Ατομική Διπλωματική Εργασία

ΜΕΛΕΤΗ ΣΤΟΥΣ ΠΟΛΥΠΥΡΗΝΟΥΣ ΕΠΕΞΕΡΓΑΣΤΕΣ ΚΑΙ ΕΥΡΕΣΗ ΚΑΛΥΤΕΡΗΣ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΣΕ ΘΕΜΑΤΑ ΕΠΙΔΟΣΗΣ ΚΑΙ ΑΠΟΔΟΣΗΣ

Νικόλαος Εξαρχάκος

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2010

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μελέτη στους πολυπύρηνους επεξεργαστές και εύρεση καλύτερης αρχιτεκτονικής σε θέματα επίδοσης και απόδοσης

Νικόλαος Εξαρχάκος

Επιβλέπων Καθηγητής

Pedro Trancoso

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2010

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου, κ. Pedro Trancoso, που με εμπιστεύθηκε για την εκπόνηση αυτής της διπλωματικής εργασίας. Τον ευχαριστώ για την συνεργασία, την υπομονή και την καθοδήγηση που μου πρόσφερε κατά την διάρκεια αυτής της εργασίας. Επίσης, θα ήθελα να ευχαριστήσω πολύ το διδακτορικό φοιτητή Παναγιώτη Πετρίδη, όπου η βοήθειά του ήταν πραγματικά πολύτιμη ώστε να αντιμετωπίσω πολλά από τα προβλήματα που συνάντησα, στη διάρκεια αυτής της διπλωματικής εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω τους φίλους μου, την οικογένειά μου και ιδιαίτερα τους γονείς μου Γιώργο και Σταυρούλα, για την αγάπη και κατανόηση που επέδειξαν όλα αυτά τα χρόνια των σπουδών μου.

Περίληψη

Οι πολυπύρηνοι επεξεργαστές, είναι ένα σχετικά νέο είδος επεξεργαστών που έχουν κατακλύσει την παγκόσμια αγορά. Η ανάγκη για μεγαλύτερη επίδοση, ανάγκασε τους επιστήμονες να στραφούν προς αυτή την κατεύθυνση και να αφήσουν πίσω τους, τους μονολιθικούς επεξεργαστές. Ψάχνοντας για την καλύτερη απόδοση, δημιούργησαν πολλούς τύπους πολυπύρηνων επεξεργαστών, με στόχο πάντα τα καλύτερα δυνατά αποτελέσματα.

Στην παρούσα διπλωματική εργασία, προσπάθησα να συγκρίνω κάποια είδη πολυπύρηνων επεξεργαστών με διαφορετικά configurations και να καταλήξω σε καταλήξω σε κάποια συμπεράσματα σχετικά με την αποδοτικότητά τους. Συγκεκριμένα, σύγκρινα out-of-order επεξεργαστές μεταβλητού issue και inorder επεξεργαστές. Αυτή η σύγκριση έγινε με τη βοήθεια του προσομοιωτή SESC και τη σουίτα εφαρμογών SPLASH2.

Μέσα από το χρόνο εκτέλεσης, το speedup και το performance vs die area για 6 διαφορετικά benchmarks, θα δούμε πόσο σημαντικό ρόλο παίζει το είδος μιας εφαρμογής και ποιο configuration μπορεί και ξεχωρίζει από τα υπόλοιπα.

Θα δούμε ότι ναι μεν, οι out-of-order επεξεργαστές μπορεί να πετυχαίνουν καλύτερο execution time ενός benchmark, όμως αυτό δε σημαίνει παράλληλα ότι είναι και πιο αποδοτικοί. Σε πολλές περιπτώσεις οι inorder επεξεργαστές μοιάζουν σαν μια πολύ καλή λύση, αφού είναι πολύ αποδοτικοί σε ένα συγκεκριμένο fixed chip area.

