ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Μελέτη αρχιτεκτονικής και εφαρμογών πολυπύρηνου υβριδικού επεξεργαστή

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

Γεωργίας Ν. Κουβέλη

Επιβλέπων: Νεκτάριος Κοζύρης
Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Δεκέμβριος 2007

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Μελέτη αρχιτεκτονικής και εφαρμογών πολυπύρηνου υβριδικού επεξεργαστή

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

Γεωργίας Ν. Κουβέλη

Επιβλέπων: Νεκτάριος Κοζύρης
Επίκουρος Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις 5 Δεκεμβρίου 2007

Ν.Κοζύρης	Ν. Παπασπύρου	Κ. Σαγώνας
Επ. Καθηγητής Ε.Μ.Π.	Επ. Καθηγητής Ε.Μ.Π.	Αν. Καθηγητής Ε.Μ.Π.

Αθήνα, Δεκέμβριος 2007

Γεωργία Ν. Κουβέλη

© Γεωργία Ν. Κουβέλη, 2007
Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν την συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

Περίληψη

Η διπλωματική εργασία έχει ως στόχο τη μελέτη και αξιολόγηση καινοτόμων υπολογιστικών συστημάτων με πολλαπλούς πυρήνες (multicore systems) όσο αφορά στις δυνατότητές τους στη παράλληλη επεξεργασία δεδομένων. Συγκεκριμένα, μελετήθηκε ο επεξεργαστής Cell, ο οποίος είναι ένας υβριδικός πολυπύρηνος επεξεργαστής που αποτελείται από ένα κεντρικό επεξεργαστή και οχτώ δευτερεύοντες συνεπεξεργαστές. Τα χαρακτηριστικά του επεξεργαστή Cell τον διαφοροποιούν από τους τυπικούς σύγχρονους επεξεργαστές και προσφέρουν αυξημένες δυνατότητες για ανάπτυξη παράλληλων εφαρμογών υψηλών επιδόσεων. Βασικός στόχος της εργασίας είναι η διερεύνηση των κατάλληλων προγραμματιστικών μοντέλων και τεχνικών που θα επιτρέψουν την εκμετάλλευση των παραπάνω δυνατοτήτων.

Στα πλαίσια της εργασίας αρχικά μελετήθηκε η αρχιτεκτονική του επεξεργαστή Cell και στη συνέχεια αναπτύχθηκαν μετροπρογράμματα για την αποτίμησή της επίδοσής του. Βασικό στοιχείο της επίδοσης των παράλληλων εφαρμογών είναι η επικοινωνία μεταξύ των επεξεργαστικών μονάδων, οπότε και ιδιαίτερη βαρύτητα δόθηκε στη μελέτη της επίδοσης διαφορετικών σχημάτων επικοινωνίας μεταξύ των επεξεργαστικών μονάδων και της κεντρικής μνήμης.

Λέξεις Κλειδιά: Πολυπύρηνος Επεξεργαστής, Παράλληλη Επεξεργασία, PowerPC Processor Element (PPE), Synergistic Processor Element (SPE), Άμεση Πρόσβαση στη Μνήμη (DMA), Element Interconnect Bus (EIB), Σχήμα Επικοινωνίας

Abstract

The purpose of this thesis is the study and evaluation of innovative multicore computing systems regarding their potential in parallel data processing. Specifically, the Cell processor was studied, which is a hybrid multicore processor consisting of one central processing unit and eight secondary coprocessors. The attributes of the Cell processor differentiate it from other typical contemporary processors and offer improved capabilities in developing high-performance parallel applications. A basic purpose of this thesis is the investigation of the appropriate programming models and techniques that will allow the exploitation of the foregoing capabilities.

The first part of this thesis was the study of the architecture of the Cell processor. The second part was the development benchmarks for the evaluation of its perfomance. A basic factor of the performance of parallel applications is the communication between the processing units, therefore the study of the performance of different communication patterns between the processing elements and main memory was of primary importance.

Keywords: Multicore Processor, Parallel Processing, Cell, PowerPC Processor Element (PPE), Synergistic Processor Element (SPE), Direct Memory Access (DMA), Element Interconnect Bus (EIB), Communication Pattern

Περιεχόμενα

Κεφάλαιο 1

Πολυπύρηνοι επεξεργαστές

1.1 Εισαγωγή

Η βιομηχανία υπολογιστών βρίσκεται συνεχώς αντιμέτωπη με τις διαρκώς αυξανόμενες απαιτήσεις σε υπολογιστική ισχύ. Η αύξηση της υπολογιστικής ισχύος των επεξεργαστών μέχρι πρόσφατα βασιζόταν κυρίως στην διαρκή αύξηση της συχνότητας λειτουργίας των επεξεργαστών και την εισαγωγή περισσότερων σταδίων στις αρχιτεκτονικές αγωγού που αυτοί χρησιμοποιούν, όμως στην προσπάθεια αυτή, αντιμετωπίζονται πλέον σημαντικά εμπόδια, τα οποία αφορούν κυρίως την κατανάλωση ισχύος και τα σχετιζόμενα θερμικά προβλήματα. Στην προσπάθεια να αντιμετωπιστούν τα παραπάνω προβλήματα, τα τελευταία χρόνια παρατηρείται μια σαφής στροφή της αγοράς προς τους πολυπύρηνους επεξεργαστές.

Ένας πολυπύρηνος επεξεργαστής είναι ένας επεξεργαστής που αποτελείται από περισσότερες από μία μονάδες επεξεργασίας. Οι πολυπύρηνοι επεξεργαστές συνδυάζουν δύο ή περισσότερους ανεξάρτητους πυρήνες σε ένα ενιαίο περίβλημα που αποτελείται από ένα μοναδικό ολοκληρωμένο κύκλωμα, ή από περισσότερες ψηφίδες που έχουν πακεταριστεί μαζί. Οι πυρήνες μπορούν να μοιράζονται μια μοναδική συναφή κρυφή μνήμη ή να έχουν χωριστές κρυφές μνήμες. Επιπλέον μοιράζονται την ίδια διεπαφή/διασύνδεση με το υπόλοιπο σύστημα. Σε κάθε έναν από τους πυρήνες μπορούν να υπάρχουν βελτιστοποιήσεις της αρχιτεκτονικής, όπως υπερβαθμωτή εκτέλεση, αρχιτεκτονικές αγωγού και υποστήριξη πολλαπλών νημάτων εκτέλεσης. Ένα σύστημα με N πυρήνες για να είναι αποδοτικό πρέπει να διαθέτει ταυτόχρονα τουλάχιστον N νήματα προς εκτέλεση. Οι πιο εμπορικοί πολυπύρηνοι επεξεργαστές που χρησιμοποιούνται σήμερα είναι εκείνοι που χρησιμοποιούνται στους προσωπικούς υπολογιστές (κυρίως από τις εταιρίες Intel και AMD) και στις μηχανές παιχνιδιών (για παράδειγμα ο επεξεργαστής Cell που χρησιμοποιείται στο Sony Playstation 3). Όμως η τεχνολογία χρησιμοποιείται και σε άλλους τεχνολογικούς τομείς, όπως τα ενσωματωμένα συστήματα, η ψηφιακή επεξεργασία σήματος και οι επεξεργαστές γραφικών (GPUs).

1.2 Επιπτώσεις για το λογισμικό

Το λογισμικό μπορεί να ωφεληθεί από τις πολυπύρηνες αρχιτεκτονικές όταν υπάρχει κώδικας που μπορεί να εκτελεστεί παράλληλα. Στα περισσότερα σύγχρονα λειτουργικά συστήματα κάθε εφαρμογή εκτελείται ως ξεχωριστή διεργασία, με αποτέλεσμα η ταυτόχρονη εκτέλεση πολλών εφαρμογών να μπορεί να επιταχυνθεί σημαντικά με χρήση πολυπύρηνων επεξεργαστών. Για να ωφεληθεί μία συγκεκριμένη εφαρμογή από τη χρήση ενός πολυπύρηνου επεξεργαστή πρέπει η ανάπτυξή της να έχει γίνει ειδικά ώστε να χρησιμοποιεί περισσότερα από ένα νήματα.

Όμως οι περισσότερες εφαρμογές που χρησιμοποιούνται σήμερα δεν έχουν γραφεί ώστε να χρησιμοποιούν αποδοτικά πολλαπλά νήματα, καθώς κάτι τέτοιο είναι απαιτητικό σε χρόνο εργασίας και σε προγραμματιστική ικανότητα. Συχνά, ακόμα και οι εφαρμογές που χρησιμοποιούν πολλά νήματα, δεν εφαρμόζουν ισορροπημένο καταμερισμό της εργασίας που πρέπει τα νήματα να εκτελέσουν, με αποτέλεσμα κάποια νήματα να έχουν πολύ πιο απαιτητική εργασία να εκτελέσουν από ό,τι κάποια άλλα. Σε αυτές τις περιπτώσεις, η εφαρμογή δεν έχει μεγάλα περιθώρια οφέλους από τη χρήση πολλαπλών πυρήνων. Η συγγραφή κώδικα που πραγματικά αξιοποιεί τη χρήση πολλαπλών νημάτων συχνά απαιτεί τον πολύπλοκο συγχρονισμό μεταξύ των νημάτων, με αποτέλεσμα η αποσφαλμάτωση τέτοιων προγραμμάτων να είναι πολύ δυσκολότερη από ό,τι στην περίπτωση απλών εφαρμογών που δε χρησιμοποιούν νήματα. Όμως η αποδοτική ανάπτυξη τέτοιου είδους εφαρμογών φαίνεται πως θα επηρεάσει σε μεγάλο βαθμό την εξέλιξη της επίδοσης των σύγχρονων υπολογιστικών συστημάτων.

Τέλος, με τους πολυπύρηνους επεξεργαστές μπορούν να χρησιμοποιηθούν υπάρχουσες τεχνικές παράλληλου προγραμματισμού. Συγκεκριμένα προγραμματιστικά μοντέλα, όπως το MPI και το OpenMP μπορούν να χρησιμοποιηθούν σε τέτοιου είδους συστήματα.

1.3 Ο επεξεργαστής Cell

Ο Cell είναι ένας ετερογενής πολυνηματικός επεξεργαστής, ο οποίος αναπτύχθηκε από τις IBM, Sony και Toshiba και αποτελείται από εννέα επεξεργαστικά στοιχεία, ένα κεντρικό στοιχείο επεξεργασίας (PPE) και οχτώ συνεπεξεργαστές (SPE), σε ένα μοναδικό ολοκληρωμένο κύκλωμα. Η ανάπτυξη του Cell έγινε με σκοπό να ξεπεραστούν τρεις σημαντικοί παράγοντες που περιορίζουν σημαντικά την επίδοση των σύγχρονων επεξεργαστών: η χρήση της μνήμης, η κατανάλωση ισχύος και η συχνότητα του επεξεργαστή.

Καταρχάς, ένα από τα σημαντικότερα προβλήματα είναι αυτό που σχετίζεται με τη μνήμη. Ενώ η συχνότητα λειτουργίας των μονάδων επεξεργασίας αυξάνεται συνεχώς, δεν παρατηρείται ανάλογη μείωση των καθυστερήσεων στην δυναμική μνήμη τυχαίας προσπέλασης (DRAM), αντίθετα οι καθυστερήσεις αυτές αυξάνονται σε κάθε νέα γενιά μνημών. Σε κάθε αστοχία κρυφής μνήμης, οπότε η πρόσβαση στην κεντρική μνήμη είναι απαραίτητη, μπορεί να καθυστερήσει η εκτέλεση εκατοντάδων εντολών. Στην προσπάθεια να αντιμετωπιστούν οι μεγάλες αυτές ποινές αστοχίας, οι επεξεργαστές χρησιμοποιούν συχνά κατ' εικασία (speculative) εκτέλεση, όμως η μέθοδος αυτή είναι ανεπαρκής για την επιτυχή απόκρυψη των καθυστερήσεων πρόσβασης στη μνήμη.

Το δεύτερο πρόβλημα είναι η συνεχώς αυξανόμενη κατανάλωση ισχύος, η οποία δημιουργεί την ανάγκη να χρησιμοποιούνται εκλεπτυσμένες μέθοδοι ψύξης. Το μέγιστο μέγεθος του συστήματος που μπορεί να είναι δεκτό, καθώς και παράγοντες όπως η μέγιστη θερμότητα που μπορεί να φεύγει από το σύστημα προς τα έξω, περιορίζουν τη μέγιστη συνολική κατανάλωση ισχύος, άρα και τη μέγιστη συχνότητα που μπορεί να χρησιμοποιηθεί. Η ταυτόχρονη αύξηση της επίδοσης και της αποδοτικότητας στην κατανάλωση ισχύος αποτελεί συνεπώς μια μεγάλη πρόκληση.

Τέλος, η βελτίωση της υπολογιστικής ισχύος δεν μπορεί πλέον να βασίζεται στην αύξηση της συχνότητας σε συνδυασμό με την αύξηση του βάθους των αρχιτεκτονικών αγωγού. Το πρόβλημα είναι ότι καθώς τα στάδια της αρχιτεκτονικής αγωγού γίνονται περισσότερα, αυξάνεται η καθυστέρηση μιας μεμονωμένης εντολής. Για να υπάρχει όφελος από τη χρήση υψηλότερων συχνοτήτων, επομένως από την έκδοση περισσότερων εντολών σε δεδομένο χρονικό διάστημα, πρέπει η αύξηση της επίδοσης που προκύπτει κατά αυτόν τον τρόπο να υπερβαίνει τον χρόνο που χάνεται λόγω των υψηλών ποινών που σχετίζονται με την αυξημένη καθυστέρηση στην εκτέλεση των εντολών. Όταν η αύξηση της συχνότητας δεν μπορεί να είναι αρκετή ώστε να αντισταθμιστούν αυτές οι ποινές, η χρήση περισσότερων σταδίων στην αρχιτεκτονική αγωγού δύναται να επιφέρει μείωση αντί για αύξηση της συχνότητας.

Στον Cell η αποδοτικότερη κατανάλωση ισχύος επιτυγχάνεται μέσω της διαφοροποίησης μεταξύ του επεξεργαστή που είναι βελτιστοποιημένος για την εκτέλεση του λειτουργικού συστήματος (PPE) και των επεξεργαστών που είναι βελτιστοποιημένοι για την εκτέλεση απαιτητικών σε υπολογισμούς προγραμμάτων (SPEs). Επίσης επιτυγχάνεται αποτελεσματική απόκρυψη των υψηλών καθυστερήσεων πρόσβασης στη μνήμη, μέσω της ιδιαίτερης ιεραρχίας τριών επιπέδων (μεγάλο ενοποιημένο αρχείο καταχωρητών στους συνεπεξεργαστές - τοπικές μνήμες στους συνεπεξεργαστές - κεντρική μνήμη) και μέσω των ασύγχρονων μεταφορών άμεσης πρόσβασης στη μνήμη, μεταξύ της κεντρικής μνήμης και των τοπικών μνημών των συνεπεξεργαστών. Τέλος, με την εξειδίκευση των επεξεργαστικών στοιχείων, τόσο το κεντρικό στοιχείο επεξεργασίας (PPE) όσο και οι συνεπεξεργαστές (SPE) σχεδιάστηκαν για υψηλές συχνότητες χωρίς υπερβολικές επιβαρύνσεις. Στο

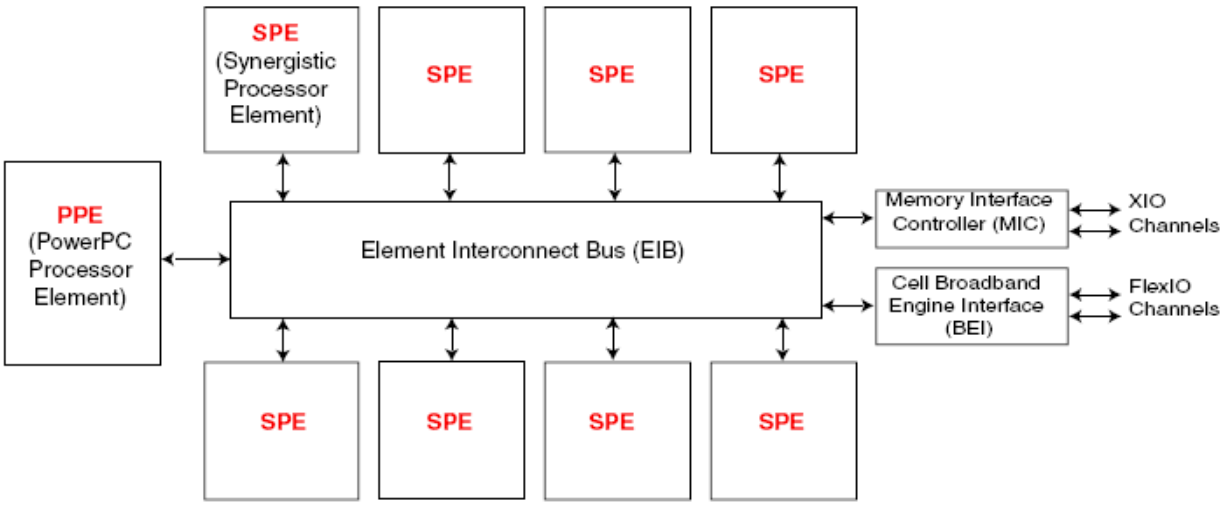
κεντρικό στοιχείο επεξεργασίας η υψηλή επίδοση επιτυγχάνεται με την βελτιστοποίηση για την εκτέλεση δύο νημάτων αντί για ένα. Στους συνεπεξεργαστές, η χρήση ενός μεγάλου ενοποιημένου αρχείου καταχωρητών υποβοηθά την παράλληλη εκτέλεση πολλών εντολών, και η χρήση ασύγχρονων μεταφορών άμεσης πρόσβασης στη μνήμη (DMA) εξαλείφει την ανάγκη για κατ' εικασία εκτέλεση και τις σχετιζόμενες επιβαρύνσεις.

Κεφάλαιο 2

Αρχιτεκτονική του επεξεργαστή Cell

2.1 Εισαγωγή

Ο Cell είναι ένας ετερογενής πολυνηματικός επεξεργαστής, ο οποίος αποτελείται από εννέα επεξεργαστικά στοιχεία σε ένα μοναδικό ολοκληρωμένο κύκλωμα. Τα επεξεργαστικά στοιχεία συνδέονται μεταξύ τους και με τις εξωτερικές συσκευές μέσω μιας συναφούς (coherent) μονάδας διαύλου υψηλού εύρους ζώνης. Στην παρακάτω εικόνα φαίνεται μια σχηματική αναπαράσταση των κυρίων συστατικών μερών του επεξεργαστή Cell:



Τα κύρια μέρη του επεξεργαστή Cell είναι τα ακόλουθα:

- Κεντρικό στοιχείο επεξεργασίας (PPE – PowerPC Processor Element)

Το PPE είναι το κύριο επεξεργαστικό στοιχείο. Περιέχει έναν πυρήνα μειωμένου συνόλου εντολών (RISC) αρχιτεκτονικής 64-bit PowerPC με ένα συμβατικό σύστημα εικονικής μνήμης. Εκτελεί το λειτουργικό σύστημα, διαχειρίζεται τους πόρους του συστήματος και προορίζεται κυρίως για διαδικασίες ελέγχου, στις οποίες περιλαμβάνονται η δημιουργία και η διαχείριση των νημάτων που εκτελούνται στους συνεπεξεργαστές. Το κεντρικό στοιχείο επεξεργασίας είναι εξειδικευμένο σε λειτουργίες ελέγχου του συστήματος και έχει πολύ καλή απόδοση σε αυτές. Υποστηρίζει το σύνολο

εντολών της αρχιτεκτονικής PowerPC και το σύνολο εντολών Vector/SIMD Multimedia Extension, επομένως μπορεί να εκτελεί λογισμικό συμβατό με την αρχιτεκτονική PowerPC.

- Ειδικευμένοι συνεπεξεργαστές (SPEs – Synergistic Processor Elements)

Οι οχτώ συνεπεξεργαστές είναι επεξεργαστές SIMD (Single Instruction Multiple Data) βελτιστοποιημένοι για εφαρμογές απαιτητικές ως προς τους υπολογισμούς και ως προς την ποσότητα των δεδομένων, οι οποίες τους ανατίθενται από την κεντρική επεξεργαστική μονάδα. Κάθε ένας από τους συνεπεξεργαστές περιέχει έναν πυρήνα μειωμένου συνόλου εντολών (RISC), μια τοπική μνήμη (local store) μεγέθους 256KB για την αποθήκευση δεδομένων και εντολών, η οποία ελέγχεται από το λογισμικό, καθώς και ένα μεγάλο ενοποιημένο αρχείο καταχωρητών, με 128 καταχωρητές των 128 bit. Οι συνεπεξεργαστές υποστηρίζουν ένα ειδικό σύνολο εντολών διανυσμάτων (SIMD) και βασίζονται σε ασύγχρονες μεταφορές άμεσης πρόσβασης στη μνήμη (Direct Memory Access – DMA) για τη μεταφορά εντολών και δεδομένων μεταξύ του κύριου χώρου διευθύνσεων (δηλαδή του συνόλου όλων των ενεργών διευθύνσεων, στις οποίες περιλαμβάνεται και η κεντρική μνήμη) και της τοπικής τους μνήμης. Στις μεταφορές αυτές οι συνεπεξεργαστές χρησιμοποιούν ενεργές διευθύνσεις (effective addresses) της αρχιτεκτονικής PowerPC. Η μετάφραση των διευθύνσεων αυτών γίνεται βάσει των πινάκων τμημάτων και σελίδων της αρχιτεκτονικής PowerPC, όπως και στην κεντρική επεξεργαστική μονάδα. Οι συνεπεξεργαστές δεν είναι εξειδικευμένοι για την εκτέλεση του λειτουργικού συστήματος.

- Δίαυλος διασύνδεσης στοιχείων (EIB – Element Interconnect Bus)

Το κεντρικό στοιχείο επεξεργασίας (PPE) και οι συνεπεξεργαστές (SPEs) επικοινωνούν συναφώς (coherently) μεταξύ τους καθώς επίσης και με την κεντρική μνήμη και τις συσκευές εισόδου-εξόδου, μέσω του διαύλου διασύνδεσης στοιχείων. Ο δίαυλος αποτελείται από τέσσερις δακτύλιους (rings) για τη μεταφορά των δεδομένων, και μια δεντρική δομή (tree structure) για τη μεταφορά των εντολών. Το εύρος ζώνης στο εσωτερικό του είναι 96 byte/cycle και μπορεί να υποστηρίζει την ταυτόχρονη εκτέλεση 128 αιτήσεων μεταφοράς από και προς τις τοπικές μνήμες των συνεπεξεργαστών.

- Ελεγκτής μνήμης (MIC – Memory Interface Controller)

Ο ελεγκτής μνήμης παρέχει τη διεπαφή μεταξύ του διαύλου διασύνδεσης των στοιχείων (EIB) και της κεντρικής μνήμης. Υποστηρίζει δύο κανάλια εισόδου-εξόδου από/προς τη μνήμη, τύπου Rambus Extreme Data Rate (XDR). Τα υποστηριζόμενα μεγέθη μεταφορών σε κάθε κανάλι είναι 1-8, 16, 32, 64, ή 128 bytes.

- Ελεγκτής συσκευών εισόδου-εξόδου (BEI – Broadband Engine Interface)

Ο ελεγκτής συσκευών εισόδου-εξόδου ελέγχει τη μεταφορά δεδομένων μεταξύ του διαύλου διασύνδεσης των στοιχείων (EIB) και των συσκευών εισόδου-εξόδου (I/O). Υποστηρίζει δύο κανάλια

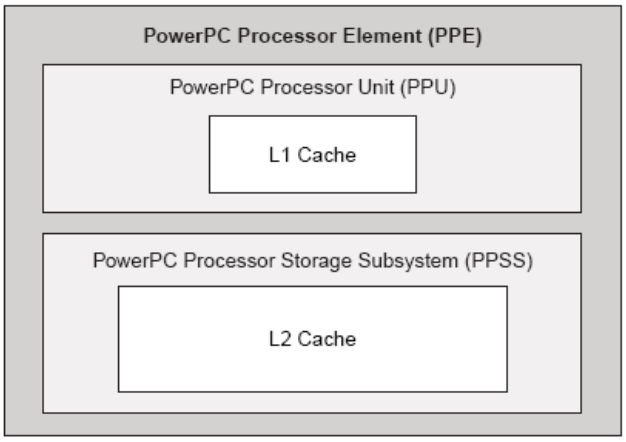
τύπου Rambus FlexIO. Το ένα από τα δύο κανάλια περιορίζεται σε μη συναφείς μεταφορές, ενώ το άλλο μπορεί να ρυθμιστεί ώστε να υποστηρίζει συναφείς μεταφορές, επεκτείνοντας το δίαυλο διασύνδεσης των στοιχείων (EIB) ώστε να συνδέεται λογικά με μία εξωτερική συμβατή συσκευή, για παράδειγμα έναν δεύτερο επεξεργαστή Cell.

Στις παραγράφους που ακολουθούν δίνεται μια αναλυτικότερη παρουσίαση των διαφόρων στοιχείων που αποτελούν τον επεξεργαστή Cell.

2.2 Κεντρικό στοιχείο επεξεργασίας (PPE)

Το κεντρικό στοιχείο επεξεργασίας είναι ένας επεξεργαστής 64-bit γενικού σκοπού που υποστηρίζει δύο νήματα εκτέλεσης. Η συχνότητα λειτουργίας του είναι 3.2 GHz. Η αρχιτεκτονική του είναι αρχιτεκτονική μειωμένου συνόλου εντολών, συμβατή με την αρχιτεκτονική PowerPC (έκδοση 2.02), με το σύνολο εντολών Vector/SIMD Multimedia Extension. Ο κύριος σκοπός του κεντρικού στοιχείου επεξεργασίας είναι ο συνολικός έλεγχος του συστήματος, επομένως είναι εκείνο που αναλαμβάνει την εκτέλεση του λειτουργικού συστήματος.

Το κεντρικό στοιχείο επεξεργασίας αποτελείται από δύο μέρη: την κεντρική μονάδα επεξεργασίας (Power Processor Unit – PPU) και το υποσύστημα διαχείρισης μνήμης (Power Processor Storage Subsystem – PPSS), όπως φαίνεται και στο σχήμα που ακολουθεί:



Η κεντρική μονάδα επεξεργασίας (PPU) αναλαμβάνει τον έλεγχο και την εκτέλεση των εντολών. Περιλαμβάνει όλους τους καταχωρητές που ορίζονται από την αρχιτεκτονική PowerPC των 64 bit, καθώς και 3 καταχωρητές διανυσμάτων των 128 bit, μια κρυφή μνήμη πρώτου επιπέδου (L1) μεγέθους 32KB για την αποθήκευση εντολών, μια κρυφή μνήμη πρώτου επιπέδου (L1) μεγέθους 32KB για την

αποθήκευση δεδομένων, μια μονάδα ελέγχου των εντολών, μια μονάδα φόρτωσης και αποθήκευσης, μια μονάδα πράξεων σταθερής υποδιαστολής, μια μονάδα πράξεων κινητής υποδιαστολής, μια μονάδα πράξεων σε διανύσματα, μια μονάδα διακλαδώσεων και μια μονάδα διαχείρισης της εικονικής μνήμης.

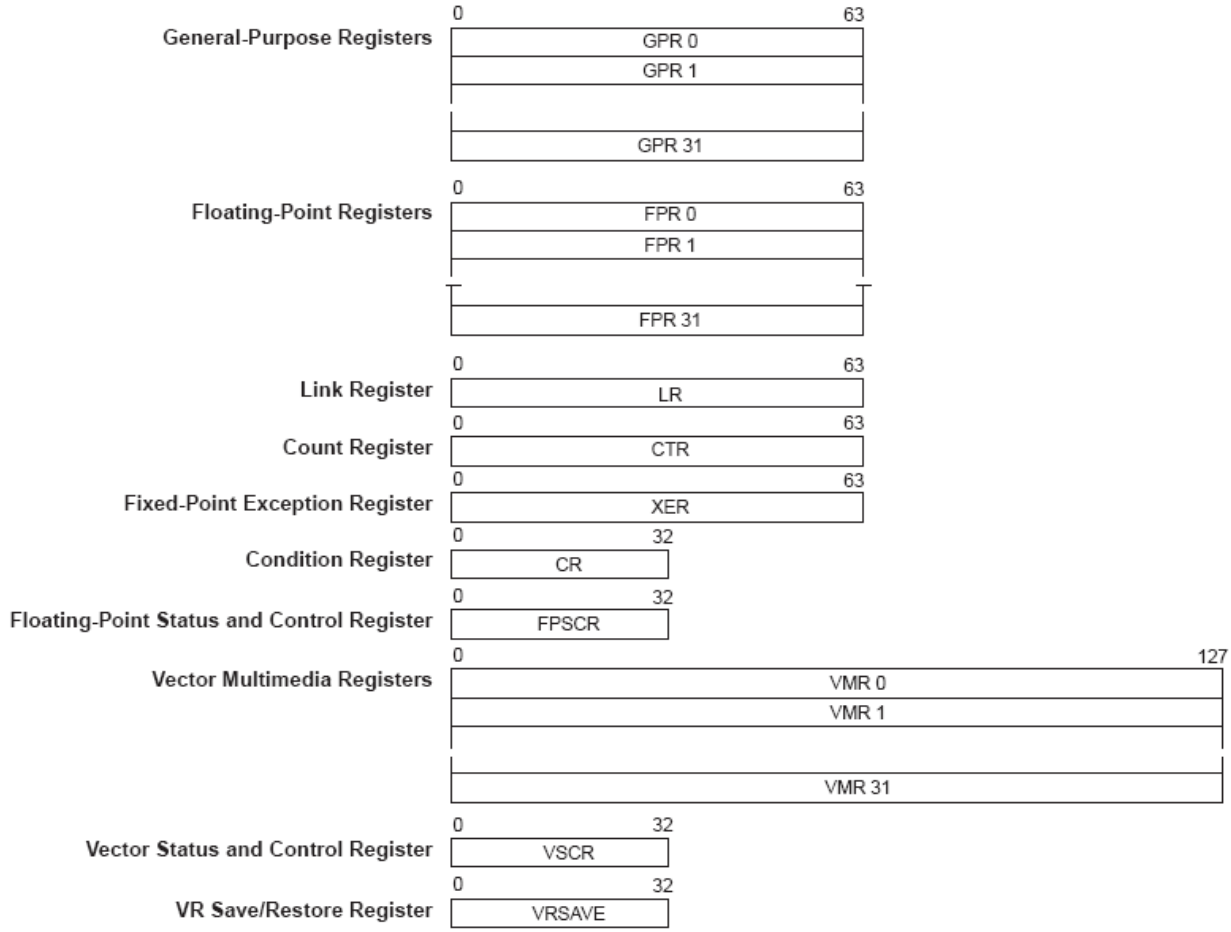
Η κεντρική μονάδα επεξεργασίας υποστηρίζει την ταυτόχρονη εκτέλεση δύο νημάτων και εμφανίζεται στο λογισμικό ως δύο ανεξάρτητοι επεξεργαστές. Όλα τα αρχιτεκτονικά στοιχεία (καταχωρητές ειδικού και γενικού σκοπού που ορίζονται από την αρχιτεκτονική, εκτός από εκείνους που σχετίζονται με τον έλεγχο των πόρων του συστήματος, όπως τη μνήμη και τον έλεγχο των νημάτων) υπάρχουν εις διπλούν. Οι υπόλοιποι πόροι, όπως οι κρυφές μνήμες, οι διάφορες ουρές, και η δίοδος δεδομένων, μοιράζονται μεταξύ των δύο νημάτων.

Το υποσύστημα διαχείρισης μνήμης (PPSS) χειρίζεται τις αιτήσεις μνήμης από το κεντρικό στοιχείο επεξεργασίας (PPE) και τις εξωτερικές αιτήσεις προς το PPE από άλλους επεξεργαστές ή από τις συσκευές εισόδου/εξόδου. Περιλαμβάνει μια ενοποιημένη κρυφή μνήμη δευτέρου επιπέδου, μεγέθους 512KB για την αποθήκευση εντολών και δεδομένων, καθώς και διάφορες ουρές αναμονής και μια μονάδα διεπαφής προς τον δίαυλο διασύνδεσης των στοιχείων. Η κύρια μνήμη παρουσιάζεται ως ένας συνεχής πίνακας από bytes με δείκτες από 0 ως 2^{64}-1. Ο δείκτης κάθε byte αποτελεί τη διεύθυνσή του. Κάθε φορά εκτελείται μία πρόσβαση στη μνήμη και η εκτέλεση των εντολών εγγραφής/ανάγνωσης στην/από την μνήμη εμφανίζεται να διατηρεί τη σειρά των εντολών του προγράμματος.

Η κρυφή μνήμη δευτέρου επιπέδου και οι κρυφές μνήμες για την μετάφραση των διευθύνσεων χρησιμοποιούν πίνακες διαχείρισης-αντικατάστασης που επιτρέπουν στο λογισμικό να διαχειρίζεται τη χρήση τους, ιδιότητα πολύ χρήσιμη για τα συστήματα πραγματικού χρόνου.

