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

Μελέτη και κατασκευή μετροπρογραμμάτων για την αξιολόγηση της επίδοσης υβριδικού πολυπύρηνου επεξεργαστή ειδικής χρήσεως

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γεράσιμος Γ. Πετεινάτος

Γεώργιος Λ. Ρόκος

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

Αθήνα, Ιούλιος 2009

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

Μελέτη και κατασκευή μετροπρογραμμάτων για την αξιολόγηση της επίδοσης υβριδικού πολυνηματικού επεξεργαστή ειδικής χρήσεως

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

των

Γεράσιμου Γ. Πετεινάτου

Γεωργίου Λ. Ρόκου

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

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 23η Ιουλίου 2009

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

Αθήνα, Ιούλιος 2009

Γεράσιμος Γ. Πετεινάτος

Φοιτητής Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Γεώργιος Λ. Ρόκος

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Αθήνα, Ιούλιος 2009

Π Ε Ρ Ι Λ Η Ψ Η

Μελέτη και κατασκευή μετροπρογραμμάτων για την αξιολόγηση της επίδοσης υβριδικού πολυνηματικού επεξεργαστή ειδικής χρήσεως

των Γεράσιμου Γ. Πετεινάτου και Γεωργίου Λ. Ρόκου

Επί κεφαλής τριμελούς εξεταστικής επιτροπής: Νεκτάριος Κοζύρης
Αναπληρωτής Καθηγητής
Τομέας Τεχνολογίας Πληροφορικής
και Υπολογιστών

Η πρώτη γενιά της Cell Broadband Engine (CBE) είναι η πρώτη υλοποίηση μίας νέας οικογένειας μικροεπεξεργαστών, οι οποίοι εναρμονίζονται με την αρχιτεκτονική Cell Broadband Engine Architecture (CBEA). Η CBEA είναι μία νέα αρχιτεκτονική που επεκτείνει την 64-bit αρχιτεκτονική PowerPC. Οι επεξεργαστές αυτής της οικογένειας ενδείκνυνται τόσο για εφαρμογές επιστημονικού ενδιαφέροντος όσο και για χρήση σε ένα ευρύτερο σύνολο εμπορικών/καταναλωτικών συσκευών (τηλεοράσεις HD, κονσόλες παιχνιδιών, πολυμεσικές εφαρμογές κ.τ.λ.).

Ο επεξεργαστής CBE είναι ένας υβριδικός, πολυπύρηνος επεξεργαστής ειδικής χρήσεως αποτελούμενος από ένα κεντρικό επεξεργαστικό στοιχείο και οκτώ μαθηματικούς συνεπεξεργαστές. Προσφέρει αυξημένες δυνατότητες για ανάπτυξη και εκτέλεση εφαρμογών υψηλής επίδοσης και διαφοροποιείται από τους συμβατικούς επεξεργαστές λόγω κάποιων ιδιαίτερων χαρακτηριστικών των συνεπεξεργαστών, όπως οι δύο ετερογενείς σωληνώσεις, η απλή αρχιτεκτονική τους, η χρήση διανυσματικών εντολών και τύπων δεδομένων και οι ελεγχόμενες από το λογισμικό τοπικές μνήμες τους, οι οποίες συνδέονται σε ένα υψηλού εύρους ζώνης δίαυλο διασύνδεσης στοιχείων και κάνουν εφικτή την ταχεία και ασύγχρονη DMA μεταφορά δεδομένων.

Η παρούσα διπλωματική εργασία αποτελεί μία προσπάθεια υλοποίησης κάποιων συγκεκριμένων εφαρμογών επιστημονικού ενδιαφέροντος στην πλατφόρμα της CBEA, αντιμετωπίζοντας και αξιοποιώντας τα ιδιαίτερα χαρακτηριστικά της, ιδιαίτερα τη χρήση διανυσματικών πράξεων και τύπων δεδομένων. Συγκεκριμένα, μελετώνται τα προβλήματα του πολλαπλασιασμού πινάκων και της εξίσωσης διάχυσης σε δύο διαστάσεις. Διερευνάται ο τρόπος υλοποίησης και η μέτρηση της επίδοσης των συγκεκριμένων εφαρμογών τόσο σε επίπεδο μίας CBE όσο και σε επίπεδο συστοιχίας από CBEs. Μέσα από αυτή τη διαδικασία αναδεικνύονται οι προγραμματιστικές τεχνικές και οι βελτιστοποιήσεις που οδηγούν στη μεγιστοποίηση της επίδοσης για αυτές τις εφαρμογές.

Η εφαρμογή του πολλαπλασιασμού πινάκων, ενός προβλήματος με μεγάλη υπολογιστική πολυπλοκότητα και χωρίς εξαρτήσεις δεδομένων, παραλληλοποιείται εύκολα και επιτυγχάνει υψηλή επίδοση τόσο στο πλαίσιο μίας CBE όσο και στο πλαίσιο συστοιχίας. Αντίθετα, η εφαρμογή της εξίσωσης διάχυσης παρουσιάζει έντονες εξαρτήσεις δεδομένων, με αποτέλεσμα τον περιορισμό της επίδοσης και τη μη αποδοτική κλιμάκωση (scaling) σε επίπεδο συστοιχίας. Σε κάθε περίπτωση, δεδομένης της αδυναμίας των σύγχρονων compilers για αποδοτική παραλληλοποίηση και διανυσματοποίηση αλγορίθμων, η επίτευξη υψηλής επίδοσης από τον επεξεργαστή προϋποθέτει τη χειροκίνητη βελτιστοποίηση του κώδικα από μέρους του προγραμματιστή.

Λέξεις Κλειδιά: πολλαπλασιασμός πινάκων, εξίσωση διάχυσης, Cell Broadband Engine, διανυσματοποίηση αλγορίθμου, παραλληλοποίηση αλγορίθμου, μεταφορές DMA, συστοιχία από CBEs.

A B S T R A C T

Study and creation of benchmark applications for the performance evaluation of a hybrid, multithreaded, purpose-specific processor

by Gerasimos G. Peteinatos and Georgios L. Rokos

Chairperson of the Supervisory Committee: Nectarios Koziris
Associate Professor
Department of
Computer Science

The first generation Cell Broadband Engine (CBE) is the first incarnation of a new family of microprocessors conforming to the Cell Broadband Engine Architecture (CBEA). The CBEA is a new architecture that extends the 64-bit PowerPC Architecture. Microprocessors of this family are intended for scientific applications as well as for use in a wider range of commercial/consumer devices (HD televisions, gaming consoles, multimedia-rich applications etc.).

The CBE is a hybrid, multi-core, special-purpose processor which consists of one main processing element and eight mathematical co-processors. It provides increased potentials for high performance application development and execution and differentiates from conventional processors due to some special characteristics of its co-processors, such as the two heterogeneous pipelines, their simplified architecture, the use of vector operations and data types and their software controlled local stores (memories), which are bound to a high-bandwidth element interconnect bus and enable the fast and asynchronous DMA data transfer.

The present diploma thesis comprises an effort to port some specific applications of scientific interest to the CBEA platform, utilizing and dealing with its special characteristics, particularly the use of vector operations and data types. More specifically, there is an effort to implement the problems of matrix multiplication and the two-dimensional diffusion equation, mainly on the purpose of looking into the way of producing source code for these applications and measuring their performance both at single CBE level as well as cluster-of-CBEs level. This process brings out the programming techniques and optimizations which lead to maximal performance for these applications.

The application of matrix multiplication, a problem of high computational complexity and without data dependencies, is easily parallelized and achieves high performance both at single-CBE and cluster level. To the contrary, the application of the diffusion equation reveals intense data dependencies, which leads to limited performance

and inefficient scaling to cluster architectures. In both cases, due to the de facto incapability of modern compilers to perform effective algorithm parallelization and vectorization, achieving high computational performance requires manual optimization of the source code by the programmer.

Keywords: matrix multiplication, diffusion equation, Cell Broadband Engine, algorithm vectorization, algorithm parallelization, DMA transfers, CBE cluster.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Ε Υ Χ Α Ρ Ι Σ Τ Ι Ε Σ

Οι συγγραφείς επιθυμούν να εκφράσουν την ειλικρινή εκτίμησή τους προς τον Αναπληρωτή Καθηγητή Νεκτάριο Κοζύρη για την ανάθεση του θέματος και τη βοήθειά του στην προετοιμασία αυτού του εγχειριδίου. Επίσης, εκφράζεται ευγνωμοσύνη προς τον Αναπληρωτή Καθηγητή Κωνσταντίνο Σαγώνα και τον Επίκουρο Καθηγητή Νικόλαο Παπασπύρου για το χρόνο που αφιέρωσαν, τη διόρθωση της μελέτης και τις χρήσιμες υποδείξεις τους. Επιπρόσθετα, απευθύνονται ειδικές ευχαριστίες προς το Μεταδιδακτορικό Ερευνητή Γεώργιο Γκούμα και τον Υποψήφιο Διδάκτορα Κορνήλιο Κούρτη για τη βοήθεια, την καθοδήγηση και την ενασχόλησή τους με το θέμα, την ανάγνωση της παρούσας μελέτης και τη διόρθωσή της με εύστοχες παρατηρήσεις. Τέλος, οι συγγραφείς ευχαριστούν θερμά την Υποψήφια Διδάκτορα Γεωργία Κουβέλη για τις συμβουλές, τη συμπαράσταση και τις χρήσιμες υποδείξεις καθ' όλη τη διάρκεια πραγματοποίησης της.

Κεφάλαιο 1

ΕΙΣΑΓΩΓΗ

Η πρώτη γενιά της Cell Broadband Engine είναι η πρώτη υλοποίηση μίας νέας οικογένειας μικροεπεξεργαστών, οι οποίοι εναρμονίζονται με την αρχιτεκτονική Cell Broadband Engine Architecture (CBEA). Η CBEA είναι μία νέα αρχιτεκτονική που επεκτείνει την 64-bit αρχιτεκτονική PowerPC. Οι CBEA και Cell Broadband Engine είναι το αποτέλεσμα της συνεργασίας μεταξύ Sony, Toshiba και IBM, γνωστής και ως STI, η οποία ξεκίνησε επίσημα στις αρχές του 2001.

Υπόβαθρο και κίνητρα

Παρ' όλο που η Cell Broadband Engine προορίζεται πρωτίστως για εφαρμογές σε κονσόλες παιχνιδιών και σε καταναλωτικές συσκευές πολυμέσων όπως τηλεοράσεις υψηλής ανάλυσης (High Definition), η αρχιτεκτονική και η υλοποίηση της Cell Broadband Engine έχουν σχεδιαστεί ώστε να προσφέρουν θεμελιώδεις προόδους στις επιδόσεις του επεξεργαστή. Οραματίζεται μία πολύ ευρύτερη χρήση αυτής της αρχιτεκτονικής.

Η Cell Broadband Engine είναι ένας πολυεπεξεργαστής, ολοκληρωμένος σε ένα τσιπ, με εννέα επεξεργαστές που δουλεύουν χρησιμοποιώντας μια μοιραζόμενη, συναφή μνήμη. Υπό αυτή την έννοια επεκτείνει τρέχουσες τάσεις στους επεξεργαστές οικιακής και server χρήσης. Το πιο διακεκριμένο χαρακτηριστικό της Cell Broadband Engine είναι ότι, παρόλο που οι εννέα επεξεργαστές μοιράζονται τον κύριο αποθηκευτικό χώρο (το χώρο διευθύνσεων που περιλαμβάνει την κύρια μνήμη), η λειτουργία τους εξειδικεύεται σε δύο τύπους:

- Τον κύριο επεξεργαστή, το PowerPC Processor Element (PPE)
- Το μαθηματικό συνεπεξεργαστή, το Synergistic Processor Element (SPE)

Η Cell Broadband Engine περιλαμβάνει:

- Ένα PPE
- Οκτώ SPEs

Το PPE (ο πρώτος τύπος επεξεργαστικού στοιχείου) είναι ένας πυρήνας αρχιτεκτονικής PowerPC 64-bit. Ακολουθεί πλήρως το πρότυπο της 64-bit PowerPC αρχιτεκτονικής και μπορεί να τρέξει 32-bit και 64-bit λειτουργικά συστήματα και εφαρμογές.

Το SPE (ο δεύτερος τύπος επεξεργαστικού στοιχείου) είναι βελτιστοποιημένο για την εκτέλεση εφαρμογών πλούσιων σε αριθμητικές πράξεις και δεν είναι αποδοτικό στην εκτέλεση ενός λειτουργικού συστήματος. Τα SPEs είναι ανεξάρτητοι επεξεργαστές, όπου ο καθένας τρέχει τα δικά του προγράμματα. Κάθε SPE έχει πλήρη πρόσβαση στη συναφή, μοιραζόμενη μνήμη, στην οποία περιλαμβάνεται και απεικονιζόμενες στη μνήμη διευθύνσεις εισόδου-εξόδου (memory mapped I/O space).

Ο χαρακτηρισμός του SPE ως *συνεργατικό* δηλώνει την αμοιβαία εξάρτηση μεταξύ του PPE και των SPEs. Τα SPEs εξαρτώνται από το PPE για την εκτέλεση του λειτουργικού συστήματος και, σε πολλές περιπτώσεις, του κύριου νήματος της εφαρμογής. Το PPE εξαρτάται από τα SPEs, τα οποία παρέχουν την υποδομή για την υψηλή απόδοση της εφαρμογής.

Τα SPEs έχουν σχεδιαστεί ώστε να προγραμματίζονται σε γλώσσες υψηλού επιπέδου και να υποστηρίζουν ένα πλούσιο σύνολο εντολών που περιλαμβάνει εκτεταμένες λειτουργίες SIMD (Single Instruction Multiple Data = Πολλαπλά Δεδομένα με Μία Εντολή). Η χρήση των SIMD δεδομένων, όπως και στους συμβατικούς επεξεργαστές με SIMD προεκτάσεις, είναι προτεινόμενη, όχι υποχρεωτική. Το PPE, ως επεξεργαστής αρχιτεκτονικής PowerPC, υποστηρίζει και αυτό SIMD επεκτάσεις.

Για έναν προγραμματιστή η Cell Broadband Engine μοιάζει με έναν επεξεργαστή εννέα πυρήνων. Το PPE είναι περισσότερο βελτιστοποιημένο για διεργασίες με περίπλοκη ροή ελέγχου και για εναλλαγή διεργασιών (task switching). Τα SPEs από την άλλη είναι περισσότερο βελτιστοποιημένα για εργασίες με πολλούς αριθμητικούς υπολογισμούς, ενώ δεν αποδίδουν πολύ καλά στην εναλλαγή διεργασιών. Άσχετα από την απόδοση, τα δύο είδη επεξεργαστικών στοιχείων μπορούν να εκτελέσουν οποιαδήποτε λειτουργία. Αυτή η εξειδίκευση επιτρέπει αυξημένη απόδοση στην υλοποίηση και του PPE και πολύ περισσότερο των SPEs. Είναι ένα καθοριστικός παράγων για την κατά τάξεως μεγέθους βελτίωση της μέγιστης επιτεύξιμης υπολογιστικής επίδοσης και της χωρικής/ενεργειακής απόδοσης που επιτυγχάνει η Cell Broadband Engine σε σχέση με άλλους συμβατικούς επεξεργαστές οικιακής χρήσεως.

Μία σημαντική διαφορά μεταξύ του PPE και των SPEs είναι ο τρόπος πρόσβασης στη μνήμη:

- Το PPE προσπελαύνει τον κύριο αποθηκευτικό χώρο με εντολές φόρτωσης (load) και αποθήκευσης (store) που μεταφέρουν δεδομένα μεταξύ των καταχωρητών (register file) και της κύριας μνήμης (η οποία μπορεί να έχει αντίγραφο στην cache του επεξεργαστή).

- Τα SPEs προσπελαύνουν την κύριο αποθηκευτικό χώρο με εντολές άμεσης πρόσβασης στη μνήμη (Direct Memory Access - DMA) οι οποίες μεταφέρουν δεδομένα μεταξύ της κύριας μνήμης και μίας «ιδιωτικής», τοπικής μνήμης (local store) που χρησιμοποιείται για την αποθήκευση τόσο των δεδομένων όσο και των εντολών του προγράμματος. Οι εντολές φόρτωσης και αποθήκευσης του SPE προσπελαύνουν αυτήν την τοπική μνήμη αντί για την κύρια. Αυτή η οργάνωση τριών επιπέδων του αποθηκευτικού χώρου (καταχωρητές, τοπική μνήμη, κύρια μνήμη), με τις ασύγχρονες DMA μεταφορές μεταξύ κύριας και τοπικής μνήμης, είναι μία ριζική αλλαγή σε σχέση με τα συμβατικά μοντέλα αρχιτεκτονικών και προγραμματισμού επειδή παραλληλοποιεί ρητά το υπολογιστικό κομμάτι της εφαρμογής και τις μεταφορές δεδομένων και εντολών.

Ο λόγος για αυτήν τη ριζική αλλαγή είναι ότι η καθυστέρηση της μνήμης, μετρούμενη σε επεξεργαστικούς κύκλους, έχει αυξηθεί κατά εκατοντάδες φορές στα τελευταία είκοσι χρόνια. Το αποτέλεσμα είναι ότι η επίδοση των εφαρμογών στις περισσότερες περιπτώσεις περιορίζεται κατά κύριο λόγο από την καθυστέρηση της μνήμης παρά από τα όρια της υπολογιστικής ισχύος ή του εύρους ζώνης. Όταν ένα σειριακό πρόγραμμα σε μία συμβατική αρχιτεκτονική εκτελεί μία εντολή φόρτωσης η οποία αποτυγχάνει στις caches η εκτέλεση του προγράμματος διακόπτεται για αρκετές εκατοντάδες κύκλων ρολογιού. Συγκρινόμενοι με αυτήν την «ποινή» οι λίγοι κύκλοι που χρειάζονται για την αρχικοποίηση μίας μεταφοράς DMA για ένα SPE είναι σχετικά αμελητέοι. Οι συμβατικοί επεξεργαστές, ακόμα και αυτοί με βαθιά και δαπανηρή πρόβλεψη (speculation), μπορούν στην καλύτερη περίπτωση να έχουν κάποιες λίγες ανεξάρτητες προσβάσεις στη μνήμη ταυτόχρονα σε εξέλιξη. Το μοντέλο των ρητών DMA μεταφορών επιτρέπει σε κάθε SPE να έχει πολλές ταυτόχρονες προσβάσεις σε εξέλιξη, χωρίς την ανάγκη για πρόβλεψη.

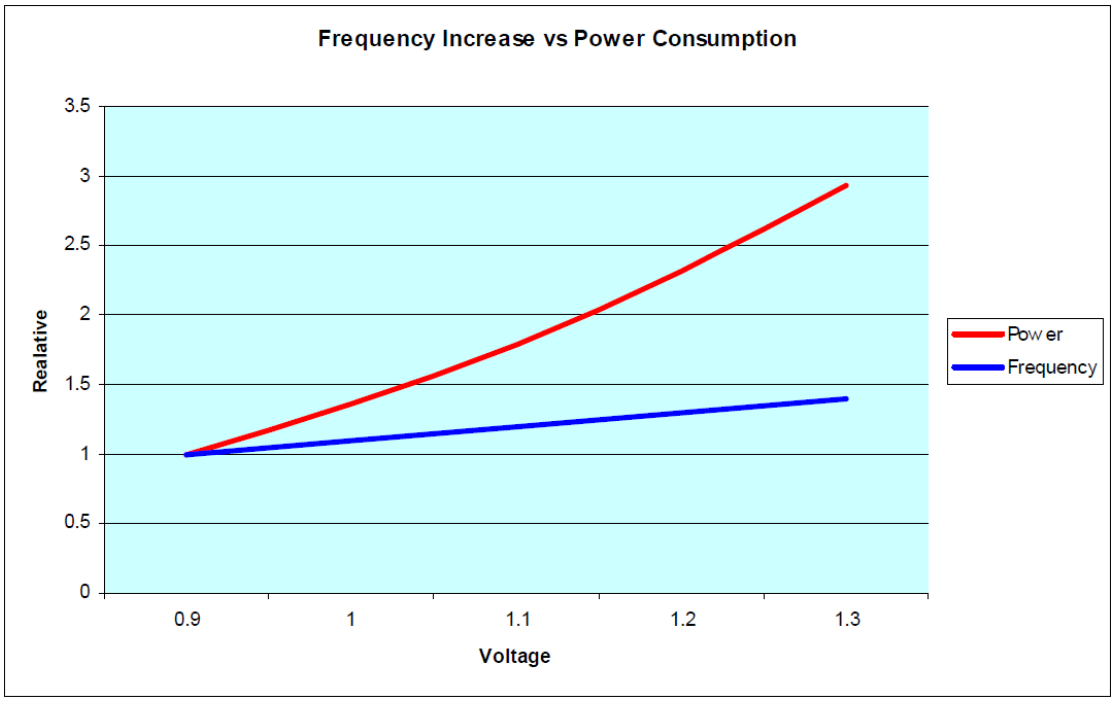
Η πιο αποδοτική αξιοποίηση του παραπάνω χαρακτηριστικού είναι το μοντέλο κατά το οποίο ένα σύνολο από εντολές για DMA μεταφορές κατασκευάζεται στην τοπική μνήμη του SPE, έτσι ώστε ο ελεγκτής DMA μπορεί να επεξεργάζεται αυτό το σύνολο ασύγχρονα ενώ το SPE παράλληλα πραγματοποιεί υπολογισμούς πάνω σε δεδομένα που έχουν έρθει από προηγούμενη DMA μεταφορά. Σε πολλές περιπτώσεις αυτή η νέα προσέγγιση στον τρόπο πρόσβασης στη μνήμη έχει ως αποτέλεσμα η επίδοση των εφαρμογών να ξεπερνά αυτή των συμβατικών επεξεργαστών κατά σχεδόν δύο τάξεις μεγέθους, νούμερο εμφανώς μεγαλύτερο από ότι θα περίμενε κάποιος κοιτάζοντας απλά την αναλογία μέγιστης επίδοσης (περίπου x10) μεταξύ της Cell Broadband Engine και άλλων συμβατικών επεξεργαστών.

Υπέρβαση των τριών παραγόντων που περιορίζουν την απόδοση

Η Cell Broadband Engine ξεπερνά τρία σημαντικά εμπόδια που περιορίζουν την απόδοση των σύγχρονων μικροεπεξεργαστών: κατανάλωση ισχύος, χρήση της μνήμης και συχνότητα επεξεργαστή.

Υπέρβαση του περιορισμού της κατανάλωσης ισχύος

Με την εξέλιξη των μικροεπεξεργαστών η απόδοση περιορίζεται κατά κύριο λόγο από τις απώλειες ισχύος (κυρίως υπό μορφή θερμότητας) και όχι τόσο από τον αριθμό των διαθέσιμων πόρων στα ολοκληρωμένα κυκλώματα (transistors και καλωδιώσεις). Ενδεικτικό είναι το παρακάτω διάγραμμα που δείχνει την αύξηση της κατανάλωσης που απαιτείται προκειμένου να υποστηριχθούν υψηλότερες συχνότητες λειτουργίας. Μία αύξηση της τάξης του 45% στη συχνότητα λειτουργίας ενός τυπικού επεξεργαστή επιφέρει τριπλασιασμό της κατανάλωσης ισχύος.



Έτσι, ο μόνος τρόπος ώστε να αυξηθεί σημαντικά η απόδοση των μικροεπεξεργαστών είναι η με σχεδόν ίδιο ρυθμό βελτίωση της ενεργειακής απόδοσης.

Ένας τρόπος για την αύξηση της ενεργειακής απόδοσης είναι η διαφοροποίηση μεταξύ:

- Επεξεργαστών βελτιστοποιημένων για εκτέλεση λειτουργικών συστημάτων και κώδικα με πολύπλοκη ροή ελέγχου

- Επεξεργαστών βελτιστοποιημένων για εκτέλεση εφαρμογών με πολλούς αριθμητικούς υπολογισμούς

Η Cell Broadband Engine επιτυγχάνει την παραπάνω διαφοροποίηση παρέχοντας ένα γενικής χρήσεως επεξεργαστικό στοιχείο (PPE) ώστε να τρέχει το λειτουργικό σύστημα και λοιπό κώδικα ελέγχου και οκτώ SPEs εξειδικευμένα στην εκτέλεση αριθμητικών εφαρμογών.

Υπέρβαση των περιορισμών της μνήμης

Στους συμμετρικούς πολυεπεξεργαστές που τρέχουν σε συχνότητες πολλών GHz (ακόμα και σε αυτούς ου ενσωματώνουν ελεγκτή μνήμης) η καθυστέρηση από τη μνήμη DRAM πλησιάζει τους 1.000 κύκλους. Ως επακόλουθο, η επίδοση των προγραμμάτων εξαρτάται σε υψηλό βαθμό από τις μεταφορές δεδομένων μεταξύ του κύριου αποθηκευτικού χώρου και του επεξεργαστή. Επίσης, οι compilers ή ακόμα και οι προγραμματιστές πρέπει ρητά να διαχειριστούν αυτές τις μεταφορές δεδομένων, παρόλο που οι hardware μηχανισμοί στις caches υποτίθεται ότι απαλλάσσουν από αυτό το καθήκον.

Τα SPEs της Cell Broadband Engine χρησιμοποιούν δύο μηχανισμούς για να αντιμετωπίσουν τις μεγάλες καθυστερήσεις σε σχέση με τη μνήμη:

- Ιεραρχία μνήμης τριών επιπέδων (κύρια μνήμη, τοπική μνήμη σε κάθε SPE και μεγάλο πλήθος καταχωρητών σε κάθε SPE)

- Ασύγχρονες μεταφορές DMA μεταξύ κύριας μνήμης και τοπικών μνημών

Αυτά τα χαρακτηριστικά επιτρέπουν στους προγραμματιστές να διαρρυθμίσουν ταυτόχρονες μεταφορές δεδομένων και κώδικα ώστε να καλυφθούν αποτελεσματικά οι μεγάλες καθυστερήσεις. Χάρη σε αυτήν την οργάνωση η Cell Broadband Engine μπορεί να υποστηρίξει 128 ταυτόχρονες μεταφορές μεταξύ των οκτώ τοπικών μνημών των SPEs και της κύριας μνήμης. Ο αριθμός αυτός ξεπερνά τον αριθμό ταυτόχρονων μεταφορών σε συμβατικούς επεξεργαστές κατά έναν παράγοντα σχεδόν είκοσι.

Υπέρβαση του περιορισμού της συχνότητας

Οι συμβατικοί επεξεργαστές απαιτούν ολοένα περισσότερο βαθιές σωληνώσεις εντολών ώστε να επιτύχουν υψηλότερες συχνότητες λειτουργίας. Η συγκεκριμένη τεχνική έχει φτάσει σε ένα σημείο όπου τα οφέλη αρχίζουν να λιγοστεύουν – μπορεί να έχουμε ακόμα και αρνητικά αποτελέσματα αν ληφθεί υπ' όψιν και η κατανάλωση ενέργειας.

Εξειδικεύοντας το PPE και τα SPEs για εργασίες ελέγχου και υπολογισμών αντίστοιχα η αρχιτεκτονική Cell Broadband Engine, πάνω στην οποία βασίζεται η Cell Broadband Engine, επιτρέπει και το PPE και τα SPEs να σχεδιάζονται για υψηλές συχνότητες

λειτουργίας χωρίς υπερβολικό τίμημα. Το PPE επιτυγχάνει υψηλή επίδοση κυρίως εκτελώντας δύο νήματα ταυτόχρονα αντί να βελτιστοποιεί την επίδοση μονού νήματος (single-thread performance). Κάθε SPE επιτυγχάνει υψηλή επίδοση χρησιμοποιώντας μεγάλο register file, το οποίο υποστηρίζει την εκτέλεση πολλών εντολών ταυτόχρονα χωρίς την επιβάρυνση που επιφέρουν η μετονομασία καταχωρητών (register renaming) ή η εκτός-σειράς εκτέλεση (out of order execution). Επίσης, κάθε SPE επιτυγχάνει επίδοση χρησιμοποιώντας ασύγχρονες DMA μεταφορές, πράγμα το οποίο συνεπάγεται υποστήριξη πολλών ταυτόχρονων εργασιών στη μνήμη χωρίς την επιβάρυνση της πρόβλεψης.

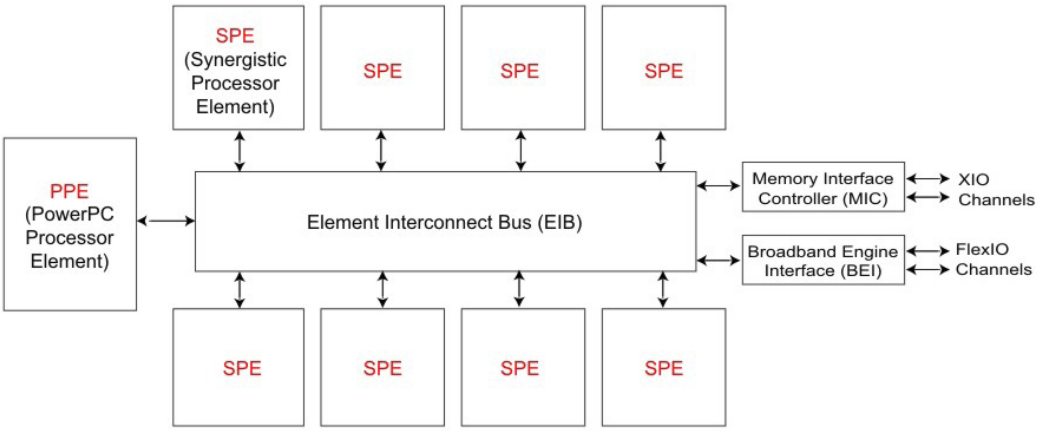
Βελτιστοποιώντας ξεχωριστά τα επεξεργαστικά στοιχεία ελέγχου ροής προγράμματος και ξεχωριστά τα στοιχεία αριθμητικών πράξεων η Cell Broadband Engine ξεπερνά σε μεγάλο βαθμό τα προβλήματα που επιβάλουν οι περιορισμοί της ισχύος, της μνήμης και της συχνότητας. Το καθαρό αποτέλεσμα είναι ένας επεξεργαστής ο οποίος, στο κόστος και στην κατανάλωση ενός συμβατικού επεξεργαστή, μπορεί να επιτύχει σχεδόν δεκαπλάσια μέγιστη επίδοση από τον τελευταίο. Φυσικά, η πραγματική επίδοση κάθε εφαρμογής ποικίλει. Μερικές εφαρμογές μπορεί να επωφεληθούν λίγο από τις SPEs, άλλες πάλι μπορεί να παρουσιάσουν αύξηση επίδοσης παραπάνω και από το δεκαπλάσιο. Γενικά, μαθηματικές εφαρμογές που χρησιμοποιούν δεδομένα των 32 bit ή μικρότερα (όπως απλής ακρίβειας κινητής υποδιαστολής και ακέραιοι) είναι εξαιρετικοί υποψήφιοι για την Cell Broadband Engine.

Κεφάλαιο 2

ΑΡΧΙΤΕΚΤΟΝΙΚΗ

Η Cell Broadband Engine αποτελείται από εννέα επεξεργαστές σε ένα και μοναδικό τσιπ, όλοι συνδεδεμένοι ο ένας με τον άλλον και με εξωτερικές συσκευές μέσω ενός συναφούς ως προς τη μνήμη διαύλου υψηλού εύρους ζώνης.

Ακολουθεί ένα μπλοκ διάγραμμα της Cell Broadband Engine:



Τα κύρια δομικά στοιχεία περιλαμβάνουν τα εξής:

- Επεξεργαστικό Στοιχείο PowerPC (PowerPC Processor Element - PPE). Το PPE είναι ο κύριος επεξεργαστής. Αποτελείται από έναν πυρήνα αρχιτεκτονικής 64-bit PowerPC μειωμένου συνόλου εντολών (RISC) με ένα παραδοσιακό υποσύστημα εικονικής μνήμης. Τρέχει το λειτουργικό σύστημα, διαχειρίζεται τους πόρους του συστήματος και προορίζεται κυρίως για επεξεργασία ροής ελέγχου, συμπεριλαμβανομένης της δημιουργίας και της διαχείρισης των νημάτων των SPEs. Μπορεί να τρέξει παλιό λογισμικό γραμμένο για αρχιτεκτονική PowerPC και παρουσιάζει καλές επιδόσεις στην εκτέλεση κώδικα ελέγχου συστήματος (system control code). Υποστηρίζει τόσο το σύνολο εντολών PowerPC όσο και το σύνολο εντολών Vector/SIMD Multimedia Extension.

- Συνεργατικά Επεξεργαστικά Στοιχεία (Synergistic Processor Elements - SPEs). Τα οκτώ SPEs είναι διανυσματικοί επεξεργαστές βελτιστοποιημένοι για

εκτέλεση πράξεων με πολλά δεδομένα που τους ανατίθενται από το PPE. Κάθε ένα από αυτά τα πανομοιότυπα επεξεργαστικά στοιχεία περιλαμβάνει έναν RISC πυρήνα, μία τοπική μνήμη για εντολές και δεδομένα, μεγέθους 256KB και ελεγχόμενη από το λογισμικό και ένα μεγάλο ενοποιημένο αρχείο καταχωρητών (128 καταχωρητές των 128 bit έκαστος). Τα SPEs υποστηρίζουν ένα ειδικό σύνολο εντολών SIMD και βασίζονται στις ασύγχρονες DMA μεταφορές για να μετακινήσουν δεδομένα και εντολές μεταξύ κύριας μνήμης και των τοπικών του μνημών. Οι DMA μεταφορές των SPEs προσπελαύνουν τον κύριο αποθηκευτικό χώρο χρησιμοποιώντας ενεργές διευθύνσεις (effective addresses) της αρχιτεκτονικής PowerPC. Όπως και στο PPE, η μετάφραση των διευθύνσεων βασίζεται στους πίνακες τμημάτων και σελίδων (segment and page tables) της αρχιτεκτονικής PowerPC. Τα SPEs δεν προορίζονται για εκτέλεση λειτουργικού συστήματος.

- Δίαυλος Διασύνδεσης Στοιχείων (Element Interconnect Bus - EIB). Το PPE και τα SPEs επικοινωνούν με συνάφεια (coherently) μεταξύ τους και με την κύρια μνήμη και τις συσκευές εισόδου/εξόδου μέσω του EIB. Ο EIB αποτελείται από μία δομή τεσσάρων δακτυλίων (δύο αριστερόστροφους και δύο δεξιόστροφους) για δεδομένα και μία δενδροειδή δομή για εντολές. Το εσωτερικό εύρος ζώνης του EIB είναι 96 bytes ανά κύκλο και μπορεί να υποστηρίξει περισσότερες από 100 εκκρεμείς αιτήσεις DMA για μεταφορά δεδομένων μεταξύ του κύριου αποθηκευτικού χώρου και των SPEs.

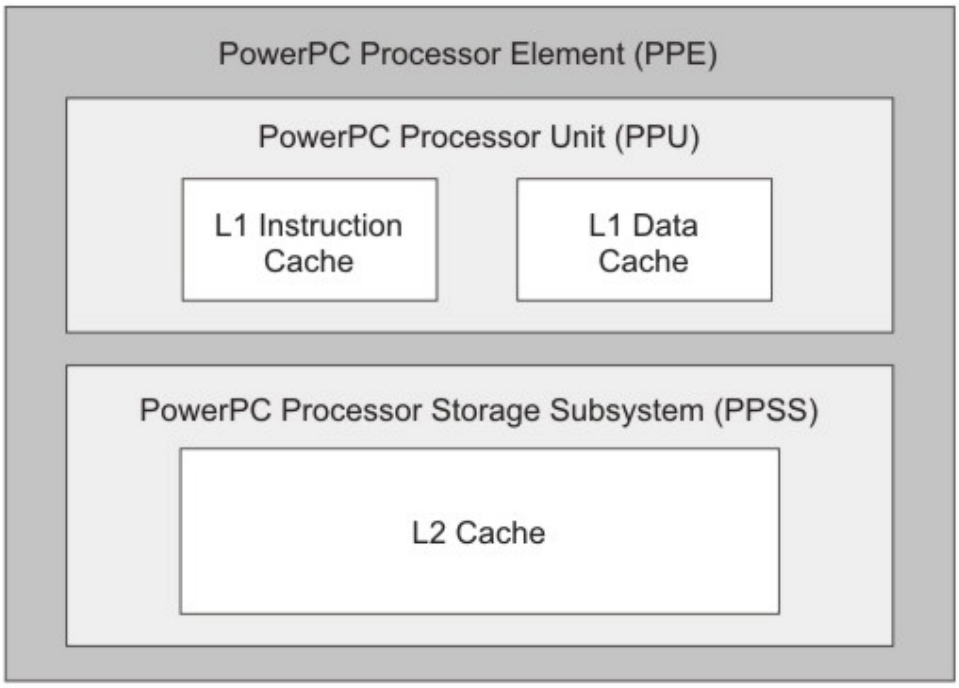
Ο συναφής ως προς τη μνήμη EIB έχεις δύο εξωτερικές διεπαφές (interfaces).

- Ο Ελεγκτής Διεπαφής Μνήμης (Memory Interface Controller - MIC) παρέχει τη διεπαφή μεταξύ του EIB και του κύριου αποθηκευτικού χώρου. Υποστηρίζει δύο κανάλια μνήμης Rambus Extreme Data Rate (XDR) I/O (XIO) και προσβάσεις στη μνήμη μεγέθους 1 έως 8, 16, 32, 64 ή 128 bytes ανά κανάλι.

- Η Διεπαφή της Cell Broadband Engine (Cell Broadband Engine Interface - BEI) διαχειρίζεται τις μεταφορές δεδομένων μεταξύ του EIB και των συσκευών εισόδου/εξόδου. Παρέχει μετάφραση διευθύνσεων, επεξεργασία εντολών, έναν εσωτερικό ελεγκτή διακοπών (interrupt controller) και διεπαφή προς το δίαυλο (bus interface). Υποστηρίζει δύο εξωτερικά κανάλια εισόδου/εξόδου τύπου Rambus FlexIO. Ένα κανάλι υποστηρίζει μόνο μη συναφείς συσκευές εισόδου/εξόδου. Το άλλο κανάλι μπορεί να παραμετροποιηθεί ώστε να υποστηρίζει είτε μη συναφείς μεταφορές είτε συναφείς μεταφορές οι οποίες συνδέουν λογικά το EIB σε άλλες συμβατές εξωτερικές συσκευές, όπως μία άλλη Cell Broadband Engine.

Το επεξεργαστικό στοιχείο PowerPC (PowerPC Processor Element)

Το Επεξεργαστικό Στοιχείο PowerPC (PowerPC Processor Element – PPE) είναι ένας γενικής χρήσεως, δύο νημάτων (dual threaded), 64bit RISC επεξεργαστής που πληροί τις προδιαγραφές της αρχιτεκτονικής PowerPC, έκδοσης 2.02, με τις επεκτάσεις Vector/SIMD Multimedia Extension. Προγράμματα που έχουν γραφτεί για τον επεξεργαστή PowerPC 970, για παράδειγμα, μπορούν να τρέξουν στη Cell Broadband Engine χωρίς τροποποιήσεις.



Όπως φαίνεται στην παραπάνω εικόνα, το PPE αποτελείται από δύο κύρια στοιχεία:

- Την Επεξεργαστική Μονάδα PowerPC (PowerPC Processor Unit – PPU)
- Το Υποσύστημα Αποθηκευτικού Χώρου του Επεξεργαστή PowerPC (PowerPC Processor Storage Subsystem - PPSS)

Το PPE είναι υπεύθυνο για το συνολικό έλεγχο του συστήματος. Τρέχει το λειτουργικό σύστημα για όλες τις εφαρμογές που τρέχουν στη Cell Broadband Engine.

Η PPU αναλαμβάνει τον έλεγχο και την εκτέλεση των εντολών. Περιλαμβάνει:

- Το πλήρες σύνολο καταχωρητών της αρχιτεκτονικής PowerPC 64bit
- 32 διανυσματικούς καταχωρητές των 128 bit
- Μία L1 cache εντολών, μεγέθους 32KB
- Μία L1 cache δεδομένων, μεγέθους 32KB
- Μία μονάδα ελέγχου εντολών (instruction control)
- Μία μονάδα φόρτωσης και αποθήκευσης (load and store)
- Μία μονάδα ακεραίων
- Μία μονάδα κινητής υποδιαστολής
- Μία μονάδα διανυσμάτων
- Μία μονάδα διακλάδωσης
- Μία μονάδα διαχείρισης εικονικής μνήμης

Η PPU υποστηρίζει την ταυτόχρονη εκτέλεση δύο νημάτων και μπορεί να ειδωθεί ως ένας πολυεπεξεργαστής δύο δρόμων με μοιραζόμενη ροή δεδομένων. Αυτό φαίνεται από την πλευρά του λογισμικού ως δύο ανεξάρτητες επεξεργαστικές μονάδες. Η κατάσταση κάθε νήματος αναπαράγεται (duplication) περιλαμβάνοντας όλους τους καταχωρητές τις αρχιτεκτονικής και τους καταχωρητές ειδικού σκοπού εκτός από αυτούς που έχουν σχέση με πόρους σε επίπεδο συστήματος, όπως λογικά διαμερίσματα, μνήμη και έλεγχο νημάτων. Οι περισσότεροι μη-αρχιτεκτονικοί (non-architected) πόροι, όπως οι caches και οι ουρές, μοιράζονται μεταξύ των δύο νημάτων, εκτός από τις περιπτώσεις όπου ένας πόρος είναι μικρός ή προσφέρει κρίσιμη βελτίωση στην επίδοση σε πολυνηματικές εφαρμογές.

Το PPSS διαχειρίζεται αιτήσεις προς τη μνήμη από το PPE και εξωτερικές αιτήσεις προς το PPE από άλλους επεξεργαστές ή συσκευές εισόδου/εξόδου. Περιλαμβάνει:

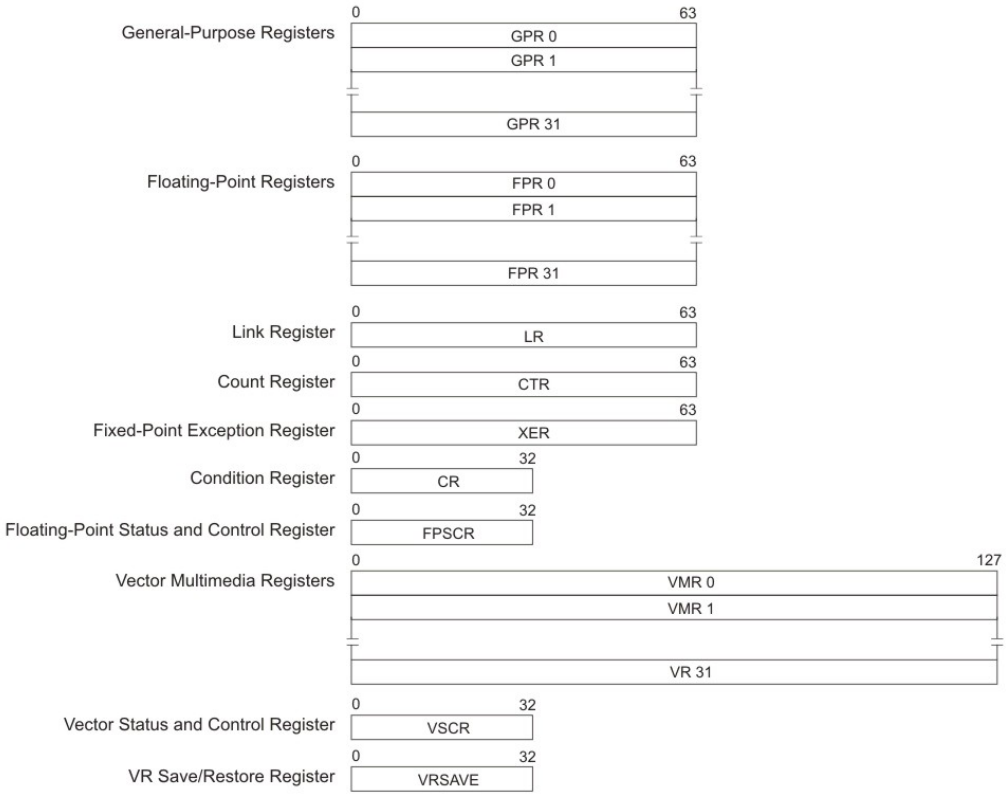
- Μία ενοποιημένη L2 cache για εντολές και δεδομένα, μεγέθους 512KB
- Διάφορες ουρές
- Μία μονάδα διεπαφής διαύλου (bus interface unit) που διαχειρίζεται (arbitration and pacing) το δίαυλο EIB

Η μνήμη θεωρείται ως ένας γραμμικός πίνακας από bytes, με διευθύνσεις από 0 έως 2^{64} – 1. Κάθε byte χαρακτηρίζεται από τη διεύθυνσή του και περιέχει μία τιμή. Κάθε φορά γίνεται μόνο μία πρόσβαση αποθήκευσης σε μία διεύθυνση και όλες οι προσβάσεις φαίνονται να γίνονται με τη σειρά που παρουσιάζονται στο πρόγραμμα.

Η L2 cache και οι caches για τις μεταφράσεις διευθύνσεων χρησιμοποιούν πίνακες στρατηγικών αντικατάστασης (replacement-management tables) που επιτρέπουν στο λογισμικό να ελέγχει τη χρήση των caches. Αυτή η δυνατότητα ελέγχου των πόρων των caches από το λογισμικό είναι ιδιαιτέρως χρήσιμη για προγραμματισμό πραγματικού χρόνου.

Καταχωρητές

Το παρακάτω σχήμα δείχνει όλους τους καταχωρητές του PPE που μπορούν να χρησιμοποιηθούν από τον προγραμματιστή (problem-state registers).



Όλες οι υπολογιστικές εντολές δουλεύουν μόνο με ορίσματα που βρίσκονται σε καταχωρητές – δεν υπάρχουν υπολογιστικές εντολές που να τροποποιούν άμεσα τον αποθηκευτικό χώρο (στη μνήμη). Προκειμένου να χρησιμοποιηθεί σε κάποιον υπολογισμό ένα όρισμα που

βρίσκεται στη μνήμη και στη συνέχεια να τροποποιηθεί η ίδια ή κάποια άλλη θέση μνήμης, το περιεχόμενο αυτό της μνήμης πρέπει να:

- Φορτωθεί σε έναν καταχωρητή
- Τροποποιηθεί
- Αποθηκευτεί πίσω στη διεύθυνση-προορισμό

Οι καταχωρητές του PPE περιλαμβάνουν:

- Καταχωρητές Γενικής Χρήσεως (General Purpose Registers – GPRs) – Οι εντολές για δεδομένα μη-κινητής υποδιαστολής χρησιμοποιούν τους GRPs σε όλο το εύρος τους (64 bits), ενώ το πλήθος αυτών είναι 32. Οι εντολές είναι ανεξάρτητες της κατάστασης λειτουργίας (mode), με εξαίρεση το 32bit mode, όπου ο επεξεργαστής χρησιμοποιεί μόνο τα 32 λιγότερο σημαντικά bits για να καθορίσει μία διεύθυνση στη μνήμη και τα bits κρατουμένου, υπερχείλισης και κατάστασης εγγραφών (record status).

- Καταχωρητές Κινητής Υποδιαστολής (Floating-Point Registers – FPRs) – Οι 32 FPRs έχουν εύρος 64 bits. Οι εσωτερική αναπαράσταση δεδομένων κινητής υποδιαστολής είναι η αναπαράστασης διπλής ακριβείας IEEE 754.

- Καταχωρητής Σύνδεσης (Link Register – LR) – Ο 64bit LR μπορεί να χρησιμοποιηθεί για να κρατήσει την ενεργή διεύθυνση ενός στόχου διακλάδωσης. Οι εντολές διακλάδωσης με το link bit (LK) ίσο με 1 (που σημαίνει ότι πρόκειται για εντολή κλήσης υπορουτίνας) αντιγράφουν τη διεύθυνση της επόμενης εντολής στον LR. Μία Εντολή Μετακίνησης Δεδομένων Σε Ειδικής Χρήσεως Καταχωρητή (Move To Special-Purpose Register instruction) μπορεί να αντιγράψει τα περιεχόμενα ενός GPR στον LR.

- Καταχωρητής Μετρητής (Count Register – CTR) – Ο 64bit CTR μπορεί να χρησιμοποιηθεί για να κρατήσει είτε τον μετρητή ενός βρόχου (loop) είτε την ενεργό διεύθυνση-στόχο μιας διακλάδωσης. Μερικές μορφές εντολών διακλάδωσης υπό συνθήκη μειώνουν τον CTR και ελέγχουν εάν η τιμή του έγινε 0. Μία Εντολή Μετακίνησης Δεδομένων Σε Ειδικής Χρήσεως Καταχωρητή (Move To Special-Purpose Register instruction) μπορεί να αντιγράψει τα περιεχόμενα ενός GPR στον CTR.

- Καταχωρητής Εξαιρέσεων Μη-Κινητής Υποδιαστολής (Fixed-Point Exception Register – XER) – Ο 64bit XER περιέχει τα bits κρατουμένου και υπερχείλισης και τον μετρητή bytes για τις εντολές move-assist. Οι περισσότερες αριθμητικές πράξεις έχουν μορφές εντολών για να θέτουν τα bits κρατουμένου και υπερχείλισης.

- Καταχωρητής Συνθήκης (Condition Register – CR) – Οι συγκρίσεις υπό συνθήκη εκτελούνται θέτοντας πρώτα έναν κωδικό συνθήκης στον 32bit CR με μια εντολή σύγκρισης ή διαχείρισης εγγραφών (recording instruction). Ο κωδικός συνθήκης είναι από εκεί και πέρα διαθέσιμος ως μία τιμή ή μπορεί να ελεγχθεί από μία εντολή διακλάδωσης για τον έλεγχο της ροής του προγράμματος. Ο CR αποτελείται από οκτώ ανεξάρτητα πεδία των 4 bits ομαδοποιημένα μαζί για βολική αποθήκευση ή ανάκτηση κατά την αλλαγή διεργασίας (context). Κάθε πεδίο μπορεί να κρατήσει πληροφορίες για την κατάσταση από μία εντολή σύγκρισης, αριθμητικής ή λογικής πράξης. Ο compiler μπορεί να διαχειριστεί τα πεδία του CR ώστε να αποφεύγονται κίνδυνοι δεδομένων (data hazards) με τον ίδιο τρόπο που διαχειρίζεται τη χρήση των GPRs. Εγγραφές στον CR πραγματοποιούνται μόνο για εντολές που τις απαιτούν ρητά. Οι περισσότερες πράξεις έχουν μορφές εντολών και recording και non-recording τύπου.

- Καταχωρητής Κατάστασης και Ελέγχου Κινητής Υποδιαστολής (Floating-Point Status and Control Register – FPSCR) – Ο επεξεργαστής ενημερώνει τον 32bit FPSCR έπειτα από κάθε πράξη κινητής υποδιαστολής ώστε να κρατήσει πληροφορίες σχετικά με το αποτέλεσμα και οποιεσδήποτε σχετιζόμενες εξαιρέσεις. Περιλαμβάνονται οι πληροφορίες κατάστασης που απαιτούνται από το πρότυπο IEEE 754 καθώς και μερικές επιπλέον πληροφορίες για το χειρισμό εξαιρέσεων.

- Διανυσματικοί Καταχωρητές (Vector Registers – VRs) – Υπάρχουν 32 Διανυσματικοί Καταχωρητές των 128bit. Χρησιμοποιούνται ως πηγή και προορισμός για όλες τις διανυσματικές πράξεις.

- Διανυσματικός Καταχωρητής Κατάστασης και Ελέγχου (Vector Status and Control Register – VSCR) – Ο 32bit VSCR διαβάζεται και γράφεται με τρόπο παρόμοιο του FPSCR. Έχει δύο ορισμένα bits, ένα bit λειτουργίας σε non-Java™ κατάσταση και ένα bit κορεσμού. Τα υπόλοιπα bits είναι δεσμευμένα. Παρέχονται ειδικές εντολές για την αντιγραφή του VSCR σε κάποιον VR καταχωρητή.

- Διανυσματικός Καταχωρητής Αποθήκευσης (Vector Save Register – VRSAVE) – Ο 32bit καταχωρητής VRSAVE βοηθάει το χρήστη και το προνομιούχο λογισμικό στην αποθήκευση και ανάκτηση της αρχιτεκτονικής κατάστασης μεταξύ αλλαγών διεργασιών (context switching).

Σύνολα Εντολών

Το PPE υποστηρίζει δύο σύνολα εντολών: το σύνολο εντολών PowerPC (PowerPC instruction set) και το σύνολο εντολών Vector/SIMD Multimedia Extension (Vector/SIMD Multimedia Extension instruction set).

Το σύνολο εντολών PowerPC χρησιμοποιεί εντολές που έχουν μήκος 4 bytes και είναι ευθυγραμμισμένες κατά λέξη (word aligned). Υποστηρίζει προσβάσεις σε δεδομένα μεγέθους byte, μισής λέξης (halfword), λέξης (word) και διπλής λέξης (doubleword) μεταξύ του αποθηκευτικού χώρου και των 32 γενικής χρήσης καταχωρητών (GPRs). Το σύνολο εντολών υποστηρίζει επίσης προσβάσεις σε δεδομένα μεγέθους λέξης (word) και διπλής λέξης (doubleword) μεταξύ του αποθηκευτικού χώρου και του συνόλου των 32 καταχωρητών κινητής υποδιαστολής (FPRs). Οι προσημασμένοι ακέραιοι αναπαρίστανται σε μορφή συμπληρώματος ως προς 2.

Το σύνολο εντολών Vector/SIMD Multimedia Extensions χρησιμοποιεί εντολές οι οποίες, όπως και οι εντολές PowerPC, έχουν μέγεθος 4 bytes και είναι ευθυγραμμισμένες κατά λέξη. Όμως, όλα τα ορίσματά τους έχουν μέγεθος 128 bits. Τα περισσότερα ορίσματα των εντολών Vector/SIMD Multimedia Extension είναι διανύσματα που περιέχουν αριθμούς κινητής υποδιαστολής μονής ακριβείας, ακεραίους, βαθμωτά δεδομένα και δεδομένα μη-κινητής υποδιαστολής μεγέθους 8, 16 και 32 bits.

Εντολές PowerPC

Κάθε φορά που παρουσιάζονται στον επεξεργαστή διευθύνσεις εντολών τα δύο λιγότερο σημαντικά bits αγνοούνται. Παρομοίως, κάθε φορά που ο επεξεργαστής δημιουργεί μία διεύθυνση εντολής τα δύο λιγότερο σημαντικά bits είναι 0. Η διεύθυνση είτε μίας εντολής είτε ενός δεδομένου μεγέθους πολλών bytes είναι το byte με τη μικρότερη αρίθμηση (lowest-numbered byte). Αυτή η διεύθυνση είναι η διεύθυνση του περισσότερο σημαντικού (most significant) byte της εντολής ή του δεδομένου (σύμβαση big endian). Η σύμβαση little endian δεν υποστηρίζεται. Η αριθμητική για τον υπολογισμό των διευθύνσεων είναι μη προσημασμένη και αγνοούνται τα κρατούμενα από το bit 0 (το περισσότερο σημαντικό bit – MSB).

Τρόποι διευθυνσιοδότησης

Όλες οι εντολές, εκτός από τις εντολές διακλάδωσης, δημιουργούν διευθύνσεις αυξάνοντας τον μετρητή προγράμματος (program counter). Όλες οι εντολές φόρτωσης και αποθήκευσης προσδιορίζουν κάποιον καταχωρητή-βάση.

Η ενεργός διεύθυνση στη μνήμη για κάποιο δεδομένο υπολογίζεται σε σχέση με τον καταχωρητή-βάση με κάποιον από τους εξής τρεις τρόπους:

- Βάση + Εκτόπιση (Base + Displacement) – Αυτές οι μορφές εντολών φόρτωσης και αποθήκευσης υπολογίζουν μία διεύθυνση η οποία είναι το άθροισμα μίας εκτόπισης καθοριζομένης άμεσα από το αντίστοιχα πεδίο της εντολής (16bit sign-extended immediate field) και του περιεχομένου κάποιου καταχωρητή-βάσης.

- Καταχωρητής + Καταχωρητής (Register + Register) – Οι δεικτοδοτημένες μορφές των εντολών φόρτωσης και αποθήκευσης υπολογίζουν μία διεύθυνση η οποία είναι το άθροισμα των περιεχομένων του καταχωρητή-δείκτη (που είναι ένας GPR) και του καταχωρητή-βάσης.

- Καταχωρητής (Register) – Οι εντολές Load String Immediate και Store String Immediate χρησιμοποιούν χωρίς τροποποίηση τα περιεχόμενα του καταχωρητή-βάσης για να υπολογίσουν μία διεύθυνση.

Οι φορτώσεις και οι αποθηκεύσεις μπορούν να προσδιορίσουν μία μορφή ενημέρωσης (update form) με την οποία επαναφορτώνεται ο καταχωρητής-βάση με τη διεύθυνση που υπολογίστηκε, εκτός και αν ο καταχωρητής-βάση είναι ο καταχωρητής-προορισμός του φορτώματος.

Οι διακλαδώσεις είναι οι μόνες εντολές που προσδιορίζουν ρητά τη διεύθυνση της επόμενης εντολής. Μία εντολή διακλάδωσης προσδιορίζει την ενεργό διεύθυνση του στόχου της διακλάδωσης με έναν από τους εξής τρόπους:

- Δεν Πραγματοποιείται Διακλάδωση (Branch Not Taken) – Η διεύθυνση της επόμενης εντολής είναι η διεύθυνση της τρέχουσας εντολής αυξημένης κατά 4.

- Απόλυτα (Absolute) – Η διεύθυνση της επόμενης εντολής δίνεται άμεσα στο αντίστοιχο πεδίο της εντολής διακλάδωσης.

- Σχετικά (Relative) – Η διεύθυνση της επόμενης εντολής δίνεται από το άθροισμα του immediate field της εντολής διακλάδωσης και της διεύθυνσης της εντολής διακλάδωσης καθ' εαυτής.

- Καταχωρητής Σύνδεσης ή Καταχωρητής Μέτρησης (Link Register or Count Register) – Η διεύθυνση της επόμενης εντολής είναι η ενεργός διεύθυνση του στόχου της διακλάδωσης που βρίσκεται αποθηκευμένη στον Καταχωρητή Σύνδεσης ή στον Καταχωρητή Μέτρησης, αντίστοιχα.

Τύποι εντολών

Οι εντολές PowerPC του Επεξεργαστικού Στοιχείου PowerPC (PPE) μπορούν να έχουν μέχρι και τρία ορίσματα. Οι περισσότερες υπολογιστικές εντολές ορίζουν δύο ορίσματα προέλευσης και ένα όρισμα προορισμού.

Οι εντολές PowerPC του PPE περιλαμβάνουν τους ακόλουθους τύπους:

- Εντολές Ακεραίων (Integer Instructions) – Αυτές περιλαμβάνουν εντολές αριθμητικών πράξεων, εντολές συγκρίσεων, εντολές λογικών πράξεων και εντολές περιστροφών/ολισθήσεων (rotate/shift).

- Εντολές Κινητής Υποδιαστολής (Floating-Point Instructions) – Αυτές περιλαμβάνουν αριθμητική κινητής υποδιαστολής, πολλαπλασιασμούς-προσθέσεις, συγκρίσεις και μεταφορές δεδομένων καθώς και εντολές που επηρεάζουν τον Καταχωρητή Κατάστασης και Ελέγχου Κινητής Υποδιαστολής (FPSCR). Οι εντολές κινητής υποδιαστολής επενεργούν πάνω σε ορίσματα απλής και διπλής ακριβείας.

- Εντολές Φόρτωσης και Αποθήκευσης (Load and Store Instructions) – Αυτές περιλαμβάνουν εντολές φόρτωσης και αποθήκευσης ακεραίων και αριθμών κινητής υποδιαστολής, με επιλογές byte-reverse, multiple και string για της φορτώσεις και αποθηκεύσεις ακεραίων.

- Εντολές Συγχρονισμού με τη Μνήμη (Memory Synchronization Instructions) – Αυτές οι εντολές ελέγχουν τη σειρά με την οποία εκτελούνται οι ενέργειες στη μνήμη ως προς ασύγχρονα γεγονότα και τη σειρά με την οποία οι άλλοι επεξεργαστές ή μηχανισμοί πρόσβασης στη μνήμη βλέπουν αυτές τις ενέργειες. Οι τύποι εντολών περιλαμβάνουν φορτώσεις και αποθηκεύσεις με κράτηση (reservation), συγχρονισμό (synchronization) και εξαναγκασμένη εντός-σειράς εκτέλεση εισόδου/εξόδου (enforced in-order execution of I/O). Είναι ιδιαιτέρως χρήσιμες για πολυεπεξεργασία.

- Εντολές Ελέγχου Ροής (Flow Control Instructions) – Αυτές περιλαμβάνουν διακλαδώσεις υπό ή χωρίς συνθήκη, παγίδες (traps) καθώς και άλλες εντολές που επηρεάζουν τη ροή των εντολών.

- Εντολές Ελέγχου του Επεξεργαστή (Processor Control Instructions) – Αυτές οι εντολές χρησιμοποιούνται για το συγχρονισμό των προσβάσεων στη μνήμη και τη διαχείριση των caches, των Translation Lookaside Buffers (TLBs), των καταχωρητών τμημάτων (segment registers) και άλλες προνομιούχες καταστάσεις του επεξεργαστή. Περιλαμβάνουν εντολές μετακίνησης προς/από καταχωρητές ειδικού σκοπού.

- Εντολές Ελέγχου Μνήμης και Cache (Memory and Cache Control Instructions) – Αυτές οι εντολές ελέγχουν τις caches, τους TLBs και τους καταχωρητές τμημάτων (segment registers).

- Εντολές Εξωτερικού Ελέγχου (External Control Instructions) – Αυτές οι εντολές επιτρέπουν σε ένα πρόγραμμα χρήστη να επικοινωνεί με μία συσκευή ειδικής χρήσεως.

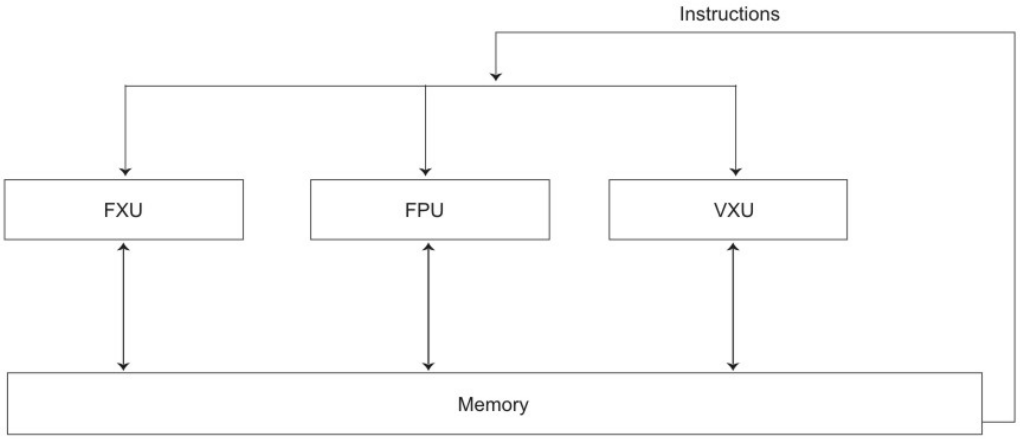
Εντολές Vector /SIMD Multimedia Extension

Η 128bit μονάδα Vector/SIMD Multimedia Extension (Vector/SIMD Multimedia Extension Unit - VXU) λειτουργεί ταυτόχρονα με τη μονάδα ακεραίων (fixed-point integer unit - FXU) και τη μονάδα εκτέλεσης πράξεων κινητής υποδιαστολής (floating-point execution unit - FPU) της PPU. Όπως και οι εντολές PowerPC, οι εντολές Vector/SIMD Multimedia Extension έχουν μέγεθος 4 bytes και είναι ευθυγραμμισμένες κατά λέξη (word aligned). Οι εντολές Vector/SIMD Multimedia Extension υποστηρίζουν ταυτόχρονη εκτέλεση πάνω σε πολλά στοιχεία τα οποία αποτελούν τα 128bit διανυσματικά ορίσματα. Αυτά τα στοιχεία των διανυσμάτων μπορεί να είναι bytes, μισές λέξεις (half words) ή λέξεις (words).

Όλες οι εντολές Vector/SIMD Multimedia Extension είναι σχεδιασμένες ώστε να υλοποιούνται εύκολα σε μία αρχιτεκτονική αγωγού (pipeline). Η παράλληλη εκτέλεση με τις εντολές ακεραίων και κινητής υποδιαστολής του PPE απλοποιείται από το γεγονός ότι οι εντολές Vector/SIMD Multimedia Extension:

- Δεν υποστηρίζουν μη ευθυγραμμισμένες προσβάσεις στη μνήμη ή περίπλοκες λειτουργίες

- Μοιράζονται λίγους μόνο πόρους και μονοπάτια επικοινωνίας με άλλες μονάδες εκτέλεσης του PPE.



Τρόποι διευθυνσιοδότησης

Το Επεξεργαστικό Στοιχείο PowerPC (PPE) υποστηρίζει όχι μόνο βασικές λειτουργίες φόρτωσης και αποθήκευσης αλλά και φορτώσεις και αποθηκεύσεις διανυσμάτων (σε μορφή left-indexed ή right-indexed).

Όλες οι πράξεις φόρτωσης και αποθήκευσης στο σύνολο εντολών Vector/SIMD Multimedia Extension χρησιμοποιούν τον τρόπο δεικτοδοτημένης διευθυνσιοδότησης Καταχωρητής + Καταχωρητής (Register + Register), όπου η διεύθυνση προκύπτει από το άθροισμα των περιεχομένων ενός GPR καταχωρητή-δείκτη και ενός GPR καταχωρητή-βάσης. Αυτός ο τρόπος διευθυνσιοδότησης είναι πολύ χρήσιμος για την πρόσβαση πινάκων.

Επιπρόσθετα στις φορτώσεις και τις αποθηκεύσεις το σύνολο εντολών Vector/SIMD Multimedia Extension παρέχει ένα πολύ ισχυρό σύνολο εντολών διαχείρισης στοιχείων – για παράδειγμα ανακατέματα(shuffles), περιστροφές (rotations) και ολισθήσεις (shifts) – ώστε να μπαίνουν τα στοιχεία αυτά στην επιθυμητή ευθυγράμμιση και διάταξη αφότου τα διανύσματα έχουν φορτωθεί στους διανυσματικούς καταχωρητές.

Τύποι εντολών

Οι περισσότερες εντολές Vector/SIMD Multimedia Extension έχουν τρία ή τέσσερα ορίσματα των 128 bits – δύο ή περισσότερα ορίσματα προέλευσης και ένα αποτέλεσμα. Επίσης, οι περισσότερες εντολές είναι από τη φύση τους διανυσματικές (SIMD).

Οι εντολές έχουν επιλεγεί για τη χρησιμότητά τους σε αλγορίθμους ψηφιακής επεξεργασίας σήματος (DSP), περιλαμβανομένων και των 3D γραφικών.

Οι εντολές Vector/SIMD Multimedia Extension περιλαμβάνουν τους ακόλουθους τύπους:

- Εντολές Διανυσμάτων Ακεραίων (Vector Integer Instructions) – Αυτές περιλαμβάνουν εντολές αριθμητικών πράξεων, εντολές σύγκρισης, εντολές λογικών πράξεων, εντολές περιστροφών (rotate) και ολισθήσεων (shift). Τα στοιχεία των διανυσμάτων-ορισμάτων τους έχουν μέγεθος byte, μισής λέξης (halfword) και λέξης (word).

- Εντολές Διανυσμάτων Κινητής Υποδιαστολής (Vector Floating-Point Instructions) – Αυτές περιλαμβάνουν εντολές αριθμητικής κινητής υποδιαστολής, πολλαπλασιασμών/προσθέσεων, στρογγυλοποιήσεων και μετατροπών, συγκρίσεων και εκτιμήσεων. Επενεργούν πάνω σε στοιχεία διανυσμάτων κινητής υποδιαστολής απλής ακρίβειας.

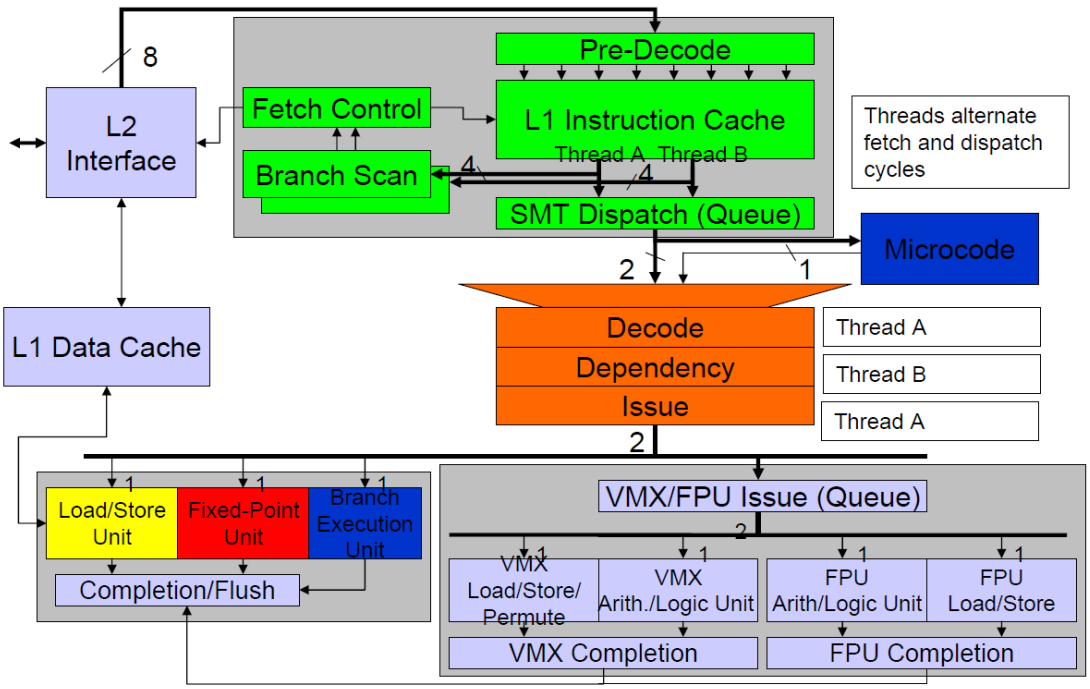
- Εντολές Φόρτωσης και Αποθήκευσης Διανυσμάτων (Vector Load and Store Instructions) – Αυτές περιλαμβάνουν μόνο βασικές εντολές φόρτωσης και αποθήκευσης ακεραίων και αριθμών κινητής υποδιαστολής. Δεν υπάρχουν ενημερωτικές μορφές (update forms) για αυτές τις εντολές. Επενεργούν σε 128bit διανύσματα.

- Εντολές Μετάθεσης και Μορφοποίησης Διανυσμάτων (Vector Permutation and Formatting Instructions) – Αυτές περιλαμβάνουν εντολές πακεταρίσματος (pack), ξεπακεταρίσματος (unpack), συγχώνευσης (merge), αναπαραγωγής (splat), μετάθεσης (permute), επιλογής (select) και ολίσθησης (shift) διανυσμάτων.

- Εντολές Ελέγχου του Επεξεργαστή (Processor Control Instructions) – Αυτές περιλαμβάνουν εντολές που διαβάζουν και γράφουν το Διανυσματικό Καταχωρητή Κατάστασης και Ελέγχου (VSCR).

- Εντολές Ελέγχου Μνήμης (Memory Control Instructions) – Αυτές περιλαμβάνουν εντολές για τη διαχείριση των caches (σε user-level και supervisor-level). Αυτές είναι εντολές "no-ops".

Το παρακάτω block διάγραμμα δείχνει τις λεπτομέρειες της αρχιτεκτονικής του PPE που αναλύθηκαν:

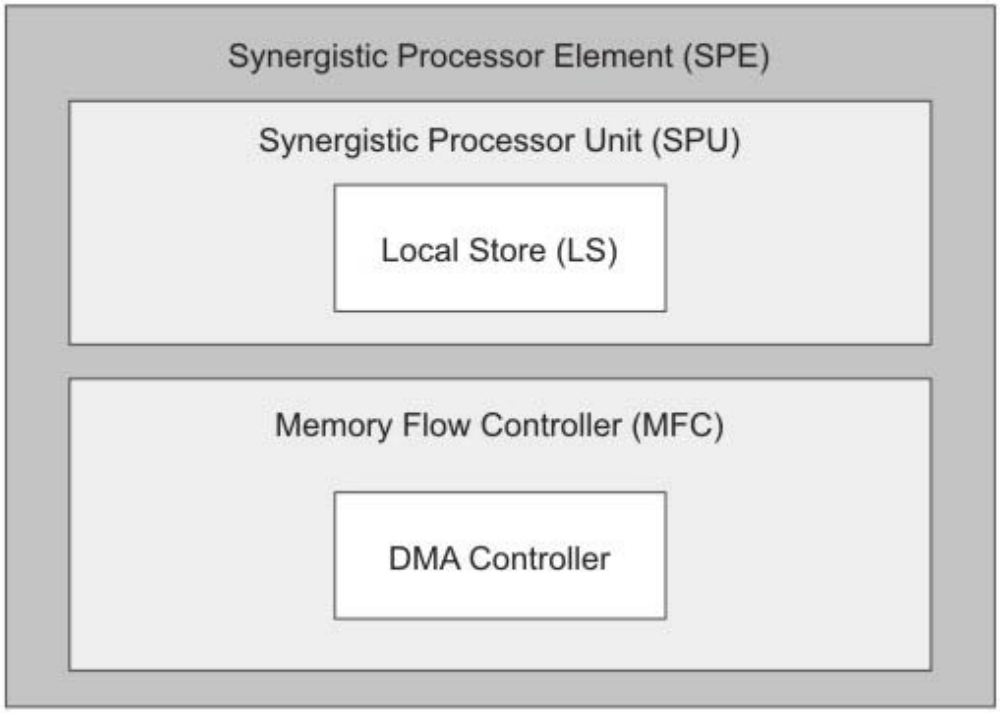


Τα Συνεργατικά Επεξεργαστικά Στοιχεία (Synergistic Processor Elements)

Κάθε ένα από τα οκτώ Συνεργατικά Επεξεργαστικά Στοιχεία (Synergistic Processor Elements - SPEs) είναι ένας RISC επεξεργαστής 128bit εξειδικευμένος σε εφαρμογές με πολλά δεδομένα και αριθμητικές διανυσματικές (SIMD) πράξεις. Όπως φαίνεται στο ακόλουθο διάγραμμα, κάθε SPE αποτελείται από δύο κύριες μονάδες:

- Τη Συνεργατική Επεξεργαστική Μονάδα (Synergistic Processor Unit - SPU)
- Τον Ελεγκτή Ροής Μνήμης (Memory Flow Controller - MFC)

Η SPU αναλαμβάνει τον έλεγχο και την εκτέλεση των εντολών. Περιλαμβάνει ένα και μόνο αρχείο καταχωρητών (register file) με 128 καταχωρητές (μήκους 128bit ο καθένας), μία ενοποιημένη (για εντολές και δεδομένα) τοπική μνήμη (Local Store - LS) μεγέθους 256KB, μία μονάδα ελέγχου εντολών (instruction control unit), μία μονάδα φόρτωσης και αποθήκευσης (load and store unit), δύο μονάδες ακεραίων, μία μονάδα κινητής υποδιαστολής και μία διεπαφή με τα κανάλια και το ελεγκτή DMA. Η SPU υλοποιεί ένα νέο σύνολο εντολών SIMD, το SPU Instruction Set Architecture, το οποίο είναι συγκεκριμένο ως προς την αρχιτεκτονική Broadband Processor Architecture.



Κάθε SPU είναι ένας ανεξάρτητος επεξεργαστής με το δικό του μετρητή προγράμματος (program counter) και είναι βελτιστοποιημένος να τρέχει SPE νήματα που δημιουργήθηκαν από το PPE. Η SPU φέρνει εντολές από και φορτώνει και αποθηκεύει δεδομένα από και προς τη δική της τοπική μνήμη. Η τοπική μνήμη, ως προς τις προσβάσεις σε αυτήν από την SPU της, είναι μη προστατευμένος και μη μεταφραζόμενος αποθηκευτικός χώρος. Ο MFC περιλαμβάνει έναν ελεγκτή DMA που υποστηρίζει μεταφορές DMA. Τα προγράμματα που τρέχουν στην SPU, στο PPE ή σε άλλη SPU χρησιμοποιούν τις μεταφορές DMA του MFC προκειμένου να μετακινήσουν δεδομένα μεταξύ της τοπικής μνήμης της SPU και του κύριου αποθηκευτικού χώρου [ως κύριος αποθηκευτικός χώρος χαρακτηρίζεται ο χώρος των ενεργών διευθύνσεων που περιλαμβάνει την κύρια μνήμη, τις τοπικές μνήμες των άλλων SPEs και απεικονιζόμενους στη μνήμη καταχωρητές (memory mapped registers) όπως διάφοροι απεικονιζόμενοι στη μνήμη καταχωρητές εισόδου/εξόδου (memory mapped I/O - MMIO)]. Ο MFC παρέχει διεπαφή μεταξύ της SPU και του EIB, υλοποιεί λειτουργίες σχετικά με τη δέσμευση εύρους ζώνης στο δίαυλο και συγχρονίζει ενέργειες μεταξύ της SPU και άλλων επεξεργαστών στο σύστημα.

Για να υποστηρίξει τις DMA μεταφορές ο MFC διατηρεί και επεξεργάζεται ουρές από εντολές DMA. Αφότου μία εντολή DMA έχει μπει στην ουρά του MFC η SPU μπορεί να συνεχίσει την εκτέλεση των εντολών του προγράμματος όσο ο MFC επεξεργάζεται την εντολή DMA αυτόνομα και ασύγχρονα. Ο MFC μπορεί επίσης να εκτελέσει αυτόνομα μία σειρά από μεταφορές DMA, όπως λίστες (scatter-gather), ως επακόλουθο μίας εντολής λίστας DMA

(DMA-list command). Αυτή η αυτόνομη εκτέλεση των εντολών DMA από τον MFC και των εντολών προγράμματος από την SPU επιτρέπει στις μεταφορές DMA να οργανώνονται με βολικό τρόπο ώστε να κρύβεται η καθυστέρηση της επικοινωνίας με τη μνήμη.