Περιεχόμενα

1. Κεφάλαιο 1 - Εισαγωγή 1
1.1 Πρόλογος 1
1.2 Μεθοδολογία 2
1.3 Σκοπός Διπλωματικής Εργασίας 3
1.4 Δομή Διπλωματικής Εργασίας 3

2. Κεφάλαιο 2 – Σχετική δουλειά 5
2.1 Δουλειά που έχει γίνει σχετική με τους επεξεργαστές 5

3. Κεφάλαιο 3- Αναδρομή στους πολυπύρηνους επεξεργαστές 9
3.1 Εισαγωγή στους πολυπύρηνους επεξεργαστές 9
3.2 XEON Intel 10
3.3 IBM POWER 5 12
3.4 Intel 80-core terascale processor 14
3.5 LARRABEE 16
3.6 PIRANHA 18
3.7 AMD OPTERON 20
3.8 IBM PowerXCell 8i 21
3.9 Γενική περιγραφή 21

4. ΚΕΦΑΛΑΙΟ 4 - Προσομοιωτής SESC 23
4.1 Τι είναι γενικά ο προσομοιωτής 23
4.2 Παλιότερη εμπειρία με προσομοιωτή 24
4.3 Εισαγωγή για τον προσομοιωτή SESC 25
4.4 Τι είναι το superscalar out-of-order pipeline 27
4.5 Πως το SESC διαχειρίζεται την προσομοίωση 29

5. Κεφάλαιο 5 – Πειραματικά Αποτελέσματα 30
5.1 Εισαγωγή 31

5.2 Επιλογή των benchmarks 32
5.3 Περιγραφή configurations των benchmarks 35
5.4 Execution time vs Simulation time 36
5.5 Ανάλυση αποτελεσμάτων για το FFT benchmark 37
5.5.1 FFT-execution time 37
5.5.2 FFT-speedup 40
5.5.3 FFT-Υπολογισμός χώρου του chip 43
5.6 Ανάλυση αποτελεσμάτων για το CHOLESKY benchmark 44
5.6.1 CHOLESKY-execution time 44
5.6.2 CHOLESKY-speedup 47
5.6.3 CHOLESKY-Υπολογισμός χώρου του chip 49
5.7 Ανάλυση αποτελεσμάτων για το LU benchmark 51
5.7.1 LU-execution time 51
5.7.2 LU-speedup 54
5.7.3 LU-Υπολογισμός χώρου του chip 56
5.8 Ανάλυση αποτελεσμάτων για το RADIX benchmark 58
5.8.1 RADIX-execution time 58
5.8.2 RADIX-speedup 61
5.8.3 RADIX-Υπολογισμός χώρου του chip 63
5.9 Ανάλυση αποτελεσμάτων για το OCEAN benchmark 64
5.9.1 OCEAN-execution time 65
5.9.2 OCEAN-speedup 68
5.9.3 OCEAN-Υπολογισμός χώρου του chip 71
5.10 Ανάλυση αποτελεσμάτων για το BARNES benchmark 72
5.10.1 BARNES-execution time 73
5.10.2 BARNES-speedup 75
5.10.3 BARNES-Υπολογισμός χώρου του chip 77
5.11 Περίληψη και σύνοψη των πειραματικών αποτελεσμάτων 79

6. Κεφάλαιο 6 - Συμπεράσματα και Μελλοντική εργασία 83
6.1 Γενικά συμπεράσματα 83
6.2 Μελλοντική εργασία 86

1. **Κεφάλαιο 1**

Εισαγωγή

1.1 Πρόλογος 1
1.2 Μεθοδολογία 2
1.3 Σκοπός Διπλωματικής Εργασίας 3
1.4 Δομή Διπλωματικής Εργασίας 3

1.1 Πρόλογος

Η παρούσα ατομική διπλωματική εργασία, έχει σαν αντικείμενο τη μελέτη των πολυπύρηνων επεξεργαστών και την εύρεση κατάλληλων configurations, ώστε να πετυχαίνουμε καλύτερη επίδοση και απόδοση. Αυτά τα configurations, θα βασιστούν στην αλλαγή του issue των επεξεργαστών, τον αριθμό πυρήνων και τους inorder και out-of-order επεξεργαστές.

