (X_c, Y_c, Z_c) της προβολής του σημείου (i, j) στο τρισδιάστατο σύστημα συντεταγμένων της *camera*. Αυτό επιτυγχάνεται εφαρμόζοντας τον ακόλουθο πίνακα περιστροφής (*rotation matrix*):

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \times \begin{bmatrix} i \\ j \\ 1 \end{bmatrix}$$

Το σχήμα 2.4 απεικονίζει γραφικά τις ιδιότητες της προβολής όταν χρησιμοποιούνται ευρυγώνιοι φακοί. Εάν υποθέσουμε ότι χρησιμοποιείται ένας ευρυγώνιος φακός με γωνία ίση με 180^o και ότι η ακτίνα από ένα σημείο του αντικειμένου έχει γωνία πρόσπτωσης ίση με 90^o, τότε το σημείο

[2]Στην βιβλιογραφία συναντάται και ως *fisheye projection model*.



Σχήμα 2.4: Γεωμετρική απεικόνιση της σχέσης μεταξύ ενός σημείου του αντικειμένου και του αντίστοιχου σημείου της εικόνας στο σύστημα συντεταγμένων της *camera*.

προβάλλεται στο εξωτερικό της κυκλικής ευρυγώνιας εικόνας. Αυτό έχει ως αποτέλεσμα το σημείο που προκύπτει να έχει την μέγιστη απόσταση από τον οπτικό άξονα. Η σχέση μεταξύ της γωνίας πρόσπτωσης και της απόστασης του αντίστοιχου σημείου από το πρωτεύον σημείο είναι σταθερή για όλη την εικόνα. Επομένως, ο λόγος που περιγράφεται από την ακόλουθη σχέση μπορεί να χρησιμοποιηθεί ως βασική εξίσωση για την ευρυγώνια προβολή:

$$\frac{\alpha}{r} = \frac{90^o}{R}, r = \sqrt{x^2 + y^2}, \tag{2.1}$$

όπου α είναι η γωνία πρόσπτωσης, r η απόσταση μεταξύ του σημείου της εικόνας και του οπτικού άξονα, R η ακτίνα της εικόνας και x, y είναι οι συντεταγμένες στον χώρο της *wide angle* εικόνας.

Με χρήση κάποιων αλγεβρικών μετασχηματισμών [48], οι εξισώσεις που περιγράφουν την προβολή στο επίπεδο της εικόνας όταν χρησιμοποιούνται ευρυγώνιοι φακοί είναι οι ακόλουθες:

$$x = \frac{\frac{2R}{\pi} \cdot \arctan(\frac{\sqrt{X_c^2+Y_c^2}}{Z_c})}{\sqrt{(\frac{Y_c}{X_c})^2 + 1}} + d_x + x_h \tag{2.2a}$$

$$y = \frac{\frac{2R}{\pi} \cdot \arctan(\frac{\sqrt{X_c^2+Y_c^2}}{Z_c})}{\sqrt{(\frac{Y_c}{X_c})^2 + 1}} + d_y + y_h, \tag{2.2b}$$

όπου (X_c, Y_c, Z_c) είναι οι συντεταγμένες του σημείου στον τρισδιάστατο χώρο των συντεταγμένων της *camera*, d_x, d_y είναι παράγοντες που μοντελοποιούν τις παραμέτρους της παραμόρφωσης που εισάγει ο εκάστοτε φακός, x_h, y_h είναι οι συντεταγμένες του πρωτεύοντος σημείου και R είναι η ακτίνα της εικόνας. Οι σχέσεις (2.2) παρέχουν μία μέθοδο για την μετατροπή των συντεταγμένων (X_c, Y_c, Z_c) ενός αντικειμένου από τον τρισδιάστατο χώρο της κεντρικής προοπτικής προβολής στον διδιάστατο χώρο των ευρυγώνιων φακών. Αυτή η διαδικασία ονομάζεται *αντίστροφη απεικόνιση - inverse mapping*. Οι προαναφερθείσες σχέσεις μπορούν να διασπαστούν σε στοιχειώδεις συναρτήσεις, όπου πλέον ενσωματώνονται και οι παράγοντες που μοντελοποιούν την παραμόρφωση που εισάγουν οι φακοί. Οι σχέσεις αυτές είναι οι ακόλουθες:

$$d = \sqrt{X_c^2 + Y_c^2} \tag{2.3a}$$

$$D_u = \frac{d}{Z_c} \tag{2.3b}$$

$$R_u = \arctan D_u \tag{2.3c}$$

$$P = k_1 \cdot R_u^4 + k_2 \cdot R_u^3 + k_3 \cdot R_u^2 + k_4 \cdot R_u + k_5 \tag{2.3d}$$

$$x = \frac{P}{d} \cdot X_c + x_h \tag{2.3e}$$

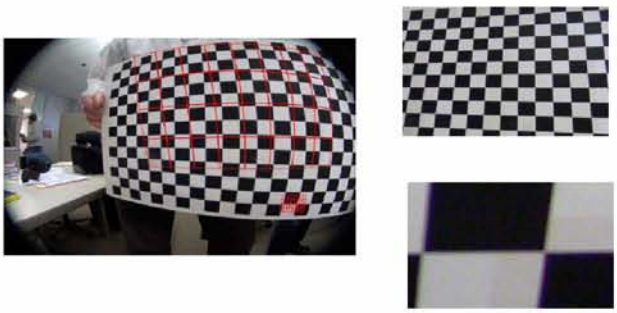
$$y = \frac{P}{d} \cdot Y_c + y_h, \tag{2.3f}$$

όπου k_i είναι παράμετροι που εξαρτώνται από τον τύπο του φακού που χρησιμοποιείται. Οι σχέσεις (2.3) μοντελοποιούν την παραμόρφωση που εισάγεται από την χρήση ενός ευρυγώνιου φακού, χρησιμοποιώντας τις παραμέτρους του αντίστοιχου αισθητήρα $k[]$. Δοθέντων των συντεταγμένων (X_c, Y_c, Z_c) ενός σημείου, χρησιμοποιούμε τις σχέσεις (2.3) ώστε να υπολογίσουμε τις αντίστοιχες συντεταγμένες του σημείου στον χώρο των ευρυγώνιων φακών.

2.3 Προτεινόμενος Αλγόριθμος Διόρθωσης

Ο προτεινόμενος αλγόριθμος λαμβάνει ως είσοδο μία εικόνα, που έχει ληφθεί από έναν ευρυγώνιο φακό και έχει υποστεί παραμόρφωση, όπως επίσης και την τιμή της παραμέτρου FoV και παράγει ως έξοδο την εικόνα όπου πλέον έχει αφαιρεθεί η παραμόρφωση. Ο χρήστης καθορίζει το κέντρο του *πεδίου ενδιαφέροντος*, Region Of Interest (ROI), με μέγεθος ίσο με $1280x960$ *pixels* και η εικόνα που παράγεται έχει μέγεθος εικόνας τύπου *VGA*, δηλαδή $640x480$ *pixels*. Το σχήμα 2.5 παρουσιάζει μία ενδεικτική χρήση του αλγορίθμου. Όπως παρατηρούμε, μικρότερες τιμές της παραμέτρου FoV έχουν ως αποτέλεσμα την ύπαρξη μεγέθυνσης, *zoom*, στην εικόνα που παράγεται.

Για να είναι δυνατή η διόρθωση της παραμόρφωσης, αρχικά θα πρέπει να γίνει η αντίστροφη απεικόνιση για τα *pixels* εντός του *πεδίου ενδιαφέροντος*. Για το σκοπό αυτό χρησιμοποιούνται οι σχέσεις (2.3). Όπως παρατηρούμε, οι εν λόγω σχέσεις δεν παράγουν ακέραιες τιμές για τις συντεταγμένες των *pixels* στο πεδίο της εικόνας. Επομένως, καθώς δεν υπάρχει απευθείας αντιστοίχιση των *pixels* του ενός χώρου στα *pixels* του άλλου χώρου, θα πρέπει να χρησιμοποιηθεί κάποιο σχήμα παρεμβολής για τον υπολογισμό των τιμών του *pixels*, χρησιμοποιώντας τιμές από γειτονικά *pixels*, τα οποία βρίσκονται σε ακέραιες θέσεις στον χώρο. Για τον υπολογισμό των τιμών του εκάστοτε *pixel* που βρίσκεται σε μη ακέραια θέση χρησιμοποιούμε το σχήμα παρεμβολής *Bicubic Interpolation* [37].



Σχήμα 2.5: Ο αλγόριθμος διόρθωσης για τιμές της παραμέτρου FoV ίσες με $FoV = 60^o$ και $FoV = 8^o$.

Αυτό το σχήμα παρεμβολής αποτελεί μία μέθοδο με μεγάλη ανοχή σε σφάλματα αλλά ταυτόχρονα και με μεγάλες υπολογιστικές απαιτήσεις. Η μέθοδος αυτή χρησιμοποιεί πολυωνυμικές συναρτήσεις τρίτου βαθμού ώστε να προσεγγιστούν οι τιμές των ενδιάμεσων σημείων. Καθώς το πολυώνυμο που χρησιμοποιείται είναι τρίτου βαθμού, απαιτούνται τέσσερεις κόμβοι για την παρεμβολή, έστω $C_i, i \in 1 \dots 4$. Δοθέντων των κόμβων της παρεμβολής, C_i, όπου η συνάρτηση παρεμβολής $g()$ είναι ίση με την, άγνωστη, παρεμβαλλόμενη συνάρτηση $f()$, η διαδικασία παρεμβολής για την τιμή της συνάρτησης $f()$ σε ένα σημείο x δύναται να περιγραφεί από τις ακόλουθες σχέσεις:

$$g(x) = C_1 \cdot U_1(s) + C_2 \cdot U_2(s) + C_3 \cdot U_3(s) + C_4 \cdot U_4(s) \tag{2.4a}$$

$$U_1 = \frac{-s^3 + 2s^2 - s}{2} \tag{2.4b}$$

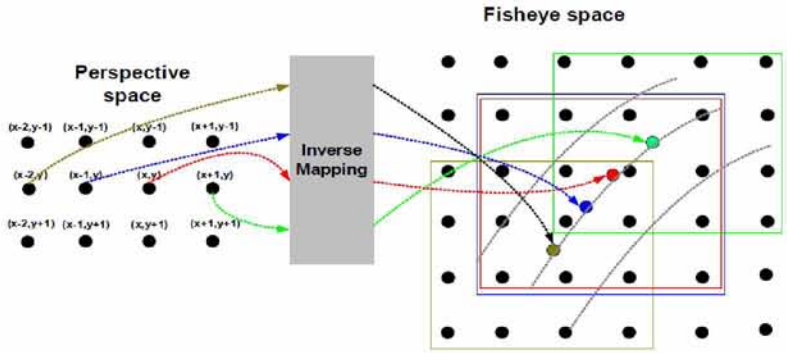
$$U_2 = \frac{3s^3 - 5s^2 + 2}{2} \tag{2.4c}$$

$$U_3 = \frac{-3s^3 + 4s^2 + s}{2} \tag{2.4d}$$

$$U_4 = \frac{s^3 - s^2}{2} \tag{2.4e}$$

όπου το σημείο x είναι τέτοιο ώστε $x_1 \leq x_2 \leq x \leq x_3 \leq x_4$ και η παράμετρος s ισούται με $s = x - x_2$.

Το σχήμα 2.6 παρουσιάζει την *αντίστροφη απεικόνιση* και την μέθοδο παρεμβολής που χρησιμοποιείται για παρεμβολή στις δύο διαστάσεις. Η παρεμβολή στις δύο διαστάσεις επιτυγχάνεται παρεμβάλλοντας πρώτα τα σημεία στην οριζόντια διάσταση και έπειτα στην κάθετη διάσταση. Για την παρεμβολή χρησιμοποιούνται τα *pixels* που βρίσκονται εντός ενός παραθύρου διαστάσεων $4x4$ γύρω από το επιθυμητό σημείο. Η συνάρτηση παρεμβολής $G()$ στο σημείο με συντεταγμένες (x, y) παρουσιάζεται στις ακόλουθες σχέσεις:



Σχήμα 2.6: Η αντίστροφη απεικόνιση και το σχήμα παρεμβολής στην 4x4 γειτονιά.

$$G(x,y) = g_1(x)\cdot V_1(t) + g_2(x)\cdot V_2(t) + g_3(x)\cdot V_3(t) + g_4(x)\cdot V_4(t) \tag{2.5a}$$

$$V_1(t) = \frac{-s^3 + 2s^2 - s}{2} \tag{2.5b}$$

$$V_2(t) = \frac{3s^3 - 5s^2 + 2}{2} \tag{2.5c}$$

$$V_3(t) = \frac{-3s^3 + 4s^2 + s}{2} \tag{2.5d}$$

$$V_4(t) = \frac{s^3 - s^2}{2} \tag{2.5e}$$

όπου $t = x - \lfloor x \rfloor$ και $g_k(x)$ είναι η τιμή που προκύπτει από την παρεμβολή στην οριζόντια διάσταση για τις γραμμές $\lfloor y \rfloor - 1$, $\lfloor y \rfloor$, $\lfloor y \rfloor + 1$ και $\lfloor y \rfloor + 2$.

Παρόλο που υπήρχε η δυνατότητα χρήσης και άλλων, απλούστερων, τεχνικών παρεμβολής, όπως παρεμβολή βάσει του κοντινότερου γειτονικού[3] *pixel* ή διγραμμική παρεμβολή[4], οι υψηλές απαιτήσεις της εφαρμογής όσον αφορά στον λόγο *Σήματος προς Θόρυβο* (*Signal to Noise ratio*), ειδικότερα για μικρές τιμές της παραμέτρου FoV, καθιστούν τη μέθοδο αυτή ως την βέλτιστη για την επίτευξη του επιθυμητού αποτελέσματος. Επίσης, υπολογιστικά πολυπλοκότερες μέθοδοι, όπως η μέθοδος *Elliptical Weighted Average (EWA)* [32], απορρίφθηκαν καθώς το κόστος αυτών ήταν μεγαλύτερο από αυτό που απαιτούνταν από την εφαρμογή ενώ ενδεχομένως καθιστούσαν και την διόρθωση της εικόνας σε πραγματικό χρόνο μη εφικτή.

Όπως αναφέρθηκε και παραπάνω, τα κύρια στάδια του αλγορίθμου, δηλαδή η *αντίστροφη απεικόνιση* και η εφαρμογή του σχήματος παρεμβολής, εκτελούνται σε μία εικόνα διαστάσεων $1280x960$. Η εκτέλεση αυτών των σταδίων υπολογισμού σε μία εικόνα μεγαλύτερων διαστάσεων από την εικόνα εξόδου συντελεί στην διατήρηση της πληροφορίας που εμπεριέχεται σε περιοχές με υψηλή χωρητική συχνότητα. Το τελευταίο στάδιο στην επεξεργασία της εικόνας είναι η δειγματοληψία προς τα κάτω της εικόνας ώστε από την αρχική ανάλυση $1280x960$ να μεταβούμε σε ανάλυση *VGA* για την εικόνα εξόδου. Αυτό επιτυγχάνεται με την εφαρμογή ενός *5-tap* βαθυπερατού φίλτρου πρώτα στην κάθετη και έπειτα στην οριζόντια κατεύθυνση με συντελεστές

[3]Nearest Neighbor
[4]Bilinear Interpolation

1: {**Input**: The frames (in the wide-angle space) to be corrected}
2: {**Output**: The corrected frames (in the perspective space)}
3: **for all** frames **do**
4: **for all** pixels in the output frame **do**
5: Compute the corresponding fractional position in the input frame (**InverseMapping**())
6: Interpolate the pixel value at that fractional position (**BicubicInterpolation**())
7: **end for**
8: Apply a 2-D low-pass filter to resize the output frame (**LPF**())
9: **end for**

Σχήμα 2.7: Ψευδοκώδικας για τον αλγόριθμο διόρθωσης (αρχική έκδοση).

$0, 0.171887, 0.5, 0.171887, 0$. Η εφαρμογή του φίλτρου σε κάθε μία κατεύθυνση έχει ως αποτέλεσμα την δειγματοληψία προς τα κάτω κατά έναν παράγοντα ίσο με 2 ώστε, συνολικά, το μέγεθος της εικόνας να υποτετραπλασιάζεται. Αντί της εφαρμογής του φίλτρου, η κλιμάκωση προς τα κάτω θα ήταν δυνατή με επιλογή κάθε δεύτερου *pixel* από την εικόνα που παράγεται μετά το στάδιο της παρεμβολής. Αυτό το σχήμα δεν θα παρήγαγε το επιθυμητό αποτέλεσμα ενώ θα στερούσε από την εφαρμογή και μία θετική παρενέργεια του βαθυπερατού φίλτρου, η οποία είναι η απομάκρυνση των υψηλής συχνότητας, τεχνητών *pixels*[5]. Σε αυτό το σημείο, θα πρέπει να αναφερθεί ότι η απομάκρυνση αυτών των *pixels* δεν αποτελεί πρωτεύων μέλημα στην σχεδίαση του αλγορίθμου αλλά το φίλτρο εφαρμόζεται ως μία καλύτερη εναλλακτική λύση για την επιθυμητή δειγματοληψία της εικόνας.

Συνοψίζοντας, τα στάδια επεξεργασίας του προτεινόμενου αλγορίθμου είναι τα ακόλουθα:

- Αντίστροφη Απεικόνιση: Σε αυτό το στάδιο τα *pixels* της εικόνας εξόδου αντιστοιχίζονται σε *pixels* στο πεδίο των ευρυγώνιων φακών.
- Παρεμβολή: Το στάδιο της παρεμβολής χρησιμοποιείται ώστε να προσεγγιστούν οι τιμές των *pixels* της εικόνας εξόδου τα οποία αντιστοιχίζονται σε κλασματικές θέσεις κατά την αντίστροφη απεικόνιση.
- Βαθυπερατό Φίλτρο: Το τελευταίο στάδιο επεξεργασίας του αλγορίθμου είναι η εφαρμογή ενός βαθυπερατού φίλτρου ώστε να γίνει η προς τα κάτω δειγματοληψία της εικόνας.

Το σχήμα 2.7 παρουσιάζει έναν ενδεικτικό, υψηλού επιπέδου, ψευδοκώδικα για τον αλγόριθμο διόρθωσης της παραμόρφωσης που προκαλείται από την χρήση ευρυγώνιων φακών.

[5]Στην βιβλιογραφία αναφέρονται και ως artifacts

Κεφάλαιο 3

Γενική Επισκόπηση του Επεξεργαστή Cell (Cell Broadband Engine)

Στο παρόν κεφάλαιο γίνεται μία εισαγωγή στον επεξεργαστή *Cell*. Αρχικά, παρουσιάζονται τα κίνητρα για την δημιουργία του επεξεργαστή, έπειτα παρουσιάζεται η αρχιτεκτονική αυτού και τέλος αναλύονται τα εργαλεία για την ανάπτυξη εφαρμογών στον επεξεργαστή.

3.1 Εισαγωγή

Οι αυξανόμενες απαιτήσεις νέων εφαρμογών, όπως εφαρμογές πολυμέσων, σε υπολογιστική δύναμη έχουν οδηγήσει σε μία συνεχή αύξηση των απαιτήσεων για επεξεργαστική ισχύ.

Κατά την προηγούμενη δεκαετία, η έλευση της τεχνολογίας Very Large Scale Integration (VLSI) και η συνεχής αλλαγή κλίμακας είχαν ως αποτέλεσμα την αύξηση της απόδοσης των μικροεπεξεργαστών κατά $50\% - 60\%$. Οι κύριοι παράγοντες που συντελούσαν σε αυτή την αύξηση ήταν η αύξηση της ταχύτητας του ρολογιού, τόσο με την μείωση των επιπέδων λογικής για κάθε κύκλο ρολογιού όσο και με την αλλαγή κλίμακας της τεχνολογίας, όπως επίσης και η εκμετάλλευση του μεγάλου αριθμού των transistors σε κάθε chip, ο οποίος διπλασιαζόταν βάσει του *Νόμου του Moore*.

Πλέον, η επίτευξη μεγάλης απόδοσης στους μικροεπεξεργαστές με χρήση συμβατικών τεχνικών θα είναι πολύ δύσκολη. Ενώ η ταχύτητα των transistors αυξάνεται δραματικά, η ταχύτητα των καλωδίων μειώνεται. Σε συνδυασμό με την αύξηση του μεγέθους του chip, η οποία παρατηρείται στους νέους μικροεπεξεργαστές, ο χρόνος για να αποσταλεί ένα σήμα θα αυξηθεί σημαντικά αφού η απόσταση την οποία θα μπορεί το σήμα να διανύσει σε έναν κύκλο ρολογιού θα μειωθεί. Έτσι, δεν θα είναι δυνατή τόσο η αύξηση των σταδίων του pipeline, η οποία τα προηγούμενα χρόνια είχε επιφέρει σημαντική βελτίωση στην απόδοση των μικροεπεξεργαστών, όσο και η προσθήκη επιπλέον μονάδων εκτέλεσης καθώς η καθυστέρηση για την μετάδοση ενός σήματος μεταξύ των σταδίων του pipeline και μεταξύ των επιμέρους υπομονάδων θα αυξηθεί. Ο συνδυασμός των ανωτέρω θα έχει άμεση επίδραση στην διεκπεραιωτική ικανότητα (*throughput*) του επεξεργαστή.

Συν τοις άλλοις, η προαναφερθείσα αύξηση της καθυστέρησης θα αυξήσει και τον χρόνο που απαιτείται για πρόσβαση στις δομικές μονάδες του επεξεργαστή, όπως στους φακέλους καταχωρητών, στην μνήμη cache ακόμη και στο παράθυρο εντολών (*instruction window*). Καθώς η καθυστέρηση πρόσβασης εξαρτάται από την χωρητικότητα της κάθε μονάδας, για να είναι δυνατή η επίτευξη χαμηλής καθυστέρησης θα πρέπει η χωρητικότητα της κάθε μονάδας να μειωθεί. Αυτός θα είναι ένας παράγοντας που θα επηρεάσει την απόδοση των μικροεπεξεργαστών καθώς δεν

θα είναι πλέον δυνατή η επίτευξη μεγαλύτερης απόδοσης μέσω της αύξησης του μεγέθους των δομικών μονάδων. Ως ενδεικτικό παράδειγμα μπορεί να αναφερθεί η εξάρτηση του παραλληλισμού που μπορεί να επιτευχθεί σε επίπεδο εντολής από το μέγεθος του παραθύρου εντολών.

Επίσης, η αύξηση της καθυστέρησης θα έχει άμεση επίδραση στον παραλληλισμό σε επίπεδο εντολών (ILP). Σε συνδυασμό με την αύξηση της συχνότητας του ρολογιού, η κατάσταση που θα μπορεί ο επεξεργαστής να εκμεταλλευτεί για δεδομένο βάθος του pipeline θα μειώνεται. Τέλος, η αύξηση της συχνότητας λειτουργίας του επεξεργαστή έχει ως αποτέλεσμα την αύξηση της καταναλισκόμενης ισχύος και ιδίως της στατικής συνιστώσας αυτής. Το κυρίως πρόβλημα που δημιουργεί η αύξηση της ισχύος είναι η αύξηση της πυκνότητας ισχύος, η οποία γίνεται εντονότερη με την αλλαγή κλίμακας των επεξεργαστών.

Λαμβάνοντας υπόψη τα ανωτέρω, οι σχεδιαστές μικροεπεξεργαστών στράφηκαν στην κατασκευή επεξεργαστών με μεγαλύτερο αριθμό από απλούστερους πυρήνες στο ίδιο *chip*. Αυτού του είδους οι επεξεργαστές αποκαλούνται επεξεργαστές πολλαπλών πυρήνων - *multicores*. Οι κυρίαρχες προσεγγίσεις στην κατασκευή επεξεργαστών πολλαπλών πυρήνων είναι δύο. Η πρώτη είναι η *Συμμετρική Πολυεπεξεργασία* (Symmetric Multiprocessing (SMP)), όπου σε ένα *chip* τοποθετούνται πολλαπλοί όμοιοι επεξεργαστές. Αυτή η προσέγγιση υιοθετείται από τις τελευταίες γενεές επεξεργαστών της *Intel* και της *AMD*. Η δεύτερη προσέγγιση συνίσταται στην δημιουργία *tiled* ή *heterogeneous* αρχιτεκτονικών. Χαρακτηριστικά παραδείγματα αυτής της προσέγγισης αποτελούν οι επεξεργαστές *Cell* [34] και *RAW* [22, 25].

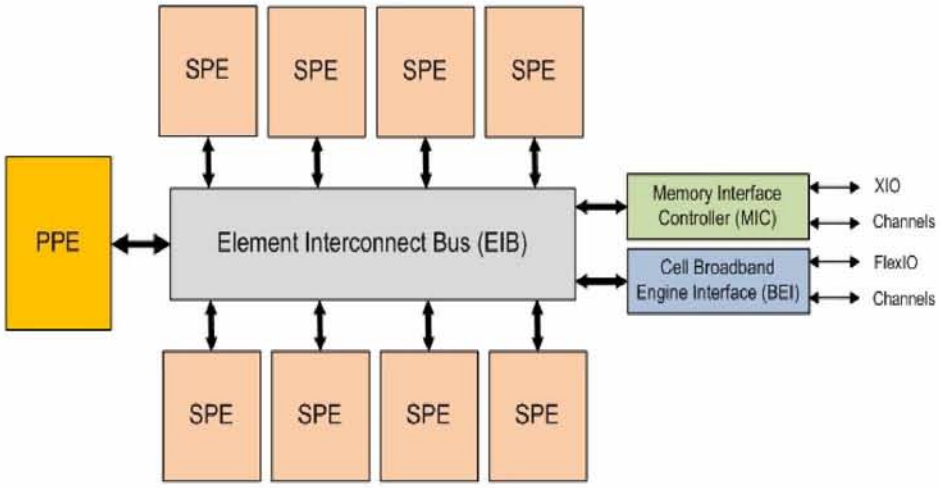
Ο επεξεργαστής *Cell* (CBE) είναι ένας μικροεπεξεργαστής που δημιουργήθηκε από την συνεργασία των εταιρειών *Sony, Toshiba* και *IBM* το 2001 ώστε να αποτελέσει τον επεξεργαστή για την νέα *game console* της *Sony*, το *PlayStation 3*. Ενώ αρχικά σχεδιάστηκε για χρήση σε *game consoles* και σε συσκευές πολυμέσων για το καταναλωτικό κοινό, είναι αρκετά ευέλικτος ώστε να χρησιμοποιηθεί ως επεξεργαστής γενικού σκοπού. Ταυτόχρονα, υπάρχει μεγάλο ερευνητικό ενδιαφέρον για την χρήση του επεξεργαστή σε μία πληθώρα άλλων εφαρμογών όπως εφαρμογές επιστημονικού υπολογισμού ή εφαρμογές υπερ-υπολογισμού (*supercomputing*).

Η πρώτη υλοποίηση αυτού αποτελεί έναν *single-chip* επεξεργαστή πολλαπλών πυρήνων με εννέα επεξεργαστές, οι οποίοι λειτουργούν σε ένα μοντέλο κοινής μνήμης (*shared memory model*), όπως παρουσιάζεται στο σχήμα 3.1. Το χαρακτηριστικό που διαφοροποιεί τον CBE από ήδη υπάρχοντες επεξεργαστές είναι ότι ενώ όλοι οι επεξεργαστές διαμοιράζονται την υπάρχουσα μνήμη, αυτοί ειδικεύονται σε δύο κατηγορίες: το Power Processing Element (PPE) και το Synergistic Processing Element (SPE). Ο επεξεργαστής *Cell* αποτελείται από έναν επεξεργαστή τύπου PPE και οχτώ επεξεργαστές τύπου SPE.

Ο πρώτος τύπος επεξεργαστικού στοιχείου, το PPE, εμπεριέχει έναν πυρήνα *64-bit* PowerPC® Architecture^{TM}. Ο πυρήνας ακολουθεί την αρχιτεκτονική *64-bit* PowerPC και μπορεί να εκτελέσει λειτουργικά συστήματα και εφαρμογές *32-bit* και *64-bit*. Ο δεύτερος τύπος επεξεργαστικού στοιχείου, το SPE, αποτελεί έναν πυρήνα βελτιστοποιημένο για την εκτέλεση Single Instruction Multiple Data (SIMD) εφαρμογών με μεγάλες απαιτήσεις όσον αφορά στην επεξεργαστική ισχύ.

Τα SPEs είναι ανεξάρτητα επεξεργαστικά στοιχεία, με τον κάθε πυρήνα να εκτελεί τις δικές του εφαρμογές ή νήματα, και έχουν πλήρη πρόσβαση στην κοινή μνήμη. Μεταξύ των SPEs και του PPE υπάρχει μία αμοιβαία εξάρτηση. Τα SPEs εξαρτώνται από το PPE για την εκτέλεση του λειτουργικού συστήματος, την εκτέλεση διαφόρων κλήσεων συστήματος και στην πλειονότητα των περιπτώσεων για την εκτέλεση του νήματος που ελέγχει τα νήματα που εκτελούνται στα SPEs. Το PPE εξαρτάται από τα SPEs τα οποία είναι υπεύθυνα για το μεγαλύτερο μέρος της απόδοσης του συστήματος.

Για τον προγραμματιστή κάποιας εφαρμογής, ο *Cell* αποτελεί έναν *dual-threaded* επεξεργαστή με οχτώ επιπλέον πυρήνες, οι οποίοι έχουν πρόσβαση στην δική τους τοπική μνήμη. Η



Σχήμα 3.1: Η αρχιτεκτονική του επεξεργαστή *Cell*.

ποικιλομορφία του *Cell*, με τον διαχωρισμό των πυρήνων σε πυρήνες βελτιστοποιημένους για λειτουργίες ελέγχου και πυρήνες βελτιστοποιημένους για υπολογισμούς, είναι το στοιχείο που επιτρέπει την δραματική βελτίωση στην μέγιστη υπολογιστική δυνατότητα που διαθέτει και τον καθιστά περισσότερο αποδοτικό, όσον αφορά στην κατανάλωση ισχύος και στην απαιτούμενη επιφάνεια για την υλοποίηση, έναντι συμβατικών μικροεπεξεργαστών.

Η κυριότερη διαφορά μεταξύ των δύο τύπων πυρήνων στον επεξεργαστή *Cell* είναι ο τρόπος με τον οποίο πραγματοποιείται η προσπέλαση μνήμης. Το μεν PPE προσπελαύνει την κύρια μνήμη (*effective address space*) με εντολές τύπου *load, store*, με τα αποτελέσματα να αποθηκεύονται σε έναν ιδιωτικό φάκελο καταχωρητών ενώ δίνεται και η δυνατότητα για *caching* των δεδομένων. Αυτή η μέθοδος πρόσβασης ομοιάζει με τον τρόπο πρόσβασης στην μνήμη όπως αυτή πραγματοποιείται στους συμβατικούς μικροεπεξεργαστές.

Κάθε SPE, σε αντίθεση, προσπελαύνει την κύρια μνήμη μέσω εντολών Direct Memory Access (DMA), οι οποίες μεταφέρουν δεδομένα μεταξύ της κύριας μνήμης και μίας τοπικής, ιδιωτικής, μνήμης, η οποία αποκαλείται Local Store (LS). Η τοπική μνήμη προσπελαύνεται απευθείας από τις εντολές προσκόμισης, φόρτωσης και αποθήκευσης ενώ δεν υπάρχει κάποια κρυφή μνήμη *cache*. Αυτή η ιεραρχία μνήμης τριών επιπέδων (φάκελος καταχωρητών, τοπική μνήμη, κύρια μνήμη), με τις ασύγχρονες εντολές DMA, αποτελεί μία ριζική αλλαγή σε σχέση με τις συμβατικές αρχιτεκτονικές και τα προγραμματιστικά μοντέλα καθώς ο παραλληλισμός μεταξύ εκτέλεσης υπολογισμών και μεταφοράς δεδομένων είναι σαφής.

Το κυρίως κίνητρο για την υιοθέτηση αυτού του μοντέλου είναι η μεγάλη καθυστέρηση που υπάρχει για την προσπέλαση της κύριας μνήμης, η οποία έχει αυξηθεί δραματικά στις τελευταίες γενεές επεξεργαστών. Αυτό έχει σαν συνέπεια η απόδοση της εφαρμογής να εξαρτάται από την καθυστέρηση της μνήμης και όχι από την υπολογιστική ικανότητα. Επί παραδείγματι, εάν η εφαρμογή έχει μία αστοχία κατά την πρόσβαση στην μνήμη *cache*, θα πρέπει να παγώσει την εκτέλεσή της για αρκετούς κύκλους ρολογιού με αποτέλεσμα την μείωση της απόδοσης. Εν αντιθέσει, ο μικρός αριθμός κύκλων ρολογιού που απαιτείται για την έναρξη της εκτέλεσης μίας εντολής DMA και η δυνατότητα επικάλυψης υπολογισμών και μεταφοράς δεδομένων με χρήση

της τεχνικής του *multi-buffering* μπορούν να αυξήσουν την απόδοση της εκάστοτε εφαρμογής.