2.2.1 Καταχωρητές

Στην κεντρική μονάδα επεξεργασίας, όπως συμβαίνει σε κάθε αρχιτεκτονική μειωμένου συνόλου εντολών (RISC), όλες οι εντολές που εκτελούν υπολογισμούς επιδρούν στα περιεχόμενα καταχωρητών και όχι σε θέσεις μνήμης. Συνεπώς, για να χρησιμοποιηθούν σε κάποιον υπολογισμό τα περιεχόμενα μιας θέσης μνήμης και να αποθηκευθεί το αποτέλεσμα σε κάποια άλλη ή στην ίδια θέση μνήμης, πρέπει τα περιεχόμενα της θέσης μνήμης να μεταφερθούν σε κάποιον καταχωρητή μέσω μιας εντολής ανάγνωσης από τη μνήμη, στη συνέχεια να εκτελεσθεί ο υπολογισμός, και τέλος με μια εντολή εγγραφής στη μνήμη να αποθηκευθεί το αποτέλεσμα στη σωστή θέση. Το σύνολο όλων των καταχωρητών που περιλαμβάνονται στην κεντρική μονάδα επεξεργασίας φαίνεται στο σχήμα που ακολουθεί:



Οι καταχωρητές αυτοί περιλαμβάνουν:

- Καταχωρητές γενικής χρήσης (General-Purpose Registers – GPRs): 32 καταχωρητές των 64 bit που χρησιμοποιούνται για πράξεις σταθερής υποδιαστολής
- Καταχωρητές κινητής υποδιαστολής (Floating-Point Registers – FPRs): 32 καταχωρητές των 64 bit που χρησιμοποιούνται για πράξεις κινητής υποδιαστολής
- Καταχωρητής σύνδεσης (Link Register – LR): καταχωρητής των 64 bit που περιέχει την ενεργό διεύθυνση προορισμού μιας εντολής διακλάδωσης
- Μετρητής (Count Register – CR): καταχωρητής των 64 bit που μπορεί είτε να λειτουργήσει ως μετρητής βρόχου είτε να περιέχει την ενεργό διεύθυνση προορισμού μιας εντολής διακλάδωσης
- Καταχωρητής εξαίρεσης κινητής υποδιαστολής (Fixed-Point Exception Register – XER): καταχωρητής των 64 bit που περιέχει τα bit κρατουμένου και υπερχείλισης για τις πράξεις

σταθερής υποδιαστολής, καθώς και τον αριθμό των byte για τις εντολές move-assist

- Καταχωρητής συνθήκης (Condition Register – CR): καταχωρητής 32 bit που εκφράζει τα αποτελέσματα συγκεκριμένων λειτουργιών και χρησιμοποιείται για έλεγχο ή διακλάδωση
- Καταχωρητής κατάστασης και ελέγχου κινητής υποδιαστολής (Floating-Point Status and Control Register – FPSCR): καταχωρητής 32 bit που ενημερώνεται μετά από κάθε πράξη κινητής υποδιαστολής και περιέχει πληροφορίες για το αποτέλεσμα της πράξης και για πιθανές εξαιρέσεις
- Διανυσματικοί καταχωρητές πολυμέσων (Vector Multimedia Registers – VMRs): 32 καταχωρητές των 32 bit, οι οποίοι χρησιμοποιούνται στις πράξεις διανυσμάτων
- Καταχωρητής κατάστασης και ελέγχου διανυσμάτων (Vector Status and Control Register – VSCR): καταχωρητής 32 bit που ενημερώνεται μετά από κάθε πράξη διανυσμάτων και περιέχει πληροφορίες για το αποτέλεσμα της πράξης
- Καταχωρητής αποθήκευσης διανυσμάτων (Vector Save Register – VRSAVE): καταχωρητής 32 bit που βοηθά στην αποθήκευση της κατάστασης του επεξεργαστή (των περιεχομένων των καταχωρητών) κατά την ανταλλαγή διεργασιών (context switch)

2.2.2 Σύνολα εντολών

2.2.2.1 Σύνολο εντολών αρχιτεκτονικής PowerPC

Οι εντολές σε αυτό το σύνολο εντολών έχουν μέγεθος 4 byte και ευθυγράμμιση λέξης. Επομένως κάθε φορά που παρουσιάζεται στον επεξεργαστή η διεύθυνση μιας εντολής, αγνοούνται τα δύο λιγότερο σημαντικά bit, ενώ κάθε φορά που η διεύθυνση μιας εντολής κατασκευάζεται από τον επεξεργαστή, τα δύο λιγότερο σημαντικά bit ισούνται με το μηδέν. Στον επεξεργαστή Cell χρησιμοποιείται η Big Endian διαρρύθμιση των bytes, επομένως η διεύθυνση μιας εντολής (και γενικότερα οποιουδήποτε δεδομένου) δείχνει στο περισσότερο σημαντικό byte της λέξης. Υποστηρίζονται μεταφορές byte, μισής λέξης, λέξης και διπλής λέξης μεταξύ των καταχωρητών γενικού σκοπού (GPRs) και της μνήμης. Επιπλέον υποστηρίζονται μεταφορές λέξης και διπλής λέξης μεταξύ των καταχωρητών κινητής υποδιαστολής (FPRs) και της μνήμης. Η αναπαράσταση των προσημασμένων αριθμών γίνεται με τη μορφή συμπληρώματος ως προς δύο. Η αναπαράσταση των δεδομένων κινητής υποδιαστολής είναι στην μορφή IEEE 754 διπλής ακρίβειας, ενώ οι αριθμοί κινητής υποδιαστολής απλής ακρίβειας διατηρούνται εσωτερικά και αυτοί στη μορφή διπλής ακρίβειας.

Οι εντολές του συνόλου εντολών αυτού μπορούν να έχουν έως τρεις τελεστές. Συνήθως οι εντολές

υπολογισμού περιέχουν δύο τελεστές προέλευσης και έναν τελεστή προορισμού.

Όλες οι εντολές φόρτωσης και αποθήκευσης καθορίζουν έναν καταχωρητή βάσης. Η ενεργός διεύθυνση (effective address) των δεδομένων που θα λάβουν μέρος στη μεταφορά μπορεί να υπολογιστεί με τρεις τρόπους: άθροισμα καταχωρητή βάσης και εκτόπισης (register + displacement), άθροισμα καταχωρητή βάσης και ενός καταχωρητή δείκτη (register + register), περιεχόμενο καταχωρητή βάσης χωρίς τροποποίηση (register).

Όλες οι εντολές, πλην των εντολών διακλάδωσης, δημιουργούν τη διεύθυνση της επόμενης εντολής αυξάνοντας έναν μετρητή προγράμματος. Οι εντολές διακλάδωσης είναι οι μόνες εντολές που καθορίζουν άμεσα την ενεργό διεύθυνση της επόμενης εντολής προς εκτέλεση, του προορισμού της διακλάδωσης δηλαδή, με έναν από τους ακόλουθους τρόπους:

- Μη επιτυχής διακλάδωση (branch not taken): η διεύθυνση της επόμενης εντολής υπολογίζεται αυξάνοντας τη διεύθυνση της τρέχουσας εντολής κατά 4
- Άμεση/απόλυτη διευθυνσιοδότηση (absolute): η διεύθυνση της επόμενης εντολής δίνεται άμεσα σε κατάλληλο πεδίο της εντολής
- Σχετική διευθυνσιοδότηση (relative): η διεύθυνση της επόμενης εντολής δίνεται από το άθροισμα του σχετικού πεδίου της εντολής και της διεύθυνσης της τρέχουσας εντολής
- Καταχωρητής σύνδεσης (LR) ή μετρητής (CR): η διεύθυνση της επόμενης εντολής καθορίζεται από τα περιεχόμενα ενός από τους δύο παραπάνω καταχωρητές

Οι εντολές του συνόλου εντολών μπορούν να καταταχθούν στις ακόλουθες ομάδες:

- Εντολές ακεραίων: περιλαμβάνουν αριθμητικές και λογικές εντολές, εντολές σύγκρισης, εντολές ολίσθησης και περιστροφής
- Εντολές κινητής υποδιαστολής: περιλαμβάνουν μεταξύ άλλων αριθμητικές εντολές και εντολές σύγκρισης που επιδρούν σε τελεστές κινητής υποδιαστολής
- Εντολές φόρτωσης-αποθήκευσης: εντολές πρόσβασης στη μνήμη τόσο για ακεραίους όσο και για αριθμούς κινητής υποδιαστολής
- Εντολές συγχρονισμού μνήμης: ελέγχουν την διάταξη των εντολών πρόσβασης στη μνήμη ως προς ασύγχρονα γεγονότα, καθώς και τη σειρά με την οποία οι προσβάσεις στη μνήμη γίνονται αντιληπτές από τους υπόλοιπους επεξεργαστές του συστήματος
- Εντολές ελέγχου ροής: περιλαμβάνουν εντολές διακλάδωσης με ή χωρίς συνθήκη, καθώς και άλλες εντολές που επηρεάζουν την ροή εκτέλεσης του προγράμματος

- Εντολές ελέγχου του επεξεργαστή: περιλαμβάνουν εντολές μεταφοράς από/προς καταχωρητές ειδικού σκοπού, στους οποίους πρέπει να έχει πρόσβαση προνομιούχο λογισμικό
- Εντολές ελέγχου μνήμης και κρυφών μνημών: ελέγχουν τις κρυφές μνήμες, τους Translation Lookaside Buffers, και τους καταχωρητές τμημάτων
- Εντολές εξωτερικού ελέγχου: επιτρέπουν στα προγράμματα χρήστη να επικοινωνούν με συσκευές ειδικού σκοπού

2.2.2.2 Σύνολο εντολών Vector/SIMD Multimedia Extension

Οι εντολές σε αυτό το σύνολο εντολών, όπως και στο σύνολο εντολών αρχιτεκτονικής PowerPC, έχουν μέγεθος 4 byte και ευθυγράμμιση λέξης. Οι τελεστές των εντολών έχουν σταθερό μέγεθος 128 bit, και είναι διανύσματα που περιέχουν στοιχεία μεγέθους 8, 16 ή 32 bits που μπορεί να είναι ακέραιοι, αριθμοί κινητής ή σταθερής υποδιαστολής. Οι εντολές αυτού του συνόλου επιτρέπουν την ταυτόχρονη εκτέλεση πράξεων στα στοιχεία των διανυσμάτων.

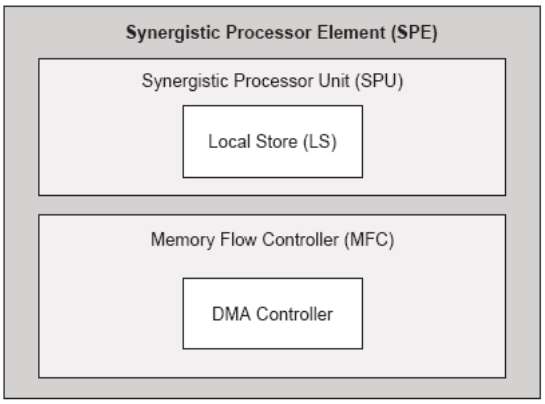
Η μονάδα εκτέλεσης των εντολών σε διανύσματα (Vector/SIMD Multimedia Extension Unit – VXU) μπορεί να λειτουργεί ταυτόχρονα με τις μονάδες σταθερής υποδιαστολής (Fixed-Point Unit – FXU) και κινητής υποδιαστολής (Floating-Point Unit – FPU) της κεντρικής μονάδας επεξεργασίας. Οι εντολές αυτές είναι σχεδιασμένες ώστε η υλοποίηση τους σε αρχιτεκτονική αγωγού να είναι εύκολη, καθώς δεν προκαλούν εξαιρέσεις, δεν υποστηρίζουν πολύπλοκες λειτουργίες ή μη ευθυγραμμισμένες προσβάσεις στη μνήμη και δεν μοιράζονται πολλούς πόρους με τις υπόλοιπες μονάδες εκτέλεσης της κεντρικής μονάδας επεξεργασίας. Οι εντολές φόρτωσης και αποθήκευσης διανυσμάτων χρησιμοποιούν διευθυνσιοδότηση μέσω αθροίσματος ενός καταχωρητή βάσης και ενός καταχωρητή δείκτη (register + register), μέθοδος που βοηθά ιδιαίτερα την πρόσβαση σε στοιχεία πινάκων. Οι περισσότερες εντολές έχουν τρεις τελεστές, συνήθως δύο τελεστές προέλευσης και έναν τελεστή προορισμού. Μπορούν να κατηγοριοποιηθούν ως εξής:

- Εντολές διανυσμάτων ακεραίων: περιλαμβάνουν αριθμητικές και λογικές εντολές, εντολές σύγκρισης, εντολές ολίσθησης και περιστροφής
- Εντολές διανυσμάτων κινητής υποδιαστολής: περιλαμβάνουν αριθμητικές εντολές, εντολές σύγκρισης, εντολές στρογγυλοποίησης και μετατροπής, που επιδρούν σε αριθμούς κινητής υποδιαστολής απλής ακρίβειας
- Εντολές φόρτωσης και αποθήκευσης διανυσμάτων: περιλαμβάνουν βασικές εντολές φόρτωσης ακεραίων και αριθμών κινητής υποδιαστολής

- Εντολές μετάθεσης και διάταξης διανυσμάτων: περιλαμβάνουν μεταξύ άλλων εντολές ολίσθησης (shift), μετάθεσης (permutation), επιλογής (select),pack, unpack, merge
- Εντολές ελέγχου του επεξεργαστή: ελέγχουν τον καταχωρητή VSCR
- Εντολές ελέγχου μνήμης

2.3 Συνεπεξεργαστές (SPE)

Κάθε ένας από τους συνεπεξεργαστές είναι ένας επεξεργαστής 128-bit αρχιτεκτονικής μειωμένου συνόλου εντολών (RISC) που ειδικεύεται σε εφαρμογές SIMD απαιτητικές ως προς τους υπολογισμούς και ως προς την ποσότητα των δεδομένων. Η συχνότητα λειτουργίας των συνεπεξεργαστών είναι 3.2 GHz. Αποτελούνται από δύο κύριες μονάδες, την μονάδα επεξεργασίας (Synergistic Processor Unit – SPU) και τον ελεγκτή πρόσβασης στη μνήμη (Memory Flow Controller – MFC), όπως φαίνεται και στο σχήμα που ακολουθεί:



Οι συνεπεξεργαστές παρέχουν ένα πλήρως ντετερμινιστικό περιβάλλον λειτουργίας. Μιας και δεν υπάρχουν κρυφές μνήμες, δεν υπάρχουν ούτε αστοχίες και απρόβλεπτες καθυστερήσεις που να σχετίζονται με αυτές και να επηρεάζουν την απόδοση των συνεπεξεργαστών. Οι κανόνες δρομολόγησης των εντολών στις σωληνώσεις (pipelines) είναι απλοί κι έτσι είναι εύκολο να υπολογιστεί στατικά η απόδοση του κώδικα. Παρότι οι εντολές φόρτωσης και αποθήκευσης από την μονάδα επεξεργασίας, οι εντολές άμεσης πρόσβασης στη μνήμη και οι προανακλήσεις εντολών ανταγωνίζονται για τη χρήση της τοπικής μνήμης, οι εντολές άμεσης πρόσβασης στη μνήμη έχουν πρόσβαση στην τοπική μνήμη μόνο το πολύ έναν κύκλο ανά οχτώ κύκλους. Επίσης η προανάκληση των εντολών μεταφέρει κάθε φορά ικανοποιητικό αριθμό εντολών, οπότε η συνολική επίδραση των

εντολών άμεσης πρόσβασης στη μνήμη και των προανακλήσεων στην απόδοση των εντολών πρόσβασης στην τοπική μνήμη και στο συνολικό χρόνο εκτέλεσης του προγράμματος είναι περιορισμένη.

2.3.1 Μονάδα επεξεργασίας (SPU)

Η μονάδα επεξεργασίας του κάθε συνεπεξεργαστή είναι ένας ανεξάρτητος υπολογιστής με το δικό του μετρητή εντολών. Αναλαμβάνει τον έλεγχο και την εκτέλεση των εντολών. Περιλαμβάνει ένα ενοποιημένο αρχείο 128 καταχωρητών, μεγέθους 128 bit ο καθένας, μια τοπική μνήμη (local store) μεγέθους 256KB, μια μονάδα ελέγχου των εντολών, μια μονάδα φόρτωσης και αποθήκευσης, δύο μονάδες πράξεων σταθερής υποδιαστολής, μια μονάδα πράξεων κινητής υποδιαστολής και μια διεπαφή προς τα κανάλια (SPU Channels) του ελεγκτή πρόσβασης στη μνήμη (MFC). Υλοποιεί ένα νέο, ειδικά σχεδιασμένο σύνολο εντολών (SPU Instruction Set Architecture). Οι εντολές που εκτελεί είναι αποθηκευμένες στη δική του τοπική μνήμη, όπως και τα δεδομένα που χρησιμοποιεί για εγγραφή/ανάγνωση. Τα δεδομένα και οι εντολές που περιέχονται στην τοπική μνήμη μεταφέρονται εκεί μέσω μεταφορών άμεσης πρόσβασης στη μνήμη (DMA), για τις οποίες η μονάδα επεξεργασίας δίνει την εντολή, ώστε να εκτελεστούν με τη βοήθεια του ελεγκτή πρόσβασης στη μνήμη (MFC).

Οι διευθύνσεις της τοπικής μνήμης μπορούν να απεικονιστούν σε ενεργές διευθύνσεις του κύριου χώρου διευθύνσεων με τη βοήθεια προνομιούχου λογισμικού που εκτελείται στο κεντρικό στοιχείο επεξεργασίας (PPE). Κύριος χώρος διευθύνσεων είναι το σύνολο των ενεργών διευθύνσεων (effective addresses), που περιλαμβάνει την κύρια μνήμη, τις τοπικές μνήμες άλλων συνεπεξεργαστών και τους καταχωρητές που έχουν απεικονιστεί στη μνήμη, για παράδειγμα καταχωρητές εισόδου-εξόδου απεικονισμένοι στην μνήμη. Οι ενεργές διευθύνσεις αυτές μπορούν στη συνέχεια να χρησιμοποιηθούν από το κεντρικό στοιχείο επεξεργασίας ή από άλλους συνεπεξεργαστές για την πρόσβαση στα δεδομένα της συγκεκριμένης τοπικής μνήμης μέσω μεταφορών άμεσης πρόσβασης στη μνήμη (DMA). Οι μεταφορές άμεσης πρόσβασης στη μνήμη μεταξύ της τοπικής μνήμης και του κύριου χώρου διευθύνσεων διατηρούν τη συνάφεια.

Η αρχιτεκτονική της μονάδας επεξεργασίας του συνεπεξεργαστή θέτει τους ακόλουθους περιορισμούς:

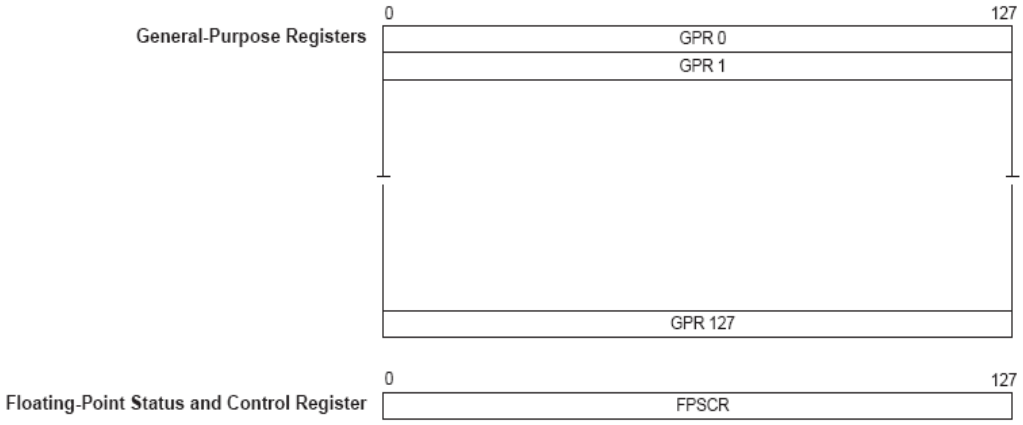
- Δεν μπορεί να έχει άμεση πρόσβαση στην κεντρική μνήμη και στον κύριο χώρο διευθύνσεων (μέσω ενεργών διευθύνσεων) γενικά. Η πρόσβαση στον κύριο χώρο διευθύνσεων γίνεται μόνο μέσω μεταφορών άμεσης πρόσβασης στη μνήμη (DMA).
- Δεν μπορεί να έχει άμεση πρόσβαση σε πόρους ελέγχου του συστήματος, όπως για παράδειγμα

στους πίνακες σελίδων. Οι πληροφορίες που χρειάζονται για τη μετάφραση των ενεργών διευθύνσεων παρέχονται στον συνεπεξεργαστή με τη βοήθεια προνομιούχου λογισμικού που εκτελείται στο κεντρικό στοιχείο επεξεργασίας (PPE).

- Οι διευθύνσεις της τοπικής μνήμης που ανήκει σε κάθε συνεπεξεργαστή δεν υπόκεινται σε κάποιο σχήμα προστασίας ή μετάφρασης (πρόσβαση από τον συνεπεξεργαστή στον οποίο ανήκει η τοπική μνήμη).

2.3.1.1 Καταχωρητές

Όλες οι εντολές που εκτελούν υπολογισμούς επιδρούν στα περιεχόμενα καταχωρητών και όχι σε θέσεις μνήμης. Το σύνολο των καταχωρητών της μονάδας επεξεργασίας φαίνεται στο σχήμα που ακολουθεί:



Οι καταχωρητές αυτοί είναι:

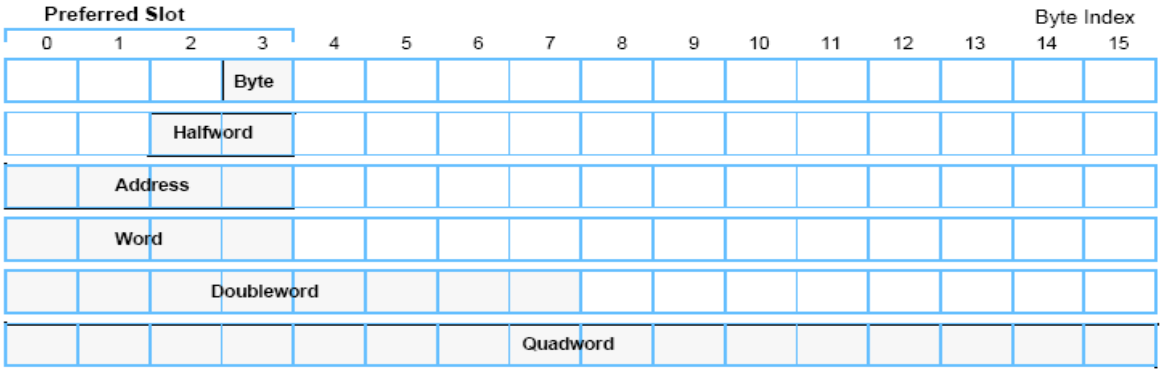
- Καταχωρητές γενικής χρήσης (General-Purpose Registers – GPRs): 128 καταχωρητές των 128 bit που χρησιμοποιούνται για αποθήκευση δεδομένων όλων των τύπων
- Καταχωρητής κατάστασης και ελέγχου κινητής υποδιαστολής (Floating-Point Status and Control Register – FPSCR): καταχωρητής 128 bit που ενημερώνεται μετά από κάθε πράξη κινητής υποδιαστολής και περιέχει πληροφορίες για το αποτέλεσμα της πράξης και για πιθανές εξαιρέσεις που σχετίζονται με αυτήν

2.3.1.2 Σύνολο εντολών

Το σύνολο εντολών που υποστηρίζει η μονάδα επεξεργασίας των συνεπεξεργαστών είναι ένα νέο

σύνολο εντολών, ειδικά σχεδιασμένο για τον επεξεργαστή Cell και ονομάζεται σύνολο εντολών SPU (SPU Instruction Set). Οι εντολές του επιδρούν κυρίως σε διανυσματικά δεδομένα, σταθερής και κινητής υποδιαστολής, ενώ υποστηρίζονται σε περιορισμένο βαθμό λειτουργίες σε μονοδιάστατα δεδομένα. Καθώς τα σύνολα εντολών που υποστηρίζουν η κεντρική μονάδα επεξεργασίας (PPU) και οι μονάδες επεξεργασίας των συνεπεξεργαστών (SPUs) είναι διαφορετικά, η μεταγλώττιση των αντίστοιχων προγραμμάτων πρέπει να γίνεται με ξεχωριστούς μεταγλωττιστές.

Στους συνεπεξεργαστές, όπως συμβαίνει γενικά στον επεξεργαστή Cell, χρησιμοποιείται η Big Endian διαρρύθμιση των bytes, οπότε για κάθε λέξη το byte με τη μικρότερη διεύθυνση είναι το περισσότερο σημαντικό byte, και το bit με τη μικρότερη αρίθμηση είναι το περισσότερο σημαντικό bit. Τα bits στους καταχωρητές αριθμούνται από το 0-127 από τα αριστερά προς τα δεξιά, με το 0 να είναι το περισσότερο σημαντικό bit και το bit 127 το λιγότερο σημαντικό. Στην SPU χρησιμοποιούνται οι ακόλουθοι τύποι δεδομένων: byte (8 bits), μισή λέξη (halfword – 16 bits), λέξη (word – 32 bits), διπλή λέξη (doubleword – 64 bits), τετραπλή λέξη (quadword – 128 bits). Στο σχήμα που ακολουθεί φαίνεται η προτιμώμενη θέση καθενός από τους παραπάνω τύπους σε έναν καταχωρητή, δηλαδή η θέση στην οποία βρίσκονται τα δεδομένα κάθε φορά που μια εντολή παράγει ή χρησιμοποιεί μονοδιάστατα δεδομένα αυτού του τύπου:



Οι 204 εντολές που ορίζονται στο σύνολο εντολών χωρίζονται ανάλογα με τη λειτουργία τους στις ακόλουθες κατηγορίες: εντολές φόρτωσης/αποθήκευσης από/προς τη μνήμη (16 εντολές), εντολές διαμόρφωσης σταθερών (6 εντολές), εντολές πράξεων σε ακεραίους και λογικές εντολές (59 εντολές), εντολές ολίσθησης και περιστροφής (31 εντολές), εντολές σύγκρισης, διακλάδωσης και τερματισμού (40 εντολές), εντολές υπόδειξης για διακλαδώσεις (3 εντολές), εντολές κινητής υποδιαστολής (28 εντολές), εντολές ελέγχου (8 εντολές), εντολές σχετικές με τα κανάλια – SPU Channels (3 εντολές), εντολές λειτουργίας διακοπών της SPU (7 εντολές) και εντολές συγχρονισμού και διάταξης (3 εντολές).

2.3.1.3 Τοπική μνήμη

Η τοπική μνήμη μπορεί να θεωρηθεί ως μια κρυφή μνήμη διαχειριζόμενη από το λογισμικό, η οποία γεμίζει και αδειάζει με μεταφορές άμεσης πρόσβασης στη μνήμη (DMA). Τα κύρια χαρακτηριστικά της τοπικής μνήμης είναι τα εξής:

- αποθήκευση δεδομένων και εντολών
- εύρος ζώνης για εντολές φόρτωσης/αποθήκευσης: 16 bytes/κύκλο, ευθυγραμμισμένα κατά τετραπλή λέξη (quadword aligned)
- εύρος ζώνης για δεδομένα από μεταφορές άμεσης πρόσβασης στη μνήμη (DMA): 128 bytes/κύκλο
- εύρος ζώνης για προανάκληση εντολών: 128 bytes/κύκλο

Όταν υπάρχει ανταγωνισμός για την πρόσβαση στην τοπική μνήμη, η μονάδα επεξεργασίας του συνεπεξεργαστή διατάσσει τις αιτήσεις πρόσβασης στην τοπική μνήμη με βάση την ακόλουθη σειρά προτεραιότητας (η μεγαλύτερη προτεραιότητα πρώτη):

- εγγραφές και αναγνώσεις από εντολές άμεσης πρόσβασης στη μνήμη (DMA)
- εντολές φόρτωσης και αποθήκευσης από την μονάδα επεξεργασίας του συνεπεξεργαστή (SPU)
- προανάκληση εντολών

Οι εγγραφές και αναγνώσεις από εντολές άμεσης πρόσβασης στη μνήμη (DMA) έχουν πάντα τη μεγαλύτερη προτεραιότητα. Επειδή το υλικό περιορίζει το ρυθμό ανάγνωσης/εγγραφής από εντολές DMA σε 8 bytes/κύκλο επεξεργαστή (16 bytes/κύκλο διαύλου), οι εγγραφές και οι αναγνώσεις αυτής της κατηγορίας συσσωρεύονται και απασχολούν την τοπική μνήμη το πολύ για έναν ανά οχτώ κύκλους. Επομένως στις περισσότερες περιπτώσεις η επίδραση των προσβάσεων αυτών στην διαθεσιμότητα της τοπικής μνήμης για προσβάσεις άλλου είδους μπορεί να θεωρηθεί αμελητέα.

Η επόμενη μεγαλύτερη προτεραιότητα δίνεται στις εντολές φόρτωσης και αποθήκευσης από την μονάδα επεξεργασίας του συνεπεξεργαστή (SPU). Η λογική στην οποία βασίζεται αυτή η σειρά προτεραιότητας είναι ότι οι εντολές αυτές συνήθως βοηθούν την συνέχιση της εκτέλεσης του προγράμματος, ενώ οι προανακλήσεις εντολών βασίζονται σε υποθέσεις. Για το λόγο αυτό στις προανακλήσεις εντολών δίνεται η μικρότερη προτεραιότητα. Οι προανακλήσεις εντολών μεταφέρουν 32 εντολές ανά αίτηση.

2.3.1.4 Αρχιτεκτονική αγωγού (pipelining)

Η μονάδα επεξεργασίας έχει δύο σωληνώσεις, που ονομάζονται άρτια (even, pipeline 0) και περιττή (odd, pipeline 1), στις οποίες μπορεί να εκδώσει και να ολοκληρώσει μέχρι δύο εντολές ανά κύκλο, μία σε κάθε σωλήνωση. Η σωλήνωση στην οποία θα εκτελεστεί μια εντολή εξαρτάται από τον τύπο της, δηλαδή εκτελείται στη σωλήνωση όπου βρίσκεται η μονάδα για την εκτέλεση της εντολής. Κάθε τύπος εντολής μπορεί να εκτελεστεί επομένως σε μία μόνο από τις δύο σωληνώσεις. Για παράδειγμα, οι εντολές φόρτωσης/αποθήκευσης εκτελούνται στην περιττή σωλήνωση, ενώ οι εντολές κινητής υποδιαστολής εκτελούνται στην άρτια σωλήνωση. Η μονάδα επεξεργασίας εκτελεί τις εντολές με τη σειρά του προγράμματος. Κάθε εντολή είναι μέρος ενός ζεύγους εντολών που είναι στοιχισμένες κατά διπλή λέξη και καλείται ομάδα ανάκλησης (fetch group). Μια τέτοια ομάδα μπορεί να περιέχει μία ή δύο έγκυρες εντολές. Η πρώτη εντολή της ομάδας, αυτή που βρίσκεται σε διεύθυνση άρτιας λέξης, προορίζεται να εκτελεστεί στην άρτια σωλήνωση, ενώ η δεύτερη εντολή, που βρίσκεται σε διεύθυνση περιττής λέξης, στην περιττή σωλήνωση. Η διπλή έκδοση των δύο εντολών επιτυγχάνεται όταν και οι δύο εντολές μπορούν να εκδοθούν, δηλαδή όταν ικανοποιούνται οι εξαρτήσεις ως προς τους καταχωρητές (δεν υπάρχουν κίνδυνοι δεδομένων) και δεν υπάρχει δομικός κίνδυνος (σύγκρουση πόρων με προηγούμενες εντολές ή εντολές άμεσης πρόσβασης στη μνήμη). Αν δεν μπορούν να εκδοθούν και οι δύο εντολές μαζί, αλλά μπορεί να εκδοθεί η πρώτη από τις δύο, η μονάδα προχωράει στην έκδοσή της και εκδίδει την δεύτερη εντολή όταν αυτό είναι δυνατό. Όταν και οι δύο εντολές έχουν εκδοθεί, φορτώνεται ένα νέο ζεύγος εντολών.

2.3.2 Ελεγκτής πρόσβασης στη μνήμη (Memory Flow Controller)

Ο ελεγκτής πρόσβασης στη μνήμη περιλαμβάνει έναν ελεγκτή άμεσης πρόσβασης στη μνήμη (DMA controller) που υποστηρίζει τις μεταφορές άμεσης πρόσβασης στη μνήμη (DMA). Οι μεταφορές αυτές χρησιμοποιούνται από προγράμματα που εκτελούνται στην μονάδα επεξεργασίας του συνεπεξεργαστή (SPU), στις μονάδες επεξεργασίας άλλων συνεπεξεργαστών, ή στην κεντρική μονάδα επεξεργασίας, για να μετακινούν εντολές και δεδομένα μεταξύ της τοπικής μνήμης και του κύριου χώρου διευθύνσεων.

Για να υποστηρίξει τις μεταφορές άμεσης πρόσβασης στη μνήμη (DMA), ο ελεγκτής πρόσβασης στη μνήμη περιέχει ουρές αναμονής για τις εντολές DMA. Με τον τρόπο αυτό, η μονάδα επεξεργασίας μπορεί να ενταλθεί την εκτέλεση κάποιας μεταφοράς και να συνεχίσει την εκτέλεση του προγράμματός της, χωρίς να περιμένει να ολοκληρωθεί η μεταφορά, η οποία εκτελείται αυτόνομα και ασύγχρονα από τον ελεγκτή πρόσβασης στη μνήμη. Ο ελεγκτής πρόσβασης στη μνήμη έχει επιπλέον τη δυνατότητα να εκτελεί μία ακολουθία εντολών, ως απάντηση σε μια εντολή άμεσης πρόσβασης στη

μνήμη τύπου λίστας (DMA list command). Η αυτόνομη εκτέλεση εντολών άμεσης πρόσβασης στη μνήμη και εντολών επεξεργασίας επιτρέπει την απόκρυψη της καθυστέρησης πρόσβασης στη μνήμη με την κατάλληλη τοποθέτηση των εντολών άμεσης πρόσβασης στη μνήμη.