Κάθε μεταφορά DMA μπορεί να έχει μέγεθος μέχρι 16KB. Παρ' όλα αυτά, μόνο ο MFC της αντίστοιχης SPU μπορεί να εκδώσει εντολές λίστας DMA. Αυτές μπορούν να περιέχουν μέχρι και 2.048 μεταφορές DMA, κάθε μία μεγέθους μέχρι 16KB. Οι μεταφορές DMA είναι συναφείς ως προς την κύριο μνήμη. Πληροφορίες για τη μετάφραση εικονικών διευθύνσεων παρέχονται σε κάθε MFC από το λειτουργικό σύστημα που τρέχει στο PPE. Οι ιδιότητες του αποθηκευτικού χώρου του συστήματος (μετάφραση διευθύνσεων και προστασία) καθορίζονται με βάση τους πίνακες σελίδων και τμημάτων (page and segment tables) της αρχιτεκτονικής PowerPC. Παρόλο που προνομιούχο λογισμικό που τρέχει στο PPE μπορεί να απεικονίσει (map) διευθύνσεις τοπικών μνημών και συγκεκριμένους πόρους των MFC στο χώρο διευθύνσεων του κύριου αποθηκευτικού χώρου, δίνοντας έτσι τη δυνατότητα στο PPE ή σε άλλες SPUs στο σύστημα να προσπελαύνουν αυτούς τους πόρους, αυτή η απεικονισμένη μνήμη δεν είναι συναφής στο σύστημα.

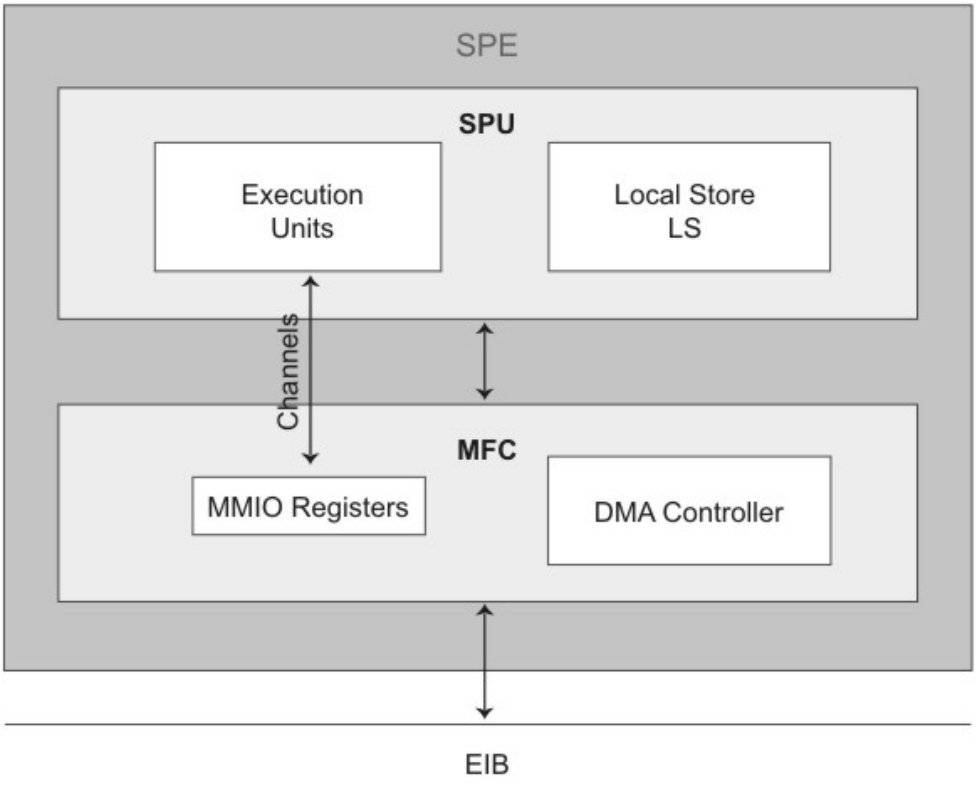
Τα SPEs παρέχουν ένα ντετερμινιστικό περιβάλλον λειτουργίας. Δεν διαθέτουν μνήμες cache, έτσι τα cache misses δεν είναι παράγοντας στην απόδοσή τους. Οι κανόνες σωλήνωσης και χρονοδιαγραφής (pipeline and scheduling) είναι απλοί, συνεπώς είναι εύκολο να αποφασίσουμε στατικά για την επιτεύξιμη επίδοση του κώδικα. Παρόλο που η τοπική μνήμη μοιράζεται μεταξύ ενεργειών ανάγνωσης και εγγραφής DMA, ενεργειών φόρτωσης και αποθήκευσης και προ-φόρτωσης εντολών, οι ενέργειες DMA είναι συσσωρευμένες και μπορούν να προσπελάσου την τοπική μνήμη το πολύ για έναν κύκλο κάθε οκτώ κύκλους. Η προ-φόρτωση εντολών φέρνει τουλάχιστον 17 εντολές στη σειρά από τη διεύθυνση-στόχο της διακλάδωσης. Έτσι, ο αντίκτυπος των ενεργειών DMA στις φορτώσεις, αποθηκεύεις και τους χρόνους εκτέλεσης του προγράμματος είναι, από σχεδιασμού, περιορισμένος.

Τα οκτώ πανομοιότυπα Συνεργατικά Επεξεργαστικά Στοιχεία (SPEs) είναι βελτιστοποιημένα για βαριές εφαρμογές αριθμητικών πράξεων στις οποίες τα απαιτούμενα από το πρόγραμμα δεδομένα και εντολές μπορούν να αναμένονται και να μεταφέρονται στις τοπικές μνήμες μέσω μεταφορών DMA καθώς το SPE εκτελεί υπολογισμούς χρησιμοποιώντας δεδομένα και εντολές που έχουν ήδη μεταφερθεί στην τοπική μνήμη από προηγούμενες DMA μεταφορές.

Τα συνεχούς ροής σύνολα δεδομένων (streaming data sets) στα 3D γραφικά, στα πολυμέσα και στις ευρυζωνικές τηλεπικοινωνίες είναι παραδείγματα εφαρμογών που τρέχουν αποδοτικά στα SPEs. Παρ' όλα αυτά, τα SPEs δεν είναι βελτιστοποιημένα να τρέχουν προγράμματα που έχουν έντονο αριθμό διακλαδώσεων, όπως ένα λειτουργικό σύστημα. Κάθε

SPE υποστηρίζει μία και μόνο διεργασία (context) ανά πάσα στιγμή. Τυπικά, το λειτουργικό σύστημα τρέχει στο PPE και τα νήματα χρήστη (user-mode threads) εκτελούνται στα SPEs.

Τα SPEs επιτυγχάνουν υψηλή επίδοση, εν μέρει εξαλείφοντας την επιβάρυνση που προκαλείται από τη μετάφραση διευθύνσεων κατά τις φορτώσεις και αποθηκεύσεις, τις ελεγχόμενες από το hardware κρυφές μνήμες (hardware-managed caches), την εκτός σειράς έκδοση εντολών (out-of-order instruction issue) και την πρόβλεψη διακλαδώσεων. Αντιθέτως, τα SPEs δίνουν έμφαση στην υψηλή υπολογιστική επίδοση που μπορεί να επιτευχθεί για εφαρμογές συνεχούς ροής δεδομένων (streaming-data applications), παρέχοντας ένα μεγάλο, ενοποιημένο αρχείο καταχωρητών (128 καταχωρητές των 128 bits έκαστος), τη δυνατότητα για διπλή έκδοση (dual issue) εντολών και υψηλό εύρος ζώνης DMA μεταξύ της τοπικής μνήμης και του κύριου αποθηκευτικού χώρου.



EIB	Element Interconnect Bus
LS	Local Store
MFC	Memory Flow Controller
SPE	Synergistic Processor Element
SPU	Synergistic Processor Unit

Το παραπάνω αποτελεί ένα πιο αναλυτικό διάγραμμα μπλοκ της αρχιτεκτονικής του SPE. Στο διάγραμμα αυτό φαίνονται οι δύο βασικές συνιστώσες του SPE:

- Συνεργατική Επεξεργαστική Μονάδα (Synergistic Processor Unit – SPU) – Η SPU εκτελεί εντολές SPU που ανακαλούνται από την τοπική της μνήμη, μεγέθους 256KB. Η SPU γεμίζει την τοπική της μνήμη με εντολές και δεδομένα χρησιμοποιώντας μεταφορές DMA οι οποίες αρχικοποιούνται από το λογισμικό που τρέχει είτε στην SPU είτε στο PPE.

- Ελεγκτής Ροής Μνήμης (Memory Flow Controller – MFC) – Ο MFC παρέχει τη διεπαφή μεταξύ της SPU και του Διαύλου Διασύνδεσης Στοιχείων (EIB), λειτουργώντας ως συνδετικός κρίκος μεταξύ της SPU και του κύριου αποθηκευτικού χώρου. Ο MFC διεκπεραιώνει μεταφορές DMA μεταξύ της τοπικής μνήμης της SPU και του κύριου αποθηκευτικού χώρου, ενώ υποστηρίζει και την ανταλλαγή μηνυμάτων μέσω mailboxes και signal-notification μεταξύ του SPE και του PPE ή άλλων συσκευών. Η SPU επικοινωνεί με τον MFC της μέσω καναλιών SPU (SPU channels). Το PPE και άλλες συσκευές (περιλαμβανομένων και άλλων SPEs) επικοινωνούν με τον MFC μέσω καταχωρητών εισόδου/εξόδου απεικονιζομένων στη μνήμη (memory-mapped I/O registers) που συνδέονται με τα κανάλια της SPU.

Συνεργατική Επεξεργαστική Μονάδα (Synergistic Processor Unit)

Κάθε ένα από τα οκτώ SPEs είναι ένας ανεξάρτητος επεξεργαστής με το δικό του μετρητή προγράμματος, αρχείο καταχωρητών και μεγέθους 256KB τοπική μνήμη.

Ένα SPE ενεργεί άμεσα σε εντολές και δεδομένα που βρίσκονται στη τοπική μνήμη του. Γεμίζει την τοπική μνήμη του αιτώντας μεταφορές DMA από τον MFC του, ο οποίος με τη σειρά του διαχειρίζεται τις μεταφορές DMA. Η SPU έχει εξειδικευμένες μονάδες για την εκτέλεση φορτώσεων και αποθηκεύσεων, μονάδα ακεραίων, μονάδα κινητής υποδιαστολής (μονής και διπλής ακρίβειας) και εντολές για τη διεπαφή με τα κανάλια.

Το μεγάλο αρχείο καταχωρητών (register file), αποτελούμενο από 128 καταχωρητές των 128 bits έκαστος, και η επίπεδο αρχιτεκτονική (όλα τα ορίσματα βρίσκονται αποθηκευμένα σε ένα και μόνο αρχείο καταχωρητών) επιτρέπουν την απόκρυψη της καθυστέρησης των εντολών χωρίς να υπάρχει πρόβλεψη (speculation). Το αρχείο καταχωρητών είναι ενοποιημένο, πράγμα που σημαίνει ότι όλοι οι τύποι δεδομένων (ακέραιοι, κινητής υποδιαστολής μονής και διπλής ακρίβειας, βαθμωτά δεδομένα, διανύσματα, λογικές τιμές, απλά bytes και άλλα) χρησιμοποιούν το ίδιο αρχείο καταχωρητών. Επίσης, το αρχείο καταχωρητών αποθηκεύει διευθύνσεις επιστροφής, αποτελέσματα συγκρίσεων και άλλα. Ως επακόλουθο του μεγάλου, ενοποιημένου αρχείου καταχωρητών δεν απαιτούνται για την

επίτευξη υψηλής επίδοσης ακριβές τεχνικές hardware, όπως η εκτός σειράς (out-of-order) εκτέλεση εντολών ή βαθιά πρόβλεψη (speculation).

Οι διευθύνσεις της τοπικής μνήμης μπορούν να απεικονιστούν από προνομιούχο λογισμικό του PPE στον κύριο χώρο ενεργών διευθύνσεων (main-storage effective-address space). Οι μεταφορές DMA μεταξύ της τοπικής μνήμης και του κύριου αποθηκευτικού χώρου είναι συναφείς στο σύστημα. Ένας δείκτης σε μια δομή δεδομένων δημιουργημένος στο PPE μπορεί να περαστεί στην SPU και αυτή μπορεί μετά να χρησιμοποιήσει αυτόν τον δείκτη για να εκδώσει μία εντολή DMA ώστε να φέρει τη δομή δεδομένων στην τοπική του μνήμη. Το λογισμικό του PPE μπορεί να χρησιμοποιήσει εντολές κλειδώματος (locking instructions) και τα mailboxes για να επιτύχει συγχρονισμό και αμοιβαίο αποκλεισμό (mutual exclusion).

Η αρχιτεκτονική της SPU έχει τους εξής περιορισμούς:

- Δεν μπορεί να γίνει απ' ευθείας πρόσβαση στην κύρια μνήμη. Η SPU προσπελαύνει τον κύριο αποθηκευτικό χώρο μόνο με χρήση των μεταφορών DMA του MFC της.

- Δεν μπορεί να γίνει απ' ευθείας έλεγχος του συστήματος, όπως για παράδειγμα έλεγχος των καταχωρήσεων του πίνακα σελίδων. Προνομιούχο λογισμικό του PPE παρέχει στην SPU τις πληροφορίες σχετικά με τη μετάφραση διευθύνσεων που χρειάζεται ο MFC της.

- Ως προς τις προσβάσεις της από την SPU της η τοπική μνήμη είναι ένας μη-προστατευόμενος και μη-μεταφραζόμενος αποθηκευτικός χώρος.

Καταχωρητές της SPU

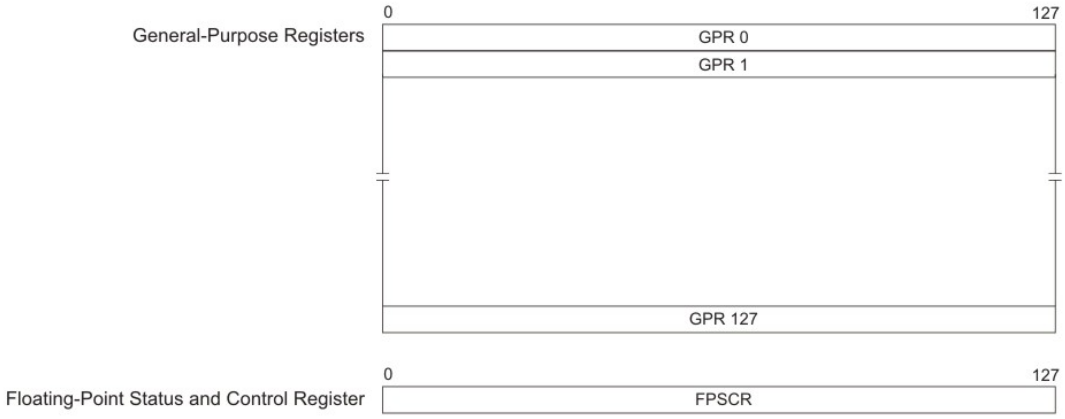
Παρακάτω δίνεται το πλήρες σύνολο των καταχωρητών χρήστη της SPU. Όλες οι υπολογιστικές εντολές επενεργούν μόνο σε καταχωρητές – δεν υπάρχουν υπολογιστικές εντολές που να τροποποιούν τον αποθηκευτικό χώρο (την τοπική μνήμη δηλαδή).

Οι καταχωρητές της SPU περιλαμβάνουν:

- Καταχωρητές Γενικού Σκοπού (General-Purpose Registers – GPRs) – Όλοι οι τύποι δεδομένων μπορούν να αποθηκευτούν στους 128bit Καταχωρητές Γενικού Σκοπού, το πλήθος των οποίων ανέρχεται στους 128.

- Καταχωρητής Κατάστασης και Ελέγχου Κινητής Υποδιαστολής (Floating-Point Status and Control Register – FPSCR) – Ο επεξεργαστής ενημερώνει τον 128bit Καταχωρητή Κατάστασης και Ελέγχου Κινητής Υποδιαστολής

έπειτα από κάθε πράξη κινητής υποδιαστολής προκειμένου να καταγράψει πληροφορίες σχετικά με το αποτέλεσμα και κάθε σχετική εξαίρεση (exception).



Πράξεις Κινητής Υποδιαστολής

Η SPU εκτελεί πράξεις κινητής υποδιαστολής τόσο απλής όσο και διπλής ακρίβειας. Οι πράξεις απλής ακρίβειας εκτελούνται με τέσσερα στοιχεία ανά διάνυσμα (4-way SIMD fashion), πλήρως pipelined, ενώ οι εντολές διπλής ακρίβειας είναι μερικώς pipelined. Οι αναπαραστάσεις των δεδομένων απλής και διπλής ακρίβειας είναι αυτές που ορίζονται από το πρότυπο IEEE Standard 754, όμως τα αποτελέσματα που υπολογίζονται από τις εντολές απλής ακρίβειας δεν συμμορφώνονται πλήρως με αυτό το πρότυπο.

Για τις πράξεις απλής ακρίβειας το εύρος των κανονικοποιημένων αριθμών έχει επεκταθεί πέρα από το πρότυπο της IEEE. Οι μη-μηδενικοί αριθμοί που μπορούν να αναπαρασταθούν κυμαίνονται από $X_{min} = 2^{126}$ έως $X_{max} = 2^{-32} \times 2^{128}$. Εάν το ακριβές αποτέλεσμα είναι μεγαλύτερο από X_{max}, το στρογγυλοποιημένο αποτέλεσμα εξισώνεται με το X_{max} με το κατάλληλο πρόσημο. Εάν το ακριβές αποτέλεσμα είναι μικρότερο από το X_{min}, το στρογγυλοποιημένο αποτέλεσμα εξισώνεται με το 0. Ένα μηδενικό αποτέλεσμα θεωρείται πάντα ως θετικό μηδέν.

Οι πράξεις κινητής υποδιαστολής απλής ακρίβειας υλοποιούν την αριθμητική IEEE 754 με τις εξής αλλαγές:

- Υποστηρίζεται μόνο ένας τρόπος στρογγυλοποίησης: στρογγυλοποίηση προς το μηδέν, γνωστό και ως truncation
- Ορίσματα με μη-ορισμένες αναπαραστάσεις θεωρούνται μηδενικά και αποτελέσματα με μη-ορισμένες αναπαραστάσεις εξισώνονται με το μηδέν

- Αριθμοί με εκθέτη που όλα του τα bits είναι 1 αντιμετωπίζονται ως κανονικοποιημένοι αριθμοί και όχι ως άπειρο (inf) ή not-a-number (NaN).

Οι πράξεις διπλής ακρίβειας δεν υποστηρίζουν την εξαίρεση IEEE precise trap. Εάν ένα αποτέλεσμα διπλής ακρίβειας με μη-ορισμένη αναπαράσταση ή τύπου NaN δεν συμμορφώνεται με το πρότυπο IEEE 754 τότε η απόκλιση καταγράφεται σε ένα από τα bits του Καταχωρητή Κατάστασης και Ελέγχου Κινητής Υποδιαστολής (FPSCR).

Οι εντολές διπλής ακρίβειας εκτελούνται ως δύο πράξεις διπλής ακρίβειας με δύο στοιχεία ανά διάνυσμα (2-way SIMD fashion). Παρ' όλα αυτά, η SPU έχει τη δυνατότητα να εκτελέσει μόνο μία πράξη διπλής ακρίβειας ανά κύκλο. Έτσι, η SPU ολοκληρώνει τις εντολές διπλής ακρίβειας διαχωρίζονται τα ορίσματα SIMD και εκτελώντας τις δύο πράξεις σε διαδοχικά instruction slots στο pipeline. Παρόλο που οι εντολές διπλής ακρίβειας έχουν καθυστέρηση (latency) 13 κύκλων, μόνο οι τελευταίοι 7 κύκλοι είναι pipelined. Καμία άλλη εντολή δεν εκδίδεται παράλληλα με εντολές διπλής ακρίβειας και καμία εντολή οποιουδήποτε τύπου δεν εκδίδεται για έξι κύκλους μετά την έκδοση μίας εντολής διπλής ακρίβειας.

Τοπική Μνήμη (Local Store)

Η Τοπική Μνήμη (Local Store – LS) μπορεί να θεωρηθεί ως μία cache ελεγχόμενη από το λογισμικό και η οποία γεμίζει και αδειάζει από δεδομένα μέσω μεταφορών DMA.

Τα κυριότερα χαρακτηριστικά της τοπικής μνήμης είναι τα εξής:

- Χρησιμοποιείται και για εντολές και για δεδομένα
- Εύρος ζώνης 16 bytes ανά κύκλο για εντολές φόρτωσης και αποθήκευσης, τα δεδομένα είναι ευθυγραμμισμένα κατά τετραπλή λέξη (quadword alignment) και μόνο
- Εύρος ζώνης 128 bytes ανά κύκλο για μεταφορές DMA
- Προφόρτωση εντολών (instruction prefetch) 128 bytes ανά κύκλο

Μπορεί να προκύψει ανταγωνισμός για την πρόσβαση στην τοπική μνήμη από:

- Εντολές φόρτωσης
- Εντολές αποθήκευσης
- Αναγνώσεις DMA

- Εγγραφές DMA
- Ανακλήσεις εντολών (instruction fetches)

Η SPU διαιτητεύει τις προσβάσεις στην τοπική μνήμη σύμφωνα με τις ακόλουθες προτεραιότητες (ξεκινώντας από την υψηλότερη προτεραιότητα):

1. Αναγνώσεις και εγγραφές DMA από το PPE ή μία συσκευή εισόδου/εξόδου
2. Φορτώσεις και αποθηκεύσεις της SPU
3. Προφόρτωση εντολών (instruction prefetch)

Ο ακόλουθος πίνακας συνοψίζει τις προτεραιότητες της διαιτησίας της τοπικής μνήμης και τα μεγέθη των μεταφορών.

Εκτέλεση	**Μέγεθος μεταφοράς (bytes)**	**Προτεραιότητα**	**Μέγιστη απασχόληση της τοπικής μνήμης (κύκλοι SPU)**	**Μέσον πρόσβασης**
MMIO	≤ 16	1 – Υψηλότερη	1/8	Line Interface
DMA	≤ 128	1		
DMA-List Transfer-Element Fetch	128	1	1/4	Quadword Interface
ECC Scrub	16	2	1/10	
SPU Load/Store	16	3	1	
Hint Fetch	128	3	1	Line Interface
Inline Fetch	128	4 - Χαμηλότερη	1/16 για inline κώδικα	

Οι αναγνώσεις και εγγραφές DMA έχουν πάντα την υψηλότερη προτεραιότητα. Επειδή το hardware υποστηρίζει αναγνώσεις και εγγραφές DMA των 128 bits αυτές οι λειτουργίες απασχολούν το πολύ έναν κύκλο κάθε οκτώ (έναν στους δεκαέξι για αναγνώσεις DMA και έναν στους δεκαέξι για εγγραφές DMA) στην τοπική μνήμη. Έτσι, εκτός των περιπτώσεων πολύ βελτιστοποιημένου κώδικα, ο αντίκτυπος των αναγνώσεων και εγγραφών DMA στη διαθεσιμότητα για φορτώσεις, αποθηκεύσεις και ανακλήσεις εντολών από τοπική μνήμη μπορεί να αγνοηθεί.

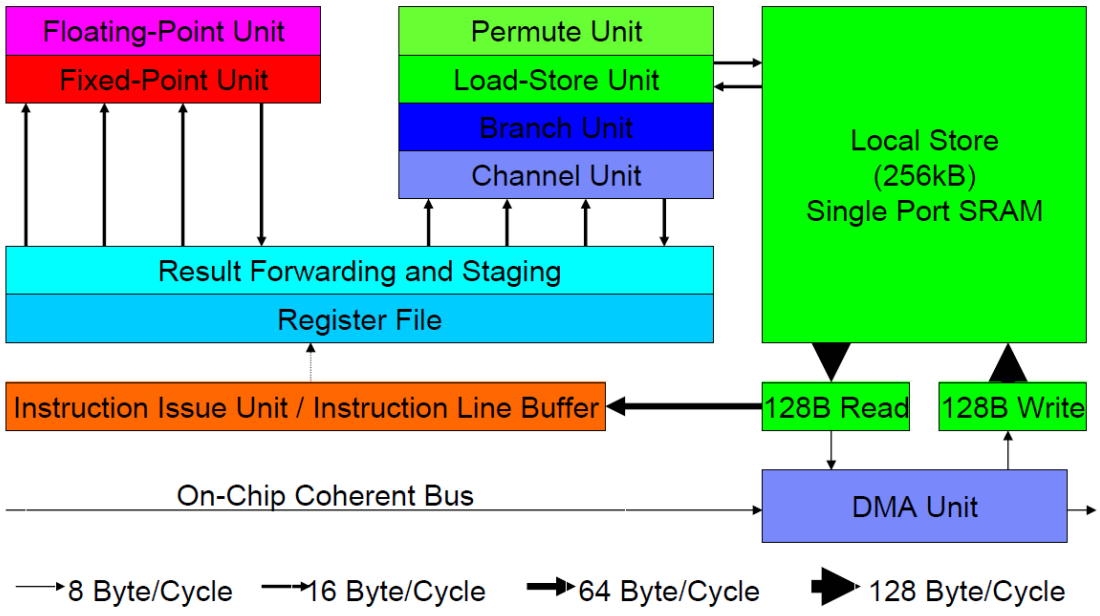
Η επόμενη υψηλότερη προτεραιότητα (για ενέργειες που προγραμματίζονται από το χρήστη) μετά τις αναγνώσεις και εγγραφές DMA δίνεται στις εντολές φόρτωσης και

αποθήκευσης. Η αιτιολόγηση για αυτό το γεγονός είναι ότι οι εντολές φόρτωσης και αποθήκευσης συνήθως βοηθούν στην γρηγορότερη εκτέλεση ενός προγράμματος, ενώ οι ανακλήσεις εντολών είναι πολύ συχνά υποθετικές. Το SPE υποστηρίζει μόνο φορτώσεις και αποθηκεύσεις των 16 bytes που είναι ευθυγραμμισμένες στα 16 bytes. Χρησιμοποιεί επίσης μία δεύτερη εντολή (ανακάτεμα bytes – byte shuffle) για να τοποθετήσει τα bytes σε κάποια άλλη σειρά εάν, για παράδειγμα, το πρόγραμμα απαιτεί μια ποσότητα μόνο 4 bytes ή μία ποσότητα με διαφορετική ευθυγράμμιση δεδομένων. Για να αποθηκευτεί κάτι το οποίο δεν είναι ευθυγραμμισμένο θα πρέπει να χρησιμοποιηθεί μία λειτουργία ανάγνωσης-τροποποίησης-εγγραφής.

Η χαμηλότερη προτεραιότητα για την πρόσβαση στην τοπική μνήμη δίνεται για την ανάκληση εντολών, της οποίας υπάρχουν τρία ήδη: ανακλήσεις μετά από καθαρισμό του pipeline (flush-initiated fetches), ανακλήσεις inline και ανακλήσεις υποδείξεων (hint fetches). Οι ανακλήσεις εντολών φορτώνουν 32 εντολές ανά αίτηση της SPU, προσπελάζοντας ταυτόχρονα όλες τις σειρές (banks) της τοπικής μνήμης. Επειδή η τοπική μνήμη έχει μόνο μία θύρα εισόδου/εξόδου είναι σημαντικό οι ενέργειες DMAκαι ανάκλησης εντολών να μεταφέρουν όσο το δυνατόν περισσότερα χρήσιμα δεδομένα σε κάθε αίτηση προς την τοπική μνήμη.

Σωληνώσεις και κανόνες διπλής έκδοσης (dual-issue)

Η SPU έχει δύο σωληνώσεις, τις επονομαζόμενες άρτια (σωλήνωση 0) και περιττή (σωλήνωση 1). Σε αυτές τις σωληνώσεις η SPU μπορεί να εκδώσει και να διεκπεραιώσει μέχρι και δύο εντολές ανά κύκλο, μία εντολή σε κάθε σωλήνωση. Οι δύο σωληνώσεις φαίνονται στο παρακάτω διάγραμμα:



Το εάν μία εντολή θα πάει στην άρτια ή στην περιττή σωλήνωση εξαρτάται από τον τύπο της εντολής, ο οποίος σχετίζεται με τη μονάδα εκτέλεσης (execution unit) που εκτελεί τη λειτουργία. Κάθε μονάδα εκτέλεσης ανατίθεται σε μία από τις δύο σωληνώσεις. Ο παρακάτω πίνακας συνοψίζει τους τύπους εντολών, τις καθυστερήσεις και τις αναθέσεις στις σωληνώσεις.

Κατηγορία Εντολής	**Περιγραφή**	**Καθυστέρηση (κύκλοι ρολογιού)**	**Σωλήνωση**
LS	Φορτώσεις και αποθηκεύσεις	6	Περιττή
HB	Hints διακλαδώσεων	15	Περιττή
BR	Ανάλυση διακλαδώσεων	4	Περιττή
CH	Διεπαφή καναλιού, ειδικής χρήσεως καταχωρητές	6	Περιττή
SP	Μονής ακρίβειας κινητής υποδιαστολής	6	Άρτια
DP	Διπλής ακρίβειας κινητής υποδιαστολής	13[2]	Άρτια
FI	Κινητής υποδιαστολής ακεραίων	7	Άρτια
SH	Ανακάτεμα (shuffle)	4	Περιττή
FX	Απλή σταθερής υποδιαστολής	2	Άρτια
WS	Περιστροφή και ολίσθηση λέξης	4	Άρτια
BO	Πράξεις σε bytes	4	Άρτια
NOP	No operation (εκτέλεση)	-	Άρτια
LNOP	No operation (φόρτωση)	-	Περιττή

Σημειώσεις:

1. Διακλαδώσεις inline ή διακλαδώσεις που έχουν προβλεφθεί σωστά έχουν μηδενική καθυστέρηση. Η ποινή για λανθασμένη πρόβλεψη διακλάδωσης είναι 18-19 κύκλοι.

2. Έξι κύκλοι σε μια εντολή κινητής υποδιαστολής διπλής ακρίβειας είναι καθυστερήσεις έκδοσης εντολής (instruction issue stalls). Καμία εντολή οποιουδήποτε τύπου δεν μπορεί να εκδοθεί για έξι κύκλους μετά την έκδοση μίας εντολής κινητής υποδιαστολής διπλής ακρίβειας.

Η SPU εκδίδει όλες τις εντολές σύμφωνα με τη σειρά του προγράμματος, ανάλογα με την ανάθεση κάθε εντολής σε μία σωλήνωση. Κάθε εντολή είναι κομμάτι ενός ζεύγους εντολών με ευθυγράμμιση διπλής λέξης (doubleword alignment) που ονομάζεται ομάδα ανάκλησης

(fetch group). Μία ομάδα ανάκλησης μπορεί να έχει μία ή δύο έγκυρες εντολές, αλλά πρέπει να είναι ευθυγραμμισμένη σε όρια διπλής λέξης. Αυτό σημαίνει ότι η πρώτη εντολή σε μία ομάδα ανάκλησης προέρχεται από μία άρτια διεύθυνση λέξης (δηλαδή διεύθυνση της μορφής $\ldots 000_{(2)}$) και η δεύτερη εντολή από μία περιττή διεύθυνση λέξης (διεύθυνση της μορφής $\ldots 100_{(2)}$).

Η SPU επεξεργάζεται τις ομάδες ανάκλησης κατά μία κάθε φορά, συνεχίζοντας στην επόμενη ομάδα ανάκλησης όταν η τρέχουσα ομάδα αδειάσει. Μία εντολή είναι εκδόσιμη (issuable) όταν οι εξαρτήσεις ως προς τους καταχωρητές ικανοποιούνται και δεν υπάρχει δομικός κίνδυνος (resource conflict) με προηγούμενες εντολές ή ενέργειες DMA ή κώδικα διόρθωσης λαθών (error-correction code – ECC).

Η διπλή έκδοση πραγματοποιείται όταν μία ομάδα ανάκλησης έχει δύο εκδόσιμες εντολές από τις οποίες η πρώτη μπορεί να εκτελεστεί στην άρτια σωλήνωση και η δεύτερη εντολή μπορεί να εκτελεστεί στην περιττή σωλήνωση. Εάν δεν μπορεί να γίνει διπλή έκδοση σε μία ομάδα ανάκλησης, αλλά μπορεί να εκδοθεί η πρώτη της εντολή, τότε η πρώτη εντολή εκδίδεται στη κατάλληλη σωλήνωση εκτέλεσης και η δεύτερη εντολή κρατείται μέχρις ότου είναι εφικτή η έκδοσή της. Μετά την έκδοση και των δύο εντολών της τρέχουσας ομάδας ανάκλησης φορτώνεται η νέα, επόμενη ομάδα.

Ελεγκτής Ροής Μνήμης (Memory Flow Controller)

Οι κύριες λειτουργίες του Ελεγκτή Ροής Μνήμης (Memory Flow Controller – MFC) είναι η σύνδεση της SPU με το δίαυλο EIB και η υποστήριξη μεταφορών DMA μεταξύ του κύριου αποθηκευτικού χώρου και της τοπικής μνήμης. Ο MFC διατηρεί και επεξεργάζεται ουρές εντολών DMA από την SPU του ή το PPE ή άλλες συσκευές. Ο ελεγκτής DMA (DMA Controller - DMAC) του MFC εκτελεί τις εντολές DMA. Αυτό το χαρακτηριστικό επιτρέπει στην SPU να συνεχίσει την εκτέλεση των εντολών της παράλληλα με τις μεταφορές DMA του MFC.

Για να είναι εφικτές οι μεταφορές DMA μεταξύ του κύριου αποθηκευτικού χώρου και της τοπικής μνήμης, προνομιούχο λογισμικό στο PPE παρέχει στους πόρους της τοπικής μνήμης και του MFC, όπως είναι οι απεικονισμένοι στη μνήμη καταχωρητές εισόδου/εξόδου (MMIO), τα «ψευδώνυμα» των ενεργών διευθύνσεων της κύριας μνήμης. Αυτό επιτρέπει στο λογισμικό του PPE ή άλλων SPUs και συσκευών να προσπελαύνουν τους πόρους του MFC και να ελέγχουν την SPU. Προνομιούχο λογισμικό στο PPE παρέχει επίσης στον MFC πληροφορίες για τη μετάφραση διευθύνσεων για χρήση στις μεταφορές DMA. Οι μεταφορές DMA είναι συναφείς ως προς τον αποθηκευτικό χώρο του συστήματος. Ιδιότητες του αποθηκευτικού χώρου του συστήματος (μετάφραση διευθύνσεων και προστασία) γίνονται σύμφωνα με τους πίνακες σελίδων και τμημάτων (page and segment tables) της αρχιτεκτονικής PowerPC.

Ο MFC υποστηρίζει κανάλια και αντιστοιχισμένους σε αυτά καταχωρητές MMIO ώστε αυτοί να χρησιμοποιούνται για τοποθέτηση στην ουρά και παρακολούθηση εντολών DMA, παρακολούθηση γεγονότων SPU, διεξαγωγή επικοινωνίας μεταξύ επεξεργαστών μέσω των mailboxes και του signal-notification, πρόσβαση βοηθητικών πόρων, όπως είναι ο χρονομετρητής της SPU (decrementer timer), και άλλες λειτουργίες. Εκτός από την υποστήριξη μεταφορών DMA, καναλιών και καταχωρητών MMIO, ο MFC υποστηρίζει επίσης λειτουργίες δέσμευσης εύρους ζώνης διαύλου (bus-bandwidth reservation features) και συγχρονίζει διαδικασίες μεταξύ της SPU και άλλων επεξεργαστικών μονάδων στο σύστημα.

Κανάλια

Τα κανάλια (channels) είναι μονόδρομες διεπαφές ανταλλαγής μηνυμάτων (unidirectional message-passing interfaces) που υποστηρίζουν 32bit μηνύματα και εντολές. Πολλά από τα κανάλια παρέχουν επικοινωνία μεταξύ του SPE και του MFC του, ο οποίος στη συνέχεια διαμεσολαβεί για την επικοινωνία με το PPE και άλλες συσκευές. Το λογισμικό της SPU χρησιμοποιεί ειδικές εντολές καναλιών (channel instructions) για να διαβάσει και να γράψει καταχωρητές και ουρές καναλιών. Το λογισμικό στο PPE και σε άλλες συσκευές χρησιμοποιεί εντολές φόρτωσης και αποθήκευσης για να διαβάσει και να γράψει τους MMIO καταχωρητές του MFC που αντιστοιχούν στα κανάλια της SPU.

Ακολουθεί πίνακας των καναλιών και των ιδιοτήτων τους. Από τον πίνακα αυτόν έχουν παραλειφθεί τα δεσμευμένα και τα προνομιούχα κανάλια.

Κανάλι	Όνομα	Μνημονικό	Μέγεθος (bits)	R/W	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 Decremeter	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

Κανάλι	Όνομα	Μνημονικό	Μέγεθος (bits)	R/W	Blocking
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 Load 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 Acknowledgment	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

Κάθε κανάλι έχει έναν αντίστοιχο μετρητή που υποδεικνύει την εναπομείνασα χωρητικότητα (το μέγιστο πλήθος εκκρεμών μεταφορών) σε αυτό το κανάλι. Αυτός ο μετρητής μειώνεται κάθε φορά που εκδίδεται στο κανάλι μία εντολή καναλιού και αυξάνεται όταν μία ενέργεια σχετιζόμενη με αυτό το κανάλι ολοκληρωθεί. Κάθε κανάλι είναι υλοποιημένο είτε με blocking είτε με non-blocking σημασιολογία. Τα blocking κανάλια προκαλούν παύση της εκτέλεσης στο SPE (αναστέλλουν την εκτέλεση και το SPE μπαίνει σε κατάσταση χαμηλής κατανάλωσης) όταν το SPE διαβάζει ή γράφει ένα κανάλι που ο μετρητής του είναι μηδέν.

Τα κυριότερα χαρακτηριστικά των ενεργειών πάνω στα κανάλια του SPE είναι τα ακόλουθα:

- Όλες οι ενέργειες στη διεπαφή του καναλιού είναι μονόδρομες
- Κάθε ενέργεια του καναλιού είναι ανεξάρτητη από κάθε άλλη ενέργεια
- Υποστηρίζονται σειριακές λειτουργίες ανάγνωσης και εγγραφής
- Η εξωτερική πρόσβαση (external access) στους ΜΜΙΟ καταχωρητές έχει υψηλότερη προτεραιότητα από τις ενέργειες των καναλιών
- Οι λειτουργίες των καναλιών πραγματοποιούνται σύμφωνα με τη σειρά του προγράμματος
- Απόπειρες ανάγνωσης από δεσμευμένα κανάλια επιστρέφουν μηδενικά
- Απόπειρες εγγραφής σε δεσμευμένα κανάλια δεν έχουν κανένα αποτέλεσμα
- Απόπειρες ανάγνωσης των μετρητών δεσμευμένων καναλιών επιστρέφουν μηδέν

Εντολές Καναλιών

Το σύνολο εντολών της αρχιτεκτονικής της SPU ορίζει τρεις εντολές καναλιών: rdch, wrch, rchcnt. Ο παρακάτω πίνακας συνοψίζει αυτές τις εντολές.

Εντολή	**Εντολή του Assembler**	**Intrinsic της Γλώσσας C**	**Περιγραφή**
Read Channel	rdch	spu_readch spu_readchqw	Προκαλεί την ανάγνωση δεδομένων από το επιλεγμένο κανάλι και την αποθήκευσή τους στον επιλεγμένο GPR

Write Channel	wrch	spu_writech spu_writechqw	Προκαλεί την ανάγνωση δεδομένων από τον επιλεγμένο GPR και την αποθήκευσή τους στο επιλεγμένο κανάλι
Read Channel Count	rchcnt	spu_readchcnt	Προκαλεί την αποθήκευση στον επιλεγμένο GPR την τιμή του αντίστοιχου μετρητή του επιλεγμένου καναλιού

Το λογισμικό που τρέχει σε ένα SPE χρησιμοποιεί τις εντολές καναλιών για να γράψει παραμέτρους και να βάλει σε ουρά εντολές του MFC. Στον παραπάνω πίνακα δίνονται τόσο οι εντολές σε assembly όσο και τα αντίστοιχα intrinsics της C.

Εάν το προς εγγραφή κανάλι είναι non-blocking τότε μία εντολή wrch μπορεί να εκδοθεί άσχετα με την τιμή του μετρητή καναλιού για το συγκεκριμένο κανάλι. Εάν το προς εγγραφή κανάλι είναι blocking τότε μία εντολή wrch η οποία εκδίδεται όταν ο μετρητής για το συγκεκριμένο κανάλι είναι ίσος με 0 θα προκαλέσει παύση λειτουργίας στο SPE. Η παύση λειτουργίας εξ αιτίας μίας εντολής wrch μπορεί να είναι χρήσιμη αφού εξοικονομεί ενέργεια, όμως, προκειμένου να αποφευχθεί η παύση αυτή, το λογισμικό θα πρέπει πριν από την έκδοση μίας wrch εντολής να διαβάζει τον μετρητή του καναλιού ώστε να επιβεβαιωθεί ότι αυτός δεν είναι μηδέν.

Η μέθοδος που χρησιμοποιείται για τον προσδιορισμό του μετρητή ενός καναλιού εξαρτάται από το πρόγραμμα. Το πρόγραμμα μπορεί να κάνει polling στον μετρητή του αντίστοιχου καναλιού χρησιμοποιώντας την εντολή rchcnt ή να εκδώσει μία εντολή wrch. Εάν το πρόγραμμα εκδώσει μία εντολή wrch, το SPE σταματά τη λειτουργία του περιμένοντας να λάβει μία απάντηση από το προς εγγραφή κανάλι.

Όταν ένα πρόγραμμα του SPE χρειάζεται να λάβει πληροφορίες χρησιμοποιεί την εντολή rdch. Συνήθως, αυτές οι πληροφορίες κρατούνται σε έναν καταχωρητή SPE. Οι πληροφορίες μπορούν να φορτωθούν σε αυτόν τον καταχωρητή μέσω της διεπαφής του καναλιού, εκτελώντας μία λειτουργία ανάγνωσης δεδομένων και φόρτωσης (read-data-load).

- Εάν το προς ανάγνωση κανάλι είναι non-blocking, τότε μία εντολή rdch μπορεί να εκδοθεί ανεξάρτητα της τιμής του μετρητή του αντίστοιχου καναλιού.

- Στο SPE, εάν το κανάλι είναι blocking, το SPE δεν μπορεί να διαβάσει από τον αντίστοιχο καταχωρητή μέχρις ότου ο μετρητής για αυτό το κανάλι δείξει

ότι τα δεδομένα είναι έγκυρα (όταν δηλαδή ο μετρητής είναι μεγαλύτερος του μηδενός).

- Εάν ο μετρητής είναι μηδέν, τότε δεν υπάρχουν δεδομένα στο κανάλι και το SPE αναστέλλει τη λειτουργία του μέχρι να συμβούν γεγονότα που σχετίζονται με αυτό το κανάλι.

Αυτές οι ενέργειες μπορούν να περιλαμβάνουν την ενημέρωση του καναλιού MFC_RdTagStat, την εγγραφή δεδομένων από το PPE στον αντίστοιχο MMIO καταχωρητή του καναλιού (όπως π.χ. ένα κανάλι mailbox) ή άλλες ενέργειες. Η μέθοδος που χρησιμοποιείται για την εξακρίβωση του μετρητή εξαρτάται από το πρόγραμμα. Το πρόγραμμα μπορεί να :

- Κάνει polling στο κανάλι μέσω του αντίστοιχου καταχωρητή χρησιμοποιώντας την εντολή rchcnt

- Να εκδώσει την εντολή rdch

Εάν το πρόγραμμα εκδώσει μία εντολή rdch, τότε το SPE αναστέλλει τη λειτουργία του, περιμένοντας μέχρι να φορτωθούν έγκυρα δεδομένα.

Οι εντολές των καναλιών αρχιτεκτονικά έχουν εύρος 128 bits, όμως στην Cell Broadband Engine το σύνολο εντολών για τα κανάλια χρησιμοποιεί μόνο τα 32 bits της πιο αριστερής λέξης (left-most word) του καταχωρητή.

M a i l b o x e s

Τα mailboxes είναι ουρές που υποστηρίζουν την ανταλλαγή 32bit μηνυμάτων μεταξύ ενός SPE και άλλων συσκευών. Κάθε ουρά mailbox έχει ένα αντιστοιχισμένο κανάλι SPE όπως και έναν αντίστοιχο για αυτό το κανάλι MMIO καταχωρητή.

Παρέχονται δύο ουρές mailbox, χωρητικότητα ενός μηνύματος έκαστη, για την αποστολή μηνυμάτων από το SPE:

- SPU Write Outbound Mailbox

- SPU Write Outbound Interrupt Mailbox

Επίσης, παρέχεται και μία ουρά mailbox, χωρητικότητας τεσσάρων μηνυμάτων, για την αποστολή μηνυμάτων προς το SPE:

- SPU Read Inbound Mailbox

Κάθε mailbox έχει ένα αντιστοιχισμένο κανάλι SPE όπως και έναν αντίστοιχο MMIO καταχωρητή. Ένα πρόγραμμα SPE χρησιμοποιεί τις εντολές rdch και wrch για να προσπελάσει το mailbox. Το PPE και άλλοι επεξεργαστές χρησιμοποιούν εντολές φόρτωσης και αποθήκευσης για να προσπελάσουν τις αντίστοιχες MMIO διευθύνσεις.

Τα δεδομένα που εγγράφονται από ένα πρόγραμμα SPE σε κάποιο από αυτά τα mailboxes χρησιμοποιώντας μία εντολή wrch είναι διαθέσιμα σε οποιονδήποτε επεξεργαστή ή συσκευή που διαβάζει τον αντίστοιχο MMIO καταχωρητή. Τα δεδομένα που εγγράφονται από κάποια συσκευή στο SPU Read Inbound Mailbox χρησιμοποιώντας μία εγγραφή MMIO είναι διαθέσιμα σε ένα πρόγραμμα SPE μέσω της ανάγνωσης αυτού του mailbox με μια εντολή rdch ή rchcnt. Μία ανάγνωση MMIO από οποιοδήποτε SPU Write Outbound Mailbox ή μία εγγραφή στο SPU Read Inbound Mailbox μπορούν να προγραμματιστούν ώστε να προκαλούν ένα γεγονός SPU (SPU Event). Το γεγονός μπορεί με τη σειρά του να προκαλέσει μία διακοπή SPU (SPU Interrupt). Μπορεί επίσης μία εντολή wrch στο SPU Write Outbound Interrupt Mailbox να προγραμματιστεί ώστε να προκαλέσει μία διακοπή σε έναν επεξεργαστή ή άλλη συσκευή.

Κάθε φορά που ένα πρόγραμμα στο PPE γράφει στην τεσσάρων-θέσεων ουρά SPU Read Inbound Mailbox, ο μετρητής του αντίστοιχου καναλιού αυξάνεται. Κάθε φορά που ένα πρόγραμμα της SPU διαβάζει την ουρά του mailbox, ο μετρητής του καναλιού μειώνεται. Το mailbox είναι μία ουρά FIFO – το πρόγραμμα του SPE διαβάζει πρώτα τα παλαιότερα δεδομένα. Εάν το πρόγραμμα του PPE γράψει περισσότερες από τέσσερεις φορές χωρίς το πρόγραμμα του SPE να διαβάσει τα δεδομένα, τότε ο μετρητής του καναλιού μένει στο τέσσερα και η τέταρτη θέση στην ουρά περιέχει τα δεδομένα που εγγράφηκαν τελευταία από το PPE. Για παράδειγμα, εάν το πρόγραμμα του PPE γράψει πέντε φορές προτού το πρόγραμμα του SPE διαβάσει τα δεδομένα, τότε τα δεδομένα που διαβάζονται είναι το πρώτο, το δεύτερο, το τρίτο και το πέμπτο στοιχείο που εγγράφηκαν. Το πέμπτο στοιχείο έχει γραφτεί πάνω από το τέταρτο, σβήνοντάς το.

Οι λειτουργίες των mailbox είναι λειτουργίες blocking: μια εγγραφή σε έναν καταχωρητή outbound mailbox που είναι ήδη γεμάτο προκαλεί αναστολή της λειτουργίας της SPE μέχρι να αδειάσει μία θέση στο mailbox από μια ανάγνωση εκ μέρους του PPE. Παρομοίως, μία ανάγνωση του SPE από ένα άδειο inbound mailbox αναστέλλει τη λειτουργία του SPE μέχρι το PPE (ή κάποιο άλλο SPE) γράψει κάτι σε αυτό το mailbox. Εάν ο μετρητής χωρητικότητας του καναλιού είναι μηδέν για ένα κανάλι το οποίο έχει ρυθμιστεί ώστε να είναι blocking, τότε μία εντολή καναλιού που εκδίδεται σε αυτό το κανάλι προκαλεί αναστολή της λειτουργίας του SPE και διακοπή της έκδοσης περαιτέρω εντολών μέχρι να ολοκληρωθεί η ανάγνωση του καναλιού. Για να αποφευχθεί η αναστολή λειτουργίας σε αυτήν την περίπτωση το πρόγραμμα του SPE χρειάζεται να διαβάσει τον καταχωρητή-μετρητή που

συνδέεται με το συγκεκριμένο mailbox και να αποφασίσει εάν θα διαβάσει από/ γράψει στο mailbox ή όχι.

Υπάρχουν τουλάχιστον τρεις τρόποι για τη διαχείριση αναμενόμενων μηνυμάτων των mailboxes:

- Το πρόγραμμα του SPE διαβάζει το κανάλι (rdch), ενέργεια η οποία θα μπλοκάρει μέχρι να φτάσει κάποιο δεδομένο

- Το πρόγραμμα του SPE διαβάζει τον μετρητή του καναλιού (rchcnt), ενέργεια η οποία θα επιστρέψει είτε 0 είτε 1, ανάλογα με το αν υπάρχουν ή όχι δεδομένα – το πρόγραμμα μπορεί μετά να αποφασίσει τι θα κάνει

- Το πρόγραμμα του SPE χρησιμοποιεί τη δυνατότητα των διακοπών ώστε να αντιδράσει σε κάποιο γεγονός που σχετίζεται με το mailbox

Παρόλο που τα mailboxes προορίζονται κυρίως για επικοινωνία μεταξύ του PPE και των SPEs, μπορούν επίσης να χρησιμοποιηθούν για επικοινωνία μεταξύ ενός SPE και άλλων SPEs, επεξεργαστών ή συσκευών. Για να συμβεί αυτό όμως, χρειάζεται το προνομιούχο λογισμικό να έχει επιτρέψει σε ένα SPE να έχει πρόσβαση στους καταχωρητές των mailboxes σε κάποιο άλλο SPE. Εάν το προνομιούχο λογισμικό δεν επιτρέψει κάτι τέτοιο, τότε μόνο οι επικοινωνίες μέσω της μνήμης του συστήματος είναι διαθέσιμες για επικοινωνία μεταξύ SPEs.

Ειδοποιήσεις μέσω Σημάτων (Signal Notification)

Τα κανάλια ειδοποιήσεων μέσω σημάτων (signal-notification channels), ή απλά σήματα (signals), είναι καταχωρητές για εισερχόμενα (σε ένα SPE) δεδομένα. Μπορούν να χρησιμοποιηθούν από άλλα SPEs, το PPE ή άλλες συσκευές προκειμένου να αποστείλουν πληροφορίες, όπως μία σημαία (flag) συγχρονισμού, σε ένα SPE.

Κάθε SPE έχει δύο 32bit καταχωρητές για ειδοποιήσεις μέσω σημάτων, κάθε ένας από τους οποίους αντιστοιχεί σε έναν καταχωρητή MMIO στον οποίον εγγράφονται τα δεδομένα του σήματος από τον αποστέλλοντα επεξεργαστή. Αντίθετα από την επικοινωνία μέσω mailboxes, οι αποστολείς σημάτων χρησιμοποιούν μία από τις τρεις ειδικές εντολές αποστολής σημάτων του MFC για να στείλουν ένα σήμα:

- sndsig

- sndsigf

- sndsigb

Ένα SPE μπορεί να διαβάσει μόνο τα τοπικά του κανάλια ειδοποίησης μέσω σημάτων. Το PPE ή άλλοι επεξεργαστές μπορούν να διαβάσουν από ή να εγγράψουν στον αντίστοιχο MMIO καταχωρητή. Αυτό επιτρέπει στο SPE-στόχο να κάνει polling, blocking ή να χρησιμοποιήσει το μηχανισμό των διακοπών προκειμένου να απαντήσει στα λαμβανόμενα σήματα. Μία ανάγνωση από μέρους του SPE σε ένα από τα δύο του κανάλια ειδοποίησης μέσω σημάτων αυτομάτως καθαρίζει το κανάλι. Μία ανάγνωση MMIO όμως δεν το καθαρίζει. Μία ανάγνωση του SPE από το κανάλι σημάτων θα προκαλέσει αναστολή της λειτουργίας όταν δεν υπάρχει διαθέσιμο σήμα τη στιγμή της απόπειρας ανάγνωσης.

Ένα κανάλι ειδοποίησης μέσω σημάτων μπορεί να ρυθμιστεί από το λογισμικό ώστε να είναι είτε σε λειτουργία overwrite είτε σε λειτουργία OR. Στη λειτουργία overwrite (που ονομάζεται και ένα-προς-ένα σηματοδότηση – one-to-one signaling) η αποστολή ενός σήματος (δηλαδή η εγγραφή στην MMIO διεύθυνση) γράφει δεδομένα πάνω από τα παλιά περιεχόμενα, σβήνοντάς τα. Στη λειτουργία OR (που ονομάζεται και πολλά-προς-ένα σηματοδότηση – many-to-one signaling) η αποστολή ενός σήματος έχει ως αποτέλεσμα το λογικό OR μεταξύ των παλιών περιεχομένων και των αποστελλόμενων δεδομένων. Στην περίπτωση της ένα-προς-ένα σηματοδότησης δεν υπάρχει συνήθως ουσιαστικά διαφορά στην επίδοση μεταξύ της χρήσης σήματος και της χρήσης ενός mailbox.

Οι διαφορές μεταξύ mailboxes και καναλιών ειδοποίησης μέσω σημάτων περιλαμβάνουν:

- Χωρητικότητα – Τα κανάλια ειδοποίησης μέσω σημάτων είναι καταχωρητές. Τα mailboxes είναι ουρές.

- Κατεύθυνση – Κάθε SPE υποστηρίζει κανάλια ειδοποίησης μέσω σημάτων τα οποία είναι μόνο εισερχόμενα προς ένα SPE. Βέβαια, ένα SPE μπορεί να στείλει σήματα σε ένα άλλο SPE χρησιμοποιώντας εντολές αποστολής σημάτων του MFC.

- Διακοπές – Ένα από τα mailboxes προκαλεί διακοπή στο PPE. Τα κανάλια ειδοποίησης μέσω σημάτων δεν έχουν αυτόματα τέτοιο χαρακτηριστικό.

- Πολλά-προς-ένα – Τα κανάλια ειδοποίησης μέσω σημάτων (αλλά όχι τα mailboxes) μπορούν να ρυθμιστούν σε λειτουργία είτε πολλά-προς-ένα (OR mode) είτε ένα-προς-ένα (overwrite mode).

- Μοναδικές εντολές – Τα κανάλια ειδοποίησης μέσω σημάτων έχουν ειδικές εντολές αποστολής σημάτων του MFC (sndsig, sndsigf, sndsigb) ώστε να εγγράψει κάποιος σε αυτά.

- Καθαρισμός – Η ανάγνωση ενός καταχωρητή καναλιού ειδοποίησης μέσω σημάτων αυτομάτως μηδενίζει τα bits του.

- Μετρητής – Οι μετρητές των καναλιών έχουν διαφορετικές ερμηνείες. Οι μετρητές των καναλιών των mailboxes δείχνουν τον αριθμό των διαθέσιμων (μη κατειλημμένων) θέσεων στην ουρά του mailbox. Ο μετρητής του καναλιού σημάτων δείχνει εάν υπάρχουν κάποια σήματα σε αναμονή (που δεν έχουν εξυπηρετηθεί).

- Πλήθος – Κάθε SPE έχει δύο κανάλια ειδοποίησης μέσω σημάτων και τρία mailboxes.

Σύνολο Εντολών της SPU

Οι προγραμματιστές που γράφουν σε μία γλώσσα υψηλού επιπέδου όπως οι C και C++ μπορούν να χρησιμοποιήσουν τις ειδικές εντολές (intrinsics) που παρέχονται ώστε να βελτιώσουν το έλεγχο που έχουν πάνω στο hardware της SPE. Οι λειτουργίες που επιτελούνται από αυτές τις ειδικές εντολές είναι πολύ στενά σχετιζόμενες με τις εντολές assembly του συνόλου εντολών της αρχιτεκτονικής της SPU.

Το σύνολο εντολών της αρχιτεκτονικής της SPU επενεργούν κυρίως πάνω σε διανυσματικά (SIMD) ορίσματα, τόσο ακέραια όσο και κινητής υποδιαστολής, με υποστήριξη και για κάποια βαθμωτά (μη διανυσματικά) ορίσματα. Και το PPE και το SPE εκτελούν εντολές SIMD, όμως οι δύο επεξεργαστές διαθέτουν διαφορετικά σύνολα εντολών και τα προγράμματα για το PPE και τα SPEs πρέπει να μεταγλωττιστούν από διαφορετικούς compilers.

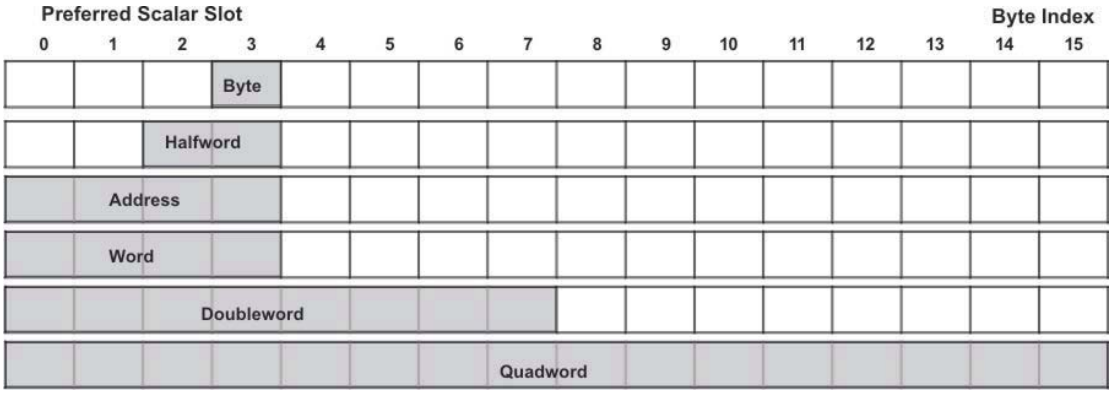
Διάταξη δεδομένων στους καταχωρητές

Το SPE υποστηρίζει τη σύμβαση big-endian για τη διάταξη των δεδομένων, μία διάταξη κατά την οποία το byte με τη χαμηλότερη διεύθυνση και το bit με τη χαμηλότερη αρίθμηση είναι το πιο σημαντικό (υψηλό) byte και bit, αντίστοιχα. Τα bits στους καταχωρητές αριθμούνται σε αύξουσα σειρά από αριστερά προς τα δεξιά, με το bit 0 να αναπαριστά το πιο σημαντικό bit (MSb) και το bit 127 το λιγότερο σημαντικό bit (LSb), όπως φαίνεται και στο ακόλουθο σχήμα. Η αρχιτεκτονική του SPE δεν ορίζει ούτε χρησιμοποιεί τη little-endian σύμβαση για τη διάταξη των δεδομένων.

Το hardware της SPU ορίζει τους ακόλουθους τύπους δεδομένων:

- Byte – 8 bits
- Μισή Λέξη (Halfword) – 16 bits
- Λέξη (Word) – 32 bits
- Διπλή Λέξη (Doubleword) – 64 bits
- Τετραπλή Λέξη (Quadword) – 128 bits

Αυτοί οι τύποι δεδομένων φαίνονται στο ακόλουθο σχήμα στις σκιασμένες περιοχές. Η πιο αριστερή λέξη (τα bytes 0, 1, 2 και 3) ενός καταχωρητή ονομάζεται προτιμώμενη θέση για βαθμωτά δεδομένα (preferred scalar slot) και δείχνεται επίσης στο σχήμα. Όταν κάποια εντολή χρησιμοποιήσει βαθμωτά δεδομένα ή διευθύνσεις, οι τιμές μπαίνουν στην προτιμώμενη θέση. Υπάρχει επίσης ένα σύνολο βοηθητικών αποθηκευτικών εντολών οι οποίες βοηθούν στην αποθήκευση bytes, μισών λέξεων, λέξεων και διπλών λέξεων.



Το μοντέλο προγραμματισμού του SPE ορίζει τους τύπους διανυσματικών δεδομένων για τη γλώσσα C που φαίνονται στον παρακάτω πίνακα. Όλοι αυτοί οι τύποι δεδομένων έχουν μήκος 128 bits και περιέχουν από 1 μέχρι 16 στοιχεία ανά διάνυσμα.

Τύπος Διανυσματικών Δεδομένων	**Περιεχόμενο**
vector unsigned char	Δεκαέξι 8bit απρόσημοι χαρακτήρες
vector signed char	Δεκαέξι 8bit προσημασμένοι χαρακτήρες
vector unsigned short	Οκτώ 16bit απρόσημες μισές λέξεις
vector signed short	Οκτώ 16bit προσημασμένες μισές λέξεις
vector unsigned int	Τέσσερεις 32bit απρόσημες λέξεις
vector signed int	Τέσσερεις 32bit προσημασμένες λέξεις

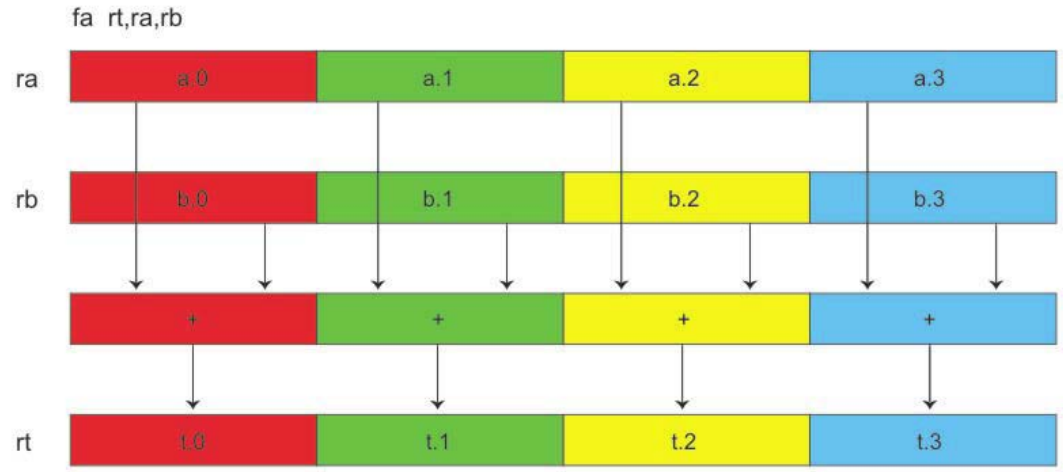
Τύπος Διανυσματικών Δεδομένων	**Περιεχόμενο**
vector unsigned long long	Δύο 64bit απρόσημες διπλές λέξεις
vector signed long long	Δύο 64bit προσημασμένες διπλές λέξεις
vector float	Τέσσερεις 32bit δεκαδικοί απλής ακρίβειας
vector double	Δύο 64bit δεκαδικοί διπλής ακρίβειας
qword	Μία τετραπλή λέξη 16 bytes (quadword)

Τύποι εντολών

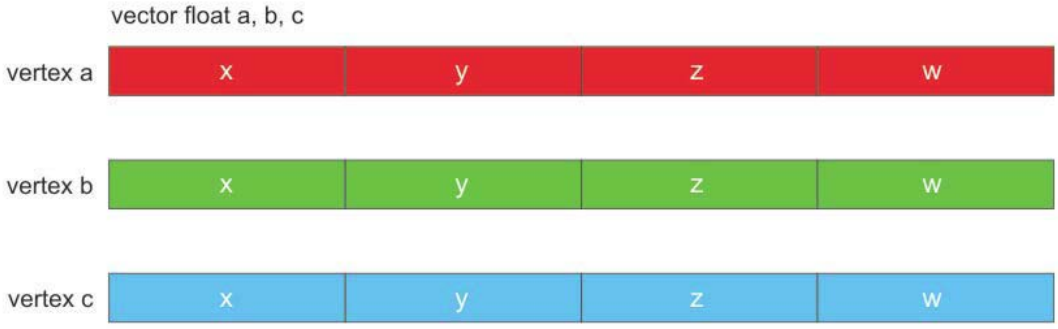
Υπάρχουν 204 εντολές στο σύνολο εντολών της αρχιτεκτονικής της SPU και ομαδοποιούνται σε 11 κατηγορίες ανάλογα με τη λειτουργικότητά τους. Οι κατηγορίες αυτές φαίνονται στον παρακάτω πίνακα.

Τύπος	**Πλήθος**
Φορτώσεις και αποθηκεύσεις από/προς τη μνήμη	16
Διαμόρφωση σταθερών	6
Πράξεις σε ακεραίους και λογικές πράξεις	59
Ολισθήσεις και περιστροφές	31
Συγκρίσεις, διακλαδώσεις και τερματισμός	40
Υποδείξεις για διακλάδωση (branch hint)	3
Κινητής υποδιαστολής	28
Έλεγχος	8
Κανάλια SPU	3
Λειτουργία διακοπών (interrupt facility)	7
Συγχρονισμός και διάταξη	3

Ακολουθεί ένα παράδειγμα μια εντολής SIMD της SPU – η εντολή πρόσθεσης κινητής υποδιαστολής fa. Αυτή η εντολή προσθέτει ταυτόχρονα τέσσερα ζεύγη διανυσματικών στοιχείων κινητής υποδιαστολής, που είναι αποθηκευμένα στους καταχωρητές ra και rb, και παράγει τέσσερα αποτελέσματα κινητής υποδιαστολής, που αποθηκεύονται στον καταχωρητή rt.

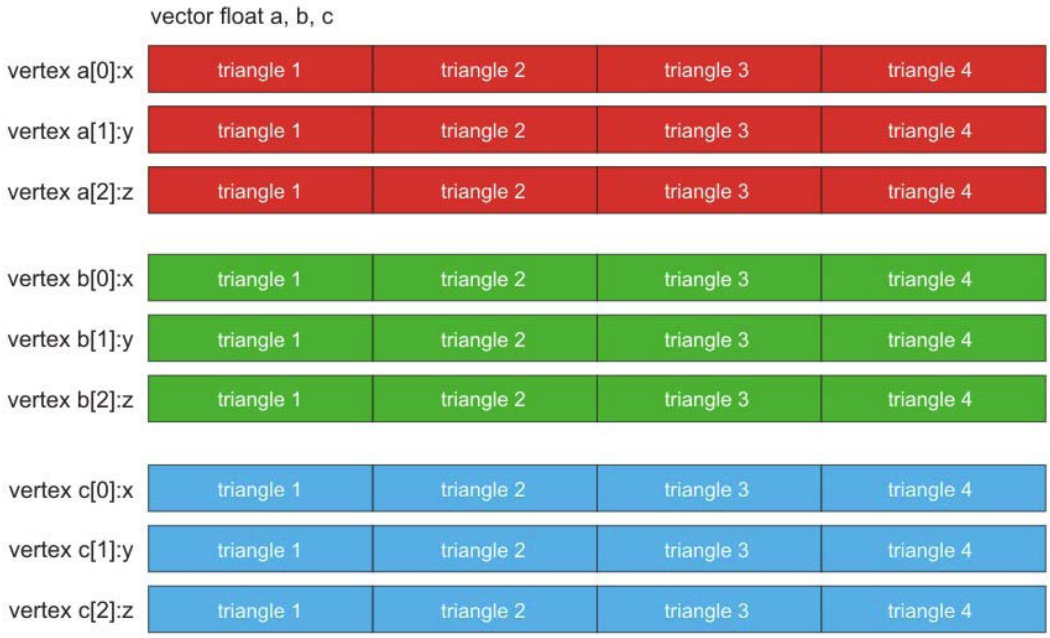


Αναλόγως των απαιτήσεων του προγραμματιστή για επίδοση και των περιορισμών για το μέγεθος του κώδικα, η κατάλληλη ομαδοποίηση των δεδομένων σε ένα διάνυσμα SIMD μπορεί να αποφέρει πλεονεκτήματα. Παρακάτω φαίνεται ένας φυσικός τρόπος χρήσης διανυσμάτων SIMD για την αποθήκευση τιμών ομοιογενών δεδομένων (x, y, z, w) για τις τρεις κορυφές (a, b, c) ενός τριγώνου σε μια εφαρμογή 3D γραφικών. Η συγκεκριμένη διάταξη των δεδομένων καλείται πίνακας από δομές (array of structures – AOS) επειδή οι τιμές των δεδομένων για κάθε κορυφή οργανώνονται σε μία μοναδική δομή και το σύνολο όλων αυτών των δομών (κορυφών) είναι ένας πίνακας.



Η παραπάνω προσέγγιση «πακεταρίσματος» δεδομένων παράγει συχνά μικρό μέγεθος κώδικα αλλά τυπικά εκτελείται με φτωχό τρόπο και γενικά χρειάζεται σημαντικό ξεδίπλωμα βρόχων (loop unrolling) ώστε να βελτιωθεί η επίδοση. Εάν οι κορυφές περιέχουν λιγότερες συνιστώσες από όσες μπορεί να κρατήσει το διάνυσμα SIMD (για παράδειγμα, τρεις συνιστώσες αντί για τέσσερεις), θα πρέπει να γίνει συμβιβασμός σχετικά με την αποδοτικότητα των SIMD.

Μία άλλη μέθοδος οργάνωσης δεδομένων σε διανύσματα SIMD είναι η δομή από πίνακες (structure of arrays – SOA).

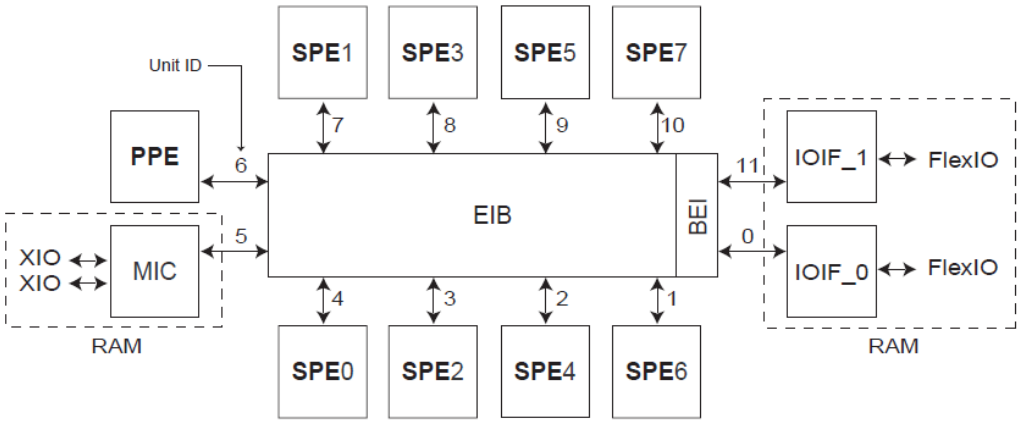


Σε αυτήν την περίπτωση, η κάθε αντίστοιχη τιμή δεδομένου για κάθε κορυφή αποθηκεύεται σε μια αντίστοιχη θέση σε ένα σύνολο από διανύσματα. Αρχικά, τα δεδομένα είναι βαθμωτά και τα διανύσματα κατασκευάζονται με ανεξάρτητα δεδομένα κατά μήκος του διανύσματος. Αυτή η προσέγγιση είναι διαφορετική από την προηγούμενη περίπτωση, όπου οι τέσσερεις τιμές της κάθε κορυφής αποθηκεύονται σε ένα διάνυσμα. Το επόμενο σχήμα δείχνει τη χρήση των διανυσμάτων SIMD για την αναπαράσταση των κορυφών x, y, z τεσσάρων τριγώνων. Εδώ δεν έχουμε μόνο τους τύπους δεδομένων που είναι οι ίδιοι κατά μήκος του διανύσματος, αλλά τώρα είναι και η σημασία των δεδομένων η ίδια. Αναλόγως του αλγορίθμου, το λογισμικό μπορεί να εκτελεστεί πιο αποδοτικά με αυτήν την SIMD οργάνωση δεδομένων από ότι θα γινόταν με την προηγούμενη μέθοδο.

Δίαυλος Διασύνδεσης Στοιχείων (Element Interconnect Bus)

Ο Δίαυλος Διασύνδεσης Στοιχείων (Element Interconnect Bus – EIB) είναι ένα μονοπάτι επικοινωνίας για εντολές και δεδομένα μεταξύ όλων των επεξεργαστών και όλων των ελεγκτών για τη μνήμη και την είσοδο/έξοδο που βρίσκονται μέσα στην Cell Broadband Engine. Όπως φαίνεται στο παρακάτω διάγραμμα, συνδέει μεταξύ τους το PPE, τα SPEs, τον ελεγκτή μνήμης (Memory Interface Controller – MIC) και τη μονάδα διεπαφής της Cell Broadband Engine (Broadband Engine Interface – BEI). Η συχνότητα λειτουργίας του διαύλου είναι η μισή της συχνότητας λειτουργίας των επεξεργαστικών στοιχείων. Ο EIB

υποστηρίζει λειτουργίες συμμετρικής πολυεπεξεργασίας (Symmetric Multiprocessing - SMP) και πλήρως συναφών με τη μνήμη. Έτσι, ένας επεξεργαστής Cell Broadband Engine έχει σχεδιαστεί ώστε να ομαδοποιείται συναφώς με άλλες Cell Broadband Engines ώστε να κατασκευαστεί μία συστοιχία από CBEs.



BEI	Cell Broadband Engine Interface	MIC	Memory Interface Controller
EIB	Element Interconnect Bus	PPE	PowerPC Processor Element
FlexIO	Rambus FlexIO Bus	RAM	Resource Allocation Management
IOIF	I/O Interface	SPE	Synergistic Processor Element
		XIO	Rambus XDR I/O (XIO) cell

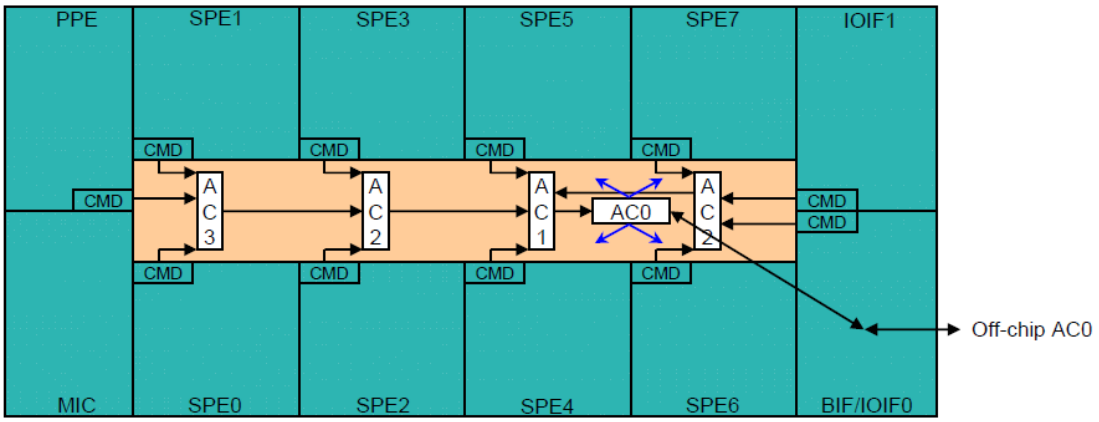
Ο EIB αποτελείται από τέσσερεις δακτυλίους εύρους 16 bytes. Κάθε δακτύλιος μεταφέρει 128 bytes (δηλαδή μία cache line του PPE) κάθε φορά. Κάθε επεξεργαστικό στοιχείο έχει ένα διάδρομο εισερχομένων από (on-ramp) και ένα διάδρομο εξερχομένων (off-ramp) προς τον EIB. Τα επεξεργαστικά στοιχεία μπορούν να στέλνουν και να δέχονται δεδομένα ταυτοχρόνως. Τα παραπάνω διάγραμμα δείχνει τους αριθμούς-ταυτότητες των μονάδων (unit ID numbers) για κάθε στοιχείο και τη σειρά με την οποία τα διάφορα στοιχεία είναι συνδεδεμένα στον EIB. Η σειρά διασύνδεσης είναι σημαντική για τους προγραμματιστές που προσπαθούν να μειώσουν την καθυστέρηση των μεταφορών στον EIB. Η καθυστέρηση είναι συνάρτηση του πλήθους των βημάτων (hops) διασύνδεσης, πράγμα που σημαίνει ότι μεταφορές μεταξύ γειτονικών στοιχείων έχουν τις μικρότερες καθυστερήσεις και μεταφορές μεταξύ στοιχείων που απέχουν κατά έξι hops έχουν τις μεγαλύτερες καθυστερήσεις.

Το μέγιστο εσωτερικό εύρος ζώνης του EIB είναι 96 bytes ανά κύκλο ρολογιού του επεξεργαστή. Σε κάθε δακτύλιο μπορούν να υπάρχουν ταυτόχρονα σε εξέλιξη πολλαπλές μεταφορές, στις οποίες περιλαμβάνονται πάνω από 100 εκκρεμείς αιτήσεις DMA μεταξύ του κύριου αποθηκευτικού χώρου και των SPEs. Ο EIB δεν υποστηρίζει κάποια ιδιαίτερη συμπεριφορά ποιότητας υπηρεσίας (Quality of Service – QoS) πέραν του να εξασφαλίζει την εξέλιξη των μεταφορών. Παρ' όλα αυτά, έχει υλοποιηθεί μέσα στον EIB μία λειτουργία διαχείρισης δέσμευσης πόρων (Resource Allocation Management Facility – RAM Facility).

Το προνομιούχο λογισμικό μπορεί να τη χρησιμοποιήσει ώστε να ρυθμίσει το ρυθμό με τον οποίον τα στοιχεία που ζητάνε πόρους (το PPE, τα SPEs και οι συσκευές εισόδου/εξόδου) μπορούν να χρησιμοποιούν πόρους της μνήμης και του I/O.

Αρχιτεκτονική

Υπάρχουν ξεχωριστά δίκτυα για την εξυπηρέτηση αιτήσεων μεταφοράς μεταξύ των συνδεδεμένων στοιχείων για δεδομένα και εντολές. Κάθε συνδεδεμένο στοιχείο έχει μία θύρα εισόδου και μία θύρα εξόδου, εκτός από την BEI, η οποία διαθέτει από μία θύρα εισόδου και μία θύρα εξόδου για κάθε διεπαφή της (IOIF_0 και IOIF_1). Κάθε στοιχείο μπορεί να στέλνει και να δέχεται ταυτόχρονα 16 bytes δεδομένων σε κάθε κύκλο.