Με αυτόν τον καινοτόμο σχεδιασμό η μικρο-αρχιτεκτονική του επεξεργαστή CBEA μπορεί να υποστηρίξει μεγάλη υπολογιστική ικανότητα μέσω της εκμετάλλευσης των παρακάτω:

- *Παραλληλισμός μεταξύ των δεδομένων* (Data Level Parallelism (DLP)) με τις μονάδες SIMD που βρίσκονται στα SPEs.
- *Παραλληλισμός μεταξύ των εντολών* (Instruction Level Parallelism (ILP)) με την *dual-issue* αρχιτεκτονική των *pipelines*.
- *Παραλληλισμός μεταξύ των νημάτων* (Thread Level Parallelism (TLP)) μέσω των πολλαπλών Synergistic Processing Units (SPUs).

Επίσης, η σχεδίαση του συστήματος της μνήμης παρέχει έναν μεγάλο βαθμό από ταυτόχρονες μεταφορές δεδομένων, οι οποίες πραγματοποιούνται υπό τον έλεγχο της εφαρμογής για την υποστήριξη μεγαλύτερης απόδοσης του συστήματος μνήμης. Με αυτά τα χαρακτηριστικά, η εκάστοτε εφαρμογή μπορεί να έχει τον πλήρη έλεγχο των διαθέσιμων πόρων της αρχιτεκτονικής ώστε να επιτευχθεί η μέγιστη δυνατή απόδοση.

Αντιμετώπιση των κύριων περιοριστικών παραγόντων της απόδοσης

Η απόδοση των σύγχρονων επεξεργαστών καθορίζεται από τρεις σημαντικούς παράγοντες: την συχνότητα λειτουργίας, την καθυστέρηση πρόσβασης στην μνήμη και την καταναλισκόμενη ισχύ. Αυτοί οι παράγοντες συχνά αναφέρονται και ως *frequency wall, memory wall* και *power wall*. Η αρχιτεκτονική του *Cell* είναι τέτοια ώστε να αντιμετωπίζει επιτυχώς αυτούς του παράγοντες.

Frequency Wall

Οι συμβατικοί επεξεργαστές απαιτούν την αύξηση του βάθους του *pipeline* ώστε να επιτύχουν υψηλές συχνότητες λειτουργίας και να αυξήσουν την απόδοσή τους. Όμως, η τεχνική αυτή έχει φθάσει στο σημείο όπου επιπλέον αύξηση οδηγεί σε μειούμενα οφέλη (*law of diminishing effects*) - ακόμη και σε αρνητικές επιπτώσεις σε περίπτωση που η κατανάλωση ισχύος ληφθεί υπόψη.

Εξειδικεύοντας την λειτουργία του PPE και του SPE για εργασίες όπου κυριαρχούν οι εντολές ελέγχου (*control-intensive tasks*) και για εργασίες όπου κυριαρχούν οι εντολές υπολογισμού (*data-intensive tasks*) αντίστοιχα, η αρχιτεκτονική στην οποία βασίζεται ο *Cell* επιτρέπει τον αποδοτικό σχεδιασμό αυτών των στοιχείων για λειτουργία σε υψηλές συχνότητες χωρίς υπερβολική επιβάρυνση.

Το PPE επιτυγχάνει υψηλή αποδοτικότητα βελτιστοποιώντας την εκτέλεση και των δύο διαθέσιμων νημάτων, αντί ενός, ενώ το SPE επιτυγχάνει υψηλή αποδοτικότητα με χρήση του μεγάλου φακέλου καταχωρητών, ο οποίος επιτρέπει την ταυτόχρονη εκτέλεση αρκετών εντολών χωρίς την επιβάρυνση από τεχνικές όπως *register renaming* ή *out-of-order execution*. Επίσης, η χρήση των ασύγχρονων εντολών DMA επιτρέπει την πρόσβαση στην μνήμη πολλαπλών εντολών χωρίς την επιβάρυνση από τον μηχανισμό του *speculation* που υπάρχει στους συμβατικούς, σύγχρονους επεξεργαστές.

Memory Wall

Οι σύγχρονοι πολυ-επεξεργαστές χαρακτηρίζονται από την μεγάλη καθυστέρηση για την πρόσβαση στην κύρια μνήμη, η οποία προσεγγίζει την τιμή των χιλίων κύκλων μηχανής. Αυτό έχει ως

αποτέλεσμα, η απόδοση των εφαρμογών να εξαρτάται από την μεταφορά δεδομένων από και προς την κύρια μνήμη. Η αντιδραστική, *reactive*, φύση της κρυφής μνήμης δεν επιφέρει κάποια βελτίωση και πλέον ο μεταγλωττιστής, ή ακόμη και ο προγραμματιστής της εφαρμογής, επιφορτίζεται με την ρητή μεταφορά των δεδομένων.

Για την αντιμετώπιση αυτής της καθυστέρησης, τα SPEs του επεξεργαστή *Cell* υιοθετούν δύο τεχνικές:

- Ιεραρχία μνήμης τριών επιπέδων (φάκελος καταχωρητών, τοπική μνήμη, κύρια μνήμη)
- Ασύγχρονες εντολές DMA

Αυτοί οι μηχανισμοί επιτρέπουν στον προγραμματιστή της εφαρμογής να επικαλύψει την καθυστέρηση πρόσβασης στην κύρια μνήμη με την ταυτόχρονη δρομολόγηση υπολογισμών και μεταφοράς δεδομένων. Ενδεικτικά αναφέρεται ότι είναι δυνατή η υποστήριξη 128 ταυτόχρονων μεταφορών, αριθμός ο οποίος υπερβαίνει τον αριθμό των ταυτόχρονων μεταφορών που υποστηρίζονται από τους συμβατικούς επεξεργαστές κατά έναν παράγοντα ίσο με είκοσι.

Power-limitation Wall

Η απόδοση των μικροεπεξεργαστών εξαρτάται σε μεγάλο βαθμό από την κατανάλωση ισχύος και όχι από τους διαθέσιμους πόρους, όπως *transistors* και καλωδιώσεις. Επομένως, ο μόνος τρόπος για την αύξηση της απόδοσης είναι η αύξηση της αποδοτικότητας στην κατανάλωση ισχύος με την ταυτόχρονη αύξηση της συχνότητας λειτουργίας.

Μία μέθοδος για την επίτευξη αποδοτικότητας στην κατανάλωση ισχύος είναι ο διαχωρισμός μεταξύ επεξεργαστών που είναι βελτιστοποιημένοι για την εκτέλεση *control-intensive tasks* και επεξεργαστών που είναι βελτιστοποιημένοι για την εκτέλεση *data-intensive tasks*.

Όπως αναφέρθηκε και παραπάνω, ο επεξεργαστής *Cell* διαθέτει έναν επεξεργαστή, το PPE, βελτιστοποιημένο για την εκτέλεση του λειτουργικού συστήματος και την εκτέλεση *control-intensive* λειτουργιών και οχτώ επεξεργαστές, τα SPEs, για την εκτέλεση *data-intensive* λειτουργιών. Η αρχιτεκτονική των SPEs είναι αρκετά απλή καθώς δεν υπάρχουν μηχανισμοί όπως *branch prediction, out-of-order execution, speculative execution* και *register renaming*, οι οποίοι υπάρχουν σε άλλους σύγχρονους επεξεργαστές και συμβάλλουν στην επιτάχυνση της εκτέλεσης των προγραμμάτων. Επομένως, ο αριθμός των *transistors* που εξοικονομείται από την απουσία αυτών αφιερώνεται για την εκτέλεση υπολογισμών καθιστώντας τα SPEs πολύ αποδοτικά.

3.2 Η Αρχιτεκτονική του Επεξεργαστή Cell

Το πρώτο πρωτότυπο του *Cell* ήταν υλοποιημένο σε τεχνολογία *90nm SOI*, αποτελούνταν από 234 εκατομμύρια *transistors*, είχε κατανάλωση ίση με 60-80 W και τάση τροφοδοσίας ίση με 1.1 V ενώ λειτουργούσε σε συχνότητα ίση με *3.2 GHz*. Η υλοποίηση του πρωτοτύπου παρουσιάζεται στο σχήμα 3.2 [65] ενώ ένα ενδεικτικό σχήμα για την δομή του *Cell* παρουσιάζεται στο σχήμα 3.3.

Power Processing Element (PPE)

Το Power Processing Element (PPE), όπως υποδηλώνει και το όνομα αυτού, αποτελεί έναν επεξεργαστή Reduced Instruction Set Computer (RISC) που υποστηρίζει την *64-bit* αρχιτεκτονική *PowerPC* με τις SIMD επεκτάσεις αυτές, γνωστές και ως επεκτάσεις *AltiVec*. Το PPE υποστηρίζει την ταυτόχρονη εκτέλεση δύο νημάτων, επιτρέποντας έτσι την ταυτόχρονη διεκπεραίωση δύο

λειτουργιών. Στον επεξεργαστή *Cell*, το PPE εκτελεί το λειτουργικό σύστημα και, στην πλειονότητα των περιπτώσεων, εκτελεί και τον κώδικα που ελέγχει και συντονίζει την λειτουργία των SPEs. Το σχήμα 3.4 παρουσιάζει την αρχιτεκτονική του PPE.

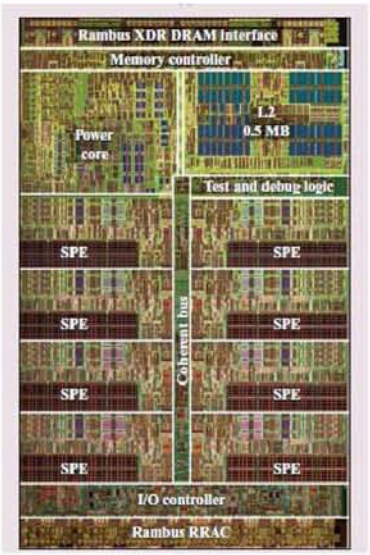
Οι κύριες υπομονάδες που αποτελούν το PPE είναι η μονάδα PowerPC Processing Unit (PPU) και το υποσύστημα PowerPC processor Storage Subsystem (PPSS), το οποίο αποτελεί το υποσύστημα για την διαχείριση της μνήμης. Η μονάδα PPU αποτελείται από τις ακόλουθες επιμέρους μονάδες:

- Instruction Unit (IU): Αυτή η υπομονάδα εκτελεί την προσκόμιση και την αποκωδικοποίηση των εντολών. Έπειτα, κατευθύνει αυτές στην κατάλληλη υπομονάδα για την εκτέλεση της εντολής. Εμπεριέχει την υπομονάδα Branch Unit (BRU) και μία κρυφή μνήμη εντολών (*L1 instruction cache*) μεγέθους $32KB$.

- Load and Store Unit (LSU): Η υπομονάδα LSU λαμβάνει αιτήσεις από την μνήμη, τις οποίες προωθεί στο υποσύστημα PPSS και εμπεριέχει μία κρυφή μνήμη δεδομένων (*L1 data cache*) μεγέθους $32KB$.

- Vector/Scalar Unit (VSU): Αυτή η υπομονάδα περιλαμβάνει την μονάδα Floating Point Unit (FPU), η οποία εκτελεί πράξεις σε βαθμωτές μεταβλητές ακεραίων κινητής υποδιαστολής, και την μονάδα Vector/SIMD Multimedia Extension Unit (VXU), η οποία εκτελεί πράξεις σε διανύσματα (*vectors*) τύπου ακεραίων κινητής υποδιαστολής. Τα διανύσματα είναι μεγέθους $128-bits$ και σε αυτή την μονάδα εκτελούνται και οι πράξεις που ορίζονται από τις επεκτάσεις *Vector/SIMD Multimedia Extensions (AltiVec)*.

- Fixed-Point Unit (FXU): Η υπομονάδα FXU εκτελεί πράξεις σε ακεραίους όπως λογικές ή αριθμητικές πράξεις.



Σχήμα 3.2: Εικόνα από ένα chip του επεξεργαστή *Cell*.

- Memory Management Unit (MMU): Η υπομονάδα MMU διαχειρίζεται το σύστημα εικονικής μνήμης του *Cell* και είναι υπεύθυνη για την μετάφραση των *ενεργών διευθύνσεων* (*effective addresses*) σε *εικονικές* και *πραγματικές διευθύνσεις*.

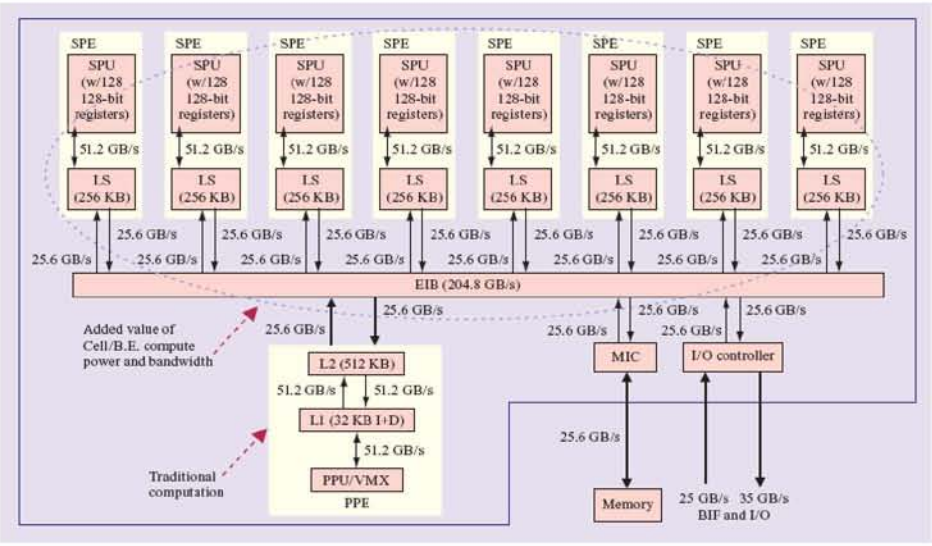
- Branch Unit (BRU): Η υπομονάδα BRU επεξεργάζεται τις εντολές διακλαδώσεων που υπάρχουν στο κώδικα της εκάστοτε εφαρμογής.

Η ιεραρχία κρυφής μνήμης του PPE αποτελείται από μία *Level-1* κρυφή μνήμη εντολών, με μέγεθος ίσο με $32\ KB$, μία *Level-1* κρυφή μνήμη δεδομένων με μέγεθος ίσο με $32\ KB$, και μία ενοποιημένη *Level-2 write-back* μνήμη εντολών και δεδομένων με μέγεθος ίσο με $512\ KB$. Οι δύο *Level-1* μνήμες υποστηρίζουν συνεκτική πρόσβαση στα δεδομένα, με κάθε φόρτωση να έχει μέγεθος ίσο με 16 ή 32 bytes.

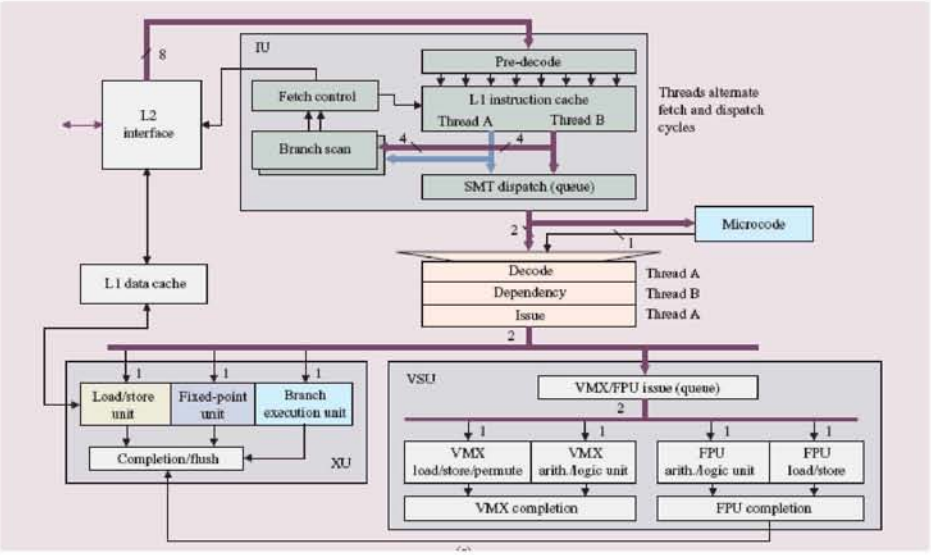
Η αρχιτεκτονική του PPE υποστηρίζει το σύνολο εντολών της αρχιτεκτονικής *PPC 970* και το πλήρες σύνολο των εντολών *AltiVec/VMX*. Όπως κάθε αρχιτεκτονική τύπου *RISC*, η πλειονότητα των εντολών εκτελούν πράξεις μεταξύ καταχωρητών ενώ συγκεκριμένες εντολές χρησιμοποιούνται για πρόσβαση στην μνήμη.

Επίσης, υποστηρίζεται ένα μικρό σύνολο διαφορετικών κωδικοποιήσεων για τις διαθέσιμες εντολές ενώ όλες οι διαφορετικές κωδικοποιήσεις έχουν το ίδιο μέγεθος. Το σύνολο καταχωρητών που είναι διαθέσιμο στον προγραμματιστή αποτελείται από 32 καταχωρητές γενικού σκοπού με μέγεθος ίσο με $64\ bits$, 32 καταχωρητές με μέγεθος ίσο με $64\ bits$ για την αποθήκευση αριθμών κινητής υποδιαστολής που ακολουθούν το πρότυπο *IEEE-754* και 32 διανυσματικούς καταχωρητές μεγέθους ίσο με $128\ bits$ για την αποθήκευση δεδομένων που χρησιμοποιούνται σε διανυσματικές πράξεις.

Όπως αναφέρθηκε παραπάνω και παρουσιάζεται και στο σχήμα 3.4, το PPE αποτελεί έναν επεξεργαστή *RISC*, με δύο *hardware threads* ενώ δεν υποστηρίζεται η εκτέλεση εντολών εκτός σειράς. Σε κάθε κύκλο ρολογιού εισέρχονται προς επεξεργασία δύο εντολές, οι οποίες κατευθύνονται προς την κατάλληλη υπομονάδα εκτέλεσης όπως αυτή καθορίζεται από την υπομονάδα IU.



Σχήμα 3.3: Δομικό διάγραμμα του επεξεργαστή *Cell*.



Σχήμα 3.4: Δομικό διάγραμμα του *PPE*.

Σε περίπτωση που δεν υπάρχουν εξαρτήσεις μεταξύ των εντολών, αυτές εκτελούνται παράλληλα οπότε αυξάνεται η ικανότητα διεκπεραίωσης του επεξεργαστή. Επίσης, κατά την φάση της αποκωδικοποίησης ελέγχεται κατά πόσο η τρέχουσα εντολή αποτελεί μία *microcoded* εντολή. Η εκτέλεση τέτοιου είδους εντολών απαιτεί τον διαχωρισμό αυτών σε επιμέρους, απλούστερες εντολές. Γι' αυτό το στάδιο απαιτούνται 11 κύκλοι ρολογιού οπότε οι *microcoded* εντολές θα πρέπει να αποφεύγονται λόγω της μεγάλης καθυστέρησης που εισάγουν.

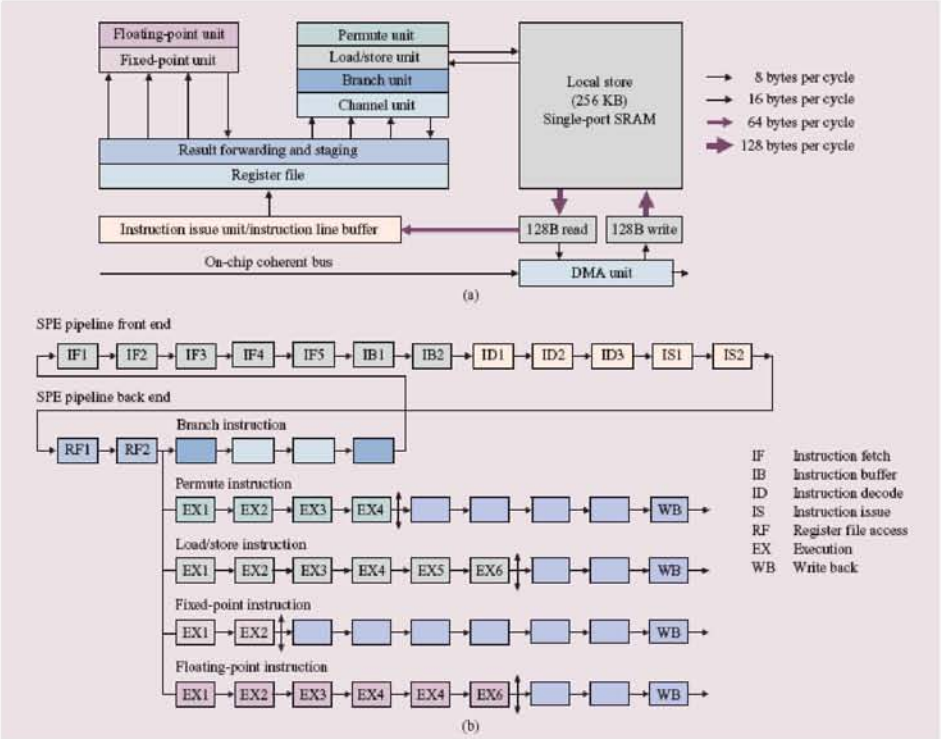
Τέλος, ένα σημαντικό τμήμα της αρχιτεκτονικής κάθε επεξεργαστή είναι η πρόβλεψη διακλαδώσεων (*branch prediction*) λόγω της μεγάλης καθυστέρησης που εισάγει μία λανθασμένη πρόβλεψη, η οποία ανέρχεται σε 23 κύκλους ρολογιού λόγω της επιβάρυνσης που εισάγεται από την διαδικασία αδειάσματος (*flushing*) του *pipeline*. Το PPE υλοποιεί την πρόβλεψη διακλαδώσεων όπως ορίζεται από την αρχιτεκτονική *PPC 970*. Η αρχιτεκτονική συνόλου εντολών του *PowerPC* εμπεριέχει μία πληθώρα εντολών διακλάδωσης, πολλές από τις οποίες χρησιμοποιούνται ώστε να παρέχουν υποδείξεις στην υπομονάδα IU ως προς το ποιο είναι το πιο πιθανό αποτέλεσμα της διακλάδωσης. Ένα χαρακτηριστικό παράδειγμα εντολής υψηλού επιπέδου είναι η εντολή *__builtin_expect*. Αυτού του είδους οι εντολές αναγνωρίζονται από τους αντίστοιχους μεταγλωττιστές οι οποίοι εισάγουν τις κατάλληλες εντολές *assembly*.

Όσον αφορά στον προγραμματισμό του PPE, αυτός πραγματοποιείται χρησιμοποιώντας τις υψηλού επιπέδου γλώσσες *C/C++* ενώ παρέχεται και ένα σύνολο εσωτερικών εντολών[1], όπως για παράδειγμα αυτές που υλοποιούν τις επεκτάσεις *AltiVec/VMX*.

Synergistic Processing Element (SPE)

Παρόλη την υπολογιστική δύναμη του PPE, η ασυνήθης υπολογιστική δύναμη του επεξεργαστή *Cell* πηγάζει από το Synergistic Processing Element (SPE). Σε αντίθεση με την αρχιτεκτονική γενικού σκοπού του PPE, η αρχιτεκτονική του SPE σχεδιάστηκε με κύριο γνώμονα την εκτέλεση διανυσματικών υπολογισμών με πολύ μεγάλη ταχύτητα. Το *datapath* της αρχιτεκτονικής έχει εύρος

[1]Οι εντολές αυτές ονομάζονται *intrinsics* και αντιστοιχούν σε μία ή περισσότερες εντολές *assembly*.



Σχήμα 3.5: Δομικό διάγραμμα του *SPE*.

ίσο με 128 $bits$ ενώ το *pipeline* είναι *dual-issue*, δηλαδή υποστηρίζεται η εκτέλεση δύο εντολών σε κάθε κύκλο ρολογιού. Όμως, η εκτέλεση δύο εντολών είναι δυνατή μόνο στην περίπτωση που δεν υπάρχουν εξαρτήσεις μεταξύ αυτών ενώ σε κάθε *pipeline* εκτελούνται εντολές από συγκεκριμένες κλάσεις εντολών. Καθένα από τα οχτώ SPEs αποτελείται από την υπομονάδα SPU και τον *ελεγκτή ροής μνήμης* (Memory Flow Controller (MFC)). Το δομικό διάγραμμα του SPE όπως επίσης και τα στάδια του *pipeline* για κάθε μία από τις διαθέσιμες κλάσεις εντολών παρουσιάζονται στο σχήμα 3.5.

Οι κύριες υπομονάδες που αποτελούν κάθε SPE είναι οι ακόλουθες:

- SPU Control Unit (SCN): Η μονάδα αυτή προσκομίζει και διανέμει τις εντολές στις αντίστοιχες μονάδες εκτέλεσης εντολών. Επίσης εκτελεί τις διάφορες εντολές ελέγχου, όπως για παράδειγμα εντολές διακλαδώσεων.

- SPU Even Fixed-Point Unit (SFX): Σε αυτή την μονάδα εκτελούνται λογικές και αριθμητικές εντολές, εντολές σύγκρισης όπως επίσης και εντολές για αντιστροφή αριθμών κινητής υποδιαστολής.

- SPU Odd Fixed-Point Unit (SFS): Η μονάδα αυτή είναι υπεύθυνη για την εκτέλεση πράξεων ολίσθησης, περιστροφής και ανακατέματος (*shuffling*).

- SPU Floating-Point Unit (SFP): Στην εν λόγω μονάδα εκτελούνται οι αριθμητικές πράξεις σε ακεραίους κινητής υποδιαστολής, απλής ή διπλής ακρίβειας, πράξεις πολλαπλασιασμού ακεραίων όπως επίσης και διάφορες πράξεις μετατροπής ακεραίων.

Τύπος Δεδομένων	Περιεχόμενα Τύπου Δεδομένων	Μέγεθος (bytes)
boolean	True/False	1
char	Χαρακτήρας ASCII	1
unsigned char	Ακέραιος (0-255)	1
short	Προσημασμένος ακέραιος τύπου short	2
unsigned short	Μη προσημασμένος ακέραιος τύπου short	2
int	Προσημασμένος Ακέραιος	4
unsigned int	Μη προσημασμένος ακέραιος	4
float	Ακέραιος κινητής υποδιαστολής, απλής ακρίβειας	4
double	Ακέραιος κινητής υποδιαστολής, διπλής ακρίβειας	8
long long / long long int	Προσημασμένος ακέραιος τύπου long long	8
unsigned long long / unsigned long long int	Μη προσημασμένος ακέραιος τύπου long long	8
qword	Εξαρτώνται από την εντολή	16

Πίνακας 3.1: Τύποι δεδομένων βαθμωτών μεταβλητών στα *SPUs*

- SPU Load/Store Unit (SLS): Η μονάδα *SLS* είναι υπεύθυνη για την διαχείριση των διαφόρων εντολών που αφορούν στην μνήμη. Εδώ εκτελούνται εντολές φόρτωσης και αποθήκευσης στην τοπική μνήμη όπως επίσης και εντολές για αιτήσεις DMA στην τοπική μνήμη.

- SPU Channel and DMA Unit (SSC): Αυτή η μονάδα εκτελεί τις διάφορες λειτουργίες επικοινωνίας με τον *ελεγκτή ροής μνήμης* (MFC) και διαχειρίζεται τις μεταφορές δεδομένων μέσω του μηχανισμού DMA.

Τα χαρακτηριστικά που διαφοροποιούν την αρχιτεκτονική του SPE από την αρχιτεκτονική του PPE είναι η απουσία κρυφής μνήμης (*cache hierarchy*) και εικονικής μνήμης, η απουσία μονάδων που επεξεργάζονται βαθμωτές μεταβλητές, καθώς όλες οι πράξεις πραγματοποιούνται σε διανύσματα, και τα δύο διαφορετικά *pipelines*.

Κάθε SPE αποτελείται από μία τοπική μνήμη, αποκλειστικής πρόσβασης και μεγέθους $256\ KB$ (LS), η οποία είναι οργανωμένη σε τέσσερεις *SRAM arrays* μεγέθους $64\ KB$. Η μνήμη έχει μόνο μία θύρα ανάγνωσης/εγγραφής και γι' αυτό οι λειτουργίες πρόσβασης σε αυτή είναι *fully pipelined* ώστε να επιτυγχάνεται ταχύτερη πρόσβαση σε αυτή. Το εύρος ζώνης της μνήμης είναι ίσο με $16\ bytes$ ανά κύκλο ενώ εντολές τύπου *DMA* μεταφέρουν από και προς την μνήμη δεδομένα που είναι οργανωμένα σε μονάδες μεγέθους $1024\ bytes$.

΄Ενα ιδιαίτερο χαρακτηριστικό της μνήμης είναι ότι η πρόσβαση σε αυτή ξεκινά από διευθύνσεις που είναι πολλαπλάσιες των $16\ bytes$ (quadword alignment). Σε περίπτωση που αυτό δεν ισχύει, εκτελούνται επιπλέον πράξεις για την ικανοποίηση των περιορισμών ευθυγράμμισης. ΄Οπως είναι εμφανές, για την ικανοποίηση των περιορισμών πρέπει να ληφθεί ειδική μέριμνα από τον προγραμματιστή της εφαρμογής προκειμένου να μην επιβαρύνεται η εκτέλεση της εφαρμογής με το κόστος που συνεπάγονται οι προαναφερθείσες πράξεις ευθυγράμμισης.

Καθώς η μνήμη έχει μόνο μία θύρα εγγραφής/ανάγνωσης, όλες οι εντολές προσπέλασης στην μνήμη, όπως εντολές φόρτωσης και αποθήκευσης, συναγωνίζονται για την πρόσβαση σε αυτή την θύρα. Για την επίλυση αυτών των συγκρούσεων, επιβάλλεται ένα σχήμα προτεραιοτήτων μεταξύ αυτών των εντολών. Οι εντολές του μηχανισμού DMA έχουν την μεγαλύτερη προτεραιότητα, μετά ακολουθούν οι εντολές φόρτωσης και αποθήκευσης ενώ την χαμηλότερη προτεραιότητα έχουν οι

Διανυσματικός Τύπος Δεδομένων	Πλήθος Στοιχείων του Καταχωρητή
vector unsigned char	16 μη προσημασμένοι χαρακτήρες, μεγέθους 8-bit
vector signed char	16 προσημασμένοι χαρακτήρες, μεγέθους 8-bit
vector unsigned short	8 μη προσημασμένοι ακέραιοι τύπου short, μεγέθους 16-bit
vector signed short	8 προσημασμένοι ακέραιοι τύπου short, μεγέθους 16-bit
vector pixel	8 μη προσημασμένες μεταβλητές τύπου halfword, μεγέθους 16-bit
vector unsigned int	4 μη προσημασμένοι ακέραιοι
vector signed int	4 προσημασμένοι ακέραιοι
vector float	4 μεταβλητές κινητής υποδιαστολής, απλής ακρίβειας
vector unsigned long long	2 μη προσημασμένες μεταβλητές τύπου long long, μεγέθους 64-bit
vector signed long long	2 προσημασμένες μεταβλητές τύπου long long, μεγέθους 64-bit
vector double	2 μεταβλητές κινητής υποδιαστολής, διπλής ακρίβειας

Πίνακας 3.2: Τύποι δεδομένων διανυσμάτων στα *SPUs*

προσπελάσεις για την προσκόμιση εντολών.

Ο φάκελος καταχωρητών σε κάθε SPE έχει μέγεθος ίσο με 128 καταχωρητές, όπου κάθε καταχωρητής έχει μήκος ίσο με 128 $bits$. Οι διαθέσιμοι τύποι δεδομένων για τις βαθμωτές μεταβλητές και τις διανυσματικές μεταβλητές σε κάθε SPU, όπως επίσης και το μέγεθος αυτών αναφέρονται στους πίνακες 3.1 και 3.2. Η αποθήκευση βαθμωτών μεταβλητών στους διαθέσιμους καταχωρητές δίνει την δυνατότητα για εκτέλεση πράξεων μεταξύ βαθμωτών μεταβλητών, παρόλο που τα SPEs έχουν σχεδιαστεί κυρίως για πράξεις μεταξύ διανυσμάτων. Για την εκτέλεση αυτών των πράξεων, οι βαθμωτές μεταβλητές πρέπει να αποθηκευθούν σε μία προκαθορισμένη θέση του αντίστοιχου καταχωρητή, η οποία ονομάζεται *preferred slot*.