Κάθε μεταφορά άμεσης πρόσβασης στη μνήμη μπορεί να έχει μέγεθος μέχρι 16KB. Εντολές άμεσης πρόσβασης στη μνήμη τύπου λίστας μπορούν να εκδοθούν μόνο από την μονάδα επεξεργασίας που αντιστοιχεί στον συγκεκριμένο ελεγκτή πρόσβασης στη μνήμη. Οι λίστες αυτές μπορούν να περιέχουν μέχρι 2048 εντολές, καθεμία από τις οποίες έχει μέγιστο μέγεθος 16KB. Οι μεταφορές DMA είναι συναφείς ως προς τον κύριο χώρο διευθύνσεων. Οι πληροφορίες που χρειάζονται για την μετάφραση των ενεργών διευθύνσεων που δίνονται ως παράμετροι στις εντολές άμεσης πρόσβασης στη μνήμη παρέχονται στον ελεγκτή πρόσβασης στη μνήμη από το κεντρικό στοιχείο επεξεργασίας (PPE), και η προστασία και η μετάφραση των διευθύνσεων αυτών βασίζεται στους πίνακες τμημάτων και σελίδων της αρχιτεκτονικής PowerPC. Παρότι το κεντρικό στοιχείο επεξεργασίας (PPE) δύναται να απεικονίσει την τοπική μνήμη κάθε συνεπεξεργαστή καθώς και κάποιους άλλους πόρους σχετιζόμενους με αυτόν σε ενεργές διευθύνσεις, επιτρέποντας την πρόσβαση σε αυτούς μέσω των διευθύνσεων αυτών, είτε η πρόσβαση γίνεται από το ίδιο το κεντρικό στοιχείο επεξεργασίας είτε από άλλους συνεπεξεργαστές, η χρήση των διευθύνσεων αυτών δεν διατηρεί την συνάφεια του συστήματος.

Η μονάδα επεξεργασίας του συνεπεξεργαστή επικοινωνεί με τον ελεγκτή πρόσβασης στη μνήμη μέσω των καναλιών (SPU Channels), ενώ το κεντρικό στοιχείο επεξεργασίας (PPE) και οι υπόλοιπες συσκευές επικοινωνούν με τον ελεγκτή πρόσβασης στη μνήμη μέσω καταχωρητών εισόδου-εξόδου απεικονισμένων στην μνήμη (MMIO registers), οι οποίοι αντιστοιχούν στα κανάλια (channels) του συνεπεξεργαστή. Υποστηρίζονται κανάλια και αντίστοιχοι καταχωρητές απεικονισμένοι στη μνήμη για τις ακόλουθες λειτουργίες: έκδοση και παρακολούθηση εντολών άμεσης πρόσβασης στη μνήμη (DMA), έλεγχος των γεγονότων που σχετίζονται με τη μονάδα επεξεργασίας (SPU Events), υλοποίηση επικοινωνίας μεταξύ των διαφόρων επεξεργαστών μέσω των SPU Mailboxes και της ειδοποίησης των συνεπεξεργαστών μέσω σημάτων (Signal-Notification), πρόσβαση σε βοηθητικούς πόρους όπως ο SPU Decrementer , και άλλες λειτουργίες.

Τέλος, εκτός από την υποστήριξη μεταφορών άμεσης πρόσβασης στη μνήμη και των καναλιών και των αντίστοιχων απεικονισμένων στη μνήμη καταχωρητών, ο ελεγκτής πρόσβασης στη μνήμη υλοποιεί λειτουργίες δέσμευσης εύρους ζώνης του διαύλου, και συντονίζει τις λειτουργίες μεταξύ του συνεπεξεργαστή και των άλλων επεξεργαστών του συστήματος.

2.3.2.1 Κανάλια (SPU Channels)

Τα κανάλια (SPU Channels) είναι μονόδρομες διεπαφές μεταφοράς μηνυμάτων που υποστηρίζουν μηνύματα και εντολές μεγέθους 32 bit. Στον πίνακα που ακολουθεί φαίνονται τα κανάλια και οι ιδιότητες τους:

Κανάλι	Πλήρες όνομα	Μνημονικό όνομα	Μέγεθος (bits)	R/W (Read/Write)	Blocking
SPU Events					
0	SPU Read Event Status	SPU_RdEventStat	32	R	Ναι
1	SPU Write Event Mask	SPU_WrEventMask	32	W	Όχι
2	SPU Write Event Acknowledgment	SPU_WrEventAck	32	W	Όχι
SPU Signal Notification					
3	SPU Signal Notification 1	SPU_RdSigNotify1	32	R	Ναι
4	SPU Signal Notification 2	SPU_RdSigNotify2	32	R	Ναι
SPU Decrementer					
7	SPU Write Decrementer	SPU_WrDec	32	W	Όχι
8	SPU Read Decrementer	SPU_RdDec	32	R	Όχι
MFC Multisource Synchronization					
9	MFC Write Multisource Synchronization Request	MFC_WrMSSyncReq	32	W	Ναι
SPU and MFC Read Mask					
11	SPU Read Event Mask	SPU_RdEventMask	32	R	Όχι
12	MFC Read Tag-Group Query Mask	MFC_RdTagMask	32	R	Όχι
SPU State Management					
13	SPU Read Machine Status	SPU_RdMachStat	32	R	Όχι
14	SPU Write State Save-and-Restore	SPU_WrSRR0	32	W	Όχι
15	SPU Read State Save-and-Restore	SPU_RdSRR0	32	R	Όχι
MFC Command Parameters					
16	MFC Local Store Address	MFC_LSA	32	W	Όχι
17	MFC Effective Address High	MFC_EAH	32	W	Όχι
18	MFC Effective Address Low or List Address	MFC_EAL	32	W	Όχι

19	MFC Transfer Size or List Size	MFC_Size	16	W	Όχι
20	MFC Command Tag Identification	MFC_TagID	16	W	Όχι
21	MFC Command Opcode or ClassID	MFC_Cmd	32	W	Ναι
MFC Tag Status					
22	MFC Write Tag-Group Query Mask	MFC_WrTagMask	32	W	Όχι
23	MFC Write Tag Status Update Request	MFC_WrTagUpdate	32	W	Ναι
24	MFC Read Tag-Group Status	MFC_RdTagStat	32	R	Ναι
25	MFC Read List Stall-and-Notify Tag Status	MFC_RdListStallStat	32	R	Ναι
26	MFC Write List Stall-and-Notify Tag Acknowledgement	MFC_WrListStallAck	32	W	Όχι
27	MFC Read Atomic Command Status	MFC_RdAtomicStat	32	R	Ναι
SPU Mailboxes					
28	SPU Write Outbound Mailbox	SPU_WrOutMbox	32	W	Ναι
29	SPU Read Inbound Mailbox	SPU_RdInMbox	32	R	Ναι
30	SPU Write Outbound Interrupt Mailbox	SPU_WrOutIntrMbox	32	W	Ναι

Το λογισμικό που εκτελείται στην μονάδα επεξεργασίας του συνεπεξεργαστή χρησιμοποιεί ειδικές εντολές για την ανάγνωση από και την εγγραφή στα κανάλια και στις σχετιζόμενες ουρές. Η κεντρική μονάδα επεξεργασίας και οι υπόλοιπες συσκευές χρησιμοποιούν εντολές φόρτωσης και αποθήκευσης για την ανάγνωση από και την εγγραφή στους καταχωρητές που σχετίζονται με τα κανάλια και απεικονίζονται στη μνήμη (MMIO registers).

Κάθε κανάλι έχει μια τιμή (count) που του αντιστοιχεί και υποδεικνύει την υπολειπόμενη χωρητικότητα του καναλιού. Αυτή η τιμή μειώνεται κατά ένα κάθε φορά που εκδίδεται μια εντολή σχετική με το κανάλι και αυξάνεται κατά ένα κάθε φορά που μια λειτουργία σχετική με αυτό ολοκληρώνεται. Κάθε κανάλι είναι είτε blocking είτε non-blocking. Τα blocking κανάλια προκαλούν το σταμάτημα (stall) της λειτουργίας της μονάδας επεξεργασίας, δηλαδή αναστολή της εκτέλεσης και παραμονή σε κατάσταση χαμηλής ισχύος, κάθε φορά που αυτή διαβάζει ή γράφει σε ένα κανάλι του οποίου η τιμή είναι 0.

Τα κύρια χαρακτηριστικά των καναλιών είναι τα ακόλουθα:

- Όλες οι ενέργειες στη διεπαφή των καναλιών είναι μονόδρομες.

- Κάθε ενέργεια σε ένα κανάλι είναι ανεξάρτητη από τις ενέργειες στα υπόλοιπα κανάλια.
- Η πρόσβαση στους καταχωρητές που είναι απεικονισμένοι στη μνήμη (και αντιστοιχούν σε κάποιο κανάλι) έχει προτεραιότητα έναντι των ενεργειών στα κανάλια από την μονάδα επεξεργασίας.
- Οι λειτουργίες που επιδρούν στα κανάλια εκτελούνται με τη σειρά του προγράμματος.
- Η εγγραφή σε κάποιο δεσμευμένο κανάλι δεν έχει κανένα αποτέλεσμα.
- Η ανάγνωση από κάποιο δεσμευμένο κανάλι επιστρέφει μηδέν.
- Η ανάγνωση της τιμής (count) κάποιου δεσμευμένου καναλιού επιστρέφει μηδέν.

Το σύνολο εντολών SPU (SPU Instruction Set) ορίζει τρεις εντολές για αλληλεπίδραση με τα κανάλια. Οι εντολές αυτές είναι οι ακόλουθες:

- rdch (Read Channel – ανάγνωση από το κανάλι): προκαλεί την ανάγνωση δεδομένων από το κανάλι που καθορίζεται και τη μεταφορά τους στον επιλεγμένο καταχωρητή γενικού σκοπού.
- wrch (Write Channel – εγγραφή στο κανάλι): προκαλεί την ανάγνωση δεδομένων από τον επιλεγμένο καταχωρητή γενικού σκοπού και τη μεταφορά τους στο κανάλι που καθορίζεται.
- rchcnt (Read Channel Count – ανάγνωση της τιμής που σχετίζεται με το κανάλι): προκαλεί την αποθήκευση της τιμής που σχετίζεται με το δεδομένο κανάλι στον επιλεγμένο καταχωρητή γενικού σκοπού.

Αν ένα κανάλι εγγραφής είναι non-blocking, τότε η εντολή wrch επιστρέφει αμέσως, όμως όταν είναι blocking, αν η τιμή που σχετίζεται με το κανάλι είναι μηδέν, η μονάδα επεξεργασίας σταματά την εκτέλεση του προγράμματος και περνά σε κατάσταση χαμηλής ισχύος, μέχρι να λάβει επιβεβαίωση από το κανάλι ότι η εγγραφή ολοκληρώθηκε. Οι εφαρμογές που πραγματοποιούν εγγραφές σε blocking κανάλια πρέπει, αν θέλουν να αποφύγουν αυτό το σταμάτημα της λειτουργίας, να εξασφαλίσουν μέσω της εντολής rchcnt ότι η τιμή του καναλιού είναι μη μηδενική πριν προχωρήσουν στην εγγραφή. Τα ίδια συμβαίνουν και με τα blocking κανάλια ανάγνωσης, με τη χρήση της εντολής rdch αντί της wrch. Εδώ, η μη μηδενική τιμή που σχετίζεται με το κανάλι δηλώνει ότι τα δεδομένα που περιέχονται στο κανάλι είναι έγκυρα. Οι εφαρμογές και σε αυτή την περίπτωση μπορούν να εφαρμόσουν είτε polling του καναλιού, είτε να εκτελέσουν κατευθείαν την εντολή ανάγνωσης από το κανάλι, με το ενδεχόμενο να σταματήσει η εκτέλεση του προγράμματος ώσπου τα δεδομένα να είναι έγκυρα.

2.3.2.2 SPE Mailboxes

Τα Mailboxes είναι ουρές που υποστηρίζουν την ανταλλαγή μηνυμάτων μεγέθους 32-bit μεταξύ ενός συνεπεξεργαστή και των υπόλοιπων συσκευών. Για την αποστολή μηνυμάτων από τον συνεπεξεργαστή υπάρχουν δύο Mailboxes χωρητικότητας ενός μηνύματος το καθένα, τα SPU Write Outbound Mailbox και SPU Write Outbound Interrupt Mailbox, ενώ για την λήψη μηνυμάτων από τον συνεπεξεργαστή, υπάρχει ένα Mailbox χωρητικότητας τεσσάρων μηνυμάτων, το SPU Read Inbound Mailbox.

Κάθε Mailbox αντιστοιχεί σε ένα κανάλι καθώς και στον ανάλογο καταχωρητή που απεικονίζεται στην μνήμη. Για την αλληλεπίδραση με τα Mailboxes η μονάδα επεξεργασίας του συνεπεξεργαστή χρησιμοποιεί τις εντολές ανάγνωσης και εγγραφής στα αντίστοιχα κανάλια, ενώ το κεντρικό στοιχείο επεξεργασίας και οι υπόλοιπες συσκευές χρησιμοποιούν εντολές φόρτωσης και αποθήκευσης από/προς τη διεύθυνση που αντιστοιχεί στον απεικονισμένο στη μνήμη καταχωρητή που σχετίζεται με το κανάλι.

Τα δεδομένα που γράφονται από τη μονάδα επεξεργασίας σε ένα από τα δύο πρώτα Mailboxes (SPU Write Outbound Mailbox, SPU Write Outbound Interrupt Mailbox) μέσω μιας εντολής εγγραφής στο αντίστοιχο κανάλι (wrch), είναι διαθέσιμα προς ανάγνωση από τους υπόλοιπους επεξεργαστές και τις συσκευές του συστήματος μέσω του αντίστοιχου απεικονισμένου στη μνήμη καταχωρητή (MMIO register). Αντίστοιχα, τα δεδομένα που γράφουν οι άλλοι επεξεργαστές και οι διάφορες συσκευές στο SPU Read Inbound Mailbox μέσω του αντίστοιχου MMIO καταχωρητή είναι διαθέσιμα στην μονάδα επεξεργασίας, η οποία μπορεί να τα διαβάσει μέσω μιας εντολής ανάγνωσης (rdch) από το αντίστοιχο κανάλι. Μια ανάγνωση από κάποιο από τα SPU Write Outbound Mailboxes ή μια εγγραφή στο SPU Read Inbound Mailbox μέσω των καταχωρητών MMIO μπορεί να προγραμματιστεί ώστε να δημιουργεί ένα γεγονός (SPE Event), το οποίο με τη σειρά του μπορεί να προκαλέσει μια διακοπή (SPE interrupt). Μια εντολή wrch στο SPU Write Outbound Interrupt Mailbox μπορεί να προγραμματιστεί ώστε να προκαλεί μια διακοπή σε έναν άλλο επεξεργαστή ή συσκευή του συστήματος.

Κάθε φορά που ένα πρόγραμμα που εκτελείται στο κεντρικό στοιχείο επεξεργασίας (PPE) γράφει μια τιμή στο SPU Read Inbound Mailbox, η τιμή που αντιστοιχεί στο κανάλι αυτό αυξάνεται κατά ένα. Κάθε φορά που η μονάδα επεξεργασίας του συνεπεξεργαστή (SPU) διαβάζει από το κανάλι μέσω της εντολής rdch, η τιμή που αντιστοιχεί στο κανάλι μειώνεται. Το κανάλι αυτό αποτελεί μια ουρά FIFO, δηλαδή η τιμή που διαβάζει η SPU είναι η πιο παλιά τιμή που γράφτηκε στο SPU Read Inbound Mailbox. Αν στο Mailbox γραφούν γίνουν από το PPE περισσότερες από τέσσερις εγγραφές, πριν τα δεδομένα διαβαστούν από την SPU, η τιμή που αντιστοιχεί στο κανάλι παραμένει ίση με τέσσερα και

τα δεδομένα που βρίσκονται στην τέταρτη θέση αντικαθίστανται από τα τελευταία δεδομένα που έγραψε το PPE. Για παράδειγμα, αν το PPE γράψει στο Mailbox πέντε φορές δεδομένα πριν τα διαβάσει η SPU, τα δεδομένα που θα διαβάσει η SPU με τέσσερις διαδοχικές εντολές ανάγνωσης από το αντίστοιχο κανάλι θα είναι με τη σειρά τα πρώτα, τα δεύτερα, τα τρίτα και τα πέμπτα δεδομένα που έγραψε το PPE.

Οι λειτουργίες στα Mailboxes είναι blocking (από τη μεριά του συνεπεξεργαστή). Μια εγγραφή σε ένα από τα SPU Write Outbound Mailboxes όταν τα προηγούμενα δεδομένα δεν έχουν ακόμα διαβαστεί, προκαλεί σταμάτημα της εκτέλεσης του προγράμματος μέχρι να αδειάσει το Mailbox (να διαβαστούν τα δεδομένα) ώστε να είναι δυνατή η ολοκλήρωση της εγγραφής. Αντίστοιχα με μια ανάγνωση του SPU Read Inbound Mailbox όταν δεν υπάρχουν διαθέσιμα δεδομένα προκαλείται και πάλι παύση της εκτέλεσης του προγράμματος μέχρι να υπάρξουν δεδομένα προς ανάγνωση. Η παύση μπορεί και στις δύο περιπτώσεις να αποφευχθεί είτε χρησιμοποιώντας την εντολή rchcnt για υλοποίηση λειτουργίας polling του Mailbox, είτε προγραμματίζοντας τον συνεπεξεργαστή ώστε να αντιδρά μέσω του συστήματος των διακοπών στα γεγονότα που σχετίζονται με τα Mailboxes.

Παρότι τα Mailboxes αποσκοπούν κυρίως στην επικοινωνία μεταξύ του PPE και των συνεπεξεργαστών, μπορούν επίσης να χρησιμοποιηθούν για επικοινωνία μεταξύ των συνεπεξεργαστών ή μεταξύ του συνεπεξεργαστή και άλλων συσκευών του συστήματος. Για να μπορεί να συμβεί αυτό, πρέπει το προνομιούχο λογισμικό που εκτελείται στο PPE να επιτρέψει σε έναν συνεπεξεργαστή να έχει πρόσβαση στα Mailboxes ενός άλλου συνεπεξεργαστή.

2.3.2.3 Κανάλια ειδοποίησης συνεπεξεργαστών μέσω σημάτων (SPE Signal Notification Channels)

Τα κανάλια ειδοποίησης μέσω σημάτων (signal-notification channels) είναι εισερχόμενοι καταχωρητές. Μπορούν να χρησιμοποιηθούν από την κεντρική μονάδα επεξεργασίας, από τους υπόλοιπους συνεπεξεργαστές ή από άλλες συσκευές για να αποστείλουν δεδομένα, όπως για παράδειγμα κάποια σημαία συγχρονισμού, σε έναν συνεπεξεργαστή. Κάθε συνεπεξεργαστής έχει δύο τέτοιους καταχωρητές, καθένας από τους οποίους έχει μέγεθος 32 bit και αντιστοιχεί σε έναν καταχωρητή απεικονισμένο στη μνήμη στον οποίο τα δεδομένα του σήματος γράφονται από τον επεξεργαστή-αποστολέα. Οι συνεπεξεργαστές για να αποστείλουν ένα σήμα σε κάποιον άλλο συνεπεξεργαστή χρησιμοποιούν μία από τις ακόλουθες τρεις εντολές: sndsig, sndsigf, sndsigb.

Ο κάθε συνεπεξεργαστής μπορεί μόνο να διαβάσει τα τοπικά του κανάλια ειδοποίησης μέσω σημάτων. Το κεντρικό στοιχείο επεξεργασίας και οι υπόλοιποι συνεπεξεργαστές μπορούν είτε να διαβάσουν είτε να γράψουν τον αντίστοιχο καταχωρητή που είναι απεικονισμένος στη μνήμη. Μια ανάγνωση ενός

από τα δύο κανάλια από τον συνεπεξεργαστή στον οποίο ανήκουν προκαλεί μηδενισμό του αντίστοιχου καταχωρητή, ενώ αυτό δε συμβαίνει στην περίπτωση της ανάγνωσής τους μέσω των καταχωρητών που είναι απεικονισμένοι στη μνήμη. Τα κανάλια είναι blocking, και έτσι μια ανάγνωσή τοπικά θα προκαλέσει παύση της λειτουργίας του συνεπεξεργαστή αν δεν υπάρχουν σήματα προς επεξεργασία. Σύμφωνα με τα παραπάνω, κάθε συνεπεξεργαστής μπορεί να χρησιμοποιήσει με τρεις τρόπους τα κανάλια αυτά: είτε εκτελώντας απευθείας εντολές ανάγνωσης, με ενδεχόμενη παύση της λειτουργίας του, είτε με polling, ελέγχοντας την τιμή που αντιστοιχεί στο κανάλι πριν διαβάσει τα πραγματικά δεδομένα, είτε ρυθμίζοντας μια διακοπή ως απάντηση στη λήψη ενός σήματος.

Υπάρχουν δύο ρυθμίσεις στις οποίες μπορεί να λειτουργούν τα κανάλια αυτά. Η πρώτη ρύθμιση ονομάζεται overwrite mode (ή σηματοδοσία ένα-προς-ένα) και σε αυτήν η αποστολή ενός σήματος προκαλεί την διαγραφή των παλιότερων δεδομένων, γράφοντας τα νέα δεδομένα “πάνω” από αυτά. Η δεύτερη ρύθμιση ονομάζεται OR mode (ή σηματοδοσία πολλά-προς-ένα). Σε αυτήν η αποστολή ενός σήματος έχει ως αποτέλεσμα την εκτέλεση της λογικής πράξης Ή (OR) μεταξύ των παλιών και των νέων δεδομένων, και την αποθήκευση του αποτελέσματος. Στην περίπτωση της πρώτης ρύθμισης (overwrite mode) η απόδοση είναι παραπλήσια της απόδοσης που έχει η χρήση των Mailboxes.

Οι διαφορές μεταξύ των καναλιών ειδοποίησης μέσω σημάτων (Signal Notification Channels) και των Mailboxes είναι οι ακόλουθες:

- Χωρητικότητα: τα κανάλια ειδοποίησης μέσω σημάτων είναι καταχωρητές, ενώ τα Mailboxes είναι ουρές.
- Κατεύθυνση: τα κανάλια ειδοποίησης μέσω σημάτων είναι μόνο εισερχόμενα, ενώ υπάρχουν και εισερχόμενα και εξερχόμενα Mailboxes. Παρόλα αυτά, ένας συνεπεξεργαστής μπορεί να χρησιμοποιήσει τις εντολές αποστολής σημάτων για να στείλει δεδομένα προς κάποιον άλλον συνεπεξεργαστή.
- Διακοπές: ένα από τα Mailboxes προκαλεί διακοπή στο κεντρικό στοιχείο επεξεργασίας, ενώ δεν υπάρχει αντίστοιχη δυνατότητα με τα κανάλια ειδοποίησης μέσω σημάτων.
- Πολλά-προς-ένα: Σε αντίθεση με τα Mailboxes, τα κανάλια ειδοποίησης μέσω σημάτων μπορούν να ρυθμιστούν ώστε να δέχονται πολλά σήματα στα οποία να εκτελείται η λογική πράξη Ή (OR mode) ή κάθε φορά το νέο σήμα αντιγράφεται πάνω από το παλιό, σβήνοντάς το (overwrite mode).
- Ξεχωριστές εντολές: τα κανάλια ειδοποίησης μέσω σημάτων διαθέτουν ειδικές εντολές για την αποστολή σημάτων σε αυτά.
- Μηδενισμός: η ανάγνωση ενός καταχωρητή ειδοποίησης μέσω σημάτων από τον

συνεπεξεργαστή στον οποίο ανήκει μηδενίζει αυτόματα τα περιεχόμενά του.

- Τιμές σχετιζόμενες με τα κανάλια: έχουν διαφορετική σημασία στις δύο περιπτώσεις. Στην περίπτωση των Mailboxes δείχνουν τον αριθμό των ελεύθερων θέσεων στην ουρά που αντιστοιχεί στο Mailbox, ενώ στην περίπτωση των καναλιών ειδοποίησης μέσω σημάτων υποδεικνύουν την ύπαρξη εκκρεμών σημάτων.
- Πλήθος: υπάρχουν τρία Mailboxes και δύο κανάλια ειδοποίησης μέσω σημάτων σε κάθε συνεπεξεργαστή.

2.4 Δίαυλος διασύνδεσης στοιχείων (EIB – Element Interconnect Bus)

Ο δίαυλος διασύνδεσης στοιχείων (EIB – Element Interconnect Bus) συνδέει έναν αριθμό από στοιχεία μεταξύ τους: το κεντρικό στοιχείο επεξεργασίας (PPE), τους συνεπεξεργαστές (SPEs), τον ελεγκτή μνήμης (MIC) και τον ελεγκτή συσκευών εισόδου-εξόδου (BIC). Η συχνότητα λειτουργίας του διαύλου είναι η μισή από τη συχνότητα λειτουργίας των στοιχείων επεξεργασίας, επομένως ισούται με 1.6 GHz.

2.4.1 Αρχιτεκτονική

Υπάρχουν ξεχωριστά δίκτυα για τη μεταφορά των δεδομένων και των εντολών (αιτήσεων για μεταφορά δεδομένων μεταξύ των στοιχείων που συνδέονται στο δίαυλο). Κάθε ένα από τα στοιχεία που συνδέονται σε αυτό έχει μία θύρα εισόδου και μία εξόδου, εκτός από τον ελεγκτή συσκευών εισόδου-εξόδου (BIC), που έχει από μία θύρα εισόδου και μία εξόδου για κάθε διεπαφή του (IOIF0 και IOIF1). Κάθε στοιχείο μπορεί σε έναν κύκλο του διαύλου να παράγει και να καταναλώνει ταυτόχρονα 16 byte δεδομένων.

Όσον αφορά τη μεταφορά των εντολών, κάθε στοιχείο συνδέεται σε έναν συγκεντρωτή διευθύνσεων (address concentrator – AC), που αναλαμβάνει την ανίχνευση και την πρόληψη συγκρούσεων και εξασφαλίζει τη δίκαιη κατανομή της πρόσβασης στο δίαυλο. Υπάρχουν πολλοί τέτοιοι συγκεντρωτές διευθύνσεων, και όλοι προωθούν τα δεδομένα σε ένα μοναδικό σειριακό σημείο ανάκλασης των εντολών (serial command reflection point), που ονομάζεται AC0. Οι άλλοι συγκεντρωτές διευθύνσεων ονομάζονται AC1, AC2 (υπάρχουν δύο), και AC3. Ο AC0 λαμβάνει και διατάσσει τις εντολές από τα διάφορα στοιχεία, στέλνει σε όλα τα στοιχεία (broadcast) τις εντολές (για snooping), και τέλος συγκεντρώνει την απάντηση και την στέλνει πάλι σε όλα τα στοιχεία. Η απάντηση στην εντολή είναι

το σήμα που στέλνεται στα κατάλληλα στοιχεία να ξεκινήσουν την μεταφορά.

Το δίκτυο για τη μεταφορά των δεδομένων είναι πιο εξεζητημένο: αποτελείται από τέσσερις δακτύλιους, ο καθένας από τους οποίους συνδέει όλα τα στοιχεία. Τα στοιχεία συνδέονται σε κάθε δακτύλιο με την ίδια σειρά. Σε κάθε δακτύλιο οι μεταφορές μπορούν να γίνονται μόνο κατά τη μία κατεύθυνση, αν για παράδειγμα σε έναν δακτύλιο επιτρέπεται να μεταφέρονται δεδομένα από το κεντρικό στοιχείο επεξεργασίας (PPE) στον συνεπεξεργαστή 1 (SPE1), δεν επιτρέπεται να μεταφέρονται δεδομένα και προς την αντίθετη κατεύθυνση. Οι δύο από τους δακτύλιους μεταφέρουν δεδομένα κατά τη φορά των δεικτών του ρολογιού και οι άλλοι δύο κατά την αντίθετη φορά. Κάθε δακτύλιος επιτρέπει να διεξάγονται σε αυτόν ταυτόχρονα έως και τρεις μεταφορές, αρκεί οι διαδρομές τους να μην αλληλοεπικαλύπτονται. Για να ξεκινήσει μια μεταφορά, το στοιχείο που δίνει τη σχετική εντολή πρέπει να ζητήσει την πρόσβαση στο δίαυλο από τον διαιτητή (arbiter) του διαύλου, ο οποίος επεξεργάζεται τις αιτήσεις και αποφασίζει σε ποιόν δακτύλιο θα ανατεθεί κάθε μεταφορά. Πάντα επιλέγεται ένας από τους δύο δακτύλιους που μεταφέρουν δεδομένα στην κατεύθυνση κατά την οποία η απόσταση μεταξύ πηγής και προορισμού είναι ελάχιστη, κι έτσι καμία μεταφορά δεν διανύει ποτέ απόσταση μεγαλύτερη από το μισό του δακτυλίου, δηλαδή 6 βήματα (αφού τα συνδεδεμένα στοιχεία είναι 12). Ο διαιτητής είναι αυτός που φροντίζει επίσης ώστε οι μεταφορές να μην αλληλοεπικαλύπτονται.

2.4.2 Θεωρητικό μέγιστο εύρος ζώνης

Κάθε στοιχείο που είναι συνδεδεμένο στο δίαυλο μπορεί να δέχεται και να λαμβάνει ταυτόχρονα 16 bytes δεδομένων σε κάθε κύκλο (δηλαδή 25.6 Gbytes/s). Το μέγιστο εύρος ζώνης δεδομένων περιορίζεται από το ρυθμό με τον οποίο τα στοιχεία του διαύλου μπορούν να “κατασκοπεύουν” τις διευθύνσεις των αιτήσεων για μεταφορές, που είναι μια διεύθυνση ανά κύκλο του διαδρόμου. Κάθε αίτηση μπορεί να αναφέρεται στη μεταφορά μέχρι 128 byte δεδομένων, έτσι με δεδομένη τη συχνότητα των 1.6 GHz, το μέγιστο εύρος ζώνης είναι 128 bytes * 1.6 GHz = 204.8 Gbytes/s.

Το μέγιστο εύρος ζώνης στην πράξη δεν επιτυγχάνεται πάντα, γιατί εξαρτάται από τους ακόλουθους παράγοντες: τις σχετικές θέσεις της πηγής και του προορισμού, την πιθανότητα μια νέα μεταφορά να συγκρούεται με άλλες που βρίσκονται ήδη σε εξέλιξη, τον αριθμό των επεξεργαστών Cell στο σύστημα, αν οι μεταφορές πραγματοποιούνται μόνο μεταξύ των συνεπεξεργαστών ή μεταξύ των συνεπεξεργαστών και της κεντρικής μνήμης, και από την αποδοτικότητα του διαιτητή του διαύλου.

Μειωμένο εύρος ζώνης έχουμε επομένως στις ακόλουθες περιπτώσεις:

- Όλες οι αιτήσεις έχουν τον ίδιο προορισμό, για παράδειγμα την κεντρική μνήμη, ή την ίδια

τοπική μνήμη. Στην περίπτωση αυτή ο προορισμός αποτελεί στένωση (bottleneck) για το σύστημα.

- Όλες οι μεταφορές εκτελούνται κατά την ίδια φορά, με αποτέλεσμα δύο από τους τέσσερις δακτυλίους να μην χρησιμοποιούνται.
- Οι περισσότερες μεταφορές έχουν μέγεθος μικρότερο από 128 byte (cache line), με αποτέλεσμα να μειώνεται η απόδοση του διαύλου.
- Όλες οι μεταφορές διανύουν απόσταση ίση με το μισό του δακτυλίου, με αποτέλεσμα να εμποδίζουν οποιαδήποτε άλλη μεταφορά στο τμήμα αυτό του δακτυλίου.

Ειδικότερα σχετικά με την σχετική θέση της πηγής και του προορισμού μια μεταφοράς μπορούν να γίνουν οι ακόλουθες παρατηρήσεις: Το κεντρικό στοιχείο επεξεργασίας (PPE) και ο ελεγκτής μνήμης (MIC) βρίσκονται το ένα δίπλα στον άλλο, οπότε οι μεταφορές μεταξύ τους τείνουν να μην επηρεάζουν τις υπόλοιπες μεταφορές. Οι συνεπεξεργαστές στο “βόρειο” μέρος του ολοκληρωμένου αριθμούνται με τους άρτιους αριθμούς 0,2,4,6, ενώ οι συνεπεξεργαστές στο “νότιο” μέρος του ολοκληρωμένου αριθμούνται με τους περιττούς αριθμούς 1,3,5,7. Επομένως, μια μεταφορά μεταξύ ενός συνεπεξεργαστή με άρτιο αριθμό και ενός με περιττό, ενδέχεται να “περνάει” από το κεντρικό στοιχείο επεξεργασίας (PPE) και τον ελεγκτή μνήμης (MIC). Σε γενικές γραμμές είναι πιο αποδοτική η επικοινωνία των συνεπεξεργαστών με άρτιο αριθμό μεταξύ τους, και αυτών με περιττό αριθμό μεταξύ τους. Για το λόγο αυτό, κατά την ανάπτυξη μιας εφαρμογής είναι χρήσιμη η κατασκευή ενός σχεδιαγράμματος στο οποίο θα φαίνονται οι διάφορες μεταφορές και το πώς αλληλεπιδρούν μεταξύ τους, ώστε να αποφεύγονται οι περιπτώσεις που μειώνουν κατά πολύ το μέγιστο εύρος ζώνης.