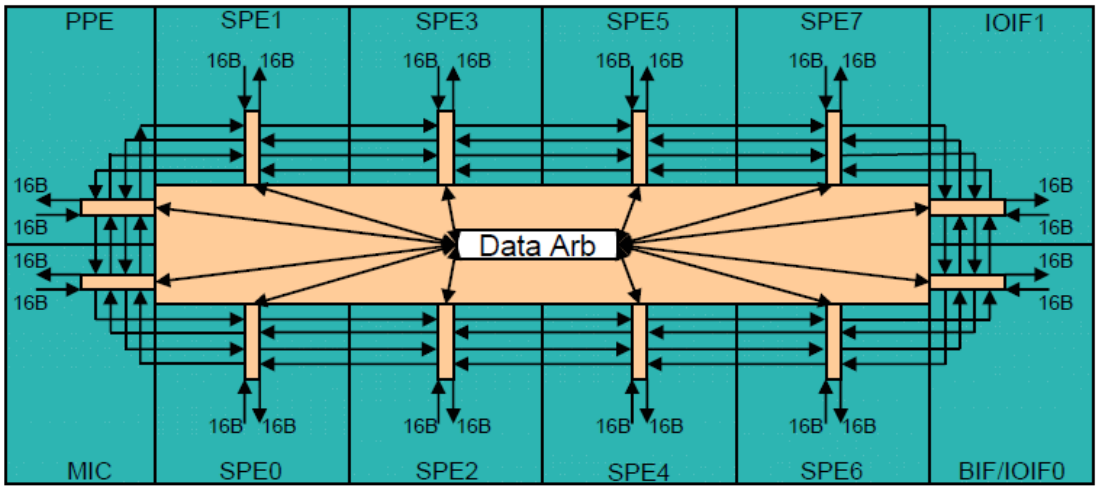


Όσον αφορά στη μεταφορά εντολών κάθε στοιχείο συνδέεται σε έναν συγκεντρωτή διευθύνσεων (address concentrator – AC) ο οποίος φροντίζει για την ανίχνευση και την πρόληψη των συγκρούσεων και εξασφαλίζει τη δίκαιη κατανομή της πρόσβασης στο δίαυλο. Όπως φαίνεται και από το παραπάνω διάγραμμα, υπάρχουν πολλοί συγκεντρωτές διευθύνσεων και όλοι προωθούν τα δεδομένα σε ένα μοναδικό σημείο σειριακής ανάκλασης εντολών (serial command reflection point), το συσσωρευτή AC0. Οι υπόλοιποι συσσωρευτές είναι οι AC1, AC2 (ο EIB διαθέτει δύο τέτοιους συσσωρευτές) και AC3. Ο AC0 λαμβάνει και διατάσσει τις εντολές από τα διάφορα στοιχεία, στέλνει (broadcast) σε όλα τα στοιχεία τις εντολές (για snooping) και συγκεντρώνει τις απαντήσεις και τις προωθεί πάλι στα διάφορα στοιχεία. Ως απάντηση στην εντολή θεωρείται το σήμα που αποστέλλεται στα εκάστοτε στοιχεία για να ξεκινήσει μία μεταφορά.

Το δίκτυο για τη μεταφορά δεδομένων είναι το πιο περίπλοκο: αποτελείται από τέσσερεις δακτυλίους, κάθε ένας από τους οποίους συνδέει μεταξύ τους όλα τα στοιχεία. Τα στοιχεία συνδέονται στον κάθε δακτύλιο με την ίδια σειρά. Οι μεταφορές σε κάθε δακτύλιο μπορούν να γίνονται μόνο προς μία κατεύθυνση. Για παράδειγμα, εάν σε ένα δακτύλιο μεταφέρονται δεδομένα από το PPE στο SPE1, τότε δεν επιτρέπεται η μεταφορά δεδομένων προς την αντίθετη κατεύθυνση. Οι δύο από τους δακτυλίους μεταφέρουν δεδομένα κατά τη

φορά των δεικτών του ρολογιού και οι άλλοι δύο κατά την αντίθετη φορά. Κάθε δακτύλιος επιτρέπει να εκτελούνται ταυτόχρονα μέχρι και τρεις μεταφορές, αρκεί οι διαδρομές τους να μην επικαλύπτονται. Για να ξεκινήσει μία μεταφορά, το στοιχείο που δίνει τη σχετική εντολή πρέπει να αιτήσει την πρόσβαση στο δίαυλο από τον «διαιτητή» (arbiter) του διαύλου, ο οποίος επεξεργάζεται τις αιτήσεις και αποφασίζει σε ποιον δακτύλιο θα ανατεθεί η διεκπεραίωση κάθε αίτησης. Πάντα επιλέγεται ένας δακτύλιος που μεταφέρει δεδομένα προς την κατεύθυνση εκείνη κατά την οποία η απόσταση μεταξύ πηγής και προορισμού είναι η ελάχιστη. Έτσι, καμία μεταφορά δεν διανύει ποτέ απόσταση μεγαλύτερη από το μισό του δακτυλίου, δηλαδή περισσότερα από έξι βήματα (το μήκος όλου του δακτυλίου είναι δώδεκα βήματα, αφού δώδεκα είναι τα συνδεδεμένα σε αυτόν στοιχεία). Ο arbiter φροντίζει επίσης ώστε να μην επικαλύπτονται μεταξύ τους οι διάφορες μεταφορές πάνω στον ίδιο δακτύλιο.

Ακολουθεί διάγραμμα στο οποίο φαίνονται τα διάφορα μονοπάτια δεδομένων μέσα στον EIB.



Εύρος ζώνης

Κάθε στοιχείο που συνδέεται στον EIB μπορεί να δέχεται και να λαμβάνει ταυτόχρονα 16 bytes δεδομένων σε κάθε κύκλο διαδρόμου. Το μέγιστο εύρος ζώνης δεδομένων περιορίζεται από τον ρυθμό με τον οποίον τα στοιχεία του διαύλου μπορούν να κάνουν επισκόπηση (snooping) των διευθύνσεων των αιτήσεων μεταφορών. Ο ρυθμός αυτός περιορίζεται στη μία διεύθυνση ανά κύκλο του διαδρόμου. Κάθε αίτηση μπορεί να αφορά σε μεταφορά έως και 128 bytes δεδομένων. Άρα, το μέγιστο εύρος ζώνης είναι 128 bytes x (συχνότητα ρολογιού / 2).

Το μέγιστο εύρος ζώνης δεν επιτυγχάνεται πάντα στην πράξη, καθώς εξαρτάται από τους ακόλουθους παράγοντες:

- Τις σχετικές θέσεις πηγής και προορισμού

- Την πιθανότητα μία νέα μεταφορά να συγκρούεται με άλλες που βρίσκονται σε εξέλιξη

- Το πλήθος των Cell Broadband Engines που βρίσκονται στο σύστημα

- Το αν οι μεταφορές πραγματοποιούνται μόνο μεταξύ SPEs ή μεταξύ SPEs και κύριας μνήμης

- Την αποδοτικότητα του bus arbiter

Άρα, μειωμένο εύρος ζώνης έχουμε στις ακόλουθες περιπτώσεις:

- Όλες οι αιτήσεις έχουν τον ίδιο προορισμό.

- Όλες οι μεταφορές εκτελούνται κατά την ίδια φορά, οπότε οι δύο από τους τέσσερεις δακτυλίους δεν χρησιμοποιούνται

- Οι περισσότερες μεταφορές έχουν μέγεθος μικρότερο από 128 bytes

- Όλες οι μεταφορές διανύουν απόσταση ίση με το μισό του δακτυλίου, οπότε εμποδίζεται κάθε άλλη μεταφορά σε αυτό το κομμάτι του δακτυλίου

Σχετικά με τις θέσεις της πηγής και του προορισμού μίας μεταφοράς μπορούν να γίνουν οι ακόλουθες παρατηρήσεις:

- Το PPE και ο MIC βρίσκονται το ένα δίπλα στο άλλο, πράγμα που σημαίνει ότι οι μεταφορές μεταξύ τους δεν επηρεάζουν σημαντικά τις υπόλοιπες μεταφορές.

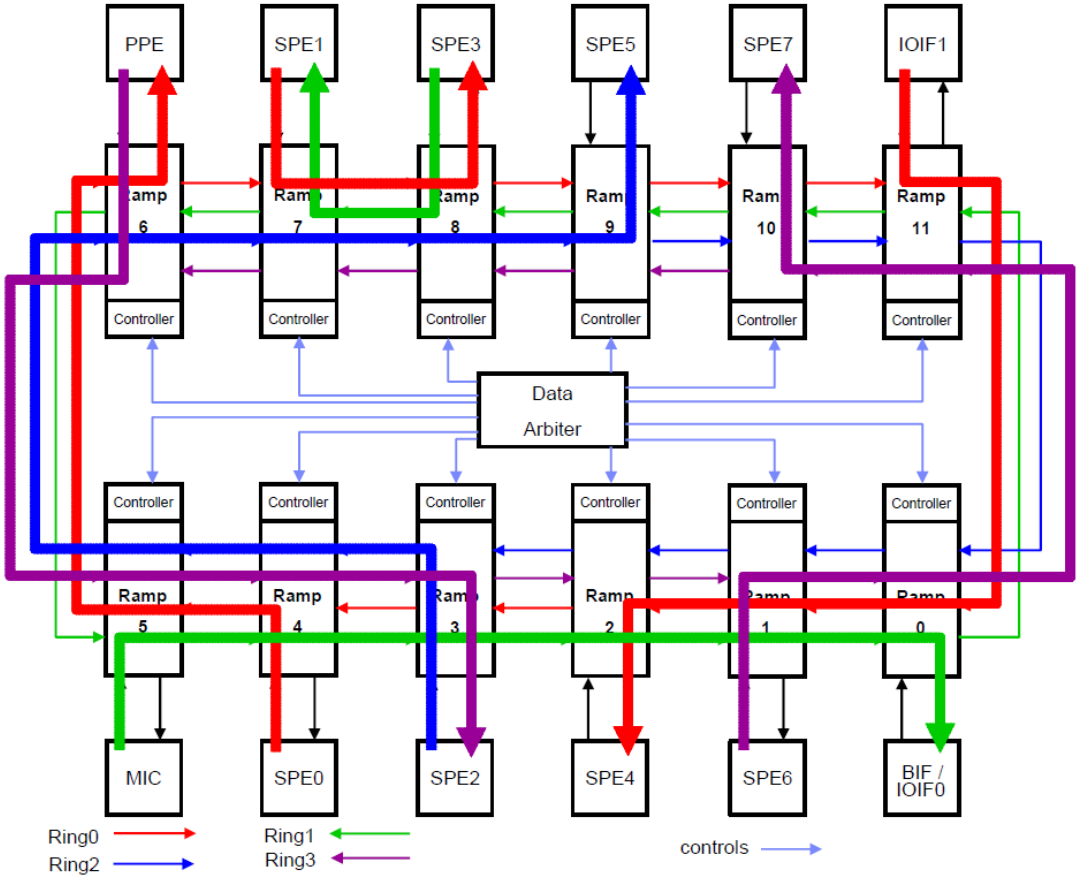
- Τα SPEs στο «βόρειο» μέρος του ολοκληρωμένου παίρνουν ως unit IDs άρτιους αριθμούς (0, 2, 4, 6), ενώ τα SPEs στο «νότιο» μέρος του ολοκληρωμένου παίρνουν περιττούς αριθμούς (1, 3, 5, 7). Έτσι, μία μεταφορά μεταξύ ενός SPE με άρτιο ID και ενός SPE με περιττό ID ενδέχεται να περνάει από το PPE ή και τον MIC.

- Είναι γενικά πιο αποδοτικό να επικοινωνούν τα SPEs με άρτιο ID μεταξύ τους και τα SPEs με περιττό ID μεταξύ τους.

Για τους παραπάνω λόγους είναι χρήσιμο κατά την ανάπτυξη μίας εφαρμογής να κατασκευάζεται ένα διάγραμμα μεταφορών ώστε να φαίνονται οι διάφορες μεταφορές και οι

μεταξύ τους αλληλεπιδράσεις, έτσι ώστε να αποφεύγονται σχεδιασμοί που θα μειώσουν κατά πολύ το μέγιστο δυνατό εύρος ζώνης.

Ακολουθεί ένα παράδειγμα οκτώ ταυτόχρονων μεταφορών μέσω του EIB. Για κάθε μεταφορά φαίνεται και ο δακτύλιος μέσω του οποίου αυτή επιτελείται.



Ελεγκτής Διεπαφής με τη Μνήμη (Memory Interface Controller)

Ο ενσωματωμένος στο ολοκληρωμένο Ελεγκτής Διεπαφής με τη Μνήμη (Memory Interface Controller – MIC) παρέχει τη διεπαφή μεταξύ του EIB και της φυσικής μνήμης. Υποστηρίζει μία ή δύο διεπαφές μνήμης Rambus Extreme Data Rate (XDR) που μαζί μπορούν να υποστηρίξουν από 64MB έως 64GB XDR DRAM μνήμη.

Οι προσβάσεις στη μνήμη σε κάθε διεπαφή είναι από 1 έως 8, 16, 32, 64 ή 128 bytes, με σειρά συναφή ως προς τη μνήμη. Μπορούν να μπουν στην ουρά μέχρι 64 αναγνώσεις και 64 εγγραφές. Ο διαχειριστής πόρων (resource-allocation token manager) παρέχει υποστήριξη σχετικά με τα επίπεδα της ουράς.

Ο MIC έχει διάφορους τρόπους λειτουργίας, ελεγχόμενους από το λογισμικό, στους οποίος περιλαμβάνονται τα fast-path mode (για βελτιωμένη καθυστέρηση όταν οι ουρές

εντολών είναι άδειες), high-priority read (για να δίνεται προτεραιότητα στις αναγνώσεις ενός SPE έναντι όλων των άλλων αναγνώσεων), early read (για να ξεκινάει μία ανάγνωση προτού ολοκληρωθεί μία προηγούμενη εγγραφή), speculative read και slow mode (σχετικά με τη διαχείριση κατανάλωσης ενέργειας). Ο MIC υλοποιεί έναν ελεγκτή κλειστών σελίδων (closed-page controller) (δηλαδή οι γραμμές της μνήμης κλείνουν μετά από κάθε ανάγνωση, εγγραφή ή ανανέωση), έναν μηχανισμό για αρχικοποίηση της μνήμης και για memory scrubbing.

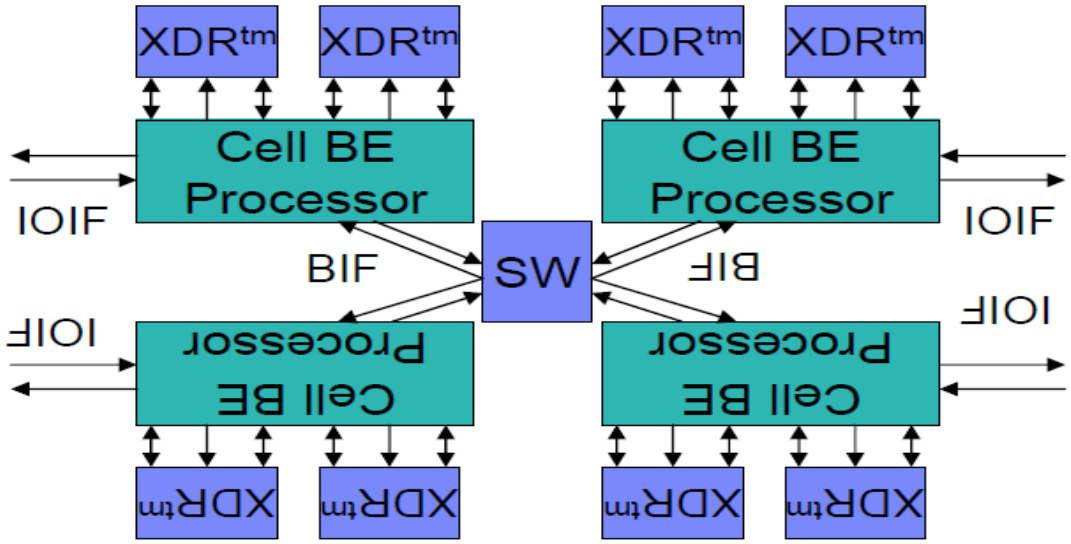
Η μνήμη XRD DRAM έχει μηχανισμό ελέγχου και διόρθωσης λαθών (Error Checking and Correction – ECC), με ανίχνευση λαθών σε ένα ή περισσότερα bits.

Μονάδα Διεπαφής της Broadband Engine (Broadband Engine Interface)

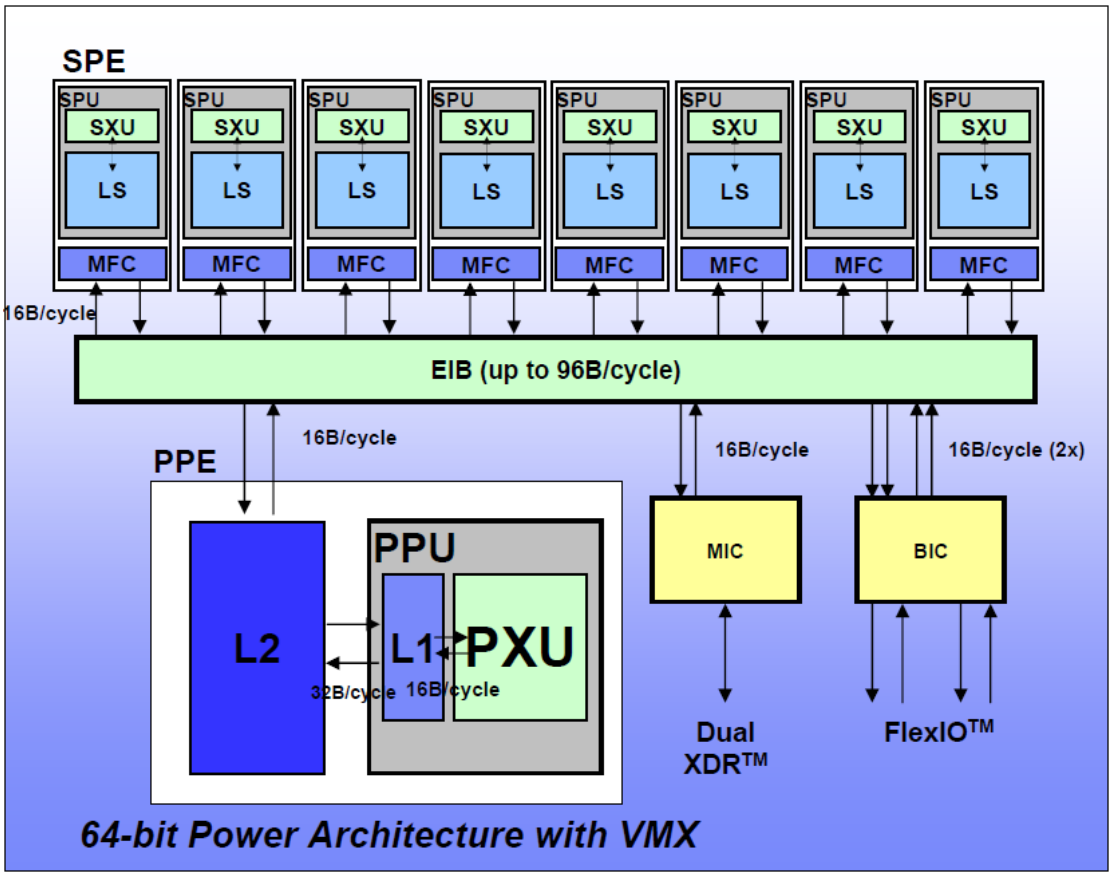
Η ενσωματωμένη στο ολοκληρωμένο μονάδα Διεπαφής της Cell Broadband Engine (Broadband Engine Interface) υποστηρίζει τη διεπαφή εισόδου/εξόδου. Περιλαμβάνει έναν ελεγκτή διεπαφής διαύλου (Bus Interface Controller – BIC), έναν ελεγκτή εισόδου/εξόδου (I/O Controller – IOC) και έναν εσωτερικό ελεγκτή διακοπών (Internal Interrupt Controller – IIC). Διαχειρίζεται τις μεταφορές δεδομένων μεταξύ του EIB και των συσκευών εισόδου/εξόδου και παρέχει μετάφραση διευθύνσεων (I/O address translation) και επεξεργασία εντολών (command processing).

Η BEI υποστηρίζει δύο διεπαφές Rambus FlexIO. Η μία από τις δύο διεπαφές (IOIF1) υποστηρίζει μόνο ένα πρωτόκολλο μη συναφούς διεπαφής εισόδου/εξόδου (I/O Interface – IOIF), το οποίο είναι κατάλληλο για συσκευές εισόδου/εξόδου. Η άλλη διεπαφή (IOIF0, που επίσης ονομάζεται και BIF/IOIF0) μπορεί να λειτουργήσει (καθοριζόμενο από το λογισμικό) είτε σύμφωνα με το μη συναφές πρωτόκολλο IOIF είτε με το συναφές ως προς τη μνήμη πρωτόκολλο της διεπαφής της Cell Broadband Engine (Broadband engine Interface – BIF). Το πρωτόκολλο BIF είναι το εσωτερικό πρωτόκολλο του EIB. Μπορεί να χρησιμοποιηθεί για τη συναφή επέκταση του EIB μέσω του IOIF0 προς μία άλλη συναφή με τη μνήμη συσκευή, όπως μία άλλη Cell Broadband Engine.

Παρακάτω δίνεται ένα παράδειγμα κατασκευής μίας συστοιχίας από Cell Broadband Engines με τη χρήση switch.



Ακολουθεί ένα αναλυτικό block διάγραμμα της αρχιτεκτονικής της Cell Broadband Engine, όπου φαίνονται οι περισσότερες από τις λεπτομέρειες που περιγράφηκαν σε αυτό το κεφάλαιο:



Κεφάλαιο 3

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Το σύνολο εντολών του PPE είναι μία επεκτεταμένη έκδοση του συνόλου εντολών PowerPC. Οι επεκτάσεις αποτελούνται από το σύνολο εντολών Vector/SIMD Multimedia Extension συν μερικές προσθήκες και αλλαγές στις εντολές PowerPC. Το σύνολο εντολών του SPE είναι παρόμοιο με το σύνολο εντολών Vector/SIMD Multimedia Extension του PPE. Παρόλο που τόσο το PPE όσο και το SPE εκτελούν εντολές SIMD, τα δύο σύνολα εντολών είναι διαφορετικά και τα προγράμματα του SPE πρέπει να μεταγλωττίζονται με διαφορετικούς compilers από ότι τα προγράμματα του PPE.

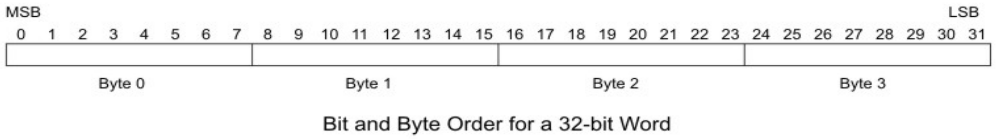
Διάταξη των bytes και αρίθμηση των bits

Η αποθήκευση δεδομένων και εντολών στην Cell Broadband Engine ακολουθεί τη σύμβαση big-endian. Η σύμβαση αυτή έχει τα ακόλουθα χαρακτηριστικά:

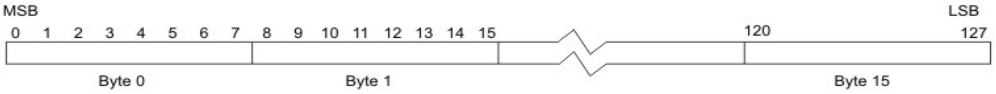
- Το περισσότερο σημαντικό byte αποθηκεύεται στη χαμηλότερη διεύθυνση και το λιγότερο σημαντικό byte αποθηκεύεται στη μεγαλύτερη διεύθυνση.

- Η αρίθμηση των bits μέσα σε ένα byte πηγαίνει από το περισσότερο σημαντικό bit (bit 0) στο λιγότερο σημαντικό bit. Αυτή η αρίθμηση διαφέρει από μερικούς άλλους big-endian επεξεργαστές.

Παρακάτω φαίνεται μία σύνοψη της διάταξης των bytes και της αρίθμησης των bits στη μνήμη, όπως επίσης και οι συμβάσεις της αρίθμησης των bits.



Bit and Byte Order for a 32-bit Word



Bit and Byte Order for a 128-bit Register

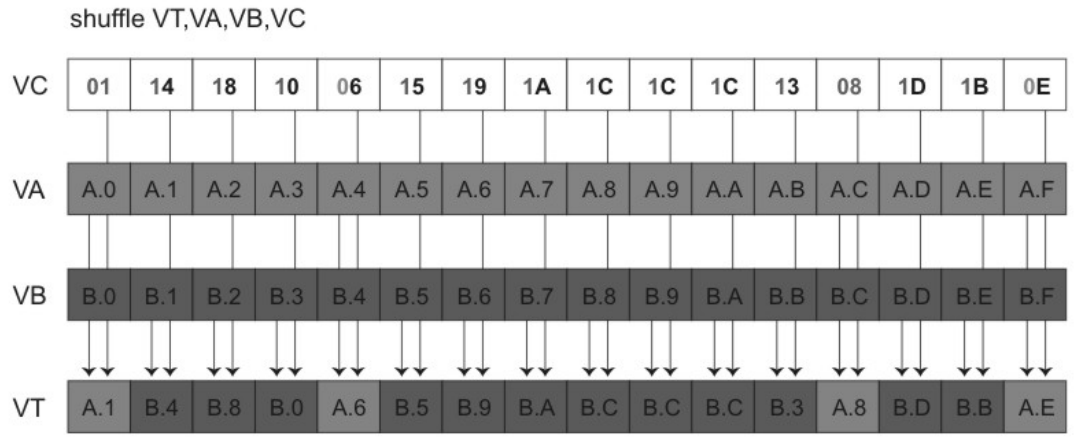
Διανυσματοποίηση SIMD (SIMD Vectorization)

Το διάνυσμα (vector) είναι ένα όρισμα εντολής που περιέχει ένα σύνολο στοιχείων δεδομένων πακεταρισμένων σε έναν μονοδιάστατο πίνακα. Τα στοιχεία αυτά μπορούν να είναι ακέραιοι ή αριθμοί κινητής υποδιαστολής. Οι περισσότερες εντολές των συνόλων Vector/SIMD Multimedia Extension και SPU ενεργούν πάνω σε διανυσματικά ορίσματα. Τα διανύσματα ονομάζονται επίσης και ορίσματα SIMD (SIMD operands) ή πακεταρισμένα ορίσματα (packed operands).

Η επεξεργασία SIMD εκμεταλλεύεται τον παραλληλισμό σε επίπεδο δεδομένων (data-level parallelism). Ο παραλληλισμός σε επίπεδο εντολών σημαίνει ότι οι πράξεις που απαιτούνται ώστε να μετασχηματιστεί ένα σύνολο στοιχείων ενός διανύσματος μπορούν να γίνουν πάνω σε όλα τα στοιχεία του διανύσματος την ίδια χρονική στιγμή. Αυτό σημαίνει ότι μία και μόνο εντολή μπορεί να εφαρμοστεί πάνω σε πολλαπλά στοιχεία δεδομένων παράλληλα. Η υποστήριξη λειτουργιών SIMD είναι κυρίαρχη στην Cell Broadband Engine. Στο PPE οι λειτουργίες SIMD υποστηρίζονται από το σύνολο εντολών Vector/SIMD Multimedia Extension και στα SPEs από το σύνολο εντολών SPU.

Τόσο στο PPE όσο και στα SPEs διανυσματικοί καταχωρητές (vector registers) κρατούν πολλαπλά στοιχεία δεδομένων ως ένα μοναδικό διάνυσμα. Τα datapaths και οι καταχωρητές που υποστηρίζουν τις λειτουργίες SIMD έχουν εύρος 128 bits, που αντιστοιχεί σε τέσσερεις πλήρεις λέξεις των 32 bits. Αυτό σημαίνει ότι τέσσερεις 32bit λέξεις μπορούν να φορτωθούν σε έναν και μόνο καταχωρητή και, για παράδειγμα, να προστεθούν σε άλλες τέσσερεις λέξεις κάποιου διαφορετικού καταχωρητή με μία και μόνο πράξη. Παρόμοιες λειτουργίες μπορούν αν εφαρμοστούν και σε διανυσματικά ορίσματα που περιέχουν 16 bytes, 8 μισές λέξεις (halfwords) ή 2 διπλές λέξεις (doublewords).

Η διαδικασία προετοιμασίας ενός προγράμματος για εκτέλεση σε ένα διανυσματικό επεξεργαστή ονομάζεται διανυσματοποίηση (vectorization ή SIMDization). Μπορεί να γίνει χειροκίνητα από τον προγραμματιστή ή αυτόματα από κάποιον compiler που κάνει αυτό-διανυσματοποίηση (auto-vectorization). Παρακάτω δίνεται ένα παράδειγμα μίας εντολής SIMD – σε αυτή την περίπτωση πρόκειται για μια εντολή ανακατέματος bytes (byte shuffle). Εδώ, τα bytes για το ανακάτεμα από τους καταχωρητές-πηγές VA και VB επιλέγονται βάσει των bytes-στοιχείων του διανύσματος ελέγχου VC, στο οποίο ένα byte 0 σημαίνει ότι για την αντίστοιχη θέση θα επιλεγεί το καθοριζόμενο byte από τον VA και ένα byte 1 σημαίνει ότι θα επιλεγεί το καθοριζόμενο byte από τον VB. Το αποτέλεσμα του ανακατέματος αποθηκεύεται στον καταχωρητή VT.



Ειδικές εντολές για τη γλώσσα C (C-language Intrinsics)

Τόσο το σύνολο εντολών Vector/SIMD Multimedia Extension όσο και το σύνολο εντολών SPU έχουν επεκτάσεις που υποστηρίζουν τις ειδικές εντολές (intrinsics) της γλώσσας C. Τα intrinsics είναι ειδικές εντολές της γλώσσας C, στη μορφή κλήσεων συναρτήσεων, οι οποίες είναι βολικά υποκατάστατα για μία ή περισσότερες inline εντολές assembly.

Σε ένα συγκεκριμένο σύνολο εντολών, τα περισσότερα ονόματα των intrinsics χρησιμοποιούν ένα σταθερό πρόθεμα στο μνημονικό τους όνομα συνοδευόμενο σε πολλές περιπτώσεις ονομάτων intrinsics από το μνημονικό μίας σχετιζόμενης εντολής assembly. Για παράδειγμα, το intrinsic που υλοποιεί την εντολή add της assembly του συνόλου εντολών Vector/SIMD Multimedia Extension ονομάζεται vec_add (vec είναι το πρόθεμα και add είναι το όνομα της αντίστοιχης assembly εντολής), ενώ το intrinsic της SPU που υλοποιεί την εντολή stop της assembly του συνόλου εντολών SPU ονομάζεται spu_stop.

Τα σύνολα εντολών Vector/SIMD Multimedia Extension (του PPE) και SPU (του SPE) έχουν και τα δύο επεκτάσεις που ορίζουν κάπως διαφορετικά σύνολα από intrinsics, όμως όλα εμπίπτουν σε μία από τις τέσσερεις κατηγορίες intrinsics που δίνονται στον παρακάτω πίνακα. Παρόλο που τα παρεχόμενα intrinsics για τα δύο σύνολα εντολών είναι παρόμοια στη λειτουργία τους, οι συμβάσεις ονομασίας και οι τρόποι κλήσης τους σε ένα πρόγραμμα διαφέρουν.

Τύπος intrinsic	**Ορισμός**	**PPE**	**SPE**
Ειδικό (specific)	Αντιστοιχία ένα-προς-ένα σε μία μοναδική εντολή assembly	X	X
Γενικό (generic)	Αντιστοιχία σε μία ή περισσότερες εντολές assembly, ανάλογα με τον τύπο των παραμέτρων εισόδου	X	X
Σύνθετο (composite)	Κατασκευάζεται από μία σειρά ειδικών ή γενικών		X

	intrinsics		
Κατηγορήματα (Predicates)	Αποτίμηση συνθηκών SIMD	X	

Νήματα και διεργασίες (Threads and tasks)

Σε ένα σύστημα που τρέχει το λειτουργικό σύστημα Linux το κύριο νήμα του προγράμματος είναι ένα νήμα Linux (Linux thread) το οποίο εκτελείται στο PPE. Το νήμα του κυρίως προγράμματος μπορεί με δημιουργήσει μία ή περισσότερες Linux διεργασίες (Linux tasks) της Cell Broadband Engine. Μία Linux διεργασία της Cell Broadband Engine έχει ένα ή περισσότερα Linux threads συσχετιζόμενα με αυτή τα οποία μπορούν να εκτελούνται είτε σε ένα PPE είτε σε ένα SPE. Ένα νήμα SPE (SPE thread) είναι ένα νήμα Linux που τρέχει σε ένα SPE. Η παρακάτω πίνακας συνοψίζει αυτούς τους όρους:

Όρος	**Ορισμός**
Νήμα Linux (Linux thread)	Ένα νήμα τα οποίο τρέχει στον περιβάλλον του λειτουργικού συστήματος Linux
Νήμα PPE (PPE thread)	Ένα νήμα Linux που τρέχει σε ένα PPE
Νήμα SPE (SPE thread)	Ένα νήμα Linux που τρέχει σε ένα SPE. Κάθε τέτοιο νήμα: • έχει το δικό του περιβάλλον SPE (SPE context) το οποίο περιλαμβάνει ένα αρχείο 128 καταχωρητών x 128 bits, έναν μετρητή προγράμματος και ουρές εντολών του MFC (MFC Command Queues) • μπορεί να επικοινωνήσει με άλλες μονάδες εκτέλεσης (ή με τη μνήμη ενεργών διευθύνσεων μέσω της διεπαφής καναλιού του MFC)

Όρος	Ορισμός
Νήμα Linux της Cell Broadband Engine (Cell Broadband Engine Linux task)	Μία διεργασία η οποία τρέχει στο PPE και στο SPE. • Κάθε τέτοια διεργασία έχει ένα ή περισσότερα νήματα Linux. • Όλα τα νήματα Linux εντός της διεργασίας μοιράζονται τους πόρους της διεργασίας.

Ένα νήμα Linux μπορεί να αλληλεπιδράσει άμεσα με ένα νήμα SPE μέσω της τοπικής μνήμης ή της problem state (σύνολο καταχωρητών ειδικής χρήσεως) του SPE. Μπορεί να αλληλεπιδράσει και εμμέσως μέσω της μνήμης του ενεργού χώρου διευθύνσεων ή της διεπαφής που παρέχεται από τις υπορουτίνες της βιβλιοθήκης SPE Runtime Management Library.

Το λειτουργικό σύστημα ορίζει το μηχανισμό και την πολιτική για τη διαχείριση των διαθέσιμων SPEs. Πρέπει να καθορίζει τις προτεραιότητες μεταξύ όλων των εφαρμογών Linux της Cell Broadband Engine στο σύστημα και να διαχειρίζεται την εκτέλεση στα SPEs ανεξάρτητα από τα συνηθισμένα νήματα Linux. Είναι επίσης υπεύθυνο κατά το χρόνο εκτέλεσης για τη φόρτωση, το πέρασμα παραμέτρων, την ειδοποίηση για γεγονότα SPE (SPE events) και λάθη, καθώς και για την υποστήριξη debugger.

Το περιβάλλον χρόνου εκτέλεσης (Runtime environment)

Το PPE τρέχει εφαρμογές και λειτουργικά συστήματα PowerPC, τα οποία μπορούν και να περιλαμβάνουν (όχι απαραίτητα) τις εντολές Vector/SIMD Multimedia Extensions. Το PPE απαιτεί ένα λειτουργικό σύστημα το οποίο έχει επεκταθεί έτσι ώστε να υποστηρίζει τα ιδιαίτερα χαρακτηριστικά του hardware της Cell Broadband Engine, όπως είναι η πολυεπεξεργασία (multiprocessing) με τα SPEs, η πρόσβαση στις λειτουργίες του Vector/SIMD Multimedia Extension του PPE, τον ελεγκτή διακοπών και όλες τις άλλες λειτουργίες της Cell Broadband Engine.

Στο σύνηθες περιβάλλον λειτουργικού συστήματος και ανάπτυξης εφαρμογών το PPE ασχολείται με τη δημιουργία νημάτων και τη διαχείριση πόρων μεταξύ των SPEs. Ο πυρήνας Linux του PPE ελέγχει την εκτέλεση προγραμμάτων στις SPUs. Τα νήματα των SPEs ακολουθούν το μοντέλο νημάτων M:N, πράγμα που σημαίνει ότι M νήματα διαμοιράζονται σε N επεξεργαστικά στοιχεία. Τυπικά, τα νήματα των SPEs τρέχουν μέχρι την ολοκλήρωσή τους, παρ' όλα αυτά είναι pre-emptible (δηλαδή μπορούν να διακοπούν ανά πάσα στιγμή και να συνεχίσουν την εκτέλεσή τους αργότερα – διαδικασία γνωστή και ως context switching) ως προς την πολιτική διαχείρισης και προτεραιοτήτων του νήματος. Το μέγεθος των κβάντων χρόνου για τα νήματα των SPEs είναι τυπικά μεγαλύτερο από τα αντίστοιχα κβάντα χρόνου

των PPE νημάτων, αφού η αλλαγή διεργασίας (context switching) στο SPE είναι μία σχετικά βαριά διαδικασία.

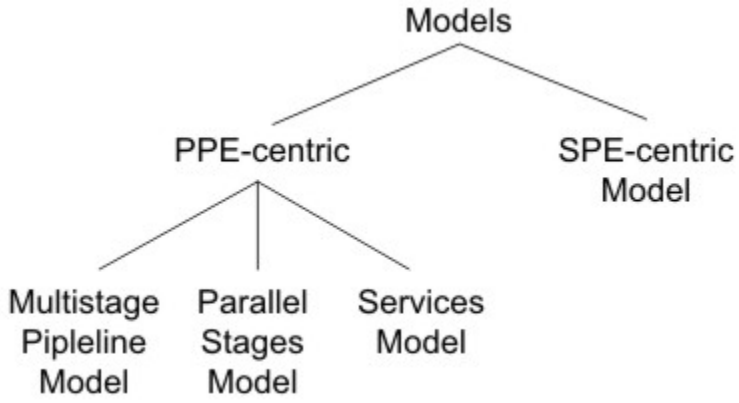
Ο πυρήνας του Linux διαχειρίζεται την εικονική μνήμη, συμπεριλαμβανομένης της απεικόνισης των τοπικών μνημών (LS) και των problem state (PS) κάθε SPE στο χώρο ενεργών διευθύνσεων. Ο πυρήνας επίσης ελέγχει την απεικόνιση στην εικονική μνήμη των πόρων των MFC όπως και τη διαχείριση των σφαλμάτων τμημάτων και σελίδων (segment-faults και page-faults) των MFC. Χαρακτηριστική είναι η υποστήριξη μεγάλων σελίδων (σελίδες των 16 MB), χρησιμοποιώντας την επέκταση hugetlbfs του Linux.

Διαμέριση εφαρμογών (Application partitioning)

Τα προγράμματα που τρέχουν στα εννέα επεξεργαστικά στοιχεία της Cell Broadband Engine τυπικά διαμερίζουν το φόρτο εργασίας στους διαθέσιμους επεξεργαστές. Στη λήψη αποφάσεων σχετικά με το πότε και πώς διαμοιράζεται ο φόρτος εργασίας και τα δεδομένα παίζουν ρόλο οι εξής παράμετροι:

- διαμοιρασμός φόρτου εργασίας
- δομή προγράμματος
- ροή δεδομένων (data flow) του προγράμματος και μοτίβα πρόσβασης (access patterns) στα δεδομένα
- κόστος, σε χρόνο και πολυπλοκότητα, της μετακίνησης κώδικα και δεδομένων μεταξύ των επεξεργαστών
- κόστος φορτώματος του διαύλου και των στοιχείων που είναι συνδεδεμένα πάνω του

Το κύριο μοντέλο που χρησιμοποιείται συνήθως για τη διαμέριση μίας εφαρμογής είναι το PPE-κεντρικό (PPE-centric), όπως φαίνεται στο ακόλουθο σχήμα:



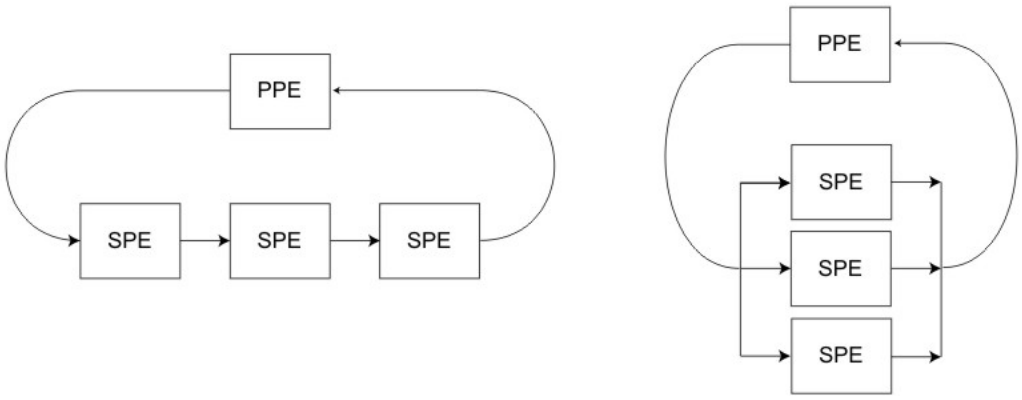
Σε αυτό το μοντέλο η κύρια εφαρμογή τρέχει στο PPE και επιμέρους διεργασίες φορτώνονται στα SPEs. Το PPE ακολούθως συντονίζει και περιμένει τα αποτελέσματα που επιστρέφονται από τα SPEs. Αυτό το μοντέλο ταιριάζει σε εφαρμογές σε σειριακά δεδομένα και παράλληλους υπολογισμούς.

Στο SPE-κεντρικό (SPE-centric) μοντέλο το μεγαλύτερο ποσοστό του κώδικα μοιράζεται στα SPEs. Το PPE λειτουργεί ως ένας κεντρικός διαχειριστής πόρων για τα SPEs. Κάθε SPE φέρνει από τον κύριο αποθηκευτικό χώρο (ή την ίδια του την τοπική μνήμη) τα επόμενα προς επεξεργασία δεδομένα όταν ολοκληρωθεί η δουλειά που κάνει με τα τρέχοντα δεδομένα.

Υπάρχουν τρεις τρόποι με τους οποίους μπορούν να χρησιμοποιηθούν τα SPEs στο PPE-κεντρικό μοντέλο:

- το μοντέλο σωλήνωσης πολλών σταδίων (multistage pipeline)
- το μοντέλο παράλληλων σταδίων (parallel stages)
- το μοντέλο υπηρεσιών (services)

Τα δύο πρώτα από τα παραπάνω μοντέλα φαίνονται στο σχήμα που ακολουθεί.



Multistage Pipeline Model

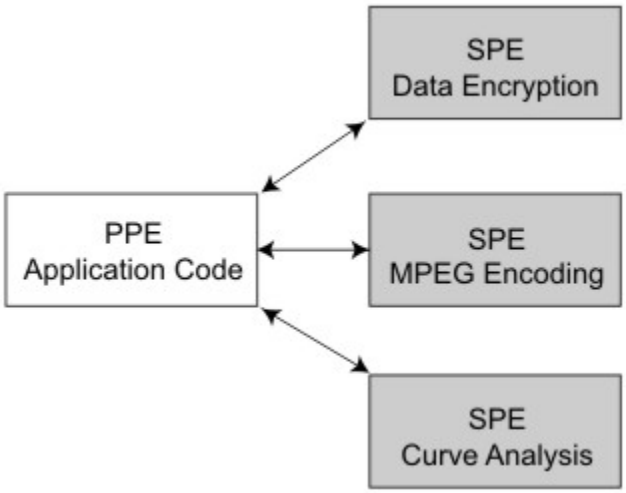
Parallel Stages Model

Εάν μία διεργασία απαιτεί σειριακή στάδια τα SPEs μπορούν να λειτουργήσουν ως στάδια μιας πολύ-σταδιακής σωλήνωσης (multistage pipeline). Αυτό το μοντέλο φαίνεται στην αριστερή πλευρά του παραπάνω σχήματος. Εδώ, η ροή των δεδομένων εισέρχεται στο πρώτο SPE, το οποίο και εκτελεί το πρώτο στάδιο της επεξεργασίας. Έπειτα, το πρώτο SPE προωθεί τα δεδομένα στο επόμενο SPE, το οποίο θα εκτελέσει το δεύτερο στάδιο της επεξεργασίας. Αφότου και το τελευταίο SPE ολοκληρώσει το τελικό στάδιο της επεξεργασίας, τα δεδομένα στέλνονται πίσω στο PPE. Όπως με κάθε αρχιτεκτονική αγωγού, η παράλληλη επεξεργασία γίνεται με διάφορα κομμάτια των δεδομένων να βρίσκονται σε διάφορα στάδια επεξεργασίας της σωλήνωσης. Το multistage pipeline γενικά αποφεύγεται εξ αιτίας της δυσκολίας διαμοιρασμού του φόρτου εργασίας. Επιπρόσθετα, αυτό το μοντέλο αυξάνει τις απαιτήσεις για τη μετακίνηση δεδομένων επειδή όλα τα δεδομένα πρέπει να μετακινηθούν για κάθε στάδιο της σωλήνωσης.

Εάν η προς εκτέλεση εφαρμογή δεν μπορεί να χωριστεί σε διακριτά στάδια, αλλά είναι μία εφαρμογή στην οποία υπάρχει ένας μεγάλος όγκος δεδομένων που μπορεί να διαμεριστεί και κάθε κομμάτι να υποστεί επεξεργασία ταυτόχρονα, τότε έχει γενικά νόημα να χρησιμοποιηθούν τα SPEs για την επεξεργασία διαφορετικών κομματιών των δεδομένων παράλληλα. Το μοντέλο παράλληλων σταδίων (parallel stages) απεικονίζεται στη δεξιά πλευρά του παραπάνω σχήματος.

Ο τρίτος τρόπος με τον οποίον τα SPEs μπορούν να χρησιμοποιηθούν στο PPE-κεντρικό μοντέλο είναι το μοντέλο υπηρεσιών (services model). Σε αυτό το μοντέλο το PPE αναθέτει διαφορετικές υπηρεσίες σε διαφορετικά SPEs και η κύρια διεργασία του PPE καλεί το κατάλληλο SPE όταν χρειάζεται κάποια συγκεκριμένη υπηρεσία. Παρακάτω φαίνεται σχηματικά το μοντέλο υπηρεσιών. Εδώ, ένα SPE πραγματοποιεί κρυπτογράφηση δεδομένων, ένα άλλο SPE κωδικοποιεί βίντεο MPEG και ένα τρίτο SPE επεξεργάζεται ανάλυση

καμπυλών. Η σταθερή, στατική ανάθεση υπηρεσιών σε SPEs πρέπει να αποφεύγεται. Αυτές οι υπηρεσίες θα πρέπει να γίνονται εικονικές και να διαχειρίζονται πάνω σε μία βάση που θα καθορίζεται από τις ανάγκες της εφαρμογής.



Το προγραμματιστικό περιβάλλον (Software development kit)

Υπάρχει διαθέσιμο για τη Cell Broadband Engine ένα περιβάλλον ανάπτυξης εφαρμογών (Software Development Kit – SDK). Το SDK περιέχει τα απαραίτητα εργαλεία για την ανάπτυξη προγραμμάτων για τη Cell Broadband Engine. Περιλαμβάνει, μεταξύ άλλων, τα ακόλουθα:

- Τον Πλήρη Εξομοιωτή Συστήματος (Full System Emulator) της IBM για τη Cell Broadband Engine, τον systemsim
- Το system root image που περιέχει το περιβάλλον εκτέλεσης Linux για χρήση εντός του systemsim
- Εργαλεία GNU που περιλαμβάνουν compilers για C και C++, linkers, assemblers και binary utilities τόσο για την PPU όσο και για την SPU
- Τους compilers της IBM xlc (για C/C++) και xlf (για Fortran), τόσο για την PPU όσο και για την SPU
- Τη βιβλιοθήκη newlib για την SPU, η οποία είναι μία standard βιβλιοθήκη της C σχεδιασμένη για χρήση σε embedded συστήματα

- Τους debuggers gdb τόσο για την PPU όσο και για την SPU, με υποστήριξη για απομακρυσμένο debugging μέσω gdbserver. Ο debugger της PPU παρέχει επίσης συνδυασμένη υποστήριξη για debugging PPU και SPU.

- Μία έκδοση Linux PPC64 με προσθήκες για την υποστήριξη της CBE

- Τη βιβλιοθήκη SPE Runtime Management Library η οποία παρέχει μία τυποποιημένη, χαμηλού επιπέδου διεπαφή για πρόσβαση στα SPEs κατά τον προγραμματισμό εφαρμογών

- Βιβλιοθήκες που βοηθούν στην ανάπτυξη και εκτέλεση παράλληλων εφαρμογών

- Διάφορα εργαλεία ανάλυσης της επίδοσης

- Ένα Ενοποιημένο Περιβάλλον Ανάπτυξης (Integrated Development Environment – IDE) βασιζόμενο στο Eclipse για την αύξηση της παραγωγικότητας του προγραμματιστή και την ενοποίηση των εργαλείων ανάπτυξης

- Τυποποιημένες SIMD μαθηματικές βιβλιοθήκες για τα σύνολα εντολών Vector/SIMD Multimedia Extension και SPU

- Βιβλιοθήκες Συστήματος Επιτάχυνσης Μαθηματικών (Mathematical Acceleration Subsystem – MASS) που υποστηρίζουν τόσο τα μεγάλα (long) όσο και τα μικρά (short) διανύσματα SIMD

- Βιβλιοθήκες εφαρμογών βελτιστοποιημένες για το Cell (Cell optimized domain-specific application libraries) που περιλαμβάνουν βιβλιοθήκη Βασικών Υποπρογραμμάτων Γραμμικής Άλγεβρας (Basic Linear Algebra Subprograms – BLAS), βιβλιοθήκη Γρήγορου Μετασχηματισμού Fourier (Fast Fourier Transform – FFT) και βιβλιοθήκη Γεννήτριας Τυχαίων Αριθμών Monte Carlo (Monte Carlo Random Number Generator)

- Παραδείγματα πηγαίου κώδικα, παραδείγματα βιβλιοθηκών, benchmarks και προγράμματα επίδειξης (demos)

Πέρα των παραπάνω, υπάρχουν και διανομές Linux για υπολογιστικά συστήματα που βασίζονται στην Cell Broadband Engine, όπως είναι τα μηχανήματα Blade της IBM ή η κονσόλα PlayStation 3. Χρησιμοποιώντας το SDK της Cell Broadband Engine σε ένα τέτοιο

σύστημα μπορεί να γίνει ανάπτυξη εφαρμογών κατ' ευθείαν στο πραγματικό μηχάνημα, χωρίς ανάγκη για χρήση του εξομοιωτή.

Επεκτάσεις της γλώσσας C/C++ για την PPU (PPU intrinsics)

Υπάρχει διαθέσιμο ένα σύνολο επεκτάσεων της γλώσσας C για τον προγραμματισμό του PPE χρησιμοποιώντας το σύνολο εντολών Vector/SIMD Multimedia Extension. Οι επεκτάσεις αυτές περιλαμβάνουν επιπρόσθετους διανυσματικούς τύπους δεδομένων και ένα μεγάλο σύνολο από εντολές (intrinsics) για βαθμωτά και διανυσματικά δεδομένα. Τα intrinsics είναι στην ουσία εντολές inline assembly υπό τη μορφή κλήσεων συναρτήσεων και οι οποίες (εντολές) έχουν σύνταξη οικεία προς τους προγραμματιστές υψηλού επιπέδου που χρησιμοποιούν τη γλώσσα C.

Τα intrinsics παρέχουν ρητό έλεγχο των εντολών PPU ή Vector/SIMD Multimedia Extension χωρίς να υπάρχει ανάγκη για απ' ευθείας διαχείριση καταχωρητών και εντολών scheduling, όπως απαιτείται σε ένα πρόγραμμα γραμμένο σε assembly. Ένας compiler που υποστηρίζει αυτές τις επεκτάσεις της C θα παράγει κώδικα βελτιστοποιημένο για την αρχιτεκτονική PPU και/ή Vector/SIMD Multimedia Extension.

Βαθμωτές επεκτάσεις (Scalar intrinsics)

Υπάρχει διαθέσιμο ένα μικρό σύνολο από συγκεκριμένες επεκτάσεις ώστε να παρέχεται πρόσβαση στον προγραμματιστή στο υποκείμενο σύνολο εντολών PPU από τη γλώσσα C. Οι επεκτάσεις αυτές δηλώνονται στο αρχείο κεφαλίδας συστήματος ppu_intrinsics.h. Περισσότερες πληροφορίες για αυτές υπάρχουν στο έγγραφο C/C++ Language Extensions for Cell Broadband Engine Architecture της IBM.

Διανυσματικοί τύποι δεδομένων

Το μοντέλο Vector/SIMD Multimedia Extension προσθέτει ένα σύνολο θεμελιωδών τύπων δεδομένων που ονομάζονται διανυσματικοί τύποι (vector types). Οι υποστηριζόμενοι διανυσματικοί τύποι δίνονται στον ακόλουθο πίνακα.

Τύπος Διανυσματικού Δεδομένου	**Σημασία**	**Τιμές**
vector unsigned char	Δεκαέξι 8bit απρόσημες τιμές	0 … 255
vector signed char	Δεκαέξι 8bit προσημασμένες τιμές	-128 … 127
vector bool char	Δεκαέξι 8bit απρόσημες Boolean	0 (false), 255 (true)
vector unsigned short	Οκτώ 16bit απρόσημες τιμές	0 … 65535
vector unsigned short int	Οκτώ 16bit απρόσημες τιμές	0 … 65535
vector signed short	Οκτώ 16bit προσημασμένες τιμές	-32768 … 32767

vector signed short int	Οκτώ 16bit προσημασμένες τιμές	-32768 … 32767
vector bool short	Οκτώ 16bit απρόσημες Boolean	0 (false), 65535 (true)
vector bool short int	Οκτώ 16bit απρόσημες Boolean	0 (false), 65535 (true)
vector unsigned int	Τέσσερεις 32bit απρόσημες τιμές	$0 \ldots 2^{32} - 1$
vector signed int	Τέσσερεις 32bit προσημασμένες τιμές	$-2^{31} \ldots 2^{31} - 1$
vector bool int	Τέσσερεις 32bit απρόσημες Boolean	0 (false), $2^{31} - 1$ (true)
vector float	Τέσσερεις 32bit απλής ακρίβειας	Τιμές κατά IEEE 754
vector pixel	Οκτώ 16bit απρόσημες τιμές	1 / 5 / 5 / 5 pixel

Στη στήλη των τιμών οι αριθμοί είναι στο δεκαδικό σύστημα. Οι διανυσματικοί καταχωρητές έχουν εύρος 128 bits και μπορούν να περιέχουν:

- Δεκαέξι 8bit τιμές, προσημασμένες ή απρόσημες
- Οκτώ 16bit τιμές, προσημασμένες ή απρόσημες
- Τέσσερεις 32bit τιμές, προσημασμένες ή απρόσημες
- Τέσσερεις τιμές κινητής υποδιαστολής απλής ακρίβειας IEEE-754

Οι διανυσματικοί τύποι χρησιμοποιούν το πρόθεμα vector μπροστά από τους τυποποιημένους τύπους δεδομένων της C – για παράδειγμα vector signed int και vector unsigned short. Ένας διανυσματικός τύπος αναπαριστά ένα διάνυσμα που περιέχει τόσους από τους προσδιοριζόμενους τύπους δεδομένων της C όσους χωράνε σε έναν καταχωρητών των 128 bits. Έτσι, το vector signed int είναι ένα όρισμα των 128 bits που περιέχει τέσσερεις 32bit προσημασμένους ακεραίους. Το vector unsigned short είναι ένα όρισμα των 128 bits που περιέχει οκτώ απρόσημες τιμές. Καθώς η λέξη "vector" είναι μία δεσμευμένη λέξη (keyword) για τους τύπους δεδομένων Vector/SIMD Multimedia Extension πρέπει να αποφεύγεται οι χρήση της σε άλλα σημεία του προγράμματος, όπως π.χ. για το όνομα μιας μεταβλητής.

Η εισαγωγή των θεμελιωδών διανυσματικών τύπων δεδομένων επιτρέπει στον compiler να παρέχει ισχυρότερο έλεγχο τύπων (type checking) και να υποστηρίζει overloaded πράξεις πάνω σε διανυσματικούς τύπους.

Διανυσματικά intrinsics

Τα intrinsics του συνόλου εντολών Vector/SIMD Multimedia Extension ομαδοποιούνται σε τρεις κατηγορίες. Αυτές είναι οι εξής:

- Ειδικά (Specific) Intrinsics – Intrinsics που έχουν αντιστοιχία ένα-προς-ένα με μία και μόνο εντολή assembly

- Γενικά (Generic) Intrinsics – Intrinsics που αντιστοιχούν σε μία ή περισσότερες εντολές assembly, αναλόγως του τύπου των παραμέτρων εισόδου

- Κατηγορηματικά (Predicate) Intrinsics – Intrinsics που συγκρίνουν τιμές και επιστρέφουν έναν ακέραιο ο οποίος μπορεί να χρησιμοποιηθεί άμεσα ως τιμή ή ως συνθήκη για διακλάδωση

Τα intrinsics και τα κατηγορήματα του συνόλου εντολών Vector/SIMD Multimedia Extension χρησιμοποιούν το πρόθεμα vec_ μπροστά από το μνημονικό μίας εντολής ή λειτουργίας assembly. Τα Κατηγορηματικά Intrinsics χρησιμοποιούν τα προθέματα vec_all και vec_any. Κατά τη μεταγλώττιση τα intrinsics παράγουν μία ή περισσότερες εντολές της Vector/SIMD Multimedia Extension assembly.

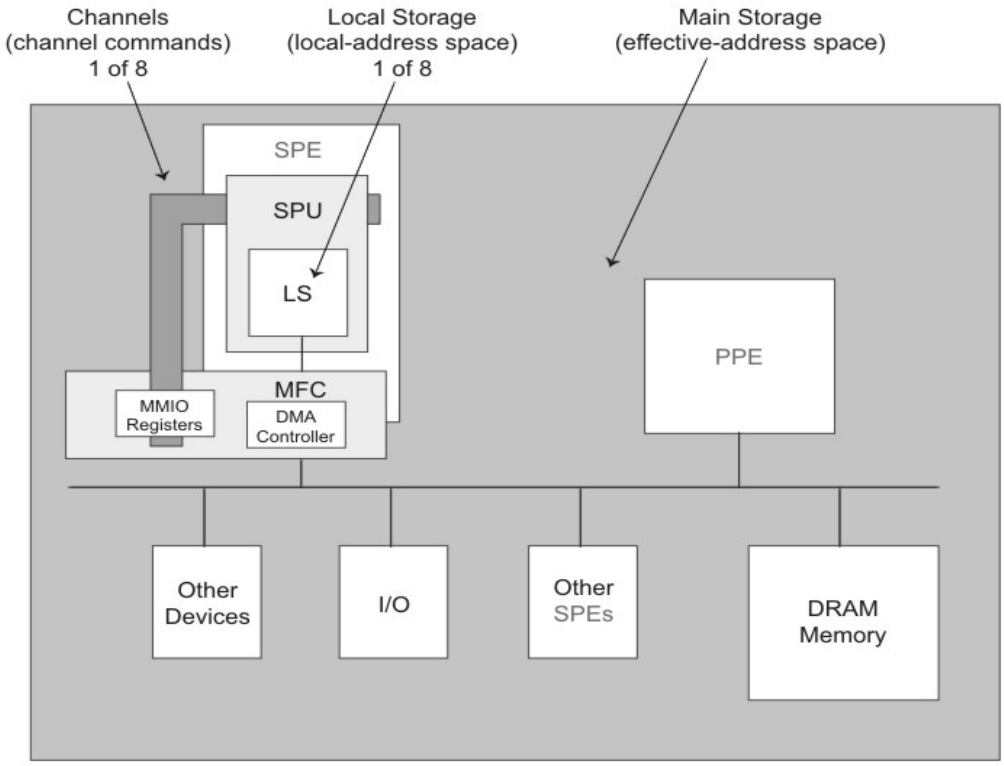
Πλήρης λίστα και επεξήγηση των διανυσματικών intrinsics υπάρχει στο έγγραφο C/C++ Language Extensions for Cell Broadband Engine Architecture της IBM.

Το PPE και τα SPEs

Ακολουθεί ανάλυση της σχέσης μεταξύ του Επεξεργαστικού Στοιχείου PowerPC (PPE) και των Συνεργατικών Επεξεργαστικών Στοιχείων (SPEs).

Πεδία Αποθήκευσης (Storage Domains)

Στη Cell Broadband Engine ορίζονται τρεις τύποι πεδίων αποθήκευσης: ένα πεδίο κύριου αποθηκευτικού χώρου (main storage domain), οκτώ πεδία τοπικής μνήμης των SPEs (SPE local store domains) και οκτώ πεδία καναλιών των SPEs (SPE channel domains). Αυτοί οι τύποι φαίνονται στο παρακάτω σχήμα. Το πεδίο του κύριου αποθηκευτικού χώρου, το οποίο αποτελεί ολόκληρο τον χώρο ενεργών διευθύνσεων, μπορεί να ρυθμιστεί από το λειτουργικό σύστημα του PPE ώστε να μοιράζεται από όλους τους επεξεργαστές και τις απεικονισμένες στη μνήμη συσκευές εισόδου/εξόδου (όλη η είσοδος/έξοδος απεικονίζεται στη μνήμη). Παρ' όλα αυτά, τα πεδία των τοπικών μνημών και των problem state των καναλιών είναι ιδιωτικά για την SPU, τη LS και τον MFC κάθε SPE.



Ένα SPE μπορεί να φέρει εντολές μόνο από τη δική του τοπική μνήμη και οι φορτώσεις και οι αποθηκεύσεις μπορούν να προσπελάσουν μόνο την τοπική μνήμη. Ένα SPE ή ένα PPE πραγματοποιεί μεταφορές δεδομένων μεταξύ της τοπικής μνήμης του SPE και του κύριου αποθηκευτικού χώρου χρησιμοποιώντας κυρίως μεταφορές DMA που ελέγχονται από τον ελεγκτή DMA του MFC για εκείνο το SPE. Το λογισμικό στην SPU του SPE αλληλεπιδρά με τον MFC μέσω καναλιών τα οποία βάζουν σε ουρά εντολές DMA και παρέχουν πρόσθετες λειτουργίες, όπως τα mailboxes, η ειδοποίηση μέσω σημάτων και η πρόσβαση με βοηθητικούς πόρους.

Ένα πρόγραμμα SPE αναφέρεται στη δική του τοπική μνήμη χρησιμοποιώντας Διευθύνσεις Τοπικής Μνήμης (Local Store Address – LSA). Στην τοπική μνήμη του κάθε SPE ανατίθεται επίσης και ένα εύρος Πραγματικών Διευθύνσεων (Real Address – RA) εντός του εύρους της μνήμης του συστήματος. Αυτό επιτρέπει στον προνομιούχο λογισμικό να απεικονίσει περιοχές της τοπικής μνήμης στο χώρο ενεργών διευθύνσεων (Effective Address – EA), έτσι ώστε το PPE, τα άλλα SPEs και άλλες συσκευές που παράγουν Ενεργές Διευθύνσεις μπορούν να προσπελάσουν τη συγκεκριμένη τοπική μνήμη.

Ο MFC του κάθε SPE λειτουργεί ως μία μηχανή διακίνησης δεδομένων. Οι αιτήσεις για μεταφορές DMA περιέχουν τόσο μία LSA όσο και μία EA. Έτσι, μπορούν να προσπελάσουν και την τοπική μνήμη ενός SPE και τον κύριο αποθηκευτικό χώρο και έτσι να

αρχικοποιήσουν μεταφορές DMA μεταξύ των αντίστοιχων πεδίων. Ο MFC διεκπεραιώνει αυτή τη διαδικασία διατηρώντας και επεξεργαζόμενος την ουρά εντολών MFC. Οι αιτήσεις DMA μπορούν να σταλούν σε έναν MFC είτε από το λογισμικό της αντίστοιχης SPU ή του PPE είτε από οποιαδήποτε άλλη επεξεργαστική μονάδα η οποία έχει πρόσβαση στους MMIO καταχωρητές του problem state του MFC.

Οι αιτήσεις που μπαίνουν στην ουρά μετατρέπονται σε μεταφορές DMA. Κάθε MFC μπορεί να διατηρήσει και να επεξεργαστεί πολλές εν εξελίξει εντολές αιτήσεων DMA και μεταφορές DMA. Ο MFC μπορεί επίσης να διαχειριστεί αυτόνομα μία σειρά από μεταφορές DMA σε απάντηση μίας εντολής λίστας DMA από την αντίστοιχή του SPU. Κάθε μεταφορά DMA έχει μία ετικέτα (tag) με ένα αναγνωριστικό ομάδας (Tag Group ID) των 5 bits. Το λογισμικό μπορεί να χρησιμοποιήσει αυτό το αναγνωριστικό για να ελέγξει ή να περιμένει την ολοκλήρωση όλων των εντολών που ανήκουν σε μία ή σε περισσότερες ομάδες.

Ο MFC υποστηρίζει μεγέθη μεταφορών που είναι φυσικώς ευθυγραμμισμένα στα 1, 2, 4 ή 8 bytes και στα πολλαπλάσια των 16 bytes, με μέγιστο μέγεθος μεταφοράς τα 16KB. Η βέλτιστη επίδοση επιτυγχάνεται για μεταφορές που τόσο η EA όσο και η LSA είναι ευθυγραμμισμένες στα 128 bytes και το μέγεθος της μεταφοράς είναι πολλαπλάσιο των 128 bytes.

Κάθε MFC έχει μία αντιστοιχισμένη μονάδα διαχείρισης μνήμης (memory management unit – MMU) η οποία διατηρεί και επεξεργάζεται τις πληροφορίες για τη μετάφραση των διευθύνσεων και την άδεια πρόσβασης που παρέχονται από το λειτουργικό σύστημα του PPE. Αυτή η MMU είναι διαφορετική και διακριτή από αυτή που χρησιμοποιεί το PPE. Προκειμένου να επεξεργαστεί μία ενεργό διεύθυνση που παρέχεται σε μία εντολή DMA η MMU χρησιμοποιεί την ίδια μέθοδο που χρησιμοποιούν οι λειτουργίες διαχείρισης μνήμης του PPE. Έτσι, οι μεταφορές DMA είναι συναφείς ως προς τον κύριο αποθηκευτικό χώρο. Οι ιδιότητες του κύριου αποθηκευτικού χώρου του συστήματος καθορίζονται από τους πίνακες σελίδων και τμημάτων της αρχιτεκτονικής PowerPC.

Το PPE ή άλλες επεξεργαστικές συσκευές μπορούν να αρχικοποιήσουν εντολές MFC σε ένα συγκεκριμένο MFC προσπελάζοντας του Καταχωρητές Παραμέτρων Εντολών του MFC (MFC Command-Parameter Registers). Οι καταχωρητές αυτοί, που περιγράφονται στον ακόλουθο πίνακα, είναι απεικονισμένοι στο χώρο πραγματικών διευθύνσεων του συστήματος. Το PPE πραγματοποιεί MMIO αναγνώσεις και εγγραφές για να έχει πρόσβαση σε αυτούς τους καταχωρητές. Οι καταχωρητές περιέχονται εντός της περιοχής μνήμης του κάθε SPE και οι εντολές αιτήσεων DMA γίνονται γράφοντας παραμέτρους σε αυτούς.

Όνομα	**Μνημονικό**	**Max. Καταχωρήσεις**	**R/W**	**Εύρος (bits)**

MFC Local-Storage Address	MFC_LSA	1	W	32
MFC Effective Address High	MFC_EAH	1	W	32
MFC Effective Address Low	MFC_EAL	1	W	32
MFC Transfer Size MFC Command Tag ID	MFC_Size MFC_TagID	1	W	32
MFC Class ID and Command Opcode	MFC_ClassID_CMD	8	W	32
MFC Command Status	MFC_CMDStatus	1	R	32

Οι καταχωρητές MFC_EAH και MFC_EAL μπορούν να εγγραφούν σε μία αποθήκευση των 64 bits. Παρομοίως, οι MFC_Size, MFC_TagID και MFC_ClassID_CMD μπορούν επίσης να εγγραφούν με μία αποθήκευση των 64 bits.

Έκδοση εντολών DMA από το PPE

Για να εκδοθεί μία εντολή DMA από το PPE ακολουθείται η εξής διαδικασία σχετικά με τους καταχωρητές Παραμέτρων Εντολών του MFC:

1. Εγγραφή της διεύθυνσης της τοπικής μνήμης στον καταχωρητή MFC_LSA

2. Εγγραφή των μερών high και low της ενεργού διεύθυνσης στους καταχωρητές MFC_EAH και MFC_EAL

3. Εγγραφή του μεγέθους της μεταφοράς και της ετικέτας ID στους καταχωρητές MFC_Size και MFC_TagID

4. Εγγραφή του αναγνωριστικού κατηγορίας (class ID) και του opcode της εντολής στον καταχωρητή MFC_ClassIS_CMD

5. Ανάγνωση του καταχωρητή MFC_CMDStatus ώστε να εξακριβωθεί η επιτυχία ή η αποτυχία της προσπάθειας τοποθέτησης της εντολής DMA στην ουρά

Τα δύο λιγότερο σημαντικά bits της τιμής της κατάστασης εντολής που επιστρέφεται από την ανάγνωση του καταχωρητή MFC_CMDStatus υποδεικνύει την επιτυχία ή κάποιο λάθος κατά την προσπάθεια εισαγωγής μία εντολής DMA στην ουρά. Οι τιμές αυτών των δύο bits έχουν την εξής ερμηνεία:

- 0 – Υποδεικνύει ότι η τοποθέτηση στην ουρά ήταν επιτυχής.

- 1 – Υποδεικνύει ότι συνέβη κάποιο λάθος σειράς κατά την τοποθέτηση στην ουρά της εντολής DMA. Για παράδειγμα, συνέβη μία διακοπή και στη

συνέχεια ξεκίνησε μία άλλη ακολουθία τοποθέτησης στην ουρά εντολής DMA από το διαχειριστή της διακοπής. Σε αυτήν την περίπτωση, η διαδικασία εισαγωγής στην ουρά πρέπει να ξεκινήσει από την αρχή.

- 2 – Υποδεικνύει ότι η τοποθέτηση στην ουρά απέτυχε εξ αιτίας ανεπαρκούς χώρου στην ουρά των εντολών.

- 3 – Υποδεικνύει ότι συνέβησαν και τα δύο προηγούμενα λάθη.

Στην περίπτωση ανεπαρκούς χώρου το λογισμικό μπορεί είτε να περιμένει μέχρι κάποιος χώρος να γίνει διαθέσιμος προτού επιχειρήσει να ξανακάνει τη μεταφορά DMA είτε να συνεχίσει την προσπάθεια εισαγωγής στην ουρά μέχρι αυτή να επιτύχει.

Βιβλιοθήκη Διαχείρισης των SPEs κατά το Χρόνο Εκτέλεσης (SPE Runtime Management Library)

Η Βιβλιοθήκη Διαχείρισης των SPEs κατά το Χρόνο Εκτέλεσης (SPE Runtime Management Library – libspe) αποτελεί την τυποποιημένη διεπαφή για τον προγραμματισμό (Application Programming Interface) εφαρμογών σε χαμηλό επίπεδο η οποία επιτρέπει στις εφαρμογές την πρόσβαση στα SPEs της Cell Broadband Engine. Αυτή η βιβλιοθήκη παρέχει ένα API το οποίο είναι ουδέτερο ως προς το υποκείμενο λειτουργικό σύστημα και τις μεθόδους του για τη διαχείριση των SPEs. Οι συναρτήσεις της ορίζονται στην κεφαλίδα συστήματος libspe2.h.

Κάποιες υλοποιήσεις της libspe μπορούν να παρέχουν επιπρόσθετη λειτουργικότητα η οποία επιτρέπει πρόσβαση στο λειτουργικό σύστημα ή σε εξαρτώμενα από την υλοποίηση κομμάτια της διαχείρισης των SPEs κατά το χρόνο εκτέλεσης. Αυτή η επιπρόσθετη λειτουργικότητα δεν υπόκειται σε τυποποίηση και μπορεί να οδηγήσει σε μη-φορητό κώδικα και σε εξαρτήσεις από συγκεκριμένες υλοποιημένες εκδόσεις της βιβλιοθήκης.

Γενικά, οι εφαρμογές δεν έχουν έλεγχο πάνω στους φυσικούς πόρους συστήματος των SPEs. Υπεύθυνο για τη διαχείριση αυτών των πόρων είναι το λειτουργικό σύστημα. Οι εφαρμογές διαχειρίζονται και χρησιμοποιούν δομές λογισμικού (software constructs) που ονομάζονται SPE contexts. Αυτά τα SPE contexts είναι μία λογική αναπαράσταση ενός SPE και είναι το βασικό αντικείμενο πάνω στο οποίο λειτουργεί η libspe. Οι εφαρμογές δεν μπορούν να τροποποιήσουν αυτή τη δομή απ' ευθείας αλλά την προσπελάζουν χρησιμοποιώντας δείκτες προς τέτοιες δομές τους οποίους περνούν ως παραμέτρους κατά την κλήση των συναρτήσεων της βιβλιοθήκης. Οι δομές αυτές λειτουργούν ως αναγνωριστικά για την κάθε SPE. Το λειτουργικό σύστημα αναθέτει (schedules) για εκτέλεση τα SPE contexts από όλες τις τρέχουσες εφαρμογές σε φυσικούς πόρους των SPEs που υπάρχουν στο σύστημα

σύμφωνα με τις προτεραιότητες και πολιτικές ανάθεσης που σχετίζονται με τα εκτελέσιμα SPE contexts. Η libspe παρέχει επίσης το μέσον για την επικοινωνία και μεταφορά δεδομένων μεταξύ του PPE και των SPEs.

Το βασικό σχήμα για μια απλή εφαρμογή που χρησιμοποιεί μία SPE είναι το ακόλουθο:

1. Δημιουργία ενός SPE context
2. Φόρτωση ενός εκτελέσιμου στην SPE αντικειμένου στην τοπική μνήμη που αντιστοιχεί στο SPE context
3. Εκτέλεση του SPE context. Ο έλεγχος μεταφέρεται στο λειτουργικό σύστημα, το οποίο διεκπεραιώνει την πραγματική ανάθεση του context σε κάποιο φυσικό SPE του συστήματος.
4. Καταστροφή του SPE context

Το τρίτο βήμα αναπαριστά μία σύγχρονη κλήση στο λειτουργικό σύστημα. Η καλούσα εφαρμογή μπλοκάρει μέχρι το SPE να σταματήσει την εκτέλεση και το λειτουργικό σύστημα να επιστρέψει από την κλήση συστήματος που προκάλεσε την εκτέλεση στο SPE.

Ταυτόχρονη χρήση πολλαπλών SPEs

Πολλές εφαρμογές χρειάζεται να χρησιμοποιήσουν πολλαπλά SPEs ταυτόχρονα. Σε αυτήν την περίπτωση οι εφαρμογές πρέπει να δημιουργήσουν τουλάχιστον τόσα νήματα όσα είναι και τα ταυτόχρονα SPE contexts που απαιτούνται. Κάθε ένα από αυτά τα νήματα μπορεί να τρέξει ένα και μόνο SPE context κάθε φορά. Εάν χρειάζονται N ταυτόχρονα SPE contexts είναι σύνηθες να υπάρχει ένα νήμα της κύριας εφαρμογής συν N νήματα που αναλαμβάνουν την εκτέλεση των SPE contexts.

Το βασικό σχήμα για μια απλή εφαρμογή που τρέχει N SPE contexts είναι το ακόλουθο:

1. Δημιουργία N SPE contexts
2. Φόρτωση του κατάλληλου εκτελέσιμου SPE αντικειμένου στις εκάστοτε τοπικές μνήμες που αντιστοιχούν στα εκάστοτε SPE contexts
3. Δημιουργία N νημάτων:
    a. Σε κάθε ένα από αυτά τα νήματα τρέχει ένα από τα SPE contexts

b. Τερματισμός νήματος όταν ολοκληρωθεί η εκτέλεση

4. Αναμονή για τον τερματισμό όλων των νημάτων

5. Καταστροφή όλων των N SPE contexts

Υπάρχουν και άλλα σχήματα που μπορούν πιθανώς να χρησιμοποιηθούν και, αναλόγως την εφαρμογή, μπορεί να είναι περισσότερο κατάλληλα.

Συναρτήσεις για το PPE

Προκειμένου να παρασχεθεί αυτή η λειτουργικότητα, η libspe περιέχει τα ακόλουθα σύνολα από συναρτήσεις για το PPE με τις οποίες:

- Δημιουργούνται και καταστρέφονται SPE contexts και SPE gang contexts (το gang context είναι μία συλλογή από contexts)

- Φορτώνονται εκτελέσιμα στο SPE αντικείμενα στην τοπική μνήμη ενός SPE για να εκτελεστούν

- Εκκίνηση της εκτέλεσης των προγραμμάτων στα SPEs και απόκτηση πληροφοριών σχετικά με τους λόγους που ένα SPE σταμάτησε την εκτέλεση

- Λήψη ασύγχρονων γεγονότων (SPE events) που δημιουργούνται από κάποιο SPE

- Πρόσβαση στις λειτουργίες της περιοχής problem state του MFC, οι οποίες περιλαμβάνουν:

  - Έκδοση εντολών του MFC

  - Λειτουργία για τον έλεγχο ολοκλήρωσης μίας ομάδας μεταφορών στον MFC

  - Λειτουργίες του Mailbox

  - Λειτουργίες της ειδοποίησης μέσω σημάτων

- Άμεση πρόσβαση των εφαρμογών στην τοπική μνήμη και τις περιοχές problem state ενός SPE

- Δήλωση για ένα πρόγραμμα SPE κλήσεων βιβλιοθήκης υποβοηθούμενων από το PPE

Δημιουργία SPE contexts

Το SPE context είναι μία από τις βασικές δομές δεδομένων της υλοποίησης της libspe. Διατηρεί όλες τις σταθερές πληροφορίες για ένα «λογικό SPE» που χρησιμοποιείται από την εφαρμογή. Αυτή η δομή δεδομένων θα πρέπει να προσπελαύνεται μέσω κλήσεων του API της libspe και όχι άμεσα.