Η εικόνα 3.6 παρουσιάζει αυτή την θέση για τους διαθέσιμους τύπους δεδομένων. Επί παραδείγματι, μία βαθμωτή μεταβλητή μήκους ίσου με 2 bytes αποθηκεύεται στις θέσεις 2 έως 3 του καταχωρητή[2]. Αυτός ο περιορισμός συνεπάγεται και επιπλέον κόστος σε περίπτωση που εκτελούνται πράξεις μεταξύ βαθμωτών μεταβλητών. Καθώς η φόρτωση από την μνήμη γίνεται σε μονάδες των 128 bits, για την αποθήκευση των βαθμωτών μεταβλητών στο *preferred slot*, ενδέχεται να απαιτούνται επιπλέον πράξεις ολισθήσεις των περιεχομένων των καταχωρητών. Αυτές οι πράξεις δημιουργούν εξαρτήσεις εντολών που έχουν ως αποτέλεσμα την ύπαρξη παγωμάτων (*stalls*) στο *pipeline*, τα οποία οδηγούν σε μείωση της επιτεύξιμης απόδοσης και αύξηση του χρόνου εκτέλεσης του προγράμματος.

Όσον αφορά στην αρχιτεκτονική του *pipeline* του SPU, όπως αναφέρθηκε και παραπάνω, αυτή αποτελείται από δύο *pipelines*, τα οποία χωρίζονται επιμέρους σε άρτιο και περιττό *pipeline*. Κατά κύριο λόγο, στο άρτιο *pipeline* εκτελούνται εντολές ακεραίων σταθερής και κινητής υποδιαστολής ενώ στο περιττό *pipeline* εκτελούνται εντολές φόρτωσης από την μνήμη όπως επίσης και

[2]Η διευθυνσιοδότηση των θέσεων των καταχωρητών γίνεται βάσει *big-endian* διάταξης

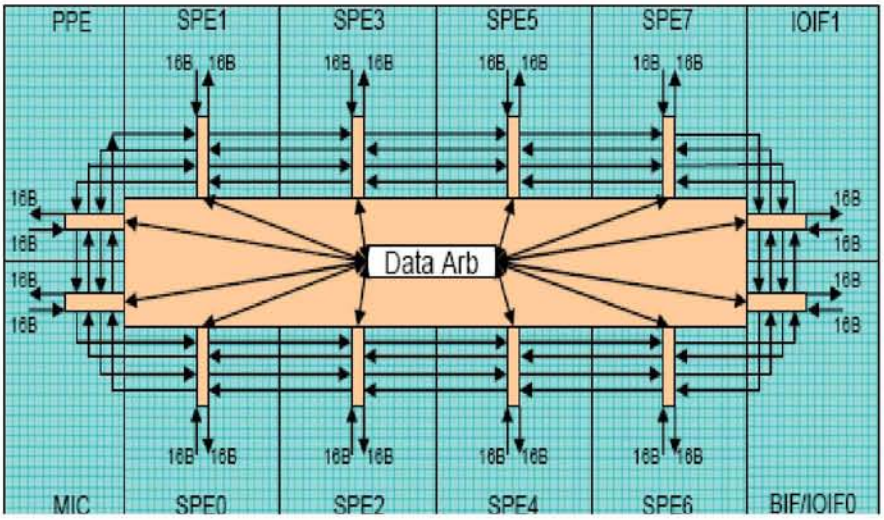
Σχήμα 3.6: Αποθήκευση βαθμωτών μεταβλητών στους καταχωρητές.

εντολές μετάθεσης περιεχομένων της μνήμης. Με χρήση αυτού του μηχανισμού, καθίσταται δυνατή η εκτέλεση δύο εντολών σε κάθε κύκλο μηχανής, υπό την προϋπόθεση ότι υπάρχουν αρκετές διαθέσιμες εντολές και δεν υπάρχουν εξαρτήσεις μεταξύ των εντολών. Κατ' αυτό τον τρόπο, είναι δυνατή η εκμετάλλευση του εγγενούς παραλληλισμού μεταξύ των εντολών (ILP) που υπάρχει στις εφαρμογές.

Η αρχιτεκτονική των SPEs, με τα προαναφερθέντα χαρακτηριστικά, επιτρέπει την εκτέλεση 6.4 Giga Floating-Point operations per second (GFlops) για πράξεις με αριθμούς κινητής υποδιαστολής διπλής ακρίβειας και 25.6 GFlops για πράξεις με αριθμούς κινητής υποδιαστολής απλής ακρίβειας. Όπως παρατηρούμε, η εκτέλεση πράξεων κινητής υποδιαστολής διπλής ακρίβειας δεν είναι τόσο αποδοτική. Αυτό οφείλεται στο ότι οι εν λόγω πράξεις δεν είναι πλήρως *pipelined* ενώ κατά την εκτέλεση τέτοιων πράξεων δεν είναι δυνατή η χρήση του *dual-issue* από άλλες εντολές. Η πιο πρόσφατη υλοποίηση της αρχιτεκτονικής CBEA, ο επεξεργαστής *PowerXCell 8i*[3] [63], ενσωματώνει μία βελτιστοποιημένη μονάδα εκτέλεσης υπολογισμών κινητής υποδιαστολής διπλής ακρίβειας στα SPEs, με αποτέλεσμα την αύξηση της υπολογιστικής ικανότητας του κάθε SPE σε 12.8 GFlops.

Ο ελεγκτής ροής μνήμης (MFC) αποτελεί το μέσο για την επικοινωνία του SPU με την εξωτερική μνήμη, άλλα επεξεργαστικά στοιχεία όπως επίσης και συσκευές εισόδου/εξόδου. Για την επικοινωνία χρησιμοποιείται το Element Interconnect Bus (EIB), το οποίο περιγράφεται στην ενότητα 3.2. Ο ελεγκτής υποστηρίζει τον μηχανισμό DMA, μέσω του οποίου καθίσταται δυνατή η μεταφορά μεγάλου όγκου δεδομένων και εντολών από και προς την τοπική μνήμη. Εκτός του μηχανισμού DMA, ο ελεγκτής ροής μνήμης υποστηρίζει τον μηχανισμό των σημάτων (*signals*) και των γραμματοκιβωτίων (*mailboxes*), οι οποίοι είναι δύο επιπλέον μηχανισμοί που προσφέρονται για την επικοινωνία μεταξύ των επεξεργαστικών στοιχείων (*inter-processor communication*). Αυτοί οι μηχανισμοί χρησιμοποιούνται για την ανταλλαγή μηνυμάτων μικρού μεγέθους μεταξύ των επεξεργαστών, όπως για παράδειγμα σήματα περάτωσης εργασίας ή διευθύνσεις δεδομένων.

[3]Ο επεξεργαστής PowerXCell 8i είναι αυτός που χρησιμοποιείται στα συστήματα *QS22 Blades* της *IBM*



Σχήμα 3.7: Δομικό διάγραμμα του *EIB*.

Element Interconnect Bus (EIB)

Το Element Interconnect Bus (EIB) είναι ένας κυκλικός δίαυλος, ο οποίος αποτελείται από τέσσερεις επιμέρους δακτυλίους με εύρος ίσο με 16 $bytes$, όσο είναι και το μήκος μίας γραμμής στην LS των SPEs, και έχει συχνότητα λειτουργίας ίση με το μισό της συχνότητας λειτουργίας του επεξεργαστή. Οι δύο δακτύλιοι μεταφέρουν δεδομένα με φορά αυτή των δεικτών του ρολογιού ενώ οι άλλοι δύο μεταφέρουν δεδομένα με αντίθετη φορά. Το EIB χρησιμοποιείται για την μεταφορά δεδομένων από το PPE και την *L2 cache* αυτού προς τα SPEs και αντίστροφα. Επίσης, συνδέεται με την διεπαφή του ελεγκτή μνήμης (*Memory Interface Controller*) και με το *FlexIO* για εξωτερική επικοινωνία. Το ακόλουθο σχήμα παρουσιάζει την αρχιτεκτονική του EIB.

Κάθε επεξεργαστής του συστήματος υποστηρίζει την ταυτόχρονη λήψη και αποστολή δεδομένων μέσω του EIB. Το μέγιστο εύρος ζώνης που υποστηρίζεται είναι ίσο με 96 $bytes$ και κάθε δακτύλιος μπορεί να υποστηρίξει ταυτόχρονα μέχρι τρεις αιτήσεις DMA υπό την προϋπόθεση ότι δεν υπάρχει επικάλυψη μεταξύ αυτών. Τέλος, ο μέγιστος αριθμός των εκκρεμών αιτήσεων που υποστηρίζεται είναι ίσος με 128 αιτήσεις για μεταφορά δεδομένων από την κύρια μνήμη προς τα SPEs και αντίστροφα.

Μνήμη και Είσοδος/Έξοδος Δεδομένων

Ο ελεγκτής διεπαφής μνήμης (*MIC*) του επεξεργαστή *Cell* υποστηρίζει μία ή δύο μνήμες υψηλής ταχύτητας τύπου Rambus Extreme Data Rate (XDR). Το εύρος κάθε καναλιού ισούται με 12.8 GB/s οπότε το μέγιστο εύρος ζώνης που δύναται να υποστηριχθεί είναι ίσο με 25.6 GB/s. Η συνολική μνήμη που μπορεί να υποστηριχθεί, εξαρτάται από την διαμόρφωση του συστήματος, και κυμαίνεται από 64 MB έως 64 GB μνήμης *XDR DRAM*.

Η μονάδα Broadband Engine Interface (BEI) του επεξεργαστή *Cell* είναι μία διεπαφή που χρησιμοποιείται από τους επεξεργαστές του συστήματος για την επικοινωνία με τις συσκευές *Εισόδου-Εξόδου* του συστήματος. Αυτή αποτελείται από την μονάδα Broadband Interface Controller (BIC), την μονάδα I/O Controller (IOC) και την μονάδα Internal Interrupt Controller (IIC).

Η μονάδα BEI υποστηρίζει δύο διεπαφές *Rambus FlexIO*. Η πρώτη διεπαφή αποτελεί μία μη συνεκτική διεπαφή για *Είσοδο/Έξοδο* και μπορεί να χρησιμοποιηθεί για την σύνδεση συσκευών όπως κάρτες γραφικών ή κάρτες ήχου. Η δεύτερη διεπαφή μπορεί να υποστηρίξει τόσο συνεκτική όσο και μη συνεκτική ανταλλαγή δεδομένων και μπορεί να χρησιμοποιηθεί για την επέκταση του BEI και την σύνδεση με έναν δεύτερο επεξεργαστή *Cell BE*, ώστε να είναι δυνατή η αποστολή και λήψη δεδομένων μεταξύ αυτών. Το εύρος ζώνης που υποστηρίζεται ισούται με $76.8\ GB/s$.

3.3 Ανάπτυξη Εφαρμογών στον Επεξεργαστή Cell

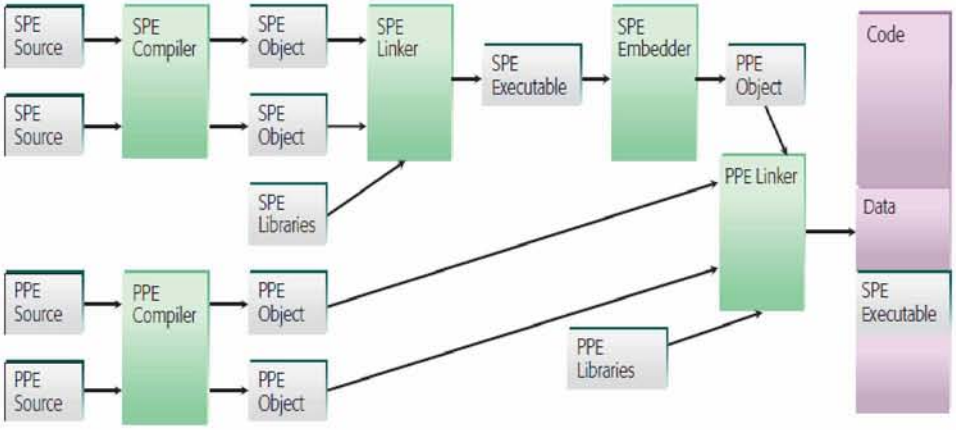
Η ιδιαίτερη αρχιτεκτονική του επεξεργαστή *Cell BE*, με την ύπαρξη ετερογενών επεξεργαστικών στοιχείων, εισάγει επιπλέον πολυπλοκότητα στην ανάπτυξη των εφαρμογών. Ήδη υπάρχοντα προγραμματιστικά μοντέλα χαμηλού επιπέδου μπορούν να χρησιμοποιηθούν αλλά θα πρέπει να τροποποιηθούν, ώστε να είναι δυνατή η εκτέλεση προγραμμάτων που κάνουν χρήση αυτών των μοντέλων στον επεξεργαστή *Cell*. Για την ανάπτυξη εφαρμογών για τον επεξεργαστή *Cell* διατίθεται από την *IBM* το περιβάλλον ανάπτυξης *Cell BE Software Development Kit (SDK)*. Το περιβάλλον επιτρέπει την ανάπτυξη εφαρμογών για τον επεξεργαστή σε διάφορες πλατφόρμες (*x86, x86-64, 64-bit PowerPC, Cell BE Blade Center*) και περιλαμβάνει:

- Toolchains για την ανάπτυξη προγραμμάτων τόσο για το PPE όσο και για το SPE, για καθεμία από τις υποστηριζόμενες πλατφόρμες.
- Βιβλιοθήκες και υποδείγματα κώδικα.
- Έναν εξομοιωτή συστήματος που περιλαμβάνει έναν επεξεργαστή *Cell*.
- Έναν πυρήνα του λειτουργικού συστήματος *Linux*.
- Ένα *IDE*, το οποίο βασίζεται στην πλατφόρμα *Eclipse*, και ενοποιεί τα *toolchains*, τους μεταγλωττιστές, τον προσομοιωτή και τα άλλα εργαλεία ανάπτυξης.
- Βοηθητικά Προγράμματα όπως το *spu-timing* το οποίο εκτελεί στατική ανάλυση του κώδικα, και το *FDPR-Pro*, το οποίο είναι εργαλείο για την βελτιστοποίηση ενός προγράμματος μέσω *feedback* από την εκτέλεση αυτού για έναν τυπικό φόρτο εργασίας.

Διαθέσιμοι Μεταγλωττιστές

Για την μεταγλώττιση προγραμμάτων που πρόκειται να εκτελεστούν στον επεξεργαστή *Cell* διατίθεται τόσο μία τροποποιημένη έκδοση του μεταγλωττιστή *GCC* από την *Sony* όσο και μία έκδοση του μεταγλωττιστή *XL C/C++* της *IBM*. Και οι δύο μεταγλωττιστές προορίζονται για την μεταγλώττιση προγραμμάτων για το PPE και το SPE.

Ο μεταγλωττιστής *IBM XL C/C++* έχει τροποποιηθεί ώστε να είναι δυνατή η εκμετάλλευση των δυνατοτήτων των επεξεργαστών της αρχιτεκτονικής CBE. Αυτός ο μεταγλωττιστής προσφέρει αυξημένες δυνατότητες, τόσο για αυτόματη όσο και για κατευθυνόμενη από τον χρήστη, παραλληλοποίηση και τεμαχισμό της εφαρμογής για την εκμετάλλευση των διαφόρων ειδών παραλληλισμού που υπάρχουν στις διάφορες εφαρμογές. Οι οδηγίες, *user directives*, που χρησιμοποιούνται από τον χρήστη για την επικοινωνία με τον επεξεργαστή βασίζονται στο *OpenMP* μοντέλο προγραμματισμού. Αυτή η προσέγγιση επιτρέπει στον προγραμματιστή την θεώρηση του συστήματος ως ένα σύστημα με κοινό χώρο διευθύνσεων μνήμης (*shared-memory address space*), όπου όλα τα δεδομένα που πρόκειται να προσπελαστούν εμπεριέχονται σε αυτό τον χώρο.



Σχήμα 3.8: Διαδικασία μεταγλώττισης προγραμμάτων.

Και οι δύο μεταγλωττιστές εμπεριέχουν ένα πλούσιο σύνολο *intrinsincs*, τα οποία μπορούν να χρησιμοποιηθούν σε συνδυασμό με τις γλώσσες προγραμματισμού *C/C++*. Αυτές οι επεκτάσεις καθιστούν τον προγραμματισμό για εκμετάλλευση των SIMD δυνατοτήτων των SPEs και του PPE αρκετά απλό, επιτρέποντας στον προγραμματιστή να έχει τον πλήρη έλεγχο τόσο των εντολών SIMD όσο και της διάταξης των δεδομένων, ενώ ο μεταγλωττιστής είναι υπεύθυνος για την δρομολόγηση των εντολών και την δέσμευση των κατάλληλων καταχωρητών. Το γεγονός αυτό προσφέρει στον προγραμματιστή το πλεονέκτημα του ελέγχου των μετασχηματισμών σε υψηλό επίπεδο, όπως για παράδειγμα το ξεδίπλωμα βρόχου (*loop unrolling*), ενώ ταυτόχρονα μπορεί να εκτελέσει και βελτιστοποιήσεις χαμηλού επιπέδου.

Η πιο προσφιλής πρακτική για την μεταγλώττιση, διασύνδεση και εκτέλεση μίας εφαρμογής στον επεξεργαστή παρουσιάζεται στο σχήμα 3.8. Αρχικά, πραγματοποιείται η μεταγλώττιση και η διασύνδεση του εκτελέσιμου για τα SPEs. Έπειτα, το εκτελέσιμο ενσωματώνεται σε ένα *object file* το οποίο θα προσπελαστεί από το PPU. Το παραγόμενο αρχείο βασίζεται στο format CBE Embedded SPE Object Format (CESOF), το οποίο είναι ένα *relocatable* αρχείο που προορίζεται για το PPU. Τέλος, πραγματοποιείται η μεταγλώττιση του κώδικα για το PPU, η διασύνδεση με τις όποιες απαραίτητες βιβλιοθήκες όπως επίσης και με το *object file* του κώδικα για τα SPEs και η παραγωγή του τελικού εκτελέσιμου αρχείου.

Προγραμματισμός των SPEs

Οι περισσότερες από τις εφαρμογές που εκτελούνται στον επεξεργαστή *Cell* χρησιμοποιούν τα διαθέσιμα SPEs στα οποία αναθέτουν επιμέρους εργασίες προς εκτέλεση. Προγραμματιστικά μοντέλα όπως το μοντέλο *Streaming* ή το μοντέλο *Pipeline* μπορούν να χρησιμοποιηθούν ανάλογα με τα χαρακτηριστικά της εφαρμογής. Ωστόσο, ανεξάρτητα από το προγραμματιστικό μοντέλο που χρησιμοποιείται, το κυρίως νήμα εκτελείται στο PPE και είναι αυτό που δημιουργεί τα επιμέρους νήματα που θα εκτελεστούν στα SPEs. Τα προγράμματα που εκτελούνται στα SPEs είναι αυτόνομα και λαμβάνουν τις απαραίτητες παραμέτρους από το κυρίως νήμα που εκτελείται στο PPE. Η διαχείριση των νημάτων και της μεταφοράς των δεδομένων εξαρτάται από το προγραμματιστικό μοντέλο.

Η διαχείριση του κώδικα που εκτελείται στα SPEs πραγματοποιείται μέσω της βιβλιοθήκης

libspe2 (SPE runtime management library) και της βιβλιοθήκης *Pthread*. Η βιβλιοθήκη *libspe2* παρέχει ένα Application Programming Interface (API) χαμηλού επιπέδου το οποίο επιτρέπει στις εφαρμογές την πρόσβαση στα SPEs για την εκτέλεση κάποιων από τα νήματα της εφαρμογής. Η βιβλιοθήκη *Pthread* χρησιμοποιείται σε συνδυασμό με την βιβλιοθήκη *libspe2* για την δημιουργία των απαραίτητων νημάτων.

Στην γενική περίπτωση, ο έλεγχος των φυσικών πόρων των SPEs πραγματοποιείται από το λειτουργικό σύστημα και όχι από τις εφαρμογές. Οι εφαρμογές ελέγχουν και χρησιμοποιούν δομές δεδομένων σε λογισμικό οι οποίες ονομάζονται *SPE contexts*. Οι δομές αυτές αποτελούν μία λογική αναπαράσταση των SPEs και εμπεριέχουν όλη την πληροφορία που απαιτείται για την λογική περιγραφή του SPE ενώ η επεξεργασία αυτών πραγματοποιείται από την βιβλιοθήκη *libspe2* και έτσι παρέχεται η απαραίτητη αφαίρεση στον προγραμματιστή της εφαρμογής. Το λειτουργικό σύστημα είναι υπεύθυνο για την δρομολόγηση των *contexts* από όλες τις εφαρμογές βάσει των προτεραιοτήτων και των πολιτικών δρομολόγησης.

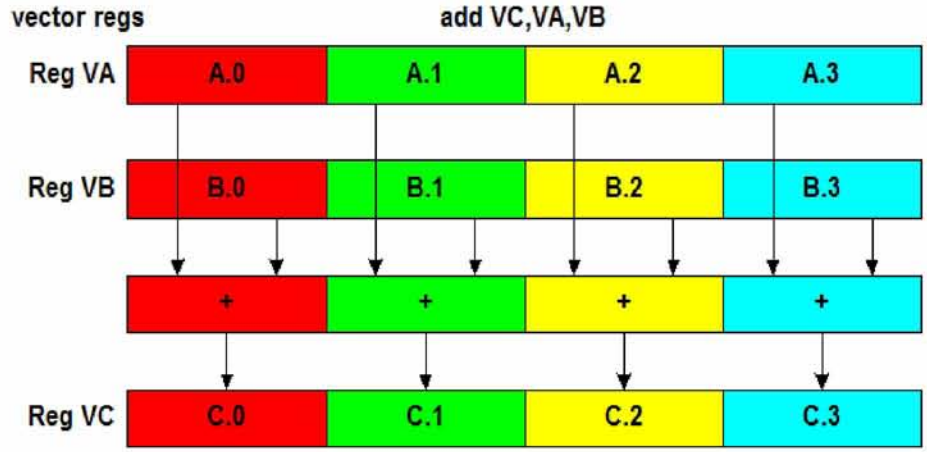
Εντολές SIMD

Η εκτέλεση υπολογισμών Single Instruction Multiple Data (SIMD) είναι ένα από τα πιο σημαντικά χαρακτηριστικά των SPEs. Η πλειονότητα των *compute-bound* εφαρμογών, όπως εφαρμογές πολυμέσων ή εφαρμογές επεξεργασίας εικόνας, εκτελούν τους ίδιους υπολογισμούς σε ένα μεγάλο σύνολο δεδομένων οπότε η ύπαρξη πράξεων SIMD είναι αναγκαία για την επιτάχυνση της εφαρμογής.

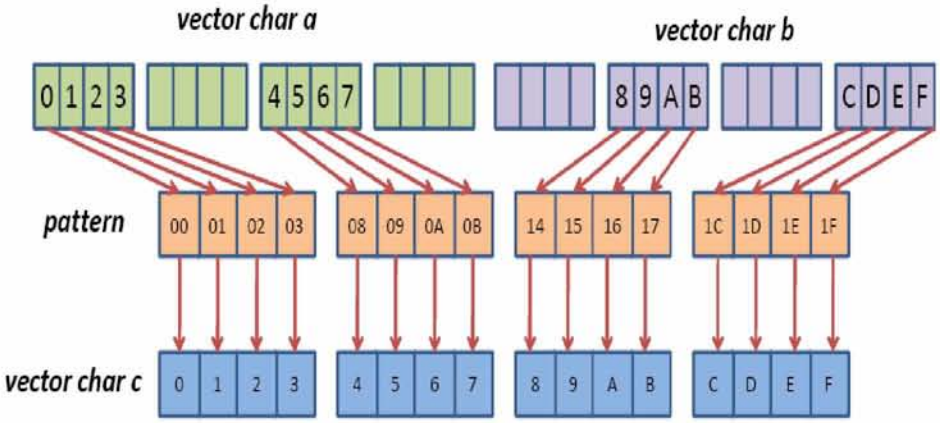
Για την εκτέλεση πράξεων SIMD, ο προγραμματιστής της εφαρμογής χρησιμοποιεί τα διαθέσιμα *intrinsics* που προσφέρονται από τους μεταγλωττιστές. Οι διαθέσιμες SIMD συναρτήσεις χωρίζονται στις ακόλουθες κατηγορίες:

- *Συναρτήσεις πρόσθεσης/αφαίρεσης*: Πρόσθεση, αφαίρεση, μερικά αθροίσματα, μέσος όρος.
- *Συναρτήσεις πολλαπλασιασμού/διαίρεσης*: Πολλαπλασιασμός, διαίρεση, πράξεις υπολοίπου και πράξεις *modulo*.
- *Συναρτήσεις αντιμετάθεσης και ολίσθησης*: Αναδιάταξη και μετακίνηση στοιχείων διανυσμάτων.
- *Βασικές συναρτήσεις ενός τελεστέου*: Απόλυτη τιμή, στρογγυλοποίηση, αντιστροφή αριθμού.
- *Λογικές Συναρτήσεις*.
- *Συναρτήσεις σύγκρισης στοιχείων διανυσμάτων*.
- *Μαθηματικές συναρτήσεις: Τριγωνομετρικές συναρτήσεις, συναρτήσεις λογαρίθμου και εκθετικές συναρτήσεις*.

Τα σχήματα 3.9 και 3.10 παρουσιάζουν δύο ενδεικτικά παραδείγματα όπου γίνεται χρήση των συναρτήσεων SIMD. Στο πρώτο σχήμα παρουσιάζεται μία διανυσματική πρόσθεση. Η συνάρτηση λαμβάνει ως ορίσματα δύο διανυσματικούς καταχωρητές και επιστρέφει έναν καταχωρητή με στοιχεία το άθροισμα των αντίστοιχων στοιχείων των τελεστέων. Στο δεύτερο σχήμα παρουσιάζεται μία πράξη διανυσματικής αναδιάταξης. Η αντίστοιχη συνάρτηση λαμβάνει ως ορίσματα τρεις καταχωρητές και επιστρέφει ως αποτέλεσμα έναν καταχωρητή ο οποίος έχει ως στοιχεία τα στοιχεία που έχουν επιλεγεί από τους δύο πρώτους καταχωρητές βάσει της μάσκας (*pattern*) που εμπεριέχεται στον τρίτο καταχωρητή.



Σχήμα 3.9: Παράδειγμα διανυσματικής πρόσθεσης.



Σχήμα 3.10: Παράδειγμα διανυσματικής αναδιάταξης.

Μεταφορά Δεδομένων

Το μέγεθος της τοπικής μνήμης των SPEs αλλά και η κατανεμημένη φύση του επεξεργαστή *Cell* δεν επιτρέπουν την αποθήκευση όλων των δεδομένων που απαιτούνται από τον κώδικα των SPEs στην τοπική μνήμη αυτών. Όπως αναφέρθηκε και στην ενότητα 3.2, η μεταφορά δεδομένων τόσο από την κύρια μνήμη προς την τοπική μνήμη των SPEs και αντίστροφα όσο και μεταξύ των SPEs πραγματοποιείται μέσω του μηχανισμού DMA. Οι εντολές που χρησιμοποιούνται είναι συγκεκριμένες εντολές του ελεγκτή μνήμης. Εντολές τύπου *get* χρησιμοποιούνται για την μεταφορά δεδομένων προς την τοπική μνήμη ενώ εντολές τύπου *put* χρησιμοποιούνται για μεταφορά δεδομένων από την τοπική μνήμη.

Αρκετές *data-intensive* εφαρμογές χρησιμοποιούν τα SPUs βάσει των παρακάτω βημάτων:

- Κάθε SPU λαμβάνει τα δεδομένα προς επεξεργασία από την κύρια μνήμη μέσω αιτήσεων DMA.

- Τα SPUs επεξεργάζονται τα δεδομένα.
- Τα SPUs μεταφέρουν τα επεξεργασμένα πλέον δεδομένα από την LS στην κύρια μνήμη του συστήματος.

Εάν αυτά τα βήματα εκτελούνται σειριακά, κάθε SPE δαπανά ένα μεγάλο ποσοστό του χρόνου, αναμένοντας αδρανές για την ολοκλήρωση των μεταφορών. Όμως, η ασύγχρονη φύση των εντολών DMA επιτρέπει την *παράλληλη* εκτέλεση αυτών των βημάτων. Η τεχνική αυτή ονομάζεται *multibuffering*. Η πιο συνήθης μορφή της τεχνικής του *multibuffering* είναι η τεχνική του *double-buffering*. Μία εφαρμογή που χρησιμοποιεί την τεχνική του *double-buffering* δεσμεύει δύο *buffers*, αντί ενός, για τα δεδομένα που πρόκειται να μεταφερθούν από και προς την LS. Ενώ το SPU επεξεργάζεται τα δεδομένα στον πρώτο *buffer*, τα δεδομένα στον δεύτερο *buffer* μεταφέρονται από ή προς την κύρια μνήμη. Έπειτα, οι ρόλοι των δύο *buffers* αντιστρέφονται: Το SPU επεξεργάζεται τα δεδομένα που έχουν μεταφερθεί στον δεύτερο *buffer* και παράλληλα δεδομένα μεταφέρονται από ή προς την LS με χρήση του πρώτου *buffer*. Κατ' αυτό τον τρόπο είναι δυνατή η μεγιστοποίηση του χρόνου κατά τον οποίο εκτελούνται υπολογισμοί με την ταυτόχρονη ελαχιστοποίηση του χρόνου όπου ο επεξεργαστής αναμένει αδρανής για την ολοκλήρωση της μεταφοράς των δεδομένων.

Ο μηχανισμός DMA είναι αρκετά απλός ώστε να διατηρηθεί η απλότητα στην αρχιτεκτονική του επεξεργαστή. Έτσι, μία εντολή DMA πραγματοποιεί μεταφορά δεδομένων από συνεχόμενες περιοχές τόσο στην τοπική μνήμη όσο και στην κύρια μνήμη ενώ απουσιάζουν εξειδικευμένα χαρακτηριστικά όπως τα μεγέθη *stride* και *span* που ενσωματώνονται σε άλλους μηχανισμούς DMA. Όμως, διατίθενται εντολές για συγχρονισμό μεταξύ πολλαπλών αιτήσεων DMA που προέρχονται από το ίδιο SPE.

Το μέγεθος των δεδομένων προς μεταφορά θα πρέπει να είναι ίσο με $1, 2, 4, 8$ ή πολλαπλάσιο των 16 $bytes$, με μέγιστο μέγεθος τα 16 KB. Συν τοις άλλοις, οι δύο διευθύνσεις, μεταξύ των οποίων μεταφέρονται τα δεδομένα, θα πρέπει να εμφανίζουν την ίδια ευθυγράμμιση. Σε περίπτωση που το μέγεθος των δεδομένων είναι μικρότερο από 16 $bytes$, οι δύο διευθύνσεις θα πρέπει να εμφανίζουν φυσική ευθυγράμμιση[4], ενώ σε διαφορετική περίπτωση οι δύο διευθύνσεις θα πρέπει να έχουν ευθυγράμμιση σε διευθύνσεις που είναι πολλαπλάσιες του 16. Η βέλτιστη μεταφορά δεδομένων επιτυγχάνεται όταν και οι δύο διευθύνσεις βρίσκονται ευθυγραμμισμένες σε διεύθυνση η οποία είναι πολλαπλάσια των 128 $bytes$. Οι απαιτήσεις ευθυγράμμισης είναι αρκετά περιοριστικές και εισάγουν επιπλέον καθυστερήσεις σε περίπτωση που το *memory access pattern* της εφαρμογής δεν τις ικανοποιεί.

Σε περίπτωση που η περιοχή της κύριας μνήμης δεν είναι συνεχόμενη, είναι δυνατή η χρήση του μηχανισμού των λιστών DMA-DMA *lists*. Οι λίστες DMA ενσωματώνουν πολλαπλές μεμονωμένες αιτήσεις DMA σε μία μόνο αίτηση, επιτρέποντας έτσι την αποφυγή του κόστους που συνεπάγεται η αρχικοποίηση του μηχανισμού DMA για κάθε επιμέρους αίτηση. Όσον αφορά στο μέγιστο πλήθος αιτήσεων, αυτό ισούται με 2048 αιτήσεις, με μέγιστο μέγεθος δεδομένων προς μεταφορά ίσο με 16 KB για κάθε αίτηση. Όμως, και οι λίστες DMA θα πρέπει να ικανοποιούν τους περιορισμούς ευθυγράμμισης του μηχανισμού DMA.

Τέλος, εκτός από τον μηχανισμό των αιτήσεων DMA, η επικοινωνία μεταξύ του PPE και των SPEs επιτυγχάνεται μέσω του μηχανισμού των γραμματοκιβωτίων (*mailboxes*) και τον μηχανισμό των σημάτων (*signals*). Αυτοί οι μηχανισμοί ενδείκνυνται σε περίπτωση που το μέγεθος των μηνυμάτων προς ανταλλαγή δεν υπερβαίνει τα 32 $bits$. Κάθε SPE διαθέτει:

[4]Φυσική ευθυγράμμιση είναι η ευθυγράμμιση των δύο διευθύνσεων σε θέση που είναι πολλαπλάσια του μεγέθους των δεδομένων που μεταφέρονται.

- 1 FIFO mailbox, τεσσάρων θέσεων, για τα εισερχόμενα μηνύματα.
- 1 mailbox, μίας θέσης, για εξερχόμενα μηνύματα.
- 1 mailbox, μίας θέσης, για εξερχόμενα μηνύματα προς το PPE με διακοπή.
- 2 καταχωρητές μεγέθους $4\ bytes$, με ειδοποίηση διακοπών, για την αποστολή μηνυμάτων στο αντίστοιχο SPE.