Κεφάλαιο 3

Προγραμματισμός του επεξεργαστή Cell

3.1 Προγραμματιστικό περιβάλλον

Για τον προγραμματισμό του επεξεργαστή Cell, παρέχεται ένα σύνολο εργαλείων για την ανάπτυξη λογισμικού για την Cell Broadband Engine, το Cell SDK (Software Development Kit). Το Cell SDK περιλαμβάνει τα εργαλεία που είναι απαραίτητα για την ανάπτυξη εφαρμογών για τον Cell. Κάποια από τα πολυάριθμα συστατικά του είναι τα ακόλουθα:

- ο προσομοιωτής της IBM για την Cell Broadband Engine (IBM Full System Simulator)
- system root image που περιλαμβάνει πυρήνα του λειτουργικού συστήματος Linux για χρήση με τον παραπάνω προσομοιωτή
- GNU εργαλεία ανάπτυξης λογισμικού (GNU tools), που περιλαμβάνουν μεταγλωττιστές C και C++, linkers, assemblers και binary utilities τόσο για την κεντρική επεξεργαστική μονάδα (PPU) όσο και για τους συνεπεξεργαστές (SPUs)
- ο μεταγλωττιστής xlc της IBM τόσο για την κεντρική επεξεργαστική μονάδα (PPU) όσο και για τους συνεπεξεργαστές (SPUs)
- βιβλιοθήκες Linux για την ανάπτυξη εφαρμογών για τον Cell (για παράδειγμα SPE Runtime Management Library)
- διάφορα εργαλεία για αποσφαλμάτωση, profiling
- παραδείγματα κώδικα και βιβλιοθηκών, βελτιστοποιημένων για τον επεξεργαστή Cell

Επιπλέον, υπάρχουν διανομές του λειτουργικού συστήματος Linux για πραγματικά συστήματα που χρησιμοποιούν τον επεξεργαστή Cell, για παράδειγμα το Sony Playstation 3. Χρησιμοποιώντας τα παραπάνω εργαλεία σε συνδυασμό με μια τέτοια διανομή, μπορεί να γίνει ανάπτυξη λογισμικού σε πραγματικό υλικό, αντί για τον προσομοιωτή.

3.2 Προγραμματισμός της κεντρικής μονάδας επεξεργασίας

Για να υποστηριχθεί το σύνολο εντολών Vector/SIMD Multimedia Extension, έχουν εισαχθεί κατάλληλες επεκτάσεις στις γλώσσες προγραμματισμού C/C++, στις οποίες περιλαμβάνονται αφενός ειδικοί τύποι διανυσματικών δεδομένων, αφετέρου εντολές για πράξεις σε διανύσματα, οι οποίες έχουν τη μορφή κλήσεων συναρτήσεων υψηλού επιπέδου, όμως στην πράξη πρόκειται για ενσωματωμένες (inline) εντολές assembly. Έτσι παράγεται με τη βοήθεια του μεταγλωττιστή βελτιστοποιημένος κώδικας που εκμεταλλεύεται πλήρως το σύνολο εντολών Vector/SIMD Multimedia Extension, χωρίς ο προγραμματιστής να χρειάζεται να χειριστεί απευθείας τους καταχωρητές, όπως απαιτεί ο προγραμματισμός σε assembly.

3.2.1 Τύποι δεδομένων

Οι νέοι τύποι διανυσματικών δεδομένων που προστίθενται έχουν το πρόθεμα *vector* μπροστά από έναν βασικό τύπο δεδομένων της γλώσσας C, για παράδειγμα κάποιοι από τους νέους τύπους δεδομένων είναι οι εξής: *vector unsigned int, vector signed int, vector float*. Οι νέοι τύποι περιέχονται στους διανυσματικούς καταχωρητές πολυμέσων (Vector Multimedia Registers – VMRs), και μπορεί να αποτελούνται από:

- δεκαέξι τιμές των 8 bit, προσημασμένες ή απρόσημες
- οχτώ τιμές των 16 bit, προσημασμένες ή απρόσημες
- τέσσερις τιμές των 32 bit, προσημασμένες ή απρόσημες
- τέσσερις τιμές κινητής υποδιαστολής απλής ακρίβειας, που αναπαριστώνται σύμφωνα με το πρότυπο IEEE-754

Κάθε ένας από τους νέους τύπους δεδομένων αντιπροσωπεύει ένα διάνυσμα τόσων στοιχείων όσα είναι τα στοιχεία του αντίστοιχου βασικού τύπου της C που χωράνε σε έναν καταχωρητή των 128 bit. Για παράδειγμα, ο τύπος *vector unsigned int* αποτελεί έναν τελεστή των 128 bit που περιέχει τέσσερις προσημασμένους ακεραίους των 32 bit.

3.2.2 Εντολές για πράξεις σε διανύσματα

Οι εντολές χωρίζονται σε τρεις κατηγορίες:

- Ειδικές εντολές (specific intrinsics) – οι εντολές αυτές αντιστοιχίζονται πάντα σε μία ακριβώς

εντολή assembly.

- Γενικές εντολές (generic intrinsics) – οι εντολές αυτές αντιστοιχίζονται είτε σε μία είτε σε περισσότερες εντολές assembly, συναρτήσει του τύπου των παραμέτρων εισόδου.
- Κατηγορηματικές εντολές (predicates intrinsics) – οι εντολές αυτές συγκρίνουν τιμές και επιστρέφουν έναν ακέραιο που μπορεί να χρησιμοποιηθεί είτε απευθείας ως τιμή είτε ως συνθήκη για κάποια εντολή διακλάδωσης.

Ένα παράδειγμα εντολής για πράξεις σε διανύσματα είναι η εντολή:

$$d = vec\_add(a,b)$$

με την οποία τα διανύσματα a και b προστίθενται και το αποτέλεσμα αποθηκεύεται στο διάνυσμα d. Η εντολή αυτή αντιστοιχίζεται σε κατάλληλες εντολές assembly ανάλογα με τον τύπο των διανυσμάτων a και b. Άλλο παράδειγμα εντολής είναι η:

$$d = vec\_all\_eq(a,b)$$

με την οποία ελέγχεται αν όλα τα στοιχεία των διανυσμάτων a και b είναι ίσα.

3.3 SPE Runtime Management Library (Βιβλιοθήκη διαχείρισης συνεπεξεργαστών κατά το χρόνο εκτέλεσης)

Η SPE Runtime Management Library (libspe) αποτελεί την τυποποιημένη διεπαφή προγραμματισμού εφαρμογών, χαμηλού επιπέδου, μέσω της οποίας οι εφαρμογές μπορούν να έχουν πρόσβαση στους συνεπεξεργαστές (SPEs) της Cell Broadband Engine. Η βιβλιοθήκη αυτή παρέχει μία διεπαφή προγραμματισμού εφαρμογών, η οποία είναι ανεξάρτητη από το υποκείμενο λειτουργικό σύστημα και τις μεθόδους που χρησιμοποιεί για τη διαχείριση των συνεπεξεργαστών.

Οι εφαρμογές δεν έχουν άμεσο έλεγχο των φυσικών πόρων των συνεπεξεργαστών του συστήματος. Υπεύθυνο για τη διαχείριση των πόρων αυτών είναι το λειτουργικό σύστημα. Αντί αυτού, οι εφαρμογές έχουν πρόσβαση σε λογικές δομές (software constructs) που ονομάζονται SPE contexts. Τα SPE contexts είναι η βασική δομή δεδομένων πάνω στην οποία λειτουργεί η SPE Runtime Management Library. Αποτελούν την λογική αναπαράσταση ενός συνεπεξεργαστή και κρατούν όλες τις πληροφορίες σχετικά με έναν “λογικό συνεπεξεργαστή” που χρησιμοποιεί μια εφαρμογή. Οι εφαρμογές δεν τροποποιούν τη δομή απευθείας, παρά μόνο χρησιμοποιούν δείκτες σε δομές τέτοιου τύπου ως αναγνωριστικά για τον “λογικό συνεπεξεργαστή” που χειρίζονται, μέσω κλήσεων στις συναρτήσεις της βιβλιοθήκης, στις οποίες περνούν τους δείκτες αυτούς ως παράμετρο. Το λειτουργικό σύστημα είναι αυτό που αναλαμβάνει να αντιστοιχίσει τα SPE contexts από όλες τις τρέχουσες

διεργασίες στους φυσικούς πόρους του συστήματος, δηλαδή στους φυσικούς συνεπεξεργαστές (SPEs), έτσι ώστε τα αντίστοιχα προγράμματα να εκτελεσθούν. Η αντιστοίχιση αυτή γίνεται σύμφωνα με τις πολιτικές και τις προτεραιότητες χρονοδρομολόγησης που αντιστοιχούν στα προς εκτέλεση SPE contexts.

Επιπρόσθετα, η SPE Runtime Management Library παρέχει τα μέσα για επικοινωνία και μεταφορά δεδομένων μεταξύ της κεντρικής επεξεργαστικής μονάδας (PPE) και των συνεπεξεργαστών (SPEs).

Η βασική μορφή μιας απλής εφαρμογής που χρησιμοποιεί έναν συνεπεξεργαστή είναι η ακόλουθη:

α) Δημιουργία ενός SPE context

β) Φόρτωση ενός εκτελέσιμου αρχείου στην τοπική μνήμη (local store) του SPE context

γ) Εκτέλεση του SPE context – εδώ μεταφέρεται ο έλεγχος στο λειτουργικό σύστημα, ζητώντας την πραγματική δρομολόγηση του context σε έναν φυσικό συνεπεξεργαστή (SPE) του συστήματος

δ) Καταστροφή του SPE context

Το βήμα γ αποτελεί μια σύγχρονη κλήση προς το λειτουργικό σύστημα. Η εφαρμογή που κάνει την κλήση θα σταματήσει τη λειτουργία της (θα "μπλοκάρει") μέχρι να σταματήσει η εκτέλεση στον συνεπεξεργαστή (είτε ομαλά, είτε λόγω κάποιου σφάλματος), οπότε το λειτουργικό σύστημα επιστρέφει από την κλήση συστήματος που προκάλεσε την εκτέλεση του context στον συνεπεξεργαστή.

Πολλές εφαρμογές απαιτούν την ταυτόχρονη χρήση περισσότερων από ενός συνεπεξεργαστές. Σε αυτή την περίπτωση, είναι απαραίτητη η δημιουργία τόσων νημάτων όσοι είναι οι συνεπεξεργαστές που προτίθεται η εφαρμογή να χρησιμοποιήσει, ώστε καθένα από τα νήματα αυτά να ζητά την εκτέλεση ενός μοναδικού SPE context κάθε φορά. Είναι κοινή πρακτική σε κάθε εφαρμογή που απαιτεί N συνεπεξεργαστές να υπάρχει ένα κυρίως νήμα, το οποίο δημιουργεί και συντονίζει τα υπόλοιπα N νήματα, καθένα από τα οποία αναλαμβάνει την εκτέλεση ενός SPE context, κατά το ακόλουθο σχήμα:

α) Δημιουργία N SPE contexts

β) Φόρτωση του κατάλληλου εκτελέσιμου αρχείου στην τοπική μνήμη (local store) καθενός εκ των SPE contexts

γ) Δημιουργία N νημάτων, σε καθένα από τα οποία εκτελείται (όπως και στην απλή περίπτωση της εφαρμογής με έναν συνεπεξεργαστή) το αντίστοιχο SPE context, και τερματισμός των νημάτων όταν η εκτέλεση ολοκληρωθεί

δ) Αναμονή για την ολοκλήρωση της εκτέλεσης των νημάτων

ε) Καταστροφή όλων των SPE contexts

Τα SPE contexts μπορούν να αποτελούν μία ομάδα (gang) με κοινές ιδιότητες που αφορούν στη δρομολόγηση και την εκτέλεσή τους. Η βασική δομή δεδομένων που χρησιμοποιεί η SPE Runtime Management Library για την ομαδοποίηση των SPE contexts είναι τα gang contexts. Αυτές οι δομές, όπως και τα SPE contexts, δεν πρέπει να τροποποιούνται απευθείας από τις εφαρμογές, οι οποίες πρέπει απλά να διατηρούν δείκτες σε δομές τέτοιου τύπου, τους οποίους να περνούν ως παραμέτρους στις συναρτήσεις της βιβλιοθήκης.

Για να παρέχει τις παραπάνω δυνατότητες, η SPE Runtime Management Library παρέχει διάφορα σύνολα συναρτήσεων που προορίζονται να καλούνται από την κεντρική επεξεργαστική μονάδα (PPE):

- Συναρτήσεις για τη δημιουργία και την καταστροφή των SPE contexts και SPE gang contexts
- Συναρτήσεις για τη φόρτωση των εκτελέσιμων αρχείων στην τοπική μνήμη των SPE contexts
- Συναρτήσεις για την έναρξη της εκτέλεσης ενός προγράμματος σε έναν συνεπεξεργαστή και για την ανάκτηση πληροφοριών σχετικών με το λόγο διακοπής της εκτέλεσής τους
- Συναρτήσεις για τη λήψη ασύγχρονων γεγονότων (SPE events) που παράγει ένας συνεπεξεργαστής
- Συναρτήσεις για την πρόσβαση στις διάφορες υπηρεσίες που παρέχει ο ελεγκτής πρόσβασης στη μνήμη, οι οποίες περιλαμβάνουν μεταξύ άλλων τα Mailboxes και την υπηρεσία ειδοποίησης μέσω σημάτων (signal notification facility)
- Συναρτήσεις για την άμεση πρόσβαση στην τοπική μνήμη (local store) και στους καταχωρητές που ελέγχουν διάφορες υπηρεσίες του ελεγκτή πρόσβασης στη μνήμη (MFC) των συνεπεξεργαστών
- Συναρτήσεις για την δήλωση κλήσεων βιβλιοθήκης για τους συνεπεξεργαστές υποβοηθούμενων από την κεντρική επεξεργαστική μονάδα

Στη συνέχεια της παραγράφου εξηγούνται πιο αναλυτικά ορισμένες από τις συναρτήσεις της SPE Runtime Management Library.

3.3.1 Δημιουργία και καταστροφή των SPE contexts και SPE gang contexts

Η δημιουργία ενός SPE gang context γίνεται μέσω κλήσης στη συνάρτηση:

spe_gang_context_ptr_t spe_gang_context_create(unsigned int flags)

όπου flags είναι σημαίες που επηρεάζουν τη δημιουργία του SPE gang context. Η καταστροφή του SPE gang context γίνεται με την ακόλουθη συνάρτηση:

int spe_gang_context_destroy(spe_gang_context_ptr_tgang)

Οι αντίστοιχες συναρτήσεις για τη δημιουργία και καταστροφή ενός SPE context είναι οι εξής:

spe_context_ptr_t spe_context_create(unsignedint flags, spe_gang_context_ptr_tgang)

(όπου flags είναι σημαίες που επηρεάζουν τη δημιουργία του SPE context και gang το SPE gang context στο οποίο θα ανήκει το SPE context)

int spe_context_destroy(spe_context_ptr_t spe)

3.3.2 Χειρισμός των SPE Program Images

Η συνάρτηση που ακολουθεί δέχεται ως όρισμα το όνομα ενός εκτελέσιμου αρχείου τύπου SPE ELF και το απεικονίζει στην μνήμη του συστήματος, και επιστρέφει έναν δείκτη που μπορεί να χρησιμοποιηθεί στη συνέχεια για τη φόρτωση του προγράμματος στην τοπική μνήμη ενός συνεπεξεργαστή:

*spe_program_handle_t * spe_image_open(constchar *filename)*

Για το κλείσιμο του εκτελέσιμου αρχείου χρησιμοποιείται η εξής συνάρτηση:

*int spe_image_close(spe_program_handle_t *program)*

Τέλος, για τη φόρτωσή του εκτελέσιμου αρχείου στην τοπική μνήμη ενός συνεπεξεργαστή, χρησιμοποιείται η ακόλουθη συνάρτηση:

*int spe_program_load(spe_context_ptr_t spe, spe_program_handle_t *program)*

3.3.3 Έλεγχος της εκτέλεσης των SPE contexts

Η συνάρτηση που ακολουθεί ζητά την εκτέλεση ενός SPE context σε έναν φυσικό επεξεργαστή του συστήματος:

*int spe_context_run(spe_context_ptr_t spe, unsigned int *entry, unsigned int runflags, void *argp, void *envp, spe_stop_info_t *stopinfo)*

Πριν την κλήση της πρέπει να έχει προηγηθεί κλήση της spe_program_load. Η παραπάνω συνάρτηση είναι blocking, επομένως το νήμα από το οποίο καλείται σταματά την εκτέλεσή του και περιμένει μέχρι η εκτέλεση στον συνεπεξεργαστή να σταματήσει είτε λόγω κανονικού τερματισμού του προγράμματος, είτε λόγω κάποιου σφάλματος στην εκτέλεση, είτε επειδή η μονάδα επεξεργασίας του συνεπεξεργαστή σταμάτησε την εκτέλεση και έστειλε κάποιο σήμα. *spe* είναι το SPE context του

οποίου η εκτέλεση ζητείται, *entry* είναι η θέση από την οποία ξεκινά η εκτέλεση του προγράμματος (συνήθως δίνεται η τιμή SPE_DEFAULT_ENTRY για ανάκτηση του προκαθορισμένου σημείου εισόδου από το εκτελέσιμο αρχείο), *runflags* είναι διάφορες σημαίες που ρυθμίζουν ειδική συμπεριφορά κατά τη εκτέλεση (ο για προκαθορισμένη συμπεριφορά), *argp* είναι δείκτης σε δεδομένα σχετικά με την εφαρμογή, *envp* είναι ένας δείκτης σε δεδομένα σχετικά με το περιβάλλον, και τέλος *stopinfo* είναι ένας δείκτης σε μια δομή που περιέχει μετά την ολοκλήρωση της *spe_context_run()* λεπτομέρειες για τον τερματισμό της εκτέλεσης του προγράμματος στο συνεπεξεργαστή, περισσότερες από αυτές που δίνει ο κωδικός επιστροφής της συνάρτησης .

3.3.4 Mailboxes

Για το χειρισμό των SPE Mailboxes παρέχονται οι ακόλουθες συναρτήσεις:

*int spe_out_mbox_read(spe_context_ptr_t spe, unsigned int *mbox_data, int count)*

Με την παραπάνω συνάρτηση διαβάζονται μέχρι και *count* μηνύματα από το SPE Outbound Mailbox. Τα δεδομένα γράφονται στις θέσεις μνήμης στις οποίες δείχνει ο δείκτης *mbox_data*. Η κλήση είναι non-blocking. Στην περίπτωση που υπάρχουν λιγότερα από *count* διαθέσιμα μηνύματα, διαβάζονται όσα είναι διαθέσιμα.

spe_out_mbox_status(spe_context_ptr_t spe)

Η παραπάνω συνάρτηση επιστρέφει τον αριθμό των μηνυμάτων που είναι διαθέσιμα προς ανάγνωση στο SPE Outbound Mailbox.

Αντίστοιχες συναρτήσεις υπάρχουν και για το SPE Outbound Interrupt Mailbox.

*spe_in_mbox_write(spe_context_ptr_t spe, unsigned int *mbox_data, int count, unsigned int behavior)*

Η παραπάνω συνάρτηση γράφει στο SPE Inbound Mailbox μέχρι και *count* μηνύματα που περιέχονται στις θέσεις μνήμης όπου δείχνει ο δείκτης *mbox_data*. Η συμπεριφορά της μπορεί να είναι είτε blocking είτε non-blocking ανάλογα με την τιμή της παραμέτρου *behavior*. Όταν η τιμή της είναι ίση με SPE_MBOX_ALL_BLOCKING, η κλήση μπλοκάρει μέχρι να ολοκληρωθεί η εγγραφή όλων των μηνυμάτων, όταν είναι ίση με SPE_MBOX_ANY_BLOCKING, η κλήση μπλοκάρει μέχρι να ολοκληρωθεί η εγγραφή τουλάχιστον ενός μηνύματος, ενώ όταν είναι ίση με SPE_MBOX_ANY_NONBLOCKING, γράφονται όσα μηνύματα είναι δυνατόν, και η κλήση τερματίζει χωρίς να μπλοκάρει.

spe_in_mbox_status(spe_context_ptr_t spe)

Η παραπάνω συνάρτηση επιστρέφει την κατάσταση του SPE Inbound Mailbox. Επιστρέφει 0 όταν αυτό είναι γεμάτο, ενώ μη μηδενική τιμή επιστροφής δηλώνει τον αριθμό των κενών θέσεων σε αυτό.

3.3.5 Λειτουργία ειδοποίησης μέσω σήματος (Signal Notification Facility)

Η αποστολή ενός σήματος σε έναν συνεπεξεργαστή μέσω εγγραφής δεδομένων σε έναν από τους δύο σχετικούς καταχωρητές (signal notification registers) γίνεται με την ακόλουθη συνάρτηση:

int spe_signal_write(spe_context_ptr_t spe, unsigned int signal_reg, unsigned int data)

όπου *spe* είναι το SPE context στο οποίο πρέπει να σταλεί το σήμα, η παράμετρος *signal_reg* δηλώνει ποιος από τους δύο καταχωρητές πρέπει να χρησιμοποιηθεί (παίρνει μία από τις τιμές SPE_SIG_NOTIFY_REG_1, SPE_SIG_NOTIFY_REG_2), και data είναι τα προς αποστολή δεδομένα.

3.3.6 Άμεση πρόσβαση στην τοπική μνήμη και τους καταχωρητές των συνεπεξεργαστών

Η συνάρτηση που ακολουθεί επιστρέφει το μέγεθος (σε πλήθος bytes) της τοπικής μνήμης του συνεπεξεργαστή στον οποίο εκτελείται το SPE context:

int spe_ls_size_get(spe_context_ptr_t spe)

Η επόμενη συνάρτηση απεικονίζει την τοπική μνήμη του συνεπεξεργαστή στον οποίο εκτελείται το SPE context στο χώρο διευθύνσεων του τρέχοντος νήματος και επιστρέφει έναν δείκτη στην πρώτη διεύθυνση της περιοχής μνήμης όπου απεικονίστηκε η τοπική μνήμη.

*void * spe_ls_area_get(spe_context_ptr_t spe)*

Η επόμενη συνάρτηση απεικονίζει ένα σύνολο από πόρους (π.χ. καταχωρητές) του συνεπεξεργαστή στον οποίο εκτελείται το SPE context στο χώρο διευθύνσεων του τρέχοντος νήματος και επιστρέφει έναν δείκτη στην πρώτη διεύθυνση της περιοχής μνήμης όπου έγινε η απεικόνιση.

*void * spe_ps_area_get(spe_context_ptr_t spe, enum ps_area area)*

Η παράμετρος *area* καθορίζει ποιοι πόροι θα απεικονιστούν. Για παράδειγμα, η τιμή SPE_SIG_NOTIFY_1_AREA έχει ως αποτέλεσμα να απεικονιστεί στη μνήμη ο πρώτος καταχωρητής για την ειδοποίηση μέσω σημάτων (signal notification register).

3.4 Προγραμματισμός των συνεπεξεργαστών

Όπως για το κεντρικό στοιχείο επεξεργασίας, έτσι και για τους συνεπεξεργαστές έχουν εισαχθεί κατάλληλες επεκτάσεις στις γλώσσες προγραμματισμού C/C++, στις οποίες περιλαμβάνονται νέοι τύποι διανυσματικών δεδομένων και εντολές για πράξεις σε διανύσματα, οι οποίες έχουν τη μορφή κλήσεων συναρτήσεων υψηλού επιπέδου, όμως στην πράξη πρόκειται για ενσωματωμένες (inline) εντολές assembly. Επίσης υπάρχουν ειδικές μακροεντολές για την έκδοση εντολών άμεσης πρόσβασης στη μνήμη (DMA). Τα παραπάνω περιγράφονται αναλυτικότερα στις παραγράφους που ακολουθούν.

3.4.1 Διανυσματικοί τύποι δεδομένων

Το μοντέλο προγραμματισμού των συνεπεξεργαστών ορίζει τους τύπους δεδομένων που παρατίθενται στην παρακάτω λίστα. Όλοι οι τύποι έχουν μέγεθος 128 bit:

- vector unsigned char – δεκαέξι απρόσημοι χαρακτήρες των 8 bit
- vector signed char – δεκαέξι προσημασμένοι χαρακτήρες των 8 bit
- vector unsigned short – οχτώ απρόσημες μισές λέξεις των 16 bit
- vector signed short – οχτώ προσημασμένες μισές λέξεις των 16 bit
- vector unsigned int – τέσσερις απρόσημες λέξεις των 32 bit
- vector signed int – τέσσερις προσημασμένες λέξεις των 32 bit
- vector unsigned long long – δύο απρόσημες λέξεις των 64 bit
- vector signed long long – δύο προσημασμένες λέξεις των 64 bit
- vector float – τέσσερις αριθμοί κινητής υποδιαστολής απλής ακρίβειας (32 bit)
- vector double – δύο αριθμοί κινητής υποδιαστολής διπλής ακρίβειας (64 bit)
- qword – τετραπλή λέξη (16 byte)

3.4.2 Επεκτάσεις των γλωσσών C/C++

Για να διευκολύνεται η εργασία του προγραμματιστή κατά την δημιουργία εφαρμογών για τους συνεπεξεργαστές, χωρίς όμως ταυτόχρονα να χάνεται η δυνατότητα εκμετάλλευσης του πλούσιου νέου συνόλου εντολών που χρησιμοποιούν, έχουν εισαχθεί κατάλληλες επεκτάσεις στις γλώσσες C/C++ υπό την μορφή συναρτήσεων, οι οποίες αντιστοιχούν είτε σε μία είτε σε περισσότερες εντολές

assembly. Οι επεκτάσεις αυτές ορίζονται στο αρχείο επικεφαλίδας spu_intrinsics.h. Ένα παράδειγμα μιας τέτοιας εντολής είναι η:

$$t = spu\_add(a, b)$$

με την οποία εκτελείται πρόσθεση των επιμέρους στοιχείων των διανυσμάτων a και b. Η εντολή αυτή αντιστοιχίζεται από τον μεταγλωττιστή στην κατάλληλη εντολή πράξης σε διανύσματα, ανάλογα με τον τύπο των διανυσμάτων (π.χ. απρόσημοι ακέραιοι/αριθμοί κινητής υποδιαστολής). Το πλήρες σύνολο αυτών των εντολών μπορεί να αναζητηθεί στο σχετικό εγχειρίδιο (C/C++ Language Extensions for Cell Broadband Engine Architecture).

3.4.3 Εντολές προς τον ελεγκτή πρόσβασης στη μνήμη και αντίστοιχες μακροεντολές

Ο ελεγκτής πρόσβασης στη μνήμη υποστηρίζει ένα σύνολο εντολών, οι οποίες παρέχουν στον συνεπεξεργαστή το μηχανισμό μέσω του οποίου επικοινωνεί με την κεντρική μνήμη και διατηρεί το συγχρονισμό του ως προς τους υπόλοιπους επεξεργαστές και συσκευές του συστήματος. Οι εντολές αυτές μπορούν να εκδοθούν είτε από κώδικα που εκτελείται στον ίδιο τον συνεπεξεργαστή, μέσω μιας ακολουθίας εντολών εγγραφής και ανάγνωσης από τα σχετικά κανάλια του ελεγκτή πρόσβασης στη μνήμη, είτε από κώδικα που εκτελείται στο κεντρικό στοιχείο επεξεργασίας ή σε άλλες συσκευές, μέσω μιας ακολουθίας εντολών φόρτωσης και αποθήκευσης από τους αντίστοιχους καταχωρητές που είναι απεικονισμένοι στη μνήμη. Ανάλογα με το ποιος τις εξέδωσε, οι εντολές τοποθετούνται σε δύο ανεξάρτητες μεταξύ τους ουρές αναμονής, την ουρά αναμονής εντολών της μονάδας επεξεργασίας του συνεπεξεργαστή (MFC SPU Command Queue) και την ουρά αναμονής απομακρυσμένων εντολών (MFC Proxy Command Queue).

Οι εντολές προς τον ελεγκτή πρόσβασης στη μνήμη που μεταφέρουν δεδομένα ονομάζονται συνήθως εντολές άμεσης πρόσβασης στη μνήμη (DMA commands). Η φορά μεταφοράς των δεδομένων πάντα ορίζεται ως προς τον συνεπεξεργαστή, έτσι οι εντολές που μεταφέρουν δεδομένα προς την τοπική μνήμη ενός συνεπεξεργαστή ονομάζονται εντολές τύπου get, ενώ οι εντολές που μεταφέρουν δεδομένα από την τοπική μνήμη ενός συνεπεξεργαστή προς τον κύριο χώρο διευθύνσεων ονομάζονται εντολές τύπου put.

Οι εντολές τύπου get που υποστηρίζονται είναι οι εξής: get, gets, getf, getb, getfs, getbs, getl, getlf, getlb, ενώ οι εντολές τύπου put που υποστηρίζονται είναι οι: put, puts, putf, putb, putfs, putbs, putl, putlf, putlb. Το επίθεμα s δηλώνει ότι αφού ολοκληρωθεί η εντολή, πρέπει να συνεχιστεί η εκτέλεση του προγράμματος στο συνεπεξεργαστή (οι εντολές με αυτό το επίθεμα δεν μπορούν να εκδοθούν τοπικά από τον ίδιο τον συνεπεξεργαστή, παρά μόνο από το κεντρικό στοιχείο επεξεργασίας ή άλλη

συσκευή του συστήματος). Τα επιθέματα f και b παρέχουν έναν τρόπο για την επιβολή διάταξης στις εντολές, και θα εξηγηθούν αναλυτικότερα παρακάτω. Το επίθεμα l δηλώνει ότι η εντολή είναι τύπου λίστας. Η λίστα μπορεί να περιέχει μέχρι 2048 στοιχεία, το καθένα από τα οποία μπορεί να μεταφέρει μέχρι 16KB δεδομένων. Οι εντολές λίστας μπορούν να εκδοθούν μόνο τοπικά, από τον ίδιο το συνεπεξεργαστή. Εκτός από τις εντολές μεταφοράς δεδομένων υπάρχουν τέσσερις ατομικές εντολές (getllar, putllc, putlluc, putqlluc), οι οποίες χρησιμοποιούνται για ατομική πρόσβαση σε θέσεις μνήμης, τρεις εντολές για αποστολή σημάτων (sndsig, sndsigb, sndsigf), οι οποίες χρησιμοποιούνται για την εγγραφή δεδομένων στους καταχωρητές ειδοποίησης μέσω σημάτων (signal notification registers) των συνεπεξεργαστών, και τρεις εντολές συγχρονισμού (barrier, mfceieio, mfcsync), οι οποίες ρυθμίζουν τη διάταξη των εντολών είτε σε έναν μόνο συνεπεξεργαστή είτε στο επίπεδο του συστήματος.

Σε όλες τις παραπάνω εντολές, πλην των getllar, putllc, putlluc, αντιστοιχίζεται μια ετικέτα (tag). Μέσω της ετικέτας αυτής μπορεί να ελεγχθεί αν έχει ολοκληρωθεί η εκτέλεση των εντολών που έχουν αυτή την ετικέτα και που ανήκουν στην ίδια ουρά αναμονής, με μία μοναδική εντολή. Έτσι το λογισμικό μπορεί να ελέγχει ή να περιμένει την ολοκλήρωση διαφόρων εντολών ομαδοποιημένων με βάση τις ετικέτες τους. Επίσης, οι εντολές που ανήκουν στην ίδια ομάδα μπορούν να συγχρονιστούν μέσω ενός barrier ή ενός fence, που δηλώνεται προσθέτοντας τα επιθέματα b και f στην εντολή, αντίστοιχα. Η εκτέλεση μιας εντολής με fence αναστέλλεται μέχρι να ολοκληρωθούν όλες οι εντολές με την ίδια ετικέτα που βρίσκονται στην ίδια ουρά και έχουν εκδοθεί πριν από τη συγκεκριμένη εντολή. Στην περίπτωση μιας εντολής με barrier, επιβάλλεται αναστολή της εντολής και όλων όσων εκδίδονται μετά από αυτήν στην ίδια ουρά αναμονής και με την ίδια ετικέτα, μέχρι να ολοκληρωθούν όλες οι εντολές με την ίδια ετικέτα που βρίσκονται στην ίδια ουρά και έχουν εκδοθεί πριν από τη συγκεκριμένη εντολή.

Για να εκδοθεί μια εντολή άμεσης πρόσβασης στη μνήμη τοπικά από τον συνεπεξεργαστή, το λογισμικό εκτελεί τις ακόλουθες εντολές εγγραφής στα κανάλια του ελεγκτή πρόσβασης στη μνήμη:

- Εγγραφή της διεύθυνσης τοπικής μνήμης στο κανάλι MFC_LSA
- Εγγραφή του άνω μισού της ενεργού διεύθυνσης στο κανάλι MFC_EAH
- Εγγραφή του κάτω μισού της ενεργού διεύθυνσης στο κανάλι MFC_EAL
- Εγγραφή του μεγέθους της μεταφοράς στο κανάλι MFC_Size
- Εγγραφή της ετικέτας στο κανάλι MFC_TagID
- Εγγραφή του αναγνωριστικού κλάσης (class ID) και του κωδικού της εντολής (command opcode) στο κανάλι MFC_Cmd

Για τη διευκόλυνση του προγραμματιστή και την αποφυγή όλων των παραπάνω εγγραφών στα κανάλια, ορίζονται στο αρχείο επικεφαλίδας spu_mfcio.h διάφορες μακροεντολές που διευκολύνουν την έκδοση των εντολών άμεσης πρόσβασης στη μνήμη. Για παράδειγμα, η έκδοση μιας εντολής τύπου put έχει τη μορφή:

mfc_put(ls, ea, size, tag, tid,rid)

ενώ με την ακολουθία εντολών:

mfc_write_tag_mask(mask)

mfc_read_tag_status_all()

είναι δυνατή η αναμονή μέχρι να ολοκληρωθούν όλες οι εντολές με ετικέτες που καθορίζονται από την επιλεγμένη μάσκα.