Προτού γίνει εφικτή η χρήση ενός SPE θα πρέπει να δημιουργηθεί και να αρχικοποιηθεί μία δομή δεδομένων του SPE context. Αυτό επιτυγχάνεται με την κλήση της συνάρτησης spe_context_create. Όταν μία εφαρμογή δεν χρειάζεται πλέον ένα συγκεκριμένο SPE context θα πρέπει να καλέσει τη συνάρτηση spe_context_destroy για να αποδεσμευθούν όλοι οι σχετιζόμενοι πόροι και να ελευθερωθεί η μνήμη που χρησιμοποιείται από τη δομή δεδομένων του SPE context.

Το SPE gang context είναι μία άλλη βασική δομή δεδομένων της υλοποίησης της libspe. Διατηρεί όλες τις σταθερές πληροφορίες σχετικά με μία ομάδα από SPE contexts η οποία θα πρέπει να αντιμετωπίζεται ως ενιαία ομάδα, δηλαδή όλα τα SPE contexts που ανήκουν στο ίδιο SPE gang context πρέπει να εκτελούνται μαζί με συγκεκριμένες ιδιότητες. Αυτή η δομή δεδομένων θα πρέπει να προσπελάζεται μόνο μέσω κλήσεων του API της libspe και όχι άμεσα.

Προτού γίνει εφικτή η χρήση ενός SPE gang context, δηλαδή προτού είναι εφικτή η κλήση της spe_context_create ώστε να προστεθούν SPE contexts ως μέλη του gang, θα πρέπει να δημιουργηθεί και να αρχικοποιηθεί η δομή δεδομένων του SPE gang context. Αυτό επιτυγχάνεται καλώντας τη συνάρτηση spe_gang_context_create. Όταν μία εφαρμογή δεν χρειάζεται πλέον ένα συγκεκριμένα SPE gang context θα πρέπει να αποδεσμεύσει όλους τους σχετιζόμενους πόρους και να ελευθερώσει τη μνήμη που χρησιμοποιείται από τη δομή δεδομένων του SPE gang context. Αυτό επιτυγχάνεται πρώτα με την κλήση της spe_context_destroy ώστε να καταστραφούν όλα τα SPE contexts που σχετίζονται με το συγκεκριμένο gang και μετά με την κλήση της συνάρτησης spe_gang_context_destroy.

Προγραμματίζοντας σε γλώσσα C, οι παραπάνω συναρτήσεις ορίζονται ως εξής:

spe_context_ptr_t spe_context_create(unsigned int flags, spe_gang_context_ptr_t gang), όπου flags είναι σημαίες που καθορίζουν ιδιότητες του SPE context

int spe_context_destroy(spe_ context_ptr_t spe)

spe_gang_context_ptr_t spe_gang_context_create(unsigned int flags), όπου flags είναι σημαίες που καθορίζουν ιδιότητες του SPE gang context

int spe_gang_context_destroy(spe_ gang_context_ptr_t gang)

Πληροφορίες για τη CPU

Οι εφαρμογές απαιτούν συχνά πληροφορίες σχετικά με το σύστημα στο οποίο τρέχουν, όπως το πλήθος των CPUs (PPEs), το πλήθος των SPEs και άλλες πληροφορίες σχετικά με το περιβάλλον επεξεργασίας. Οι πληροφορίες αυτές αποκτώνται με την κλήση της συνάρτησης spe_cpu_info_get.

int spe_cpu_info_get(unsigned int info_requested, int cpu_node), όπου info_requested είναι ο τύπος των πληροφοριών που ζητούνται και cpu_node ο κόμβος (σε συστήματα NUMA) για τον οποίον ζητούνται οι πληροφορίες αυτές

Χειρισμός των SPE program images

Προτού γίνει εφικτή η εκτέλεση εντός SPE context θα πρέπει να φορτωθεί ένα πρόγραμμα SPE στην τοπική μνήμη ενός SPE. Αυτό γίνεται καλώντας τη συνάρτηση spe_program_load. Το πρόγραμμα SPE μπορεί να είναι είτε ένα ανεξάρτητο ELF image σε ένα αρχείο ή μπορεί να είναι ενσωματωμένο σε ειδικές θέσεις μέσα στο εκτελέσιμο του κύριου νήματος. Η πρώτη περίπτωση απαιτεί το φόρτωμα του SPE program image στη μνήμη καλώντας αρχικά τη συνάρτηση spe_image_open, η οποία επιστρέφει ένα δείκτη που χρησιμοποιείται στη συνέχεια για τη φόρτωση του προγράμματος. Η αντίστοιχη συνάρτηση που καλείται για το κλείσιμο του εκτελέσιμου αρχείου είναι η spe_image_close.

spe_program_handle_t * spe_image_open(const char * filename)

int spe_image_close(spe_program_handle_t * program)

int spe_program_load(spe_context_ptr_t spe, spe_program_handle_t * program)

Έλεγχος της εκτέλεσης στα SPEs

Αφότου η εφαρμογή έχει δημιουργήσει ένα SPE context και έχει φορτώσει ένα πρόγραμμα SPE στην τοπική της μνήμη μπορεί να καλέσει τη συνάρτηση spe_context_run για να τρέξει το SPE context. Ένα νήμα το οποίο εκτελεί ένα SPE context ονομάζεται νήμα SPE. Η συνάρτηση του API για την εκτέλεση ενός context είναι μία σύγχρονη, blocking κλήση από την προοπτική του νήματος που την χρησιμοποιεί πράγμα που σημαίνει ότι όσο εκτελείται ένα πρόγραμμα SPE το σχετιζόμενο νήμα του PPE μπλοκάρει και συνήθως μεταβαίνει σε κατάσταση "sleep" από το λειτουργικό σύστημα. Όταν το πρόγραμμα SPE τερματίσει, είτε επειδή έφτασε το «κανονικό» σημείο εξόδου ή σε μία εντολή stop and signal

είτε επειδή συνέβη κάποια συνθήκη λάθους, η συνάρτηση spe_context_run επιστρέφει και η τιμή του επιστρεφόμενου αποτελέσματος προσδιορίζει την ακριβή συνθήκη υπό την οποία τερμάτισε το πρόγραμμα SPE.

Πολλές εφαρμογές χρειάζονται να χρησιμοποιήσουν πολλαπλά SPEs ταυτόχρονα. Σε αυτήν την περίπτωση, η εφαρμογή πρέπει να δημιουργήσει τουλάχιστον τόσα νήματα όσα και τα ταυτόχρονα SPE contexts που απαιτούνται, χρησιμοποιώντας τυποποιημένες μεθόδους του λειτουργικού συστήματος. Κάθε ένα από αυτά τα νήματα μπορεί να τρέξει ένα και μόνο SPE context ανά πάσα στιγμή. Εάν χρειάζονται N ταυτόχρονα SPE contexts είναι σύνηθες να χρησιμοποιούνται N+1 νήματα – ένα κύριο νήμα της εφαρμογής το οποίο θα «διευθύνει» την εκτέλεση των N νημάτων στα SPEs.

Σε ένα πολυνηματικό περιβάλλον είναι συχνά βολικό να χρησιμοποιείται ένας μηχανισμός γεγονότων (event mechanism) για την ειδοποίηση σχετικά με συγκεκριμένα γεγονότα που προκαλούνται από τα ασυγχρόνως εκτελούμενα νήματα στα SPEs. Ένα συγκεκριμένο γεγονός χρησιμοποιείται για να υποδείξει ότι ένα SPE context σε ένα νήμα SPE έχει σταματήσει. Η συνάρτηση spe_stop_info_read επιτρέπει στο κύριο νήμα να διαβάσει τις πλήρεις πληροφορίες που επεξηγούν γιατί σταμάτησε το SPE context.

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

όπου spe είναι το προς εκτέλεση SPE context, entry είναι η θέση από την οποία ξεκινάει η εκτέλεση του προγράμματος, runflags είναι διάφορες σημαίες που καθορίζουν τη συμπεριφορά κατά το χρόνο εκτέλεσης, argp είναι ένας δείκτης σε δεδομένα που χρησιμοποιούνται ως παράμετροι του προγράμματος SPE, envp είναι ένας άλλος δείκτης σε δεδομένα που σχετίζονται σε το περιβάλλον εκτέλεσης και stopinfo είναι μία δομή στην οποία αποθηκεύονται μετά τον τερματισμό του SPE προγράμματος πληροφορίες σχετικά με αυτόν τον τερματισμό.

```
int spe_stop_info_read(spe_context_ptr_t spe, spe_stop_info_t * stopinfo)
```

Συναρτήσεις για τα mailboxes των SPEs

Υπάρχει ένα σύνολο συναρτήσεων με τις οποίες επιτρέπεται στο κύριο νήμα να επικοινωνήσει με ένα SPE μέσω του μηχανισμού των mailboxes του. Η ονομασία των mailboxes βασίζεται στην οπτική από την πλευρά του SPE. Έτσι, για παράδειγμα, το out_mbox είναι το mailbox εξερχομένων για αυτό το SPE και η αντίστοιχη συνάρτηση της βιβλιοθήκης _spe_out_mbox_read χρησιμοποιείται για την ανάγνωση του μηνύματος του mailbox από το κύριο νήμα.

int spe_out_mbox_read(spe_context_ptr_t spe, unsigned int * mbox_data, int count), πρόκειται για μία συνάρτηση η οποία διαβάζει το πολύ count μηνύματα από το SPE Outbound Mailbox του SPE που καθορίζεται από τη μεταβλητή spe. Τα δεδομένα που διαβάζονται αποθηκεύονται στη θέση που δείχνει ο δείκτης mbox_data. Η κλήση αυτής της συνάρτησης είναι non-blocking. Επιστρέφεται το πλήθος των μηνυμάτων που διαβάστηκαν (μπορεί να είναι μικρότερο από count).

int spe_out_mbox_status(spe_context_ptr_t spe), πρόκειται για μία συνάρτηση η οποία επιστρέφει το πλήθος των μηνυμάτων τα οποία είναι διαθέσιμα προς ανάγνωση στο SPE Outbound Mailbox του SPE που καθορίζεται από τη μεταβλητή spe.

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

πρόκειται για μία συνάρτηση η οποία γράφει το πολύ count μηνύματα στο SPE Inbound Mailbox του SPE που καθορίζεται από τη μεταβλητή spe. Η κλήση αυτής της συνάρτησης μπορεί να είναι είτε blocking είτε non-blocking, ανάλογα με την τιμή της μεταβλητής behavior. Η τιμή SPE_MBOX_ALL_BLOCKING προκαλεί blocking μέχρι να εγγραφούν όλα τα δεδομένα (count στο πλήθος), η τιμή SPE_MBOX_ANY_BLOCKING προκαλεί blocking μέχρι να εγγραφεί τουλάχιστον ένα μήνυμα, ενώ τέλος με την τιμή SPE_MBOX_ANY_NONBLOCKING εγγράφονται όσα μηνύματα μπορούν να εγγραφούν και η κλήση επιστρέφει αμέσως. Επιστρέφεται το πλήθος των μηνυμάτων που τελικά γράφτηκαν.

int spe_out_mbox_status(spe_context_ptr_t spe), πρόκειται για μία συνάρτηση η οποία επιστρέφει το πλήθος των μηνυμάτων που μπορούν να εγγραφούν στο SPE Inbound Mailbox του SPE που καθορίζεται από τη μεταβλητή spe. Εάν το mailbox είναι γεμάτο επιστρέφεται η τιμή 0, ενώ κάθε άλλη μη μηδενική τιμή δηλώνει τον αριθμό των κενών θέσεων.

Στην κεφαλίδα cbe_mfc.io ορίζονται παρόμοιες συναρτήσεις για τη διαχείριση των mailboxes. Εκτελούνται γρηγορότερα από τις συναρτήσεις της libspe, ενώ ως ορίσματα δέχονται δείκτες προς τις περιοχές problem state των MFCs (από άποψη απεικόνισης στη μνήμη, τα mailboxes ανήκουν σε αυτές τις περιοχές). Οι κλήσεις των συναρτήσεων _spe_out_mbox_read και _spe_in_mbox_write είναι blocking.

unsigned int _spe_out_mbox_read(spe_spu_control_area_t * ps_area)

int _spe_out_mbox_status(spe_spu_control_area_t * ps_area)

void _spe_in_mbox_write(spe_spu_control_area_t * ps_area, unsigned int mbox_data)

int _spe_in_mbox_status(spe_spu_control_area_t * ps_area)

Συναρτήσεις για την ειδοποίηση μέσω σημάτων των SPEs

Υπάρχουν διαθέσιμες κάποιες συναρτήσεις για τη λειτουργία της ειδοποίησης μέσω σημάτων (SPE SPU signal notification). Το PPE μπορεί να στείλει μήνυμα σε ένα SPE χρησιμοποιώντας τη συνάρτηση spe_signal_write.

int spe_signal_write(spe_context_ptr_t spe, unsigned int signal_reg, unsigned int data), πρόκειται για μία συνάρτηση η οποία αποστέλλει το σήμα data στον καταχωρητή σήματος που ορίζεται από την παράμετρο signal_reg (μπορεί να πάρει τις τιμές SPE_SIGNAL_NOTIFY_REG_1 και SPE_SIGNAL_NOTIFY_REG_2) του SPE που καθορίζεται από τη μεταβλητή spe.

Όπως στην περίπτωση των mailboxes, ορίζονται στην κεφαλίδα cbe_mfc.h και οι εξής συναρτήσεις:

unsigned int _spe_sig_notify_1_write(spe_sig_notify_1_area_t *ps_area, unsigned int data)

unsigned int _spe_sig_notify_2_write(spe_sig_notify_2_area_t *ps_area, unsigned int data)

για την αποστολή μηνυμάτων στους αντίστοιχους καταχωρητές. Ως ορίσματα δέχονται δείκτες προς τους MMIO καταχωρητές σημάτων της περιοχής problem sate του ζητούμενου MFC.

Άμεση πρόσβαση των εφαρμογών στα SPEs

Διατίθενται ορισμένες συναρτήσεις μέσω των οποίων οι εφαρμογές μπορούν να έχουν άμεση πρόσβαση στην τοπική μνήμη και τους διάφορους καταχωρητές του problem state ενός SPE. Η συνάρτηση spe_ls_area_get απεικονίζει την τοπική μνήμη ενός SPE στο χώρο διευθύνσεων του κύριου νήματος. Έτσι, η πρόσβαση μετά στην τοπική μνήμη γίνεται με τον ίδιο τρόπο που γίνεται και στην κανονική μνήμη του συστήματος. Δεν συνίσταται για τακτική χρήση καθώς οι μεταφορές DMA από και προς τις τοπικές μνήμες είναι γενικά πιο αποδοτικές. Μία πιο συνηθισμένη χρήση της απεικόνισης της τοπικής μνήμης είναι για την αποστολή της ενεργού διεύθυνσης της τοπικής μνήμης ενός SPE σε ένα άλλο SPE, καθιστώντας έτσι εφικτή τη χρήση ενεργειών DMA για την απ' ευθείας μεταφορά δεδομένων προς και από μία άλλη τοπική μνήμη. Αυτός ο τρόπος μεταφοράς δεδομένων είναι πολύ αποδοτικός καθώς οι μεταφορές DMA πηγαίνουν απ' ευθείας από το ένα SPE στο άλλο, χωρίς να περνάνε από την κύρια μνήμη.

Η συνάρτηση spe_ps_area_get απεικονίζει στο χώρο διευθύνσεων του κύριου νήματος τους καταχωρητές του problem state ενός SPE. Ο δείκτης προς την περιοχή problem state

μπορεί να χρησιμοποιηθεί για την πρόσβαση στις λειτουργίες του problem state χωρίς να χρησιμοποιούνται κλήσεις συστήματος της βιβλιοθήκης. Το problem state περιλαμβάνει λειτουργίες όπως συγχρονισμός πολλών πηγών (multi-source synchronization), proxy DMAs, mailboxes και καταχωρητές ειδοποίησης μέσω σημάτων. Αυτοί οι δείκτες, μαζί με τους δείκτες προς τις τοπικές μνήμες, μπορούν επίσης να χρησιμοποιηθούν για τη διεξαγωγή και τον έλεγχο επικοινωνίας SPE-προς-SPE μέσω mailboxes, DMAs και ειδοποίησης μηνυμάτων.

void * spe_ls_area_get(spe_context_ptr_t spe), πρόκειται για μία συνάρτηση η οποία απεικονίζει την τοπική μνήμη του SPE που καθορίζεται από την παράμετρο spe στο χώρο διευθύνσεων του νήματος και επιστρέφει ένα δείκτη στην αρχή της τοπικής μνήμης.

int spe_ls_size_get(spe_context_ptr_t spe), πρόκειται για μία συνάρτηση η οποία επιστρέφει το μέγεθος της τοπικής μνήμης. Η αρχιτεκτονική της Cell Broadband Engine δεν ορίζει κάποιο σταθερό μέγεθος για την τοπική μνήμη, έτσι οι εφαρμογές που προορίζονται για εκτέλεση σε διάφορες υλοποιήσεις της CBEA πρέπει να καλούν αυτή τη συνάρτηση για να διαβάζουν το ακριβές μέγεθος.

void * spe_ps_area_get(spe_context_ptr_t spe, enum ps_area area), πρόκειται για μία συνάρτηση η οποία απεικονίζει το problem state του SPE που καθορίζεται από την παράμετρο spe στο χώρο διευθύνσεων του κύριου νήματος και επιστρέφει ένα δείκτη στο προς το ζητούμενο από την παράμετρο area καταχωρητή.

Επεκτάσεις C/C++ για τα SPEs

Ένα μεγάλο σύνολο από επεκτάσεις (intrinsics) της C/C++ κάνουν διαθέσιμα με βολικό τρόπο προς τον προγραμματιστή τα υποκείμενα χαρακτηριστικά της αρχιτεκτονικής συνόλου εντολών SPU και του λοιπού hardware. Αυτές οι επεκτάσεις μπορούν να χρησιμοποιηθούν στη θέση κώδικα assembly κατά τον προγραμματισμό σε C/C++.

Οι επεκτάσεις είναι ουσιαστικά εντολές inline assembly υπό τη μορφή κλήσεων συναρτήσεων της γλώσσας C. Παρέχουν στον προγραμματιστή το ρητό έλεγχο των εντολών SIMD του SPE χωρίς την ανάγκη για επ' ευθείας διαχείριση των καταχωρητών. Ένας καλογραμμένος compiler που υποστηρίζει αυτά τα intrinsics θα παράγει αποδοτικό κώδικα για την αρχιτεκτονική του SPE. Οι τεχνικές που χρησιμοποιούνται από τους compilers για την παραγωγή αποδοτικού κώδικα περιλαμβάνουν:

- Χρωματισμός καταχωρητών (register coloring)
- Δρομολόγηση εντολών – βελτιστοποίησης διπλής έκδοσης (instruction scheduling – dual-issue optimization)

- Φορτώσεις και αποθηκεύσεις δεδομένων

- Εφαρμογή blocking, σύντηξης και ξεδιπλώματος βρόχων (loop blocking, fusion, unrolling)

- Σωστή τοποθέτηση των υποδείξεων για άλματα

- Κατασκευή πραγματικών διανυσμάτων

Για παράδειγμα, ένας compiler για την SPU παρέχει το intrinsic t = spu_add(a, b) ως υποκατάστατο της assembly εντολής fa rt, ra, rb. Για αυτό το intrinsic ο compiler θα παράγει την αντίστοιχη εντολή assembly, υποθέτοντας ότι τat, a και b είναι μεταβλητές τύπου vector float. Αυτές οι επεκτάσεις ορίζονται στην κεφαλίδα συστήματος spu_intrinsics.h.

Κατηγορίες επεκτάσεων

Οι επεκτάσεις για την SPU ομαδοποιούνται σε τρεις κατηγορίες:

- Ειδικές (Specific) Επεκτάσεις – Επεκτάσεις που έχουν αντιστοιχία ένα-προς-ένα με μία εντολή assembly. Οι προγραμματιστές σπάνια χρειάζονται αυτήν την κατηγορία επεκτάσεων για την κατασκευή κώδικα inline assembly καθώς το Joint Reference Environment (JSRE) έχει υιοθετήσει inline assembly στο στυλ του gcc.

- Γενικές (Generic) Επεκτάσεις – Επεκτάσεις που έχουν αντιστοιχία με μία ή περισσότερες εντολές assembly αναλόγως του τύπου των ορισμάτων τους.

- Σύνθετες (Composite) Επεκτάσεις – Επεκτάσεις που κατασκευάζονται από μία σειρά ειδικών ή γενικών intrinsics.

Ωστόσο, δεν παρέχονται intrinsics για όλες τις εντολές assembly. Κάποιες εντολές, όπως είναι οι διακλαδώσεις, οι υποδείξεις διακλαδώσεων και η επιστροφή από συνάρτηση, προκύπτουν με φυσικό τρόπο από τη σημασιολογία της γλώσσας C.

Τα ειδικά intrinsics έχουν όνομα που προκύπτει από το μνημονικό της αντίστοιχης εντολής assembly με το πρόθεμα si_. Για παράδειγμα, το intrinsic που υλοποιεί την εντολή assembly stop ονομάζεται si_stop. Ειδικά intrinsics παρέχονται για σχεδόν όλες τις εντολές εκτός από τις διακλαδώσεις, τις υποδείξεις διακλαδώσεων και τις εντολές για διακοπές και επιστροφή από συνάρτηση. Επίσης, για κάθε ειδικό intrinsic υπάρχει και αντίστοιχο γενικό, εκτός από κάποιες εξαιρέσεις.

Το όνομα των γενικών intrinsics προκύπτει από το μνημονικό της αντίστοιχης εντολής assembly με το πρόθεμα spu_. Για παράδειγμα, το intrinsic που υλοποιεί την εντολή assembly stop ονομάζεται spu_stop. Συχνά πολλά από τα γενικά intrinsics υλοποιούνται ως built-ins του compiler. Παρέχονται για όλες τις εντολές, εκτός από τις διακλαδώσεις, τις υποδείξεις διακλαδώσεων, την επιστροφή από διακοπές, την παραγωγή εντολών για αποθήκευση βαθμωτών δεδομένων, τη διαμόρφωση σταθερών, τις εντολές no-op, τις φορτώσεις και αποθηκεύσεις στη μνήμη και τα stop and signal με εξαρτήσεις (stopd).

Τα σύνθετα intrinsics είναι τρία: το spu_mfcdma32(ls, ea, size, tagid, cmd) που αρχικοποιεί DMA μεταφορές προς ή από 32bit ενεργές διευθύνσεις, το spu_mfcdma64(ls, eahi, ealow, size, tagid, cmd) που αρχικοποιεί DMA μεταφορές προς ή από 64bit ενεργές διευθύνσεις και το spu_mfcstat(type) που διαβάζει την κατάσταση (status) για μία ομάδα (tag group).

Αναλυτικά τα intrinsics και η λειτουργία τους περιγράφονται στο έγγραφο της IBM C/C++ Language Extensions for Cell Broadband Engine Architecture.

Προβιβασμός βαθμωτών τύπων δεδομένων σε διανυσματικούς τύπους

Η SPU φορτώνει και αποθηκεύει μία τετραπλή λέξη κάθε φορά. Όταν κάποιες εντολές χρησιμοποιούν ή παράγουν βαθμωτά ορίσματα (συμπεριλαμβανομένων των διευθύνσεων), η τιμή κρατείται στην προτιμώμενη θέση (preferred slot) ενός καταχωρητή SIMD. Οι φορτώσεις και αποθηκεύσεις βαθμωτών δεδομένων (δηλαδή δεδομένων μικρότερων από το μέγεθος ενός διανύσματος) απαιτούν αρκετές εντολές προκειμένου να διαμορφωθούν τα δεδομένα που θα χρησιμοποιηθούν στην αρχιτεκτονική SIMD του SPE. Οι φορτώσεις βαθμωτών δεδομένων πρέπει να περιστραφούν ώστε το βαθμωτό δεδομένα να έρθει στην προτιμώμενη θέση. Οι αποθηκεύσεις βαθμωτών δεδομένων απαιτούν μία ανάγνωση, μία εισαγωγή τιμής στο αναγνωσμένο διάνυσμα και μία εγγραφή. Όλες αυτές οι επιπρόσθετες λειτουργίες διαμόρφωσης μειώνουν την επίδοση.

Οι διανυσματικές πράξεις πάνω σε βαθμωτά δεδομένα δεν είναι αποδοτικές. Για την αύξηση της επίδοσης μπορούν να ακολουθηθούν οι παρακάτω στρατηγικές:

- Προσπάθεια για αλλαγή των βαθμωτών δεδομένων σε διανυσματικά. Εξαλείφοντας τις τρεις επιπρόσθετες εντολές που σχετίζονται με τις φορτώσεις και τις αποθηκεύσεις βαθμωτών δεδομένων μειώνεται τόσο το μέγεθος του κώδικα όσο και ο απαιτούμενος χρόνος εκτέλεσης.

- Ομαδοποίηση βαθμωτών δεδομένων και φόρτωση πολλών βαθμωτών δεδομένων τη φορά χρησιμοποιώντας μία πρόσβαση μεγέθους τετραπλής

λέξης στη μνήμη. Στη συνέχεια χειροκίνητη εξαγωγή ή εισαγωγή αναλόγως των απαιτήσεων. Έτσι μειώνονται κατά πολύ οι φορτώσεις και οι αποθηκεύσεις.

Παρακάτω δίνονται τα intrinsics που παρέχονται για την αποδοτική διαχείριση και μετατροπή βαθμωτών δεδομένων σε διανύσματα και διανυσμάτων σε βαθμωτά δεδομένα.

Εντολή	**Περιγραφή**
d = spu_insert	Εισάγει ένα βαθμωτό δεδομένο στην προσδιοριζόμενη θέση του διανύσματος
d = spu_promote	Προβιβασμός ενός βαθμωτού δεδομένου σε διάνυσμα
d = spu_extract	Εξαγωγή ενός στοιχείου του διανύσματος από την προσδιοριζόμενη θέση

Διαφορές στην υποστήριξη SIMD μεταξύ PPE και SPE

Οι κυριότερες διαφορές στην υποστήριξη SIMD μεταξύ PPE και SPE είναι οι ακόλουθες:

- Το SPE διαθέτει 128 SIMD καταχωρητές, ενώ το PPE 32

- Το αρχείο καταχωρητών στο SPE είναι ενοποιημένο, ενώ στο PPE υπάρχουν ξεχωριστοί καταχωρητές για ακεραίους, για αριθμούς κινητής υποδιαστολής και για διανυσματικούς τύπους δεδομένων

- Στο SPE η καθυστέρηση για τη φόρτωση ενός δεδομένου από την τοπική μνήμη είναι σταθερή, ενώ στο PPE είναι μεταβλητή και εξαρτάται από το αν το δεδομένο βρίσκεται σε κάποια cache ή όχι

- Το σύνολο εντολών του PPE είναι πιο «ορθογώνιο», ενώ αυτό του SPE είναι βελτιστοποιημένο για πράξεις κινητής υποδιαστολής απλής ακρίβειας

- Το εύρος τιμών της κινητής υποδιαστολής απλής ακρίβειας είναι αυτό που ορίζεται από το πρότυπο IEEE 754 – 1985 για το PPE, ενώ στο SPE το εύρος αυτό έχει επεκταθεί

- Το PPE υποστηρίζει του τύπους δεδομένων vector bool char, vector bool short, vector pixel και vector bool int τους οποίους δεν υποστηρίζει το SPE. Από την άλλη, το SPE υποστηρίζει τους τύπους vector unsigned long long, vector signed long long και vector double τους οποίους δεν υποστηρίζει το PPE.

- Το σύνολο εντολών του PPE υποστηρίζει μαθηματικές πράξεις που δεν υποστηρίζονται από το σύνολο εντολών του SPE, όπως άθροισμα στοιχείων ενός διανύσματος, λογαριθμικές και εκθετικές με βάση το 2 και στρογγυλοποιήσεις ceiling και floor. Αντίθετα, το σύνολο εντολών του SPE υποστηρίζει μαθηματικές πράξεις έναντι του PPE, όπως άθροισμα απόλυτων διαφορών, αρίθμηση άσσων σε bytes, ισότητα, nand, συμπλήρωμα or, επέκταση προσήμου, πολλαπλασιασμός και συσσώρευση ακεραίων.

Οδηγίες προς τον compiler

Μαζί με τις επεκτάσεις της C/C++ υπάρχουν και μερικές οδηγίες (directives) προς τον compiler οι οποίες είναι πολύ σημαντικές κατά τον προγραμματισμό εφαρμογών.

Ο qualifier τύπων restrict είναι γνωστός σε πολλές υλοποιήσεις της C/C++ και είναι μέρος της επέκτασης του compiler για την SPU. Όταν η δεσμευμένη λέξη restrict χρησιμοποιείται ως qualifier για ένα δείκτη προσδιορίζει ότι όλες οι προσβάσεις στο αντικείμενο στο οποίο αναφέρεται ο δείκτης γίνονται μόνο μέσω αυτού το δείκτη. Για παράδειγμα, στη δήλωση void * memcpy(void * restrict s1, void * restrict s2, size_t n) προσδιορίζεται ότι οι περιοχές πηγής και προορισμού (εκεί που δείχνουν οι restricted δείκτες s1 και s2 αντίστοιχα) δεν επικαλύπτονται.

Μία άλλη οδηγία είναι η __builtin_expect. Από τη στιγμή που οι λανθασμένες προβλέψεις διακλαδώσεων είναι σχετικά ακριβές, η οδηγία __builtin_expect παρέχει έναν τρόπο για τον προγραμματιστή να κατευθύνει την πρόβλεψη. Για παράδειγμα, η δήλωση int __builtin_expect(int exp, int value) επιστρέφει το αποτέλεσμα της αποτίμησης του exp και ο προγραμματιστής περιμένει το exp να ισούται με value. Το value μπορεί να είναι μία σταθερά για πρόβλεψη στο χρόνο μεταγλώττισης ή μία μεταβλητή για πρόβλεψη στο χρόνο εκτέλεσης.

Δύο ακόμα οδηγίες είναι η ιδιότητα aligned και η οδηγία _align_hint. Η ιδιότητα aligned χρησιμοποιείται για να εξασφαλίσει την κατάλληλη ευθυγράμμιση, ώστε να μεταφέρονται αποδοτικά τα δεδομένα μέσω DMA. Η σύνταξή της είναι η ίδια όπως σε πολλές υλοποιήσεις του gcc: float factor __attribute__((aligned(16))) – ευθυγραμμίζει τη μεταβλητή factor σε μία τετραπλή λέξη (16 bytes).

Η οδηγία _align_hint βοηθάει τους compilers να κάνουν αυτόματα vectorization. Παρόλο που μοιάζει περισσότερο με intrinsic, στην πραγματικότητα περιγράφεται καλύτερα ως οδηγία προς τον compiler, αφού δεν παράγεται κάποιος κώδικας από τη χρήση της. Για παράδειγμα, το _align_hint(ptr, base, offset) πληροφορεί τον compiler ότι ο δείκτης ptr δείχνει σε δεδομένα που είναι ευθυγραμμισμένα με βάση το base και απέχουν από τη βασική ευθυγράμμιση κατά offset bytes. Η βάση πρέπει να είναι δύναμη του 2. Η τιμή 0 δηλώνει ότι ο δείκτης δεν έχει κάποια γνωστή ευθυγράμμιση. Το offset πρέπει φυσικά να είναι μικρότερο

από τη βάση ή 0. Η οδηγία _align_hint δεν πρέπει να χρησιμοποιείται με δείκτες που δεν έχουν φυσική ευθυγράμμιση.

Εντολές MFC

Ο MFC υποστηρίζει ένα σύνολο εντολών MFC. Αυτές οι εντολές παρέχουν τον κύριο μηχανισμό που επιτρέπει στον κώδικα που εκτελείται σε μία SPU να έχει πρόσβαση στον κύριο αποθηκευτικό χώρο και να διατηρεί συγχρονισμό με άλλους επεξεργαστές και συσκευές στο σύστημα. Οι εντολές MFC μπορούν να εκδοθούν είτε από κώδικα που τρέχει στην αντίστοιχη SPU του MFC ή από κώδικα που τρέχει στο PPE ή άλλη συσκευή:

- Ο κώδικας που τρέχει στην SPU εκδίδει μία εντολή MFC εκτελώντας μία σειρά από εγγραφές χρησιμοποιώντας τις εντολές καναλιών.

- Ο κώδικας που τρέχει στο PPE ή σε άλλες συσκευές εκδίδει μία εντολή MFC εκτελώντας μία σειρά αποθηκεύσεων και φορτώσεων στους MMIO καταχωρητές του εν λόγω MFC.

Οι εντολές μπαίνουν σε μια από τις δύο ανεξάρτητες ουρές εντολών του MFC:

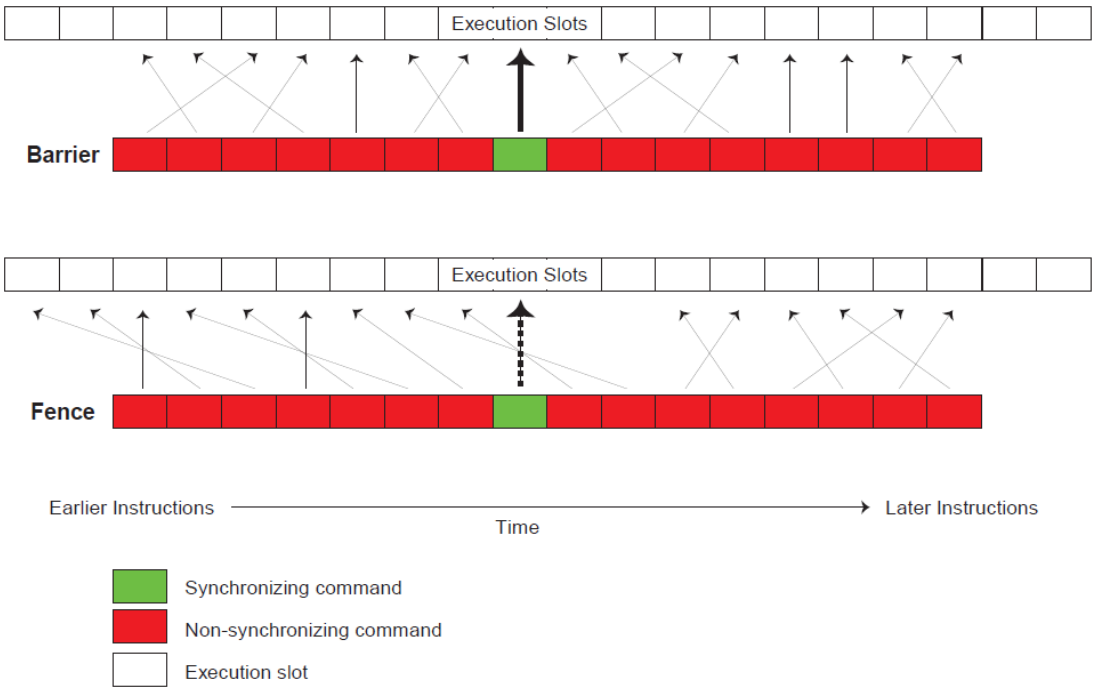
- Ουρά Εντολών MFC SPU (MFC SPU Command Queue) – Για εντολές που ξεκινάνε μέσω εγγραφών στα κανάλια από την αντίστοιχη SPU

- Ουρά Εντολών MFC Proxy (MFC Proxy Command Queue) – Για εντολές που έχουν εκδοθεί με MMIO τρόπο από το PPE ή άλλη συσκευή

Οι εντολές MFC που μεταφέρουν δεδομένα αναφέρονται ως εντολές DMA. Η κατεύθυνση της μεταφοράς δεδομένων για τις εντολές DMA του MFC αναφέρεται πάντα από την πλευρά του SPE. Έτσι, εντολές που μεταφέρουν δεδομένα προς ένα SPE (από την κύρια μνήμη στην τοπική μνήμη) θεωρούνται εντολές τύπου get και μεταφορές δεδομένων από ένα SPE (από την τοπική μνήμη στην κύρια μνήμη) θεωρούνται εντολές τύπου put. Οι εντολές get και put μπορούν προαιρετικά να ακολουθούνται από τα επιθέματα που φαίνονται στον παρακάτω πίνακα. Συνολικά, οι εντολές που μπορούν να κληθούν είναι οι εξής: put, puts, putf, putb, putfs, putbs, putl, putlf, putlb, get, gets, getf, getb, getfs, getbs, getl, getlf, getlb. Τα επιθέματα επεκτείνουν ή προσδιορίζουν καλύτερα τη λειτουργία μίας εντολής. Για παράδειγμα, μία εντολή put μεταφέρει δεδομένα από την τοπική μνήμη στον ενεργό χώρο διευθύνσεων. Η εντολή puts μεταφέρει δεδομένα από την τοπική μνήμη στον ενεργό χώρο διευθύνσεων και ξεκινάει την SPU μετά την ολοκλήρωση της μεταφοράς DMA. Εντολές με επίθεμα s μπορούν να εκδοθούν μόνο από το PPE. Οι εντολές με όλα τα υπόλοιπα επιθέματα/συνδυασμους επιθεμάτων μπορούν να εκδοθούν είτε από το PPE είτε από το SPE.

Επίθεμα	Περιγραφή
s	Ξεκινάει την εκτέλεση στην SPU από την εντολή που υποδεικνύεται από τον καταχωρητή-μετρητή προγράμματος μετά την ολοκλήρωση της μεταφοράς των δεδομένων στην ή από την τοπική μνήμη.
f	Φράχτης εξαρτώμενος από την ετικέτα (tag specific fence). Οι εντολές με κάποιο φράχτη ταξινομούνται τοπικά μετά από όλες τις εντολές που έχουν εκδοθεί προηγουμένως και με τις οποίες ανήκουν στο ίδιο tag group και την ίδια ουρά εντολών.
b	Φράγμα εξαρτώμενο από την ετικέτα (tag specific barrier). Οι εντολές με κάποιο φράγμα ταξινομούνται τοπικά μετά από όλες τις εντολές που έχουν εκδοθεί προηγουμένως και πριν από όλες τις εντολές που θα εκδοθούν μετέπειτα και με τις οποίες ανήκουν στο ίδιο tag group και την ίδια ουρά εντολών.
l	Εντολή λίστας. Εκτελεί μία λίστα από στοιχεία μεταφορών DMA που βρίσκεται στην τοπική μνήμη. Το μέγιστο πλήθος αυτών των στοιχείων είναι 2.048 και κάθε στοιχείο μπορεί να αναφέρεται σε μία μεταφορά μέχρι και 16KB.

Ο MFC μπορεί να αναδιατάξει τις εντολές που βρίσκονται μέσα στην ουρά εντολών. Δηλαδή, οι μεταφορές δεν εκτελούνται απαραίτητα στη σειρά με την οποία ορίζει το πρόγραμμα. Για τον έλεγχο αυτής της σειράς υπάρχουν οι μορφές fence και barrier των εντολών put και get. Και τις δύο αυτές μορφές η εντολή τοποθετείται στην ουρά μετά από όλες τις εντολές που υπάρχουν νωρίτερα στο πρόγραμμα. Η διαφορά μεταξύ fence και barrier περιγράφεται σχηματικά παρακάτω. Ουσιαστικά, στην περίπτωση μίας εντολής της μορφής fence, εντολές που εκδίδονται μετά από αυτή μπορούν κατά την αναδιάταξη να μπουν πριν από αυτή. Στην περίπτωση της barrier μορφής αυτό δεν συμβαίνει. Οι μορφές fence και barrier επηρεάζουν μόνο τις εντολές που βρίσκονται στην ίδια ομάδα (δηλαδή είναι στο ίδιο tag-group). Δεν επηρεάζεται η σειρά μεταξύ εντολών που ανήκουν σε διαφορετικά tag-groups.



Υπάρχουν άλλες δύο κατηγορίες εντολών MFC. Η πρώτη περιλαμβάνει τις Εντολές Συγχρονισμού MFC (MFC Synchronization Commands). Περιληπτικά, οι εντολές αυτής της κατηγορίας είναι οι εξής:

- barrier, mfceieio και mfcsync που αποτελούν παραλλαγές της ίδιας λειτουργίας και χρησιμοποιούνται για τη διάταξη των εντολών στις ουρές του MFC ενός SPE ή μεταξύ των ουρών όλων των MFCs που υπάρχουν στο σύστημα. Η αναδιάταξη επηρεάζει όλες τις εντολές, ανεξαρτήτως tag group.

- sndsig, sndsigb, sndsigf που χρησιμοποιούνται για την αποστολή σημάτων (με ή χωρίς fence ή barrier), εγγράφοντας δεδομένα στους Signal Notification Registers κάποιου άλλου SPE ή συσκευής εισόδου/εξόδου.

Όλες οι εντολές αυτής της κατηγορίας μπορούν να εκδοθούν τόσο από το PPE όσο και από το SPE. Η δεύτερη κατηγορία εντολών περιλαμβάνει τις Ατομικές Εντολές του MFC (MFC Atomic Commands). Οι εντολές αυτές είναι οι εξής: getllar, putllc, putlluc και putqlluc. Χρησιμοποιούνται για την ατομική (αποκλειστική) πρόσβαση σε θέσεις μνήμης που χρησιμοποιούνται από κοινού από πολλές επεξεργαστικές μονάδες και συσκευές και μπορούν να εκδοθούν μόνο από τα SPEs.

Ομάδες ετικετών (tag groups) εντολών DMA

Όλες οι εντολές DMA εκτός των getllar, putllc και putlluc μπορούν να χαρακτηρισθούν με μία 5bit Ετικέτα-ID Ομάδας (Tag Group ID). Τοποθετώντας μία εντολή ή μία ομάδα εντολών DMA σε διαφορετικά tag groups, μπορεί να προσδιοριστεί η κατάσταση ολόκληρου του tag group εντός μίας ουράς εντολών (είτε της MFC SPU Command Queue είτε της MFC Proxy Command Queue). Το λογισμικό μπορεί να χρησιμοποιήσει αυτήν την ετικέτα για να ελέγξει ή να περιμένει για την ολοκλήρωση όλων των εντολών που ανήκουν στο ίδιο tag group σε μια ουρά. Η χρήση των ετικετών είναι προαιρετική αλλά μπορεί να είναι χρήσιμη όταν χρησιμοποιούνται fences και barriers για τον έλεγχο της σειράς των εντολών MFC μέσα σε μία ουρά εντολών.

Μακροεντολές εισόδου/εξόδου του MFC

Οι επεκτάσεις της C/C++ για την αρχιτεκτονική της Cell Broadband Engine ορίζουν επίσης ένα σύνολο μακροεντολών οι οποίες βοηθούν στην πρόσβαση σε λειτουργίες της SPU και του MFC που παρέχονται μέσω της διεπαφής των καναλιών. Αυτές οι μακροεντολές ορίζονται στην κεφαλίδα spu_mfcio.h και μπορούν να υλοποιούνται είτε ως μακροεντολές είτε ως built-in συναρτήσεις στον compiler. Για παράδειγμα, οι μακροεντολές που υλοποιούν τις λειτουργίες put, putf και putb είναι αντίστοιχα οι mfc_put(ls, ea, size, tag, tid, rid), mfc_putf(ls, ea, size, tag, tid, rid) και mfc_putb(ls, ea, size, tag, tid, rid). Αναλυτικά οι μακροεντολές αυτές περιγράφονται στο έγγραφο της IBM C/C++ Language Extensions for Cell Broadband Engine Architecture.

Μεταφορές DMA

Οι μεταφορές DMA μετακινούν δεδομένα μεταξύ της τοπικής μνήμης και του κύριου αποθηκευτικού χώρου. Η διεύθυνση της τοπικής μνήμης δίνεται στην εντολή DMA με το όρισμα local store address (LSA). Το μέγεθος μίας μεταφοράς DMA περιορίζεται στα 16KB. Η τοπική μνήμη προσπελάζεται σειριακά με ελάχιστο βήμα τη μία τετραπλή λέξη. Το λογισμικό σε ένα SPE έχει πρόσβαση στις λειτουργίες μεταφορών DMA του MFC του μέσω των καναλιών. Για να μπει στην ουρά μία μεταφορά DMA το λογισμικό πρέπει να γράψει με την εντολή wrch τους καταχωρητές των καναλιών των παραμέτρων των εντολών (MFC Command Parameter Registers) με την εξής σειρά:

1. Εγγραφή της διεύθυνσης της τοπικής μνήμης στο κανάλι MFC_LSA

2. Εγγραφή του άνω ημίσεως της ενεργού διεύθυνσης (EA-high – EAH) στο κανάλι MFC_EAH

3. Εγγραφή του κάτω ημίσεως της ενεργού διεύθυνσης (EA-low – EAL) στο κανάλι MFC_EAL

4. Εγγραφή του μεγέθους της μεταφοράς στο κανάλι MFC_Size

5. Εγγραφή του αναγνωριστικού ετικέτας (tag ID) στο κανάλι MFC_TagID

6. Εγγραφή του αναγνωριστικού κλάσης (class ID) και του opcode της εντολής στο κανάλι MFC_Cmd

Αντί για όλη την παραπάνω διαδικασία μπορεί να χρησιμοποιηθεί κάποια κατάλληλη μακροεντολή. Για παράδειγμα, μία μεταφορά DMA από την τοπική μνήμη προς τον κύριο αποθηκευτικό χώρο αρχικοποιείται χρησιμοποιώντας τη μακροεντολή mfc_put(lsa, ea, size, tag, tid, rid), ενώ ο έλεγχος για την ολοκλήρωσή της γίνεται με την ακολουθία εντολών mfc_write_tag_mask(1 << tag) και mfc_read_tag_status_all(). Γενικότερα, η μακροεντολή mfc_write_tag_mask(mask) δέχεται ως όρισμα μία μάσκα η οποία δημιουργείται τοποθετώντας 1 στα bits εκείνα που αντιστοιχούν στις ομάδες ετικετών των οποίων αναμένεται η ολοκλήρωση. Η κλήση της mfc_read_tag_status_all() μπλοκάρει μέχρι να ολοκληρωθούν όλες οι μεταφορές του tag group ή των tag groups που έχουν καθοριστεί στη μάσκα.

Μεταφορές λίστας DMA

Μία λίστα DMA είναι μία ακολουθία από στοιχεία μεταφοράς (transfer elements) η οποία, μαζί με την εντολή που αρχικοποιεί τη λίστα DMA, προσδιορίζουν μία ακολουθία μεταφορών DMA μεταξύ μίας και μόνο περιοχής της τοπικής μνήμης και πιθανόν μη συνεχόμενες περιοχές του κύριου αποθηκευτικού χώρου. Τέτοιες λίστες DMA αποθηκεύονται στην τοπική μνήμη ενός SPE και οι ακολουθία των μεταφορών αρχικοποιείται από μία εντολή DMA-list όπως είναι οι getl και οι putl. Οι εντολές λίστας DMA μπορούν να εκδοθούν μόνο από προγράμματα που τρέχουν στο SPE, όμως το PPE ή άλλες συσκευές μπορούν να δημιουργήσουν και να αποθηκεύσουν τις λίστες στην τοπική μνήμη ενός SPE. Οι λίστες DMA μπορούν να χρησιμοποιηθούν για να υλοποιηθούν λειτουργίες scatter-gather μεταξύ του κύριου αποθηκευτικού χώρου και της τοπικής μνήμης.

Κατασκευή της λίστας DMA

Κάθε στοιχείο μεταφοράς (transfer element) στη λίστα DMA περιέχει το μέγεθος της μεταφοράς, το κάτω ήμισυ της ενεργού διευθύνσεως και ένα bit stall-and-notify το οποίο μπορεί να χρησιμοποιηθεί για να διακόψει την εκτέλεση των μεταφορών της λίστας αφότου εκτελεσθεί μία μεταφορά της οποίας το transfer element είχε το bit stall-and-notify ίσο με 1. Κάθε μεταφορά DMA που ορίζεται σε μια λίστα μπορεί να μεταφέρει μέχρι και 16KB δεδομένων και η ίδια η λίστα μπορεί να έχει μέχρι και 2.048 στοιχεία μεταφοράς.

Το λογισμικό δημιουργεί τη λίστα και την αποθηκεύει στην τοπική μνήμη. Οι λίστες πρέπει να αποθηκεύονται στη μνήμη ευθυγραμμισμένες στα 8 bytes. Η μορφή ενός στοιχείου μεταφοράς είναι {μέγεθος μεταφοράς LTS, ενεργός διεύθυνση EAL}.

- Η πρώτη λέξη (το list transfer size – LTS) είναι το μέγεθος της μεταφοράς και το περισσότερο σημαντικό bit της είναι το bit stall-and-notify.

- Η δεύτερη λέξη (effective address low – EAL) είναι τα 32 χαμηλότερα bits της ενεργού διευθύνσεως.

Τα στοιχεία της λίστας υπόκεινται σε επεξεργασία σειριακά, με την ίδια σειρά που έχουν αποθηκευτεί. Εάν η σημαία stall-and-notify είναι ενεργοποιημένη για κάποιο στοιχείο μεταφοράς, ο MFC θα σταματήσει να επεξεργάζεται τη λίστα DMA μετά την εκτέλεση της μεταφοράς που αντιστοιχεί σε αυτό το στοιχείο μέχρι το πρόγραμμα του SPE να καθαρίσει το γεγονός (event) DMA Command Stall-And-Notify από το κανάλι SPU Read Event Status. Αυτό δίνει στα προγράμματα την ευκαιρία να τροποποιήσουν τα επόμενα στοιχεία μεταφοράς προτού αυτά περάσουν για επεξεργασία από τον MFC.

Αρχικοποίηση των μεταφορών που ορίζονται στη λίστα DMA

Αφότου η λίστα αποθηκευθεί στην τοπική μνήμη αρχικοποιείται η εκτέλεση της λίστας μέσω μίας εντολής DMA-list, όπως είναι οι getl και putl, από το SPE στου οποίου την τοπική μνήμη περιέχεται η λίστα.

Οι εντολές DMA-list, όπως και οι εντολές για τις απλές μεταφορές DMA, απαιτούν κάποιες παραμέτρους οι οποίες πρέπει να εγγραφούν στα κανάλια MFC Command Parameter. Η εγγραφή αυτή γίνεται με τον ίδιο τρόπο όπως στις απλές μεταφορές, όμως οι εντολές DMA-list απαιτούν δύο διαφορετικούς τύπους παραμέτρων:

- MFC_EAL: Αυτή η παράμετρος πρέπει να περιέχει τη διεύθυνση της τοπικής μνήμης στην οποία αρχίζει η λίστα και όχι με την EAL (η EAL περιέχεται στο κάθε στοιχείο μεταφοράς).

- MFC_Size: Αυτή η παράμετρος πρέπει να περιέχει το μέγεθος της λίστας και όχι το μέγεθος της μεταφοράς (το μέγεθος της μεταφοράς περιέχεται στο κάθε στοιχείο μεταφοράς). Το μέγεθος της λίστας ισούται με το πλήθος των στοιχείων μεταφοράς πολλαπλασιαζόμενο με το μέγεθος της δομής του transfer-element (8 bytes).

Η διεύθυνση της τοπικής μνήμης (LSA) και το άνω ήμισυ της ενεργού διευθύνσεως (EAH) προσδιορίζονται μόνο μία φορά, στην εντολή DMA-list που αρχικοποιεί τις μεταφορές. Η

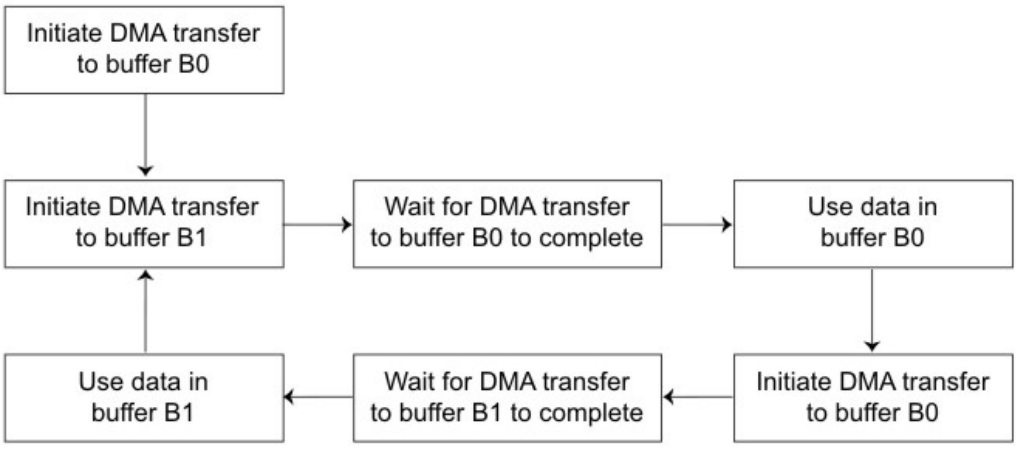
LSA μετά αυξάνεται εσωτερικά με βάση το μέγεθος των δεδομένων που μεταφέρθηκαν από κάθε στοιχείο μεταφοράς. Η EAL για κάθε στοιχείο βρίσκεται στην 4GB περιοχή μνήμης που ορίζεται από την παράμετρο EAH.

Η τεχνική της χρήσης διπλών ενταμιευτών (Double Buffering)

Τα προγράμματα SPE χρησιμοποιούν μεταφορές DMA για να μετακινήσουν δεδομένα μεταξύ του κύριου αποθηκευτικού χώρου και της τοπικής μνήμης του SPE. Έστω ένα πρόγραμμα SPE το οποίο απαιτεί μεγάλο πλήθος δεδομένων από την κύρια μνήμη. Ένα απλό σχήμα για αυτή τη μεταφορά δεδομένων είναι το εξής:

1. Αρχικοποίηση μίας μεταφοράς DMA από την κύρια μνήμη στον buffer B της τοπικής μνήμης
2. Αναμονή για την ολοκλήρωση της μεταφοράς
3. Χρήση των δεδομένων στον buffer B
4. Επανάληψη των βημάτων 1-3

Αυτός ο τρόπος σπαταλάει μεγάλο μέρος του χρόνου περιμένοντας να ολοκληρωθούν οι μεταφορές DMA. Η όλη διαδικασία μπορεί να επιταχυνθεί σημαντικά χρησιμοποιώντας δύο buffers, τους B_0 και B_1, και επικαλύπτοντας τον υπολογισμό στα δεδομένα του ενός buffer και τη μεταφορά δεδομένων στον άλλον. Αυτή η τεχνική καλείται Τεχνική Διπλών Ενταμιευτών (Double Buffering) και περιγράφεται σχηματικά στο παρακάτω σχήμα. Η χρήση διπλών ενταμιευτών είναι μία μορφή της χρήσης πολλών ενταμιευτών (multi-buffering), η οποία είναι η μέθοδος κατά την οποία χρησιμοποιούνται πολλαπλοί ενταμιευτές σε κυκλική σειρά για να επικαλυφθούν οι υπολογισμοί και οι μεταφορές δεδομένων.



Για να είναι αποδοτική η χρήση του double buffering πρέπει να ακολουθούνται οι εξής κανόνες για τις μεταφορές DMA του SPE:

- Χρήση πολλαπλών buffers στην τοπική μνήμη
- Χρήση μοναδικών tag IDs των DMA, ένα για κάθε buffer ή λογική ομάδα από buffers
- Χρήση των fenced μορφών των εντολών DMA για την ταξινόμηση των μεταφορών DMA εντός μίας ομάδας (tag group)
- Χρήση της εντολής συγχρονισμού barrier του MFC για την ταξινόμηση των μεταφορών του ελεγκτή DMA ενός MFC

Ο σκοπός του double buffering είναι η μεγιστοποίηση του χρόνου της υπολογιστικής φάσης ενός προγράμματος και η ελαχιστοποίηση του χρόνο που καταναλώνεται περιμένοντας να ολοκληρωθούν οι μεταφορές DMA. Εάν τ_t είναι ο χρόνος που απαιτείται για τη μεταφορά ενός buffer B και τ_c ο χρόνος που απαιτείται για να πραγματοποιηθούν οι υπολογισμοί τα δεδομένα αυτού του buffer. Γενικά, όσο μεγαλύτερος είναι ο λόγος τ_t/τ_c για μία εφαρμογή, τόσο περισσότερο θα επωφεληθεί ως προς την απόδοσή της με τη χρήση του σχήματος του double buffering.

Γεγονότα του SPE (SPE Events)

Το γεγονός, από τη σκοπιά του λογισμικού του SPE, είναι γενικά μία ασύγχρονη αλλαγή στην κατάσταση ενός πόρου ή μίας μονάδας στον επεξεργαστή Cell Broadband Engine που είναι εξωτερική για αυτό το SPE. Κάθε γεγονός που μπορεί να έχει κάποιο ενδιαφέρον για ένα SPE έχει ένα αντίστοιχο bit στο κανάλι Event Status του SPE. Για παράδειγμα, η άφιξη ενός μηνύματος mailbox από ένα άλλο επεξεργαστικό στοιχείο θέτει το mailbox bit στο Event Status και ίσως ξεκινήσει μία διακοπή στο SPE. Το λογισμικό του SPE μπορεί να επιλέξει να παρακολουθεί και να απαντάει σε γεγονότα σε σύγχρονο (κάνοντας polling ή stalling) ή ασύγχρονο (ενεργοποιώντας τη διακοπή γεγονότων) τρόπο.

Κάθε SPE υποστηρίζει μία λειτουργία γεγονότων που έχει τη δυνατότητα να προσθαφαιρεί γεγονότα από μια μάσκα, να περιμένει την εκδήλωση γεγονότων, να κάνει polling για γεγονότα και να προκαλεί διακοπές στο SPE όταν συμβεί ένα συγκεκριμένο γεγονός. Εάν οι διακοπές είναι ενεργοποιημένες και συμβεί ένα γεγονός το οποίο δεν είναι στη μάσκα τότε καλείται ο διαχειριστής διακοπών του SPE. Οι διακοπές αυτές ονομάζονται διακοπές του SPE και δεν έχουν σχέση με τις διακοπές του PPE.

Η κύρια διεπαφή μεταξύ του λογισμικού του SPE και του hardware που παρακολουθεί τα γεγονότα παρέχεται μέσω τεσσάρων καναλιών τα οποία ελέγχουν τη διαχείριση των γεγονότων και αναφέρουν την κατάσταση ενός γεγονότος. Το λογισμικό του SPE αντιμετωπίζει τα γεγονότα χρησιμοποιώντας τέσσερεις βασικές ενέργειες που αφορούν σε αυτά τα τρία κανάλια:

- Αρχικοποίηση της διαχείρισης των γεγονότων: εγγραφή στο κανάλι Event Mask, κανάλι 1
- Αναγνώριση γεγονότων που έχουν συμβεί: ανάγνωση του καναλιού Event Status, κανάλι 0
- Καθαρισμός γεγονότων: εγγραφή στο κανάλι Event Acknowledge, κανάλι 2
- Εξυπηρέτηση γεγονότων: εκτέλεση κώδικα ανάλογα με την εφαρμογή

Τα κανάλια τα οποία σχετίζονται με τα γεγονότα είναι τα εξής:

- SPU Read Event Status (SPU_RdEventStat) – κανάλι 0
- SPU Write Event Mask (SPU_WrEventMask) – κανάλι 1
- SPU Write Event Acknowledgment (SPU_WrEventAck) – κανάλι 2
- SPU Read Event Mask (SPU_RdEventMask) – κανάλι 11

Επιπρόσθετα, υπάρχει ένας εσωτερικός καταχωρητής, ο Pending Event Register, ο οποίος είναι κρυμμένος από το λογισμικό του SPE αλά είναι ορατός από το προνομιούχο λογισμικό που εκτελείται στο PPE.

Τυπικά, ένα πρόγραμμα SPE ενδιαφέρεται μόνο για μερικά από τα πιθανά γεγονότα SPE. Προκειμένου να ενεργοποιήσει μόνο εκείνα τα γεγονότα τα οποία σχετίζονται με τη λειτουργία του προγράμματος, το λογισμικό του SPE αρχικοποιεί μία μάσκα στην οποία θέτει εκείνα τα bits που αντιστοιχούν στα σχετικά γεγονότα και την οποία γράφει στο κανάλι SPU Write Event Mask. Μετά την αναγνώριση ενός γεγονότος από το λογισμικό, γράφεται μία τιμή γνωστοποίησης (acknowledgement value) στο κανάλι SPU Write Event Acknowledgment, δηλώνοντας ότι έχει ολοκληρωθεί ο χειρισμός κάποιου γεγονότος, έτσι ώστε αυτό να μπορεί να αναγνωρισθεί πάλι όταν ξανασυμβεί. Τέλος, το κανάλι SPU Read Event Status παρέχει πληροφορίες σχετικά με όλα τα ενεργοποιημένα από τη μάσκα γεγονότα.

Το λογισμικό του SPE μπορεί να επιλέξει να αναγνωρίζει γεγονότα είτε με σύγχρονο είτε με ασύγχρονο τρόπο. Για σύγχρονη διαχείριση γεγονότων το λογισμικό μπορεί να κάνει είτε polling για επικείμενα γεγονότα (για παράδειγμα να ελέγχει ένα μετρητή καναλιού σε ένα βρόχο) ή να μπλοκάρει όταν δεν υπάρχουν ενεργοποιημένα γεγονότα που να περιμένουν εξυπηρέτηση (για παράδειγμα, να αναστέλλεται η λειτουργία της SPU ως συνέπεια της προσπάθειας ανάγνωσης από άδειο κανάλι). Για ασύγχρονο χειρισμό γεγονότων το λογισμικό πρέπει να ενεργοποιήσει το μηχανισμό των διακοπών και να παράσχει κώδικα διαχείρισής των (interrupt handler). Η SPU υποστηρίζει μία μοναδική συνάρτηση χειρισμού διακοπών με προκαθορισμένο σημείο εισόδου, το οποίο αντιστοιχεί στη διεύθυνση 0 της τοπικής μνήμης. Όταν προκαλείται μία διακοπή ως αποτέλεσμα κάποιου γεγονότος, ο έλεγχος περνάει σε αυτήν τη συνάρτηση και απενεργοποιούνται οι διακοπές, ενώ ταυτόχρονα αποθηκεύεται στον κατάλληλο καταχωρητή η διεύθυνση της επόμενης εντολής του προγράμματος SPE ώστε να είναι δυνατή η επιστροφή από τη συνάρτηση χειρισμού των διακοπών και η συνέχιση της εκτέλεσης του προγράμματος. Για να είναι δυνατός ο χειρισμός πολλών διαφορετικών γεγονότων, η λειτουργία της συνάρτησης χειρισμού διακοπών χωρίζεται σε δύο μέρη: στο Διαχειριστή Γεγονότων Πρώτου Επιπέδου (First Level Interrupt Handler – FLIH) ο οποίος είναι κοινός για όλα τα γεγονότα και στο Διαχειριστή Γεγονότων Δεύτερου Επιπέδου (Second Level Interrupt Handler – SLIH) ο οποίος είναι διαφορετικός για κάθε γεγονός και είναι αυτός που υλοποιεί τον τρόπο με τον οποίον ανταποκρίνεται το SPE σε ένα συγκεκριμένο γεγονός. Ο FLIH, δηλαδή, ελέγχει ποια συγκεκριμένα γεγονότα έχουν συμβεί και αναλόγως καλεί τον κατάλληλο SLIH.

Μειώνοντας τον αντίκτυπο των διακλαδώσεων

Το hardware της SPU υποθέτει γραμμική ροή εντολών και είναι βελτιστοποιημένο ώστε να μην παράγει καμία καθυστέρηση (stall) όταν τηρείται η γραμμική ροή. Μία εντολή διακλάδωσης μπορεί να χαλάσει την υποτιθέμενη γραμμική ροή. Οι διακλαδώσεις που προβλέπονται σωστά εκτελούνται σε έναν κύκλο, όμως οι διακλαδώσεις (είτε υπό συνθήκη είτε χωρίς συνθήκη) που δεν προβλέπονται σωστά επιφέρουν ποινή περίπου 18-19 κύκλων. Λαμβάνοντας υπ' όψιν την τυπική καθυστέρηση των εντολών SPU, που ανέρχεται στους 2-7 κύκλους, οι διακλαδώσεις που δεν προβλέπονται σωστά μπορούν να υποβαθμίσουν σοβαρά την επίδοση ενός προγράμματος. Οι διακλαδώσεις δημιουργούν επίσης φράγματα στο scheduling των εντολών, μειώνοντας τις ευκαιρίες για διπλή έκδοση και κάλυψη των καθυστερήσεων λόγω εξαρτήσεων (dependency stalls).

Ο πιο αποδοτικός τρόπος μείωσης του αντίκτυπου των διακλαδώσεων είναι η εξαφάνισή τους, χρησιμοποιώντας τρεις βασικές μεθόδους – inlining συναρτήσεων, ξεδίπλωμα (unrolling) βρόχων και δηλώσεις (predication). Το επόμενο αποτελεσματικό βήμα προς αυτήν την κατεύθυνση είναι η χρήση των εντολών υπόδειξης διακλαδώσεων.

Το inlining συναρτήσεων (function inlining) και το ξεδίπλωμα βρόχων (loop unrolling) είναι δύο τεχνικές που χρησιμοποιούνται συχνά για την αύξηση του μεγέθους των βασικών block (ακολουθιών από διαδοχικές εντολές χωρίς διακλαδώσεις), ενέργεια που έχει ως αποτέλεσμα αυξημένες ευκαιρίες για αποδοτικό scheduling εντολών. Το inlining συναρτήσεων εξαλείφει τις δύο διακλαδώσεις που σχετίζονται με το linking των συναρτήσεων, δηλαδή μία διακλάδωση κατά την κλήση της συνάρτησης και μία διακλάδωση κατά την επιστροφή από την κλήση της συνάρτησης. Το ξεδίπλωμα βρόχων εξαλείφει τις διακλαδώσεις μειώνοντας το πλήθος των επαναλήψεων του βρόχου. Το ξεδίπλωμα μπορεί να γίνει είτε χειροκίνητα είτε υποδεικνύοντάς την στον compiler είτε αυτόματα από τον compiler. Γενικά, οι διακλαδώσεις που σχετίζονται με τους βρόχους δεν έχουν τόσο μεγάλο αντίκτυπο καθώς είναι ως επί το πλείστον προβλέψιμες. Παρ' όλα αυτά, εάν κάποιος βρόχος μπορεί να αναδιπλωθεί πλήρως τότε όλες οι διακλαδώσεις εξαλείφονται – συμπεριλαμβανομένης και της τελευταίας μη-προβλεπόμενης διακλάδωσης.

Κεφάλαιο 4

ΠΑΡΑΛΛΗΛΑ ΠΡΟΓΡΑΜΜΑΤΑ ΣΤΗ CELL BROADBAND ENGINE

Η διαφορετικότητα της CBE

Η αρχιτεκτονική και η φιλοσοφία της Cell Broadband Engine διαφέρει σημαντικά από άλλες συμβατικές αρχιτεκτονικές, όπως για παράδειγμα η πλατφόρμα x86. Συγκεκριμένα, η Cell Broadband Engine παρέχει ένα σύνολο εντολών βελτιστοποιημένων για πράξεις SIMD ή δυνατότητες όπως οι ασύγχρονες μεταφορές DMA, χαρακτηριστικά τα οποία είτε δεν περιλαμβάνονται σε άλλες συμβατικές αρχιτεκτονικές είτε η υλοποίησή τους δεν είναι τόσο αποδοτική. Λαμβάνοντας υπ' όψιν τέτοιες παραμέτρους γίνεται αντιληπτό ότι η Cell Broadband Engine είναι ιδιαίτερα αποδοτική σε συγκεκριμένες κατηγορίες προβλημάτων, ενώ υστερεί σε κάποιες άλλες, και ότι συγκεκριμένες σχεδιαστικές αποφάσεις για αυτά τα προβλήματα μπορούν να αξιοποιήσουν καλύτερα το υποκείμενο hardware, εκμεταλλευόμενες τις ιδιαιτερότητές του. Την ίδια στιγμή, άλλες τεχνικές σχεδίασης και προγραμματισμού, η εφαρμογή των οποίων είναι κοινή πρακτική για μια συμβατική αρχιτεκτονική, μπορούν να έχουν από μηδενική έως και αρνητική επίδραση στην επίδοση ενός συστήματος Cell Broadband Engine.

Ένα χαρακτηριστικό παράδειγμα αποτελεί ο πολλαπλασιασμός πινάκων. Μία τεχνική για τη βελτίωση της πολυπλοκότητας των υπολογισμών είναι η εφαρμογή του αλγορίθμου του Strassen. Στον κλασσικό αλγόριθμο απαιτούνται 8 πολλαπλασιασμοί και 4 προσθέσεις για τον πολλαπλασιασμό δύο υποπινάκων 2 x 2. Ο αλγόριθμος του Strassen μειώνει τους απαιτούμενους πολλαπλασιασμούς σε 7, αυξάνει όμως τις προσθέσεις σε 16. Μία συμβατική αρχιτεκτονική, στην οποία ο πολλαπλασιασμός κινητής υποδιαστολής απαιτεί πολλούς περισσότερους πόρους και κύκλους ρολογιού από το σύστημα εν συγκρίσει με την πρόσθεση, θα ωφεληθεί από την εφαρμογή του αλγορίθμου του Strassen. Στη Cell Broadband Engine, όμως, τα κόστη ενός πολλαπλασιασμού και μίας πρόσθεσης είναι απολύτως ισοδύναμα. Έτσι, από την πλευρά του επεξεργαστή, ο κλασσικός αλγόριθμος απαιτεί 12 πράξεις κινητής υποδιαστολής ενώ ο αλγόριθμος του Strassen θα αυξήσει τις απαιτούμενες πράξεις σε 23.

Χρήση multi-buffering

Τα SPEs διαφέρουν αρκετά από της συμβατικές αρχιτεκτονικές πυρήνων λόγω της χρήσης μίας αυτόνομης, ελεγχόμενης από το λογισμικό τοπικής μνήμης, σε αντίθεση με την παραδοσιακή ιεραρχία των ελεγχόμενων από το hardware κρυφών μνημών που χρησιμοποιεί το PPE. Οι καθυστερήσεις των μεταφορών δεν κρύβονται με χρήση pre-fetching, αλλά

αξιοποιώντας τις ελεγχόμενες από το λογισμικό μηχανές DMA οι οποίες αποζεύουν τις μεταφορές μεταξύ κύριας και τοπικής μνήμης από την εκτέλεση των εντολών.

Ο συμβατικός τρόπος μεταφοράς δεδομένων, κατά τον οποίον ένα δεδομένο μεταφέρεται ακριβώς πριν χρησιμοποιηθεί, καθηλώνει την επίδοση του επεξεργαστή, καθώς ο τελευταίος αδρανεί για μεγάλο μέρος του χρόνου, περιμένοντας να ολοκληρωθούν οι μεταφορές DMA. Η χρήση τεχνικών multi-buffering εκμεταλλεύεται τα ιδιαίτερα χαρακτηριστικά της CBE και επικαλύπτει την εκτέλεση πράξεων στα δεδομένα κάποιων buffers με τη μεταφορά δεδομένων από/προς τους υπολοίπους. Αυτή η προσέγγιση επιτρέπει την αποδοτικότερη αξιοποίηση τόσο του εύρους ζώνης όσο και της δυνατότητας της CBE για ανεξάρτητη εκτέλεση υπολογισμών και μεταφορών. Για αυτούς του λόγους, η χρήση multi-buffering είναι κρίσιμης σημασίας για την επίτευξη μέγιστης επίδοσης, αυξάνει, όμως, παράλληλα την πολυπλοκότητα του προγραμματιστικού μοντέλου.

Στις εφαρμογές αυτής της μελέτης έχουν χρησιμοποιηθεί οι τεχνικές double-, triple- και quadruple-buffering. Για τα δεδομένα εκείνα που απλά διαβάζονται αρκεί η εφαρμογή του double-buffering για την επικάλυψη υπολογισμών – επικοινωνίας. Για δεδομένα τα οποία χρησιμοποιούνται για ανάγνωση και εγγραφή ενδέχεται η χρήση triple-buffering να ωφελήσει την επίδοση, έτσι ώστε ένας buffer να χρησιμοποιείται για την προ-φόρτωση δεδομένων, ένας για την αποθήκευση στην κύρια μνήμη αποτελεσμάτων που έχουν προηγουμένως εξαχθεί και ένας για την επεξεργασία των τρεχόντων δεδομένων. Σε κάποιες περιπτώσεις, όπως στον πολλαπλασιασμό πινάκων που επεξηγείται παρακάτω, υπαγορεύεται η χρήση quadruple-buffering.

Χρήση διανυσματικών πράξεων

Ένα χαρακτηριστικό της CBE είναι ότι τα SPEs είναι αποκλειστικά διανυσματικά επεξεργαστικά στοιχεία. Αυτό σημαίνει ότι το σύνολο της αρχιτεκτονικής της SPU είναι βελτιστοποιημένο για διανυσματικές πράξεις και διανυσματικούς τύπους δεδομένων, ενώ υπάρχει σημαντικό overhead στη διαχείριση βαθμωτών (scalars). Η χρήση διανυσματικών πράξεων και τύπων δεδομένων έχει το πλεονέκτημα της ταυτόχρονης εφαρμογής μίας πράξης πάνω σε περισσότερα δεδομένα. Για παράδειγμα, η εκτέλεση μίας εντολής πρόσθεσης σε δεδομένα κινητής υποδιαστολής απλής ακρίβειας σε διανυσματική μορφή πραγματοποιεί στην ουσία τέσσερεις προσθέσεις ταυτόχρονα, τετραπλασιάζοντας την επίδοση.

Η χρήση διανυσμάτων δεν είναι απλώς προτεινόμενη, αλλά επιβεβλημένη. Οι βαθμωτοί τύποι δεδομένων, πέραν του ότι δεν αξιοποιούν την παραπάνω παραλληλία, εισάγουν πρόσθετο φόρτο από τις αναγκαίες εντολές διαμόρφωσης (shuffle, rotate, shift), διότι τα δεδομένα φορτώνονται/αποθηκεύονται από/προς την τοπική μνήμη ανά τετραπλή λέξη (δηλαδή ανά διάνυσμα) και ότι οι βαθμωτές πράξεις περιμένουν τα δεδομένα σε συγκεκριμένες θέσεις μέσα στο διάνυσμα (preferred slots). Συμπερασματικά, η χρήση

διανυσμάτων δεν μπορεί να θεωρηθεί μόνο ως βελτιστοποίηση, αλλά και ως απαραίτητη προσέγγιση για τον περιορισμό βοηθητικών εντολών, οι οποίες επιβαρύνουν την επίδοση.

Εκμετάλλευση των δύο ετερογενών pipelines

Κάθε SPU διαθέτει δύο ετερογενή pipelines, τα οποία λειτουργούν πλήρως ανεξάρτητα το ένα από το άλλο. Η λειτουργία του ενός pipeline συνοψίζεται στην εκτέλεση υπολογιστικών εντολών, ενώ του άλλου στην φόρτωση, αποθήκευση και μορφοποίηση των διανυσμάτων. Οι δύο αυτές λειτουργίες αντικατοπτρίζουν τον τρόπο με τον οποίον γίνεται (συνήθως) η επεξεργασία δεδομένων. Σε μία συμβατική αρχιτεκτονική, οι εντολές για τις δύο λειτουργίες θα διαδέχονταν η μία την άλλη. Στην SPU, όμως, οι δύο σωληνώσεις μπορούν να λειτουργήσουν ανεξάρτητα, αρκεί να μην υπάρχουν εξαρτήσεις και κίνδυνοι δεδομένων. Χάρη σε αυτό το χαρακτηριστικό, οι δύο συνιστώσες της επεξεργασίας δεδομένων μπορούν να υλοποιηθούν και να προγραμματιστούν παράλληλα και, κυρίως, ανεξάρτητα. Το σχήμα που ακολουθείται για τη μεγιστοποίηση της επίδοσης είναι η έγκαιρη διαχείριση (φόρτωση, αποθήκευση, διαμόρφωση) των δεδομένων από το περιττό pipeline, έτσι ώστε αυτά να είναι έτοιμα προς χρήση ως ορίσματα – αποτελέσματα για τις υπολογιστικές εντολές του άρτιου pipeline.

Για να είναι εφικτή η υλοποίηση αυτού του σχήματος είναι απαραίτητη η ανάπτυξη του κώδικα με ξεδιπλωμένους βρόχους. Μία εφαρμογή συνίσταται συνήθως στην επαναληπτική εκτέλεση (μονός, διπλός, τριπλός, …, πολλαπλός βρόχος) μίας συγκεκριμένης ακολουθίας εντολών. Η ακολουθία αυτή παρουσιάζει τη δομή: φόρτωση δεδομένων σε καταχωρητές → εκτέλεση πράξεων → αποθήκευση δεδομένων στη μνήμη. Η προσέγγιση που έχει χρησιμοποιηθεί στις εφαρμογές αυτής της μελέτης είναι η προ-φόρτωση σε ένα στιγμιότυπο της επανάληψης των δεδομένων που απαιτούνται στην επόμενη επανάληψη και η αποθήκευση στη μνήμη των αποτελεσμάτων της προηγούμενης επανάληψης.

Δυστυχώς, η αποτελεσματική δρομολόγηση εντολών και διαχείριση των σωληνώσεων απαιτούν εις βάθος ανάλυση της ροής του κώδικα και των δεδομένων. Η ανάλυση αυτή είναι πολύ ιδιαίτερη για κάθε εφαρμογή και αρκετά πολύπλοκη ώστε να υλοποιηθεί από έναν compiler. Έτσι, η κατασκευή αποδοτικού κώδικα επιβαρύνει τον προγραμματιστή, ο οποίος πρέπει να αφιερώσει περισσότερο χρόνο και να αντιμετωπίσει κώδικα σημαντικά μεγαλύτερου μεγέθους και πολυπλοκότητας.