Επικαλυπτόμενα Τμήματα Κώδικα

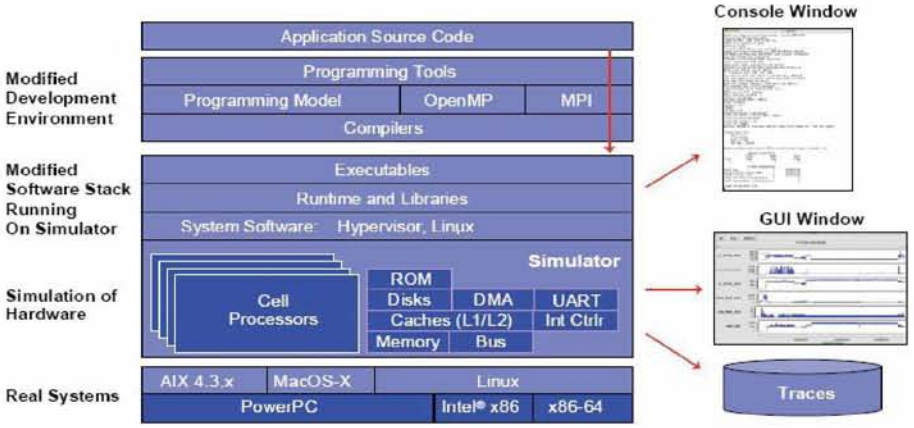
Με δεδομένο το περιορισμένο μέγεθος της τοπικής μνήμης των SPEs και της αρχιτεκτονικής αυτής - ενοποιημένη μνήμη εντολών και δεδομένων - υπάρχει το ενδεχόμενο της ανεπάρκειας της μνήμης για την αποθήκευση ενός εκτελέσιμου για το SPE. Σε αυτή την περίπτωση, τα επικαλυπτόμενα τμήματα κώδικα είναι ένας πολύ χρήσιμος μηχανισμός. Αυτή η προσέγγιση, τεμαχίζει τον κώδικα της εφαρμογής σε επιμέρους τμήματα ενώ καταλαμβάνεται ένα τμήμα της τοπικής μνήμης των SPEs για τον διαχειριστή των επικαλυπτόμενων τμημάτων. Ο προσδιορισμός των τμημάτων γίνεται από τον προγραμματιστή, ο οποίος παρέχει το κατάλληλο αρχείο σεναρίου (*script file*) στον *linker*. Κατά την εκτέλεση της εφαρμογής, ο διαχειριστής των τμημάτων είναι υπεύθυνος για την μεταφορά των τμημάτων από την κύρια μνήμη στην τοπική μνήμη των SPEs, όποτε αυτό είναι απαραίτητο. Η μεταφορά των τμημάτων απαιτεί έναν σημαντικό αριθμό από κύκλους μηχανής αλλά αυτός είναι ο μόνος και ο πιο αποδοτικός τρόπος για την εκτέλεση εφαρμογών για τα SPEs που απαιτούν περισσότερα από $256\ KB$ μνήμης.

Ο Προσομοιωτής

Ο προσομοιωτής *IBM Full-System Simulator*, ο οποίος παρέχεται με το *Cell BE SDK*, είναι μία εφαρμογή λογισμικού που προσομοιώνει την συμπεριφορά ενός συστήματος με έναν επεξεργαστή *Cell BE*. Ο χρήστης μπορεί να εκκινήσει το λειτουργικό σύστημα *Linux* μέσω του *image* που παρέχεται και να εκτελέσει τις εφαρμογές στο σύστημα που προσομοιώνεται. Επίσης, ο προσομοιωτής υποστηρίζει την εκτέλεση *statically linked* προγραμμάτων χωρίς την εκτέλεση του λειτουργικού συστήματος, ενώ παρέχονται και διαγνωστικές λειτουργίες, όπως η λειτουργία αποσφαλμάτωσης. Το σχήμα 3.11 παρουσιάζει το διάγραμμα της στοίβας προσομοίωσης για τον επεξεργαστή *Cell*.

Η υποδομή του προσομοιωτή έχει σχεδιασθεί ώστε να είναι δυνατή η μοντελοποίηση του επεξεργαστή και της αρχιτεκτονικής σε διάφορα επίπεδα αφαίρεσης, τα οποία ποικίλουν από λειτουργική προσομοίωση του συστήματος μέχρι λεπτομερή προσομοίωση της συνολικής επίδοσης του συστήματος.

- **Λειτουργική Προσομοίωση**: Αυτού του είδους η προσομοίωση μοντελοποιεί τα ορατά αποτελέσματα της εκτέλεσης των εντολών του προγράμματος, χωρίς να μοντελοποιεί τον χρόνο που απαιτείται για την εκτέλεση αυτών των εντολών. Για την εκτέλεση, γίνεται η υπόθεση ότι κάθε εντολή μπορεί να εκτελεστεί σε ένα σταθερό αριθμό κύκλων ρολογιού. Επίσης, οι προσπελάσεις της μνήμης είναι *σύγχρονες* και εκτελούνται σε σταθερό αριθμό κύκλων ρολογιού. Αυτό το μοντέλο προσομοίωσης είναι χρήσιμο για ανάπτυξη εφαρμογών και αποσφαλμάτωση αυτών όταν η ακριβής μέτρηση του χρόνου εκτέλεσης δεν είναι απαραίτητη.
- **Λεπτομερής Προσομοίωση**: Για την ανάλυση της επίδοσης της εφαρμογής και του συστήματος χρησιμοποιείται η λεπτομερής προσομοίωση. Αυτού του είδους η προσομοίωση είναι



Σχήμα 3.11: Η στοίβα προσομοίωσης για τον επεξεργαστή *Cell*.

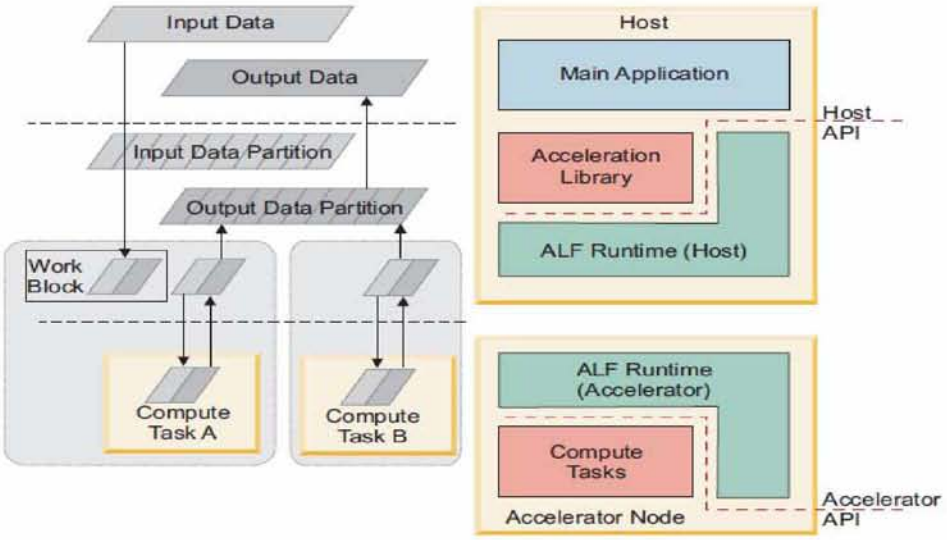
αρκετά πιο χρονοβόρα από την λειτουργική προσομοίωση καθώς μοντελοποιούνται όλα τα στοιχεία του συστήματος, μέσω ενός κατάλληλου μοντέλου ανάλυσης. Η μοντελοποίηση των διαφορών καθυστερήσεων γίνεται δυναμικά ώστε να ληφθούν υπόψη τόσο ο χρόνος επεξεργασίας που απαιτείται όσο και οι διάφοροι περιορισμοί των πόρων του συστήματος.

Η τρέχουσα έκδοση του προσομοιωτή υποστηρίζει την προσομοίωση σε επίπεδο κύκλου ρολογιού (*cycle-accurate simulation*) για όλο το σύστημα εκτός από το PPE. Τα μοντέλα που προσομοιώνουν την λειτουργία των SPEs μοντελοποιούν με μεγάλη ακρίβεια την ροή των εντολών καθώς αυτές διέρχονται από τα δύο *pipelines* των SPEs. Αυτό το χαρακτηριστικό επιτρέπει την λήψη ενός μεγάλου αριθμού από χρήσιμες πληροφορίες που αφορούν στην εκτέλεση του προγράμματος, όπως για παράδειγμα τον αριθμό των εντολών που εκτελέστηκαν παράλληλα, τον αριθμό των *pipeline stalls* και την αιτία αυτών, τον αριθμό των λανθασμένων προβλέψεων διακλάδωσης, κ.α.

Accelerated Library Framework (ALF)

Το Accelerated Library Framework (ALF) είναι μία προγραμματιστική διεπαφή (API) που δημιουργήθηκε για να επιταχύνει την διαδικασία ανάπτυξης εφαρμογών, παρέχοντας ένα επίπεδο αφαίρεσης των παράλληλων εφαρμογών σε συστήματα πολλαπλών πυρήνων. Η υλοποίηση του *framework* επικεντρώνει στην επίλυση προβλημάτων με *παραλληλισμό μεταξύ δεδομένων* (*data parallelism*) σε υβριδικά συστήματα *host-accelerator*. Το ALF υποστηρίζει το προγραμματιστικό μοντέλο Multiple Instruction Multiple Data (MIMD), όπου είναι δυνατή η ταυτόχρονη εκτέλεση πολλαπλών προγραμμάτων σε πολλαπλούς επεξεργαστές. Τα σημαντικότερα χαρακτηριστικά αυτού του *framework* είναι η διαχείριση της μεταφοράς των δεδομένων, η διαχείριση των παράλληλων εργασιών, η διαχείριση των πολλαπλών *buffers* για την τεχνική του *multi-buffering* όπως επίσης και ο τεμαχισμός των δεδομένων για κάθε επεξεργαστή.

Το σχήμα 3.12 παρουσιάζει μία επισκόπηση των συστατικών στοιχείων του ALF. Στον κεντρικό επεξεργαστή (*host processor*) γίνεται ο τεμαχισμός των δεδομένων εισόδου και των αντίστοιχων δεδομένων εξόδου σε μικρότερες μονάδες που αποκαλούνται *work blocks*. Με το παρεχόμενο API, τα *work blocks* αποθηκεύονται στην αντίστοιχη *work queue* και εν συνεχεία κατανέμονται στους επιμέρους επεξεργαστές. Έπειτα, οι επεξεργαστές εκτελούν την επεξεργασία στα *work*



Σχήμα 3.12: Επισκόπηση των συστατικών στοιχείων του *ALF*.

blocks και επιστρέφουν τα δεδομένα εξόδου στον κεντρικό επεξεργαστή. Το ALF υποστηρίζει μόνο σύνολα δεδομένων εισόδου τα οποία μπορούν να τεμαχιστούν σε *work blocks* με κατάλληλο μέγεθος ώστε να είναι δυνατή η αποθήκευση αυτών στην τοπική μνήμη των SPEs.

Συνοψίζοντας, το ALF διευκολύνει τους προγραμματιστές των εφαρμογών καθώς αναλαμβάνει την διαχείριση της μεταφοράς των δεδομένων, την διαχείριση των πόρων και των εργασιών αλλά και τα διάφορα θέματα συγχρονισμού που ενδέχεται να υπάρχουν. Παρόλα αυτά, η υλοποίηση ενός καλού τεμαχισμού των δεδομένων και η βελτιστοποίηση των υπολογισμών είναι καθήκον του προγραμματιστή. Ο τεμαχισμός των δεδομένων είναι πολύ σημαντικός τόσο για το ALF όσο και για την αρχιτεκτονική του επεξεργαστή, καθώς θα πρέπει να ληφθεί υπόψη το περιορισμένο μέγεθος της τοπικής μνήμης των SPEs και η μνήμη που απαιτείται για το *double-buffering*. Τέλος, οι υπολογιστικοί πυρήνες (*computational kernels*) θα πρέπει να βελτιστοποιηθούν από τον προγραμματιστή για την αρχιτεκτονική των SPEs.

Κεφάλαιο 4

Υλοποίηση και Βελτιστοποίηση

Αυτό το κεφάλαιο περιγράφει την διαδικασία υλοποίησης και βελτιστοποίησης του αλγορίθμου για την διόρθωση της παραμόρφωσης που προκαλείται από ευρυγώνιους φακούς στον επεξεργαστή *Cell*.

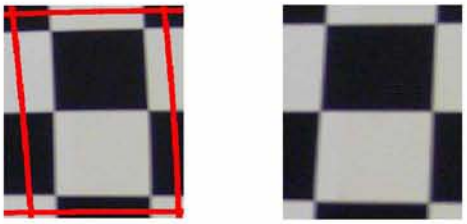
4.1 Στάδια Υλοποίησης

Το πρώτο στάδιο στην διαδικασία υλοποίησης του αλγορίθμου είναι η εφαρμογή μετασχηματισμών υψηλού επιπέδου για την εκμετάλλευση του παραλληλισμού που υπάρχει στην εφαρμογή. Μία πολύ σημαντική παρατήρηση από την ανάλυση του αλγορίθμου στο Κεφάλαιο 2 είναι ότι οι συντεταγμένες των *pixels* στις μη ακέραιες θέσεις ακολουθούν ένα αρκετά περίπλοκο και μη γραμμικό πρότυπο (Σχήμα 2.6). Η ακριβής μορφή της τροχιάς εξαρτάται από αρκετούς παράγοντες συμπεριλαμβανομένου του FoV, της θέσης του *pixel* και των παραμέτρων που μοντελοποιούν την παραμόρφωση που εισάγεται από τον εκάστοτε φακό που χρησιμοποιείται. Η μορφή της τροχιάς δεν εξαρτάται από την εικόνα που ο αλγόριθμος λαμβάνει ως είσοδο.

Επομένως, δοθέντων των διαφόρων παραμέτρων και της *περιοχής ενδιαφέροντος* (ROI) όπου ο χρήστης επιθυμεί την διόρθωση, είναι δυνατός ο εκ των προτέρων υπολογισμός της τροχιάς που ακολουθείται. Όμως, το αρκετά περίπλοκο πρότυπο προσπέλασης στην μνήμη, οι περιορισμοί ευθυγράμμισης της προσπέλασης στην μνήμη που επιβάλλει ο επεξεργαστής *Cell* και η μικρή χωρητικότητα της LS καθιστούν την εκ των προτέρων μεταφορά των δεδομένων, γνωστή και ως *prefetching*, αδύνατη.

Αν και δεν είναι δυνατή η εφαρμογή *prefetching*, ο αλγόριθμος διόρθωσης εμφανίζει μεγάλο βαθμό επαναχρησιμοποίησης των δεδομένων. Δεδομένα της $4x4$ γειτονιάς που χρησιμοποιήθηκαν για την παρεμβολή της τιμής ενός *pixel*, χρησιμοποιούνται και για την παρεμβολή γειτονικών *pixels*. Ενδεικτικό παράδειγμα αποτελούν τα *pixels* στις θέσεις $(x-1, y)$ και (x, y) του σχήματος 2.6 όπου χρησιμοποιείται η ίδια γειτονιά για την εφαρμογή του σχήματος παρεμβολής. Αυτού του είδους η επαναχρησιμοποίηση μεγιστοποιείται με χρήση του *2-D tiling* σε κάθε *frame*.

Η τεχνική του *tiling* εκμεταλλεύεται την *χωρική τοπικότητα* (*spatial locality*) και χρησιμοποιείται από μεταγλωττιστές βελτιστοποίησης (*optimizing compilers*) για την αύξηση του ποσοστού επιτυχίας στις προσβάσεις στην μνήμη *cache*. Κάθε *frame* τεμαχίζεται σε *blocks* ίσου μεγέθους και ο αλγόριθμος διόρθωσης εφαρμόζεται διαδοχικά στα *pixels* του κάθε *block*. Κατ' αυτό τον τρόπο επιτυγχάνεται η βελτιστοποίηση των υπολογισμών καθώς υπάρχει επαναχρησιμοποίηση των δεδομένων σε επίπεδο *block*. Το σχήμα 4.1 παρουσιάζει ένα από τα *tiles* του διορθωμένου *frame*, όπου το *frame* εισόδου είναι αυτό που παρουσιάζεται στο σχήμα 2.5, και το αντίστοιχο



Σχήμα 4.1: Ένα *tile* από την εικόνα εισόδου και το αντίστοιχο διορθωμένο *block*.

tile της παραμορφωμένης εικόνας (παρουσιάζεται με κόκκινο περίγραμμα) που απαιτείται για την εφαρμογή του σχήματος παρεμβολής ώστε να παραχθούν τα *pixels* του διορθωμένου *tile*. Όπως παρατηρούμε, γειτονικά *tiles* θα εμφανίζουν κάποια επικάλυψη στον χώρο των ευρυγώνιων φακών λόγω της κύρτωσης που εμφανίζεται. Αυτή η επικάλυψη έχει ως αποτέλεσμα τα *pixels* που βρίσκονται κοντά στις ακμές να μεταφέρονται περισσότερες από μία φορές από την κύρια μνήμη. Το επιπλέον κόστος για επικοινωνία αποτελεί και το μόνο μειονέκτημα της εφαρμογής του *tiling* στον αλγόριθμο.

Η εικόνα που χρησιμοποιήθηκε για όλες τις εκτελέσεις και τις διάφορες προσομοιώσεις της εφαρμογής, όπως παρουσιάζεται στο σχήμα 2.1, ακολουθεί το *format RGB* και έχει διαστάσεις $2592x1944$. Τα στάδια της αντίστροφης απεικόνισης, της παρεμβολής και του βαθυπερατού φίλτρου εφαρμόζονται σε μία περιοχή διαστάσεων $1280x960$, καθώς αυτό είναι το μέγεθος της ROI. Έπειτα, η εικόνα κλιμακώνεται προς τα κάτω, σε μία εικόνα ανάλυσης *VGA*.

Ο χρόνος εκτέλεσης του αλγορίθμου αξιολογήθηκε για τιμές της παραμέτρου FoV από 1.0^o έως 60.0^o και για όλες τις πιθανές ROI της εικόνας αλλά δεν παρουσιάστηκε κάποια μεταβολή. Αυτό ήταν αναμενόμενο καθώς το μέγεθος και η ανάλυση της εικόνας εξόδου είναι καθορισμένα και ανεξάρτητα από άλλες παραμέτρους του αλγορίθμου. Επίσης, το πλήθος των υπολογισμών που εκτελούνται για κάθε *pixel* της εικόνας εξόδου είναι σταθερό και δεν εξαρτάται από τις παραμέτρους και την εικόνα εισόδου. Βάσει αυτών, στις διάφορες εκτελέσεις του αλγορίθμου χρησιμοποιήθηκε η τιμή FoV = 40.0^o.

Το επόμενο βήμα στην διαδικασία υλοποίησης του αλγορίθμου είναι η εύρεση του παραλληλισμού σε υψηλό επίπεδο ώστε να είναι δυνατή η εκμετάλλευση των διαθέσιμων επεξεργαστών. Μετά την εφαρμογή του *tiling*, τα δύο εμφανή είδη παραλληλισμού είναι ο παραλληλισμός μεταξύ διαδοχικών *frames* και ο παραλληλισμός μεταξύ των *blocks* ενός *frame* καθώς δεν εμφανίζονται εξαρτήσεις ούτε μεταξύ των *frames* αλλά ούτε και μεταξύ των *blocks*. Για την εκμετάλλευση του παραλληλισμού μεταξύ *frames*, σε κάθε επεξεργαστή ανατίθεται ένα *frame* προς διόρθωση. Όμως, το μέγεθος του ROI ($1280x960$) και το περιορισμένο μέγεθος της LS δεν επιτρέπουν την αποθήκευση των απαιτούμενων *pixels* για την διόρθωση στα SPEs. Έτσι, καταλήγουμε στην εκμετάλλευση του παραλληλισμού που εμφανίζεται εντός ενός *frame*.

Έπειτα, ακολουθεί το στάδιο του τεμαχισμού και της αντιστοίχισης όπου γίνεται ο διαμερισμός του προβλήματος σε επιμέρους τμήματα και η αντιστοίχιση εργασιών στους διαθέσιμους επεξεργαστές, με έμφαση στην ταυτόχρονη εργασία και στο ελάχιστο επικοινωνιακό κόστος. Το σχήμα τεμαχισμού που χρησιμοποιήθηκε είναι το *data-cyclic*, ή *block-cyclic*, σχήμα όπου κάθε επεξεργαστής λαμβάνει με κυκλικό τρόπο ένα *tile* από την κύρια μνήμη, εκτελεί την απαραίτητη επεξεργασία και αποστέλλει στην κύρια μνήμη το αντίστοιχο διορθωμένο *tile* της εικόνας εξόδου.

Η τελευταία σχεδιαστική απόφαση που πρέπει να ληφθεί για την υλοποίηση του αλγορίθμου είναι η ακρίβεια στους διάφορους υπολογισμούς που εκτελούνται. Ο αλγόριθμος εμπεριέχει

αρκετές πράξεις αριθμών κινητής υποδιαστολής σε όλα τα στάδια της διόρθωσης. Επομένως, η ακρίβεια που χρησιμοποιείται είναι πολύ σημαντική για το τελικό αποτέλεσμα. Ο αλγόριθμος εκτελέστηκε τόσο χρησιμοποιώντας αριθμούς κινητής υποδιαστολής διπλής ακρίβειας όσο και αριθμούς κινητής υποδιαστολής απλής ακρίβειας. Για την αξιολόγηση των αποτελεσμάτων χρησιμοποιήθηκε ο λόγος Peak Signal to Noise Ratio (PSNR) ώστε να συγκριθούν οι παραγόμενες εικόνες. Όπως παρατηρήθηκε, οι εικόνες που παρήχθησαν με χρήση απλής ακρίβειας ήταν όμοιες σε ποσοστό 99.6% με τις αντίστοιχες που παρήχθησαν με χρήση διπλής ακρίβειας. Δεδομένης και της περιορισμένης αποδοτικότητας των SPEs στις πράξεις κινητής υποδιαστολής διπλής ακρίβειας, οι πράξεις που χρησιμοποιούνται σε όλες τις πράξεις αριθμών κινητής υποδιαστολής είναι πράξεις απλής ακρίβειας.

Το υπόλοιπο αυτής της ενότητας περιγράφει την διαδικασία υλοποίησης του αλγορίθμου σε έναν επεξεργαστή *Intel x86* και στον επεξεργαστή CBEA.

Υλοποίηση του αλγορίθμου σε αρχιτεκτονική Intel x86

Ο *baseline*[1] κώδικας που χρησιμοποιήθηκε για την αρχική υλοποίηση του αλγορίθμου είναι αυτός που χρησιμοποιείται και στην εργασία που περιγράφεται στο [8] όπου παρουσιάζεται η υλοποίηση του αλγορίθμου σε μία FPGA.

Για να αξιολογηθεί κατά πόσο είναι δυνατή υλοποίηση του αλγορίθμου για την επίτευξη της διόρθωσης παραμόρφωσης σε πραγματικό χρόνο, δηλαδή διόρθωση $25 - 30$ *frames* ανά δευτερόλεπτο, με έναν συμβατικό επεξεργαστή γενικού σκοπού, υλοποιήθηκε και εκτελέστηκε μία παράλληλη, βελτιστοποιημένη έκδοση του αλγορίθμου σε ένα σύστημα που εμπεριέχει έναν επεξεργαστή *Intel Core2 Duo T7500* με συχνότητα λειτουργίας ίση με $2.2\ GHz$. Το σύστημα ενσωματώνει μνήμη *RAM* με μέγεθος ίσο με $2\ GB$ και χρησιμοποιείται το λειτουργικό *Linux* (διανομή λειτουργικού *OpenSUSE 10.3*, με τον πυρήνα 2.6.22). Για την μεταγλώττιση του κώδικα της εφαρμογής χρησιμοποιήθηκε τόσο ο μεταγλωττιστής *icc* από το *Intel compiler suite* [14] όσο και ο μεταγλωττιστής *gcc*. Για την παραγωγή του τελικού εκτελέσιμου σε κάθε μεταγλωττιστή, χρησιμοποιήθηκαν οι επιλογές που οδηγούσαν στο εκτελέσιμο με τον μικρότερο χρόνο εκτέλεσης. Η επίδοση των εκτελέσιμων που παρήχθησαν με χρήση του *icc* ήταν καλύτερη, οπότε παρουσιάζονται μόνο οι χρόνοι εκτέλεσης αυτών.

Ο κώδικας για τον επεξεργαστή παραλληλοποιήθηκε χρησιμοποιώντας την βιβλιοθήκη *OpenMP*. Κάθε *tile* της εικόνας εξόδου παράγεται ανεξάρτητα, ενδεχομένως από ένα διαφορετικό πλαίσιο εκτέλεσης του επεξεργαστή. Εκτός από την εφαρμογή του *tiling* για την εικόνα εισόδου και την εικόνα εξόδου, εφαρμόσθηκαν όλες οι δυνατές βελτιστοποιήσεις που περιγράφονται στην ενότητα 4.2 και πιο συγκεκριμένα η εφαρμογή της μη αυτόματης διανυσματοποίησης (*vectorization*) των υπολογισμών χρησιμοποιώντας την αρχιτεκτονική συνόλου εντολών Streaming SIMD Extensions (SSE) (*SSE ISA*) [35] και το ξεδίπλωμα βρόχου. Συν τοις άλλοις, καθώς και η αρχιτεκτονική συνόλου εντολών SSE παρουσιάζει κάποιους περιορισμούς ευθυγράμμισης, δόθηκε αρκετή προσοχή στον χειρισμό των φορτώσεων μη ευθυγραμμισμένων δεδομένων σε διανυσματικούς καταχωρητές.

Για την ανάλυση του χρόνου εκτέλεσης της εφαρμογής χρησιμοποιήθηκε το λογισμικό *Intel VTune Performance Analyzer for Linux, 9.1* [16]. Η εφαρμογή επεξεργάζεται 2.38 Frames per Second (fps) εάν χρησιμοποιηθεί ένα νήμα, ενώ είναι δυνατή η επεξεργασία 4.17 *frames* το δευτερόλεπτο εάν χρησιμοποιηθούν και τα δύο διαθέσιμα νήματα. Τα αποτελέσματα αυτά αποτελούν το κίνητρο για την χρήση ενός μη συμβατικού επεξεργαστή υψηλής απόδοσης, όπως ο

[1] Ο *baseline* κώδικας είναι ο κώδικας που περιγράφει την βασική λειτουργικότητα της εφαρμογής όπου δεν έχει εφαρμοσθεί κάποια βελτιστοποίηση χαμηλού επιπέδου.

επεξεργαστής CBEA, για την επίτευξη της επιθυμητής απόδοσης.

Υλοποίηση του αλγορίθμου στον επεξεργαστή Cell Broadband Engine

Για την πειραματική αξιολόγηση της εφαρμογής στον επεξεργαστή CBEA χρησιμοποιήθηκε ένα *PlayStation3* το οποίο ενσωματώνει έναν επεξεργαστή *Cell*, όπου είναι διαθέσιμα μόνο τα 6 από τα 8 SPEs. Από τα υπόλοιπα 2 SPEs, το ένα χρησιμοποιείται ως πλεονασματικό για την αύξηση του *yield* ενώ το άλλο χρησιμοποιείται από το λειτουργικό σύστημα για λόγους ασφαλείας [68] (χρησιμοποιείται ως ο *hypervisor* όπου εκτελείται το εναλλακτικό λειτουργικό σύστημα). Το λειτουργικό που χρησιμοποιείται είναι η διανομή *Yellow Dog Linux* με τον πυρήνα 2.6.23. Όπως προαναφέρθηκε, το λειτουργικό και οι εφαρμογές ελέγχονται από τον *hypervisor* του συστήματος, οπότε δεν είναι δυνατή η πρόσβαση στους *hardware performance counters* που διαθέτει ο επεξεργαστής.

Για την λήψη πιο λεπτομερών αποτελεσμάτων και την αξιολόγηση της εκτέλεσης της εφαρμογής για 8 SPEs χρησιμοποιήθηκε ο προσομοιωτής *IBM Full-System Simulator*, ο οποίος παρέχεται με το *Cell BE SDK*. Η έκδοση του προσομοιωτή που χρησιμοποιήθηκε είναι η έκδοση 3.1, η οποία εκτελείται στο λειτουργικό *Fedora Linux 7*. Αυτή η έκδοση του προσομοιωτή είναι ακριβής σε επίπεδο κύκλου ρολογιού (*cycle-accurate*) για όλα τα στοιχεία του συστήματος, στοιχείο που επιτρέπει την λήψη αρκετά ακριβών μετρήσεων για την εκτέλεση της εφαρμογής. Για την μεταγλώττιση του κώδικα χρησιμοποιήθηκαν οι μεταγλωττιστές *gcc* (έκδοση 4.2.1) και *xlc* (έκδοση 9.0), με επιλογές μεταγλώττισης εκείνες που οδηγούσαν στην παραγωγή του εκτελέσιμου με τον πιο μικρό χρόνο εκτέλεσης. Η απόδοση των εκτελέσιμων που παρήχθησαν με τον *gcc* ήταν καλύτερη, οπότε παρουσιάζονται μόνο οι χρόνοι εκτέλεσης αυτών.

Εκτέλεση του αλγορίθμου στο PPE

Μετά την εφαρμογή της τεχνικής του *tiling* ώστε οι υπολογισμοί να πραγματοποιούνται σε *blocks*, το επόμενο στάδιο για την υλοποίηση του αλγορίθμου στον επεξεργαστή CBEA είναι η διαδικασία του *profiling* ώστε να βρεθούν οι υπολογιστικοί πυρήνες όπου το πρόγραμμα δαπανά το μεγαλύτερο μέρος του χρόνου εκτέλεσης. Από αυτή την διαδικασία θα προκύψουν οι πιο χρονοβόροι υπολογιστικοί πυρήνες, οι οποίοι εν συνεχεία θα ανατεθούν στα SPEs, ώστε να είναι δυνατή η εκμετάλλευση των δυνατοτήτων αυτών. Για την διαδικασία του *profiling* στο PPE χρησιμοποιήθηκε το πρόγραμμα *GProf* [50]. Για τις απαραίτητες μετρήσεις εκτελέσθηκε ο κώδικας της εφαρμογής με ένα νήμα τόσο στον επεξεργαστή *Core2 Duo* όσο και στο PPE ενώ δεν χρησιμοποιήθηκαν οι επεκτάσεις *AltiVec* για τον κώδικα στο PPE. Το σχήμα 4.2 παρουσιάζει την αναλυτική παράθεση του συνολικού χρόνου και των χρόνων εκτέλεσης των υπολογιστικά πολυπλοκότερων πυρήνων της εφαρμογής για τους δύο επεξεργαστές.

Όπως παρατηρούμε, οι υπολογιστικά πολυπλοκότεροι πυρήνες της εφαρμογής και για τους δύο επεξεργαστές είναι οι συναρτήσεις *Inverse Mapping*, *Bicubic Interpolation* και *Low-Pass Filter* που υλοποιούν την *αντίστροφη απεικόνιση*, την *παρεμβολή* και το *βαθυπερατό φίλτρο* αντίστοιχα. Η συνάρτηση που υλοποιεί το σχήμα παρεμβολής είναι ο πιο πολύπλοκος υπολογιστικός πυρήνας με συνεισφορά στον χρόνο εκτέλεσης της εφαρμογής μεγαλύτερη του 60% και για τους δύο επεξεργαστές. Η συνάρτηση που υλοποιεί την *αντίστροφη απεικόνιση* έχει μικρή συνεισφορά και αυτό οφείλεται στις αρκετά αποδοτικές μονάδες που εκτελούν πράξεις αριθμών κινητής υποδιαστολής που εμπεριέχονται στο PPE και στον *Core2 Duo*. Επίσης, ο χρόνος εκτέλεσης της εφαρ-

Σχήμα 4.2: Ο χρόνος εκτέλεσης της εφαρμογής στο *PPE* (με χρήση του *gcc*) και στον *Core2 Duo* (με χρήση του *icc*).

μογής στο PPE είναι μεγαλύτερος από τον χρόνο εκτέλεσης στον *Core2 Duo*, οπότε η διόρθωση σε πραγματικό χρόνο δεν είναι εφικτή σε περίπτωση που ο αλγόριθμος εκτελείται εξολοκλήρου στο PPE, δηλαδή χωρίς την χρήση των διαθέσιμων SPEs.

Εδώ θα πρέπει να αναφερθεί ότι η συνάρτηση *Inverse Mapping* πρέπει να καλείται μόνο σε περίπτωση που οι παράμετροι FoV ή ROI μεταβάλλονται. Για ένα ρεαλιστικό σενάριο χρήσης του αλγορίθμου, η συχνότητα μεταβολής αυτών των παραμέτρων δεν αναμένεται να είναι μεγάλη. Τα αποτελέσματα του σχήματος 4.2 υποθέτουν την εκτέλεση του *Inverse Mapping* για κάθε *frame* στο οποίο εκτελείται η διόρθωση. Στο Κεφάλαιο 5 αξιολογείται η περίπτωση όπου η διαδικασία του *Inverse Mapping* εκτελείται μόνο μία φορά, οι συντεταγμένες αποθηκεύονται στην κύρια μνήμη και μεταφέρονται στα SPEs για την διόρθωση κάθε *tile*.