3.5 Σύστημα αρχείων spufs (SPU file system)

Το σύστημα αρχείων spufs είναι ένα εικονικό σύστημα αρχείων μέσω του οποίου μας παρέχονται πληροφορίες για τα SPE contexts καθώς επίσης και για τους συνεπεξεργαστές στους οποίους το καθένα από αυτά βρίσκεται σε εκτέλεση. Το σύνηθες mount point του spufs είναι ο φάκελος /spu. Κάτω από αυτόν (ή όποιον άλλο αποτελεί τη ρίζα του spufs, σε περίπτωση που δεν είναι αυτός) βρίσκονται διάφοροι φάκελοι που αντιστοιχούν στα SPE gang contexts, και μέσα σε καθέναν από αυτούς άλλοι φάκελοι που αντιστοιχούν στα SPE contexts που ανήκουν στο SPE gang context του parent directory. Ο κάθε ένας από τους φακέλους που αντιστοιχεί σε ένα SPE context περιέχει έναν αριθμό προκαθορισμένων αρχείων και φακέλων, μέσω των οποίων μας παρέχονται οι πληροφορίες που παρέχει το σύστημα αρχείων. Ένα αρχείο που παρουσιάζει ιδιαίτερο ενδιαφέρον είναι το phys-id, τα περιεχόμενα του οποίου είναι απλά ένας αριθμός, ο αριθμός του συνεπεξεργαστή στον οποίον εκτελείται το SPE context. Με τη βοήθεια αυτής της πληροφορίας η συνάρτηση που ακολουθεί και δέχεται ως όρισμα έναν πίνακα από δείκτες σε SPE contexts, έχει τη δυνατότητα να ταξινομεί τους δείκτες αυτούς ώστε η θέση στην οποία βρίσκεται ένας δείκτης σε ένα SPE context στον πίνακα να αντιστοιχεί στον αριθμό του συνεπεξεργαστή στον οποίο αυτό εκτελείται:

```
int spe_get_physical_ids(spe_gang_context_ptr_t gang,
                                                spe_context_ptr_t speid[]) {
        int physid[SPE_THREADS];
        char str[128];
        struct dirent **files;
        int i, k, j, num_files, l, lstr;
        char folder[128];
```

```
        spe_context_ptr_t tmp[MAX_SPES];

        num_files = scandir("/spu/", &files, 0, alphasort);
        /* εύρεση του φακέλου που αντιστοιχεί στο gang context */
        sprintf(str, "%ld", (long) gang);
        lstr = strlen(str);
        for (i=0;i<num_files;++i) {
                l = strlen(files[i]->d_name);
                if (l>=lstr && !strcmp(files[i]->d_name+l-lstr, str)) {
                        sprintf(folder, "/spu/%s/",files[i]->d_name);
                        free(files);
                        num_files = scandir(folder, &files, 0, alphasort);
                        break;
                }
        }
        /* εύρεση της ζητούμενης πληροφορίας για κάθε ένα από τα contexts
         * και αποθήκευσή της στον πίνακα physid (πίνακας με στοιχεία int) */
        for(j=0; j<SPE_THREADS; j++) {
                sprintf(str, "%ld", (long)speid[j]);
                lstr = strlen(str);
                for(k=0; k<num_files; k++) {
                        l = strlen(files[k]->d_name);
                        if (l>=lstr && !strcmp(files[k]->d_name+l-lstr, str)) {
                                sprintf(str, "%s%s/phys-id", folder,
                                                        files[k]->d_name);
                                FILE *f = fopen(str, "r");
                                fgets(str, 128, f);
                                sscanf(str+2, "%x", &physid[j]);
                                fclose(f);
                        }
                }
        }
        free(files);

        /* αντιγραφή των ids των spe contexts στον πίνακα tmp */
        for(j=0; j<MAX_SPES; ++j) {
                tmp[j] = speid[j];
                speid[j] = NULL; /* έτσι ώστε να μείνουν ίσες με NULL οι θέσεις του
                                  * πίνακα με δείκτη που αντιστοιχεί σε μη ενεργό spe */
        }
        /* αναδιάταξη των ids των contexts με βάση τα physical ids των spes */
        for(j=0; j<SPE_THREADS; ++j) {
                speid[physid[j]] = tmp[j];
        }

        return 0;
}
```

3.6 SPU Events

Οι συνεπεξεργαστές διαθέτουν έναν μηχανισμό ειδοποίησης για ασύγχρονες αλλαγές που συμβαίνουν σε διάφορους πόρους του επεξεργαστή Cell. Οι ασύγχρονες αυτές αλλαγές ονομάζονται Events. Παραδείγματα SPU Events αποτελούν η άφιξη ενός μηνύματος από ένα άλλο στοιχείο επεξεργασίας μέσω των Mailboxes ή μέσω ενός από τους Signal Notification Registers. Το λογισμικό που εκτελείται στην κεντρική μονάδα επεξεργασίας των συνεπεξεργαστών μπορεί να παρακολουθεί και να χειρίζεται τα Events είτε σύγχρονα (με polling ή blocking) είτε ασύγχρονα (ενεργοποιώντας τις διακοπές που σχετίζονται με τα Events και υλοποιώντας κατάλληλη συνάρτηση χειρισμού των διακοπών).

Κάθε συνεπεξεργαστής έχει τη δυνατότητα να ενεργοποιεί και να απενεργοποιεί τα SPU Events μέσω μιας μάσκας, να περιμένει να συμβεί κάποιο SPU Event και να ελέγχει αν έχει ήδη συμβεί κάποιο SPU Event. Η αλληλεπίδραση της κεντρικής μονάδας επεξεργασίας με το μηχανισμό των SPU Events γίνεται μέσω των ακόλουθων τεσσάρων καναλιών:

- SPU Read Event Status Channel
- SPU Write Event Mask Channel
- SPU Read Event Mask Channel
- SPU Write Event Acknowledgment Channel

Συγκεκριμένα το SPU Read Event Status Channel περιέχει πληροφορίες για την κατάσταση όλων των Events που είναι ενεργοποιημένα, ανάλογα με τη μάσκα που περιέχεται στο SPU Write Event Mask Channel. Το SPU Write Event Acknowledgment Channel χρησιμοποιείται για να δηλώνεται ότι έχει ολοκληρωθεί ο χειρισμός κάποιου Event και να αρχικοποιείται ξανά η κατάστασή του.

Η μονάδα επεξεργασίας του συνεπεξεργαστή (SPU) μπορεί να επιλέξει να χειρίζεται τα Events σύγχρονα είτε ελέγχοντας σε έναν βρόχο την τιμή (channel count) του SPU Read Event Status Channel (polling), είτε εκτελώντας απευθείας μια εντολή ανάγνωσης από το κανάλι αυτό, με το ενδεχόμενο σταματήματος (stalling) της εκτέλεσης του λογισμικού μέχρι να υπάρξει διαθέσιμο προς χειρισμό Event. Στην περίπτωση επιλεγεί ο ασύγχρονος χειρισμός των Events, πρέπει να ενεργοποιηθούν οι διακοπές, αφού έχει δημιουργηθεί η κατάλληλη συνάρτηση χειρισμού τους (interrupt handler). Η μονάδα επεξεργασίας του συνεπεξεργαστή (SPU) υποστηρίζει μία μοναδική συνάρτηση χειρισμού των διακοπών, της οποίας το σημείο εισόδου είναι προκαθορισμένο και αντιστοιχεί στη διεύθυνση 0 της τοπικής μνήμης. Όταν ενεργοποιηθεί κάποιο Event προκαλώντας έτσι διακοπή, η SPU εκτελεί εντολή διακλάδωσης στη διεύθυνση 0 και απενεργοποιεί τις διακοπές, αποθηκεύοντας ταυτόχρονα σε κατάλληλο καταχωρητή τη διεύθυνση της επόμενης εντολής, ώστε να χρησιμοποιηθεί κατά την

επιστροφή από τη συνάρτηση χειρισμού των διακοπών. Για να είναι δυνατός ο χειρισμός πολλών διαφορετικών Events συγχρόνως, η λειτουργία χειρισμού τους διαχωρίζεται σε δύο συναρτήσεις, μια πρώτου επιπέδου (κοινή για όλα τα Events) και μία δεύτερου επιπέδου, η οποία διαφοροποιείται ανάλογα με το συγκεκριμένο Event. Με τον τρόπο αυτό, στη συνάρτηση πρώτου επιπέδου χειρισμού των Events ελέγχεται συγκεκριμένα ποια Events έχουν ενεργοποιηθεί, και ο έλεγχος περνάει στις κατάλληλες συναρτήσεις δεύτερου επιπέδου χειρισμού των συγκεκριμένων Events. Κώδικας για την συνάρτηση πρώτου επιπέδου χειρισμού των Events και βοηθητικός κώδικας για την εγκατάσταση των συναρτήσεων χειρισμού τους δευτέρου επιπέδου, υπάρχει στα παραδείγματα κώδικα που περιέχονται στο Cell SDK και επεξηγείται στο 18ο κεφάλαιο του εγγράφου Cell Broadband Engine Programming Handbook.

3.7 Προγραμματιστικά μοντέλα που μπορούν να χρησιμοποιηθούν στον Cell

Λόγω της ιδιαίτερης αρχιτεκτονικής του επεξεργαστή Cell, παρέχεται η δυνατότητα χρήσης πολλών διαφορετικών προγραμματιστικών μοντέλων κατά την ανάπτυξη εφαρμογών για αυτόν. Μερικά από τα μοντέλα αυτά είναι τα ακόλουθα:

- Μοντέλο κλήσεων απομακρυσμένων διαδικασιών (Remote Procedure Calls – RPC): στο μοντέλο αυτό, η κεντρική εφαρμογή εκτελείται στο κεντρικό στοιχείο επεξεργασίας (PPE) και καλεί επιλεγμένες συναρτήσεις, οι οποίες εκτελούνται σε έναν ή περισσότερους συνεπεξεργαστές. Το κεντρικό στοιχείο επεξεργασίας καλεί τις συναρτήσεις αυτές όπως θα καλούσε οποιαδήποτε τοπική συνάρτηση. Με τον τρόπο αυτό, οι λεπτομέρειες χαμηλού επιπέδου σχετικά με την επικοινωνία των στοιχείων επεξεργασίας και την εκτέλεση εντολών άμεσης πρόσβασης στη μνήμη (DMA) αποκρύπτονται από το προγραμματιστή. Ο Interface Language Definition (IDL) Compiler είναι ένας μεταγλωττιστής που περιέχεται στο Cell SDK με σκοπό να απλοποιήσει την χρήση του συγκεκριμένου προγραμματιστικού μοντέλου.
- Μοντέλο επέκτασης συσκευών (Device-Extension Model): αποτελεί μια ειδική περίπτωση του προηγούμενου μοντέλου, στο οποίο οι συνεπεξεργαστές λειτουργούν ως συσκευές εισόδου-εξόδου, είτε δρουν ως ευφυείς διαμεσολαβητές (front ends) μεταξύ των υπόλοιπων στοιχείων επεξεργασίας και της συσκευής εισόδου-εξόδου. Σε αυτή την περίπτωση, το λογισμικό που εκτελείται στο συνεπεξεργαστή αποτελεί συνήθως μέρος του λειτουργικού συστήματος και έχει προνομιούχο πρόσβαση σε πόρους του συστήματος (π.χ. καταχωρητές κάποιας συσκευής).
- Μοντέλο επιτάχυνσης υπολογισμών (Computation Acceleration Model): χρησιμοποιείται για

τη βελτίωση της επίδοσης εφαρμογών που χρησιμοποιούν απαιτητικούς μαθηματικούς υπολογισμούς. Το μεγαλύτερο μέρος των υπολογισμών εκτελείται στους συνεπεξεργαστές, με το κεντρικό στοιχείο επεξεργασίας να εκτελεί λειτουργίες συντονισμού των συνεπεξεργαστών και αλληλεπίδρασης με το λειτουργικό σύστημα. Απαιτεί να ενσωματωθούν στο πρόγραμμα που εκτελείται στους συνεπεξεργαστές κατάλληλες εντολές άμεσης πρόσβασης στη μνήμη για την μεταφορά εντολών και δεδομένων. Ο καταμερισμός των υπολογισμών μπορεί να γίνει είτε "χειροκίνητα" από τον προγραμματιστή, είτε αυτοματοποιημένα με τη βοήθεια εξειδικευμένων μεταγλωττιστών.

- Μοντέλο Streaming: κάθε συνεπεξεργαστής αποτελεί μέρος μιας σωλήνωσης (pipeline) και δέχεται δεδομένα υπό μορφή "ρεύματος" (stream), στα οποία εκτελεί υπολογισμούς και στη συνέχεια τα περνάει στο επόμενο στάδιο της σωλήνωσης. Το εύρος ζώνης για την επικοινωνία των στοιχείων που βρίσκονται μέσα στο ολοκληρωμένο κύκλωμα του Cell ξεπερνάει κατά πολύ το εύρος ζώνης για την επικοινωνία με τις συσκευές που είναι εξωτερικές ως προς τον Cell, για το λόγο αυτό αν όλοι οι συνεπεξεργαστές εκτελούν ανάλογο όγκο υπολογισμών, η μεταφορά των δεδομένων γίνεται κυρίως μέσα στον Cell, περιορίζοντας την επικοινωνία με τις εξωτερικές συσκευές και την κεντρική μνήμη και άρα υπάρχει αποδοτικότερη εκμετάλλευση του διαύλου.
- Μοντέλο διαμοιραζόμενης μνήμης: ο επεξεργαστής Cell μπορεί να προγραμματιστεί ως ένας πολυεπεξεργαστής διαμοιραζόμενης μνήμης που χρησιμοποιεί δύο διαφορετικά σύνολα εντολών, με διατήρηση της συνάφειας της μνήμης, μιας και οι εντολές άμεσης πρόσβασης στη μνήμη (DMA) διατηρούν την συνάφεια της κεντρικής μνήμης. Οι εντολές ανάγνωσης από την κεντρική μνήμη για λογαριασμό ενός συνεπεξεργαστή, αντικαθιστώνται από μία εντολή DMA μεταφοράς δεδομένων από την κεντρική μνήμη στην τοπική μνήμη ακολουθούμενη από μια εντολή ανάγνωσης από την τοπική μνήμη, ενώ οι εντολές εγγραφής στην κεντρική μνήμη αντικαθιστώνται από μια εντολή εγγραφής στην τοπική μνήμη ακολουθούμενη από μία εντολή DMA μεταφοράς δεδομένων από την τοπική στην κεντρική μνήμη.
- Μοντέλο ασύμμετρων νημάτων εκτέλεσης: στο μοντέλο αυτό μπορούν να δημιουργηθούν νήματα προς εκτέλεση είτε στο κεντρικό στοιχείο επεξεργασίας είτε στους συνεπεξεργαστές, με κάθε συνεπεξεργαστή να μπορεί να εκτελεί μόνο ένα νήμα κάθε φορά. Το κλασικό μοντέλο των νημάτων επεκτείνεται ώστε να υποστηρίζονται τα δύο διαφορετικά σύνολα εντολών του Cell. Το μοντέλο αυτό υλοποιείται από τη βιβλιοθήκη SPE Runtime Management Library.

- Μοντέλο νημάτων επιπέδου χρήστη: στο μοντέλο αυτό κάθε συνεπεξεργαστής μπορεί να εκτελεί περισσότερα από ένα νήματα, τα οποία δεν υποστηρίζονται από το λειτουργικό σύστημα αλλά από το λογισμικό χρήστη.

Κεφάλαιο 4

Μελέτη της επίδοσης του Cell

4.1 Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζονται διάφορες εφαρμογές που αναπτύχθηκαν για την μελέτη της επίδοσης του επεξεργαστή Cell, καθώς και τα αποτελέσματα που ελήφθησαν με τη χρήση των εφαρμογών αυτών. Αναπτύχθηκε μία εφαρμογή για τη μελέτη της επίδοσης διαφορετικών σχημάτων επικοινωνίας μεταξύ των επεξεργαστικών μονάδων του Cell και της κεντρικής μνήμης, μία απλή παράλληλη εφαρμογή χωρίς επικοινωνία των συνεπεξεργαστών μεταξύ τους και μια απλή παράλληλη εφαρμογή που χρησιμοποιεί το streaming μοντέλο.

4.2 Μετρήσεις καθυστερήσεων και εύρους ζώνης σε διαφορετικά σχήματα επικοινωνίας

4.2.1 Εισαγωγή

Αρχικά παρουσιάζεται μία εφαρμογή που αναπτύχθηκε για τη μελέτη της επίδοσης διαφορετικών σχημάτων επικοινωνίας μεταξύ των επεξεργαστικών μονάδων του Cell και της κεντρικής μνήμης. Η εφαρμογή που αναπτύχθηκε δέχεται ως είσοδο ένα σχήμα επικοινωνίας και αναλαμβάνει την εκτέλεσή του και τη συλλογή μετρήσεων για την επίδοσή του. Η μεταφορά των δεδομένων πραγματοποιείται από ή/και προς τις τοπικές μνήμες (local stores) των συνεπεξεργαστών (SPEs) μέσω εντολών άμεσης πρόσβασης στη μνήμη (Direct Memory Access - DMA). Χρησιμοποιήθηκε η γλώσσα προγραμματισμού C, με χρήση των βιβλιοθηκών που αφορούν ειδικά τον επεξεργαστή Cell. Αναπτύχθηκε ένα πρόγραμμα για το κεντρικό στοιχείο επεξεργασίας (PPE) και ένα πρόγραμμα για τους συνεπεξεργαστές (SPE). Το πρόγραμμα που εκτελείται στο κεντρικό στοιχείο επεξεργασίας αναλαμβάνει τη δημιουργία και το συντονισμό των νημάτων που εκτελούνται στους συνεπεξεργαστές, καθώς και την αποστολή σε αυτούς των στοιχείων σχετικά με τις μεταφορές που πρέπει να πραγματοποιήσουν. Το πρόγραμμα που εκτελείται στους συνεπεξεργαστές λαμβάνει τα στοιχεία για τις μεταφορές που πρόκειται να εκτελεστούν και αν αυτό είναι αναγκαίο τα επεξεργάζεται ώστε να τηρούνται οι περιορισμοί που αφορούν στις μεταφορές άμεσης πρόσβασης στη μνήμη (DMA). Στη συνέχεια προχωρεί στην εκτέλεση συγκεκριμένου αριθμού επαναλήψεων ενός βρόχου, στον οποίο

πραγματοποείται η επικοινωνία βάσει του δεδομένου σχήματος, με προαιρετικό συγχρονισμό μεταξύ των συνεπεξεργαστών. Στις παραγράφους που ακολουθούν δίνονται πιο αναλυτικά κάποιες λεπτομέρειες για την εφαρμογή. Τέλος, παρουσιάζονται οι μετρήσεις που προέκυψαν ως αποτέλεσμα της εκτέλεσης της εφαρμογής για διάφορα σχήματα επικοινωνίας σε ένα Sony Playstation 3 (στο συγκεκριμένο σύστημα ο αριθμός των διαθέσιμων για την εκτέλεση προγραμμάτων συνεπεξεργαστών είναι 6).

4.2.2 Μορφή αρχείου εισόδου

Το σχήμα επικοινωνίας παρέχεται στο πρόγραμμα με τη βοήθεια ενός αρχείου εισόδου, το οποίο περιλαμβάνει μία ή περισσότερες γραμμές της ακόλουθης μορφής:

συνεπεξεργαστής-πηγή διεύθυνση-πηγή συνεπεξεργαστής-προορισμός διεύθυνση-προορισμός μέγεθος σημαία

όπου:

- *συνεπεξεργαστής-πηγή* είναι ο αριθμός του συνεπεξεργαστή από την τοπική μνήμη του οποίου θα μεταφερθούν τα δεδομένα,
- *διεύθυνση-πηγή* είναι η διεύθυνση στην τοπική μνήμη του συγκεκριμένου συνεπεξεργαστή στην οποία βρίσκονται αποθηκευμένα τα προς μεταφορά δεδομένα,
- *συνεπεξεργαστής-προορισμός* είναι ο αριθμός του συνεπεξεργαστή στην τοπική μνήμη του οποίου θα μεταφερθούν τα δεδομένα,
- *διεύθυνση-προορισμός* η αντίστοιχη διεύθυνση στην τοπική μνήμη του,
- *μέγεθος* είναι το μέγεθος της μεταφοράς,
- η *σημαία* (flag) ισούται με 0 όταν επιθυμούμε η μεταφορά να γίνεται με πρωτοβουλία του συνεπεξεργαστή προέλευσης των δεδομένων, με εντολή άμεσης πρόσβασης στη μνήμη (DMA) τύπου put, ή με 1 όταν επιθυμούμε η μεταφορά να γίνεται με πρωτοβουλία του συνεπεξεργαστή στον οποίο θα καταλήξουν τα δεδομένα, με εντολή άμεσης πρόσβασης στη μνήμη (DMA) τύπου get.

Σημειώνεται ότι τα πεδία *συνεπεξεργαστής-πηγή*, *συνεπεξεργαστής-προορισμός* μπορούν να πάρουν τιμές από 0 ως MAX_SPES-1, όπου MAX_SPES είναι ο μέγιστος αριθμός συνεπεξεργαστών στο συγκεκριμένο επεξεργαστή Cell, ενώ επίσης υπάρχει η δυνατότητα να έχουν την τιμή -1, δηλώνοντας ότι η πηγή ή ο προορισμός της μεταφοράς είναι η κύρια μνήμη. Σε αυτή την περίπτωση, η αντίστοιχη διεύθυνση δεν έχει σημασία, καθώς για τις μεταφορές από/προς την κεντρική μνήμη, για τη

διευθυνσιοδότηση της οποίας χρησιμοποιούνται εικονικές διευθύνσεις, χρησιμοποιούνται ενταμιευτές (buffers) που δεσμεύονται δυναμικά.

4.2.3 Παράμετροι εκτέλεσης

Το πρόγραμμα δέχεται κατά την εκτέλεση τρεις παραμέτρους. Η πρώτη παράμετρος είναι το όνομα του αρχείου εισόδου, που περιέχει την περιγραφή του σχήματος επικοινωνίας, όπως αναλύθηκε παραπάνω. Η δεύτερη παράμετρος μας παρέχει τη δυνατότητα να επιλέξουμε τον αριθμό των εντολών άμεσης πρόσβασης στη μνήμη (DMA) που εκδίδονται προτού η εκτέλεση του προγράμματος σταματήσει για να περιμένουμε την ολοκλήρωση των μεταφορών. Όταν η τιμή της παραμέτρου ισούται με τη μονάδα, έχουμε απλή blocking επικοινωνία. Αυξάνοντάς την σταδιακά, εκμεταλλευόμαστε τη δυνατότητα ύπαρξης πολλών μεταφορών σε εξέλιξη ταυτόχρονα, δυνατότητα που μας παρέχει ο διάδρομος διασύνδεσης (EIB) και οι ελεγκτές πρόσβασης στη μνήμη (Memory Flow Controllers – MFC) του κάθε συνεπεξεργαστή. Συγκεκριμένα είναι δυνατόν για κάθε συνεπεξεργαστή να βρίσκονται ταυτόχρονα σε εξέλιξη 16 εντολές άμεσης πρόσβασης στη μνήμη (DMA), φτάνοντας έτσι τον συνολικό μέγιστο αριθμό των 128 εντολών για ολόκληρο τον επεξεργαστή. Την παραπάνω δυνατότητα την εκμεταλλευόμαστε δίνοντας στην εν λόγω παράμετρο μια μεγάλη τιμή, προσεγγίζοντας έτσι τη non-blocking επικοινωνία, με αποτέλεσμα την αύξηση του εύρους ζώνης. Η τρίτη παράμετρος είναι ο αριθμός των επαναλήψεων του βρόχου που αποτελεί το κυρίως σώμα του προγράμματος που εκτελούν οι συνεπεξεργαστές, δηλαδή ο αριθμός που θα επαναληφθεί η εκτέλεση του δεδομένου συνόλου μεταφορών.

4.2.4 Περιορισμοί των εντολών άμεσης πρόσβασης στη μνήμη (DMA) και τρόπος εξασφάλισης της τήρησής τους

Σε κάθε μεταφορά άμεσης πρόσβασης στη μνήμη (DMA) από/προς μια τοπική μνήμη (local store) πρέπει να τηρούνται οι ακόλουθοι περιορισμοί:

- Το μέγεθος μιας μεταφοράς μπορεί να είναι 1, 2, 4, 8, 16 bytes ή οποιοδήποτε πολλαπλάσιο των 16 bytes με μέγιστο τα 16 KB.
- Οι μεταφορές DMA που έχουν μέγεθος μικρότερο από 16 bytes πρέπει να είναι φυσικά ευθυγραμμισμένες, που σημαίνει ότι οι διευθύνσεις προέλευσης και προορισμού πρέπει να διαιρούνται από το μέγεθος της μεταφοράς.
- Οι μεταφορές DMA που έχουν μέγεθος ίσο με 16 bytes ή μεγαλύτερο, πρέπει να είναι ευθυγραμμισμένες (στοιχισμένες) κατά 16 bytes, δηλαδή οι διευθύνσεις προέλευσης και

προορισμού πρέπει να έχουν τα τέσσερα λιγότερο σημαντικά ψηφία ίσα με 0.

- Τέλος, πρέπει τα τέσσερα λιγότερο σημαντικά ψηφία της effective address να είναι ίσα με τα αντίστοιχα ψηφία της διεύθυνσης στην τοπική μνήμη (local store address).

Αν κάποια μεταφορά δεν πληροί κάποια από τις παραπάνω προϋποθέσεις, κατά την εκτέλεσή της παρουσιάζεται εξαίρεση (alignment exception). Αυτό που τελικά παρατηρεί ο χρήστης είναι σφάλμα διαδρόμου (bus error).

Στο πρόγραμμα που αναπτύχθηκε, οι αρχικές μεταφορές που δίνονται προς εκτέλεση μέσω του αρχείου εισόδου ελέγχονται απλά ως προς τον τελευταίο περιορισμό (δηλαδή η effective address και η διεύθυνση τοπικής μνήμης να έχουν την ίδια ευθυγράμμιση μέσα σε μία τετραπλή λέξη - quadword). Σε κατάλληλο τμήμα του προγράμματος φροντίζεται αν οι μεταφορές δεν συμμορφώνονται ως προς κάποιον από τους υπόλοιπους περιορισμούς, να χωριστούν με τέτοιο τρόπο ώστε η εκτέλεση της μεταφοράς να μη γίνεται πλέον από μια απλή εντολή άμεσης πρόσβασης στη μνήμη (DMA), αλλά από μια εντολή λίστας, με την οποία θα μεταφέρονται κατά τμήματα τα δεδομένα που αντιστοιχούν στις δεδομένες διευθύνσεις, χωρισμένα σε μικρότερες μεταφορές συμβατές με τους παραπάνω περιορισμούς. Για παράδειγμα, μια μεταφορά με μέγεθος 3 byte δε θα απορρίπτεται ως λανθασμένη, αλλά θα φροντίζεται να εκτελεστεί σε δύο τμήματα, ένα του ενός και ένα των δύο byte, λαμβάνοντας φυσικά υπόψη και τυχόν απαραίτητες διορθώσεις ως προς τη στοίχιση (alignment).

4.2.5 Συγχρονισμός και επικοινωνία μεταξύ των επεξεργαστικών μονάδων

Στην εφαρμογή χρειάστηκαν δύο είδη συγχρονισμού, αφενός ένα barrier ώστε οι συνεπεξεργαστές να ξεκινούν και να ολοκληρώνουν την εκτέλεση του προγράμματος συγχρονισμένα, αφετέρου για να υπάρχει κάποια δίκαιη κατανομή του εύρους ζώνης στην περίπτωση που περισσότεροι από ένας συνεπεξεργαστές εκτελούν μεταφορές από/προς την ίδια διεύθυνση σε μία τοπική μνήμη κάποιου άλλου συνεπεξεργαστή. Το barrier υλοποιήθηκε με τη βοήθεια του κεντρικού στοιχείου επεξεργασίας, που επικοινωνεί με τους συνεπεξεργαστές μέσω των inbound και outbound SPE Mailboxes, συντονίζοντας τους ώστε να μην προχωρήσουν με την εκτέλεση του προγράμματος πέραν του barrier μέχρι να έχουν φτάσει στο ίδιο σημείο εκτέλεσης (barrier) όλοι οι υπόλοιποι συνεπεξεργαστές. Η δεύτερη περίπτωση συγχρονισμού υλοποιήθηκε με αποστολή σημάτων μέσω του SPE Signal Notification Register 1.

Σε όποια επιπλέον σημεία του προγράμματος χρειάζεται επικοινωνία-συγχρονισμός μεταξύ του κεντρικού στοιχείου επεξεργασίας και των συνεπεξεργαστών, αυτό γίνεται με τον μηχανισμό των SPE Mailboxes.

4.2.6 Μέθοδος λήψης των μετρήσεων χρόνου

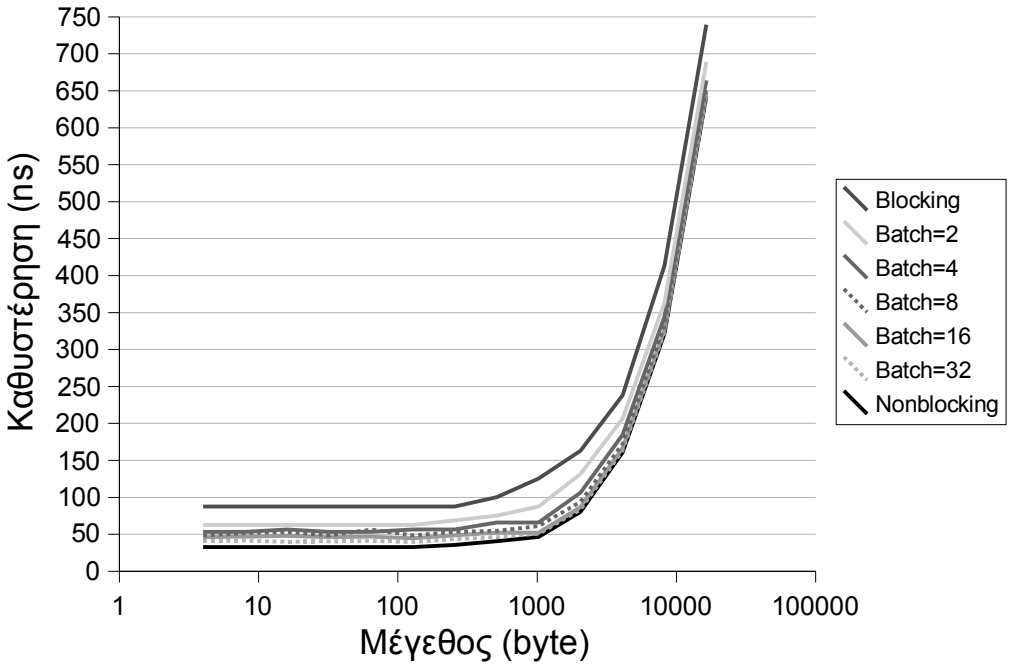
Οι μετρήσεις χρόνου στο παραπάνω πρόγραμμα έγιναν για λόγους ακρίβειας με τη βοήθεια ειδικών καταχωρητών, του Timebase Register, ο οποίος βρίσκεται στο κεντρικό στοιχείο επεξεργασίας, και των SPE Decrementers. Κάθε συνεπεξεργαστής διαθέτει έναν SPE Decrementer. Τόσο ο Timebase Register όσο και οι SPE Decrementers αλλάζουν την τιμή τους κατά 1 με την ίδια συχνότητα, η οποία για το σύστημα που χρησιμοποιήθηκε για τις μετρήσεις είναι 79800000 κύκλοι/δευτερόλεπτο, με τη διαφορά ότι η τιμή του Timebase Register αυξάνεται σε κάθε κύκλο, ενώ των SPE Decrementers μειώνεται. Στην περίπτωση των μεταφορών μεταξύ ενός μόνο ζεύγους συνεπεξεργαστών ή μεταξύ ενός συνεπεξεργαστή και της κεντρικής μνήμης, όπου μας ενδιέφερε να μετρήσουμε την επίδοση μεμονωμένων μεταφορών (blocking και non-blocking), μετρήθηκε με χρήση του SPE Decrementer ο αριθμός κύκλων που χρειάζονται για την ολοκλήρωση μίας ή περισσότερων εντολών που εκδίδονται ταυτόχρονα (ανάλογα με την τιμή της δεύτερης παραμέτρου που δίνεται στο πρόγραμμα). Στην περίπτωση των συλλογικών σχημάτων επικοινωνίας, στον υπολογισμό του εύρους ζώνης λαμβάνεται υπ' όψιν ο συνολικός χρόνος εκτέλεσης του σχήματος επικοινωνίας, μετρώντας την τιμή του Timebase Register πριν και μετά τα διαδοχικά barriers που καλούνται στο κεντρικό στοιχείο επεξεργασίας για το συγχρονισμό των συνεπεξεργαστών.