Οι πολυπύρηνοι επεξεργαστές προήλθαν από τις ανάγκη για δυνατότερους υπολογιστές χωρίς το μειονέκτημα τις υψηλής κατανάλωσης και τις υψηλής θερμοκρασίας.

Λίγο μετά το 2000 τα ερευνητικά εργαστήρια ανακάλυψαν ότι η αύξηση της συχνότητας των επεξεργαστών , που αποτελούσε μέχρι τότε το κύριο μέσο τις αύξησης τις επίδοσης, είχε περιορισμούς και δεν θα μπορούσε να συνεχίσει επ' άπειρο. Ο λόγος για αυτό το μεγάλο εμπόδιο ήταν η αύξηση της θερμοκρασίας όσο αυξανόταν η συχνότητα και η μεγάλη ανάγκη για κατανάλωση για την ψύξη αυτών των επεξεργαστών. Η λύση σε αυτό το πρόβλημα ήταν η δημιουργία επεξεργαστών που αντί για ένα δυνατό επεξεργαστή ενσωμάτωναν δύο ή και παραπάνω μικρότερης ισχύς. Αυτοί οι επεξεργαστές μπορούσαν να δουλεύουν παράλληλα βελτιώνοντας την συνολική απόδοση του συστήματος.

1.2 Μεθοδολογία

Στην ατομική διπλωματική μου εργασία, αρχικά γίνεται μια αναδρομή στους πολυπύρηνους επεξεργαστές, περιγράφοντας κάποια κύρια χαρακτηριστικά τους και στη συνέχεια παρουσιάζω το πειραματικό στάδιο της διπλωματικής μου. Για τις διάφορες μετρήσεις και την προσομοίωση των benchmarks,χρησιμοποίησα τον προσομοιωτή SESC και τη σουίτα εφαρμογών SPLASH2.

Τα configurations που δοκίμασα, είχαν να κάνουν με τον αριθμό των επεξεργαστών, το issue και τους inorder και out-of-order επεξεργαστές. Μέσα από συγκρίσεις των αποτελεσμάτων, προσπαθώ να καταλήξω στο ποια είναι τα ιδανικά configurations για πολυπύρηνους επεξεργαστές και με ποιο κόστος μπορεί κάτι τέτοιο να επιτευχθεί. Αλλάζοντας τα configurations, μια καλύτερη επίδοση σε χρόνο, δε σήμαινε παράλληλα πως κάτι τέτοιο είναι και ταυτόχρονα αποδοτικό και γι' αυτό το λόγο, έπρεπε να μελετήσω σε βάθος τα αποτελέσματα των benchmarks που προσομοίωσα ώστε να καταλήξω στα συμπεράσματά μου.

1.3 Σκοπός Διπλωματικής Εργασίας

Η διπλωματική μου εργασία, έχει σαν στόχο την μελέτη συστημάτων με πολυπύρηνους επεξεργαστές. Με βάση την έρευνα που θα κάνω, θα πειραματιστώ προσπαθώντας να πετύχω την καλύτερη επίδοση, αλλάζοντας τα configurations και τον αριθμό των επεξεργαστών.

Στόχος λοιπόν της διπλωματικής μου εργασίας, είναι να μελετήσω τα όρια των πολυπύρηνων επεξεργαστών και να δω μέσα από πειράματα και δοκιμές ποια αρχιτεκτονική είναι καλύτερη σε θέμα απόδοσης. Για το λόγο αυτό, θα συγκριθούν πολυπύρηνοι επεξεργαστές οι οποίοι θα έχουν διαφορετικό issue, διαφορετικό αριθμό επεξεργαστών και θα συγκρίνω τους inorder και out-of-order επεξεργαστές. Για την καλύτερη και ακριβέστερη συλλογή αποτελεσμάτων, χρησιμοποιήθηκαν 6 benchmarks (4 kernels,2 applications), αφού τα αποτελέσματα της παραπάνω μελέτης, έχει άμεση σχέση με την εφαρμογή που θα χρησιμοποιηθεί.