Εκμετάλλευση του παραλληλισμού σε επίπεδο tile

Όπως προέκυψε από το σχήμα 4.2, οι συναρτήσεις που είναι υπεύθυνες για την πλειονότητα του χρόνου εκτέλεσης του αλγορίθμου είναι οι συναρτήσεις *Inverse Mapping*, *Bicubic Interpolation* και *Low-Pass Filter*. Αυτές είναι και οι συναρτήσεις που θα υλοποιηθούν στα SPEs. Το PPE εκτελεί τον απαραίτητο κώδικα για την αρχικοποίηση του συστήματος, όπου συμπεριλαμβάνεται η φόρτωση της εικόνας εισόδου, και δημιουργεί τα απαραίτητα νήματα για την εκτέλεση του αντίστοιχου κώδικα στα SPEs.

Το σχήμα διαμοιρασμού των *tiles* στα διαθέσιμα SPEs είναι το σχήμα *block-cyclic* που περιγράφεται στην ενότητα 4.1. Αυτό το σχήμα τεμαχισμού επιτρέπει στο κάθε SPE να επεξεργάζεται τα δεδομένα ανεξάρτητα τόσο από τα υπόλοιπα SPEs όσο και από το PPE. Ο μόνος συγχρονισμός που απαιτείται είναι αυτός στο τέλος της επεξεργασίας του κάθε *frame*. Το μέγεθος του *tile* ορίζεται σε 256 στήλες και 48 γραμμές και αυτό το μέγεθος χρησιμοποιείται για όλες τις εκτελέσεις της εφαρμογής. Στο Κεφάλαιο 5 εξετάζονται διάφορα μεγέθη των *tiles* για την αξιολόγηση του μεγέθους του *tile* στον χρόνο εκτέλεσης του αλγορίθμου.

Για την μεταφορά των *pixels* που απαρτίζουν κάθε *tile* από την κύρια μνήμη στην LS του κάθε SPE, χρησιμοποιείται ο μηχανισμός των ασύγχρονων αιτήσεων DMA, όπως αυτός περιγράφηκε στην ενότητα 3.3. Κάθε SPE αρχικοποιεί τις απαραίτητες αιτήσεις για την μεταφορά των δεδο-

μένων μεταξύ της κύριας και της τοπικής μνήμης. Η επιλογή της αρχικοποίησης των αιτήσεων DMA από τα SPEs είναι επιβεβλημένη καθώς σε περίπτωση που οι αιτήσεις αρχικοποιούνταν από το PPE, το PPE θα αποτελούσε πιθανό σημείου συνωστισμού, λόγω του μεγάλου αριθμού αιτήσεων DMA που απαιτούνται για την μεταφορά των δεδομένων.

Όσον αφορά στην μορφή των δεδομένων που μεταφέρονται, αυτά αποτελούν μεταβλητές τύπου *unsigned char*. Αυτή η μορφή των δεδομένων συντελεί στην ταχύτερη μεταφορά των δεδομένων λόγω του μικρού μεγέθους. Όμως, για την εκτέλεση των μετέπειτα πράξεων, όπως για παράδειγμα το σχήμα παρεμβολής, είναι απαραίτητη η μετατροπή των δεδομένων σε αριθμούς κινητής υποδιαστολής απλής ακρίβειας.

Το σχήμα 4.3 παρουσιάζει τον χρόνο εκτέλεσης της αρχικής έκδοσης του κώδικα της εφαρμογής, όπου δεν χρησιμοποιείται η τεχνική του *double-buffering*, και της τελικής έκδοσης αυτού, όπου έχουν εφαρμοσθεί όλες οι βελτιστοποιήσεις. Το διάγραμμα είναι σε λογαριθμική κλίμακα και οι μετρήσεις για 7 και 8 SPEs αποκτήθηκαν με χρήση του προσομοιωτή. Οι μετρήσεις για $1-6$ SPEs στο *PlayStation 3* συμπίπτουν με τις μετρήσεις στον προσομοιωτή και αυτό μας επιτρέπει να χρησιμοποιήσουμε και τις μετρήσεις για 7 και 8 SPEs ως ακριβείς μετρήσεις για τον χρόνο εκτέλεσης της εφαρμογής.

Η εφαρμογή εκτελέσθηκε για 1 έως 8 SPEs και παρατηρήθηκε ότι η αύξηση του αριθμού των SPEs που χρησιμοποιούνται για την επεξεργασία αποφέρει μεγάλα οφέλη στην απόδοση της εφαρμογής. Αυτό οφείλεται κυρίως στον μεγάλο βαθμό παραλληλισμού που παρουσιάζει η εφαρμογή. Στην πραγματικότητα, ο χρόνος εκτέλεσης μειώνεται γραμμικά με την αύξηση του αριθμού των SPEs από $4.93\ secs/frame$ όταν χρησιμοποιείται 1 SPE σε $0.62\ secs/frame$ όταν χρησιμοποιούνται 8 SPEs. Το *speedup* που επιτυγχάνεται είναι σχεδόν τέλειο ($\simeq 8$) και αυτό δικαιολογεί την επιλογή του σχήματος τεμαχισμού που χρησιμοποιείται. Από το σχήμα 4.3 παρατηρούμε επίσης ότι η αρχιτεκτονική των SPEs δεν είναι βελτιστοποιημένη για την εκτέλεση κώδικα που αποτελείται από βαθμωτές πράξεις. Η αρχιτεκτονική του PPE είναι πιο αποδοτική στην εκτέλεση βαθμωτών πράξεων: 1 νήμα εκτέλεσης στο PPE επεξεργάζεται ένα *frame* σε $1.8\ secs$ ενώ ένα SPE επεξεργάζεται 1 *frame* σε $4.93\ secs$. Σε αυτό συντελεί και ο χρόνος που το SPE δαπανά αναμένοντας αδρανές για την ολοκλήρωση των μεταφορών.

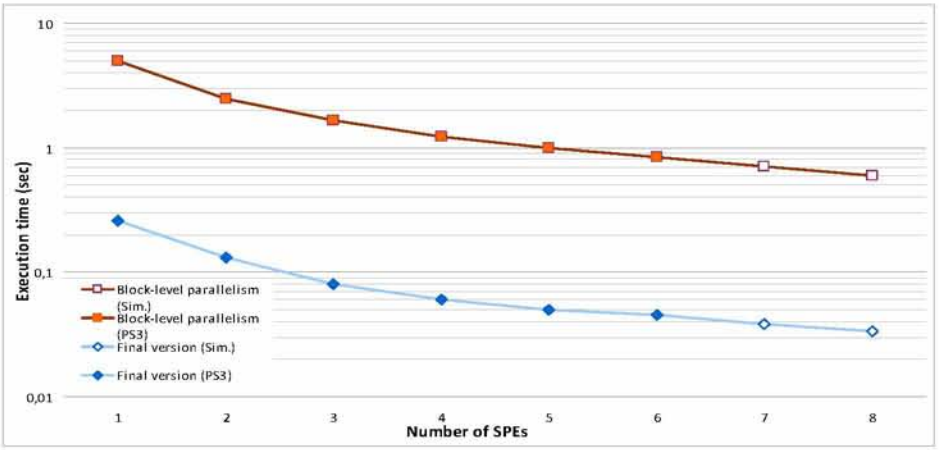
Όσον αφορά στον χρόνο εκτέλεσης του κάθε υπολογιστικού πυρήνα στα SPEs, αυτός παρουσιάζεται αναλυτικά στο σχήμα 4.4. Από το σχήμα παρατηρούμε ότι ο υπολογιστικά πολυπλοκότερος πυρήνας είναι η διαδικασία του *Inverse Mapping* και αυτό οφείλεται στις πολύπλοκες τριγωνομετρικές συναρτήσεις που χρησιμοποιούνται για την *αντίστροφη απεικόνιση*. Παρά την αποτελεσματικότητα στην εκτέλεση των συναρτήσεων της βιβλιοθήκης *SIMD Math Library* [57], δεν ήταν δυνατή η περαιτέρω βελτίωση του χρόνου εκτέλεσης της εφαρμογής.

4.2 Στάδια Βελτιστοποίησης

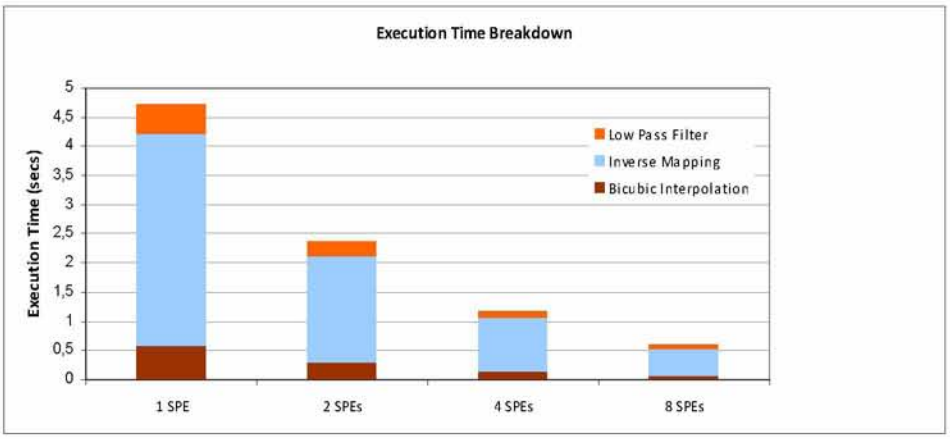
Από την ανάλυση στην ενότητα 4.1 είναι κατανοητό ότι δεν είναι δυνατή η επίτευξη διόρθωσης σε πραγματικό χρόνο χωρίς την εφαρμογή κάποιων μετασχηματισμών βελτιστοποίησης, οι οποίοι θα έχουν ως αποτέλεσμα την μείωση του χρόνου εκτέλεσης της εφαρμογής. Η παρούσα ενότητα αναλύει όλους τους μετασχηματισμούς που πραγματοποιήθηκαν στον πηγαίο κώδικα της εφαρμογής.

DMA Double Buffering

Στην αρχική έκδοση του πηγαίου κώδικα του αλγορίθμου, η μεταφορά των δεδομένων δεν πραγματοποιείται ταυτόχρονα με τους αντίστοιχους υπολογισμούς. Αυτό προσθέτει ένα επιπλέον



Σχήμα 4.3: Ο χρόνος εκτέλεσης της αρχικής και της τελικής έκδοσης του κώδικα της εφαρμογής συναρτήσει του αριθμού των *SPEs*.



Σχήμα 4.4: Αναλυτική παράθεση του χρόνου εκτέλεσης κάθε υπολογιστικού πυρήνα συναρτήσει του αριθμού των *SPEs*.

κόστος, της τάξεως των 0.005 $secs/frame$, στον χρόνο εκτέλεσης της εφαρμογής. Το μικρό επιπλέον κόστος οφείλεται στο εύρος ζώνης του διαύλου EIB (96 $B/cycle$), το οποίο είναι ικανό να ικανοποιήσει όλες τις αιτήσεις που πραγματοποιούνται με πολύ μικρή καθυστέρηση.

Παρά το μικρό επιπλέον κόστος, η διαδικασία της μεταφοράς δεδομένων επιταχύνεται με την εφαρμογή της τεχνικής του *double-buffering*. Για την μεταφορά των δεδομένων, δεσμεύονται δύο ξεχωριστοί *buffers*. Κατ' αυτό τον τρόπο είναι δυνατή η επικάλυψη των υπολογισμών για το *N*-οστό *tile* με την μεταφορά των δεδομένων για το *(N+1)*-οστό *tile* προς την LS και την μεταφορά των δεδομένων για το *(N-1)*-οστό *tile* προς την κύρια μνήμη. Η ενσωμάτωση αυτής της τεχνικής επικαλύπτει δραστικά την καθυστέρηση για την πρόσβαση στην μνήμη, αυξάνοντας όμως τόσο τις απαιτήσεις σε μνήμη για την αποθήκευση των *pixels* όσο και την πολυπλοκότητα του κώδικα καθώς απαιτείται η εναλλαγή μεταξύ των διαθέσιμων *buffers*. Η συνεισφορά της εξάλειψης αυτού του μικρού κόστους είναι περισσότερο σημαντική στην τελική έκδοση του κώδικα.

Διανυσματοποίηση Υπολογισμών και Ξεδίπλωμα Βρόχου

Το επόμενο βήμα στην διαδικασία βελτιστοποίησης είναι η βελτιστοποίηση του κώδικα που εκτελείται στα SPEs. Για την βελτιστοποίηση του κώδικα χρησιμοποιήθηκε τόσο μία σειρά από συμβατικούς, μη αυτόματους μετασχηματισμούς για την εκμετάλλευση στοιχείων της αρχιτεκτονικής των SPEs όπως οι διανυσματικές μονάδες επεξεργασίας ή το *dual-issue pipeline* όσο και μη συμβατικούς μετασχηματισμούς που δεν εφαρμόζονται τυπικά σε βαθμωτούς επεξεργαστές γενικού σκοπού. Τόσο η παρούσα όσο και οι υπόλοιπες υποενότητες περιγράφουν τις βελτιστοποιήσεις που εφαρμόσθηκαν. Εδώ θα πρέπει να αναφερθεί ότι, παρά τις εξελιγμένες λειτουργίες για την παραγωγή κώδικα που ενσωματώνονται στους δύο μεταγλωττιστές, *gcc* και *xlc*, που χρησιμοποιήθηκαν για την μεταγλώττιση του κώδικα για τον CBEA, δεν ήταν δυνατή η αυτόματη εφαρμογή κάποιων από τους απαιτούμενους μετασχηματισμούς, όπως για παράδειγμα η *διανυσματοποίηση των υπολογισμών*.

Η πλειονότητα των υπολογισμών που πραγματοποιούνται στα SPEs εσωκλείονται σε διπλά *for-loops*. Το εξωτερικό *loop*, το οποίο διατρέχει όλα τα *pixels* (*pixel scan*), αρχικά υπολογίζει τις κλασματικές συντεταγμένες για όλα τα *pixels* εντός του *tile* (*Inverse Mapping*) και έπειτα εφαρμόζει την συνάρτηση *Bicubic Interpolation* για την εφαρμογή του σχήματος παρεμβολής. Και το σχήμα παρεμβολής χρησιμοποιεί δύο εμφωλευμένα *for-loops*, όπου το ένα *loop* διατρέχει τις τρεις συνιστώσες χρώματος της εικόνας (*RGB*). Ομοίως, και στην συνάρτηση *Low-Pass Filter* χρησιμοποιούνται δύο εξωτερικά *for-loops*, ένα για κάθε μία κατεύθυνση εφαρμογής του φίλτρου (οριζόντια και κάθετη). Καθένα από αυτά τα *loops* περικλείει ένα εσωτερικό *loop* με τρεις επαναλήψεις για τις τρεις συνιστώσες του χρώματος της εικόνας.

Για την επιτάχυνση των υπολογισμών που πραγματοποιούνται, εκμεταλλευόμαστε τον *παραλληλισμό μεταξύ δεδομένων* (DLP) και τις δυνατότητες διανυσματικών υπολογισμών των SPEs, ομαδοποιώντας τέσσερεις τελεστέους κινητής υποδιαστολής, απλής ακρίβειας σε έναν διανυσματικό καταχωρητή. Με χρήση αυτού του σχήματος, η συνάρτηση *Inverse Mapping* παράγει τέσσερα ζεύγη συντεταγμένων σε μία επανάληψη, ενώ το μονοδιάστατο *stencil* για το σχήμα παρεμβολής εφαρμόζεται σε τέσσερα *pixels* σε κάθε επανάληψη. Καθώς τα SPEs δαπανούν τον περισσότερο χρόνο στα προαναφερθέντα *loops*, η παραλληλοποίηση με χρήση πράξεων SIMD μπορεί να επιταχύνει την εκτέλεση των υπολογισμών, με μέγιστο παράγοντα επιτάχυνσης ίσο με 4.

Αυτό το έμμεσο ξεδίπλωμα βρόχου (*loop unrolling*) λόγω της διανυσματοποίησης των υπολογισμών έχει το επιπλέον επιθυμητό αποτέλεσμα της μείωσης των διακλαδώσεων που πραγματοποιούνται προς την αρχή του κάθε βρόχου κατά έναν παράγοντα ίσο με 4. Η λανθασμένη πρόβλεψη διακλαδώσεων συνεπάγεται μία αρκετά μεγάλη καθυστέρηση, περίπου 20 κύκλους ρολογιού, ενώ

η καθυστέρηση στην εκτέλεση μίας τυπικής εντολής κυμαίνεται μεταξύ 2 και 7 κύκλων ρολογιού. Η καθυστέρηση από την λανθασμένη πρόβλεψη διακλαδώσεων υπάρχει πάντα στην περίπτωση των προς τα πίσω διακλαδώσεις που υπάρχουν στα *for-loops* καθώς τα SPEs προβλέπουν[2] πάντα τις διακλαδώσεις ως *μη αληθείς* (*not taken*).

Για την περαιτέρω εκμετάλλευση των θετικών αποτελεσμάτων της εξάλειψης των εντολών διακλάδωσης πραγματοποιήθηκε ξεδίπλωμα των εσωτερικών βρόχων που διατρέχουν τις τρεις συνιστώσες χρώματος (*R*, *G* και *B*), με παράγοντα ξεδιπλώματος ίσο με 3. Αυτό το ξεδίπλωμα είχε ως αποτέλεσμα την πλήρη εξάλειψη των εντολών διακλάδωσης που εμπεριέχονταν σε αυτούς τους βρόχους.

Εκτός από την εξάλειψη των εντολών διακλάδωσης, το ξεδίπλωμα βρόχου έχει ως αποτέλεσμα την ύπαρξη περισσοτέρων εντολών προς δρομολόγηση. Κατ' αυτό τον τρόπο, ο μεταγλωττιστής μπορεί να εκτελέσει μία καλύτερη δρομολόγηση των εντολών, εξαλείφοντας έτσι πιθανά παγώματα στο *pipeline* λόγω εξαρτήσεων μεταξύ εντολών. Το μόνο μειονέκτημα της τεχνικής του *ξεδιπλώματος βρόχου* είναι η πίεση που υπάρχει στην χρήση των καταχωρητών (*register pressure*), η οποία ενδέχεται να οδηγήσει στην απώλεια όλων ή μερικών από τα πλεονεκτήματα αυτής της τεχνικής. Όπως προέκυψε με χρήση του προσομοιωτή, η χρήση των καταχωρητών αυξήθηκε όταν χρησιμοποιήθηκε η τεχνική του *ξεδιπλώματος βρόχου*. Όμως, ο μεγάλος αριθμός των διαθέσιμων καταχωρητών στα SPEs, καθιστά δυνατή την αποθήκευση όλων των *ενεργών τιμών* (*live values*) που εμφανίζονται εντός του βρόχου, επιτρέποντας την πλήρη εκμετάλλευση των δυνατοτήτων της τεχνικής του *ξεδιπλώματος βρόχου*.

Ο πίνακας 4.1 παρουσιάζει τα αποτελέσματα από τις δύο προαναφερθείσες βελτιστοποιήσεις του κώδικα με χρήση των 6 SPEs του *PS3*. Όπως παρατηρούμε, η πρώτη βελτιστοποίηση επιτρέπει την επιτάχυνση της εφαρμογής με παράγοντα ίσο με 9.03 σε σύγκριση με την αρχική έκδοση του κώδικα και μειώνει το κόστος λόγω των παγωμάτων του *pipeline* που δημιουργούνται από την λανθασμένη πρόβλεψη διακλαδώσεων από 12.5% σε 3.1% του συνολικού αριθμού των εντολών. Το ξεδίπλωμα των εσωτερικών βρόχων μειώνει περισσότερο τον χρόνο εκτέλεσης της εφαρμογής σε $0.07\ secs/frame$, ο οποίος συνεπάγεται έναν παράγοντα επιτάχυνσης ίσο με $12x$. Επίσης, και οι δύο βελτιστοποιήσεις οδηγούν σε αύξηση της χρησιμοποίησης του *dual-issue* καθώς υπάρχουν περισσότερες διαθέσιμες εντολές προς δρομολόγηση. Επιπλέον ξεδίπλωμα των βρόχων δεν οδηγούσε σε σημαντική μείωση του χρόνου εκτέλεσης της εφαρμογής ενώ είχε ως αποτέλεσμα την σημαντική αύξηση του μεγέθους του κώδικα.

Μη Ευθυγραμμισμένες Φορτώσεις από την Μνήμη

Ένα ιδιαίτερο χαρακτηριστικό της συνάρτησης *Bicubic Interpolation* είναι οι μη ευθυγραμμισμένες προσβάσεις στην μνήμη. Στις περισσότερες περιπτώσεις, το επάνω αριστερά *pixel* της $4x4$ γειτονιάς που χρησιμοποιείται για την παρεμβολή βρίσκεται σε διεύθυνση η οποία δεν είναι πολλαπλάσια των 16 $bytes$ (*vector alignment*). Αυτό αποτελεί μία πρόκληση κατά την διαδικασία της διανυσματοποίησης των υπολογισμών. Στην περίπτωση της φόρτωσης των βαθμωτών μεταβλητών από μη ευθυγραμμισμένες διευθύνσεις μνήμης, ο μεταγλωττιστής του εκτελέσιμου για τα SPEs εισάγει επιπλέον βοηθητικές εντολές και εντολές μετατόπισης ώστε να μεταφερθεί η βαθμωτή μεταβλητή στο *preferred slot*.

Για να διαπιστωθεί η επίδραση αυτών των φορτώσεων από την μνήμη χρησιμοποιήθηκε το εργαλείο *spu_timing*, το οποίο παρέχεται με το *Cell BE SDK*, για την στατική ανάλυση του κώδικα της εφαρμογής. Η στατική ανάλυση του *pipeline* των SPEs έδειξε ότι η πλειονότητα των παγωμάτων

[2]Η πρόβλεψη της κατεύθυνσης των διακλαδώσεων στα SPEs δύναται να αλλαχθεί είτε μέσω εντολών που εισάγει ο προγραμματιστής (*intrinsics*) είτε μέσω εντολών *assembly* που εισάγονται από τον μεταγλωττιστή.

Optimization	Execution Time (sec)	Speed Up compared with (A)	Branch Misprediction stalls (% cycles)	Dual Issue
(A) Coarse grain parallelization	0.84	1	12.5%	11.6%
(B) Vectorization and 4x unroll	0.093	9.03	3.1%	20.1%
(C) 3x inner loop unroll	0.07	12	2.6%	22.6%

Πίνακας 4.1: Χρόνος εκτέλεσης, παράγοντας επιτάχυνσης, κόστος λανθασμένων προβλέψεων διακλάδωσης και χρήση του *dual-issue* πριν και μετά την εφαρμογή της τεχνικής του *ξεδιπλώματος βρόχου*.

του *pipeline* οφειλόταν στις μη ευθυγραμμισμένες προσβάσεις στην μνήμη που απαιτούνται για τον σχηματισμό των $4x4$ γειτονιών. Τα τέσσερα *pixels* στην κάθε γραμμή της γειτονιάς, για κάθε χρώμα, φορτώνονται ως αριθμοί κινητής υποδιαστολής απλής ακρίβειας στον ίδιο διανυσματικό καταχωρητή. Αυτή η μετατροπή από ακεραίους σε αριθμούς κινητής υποδιαστολής απλής ακρίβειας είναι απαραίτητη καθώς μετέπειτα υπολογισμοί στα *pixels* χρησιμοποιούν αριθμητική κινητής υποδιαστολής. Καθώς κάθε φόρτωση στον εκάστοτε διανυσματικό καταχωρητή εκτελείται ατομικά για κάθε *pixel* στην ίδια σειρά, οι εξαρτήσεις που δημιουργούνται από προηγούμενες φορτώσεις εισάγουν παγώματα σε μετέπειτα φορτώσεις στον ίδιο καταχωρητή, μετατρέποντας τον διανυσματικό καταχωρητή σε ένα *σημείο ανταγωνισμού* (*contention point*).

Για την αποφυγή των παγωμάτων λόγω των εξαρτήσεων που δημιουργούνται, αντιστρέφουμε την σειρά σύμφωνα με την οποία πραγματοποιούνται οι φορτώσεις των καταχωρητών. Αρχικά πραγματοποιείται η φόρτωση της πρώτης στήλης από *pixels*, έπειτα η φόρτωση της δεύτερης στήλης, κοκ. Και σε αυτό το σχήμα φόρτωσης συνεχίζουμε να χρησιμοποιούμε έναν καταχωρητή για την αποθήκευση κάθε γραμμής από *pixels* αλλά τροποποιούμε την ακολουθία βάσει της οποίας πραγματοποιείται η φόρτωση των δεδομένων (Σχήμα 2.6). Η τροποποίηση αυτή διαχωρίζει τις διαδοχικές εντολές που χρησιμοποιούν τον ίδιο καταχωρητή, εισάγοντας έναν ικανό αριθμό από εντολές ενδιάμεσα αυτών και αυξάνοντας το χρονικό διάστημα μεταξύ της φόρτωσης ενός *pixels* σε έναν καταχωρητή και της φόρτωσης του επόμενου *pixel* στο ίδιο καταχωρητή.

Θα πρέπει να σημειωθεί ότι αυτή η δρομολόγηση των εντολών αντιβαίνει στην κοινή διαίσθηση, καθώς οι βελτιστοποιήσεις που εισάγουν οι μεταγλωττιστές τείνουν να οργανώνουν τις προσβάσεις στην μνήμη κατά σειρές. Αυτή η βελτιστοποίηση είχε ως αποτέλεσμα την μείωση του χρόνου εκτέλεσης της εφαρμογής σε $0.06\ secs/frame$.

Διανυσματοποίηση Συνθηκών Βρόχων και Χειροκίνητη Δρομολόγηση Εντολών

Η συνάρτηση *Bicubic Interpolation* εμπεριέχει την ακόλουθη συνθήκη βρόχου ώστε να διαπιστωθεί κατά πόσο ένα σημείο με συντεταγμένες (X, Y) βρίσκεται εκτός των ορίων του *frame*. Σε περίπτωση που αυτό ισχύει, το χρώμα του αντίστοιχου *pixel* τίθεται σε μαύρο, το οποίο αντιστοιχεί σε μηδενική τιμή για κάθε μία από τις συνιστώσες χρώματος του συστήματος *RGB*. Οι τιμές αυτές αποθηκεύονται στον πίνακα *lpf*, ο οποίος χρησιμοποιείται ως ενδιάμεσος *buffer* όπου αποθηκεύονται τα αποτελέσματα από την εφαρμογή του βαθυπερατού φίλτρου στην κάθετη διεύθυνση:

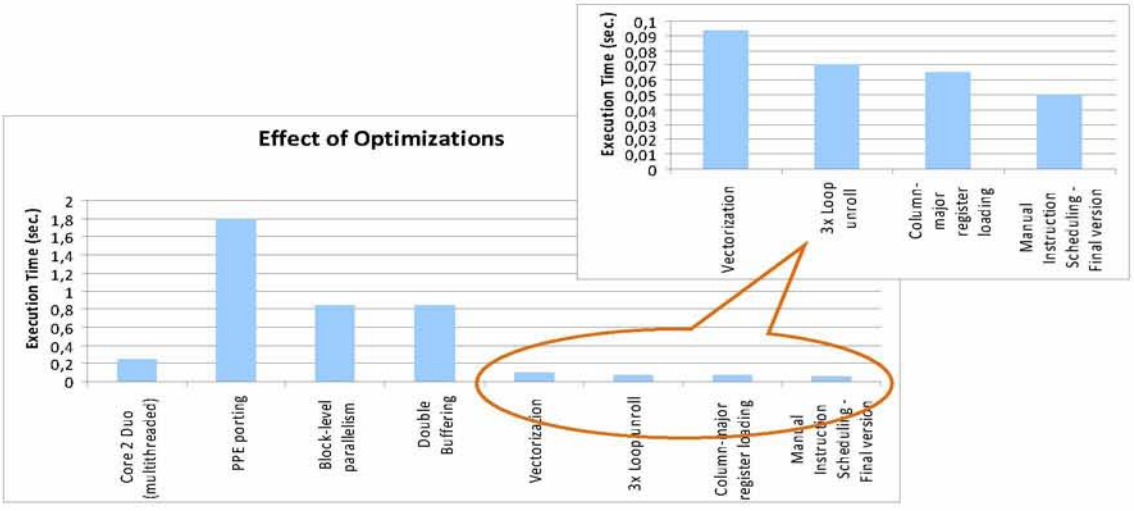
```
if ((X < 0) || (X > (Width − 1)) ||
    (Y < 0) || (Y > (Height− 1)))
    lpf[loc][0]=lpf[loc][1]=lpf[loc][2]=0;
```

Μετά την εφαρμογή των βελτιστοποιήσεων για την χρήση SIMD υπολογισμών, υπάρχουν τέσσερεις συνθήκες *if* στον εξωτερικό βρόχο όπως επίσης και οχτώ συντεταγμένες, ένα ζεύγος συντεταγμένων (X, Y) για καθένα από τα στοιχεία του διανυσματικού καταχωρητή, όπου οι τετμημένες αποθηκεύονται σε έναν καταχωρητή ενώ οι τεταγμένες αποθηκεύονται σε δεύτερο καταχωρητή. Και εδώ αντιμετωπίζουμε ένα πρόβλημα παρόμοιο με αυτό των μη ευθυγραμμισμένων φορτώσεων καθώς απαιτείται η εξαγωγή των συντεταγμένων από τον κάθε καταχωρητή, ο υπολογισμός της τιμής της συνθήκης και η εκτέλεση ή όχι των εντολών που βρίσκονται εντός της συνθήκης *if*. Όπως η φόρτωση στοιχείων στους καταχωρητές έτσι και η εξαγωγή στοιχείων από αυτούς, εισάγει παγώματα στο *pipeline* σε περίπτωση που διαδοχικές εντολές προσπελαύνουν τον ίδιο καταχωρητή.

Η προσέγγιση που ακολουθήσαμε για την βελτιστοποίηση αυτών των υπολογισμών, είναι η χρήση διανυσματικών πράξεων για τον υπολογισμό των αποτελεσμάτων των συνθηκών και η μεταφορά αυτών των υπολογισμών στην αρχή του εξωτερικού βρόχου. Ακόμη υπάρχει η ανάγκη για εξαγωγή των αποτελεσμάτων από τους καταχωρητές αλλά πλέον αυτό πραγματοποιείται στην αρχή του εξωτερικού βρόχου, αρκετούς κύκλους πριν από το σημείο όπου χρησιμοποιούνται αυτές οι τιμές.

Παρά την εξάλειψη των περισσοτέρων παγωμάτων του *pipeline* από την εφαρμογή των ανωτέρω βελτιστοποιήσεων, ο υπολειπόμενος αριθμός παγωμάτων, όπως αυτός προέκυπτε με χρήση του εργαλείου *spu_timing*, ήταν σημαντικός και επιβάρυνε αρκετά την απόδοση της εφαρμογής. Το τελευταίο στάδιο βελτιστοποίησης ήταν η εξάλειψη αυτών των παγωμάτων με χειροκίνητη δρομολόγηση των εντολών. Η διαδικασία που ακολουθήθηκε αποτελείται από τα ακόλουθα βήματα:

- Παραγωγή του κώδικα *assembly* για τον κώδικα που εκτελείται στα SPEs.



Σχήμα 4.5: Η απόδοση του αλγορίθμου διόρθωσης της παραμόρφωσης που προκαλείται από τους ευρυγώνιους φακούς στον επεξεργαστή *Core2 Duo* και στην αρχιτεκτονική *CBEA* για όλες τις βελτιστοποιήσεις.

Βελτιστοποίηση	Κύρια επίδραση στην απόδοση
DMA Double-Buffering	Μείωση του αδρανούς χρόνου και επικάλυψη μεταφοράς δεδομένων με υπολογισμούς
Διανυσματοποίηση υπολογισμών (έμμεσο ξεδίπλωμα βρόχου με παράγοντα 4x)	Μείωση των εντολών διακλάδωσης και αύξηση των διαθέσιμων εντολών προς δρομολόγηση
Ξεδίπλωμα εσωτερικού βρόχου	Όμοια με την προηγούμενη βελτιστοποίηση
Φόρτωση καταχωρητών κατά στήλες	Εξάλειψη των παγωμάτων λόγω μη ευθυγραμμισμένων φορτώσεων και διαδοχικών χρήσεων του ιδίου καταχωρητή
Χειροκίνητη δρομολόγηση εντολών	Μείωση αρκετών παγωμάτων του *pipeline* μέσω της ταυτόχρονης δρομολόγησης ανεξάρτητων εντολών και της εξάλειψης των διαφόρων εξαρτήσεων

Πίνακας 4.2: Οι βελτιστοποιήσεις που πραγματοποιήθηκαν και η κύρια επίδραση αυτών στην απόδοση της εφαρμογής.