Αναπαράσταση αριθμών κινητής υποδιαστολής

Στις εφαρμογές που ακολουθούν έχει χρησιμοποιηθεί για την αναπαράσταση των αριθμών κινητής υποδιαστολής το πρότυπο απλής ακρίβειας. Η επιλογή αυτή βασίζεται στο γεγονός ότι η τρέχουσα υλοποίηση της Cell Broadband Engine, αν και υποστηρίζει SIMD πράξεις με αριθμούς διπλής ακρίβειας, εντούτοις η επίδοση που επιτυγχάνεται είναι πολύ

φτωχή και δεν επιτρέπει τη μελέτη άλλων χαρακτηριστικών του επεξεργαστή, όπως οι ασύγχρονες μεταφορές DMA, το υψηλό εύρος ζώνης κ.τ.λ.. Ο κώδικας των υπολογισμών της SPU μπορεί να επιτύχει την ακόλουθη επίδοση:

- Floats: Όπως προκύπτει από τα τεχνικά χαρακτηριστικά της αρχιτεκτονικής, οι εντολές για αριθμούς κινητής υποδιαστολής απλής ακρίβειας είναι πλήρως pipelined, πράγμα που σημαίνει ότι ανά κύκλο ρολογιού παράγεται ένα αποτέλεσμα-διάνυσμα, δηλαδή τέσσερα αποτελέσματα κινητής υποδιαστολής απλής ακρίβειας.
- Doubles: Οι εντολές για αριθμούς κινητής υποδιαστολής διπλής ακρίβειας δεν είναι πλήρως pipelined, αλλά υπάρχει μία καθυστέρηση (stall) έξι κύκλων μεταξύ της έκδοσης μίας εντολής κινητής υποδιαστολής διπλής ακρίβειας και της επόμενής της. Το γεγονός αυτό από μόνο του έχει ως αποτέλεσμα την παραγωγή ενός αποτελέσματος-διανύσματος ανά επτά κύκλους και καθιστά τον κώδικα των υπολογισμών επτά φορές αργότερο. Συνυπολογίζοντας ότι κάθε αποτέλεσμα-διάνυσμα περιέχει δύο αντί για τέσσερεις αριθμούς κινητής υποδιαστολής διπλής ακρίβειας, συνεπάγεται ότι η χρήση doubles μειώνει το θεωρητικό μέγιστο της επίδοσης του κώδικα υπολογισμών κατά δώδεκα φορές.

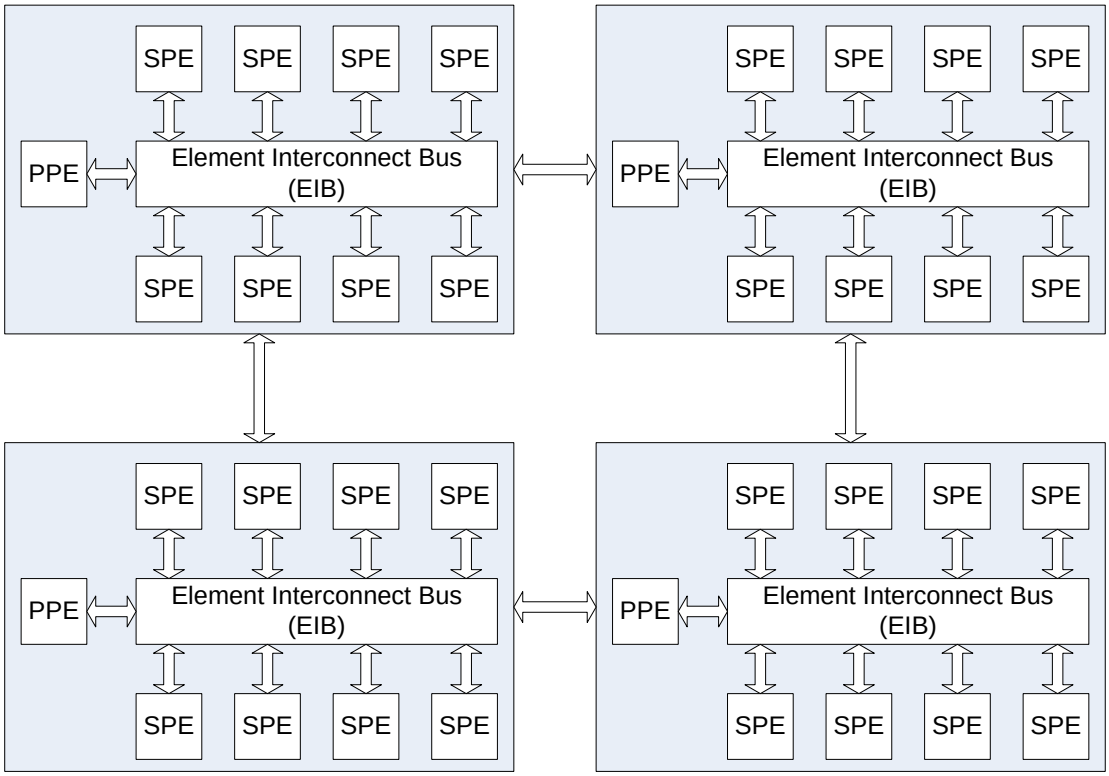
Στην υλοποίηση της Cell Broadband Engine που χρησιμοποιήθηκε ο επεξεργαστής λειτουργεί στα 3.2GHz και τα θεωρητικά μέγιστα της επίδοσης, αξιοποιώντας τις εντολές που πραγματοποιούν δύο πράξεις κινητής υποδιαστολής ταυτόχρονα (π.χ. πολλαπλασιασμό δύο διανυσμάτων και πρόσθεση του γινομένου σε τρίτο διάνυσμα σε μία εντολή), έχουν ως εξής:

- Floats: $3.2GHz \times 1 \frac{\text{αποτ έλεσμα } -\text{διάνυσμα}}{\text{κύκλο}} \times 4 \frac{\text{αποτελ έσματα}}{\text{αποτ έλεσμα } -\text{διάνυσμα}} \times 2 \frac{\text{πράξεις κινητής υποσιαστολής}}{\text{αποτ έλεσμα}} = 25.6GFLOPS$
- Doubles: $3.2GHz \times \frac{1 \text{ αποτ έλεσμα } -\text{διάνυσμα}}{7 \text{ κύκλους}} \times 2 \frac{\text{αποτελ έσματα}}{\text{αποτ έλεσμα } -\text{διάνυσμα}} \times 2 \frac{\text{πράξεις κινητής υποσιαστολής}}{\text{απο τέλεσμα}} = 1.83GFLOPS$

Υπάρχει μία νεώτερη υλοποίηση της Cell Broadband Engine, η PowerXCell 8i, η οποία υπερβαίνει το πρόβλημα των καθυστερήσεων στην έκδοση εντολών κινητής υποδιαστολής διπλής ακρίβειας, πράγμα που σημαίνει ότι οι εντολές αυτές είναι πλήρως pipelined και το θεωρητικό μέγιστο των υπολογιστικών εφαρμογών ανέρχεται στα 12.8GFLOPS.

Παραλληλοποίηση προβλημάτων

Κάθε Cell Broadband Engine αποτελείται από 8 SPEs. Στην περίπτωση μας, όπου χρησιμοποιήθηκε μία έκδοση της CBE που ενσωματώνεται στην κονσόλα PlayStation 3, έχουμε ενεργά μόνο τα 6 από τα 8 SPEs. Καθώς το κάθε SPE μπορεί να εκτελεί κώδικα αυτόνομα, η ίδια η CBE μπορεί να θεωρηθεί ως ένα σύστημα παράλληλης επεξεργασίας δεδομένων. Δημιουργώντας μία συστοιχία από CBEs ουσιαστικά έχουμε ένα παράλληλο σύστημα στο οποίο η κάθε CBE εσωτερικά είναι ένα σύστημα παράλληλης επεξεργασίας. Τα δύο επίπεδα παραλληλοποιήσεων, ένα μεταξύ των κόμβων της συστοιχίας και ένα μεταξύ των SPEs εντός του κάθε κόμβου, γίνονται ανεξάρτητα και αυτόνομα το ένα από το άλλο, τόσο σχεδιαστικά όσο και προγραμματιστικά. Η υλοποίηση των δύο αυτόνομων, παράλληλων συστημάτων, φαίνεται σχηματικά παρακάτω.



Η παραλληλοποίηση μίας εφαρμογής απαιτεί τη διάτμηση των δεδομένων της σε μικρότερα κομμάτια – blocks, κάθε ένα από τα οποία θα υπόκειται σε επεξεργασία από έναν κόμβο ή ένα SPE ανεξάρτητα (ή σχεδόν ανεξάρτητα) από τα υπόλοιπα. Για τα προβλήματα όπου υπάρχει εξάρτηση μεταξύ των δεδομένων, όπως είναι το πρόβλημα θερμικής διάχυσης, απαιτείται μία πρόσθετη μεταφορά δεδομένων (π.χ. συνοριακών τιμών) μεταξύ των επεξεργαστικών μονάδων.

Διαμέριση Δεδομένων σε Blocks

Υπάρχουν δύο προσεγγίσεις για τον χωρισμό των δεδομένων σε blocks. Η πρώτη προσέγγιση είναι η δημιουργία blocks σταθερού μεγέθους (και διαστάσεων, εάν το πρόβλημα έχει γεωμετρία). Το μέγεθος αυτό καθορίζεται από τον τρόπο σχεδιασμού της εφαρμογής και των λεπτομερειών της εκάστοτε αρχιτεκτονικής. Η δεύτερη προσέγγιση συνίσταται στη δημιουργία blocks σταθερού πλήθους. Συνήθως, το πλήθος αυτό καθορίζεται από το πλήθος των επεξεργαστικών μονάδων που υπάρχουν στο σύστημα.

Blocks σταθερού μεγέθους

Σε αυτήν την προσέγγιση παραλληλοποίησης τα δεδομένα του προβλήματος χωρίζονται σε blocks σταθερού μεγέθους / διαστάσεων, το πλήθος των οποίων (blocks) προκύπτει από τις διαστάσεις του προβλήματος. Το πλεονέκτημα αυτής της προσέγγισης είναι ότι, λόγω του σταθερού μεγέθους (και της εκ των προτέρων γνωστής γεωμετρίας στα αντίστοιχα προβλήματα), ο προγραμματιστής έχει τη δυνατότητα να οργανώσει τον κώδικα του προγράμματος και τις απαραίτητες πράξεις μέσα στο block με τέτοιον τρόπο ώστε η επίδοση του προγράμματος να προσεγγίζει όσο περισσότερο γίνεται την μέγιστη δυνατή.

Όμως, το σταθερό μέγεθος block επιβάλλει περιορισμούς στις συνολικές διαστάσεις του προβλήματος, οι οποίες πρέπει να είναι πολλαπλάσια των διαστάσεων του block. Όσο αυξάνεται το μέγεθος του block γίνονται αυστηρότεροι οι παραπάνω περιορισμοί, ενώ μειώνεται και το πλήθος των blocks, γεγονός που δυσκολεύει την παράλληλη επεξεργασία. Από την άλλη πλευρά, μικρό μέγεθος συνεπάγεται μεγάλο αριθμό από blocks, γεγονός που αυξάνει τις εξαρτήσεις δεδομένων, την απαιτούμενη επικοινωνία μεταξύ επεξεργαστικών μονάδων και μνήμης και τον συγχρονισμό των επεξεργαστικών μονάδων μεταξύ τους.

Στις εφαρμογές που μελετήθηκαν χρησιμοποιήθηκε η προσέγγιση των blocks σταθερού μεγέθους για την παραλληλοποίηση σε επίπεδο εσωτερικά των κόμβων, δηλαδή σε επίπεδο SPEs.

Blocks σταθερού πλήθους

Σε αυτήν την προσέγγιση παραλληλοποίησης τα δεδομένα του προβλήματος χωρίζονται σε τόσα ίσα μέρη όσοι είναι και οι επεξεργαστές που χρησιμοποιούνται και με τέτοια γεωμετρία όπως είναι η γεωμετρία του πλέγματος (grid) των επεξεργαστών. Μια τέτοια προσέγγιση έχει ως αποτέλεσμα το ελάχιστο δυνατό πλήθος από blocks, πράγμα που περιορίζει τις εξαρτήσεις, τις μεταφορές από και προς τη μνήμη και το συγχρονισμό μεταξύ των επεξεργαστικών μονάδων του συστήματος.

Το μειονέκτημα του μικρού πλήθους blocks είναι η αυξημένη δυσκολία στην παράλληλη επεξεργασία. Επίσης, λόγω του ασαφούς μεγέθους που θα έχει το κάθε block, δεν

είναι δυνατή η παραγωγή ενός απόλυτα βελτιστοποιημένου κώδικα, αλλά επιβάλλεται η χρήση μίας πιο «γενικής» (generic) υλοποίησης.

Στις εφαρμογές που μελετήθηκαν χρησιμοποιήθηκε η προσέγγιση των blocks σταθερού πλήθος για την παραλληλοποίηση σε επίπεδο συστοιχίας, δηλαδή μεταξύ των κόμβων.

Αδρομερής παραλληλισμός

Ένα ενδιαφέρον ζήτημα στο πρόβλημα της παραλληλοποίησης εφαρμογών είναι η καταπίεση του κόστους επικοινωνίας και ο έλεγχος με αποδοτικό τρόπο της ισορροπίας στο ισοζύγιο μεταξύ υπολογισμού και επικοινωνίας. Η πρώτη προσέγγιση στο ζήτημα είναι η εκτέλεση κάθε επανάληψης ξεχωριστά και η επικοινωνία μεταξύ των επεξεργαστών σε κάθε μία επανάληψη (fine grain parallelism). Αν το υπό μελέτη πρόβλημα έχει μεγάλο υπολογιστικό όγκο, δηλαδή οι μεταφορές δεδομένων αποτελούν ένα μικρό κλάσμα του συνολικού φόρτου εκτέλεσης, και η χρονική στιγμή κατά την οποία θα ξεκινήσει μια επεξεργαστική μονάδα την εκτέλεση εξαρτάται άμεσα από το πότε θα βγάλει αποτελέσματα κάποια προηγούμενη επεξεργαστική μονάδα (εξάρτηση δεδομένων) τότε το fine grain parallelism είναι γενικά προτιμότερο. Αν, όμως, ο φόρτος υπολογισμού δεν είναι αρκετά μεγάλος σε σχέση με το φόρτο μεταφορών ή το πλήθος των μεταφορών είναι μεγάλο, τότε η μεταφορά δεδομένων συχνά μπορεί να αποτελέσει τροχοπέδη στην fine grain προσέγγιση.

Στον αδρομερή παραλληλισμό (coarse grain parallelism) η μείωση του κόστους επικοινωνίας γίνεται με την εφαρμογή του σχήματος του tiling. Με αυτόν τον τρόπο k διαδοχικές επαναλήψεις ομαδοποιούνται και κατασκευάζουν μία «υπέρ-επανάληψη» η οποία μπορεί να εκτελεστεί ενιαία και χωρίς επικοινωνία στο ενδιάμεσο με άλλους επεξεργαστές. Λέμε ότι έχει εφαρμοστεί tiling μεγέθους k. Οι ανταλλαγές δεδομένων με κάθε γειτονικό επεξεργαστή επίσης ομαδοποιούνται σύμφωνα με την ομαδοποίηση των επαναλήψεων και τα δεδομένα στέλνονται συγκεντρωτικά στο τέλος της κάθε υπέρ-επανάληψης. Με αυτό τον τρόπο υποπολλαπλασιάζεται κατά έναν παράγοντα k το πλήθος των μεταφορών που πρέπει να διεξαχθούν. Αυτό μπορεί να επιφέρει αξιοσημείωτη αύξηση στην επίδοση του παράλληλου συστήματος, καθώς κάθε μεταφορά συνοδεύεται από κάποιο overhead. Ομαδοποιώντας τις μεταφορές μειώνεται το overhead αυτό, πράγμα που ωφελεί την επίδοση, παρόλο που ο όγκος των μεταφορών παραμένει ίδιος. Η συνεχής αύξηση, όμως, του παράγοντα tiling αρχίζει και δημιουργεί πρόβλημα, από κάποιο σημείο και μετά, στην επίδοση του παράλληλου συστήματος, καθώς με τη δημιουργία μεγάλων «υπέρ-επαναλήψεων» k-πλασιάζεται ο χρόνος που απαιτείται για να υπολογιστούν και να είναι έτοιμα προς αποστολή τα δεδομένα επικοινωνίας από έναν κόμβο προς τους κόμβους που εξαρτώνται από αυτόν. Επίσης, μειώνεται μεν το πλήθος των μεταφορών, αφού αυτές ομαδοποιούνται σε λίγες μεγάλες μεταφορές, η αύξηση όμως του μεγέθους της κάθε μεταφοράς επιβαρύνει τα συστήματα που

επικοινωνούν μεταξύ τους με σύγχρονο και όχι ασύγχρονο τρόπο, αφού η κάθε μεταφορά απαιτεί περισσότερο χρόνο για να ολοκληρωθεί.

Υπολογιστικό σύστημα μετρήσεων

Το σύστημα που χρησιμοποιήθηκε για τη διεξαγωγή των μετρήσεων είναι μία συστοιχία από κονσόλες PlayStation 3, συνδεδεμένες μέσω switch. Κάθε PlayStation περιέχει έναν επεξεργαστή Cell Broadband Engine με ενεργά τα έξι από τα οκτώ SPEs που προβλέπει η Cell Broadband Engine Architecture. Υπάρχει και ένα έβδομο SPE, το οποίο είναι δεσμευμένο από το λειτουργικό σύστημα (hypervisor) της κονσόλας και είναι απρόσιτο στον προγραμματιστή. Επίσης, η CBE λειτουργεί στη συχνότητα των 3.2GHz. Το σύστημα εξοπλίζεται με 256MB XDR μνήμης και κάρτα δικτύου Gigabit Ethernet. Όλοι οι κόμβοι της συστοιχίας τρέχουν λειτουργικό σύστημα Debian/GNU Linux, με πυρήνα 2.6.24.

Κεφάλαιο 5

ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΣ ΠΙΝΑΚΩΝ

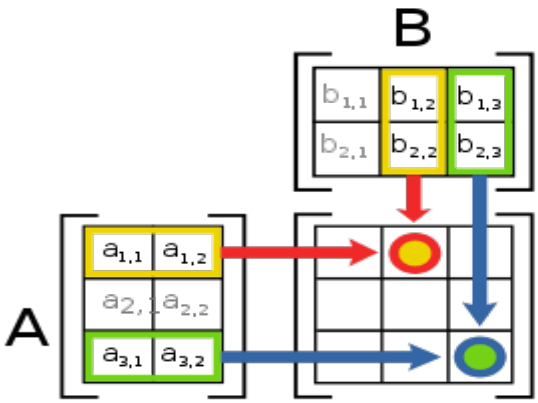
Περιγραφή του προβλήματος

Ο πολλαπλασιασμός πινάκων είναι ένα πρόβλημα γραμμικής άλγεβρας που απαντάται σε πολλές σύγχρονες εφαρμογές. Η πράξη του πολλαπλασιασμού μεταξύ δύο πινάκων ορίζεται ως εξής: Έστω πίνακας A, διαστάσεων i x k, και πίνακας B, διαστάσεων k x j. Είναι απαραίτητο η διάσταση k να είναι ίδια μεταξύ των δύο πινάκων, δηλαδή το πλήθος των στηλών του πίνακα A πρέπει να είναι ίδιο με το πλήθος των γραμμών του πίνακα B. Το αποτέλεσμα είναι ένας πίνακας C, διαστάσεων i x j. Κάθε στοιχείο (x, y) του πίνακα C προκύπτει από το άθροισμα των γινομένων μεταξύ των στοιχείων των αντίστοιχων θέσεων της x-οστής γραμμής του πίνακα A και της y-οστής στήλης του πίνακα B. Έστω το παρακάτω παράδειγμα, με έναν πίνακα A, διαστάσεων 3 x 4, και έναν πίνακα B, διαστάσεων 4 x 5. Το γινόμενο είναι ο πίνακας C, διαστάσεων 3 x 5. Στο σχήμα δίνεται και ο τρόπος υπολογισμού του στοιχείου (3, 4) του πίνακα γινομένου.

$$\begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 1 & 2 & 3 & 4 \end{bmatrix} \begin{bmatrix} \cdot & \cdot & \cdot & a & \cdot \\ \cdot & \cdot & \cdot & b & \cdot \\ \cdot & \cdot & \cdot & c & \cdot \\ \cdot & \cdot & \cdot & d & \cdot \end{bmatrix} = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & x_{3,4} & \cdot \end{bmatrix}$$

$$x_{3,4} = (1,2,3,4) \cdot (a,b,c,d) = 1 \times a + 2 \times b + 3 \times c + 4 \times d.$$

Με τον ίδιο τρόπο υπολογίζεται οποιοδήποτε στοιχείο του πίνακα γινομένου, όπως φαίνεται στο παρακάτω σχήμα.



Ο πολλαπλασιασμός πινάκων χαρακτηρίζεται ως ένα πρόβλημα με πολύ μεγάλο πλήθος υπολογισμών (compute-intensive). Ο κλασσικός αλγόριθμος πολλαπλασιασμού έχει πολυπλοκότητα $O(n^3)$. Ταυτόχρονα, δεν υπάρχουν εξαρτήσεις μεταξύ των δεδομένων, δηλαδή κάθε κομμάτι του πίνακα γινομένου μπορεί να υπολογιστεί ανεξάρτητα από τα υπόλοιπα, όπως φαίνεται και στο παραπάνω σχήμα, αρκεί ο υπολογισμός κάθε τέτοιου κομματιού να υπάγεται σε μία και μόνο επεξεργαστική μονάδα. Τα δύο παραπάνω χαρακτηριστικά κάνουν τον υπολογισμό του πολλαπλασιασμού πινάκων ένα πολύ καλό παράδειγμα παράλληλης επεξεργασίας. Εμπεριέχει πολλές πράξεις κινητής υποδιαστολής, οπότε η παραλληλοποίηση επιταχύνει τη διαδικασία, ενώ η απουσία εξαρτήσεων κάνει την υλοποίηση της παραλληλοποίησης σχετικά εύκολη.

Δεδομένου ότι ο όγκος των πράξεων είναι πολύ μεγαλύτερος από τις μεταφορές δεδομένων θεωρείται σκόπιμο να βελτιστοποιηθεί κατά το δυνατόν η επίδοση του υπολογιστικού μέρους του αλγορίθμου. Κάτι τέτοιο είναι προφανές ότι μπορεί να επιτευχθεί εάν χωριστεί ο πίνακας σε blocks σταθερού μεγέθους.

Πολλαπλασιασμός πινάκων σε επίπεδο ενός κόμβου

Διαμέριση των πινάκων σε blocks

Η πρώτη σχεδιαστική απόφαση για την υλοποίηση του πολλαπλασιασμού πινάκων στη Cell Broadband Engine είναι η επιλογή του μεγέθους του block και του τρόπου ανάθεσης των διαφόρων blocks στις επεξεργαστικές μονάδες, έτσι ώστε να επιτευχθεί η καλύτερη δυνατή επίδοση. Το μέγεθος του block καθορίζεται κυρίως από το διαθέσιμο χώρο στην τοπική μνήμη ενός SPE. Το ζητούμενο είναι η μέγιστη αξιοποίηση αυτού του χώρου. Στην υλοποίηση της Cell Broadband Engine που χρησιμοποιήθηκε, το μέγεθος της τοπικής μνήμης είναι ίσο με 256KB. Σε όλα τα προγράμματα που μελετήθηκαν έγινε η (έστω και αυθαίρετη) υπόθεση ότι ο χώρος της τοπικής μνήμης διατίθεται κατά το ήμισυ (128KB) για τις εντολές του προγράμματος και κατά το άλλο ήμισυ (128KB) για τα δεδομένα. Εκ των υστέρων αποδεικνύεται ότι η υπόθεση αυτή δεν απέχει από την πραγματικότητα.

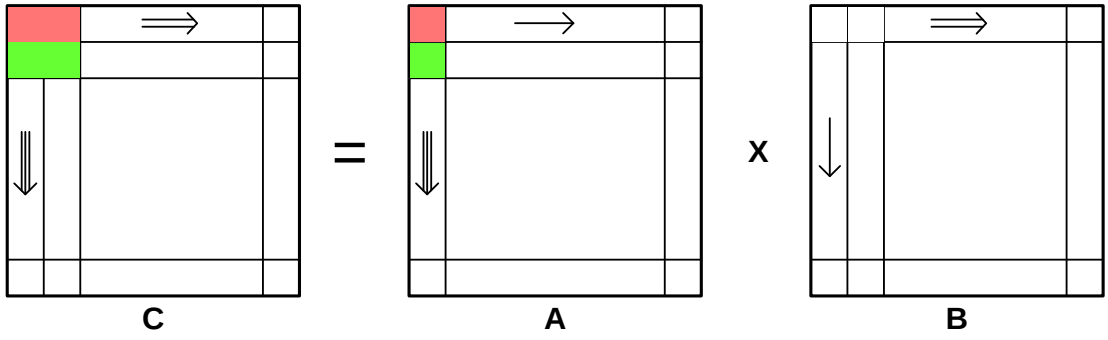
Blocks σταθερού μεγέθους

Ακολουθεί η περιγραφή της διαμέρισης των πινάκων κατά την προσέγγιση των blocks σταθερού μεγέθους.

Διαίρεση σε blocks 64 x 64

Έστω ότι A και B είναι οι πίνακες εισαγωγής και C ο πίνακας γινόμενο. Ένας τρόπος χωρισμού του προβλήματος είναι η διαμέριση των πινάκων σε ισομεγέθη και τετράγωνα blocks. Προκειμένου να αξιοποιηθεί η δυνατότητα για double buffering, που είναι αναγκαία για την υψηλή επίδοση του κώδικα, όπως έχει αναφερθεί, είναι απαραίτητο να υπάρχουν δύο buffers για τα blocks του πίνακα A και άλλοι δύο για τα blocks του πίνακα B. Όμως, όπως

φαίνεται στο παρακάτω σχήμα, από τα 2 + 2 blocks των πινάκων A και B υπολογίζονται μερικά αθροίσματα-αποτελέσματα για 4 blocks του πίνακα C. Άρα, συνολικά πρέπει να χρησιμοποιηθούν 2 + 2 + 4 = 8 buffers και στον πίνακα C χρησιμοποιείται τεχνική quadruple buffering. Με την υπόθεση ότι υπάρχουν διαθέσιμα 128KB στην τοπική μνήμη για αυτούς τους buffers, κάθε ένας μπορεί να έχει μέγιστο μέγεθος 16KB. Χρησιμοποιώντας αριθμούς κινητής υποδιαστολής απλής ακρίβειας (4 bytes έκαστος) προκύπτει ότι το μέγιστο block χωράει 4.096 αριθμούς, άρα οι διαστάσεις των blocks προκύπτουν ίσες με 64 x 64. Το μέγεθος αυτό είναι συμβατό με το μέγεθος ενός διανύσματος, ενώ το μέγεθος σε bytes μίας γραμμής του block είναι 256 bytes, πράγμα που σημαίνει ότι είναι πολλαπλάσιο της cacheline της CBE και έχουμε τη μέγιστη επίδοση κατά τη μεταφορά των δεδομένων από και προς τη μνήμη.



Ένας τρόπος για τη σάρωση των πινάκων είναι αυτός που φαίνεται στο παραπάνω σχήμα. Το βελάκι με τη μονή γραμμή αναπαριστά τον εσωτερικότερο βρόχο (loop), ενώ το βελάκι με την τριπλή γραμμή αναπαριστά τον πιο εξωτερικό βρόχο. Με αυτόν τον τρόπο κάθε block του πίνακα του αποτελέσματος (ή, πιο συγκεκριμένα, κάθε τετράδα από blocks) υπολογίζεται σε μία και μόνο εκτέλεση του εσωτερικότερου loop του αλγορίθμου. Κάτι τέτοιο κάνει δυνατό το σχηματισμό του αποτελέσματος για μία τετράδα blocks του C από ένα και μόνο SPE, συμβάλλοντας σημαντικά στην συνάφεια και την συνέπεια του προβλήματος χωρίς περαιτέρω κώδικα για έλεγχο και συγχρονισμό μεταξύ των SPEs. Η έλλειψη σημείων συγχρονισμού δείχνει ότι είναι δυνατή η επ' άπειρο επέκταση των επεξεργαστικών μονάδων που μπορεί κανείς να χρησιμοποιήσει, διατηρώντας τη γραμμικότητα στην αύξηση της επίδοσης. Η μοναδική παράμετρος που περιορίζει αυτή την αύξηση είναι το ίδιο το μέγεθος των υπό μελέτη πινάκων. Αν το μέγεθος των πινάκων γίνει αρκετά μικρό σε σχέση με το πλήθος των επεξεργαστικών μονάδων, τότε είναι προφανές ότι δεν αναμένεται η ίδια βελτίωση στην επίδοση.

Σε κώδικα C - ψευδοκώδικα το παραπάνω σχήμα έχει ως εξής:

```
/*
  (i/j)_blocks: πλήθος blocks στον άξονα (i/j)
  spe_id: αύξων αριθμός SPE
  spe_threads: πλήθος SPEs που τρέχουν
  (A/B/C)_buffer_(0/1/2/3)_(i/j/k): offset-block στον πίνακα (A/B/C)
                          στον άξονα (i/j/k) για τον buffer (0/1/2/3)
  (A/B/C)_buffer[(0/1/2/3)]:buffer(0/1/2/3) του πίνακα (A/B/C)
  tags: πίνακας με ετικέτες για τις μεταφορές DMA
*/

i = 2 * (spe_id / (j_blocks / 2));
j = 2 * (spe_id % (j_blocks / 2));

A_buffer_0_i = i;
A_buffer_0_k = 0;
A_buffer_1_i = i + 1;
A_buffer_1_k = 0;
B_buffer_0_k = 0;
B_buffer_0_j = j;
B_buffer_1_k = 0;
B_buffer_1_j = j + 1;
C_buffer_0_i = i;
C_buffer_0_j = j;
C_buffer_1_i = i;
C_buffer_1_j = j + 1;
C_buffer_2_i = i + 1;
C_buffer_2_j = j;
C_buffer_3_i = i + 1;
C_buffer_3_j = j + 1;

load_block(matrix_A, A_buffer[0], A_buffer_0_i, A_buffer_0_k, tags[0]);
load_block(matrix_B, B_buffer[0], B_buffer_0_k, B_buffer_0_j, tags[0]);
load_block(matrix_C, C_buffer[0], C_buffer_0_i, C_buffer_0_j, tags[0]);

load_block(matrix_B, B_buffer[1], B_buffer_1_k, B_buffer_1_j, tags[1]);
load_block(matrix_C, C_buffer[1], C_buffer_1_i, C_buffer_1_j, tags[1]);

while(i < i_blocks)
{
  load_block(matrix_C, C_buffer[2], C_buffer_2_i, C_buffer_2_j, tags[2]);
  load_block(matrix_C, C_buffer[3], C_buffer_3_i, C_buffer_3_j, tags[2]);

  for(k = 1; k < k_blocks; k++)
  {
    wait_dma_group(tags[0]);
    block_mult(A_buffer[0], B_buffer[0], C_buffer[0]);

    load_block(matrix_A, A_buffer[1], A_buffer_1_i, A_buffer_1_k + k - 1, tags[2]);

    wait_dma_group(tags[1]);
    block_mult(A_buffer[0], B_buffer[1], C_buffer[1]);

    load_block(matrix_A, A_buffer[0], A_buffer_0_i, A_buffer_0_k + k, tags[0]);

    wait_dma_group(tags[2]);
    block_mult(A_buffer[1], B_buffer[0], C_buffer[2]);

    load_block(matrix_B, B_buffer[0], B_buffer_0_k + k, B_buffer_0_j, tags[0]);

    block_mult(A_buffer[1], B_buffer[1], C_buffer[3]);
```

```
    load_block(matrix_B, B_buffer[1], B_buffer_1_k + k, B_buffer_1_j, tags[1]);
  }

  j += 2 * spe_threads;

  while(j >= j_blocks)
  {
    i += 2;
    j -= j_blocks;
  }

  if(i < i_blocks)
  {
    update block offsets;
    perform for-loop sequence one last time;
    store C blocks;
  }
  else
  {
    perform for-loop sequence one time without loading matrices;
    store C blocks;
  }
}
```

Το σταθερό μέγεθος του block βοηθάει στη βελτιστοποίηση του κώδικα του πολλαπλασιασμού, εφαρμόζοντας κατάλληλα διάφορες τεχνικές, όπως ξεδίπλωμα βρόχων, διάταξη εντολών ώστε να διευκολύνεται η διπλή έκδοση και άλλα. Όπως έχει αναλυθεί, για την σωστή εκμετάλλευση των δύο ετερογενών pipelines της CBE είναι απαραίτητη η εφαρμογή loop-unrolling στον κώδικα των βρόχων. Επιπρόσθετα, με τον συγκεκριμένο τρόπο χωρισμού του προβλήματος, τα blocks του πίνακα C μεταφέρονται μόνο μία φορά από την κύρια μνήμη στην τοπική και μόνο μία ακόμα φορά από την τοπική μνήμη στην κύρια.

Το μειονέκτημα αυτού του τρόπου χωρισμού του προβλήματος είναι ότι οι πίνακες A και B μεταφέρονται πολλές φορές από και προς την τοπική μνήμη κάθε SPE. Στην πράξη, ο αντίκτυπος των αυξημένων μεταφορών δεν φαίνεται στην επίδοση της εφαρμογής, αφού η αυξημένη πολυπλοκότητα του πολλαπλασιασμού των blocks και οι τεχνικές multi-buffering στα δεδομένα κρύβουν αποτελεσματικά την επικοινωνία των επεξεργαστικών μονάδων με τη μνήμη. Επίσης, καθώς μεταφέρουμε blocks που περιλαμβάνουν διαφορετικές γραμμές (άρα μη συνεχόμενα στοιχεία στη μνήμη), είναι απαραίτητη η χρήση μεταφορών DMA-lists, με όποια επιβάρυνση σε σχέση με τη γραμμική μεταφορά αυτό συνεπάγεται.

Αξιοσημείωτος από αυτή την υλοποίηση είναι ο τρόπος με τον οποίο γίνεται η πράξη πολλαπλασιασμού μεταξύ δύο blocks. Ο κώδικας σε C του αλγορίθμου φαίνεται παρακάτω.

```
void block_mult(float * A_block, float * B_block, float * C_block)
{
  int i, j, k;
  register vector float row_a, splat_a;
  register vector float row_b0, row_b1, row_b2, row_b3;
  register vector float row_c;
  register const vector unsigned char splat_word0 = { 0x00, 0x01, 0x02, 0x03, 0x00, 0x01, 0x02, 0x03,
                          0x00, 0x01, 0x02, 0x03, 0x00, 0x01, 0x02, 0x03 };
  register const vector unsigned char splat_word1 = { 0x04, 0x05, 0x06, 0x07, 0x04, 0x05, 0x06, 0x07,
                          0x04, 0x05, 0x06, 0x07, 0x04, 0x05, 0x06, 0x07 };
  register const vector unsigned char splat_word2 = { 0x08, 0x09, 0x0A, 0x0B, 0x08, 0x09, 0x0A, 0x0B,
                          0x08, 0x09, 0x0A, 0x0B, 0x08, 0x09, 0x0A, 0x0B };
  register const vector unsigned char splat_word3 = { 0x0C, 0x0D, 0x0E, 0x0F, 0x0C, 0x0D, 0x0E, 0x0F,
                          0x0C, 0x0D, 0x0E, 0x0F, 0x0C, 0x0D, 0x0E, 0x0F };
  register float * pA, * pB, * pC;

  pC = C_block;

  /* BLOCK_DIM είναι η διάσταση του block – στην προκειμένη περίπτωση BLOCK_DIM = 64 */

  for(i = 0; i < BLOCK_DIM; i++)
  {
    for(j = 0; j < BLOCK_DIM / 4; j++)
    {
      pA = A_block + i * BLOCK_DIM;
      pB = B_block + j * 4;
      row_c = *((vector float *) pC);

      for(k = 0; k < BLOCK_DIM / 4; k++)
      {
        row_a = *((vector float *) pA);
        row_b0 = *((vector float *) pB);
        row_b1 = *((vector float *) (pB + BLOCK_DIM));
        row_b2 = *((vector float *) (pB + BLOCK_DIMX2));
        row_b3 = *((vector float *) (pB + BLOCK_DIMX3));

        splat_a = spu_shuffle(row_a, row_a, splat_word0);
        row_c = spu_madd(splat_a, row_b0, row_c);
        splat_a = spu_shuffle(row_a, row_a, splat_word1);
        row_c = spu_madd(splat_a, row_b1, row_c);
        splat_a = spu_shuffle(row_a, row_a, splat_word2);
        row_c = spu_madd(splat_a, row_b2, row_c);
        splat_a = spu_shuffle(row_a, row_a, splat_word3);
        row_c = spu_madd(splat_a, row_b3, row_c);

        pA += 4;
        pB += BLOCK_DIMX4;
      }

      *((vector float *) pC) = row_c;
      pC += 4;
    }
  }

  return;
}
```

Επισημαίνεται ότι αυτός ο κώδικας δίνεται προς κατανόηση του αλγορίθμου. Ο κώδικας που πραγματικά χρησιμοποιήθηκε προκύπτει από τον παρακάτω με τις κατάλληλες βελτιστοποιήσεις, σύμφωνα με την ανάλυση που έχει προηγηθεί (loop-unrolling, αναδιάταξη εντολών για εξασφάλιση συνεχούς dual-issuing, κατάλληλη προ-φόρτωση δεδομένων στους καταχωρητές ώστε το άρτιο pipeline να τα βρίσκει στις θέσεις τους όταν τα χρειαστεί και να τροφοδοτείται συνεχώς με δεδομένα, φτάνοντας την απόδοσή του κοντά στο θεωρητικό μέγιστο).

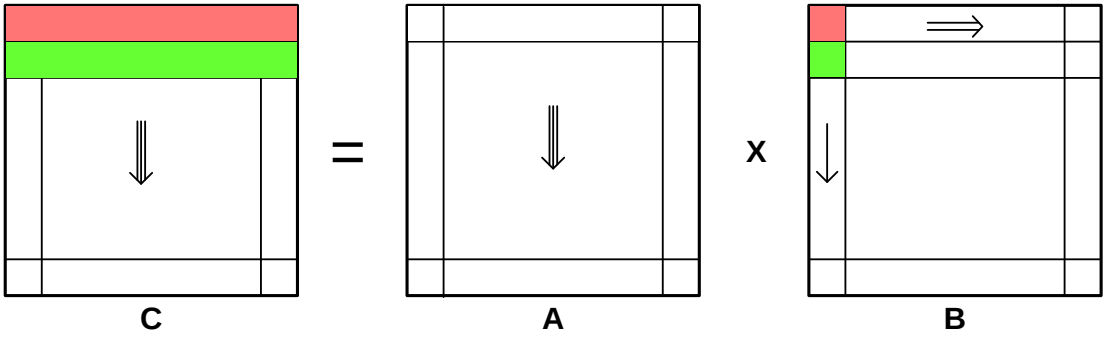
Τονίζεται ότι για την υλοποίηση των αριθμητικών πράξεων σε κάθε SPE ενδείκνυται έως και επιβάλλεται η χρήση διανυσματικών πράξεων και δεδομένων. Η χρήση διανυσμάτων, ακόμα και για ένα straight-forward πρόβλημα όπως ο πολλαπλασιασμός πινάκων, έχει κάποιες δυσκολίες ως προς τη διαμόρφωση των κατάλληλων διανυσμάτων. Στον παραπάνω κώδικα λαμβάνονται τέσσερα διανύσματα από τον πίνακα B (έστω B_0, B_1, B_2, B_3) και ένα διάνυσμα από τον πίνακα A (έστω A_X). Για κάθε στοιχείο του διανύσματος A_X δημιουργείται ένα καινούριο διάνυσμα, σε κάθε θέση του οποίου έχει αναπαραχθεί το εκάστοτε στοιχείο (προκύπτουν δηλαδή τα διανύσματα $A_0 = \{A_X[0], A_X[0], A_X[0], A_X[0]\}$, $A_1 = \{A_X[1], A_X[1], A_X[1], A_X[1]\}$, $A_2 = \{A_X[2], A_X[2], A_X[2], A_X[2]\}$ και $A_3 = \{A_X[3], A_X[3], A_X[3], A_X[3]\}$). Κάθε ένα από τα νέα αυτά διανύσματα $A_{(0, 1, 2, 3)}$ πολλαπλασιάζεται με το αντίστοιχο διάνυσμα $B_{(0, 1, 2, 3)}$. Το γινόμενο που προκύπτει προστίθεται στο υπάρχον μερικό άθροισμα του πίνακα C για τις αντίστοιχες θέσεις.

Blocks σταθερού πλήθους

Ακολουθεί η περιγραφή της διαμέρισης των πινάκων κατά την προσέγγιση των blocks σταθερού πλήθους. Ο πρώτος από αυτούς, η «Μεταφορά μεγάλων κομματιών του A και λίγων στοιχείων του B», είναι ο τρόπος που χρησιμοποιείται και για τη διαμέριση του προβλήματος μεταξύ των κόμβων της συστοιχίας, όπως περιγράφεται στην αντίστοιχη ενότητα.

Μεταφορά μεγάλων κομματιών του A και λίγων στοιχείων του B

Ένας άλλος τρόπος χωρισμού του προβλήματος είναι ο εξής: από τον πίνακα A μεταφέρεται σε κάθε SPE ένα block όσο μεγαλύτερο γίνεται ώστε να χωράει στην τοπική μνήμη. Το block αυτό είναι διαφορετικό για κάθε SPE. Επίσης, μεταφέρονται σε κάθε βήμα της επανάληψης από τον πίνακα B τόσα στοιχεία όσες και οι γραμμές που περιλαμβάνει το block από τον A. Οι πίνακες διατρέχονται κατά τη φορά που δείχνουν τα βέλη.



Σε ψευδοκώδικα το παραπάνω σχήμα υλοποιείται όπως φαίνεται παρακάτω. Ο ψευδοκώδικας αυτός δίνεται για να φανούν οι μεταφορές δεδομένων και οι σειρά με την οποία διατρέχονται οι πίνακες. Ο ίδιος ο πολλαπλασιασμός, έτσι όπως φαίνεται στον ψευδοκώδικα, είναι βαθμωτός και όχι διανυσματικός. Η vectorized μορφή είναι αρκετά πιο πολύπλοκη και, δεδομένου ότι αυτός ο τρόπος διαχωρισμού υστερεί φανερά, όπως επεξηγείται παρακάτω, έναντι του της μεθόδου «Διαίρεση σε blocks 64 x 64» και δεν μελετήθηκε περαιτέρω, όλη η ανάλυση δίνεται για θεωρητικούς σκοπούς. Η προσέγγιση αυτή έχει χρησιμοποιηθεί για τη διαμέριση των πινάκων σε επίπεδο συστοιχίας.

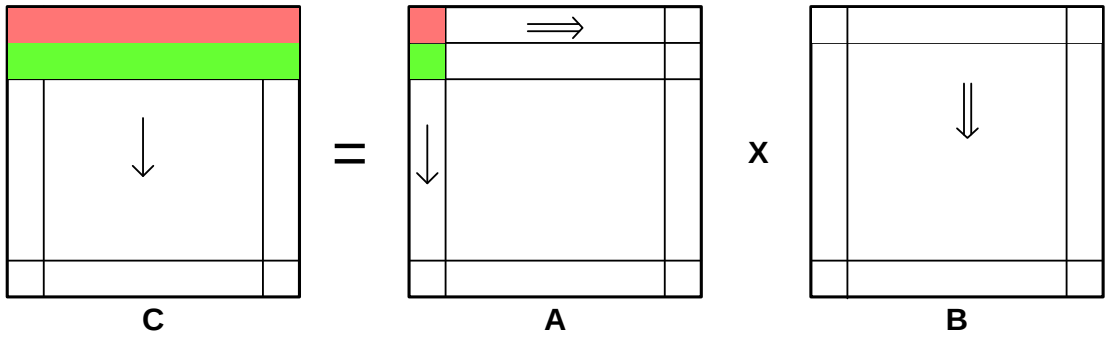
```
/*
  spe_id = αύξων αριθμός SPE
  matrix_i_dim = πλήθος γραμμών του πίνακα A και C
  matrix_k_dim = πλήθος στηλών του πίνακα A και C
  spes = πλήθος SPE που δουλεύουν
  Κάθε SPE αναλαμβάνει από τους A και C (matrix_i_dim / spes) γραμμές, που ξεκινάνε από τη γραμμή
                spe_id * (matrix_i_dim / spes) του εκάστοτε πίνακα
  A_lines = πλήθος γραμμών του A που μεταφέρονται κάθε φορά = πλήθος γραμμών του C
  blocks_A = πλήθος γραμμών του A / A_lines
  columns = πλήθος στηλών του B
  rows = πλήθος γραμμών του B
*/



for kk = 0 to blocks_A
{
  load A_block[kk]
  load C_block[kk]

  for j = 0 to columns
    for i = 0 to rows
    {
      load B[i][j]

      for k=0 to A_lines
        C_block[kk][j + k*matrix_k_dim] += A_block[kk][j + k*matrix_k_dim] * B[i][j]
    }

  store C_block[j]
}
```

Με αυτό τον τρόπο τα στοιχεία των πινάκων A και C μεταφέρεται μόνο μια φορά προς και από τις τοπικές μνήμες. Και σε αυτή τη περίπτωση είναι αρκετά εύκολη η παραλληλοποίηση του αλγορίθμου διότι κάθε επεξεργαστική μονάδα λαμβάνει διαφορετικές γραμμές του πίνακα A και εξάγει σαν αποτέλεσμα τις αντίστοιχες γραμμές του πίνακα C. Τα συγκεκριμένα αποτελέσματα του πίνακα C προκύπτουν «κατ' ευθείαν», δηλαδή χωρίς να παράγονται ενδιάμεσα μερικά (partial) αθροίσματα. Κάτι τέτοιο κάνει δυνατό το σχηματισμό του αποτελέσματος για ένα block του C από ένα και μόνο SPE, συμβάλλοντας σημαντικά στην συνάφεια και την συνέπεια του προβλήματος χωρίς περαιτέρω κώδικα για έλεγχο και συγχρονισμό μεταξύ των SPEs.

Το μειονέκτημα της συγκεκριμένης υλοποίησης είναι ότι τα στοιχεία του πίνακα B μεταφέρονται πολλές φορές στις τοπικές μνήμες των SPEs. Επίσης, ο κώδικας του πολλαπλασιασμού των blocks δεν είναι αποδοτικός, καθώς κάθε στοιχείο του B πολλαπλασιάζεται με μία στήλη του A. Καθώς οι πίνακες είναι αποθηκευμένοι κατά γραμμή, τα στοιχεία της στήλης του A δεν είναι γειτονικά στη μνήμη και απαιτούνται πολλοί κύκλοι ρολογιού ώστε να αναδιαταχθούν και να σχηματίσουν ένα διάνυσμα. Τέλος, στην πλειοψηφία των περιπτώσεων, τα στοιχεία που μεταφέρονται κάθε φορά από τον πίνακα B δεν αρκούν για να καλύψουν το μέγεθος μίας cacheline της CBE, γεγονός που καθιστά τις μεταφορές DMA για τα στοιχεία αυτά μη αποδοτικές.

Μεταφορά μεγάλων κομματιών του B και λίγων στοιχείων των A και C

Μια τρίτη μέθοδος διαμέρισης του προβλήματος είναι η συμμετρική της προηγούμενης μεθόδου: από τον πίνακα B μεταφέρεται σε κάθε SPE ένα block όσο μεγαλύτερο γίνεται ώστε να χωράει στην τοπική μνήμη. Το block αυτό είναι διαφορετικό για κάθε SPE. Επίσης, μεταφέρονται σε κάθε βήμα της επανάληψης από τον πίνακα A τόσα στοιχεία όσες και οι γραμμές που περιλαμβάνει το block από τον B. Τα στοιχεία του A πολλαπλασιάζονται με το block του B ενώ ο πίνακας A διατρέχεται κατά στήλες, όπως φαίνονται και από τα βέλη του παρακάτω σχήματος.



Σε ψευδοκώδικα το παραπάνω σχήμα υλοποιείται όπως φαίνεται παρακάτω. Όπως στην προηγούμενη περίπτωση, ο ψευδοκώδικας αυτός δίνεται σε απλή, μη vectorized μορφή μόνο για να φανούν οι μεταφορές δεδομένων και οι σειρά με την οποία διατρέχονται οι πίνακες.

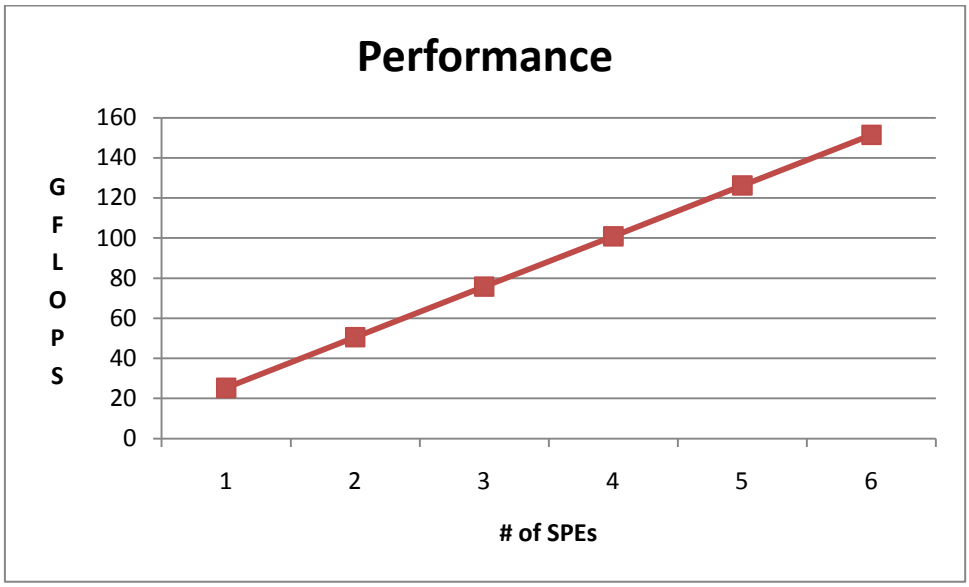
```
/*
  B_lines = πλήθος γραμμών του B που μεταφέρονται κάθε φορά
  blocks_B = πλήθος γραμμών του B / B_lines
  rows = πλήθος γραμμών του A = πλήθος γραμμών του C
*/

for kk = 0 to blocks_B
{
  load B_block[kk]
  for i = 0 to rows
  {
    load από τη γραμμή i του A τα στοιχεία που βρίσκονται
        σε στήλες αντίστοιχες των γραμμών του B_block[kk] (συνολικά B_lines τω πλήθος στοιχεία)
    load τη γραμμή i του C

    for k = 0 to B_lines
      C[i] += A[i][k] * B_block[k] (το εκάστοτε στοιχείο του A πολλαπλασιάζεται με όλη την γραμμή του B)

    store C[i]
  }
}
```

Ένα βασικό πλεονέκτημα της συγκεκριμένης μεθόδου είναι ότι ελαχιστοποιεί τις μεταφορές δεδομένων. Αυτό επιτυγχάνεται λόγο του ότι τα στοιχεία των πινάκων A και B μεταφέρονται μόνο μία φορά στην τοπική μνήμη και τα στοιχεία που μεταφέρονται είναι γειτονικά στη κύρια μνήμη, πράγμα που σημαίνει ότι δεν χρειάζονται μεταφορές λίστας DMA και το overhead που αυτές συνεπάγονται.

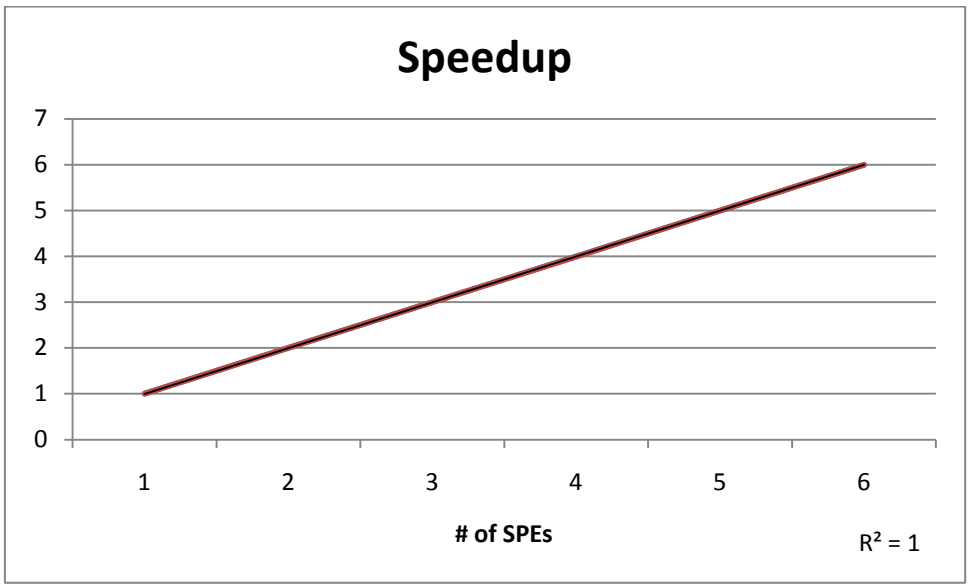
Όμως, κάθε στοιχείο του πίνακα αποτελέσματος προκύπτει σαν άθροισμα πράξεων που γίνονται σε διαφορετικά σημεία του κώδικα και σε παράλληλες αρχιτεκτονικές δύο ή περισσότεροι επεξεργαστές μπορεί να προσπαθούν να εξάγουν ενδιάμεσα αποτελέσματα για το ίδιο block του C. Σε μια τέτοια περίπτωση πρέπει να υπάρξουν έλεγχοι και συγχρονισμοί μεταξύ των SPEs καθώς και η διαδικασία κατά την οποία κάποιο από όλα τα SPEs θα συγκεντρώσει και θα αθροίσει όλα τα μερικά αποτελέσματα που έχουν προκύψει για κάθε ένα από τα blocks του C. Άρα, η προσέγγιση αυτή δυσχεραίνει την παραλληλοποίηση. Επίσης, στην πλειοψηφία των περιπτώσεων, τα στοιχεία που μεταφέρονται κάθε φορά από τον πίνακα A δεν αρκούν για να καλύψουν το μέγεθος μίας cacheline της CBE, γεγονός που καθιστά τις μεταφορές DMA για τα στοιχεία αυτά μη αποδοτικές. Μπορεί η μέθοδος αυτή να περιορίζει τις μεταφορές δεδομένων, που είναι η αχίλλειος πτέρνα των σημερινών επεξεργαστών σε μία παράλληλη αρχιτεκτονική, επιφέρει όμως σημαντικές δυσκολίες στην υλοποίηση και ενδεχομένως να αποτύχει εν συγκρίσει με άλλες μεθόδους, όπως επιβεβαιώθηκε και από τα πειραματικά αποτελέσματα.

Μετρήσεις σε επίπεδο ενός κόμβου

Οι μετρήσεις για την αξιολόγηση της εφαρμογής πολλαπλασιασμού πινάκων σε έναν κόμβο περιλαμβάνουν μετρήσεις της επίδοσης συναρτήσει του αριθμού των χρησιμοποιούμενων SPEs και μετρήσεις της επίδοσης συναρτήσει του μεγέθους των πινάκων.

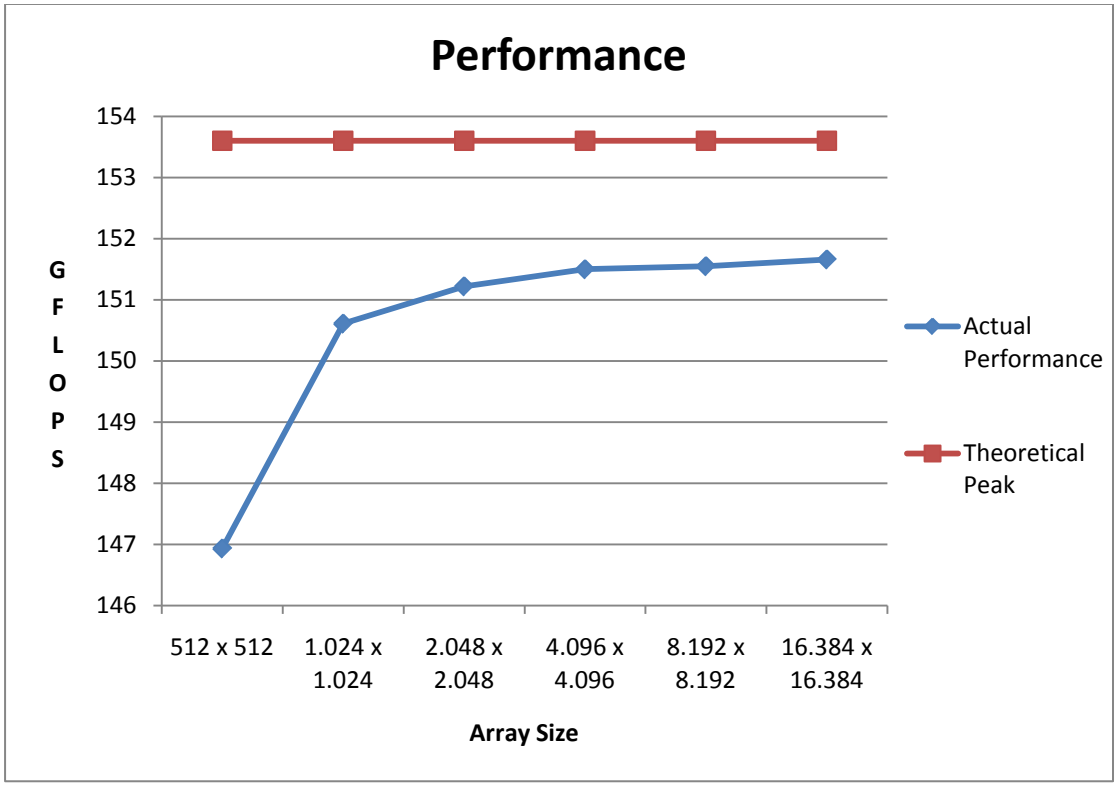


Πιο συγκεκριμένα, μετρήθηκε η επίδοση της εφαρμογής χρησιμοποιώντας 1, 2, 3, 4, 5 και 6 SPEs για μέγεθος πινάκων εισόδου 4.096 x 4.096. Η επίδοση για αυτή τη σειρά μετρήσεων φαίνεται στο παραπάνω διάγραμμα. Σε κάθε περίπτωση, ο χρόνος που χρειάζεται κάθε SPE για τη εκτέλεση της εφαρμογής αφιερώνεται σχεδόν αποκλειστικά στο υπολογιστικό κομμάτι, ενώ οι καθυστερήσεις λόγω επικοινωνίας με την κύρια μνήμη είναι αμελητέες. Συγκεκριμένα, το ποσοστό του χρόνου που αφιερώνεται στους υπολογισμούς κυμαίνεται από 99.38% έως 99.59%.



Το υψηλό ποσοστό αξιοποίησης του επεξεργαστή και η απουσία καθυστερήσεων λόγω επικοινωνίας εξηγούν την πλήρη γραμμικότητα του διαγράμματος επιτάχυνσης. Ο συντελεστής συσχέτισης μεταξύ της καμπύλης μετρήσεων και της ευθείας που προκύπτει από τη γραμμικοποίηση των μετρήσεων ισούται με τη μονάδα. Θεωρητικά και με την προϋπόθεση ότι το μέγεθος των πινάκων είναι αρκούντως μεγάλο η γραμμική επιτάχυνση μπορεί να συνεχιστεί για αρκετά μεγάλο αριθμό επεξεργαστικών μονάδων. Έτσι, επιβεβαιώνεται η φύση του προβλήματος του πολλαπλασιασμού πινάκων ως μίας εφαρμογής κατάλληλης για αποδοτική παραλληλοποίηση.

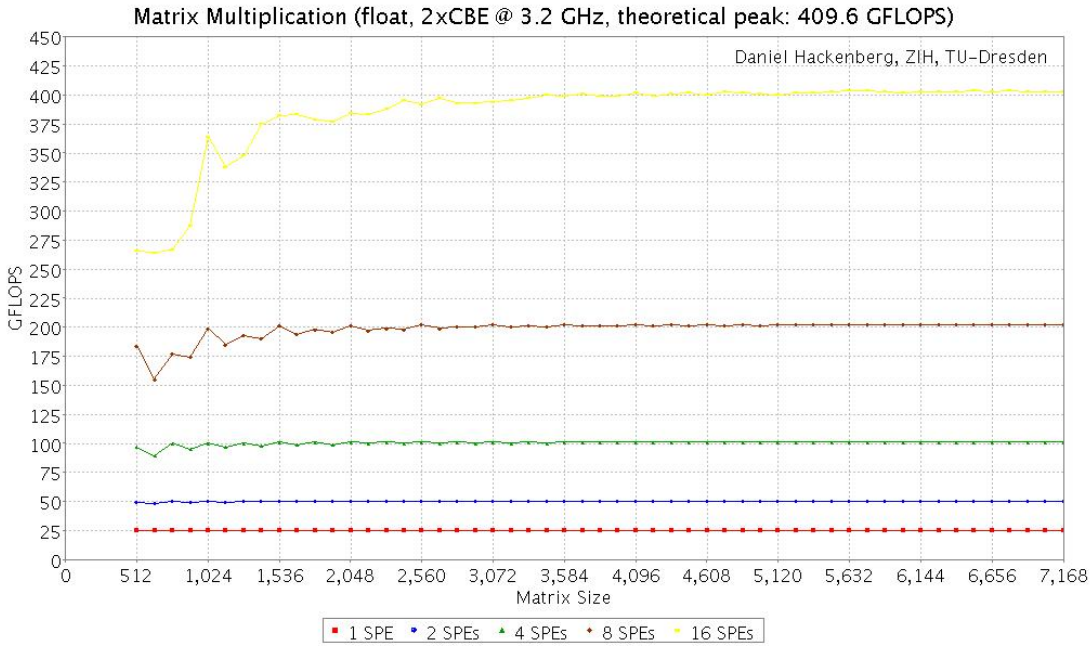
Ακολουθεί μία δεύτερη σειρά μετρήσεων της επίδοσης της εφαρμογής σε έξι SPEs, συναρτήσει του μεγέθους των πινάκων εισόδου. Συγκεκριμένα, μελετήθηκαν τα μεγέθη 512 x 512, 1.024 x 1.024, 2.048 x 2.048, 4.096 x 4.096, 8.192 x 8.192 και 16.384 x 16.384.



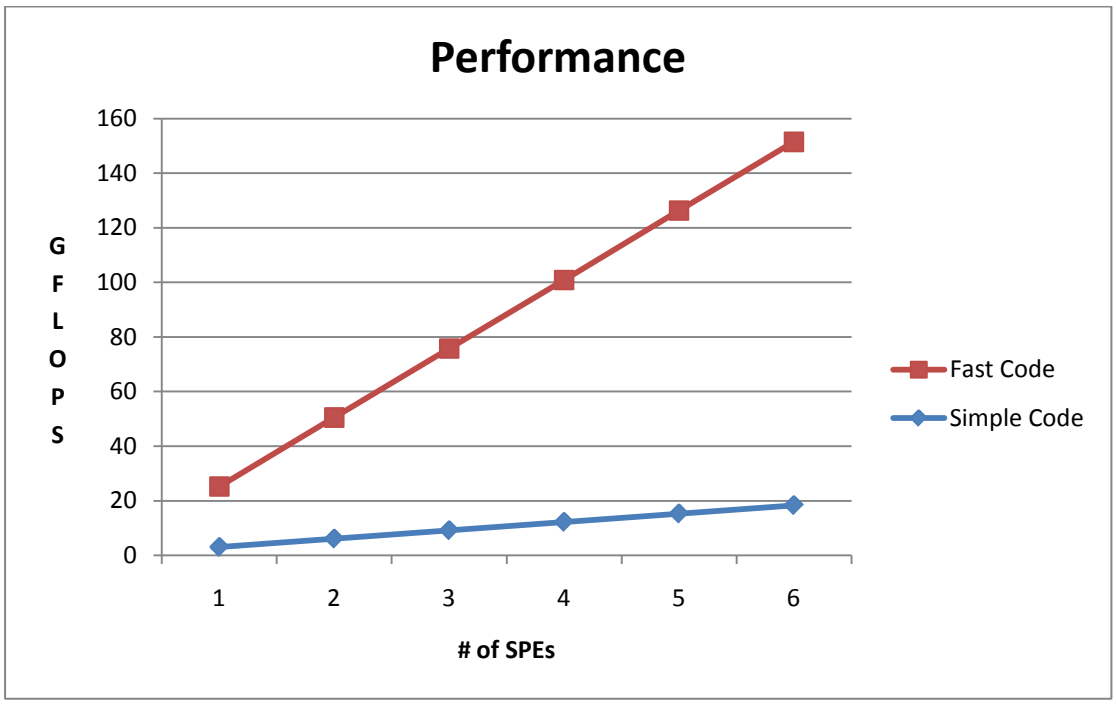
Η χαμηλή επίδοση στις μικρές διαστάσεις οφείλεται στο γεγονός ότι σχηματίζονται σχετικά λίγα blocks. Σε μεγαλύτερες διαστάσεις η επίδοση τείνει ασυμπτωτικά στα 152GFLOPS, νούμερο που υπολείπεται ελάχιστα του θεωρητικού μεγίστου της Cell Broadband Engine για έξι SPEs, το οποίο είναι 25.6GFLOPS x 6 = 153.6GFLOPS.

Ακολουθεί το διάγραμμα επίδοσης μίας υλοποίησης του πολλαπλασιασμού πινάκων του φοιτητή Daniel Hackenberg από το Τεχνικό Πανεπιστήμιο της Δρέσδης (Technische Universität Dresden), η οποία είναι πανομοιότυπη με την τρέχουσα υλοποίηση, με μόνη

διαφορά ότι ο κώδικας για τον πολλαπλασιασμό των blocks έχει γραφτεί εξ ολοκλήρου σε assembly αντί για C.



Όπως προκύπτει από τη σύγκριση των δύο εκδόσεων, η επίδοση είναι σχεδόν η ίδια, με διαφορές μικρότερες του 0.6%. Αποδεικνύεται, έτσι, η καλά υλοποιημένη δυνατότητα για προγραμματισμό χαμηλού επιπέδου με χρήση των intrinsics της C/C++ που παρέχονται στον προγραμματιστή.



Πρέπει να σημειωθεί, ωστόσο, ότι τόσο ο παρών κώδικας όσο και ο κώδικας του Daniel Hackenberg είναι κώδικες μεγάλης έκτασης, με χειροκίνητη αναδιάταξη και scheduling εντολών, εξάλειψη διακλαδώσεων, inlining συναρτήσεων και ξεδίπλωμα βρόχων. Συγκριτικά, η χρήση ενός πιο απλού κώδικα και συγκεκριμένα του κώδικα C που παρατίθεται στην περιγραφή του αλγορίθμου επιτυγχάνει την επίδοση που φαίνεται στο παραπάνω διάγραμμα. Η διαφορά των δύο υλοποιήσεων πλησιάζει τη μία τάξη μεγέθους, οπότε οποιαδήποτε σύγκριση μεταξύ τους δεν έχει νόημα.

Πολλαπλασιασμός πινάκων σε επίπεδο συστοιχίας από CBEs

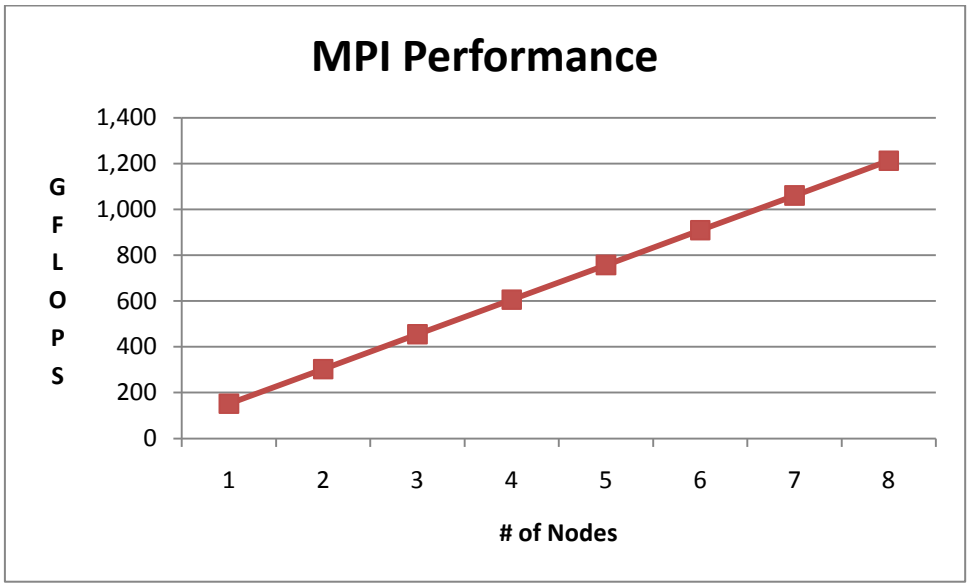
Στη υλοποίηση του προβλήματος πολλαπλασιασμού πινάκων σε επίπεδο συστοιχίας από Cell Broadband Engines ακολουθήθηκε η εξής στρατηγική ως προς τον χωρισμό των πινάκων: ο πίνακας A χωρίζεται κατά γραμμές σε τόσα τμήματα όσοι και οι κόμβοι που θα χρησιμοποιούνται, αποδίδοντας σε κάθε κόμβο ένα ξεχωριστό κομμάτι, όπως ακριβώς περιγράφεται στη μέθοδο χωρισμού των πινάκων «Μεταφορά μεγάλων κομματιών του A και λίγων στοιχείων του B». Κάθε κόμβος, λοιπόν, χρησιμοποιεί έναν υποπίνακα του «αρχικού πίνακα A» ως τον «δικό του πίνακα A», έστω A_n, n = αύξων αριθμός του κόμβου, και όλον τον αρχικό πίνακα B. Με αυτήν την προσέγγιση κάθε κόμβος αναλαμβάνει τον αποκλειστικό υπολογισμό τμημάτων του πίνακα C, έστω C_n, αντίστοιχων των τμημάτων του πίνακα A που αναλαμβάνει κάθε κόμβος. Από εκεί και πέρα, εσωτερικά σε κάθε κόμβο πραγματοποιείται πολλαπλασιασμός μεταξύ των πινάκων A_n και B προς παραγωγή του αποτελέσματος C_n με τη μεθοδολογία «Διαίρεση σε blocks 64 x 64», όπως ακριβώς θα γινόταν και στην περίπτωση του ενός κόμβου.

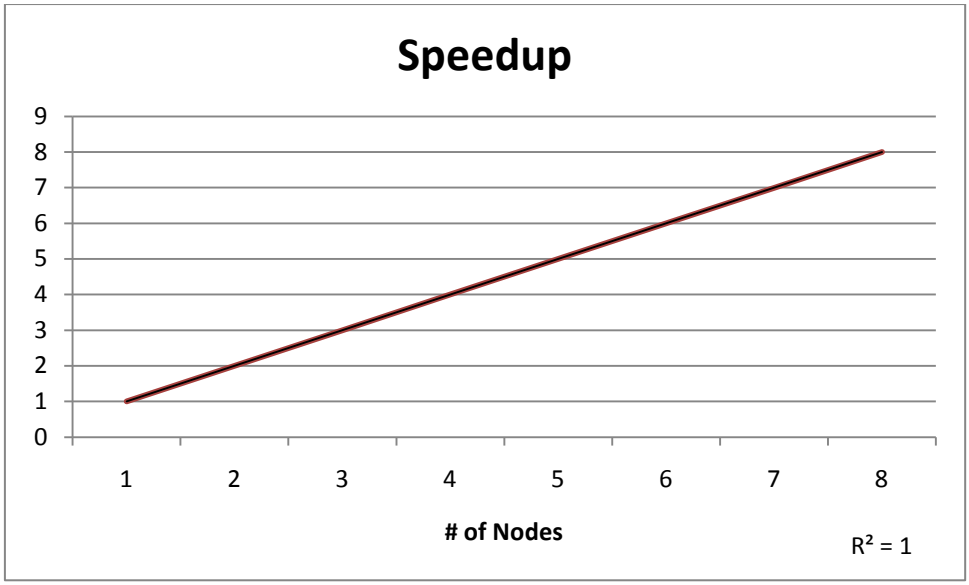
Με αυτό τον τρόπο είναι πολύ εύκολη η παραλληλοποίηση του προβλήματος χωρίς ιδιαίτερα πολύπλοκους μηχανισμούς ελέγχου και συγχρονισμού, αφού οι υπολογισμοί σε κάθε κόμβο γίνονται ανεξάρτητα από τους άλλους κόμβους, όπως οι υπολογισμοί σε κάθε SPE γίνονταν ανεξάρτητα από τα άλλα SPEs.Οι μεταφορές δεδομένων μεταξύ κόμβων στο προγραμματιστικό περιβάλλον MPI γίνονται με μικρή ταχύτητα και μεγάλη καθυστέρηση συγκριτικά με τις ταχύτητες και το εύρος ζώνης εσωτερικά της Cell Broadband Engine. Με αυτό τον τρόπο χωρισμού του προβλήματος, δηλαδή με στατικό εξ αρχής διαμοιρασμό των πινάκων, οι μεταφορές αυτές γίνονται οι ελάχιστες δυνατές και συγκεκριμένα το μόνο που απαιτείται είναι μία αποστολή στην αρχή της εκτέλεσης των πινάκων A_n και B από τον root κόμβο προς τους υπολοίπους και μία αποστολή στο τέλος της εκτέλεσης από κάθε κόμβο προς τον root κόμβο των υποπινάκων C_n.

Μετρήσεις σε επίπεδο συστοιχίας

Οι μετρήσεις για την αξιολόγηση της εφαρμογής πολλαπλασιασμού πινάκων σε επίπεδο συστοιχίας περιλαμβάνουν μετρήσεις της επίδοσης συναρτήσει του αριθμού των χρησιμοποιούμενων κόμβων και μετρήσεις της επίδοσης συναρτήσει του μεγέθους των

πινάκων. Πιο συγκεκριμένα, μετρήθηκε η επίδοση της εφαρμογής χρησιμοποιώντας 1, 2, 3, 4, 5, 6, 7 και 8 κόμβους, με έξι SPEs ανά κόμβο, για μέγεθος πινάκων εισόδου 16.384 x 16.384. Η επίδοση για αυτή τη σειρά μετρήσεων φαίνεται στο παρακάτω διάγραμμα.

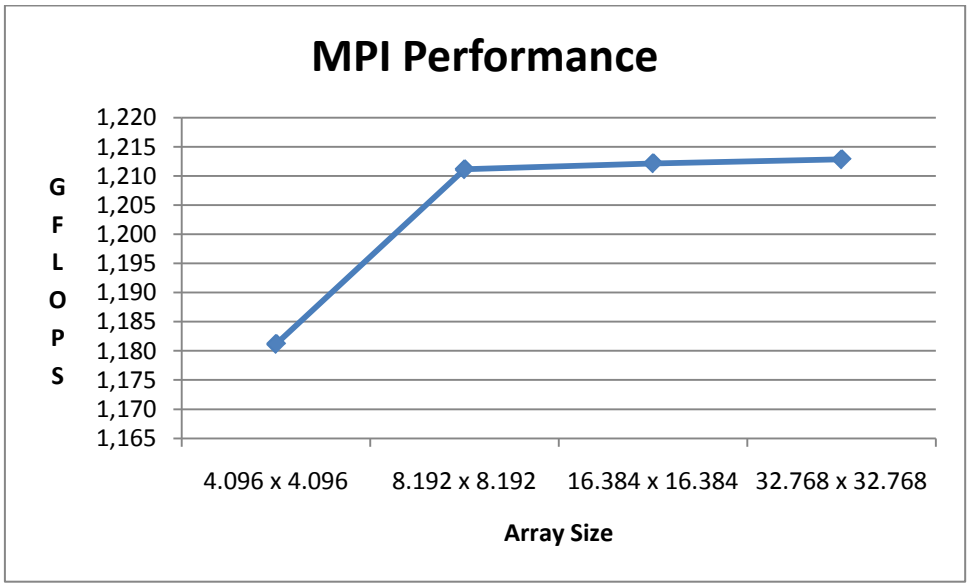




Σε κάθε περίπτωση, ο χρόνος που χρειάζεται κάθε κόμβος για τη εκτέλεση της εφαρμογής αφιερώνεται αποκλειστικά στο υπολογιστικό κομμάτι, αφού το μοίρασμα των πινάκων γίνεται στατικά και δεν υπάρχουν επικοινωνίες μεταξύ των κόμβων κατά τη διάρκεια της εκτέλεσης. Για το λόγο αυτό η επιτάχυνση συναρτήσει του αριθμού των κόμβων είναι γραμμική, όπως φαίνεται και από το συντελεστή συσχέτισης που ισούται με τη μονάδα.

Ακολουθεί μία δεύτερη σειρά μετρήσεων της επίδοσης της εφαρμογής σε οκτώ κόμβους, συναρτήσει του μεγέθους των πινάκων εισόδου. Συγκεκριμένα, μελετήθηκαν τα

μεγέθη 4.096 x 4.096, 8.192 x 8.192, 16.384 x 16.384 και 32.768 x 32.768, χρησιμοποιώντας έξι SPEs ανά κόμβο.



Όπως σε επίπεδο SPEs, στους μικρούς πίνακες υπάρχει μία πτώση της επίδοσης, η οποία όμως τείνει ασυμπτωτικά για μεγαλύτερα μεγέθη πινάκων στα 1.215GFLOPS, τιμή ελάχιστα χαμηλότερη από το θεωρητικό μέγιστο για οκτώ κόμβους, που είναι τα 153.6GFLOPS x 8 = 1.228,8GFLOPS.

Συμπεράσματα

Ο πολλαπλασιασμός πινάκων είναι ένα πρόβλημα με μεγάλη υπολογιστική πολυπλοκότητα και χωρίς εξαρτήσεις δεδομένων. Ως επακόλουθο παραλληλοποιείται εύκολα και επιτυγχάνει υψηλή επίδοση τόσο στο πλαίσιο μίας CBE όσο και στο πλαίσιο συστοιχίας. Η πλήρης επικάλυψη υπολογισμών και επικοινωνίας έχει ως αποτέλεσμα την πλήρη γραμμικότητα στα διαγράμματα της επιτάχυνσης και για το λόγο αυτό το υπολογιστικό πλέγμα για αυτήν την εφαρμογή μπορεί να επεκταθεί επ' άπειρον. Παρ' όλα αυτά, για την επίτευξη των παραπάνω επιδόσεων είναι αναγκαία η χειροκίνητη βελτιστοποίηση του κώδικα και η εφαρμογή των τεχνικών που έχουν αναφερθεί (multi-buffering, vectorization, loop unrolling).

Κεφάλαιο 6

ΕΞΙΣΩΣΗ ΔΙΑΧΥΣΗΣ

Διάχυση είναι το φυσικό φαινόμενο της διακίνησης και διασποράς κάποιου φυσικού μεγέθους (π.χ. θερμοκρασία, ηλεκτρομαγνητικά πεδία κ.τ.λ.) από περιοχές υψηλότερης συγκέντρωσης προς τις περιοχές χαμηλότερης συγκέντρωσης. Τα φαινόμενα διάχυσης μελετούνται σε διάφορες επιστήμες όπως τη φυσική (π.χ. διάχυση θερμότητας), τη χημεία (π.χ. διάχυση υγρών και αερίων) ή τη μετεωρολογία (π.χ. διακίνηση μολυσμένων σωματιδίων στην ατμόσφαιρα) και άλλες.