4.2.7 Μετρήσεις - ερμηνεία

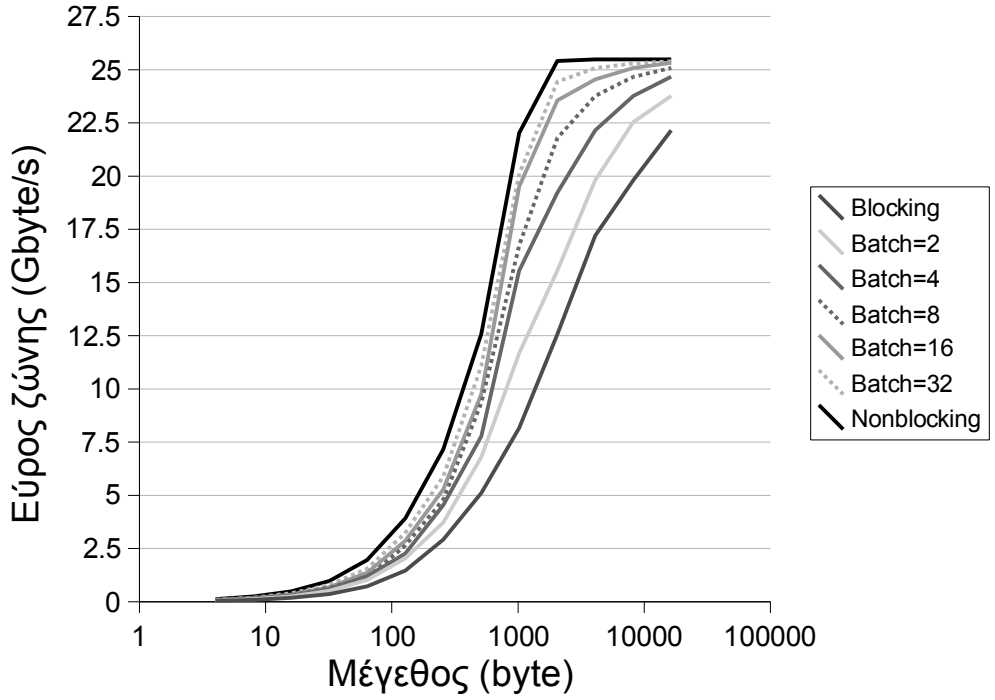
Με τη βοήθεια της εφαρμογής, εκτελέστηκαν μετρήσεις για τις ακόλουθες περιπτώσεις:

4.2.7.1 Μεταφορές DMA μεταξύ ενός ζεύγους συνεπεξεργαστών

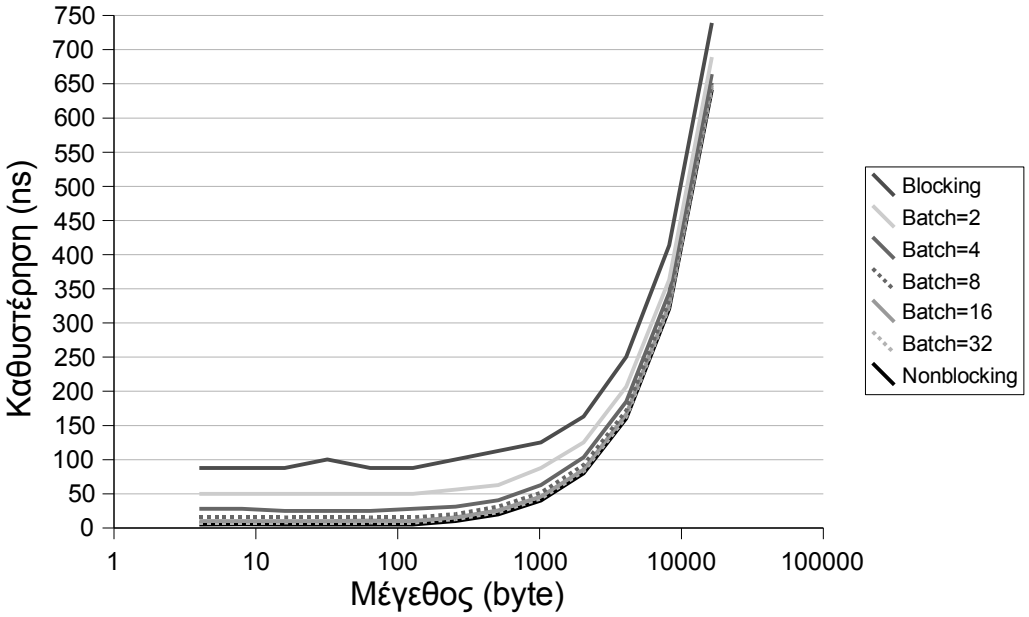
Σε αυτή την κατηγορία μετρήσεων εκτελέστηκαν μεταφορές μεταξύ των συνεπεξεργαστών (SPE) 0 (προέλευση) και 1 (προορισμός). Οι μετρήσεις που ελήφθησαν αφορούν διάφορα μεγέθη μεταφορών μεταξύ 4 byte και 16 KB. Επίσης ελήφθησαν μετρήσεις για την περίπτωση όπου περιμένουμε να ολοκληρωθεί η μεταφορά μετά από την έκδοση κάθε εντολής, μετά από την έκδοση ενός συγκεκριμένου αριθμού εντολών (συγκεκριμένα 2, 4, 8, 16 και 32) και τέλος για την περίπτωση non-blocking επικοινωνίας (η οποία προσεγγίζεται αυξάνοντας τον αριθμό των εντολών που εκδίδονται σε 16384). Τα αποτελέσματα των μετρήσεων συνοψίζονται στα διαγράμματα που ακολουθούν:



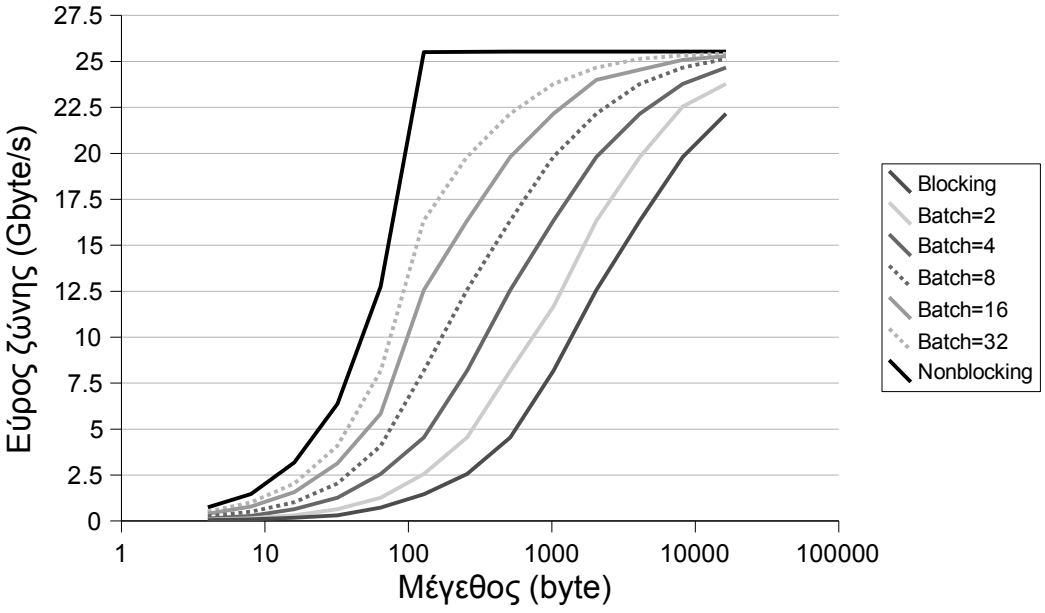
Καθυστερήσεις εντολών τύπου put



Εύρος ζώνης εντολών τύπου put



Καθυστερήσεις εντολών τύπου get



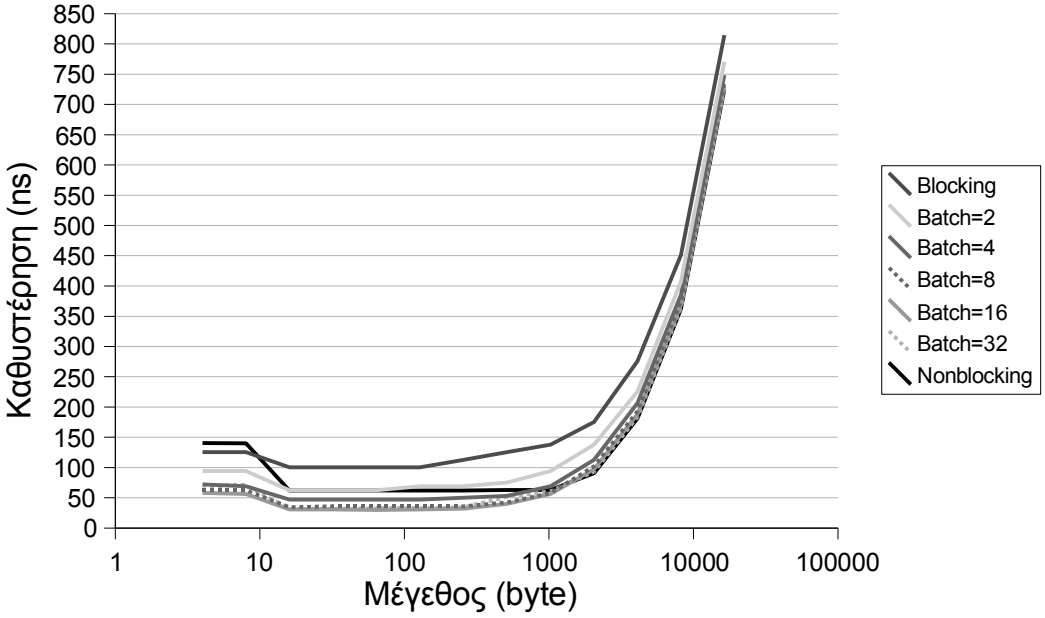
Εύρος ζώνης εντολών τύπου get

Από τα παραπάνω διαγράμματα φαίνεται, όπως είναι αναμενόμενο, πως σε όλες τις περιπτώσεις το εύρος ζώνης αυξάνεται με τη σταδιακή αύξηση του μεγέθους της μεταφοράς, φτάνοντας τελικά πολύ κοντά στο θεωρητικό μέγιστο των 25.6 Gbytes/s. Στην περίπτωση των εντολών τύπου get το μέγιστο αυτό επιτυγχάνεται στην περίπτωση της non-blocking επικοινωνίας για μεγέθη μεταφορών από 128 bytes και πάνω, ενώ στην περίπτωση των μεταφορών τύπου put για μεγέθη μεταφορών από 1024 bytes και πάνω. Επίσης παρατηρούμε πως η ταυτόχρονη έκδοση πολλαπλών εντολών, ακόμα και όταν δεν πρόκειται για non-blocking επικοινωνία καθαρά, βελτιώνει σημαντικά το εύρος ζώνης. Το αποτέλεσμα αυτό είναι περισσότερο εμφανές στην περίπτωση εντολών τύπου get, όπου ειδικά για μεγέθη μεταφορών μεγαλύτερα ή ίσα με 128 bytes η βελτίωση που επιφέρει η έκδοση πολλαπλών εντολών είναι θεαματική. Ακόμα, οι εντολές τύπου get φαίνονται να έχουν καλύτερη συμπεριφορά από τις εντολές τύπου put στην συγκεκριμένη περίπτωση. Προκύπτει τελικά το συμπέρασμα πως είναι προτιμότερο να χρησιμοποιούνται εντολές τύπου get μεγέθους 128 byte ή πολλαπλάσιου των 128 byte και να αξιοποιείται, στο βαθμό που αυτό είναι εφικτό, η δυνατότητα να βρίσκονται πολλαπλές εντολές σε εξέλιξη.

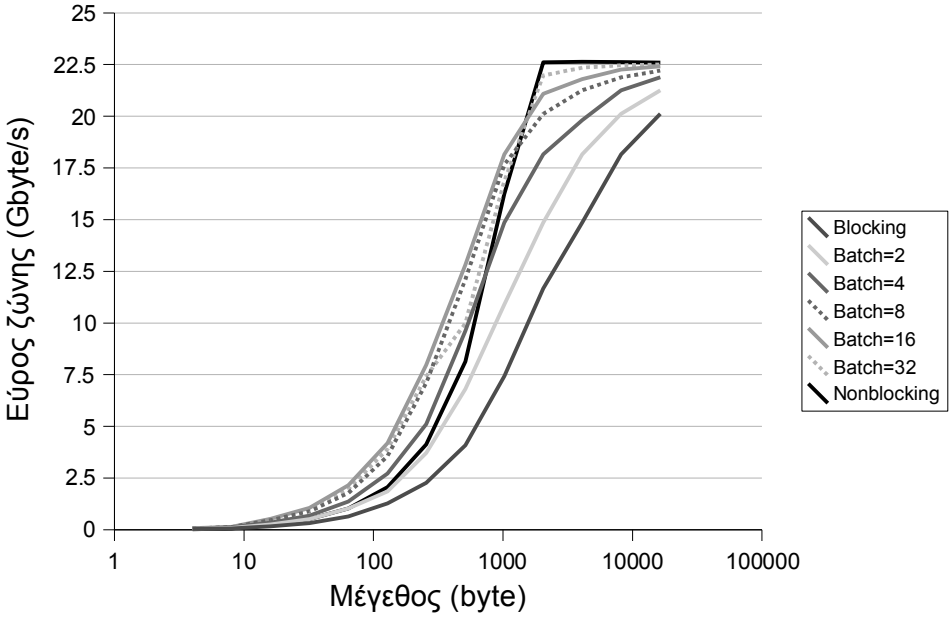
Τέλος, πρέπει να αναφερθεί πως μετρήσεις μεταξύ διαφορετικών ζευγών συνεπεξεργαστών δεν έδειξαν σημαντική διαφορά στην επίδοση.

4.2.7.2 Μεταφορές DMA μεταξύ ενός συνεπεξεργαστή και μνήμης

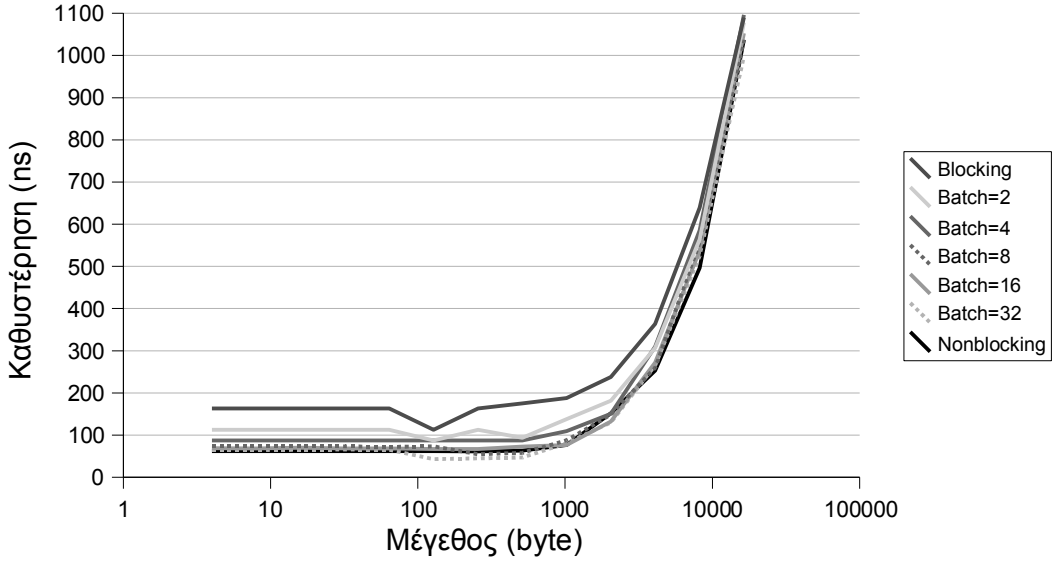
Για τις μεταφορές αυτής της κατηγορίας χρησιμοποιήθηκε ο συνεπεξεργαστής (SPE) 0. Οι μετρήσεις που ελήφθησαν αφορούν διάφορα μεγέθη μεταφορών μεταξύ 4 byte και 16 KB. Επίσης ελήφθησαν μετρήσεις για την περίπτωση όπου περιμένουμε να ολοκληρωθεί η μεταφορά μετά από την έκδοση κάθε εντολής, μετά από την έκδοση ενός συγκεκριμένου αριθμού εντολών (συγκεκριμένα 2, 4, 8, 16 και 32) και τέλος για την περίπτωση non-blocking επικοινωνίας (η οποία προσεγγίζεται αυξάνοντας τον αριθμό των εντολών που εκδίδονται σε 16384). Τα αποτελέσματα των μετρήσεων παρουσιάζονται στα διαγράμματα που ακολουθούν:



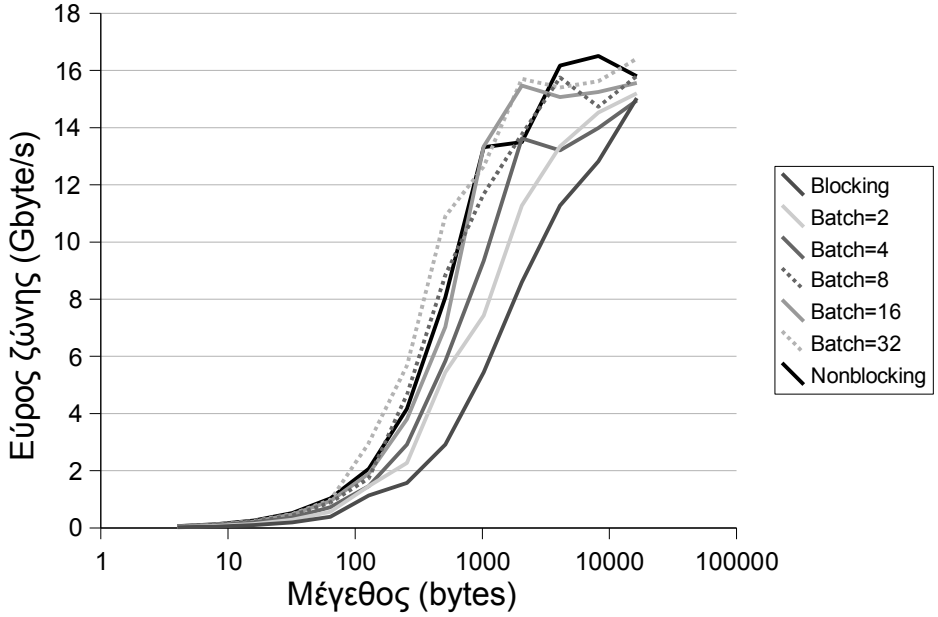
Καθυστερήσεις εντολών τύπου put



Εύρος ζώνης εντολών τύπου put



Καθυστερήσεις εντολών τύπου get

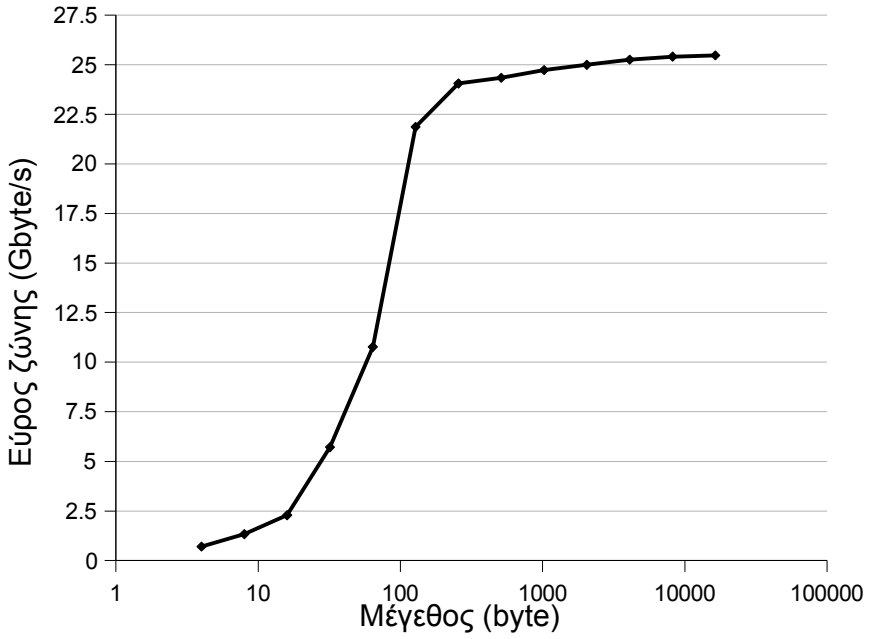


Εύρος ζώνης εντολών τύπου get

Για τις μεταφορές από και προς την κύρια μνήμη ισχύουν σε γενικές γραμμές οι παρατηρήσεις που έγιναν και για τις μεταφορές από/προς τις τοπικές μνήμες. Παρατηρούμε και σε αυτή την περίπτωση πως η non-blocking επικοινωνία έχει βελτιωμένες επιδόσεις. Μία ακόμα σημαντική παρατήρηση είναι πως οι εντολές εγγραφής στην κεντρική μνήμη (put) παρουσιάζουν αρκετά καλύτερη συμπεριφορά από τις εντολές ανάγνωσης από την κεντρική μνήμη (get), οι οποίες δεν καταφέρνουν να πλησιάσουν το θεωρητικό μέγιστο (25.6 Gbytes/s). Καταλληλότερα φαίνεται να είναι τα μηνύματα μεγέθους 1024 byte και μεγαλύτερα, καθώς με αυτά επιτυγχάνεται μεγαλύτερο εύρος ζώνης.

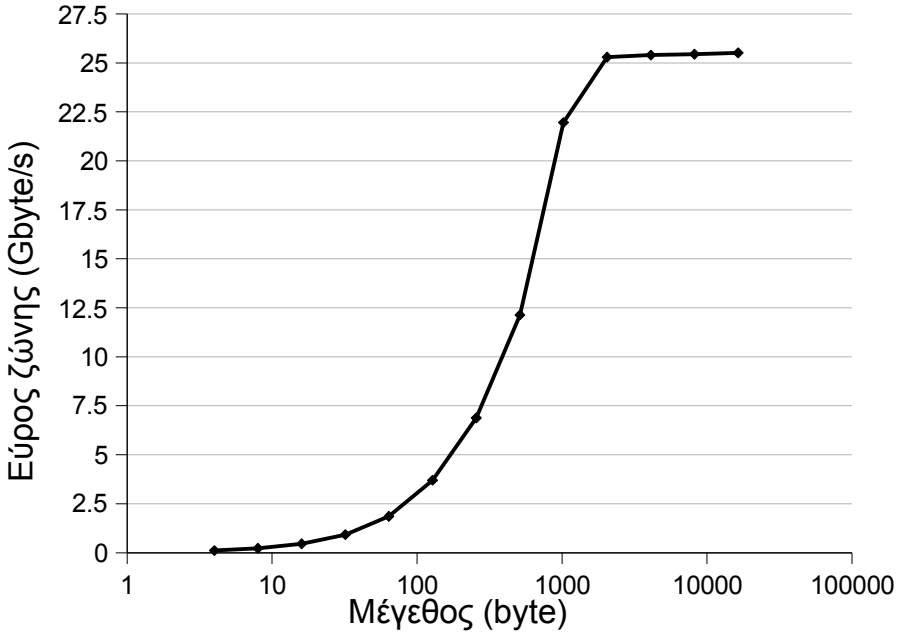
4.2.7.3 Μεταφορές DMA από πολλούς συνεπεξεργαστές προς έναν συνεπεξεργαστή

Σε αυτή την κατηγορία μετρήσεων εκτελέστηκαν μεταφορές από/προς τον συνεπεξεργαστή 0 από έναν αριθμό συνεπεξεργαστών μεταξύ του 2 και του μέγιστου αριθμού διαθέσιμων συνεπεξεργαστών (6 για το Sony Playstation 3), πλην του συνεπεξεργαστή 0 φυσικά. Μελετήθηκε η επίδραση του αριθμού των συνεπεξεργαστών που λαμβάνουν μέρος στην επικοινωνία, για σταθερό μέγεθος μεταφορών ίσο με 16Kbyte, καθώς επίσης και η επίδραση του μεγέθους του μηνύματος, στην περίπτωση που χρησιμοποιούνται όλοι οι διαθέσιμοι συνεπεξεργαστές. Για όλες τις μετρήσεις χρησιμοποιήθηκε non-blocking επικοινωνία. Ακολουθούν τα διαγράμματα με τα αποτελέσματα των μετρήσεων:



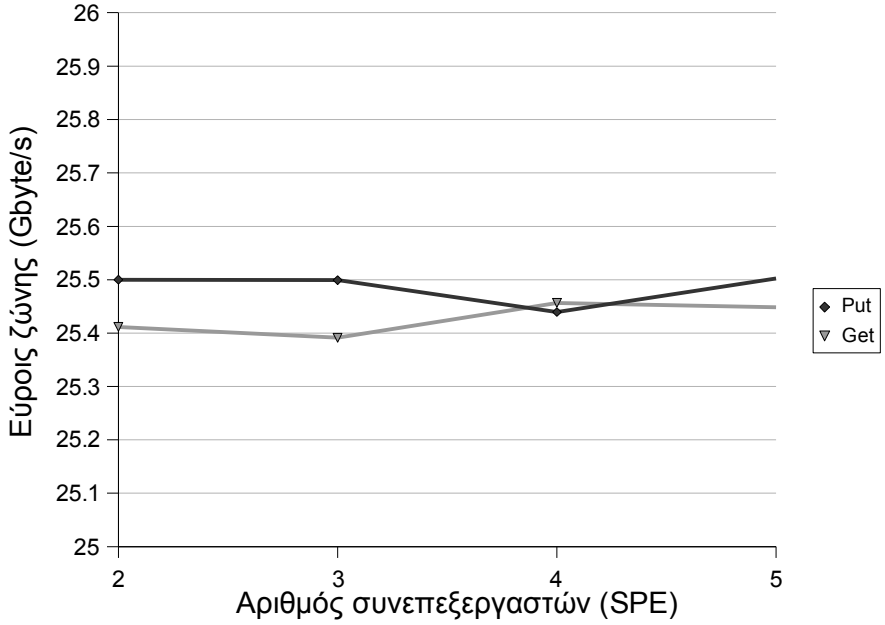
Εύρος ζώνης συναρτήσει μεγέθους μηνύματος για μεταφορές τύπου get

(μετρήσεις με 5 συνεπεξεργαστές να μεταφέρουν δεδομένα από τον συνεπεξεργαστή 0)



Εύρος ζώνης συναρτήσει μεγέθους μηνύματος για μεταφορές τύπου put

(μετρήσεις με 5 συνεπεξεργαστές να μεταφέρουν δεδομένα προς τον συνεπεξεργαστή 0)



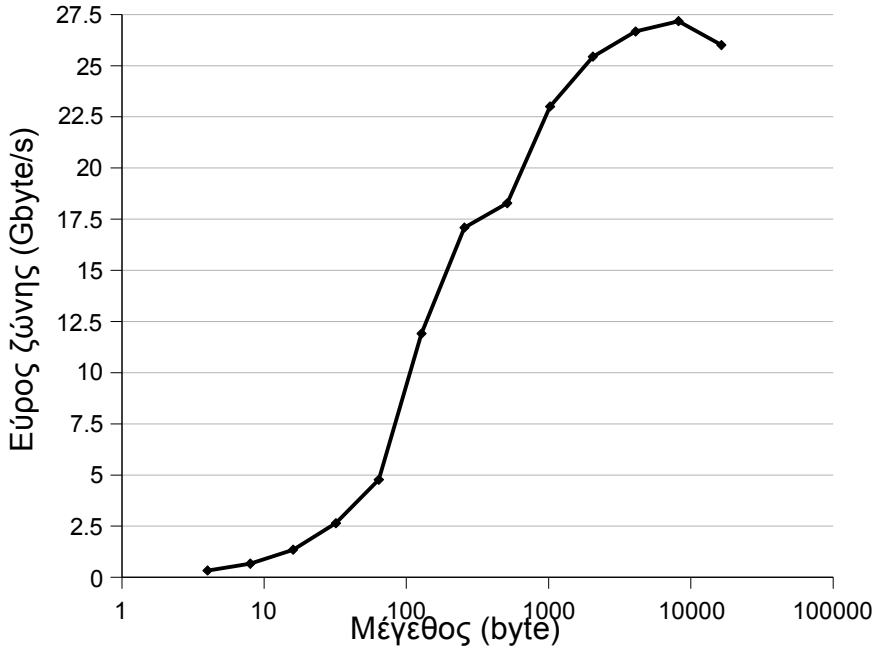
Εύρος ζώνης συναρτήσει του αριθμού των συνεπεξεργαστών

(non-blocking επικοινωνία με μηνύματα μεγέθους 16Kbyte)

Παρατηρούμε ότι στις μεταφορές DMA τύπου get, όταν χρησιμοποιούνται όλοι οι συνεπεξεργαστές, επιτυγχάνεται εύρος ζώνης κοντά στο θεωρητικό μέγιστο (25.6 Gbytes/s) για μηνύματα μεγέθους 128 byte και μεγαλύτερα. Για τις μεταφορές DMA τύπου put, παρατηρούμε το ίδιο αποτέλεσμα για μηνύματα μεγέθους 1024 byte και μεγαλύτερα. Στην περίπτωση που μετρήσαμε το εύρος ζώνης συναρτήσει του αριθμού των συνεπεξεργαστών που λαμβάνουν μέρος στο σχήμα επικοινωνίας, παρατηρούμε ότι το εύρος ζώνης που επιτυγχάνεται είναι πολύ κοντά στο θεωρητικό μέγιστο σε όλες τις περιπτώσεις, με ασήμαντες διαφορές μεταξύ των μεταφορών τύπου get και put.

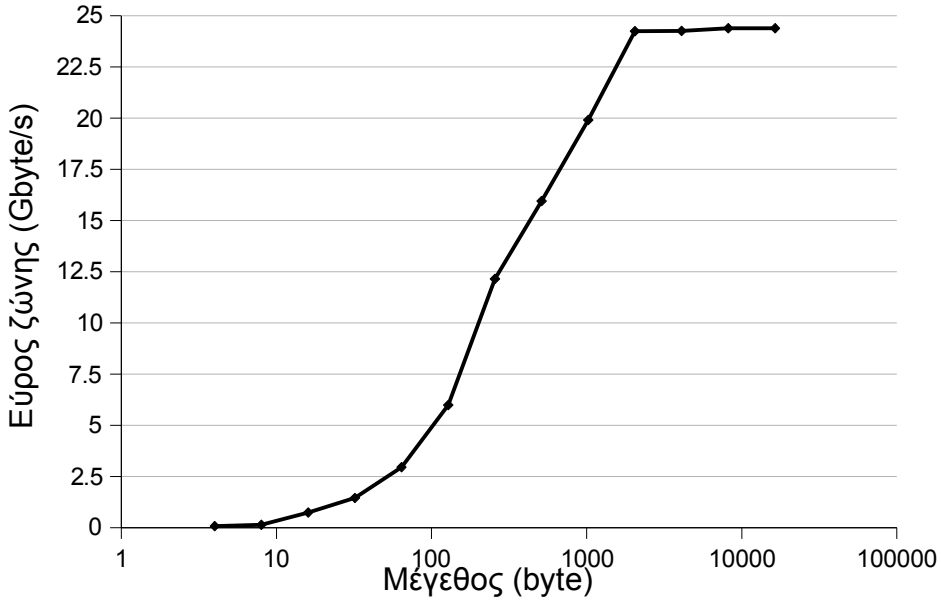
4.2.7.4 Μεταφορές DMA από πολλούς συνεπεξεργαστές προς τη μνήμη και αντίστροφα

Σε αυτή την κατηγορία μετρήσεων εκτελέστηκαν μεταφορές από/προς την κεντρική μνήμη από έναν αριθμό συνεπεξεργαστών μεταξύ του 2 και του μέγιστου αριθμού διαθέσιμων συνεπεξεργαστών (6 για το Sony Playstation 3). Μελετήθηκε η επίδραση του αριθμού των συνεπεξεργαστών που λαμβάνουν μέρος στην επικοινωνία, για σταθερό μέγεθος μεταφορών ίσο με 16Kbyte, καθώς επίσης και η επίδραση του μεγέθους του μηνύματος, στην περίπτωση που χρησιμοποιούνται όλοι οι διαθέσιμοι συνεπεξεργαστές. Για όλες τις μετρήσεις χρησιμοποιήθηκε non-blocking επικοινωνία. Ακολουθούν τα διαγράμματα με τα αποτελέσματα των μετρήσεων:



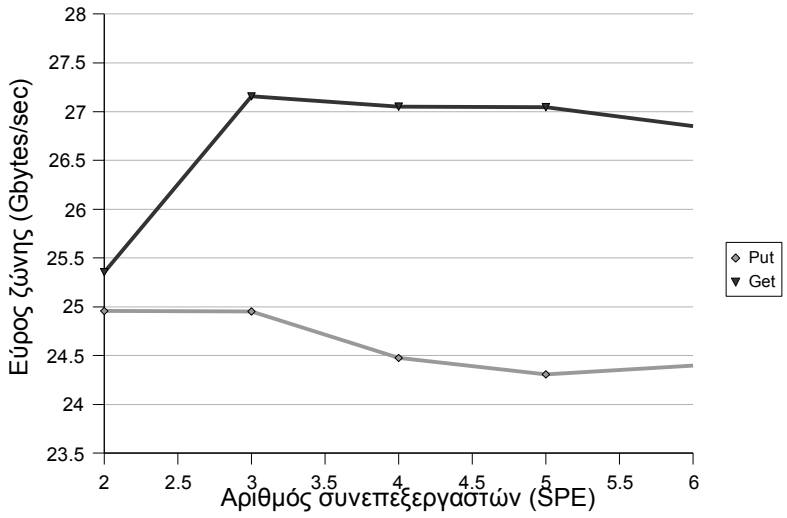
Εύρος ζώνης συναρτήσει μεγέθους μηνύματος για μεταφορές τύπου get

(μετρήσεις με 6 συνεπεξεργαστές να μεταφέρουν δεδομένα από την κεντρική μνήμη)



Εύρος ζώνης συναρτήσει μεγέθους μηνύματος για μεταφορές τύπου put

(μετρήσεις με 6 συνεπεξεργαστές να μεταφέρουν δεδομένα προς την κεντρική μνήμη)



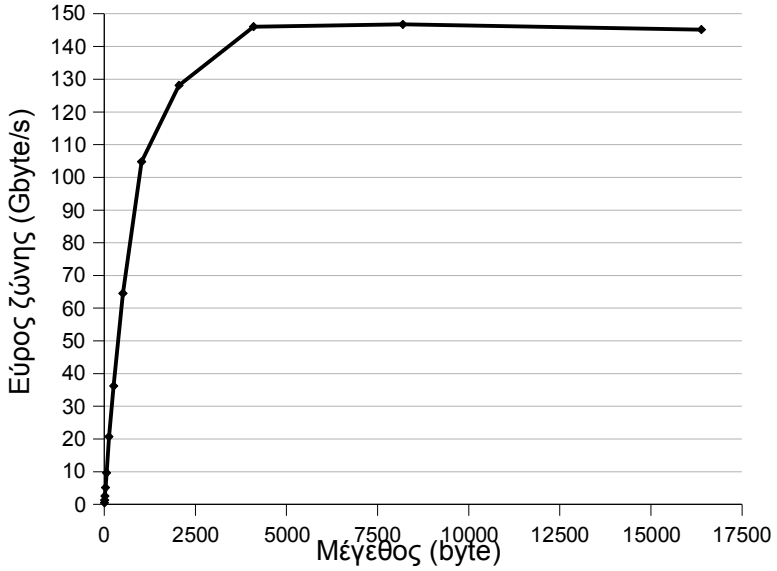
Εύρος ζώνης συναρτήσει του αριθμού των συνεπεξεργαστών

(non-blocking επικοινωνία με μηνύματα μεγέθους 16Kbyte)

Παρατηρούμε ότι στις μεταφορές DMA τύπου get, όταν χρησιμοποιούνται όλοι οι συνεπεξεργαστές, επιτυγχάνεται εύρος ζώνης κοντά στο θεωρητικό μέγιστο (25.6 Gbytes/s) για μηνύματα μεγέθους 1024 byte και μεγαλύτερα. Για τις μεταφορές DMA τύπου put, παρατηρούμε το ίδιο αποτέλεσμα για μηνύματα μεγέθους 2048 byte και μεγαλύτερα.