- Χρήση του εργαλείου *spu_timing* για την επισημείωση του κώδικα *assembly* με τις πληροφορίες εκτέλεσης της κάθε εντολής στο *pipeline*.
- Από τις πληροφορίες που παράγονταν στο δεύτερο βήμα εντοπιζόντουσαν σημεία όπου υπήρχαν πολλά παγώματα ή μικρή χρήση του *dual-issue* και υπήρχε η δυνατότητα αναδιάταξης των εντολών. Έπειτα, αυτή η αναδιάταξη πραγματοποιούνταν στο υψηλότερο επίπεδο του κώδικα, όπου γινόταν αναδιάταξη των εντολών της γλώσσας *C* με σκοπό την εξάλειψη των παγωμάτων του *pipeline*.

Ακολουθώντας την ανωτέρω διαδικασία, πραγματοποιήσαμε την διεμπλοκή, *interleaving*, των εντολών για μη ευθυγραμμισμένες προσβάσεις στην μνήμη με εντολές που πραγματοποιούν υπολογισμούς επιτρέποντας στον μεταγλωττιστή μία πιο αποδοτική δρομολόγηση των διαθέσιμων εντολών. Το ξεδίπλωμα του εσωτερικού βρόχου (ο βρόχος που διατρέχει τις τρεις συνιστώσες χρώματος), όπως περιγράφεται στην υποενότητα 4.2, διευκολύνει την χειροκίνητη δρομολόγηση των εντολών μέσω του μεγάλου αριθμού των εντολών για τις οποίες υπάρχει η δυνατότητα διεμπλοκής. Εδώ θα πρέπει να αναφερθεί ότι ο μεταγλωττιστής δεν απεδείχθη αρκετά αποδοτικός στην δρομολόγηση ανεξάρτητων εντολών καθώς η δρομολόγηση των εντολών πραγματοποιούνταν μόνο σε τοπικό επίπεδο.

Ο πίνακας 4.2 συνοψίζει τις βελτιστοποιήσεις που πραγματοποιήθηκαν και την κύρια επίδραση αυτού στην απόδοση της εφαρμογής και το σχήμα 4.5 απεικονίζει την απόδοση του αλγορίθμου μετά από την εφαρμογή κάθε μετασχηματισμού βελτιστοποίησης που περιγράφηκε στην παρούσα ενότητα. Όπως παρατηρούμε, η εκμετάλλευση του *coarse-grain* παραλληλισμού μεταξύ των δεδομένων με χρήση των διαθέσιμων SPEs, του *fine-grain* παραλληλισμού, όπως η διανυσματοποίηση των υπολογισμών, και η εξάλειψη των εντολών διακλάδωσης προς τα πίσω που υπάρχουν στους βρόχους αποδείχθηκαν ως οι πιο αποτελεσματικές βελτιστοποιήσεις. Η τελική έκδοση του κώδικα απαιτεί μόνο $0.05\ secs$ για την επεξεργασία ενός *frame* ώστε επιτυγχάνεται η διόρθωση $20\ frames/sec$. Χαρακτηριστική είναι και η αύξηση της χρήσης του *dual-issue* από 22.6% σε 34.6% μετά την εφαρμογή της χειροκίνητης δρομολόγησης των εντολών, όπως προκύπτει από το *profiling*

της εφαρμογής στον προσομοιωτή.

Βάσει των αποτελεσμάτων, παρόλο που η εκτέλεση της εφαρμογής είτε με ένα είτε με δύο νήματα στον επεξεργαστή *Core2 Duo* ήταν αποτελεσματικότερη από την εκτέλεση αυτής στο PPE κατά έναν παράγοντα ίσο με $4.29x$ και $7.5x$ αντίστοιχα, ο συνδυασμός του PPE (ως επεξεργαστή ελέγχου) και των 6 διαθέσιμων SPEs (ως επεξεργαστές για την εκτέλεση των υπολογισμών) οδηγεί στην πιο γρήγορη εκτέλεση της εφαρμογής κατά έναν παράγοντα ίσο με $4.8x$ σε σύγκριση με την πολυνηματική εκτέλεση στον επεξεργαστή *Core2 Duo* όταν ο κώδικας έχει βελτιστοποιηθεί (βλ. την γραμμή με το μαύρο χρώμα στο σχήμα 4.3). Αυτή η παρατήρηση επιβεβαιώνει την δυνατότητα της αρχιτεκτονικής CBEA για την εκμετάλλευση του παραλληλισμού σε πολλαπλά επίπεδα που υπάρχει στις διάφορες εφαρμογές. Όμως, για την εκμετάλλευση αυτού του παραλληλισμού είναι αναγκαία η εφαρμογή εκτεταμένων βελτιστοποιήσεων στον κώδικα της εφαρμογής.

4.3 Σχετικές Εργασίες

Ο επεξεργαστής CBE αρχικά προοριζόταν για την αγορά των πολυμέσων και των ηλεκτρονικών παιχνιδιών. Ωστόσο σύντομα προσέλκυσε το ενδιαφέρον της επιστημονικής κοινότητας λόγω της μεγάλης υπολογιστικής δύναμης και της μεγάλης αποδοτικότητας τόσο σε σχέση με την ισχύ που καταναλώνεται όσο και με την τιμή αυτού.

Έχει χρησιμοποιηθεί επιτυχώς σε μία πλειάδα εφαρμογών όπως δημιουργία ιατρικών εικόνων [39], υπολογιστική βιολογία [45], προσομοιώσεις μεταφοράς σωματιδίων [23], εξόρυξη γνώσης [6, 11], on-line συστήματα ανίχνευσης εισβολών σε δίκτυα υπολογιστών [46], εικονοαπόδοση του περιβάλλοντος (*terrain rendering*) [43] ακόμη και σε εφαρμογές υπολογιστικής άλγεβρας [38]. Είναι επίσης μία δημοφιλής πλατφόρμα για την ανάπτυξη εφαρμογών επεξεργασίας πολυμέσων όπως αποκωδικοποίηση βίντεο υψηλής ανάλυσης [7] και αναγνώριση ομιλίας [30]. Αυτού του είδους οι εφαρμογές χαρακτηρίζονται από μεγάλες απαιτήσεις σε υπολογιστική ισχύ, μικρό λόγο μεταφοράς δεδομένων προς υπολογισμούς και συχνά απαιτούν επεξεργασία σε πραγματικό χρόνο.

Οι αυστηρές απαιτήσεις ευθυγράμμισης που θέτει η αρχιτεκτονική του επεξεργαστή συχνά αποτελεί έναν σημαντικό ανασχετικό παράγοντα της απόδοσης υπολογιστικών πυρήνων που βασίζονται σε υπολογισμούς με *stencils*. Στις εργασίες [28, 70] οι συγγραφείς αποτιμούν την εκτέλεση τέτοιου είδους εφαρμογών στον επεξεργαστή CBE και προτείνουν την εφαρμογή των τεχνικών του *blocking* και του *time skewing* [71] για την αύξηση της τοπικότητας και την αύξηση των υπολογισμών έναντι της επικοινωνίας που απαιτεί η εφαρμογή.

Η μη ώριμη τεχνολογία των υπαρχόντων μεταγλωττιστών δημιούργησε έντονο ενδιαφέρον για την ανάπτυξη μεταγλωττιστών με εξειδικευμένες δυνατότητες για τον επεξεργαστή CBE. Οι συγγραφείς στην εργασία [19] προτείνουν τεχνικές μεταγλώττισης που στοχεύουν στην αυτόματη παραγωγή αποδοτικού κώδικα για τον επεξεργαστή *Cell*. Αυτές οι τεχνικές περιλαμβάνουν ευθυγράμμιση μνήμης και πρόβλεψη διακλαδώσεων υποβοηθούμενες από τον μεταγλωττιστή, παραλληλοποίηση σε επίπεδο νημάτων και σε επίπεδο πράξεων SIMD και *caching* με λογισμικό που ελέγχεται από τον μεταγλωττιστή. Παρόμοια, οι *Zhao* και *Kennedy* [74] παρουσιάζουν έναν μεταγλωττιστή για τον επεξεργαστή *Cell* ο οποίος προσφέρει αυτοματοποιημένη παραλληλοποίηση των βρόχων που βασίζεται σε ανάλυση των εξαρτήσεων και αυτόματη SIMD παραλληλοποίηση των υπολογισμών.

Κεφάλαιο 5

Σχεδιαστικά Trade-offs και Ανάλυση Ευαισθησίας

Η ανάπτυξη μίας εφαρμογής με εγγενή πολυεπίπεδο παραλληλισμό σε μία πολύπλοκη και ετερογενή παράλληλη αρχιτεκτονική, όπως ο επεξεργαστή CBE, απαιτεί την λήψη διαφόρων σχεδιαστικών αποφάσεων από τον προγραμματιστή. Σε αυτό το κεφάλαιο αναλύονται οι διάφοροι *αντιπραγματισμοί* (*trade-offs*) που αντιμετωπίσθηκαν κατά την ανάπτυξη της εφαρμογής και πραγματοποιείται μία ανάλυση της ευαισθησίας της εφαρμογής σε διάφορες σχεδιαστικές παραμέτρους.

5.1 Μέγεθος των Tiles

Το μέγεθος και το σχήμα των *tiles* είναι ένας πολύ σημαντικός παράγοντας σε εφαρμογές όπου χρησιμοποιείται η τεχνική του *tiling*. Το μέγεθος του *tile* θα πρέπει να είναι αρκετά μεγάλο ώστε να εκμεταλλευτεί όλη την διαθέσιμη μνήμη *cache*, ή την διαθέσιμη LS στην αρχιτεκτονική του CBEA, προκειμένου να μεγιστοποιηθεί η επαναχρησιμοποίηση των δεδομένων και το σύνολο εργασίας, με την ταυτόχρονη ελαχιστοποίηση του κόστους που απαιτείται για την επικοινωνία. Η τελευταία παρατήρηση βασίζεται στο γεγονός ότι τεμαχισμός της εικόνας σε μικρότερο αριθμό από *tiles* έχει ως επιπλέον αποτέλεσμα την ελαχιστοποίηση του κόστους που απαιτείται για την αρχικοποίηση των απαιτούμενων αιτήσεων DMA για την μεταφορά των *tiles*. Συν τοις άλλοις, το μέγεθος των *tiles* θα πρέπει να είναι μικρό λόγω της περιορισμένης χωρητικότητας της *LS* των SPEs αλλά και της κύρτωσης των *tiles* της εικόνας εισόδου.

Όμως, στην περίπτωση της διόρθωσης της παραμόρφωσης που προκαλείται από την χρήση ευρυγώνιων φακών, εκτός από το μέγεθος της τοπικής μνήμης των SPEs και του *αποτυπώματος της μνήμης* (*memory footprint*) των άλλων δεδομένων που πρέπει να αποθηκευθούν, θα πρέπει να ληφθούν υπόψη και άλλες παράμετροι του προβλήματος. Στις περισσότερες περιπτώσεις, το τμήμα της εικόνας εισόδου που πρέπει να μεταφερθεί είναι κυρτό και όχι ορθογώνιο, όπως παρουσιάζεται στο Σχήμα 4.1. Δεδομένων των περιορισμών ευθυγράμμισης στην πρόσβαση της μνήμης και των περιορισμένων δυνατοτήτων του μηχανισμού DMA, ο χειρισμός των κυρτών περιοχών καθίσταται αρκετά περίπλοκος και εισάγει επιπλέον καθυστερήσεις στην εκτέλεση της εφαρμογής. Για την αποφυγή αυτού του κόστους, κάθε SPE μεταφέρει την ορθογώνια περιοχή που περικλείει το αντίστοιχο *tile*.

Συν τοις άλλοις, επιπλέον περιορισμοί επιβάλλονται από τα χαρακτηριστικά της αρχιτεκτονικής του επεξεργαστή στο σχήμα και στο μέγεθος της ορθογώνιας περιοχής που περικλείει κάθε *tile*. Επί παραδείγματι, το μήκος της ορθογώνιας περιοχής θα πρέπει να είναι πολλαπλάσιο του 4,

ώστε να διευκολυνθεί η διαδικασία της διανυσματοποίησης των υπολογισμών.

Επίσης, κάθε γραμμή από *pixels* της περιοχής θα πρέπει να ξεκινά από διεύθυνση η οποία είναι πολλαπλάσια των $16B$, καθώς αυτός είναι ο ελάχιστος περιορισμός ευθυγράμμισης που θα πρέπει να ικανοποιείται ώστε να είναι δυνατή η μεταφορά δεδομένων με μέγεθος μεγαλύτερο των $16B$ μέσω του μηχανισμού DMA.

Λόγω της κυρτότητας που παρουσιάζει η εικόνα εισόδου, *tiles* πολύ μικρού μεγέθους έχουν ως αποτέλεσμα την μεταφορά ενός μεγάλου αριθμού από *pixels* τα οποία βρίσκονται εκτός της περιοχής ενδιαφέροντος, αλλά πρέπει να μεταφερθούν καθώς βρίσκονται εντός της ορθογώνιας περιοχής. Τα επιπλέον *pixels* θα πρέπει να διορθωθούν οπότε αυξάνεται τόσο ο χρόνος μεταφοράς όσο και ο χρόνος της επεξεργασίας των *tiles*.

Το ίδιο αποτέλεσμα έχουν και *tiles* τα οποία είναι πολύ εκτεταμένα προς την μία ή την άλλη κατεύθυνση. Γι' αυτό τον λόγο δεν ήταν δυνατή και η ικανοποίηση του περιορισμού ευθυγράμμισης των 128 $bytes$ για την βέλτιστη μεταφορά δεδομένων, όπως αναφέρεται στην υποενότητα 3.3. Σε αυτή την περίπτωση, θα έπρεπε το μήκος της κάθε ορθογώνιας περιοχής να είναι πολλαπλάσιο του 128, κάτι που οδηγεί στα προαναφερθέντα, μη επιθυμητά αποτελέσματα.

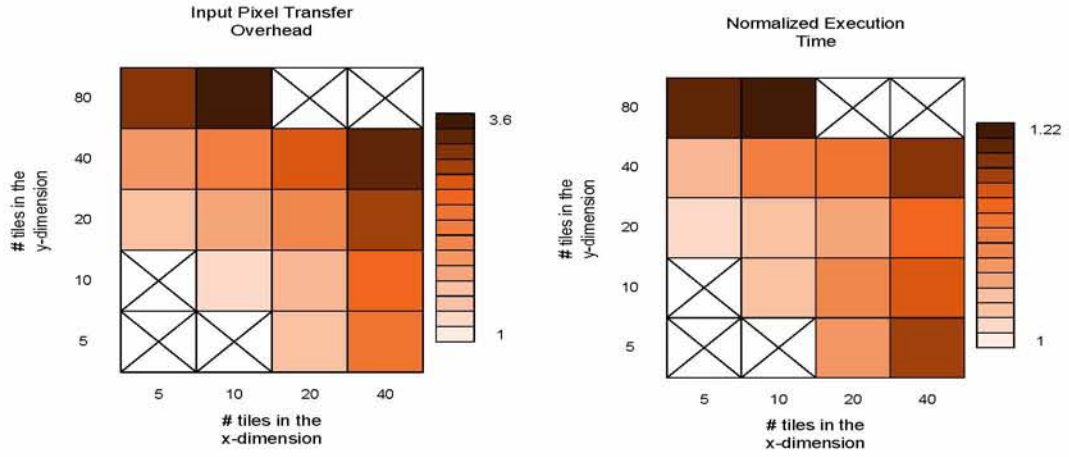
Το διάγραμμα (a) της Εικόνας 5.1 παρουσιάζει το επιπλέον κόστος (ο αριθμός των *pixels* που μεταφέρονται προς τον συνολικό αριθμό των *pixels* του *frame* - *transfer overhead*) για την μεταφορά των *pixels* εισόδου που απαιτούνται για την διόρθωση ενός *frame*, για όλα τα δυνατά μεγέθη και σχήματα των *tiles* της εικόνας εξόδου, βάσει των προαναφερθέντων περιορισμών που επιβάλλονται. Το διάγραμμα (b) απεικονίζει τον κανονικοποιημένο[1] χρόνο εκτέλεσης της εφαρμογής με χρήση των 6 διαθέσιμων SPEs του *PS3*. Οι μη εφικτοί συνδυασμοί λόγω των διαφόρων περιορισμών της αρχιτεκτονικής του επεξεργαστή CBEA επισημειώνονται με X.

Τα δύο διαγράμματα καταδεικνύουν μία στενή σχέση μεταξύ του χρόνου εκτέλεσης της εφαρμογής και του μεγέθους των δεδομένων που μεταφέρονται για τα διάφορα μεγέθη των *tiles*. Επιπλέον, είναι εμφανές ότι *tiles* με πολύ μικρό μέγεθος ή *tiles* τα οποία είναι πολύ εκτεταμένα προς την μία ή την άλλη κατεύθυνση (συνδυασμοί που αντιστοιχούν στην άνω αριστερή ή στην κάτω δεξιά γωνία) δεν είναι επιθυμητά. Το ποσοστό των επιπλέον *pixels* που μεταφέρονται κυμαίνεται από 9% έως 331%. Η δυνητική αύξηση του χρόνου εκτέλεσης της εφαρμογής, αναφορικά με τον χρόνο εκτέλεσης της εφαρμογής για το βέλτιστο μέγεθος των *tiles* ισούται με 22%. Βάσει αυτών των παρατηρήσεων και των αποτελεσμάτων, ο τεμαχισμός του *frame* εξόδου που χρησιμοποιείται αποτελείται από 5 *tiles* στην οριζόντια κατεύθυνση (*x-dimension*) και 20 *tiles* στην κάθετη διεύθυνση (*y-dimension*).

5.2 Επιμερισμός του Κόστους της Αντίστροφης Απεικόνισης

Η *αντίστροφη απεικόνιση*, δηλαδή η διαδικασία του υπολογισμού για κάθε *pixel* του *frame* εξόδου του αντιστοίχου κλασματικού *pixel* στο *frame* εισόδου, είναι μία διαδικασία που συγκαταλέγεται στους υπολογιστικά πολυπλοκότερους πυρήνες της εφαρμογής (Σχήμα 4.2). Όπως προαναφέρθηκε, η αντιστοιχία των συντεταγμένων των *pixels* μεταξύ εισόδου και εξόδου εξαρτάται αποκλειστικά από την *περιοχή ενδιαφέροντος* (ROI) και το *πεδίο θέασης* (FoV). Αυτές οι δύο παράμετροι του αλγορίθμου διόρθωσης μπορούν να αλλαχθούν διαδραστικά κατά την εκτέλεση του αλγορίθμου. Αυτό όμως είναι κάτι που είτε δεν συμβαίνει ποτέ είτε συμβαίνει με πολύ μικρή συχνότητα σε τυπικά σενάρια χρήσης των εφαρμογών *videoconferencing*. Επομένως, μπορούμε να επιμερίσουμε

[1] Ο χρόνος εκτέλεσης είναι κανονικοποιημένος αναφορικά με τον βέλτιστο χρόνο εκτέλεσης που επιτυγχάνεται όταν το *frame* εξόδου αποτελείται από 5 *tiles* στην οριζόντια κατεύθυνση (*x-dimension*) και από 20 *tiles* στην κάθετη διεύθυνση.



Σχήμα 5.1: Αριθμός *pixels* που μεταφέρονται προς τον συνολικό αριθμό των *pixels* του *frame* (διάγραμμα (a)) και κανονικοποιημένος χρόνος εκτέλεσης της εφαρμογής (διάγραμμα (b)).

το κόστος για την διαδικασία της *αντίστροφης απεικόνισης* μέσω της εκτέλεσης αυτού του υπολογισμού μόνο για 1 *frame* και της επαναχρησιμοποίησης των συντεταγμένων που παράγονται στα επόμενα *frames*, έως ότου μεταβληθεί κάποια από τις προαναφερθείσες παραμέτρους.

Αυτή η επαναχρησιμοποίηση έχει ως αποτέλεσμα την αύξηση των απαιτήσεων σε μνήμη της εφαρμογής: το μέγεθος της δομής δεδομένων που χρησιμοποιείται για την αποθήκευση των συντεταγμένων ισούται με $4.8\ MB$ καθώς πρέπει να αποθηκευθούν $1280x960$ ζεύγη αριθμών κινητής υποδιαστολής απλής ακρίβειας. Μία τέτοιου μεγέθους δομή δεδομένων δεν θα μπορούσε να αποθηκευθεί στην τοπική μνήμη των SPEs. Βάσει αυτών, αξιολογήθηκε την εναλλακτική λύση της αποθήκευσης των συντεταγμένων στην κύρια μνήμη και την μεταφορά των αντιστοίχων *blocks* κλασματικών συντεταγμένων στην τοπική μνήμη του κάθε SPE, βάσει του *tile* της εικόνας εξόδου που αυτό επεξεργάζεται.

Αυτό το *trade-off* που εμφανίζεται μεταξύ υπολογισμών και επικοινωνίας παρουσιάζει ιδιαίτερο ενδιαφέρον, δεδομένης της μεγάλης υπολογιστικής δύναμης του CBEA. Χρησιμοποιώντας την βιβλιοθήκη *SPU Timer Library* [60] βρέθηκε ότι οι αυξημένες απαιτήσεις σε επικοινωνία, ακόμη και μετά την εφαρμογή της τεχνικής του *double-buffering*, είχαν ως αποτέλεσμα τον τετραπλασιασμό του αριθμού των κύκλων ρολογιού που η εφαρμογή δαπανά κατά την αναμονή για την ολοκλήρωση των αιτήσεων DMA. Παρά την αύξηση του χρόνου που απαιτείται για την ολοκλήρωση των αιτήσεων, ο χρόνος εκτέλεσης της εφαρμογής για μία ακολουθία 10 *frames* μειώθηκε κατά μέσο όρο σε $0.045\ sec/frame$ όταν χρησιμοποιούνταν και τα 6 διαθέσιμα SPEs του *PS3*, αυξάνοντας έτσι την απόδοση της εφαρμογής σε $22\ frames/sec$.

Η τροποποιημένη έκδοση της εφαρμογής εκτελέστηκε στον προσομοιωτή χρησιμοποιώντας και τα 8 διαθέσιμα SPEs αυτού. Με χρήση των δυνατοτήτων *profiling* υπολογίσθηκε ότι απαιτούνταν $0.033\ secs$ για την διόρθωση ενός *frame*, το οποίο μεταφράζεται σε $30\ frame/sec$. Αυτός ο χρόνος εκτέλεσης αντιστοιχεί σε μία επιτάχυνση ίση με $7.27x$ αναφορικά με την πολυνηματική έκδοση της εφαρμογής στον επεξεργαστή *Core2 Duo*. Αυτή είναι και η τροποποίηση που επιτρέπει την εκτέλεση της εφαρμογής σε *πραγματικό χρόνο*. Το Σχήμα 5.2 παρουσιάζει τον αντίστοιχο ψευδοκώδικα για την έκδοση της εφαρμογής μετά από αυτή την τροποποίηση.

Μία άλλη παράμετρος του *trade-off* μεταξύ υπολογισμών και επικοινωνίας είναι ο υπολογισμός

```
1: {Input: The frames (in the wide-angle space) to be corrected}
2: {Output: The corrected frames (in the perspective space)}
3: Partition the output frame to blocks
4: for all frames do
5:    if FoV has changed then
6:       for all pixels in the output frame do
7:          Compute the corresponding fractional position in the input frame ( InverseMapping())
8:       end for
9:       Calculate the area of the input frame required for the calculation of each output frame block
10:   end if
11:   {Output blocks are statically partitioned to SPEs and processed concurrently on different SPEs}
12:   for all output blocks i do
13:      Fetch (async. DMA) input data for output block i + 1
14:      Fetch (async. DMA) fractional coordinates in the wide-angle space for each pixel of the output
         block i + 1
15:      Store (async. DMA) output block i − 1
16:      for all pixels in output block i do
17:         Interpolate the pixel value at that fractional position ( BicubicInterpolation(), SIMDized)
18:      end for
19:      Apply a 2-D low-pass filter to resize the output block ( LPF(), SIMDized)
20:   end for
21: end for
```

Σχήμα 5.2: Ψευδοκώδικας για τον αλγόριθμο διόρθωσης (τελική, βελτιστοποιημένη έκδοση).

εκ των προτέρων και η αποθήκευση στην κύρια μνήμη των συντελεστών $U_i(s), V_j(t)$ της παρεμβολής αντί των συντεταγμένων s και t. Αυτοί οι συντελεστές εξαρτώνται μόνο από τις τιμές των συντεταγμένων s και t οπότε είναι δυνατός ο υπολογισμός αυτών για μία φορά στο PPE και η μεταφορά αυτών στα SPEs για την εκτέλεση της *bicubic* παρεμβολής. Το μειονέκτημα αυτής της προσέγγισης είναι ότι πλέον απαιτείται η μεταφορά και η αποθήκευση στα SPEs οχτώ συντελεστών, δηλαδή οχτώ αριθμών κινητής υποδιαστολής απλής ακρίβειας. Αυτό θα απαιτούσε τον τεμαχισμό της εικόνας σε *tiles* μικρότερου μεγέθους για την αύξηση του διαθέσιμου χώρου στην LS των SPEs, με τα μη επιθυμητά αποτελέσματα που αυτό έχει στον χρόνο εκτέλεσης της εφαρμογής, όπως αναλύθηκε στην ενότητα 5.1.

5.3 Προ-επεξεργασία των Δεδομένων Εισόδου στο PPE

Όπως προαναφέρθηκε, οι μη ευθυγραμμισμένες προσβάσεις στην μνήμη είναι ένας παράγοντας που επηρεάζει σημαντικά την απόδοση στην εκτέλεση του κώδικα των SPEs. Αυτού του είδους οι προσβάσεις είναι χαρακτηριστικό των εφαρμογών που εμπεριέχουν υπολογισμούς με χρήση *stencil* (στην παρούσα εφαρμογή ως *stencil* νοείται η $4x4$ γειτονιά), όπου το *stencil* διατρέχει τα δεδομένα εισόδου. Τα δεδομένα κάθε *frame* εισόδου οργανώνονται ως ένας $2D$ πίνακας από *pixels* με τρία *bytes* για κάθε *pixel*. Κάθε *byte* αποτελεί την τιμή για την αντίστοιχη συνιστώσα χρώματος στο *RGB format*. Ο κώδικας που χρησιμοποιείται για την φόρτωση των δεδομένων στους καταχωρητές των SPEs εκτελεί την ακόλουθη διαδικασία:

- Αποπολύπλεξη των *bytes* που αντιστοιχούν σε διαφορετικές συνιστώσες χρώματος, καθώς κάθε συνιστώσα χρώματος φορτώνεται σε έναν διαφορετικό καταχωρητή των $128\ bits$ ώστε να είναι δυνατή η επεξεργασία αυτής ανεξάρτητα από τις άλλες συνιστώσες.

- Μετατροπή (αναβάθμιση) των $8-bit$ ακεραίων σε $32-bits$ αριθμούς κινητής υποδιαστολής για την εκτέλεση των αντίστοιχων πράξεων.

- Αποθήκευση σε έναν καταχωρητή τεσσάρων τιμών που αντιστοιχούν σε μία συνιστώσα χρώματος και σε τέσσερα διαδοχικά *pixels*.

Για την ελαχιστοποίηση της προ-επεξεργασίας των δεδομένων στα SPEs, διερευνήσαμε την προ-επεξεργασία των δεδομένων στο PPE. Τα δεδομένα εισόδου μετατρέπονται σε αριθμούς κινητής υποδιαστολής και αποθηκεύονται στην κύρια μνήμη κατά την φόρτωση του *frame* ενώ πραγματοποιείται και η απαραίτητη αποπολύπλεξη των διαφορετικών συνιστωσών χρώματος, όπου κάθε συνιστώσα αποθηκεύεται σε έναν ξεχωριστό $1D$ πίνακα. Με αυτή την προεπεξεργασία η πιθανότητα ευθυγραμμισμένης φόρτωσης δεδομένων σε έναν καταχωρητή είναι ίση με 25%. Για τον διπλασιασμό της πιθανότητας ευθυγραμμισμένης πρόσβασης, δημιουργήσαμε ένα δεύτερο αντίγραφο των *tiles* εισόδου στο οποίο αποθηκεύουμε τα ολισθημένα κατά δύο θέσεις αριστερά δεδομένα του πρώτου μισού του πίνακα. Με αυτό το σχήμα πλεονασμού επιτυγχάνεται πιθανότητα ευθυγραμμισμένης πρόσβασης ίση με 50%.

Τα κύρια μειονεκτήματα από αυτή την προσέγγιση είναι ο πολλαπλασιασμός με έναν παράγοντα ίσο με 8 τόσο του απαιτούμενου αποθηκευτικού χώρου (*memory footprint*) του κάθε *tile* στην LS των SPEs όσο και του όγκου των δεδομένων για τα *tiles* εισόδου που πρέπει να μεταφερθούν προς επεξεργασία στα SPEs.

Η ανωτέρω τροποποίηση είχε ως αποτέλεσμα την αύξηση του χρόνου εκτέλεσης της εφαρμογής κατά 30%. Για την εύρεση του κύριου περιοριστικού παράγοντα στην απόδοση της εφαρμογής χρησιμοποιήθηκε ο προσομοιωτής συστήματος. Όπως προέκυψε, η αύξηση του χρόνου εκτέλεσης οφειλόταν στην μη επικάλυψη του αυξημένου χρόνου που απαιτούνταν για την μεταφορά των δεδομένων με την εκτέλεση υπολογισμών. Συν τοις άλλοις, το επιπλέον κόστος του κώδικα για την προ-επεξεργασία των δεδομένων, καθιστούσε το PPE τον κύριο ανασχετικό παράγοντα της εφαρμογής, με αποτέλεσμα τον περαιτέρω περιορισμό της απόδοσης της εφαρμογής.

5.4 Χρήση DMA Lists για την Μεταφορά των Δεδομένων

Ο μηχανισμός που χρησιμοποιείται για την μεταφορά των δεδομένων του κάθε *tile* από το PPE στο αντίστοιχο SPE είναι αυτός των ασύγχρονων αιτήσεων DMA. Για κάθε γραμμή του εκάστοτε *tile* πραγματοποιείται μία ξεχωριστή αίτηση ενώ αιτήσεις για γραμμές που ανήκουν στο ίδιο *tile* λαμβάνουν το ίδιο *αναγνωριστικό αίτησης* (*tag id*). Καθώς ο μηχανισμός των αιτήσεων DMA είναι ασύγχρονος, αυτό το αναγνωριστικό χρησιμοποιείται για την αναμονή προς ολοκλήρωση των αντιστοίχων αιτήσεων. Αυτές οι συνεχόμενες αιτήσεις εισάγουν ένα επιπλέον κόστος στην εκτέλεση της εφαρμογής, το οποίο προέρχεται από τον χρόνο που απαιτείται για την αρχικοποίηση του μηχανισμού. Επίσης, σε πολλές περιπτώσεις, ο περιορισμένος αποθηκευτικός χώρος που διαθέτει το κάθε SPE για την αποθήκευση των εκκρεμών αιτήσεων ενδέχεται να μην επαρκεί.

Για την αποφυγή αυτού του κόστους, εκμεταλλευτήκαμε την συνέχεια του χώρου μνήμης στην LS όπου αποθηκεύονται τα *pixels* για κάθε *tile* όπως επίσης και την ασυνέχεια του αντίστοιχου χώρου στην κύρια μνήμη ώστε να χρησιμοποιήσουμε τον μηχανισμό των *λιστών DMA* (*DMA lists*) για την μεταφορά των δεδομένων προς την LS. Έτσι, πολλαπλές συνεχόμενες αιτήσεις DMA αντικαθίστανται από μία αίτηση για την μεταφορά μίας λίστας DMA. Όσον αφορά στην μεταφορά δεδομένων από την LS προς την κύρια μνήμη, η ασυνέχεια του χώρου της LS από τον οποίο μεταφέρονται δεδομένα δεν επιτρέπει την χρήση των *λιστών DMA*.

5.5 Εκμετάλλευση της Ομοιότητας μεταξύ Διαδοχικών Frames

Το κυρίως κίνητρο για την βελτιστοποίηση του αλγορίθμου είναι η εφαρμογή του σε συστήματα *videoconferencing*. Όπως γνωρίζουμε, ένα *video* είναι μία ακολουθία από *frames*. Βάσει αυτού, μία προφανής βελτιστοποίηση σε αλγοριθμικό επίπεδο είναι η εκμετάλλευση της ομοιότητας που εμφανίζεται μεταξύ διαδοχικών *frames*. Σε αυτή την περίπτωση, δεν απαιτείται η εκ νέου εφαρμογή του αλγορίθμου σε *tiles* διαδοχικών *frames* τα οποία δεν μεταβάλλονται σε μεγάλο βαθμό.