Η διάχυση μοντελοποιείται με τη χρήση της εξίσωσης διάχυσης, η οποία αποτελεί μία μερική διαφορική εξίσωση που περιγράφει την κίνηση ενός βαθμωτού μεγέθους (π.χ. θερμότητα) καθώς αυτό κινείται μέσα σε ένα γνωστό πεδίο (π.χ. το υλικό στο οποίο διαχέεται η θερμότητα). Η εξίσωση αυτή εκφράζεται μαθηματικά ως εξής:

$$\frac{\partial v}{\partial t} = \vec{a}\nabla \mathrm{v}\,,$$

όπου $\vec{a}$ είναι το διάνυσμα του πεδίου.

Η εξίσωση διάχυσης σε δύο διαστάσεις

Σε δύο χωρικές διαστάσεις η παραπάνω διαφορική εξίσωση γίνεται:

$$\frac{\partial v}{\partial t} = a_x \frac{\partial v}{\partial x} + a_y \frac{\partial v}{\partial y}\,.$$

Για να μελετήσουμε τη διάχυση σε ένα χωρίο X × Y για χρονικό παράθυρο T διαμερίζουμε το χωρίο σε ένα διακριτό και ομοιόμορφο πλέγμα. Για τη διακριτοποίηση χρησιμοποιείται χρονικό βήμα Δt και χωρικά βήματα Δx και Δy. Η διαφορική εξίσωση μπορεί μετά να διαμεριστεί σε διακριτά κομμάτια χρησιμοποιώντας πεπερασμένα σχήματα, όπως το σχήμα Euler – Forward, στο οποίο η χρονική παράγωγος αντικαθίσταται από το κλάσμα διαφορών

$$\frac{\partial v}{\partial t} = \frac{v_{ij}^{n+1} - v_{ij}^{n}}{\Delta t}\,.$$

Η φυσική του προβλήματος επιτρέπει την εφαρμογή διαφορικών σχημάτων τύπου upwind για τις χωρικές μερικές παραγώγους, τα οποία σχήματα περιλαμβάνουν υπολογισμούς με

«προηγούμενα» χωρικά σημεία του πλέγματος. Έτσι, οι χωρικές μερικές παράγωγοι μπορούν να αντικατασταθούν με την έκφραση

$$\frac{\partial v}{\partial x} = \frac{v_{ij}^{n} - v_{(i-1)j}^{n}}{\Delta x}.$$

Τελικά, υποθέτοντας ομοιόμορφο υπολογιστικό πλέγμα στις δύο χωρικές διαστάσεις, δηλαδή $\Delta x = \Delta y$ και $a_x = a_y = a$, η διαφορική εξίσωση στις δύο χωρικές διαστάσεις γίνεται

$$v_{ij}^{n+1} = \left(1 + 2a\frac{\Delta t}{\Delta x}\right) v_{ij}^{n} - a\frac{\Delta t}{\Delta x}\left(v_{(i-1)j}^{n} + v_{i(j-1)}^{n}\right).$$

Τα μεγέθη v_{ij}^{0}, v_{0j}^{n}, v_{i0}^{n} είναι γνωστά από τις αρχικές και συνοριακές συνθήκες του προβλήματος της μερικής διαφορικής εξίσωσης.

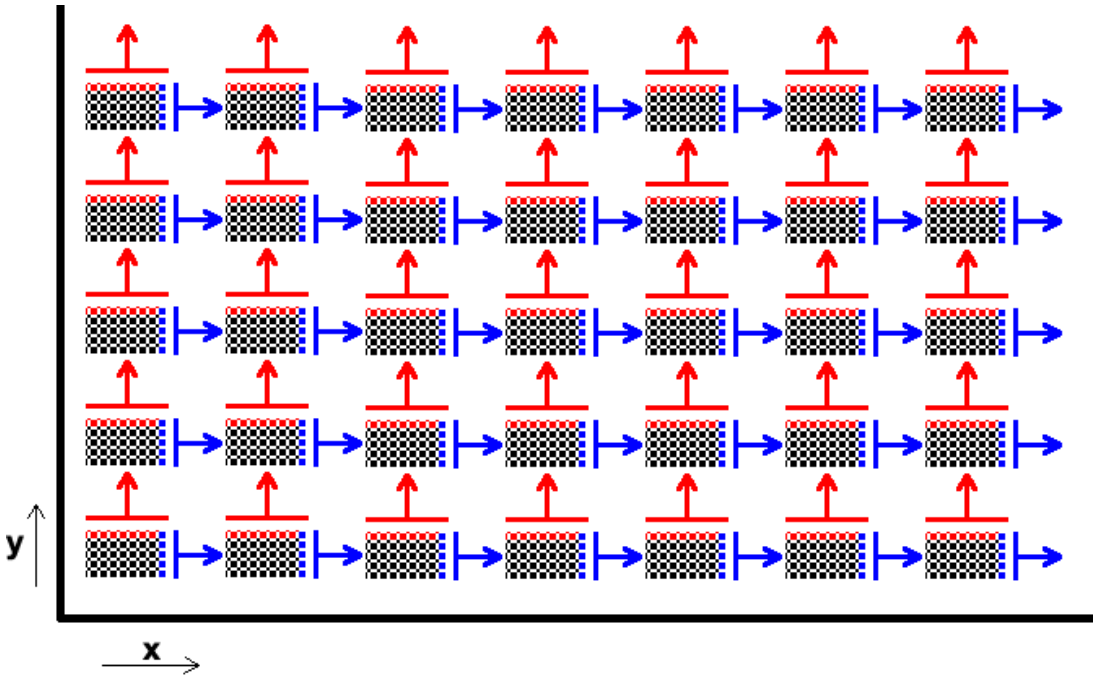
Το πρόβλημα στη διακριτή του μορφή είναι ένα πρόβλημα αριθμητικής ανάλυσης, πράγμα που σημαίνει ότι για την εξαγωγή κάποιου αποτελέσματος που συγκλίνει στην πραγματική λύση απαιτείται να γίνει ένας ικανός αριθμός επαναλήψεων των υπολογισμών πάνω στο χωρίο. Ο αριθμός αυτός των επαναλήψεων εξαρτάται από τις απαιτήσεις για την ακρίβεια της σύγκλισης, το μέγεθος του χωρίου, τον αλγόριθμο που χρησιμοποιείται για τους υπολογισμούς κ.τ.λ.. Έχουν χρησιμοποιηθεί δύο αλγόριθμοι, οι οποίοι αναλύονται παρακάτω. Πρόκειται για τους αλγορίθμους "in-place" και "out-of-place".

Διαμέριση του προβλήματος στα SPEs – επίπεδο ενός κόμβου

Βάση της πρότερης ανάλυσης του προβλήματος και της μεθοδολογίας αντιμετώπισης του πολλαπλασιασμού πινάκων προκύπτουν ιδέες και χρήσιμα συμπεράσματα για την εφαρμογή της εξίσωσης διάχυσης. Η διαμέριση του προβλήματος της διάχυσης προϋποθέτει κατ' αρχάς τον χωρισμό του πίνακα-χωρίου σε blocks. Το μέγεθος και το σχήμα τους εξαρτάται από το χρησιμοποιούμενο αλγόριθμο, όπως αναλύεται παρακάτω. Προκειμένου να αξιοποιηθεί η δυνατότητα της CBE για double-buffering των δεδομένων πρέπει να ακολουθηθεί ο κατάλληλος τρόπος μεταφοράς των blocks από και προς την κύρια μνήμη. Τέλος, ο κώδικας του αλγορίθμου πρέπει να υλοποιηθεί και να αναπτυχθεί με τέτοιον τρόπο ούτως ώστε να είναι δυνατή η αξιοποίηση των ετερογενών pipelines και η χρήση διανυσματικών πράξεων και τύπων δεδομένων.

Ο χωρισμός ενός χωρίου σε blocks απεικονίζεται στο παρακάτω σχήμα. Συνέπεια του χωρισμού είναι η δημιουργία εξαρτήσεων μεταξύ γειτονικών blocks. Αντίθετα από τον πολλαπλασιασμό πινάκων, στην εφαρμογή της εξίσωσης διάχυσης πρέπει να υπάρχει αποστολή και λήψη συνοριακών τιμών (συνόρων) κατά μήκος κάθε διάστασης των blocks, δηλαδή και οριζοντίως και καθέτως. Η απαιτούμενες μεταφορές συνόρων φαίνονται επίσης

στο παρακάτω σχήμα. Οι μεταφορές αυτές υποδεικνύουν και τις εξαρτήσεις που δημιουργούνται.

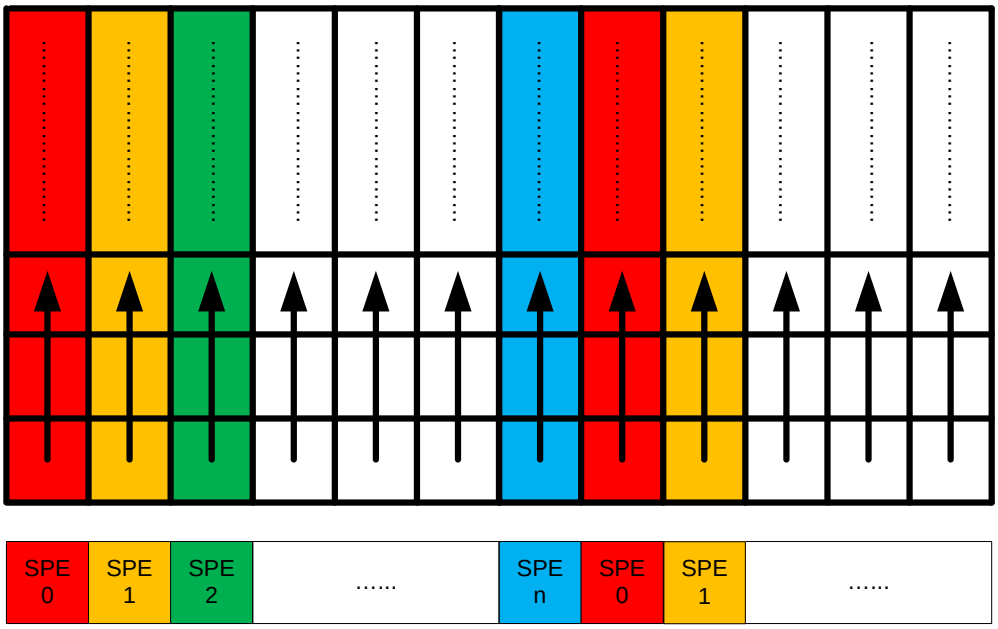


Για τον υπολογισμό του κάθε block, πλην των οριακών περιπτώσεων, κατά την t-οστή επανάληψη απαιτούνται οι συνοριακές τιμές των δύο γειτονικών του blocks (του «κάτω» και του «αριστερού», έτσι όπως δίνονται στο σχήμα) από την ίδια επανάληψη.

Άλλη μία εξάρτηση για τον υπολογισμό ενός block κατά την t-οστή επανάληψη προκύπτει από την απαίτηση για την ύπαρξη των τιμών του ίδιου block από την (t-1)-οστή επανάληψη. Άρα, εκτός από τις «οριζόντιες» και «κάθετες» εξαρτήσεις, που μπορούμε να πούμε ότι αποτελούν «χωρικές εξαρτήσεις» ή «εξαρτήσεις στις χωρικές διαστάσεις», υπάρχουν και οι «εγκάρσιες» (ως προς το παραπάνω σχήμα) εξαρτήσεις, δηλαδή οι εξαρτήσεις μεταξύ των επαναλήψεων, που μπορούμε να πούμε ότι αποτελούν «χρονικές εξαρτήσεις» ή «εξαρτήσεις στη χρονική διάσταση».

Από τα παραπάνω είναι φανερό ότι η ύπαρξη συνόρων και επικοινωνίας μεταξύ των επεξεργαστικών στοιχείων δυσχεραίνει την υψηλής επίδοσης παραλληλοποίηση της εφαρμογής, τόσο σε επίπεδο κώδικα όσο και σε επίπεδο λογικής.

Η υλοποίηση της διαμέρισης του χωρίου που επιλέχθηκε για την συγκεκριμένη εφαρμογή και ο τρόπος με τον οποίον ανατίθενται στα SPEs τα διάφορα blocks φαίνεται σχηματικά παρακάτω.



Η επιφάνεια χωρίζεται σε blocks, η διάσταση και η γεωμετρία των οποίων εξαρτάται από την υλοποίηση που ακολουθείται κατά περίπτωση (αλγόριθμος in-place ή αλγόριθμος out-of-place). Κάθε SPE αναλαμβάνει τον υπολογισμό ολόκληρων στηλών από blocks, δηλαδή από το πρώτο έως και το τελευταίο block της στήλης, όπως φαίνεται και στο σχήμα, το οποίο αναφέρεται στην περίπτωση που χρησιμοποιούνται n SPEs. Η ανάθεση block-στηλών στα SPEs γίνεται κυκλικά, πράγμα που σημαίνει ότι το πρώτο SPE αναλαμβάνει την πρώτη block-στήλη, το δεύτερο SPE αναλαμβάνει τη δεύτερη στήλη, …, το n-οστό SPE αναλαμβάνει την n-οστή στήλη. Τη (n+1)-οστή στήλη την αναλαμβάνει και πάλι το πρώτο SPE κ.ο.κ.. Η επιλογή για αυτόν τον τρόπο διαμέρισης έχει να κάνει με κάποιες λεπτομέρειες που αφορούν τόσο σε περιορισμούς του υποκείμενου hardware όσο και στις ιδιαιτερότητες των SPUs της Cell Broadband Engine οι οποίες μπορούν έτσι να αξιοποιηθούν πιο αποδοτικά. Φυσικά, θα μπορούσε να είχε επιλεγεί και η συμμετρική διαμέριση, δηλαδή η διαμέριση κατά την οποία το μοίρασμα γίνεται κατά γραμμές αντί για στήλες. Το αποτέλεσμα στην επίδοση και στον κώδικα είναι ακριβώς το ίδιο.

Με αυτή τη διαμέριση οι εξαρτήσεις δεδομένων έχουν ως εξής: οι απαιτούμενες συνοριακές τιμές ενός block κατά τον κατακόρυφο άξονα μεταφέρονται από το οποιοδήποτε SPE στο επόμενό του, μετά το πέρας των υπολογισμών για αυτό το block, ενώ οι συνοριακές τιμές κατά τον οριζόντιο άξονα διατηρούνται εσωτερικά σε κάθε SPE, οπότε πρακτικά δεν συνίσταται οριζόντια εξάρτηση. Επίσης, κάθε SPE αναλαμβάνει σε κάθε επανάληψη τα ίδια ακριβώς blocks που ανέλαβε και στην προηγούμενη, οπότε εξαλείφεται και η χρονική εξάρτηση (προφανώς κάθε SPE έχει εκτελέσει ήδη την (t-1)-οστή επανάληψη για ένα block

προτού φτάσει στην t-οστή επανάληψη για το ίδιο block). Τελικά, από τα τρία είδη εξαρτήσεων απομένουν μόνο οι εξαρτήσεις στον κατακόρυφο άξονα.

Για την εκτέλεση της εφαρμογής στη Cell Broadband Engine χρησιμοποιείται κώδικας που αξιοποιεί πολλές από τις δυνατότητες της συγκεκριμένης αρχιτεκτονικής. Για τη φόρτωση και αποθήκευση των blocks του πίνακα-χωρίου χρησιμοποιείται η τεχνική double-buffering (για την υλοποίηση του αλγορίθμου in-place και για μία από τις υλοποιήσεις του αλγορίθμου out-of-place) ή quadruple-buffering (για τη δεύτερη υλοποίηση του out-of-place αλγορίθμου). Για την ανταλλαγή συνοριακών τιμών χρησιμοποιείται η τεχνική triple-buffering για τα σύνορα κατά μήκος του κατακόρυφου άξονα, ανεξαρτήτως αλγορίθμου. Η επικοινωνία μεταξύ των SPEs για το συγχρονισμό και την ανταλλαγή συνοριακών τιμών γίνεται με τη χρήση τόσο των mailboxes όσο και του signal-notification.

Ο τρόπος με τον οποίον εκτελείται κάθε επανάληψη πάνω στο χωρίο είναι αυτός που περιγράφεται από τον ακόλουθο ψευδοκώδικα. Ο ψευδοκώδικας αυτός αφορά στο παράλληλο πρόγραμμα, δηλαδή τρέχει ταυτόχρονα σε όλα τα SPEs. Οι συγχρονισμοί μεταξύ των SPEs «κρύβονται» μέσα στις συναρτήσεις του ψευδοκώδικα receive_borders και send_borders. Δίνεται η έκδοση η οποία χρησιμοποιεί την τεχνική double-buffering για τα blocks του πίνακα-χωρίου.

```
/*
  spe_id: αύξων αριθμός SPE
  spe_threads: πλήθος των SPEs που λειτουργούν
  i_blocks: πλήθος blocks μίας στήλης
  j_blocks: πλήθος blocks στον οριζόντιο άξονα
  buffer[0/1]: buffers για τα blocks
  border[0/1]: buffers για τις συνοριακές τιμές
  load_block(i, j): συνάρτηση που φορτώνει το block με συνεταγμένες (i, j)
  store_block: συνάρτηση που αποθηεύει το block
  receive_borders: συνάρτηση που λαμβάνει τις απαιτούμενες συνοριακές τιμές από
          το προηγούμενο SPE, όταν αυτές οι τιμές είναι πλέον διαθέσιμες
  send_borders: συνάρτηση που στέλενει στο επόμενο SPE τις απαιτούμενες συνοριακές
          τιμές, όταν αυτό μπορεί να τις λάβει
*/

  j = spe_id;

  load_block(0, j) --> buffer[0]
  receive_borders(spe_id - 1) --> border[0]

  while(j < j_blocks)
  {
    i = 0

    while(++i < i_blocks - 1)
    {
      load_block(i, j) --> buffer[i % 2]
      receive_borders(spe_id - 1) --> border[i % 3]

      calculate(buffer[(i - 1) % 2], borders[(i - 1) % 3])
```

```
      store_block <-- buffer[(i - 1) % 2]
      send_borders(spe_id + 1)
    }

    j += spe_threads

    if(j < j_blocks)
    {
      load_block(0, j) --> buffer[(i_blocks - 1) % 2]
      receive_borders(spe_id - 1) --> border[(i_blocks - 1) % 3]
    }

    calculate(buffer[i_blocks % 2], borders[i_blocks % 3])

    store_block <-- buffer[i_blocks % 2]
    send_borders(spe_id + 1)
  }
```

In-place αλγόριθμος για την εξίσωση διάχυσης

Περιγραφή του αλγορίθμου

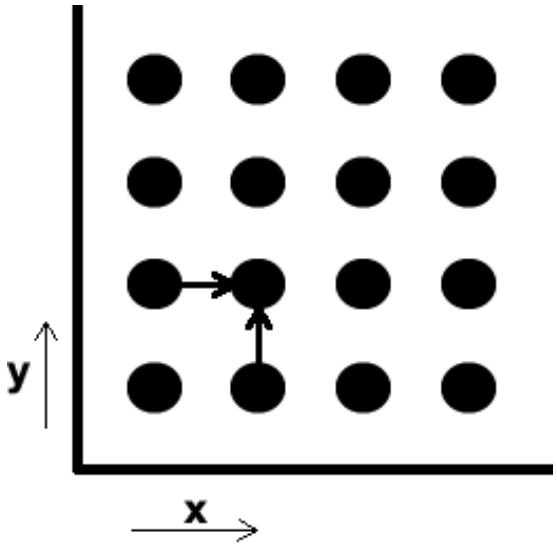
In-place αλγόριθμος για την εξίσωση διάχυσης καλείται ο αλγόριθμος εκείνος κατά τον οποίον ο υπολογισμός ενός σημείου U[t][y][x] εξαρτάται από τις τιμές U[t-1][y][x], U[t][y-1][x] και U[t][y][x-1]. Ο in-place αλγόριθμος σε μορφή ψευδοκώδικα για την διακριτή μερική διαφορική εξίσωση της διάχυσης είναι ο ακόλουθος.

```
while(loops < MAXLOOPS)
{
  t = (++loops)%2;

  for(y = 1; y < Y; y++)
    for(x = 1; x < X; x++)
      U[t][y][x] = (1+2*a*dt/dx)*U[(t-1)%2][y][x] -  a*dt/dx*(U[t][y-1][x] + U[t][y][x-1]);
}
```

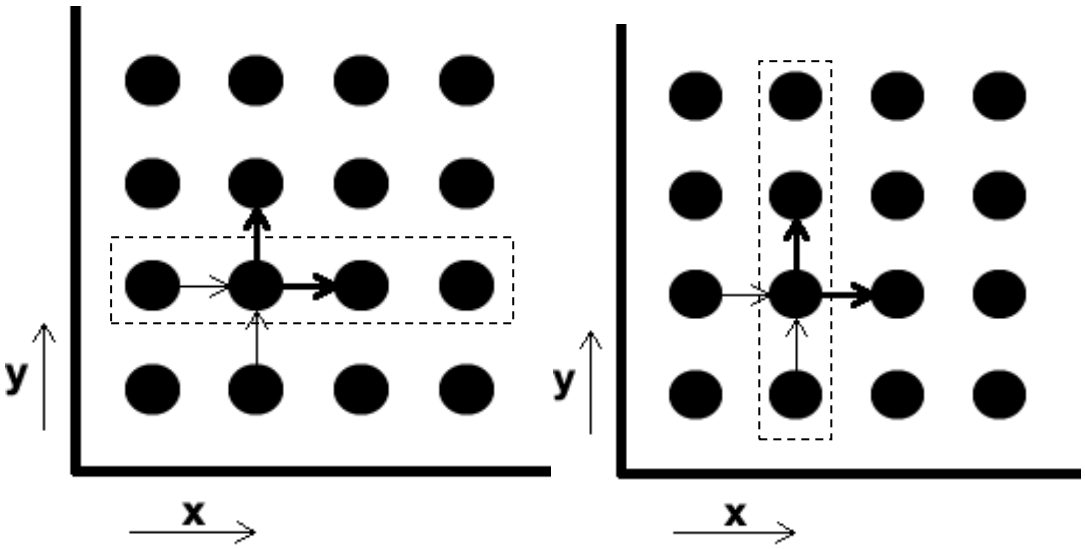
Γνώμονας περαιτέρω διερεύνησης της υλοποίησης του αλγορίθμου είναι η χρήση διανυσματικών πράξεων και τύπων δεδομένων, όπως και η αποδοτικότερη δυνατή αξιοποίηση των ετερογενών pipelines. Για την εφαρμογή αυτών των χαρακτηριστικών πρέπει να ληφθούν ορισμένες ειδικές σχεδιαστικές αποφάσεις.

Ειδικές σχεδιαστικές αποφάσεις

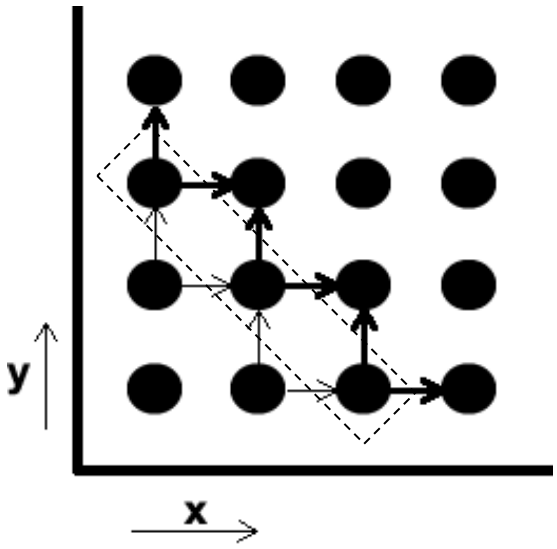
Θεωρώντας ότι η τιμή U[t-1][y][x], δηλαδή η τιμή του σημείου (x, y) του πλέγματος κατά την προηγούμενη επανάληψη του αλγορίθμου, υπάρχει για κάθε σημείο (x, y), τότε ο υπολογισμός της τρέχουσας τιμής κάθε σημείου εξαρτάται από τις τιμές των δύο προηγούμενων (ένα σε κάθε άξονα) γειτονικών σημείων του. Αυτή η εξάρτηση φαίνεται παρακάτω. Σε όλα τα σχήματα εξαρτήσεων, ένα βελάκι εισερχόμενο σε ένα σημείο υποδηλώνει ότι το σημείο αυτό εξαρτάται από το σημείο από το οποίο εξέρχεται το βελάκι.



Η εφαρμογή του παραπάνω ψευδοκώδικα παρουσιάζει το εξής πρόβλημα: είτε γίνει σάρωση του χωρίου κατά γραμμές είτε κατά στήλες (δηλαδή όποια και αν είναι η σειρά των δύο for-loop), η εξάρτηση κάθε σημείου από τις 2 ακριβώς προηγούμενες τιμές δεν επιτρέπει τη χρήση διανυσματικών (SIMD) πράξεων που παρέχει η Cell Broadband Engine. Όπως φαίνεται και σχηματικά, σε κάθε διάνυσμα υπάρχουν στοιχεία των οποίων ο υπολογισμός απαιτεί τον προγενέστερο υπολογισμό άλλων στοιχείων που επίσης βρίσκονται στο ίδιο διάνυσμα.



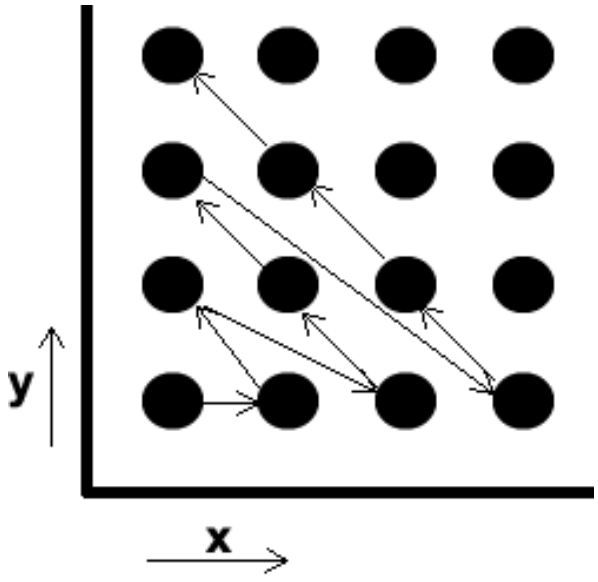
Η μη αξιοποίηση των διανυσματικών λειτουργιών της αρχιτεκτονικής της Cell Broadband Engine θα επιφέρει άμεση επίπτωση στην επίδοση του αλγορίθμου, η οποία θεωρητικά μειώνεται στο ένα τέταρτο για αναπαράσταση με αριθμούς κινητής υποδιαστολής απλής ακρίβειας και στο μισό για αριθμούς κινητής υποδιαστολής διπλής ακρίβειας, καθώς κάθε διάνυσμα στη CBE περιέχει τέσσερεις floats ή δύο doubles, αντίστοιχα.



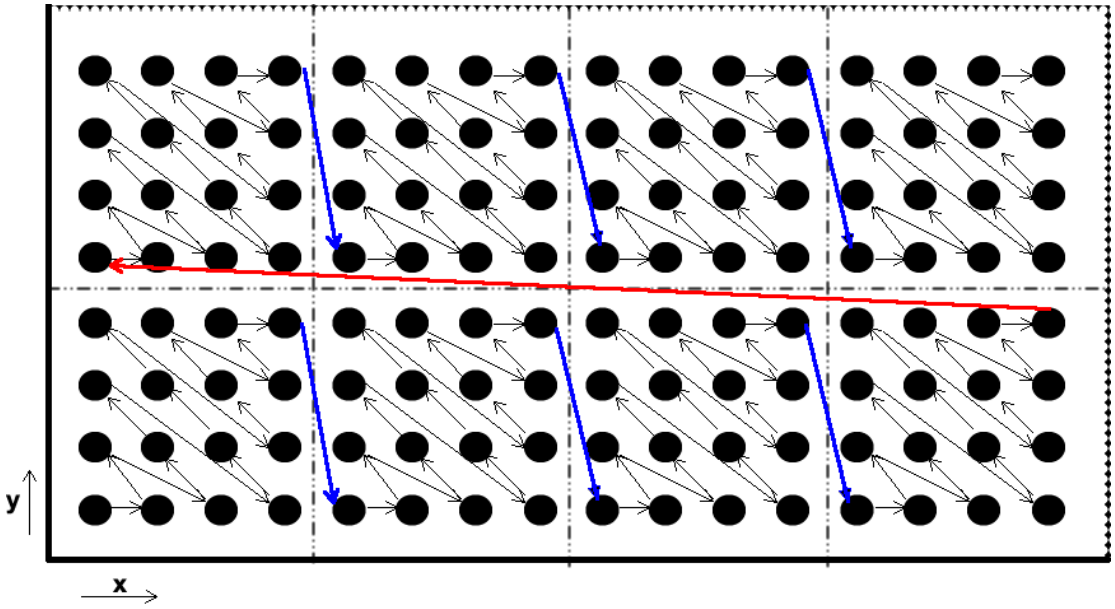
Παρατηρείται από το παραπάνω σχήμα ότι τα στοιχεία κατά μήκος μίας διαγωνίου δεν έχουν εξαρτήσεις μεταξύ τους, αλλά μπορούν να υπολογιστούν ταυτόχρονα αν έχουν ήδη υπολογιστεί τα στοιχεία της προηγούμενης διαγωνίου. Σαρώνοντας, λοιπόν, τον πίνακα κατά διαγωνίους, αντί κατά γραμμές ή κατά στήλες, είναι δυνατή η χρήση των διανυσματικών πράξεων, κάτι που θα θεωρητικά θα πολλαπλασιάσει την επίδοση.

Διασχίζοντας, όμως, τον πίνακα κατά διαγωνίους δημιουργείται ένα σημαντικό πρόβλημα. Στη γλώσσα προγραμματισμού C, όπως και στην πλειοψηφία των γλωσσών προγραμματισμού, τα στοιχεία ενός πίνακα αποθηκεύονται κατά γραμμές και σε διαδοχικές θέσεις μνήμης. Τα στοιχεία κατά μήκος μίας διαγωνίου βρίσκονται συνεπώς σε μη διαδοχικές θέσεις μνήμης και συνεπώς δεν σχηματίζουν «έτοιμο» διάνυσμα. Η δημιουργία διανυσμάτων από αυτά τα στοιχεία απαιτεί μία μεγάλη σειρά εντολών φόρτωσης / αποθήκευσης και διαμόρφωσης (rotate, shift, shuffle) οι οποίες επιβραδύνουν την επίδοση και αντισταθμίζουν μεγάλο ποσοστό από το όποιο κέρδος αποφέρει η χρήση εντολών SIMD.

Η λύση σε αυτό το πρόβλημα έρχεται υιοθετώντας μία τεχνική κατά την οποία αναδιατάσσονται στην κύρια μνήμη τα στοιχεία του χωρίου, έτσι ώστε να είναι αποθηκευμένα «κατά διαγωνίους» (ορολογία αντίστοιχη με τους όρους «κατά γραμμές» ή «κατά στήλες»), όπως φαίνεται στο παρακάτω σχήμα.

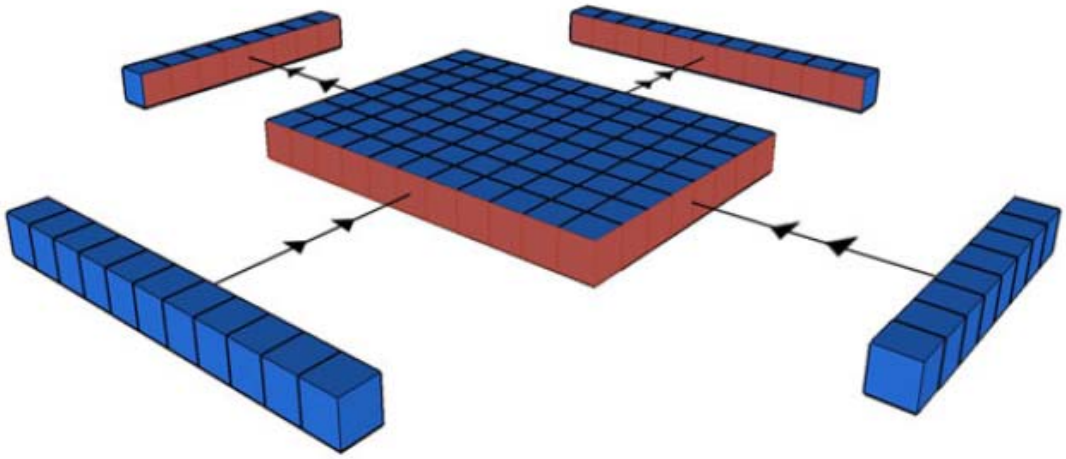


Τονίζεται ότι η αναδιάταξη των στοιχείων κατ' αυτόν τον τρόπο γίνεται σε κάθε block ξεχωριστά και όχι στο πλαίσιο όλου του πίνακα. Η διάταξη των στοιχείων στο πλαίσιο του συνολικού πίνακα φαίνεται παρακάτω.



Επιλογή του μεγέθους των blocks

Όπως έχει αναφερθεί τα διανύσματα περιέχουν τέσσερα στοιχεία κινητής υποδιαστολής απλής ακρίβειας. Για το λόγο αυτό είναι σκόπιμο να χωρίζεται ο πίνακας σε blocks, η διάσταση των οποίων είναι πολλαπλάσιο του τέσσερα. Η τοπική μνήμη της τρέχουσας υλοποίησης της Cell Broadband Engine περιορίζεται στα 256KB. Αν υποθέσουμε ότι η τοπική μνήμη κατανέμεται ισόποσα στις εντολές του προγράμματος και στα δεδομένα, υπάρχουν διαθέσιμα 128KB μνήμης για την αποθήκευση των δεδομένων του προβλήματος.

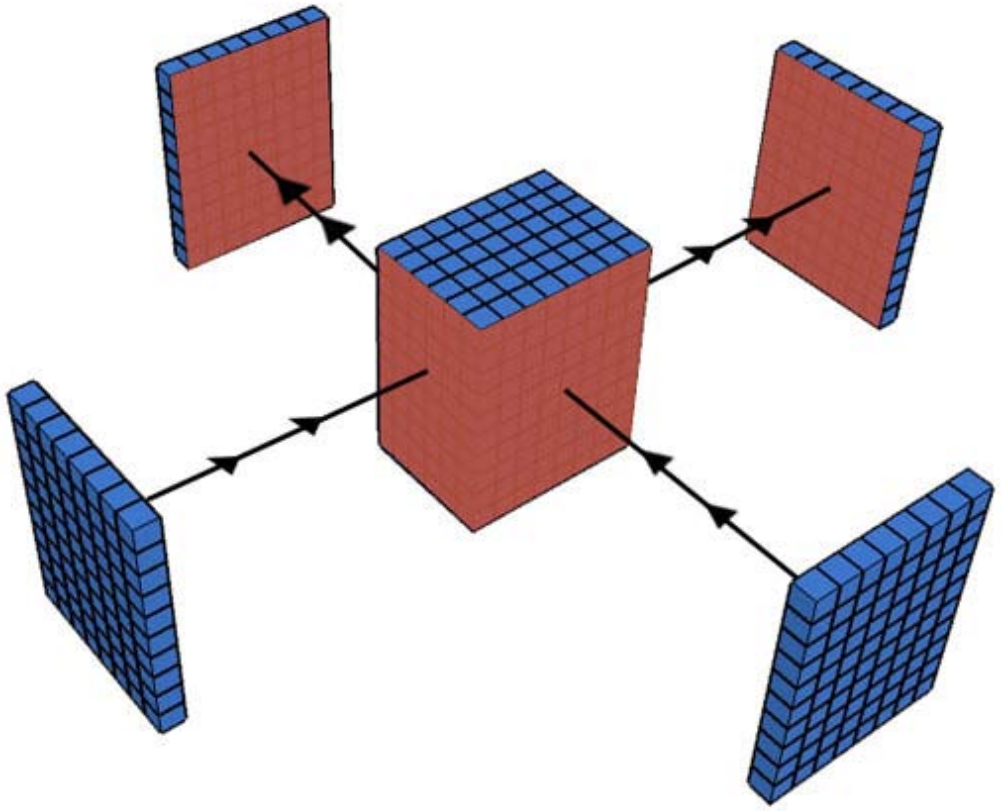


Στην περίπτωση της εξίσωσης διάχυσης ο χωρισμός των υπό επεξεργασία πινάκων γίνεται πιο πολύπλοκα από ότι στον πολλαπλασιασμό πινάκων και έχει περιγραφεί παραπάνω. Η διαδικασία υπολογισμού της διάχυσης σε κάθε block χρησιμοποιεί ως δεδομένα εισαγωγής (input data) τα εξής:

1. Τις τιμές την σημείων του χωρίου που προκύπτουν από τον υπολογισμό της προηγούμενης επανάληψης
2. Συνοριακές τιμές από τα προηγούμενα blocks τόσο ως προς τον άξονα x όσο και ως προς τον άξονα y.

Επιπρόσθετα, η εξίσωση διάχυσης επιλύεται μέσα από μια επαναληπτική διαδικασία. Αν, λοιπόν, συγκεντρωθούν σε μία «υπέρ-επανάληψη» n βήματα-επαναλήψεις για κάθε block, δηλαδή αν εφαρμοστεί tiling μεγέθους n, περιορίζεται τότε η μεταφορά δεδομένων της πρώτης κατηγορίας (τιμές του block από την προηγούμενη επανάληψη) από και προς τη μνήμη κατά n φορές.

Από την άλλη, η εφαρμογή tiling επιβάλλει τη διατήρηση μέσα στην τοπική μνήμη κάθε SPE των συνοριακών τιμών όχι μόνο για μία επανάληψη του υπολογισμού αλλά για n επαναλήψεις. Έτσι, n-πλασιάζεται το μέγεθος των buffers που χρησιμοποιούνται για την αποθήκευση των συνοριακών τιμών και απομένει λιγότερος χώρος στην τοπική μνήμη για τους buffers στους οποίους αποθηκεύονται τα blocks του πίνακα-χωρίου. Αυτό επηρεάζει άμεσα την απόφαση για το μέγεθος των blocks.



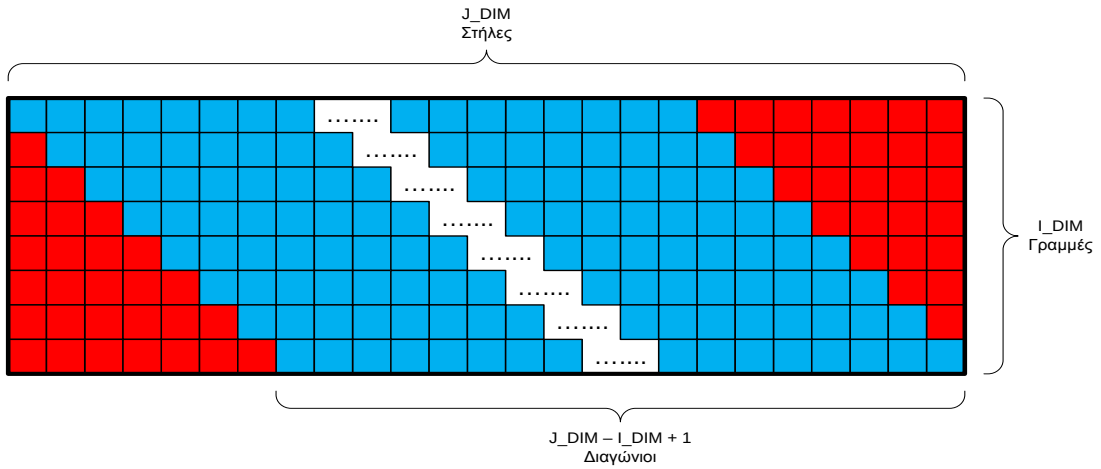
Όπως προέκυψε από τη θεωρητική ανάλυση σε συνδυασμό με μετρήσεις-πειράματα, η καλύτερη δυνατή αξιοποίηση των SPEs γίνεται με blocks μεγέθους 16KB, το οποίο μεταφράζεται σε 4.096 αριθμούς κινητής υποδιαστολής απλής ακρίβειας (ίδιο μέγεθος με την εφαρμογή του πολλαπλασιασμού πινάκων). Το μέγεθος αυτό είναι και το μέγιστο μέγεθος για μία μεταφορά DMA του MFC, οπότε μειώνεται το overhead των μεταφορών και αξιοποιείται αποδοτικότερα το εύρος ζώνης.

Επιλογή του σχήματος των blocks

Σε πρώτη προσέγγιση διερευνάται ο χωρισμός του πίνακα-χωρίου σε τετράγωνα blocks, μεγέθους 64 x 64. Η επιλογή αυτή αποδεικνύεται προβληματική. Πιο συγκεκριμένα, στον αλγόριθμο in-place επιδιώκεται η σάρωση των στοιχείων του block κατά διαγωνίους. Ένα τετράγωνο block, όμως, δεν έχει ποτέ διαγωνίους σταθερού μεγέθους. Κάθε διαγώνιος έχει ένα στοιχείο περισσότερο (από την αρχή μέχρι την κεντρική διαγώνιο του τετραγώνου) ή λιγότερο (από την κεντρική διαγώνιο του τετραγώνου μέχρι το τέλος) από την προηγούμενή της. Η ανομοιομορφία αυτή των διαγωνίων δυσκολεύει πολύ την κατασκευή αποδοτικού κώδικα.

Επίσης, ανά τέσσερεις διαγωνίους μόνο η μία έχει πλήθος στοιχείων πολλαπλάσιο του τέσσερα, ώστε να χωρίζεται ακριβώς σε διανύσματα. Στις υπόλοιπες τρεις διαγωνίους, τα στοιχεία που περισσεύουν θα πρέπει να υπολογιστούν είτε ένα-ένα (με βαθμωτό τρόπο δηλαδή, γεγονός που μειώνει την επίδοση των υπολογισμών σε αυτά τα σημεία στο ¼ είτε με κάποιον πολύπλοκο και δύσχρηστο κώδικα που θα προσπαθήσει κατά το δυνατόν να χρησιμοποιήσει διανύσματα (όπου και πάλι η επίδοση για αυτά τα σημεία δεν θα ξεπεράσει το ½ της επιθυμητής).

Η απαίτηση για διαγωνίους σταθερού μήκους σε κατά το δυνατόν μεγαλύτερη έκταση υποδεικνύει τη χρήση παραλληλεπιπέδων blocks, δηλαδή blocks με «συμπιεσμένη» τη μία διάσταση και εκτεταμένη την άλλη. Αυτή η προσέγγιση φαίνεται στο παρακάτω σχήμα.



Οι κυανές περιοχές αποτελούν τις σταθερού μεγέθους διαγωνίους, ενώ οι κόκκινες απεικονίζουν τις «προβληματικές» διαγωνίους. Εάν το block έχει διαστάσεις I_DIM x J_DIM, τότε οι κυανές διαγώνιοι ανέρχονται στις J_DIM – I_DIM + 1. Είναι προφανές ότι όσο περισσότερο συμπιεστεί η διάσταση I_DIM τόσες περισσότερες διαγώνιοι θα προκύψουν και τόσο θα ελαττωθούν οι κόκκινες, τριγωνικές περιοχές (τρίγωνα lead-in και lead-out). Για παράδειγμα, σε ένα block μεγέθους 16 x 256 οι μη πλήρεις διαγώνιοι είναι 15 + 15 = 30 ενώ οι πλήρες διαγώνιοι είναι 241. Σε ένα πιο πεπλατυσμένο block μεγέθους 4 x 1.024 οι μη πλήρεις διαγώνιοι είναι 3 + 3 = 6 ενώ οι πλήρες διαγώνιοι είναι 1.021. Όσο συμπιέζεται η μία εκ των δύο διαστάσεων, διατηρώντας σταθερό το πλήθος των στοιχείων του block, τόσο μεγαλύτερη αναλογία από τα στοιχεία αυτά θα ανήκουν σε πλήρης, κυανές διαγωνίους.

Ο κώδικας για τον υπολογισμό του block πρέπει να «σπάσει» σε τρία κομμάτια: ένα κομμάτι για τον υπολογισμό του κάτω αριστερού τριγώνου (αργός, μη αποδοτικός υπολογισμός), ένα για τον υπολογισμό του μεσαίου παραλληλογράμμου (γρήγορος, vectorized και pipelined υπολογισμός) και ένα για τον υπολογισμό του άνω δεξιού τριγώνου (αργός, μη αποδοτικός υπολογισμός).

Εδώ φαίνεται καλύτερα το πρόβλημα των τετράγωνων blocks. Ένα τετράγωνο block αποτελείται μόνο από τα lead-in και lead-out τρίγωνα και δεν υπάρχει καθόλου η κυανή περιοχή. Έτσι, ολόκληρο το block υπολογίζεται με αργό, μη αποδοτικό τρόπο και δεν γίνεται ικανοποιητική εκμετάλλευση του διανυσματικού αλγορίθμου.

Τέλος, πρέπει να αναφερθεί ότι η επιλογή να συμπιεστεί η κατακόρυφη διάσταση και όχι η οριζόντια έχει να κάνει τόσο με τον τρόπο που έχει διαμεριστεί ο πίνακας στα SPEs όσο και με την κατά γραμμές αποθήκευση ενός πίνακα στη C/C++. Τα κατακόρυφα σύνορα είναι αυτά που ανταλλάσσονται μεταξύ των SPEs, οπότε το μικρό τους μέγεθος βοηθάει στην επίδοση της εφαρμογής. Επίσης, για να μεταφερθεί ένα block απαιτούνται τόσες μεταφορές DMA, υπό μορφή λίστας DMA, όσες και οι γραμμές που απαρτίζουν το block. Συμπιέζοντας την κατακόρυφη διάσταση μειώνεται αντίστοιχα και το πλήθος των μεταφορών αυτών, παρόλο που αυξάνεται ο όγκος κάθε μεταφοράς. Κάθε μεταφορά DMA έχει ένα overhead και είναι πιο αποδοτικό να υπάρχουν λίγες και μεγάλες μεταφορές, παρά πολλές και μικρές.

Ένα στοιχείο που πρέπει να ληφθεί υπ' όψιν για τον καθορισμό των διαστάσεων του block είναι ότι τα διανύσματα αριθμών κινητής υποδιαστολής απλής ακρίβειας αποτελούνται από τέσσερα στοιχεία, οπότε, για να προκύψουν διαγώνιοι που χωρίζονται ακριβώς σε διανύσματα, πρέπει ο αριθμός των γραμμών στην συμπιεσμένη διάσταση να είναι πολλαπλάσιο του τέσσερα.

Στην επιλογή της κατακόρυφης διάστασης παίζει σημαντικό ρόλο και η λειτουργία του άρτιου pipeline της SPU. Στην τρέχουσα υλοποίηση της Cell Broadband Engine απαιτούνται

έξι κύκλοι για την ολοκλήρωση της εκτέλεσης μιας εντολής, δηλαδή από την έκδοσή της μέχρι την παραλαβή των αποτελεσμάτων της μεσολαβούν έξι κύκλοι ρολογιού. Για να μπορέσει να αξιοποιηθεί πλήρως το pipeline θα πρέπει να υπάρχει η δυνατότητα για έκδοση τουλάχιστον πέντε ακόμη εντολών, χωρίς εξαρτήσεις μεταξύ τους, προτού χρησιμοποιηθούν τα αποτελέσματα της πρώτης εντολής. Για τον υπολογισμό των στοιχείων μίας διαγωνίου πρέπει να ολοκληρωθεί ο υπολογισμός των στοιχείων της προηγούμενής της.

Η επιλογή πολύ μικρής διάστασης αυξάνει μεν την κυανή περιοχή, μειώνοντας την έκταση των προβληματικών τριγώνων, δημιουργεί όμως άλλα προβλήματα στην επίδοση του κώδικα που απορρέουν από τις εξαρτήσεις μεταξύ των διαγωνίων. Στην περίπτωση ενός block 4 x 1.024, για παράδειγμα, έχοντας μόνο ένα διάνυσμα ανά διαγώνιο, δεν μπορεί να αξιοποιηθεί η δυνατότητα της SPU για pipeline των εντολών κινητής υποδιαστολής απλής ακρίβειας, αφού πρέπει πρώτα να ολοκληρωθεί μία εντολή προτού εκδοθεί η επόμενή της, η οποία χρειάζεται τα αποτελέσματα της πρώτης. Αυτό στοιχίζει στην επίδοση του κώδικα στην κυανή περιοχή, ο οποίος γίνεται έξι φορές αργότερος από το θεωρητικό του μέγιστο. Ακόμα και με αναδιάταξη των εντολών για βελτιστοποίηση της επίδοσης, το καλύτερο αποτέλεσμα που δύναται να επιτευχθεί είναι το ¼ του θεωρητικού μεγίστου.

Στην περίπτωση του block διαστάσεων 4 x 1.024 υπάρχει και το πρόσθετο πρόβλημα της ευθυγράμμισης των δεδομένων. Με μία τέτοια κατακόρυφη διάσταση τα κόκκινα τρίγωνα του block αποτελούνται από έξι στοιχεία. Αυτό σημαίνει ότι οι κυανές διαγώνιοι θα ξεκινούν από τις θέσεις 6, 10, 14, 18, … (υποθέτοντας C-style αρίθμηση που ξεκινάει από το 0). Οι θέσεις αυτές, όμως, δεν έχουν ευθυγράμμιση διανύσματος, δηλαδή δεν είναι πολλαπλάσια του 4. Με άλλα λόγια, δεν γίνεται να φορτωθούν τα στοιχεία μίας διαγωνίου από την τοπική μνήμη σε έναν καταχωρητή με μία εντολή. Αυτό το πρόβλημα ισχύει για όλες τις κατακόρυφες διαστάσεις που είναι περιττά πολλαπλάσια του 4 (δηλαδή 4, 12, 20, 28 κ.τ.λ.).

Μία λύση θα ήταν η μεταφορά των δύο τελευταίων στοιχείων του block (που ούτως ή άλλως δεν ανήκουν στις κυανές περιοχές) στην αρχή του, ούτως ώστε τα μη διαγώνια στοιχεία της lead-in περιοχής να ξεκινούν από τη δεύτερη θέση του block και να καταλαμβάνουν μέχρι και την έβδομη θέση (υποθέτοντας C-style αρίθμηση που ξεκινάει από το 0). Αυτή η ανακατανομή των στοιχείων περιπλέκει πολύ τον κώδικα του αλγορίθμου.

Μία καλύτερη επιλογή για την κατακόρυφη διάσταση των blocks είναι οι οκτώ γραμμές, οπότε προκύπτουν blocks διαστάσεων 8 x 512. Σε αυτήν την περίπτωση, τα κόκκινα τρίγωνα περιλαμβάνουν 28 στοιχεία έκαστο, τιμή που είναι και σχετικά μικρή εν συγκρίσει με το συνολικό αριθμό στοιχείων του block και πολλαπλάσιο του τέσσερα. Με αυτό τον τρόπο, χωρίς να αλλοιωθεί πολύ η αναλογία κυανών – κόκκινων περιοχών του block, εξαλείφεται το πρόβλημα της ευθυγράμμισης. Γενικά, οποιοδήποτε άρτιο πολλαπλάσιο του 4, δηλαδή οποιοδήποτε πολλαπλάσιο του 8, δεν παρουσιάζει προβλήματα ευθυγράμμισης.

Η επιλογή του block 8 x 1.024 βελτιώνει και το θέμα της επίδοσης του κώδικα. Πλέον, κάθε διαγώνιος αποτελείται από δύο ανεξάρτητα μεταξύ τους διανύσματα, τα οποία μπορούν να αξιοποιηθούν ταυτόχρονα. Εκ πρώτης όψεως η επίδοση του κώδικα στην κυανή περιοχή ανέρχεται στο 1/3 του θεωρητικού μεγίστου και με μια αναδιάταξη των εντολών αυξάνεται περεταίρω στο ½ του θεωρητικού μεγίστου.

Η ελάχιστη κατακόρυφη διάσταση για την πλήρη αξιοποίηση του pipeline της SPU είναι οι 24 γραμμές, δηλαδή οι κυανές διαγώνιοι θα περιέχουν 24 στοιχεία που σχηματίζουν έξι ανεξάρτητα μεταξύ τους διανύσματα. Το ίδιο αποτέλεσμα, χωρίς προβλήματα ευθυγράμμισης, επιτυγχάνεται και με την επιλογή 16 γραμμών για την κατακόρυφη διάσταση και ταυτόχρονα με πολύ λεπτομερή και προσεκτική αναδιάταξη εντολών στον κώδικα. Οι παραπάνω διαστάσεις, όμως, ήδη χαλάνε αρκετά την επιθυμητή αναλογία κυανών και κόκκινων περιοχών του block.

Λαμβάνοντας υπ' όψιν όλα τα παραπάνω, η χρυσή τομή για την επιλογή των διαστάσεων των blocks βρίσκεται στα blocks 8 x 512. Αυτή είναι η διάσταση που χρησιμοποιήθηκε για την κατασκευή του κώδικα και για τις μετρήσεις της επίδοσης, τόσο σε επίπεδο μονής Cell Broadband Engine όσο και σε επίπεδο συστοιχίας.

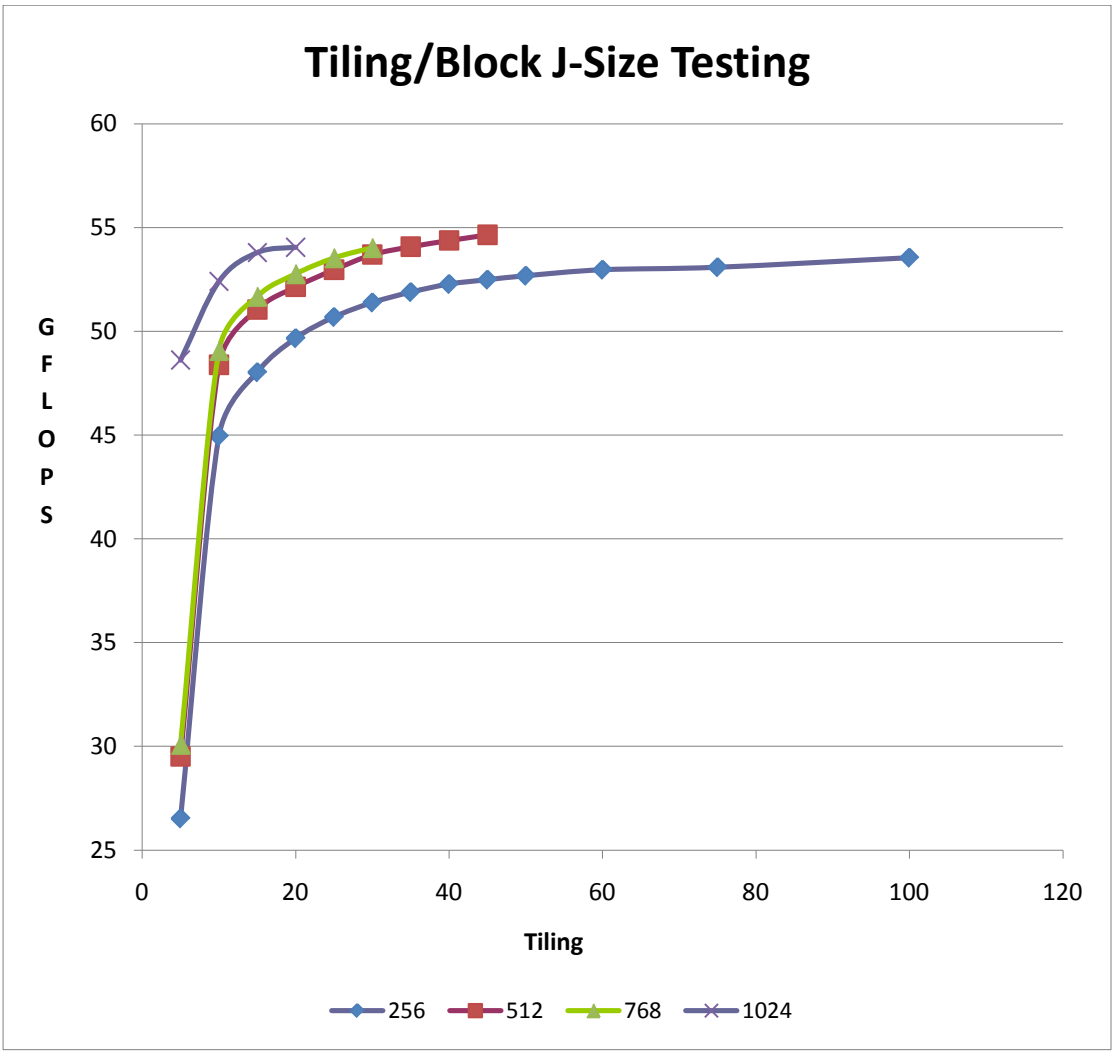
Έχοντας καθορίσει το μέγεθος και το σχήμα των blocks, ο υπολογισμός της διάχυσης εντός του block υλοποιείται όπως φαίνεται από τον ακόλουθο ψευδοκώδικα.

```
/*
  const_1 = 1 + 2 * a * dt / dx
  const_2 = - a * dt / dx
  lower_vector_0, upper_vector_0,
    lower_vector_aux_1, upper_vector_aux_1,
    lower_vector_aux_2, upper_vector_aux_2,
    horizontal_border: μεταβλητές-διανύσματα
*/

calculate in a scalar way the lower-left triangle using the in-place algorithm

for i = 0 to BLOCK_J_DIM - BLOCK_I_DIM
{
  load into lower_vector_0 the lower vector of the current diagonal
  load into upper_vector_0 the upper vector of the current diagonal
  load value of lower horizontal buffer into horizontal_border

  lower_vector_0 = spu_mul(lower_vector_0, const_1)
  upper_vector_0 = spu_mul(upper_vector_0, const_1)

  load into lower_vector_aux_1 the lower vector of the previous diagonal
  load into upper_vector_aux_1 the upper vector of the previous diagonal

  lower_vector_0 = spu_madd(lower_vector_aux_1, const_2, lower_vector_0)
  upper_vector_0 = spu_madd(upper_vector_aux_1, const_2, upper_vector_0)

  lower_vector_aux_2 = spu_shuffle(lower_vector_aux_1, upper_vector_aux_1)
  upper_vector_aux_2 = spu_shuffle(upper_vector_aux_1, horizontal_border)
```

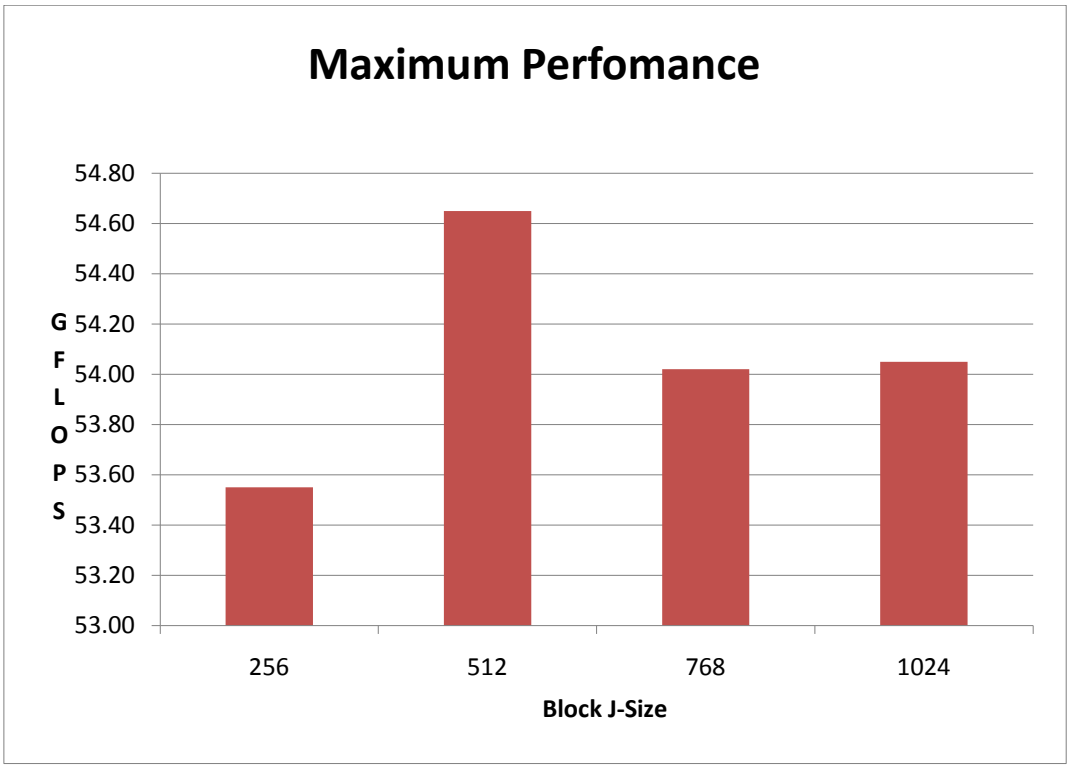
```
  lower_vector_0 = spu_madd(lower_vector_aux_2, const_2, lower_vector_0)
  upper_vector_0 = spu_madd(upper_vector_aux_2, const_2, upper_vector_0)

  horizontal_border = spu_extract(upper_vector_0, 3)

  store lower_vector_0
  store upper_vector_0
  store horizontal_border in upper horizontal buffer
}

calculate in a scalar way the upper-right triangle using the in-place algorithm
```

Μετρήσεις σε επίπεδο ενός κόμβου

Για την επιλογή του καταλληλότερου μεγέθους των blocks στην εφαρμογή της εξίσωσης διάχυσης πραγματοποιήθηκε μία σειρά μετρήσεων της επίδοσης συναρτήσει της οριζόντιας διάστασης του block. Για τους λόγους που έχουν εξηγηθεί, η κατακόρυφη διάσταση παραμένει σταθερή και ίση με οκτώ γραμμές. Συγκεκριμένα, μετρήθηκε η επίδοση της εφαρμογής για διαστάσεις block 8 x 256, 8 x 512, 8 x 768 και 8 x 1.024.

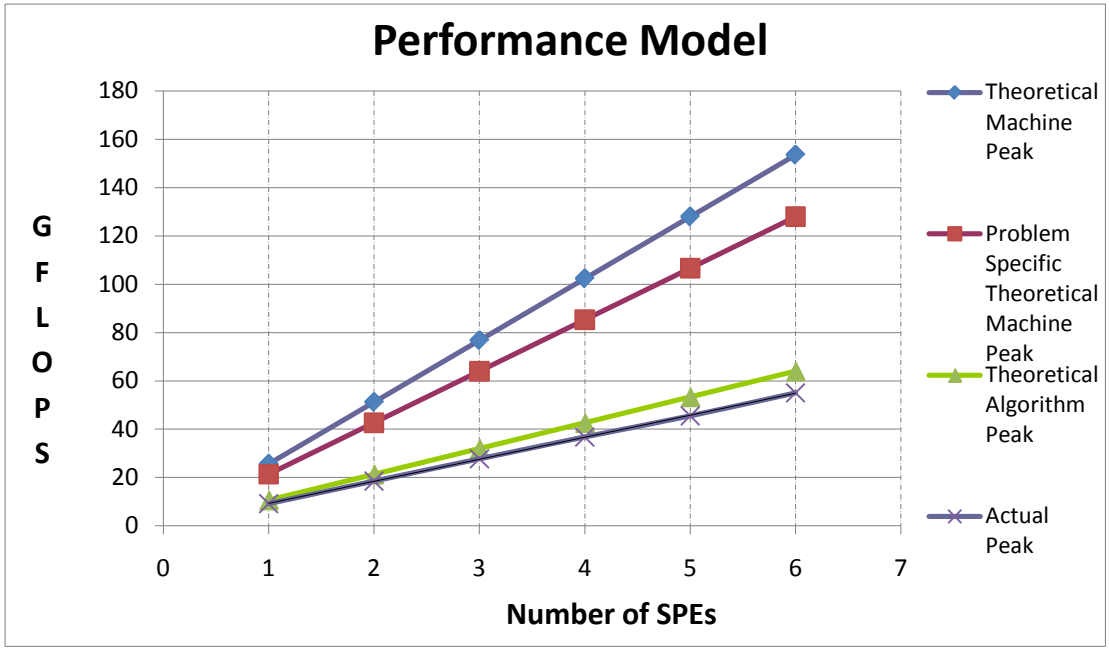


Μεγαλώνοντας την οριζόντια διάσταση των blocks αυξάνεται και το μέγεθος των συνόρων. Κατ' επέκταση αυξάνεται ο χώρος στην τοπική μνήμη τόσο για την αποθήκευση των blocks όσο και για την αποθήκευση των συνόρων. Έτσι, περιορίζεται η μέγιστη τιμή που μπορεί να πάρει το tiling μεταξύ των SPEs. Από το παραπάνω διάγραμμα προκύπτει ότι για μικρές τιμές του tiling η επίδοση της εφαρμογής ευνοείται πάρα πολύ από μέγεθος block 8 x 1.024. Συνολικά, όμως, ο καλύτερος συνδυασμός είναι το μέγεθος block 8 x 512 με tiling 45, με τον οποίον επιτυγχάνεται η υψηλότερη επίδοση από οποιονδήποτε άλλο συνδυασμό. Οι υψηλότερες τιμές της επίδοσης για κάθε μέγεθος block που εξετάστηκε φαίνονται παρακάτω.



Το μέγεθος 8 x 512 για τα blocks είναι το μέγεθος με το οποίο έγιναν όλες οι παρακάτω μετρήσεις για τον in-place αλγόριθμο, τόσο σε επίπεδο κόμβου όσο και σε επίπεδο συστοιχίας.

Ακολουθεί η περιγραφή του αναλυτικού μοντέλου επίδοσης, η οποία φαίνεται σχηματικά στο παρακάτω διάγραμμα:



Οι τέσσερεις καμπύλες απεικονίζουν τις εξής έννοιες:

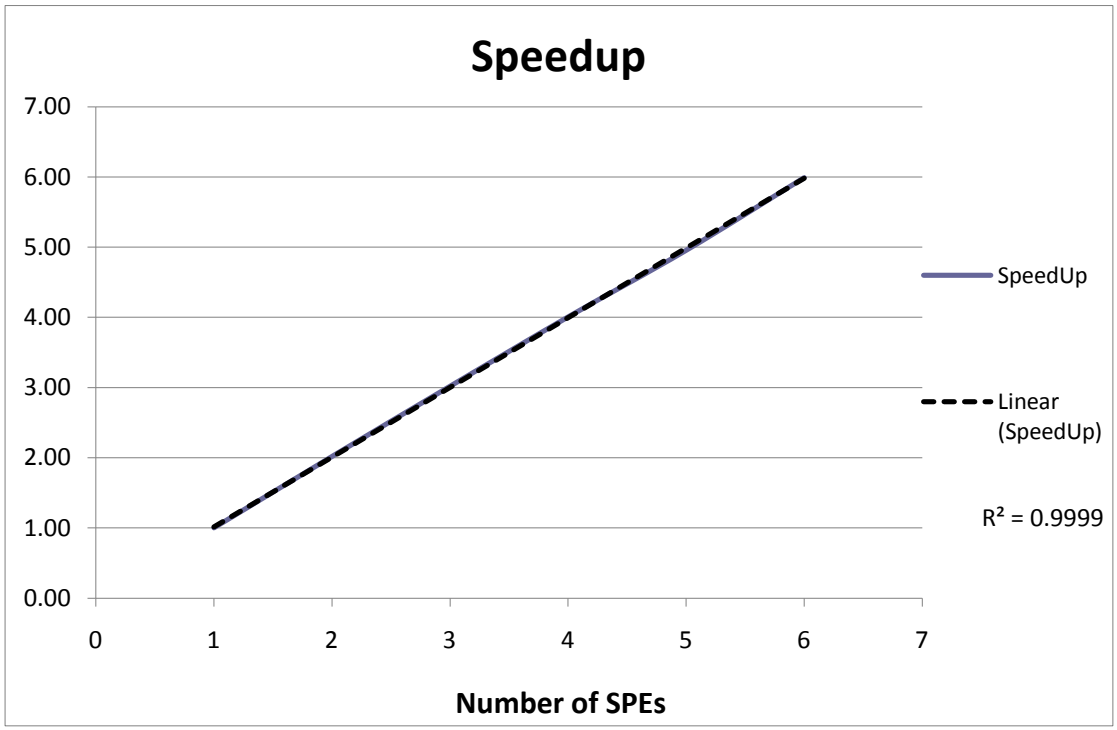
- Το Theoretical Machine Peak είναι η μέγιστη επίδοση, ανεξαρτήτως προβλήματος, που μπορεί να φτάσει ένας επεξεργαστής Cell Broadband Engine αναλόγως του πλήθους των SPEs που χρησιμοποιούνται, π.χ. για ένα SPE η μέγιστη επίδοση είναι 3.2 GHz x (1 vector-αποτέλεσμα / κύκλο) x (4 στοιχεία / vector) x (2 πράξεις κινητής υποδιαστολής [multiply-add] / στοιχείο) = 25.6 GFLOPS.

- Το Problem Specific Theoretical Machine Peak είναι η μέγιστη επίδοση στο συγκεκριμένο πρόβλημα, όπου για τον υπολογισμό ενός αποτελέσματος δεν μπορούν να χρησιμοποιηθούν μόνο πράξεις τύπου multiply-add, αλλά χρειάζονται 2 multiply-add και μία απλή multiply. Άρα στους τρεις κύκλους ρολογιού δεν εκτελούνται έξι πράξεις κινητής υποδιαστολής, αλλά πέντε. Έτσι, για ένα SPE η μέγιστη επίδοση είναι $^2/_3$ x 3,2 GHz x (1 vector-αποτέλεσμα / κύκλο) x (4 στοιχεία / vector) x (2 πράξεις κινητής υποδιαστολής / στοιχείο) + $^1/_3$ x 3,2 GHz x (1 vector-αποτέλεσμα / κύκλο) x (4 στοιχεία / vector) x (1 πράξη κινητής υποδιαστολής / στοιχείο)= 21,33 GFLOPS.

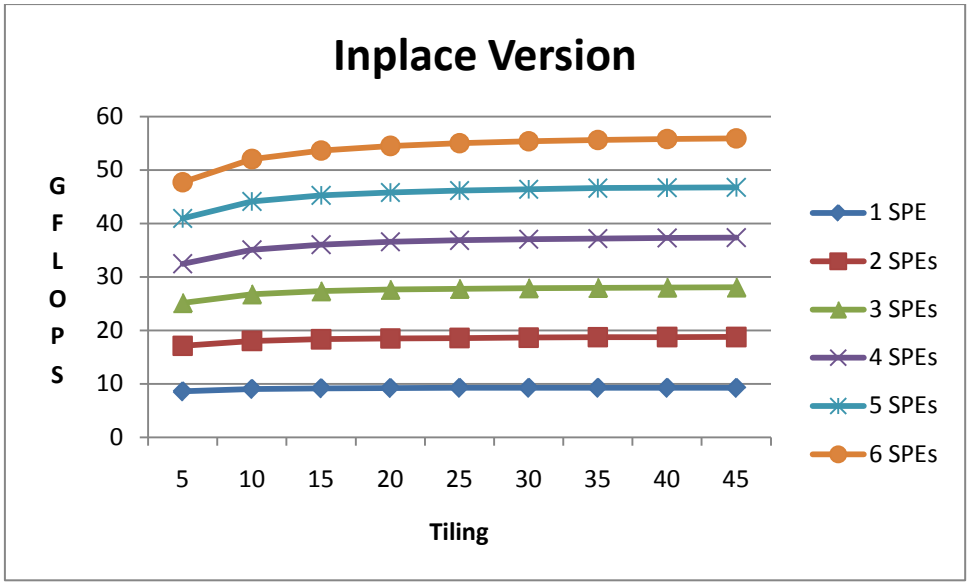
- Το Theoretical Algorithm Peak είναι το θεωρητικό μέγιστο της συγκεκριμένης υλοποίησης του αλγορίθμου. Δεδομένων των έντονων εξαρτήσεων του προβλήματος διάχυσης, ο αλγόριθμος που υλοποιήθηκε επιτυγχάνει τις πέντε πράξεις που προαναφέρθηκαν όχι σε τρεις αλλά, στην καλύτερη περίπτωση, έξι κύκλους ρολογιού. Άρα, το Theoretical Algorithm Peak είναι ακριβώς το μισό του Problem Specific Theoretical Machine Peak.

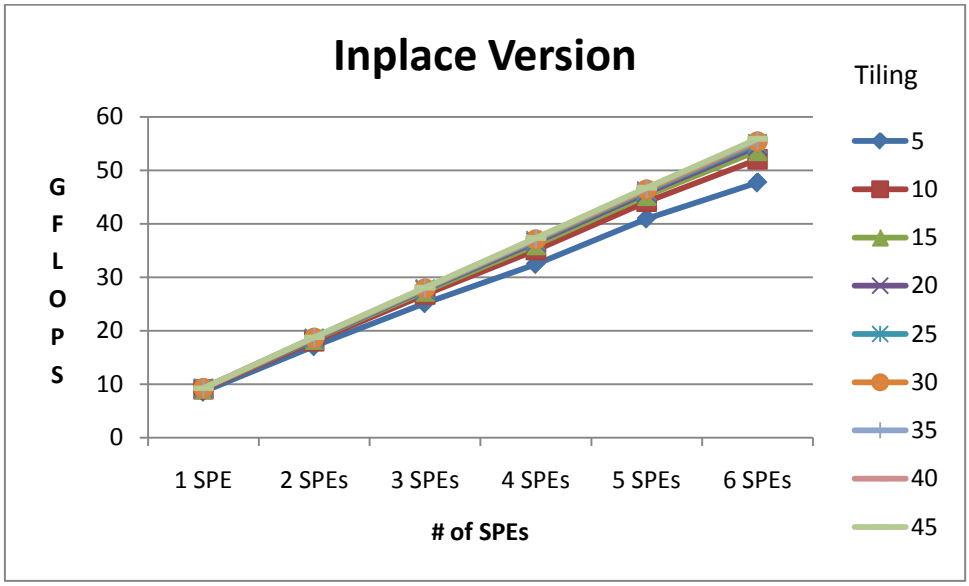
- Το Actual Peak είναι η μέγιστη μετρούμενη επίδοση που επιτυγχάνει το πρόγραμμα για μεγάλο μέγεθος πίνακα (6.144 x 6.144) και αριθμό επαναλήψεων (9.000). Η απόκλιση από τη θεωρητική καμπύλη οφείλεται ως επί το πλείστον στις τριγωνικές περιοχές lead-in και lead-out των blocks, όπου ο υπολογισμός είναι αργός.

Ακολουθεί διάγραμμα που δείχνει την επιτάχυνση συναρτήσει του αριθμού των SPEs που χρησιμοποιούνται, για πίνακα διαστάσεων 6.144 x 6.144 και 12.600 επαναλήψεις. Στο ίδιο διάγραμμα φαίνεται με διακεκομμένη γραμμή η ευθεία που προκύπτει από τη γραμμικοποίηση των μετρήσεων (γραμμική παλινδρόμηση). Παρατηρείται ότι η αύξηση της επίδοσης είναι απολύτως γραμμική όπως φαίνεται και από το συντελεστή συσχέτισης (R^2 = 0.9999).



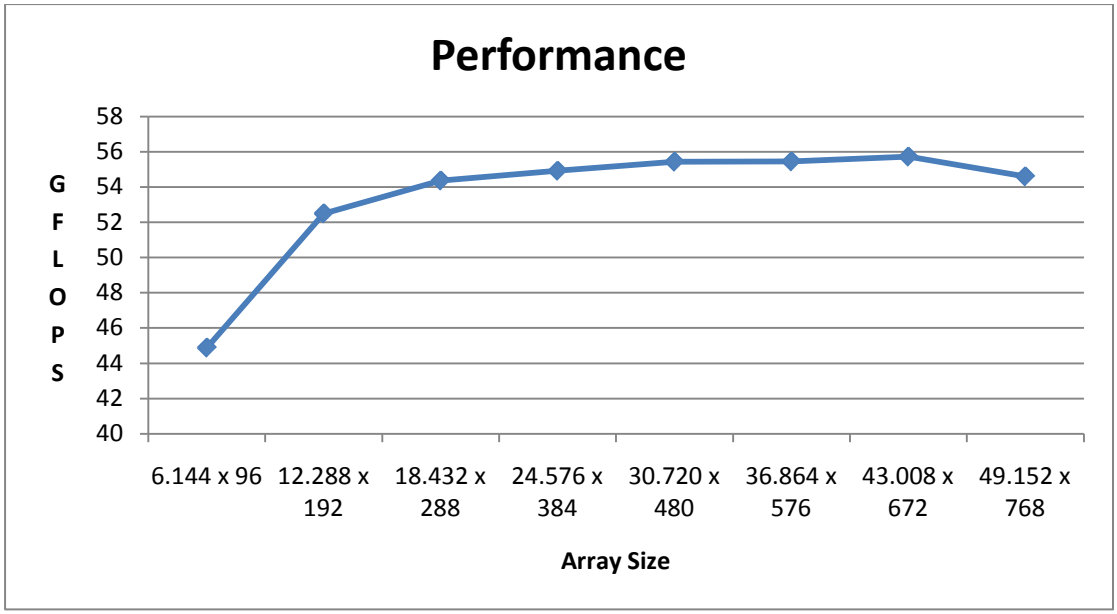
Σημαντικό ρόλο στην επίδοση της εφαρμογής παίζει και το μέγεθος του tiling. Συγκεκριμένα, έγιναν μετρήσεις της επίδοσης σε πίνακα 6.144 x 6.144 για τιμές tiling 5, 10, 15, 20, 25, 30, 35, 40 και 45, με 12.600 συνολικές επαναλήψεις και για αριθμό χρησιμοποιούμενων SPEs από ένα έως έξι. Ακολουθούν διαγράμματα της επίδοσης συναρτήσει του tiling για ένα έως έξι SPEs και της επίδοσης συναρτήσει των SPEs για τις προαναφερόμενες τιμές tiling.





Όπως φαίνεται από το δεύτερο διάγραμμα, το υψηλό tiling ευνοεί περισσότερο τις εκτελέσεις της εφαρμογής όσο αυξάνεται το πλήθος των χρησιμοποιούμενων SPEs, γεγονός αναμενόμενο, αφού μεγάλο πλήθος SPEs συνεπάγεται περισσότερες μεταφορές δεδομένων.