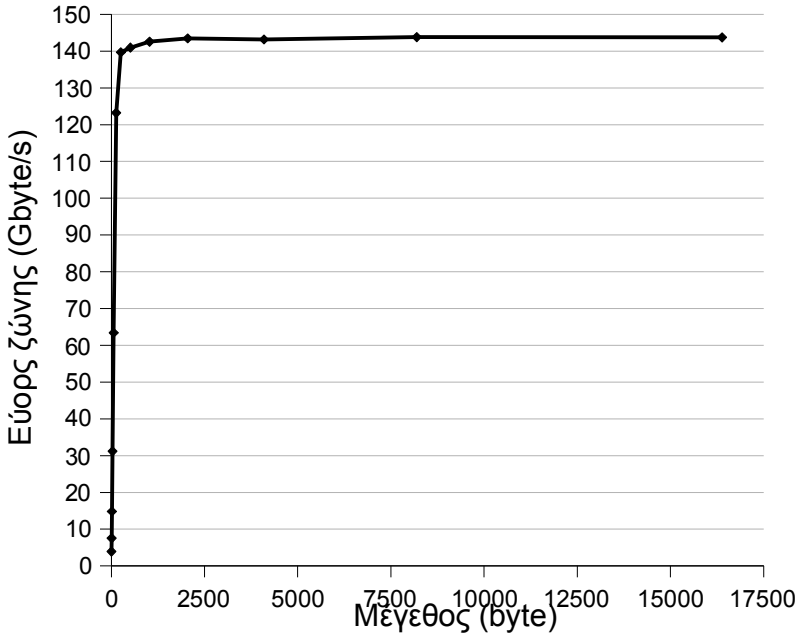
4.2.7.5 Επικοινωνία μεταξύ ζευγών συνεπεξεργαστών (pairwise pattern)

Οι μετρήσεις αυτής της κατηγορίας αφορούν μεταφορές DMA μεταξύ ζευγών συνεπεξεργαστών. Αρχικά επιλέχτηκε ο συνδυασμός (SPE0,SPE1), (SPE2,SPE3), (SPE4,SPE5) και μελετήθηκε το εύρος ζώνης για non-blocking επικοινωνία και για διάφορα μεγέθη μηνυμάτων μεταξύ 4 byte και 16 KB. Στα διαγράμματα που ακολουθούν παρουσιάζονται οι μετρήσεις που ελήφθησαν:



Εύρος ζώνης συναρτήσει μεγέθους μηνύματος για μεταφορές τύπου put

Συνδυασμός (SPE0,SPE1), (SPE2,SPE3), (SPE4,SPE5)

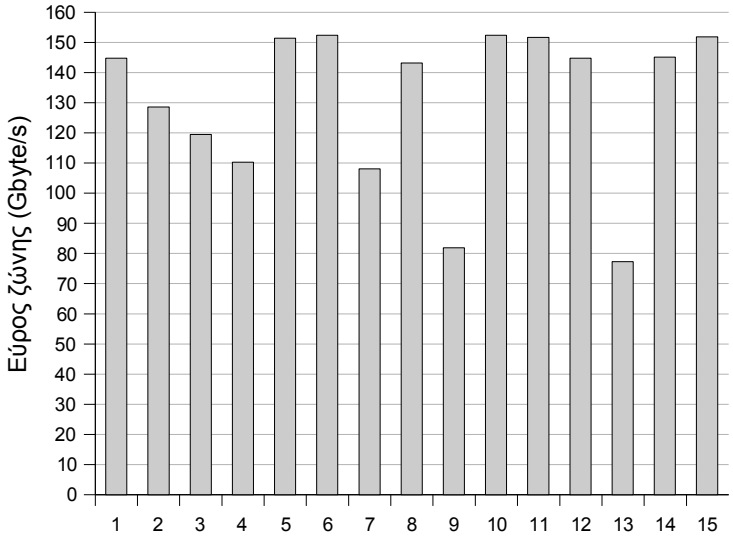


Εύρος ζώνης συναρτήσει μεγέθους μηνύματος για μεταφορές τύπου get

Συνδυασμός (SPE0,SPE1), (SPE2,SPE3), (SPE4,SPE5)

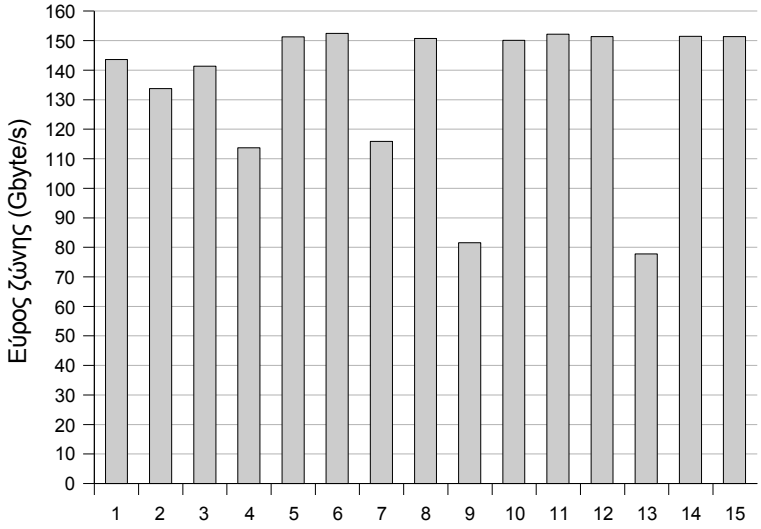
Το θεωρητικό μέγιστο εύρος ζώνης σε αυτή την περίπτωση ισούται με 6 x 25.6 Gbytes/s = 153.6 Gbytes/s. Όπως φαίνεται, στις μεταφορές DMA τύπου put, για το συγκεκριμένο συνδυασμό των συνεπεξεργαστών σε ζεύγη, το μέγιστο εύρος ζώνης επιτυγχάνεται για μηνύματα μεγέθους 1024 byte και μεγαλύτερα. Για τις μεταφορές DMA τύπου put, παρατηρούμε το ίδιο αποτέλεσμα για μηνύματα μεγέθους 4096 byte και μεγαλύτερα.

Στη συνέχεια έγιναν μετρήσεις για όλους τους πιθανούς συνδυασμούς των συνεπεξεργαστών ανά δύο, που είναι 15 για 6 διαθέσιμους συνεπεξεργαστές, ώστε να διαπιστωθεί αν κάποιοι από τους συνδυασμούς αυτούς ευνοούνται ως προς τις επιδόσεις.



Εύρος ζώνης για τους διαφορετικούς συνδυασμούς των συνεπεξεργαστών σε ζεύγη

(μεταφορές put, non-blocking με μέγεθος μηνύματος 16Kbyte)



Εύρος ζώνης για τους διαφορετικούς συνδυασμούς των συνεπεξεργαστών σε ζεύγη

(μεταφορές get, non-blocking με μέγεθος μηνύματος 16Kbyte)

Αριθμός συνδυασμού	Ζεύγος 1		Ζεύγος 2		Ζεύγος 3	
1	0	1	2	3	4	5
2	0	1	2	4	3	5
3	0	1	2	5	3	4
4	0	2	1	3	4	5
5	0	2	1	4	3	5
6	0	2	1	5	3	4
7	0	3	2	1	4	5
8	0	3	2	4	1	5
9	0	3	2	5	1	4
10	0	4	2	3	1	5
11	0	4	2	1	3	5
12	0	4	2	5	1	3
13	0	5	2	3	1	4
14	0	5	2	4	1	3
15	0	5	1	2	3	4

Συνδυασμοί συνεπεξεργαστών σε ζεύγη

Από τα παραπάνω διαγράμματα παρατηρούμε ότι ορισμένοι συνδυασμοί ευνοούνται (επιτυγχάνεται το μέγιστο εύρος ζώνης), ενώ άλλοι όχι. Αυτό οφείλεται στην κατασκευή του διαδρόμου διασύνδεσης (EIB) και στις σχετικές θέσεις μεταξύ των συνεπεξεργαστών που επικοινωνούν μεταξύ τους. Όπως έχει αναφερθεί κατά την περιγραφή της αρχιτεκτονικής του επεξεργαστή Cell, οι συνεπεξεργαστές στο “βόρειο” μέρος του ολοκληρωμένου αριθμούνται με τους άρτιους αριθμούς 0,2,4,6, ενώ οι συνεπεξεργαστές στο “νότιο” μέρος του ολοκληρωμένου αριθμούνται με τους περιττούς αριθμούς 1,3,5,7 και σε γενικές γραμμές είναι πιο αποδοτική η επικοινωνία των συνεπεξεργαστών με άρτιο αριθμό μεταξύ τους, και αυτών με περιττό αριθμό μεταξύ τους. Επίσης όσο μεγαλύτερη είναι η απόσταση μεταξύ δύο στοιχείων που επικοινωνούν τόσο πιο πιθανό είναι η μεταφορά αυτή να

συγκρούεται με άλλες που βρίσκονται ήδη σε εξέλιξη. Αυτός είναι και ο λόγος που η χειρότερη απόδοση παρατηρείται με εκείνα τα ζεύγη συνεπεξεργαστών για τα οποία οι αποστάσεις που διανύουν τα δεδομένα που μεταφέρονται είναι οι μέγιστες.

4.3 Μέτρηση της επίδοσης του Cell σε απλή παράλληλη εφαρμογή χωρίς επικοινωνία μεταξύ των συνεπεξεργαστών

4.3.1 Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζεται μία απλή παράλληλη εφαρμογή που υλοποιήθηκε με σκοπό την μέτρηση της επιτάχυνσης που μπορεί να επιτευχθεί σε απλούς υπολογισμούς με χρήση των συνεπεξεργαστών. Η εφαρμογή αυτή υπολογίζει το άθροισμα δύο πολύ μεγάλων διανυσμάτων αφενός σειριακά στο κεντρικό στοιχείο επεξεργασίας (PPE) και αφετέρου παράλληλα με τη χρήση των συνεπεξεργαστών και λαμβάνει μετρήσεις της επίδοσης, για διάφορα μεγέθη των διανυσμάτων και των ενταμιευτών που χρησιμοποιούν οι συνεπεξεργαστές, καθώς και με μεταβλητό τον αριθμό των συνεπεξεργαστών που συμμετέχουν στους υπολογισμούς. Το πρόγραμμα που αναπτύχθηκε παρότι δεν έχει κάποια ιδιαίτερη πρακτική σημασία, επιδεικνύει με απλό τρόπο την βελτίωση της επίδοσης που μπορεί να επιφέρει ο επεξεργαστής Cell στην εκτέλεση εφαρμογών απαιτητικών σε υπολογισμούς.

4.3.2 Χρήση διπλών ενταμιευτών (double buffering)

Τα προγράμματα που εκτελούνται σε κάποιον συνεπεξεργαστή συχνά, όπως στην περίπτωση του προγράμματος στο οποίο αναφέρεται το κεφάλαιο αυτό, απαιτούν τη μεταφορά μεγάλου όγκου δεδομένων από την κεντρική μνήμη στην τοπική τους μνήμη, από όπου η μονάδα επεξεργασίας του μπορεί να έχει πρόσβαση σε αυτά για να τα επεξεργαστεί. Ένα απλό σχήμα που υλοποιεί την παραπάνω λειτουργία είναι το ακόλουθο (σε ψευδοκώδικα):

```
/*
 * mem: θέση στη μνήμη από όπου μεταφέρονται τα δεδομένα
 * buffer: ενταμιευτής
 * buffer_size: μέγεθος ενταμιευτή
 * tag: ετικέτα που αντιστοιχεί στη μεταφορά DMA
 */

while (not_finished){
    dma_get_start(mem, buffer, buffer_size, tag); // έναρξη μεταφοράς DMA
    mem += buf_size;                      // ενημέρωση του δείκτη στη μνήμη
    dma_wait_for_transfer(tag);           // αναμονή για ολοκλήρωση της μεταφοράς
    process(buffer);                      // επεξεργασία των δεδομένων
```

```
}
```

Συνεπώς, μεγάλο ποσοστό του χρόνου σπαταλιέται στην αναμονή για την ολοκλήρωση της μεταφοράς. Ο συνολικός χρόνος εκτέλεσης μπορεί να μειωθεί αισθητά με την δέσμευση χώρου στην τοπική μνήμη για δύο ενταμιευτές (buffers) και την αλληλοεπικάλυψη της επεξεργασίας των δεδομένων που βρίσκονται στον έναν ενταμιευτή με τη μεταφορά δεδομένων στον άλλο. Η διαδικασία αυτή ονομάζεται χρήση διπλών ενταμιευτών (double buffering). Ακολουθεί ένα τμήμα ψευδοκώδικα που περιγράφει τα βήματα αυτής της διαδικασίας:

```
/*
 * mem: θέση στη μνήμη από όπου μεταφέρονται τα δεδομένα
 * buffers: πίνακας δύο ενταμιευτών
 * buffer_size: μέγεθος ενταμιευτών
 * tags: ετικέτες που αντιστοιχούν στις μεταφορές DMA
 * i: τρέχων δείκτης ενταμιευτή σε χρήση
 */

i = 0;
dma_get_start(mem, buffers[i], buffer_size, tags[i]); // έναρξη μεταφοράς DMA
mem += buf_size;                          // ενημέρωση του δείκτη στη μνήμη

while (not_finished){
    dma_get_start(mem, buffers[i^1], buffer_size, tags[i^1]);
                                          // έναρξη μεταφοράς DMA
    mem += buf_size;                      // ενημέρωση του δείκτη στη μνήμη
    dma_wait_for_transfer(tags[i]);       // αναμονή για ολοκλήρωση της μεταφοράς
    process(buffers[i]);                  // επεξεργασία των δεδομένων
    i ^= 1;                               // ενημέρωση δείκτη
}
```

Με τον τρόπο αυτό, επιτυγχάνεται η αύξηση του λόγου του χρόνου εκτέλεσης υπολογισμών προς το συνολικό χρόνο εκτέλεσης. Γενικά, όταν η μεταφορά των δεδομένων απαιτεί πολύ χρόνο συγκριτικά με την εκτέλεση υπολογισμών σε αυτά, δηλαδή όταν ο λόγος του χρόνου που απαιτεί η επεξεργασία των δεδομένων προς το συνολικό χρόνο εκτέλεσης είναι μικρός, η απόδοση της εφαρμογής έχει μεγάλα περιθώρια βελτίωσης με τη χρήση διπλών ενταμιευτών, μιας και επιτυχής απόκρυψη του χρόνου μεταφοράς θα μειώσει σημαντικά το συνολικό χρόνο εκτέλεσης.

4.3.3 Καταμερισμός των δεδομένων – υπολογισμοί

Έστω ότι τα αρχικά διανύσματα των οποίων το άθροισμα θέλουμε να υπολογίσουμε είναι τα a, b με μέγεθος ίσο με n. Το τελικό αποτέλεσμα αποθηκεύεται στο διάνυσμα c, το οποίο έχει επίσης μέγεθος c και τα στοιχεία του υπολογίζονται με τον ακόλουθο τύπο:

$$c_i = a_i + b_i, i = 0, \ldots, n-1$$

Έστω ότι χρησιμοποιούνται p συνεπεξεργαστές και ότι σε κάθε επανάληψη ο καθένας από αυτούς μεταφέρει δεδομένα και εκτελεί υπολογισμούς σε ενταμιευτές μεγέθους k στοιχείων. Είναι προφανές πως δε χρειάζεται επικοινωνία μεταξύ των συνεπεξεργαστών, παρά μόνο μεταφορά δεδομένων μεταξύ της τοπικής μνήμης του καθενός και της κεντρικής μνήμης.

Για να μελετηθεί η επίδραση που έχει η χρήση διπλών ενταμιευτών (double buffering) στην απόδοση της παράλληλης εφαρμογής, αναπτύχθηκαν δύο εκδόσεις του προγράμματος, μια απλή και μια με χρήση double buffering. Η απλή έκδοση ακολουθεί το παρακάτω σχήμα (σε ψευδοκώδικα):

```
/*
 * a_mem: θέση μνήμης από όπου μεταφέρονται τα στοιχεία του διανύσματος a
 * b_mem: θέση μνήμης από όπου μεταφέρονται τα στοιχεία του διανύσματος b
 * c_mem: θέση μνήμης όπου μεταφέρονται τα στοιχεία του διανύσματος c (αποτέλεσμα)
 * a_buf: ενταμιευτής για το τμήμα του διανύσματος a
 * b_buf: ενταμιευτής για το τμήμα του διανύσματος b
 * c_buf: ενταμιευτής για το τμήμα του διανύσματος c
 * k: μέγεθος ενταμιευτών (σε στοιχεία)
 * tag: ετικέτα που αντιστοιχεί στη μεταφορά DMA
 * loops: αριθμός επαναλήψεων
 * rank: αριθμός συνεπεξεργαστή
 * p: πλήθος συνεπεξεργαστών
 * n: πλήθος στοιχείων διανυσμάτων
 * datatype: τύπος στοιχείων των διανυσμάτων
 */

//αρχικοποίηση - υπολογισμός πλήθους επαναλήψεων και τροποποίηση αρχικών δεικτών
initialize(loops, a_mem, b_mem, c_mem);

while (--loops){
    dma_get_start(a_mem, a_buf, k*sizeof(datatype), tag); // έναρξη μεταφορών DMA
    dma_get_start(b_mem, b_buf, k*sizeof(datatype), tag);
    dma_wait_for_transfer(tag);           // αναμονή για ολοκλήρωση των μεταφορών
    c_buf = add(a_buff, b_buff);          // πρόσθεση των δεδομένων
    dma_put_start(c_mem, c_buf, k*sizeof(datatype), tag);
                                          // έναρξη μεταφοράς DMA αποτελέσματος
    update(a_mem, b_mem, c_mem);          // ενημέρωση των δεικτών στη μνήμη
}
dma_wait_for_transfer(tag);               // αναμονή για ολοκλήρωση της μεταφοράς
```

Το αντίστοιχο πρόγραμμα με χρήση αυτή τη φορά double buffering είναι το ακόλουθο:

```
/*
 * a_mem: θέση μνήμης από όπου μεταφέρονται τα στοιχεία του διανύσματος a
 * b_mem: θέση μνήμης από όπου μεταφέρονται τα στοιχεία του διανύσματος b
 * c_mem: θέση μνήμης όπου μεταφέρονται τα στοιχεία του διανύσματος c (αποτέλεσμα)
 * a_bufs: πίνακας δύο ενταμιευτών για το τμήμα του διανύσματος a
 * b_bufs: πίνακας δύο ενταμιευτών για το τμήμα του διανύσματος b
 * c_bufs: πίνακας δύο ενταμιευτών για το τμήμα του διανύσματος c
 * k: μέγεθος ενταμιευτών (σε στοιχεία)
 * tags: ετικέτες που αντιστοιχούν στις μεταφορές DMA
 * i: τρέχων δείκτης ενταμιευτή σε χρήση
```

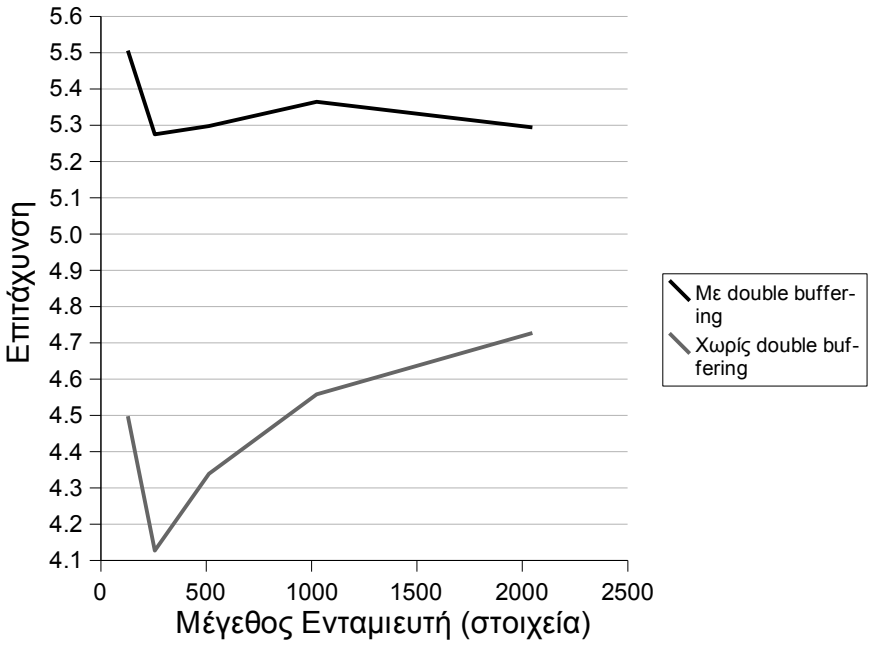
```
 * loops: αριθμός επαναλήψεων
 * rank: αριθμός συνεπεξεργαστή
 * p: πλήθος συνεπεξεργαστών
 * n: πλήθος στοιχείων διανυσμάτων
 * datatype: τύπος στοιχείων των διανυσμάτων
 */

//αρχικοποίηση - υπολογισμός πλήθους επαναλήψεων και τροποποίηση αρχικών δεικτών
initialize(loops, a_mem, b_mem, c_mem);

i = 0;
// έναρξη μεταφορών DMA
dma_get_start(a_mem, a_bufs[i], k*sizeof(datatype), tags[i]);
dma_get_start(b_mem, b_bufs[i], k*sizeof(datatype), tags[i]);

while (loops--){
    // έναρξη μεταφορών DMA
    dma_get_start(a_mem, a_bufs[i^1], k*sizeof(datatype), tags[i^1]);
    dma_get_start(b_mem, b_bufs[i^1], k*sizeof(datatype), tags[i^1]);
    dma_wait_for_transfer(tags[i]);       // αναμονή για ολοκλήρωση των μεταφορών
    c_bufs[i] = add(a_buffs[i], b_buffs[i]);          // πρόσθεση των δεδομένων
    // έναρξη μεταφοράς DMA αποτελέσματος
    dma_put_start(c_mem, c_bufs[i], k*sizeof(datatype), tags[i]);
    update(a_mem, b_mem, c_mem);          // ενημέρωση των δεικτών στη μνήμη
    i ^= 1;                               // ενημέρωση δείκτη
}
dma_wait_for_transfer(tags[i]);           // αναμονή για ολοκλήρωση των μεταφορών
c_bufs[i] = add(a_buffs[i], b_buffs[i]);  // πρόσθεση των δεδομένων
// έναρξη μεταφοράς DMA αποτελέσματος
dma_put_start(c_mem, c_bufs[i], k*sizeof(datatype), tags[i]);
dma_wait_for_transfer(tags[i]);           // αναμονή για ολοκλήρωση της μεταφοράς
```

4.3.4 Μετρήσεις

Με τη βοήθεια του προγράμματος που υλοποιήθηκε, ελήφθησαν στο Playstation 3 μετρήσεις του συνολικού χρόνου εκτέλεσης του προγράμματος για την περίπτωση που οι υπολογισμοί γίνονται σειριακά στο κεντρικό στοιχείο επεξεργασίας (PPE) και για την περίπτωση που χρησιμοποιούνται 1, 2, 3, 4, 5 ή 6 συνεπεξεργαστές, για διάφορα μεγέθη των διανυσμάτων και διάφορα μεγέθη ενταμιευτών, για πράξεις σε ακέραιους (int – 4 bytes) και για πράξεις σε αριθμούς κινητής υποδιαστολής διπλής ακρίβειας (double – 8 bytes). Σε κάθε περίπτωση υπολογίστηκε η επιτάχυνση (speedup), η οποία ισούται με το λόγο του χρόνου εκτέλεσης όταν χρησιμοποιείται ένας συνεπεξεργαστής προς το χρόνο εκτέλεσης όταν χρησιμοποιούνται περισσότεροι από ένας συνεπεξεργαστές. Οι μετρήσεις έγιναν με τη βοήθεια του Timebase Register στο κεντρικό στοιχείο επεξεργασίας (PPE) και των SPE Decrementers στους συνεπεξεργαστές.

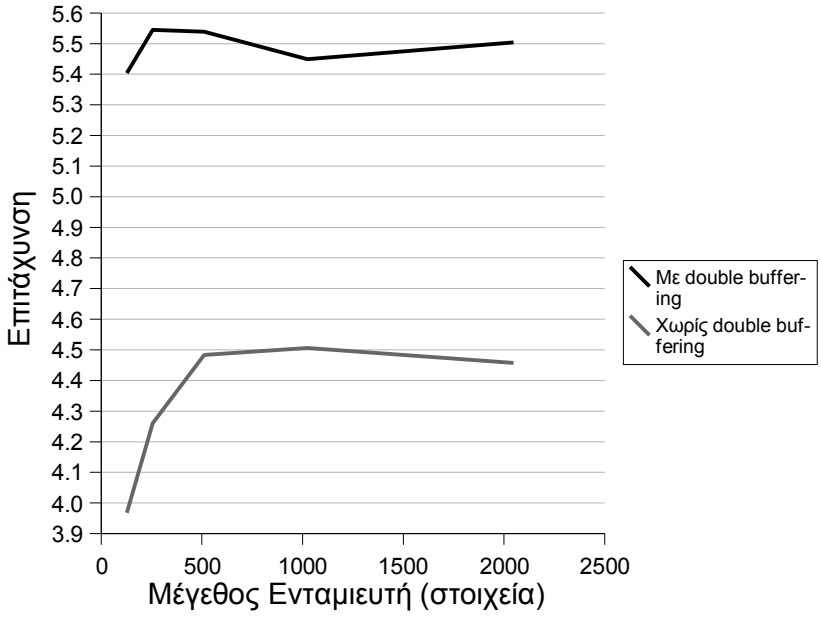
Αρχικά παρουσιάζονται κάποια διαγράμματα στα οποία φαίνεται η εξάρτηση της επιτάχυνσης από το μέγεθος του ενταμιευτή. Οι μετρήσεις αφορούν διανύσματα μεγέθους 4194304 (2^{22}) στοιχείων και

υπολογισμούς με 6 συνεπεξεργαστές.



Επιτάχυνση συναρτήσει του μεγέθους των ενταμιευτών

(μετρήσεις για διανύσματα με 2^{22} ακέραια (int) στοιχεία και χρήση 6 συνεπεξεργαστών)

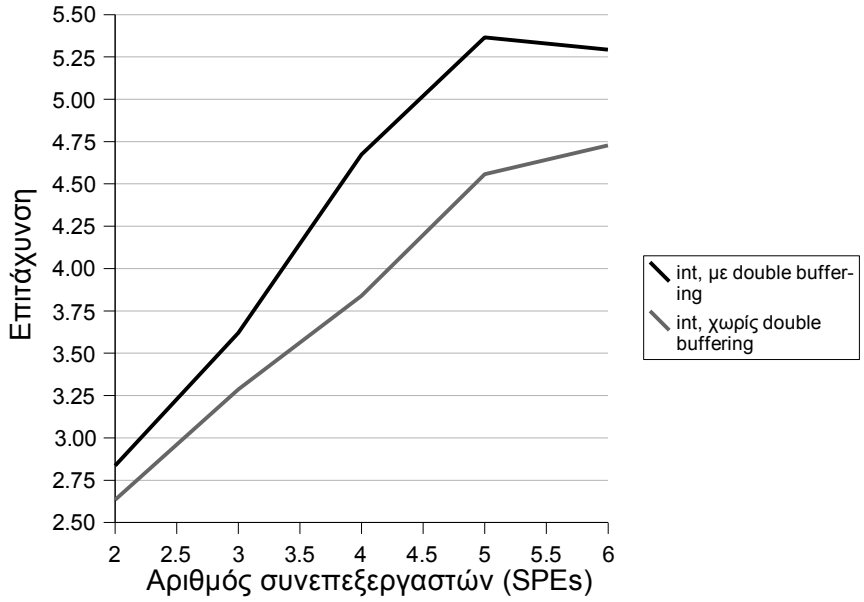


Επιτάχυνση συναρτήσει του μεγέθους των ενταμιευτών

(μετρήσεις για διανύσματα με 2^{22} στοιχεία κινητής υποδιαστολής διπλής ακρίβειας (double)

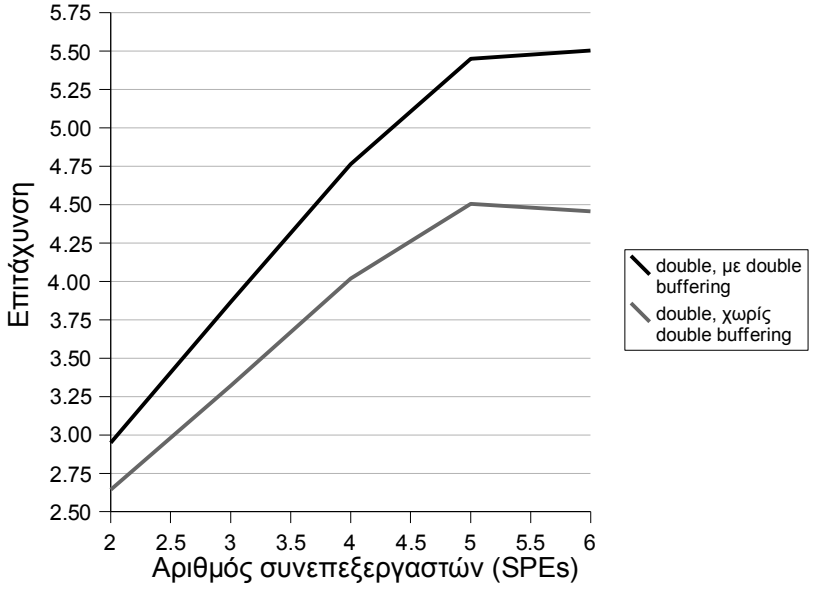
και χρήση 6 συνεπεξεργαστών)

Στα επόμενα διαγράμματα φαίνεται η εξάρτηση της επιτάχυνσης από τον αριθμό των συνεπεξεργαστών. Οι μετρήσεις αφορούν διανύσματα μεγέθους 4194304 (2^{22}) στοιχείων και ενταμιευτές μεγέθους 2048 στοιχείων.