Για την εύρεση των διαφορών μεταξύ διαδοχικών *tiles* είναι απαραίτητη η εφαρμογή ενός είδους *εκτίμηση κίνησης* (*Motion Estimation*). Οι αλγόριθμοι *εκτίμησης κίνησης* χρησιμοποιούνται σε πρότυπα για συμπίεση/αποσυμπίεση βίντεο (π.χ. *H.264, AVS*) και τείνουν να έχουν μεγάλη υπολογιστική πολυπλοκότητα και απαιτήσεις σε εύρος ζώνης. Επί παραδείγματι, η πλειονότητα των αλγορίθμων απαιτεί την μεταφορά του προηγουμένου *frame* πολλαπλές φορές λόγω της επικάλυψης που υπάρχει μεταξύ των διαφόρων *παραθύρων ενδιαφέροντος*. Δεδομένης της μικρής χωρητικότητας της LS των SPEs, η πιθανότητα για σημαντική επιβράδυνση της εφαρμογής λόγω των πολλαπλών μεταφορών από την κύρια μνήμη στην LS είναι σημαντική. Επίσης, λόγω της μεγάλης υπολογιστικής πολυπλοκότητας είναι πολύ πιθανό ο χρόνος εκτέλεσης του αλγορίθμου *εκτίμησης κίνησης* να ξεπερνά κατά πολύ τον χρόνο που εξοικονομείται από την μη εφαρμογή του αλγορίθμου διόρθωσης στα αντίστοιχα *tiles*.

Κεφάλαιο 6

Μελέτη της Απόδοσης της Εφαρμογής

Σε αυτό το κεφάλαιο επιχειρείται μία λεπτομερής ανάλυση της απόδοσης της εφαρμογής στον επεξεργαστή *Cell*, σε έναν επεξεργαστή *Core2 Quad* από την *Intel* και σε μία *Virtex-4 LX80* FPGA. Το κεφάλαιο αυτό διαφέρει από το Κεφάλαιο 4 καθώς όλα τα αποτελέσματα αναφέρονται στην εκτέλεση της εφαρμογής σε πραγματικό *hardware* χωρίς την ύπαρξη αποτελεσμάτων από προσομοιώσεις. Με την εκτέλεση της εφαρμογής σε πραγματικό *hardware* επιτυγχάνεται η λήψη πιο λεπτομερών μετρήσεων τόσο για μετρικές που αναφέρθηκαν και στο Κεφάλαιο 4 όσο και για μετρικές χαμηλού επιπέδου οι οποίες δεν ήταν δυνατόν να ληφθούν μέσω προσομοιώσεων. Αρχικά περιγράφεται το σύστημα με τον επεξεργαστή *Cell* που χρησιμοποιήθηκε, ο επεξεργαστής *Intel Core2 Quad* και η *Virtex-4 LX80* FPGA και έπειτα γίνεται η ανάλυση των διαφόρων μετρήσεων. Η ανάλυση αυτή στοχεύει στον χαρακτηρισμό της εφαρμογής σε αντιπροσωπευτικές πλατφόρμες από τις πιο σύγχρονες κατηγορίες συστημάτων που χρησιμοποιούνται για την ανάπτυξη εφαρμογών.

6.1 Περιγραφή των Συστημάτων

IBM BladeCenter QS20

Το σύστημα *IBM BladeCenter QS20* αποτελείται από ένα μοναδικό *blade* που ενσωματώνει δύο επεξεργαστές CBEA. Περισσότερα του ενός συστήματα *IBM BladeCenter QS20* μπορούν να συνδεθούν μεταξύ τους για την δημιουργία ενός πιο ισχυρού υπολογιστικού συστήματος. Το σύστημα που χρησιμοποιήθηκε για την εκτέλεση της εφαρμογής και την λήψη των μετρήσεων για τον επεξεργαστή CBEA ενσωματώνει εξωτερική μνήμη *DRAM* μεγέθους ίσου με $1\ GB$.

Intel Core2 Quad

Ο επεξεργαστής *Intel Core2 Quad Q9300* είναι ένας αντιπροσωπευτικός επεξεργαστής που ανήκει στους ομογενείς επεξεργαστές πολλαπλών πυρήνων. Η συχνότητα λειτουργίας του επεξεργαστή είναι ίση με $2.5\ GHz$ και η συχνότητα λειτουργίας του Front-Side Bus (FSB) είναι ίση με $1.3\ GHz$. Ο επεξεργαστής είναι οργανωμένος ως δύο *blocks* επεξεργαστών διπλού πυρήνα, οι οποίοι επικοινωνούν με χρήση του FSB. Κάθε ένα από τα δύο *blocks* διπλού πυρήνα εμπεριέχει μία *L2 cache* μεγέθους $3\ MB$, (12 *way set associative*), με μέγεθος *cache line* ίσο με $64\ bytes$ η οποία διαμοιράζεται μεταξύ των δύο πυρήνων του *block*. Επιπλέον, κάθε πυρήνας έχει αποκλειστική πρόσβαση σε μία *L1 cache* μεγέθους $64\ KB$ ($32\ KB$ για δεδομένα και $32\ KB$ για τον κώδικα, 8 *way set associative*, με μέγεθος *cache line* ίσο με $64\ bytes$). Ο επεξεργαστής υποστηρίζει το

σύνολο εντολών SSE 4.1 και η μέση κατανάλωση ισχύος του επεξεργαστή είναι ίση με $95\ W$. Το σύστημα που χρησιμοποιήθηκε για την συγκέντρωση των αντίστοιχων αποτελεσμάτων εμπεριέχει μία μνήμη *RAM*, μεγέθους $2\ GB$ και με συχνότητα λειτουργίας ίση με $667\ MHz$.

Αναδιατασσόμενη Λογική

Σε αντίθεση με τις δύο προαναφερθείσες πλατφόρμες *αμετάβλητης λογικής* (*fixed logic*), η τρίτη πλατφόρμα που χρησιμοποιήθηκε είναι μία συσκευή αναδιατασσόμενης λογικής (FPGA). Μία τέτοιου είδους συσκευή είναι ουσιαστικά ένας υψηλής πυκνότητας πίνακας με *blocks* λογικής τα οποία διαμορφώνονται μετά την κατασκευή της συσκευής [13]. Η λειτουργικότητα των FPGAs καθορίζεται με χρήση των λεγόμενων *configuration bits*, τα οποία καθορίζουν την λειτουργικότητα των επιμέρους λογικών μονάδων και της λογικής που χρησιμοποιείται για την διασύνδεση των λογικών μονάδων. Πρόσφατες FPGAs ενσωματώνουν και τμήματα από σταθερή λογική (*hard IP logic*) όπως μικροεπεξεργαστές γενικού σκοπού, τμήματα για Digital Signal Processing (DSP) και λογική για την υλοποίηση Static Random Access Memories (SRAMs) εντός της FPGA.

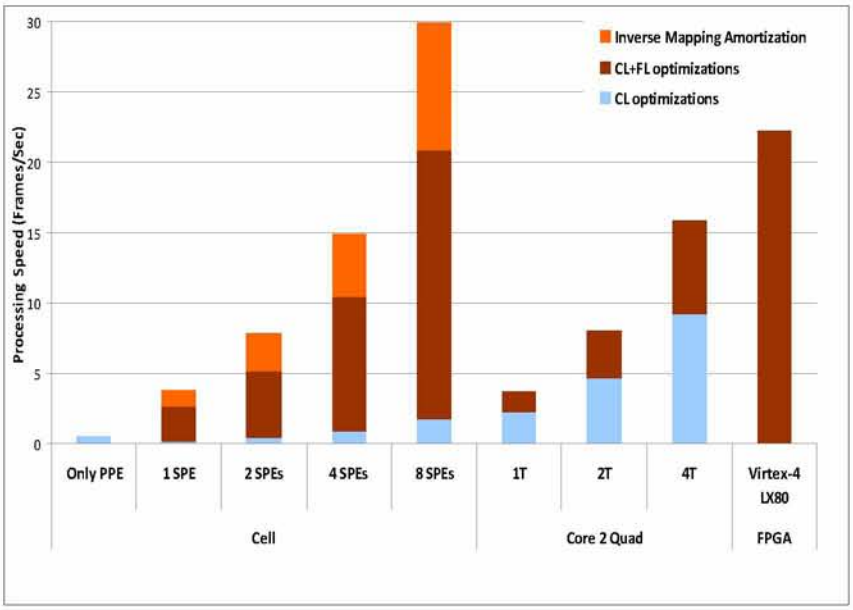
Σε σύγκριση με τα συστήματα Application Specific Instruction Set (ASIC) τα οποία δημιουργούνται με γνώμονα τις ανάγκες συγκεκριμένης εφαρμογής ή κλάσης εφαρμογών οπότε προσφέρουν ελάχιστη ευελιξία, οι FPGAs προσφέρουν το πλεονέκτημα την μεγάλης ευελιξίας όσον αφορά στην προσαρμογή της αρχιτεκτονικής στις ανάγκες της εκάστοτε εφαρμογής. Σε αντίθεση με τους επεξεργαστές *Core2 Quad* και CBEA, οι FPGAs δεν ακολουθούν την παραδοσιακή αρχιτεκτονική συνόλου εντολών που βασίζεται στο μοντέλο *von-Neumann*. Επομένως, υπάρχει το *trade-off* μεταξύ των διαθέσιμων υπολογιστικών πόρων και της απόδοσης της εφαρμογής καθώς ο προγραμματιστής έχει την δυνατότητα να επιλέξει το απαραίτητο επίπεδο παραλληλισμού για την εφαρμογή. Αυτή η ευελιξία καθιστά τις FPGAs ενεργειακά αποδοτικότερες, συγκρινόμενες με οποιοδήποτε σύστημα γενικού σκοπού.

Τα κύρια μειονεκτήματα αυτής της αρχιτεκτονικής είναι η μεγάλη πολυπλοκότητα για την ανάπτυξη εφαρμογών και η μικρή συχνότητα λειτουργίας παρά την μεγάλη κλιμάκωση της τεχνολογίας κατασκευής των FPGAs[1]. Το πρώτο μειονέκτημα προκύπτει από την φύση των FPGAs καθώς ο προγραμματισμός τους γίνεται με χρήση Hardware Description Languages (HDL) (*VHDL* ή *Verilog*). Αυτές οι γλώσσες απαιτούν αφενός μεν αρκετό χρόνο για την ανάπτυξη της εφαρμογής, αφετέρου δε την εξειδικευμένη γνώση λεπτομερειών χαμηλού επιπέδου της αρχιτεκτονικής του συστήματος. Παρ' όλο που υπάρχει μία συνεχής τάση για τον προγραμματισμό των FPGAs με χρήση γλωσσών προγραμματισμού υψηλού επιπέδου, οι περισσότεροι προγραμματιστές συνεχίζουν να χρησιμοποιούν γλώσσες HDL.

Όσον αφορά στην συχνότητα λειτουργίας του ρολογιού, αυτή είναι μικρότερη (τουλάχιστον κατά μία τάξη μεγέθους) από τις συχνότητες λειτουργίας των επεξεργαστών υψηλής απόδοσης που έχουν σχεδιασθεί χρησιμοποιώντας *full custom* σχεδίαση. Στην πραγματικότητα, όλες οι σχεδιάσεις FPGA λειτουργούν σε συχνότητα ρολογιού μικρότερη των $200\ MHz$.

Για την υλοποίηση του *module* που υλοποιεί τον αλγόριθμο διόρθωσης, χρησιμοποιήθηκε μία *Virtex-4 LX80* FPGA. Αυτή η συσκευή εμπεριέχει 80460 λογικά *cells*, 200 SRAMs εντός του *chip* με μέγεθος $18\ Kbit$ έκαστη, και 80 τμήματα για την εκτέλεση DSP υπολογισμών. Το *module* που υλοποιεί τον αλγόριθμο είναι τμήμα ενός ενσωματωμένου *Συστήματος σε Chip* (System on Chip (SoC)), το οποίο εμπεριέχει έναν επεξεργαστή *Microblaze*, έναν *multi-port* ελεγκτή μνήμης για την παροχή υψηλού εύρους ζώνης πρόσβαση στην εξωτερική μνήμη *SDRAM* και στα υπόλοιπα περιφερειακά. Η συσκευή χρησιμοποιεί ένα ρολόι συχνότητας $62.5\ MHz$ [72].

[1]Οι νεότερης τεχνολογίας Virtex-6 FPGAs από την Xilinx θα υλοποιηθούν σε τεχνολογία CMOS 40nm.



Σχήμα 6.1: Χρόνος εκτέλεσης και κλιμάκωση της απόδοσης.

Η δημιουργία του *module* είναι πλήρως αυτοματοποιημένη και πραγματοποιήθηκε με την χρήση ενός εργαλείου που βασίζεται στο *architectural synthesis* μοντέλο προγραμματισμού [8]. Το εργαλείο Computer Aided Design (CAD) παράγει *hardware accelerators* οι οποίοι ακολουθούν το *streaming* παράδειγμα αρχιτεκτονικής [3]. Αυτή η τεχνική παράγει τις απαραίτητες μονάδες για την ανάγνωση/αποθήκευση των δεδομένων που χρησιμοποιούνται για το *prefetching* των δεδομένων όπως και το *data path* το οποίο χρησιμοποιείται για την εκτέλεση του προγράμματος και διατυπώνεται μέσω ενός *streaming διαγράμματος ροής δεδομένων* (*streaming Data Flow Graph - sDFG*).

Η υλοποίηση του αλγορίθμου στην FPGA εκμεταλλεύεται την ευελιξία των διαθέσιμων πόρων οπότε είναι δυνατή η δρομολόγηση ενός μεγάλου αριθμού από πράξεις των *sDFG* διαγραμμάτων (περίπου 400 πράξεις σε όλα τα στάδια του *pipeline*). Η δρομολόγηση των εντολών σε κάθε κύκλο ρολογιού δημιουργείται από το εργαλείο CAD χρησιμοποιώντας το σχήμα *modulo scheduling* [44].

6.2 Ανάλυση της Απόδοσης της Εφαρμογής

Σε αυτή την ενότητα αξιολογείται η απόδοση του αλγορίθμου διόρθωσης στις πλατφόρμες που περιγράφηκαν στην ενότητα 6.1. Όλα τα αποτελέσματα έχουν ληφθεί από την εκτέλεση της εφαρμογής σε πραγματικό υλικό παρά σε προσομοιωτές.

Η παραλληλοποίηση της εφαρμογής στον επεξεργαστή *Core2 Quad* έγινε με χρήση της βιβλιοθήκης *POSIX Threads* καθώς η βιβλιοθήκη *OpenMP* στο λειτουργικό *Linux* παρουσίαζε προβλήματα στην κλιμάκωση της εφαρμογής για χρήση περισσοτέρων των δύο νημάτων. Για την ανάλυση της απόδοσης της εφαρμογής χρησιμοποιήθηκαν το λογισμικό *Intel VTune Performance Analyzer* [16] και το λογισμικό *Intel Thread Profiler* [15].

Όσον αφορά στην συλλογή των χαμηλού επιπέδου δεδομένων για την απόδοση της εφαρμογής

στον επεξεργαστή CBEA χρησιμοποιήθηκε το εργαλείο *Cell Performance Counter* (CPC) το οποίο χρησιμοποιείται για την αρχικοποίηση και την παρακολούθηση των *hardware performance counters* στον επεξεργαστή CBEA. Αυτοί οι *counters* χρησιμοποιούνται για τον ακριβή προσδιορισμό συγκεκριμένων γεγονότων στο *hardware* που πραγματοποιούνται κατά την εκτέλεση της εφαρμογής. Σε αντίθεση με το *PlayStation3*, στο σύστημα *IBM QS20* είναι δυνατή η χρήση των *hardware performance counters* λόγω της απουσίας του *hypervisor*.

Για την μεταγλώττιση του κώδικα για τους δύο επεξεργαστές χρησιμοποιήθηκαν τα εργαλεία και οι επιλογές μεταγλώττισης που περιγράφονται στο Κεφάλαιο 4.

Μελέτη Απόδοσης και Κλιμάκωσης του Αλγορίθμου

Το σχήμα 6.1 απεικονίζει την απόδοση της εφαρμογής σε κάθε πλατφόρμα βάσει του αριθμού των *frames* που είναι δυνατόν να επεξεργαστούν σε κάθε δευτερόλεπτο μετά την εφαρμογή της κάθε κατηγορίας βελτιστοποιήσεων. Στην κατηγορία των *Coarse-Level* (CL) βελτιστοποιήσεων ανήκει η εφαρμογή του *2D tiling* ενώ στην κατηγορία των *Fine-Level* (FL) βελτιστοποιήσεων ανήκουν όλες οι βελτιστοποιήσεις που εφαρμόζονται σε πιο χαμηλό επίπεδο όπως για παράδειγμα η εκμετάλλευση του DLP που υπάρχει στην εφαρμογή.

Οι *FL* βελτιστοποιήσεις αφορούν μόνο στους δύο επεξεργαστές καθώς η μόνη βελτιστοποίηση που έχει εφαρμοσθεί στην FPGA είναι η εφαρμογή της τεχνικής του *tiling*. Η οργάνωση των δεδομένων σε *tiles* επιτρέπει την εκτέλεση των διαφορετικών σταδίων του αλγορίθμου σε ένα *pipeline* σχήμα οπότε σε κάθε χρονική στιγμή πολλαπλά *tiles* επεξεργάζονται στα διάφορα στάδια του *pipeline*. Επί παραδείγματι, όταν το *tile* N βρίσκεται στο στάδιο της παρεμβολής, το *tile* $N+1$ βρίσκεται στο στάδιο της *αντίστροφης απεικόνισης*. Όσον αφορά στην βελτιστοποίηση *Inverse Mapping Amortization* αυτή αναφέρεται στον επιμερισμό του κόστους της *αντίστροφης απεικόνισης* όπως αυτός περιγράφεται στην ενότητα 5.2.

Μία άμεση παρατήρηση είναι η γραμμική εξάρτηση της ταχύτητας εκτέλεσης της εφαρμογής από τον αριθμό των διαθέσιμων νημάτων εκτέλεσης που προσφέρει η κάθε πλατφόρμα. Αυτό επαληθεύει και τα αποτελέσματα που είχαν ληφθεί από τον προσομοιωτή όπως αυτά περιγράφονται στην υποενότητα 4.1. Σε αυτό συμβάλλουν και οι ιεραρχίες μνήμης και διαύλου οι οποίες καθιστούν δυνατή την πλήρη εκμετάλλευση των δυνατοτήτων των *data paths* το οποίο σημαίνει ότι η εφαρμογή είναι *compute bound* για όλες τις περιπτώσεις που μετρήθηκαν. Εάν υποθέσουμε ότι απαιτούμε διόρθωση 20 *frames* το δευτερόλεπτο, τότε μόνο ο επεξεργαστής CBEA και η FPGA μπορούν να έχουν την επιθυμητή απόδοση. Σε όλες τις περιπτώσεις η αναδιατασσόμενη λογική είναι αυτή που υπερέχει έναντι των Central Processing Units (CPUs) που ακολουθούν το μοντέλο *von-Neumann*, καθώς είναι 37.5 φορές πιο γρήγορη για κάθε *Hz* από τον CBEA και 56 φορές πιο γρήγορη από τον επεξεργαστή *Intel Core2 Quad*.

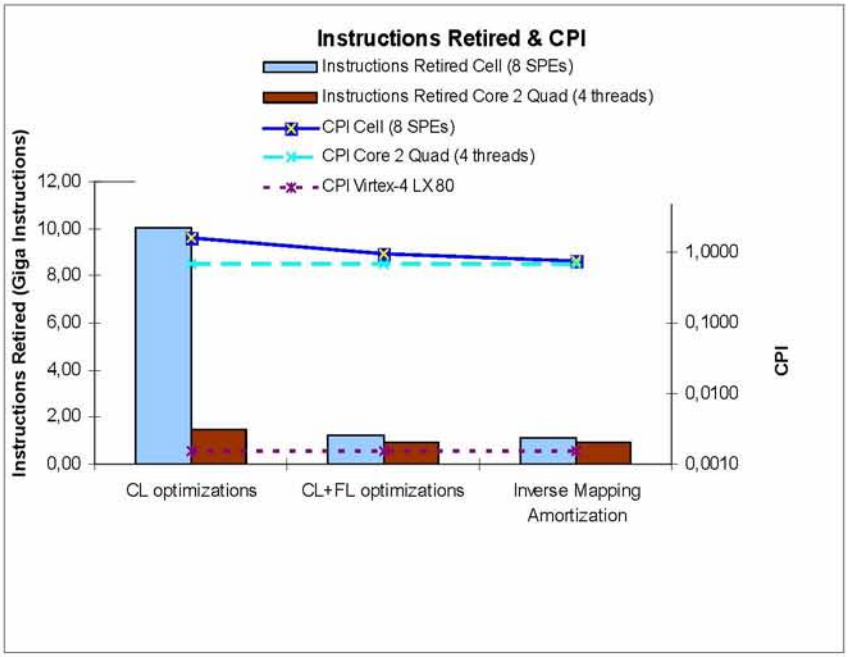
Η εκμετάλλευση του *fine-level* παραλληλισμού μέσω της διανυσματοποίησης των υπολογισμών και της τεχνικής του ξεδιπλώματος βρόχου αποφέρει επιτάχυνση της εφαρμογής ίση με $12.4x$ για τον επεξεργαστή CBEA και επιτάχυνση ίση με $1.7x$ για τον επεξεργαστή *Core2 Quad* σε σύγκριση μόνο με την εφαρμογή των *CL* βελτιστοποιήσεων. Αυτό αποτελεί απόδειξη της προσεχτικής επανεγγραφής του κώδικα (συνήθως μη αυτόματα) που πρέπει να υπάρξει προκειμένου να βελτιστοποιηθεί η εκτέλεση της εφαρμογής στα SPEs του επεξεργαστή CBEA. Η έλλειψη μηχανισμού για την πρόβλεψη διακλαδώσεων και η μη εξελιγμένη τεχνολογία των μεταγλωττιστών για την αυτόματη διανυσματοποίηση των υπολογισμών και την αποδοτική δρομολόγηση των εντολών επιβαρύνουν τον προγραμματιστή της εφαρμογής με το καθήκον της παραγωγής όσο το δυνατόν περισσότερο βελτιστοποιημένου κώδικα.

Στο σχήμα 6.2 παρουσιάζεται ο συνολικός αριθμός από εντολές που εκτελούνται και η τιμή

της μετρικής Cycles Per Instruction (CPI) για κάθε μία πλατφόρμα όταν όλοι οι διαθέσιμοι πόροι χρησιμοποιούνται. Θα πρέπει να σημειωθεί ότι η τιμή CPI (περίπου 0.0015) για την FPGA είναι περίπου 500 φορές μικρότερη από την τιμή CPI των άλλων δύο CPUs.

Για την καλύτερη κατανόηση της διαφορετικής επίδρασης που έχουν οι βελτιστοποιήσεις SIMD στους επεξεργαστές CBEA και *Core2 Quad* παρουσιάζουμε την λεπτομερή ανάλυση των χρόνων εκτέλεσης για κάθε συνάρτηση του κώδικα της εφαρμογής στο σχήμα 6.3 (η βελτιστοποίηση *IMA* αναφέρεται στον επιμερισμό του κόστους της αντίστροφης απεικόνισης). Όπως παρατηρούμε, τα SPEs του επεξεργαστή *Cell* δαπανούν το 75% του χρόνου για την εκτέλεση της *αντίστροφης απεικόνισης* στη περίπτωση που εφαρμόζονται οι *CL* βελτιστοποιήσεις ενώ ο επεξεργαστής *Intel Core2 Quad* δαπανά μόνο το 18% του αντίστοιχου χρόνου εκτέλεσης. Τα ποσοστά αυτά μειώνονται σε 15% και 0.3% αντίστοιχα όταν εφαρμόζονται οι *FL* βελτιστοποιήσεις.

Αυτή η διαφορά στους χρόνους εκτέλεσης οφείλεται κυρίως στην υπερέχουσα απόδοση της μονάδας για εκτέλεση πράξεων κινητής υποδιαστολής της αρχιτεκτονικής $x86$ και στην ικανότητα αυτής για εκτέλεση των στοιχειωδών συναρτήσεων στις σχέσεις (2.3) στο υλικό, σε αντίθεση με τον επεξεργαστή *Cell* όπου η εκτέλεση αυτών των συναρτήσεων πραγματοποιείται μέσω βιβλιοθηκών λογισμικού. Σε αυτή την περίπτωση, η εφαρμογή βελτιστοποιήσεων που διευκολύνουν την εκτέλεση διανυσματικών πράξεων είναι ιδιαιτέρως σημαντική για την μείωση του χρόνου εκτέλεσης της εφαρμογής. Όπως αναφέρθηκε στην αρχή της παρούσας υποενότητας, η εκτέλεση του αλγορίθμου στην FPGA είναι *pipelined* οπότε ο χρόνος εκτέλεσης είναι πάντα ίσος με τον χρόνο εκτέλεσης του πιο αργού τμήματος του αλγορίθμου, το οποίο είναι το σχήμα της *bicubic* παρεμβολής. Αυτό σημαίνει ότι στο σχήμα 6.3 υπονοείται πως τα στάδια του *βαθυπερατού φίλτρου* και της *αντίστροφης απεικόνισης* τελειώνουν πιο νωρίς και αναμένουν για την περάτωση του σταδίου της παρεμβολής.



Σχήμα 6.2: Ο συνολικός αριθμός εκτελούμενων εντολών και η τιμή του μεγέθους *CPI* για την βέλτιστη έκδοση του κώδικα της εφαρμογής.

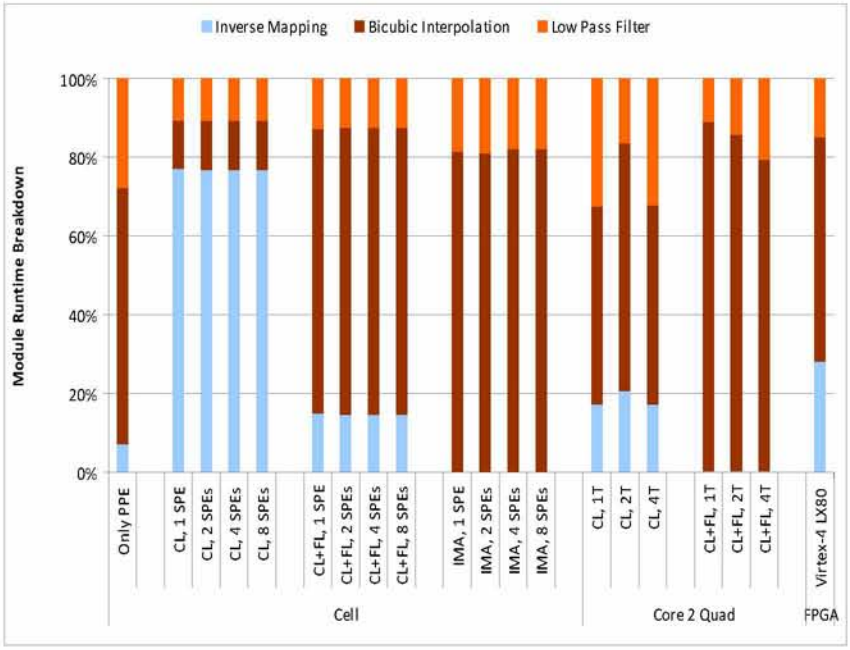
Μελέτη Απόδοσης του Συστήματος Μνήμης

΄Ενας σημαντικός περιοριστικός παράγοντας σε αρκετές *streaming* εφαρμογές είναι οι απαιτήσεις για το *off-chip* εύρος ζώνης. Το σχήμα 6.4 απεικονίζει τον όγκο των δεδομένων ανά δευτερόλεπτο που μεταφέρονται από την κύρια μνήμη προς τον επεξεργαστή για κάθε μία από τις τρεις αρχιτεκτονικές.

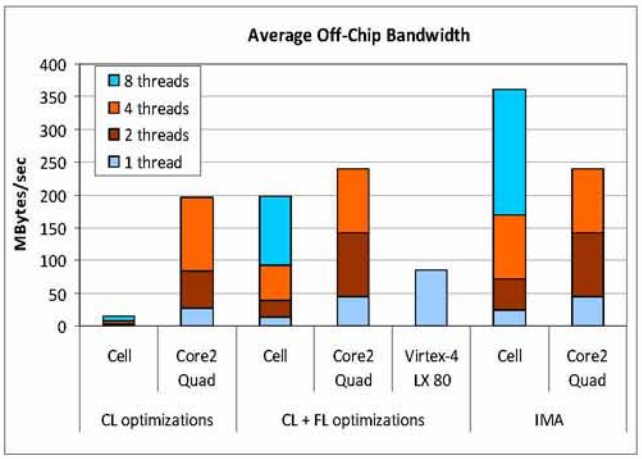
Ο όγκος των δεδομένων είναι σχεδόν πλήρως προβλέψιμος. Για την δημιουργία και αποθήκευση στην κύρια μνήμη ενός *frame* μεγέθους 900 $Kbytes$ συνολικά απαιτούνται περί τα 4.98 $MBytes$ από δεδομένα εισόδου. Ο επιμερισμός του κόστους της *αντίστροφης απεικόνισης* σε διαφορετικά *frames* ενδέχεται να δημιουργήσει επιπλέον ανάγκη για την μεταφορά δεδομένων. Σε περίπτωση που η χωρητικότητα των *caches* εξωτερικού επιπέδου - δηλαδή η χωρητικότητα της *L2 cache* και η συνολική χωρητικότητα των LSs του CBEA - δεν επαρκούν για την αποθήκευση του *συνόλου εργασίας* για τον υπολογισμό του *frame*, οι κλασματικές συντεταγμένες, το μέγεθος των οποίων ανέρχεται σε επιπλέον 1.32 MB για κάθε *frame*, θα πρέπει να μεταφερθούν από την κύρια μνήμη.

Η μέση τιμή του εύρους ζώνης που απαιτείται προσδιορίζεται κυρίως από τον χρόνο εκτέλεσης του αλγορίθμου διόρθωσης για κάθε *frame*. Η χειρότερη περίπτωση, τόσο σε όγκο δεδομένων που πρέπει να μεταφερθούν όσο και σε χρόνο εκτέλεσης της εφαρμογής, επιβάλλει την μεταφορά περισσοτέρων από 350 $MBytes$ το δευτερόλεπτο. Επομένως, η πίεση που επιβάλλεται στο σύστημα της μνήμης είναι κατά πολύ χαμηλότερη από το εύρος ζώνης που προσφέρεται και στις τρεις αρχιτεκτονικές.

Αυτή η παρατήρηση έρχεται σε αντίθεση με την κοινή πεποίθηση που θεωρεί την απόδοση της μνήμης *cache* ως τον κύριο περιοριστικό παράγοντα για εφαρμογές που ανήκουν στον *streaming* τομέα [12]. Ο όγκος των δεδομένων ανά εντολή που μεταφέρονται *off-chip* παρουσιάζεται



Σχήμα 6.3: Αναλυτική παράθεση του χρόνου εκτέλεσης κάθε συνάρτησης.



Σχήμα 6.4: Μέσο *off-chip* εύρος ζώνης.

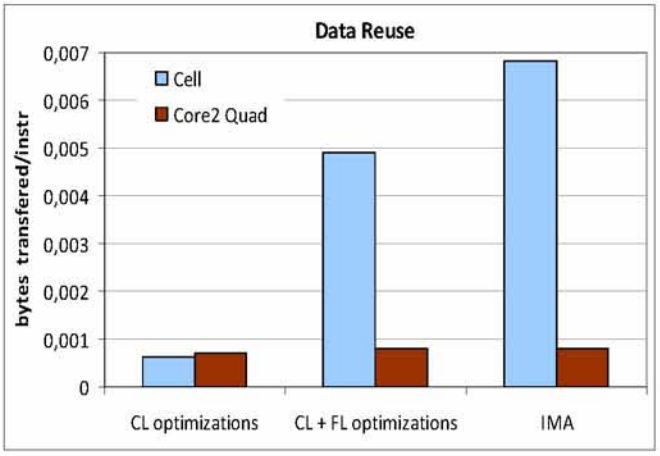
στο σχήμα 6.5. Οι ρυθμοί αναφέρονται στην εκτέλεση της εφαρμογής με τον μέγιστο βαθμό παραλληλισμού που επιτρέπει η κάθε πλατφόρμα (4 νήματα για τον επεξεργαστή *Core2 Quad* και 8 SPEs για τον επεξεργαστή CBEA). Τα αποτελέσματα δεν μεταβάλλονται σε περίπτωση που χρησιμοποιηθούν λιγότερα νήματα ή SPEs.

Ο μέγιστος αριθμός των *bytes* που απαιτείται να μεταφερθούν από την κύρια μνήμη για κάθε εντολή ανέρχεται σε 0.007 $bytes$. Αυτό αποδεικνύει τόσο την αποδοτικότητα της τεχνικής του *2D tiling* στην επαναχρησιμοποίηση των δεδομένων όσο και την *compute-intensive* φύση της εφαρμογής.