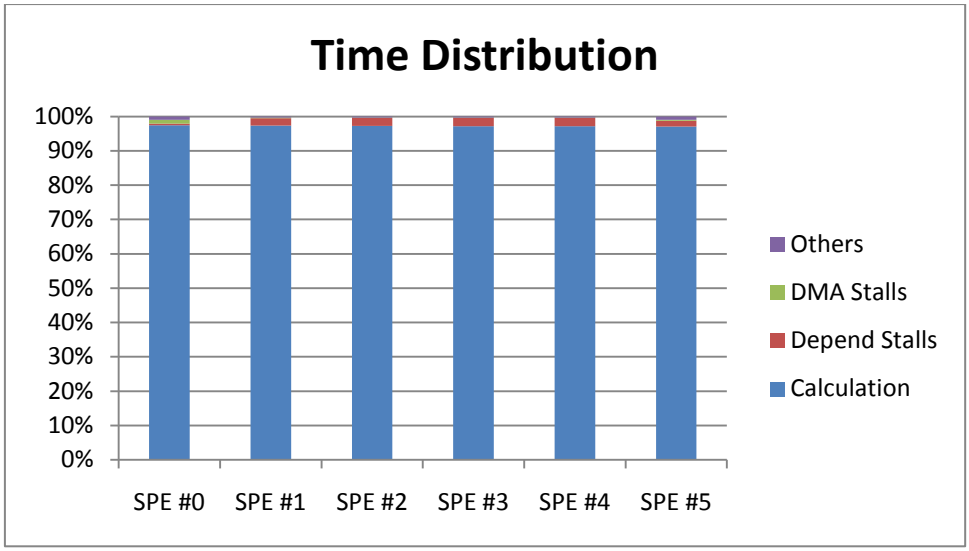
Πραγματοποιήθηκε άλλη μία σειρά μετρήσεων της επίδοσης της εφαρμογής συναρτήσει του μεγέθους του πίνακα-χωρίου. Συγκεκριμένα, μελετήθηκαν τα χωρία 6.144 x 96, 12.288 x 192, 18.432 x 288, 24.576 x 384, 30.720 x 480, 36.864 x 576, 43.008 x 672 και 49.152 x 768 με χρήση έξι SPEs και 12.600 επαναλήψεων. Η μορφή των μεγεθών αυτών έχει επιλεγεί λόγω του «ιδιότροπου» σχήματος των blocks που χρησιμοποιεί ο in-place αλγόριθμος. Αν αναχθούν αυτές οι διαστάσεις σε πλέγματα blocks, τα πλέγματα αυτά είναι τετράγωνα (π.χ. το χωρίο 6.144 x 96 χωρίζεται σε ένα πλέγμα 12 x 12 από blocks).



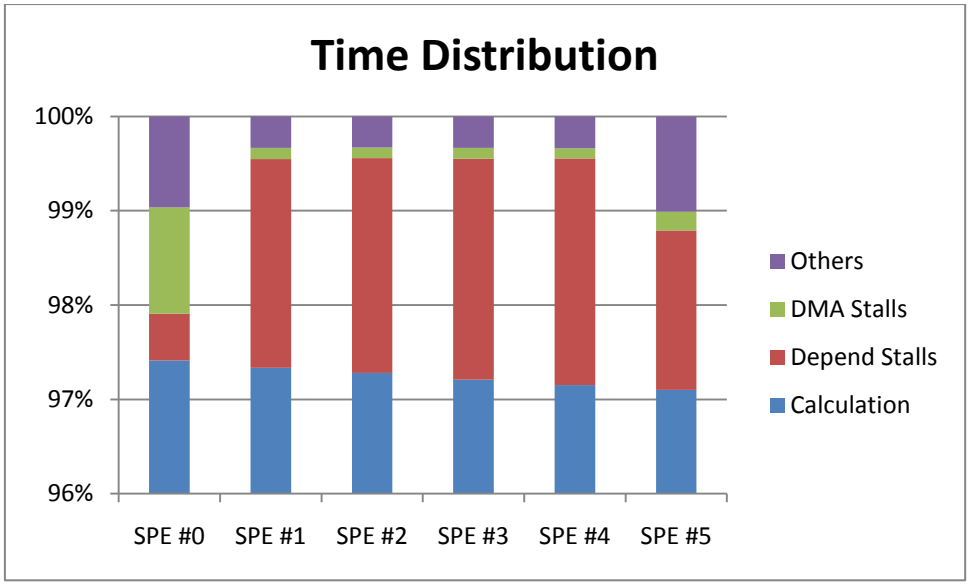
Γενικά, όσο αυξάνεται το μέγεθος του χωρίου τόσο αυξάνεται και η επίδοση της εφαρμογής. Η πτώση που παρατηρείται στο τελευταίο, μεγάλο χωρίο οφείλεται στην περιορισμένη μνήμη που έχει εγκατεστημένη το PlayStation 3. Αυτό το μέγεθος είναι στα όρια των πινάκων που μπορούν να χρησιμοποιηθούν για προγραμματισμό στη συγκεκριμένη κονσόλα, πριν εξαντληθεί ο χώρος της RAM και αρχίσει να χρησιμοποιείται το αργό swap file.

Ένα ακόμη σημαντικό στοιχείο για την ανάλυση της εφαρμογής είναι η κατανομή του συνολικού χρόνου εκτέλεσης σε επιμέρους χρόνους. Ακολουθεί ένα διάγραμμα με μπάρες (μία για κάθε SPE) όπου φαίνεται, σε πρόβλημα με διαστάσεις 6.144 x 6.144 και 12.600 επαναλήψεις, η κατανομή πάνω στο συνολικό χρόνο εκτέλεσης των επιμέρους χρόνων

- Υπολογισμού
- Καθυστερήσεων λόγω μεταφορών DMA
- Καθυστερήσεων λόγω εξαρτήσεων του προβλήματος (π.χ. το δεύτερο SPE δεν μπορεί να ξεκινήσει υπολογισμούς προτού το πρώτο SPE υπολογίσει τουλάχιστον ένα block)
- Λοιπών ενεργειών που γίνονται κατά την εκτέλεση του προγράμματος



Επειδή ο χρόνος που αφιερώνεται στους υπολογισμούς καταλαμβάνει πάνω από το 95% του συνολικού χρόνου, με αποτέλεσμα οι υπόλοιπες συνιστώσες του διαγράμματος να μην είναι ευδιάκριτες, ακολουθεί και δεύτερο διάγραμμα που αποτελεί μεγέθυνση του πρώτου στην περιοχή 96% έως 100%. Όπως φαίνεται, οι καθυστερήσεις λόγω μεταφορών DMA είναι ελάχιστες, της τάξης του 0.2%.



Η μεγάλη διαφοροποίηση στα DMA stalls που παρουσιάζει το πρώτο SPE οφείλεται στον εξής λόγο: Κάθε SPE πρέπει να πάρει τρεις πίνακες για να κάνει τους υπολογισμούς του.

- Έναν πίνακα 16KB με τις τιμές της προηγούμενης επανάληψης
- Έναν πίνακα για το οριζόντιο σύνορο (για tiling=45 αυτός έχει μέγεθος 45 x 512 x 4bytes/float = 92KB)

- Έναν για το κατακόρυφο σύνορο (για tiling=45 αυτός έχει μέγεθος 45 x 8 x 4bytes/float = 1.5KB).

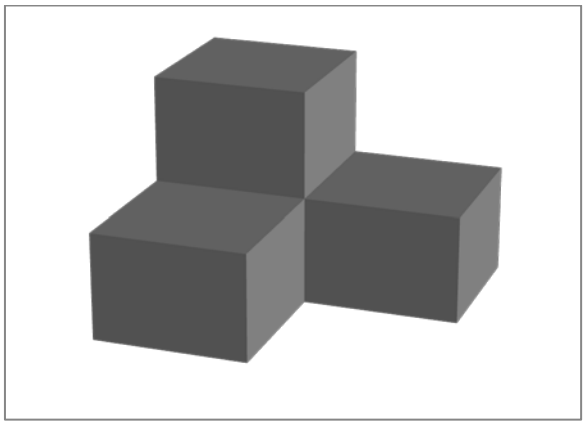
Το οριζόντιο σύνορο είναι πολύ μεγάλο και αν δεν φορτωθεί εγκαίρως στην τοπική μνήμη τότε θα επιβαρύνει πολύ το χρόνο επικοινωνίας. Το πρώτο SPE αναγκαστικά δεν μπορεί να έχει προφορτώσει αυτό το σύνορο, αφού το ίδιο το σύνορο δεν είναι διαθέσιμο στην κύρια μνήμη. Ο MPI κόμβος, δηλαδή, δεν το έχει λάβει ακόμα από το γειτονικό του κόμβο (η υλοποίηση της εφαρμογής είναι κοινή για μονό κόμβο και συστοιχία, οπότε το πρώτο SPE δεν προφορτώνει το σύνορο από τη RAM ούτε στην περίπτωση του μονού κόμβου, παρόλο που θα μπορούσε). Όταν το σύνορο έρθει, τότε ξεκινάνε τα SPEs τους υπολογισμούς. Για το πρώτο SPE, λοιπόν, υπάρχει η καθυστέρηση μέχρι να έρθει από τη RAM στην τοπική μνήμη αυτό το μεγάλο κομμάτι δεδομένων. Τα υπόλοιπα SPEs ούτως ή άλλως δεν μπορούν να κάνουν υπολογισμούς προτού το πρώτο SPE τελειώσει το πρώτο του block, ώστε να στείλει το κατακόρυφο σύνορο στο διπλανό του SPE κ.τ.λ.. Έτσι, τα υπόλοιπα SPEs αναγκαστικά περιμένουν (αυτή είναι και η έννοια των dependency stalls) και σε αυτόν τον ούτως ή άλλως νεκρό χρόνο έχουν την ευκαιρία να φέρουν τα απαραίτητα δεδομένα στην τοπική μνήμη. Άρα, όταν το πρώτο SPE τελειώσει το πρώτο του block και στείλει το κατακόρυφο σύνορο στο δεύτερο SPE, αυτό έχει ήδη φορτώσει το οριζόντιο σύνορο και ξεκινάει αμέσως υπολογισμούς. Με άλλα λόγια, αυτό το μεγαλύτερο ποσοστό DMA stalls που παρουσιάζει το πρώτο SPE "κρύβεται" στα υπόλοιπα SPEs πίσω από το χρόνο που προσμετρείται ως dependency stall.

Ένα ενδιαφέρον σχόλιο που μπορεί να γίνει από τις παραπάνω μετρήσεις και συγκεκριμένα από το αναλυτικό μοντέλο επίδοσης έχει να κάνει με την καμπύλη Theoretical Algorithm Peak. Όπως φαίνεται, η θεωρητική απόδοση της παρούσας υλοποίησης του αλγορίθμου in-place είναι η μισή από αυτήν που θεωρητικά μπορεί να επιτύχει το μηχάνημα. Αυτό, όπως έχει αναλυθεί, οφείλεται στις πολλές εξαρτήσεις μεταξύ των στοιχείων εντός του κάθε block, οι οποίες καθυστερούν τα pipelines των SPEs. Προς υπέρβαση αυτών των προβλημάτων προτείνονται οι λύσεις της συγκεντρωτικής επεξεργασίας επαναλήψεων και του αλγορίθμου out-of-place.

Θεωρητική ανάλυση της συγκεντρωτικής επεξεργασίας επαναλήψεων

Ακολουθεί η θεωρητική ανάλυση μίας άλλης προσέγγισης του θέματος, η οποία δεν έχει υλοποιηθεί αλλά αφήνεται για μελλοντική διερεύνηση. Όπως έχει ήδη αναφερθεί από την ανάλυση των εξαρτήσεων των δεδομένων, ο υπολογισμός μίας τιμής του χωρίου επιτρέπει τον υπολογισμό όχι μόνο της τιμής των δύο γειτονικών του σημείων αλλά και της τιμής του ίδιου σημείου που αντιστοιχεί στην επόμενη επανάληψη. Αυτή η εξάρτηση των δεδομένων αναπαρίσταται στο σχήμα που ακολουθεί, το οποίο ο εγκάρσιος (ως προς το σχήμα) άξονας αναπαριστά το χρόνο, δηλαδή την αλληλουχία των επαναλήψεων. Σε αυτό το σχήμα φαίνονται

τα τρία σημεία που μπορούν να υπολογιστούν όταν έχει ολοκληρωθεί ο υπολογισμός του σημείου που αυτά «περικυκλώνουν».



Έτσι, μπορεί να κατασκευαστεί αλγόριθμος που σαρώνει το block όχι μόνο χωρικά, αλλά την ίδια στιγμή και χρονικά – κατά μήκος του άξονα των επαναλήψεων. Η προσέγγιση αυτή απαιτεί την εφαρμογή tiling που ως γνωστόν περιορίζει τις μεταφορές δεδομένων από και προς τη μνήμη και αυξάνει την αναλογία των πράξεων ως προς μεταφορές, μία αναλογία που, όπως γνωρίζουμε, είναι επιβεβαρυμμένη στα προβλήματα διάχυσης.

Η εφαρμογή tiling προσφέρει τα πλεονεκτήματα και τα μειονεκτήματα που έχουν αναφερθεί. Έχοντας tiling βήματος N, για κάθε block που μεταφέρεται στην τοπική μνήμη του κάθε SPE εκτελούνται όλες οι N επαναλήψεις του υπολογισμού και αποθηκεύονται πίσω στη μνήμη τα αποτελέσματα. Προφανώς, οι μεταφορές δεδομένων του πίνακα-χωρίου περιορίζονται στο 1/N, αφού πραγματοποιούνται μία φορά ανά N βήματα. Στον αντίποδα, αυξάνεται ο χώρος που απαιτείται για την αποθήκευση των συνοριακών τιμών κατά N φορές, οπότε απαιτείται μια ισορροπημένη τιμή για το βαθμό του tiling.

Σε αυτό το σημείο κρίνεται σκόπιμο να διευκρινιστεί η διαφορά μεταξύ tiling και συγκεντρωτικής επεξεργασίας επαναλήψεων (ΣΕΕ):

- Όταν υπάρχει απλό tiling, κάθε SPE υπολογίζει τις τιμές για όλα τα σημεία του block και αφού ολοκληρωθεί αυτή η διαδικασία τότε το SPE περνάει στην επόμενη επανάληψη. Η επικοινωνία με άλλες επεξεργαστικές μονάδες γίνεται όταν ολοκληρωθεί ένας αριθμός επαναλήψεων. Ο αριθμός αυτός είναι το «βήμα tiling».
- Όταν υπάρχει συγκεντρωτική επεξεργασία επαναλήψεων, πάλι ομαδοποιείται ένας αριθμός επαναλήψεων σε μία «υπέρ-επανάληψη», δηλαδή η έννοια της συγκεντρωτικής επεξεργασίας επαναλήψεων εμπεριέχει την έννοια του tiling, όμως κάθε SPE εκτελεί υπολογισμούς σε πολλά διαφορετικά χρονικά βήματα – επαναλήψεις ταυτόχρονα, δηλαδή δεν εκτελεί την κάθε επανάληψη

μεμονωμένα, αλλά έχει σε εξέλιξη πολλά επίπεδα-βήματα ταυτόχρονα. Ο αριθμός των βημάτων αυτών είναι το «βήμα ΣΕΕ».

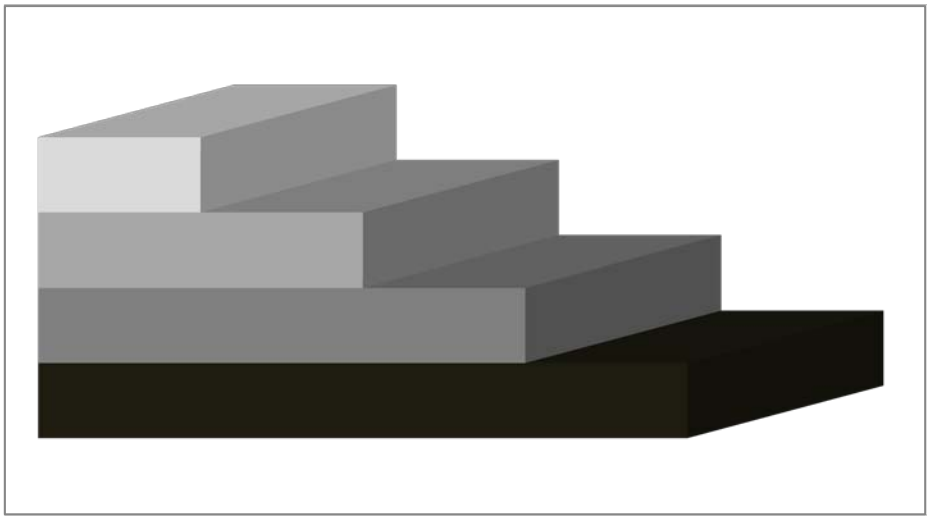
Από τη στιγμή που εφαρμόζεται η συγκεντρωτική εκτέλεση επαναλήψεων απαιτείται η εφαρμογή tiling. Έτσι, εισάγεται αυτομάτως ένας περιορισμός, ότι το βήμα του tiling θα πρέπει να είναι πολλαπλάσιο του βήματος ΣΕΕ. Η υλοποίηση της ΣΕΕ επιφέρει κάποια σημαντικά πλεονεκτήματα στην ταχύτητα επεξεργασίας, κρύβει όμως και αρκετές προγραμματιστικές δυσκολίες. Για χάρη της διερεύνησης της ΣΕΕ θεωρείται ότι το βήμα ΣΕΕ είναι ίσο με τέσσερα. Πρόκειται για μία λογική τιμή, αφού σε ένα block διαστάσεων 8 x 512 σχηματίζονται δύο ανεξάρτητα μεταξύ τους διανύσματα ανά διαγώνιο, οπότε έχοντας σε ταυτόχρονη επεξεργασία τέσσερα επίπεδα προκύπτουν συνολικά οκτώ ανεξάρτητα μεταξύ τους διανύσματα. Το νούμερο αυτό υπερκαλύπτει τις απαιτήσεις για συνεχή τροφοδοσία του άρτιου pipeline (έξι διανύσματα), άρα πληρούνται οι προϋποθέσεις για την επίτευξη επίδοσης κοντά στο θεωρητικό μέγιστο.

Η αρχιτεκτονική της SPU περιλαμβάνει δύο ανεξάρτητες σωληνώσεις, κάθε μία από τις οποίες αναλαμβάνει την εκτέλεση διαφορετικών τύπων εντολών. Η άρτια σωλήνωση είναι μεταξύ άλλων υπεύθυνη για την εκτέλεση εντολών κινητής υποδιαστολής. Από την άλλη, η περιττή σωλήνωση εκτελεί εντολές όπως φόρτωση δεδομένων από την τοπική μνήμη στους καταχωρητές, εντολές μορφοποίησης διανυσμάτων (rotate, shift, shuffle) και αποθηκεύσεις διανυσμάτων από τους καταχωρητές στην τοπική μνήμη. Η μέγιστη δυνατή επίδοση ως προς τις πράξεις κινητής υποδιαστολής επιτυγχάνεται όταν το πλήθος των εντολών του άρτιου pipeline είναι μεγαλύτερο ή ίσο με τον αριθμό των εντολών του περιττού pipeline. Σε αντίθετη περίπτωση, αν δηλαδή οι εντολές για φόρτωση/διαμόρφωση/αποθήκευση είναι περισσότερες από τις εντολές κινητής υποδιαστολής, τότε σε κάποιους κύκλους δεν θα υπάρχουν διαθέσιμα δεδομένα για να τροφοδοτηθεί το άρτιο pipeline. Αυτό θα δημιουργήσει stalls στη λειτουργία του άρτιου pipeline και η απόδοση θα μειωθεί.

Συγκεκριμένα για το πρόβλημα της εξίσωσης διάχυσης, στην περίπτωση που δεν υπάρχει ΣΕΕ, αλλά κάθε επίπεδο υπόκειται μεμονωμένα σε επεξεργασία, οι εντολές του περιττού pipeline είναι περισσότερες. Για τον υπολογισμό της τιμής κάθε στοιχείου απαιτούνται πέντε πράξεις κινητής υποδιαστολής, όπως φαίνεται από τον ψευδοκώδικα του αλγορίθμου, οι οποίες πραγματοποιούνται από τρεις εντολές της SPU (δύο multiply_add και μία multiply). Για κάθε τρεις εντολές πράξεων κινητής υποδιαστολής μέσα στις κυανές περιοχές του block χρειάζονται μια εντολή φόρτωσης, μία εντολή αναδιάταξης του διανύσματος και μια εντολή αποθήκευσης. Επιπλέον, ανά τέσσερεις διαγωνίους χρειάζονται άλλες πέντε εντολές του περιττού pipeline για τη διαχείριση των συνοριακών τιμών. Συνολικά, δηλαδή, για κάθε διαγώνιο απαιτούνται 1.25 παραπάνω εντολές του περιττού pipeline (ή αλλιώς 5 εντολές ανά τέσσερις διαγωνίους) σε σχέση με τις εντολές του άρτιου pipeline.

Εκτελώντας υπολογισμούς σε τέσσερα επίπεδα ταυτόχρονα, οι οκτώ εντολές φόρτωσης και οι οκτώ αποθήκευσης (2 διανύσματα x 4 επίπεδα) μειώνονται σε δύο και δύο αντίστοιχα (κέρδος 6 + 6 εντολές). Η μείωση αυτή προκύπτει από το γεγονός ότι όταν εξαχθεί ένα αποτέλεσμα χρησιμοποιείται αμέσως για τον υπολογισμό του επόμενου επιπέδου. Σε διαφορετική περίπτωση, το αποτέλεσμα θα αποθηκευόταν πίσω στην τοπική μνήμη του SPE και θα ξαναφορτωνόταν κατά τον υπολογισμό της επόμενης επανάληψης. Έτσι προκύπτει ένα κέρδος δώδεκα εντολών ανά διαγώνιο στο περιττό pipeline. Βέβαια, με την επεξεργασία τεσσάρων επιπέδων ταυτόχρονα, τετραπλασιάζονται οι εντολές για τη διαχείριση των διανυσμάτων των συνοριακών τιμών. Συγκεκριμένα απαιτούνται 20 εντολές ανά τέσσερεις διαγωνίους. Η χρήση ΣΕΕ ήδη μειώνει κατά 48 [(6 + 6) x 4 διαγωνίους] τις εντολές στο περιττό pipeline, άρα συνολικά οι εντολές για αυτό το pipeline καταλήγουν να είναι λιγότερες από αυτές για το άρτιο pipeline. Με αυτόν τον τρόπο ικανοποιούνται οι προϋποθέσεις για την επίτευξη υπολογιστικής επίδοσης που πλησιάζει το θεωρητικό μέγιστο.

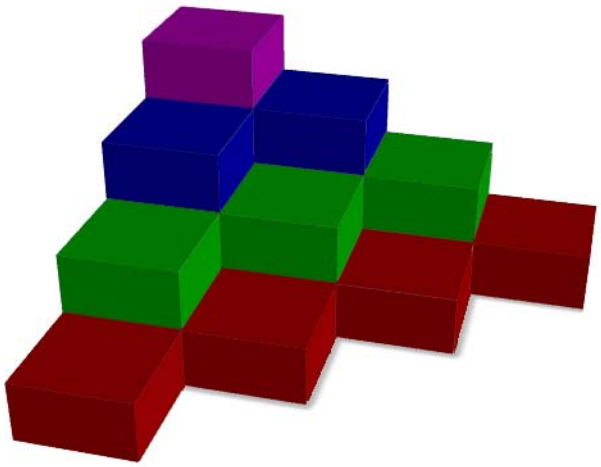
Στον αντίποδα, ο αλγόριθμος αυτός παρουσιάζει προγραμματιστικά κάποιες δυσκολίες. Η ταυτόχρονη επεξεργασία των τεσσάρων επιπέδων μπορεί να απεικονισθεί με το κάτωθι σχήμα (χάριν απλοποίησης δεν απεικονίζεται η κατά διαγωνίους διάσχιση του πίνακα, αλλά φαίνεται μία διάσχιση κατά στήλες).



Όπως φαίνεται και στο σχήμα, τα τέσσερα επίπεδα που υπολογίζονται σε κάθε βήμα ΣΕΕ απέχουν το ένα από το άλλο κατά μια διαγώνιο. Για τον υπολογισμό κάθε διαγωνίου πρέπει, λόγω των εξαρτήσεων του προβλήματος, να έχει ήδη υπολογιστεί η τιμή για αυτή τη διαγώνιο στο προηγούμενο επίπεδο-επανάληψη. Το χαρακτηριστικό αυτό κάνει αρκετά πολύπλοκο τον κώδικα που διαχειρίζεται τις περιοχές lead-in και lead-out των blocks.

Όπως έχει προαναφερθεί, τα στοιχεία αυτών των περιοχών θα πρέπει να υπολογιστούν είτε ένα-ένα (βαθμωτά) είτε με κάποιον πολύπλοκο και δύσχρηστο κώδικα που θα προσπαθήσει κατά το δυνατόν να χρησιμοποιήσει διανύσματα. Το πρόβλημα αυτό

επιδεινώνεται όταν υπάρχουν τέσσερα επίπεδα προς ταυτόχρονο υπολογισμό. Η έκταση των lead-in και lead-out περιοχών σε τέσσερα επίπεδα δημιουργεί ένα αρκετά πολύπλοκο μέτωπο επεξεργασίας, το οποίο μπορεί να απεικονιστεί σχηματικά ως εξής:



Άρα η κατασκευή ενός κώδικα για την επεξεργασία των περιοχών εκτός των κυανών διαγωνίων των blocks, που αξιοποιεί τα αρχιτεκτονικά χαρακτηριστικά της CBE με τον καλύτερο δυνατό τρόπο, είναι αρκετά δύσκολη και πολύπλοκη.

Όπως συνεπάγεται από την παραπάνω ανάλυση, η προσέγγιση της συγκεντρωτικής επεξεργασίας επαναλήψεων δίνει, θεωρητικά τουλάχιστον, μία λύση στο πρόβλημα του χαμηλού Theoretical Algorithm Peak, είναι όμως προγραμματιστικά πολύπλοκη. Η δυσκολία στην υλοποίηση παρουσιάζει ως μία πιο βιώσιμη λύση για αύξηση της επίδοσης την προσέγγιση του αλγορίθμου out-of-place.

Out-of-place αλγόριθμος για την εξίσωση διάχυσης

Περιγραφή αλγορίθμου

Out-of-place αλγόριθμος για την εξίσωση διάχυσης καλείται ο αλγόριθμος εκείνος κατά τον οποίον ο υπολογισμός ενός σημείου U[t][y][x] γίνεται λαμβάνοντας τις τιμές U[t-1][y][x], U[t-1][y-1][x] και U[t-1][y][x-1]. Η διαφορά του από τον in-place αλγόριθμο είναι ότι όλες οι εξαρτήσεις αναφέρονται σε σημεία της προηγούμενες επανάληψης, ενώ δεν υπάρχουν εξαρτήσεις από σημεία τις τρέχουσας επανάληψης. Ο out-of-place αλγόριθμος σε μορφή ψευδοκώδικα για την διακριτή μερική διαφορική εξίσωση της διάχυσης είναι ο ακόλουθος.

```
while(loops < MAXLOOPS)
{
  t = (++loops)%2;

  for(y = 1; y < Y; y++)
    for(x = 1; x < X; x++)
      U[t][y][x] = (1+2*a*dt/dx)*U[(t-1)%2][y][x] -  a*dt/dx*(U[t-1][y-1][x] + U[t-1][y][x-1]);
}
```

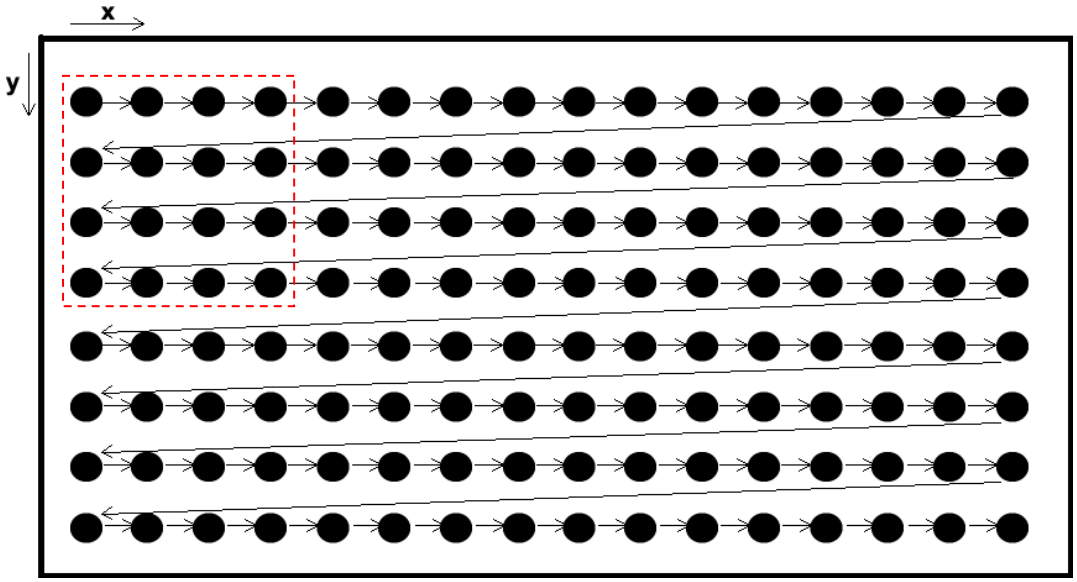
Από τα ανωτέρω προκύπτει ότι όλα τα δεδομένα που απαιτούνται για τον υπολογισμό κάθε σημείου (x, y) έχουν ήδη υπολογιστεί κατά την προηγούμενη επανάληψη του αλγορίθμου. Το γεγονός αυτό μας εξαλείφει τις εξαρτήσεις κάθε σημείου από τις τιμές των δύο γειτονικών του. Είναι, έτσι, δυνατή η χρήση διανυσματικών (SIMD) εντολών χωρίς τη διάσχιση του πίνακα κατά διαγωνίους, όπως απαιτείτο στον in-place αλγόριθμο. Αυτό συνεπάγεται την εξάλειψη όλων των προβλημάτων του in-place αλγορίθμου, όπως η ύπαρξη προβληματικών τριγωνικών περιοχών μέσα στα blocks, αφού πλέον το block μπορεί να διασχιστεί κατά γραμμές (ή κατά στήλες, σε κάποια γλώσσα προγραμματισμού που η αποθήκευση ενός πίνακα γίνεται κατά στήλες). Χωρίς πλέον τους σημαντικότερους περιορισμούς που υπάρχουν για τον in-place αλγόριθμο είναι δυνατή η εφαρμογή της ίδιας προσέγγισης με τον πολλαπλασιασμό πινάκων.

Επιλογή του μεγέθους και του σχήματος των blocks

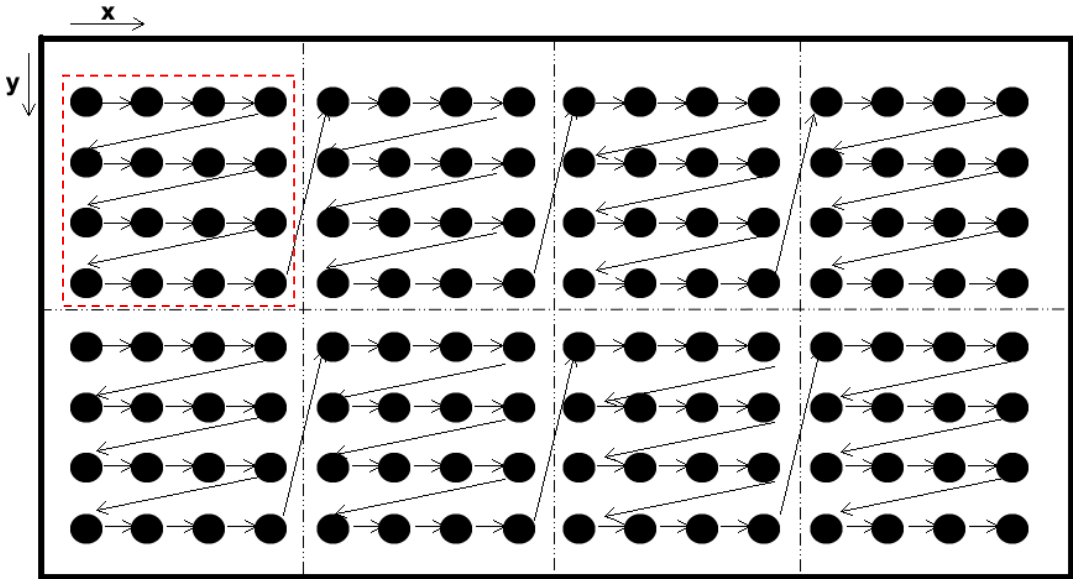
Η επιλογή και το σχήμα των blocks γίνεται όπως και στον πολλαπλασιασμό πινάκων, δηλαδή τετράγωνα blocks, διαστάσεων 64 x 64. Η διαμέριση του προβλήματος στα SPEs είναι η ίδια όπως και στον in-place αλγόριθμο. Μια μικρή διαφορά έγκειται στο γεγονός ότι κατά την ανταλλαγή συνοριακών τιμών δεν μεταφέρονται οι τιμές των συνόρων της τρέχουσας επανάληψης, αλλά της προηγούμενης.

Αποθήκευση του πίνακα κατά row-major και block-major layout

Στην εξ ορισμού προσέγγιση της C/C++ ένας πίνακας αποθηκεύεται στη μνήμη κατά γραμμές. Η σύμβαση αυτή αναφέρεται ως row-major layout. Όπως δείχνεται και στο σχήμα, τα στοιχεία μίας γραμμής αποθηκεύονται σε διαδοχικές θέσεις μνήμης.



Για τη μεταφορά, όμως, ενός block (τα στοιχεία που περικλείονται από την κόκκινη διακεκομμένη γραμμή) από την κύρια μνήμη στην τοπική μνήμη ενός SPE, δεδομένου ότι τα στοιχεία του block δεν βρίσκονται όλα σε διαδοχικές θέσεις, απαιτείται ένα πλήθος μεταφορών ίσο με το πλήθος των γραμμών του block μέσω μίας λίστας μεταφορών DMA (μία εντολή μεταφοράς DMA για κάθε γραμμή). Στην περίπτωση ενός block 8 x 512, οι γραμμές που πρέπει να μεταφερθούν είναι λίγες, οπότε δεν αναζητήθηκε κάποια καλύτερη προσέγγιση στο θέμα της επικοινωνίας μεταξύ SPE και κύριου αποθηκευτικού χώρου. Στην περίπτωση, όμως, των blocks 64 x 64 το πλήθος των μεταφορών αρχίζει και γίνεται μεγάλο, προκαλώντας συνωστισμό στον EIB. Η σχετικά μικρή πολυπλοκότητα των πράξεων πάνω στα δεδομένα των blocks προδιαθέτει για τον αντίκτυπο αυτού του συνωστισμού στην επίδοση της εφαρμογής, γεγονός που αποκαλύπτεται και από τις πειραματικές μετρήσεις.



Μία καλύτερη από άποψη επίδοσης προσέγγιση είναι η διάταξη block-major layout, κατά την οποία ο πίνακας αποθηκεύεται κατά blocks, όπως φαίνεται και από το προηγούμενο σχήμα. Με αυτήν την διάταξη, τα στοιχεία ενός block βρίσκονται πλέον αποθηκευμένα σε συνεχόμενες θέσεις μνήμης, οπότε ολόκληρο το block μπορεί να μεταφερθεί προς και από την τοπική μνήμη με μία και μόνη μεταφορά DMA, αρκεί το μέγεθος της μεταφοράς να ικανοποιεί τους περιορισμούς του MFC (≤16KB). Σε διαφορετική περίπτωση, το block θα πρέπει και πάλι να χωριστεί σε περισσότερα κομμάτια, κάθε ένα από τα οποία θα ικανοποιεί τους περιορισμούς αυτούς. Στο σημείο αυτό δικαιώνεται η επιλογή του μεγέθους του block, τόσο για τον πολλαπλασιασμό πινάκων όσο και για την εξίσωση διάχυσης. Το όφελος στην επίδοση της εφαρμογής είναι σημαντικό, όπως φάνηκε από τις μετρήσεις που έγιναν.

Στην περίπτωση του in-place αλγορίθμου, όπου το block αποτελείται από οκτώ γραμμές, η εφαρμογή του block-major layout δεν προσφέρει κάτι στην επίδοση. Το ίδιο ισχύει και στον πολλαπλασιασμό πινάκων όπου, αν και το block αποτελείται από 64 γραμμές, εντούτοις το πλήθος των πράξεων στα δεδομένα του block [της τάξεως του $O(n^3)$] υπερκαλύπτει το χρόνο των μεταφορών και ο ενδεχόμενος συνωστισμός στον EIB κρύβεται από την τελική επίδοση.

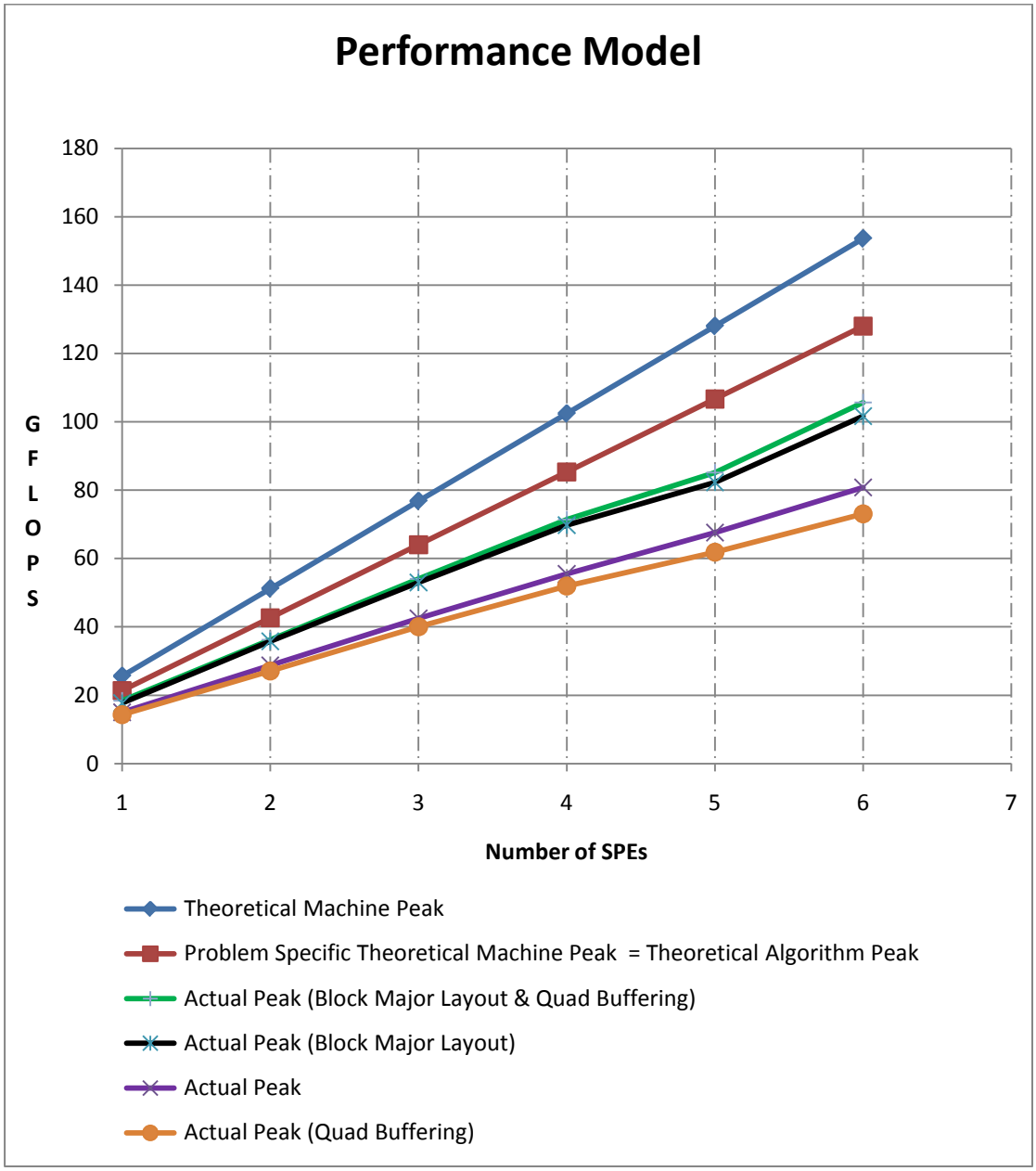
Το μειονέκτημα του αλγορίθμου out-of-place είναι η ταχύτητα σύγκλισης των υπολογισμών στο πραγματικό αποτέλεσμα, η οποία υπολείπεται της αντίστοιχης ταχύτητας του in-place αλγορίθμου, δηλαδή η out-of-place υλοποίηση χρειάζεται περισσότερες επαναλήψεις για την επίλυση της εξίσωσης.

Ο υπολογισμός της διάχυσης εντός του block υλοποιείται όπως φαίνεται από τον ακόλουθο ψευδοκώδικα.

```
/*
  const_1 = 1 + 2 * a * dt / dx
  const_2 = - a * dt / dx
  lower_vector, left_vector, current_vector, temp_vector: μεταβλητές-διανύσματα
*/

for i = 0 to BLOCK_I_DIM
  for j = 0 to (BLOCK_J_DIM / 4)
  {
    load into current_vector the j-th vector of the current line from U_old
    load into left_vector the (j-1)-th vector of the current line from U_old
    load into lower_vector the j-th vector of the previous line from U_old

    temp_vector = spu_shuffle(current_vector, left_vector) // --> {left_vector[3], current_vector[0],
                                                                  current_vector[1], current_vector[2]}
    current_vector = spu_mul(current_vector, const_1)
    current_vector = spu_madd(lower_vector, const_2, current_vector)
    current_vector = spu_madd(temp_vector, const_2, current_vector)

    store current_vector in U_new
  }
```

Μετρήσεις σε επίπεδο ενός κόμβου

Ακολουθεί η περιγραφή του αναλυτικού μοντέλου επίδοσης για τον αλγόριθμο out-of-place, η οποία φαίνεται σχηματικά στο παρακάτω διάγραμμα. Η μείωση του μεγέθους των συνόρων επιτρέπει μεγαλύτερες τιμές για tiling. Έτσι, οι μετρήσεις έγιναν για αριθμό SPEs από 1 έως έξι, για χωρίο διαστάσεων 6.144 x 6.144, 12.600 επαναλήψεις και tiling 60.



Σε σχέση με τον in-place αλγόριθμο, σε αυτήν την περίπτωση εντοπίζονται οι εξής διαφορές:

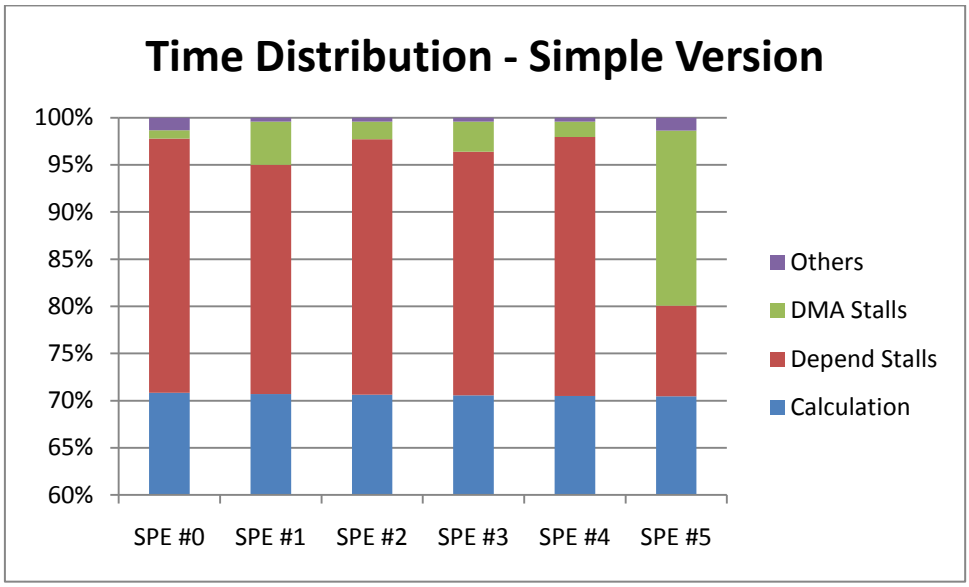
- Οι λιγότερες εξαρτήσεις του προβλήματος διάχυσης κατά τον out-of-place αλγόριθμο επιτρέπουν την εκτέλεση των πέντε πράξεων που χρειάζονται για τον υπολογισμό ενός διανύσματος-αποτελέσματος σε έξι κύκλους ρολογιού. Άρα, το

Theoretical Algorithm Peak ισούται πλέον με το Problem Specific Theoretical Machine Peak.

- Το Actual Peak σε αυτήν την περίπτωση απέχει πολύ από το θεωρητικό μέγιστο. Η εξέταση της κατανομής του χρόνου οδήγησε στο συμπέρασμα ότι η απόκλιση αυτή οφείλεται στις μεταφορές δεδομένων. Έτσι, διερευνήθηκε περισσότερο το πρόβλημα, με την εφαρμογή των τεχνικών quadruple-buffering και block-major layout στον πίνακα-χωρίο.

Η πρώτη προσέγγιση για την αντιμετώπιση του συνωστισμού στον EIB ήταν η χρήση quadruple-buffering αντί για double, έτσι ώστε τα δεδομένα να έχουν περισσότερο χρόνο διαθέσιμο για τη μεταφορά τους στην τοπική μνήμη, μέχρι να απαιτηθεί η χρήση τους. Η αποκλειστική εφαρμογή του quadruple-buffering προκαλεί μείωση της επίδοσης, διότι οι διπλάσιοι σε πλήθος buffers που χρησιμοποιούνται στην τοπική μνήμη του κάθε SPE μειώνουν το διαθέσιμο χώρο, με αποτέλεσμα να περιορίζεται το εφικτό μέγεθος tiling από 60 σε 45. Αντιθέτως, η εφαρμογή του block-major layout στην αποθήκευση του πίνακα-χωρίου βελτιώνει την επίδοση από 18% έως 26%, αναλόγως του πλήθους των χρησιμοποιούμενων SPEs. Η εφαρμογή και των δύο τεχνικών ταυτόχρονα επιφέρει τα καλύτερα συνολικά αποτελέσματα.

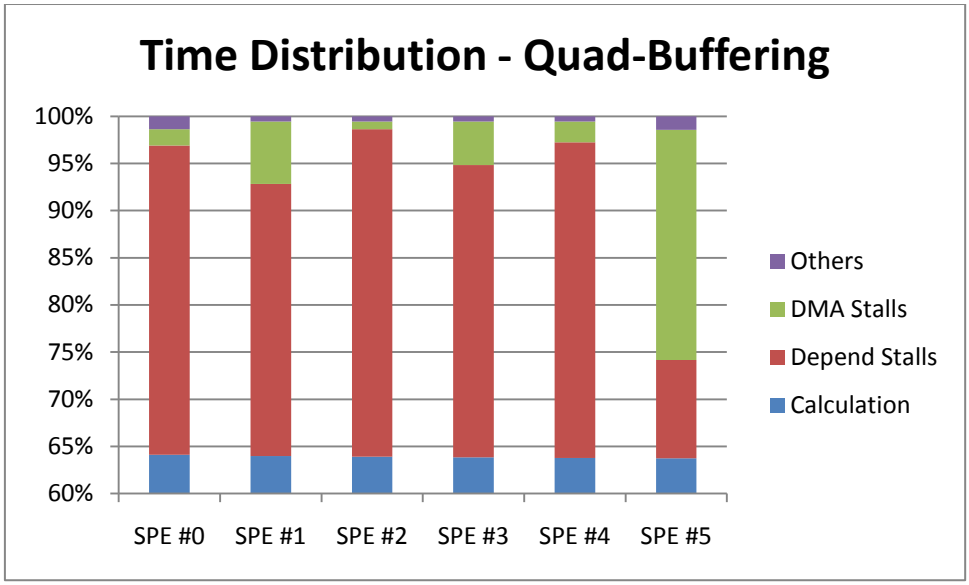
Ακολουθούν τα διαγράμματα κατανομής χρόνου για κάθε συνδυασμό των τεχνικών που αναφέρθηκαν. Παρατίθεται μόνο η μεγεθυμένη στην υψηλότερη περιοχή ποσοστών έκδοση των διαγραμμάτων.



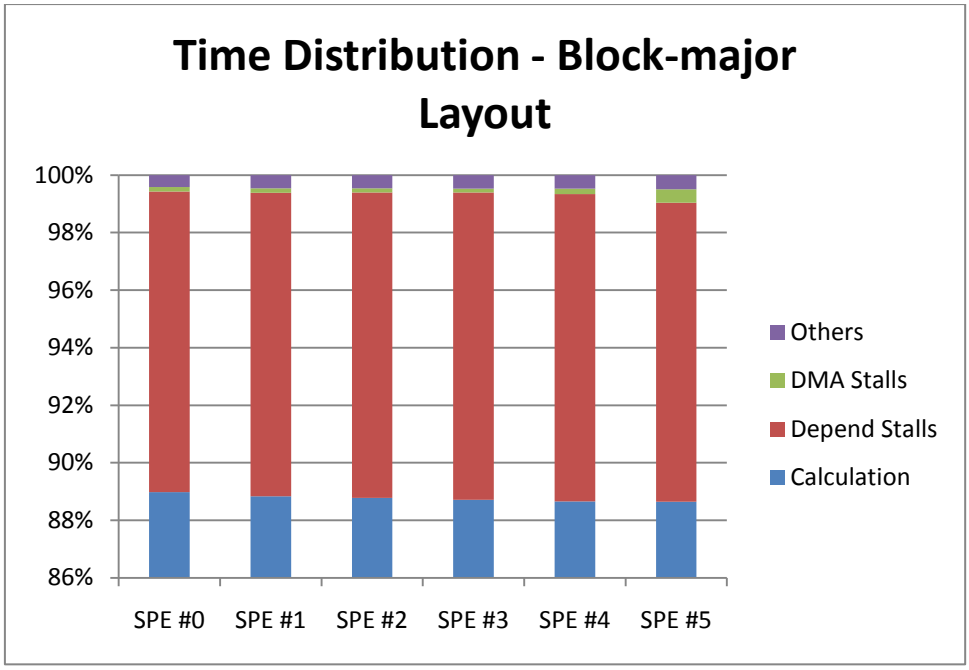
Παρατηρείται ότι μεγάλο ποσοστό του χρόνου, από 1.5% έως 4.5%, αφιερώνεται στις καθυστερήσεις των μεταφορών DMA. Δεδομένου ότι το μέγεθος των μεταφορών δεν διαφέρει

από το αντίστοιχο μέγεθος του in-place αλγορίθμου, η αιτία των καθυστερήσεων αναζητείται στη μορφή των blocks. Στον in-place αλγόριθμο, κάθε block αποτελείται από οκτώ γραμμές, πράγμα που σημαίνει ότι απαιτούνται οκτώ εντολές υπό μορφή λίστας για τη μεταφορά από και προς την κύρια μνήμη. Στην περίπτωση του out-of-place αλγορίθμου, οι μεταφορές αυτές αυξάνονται σε 64 με αποτέλεσμα, όπως προκύπτει στις μετρήσεις, το συνωστισμό στον EIB. Ειδικά για το SPE #5, το οποίο στέλνει το κατακόρυφο σύνορό του στην κύρια μνήμη αντί σε κάποιο άλλο SPE, οι καθυστερήσεις αυτές αποτελούν το 18.5% του συνολικού χρόνου εκτέλεσης.

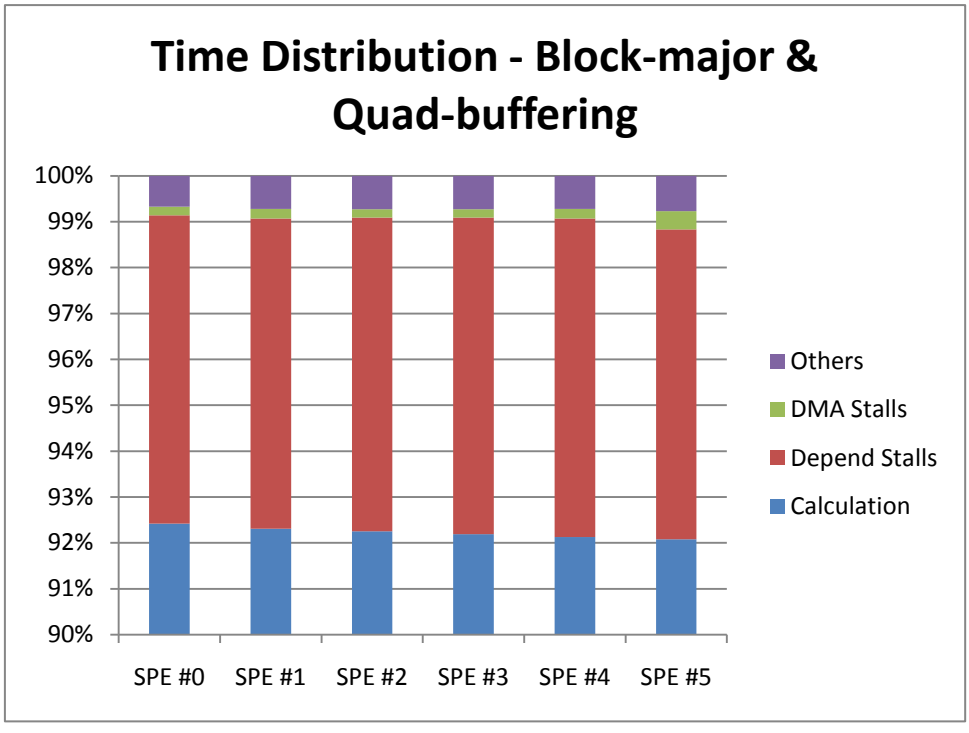
Ο out-of-place αλγόριθμος είναι υπολογιστικά σχεδόν δύο φορές γρηγορότερος από τον in-place (Theoretical Algorithm Peak), άρα οι καθυστερήσεις των μεταφορών δεν κρύβονται πλέον εύκολα. Η εφαρμογή της τεχνικής του quadruple-buffering δεν βελτιώνει την κατάσταση, απεναντίας προκαλεί μείωση στην επίδοση λόγω της μειωμένης τιμής tiling που μπορεί να χρησιμοποιηθεί.



Το πρόβλημα των αυξημένων γραμμών ενός block λύνεται με την εφαρμογή της τεχνικής block-major layout. Με αυτόν τον τρόπο, η μεταφορά κάθε block γίνεται με μία και μόνο εντολή μεταφοράς DMA. Το κέρδος στη μείωση των DMA stalls και κατ' επέκταση στην επίδοση του κώδικα είναι εμφανές:

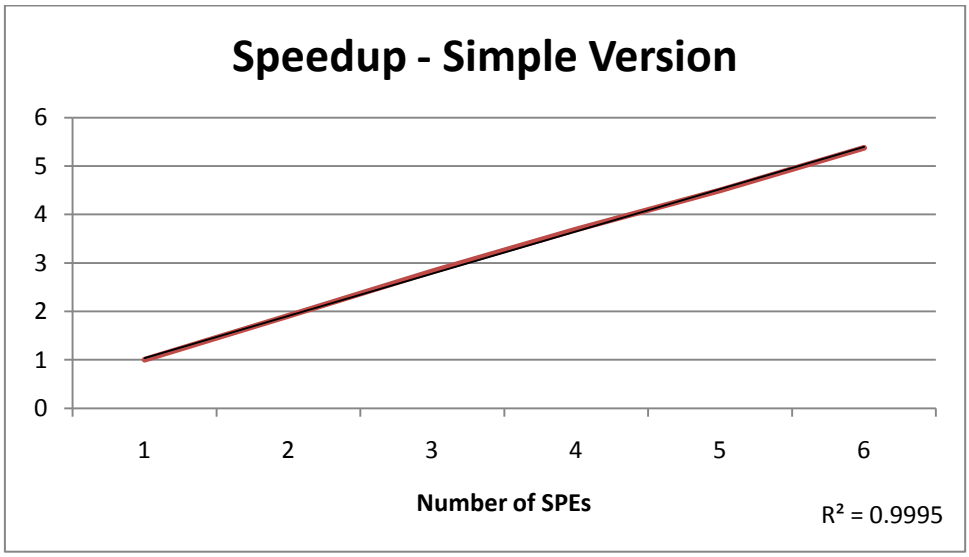


Η συνδυασμένη εφαρμογή του quadruple-buffering και του block-major layout επιφέρει μία περεταίρω μικρή βελτίωση της επίδοσης, παρόλο που το μειωμένο επιτρεπόμενο tiling έχει ως αποτέλεσμα την σχεδόν αμελητέα αύξηση του ποσοστού των DMA stalls.

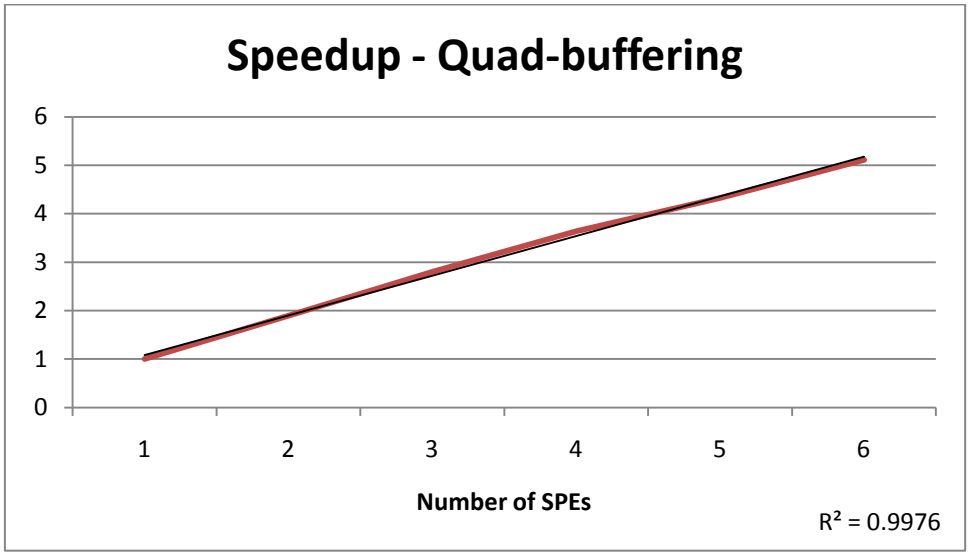


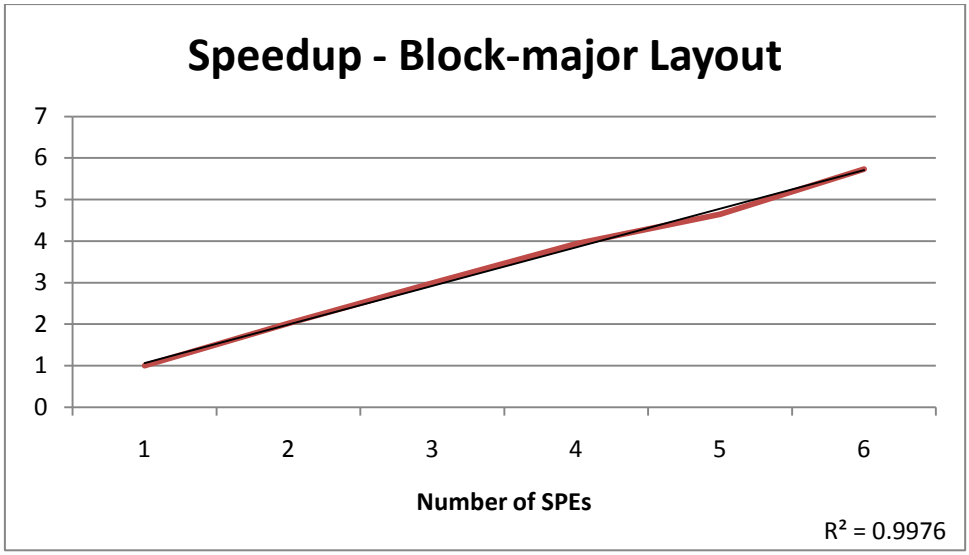
Ακολουθούν διαγράμματα που δείχνουν την επιτάχυνση συναρτήσει του αριθμού των SPEs που χρησιμοποιούνται, για πίνακα διαστάσεων 6.144 x 6.144 και 12.600 επαναλήψεις, για κάθε έκδοση του out-of-place αλγορίθμου. Σε κάθε διάγραμμα φαίνεται με διακεκομμένη

γραμμή η ευθεία που προκύπτει από τη γραμμικοποίηση των μετρήσεων (γραμμική παλινδρόμηση).

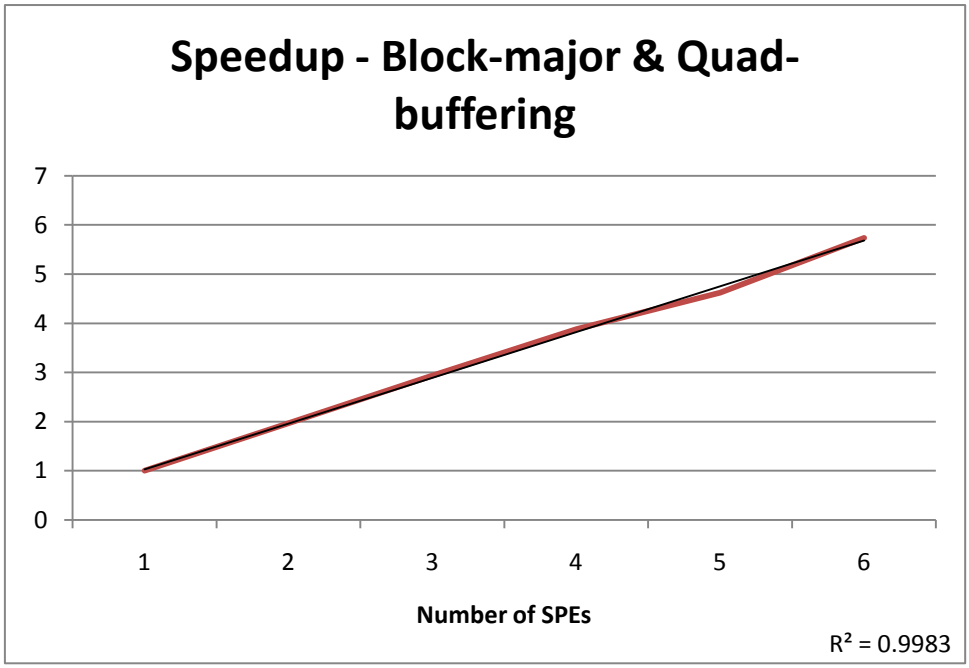


Παρατηρείται ότι η αύξηση της επίδοσης για την απλή έκδοση της εφαρμογής είναι σχεδόν γραμμική όπως φαίνεται και από το συντελεστή συσχέτισης ($R^2 = 0.9995$). Η έκδοση με quadruple-buffering και η έκδοση με block-major layout παρουσιάζουν επίσης σχεδόν γραμμικές επιταχύνσεις, με ελαφρώς μικρότερο συντελεστή ($R^2 = 0.9976$).

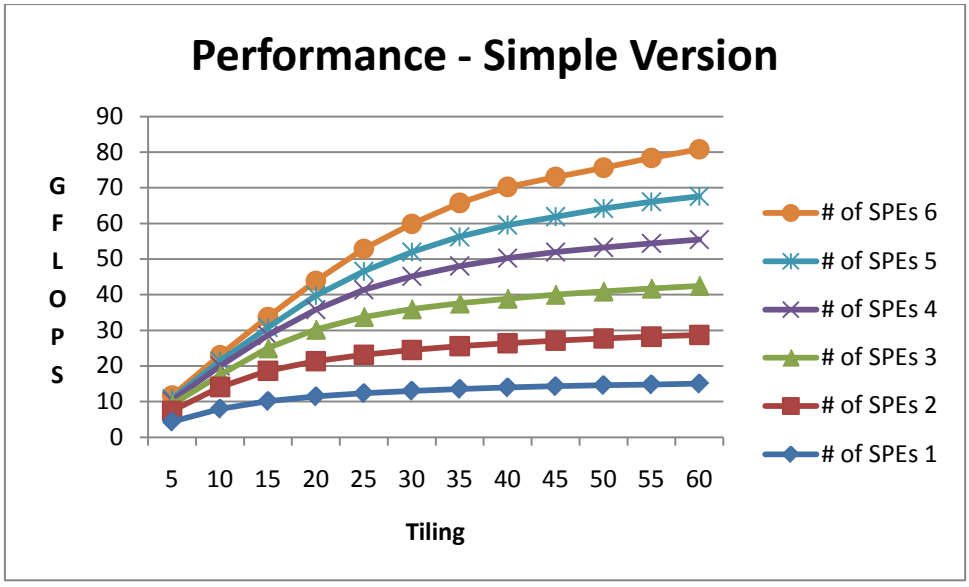


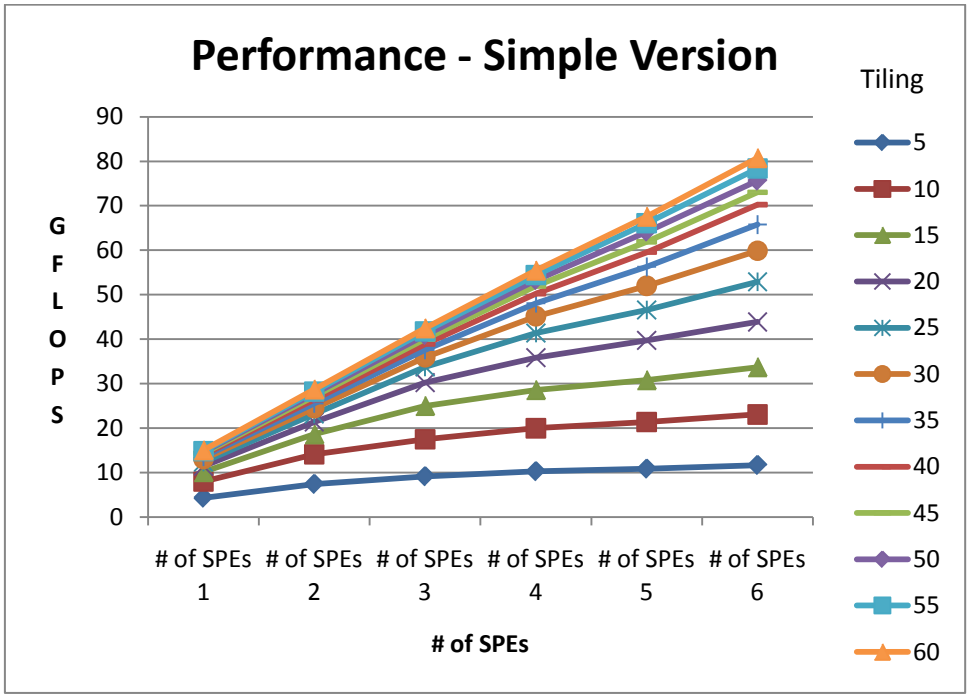


Τέλος, η έκδοση του συνδυασμού block-major layout και quadruple-buffering είναι παρουσιάζει επίσης γραμμική επιτάχυνση ($R^2 = 0.9983$).

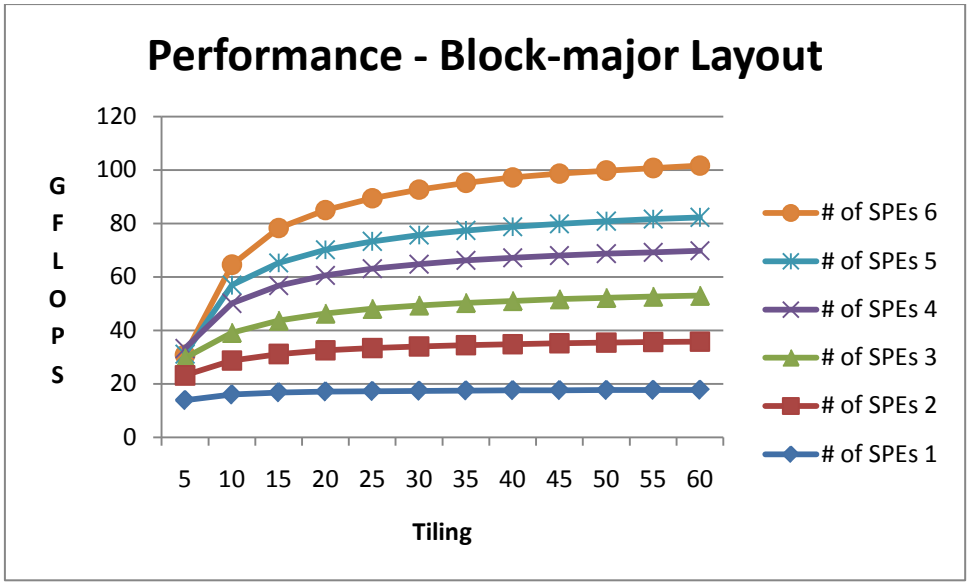


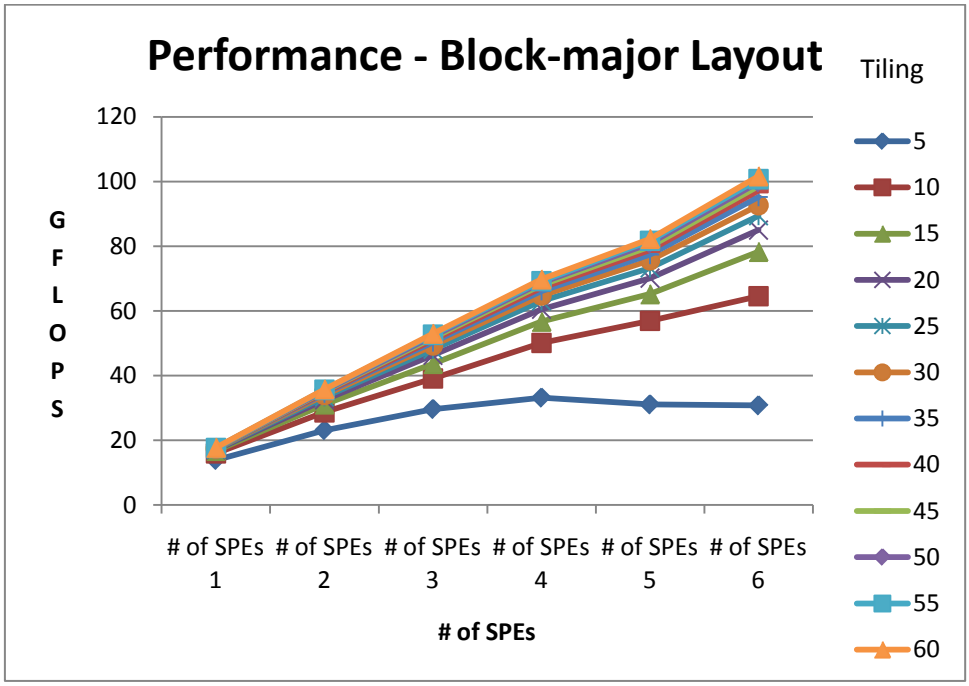
Ακολουθεί διερεύνηση της επίδοσης συναρτήσει του μεγέθους tiling. Συγκεκριμένα, για κάθε έκδοση της εφαρμογής έγιναν μετρήσεις της επίδοσης σε πίνακα 6.144 x 6.144 με 12.600 συνολικές επαναλήψεις και για αριθμό χρησιμοποιούμενων SPEs από ένα έως έξι. Στις εκδόσεις με quadruple-buffering το tiling παίρνει τιμές 5, 10, 15, 20, 25, 30, 35, 40 και 45, ενώ στις εκδόσεις χωρίς quadruple-buffering το tiling παίρνει επιπλέον και τις τιμές 50, 55 και 60. Ακολουθούν διαγράμματα της επίδοσης συναρτήσει του tiling για ένα έως έξι SPEs και της επίδοσης συναρτήσει των SPEs για τις προαναφερόμενες τιμές tiling σε κάθε περίπτωση.



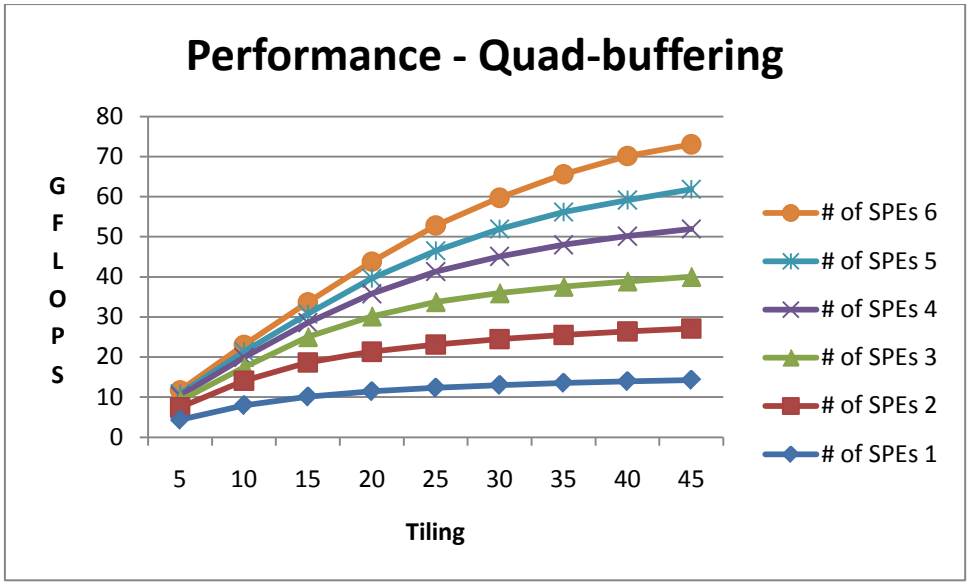


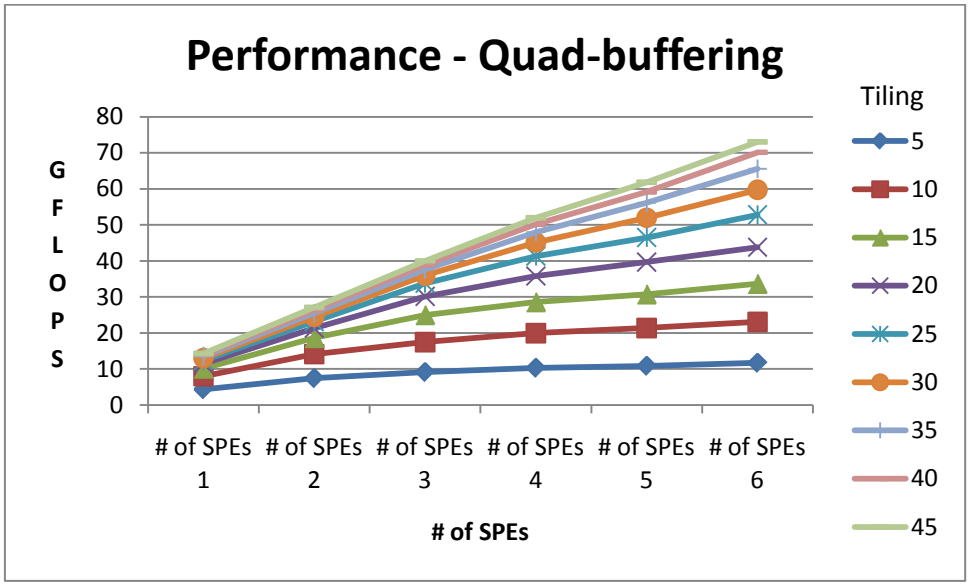
Όπως φαίνεται από το δεύτερο διάγραμμα, το υψηλό tiling ευνοεί περισσότερο τις εκτελέσεις της εφαρμογής όσο αυξάνεται το πλήθος των χρησιμοποιούμενων SPEs, γεγονός αναμενόμενο, αφού μεγάλο πλήθος SPEs συνεπάγεται περισσότερες μεταφορές δεδομένων. Το ίδιο ισχύει και για τις άλλες τρεις εκδόσεις:



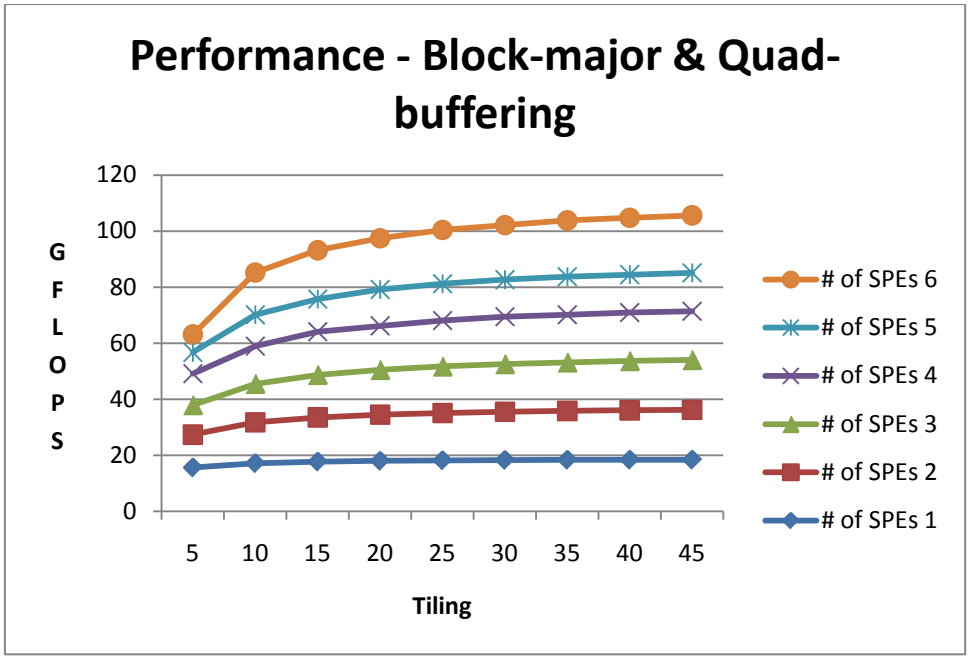


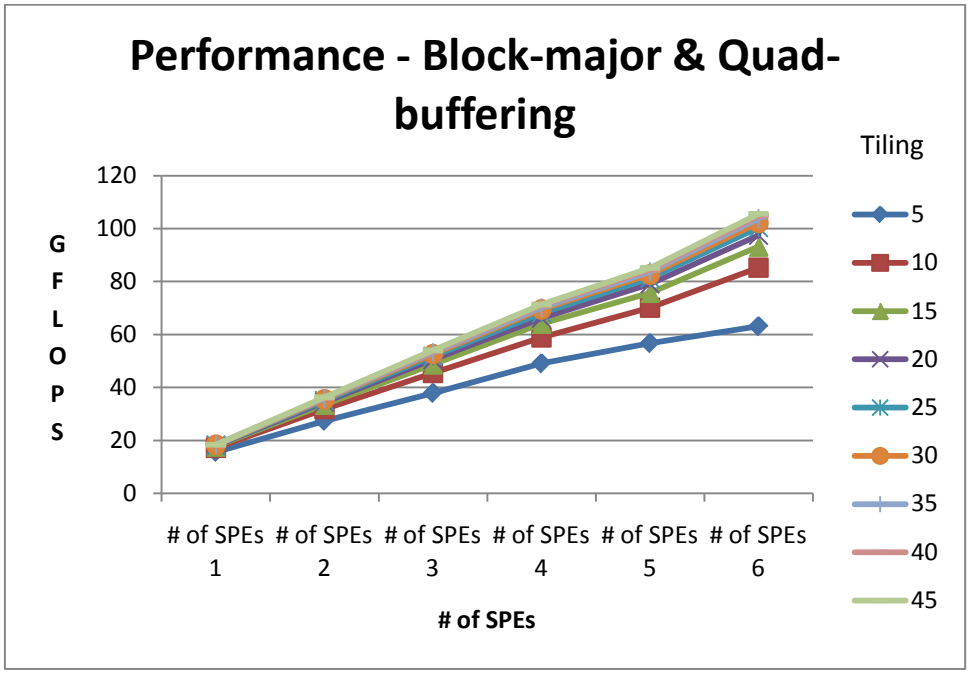
Παρατηρείται ότι η χαμηλή τιμή tiling 5 επιφέρει πτώση της επίδοσης όσο αυξάνεται ο αριθμός των SPEs. Από την άλλη, οι διαφορές μεταξύ μεγεθών tiling σε πολλές SPEs δεν είναι τόσο εμφανής όσο στην απλή έκδοση του προγράμματος. Αποδεικνύεται, έτσι, ότι η εφαρμογή του block-major layout βοηθάει στην αποσυμφόρηση του EIB.