Επιτάχυνση συναρτήσει του αριθμού των συνεπεξεργαστών

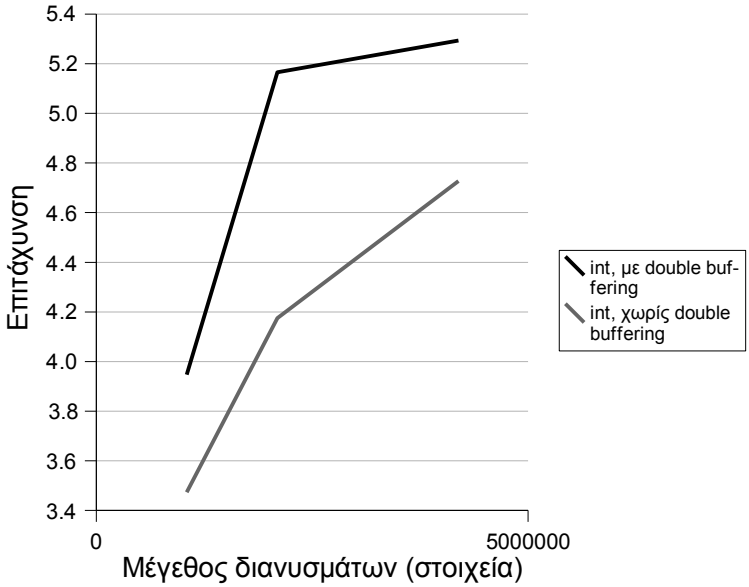
(μετρήσεις με διανύσματα μεγέθους 2^{22} ακεραίων στοιχείων, ενταμιευτές μεγέθους 2048 στοιχείων)



Επιτάχυνση συναρτήσει του αριθμού των συνεπεξεργαστών

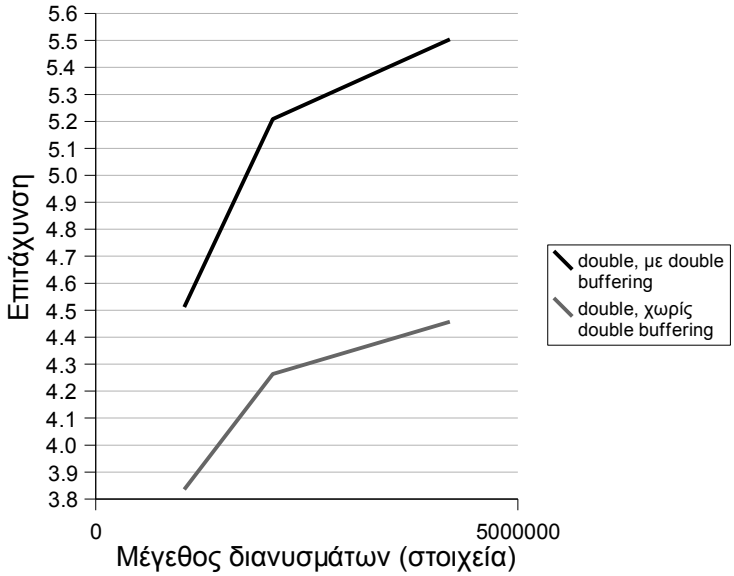
(μετρήσεις με διανύσματα μεγέθους 2^{22} στοιχείων κινητής υποδιαστολής διπλής ακρίβειας, ενταμιευτές μεγέθους 2048 στοιχείων)

Η επιτάχυνση εξαρτάται επίσης και από το μέγεθος των διανυσματων, όπως φαίνεται στα ακόλουθα διαγράμματα:



Επιτάχυνση συναρτήσει του μεγέθους των διανυσμάτων, για πράξεις σε ακέραιους αριθμούς

(μετρήσεις με χρήση 6 συνεπεξεργαστών και ενταμιευτές 2048 στοιχείων)

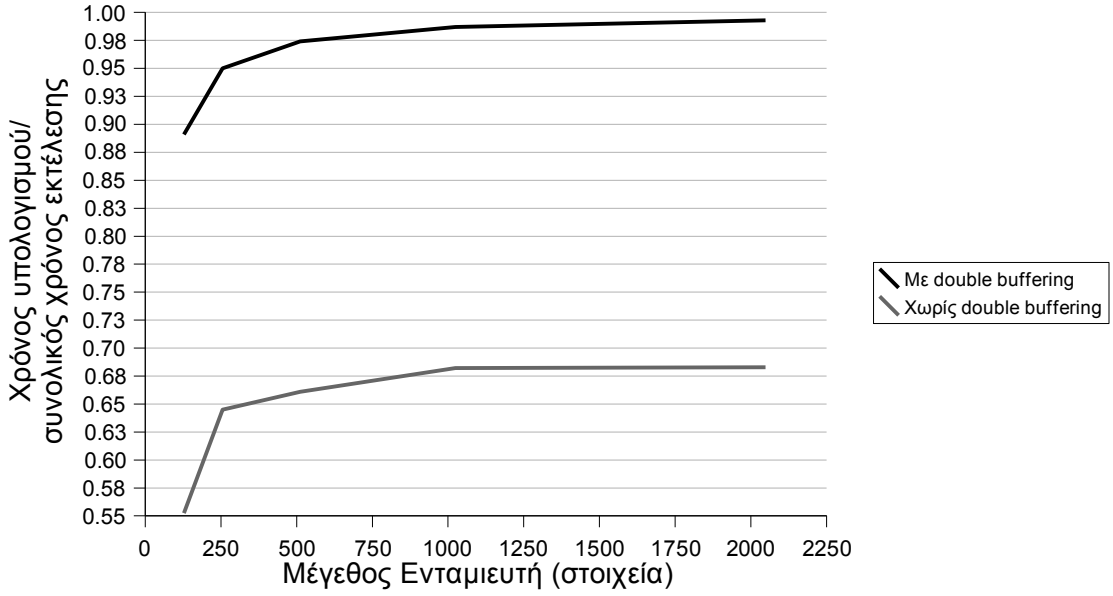


Επιτάχυνση συναρτήσει του μεγέθους των διανυσμάτων,

για πράξεις σε αριθμούς κινητής υποδιαστολής διπλής ακρίβειας

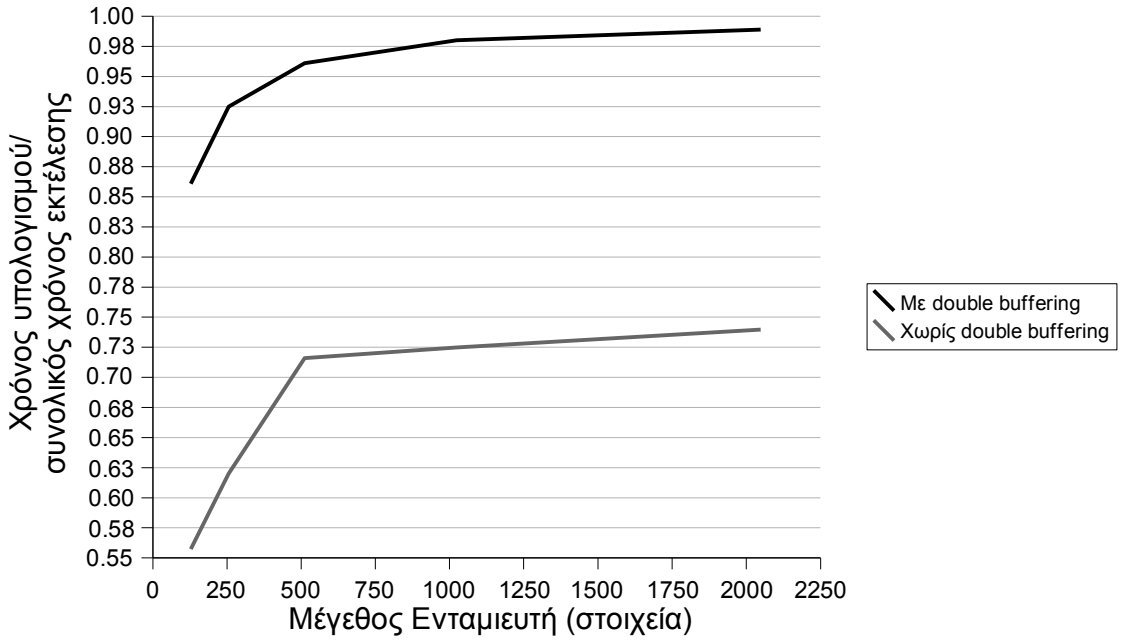
(μετρήσεις με χρήση 6 συνεπξεργαστών και ενταμιευτές 2048 στοιχείων)

Όπως είναι αναμενόμενο, παρατηρείται μεγαλύτερη επιτάχυνση όταν χρησιμοποιείται μεγαλύτερος αριθμός επεξεργαστών. Επίσης, για μεγαλύτερο μέγεθος διανυσμάτων παρατηρείται επίσης μεγαλύτερη επιτάχυνση, επομένως όσο πιο απαιτητική είναι η εφαρμογή ως προς τον όγκο των δεδομένων τόσο περισσότερο επωφελείται από τη χρήση παραλληλίας. Επίσης σε κάθε περίπτωση παρατηρείται σημαντικά μεγαλύτερη επιτάχυνση στην περίπτωση χρήσης double buffering, καθώς ο χρόνος που χρειάζεται για τη μεταφορά των δεδομένων από την κεντρική μνήμη στην τοπική μνήμη (η οποία κατά κάποιο τρόπο λειτουργεί ως μια μεγάλη κρυφή μνήμη) αποκρύπτεται σε μεγάλο βαθμό, με την επιτυχή επικάλυψη μεταφοράς δεδομένων και υπολογισμών. Η αιτία της βελτιωμένης επιτάχυνσης μπορεί να γίνει σαφέστερη με την μέτρηση του λόγου του χρόνου εκτέλεσης υπολογισμών προς το συνολικό χρόνο εκτέλεσης. Οι μετρήσεις αφορούν διανύσματα 4194304 (2^{22}) στοιχείων, ενταμιευτές 2048 στοιχείων και υπολογισμούς που έγιναν με 6 συνεπεξεργαστές. Τα σχετικά αποτελέσματα παρουσιάζονται στα διαγράμματα που ακολουθούν:



Χρόνος υπολογισμού/χρόνος συνολικής εκτέλεσης

(μετρήσεις με διανύσματα μεγέθους 2^{22} στοιχείων κινητής υποδιαστολής διπλής ακρίβειας, ενταμιευτές μεγέθους 2048 στοιχείων και χρήση 6 συνεπεξεργαστών)



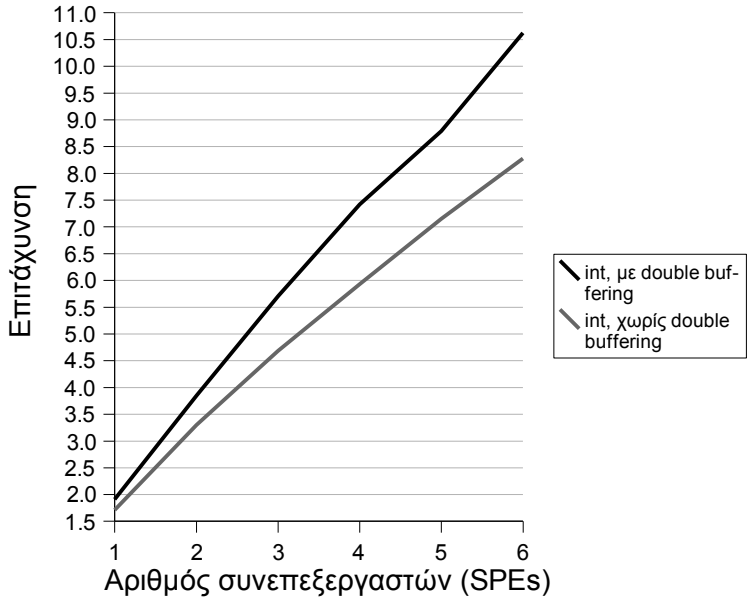
Χρόνος υπολογισμού/χρόνος συνολικής εκτέλεσης

(μετρήσεις με διανύσματα μεγέθους 2^{22} ακέραιων στοιχείων,

ενταμιευτές μεγέθους 2048 στοιχείων και χρήση 6 συνεπεξεργαστών)

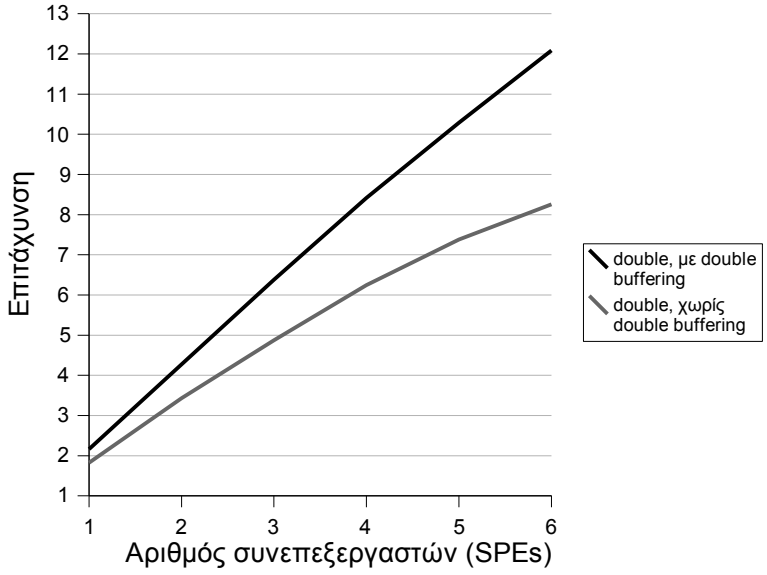
Συγκεκριμένα ο λόγος του χρόνου εκτέλεσης υπολογισμών προς το συνολικό χρόνο εκτέλεσης στην περίπτωση χρήσης double buffering είναι αρκετά υψηλότερος από ό,τι στην απλή περίπτωση (χωρίς double buffering), δηλαδή επιτυγχάνεται ο σκοπός της χρήσης του double buffering, η επικάλυψη μεταφοράς δεδομένων και υπολογισμών, που επιφέρει μείωση του συνολικού χρόνου εκτέλεσης και αύξηση της επιτάχυνσης (speedup) που παρατηρείται.

Τέλος, μετρήθηκε και η επιτάχυνση ως προς το κεντρικό στοιχείο επεξεργασίας, δηλαδή ο λόγος του χρόνου εκτέλεσης όταν οι υπολογισμοί γίνονται σειριακά στο κεντρικό στοιχείο επεξεργασίας προς το χρόνο εκτέλεσης όταν χρησιμοποιούνται οι συνεπεξεργαστές.



Επιτάχυνση ως προς το κεντρικό στοιχείο επεξεργασίας (PPE)

για ακέραιους αριθμούς, διανύσματα μεγέθους 2^{22} στοιχεία, ενταμιευτές μεγέθους 2048 στοιχείων



Επιτάχυνση ως προς το κεντρικό στοιχείο επεξεργασίας (PPE)

για πράξεις σε αριθμούς κινητής υποδιαστολής διπλής ακρίβειας,

διανύσματα μεγέθους 2^{22} στοιχείων, ενταμιευτές μεγέθους 2048 στοιχείων

Σχεδόν σε όλες τις περιπτώσεις παρατηρείται υπεργραμμική επιτάχυνση. Ο μεγάλος όγκος των δεδομένων έχει ως αποτέλεσμα να μην είναι δυνατή η ταυτόχρονη παρουσία του συνόλου τους στην κρυφή μνήμη της κεντρικής μονάδας επεξεργασίας, έτσι ώστε οι ποινές αστοχίας (miss penalties) έχουν τελικά σημαντική αρνητική επίδραση στο συνολικό χρόνο εκτέλεσης, στην περίπτωση της σειριακής εκτέλεσης των υπολογισμών. Επίσης οι αρχιτεκτονικές διαφορές επηρεάζουν σημαντικά τη διαμόρφωση του χρόνου εκτέλεσης. Είναι εμφανές πώς οι συνεπεξεργαστές έχουν πλεονέκτημα σε υπολογισμούς τέτοιου είδους έναντι της κεντρικής μονάδας επεξεργασίας.

4.4 Μέτρηση της επίδοσης του Cell σε απλή εφαρμογή που χρησιμοποιεί το streaming μοντέλο

4.4.1 Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζεται μία απλή παράλληλη εφαρμογή που χρησιμοποιεί το streaming μοντέλο. Η εφαρμογή αυτή αναλαμβάνει τον υπολογισμό του αθροίσματος ομάδων διανυσμάτων αφενός σειριακά στο κεντρικό στοιχείο επεξεργασίας (PPE) και αφετέρου παράλληλα με τη χρήση των συνεπεξεργαστών σε μοντέλο streaming. Ο πρώτος συνεπεξεργαστής σε κάθε επανάληψη της διαδικασίας μεταφέρει από την κεντρική μνήμη δύο από τα διανύσματα της ομάδας, τα προσθέτει και ειδοποιεί τον επόμενο στη σειρά, ο οποίος μεταφέρει από την κεντρική μνήμη ένα ακόμα από τα διανύσματα, και το άθροισμα που υπολόγισε ο προηγούμενος, τα αθροίζει και ειδοποιεί τον επόμενο και ούτω καθεξής. Ο τελευταίος στη σειρά συνεπεξεργαστής αφού υπολογίσει με αντίστοιχο τρόπο το τελικό άθροισμα, το μεταφέρει στην κεντρική μνήμη. Η διαδικασία επαναλαμβάνεται n φορές, για n ομάδες που αποτελούνται από τόσα διανύσματα, όσα ο αριθμός των συνεπεξεργαστών που χρησιμοποιούνται αυξημένος κατά ένα. Με τον τρόπο αυτό, αν για παράδειγμα χρησιμοποιούνται 6 συνεπεξεργαστές, ενώ ο πρώτος συνεπεξεργαστής έχει ξεκινήσει υπολογισμούς για το άθροισμα της έκτης ομάδας διανυσμάτων, ο δεύτερος εκτελεί υπολογισμούς για την πέμπτη ομάδα διανυσμάτων, ο τρίτος εκτελεί υπολογισμούς για την τέταρτη ομάδα διανυσμάτων, ο τέταρτος για την τρίτη, ο πέμπτος για την δεύτερη και ο έκτος ολοκληρώνει τον υπολογισμό του αθροίσματος της πρώτης ομάδας. Έτσι, τα στάδια των υπολογισμών παρουσιάζουν σημαντική αλληλοεπικάλυψη, με αποτέλεσμα την σημαντική μείωση του συνολικού χρόνου εκτέλεσης.

4.4.2 Ανάλυση της εφαρμογής

Σε κάθε επανάληψη του αλγορίθμου που υλοποιεί η εφαρμογή, προστίθενται τόσα διανύσματα όσα ο αριθμός των συνεπεξεργαστών που χρησιμοποιούνται αυξημένος κατά ένα. Οι πράξεις αφορούν αριθμούς κινητής υποδιαστολής διπλής ακρίβειας (double). Για τις ανάγκες του προγράμματος χρειάστηκε να υπάρχει επικοινωνία μεταξύ των συνεπεξεργαστών, ώστε οι μεταφορές των μερικών αθροισμάτων μεταξύ των τοπικών μνημών να γίνονται με σωστό συγχρονισμό. Ο συγχρονισμός υλοποιήθηκε με αποστολή σημάτων μέσω των Signal Notification Registers. Ο αλγόριθμος επεξηγείται αναλυτικά με τη βοήθεια του ακόλουθου ψευδοκώδικα:

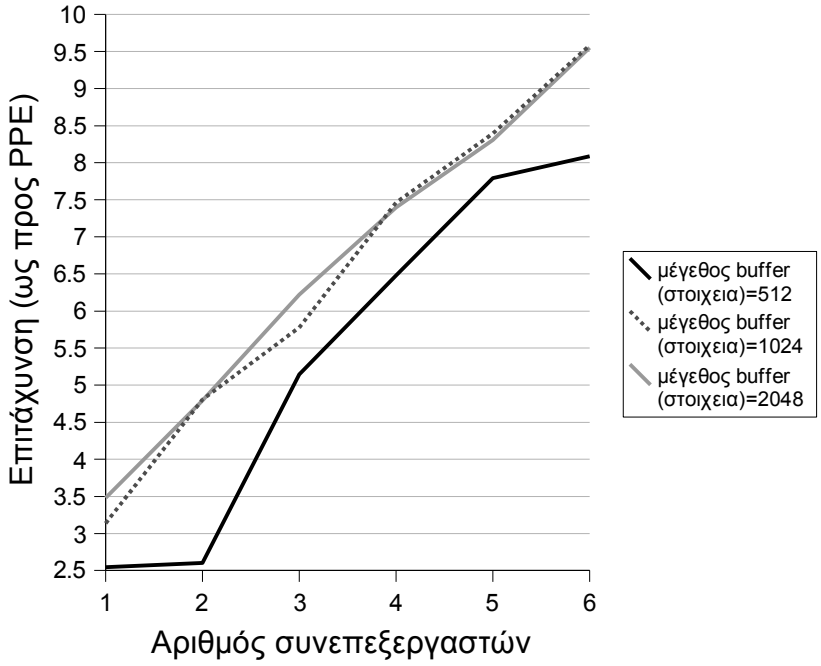
```
/*
 * mem: θέση μνήμης από όπου μεταφέρονται τα δεδομένα
 * result: θέση μνήμης όπου αποθηκεύονται τα αποτελέσματα
 * vec: ενταμιευτής για το διάνυσμα που θα προστεθεί στο μερικό σύνολο
 * prev_sum: ενταμιευτής για το προηγούμενο μερικό σύνολο
 * new_sum: ενταμιευτής για το νέο σύνολο
 * k: μέγεθος διανύσματος (σε στοιχεία)
 * tag: ετικέτα που αντιστοιχεί στη μεταφορά DMA
 * put_tag: ετικέτα που αντιστοιχεί στη μεταφορά του αποτελέσματος στην μνήμη
 * num: συνολικός αριθμός επαναλήψεων
 * loops: αριθμός επαναλήψεων που απομένουν
 * rank: αριθμός συνεπεξεργαστή
 * p: πλήθος συνεπεξεργαστών
 */

//αρχικοποίηση πλήθους επαναλήψεων και τροποποίηση αρχικών δεικτών
loops = num;
initialize(mem);

barrier();

if(rank==0) { // συνεπεξεργαστής 0
  while (--loops){
    dma_get_start(mem, vec, k*sizeof(double), tag);      // έναρξη μεταφοράς DMA
    update(mem);                          // ενημέρωση του δείκτη στη μνήμη
    dma_get_start(mem, prev_sum, k*sizeof(double), tag); // έναρξη μεταφοράς DMA
    update(mem);                          // ενημέρωση του δείκτη στη μνήμη
    dma_wait_for_transfer(tag);           // αναμονή για ολοκλήρωση των μεταφορών
    if (loops<num-1)
      wait_for_signal(next);              // αναμονή σήματος από τον επόμενο ότι
                                          // ολοκλήρωσε τη μεταφορά του prev_sum
                                          // στη δική του τοπική μνήμη
    for(i=0;i<k;++i)
        new_sum[i] = prev_sum[i] + vec[i];            // πρόσθεση των δεδομένων
    send_signal(next);                    // αποστολή σήματος στον επόμενο για να
                                          // ξεκινήσει τη μεταφορά του prev_sum
                                          // στη δική του τοπική μνήμη
  }
} else if (rank==p-1){ // συνεπεξεργαστής p-1
  while (--loops){
    dma_get_start(mem, vec, k*sizeof(double), tag);      // έναρξη μεταφοράς DMA
    update(mem);                          // ενημέρωση του δείκτη στη μνήμη
```

```
    wait_for_signal(prev);                // αναμονή σήματος από τον προηγούμενο
                                          // ότι επιτρέπει τη μεταφορά του prev_sum
                                          // στην εδώ τοπική μνήμη
    dma_get_start(prev_ls+new_sum, prev_sum, k*sizeof(double), tag);
                                          // έναρξη μεταφοράς DMA
    dma_wait_for_transfer(tag);           // αναμονή για ολοκλήρωση των μεταφορών
    if (loops<num-1)
      dma_wait_for_transfer(put_tag);     // αναμονή για ολοκλήρωση της μεταφοράς
                                          // του αποτελέσματος
    if (loops!=0)
      send_signal(prev);                  // ειδοποίηση του προηγούμενου ότι έχει
                                          // ολοκληρωθεί η μεταφορά του prev_sum
    for(i=0;i<k;++i)
        new_sum[i] = prev_sum[i] + vec[i];            // πρόσθεση των δεδομένων
    dma_begin_put(result, new_sum, k*sizeof(double), put_tag);
                                       // έναρξη μεταφοράς DMA του αποτελέσματος
    update(result);                    // ενημέρωση του δείκτη στη μνήμη
  }
  dma_wait_for_transfer(put_tag);      // αναμονή για ολοκλήρωση της μεταφοράς
} else { // υπόλοιποι συνεπεξεργαστές
    dma_get_start(mem, vec, k*sizeof(double), tag);      // έναρξη μεταφοράς DMA
    update(mem);                          // ενημέρωση του δείκτη στη μνήμη
    wait_for_signal(prev);                // αναμονή σήματος από τον προηγούμενο
                                          // ότι επιτρέπει τη μεταφορά του prev_sum
                                          // στην εδώ τοπική μνήμη
    dma_get_start(prev_ls+new_sum, prev_sum, k*sizeof(double), tag);
                                          // έναρξη μεταφοράς DMA
    dma_wait_for_transfer(tag);       // αναμονή για ολοκλήρωση των μεταφορών
    if (loops!=0)
      send_signal(prev);              // ειδοποίηση του προηγούμενου ότι έχει
                                      // ολοκληρωθεί η μεταφορά του prev_sum
    if (loops<num-1)
      wait_for_signal(next);              // αναμονή σήματος από τον επόμενο ότι
                                          // ολοκλήρωσε τη μεταφορά του prev_sum
                                          // στη δική του τοπική μνήμη
    for(i=0;i<k;++i)
        new_sum[i] = prev_sum[i] + vec[i];// πρόσθεση των δεδομένων
    send_signal(next);                    // αποστολή σήματος στον επόμενο για να
                                          // ξεκινήσει τη μεταφορά του prev_sum
                                          // στη δική του τοπική μνήμη
}
barrier();
```

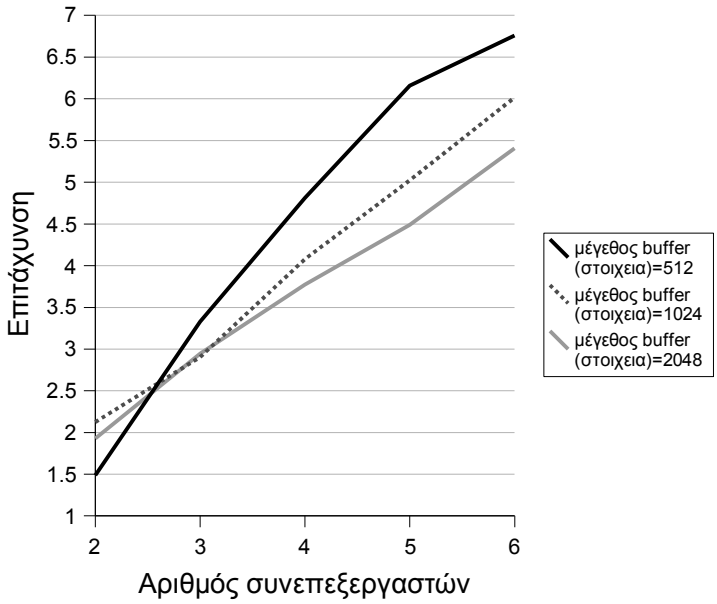
4.4.3 Μετρήσεις

Με τη βοήθεια της απλής αυτής εφαρμογής, ελήφθησαν στο Playstation 3 μετρήσεις του συνολικού χρόνου εκτέλεσης του προγράμματος για την περίπτωση που οι υπολογισμοί γίνονται σειριακά στο κεντρικό στοιχείο επεξεργασίας (PPE) και για την περίπτωση που χρησιμοποιούνται 1, 2, 3, 4, 5 ή 6 συνεπεξεργαστές. Μετρήθηκε η επιτάχυνση (speedup) ως προς το κεντρικό στοιχείο επεξεργασίας (PPE), η οποία ισούται με το λόγο του χρόνου σειριακής εκτέλεσης στο κεντρικό στοιχείο επεξεργασίας προς το χρόνο παράλληλης εκτέλεσης, για διάφορα μεγέθη των διανυσμάτων.

Επιλέχθηκε αριθμός επαναλήψεων αρκετά μεγάλος (1000) ώστε να εμφανίζεται σημαντική επιτάχυνση. Επίσης υπολογίστηκαν τα MFLOPS στην περίπτωση των υπολογισμών με χρήση των συνεπεξεργαστών, διαιρώντας τον αριθμό των πράξεων με τον συνολικό χρόνο εκτέλεσης, και με βάση αυτά υπολογίστηκε η επιτάχυνση ως προς τον έναν συνεπεξεργαστή, κανονικοποιημένη ως προς τον αριθμό των πράξεων, δηλαδή ο λόγος των MFLOPS που επιτυγχάνονται στην περίπτωση χρήσης πολλών συνεπεξεργαστών προς τα MFLOPS που επιτυγχάνονται στην περίπτωση χρήσης ενός συνεπεξεργαστή. Οι μετρήσεις έγιναν με τη βοήθεια του Timebase Register στο κεντρικό στοιχείο επεξεργασίας (PPE). Τα αποτελέσματα των μετρήσεων παρουσιάζονται συνοπτικά στα διαγράμματα που ακολουθούν:



Επιτάχυνση ως προς το κεντρικό στοιχείο επεξεργασίας



Επιτάχυνση ως προς το χρόνο εκτέλεσης σε έναν συνεπεξεργαστή

(κανονικοποιημένο ως προς τον αριθμό των πράξεων)

Στα παραπάνω διαγράμματα είναι εμφανής η βελτίωση της επίδοσης με τη χρήση περισσότερων συνεπεξεργαστών. Παρατηρείται επίσης πως το streaming μοντέλο επιτυγχάνει σημαντική επιτάχυνση, και είναι ιδιαίτερα κατάλληλο για χρήση με τον επεξεργαστή Cell.

Κεφάλαιο 5

Επίλογος

5.1 Περίληψη

Στα πλαίσια της παρούσης εργασίας αρχικά μελετήθηκε η ιδιαίτερη αρχιτεκτονική του επεξεργαστή Cell και ο τρόπος με τον οποίο προγραμματίζονται εφαρμογές ειδικά για αυτόν. Στη συνέχεια, αναπτύχθηκαν διάφορες εφαρμογές με χρήση των οποίων έγιναν μετρήσεις για την επίδοση του επεξεργαστή Cell.

Συγκεκριμένα, σε πρώτη φάση ελήφθησαν μετρήσεις των καθυστερήσεων και του εύρους ζώνης κατά τη διεξαγωγή ασύγχρονων μεταφορών άμεσης πρόσβασης στη μνήμη σύμφωνα με διάφορα σχήματα επικοινωνίας. Στη συνέχεια, αναπτύχθηκε μια απλή παράλληλη εφαρμογή χωρίς επικοινωνία/συγχρονισμό μεταξύ των επεξεργαστικών μονάδων, και διερευνήθηκε η επιτάχυνση συναρτήσει παραμέτρων όπως ο αριθμός των συνεπεξεργαστών που χρησιμοποιούνται και το μέγεθος των μεταφορών που εκτελούνται. Τέλος, μελετήθηκε η επίδοση μιας απλής εφαρμογής που επιδεικνύει τη χρήση του streaming προγραμματιστικού μοντέλου, το οποίο κρίθηκε ιδιαίτερα κατάλληλο για τον επεξεργαστή Cell.

5.2 Συμπεράσματα

Ο επεξεργαστής Cell αποτελεί χαρακτηριστικό παράδειγμα της στροφής της βιομηχανίας των υπολογιστών προς τους πολυπύρηνους επεξεργαστές, στην προσπάθεια να αντιμετωπιστούν τα προβλήματα που καθιστούν δύσκολη την ολοένα και μεγαλύτερη απαίτηση για αύξηση της υπολογιστικής ισχύος. Με τον σχεδιασμό του ώστε να αντιμετωπίζει τα προβλήματα που σχετίζονται με την χρήση της μνήμης, την κατανάλωση ισχύος και τη συχνότητα του επεξεργαστή, επιτυγχάνει επιδόσεις πολύ υψηλότερες από έναν τυπικό σύγχρονο επεξεργαστή.

Από τα αποτελέσματα των μετρήσεων που ελήφθησαν στα πλαίσια της εργασίας, φαίνεται πως ο επεξεργαστής Cell έχει αυξημένες δυνατότητες για ανάπτυξη παράλληλων εφαρμογών επιδόσεων, λόγω των ιδιαίτερων χαρακτηριστικών του (εξειδίκευση των στοιχείων επεξεργασίας, ιεραρχία

μνήμης τριών επιπέδων, ασύγχρονες μεταφορές άμεσης πρόσβασης στη μνήμη, δίαυλος διασύνδεσης υψηλού εύρους ζώνης), τα οποία βεβαίως καθιστούν τον προγραμματισμό του δυσκολότερο από ό,τι ενός τυπικού σύγχρονου επεξεργαστή. Ιδιαίτερα τα σχήματα επικοινωνίας τα οποία αξιοποιούν πλήρως τις δυνατότητες του διαύλου διασύνδεσης, αποφεύγοντας τις συγκρούσεις μεταξύ των μεταφορών, μπορούν να οδηγήσουν σε πολύ καλές επιταχύνσεις του χρόνου εκτέλεσης. Τέλος, μια κατηγορία εφαρμογών που μπορούν να ωφεληθούν από την ιδιαίτερη αρχιτεκτονική του επεξεργαστή Cell είναι οι εφαρμογές που χρησιμοποιούν το streaming προγραμματιστικό μοντέλο.

Βιβλιογραφία

- Introduction to the Cell multiprocessor, J. A. Kahle, M. N. Day, H. P. Hofstee, C. R. Johns, T. R. Maeurer, and D. Shippy, http://www.research.ibm.com/journal/rd/494/kahle.html
- Cell Broadband Engine Architecture,

  http://www-128.ibm.com/developerworks/power/cell/
- Cell Broadband Engine Programming Tutorial,

  http://www-128.ibm.com/developerworks/power/cell/
- Cell Broadband Engine Programming Handbook,

  http://www-128.ibm.com/developerworks/power/cell/
- SPU C/C++ Language Extensions,

  http://www-128.ibm.com/developerworks/power/cell/
- SPE Runtime Management Library 2.2,

  http://www-128.ibm.com/developerworks/power/cell/
- Cell Multiprocessor Communication Network: Built for Speed, Michael Kistler, Michael Perrone, Fabrizio Petrini, IEEE Micro 26(3): 10-23 (2006)
- Cell Broadband Engine Architecture and its first implementation, Thomas Chen, Ram Raghavan, Jason Dale, Eiji Iwata,

  http://www.ibm.com/developerworks/power/library/pa-cellperf/
- Just like being there: Papers from the Fall Processor Forum 2005: Unleashing the Cell Broadband Engine Processor: The Element Interconnect Bus, David Krolak, http://www.ibm.com/developerworks/power/library/pa-fpfeib/