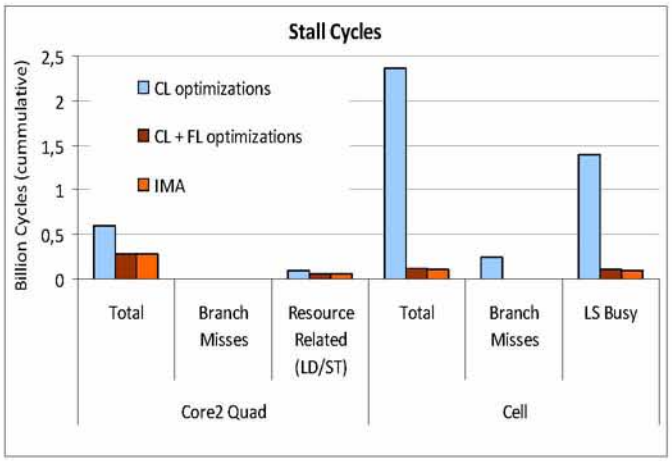
Συγκρινόμενος με τον ρυθμό για τον επεξεργαστή CBEA, ο αντίστοιχος ρυθμός είναι κατά πολύ μικρότερος για τον επεξεργαστή *Core2 Quad* λόγω του συνολικού μεγέθους της *cache* στο τελευταίο επίπεδο της ιεραρχίας (6 MB *L2 cache* έναντι 2 MB συνολικής μνήμης στις LS των SPEs).

Για τον επεξεργαστή CBEA, το σχήμα 6.5 αποκαλύπτει ότι ο επιμερισμός του κόστους της *αντίστροφης απεικόνισης* έχει ως αποτέλεσμα την μεταφορά επιπλέον δεδομένων για κάθε εντολή που εκτελείται. Αυτό είναι αναμενόμενο καθώς οι LS δεν επαρκούν για την αποθήκευση των δεδομένων εισόδου, εξόδου και των κλασματικών συντεταγμένων για κάθε *frame*. Αυτό έχει ως συνέπεια την συνεχή μεταφορά των κλασματικών συντεταγμένων από την κύρια μνήμη προς το κάθε SPE για την διόρθωση του εκάστοτε *tile*. Το διάγραμμα εμφανίζει επίσης ότι ο κώδικας της εφαρμογής για τον CBEA δεν είναι αρκετά αποδοτικός, όσον αφορά στην απόδοση της μνήμης, όταν εφαρμόζονται οι *FL* βελτιστοποιήσεις. Όμως, αυτό δεν είναι αληθές καθώς η παρατηρούμενη διαφορά μπορεί να αποδοθεί στην σημαντική μείωση του αριθμού των εντολών που συνεπάγεται η *διανυσματοποίηση των υπολογισμών* και η εφαρμογή της τεχνικής του *ξεδιπλώματος βρόχου*.

Θα πρέπει να αναφερθεί ότι η παρούσα ενότητα αναφέρεται στο εύρος ζώνης που απαιτεί η εφαρμογή *κατά μέσο όρο*. Ωστόσο, οι αιτήσεις για μεταφορά δεδομένων από την μνήμη τείνουν να πραγματοποιούνται κατά *ριπές* (*bursts*). Αυτό έχει ως αποτέλεσμα την δημιουργία παγωμάτων σε περίπτωση που το σύστημα μνήμης δεν έχει την ικανότητα να ικανοποιήσει αποδοτικά τις *ριπές* των αιτήσεων.



Σχήμα 6.5: Μεταφορές δεδομένων (σε *bytes*) ανά εντολή από την *off-chip* μνήμη.



Σχήμα 6.6: Αριθμός παγωμάτων (συνολικός αριθμός, παγώματα που σχετίζονται με τις διακλαδώσεις και παγώματα που σχετίζονται την μνήμη).

Ανάλυση των Παγωμάτων

Ο αριθμός των *παγωμάτων* (*stalls*) είναι μία μετρική που ποσοτικοποιεί τις καθυστερήσεις στην εκτέλεση της εφαρμογής που οφείλονται είτε σε έλλειψη πόρων είτε σε ανασχετικούς παράγοντες (*bottlenecks*) της αρχιτεκτονικής του συστήματος. Το σχήμα 6.6 παρουσιάζει τον συνολικό αριθμό από παγώματα και τον αριθμό των παγωμάτων που προκαλούνται από δύο πολύ σημαντικούς παράγοντες - την αλληλεπίδραση της εφαρμογής με το σύστημα της μνήμης και την λανθασμένη πρόβλεψη διακλαδώσεων - για τις διαφορετικές βελτιστοποιήσεις που εφαρμόσθηκαν στους δύο επεξεργαστές. Ο αριθμός των παγωμάτων είναι συγκεντρωτικός (για όλα τα ενεργά πλαίσια εκτέλεσης - *execution contexts*) και αναφέρεται στην εκτέλεση της εφαρμογής με τον μέγιστο βαθμό παραλληλισμού που επιτρέπει η κάθε πλατφόρμα (4 νήματα για τον επεξεργαστή *Core2 Quad* και 8 SPEs για τον επεξεργαστή CBEA). Τα αποτελέσματα δεν μεταβάλλονται σε περίπτωση που χρησιμοποιηθούν λιγότερα πλαίσια εκτέλεσης.

Εύκολα παρατηρούμε ότι ο επεξεργαστής CBEA είναι λιγότερο ανεκτικός στην εκτέλεση μη βέλτιστου κώδικα. Αυτό οφείλεται κυρίως στην αρκετά απλοποιημένη αρχιτεκτονική των SPEs, η οποία όμως συμβάλλει τα μέγιστα στην απόδοση του επεξεργαστή. Ο αριθμός των παγωμάτων του κώδικα όπου έχει εφαρμοσθεί το *tiling* είναι πολύ μεγαλύτερος για τον CBEA σε σχέση με τον επεξεργαστή *Core2 Quad*. Ωστόσο, η εφαρμογή των *Fine-Level* βελτιστοποιήσεων έχει ως αποτέλεσμα την εξάλειψη του μεγαλύτερου αριθμού των παγωμάτων για τον επεξεργαστή CBEA. Το ίδιο επίπεδο βελτιστοποιήσεων μειώνει περίπου κατά 50% τον αριθμό παγωμάτων στον επεξεργαστή *Core2 Quad*.

Ο εξελιγμένος μηχανισμός για την πρόβλεψη των διακλαδώσεων που ενσωματώνει ο επεξεργαστής *Intel* κατορθώνει να εξαλείψει τον αριθμό των παγωμάτων λόγω λανθασμένης πρόβλεψης διακλάδωσης. Στην αντίπερα όχθη, ο επεξεργαστής CBEA δεν ενσωματώνει κάποιο μηχανισμό πρόβλεψης διακλάδωσης. Κάθε διακλάδωση προβλέπεται ως ψευδής[2] και σε περίπτωση λανθασμένης πρόβλεψης η καθυστέρηση είναι ίση με 20 κύκλους ρολογιού ενώ η τυπική καθυστέρηση των εντολών κυμαίνεται από 2 έως 7 εντολές. Η εκτέλεση του κώδικα επιβαρύνεται με αυτή την καθυστέρηση στην περίπτωση των διακλαδώσεων προς τα πίσω που υπάρχουν σε κάθε βρόχο. Οι *Fine-Level* βελτιστοποιήσεις και κυρίως η *διανυσματοποίηση των υπολογισμών* και το *ξεδίπλωμα βρόχου* μειώνουν τον αριθμό των επαναλήψεων των βρόχων και τον αριθμό των προς τα πίσω διακλαδώσεων. Αυτού του είδους τα παγώματα μειώνονται περαιτέρω μέσω των κατάλληλων εντολών λογισμικού (*software hinting*).

Εκτός από τα παγώματα λόγω της πρόβλεψης διακλαδώσεων, ένας ακόμη σημαντικός παράγοντας που συμβάλλει στον αριθμό των παγωμάτων είναι η αλληλεπίδραση μεταξύ του επεξεργαστή και της ιεραρχίας μνήμης. Πρότυπα προσπέλασης στην μνήμη που πραγματοποιούν αιτήσεις κατά *ριπές*, εισάγουν παγώματα είτε λόγω συγκρούσεων για την πρόσβαση στις διαθέσιμες θύρες της μνήμης είτε λόγω έλλειψης διαθέσιμων θέσεων στις ουρές του επεξεργαστή όπου αποθηκεύονται οι αιτήσεις για ανάγνωση ή εγγραφή στην μνήμη.

Το πρόβλημα αυτό εντείνεται στον επεξεργαστή CBEA, όπου η LS διαθέτει μία μοναδική θύρα πρόσβασης. Οι αιτήσεις που δημιουργούνται από το SPE ενδέχεται να συγκρούονται μεταξύ τους ή με άλλες εντολές που αιτούνται πρόσβαση στην μνήμη, όπως για παράδειγμα οι αιτήσεις του μηχανισμού DMA. Για ακόμη μία φορά, παρατηρούμε ότι οι χαμηλού επιπέδου βελτιστοποιήσεις οδηγούν σε μείωση και αυτών των παγωμάτων. Η τεχνική του *ξεδιπλώματος βρόχου* επιτρέπει την δημιουργία πιο αποδοτικών δρομολογήσεων των εντολών καθώς υπάρχουν περισσότερες διαθέσιμες εντολές προς δρομολόγηση. Η μη αυτόματη δρομολόγηση των εντολών από τον προγραμματιστή, ιδίως στην περίπτωση του CBEA, μειώνει περαιτέρω τις συγκρούσεις μεταξύ των εντολών σε περιπτώσεις που ο μεταγλωττιστής αποδεικνύεται αρκετά συντηρητικός.

Οι αιτήσεις του μηχανισμού DMA για μεταφορές δεδομένων μεταξύ των LSs και της κύριας μνήμης ενδέχεται να εισάγουν επιπλέον καθυστέρηση, η οποία οφείλεται είτε στον ανταγωνισμό για τα διαθέσιμα *κανάλια* (*channels*) του EIB είτε στην καθυστέρηση που εισάγουν καθαυτές οι εντολές DMA. Η τεχνική του *double-buffering* αποδεικνύεται αρκετά αποδοτική στην επικάλυψη αυτών των καθυστερήσεων με την εκτέλεση υπολογισμών. Ο συνολικός αριθμός των παγωμάτων που οφείλεται στις αιτήσεις DMA είναι περιορισμένος και, στην χειρότερη περίπτωση, ανέρχεται σε 600 κύκλους.

[2]Η πρόβλεψη της διακλάδωσης δύναται να αλλαγεί στο επίπεδο του λογισμικού μέσω κατάλληλων *intrinsics* (*branch hints*) ή των αντίστοιχων εντολών *assembly*.

Κόστος Υλοποίησης

Το κόστος που απαιτείται για την υλοποίηση της εφαρμογής είναι ένας σημαντικός παράγοντας που πρέπει να ληφθεί υπόψη κατά την χρήση μίας νέας πλατφόρμας για την ανάπτυξη εφαρμογών. Μία πτυχή στην σύγκριση των προγραμματιστικών μοντέλων για τις τρεις χρησιμοποιούμενες αρχιτεκτονικές είναι η προσπάθεια που πρέπει να καταβληθεί από τον προγραμματιστή της εφαρμογής. Καθώς είναι αρκετά δύσκολο να υπολογισθούν ακριβή στατιστικά για τον χρόνο που απαιτείται για την ανάπτυξη μίας εφαρμογής, χρησιμοποιούμε το μέγεθος *Lines-of-Code* (*LOC*) ως την κύρια μετρική για την ποσοτικοποίηση της προγραμματιστικής προσπάθειας.

Ο αρχικός κώδικας της εφαρμογής στην γλώσσα προγραμματισμού *C*, όπου χρησιμοποιείται μόνο ένα νήμα εκτέλεσης, αποτελείται από 800 γραμμές πηγαίου κώδικα (*LOC*). Ο πλήρως βελτιστοποιημένος κώδικας της εφαρμογής για τον επεξεργαστή CBEA απαιτεί 1500 επιπλέον γραμμές κώδικα ενώ ο αντίστοιχος κώδικας για τον επεξεργαστή *x86* απαιτεί μόνο 500 επιπλέον γραμμές πηγαίου κώδικα. Η περιγραφή του Streaming Data Flow Graph (sDFG) για την FPGA αποτελείται από 800 γραμμές κώδικα που ομοιάζει με εντολές *assembly*. Συν τοις άλλοις, η υλοποίηση της εφαρμογής στην FPGA απαιτεί πολλαπλές και αρκετά χρονοβόρες επαναλήψεις για τις διαδικασίες της *σύνθεσης* (*synthesis*) και του *place & route*, οι οποίες θα πρέπει να ληφθούν υπόψη στο συνολικό κόστος της ανάπτυξης της εφαρμογής.

Βασιζόμενοι σε αυτές τις παρατηρήσεις, η αρχιτεκτονική του επεξεργαστή *Core2 Quad* προσφέρει ένα καλύτερο μοντέλο προγραμματισμού ενώ ο επεξεργαστής CBEA έχει ένα σχετικό πλεονέκτημα έναντι της FPGA. Παρόλα αυτά, βασιζόμενοι στον συνολικό χρόνο ανάπτυξης της εφαρμογής, πιστεύουμε ότι η σχεδίαση και η υλοποίηση αναδιατασσόμενων συστημάτων με χρήση γλωσσών υψηλού επιπέδου θα είναι αρκετά ανταγωνιστική σε σχέση με τα *manycore* και τα *multicore* συστήματα εάν το κόστος υλοποίησης για την εκάστοτε εφαρμογή ληφθεί υπόψη.

Κεφάλαιο 7

Επίλογος

Η πρωτοφανής υπολογιστική δύναμη του ετερογενούς επεξεργαστού πολλαπλών πυρήνων Cell Broadband Engine Architecture (CBEA) επέτρεψε την ανάπτυξη λύσεων σε λογισμικό για την επίλυση υπολογιστικά απαιτητικών αλγορίθμων με απαιτήσεις πραγματικού χρόνου (*real-time constraints*), οι οποίοι μέχρι πρότινος απαιτούσαν ειδική υποστήριξη από το υλικό. Η παρούσα εργασία επικεντρώνει στην ανάπτυξη ενός αλγορίθμου για την διόρθωση της παραμόρφωσης που προκαλείται από ευρυγώνιους φακούς σε πραγματικό χρόνο. Η διόρθωση παραμόρφωσης που προκαλείται από τους ευρυγώνιους φακούς είναι μία εφαρμογή *στρέβλωσης εικόνας* (*image warping application*) με εφαρμογή σε διάφορους επιστημονικούς τομείς. Κύριο χαρακτηριστικό αυτής της εφαρμογής είναι το στατικό αλλά μη κανονικό πρότυπο πρόσβασης στην μνήμη το οποίο καθιστά την προφόρτωση των δεδομένων αρκετά δύσκολη.

Σε αυτή την εργασία περιγράψαμε και αξιολογήσαμε πειραματικά τα αλγοριθμικά βήματα και τις οδηγούμενες από την αρχιτεκτονική του επεξεργαστή βελτιστοποιήσεις που απαιτούνται για την επίτευξη διόρθωσης σε πραγματικό χρόνο ($30\ fps$) για μία ακολουθία εικόνων που βασίζεται στο *format RGB*, με μέγεθος *frame* εισόδου $2592x1944$ και μέγεθος *frame* εξόδου $640x480$. Επίσης, διερευνήσαμε τα αποτελέσματα στην απόδοση της εφαρμογής διαφόρων *coarse* και *fine-grain* βελτιστοποιήσεων. Ξεκινώντας από απόδοση $4.17\ fps$ σε έναν επεξεργαστή *Core2 Duo*, εκτελέσαμε και βελτιστοποιήσαμε σταδιακά τον κώδικα της εφαρμογής στον επεξεργαστή CBE με αποτέλεσμα την επίτευξη απόδοσης ίση με $30\ fps$. Κατά την διάρκεια των βελτιστοποιήσεων, οι πιο προσοδοφόρες βελτιστοποιήσεις ήταν η απεικόνιση των υπολογιστικά πολυπλοκότερων πυρήνων της εφαρμογής στα SPEs (*task offloading*), η διανυσματοποίηση των υπολογισμών (*vectorization*) και η εξάλειψη των εντολών διακλάδωσης (*branch elimination*).

Συν τοις άλλοις, διερευνήσαμε αρκετές σχεδιαστικές επιλογές για την εκτίμηση της επίδρασης εναλλακτικών μετασχηματισμών του λογισμικού στην απόδοση της εφαρμογής. Όπως προέκυψε, η βέλτιστη απόδοση επιτυγχάνεται όταν η διαδικασία της *αντίστροφης απεικόνισης* (*Inverse Mapping*) υλοποιούνταν στο PPE (με χρήση των επεκτάσεων *AltiVec*) και έπειτα πραγματοποιούνταν η διανομή των κλασματικών συντεταγμένων στα SPEs. Αυτή η προσέγγιση επιτυγχάνει τον ελάχιστο χρόνο εκτέλεσης, ο οποίος ισούται με $0.033\ secs/frame$. Επίσης, πραγματοποιήσαμε μία λεπτομερή ανάλυση της απόδοσης της εφαρμογής στον επεξεργαστή *Cell*, σε έναν επεξεργαστή *Core2 Quad* από την *Intel* και σε μία *Virtex-4 LX80* FPGA.

Μία ενδιαφέρουσα διαπίστωση που προέκυψε από την υλοποίηση της εφαρμογής στον επεξεργαστή CBEA είναι ο σημαντικός βαθμός μη αυτόματης δρομολόγησης των εντολών που απαιτείται από την προγραμματιστή ώστε να μειωθούν τα παγώματα του *pipeline* που οφείλονται τόσο σε εξαρτήσεις μεταξύ εντολών όσο και στην μη βέλτιστη δρομολόγηση των εντολών για την

εκμετάλλευση του *dual-issue pipeline*. Αυτό επιδεικνύει την ανάγκη για πιο ώριμη τεχνολογία μεταγλωττιστών, οι οποίοι θα μπορούν να παράγουν βελτιστοποιημένο κώδικα.

Ένα ακόμη σημαντικό συμπέρασμα είναι οι ελάχιστες δυνατότητες που προσφέρει ο μηχανισμός DMA της αρχιτεκτονικής CBEA. Κυρίως για εφαρμογές που ανήκουν στο *streaming domain* και χαρακτηρίζονται από ένα μη κανονικό πρότυπο προσπέλασης στην μνήμη και τον μεγάλο όγκο δεδομένων που μεταφέρονται, η έλλειψη χαρακτηριστικών όπως τα μεγέθη *skip* ή *stride* καθιστούν την μεταφορά των δεδομένων ένα προκλητικό εγχείρημα για τον προγραμματιστή της εφαρμογής.

Τέλος, όπως προκύπτει από την πλειονότητα των βελτιστοποιήσεων, ο προγραμματιστής των εφαρμογών για επεξεργαστές πολλαπλών πυρήνων, και κυρίως για ετερογενείς επεξεργαστές πολλαπλών πυρήνων, θα πρέπει να έχει μία άριστη γνώση της αρχιτεκτονικής του συστήματος ώστε να μπορεί να εφαρμόζει τους απαραίτητους μετασχηματισμούς για την βελτιστοποίηση του κώδικα της εκάστοτε εφαρμογής.

Βιβλιογραφία

[1] V. Agarwal, M. S. Hrishikesh, S. W. Keckler, and D. Burger. Clock Rate versus IPC: The End of the Road for Conventional Microarchitectures. In *Proceedings of the* 27^{th} *Annual International Symposium on Computer Architecture*, 2000.

[2] S. Amarasinghe. Multicore Programming Primer: Learn and Compete in Programming the PLAYSTATION®3 Cell Processor. http://www.cag.csail.mit.edu/ps3/index.html, 2007.

[3] S. Amarasinghe and B. Thies. Architectures, Languages and Compilers for the Streaming Domain. *Tutorial at the 12th Annual International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2003.

[4] S. B. Baden. Parallel Computation (lecture slides). http://www.cse.ucsd.edu/classes/wi08/cse260/index.html, 2008.

[5] D. A. Bader. One-Day IBM Cell Programming Workshop at Georgia Tech. http://www.cc.gatech.edu/ bader/index.html, 2007.

[6] D. A. Bader, V. Agarwal, and K. Madduri. On the Design and Analysis of Irregular Algorithms on the Cell Processor: A Case Study of List Ranking. In *Proceedings of the 2007 International Parallel and Distributed Processing Symposium (IPDPS 2007)*, page 76, 2007.

[7] H. Baik, K-H. Sihn, Y-I. Kim, S. Bae, N. Han, and H. J. Song. Analysis and Parallelization of H.264 decoder on Cell Broadband Engine Architecture. In *Proceedings of the 2007 IEEE International Symposium on Signal Processing and Information Technology*, pages 791--795, December 2007.

[8] N. Bellas, S. Chai, M. Dwyer, and D. Linzmeier. Real-Time Fisheye Lens Distortion Correction Using Automatically Generated Streaming Accelerators. *Field-Programmable Custom Computing Machines, Annual IEEE Symposium on*, 0, 2009.

[9] F. Blagojevic, D. S. Nikolopoulos, A. Stamatakis, and C. D. Antonopoulos. Dynamic Multigrain Parallelization on the Cell Broadband Engine. In *ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP07)*, 2007.

[10] F. Blagojevic, D. S. Nikolopoulos, A. Stamatakis, and C. D. Antonopoulos. RAxML-Cell: Parallel Phylogenetic Tree Inference on the Cell Broadband Engine. In *21st IEEE International Parallel and Distributed Processing Symposium (IPDPS-07)*, 2007.

[11] G. Buehrer, S. Parthasarathy, and M. Goyder. Data Mining on the Cell Broadband Engine. In *Proceedings of the 22nd Annual International Conference on Supercomputing (ICS '08)*, pages 26--35, June 2008.

[12] D. Chiou, D. Chiouy, L. Rudolph, L. Rudolphy, S. Devadas, S. Devadasy, B. S. Ang, and B. S. Angz. Dynamic Cache Partitioning via Columnization. Laboratory for Computer Science, Massachusetts Institute of Technology, Computation Structures Group Memo 430, November 1999.

[13] K. Compton and S. Hauck. Reconfigurable Computing: A Survey of Systems and Software. *ACM Computing Surveys*, 34(2):171--210, June 2002.

[14] Intel Corporation. Intel Compiler Suite. http://www3.intel.com/cd/software/products/asmo-na/eng/index.htm.

[15] Intel Corporation. Intel Thread Profiler. Document number 300638-001.

[16] Intel Corporation. Intel VTune Performance Analyzer. http://www3.intel.com/cd/software/products/asmo-na/eng/index.htm.

[17] A. Arevalo et al. *Programming the Cell Broadband Engine: Examples and Best Practices*. International Business Machines Corporation (IBM), October 2008.

[18] A. Buttari et al. SCOP 3: A Rough Guide to Scientific Computing On the PlayStation 3. Technical Report UT-CS-07-595, Innovative Computing Laboratory, University of Tennessee, Knoxville, 2007.

[19] A. E. Eichenberger et al. Using Advanced Compiler Technology to Exploit the Performance of the Cell Broadband Engine Architecture. *IBM Systems Journal*, 45(1):59--84, January 2006.

[20] B. Flachs et al. A Streaming Processing Unit for a CELL Processor. *ISSCC Dig. Tech. Papers*, pages 134--135, 2005.

[21] D. Pham et al. The Design and Implementation of a First-Generation CELL Processor. *ISSCC Dig. Tech. Papers*, pages 184--185, 2005.

[22] E. Waingold et al. Baring It All to Software: Raw Machines. In *IEEE Computer*, volume 30, 1997.

[23] F. Petrini et al. Multicore Surprises: Lessons Learned from Optimizing Sweep3D on the Cell Broadband Engine. In *Proceedings of the 2007 International Parallel and Distributed Processing Symposium (IPDPS 2007)*, March 2007.

[24] J. A. Kahle et al. Introduction to the CELL Multiprocessor. *IBM Journal of Research and Development*, 49(4/5):589--604, July/September 2005.

[25] M. B. Taylor et al. The Raw Microprocessor: A Computational Fabric for Software Circuits and General Purpose Programs. In *IEEE Micro*, volume 22, 2002.

[26] O. Takahashi et al. Power-Conscious design of the Cell Processor's Synergistic Processor Element. *IEEE Micro*, 25(5):10--18, 2005.

[27] S. H. Dhong et al. A 4.8GHz Fully Pipelined Embedded SRAM in the Streaming Processor of a CELL Processor. *ISSCC Dig. Tech. Papers*, pages 487--487, 612, 2005.

[28] S. Kamil et al. Implicit and Explicit Optimizations for Stencil Computations. In *Proceedings of the 2006 Workshop on Memory System Performance and Correctness (MSPC-06)*, pages 51--60, October 2006.

[29] T. Asano et al. Low-Power Design Approach of 11FO4 256-Kbyte Embedded SRAM for the Synergistic Processor Element of a Cell Processor. *IEEE Micro*, 25(5):30--38, 2005.

[30] Y. Liu et al. Speech Recognition Systems on the Cell Broadband Engine Processor. *IBM Journal of Research and Development*, 51(5):583--591, September 2007.

[31] G. C. Evans and D. E. Coombe. Hemispherical and woodland canopy photography and the light climate. *Journal of Ecology*, 47:103--113, 1959.

[32] N. Greene and P. S. Heckbert. Creating raster Omnimax images from multiple perspective views using the elliptical weighted average filter. *IEEE Computer Graphics and Applications*, 6(6):21--27, June 1986.

[33] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, fourth edition, 2007.

[34] H. P. Hofstee. Power Efficient Processor Architecture and The Cell Processor. *International Symposium on High-Performance Computer Architecture*, 0:258--262, 2005.

[35] Intel Corporation. Intel64 and IA32 Architectures Software Developer's Manual. http://developer.intel.com/products/processor/manuals/index.htm.

[36] N. D. Jankovic and M. D. Naish. Developing a Modular Active Spherical Vision System. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, Barcelona, Spain*, April 2005.

[37] R. Keyes. Cubic Convolution Interpolation for Digital Image Processing. In *IEEE Transactions on Acoustics, Speech ,and Signal Processing*, volume ASSP-29, December 1981.

[38] M. Kistler, J. Gunnels, D. Brokenshire, and B. Benton. Programming the Linpack benchmark for the IBM PowerXCell 8i processor. *Scientific Programming*, 17:43--57, 2009.

[39] M. Knaup, S. Steckmann, O. Bockenbach, and M. Kachelrieß. Tomographic Image Reconstruction using the Cell Broadband Engine (CBE) General Purpose Hardware. In *Proceedings Electronic Imaging, Computational Imaging V, SPIE*, volume 6498, 64980, pages 1--10, January 2007.

[40] K. Krewell. CELL MOVES INTO THE LIMELIGHT. *Microprocessor Report*, 2005.

[41] J. B. Kuang. A Double-Precision Multiplier with Fine-Grained Clock-Gating Support for a First-Generation CELL Processor. *ISSCC Dig. Tech. Papers*, pages 378--379,605, 2005.

[42] D. P. Kuban, H. L. Martin, S. D. Zimmermann, and N. Busko. Omniview motionless camera surveillance system. US Patent Trademark Office, US5359363, Feb. 1993.

[43] B. Minor, G. Fossum, and V. To. Terrain Rendering Engine (TRE): Cell Broadband Engine Optimized Real-time Ray-caster. Online Documentation and Articles for the Cell Broadband Engine Architecture, http://www.ibm.com/developerworks/power/cell/.

[44] B. Ramakrishna Rau. Iterative modulo scheduling: an algorithm for software pipelining loops. In *MICRO 27: Proceedings of the 27th annual international symposium on Microarchitecture*, pages 63--74, New York, NY, USA, 1994. ACM.

[45] V. Sachdeva, M. Kistler, E. Speight, and T.-H. K. Tzang. Exploring the Viability of the Cell Broadband Engine for Bioinformatics Applications. In *Proceedings of the 2007 IEEE International Workshop on High Performance Computational Biology (HiCOMB '07)*, pages 1--8, March 2007.

[46] D. P. Scarpazza, O. Villa, and F. Petrini. Peak-Performance DFA-based String Matching on the Cell Processor. In *Proceedings of the 3rd IEEE/ACM Intl. Workshop on System Management Techniques, Processes, and Services (SMTPS 2007)*, pages 1--8, March 2007.

[47] D. Schneider and H.-G. Maas. Development and application of an extended geometrical model for high resolution panoramic cameras. *International archives of Photogrammetry, Remote Sensing and Spatial Information Sciences.*, XXXV, 2004.

[48] E. Schwalbe. Geometric Modeling and Calibration of FishEye Lens Camera Systems. In *Proceedings of the ISPRS Working Group, Panoramic Photogrammetry Workshop, Berlin, Germany*, volume 34-5/W8, February 2005.

[49] A. Stamatakis, F. Blagojevic, D. S. Nikolopoulos, and C. D. Antonopoulos. Exploring new Search Algorithms and Hardware for Phylogenetics: RAxML meets the IBM Cell. *Journal of VLSI Signal Processing*, 48(3):271--286, 2007.

[50] GProf GNU Binutils. http://www.gnu.org/software/binutils/.

[51] International Business Machines Corporation (IBM). PowerPC microprocessor family: Vector/SIMD Multimedia Extension Technology programming environments manual, version 2.07c, 2006.

[52] International Business Machines Corporation (IBM). Preventing Synergistic Processor Element indefinite stalls resulting from instruction depletion in the Cell Broadband Engine Processor for CMOS SOI 90 nm, version 1.0, 2007.

[53] International Business Machines Corporation (IBM). Cell Broadband Engine Architecture, version 1.02, 2007.

[54] International Business Machines Corporation (IBM). Cell Broadband Engine Programming Handbook. version 1.1, 2007.

[55] International Business Machines Corporation (IBM). IBM online course on Cell. http://publib.boulder.ibm.com/infocenter/ieduasst/stgv1r0/topic/com.ibm.iea.cbe/cbe_coverpage.html, 2007.

[56] International Business Machines Corporation (IBM). IBM Hands-On Tutorial on Cell programming. http://publib.boulder.ibm.com/infocenter/ieduasst/stgv1r0/topic/com.ibm.iea.cbe/cbe/1.0/Programming.html, 2007.

[57] International Business Machines Corporation (IBM). SIMD Math Library Specification for Cell Broadband Engine Architecture, version 1.1, 2007.

[58] International Business Machines Corporation (IBM). Software Development Kit for Multicore Acceleration, Programmer's Guide, version 3.0, 2007.

[59] International Business Machines Corporation (IBM). Software Development Kit for Multicore Acceleration, Programming Tutorial, version 3.0, 2007.

[60] International Business Machines Corporation (IBM). SPU Timer Library Programmers Guide and API Reference, version 3.0, 2007.

[61] International Business Machines Corporation (IBM). Synergistic Processor Unit Instruction Set Architecture, version 1.2, 2007.

[62] International Business Machines Corporation (IBM). C/C++ Language Extensions for Cell Broadband Engine Architecture, version 2.5, 2008.

[63] International Business Machines Corporation (IBM). PowerXCell 8i processor product brief. http://www-03.ibm.com/technology/cell/pdf/PowerXCell_PB_7May2008_pub.pdf, 2008.

[64] International Business Machines Corporation (IBM). Online documentation and articles for the Cell Broadband Engine Architecture, http://www.ibm.com/developerworks/power/cell/, 2009.

[65] International Business Machines Corporation (IBM). The Cell project at IBM Research. http://www.research.ibm.com/cell/home.html, 2009.

[66] O. Villa, D. P. Scarpazza, F. Petrini, and J. F. Peinador. Challenges in Mapping Graph Exploration Algorithms on Advanced Multi-core Processors. In *Proceedings of the 2007 International Parallel and Distributed Processing Symposium (IPDPS 2007)*, page 1, March 2007.

[67] S. Wagner. Calibration of grey values of hemispherical photographs for image analysis. *Agricultural and Forest Meteorology*, 90(1/2):103--117, 1998.

[68] Wikipedia. Article for the PlayStation 3. http://en.wikipedia.org/wiki/PlayStation_3.

[69] Wikipedia. Image Distortion. http://en.wikipedia.org/wiki/Image_distortion.

[70] S. Williams, J. Shalf, L. Oliker, S. Kamil, P. Husbands, and K. Yelick. The Potential of the Cell Processor for Scientific Computing. In *Proceedings of the 3rd Conference on Computing Frontiers*, pages 9--20, 2006.

[71] D. Wonnacott. Using Time Skewing to Eliminate Idle Time due to Memory Bandwidth and Network Limitations. In *Proceedings of the 2000 International Parallel and Distributed Processing Symposium (IPDPS 2000)*, pages 171--180, May 2000.

[72] Xilinx. Virtex-4 Handbook. http://www.xilinx.com, august 2004.

[73] T. Yamamoto and M. Doi. Design and Implementation of Panoramic Movie System by Using Commodity 3D Graphics Hardware. In *Computer Graphics International (CGI) Tokyo, Japan*, pages 14--19, July 2003.

[74] Y. Zhao and K. Kennedy. Dependence-based Code Generation for a CELL Processor. In *Proceedings of the 19th International Workshop on Languages and Compilers for Parallel Computing (LCPC)*, pages 64--79, November 2006.

[75] S. Zimmermann and D. Kuban. A video pan/tilt/magnify/rotate system with no moving parts. In *IEEE Digital Avionics Systems Conference*, pages 523--531, October 1992.