Παρατηρείται ότι σε αυτήν την περίπτωση, ακόμα και για μικρό αριθμό χρησιμοποιούμενων SPEs, η αύξηση στην επίδοση που επιφέρει ένα μεγάλο μέγεθος tiling είναι αξιοσημείωτη.

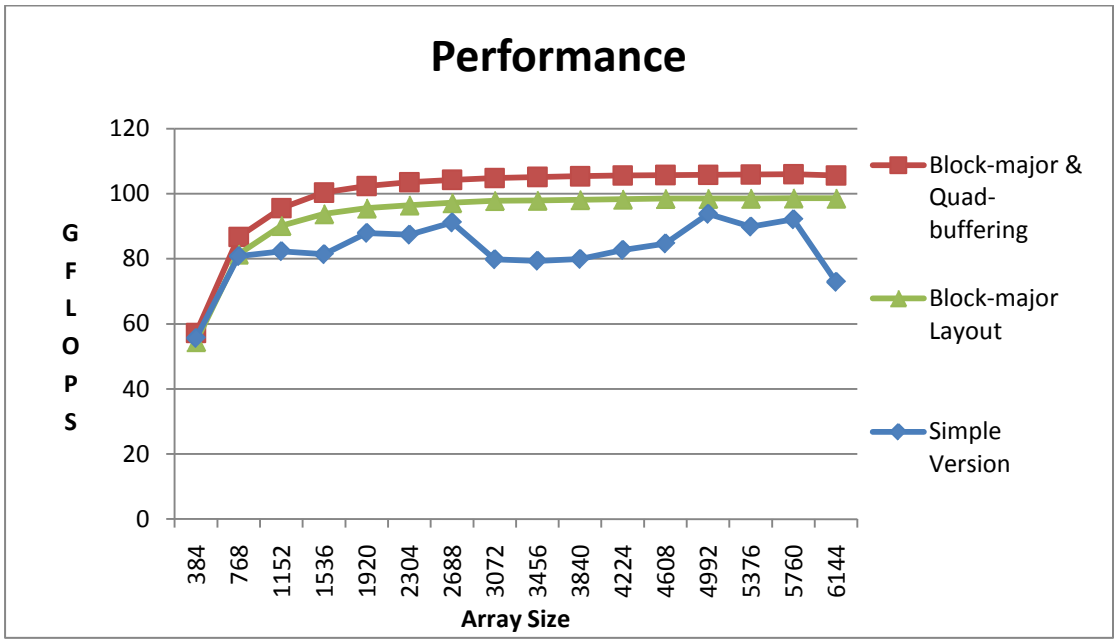




Και πάλι φαίνεται το κέρδος που αποκομίζεται από τη χρήση του block-major layout προς αποσυμφόρηση του EIB.

Πραγματοποιήθηκε άλλη μία σειρά μετρήσεων της επίδοσης κάθε έκδοσης της εφαρμογής συναρτήσει του μεγέθους του πίνακα-χωρίου. Συγκεκριμένα, μελετήθηκαν τα τετράγωνα χωρία 384 x 384, 768 x 768, 1.152 x 1.152, 1.536 x 1.536, 1.920 x 1.920, 2.304 x 2.304, 2.688 x 2.688, 3.072 x 3.072, 3.456 x 3.456, 3.840 x 3.840, 4.224 x 4.224, 4.608 x 4.608, 4.992 x 4.992, 5.376 x 5.376, 5.760 x 5.760 και 6.144 x 6.144 με χρήση έξι SPEs και

12.600 επαναλήψεων. Στο ακόλουθο διάγραμμα φαίνεται η επίδοση για όλες τις εκδόσεις του προγράμματος, εκτός από την έκδοση με το quadruple-buffering, όπου δεν κρίθηκε σκόπιμη η περεταίρω διερεύνησή της.



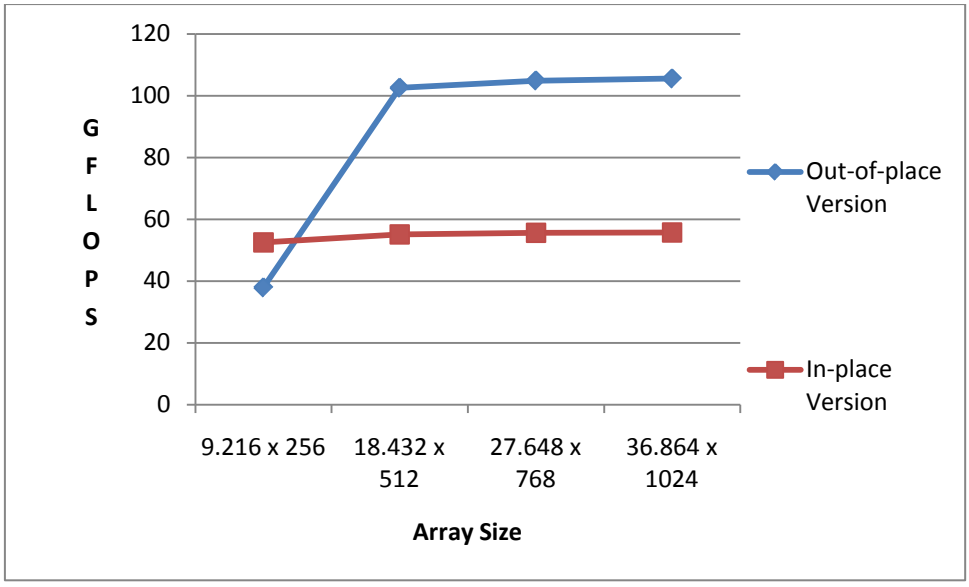
Όπως προκύπτει από το παραπάνω διάγραμμα, η απλή έκδοση της εφαρμογής παρουσιάζει ασταθή συμπεριφορά ως προς την επίδοση για τα διάφορα μεγέθη πινάκων. Αυτό μπορεί να οφείλεται στο συνωστισμό του EIB, γεγονός που δίνει στις εκτελέσεις του προγράμματος μία μη ντετερμινιστική συμπεριφορά. Από την άλλη πλευρά, οι εκδόσεις με block-major layout στην αναπαράσταση του πίνακα-χωρίου εμφανίζουν την αναμενόμενη συμπεριφορά.

Σύγκριση των αλγορίθμων in-place και out-of-place

Οι δύο αλγόριθμοι που εξετάστηκαν διαφέρουν τόσο ως προς την ταχύτητα εκτέλεσης στη CBE όσο και ως προς την ταχύτητα σύγκλισης. Ακολουθεί μία σύγκριση αυτών των δύο παραμέτρων.

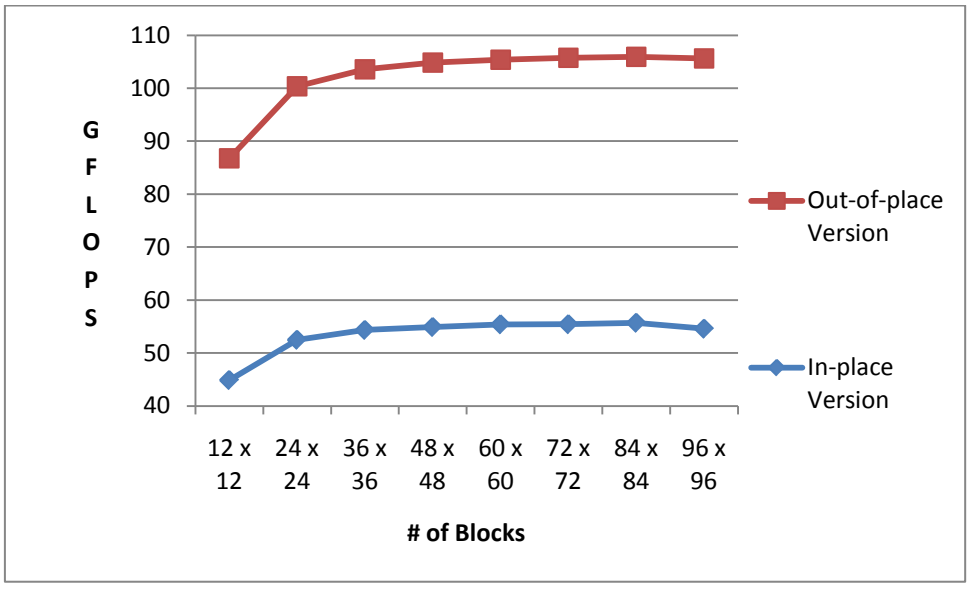
Σύγκριση ως προς την επίδοση

Για την καλύτερη οπτική σύγκριση των δύο αλγορίθμων υπολογισμού της εξίσωσης διάχυσης ακολουθεί μία σειρά μετρήσεων της επίδοσης σε πίνακες διαστάσεων 9.216 x 256, 18.432 x 512, 27.648 x 768 και 36.864 x 1024, με έξι SPEs και 12.600 επαναλήψεις.



Παρατηρείται ότι, εκτός από πολύ μικρά μεγέθη πίνακα, ο αλγόριθμος out-of-place (block-major layout & quadruple-buffering) επιτυγχάνει σχεδόν διπλάσια επίδοση από τον αλγόριθμο in-place. Τα «ιδιότροπα» αυτά μεγέθη έχουν επιλεγεί σύμφωνα με τις απαιτήσεις του αλγορίθμου in-place, παραμένοντας συμβατά με τις απαιτήσεις του αλγορίθμου out-of-place.

Μία πιο αντικειμενική σύγκριση των δύο αλγορίθμων προκύπτει από την εκτέλεση των προγραμμάτων σε πίνακες όχι κοινού για τους δύο αλγορίθμους μεγέθους, αλλά κοινού πλέγματος blocks.



Συγκεκριμένα, πραγματοποιήθηκαν μετρήσεις σε πλέγματα blocks 12 x 12, 24 x 24, 36 x 36, 48 x 48, 60 x 60, 72 x 72, 84 x 84 και 96 x 96, με έξι SPEs και 12.600 επαναλήψεις. Και σε

αυτήν την περίπτωση φαίνεται ότι ο αλγόριθμος out-of-place (block-major layout & quadruple-buffering) επιτυγχάνει σχεδόν διπλάσια υπολογιστική επίδοση από τον αλγόριθμο in-place.

Σύγκριση ως προς την ταχύτητα σύγκλισης

Το μειονέκτημα του out-of-place αλγορίθμου είναι η πιο αργή ταχύτητα σύγκλισης στο πραγματικό αποτέλεσμα. Για τη διερεύνηση της ταχύτητας αυτής πραγματοποιήθηκαν δύο μετρήσεις, μία σε πίνακα 3.072 x 3.072 και μία σε πίνακα 6.144 x 6.144, με ακαθόριστο αριθμό επαναλήψεων. Ως κριτήριο για τον τερματισμό του προγράμματος τέθηκαν οι εξής δύο έλεγχοι:

- Η τιμή του πιο απομακρυσμένου σημείου του χωρίου να φτάσει τουλάχιστον στο 60% της πραγματικής τιμής

- Η διαφορά της τιμής του ίδιου σημείου μεταξύ δύο διαδοχικών επαναλήψεων να μην υπερβαίνει το 10^{-6}

Τα αποτελέσματα της σύγκρισης φαίνονται στον παρακάτω πίνακα:

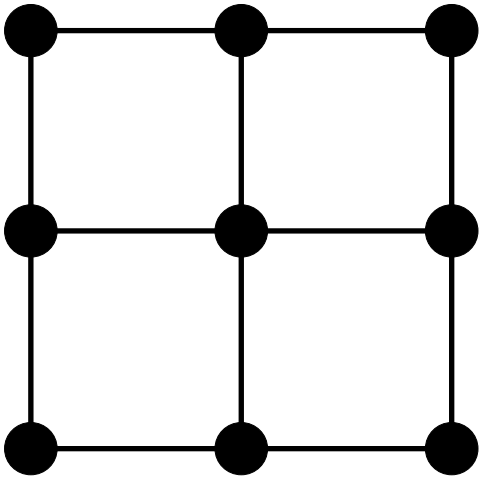
	In-place		**Out-of-place**	
	Επαναλήψεις	Επίδοση	Επαναλήψεις	Επίδοση
3.072 x 3.072	6.615	55.26GFLOPS	12.780	104.79GFLOPS
6.144 x 6.144	12.375	55.90GFLOPS	24.615	105.62GFLOPS

Όπως φαίνεται, ο αλγόριθμος out-of-place απαιτεί σχεδόν διπλάσιες επαναλήψεις για την εξαγωγή αποτελέσματος από ότι ο αλγόριθμος in-place. Από την άλλη, ο αλγόριθμος out-of-place επιτυγχάνει σχεδόν διπλάσια υπολογιστική επίδοση. Τελικά, οι δύο αλγόριθμοι θα φτάσουν στο πραγματικό αποτέλεσμα σχεδόν στον ίδιο χρόνο. Για τον in-place αλγόριθμο, όμως, υπάρχει η μελλοντική βελτίωση με χρήση συγκεντρωτικής επεξεργασίας επαναλήψεων, η οποία θεωρητικά διπλασιάζει την υπολογιστική του επίδοση στη Cell Broadband Engine. Εάν αυτή η θεωρία επαληθευτεί στην πράξη έστω και μερικώς, ο αλγόριθμος in-place θα προηγείται σαφώς του αλγορίθμου out-of-place.

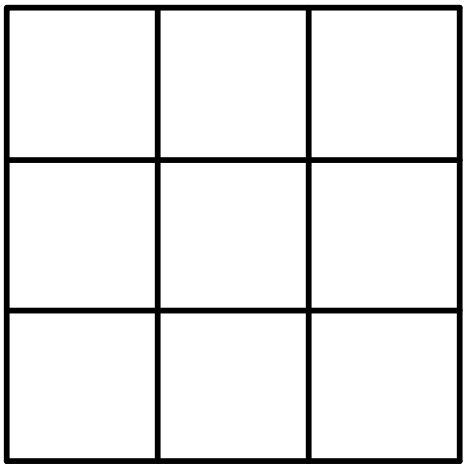
Η εξίσωση διάχυσης σε επίπεδο συστοιχίας από CBEs

Διαμέριση του προβλήματος

Στη υλοποίηση του προβλήματος της εξίσωσης διάχυσης σε επίπεδο συστοιχίας από Cell Broadband Engines ακολουθήθηκε η εξής στρατηγική ως προς τον χωρισμό πίνακα-χωρίου: Ανάλογα με το πλήθος των κόμβων που απαρτίζουν τη συστοιχία, προκύπτει μία διάταξή τους σε ένα πλέγμα κόμβων (node grid). Παρακάτω δίνεται μία διάταξη εννέα κόμβων.

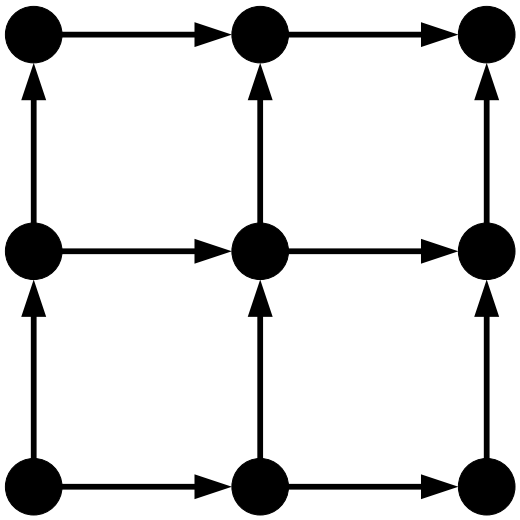


Αναλόγως της μορφής του πλέγματος που προκύπτει, ο πίνακας-χωρίο μοιράζεται σε αντίστοιχα κομμάτια. Στη διάταξη των εννέα κόμβων που φαίνεται παραπάνω ο πίνακας διαμερίζεται με τον παρακάτω τρόπο.



Κάθε ένας από του επεξεργαστές του υπολογιστικού πλέγματος αναλαμβάνει ένα κομμάτι του πίνακα-χωρίου πάνω στο οποίο θα εκτελέσει του υπολογισμούς. Οι εξαρτήσεις που προκύπτουν μεταξύ γειτονικών επεξεργαστών επιβάλλουν τη μεταξύ τους επικοινωνία. Συγκεκριμένα, απαιτείται η αποστολή και λήψη συνόρων κατά μήκος κάθε διάστασης όπου

συνορεύουν γειτονικοί κόμβοι, ακριβώς όπως ισχύει μεταξύ των SPEs. Δηλαδή, κάθε κόμβος λαμβάνει σύνορα από τους γειτονικούς του κόμβους προς τα κάτω και αριστερά και στέλνει σύνορα στους γειτονικούς του προς τα πάνω και δεξιά.



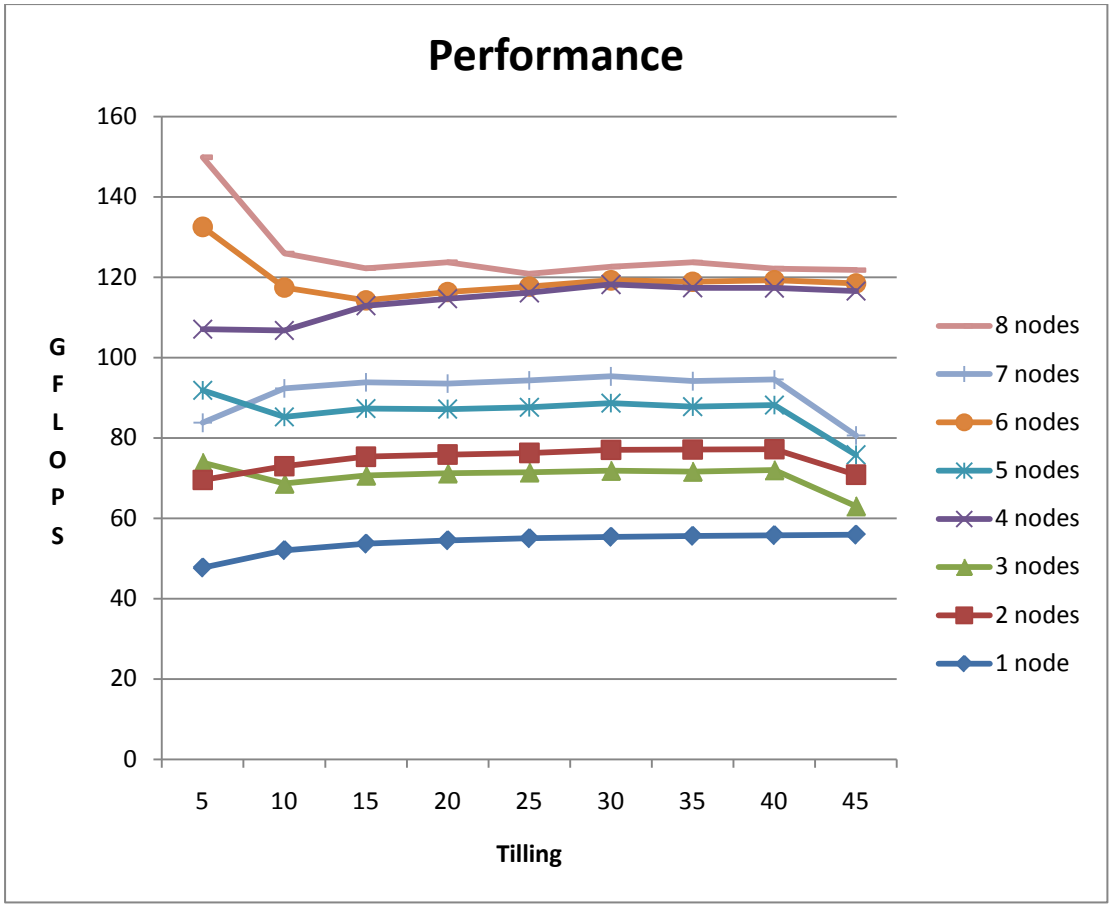
Εντός του κάθε κόμβου, το πρόβλημα υπολογίζεται όπως ακριβώς έχει περιγραφεί παραπάνω, σε επίπεδο SPEs.

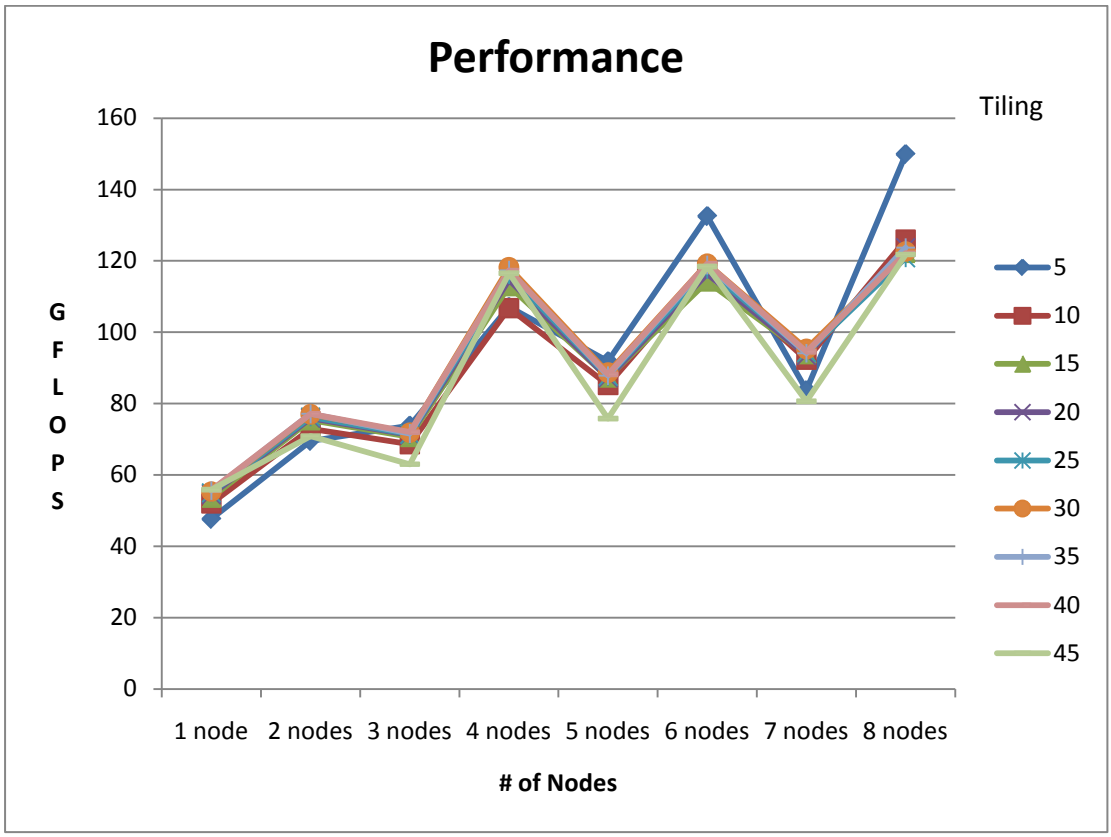
Μετρήσεις σε επίπεδο συστοιχίας

Ακολουθούν οι μετρήσεις της επίδοσης των δύο αλγορίθμων, in-place και out-of-place, στη συστοιχία των CBEs.

Αλγόριθμος in-place

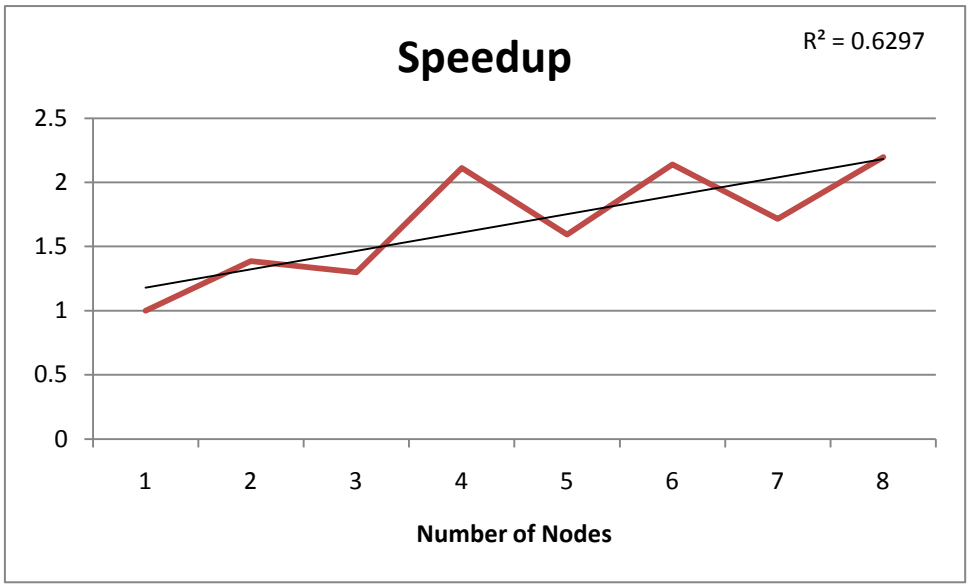
Για την αξιολόγηση της επίδοσης της εφαρμογής του in-place αλγορίθμου σε επίπεδο συστοιχίας, πραγματοποιήθηκε σε πρώτη φάση μία σειρά μετρήσεων για τη διερεύνηση του παράγοντα του tiling. Συγκεκριμένα, μετρήθηκε η επίδοση σε πίνακα-χωρίο διαστάσεων 6.144 x 6.144 με έναν έως οκτώ κόμβους MPI, με έξι SPEs ανά κόμβο και 12.600 επαναλήψεις, για τιμές του tiling 5, 10, 15, 20, 25, 30, 35, 40 και 45. Ακολουθούν διαγράμματα της επίδοσης συναρτήσει του tiling για έναν έως οκτώ κόμβους και της επίδοσης συναρτήσει των κόμβων για τις προαναφερόμενες τιμές tiling σε κάθε περίπτωση.


Performance
160
140
120
100
80
60
40
20
0
GFLOPS
5
10
15
20
25
30
35
40
45
Tilling
8 nodes
7 nodes
6 nodes
5 nodes
4 nodes
3 nodes
2 nodes
1 node

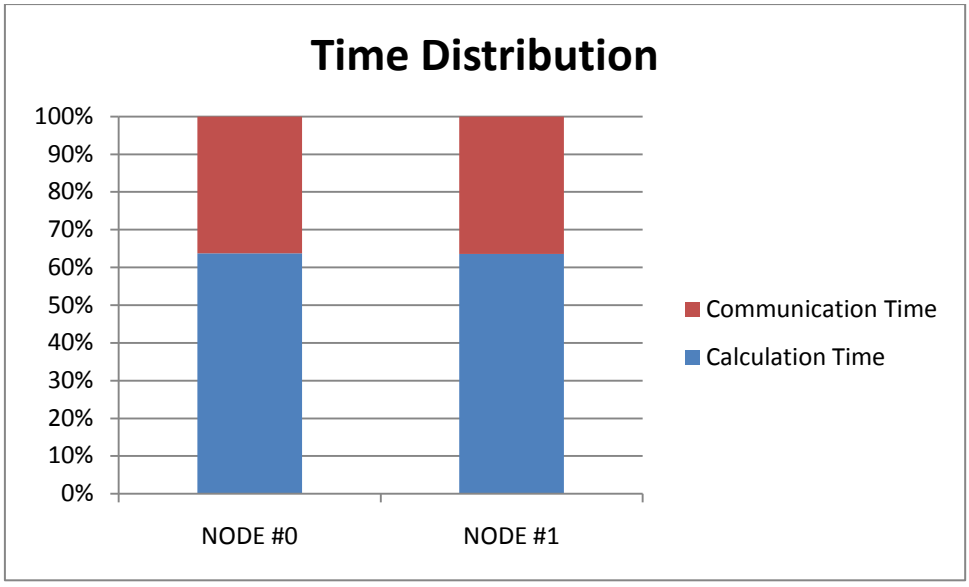

Performance
160
140
120
100
80
60
40
20
0
GFLOPS
1 node
2 nodes
3 nodes
4 nodes
5 nodes
6 nodes
7 nodes
8 nodes
# of Nodes
Tiling
5
10
15
20
25
30
35
40
45

Όπως φαίνεται από τα παραπάνω διαγράμματα, η επίδοση είναι μειωμένη για 3, 5 και 7 κόμβους λόγω του γεγονότος ότι σε αυτές τις τοπολογίες το υπολογιστικό πλέγμα (node grid) εκφυλίζεται σε απλή ακολουθία κόμβων. Μία ενδιαφέρουσα παρατήρηση είναι ότι όσο αυξάνονται οι κόμβοι του συστήματος, πράγμα που σημαίνει ότι αυξάνεται και η καθυστέρηση έναρξης των υπολογισμών σε κάθε κόμβο, μειώνεται η καταλληλότερη τιμή του tiling για την οποία επιτυγχάνεται η μέγιστη επίδοση. Για παράδειγμα, στην περίπτωση των δύο κόμβων η μέγιστη επίδοση επιτυγχάνεται για τιμή του tiling ίση με 40, ενώ στην περίπτωση των οκτώ κόμβων η επίδοση μεγιστοποιείται για tiling ίσο με 5.

Επίσης, μετρήθηκε η επιτάχυνση συναρτήσει του αριθμού των κόμβων στο σύστημα, από έναν έως οκτώ κόμβους MPI, για πίνακα 6.144 x 6.144, με έξι SPEs ανά κόμβο, 12.600 επαναλήψεις και μία ενδιάμεση τιμή του tiling, δηλαδή 25. Παρατηρείται η αυξομείωση στην επίδοση λόγω των ιδιαίτερων τιμών για το πλήθος των κόμβων 3, 5 και 7. Η επιτάχυνση δεν είναι πλέον γραμμική, αλλά ο συντελεστής συσχέτισης έχει μειωθεί (R^2 = 0.6297). Παρατηρείται, επίσης, ότι η μέγιστη επιτάχυνση, η οποία επιτυγχάνεται με τη χρήση οκτώ κόμβων, είναι μόλις 2.2



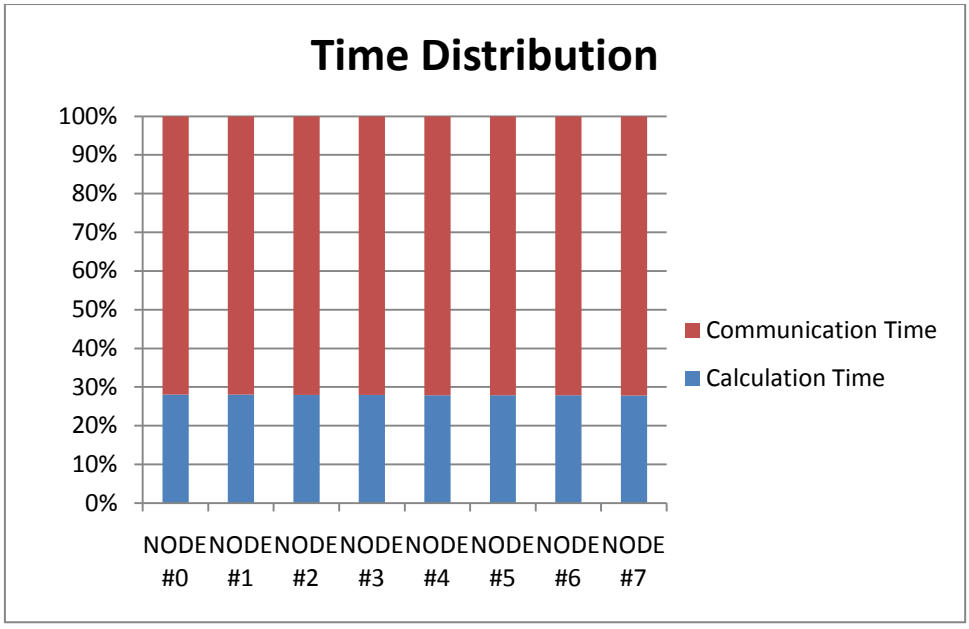
Για να βρεθεί η μέγιστη δυνατή επιτάχυνση, καθώς και το πλήθος των κόμβων που έχει νόημα να χρησιμοποιηθούν για να επιτευχθεί αυτή, παρατίθεται το διάγραμμα κατανομής χρόνου στην περίπτωση της χρήσης δύο κόμβων, στην ίδια εκτέλεση που αναφέρθηκε παραπάνω.



Από την κατανομή αυτή προκύπτει ότι ο μέγιστος βαθμός παραλληλοποίηση του προβλήματος μεταξύ δύο κόμβων είναι περίπου 63.6%, όσο δηλαδή το ποσοστό του χρόνου που αφιερώνεται στους υπολογισμούς. Από το νόμο του Amdahl:

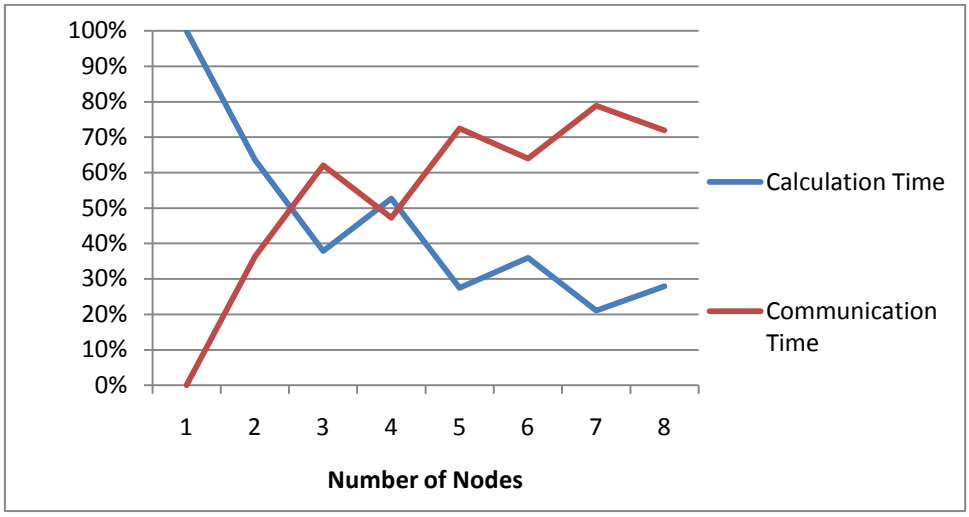
$SpeedUp \leq \frac{1}{(1-P)+\frac{P}{N}}$, P = ποσοστό παραλληλοποίησης εφαρμογής, N = πλήθος κόμβων,

προκύπτει ότι η μέγιστη δυνατή επιτάχυνση σε ένα τέτοιο σύστημα και με ένα τέτοιο πρόβλημα είναι (για N = ∞) 2.75, που αντιστοιχεί σε επίδοση 151.25GFLOPS. Αυτό συνεπάγεται ότι για την επίτευξη επιτάχυνσης ίσης με 2.7 απαιτούνται 100 κόμβοι στο σύστημα.



Ενδεικτικά, παρατίθεται το διάγραμμα κατανομής χρόνου για την περίπτωση χρήσης οκτώ κόμβων σε κάθε κόμβο της ίδιας εκτέλεσης, από όπου φαίνεται ότι ήδη κυριαρχεί η επικοινωνία μεταξύ των επεξεργαστών του πλέγματος έναντι των υπολογισμών.

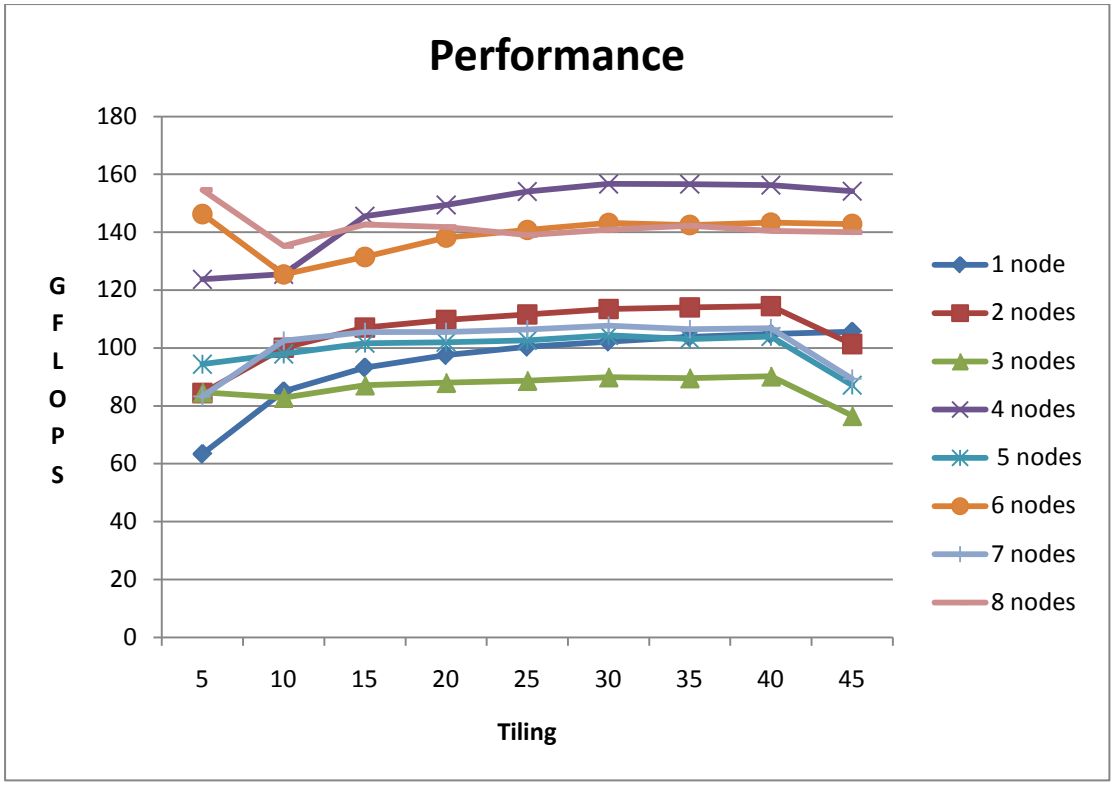
Τέλος, παρατίθεται το διάγραμμα που δείχνει τη μέση κατανομή επικοινωνίας-υπολογισμών για όλες τις περιπτώσεις πλήθους κόμβων που εξετάστηκαν.

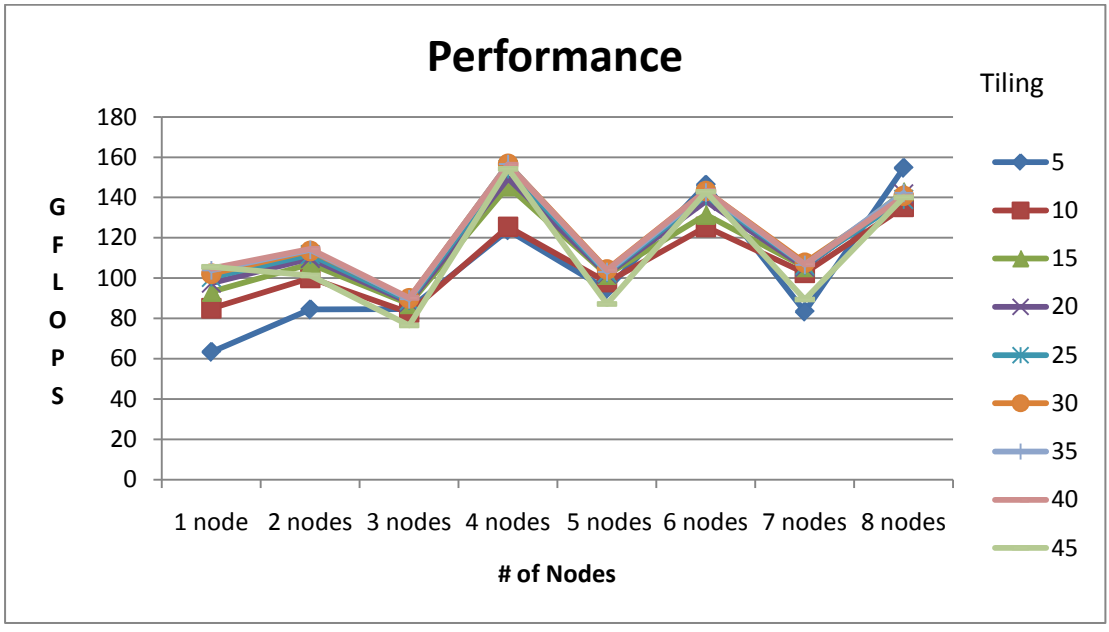


Αλγόριθμος out-of-place

Για την αξιολόγηση της επίδοσης της εφαρμογής του out-of-place αλγορίθμου, έκδοση με block-major layout και quadruple-buffering, σε επίπεδο συστοιχίας, πραγματοποιήθηκε σε πρώτη φάση μία σειρά μετρήσεων για τη διερεύνηση του παράγοντα του tiling. Συγκεκριμένα, μετρήθηκε η επίδοση σε πίνακα-χωρίο διαστάσεων 6.144 x 6.144 με έναν έως οκτώ κόμβους MPI, με έξι SPEs ανά κόμβο και 12.600 επαναλήψεις, για τιμές του tiling 5, 10, 15, 20, 25, 30, 35, 40 και 45. Ακολουθούν διαγράμματα της επίδοσης συναρτήσει του tiling για έναν έως οκτώ κόμβους και της επίδοσης συναρτήσει των κόμβων για τις προαναφερόμενες τιμές tiling σε κάθε περίπτωση.

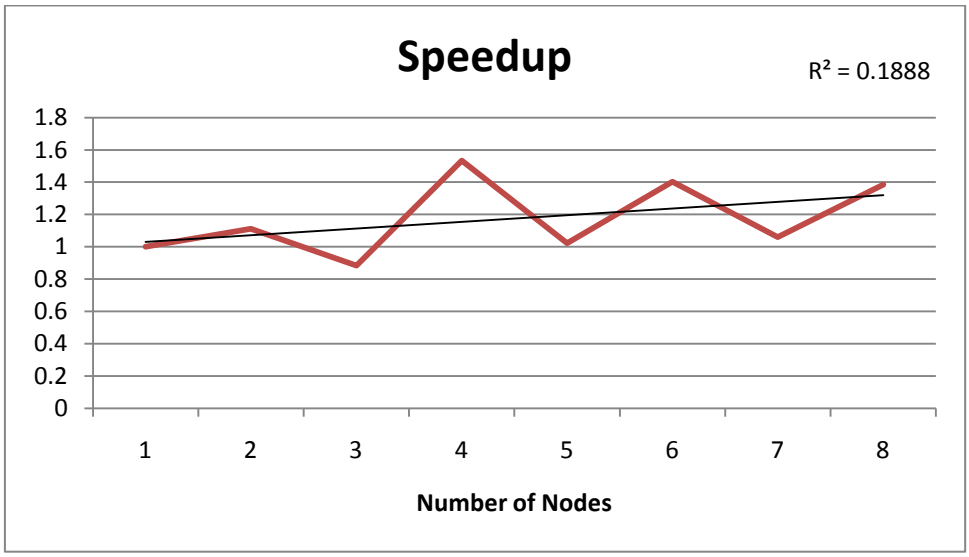
Όπως φαίνεται από τα παρακάτω διαγράμματα, η επίδοση είναι μειωμένη για 3, 5 και 7 κόμβους, όπως και στην περίπτωση του in-place αλγορίθμου.



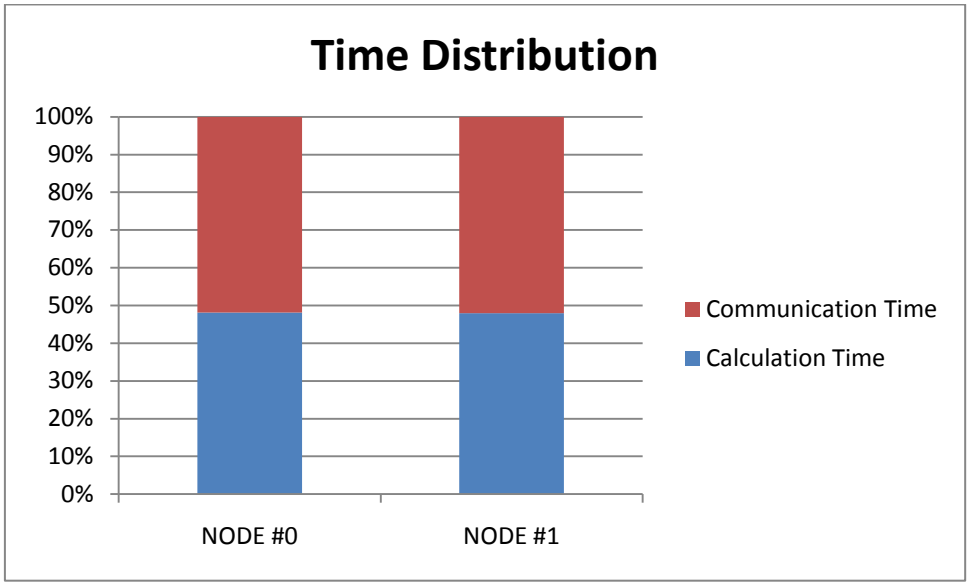


Επίσης, μετρήθηκε η επιτάχυνση συναρτήσει του αριθμού των κόμβων στο σύστημα, από έναν έως οκτώ κόμβους MPI, για πίνακα 6.144 x 6.144, με έξι SPEs ανά κόμβο, 12.600 επαναλήψεις και μία ενδιάμεση τιμή του tiling, δηλαδή 25. Παρατηρείται η αυξομείωση στην επίδοση λόγω των ιδιαίτερων τιμών για το πλήθος των κόμβων 3, 5 και 7. Η επιτάχυνση δεν είναι πλέον γραμμική, αλλά ο συντελεστής συσχέτισης έχει μειωθεί (R^2 = 0.1888).

Παρατηρείται, επίσης, ότι η μέγιστη επιτάχυνση, η οποία επιτυγχάνεται με τη χρήση τεσσάρων κόμβων, είναι μόλις 1.4, ενώ υπάρχουν και τιμές κάτω της μονάδας.



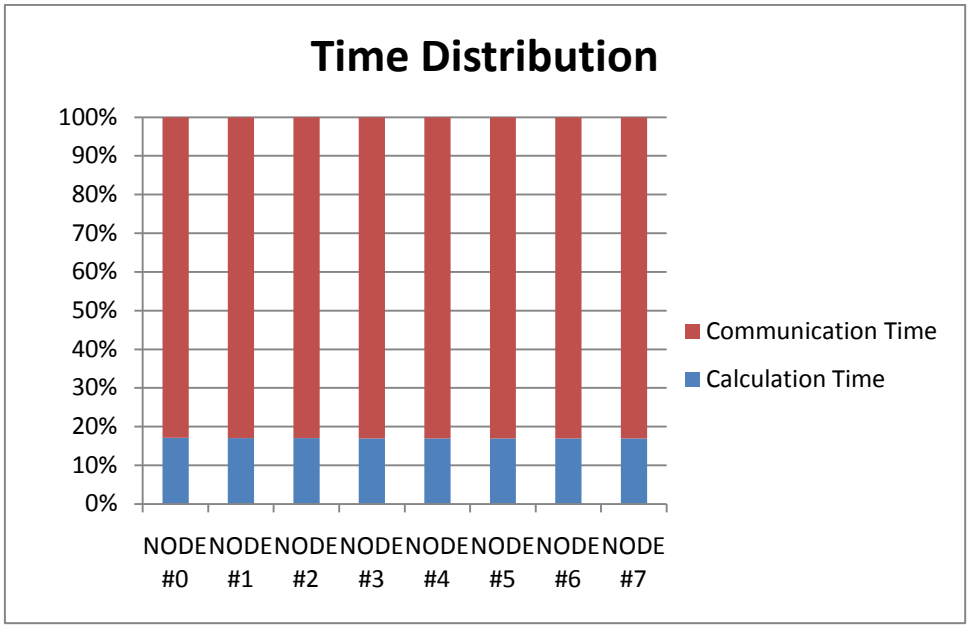
Για να βρεθεί η μέγιστη δυνατή επιτάχυνση, καθώς και το πλήθος των κόμβων που έχει νόημα να χρησιμοποιηθούν για να επιτευχθεί αυτή, παρατίθεται το διάγραμμα κατανομής χρόνου στην περίπτωση της χρήσης δύο κόμβων, στην ίδια εκτέλεση που αναφέρθηκε παραπάνω.



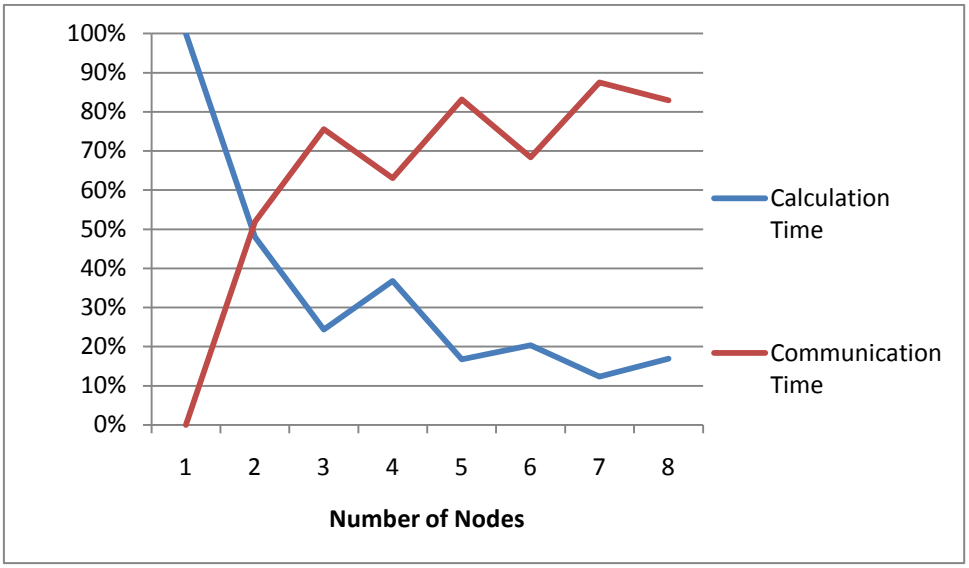
Από την κατανομή αυτή προκύπτει ότι ο μέγιστος βαθμός παραλληλοποίηση του προβλήματος μεταξύ δύο κόμβων είναι περίπου 48.0%, όσο δηλαδή το ποσοστό του χρόνου που αφιερώνεται στους υπολογισμούς. Η διαφορά αυτού του ποσοστού από το αντίστοιχο του αλγορίθμου in-place εξηγείται λαμβάνοντας υπ' όψιν ότι ο out-of-place αλγόριθμος είναι εγγενώς ταχύτερος υπολογιστικά στην εκτέλεση, οπότε οι καθυστερήσεις για την επικοινωνία

στο MPI δεν κρύβονται τόσο καλά. Από το νόμο του Amdahl προκύπτει ότι η μέγιστη δυνατή επιτάχυνση σε ένα τέτοιο σύστημα και με ένα τέτοιο πρόβλημα είναι 1.92, , που αντιστοιχεί σε επίδοση 192.79GFLOPS. Αυτό συνεπάγεται ότι για την επίτευξη επιτάχυνσης ίσης με 1.9 απαιτούνται 76 κόμβοι στο σύστημα.

Ενδεικτικά, παρατίθεται το διάγραμμα κατανομής χρόνου για την περίπτωση χρήσης οκτώ κόμβων σε κάθε κόμβο της ίδιας εκτέλεσης, από όπου φαίνεται ότι ήδη κυριαρχεί η επικοινωνία μεταξύ των επεξεργαστών του πλέγματος έναντι των υπολογισμών σε ποσοστό μεγαλύτερο του 80%.

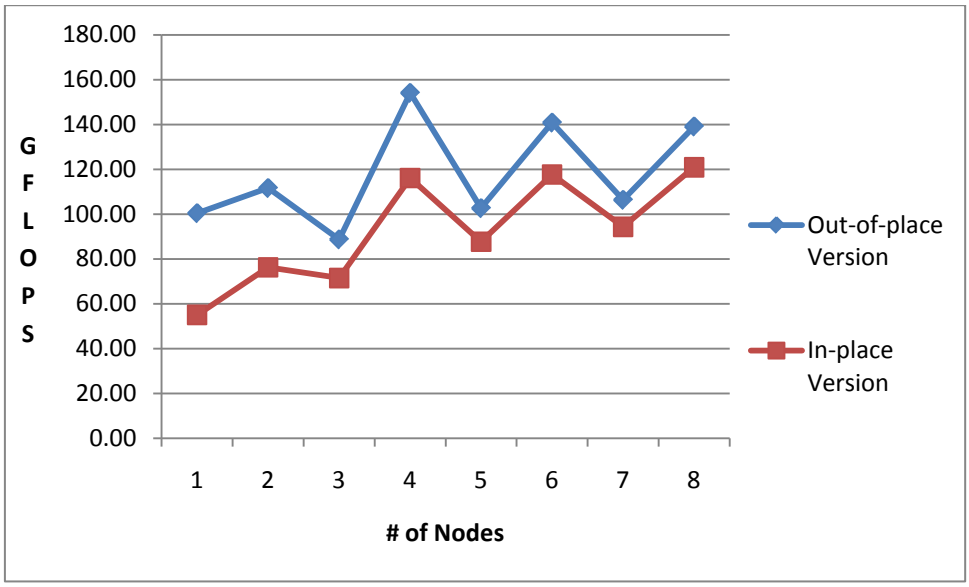


Τέλος, παρατίθεται το διάγραμμα που δείχνει τη μέση κατανομή επικοινωνίας-υπολογισμών για όλες τις περιπτώσεις πλήθους κόμβων που εξετάστηκαν.



Σύγκριση αλγορίθμων in-place και out-of-place σε επίπεδο συστοιχίας

Πέρα από τη σύγκριση των δύο αλγορίθμων ως προς την ταχύτητα σύγκλισης, ενδιαφέρον παρουσιάζει και η σύγκρισή τους σε επίπεδο συστοιχίας. Βάζοντας μαζί τα διαγράμματα επίδοσης συναρτήσει του πλήθους των κόμβων, για έναν έως οκτώ κόμβους, σε χωρίο 6.144 x 6.144, με έξι SPEs ανά κόμβο, tiling ίσο με 25 και 12.600 επαναλήψεις προκύπτει το παρακάτω διάγραμμα:



Παρατηρείται ότι η υπολογιστική επίδοση των δύο αλγορίθμων τείνει να συγκλίνει όσο αυξάνεται το πλήθος των κόμβων. Λαμβάνοντας, όμως, υπ' όψιν ότι ο αλγόριθμος in-place έχει σχεδόν διπλάσια ταχύτητα σύγκλισης από τον αλγόριθμο out-of-place προκύπτει και πάλι ότι η in-place προσέγγιση είναι η πιο συμφέρουσα. Για την ακρίβεια, η μέγιστη θεωρητική επίδοση του in-place αλγορίθμου είναι τα 151.25GFLOPS, ενώ του out-of-place αλγορίθμου μακράν απέχει από το να είναι διπλάσια (192.79GFLOPS), άρα ο αλγόριθμος in-place είναι ξεκάθαρα ο καλύτερος σε επίπεδο συστοιχίας.

Συμπεράσματα

Η εξίσωση διάχυσης είναι ένα πρόβλημα με σχετικά μικρή υπολογιστική πολυπλοκότητα και έντονες εξαρτήσεις δεδομένων. Ως επακόλουθο, η παραλληλοποίησή της παρουσιάζει δυσκολίες και η επίτευξη υψηλής επίδοσης απαιτεί μεγαλύτερη διερεύνηση. Ένας τρόπος για την επιτάχυνση του in-place αλγορίθμου είναι η εφαρμογή της Συγκεντρωτικής Επεξεργασίας Επαναλήψεων, ούτως ώστε να αξιοποιηθούν καλύτερα οι σωληνώσεις των SPEs. Στην περίπτωση του out-of-place αλγορίθμου η επίδοση είναι σαφώς μεγαλύτερη, αλλά για τη βελτίωσή της απαιτείται η εφαρμογή της σύμβασης block-major layout, προκειμένου να περιοριστεί το overhead των μεταφορών.

Στην περίπτωση της συστοιχίας, οι έντονες εξαρτήσεις δεδομένων καθηλώνουν την επίδοση και το scaling της εφαρμογής σε μεγάλο πλήθος κόμβων δεν έχει ιδιαίτερα αποτελέσματα. Όπως και στον πολλαπλασιασμό πινάκων, για την επίτευξη των παραπάνω επιδόσεων είναι αναγκαία η χειροκίνητη βελτιστοποίηση του κώδικα και η εφαρμογή των τεχνικών που έχουν αναφερθεί (multi-buffering, vectorization, loop unrolling).

Η εξίσωση διάχυσης στην τριδιάστατη μορφή της

Η παραπάνω ανάλυση της εξίσωσης διάχυσης σε δύο διαστάσεις μπορεί να επεκταθεί και στον τριδιάστατο χώρο, όπως επίσης και σε οποιονδήποτε n-διάστατο χώρο. Σε τρεις διαστάσεις, η διακριτή μορφή της εξίσωσης διάχυσης, χρησιμοποιώντας την in-place προσέγγιση, είναι η εξής:

$$\mathrm{U[t][z][y][x]} = \left(1 + 2 \times \mathrm{a} \times \frac{\mathrm{dt}}{\mathrm{dx}}\right) \times \mathrm{U[t-1][z][y][x]} - \mathrm{a} \times \frac{\mathrm{dt}}{\mathrm{dx}} \times (\mathrm{U[t][z-1][y][x]} + \mathrm{U[t][z][y-1][x]} + \mathrm{U[t][z][y][x-1]})$$

Για τον out-of-place αλγόριθμο ισχύει αντίστοιχα ο τύπος:

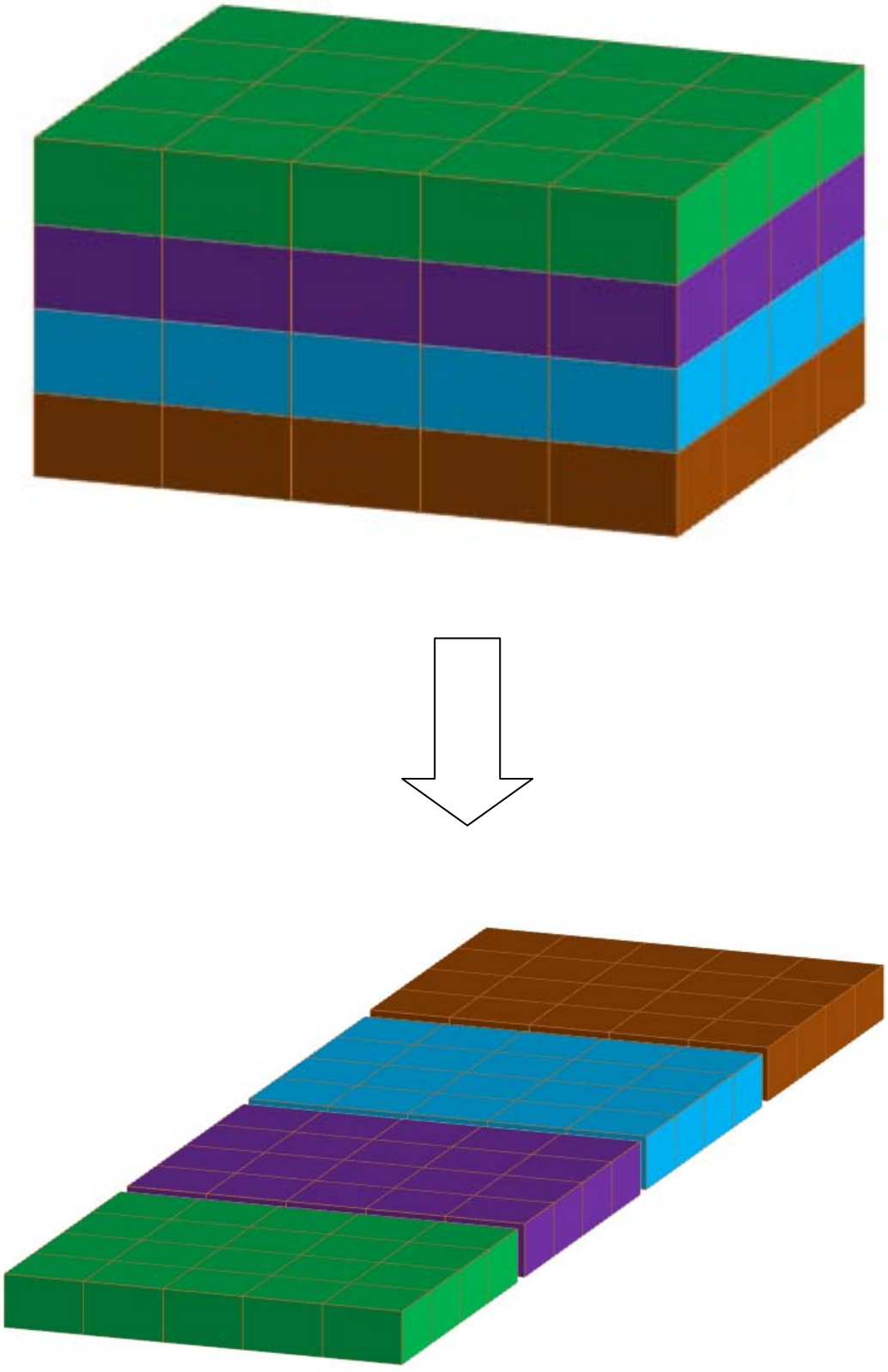
$$\mathrm{U[t][z][y][x]} = \left(1 + 2 \times \mathrm{a} \times \frac{\mathrm{dt}}{\mathrm{dx}}\right) \times \mathrm{U[t-1][z][y][x]} - \mathrm{a} \times \frac{\mathrm{dt}}{\mathrm{dx}} \times (\mathrm{U[t-1][z-1][y][x]} + \mathrm{U[t-1][z][y-1][x]} + \mathrm{U[t-1][z][y][x-1]})$$

Ακολουθεί η θεωρητική ανάλυση του θέματος, η οποία δεν έχει υλοποιηθεί αλλά αφήνεται για μελλοντική διερεύνηση. Μία πρώτη διαφοροποίηση αυτής της περίπτωσης από την αντίστοιχη σε δύο διαστάσεις έχει να κάνει με τις διαστάσεις των blocks που μεταφέρονται. Το πλήθος των στοιχείων που μπορεί να έχει ένα block πρέπει να είναι ένας αριθμός που να συμβιβάζει τα πλεονεκτήματα και τους περιορισμούς της εκάστοτε αρχιτεκτονικής. Οι ιδανικές διαστάσεις έχουν αναλυθεί παραπάνω. Προκειμένου να διατηρηθεί σταθερό το πλήθος αυτό, πρέπει να συρρικνωθούν όλες οι διαστάσεις, ούτως ώστε να «χωρέσει» και μία τρίτη. Αυτό συνιστά ένα πρώτο πρόβλημα για εφαρμογή αλγορίθμων όπως ο in-place, ο οποίος χρειάζεται τουλάχιστον μία κατά το δυνατόν μεγάλη και μία κατά το δυνατόν μικρή διάσταση.

Ένα δεύτερο πρόβλημα έχει να κάνει με τις συνοριακές τιμές που μεταφέρονται μεταξύ των επεξεργαστικών μονάδων. Στην διδιάστατη έκδοση, οι συνοριακές τιμές αποθηκεύονται σε δύο (το πολύ) μονοδιάστατους πίνακες. Από την άλλη, στην τριδιάστατη έκδοση τα σύνορα αποθηκεύονται σε τρεις (το πολύ) διδιάστατους πίνακες-επίπεδα. Ο χώρος, λοιπόν, που απαιτείται στην τοπική μνήμη για τις συνοριακές τιμές αυξάνεται υπέρμετρα.

Για τον in-place αλγόριθμο, η υλοποίηση της εφαρμογής είναι αρκετά πολύπλοκη. Η προσέγγιση της ΣΕΕ γίνεται πλέον υποχρεωτική, με τα μαζικά επίπεδα να αντιπροσωπεύουν τώρα διαφορετικά επίπεδα του χώρου και όχι διαφορετικές επαναλήψεις. Με αυτόν τον τρόπο πρέπει να αντιμετωπιστούν όλα τα προγραμματιστικά προβλήματα της ΣΕΕ, χωρίς να υπάρχει πιο απλή και εξίσου αποδοτική προσέγγιση στο θέμα.

Για την εφαρμογή του out-of-place αλγορίθμου, ο τριδιάστατος πίνακας-χώρος μπορεί να χωριστεί σε blocks-κύβους. Ο υπό μελέτη χώρος δεδομένων μπορεί να χωριστεί σε επίπεδα (αντιστοιχούν στα διαφορετικά χρώματα του σχήματος) με ύψος όσο το ύψος των blocks-κύβων και στη συνέχεια να γίνει ένα «ξεδίπλωμα» των επιπέδων αυτόν προς σχηματισμό ενός μεγάλου «υπέρ-επιπέδου» δύο διαστάσεων. Από εκεί και πέρα, το πρόβλημα ανάγεται σε ένα πρόβλημα εξίσωσης διάχυσης σε δύο διαστάσεις, με τροποποιημένες μόνο τις μεταφορές των συνοριακών τιμών και τον τρόπο υπολογισμού της διάχυσης εντός των blocks, τα οποία πλέον είναι τριδιάστατα και θα πρέπει να χρησιμοποιηθεί η μορφή της εξίσωσης που αναφέρεται παραπάνω.



Κεφάλαιο 7

ΣΥΜΠΕΡΑΣΜΑΤΑ - ΕΠΙΛΟΓΟΣ

Η παρούσα εργασία αποτελεί μία πρώτη προσέγγιση στην εφαρμογή διανυσματικών και παράλληλων υπολογισμών στη Cell Broadband Engine, τόσο σε επίπεδο μίας CBE όσο και σε επίπεδο συστοιχίας από CBEs. Μέσα από τη μελέτη του προγραμματισμού της CBE πάνω στη βάση της αρχιτεκτονικής της (architecture specific programming) προέκυψαν τα ακόλουθα συμπεράσματα.

Συμπεράσματα από τη μελέτη της Cell Broadband Engine

Με την αδυναμία των σύγχρονων compilers για αποδοτική παραλληλοποίηση και διανυσματοποίηση αλγορίθμων, η επίτευξη της μέγιστης επίδοσης από τον επεξεργαστή προϋποθέτει τη χειροκίνητη βελτιστοποίηση του κώδικα από μέρους του προγραμματιστή. Τεχνικές όπως το ξεδίπλωμα βρόχων, το inlining συναρτήσεων, η εξάλειψη διακλαδώσεων, η χειροκίνητη αναδιάταξη και δρομολόγηση εντολών και η κατά το δυνατόν χρήση σταθερών εκφράσεων (π.χ. χρήση σταθερών και όχι μεταβλητών στους δείκτες των πινάκων) είναι απαραίτητες για την επίτευξη μέγιστης επίδοσης, παρ' όλο που αυξάνεται κατακόρυφα η δυσκολία προγραμματισμού και ο απαιτούμενος χρόνος ανάπτυξης των εφαρμογών. Η χρήση παραμέτρων (flags) βελτιστοποίησης στον compiler ελάχιστα ωφελεί την επίδοση, ενώ κατά περιπτώσεις τα αποτελέσματα είναι αρνητικά.

Ένα ενδιαφέρον εύρημα έχει να κάνει με την τεχνική του block-major layout για την αναπαράσταση στη μνήμη των δεδομένων. Σε περιπτώσεις όπου το πλήθος των μεταφορών γίνεται μεγάλο, η χρήση της συγκεκριμένης τεχνικής βοηθάει στην αποσυμφόρηση του EIB και στην καλύτερη αξιοποίηση του εύρους ζώνη, κατ' επέκταση συμβάλλει σε μεγάλο βαθμό στην αύξηση της επίδοσης της εφαρμογής.

Από τη μελέτη του προβλήματος της εξίσωσης διάχυσης σε δύο διαστάσεις προέκυψαν τα οφέλη από την ελαχιστοποίηση των διανυσμάτων εξάρτησης. Στην κλασσική προσέγγιση της παραλληλοποίησης του προβλήματος, όπου το χωρίο υπολογισμού χωρίζεται σε πλέγμα επεξεργαστικών μονάδων, υπάρχουν διανύσματα εξάρτησης και προς τις δύο διαστάσεις. Στην υλοποίηση της εφαρμογής σε επίπεδο CBE, ο τρόπος χωρισμού του χωρίου αφήνει μόνο ένα διάνυσμα εξάρτησης, αυτό κατά μήκος της κατακόρυφης διάστασης. Τα οφέλη αυτού του τρόπο φαίνονται ξεκάθαρα στα διαγράμματα κατανομής χρόνων, όπου οι επικοινωνίες μεταξύ επεξεργαστικών μονάδων καταλαμβάνουν ελάχιστα ποσοστά. Αντιθέτως, στην υλοποίηση σε επίπεδο συστοιχίας, όπου ακολουθήθηκε η κλασσική προσέγγιση με δύο διανύσματα εξάρτησης, τα αποτελέσματα ήταν τελείως διαφορετικά.

Ένα ακόμη χρήσιμο συμπέρασμα αφορά στη διαφορετική φύση των επικοινωνιών μεταξύ SPEs και μεταξύ κόμβων μίας συστοιχίας. Αυτό αντικατοπτρίζεται στα αντικρουόμενα «συμφέροντα» για το μέγεθος του tiling. Σε επίπεδο επικοινωνίας μεταξύ των SPEs, μεγαλύτερο βήμα tiling συνεπάγεται βελτίωση της επίδοσης. Απεναντίας, στο επίπεδο κόμβων της συστοιχίας η αύξηση του παράγοντα tiling έχει αρνητικά αποτελέσματα από κάποιο μέγεθος και μετά. Αυτή η διαπίστωση είναι σύμφωνη και με το εντελώς διαφορετικό εύρος ζώνης που υπάρχει για την επικοινωνία των επεξεργαστικών μονάδων σε κάθε επίπεδο. Τα SPEs επικοινωνούν μέσω ενός εύρους ζώνης που θεωρητικά φτάνει τα 204.8GB/s τη στιγμή που οι κόμβοι της συστοιχίας περιορίζονται στη διαμεταγωγή των 125MB/s που προσφέρει το Gigabit Ethernet.

Ένα τελευταίο συμπέρασμα, το οποίο είναι γενικότερη παρατήρηση και δεν αφορά αποκλειστικά στην Cell Broadband Engine, είναι ότι ο αλγόριθμος in-place για τον υπολογισμό της εξίσωσης διάχυσης στις δύο διαστάσεις, αν και σε πρώτη φάση φαίνεται δύσχρηστος και αργός, αποδεικνύεται τελικά ότι συμφέρει έναντι του αλγορίθμου out-of-place, τόσο σε επίπεδο μονού κόμβου όσο και σε επίπεδο συστοιχίας.

Σύνοψη των βελτιστοποιήσεων που εφαρμόζονται στα προγράμματα

Βελτιστοποίηση	Περιγραφή	Αποτέλεσμα
Vectorization	Χρήση διανυσματικών (SIMD) εντολών και τύπων δεδομένων	Εκτέλεση πολλαπλών πράξεων με μία εντολή και περιορισμός του overhead των βαθμωτών
Loop-unrolling & Instruction re-scheduling	Ξεδίπλωμα βρόχων, αναδιάταξη εντολών, αποδοτική αξιοποίηση των ετερογενών pipelines	Περιορισμός διακλαδώσεων, έγκαιρη φόρτωση/αποθήκευση δεδομένων που θα χρειαστούν σε μετέπειτα επαναλήψεις, συνεχής και αδιάλειπτη τροφοδότηση του άρτιου pipeline
Multi-buffering	Χρήση πολλαπλών buffers για επικάλυψη υπολογισμών - επικοινωνίας	Περιορισμός του χρόνου κατά τον οποίον η επεξεργαστική μονάδα μένει αδρανής, περιμένοντας τη μεταφορά δεδομένων
Block-major Layout	Απεικόνιση όλων των στοιχείων ενός block σε συνεχόμενες θέσεις μνήμης	Μεταφορά του block ως ένα ενιαίο κομμάτι, περιορισμός του πλήθους των μεταφορών κατά έναν παράγοντα όσες και οι γραμμές του block
Tiling	Ανταλλαγή συνόρων από πολλές επαναλήψεις με μία μεταφορά	Μείωση του πλήθους των μεταφορών συνοριακών τιμών και του πλήθους των μεταφορών των blocks του χωρίου, άρα αύξηση του λόγου εκτελούμενων πράξεων / όγκο μεταφορών
Σ.Ε.Ε.	Συγκεντρωτική Επεξεργασία Επαναλήψεων	Πλήρης αξιοποίηση των ετερογενών σωληνώσεων για παραγωγή υπολογιστικού αποτελέσματος σε κάθε κύκλο ρολογιού

Προτάσεις για μελλοντική εργασία

Η παρούσα ενασχόληση με τη Cell Broadband Engine καταλήγει σε ενδιαφέροντα συμπεράσματα, παράλληλα όμως δημιουργεί κάποιες σκέψεις και εγείρει διάφορα ερωτήματα για περεταίρω διερεύνηση:

- Μετατροπή του υπολογιστικού μέρους του κώδικα των εφαρμογών για χρήση αριθμών κινητής υποδιαστολής διπλής ακρίβειας και διεξαγωγή μετρήσεων σε επεξεργαστές PowerXCell 8i

- Υλοποίηση της προσέγγισης της συγκεντρωτικής επεξεργασίας επαναλήψεων στον αλγόριθμο in-place για τον υπολογισμό της εξίσωσης διάχυσης και

διερεύνηση του κατά πόσο έχει νόημα μία τέτοια υλοποίηση σε επίπεδο συστοιχίας

- Υλοποίηση στη Cell Broadband Engine της τριδιάστατης έκδοσης του προβλήματος διάχυσης

- Διεξαγωγή μετρήσεων σε συστοιχία με ανώτερο δικτυακό hardware, στο οποίο θα έχουν νόημα έννοιες όπως η επικαλυπτόμενη δρομολόγηση

- Διερεύνηση του προβλήματος εξίσωσης διάχυσης σε επίπεδο συστοιχίας, με διαμερίσεις που ελαχιστοποιούν τα διανύσματα εξάρτησης

- Εύρεση τρόπων επικοινωνίας που περιορίζουν τον όγκο των μεταφερόμενων δεδομένων μεταξύ των κόμβων της συστοιχίας

- Σύγκριση της επίδοσης των εφαρμογών σε σχέση με τις ίδιες εφαρμογές σε συστοιχίες διαφορετικών αρχιτεκτονικών

- Υλοποίηση περισσότερων εφαρμογών επιστημονικού ενδιαφέροντος

IBM, *Cell Broadband Engine Programming Tutorial* (2007),
http://www-128.ibm.com/developerworks/power/cell/

IBM, *Cell Broadband Engine Programming Handbook v1.1* (2007),
http://www-128.ibm.com/developerworks/power/cell/

IBM, *C/C++ Language Extensions for Cell Broadband Engine Architecture v2.5* (2008),
http://www-128.ibm.com/developerworks/power/cell/

IBM, *SPU Assembly Language Specification v1.6* (2007),
http://www-128.ibm.com/developerworks/power/cell/

IBM, *SPU Runtime Management Library v2.2* (2007),
http://www-128.ibm.com/developerworks/power/cell/

Daniel Hackenberg, *Fast Matrix Multiplication on Cell (SMP) Systems* (2007),
http://www.tu-dresden.de/zih/cell/matmul/

Γεωργία Κουβέλη,
Μελέτη αρχιτεκτονικής και εφαρμογών πολυπύρηνου υβριδικού επεξεργαστή (2007),
http://artemis.cslab.ntua.gr/Dienst/UI/1.0/Display/artemis.ntua.ece/DT2007-0221

Νικόλαος Ιωάννου,
Επικάλυψη Υπολογισμών και Επικοινωνίας σε Συστοιχίες από Πολυνηματικούς Επεξεργαστές (2008),
http://artemis.cslab.ntua.gr/Dienst/UI/1.0/Display/artemis.ntua.ece/DT2008-0105

IBM, *Cell Architecture, Course Code L1T1H1-10* (2006),
http://www.power.org/resources/devcorner/cellcorner/

Wikipedia, *Matrix Multiplication* (2009),
http://en.wikipedia.org/wiki/Matrix_multiplication/

Wikipedia, *Cell (microprocessor)* (2009),
http://en.wikipedia.org/wiki/Cell_(microprocessor)/

Βικιπαίδεια, *Διάχυση* (2009),
http://el.wikipedia.org/wiki/Διάχυση/

Kaushik Datta, Mark Murphy, Vasily Volkov, Samule Williams, Jonathan Carter, Leonid Oliker, David Patterson, John Shalf, Katherine Yelick, *Stencil Computation Optimization and Auto-tuning on State-of-the-Art Multicore Architectures,* Conference of High Performance Networking and Computing, Proceedings of the 2008 ACM/IEEE conference of Supercomputing, Austin, Texas, November 15 – 21, 2008

Shoaib Kamil, Kaushik Datta, Samule Williams, Leonid Oliker, John Shalf, Katherine Yelick, *Implicit and Explicit Optimizations for Stencil Computations,* Memory System Performance and Correctness ACM, Proceedings of the 2006 workshop on Memory system performance and correctness, San Jose, California, October 22, 2006

Samule Williams, John Shalf, Leonid Oliker, Shoaib Kamil, Parry Husbands, Katherine Yelick, *The Potential of the Cell Processor for Scientific Computing,* Conference On Computing Frontiers, Proceedings of the third conference on Computing Frontiers, Ischia, Italy, May 3 – 5, 2006

S. Hunold, T. Rauber, G. Rünger, *Combining building blocks for parallel multi-level matrix multiplication,* Parallel Computing 34: 411 – 426 (2008)