1.4 Δομή Διπλωματικής Εργασίας

Η παρούσα ατομική διπλωματική εργασία, με θέμα Μελέτη στους Πολυπύρηνους Επεξεργαστές και Εύρεση Καλύτερης Αρχιτεκτονικής σε Θέματα Επίδοσης και Απόδοσης αποτελείται από 6 κεφάλαια.

Το *πρώτο* κεφάλαιο είναι η εισαγωγή της διπλωματικής εργασίας και περιγράφω το στόχο και τη σχετική δουλειά που έχει γίνει πάνω στους πολυπύρηνους επεξεργαστές.
Στο *δεύτερο* κεφάλαιο, παρουσιάζεται η σχετική δουλειά που έχει γίνει πάνω στους επεξεργαστές.
Στο *τρίτο* κεφάλαιο, γίνεται μια αναδρομή στους πολυπύρηνους επεξεργαστές, περιγράφοντας κάποια γνωστά μοντέλα από multicores, που είχαν διατεθεί στο εμπόριο.
Στο *τέταρτο* κεφάλαιο αναλύεται, ο τρόπος λειτουργίας ενός προσομοιωτή και εξηγείται εκτενέστερα ο προσομοιωτής SESC πάνω στον οποίο στηρίχθηκε η παρούσα διπλωματική εργασία.

Στο *πέμπτο* κεφάλαιο δίνονται τα πειραματικά αποτελέσματα, τα οποία εξήγαγα από την προσομοίωση των benchmarks, τα οποία έχουν να κάνουν με το execution time , το speedup, το IPC και το χώρο του chip.

Στο *έκτο* κεφάλαιο περιγράφονται τα συμπεράσματα, τα οποία έβγαλα σύμφωνα με τη συλλογή και τη σύγκριση των αποτελεσμάτων και επίσης παρουσιάζεται η μελλοντική εργασία που μπορεί να γίνει.

2. **Κεφάλαιο 2**

Σχετική δουλειά

2.1 Δουλειά που έχει γίνει σχετική με τους επεξεργαστές.....5

2.1 Δουλειά που έχει γίνει σχετική με τους επεξεργαστές

Πολλοί ερευνητές και καθηγητές, εστίασαν τη δουλειά τους, στο πως θα καταφέρουν να αυξήσουν την απόδοση των επεξεργαστών. Για το λόγο αυτό, χρησιμοποιήθηκαν πολλές διαφορετικές τεχνικές, εστιάζοντας στα χαρακτηριστικά του hardware και στις ιδιότητες των εφαρμογών.

Το pipeline execution model, ήταν ικανό να μειώσει τις καθυστερήσεις στα διάφορα στάδια εκτέλεσης μιας εντολής και στόχευσε στην απόδοση του throughput που εκμεταλεύεται το διαθέσιμο παραλληλισμό στο instruction – level (ILP). Διαφορετικές παραλλαγές αυτής της τεχνικής, έχουν προταθεί κατά τη διάρκεια των τελευταίων ετών, όπως ο επεξεργαστής VLIW.

Όσον αφορά το pipeline, η τεχνική αυτή επιτρέπει στις εντολές να γίνονται issue και να εκτελούνται ανεξάρτητα, με στόχο την αποφυγή των διαφόρων stalls που θα

συναντούσαμε αν τις εκτελούσαμε όλες μαζί σε ένα μόνο στάδιο. Γι' αυτό το λόγο, προτάθηκαν διάφορες λύσεις και τεχνικές. Μια τεχνική που προτάθηκε ήταν το Tomasulo, όπου πρότεινε έναν επεξεργαστή με διάφορα functional units, επιτρέποντας στις εντολές να γίνονται issue in-order, αλλά να εκτελούνται out-of-order, χωρίς να παραβιάζονται οι εξαρτήσεις των δεδομένων, έτσι ώστε να εξασφαλίζεται η σωστή εκτέλεση της εφαρμογής. Αυτή η τεχνική, απαλείφει τις καθυστερήσεις από τις μεγάλες εντολές, αφού εκτελεί άλλες εντολές στους διαθέσιμους πόρους την ίδια χρονική στιγμή και έτσι αυξάνεται το throughput του επεξεργαστή.

Οι out-of-order επεξεργαστές κατάφεραν να εκμεταλλευτούν το διαθέσιμο παραλληλισμό των εντολών, γνωστό και ως Instruction-Level Parallelism (ILP). Πολλά ερευνητικά προγράμματα, εστίασαν στο ILP και στο πώς να επιτύχουν υψηλά επίπεδα απόδοσης. Φτάνοντας όμως στα όρια του ILP, οι ερευνητές στράφηκαν στη μελέτη εναλλακτικών μεθόδων, για επίτευξη υψηλού παραλληλισμού.

Ο πρόσθετος παραλληλισμός, βρέθηκε στα threads, γνωστό και ως thread level Parallelism (TLP). Ο στόχος αυτών των επεξεργαστών, ήταν να εκμεταλλευτούν όχι μόνο το διαθέσιμο ILP άλλα επίσης και το διαθέσιμο παραλληλισμό μεταξύ των threads. Πολλά projects, εστίασαν στο πως θα βελτιώσουν την απόδοση του throughput στους TLP επεξεργαστές. Γι' αυτό το λόγο προτάθηκαν και οι Simultaneous Multithreading processors (SMT). Οι SMT επεξεργαστές επιτρέπουν σε αρκετά από τα ανεξάρτητα threads, να κάνουν issue εντολές σε πολλαπλά functional units του superscalar, σε ένα κύκλο εντολών. Αυτή η τεχνική, αυξάνει τη χρησιμοποίηση του επεξεργαστή, παρά τις μεγάλες καθυστερήσεις της μνήμης και τον περιορισμένο διαθέσιμο παραλληλισμό κάθε thread. Η αύξηση της απόδοσης τέτοιων επεξεργαστών, επιτεύχθηκε με την αύξηση της συχνότητας του επεξεργαστή, μια τεχνική που χρησιμοποιείται επίσης και για τα ILP συστήματα.

Οι παραδοσιακοί μονολιθικοί επεξεργαστές είχαν πια φτάσει στα τελευταία στάδια εξέλιξης και γι' αυτό οι ερευνητές, αποφάσισαν να στρέψουν την προσοχή τους σε άλλες τεχνολογίες. Με στόχο λοιπόν να βελτιωθεί η απόδοση, όπως μάλιστα

υπαγορεύεται από το νόμο του Moore, οι ερευνητές ασχολήθηκαν με τους Chip Multiprocessors (CMP), μια αποδοτική λύση, όχι μόνο στην πτυχή της απόδοσης αλλά και στην πτυχή της δύναμης και της θερμικής αποδοτικότητας.

Τα μελλοντικά multi-core συστήματα αναμένεται να αποτελούνται από 100 πυρήνες στο ίδιο die. Ένα σοβαρό θέμα σε αυτά τα συστήματα, είναι ο τρόπος που οι εφαρμογές θα είναι σε θέση να εκμεταλλευτούν τους διαθέσιμους πόρους, ώστε να έχουν μια αποδοτική εκτέλεση. Μέσα από τη συγκεκριμένη διπλωματική εργασία, θα εστιάσω πάνω σε αυτό το θέμα και θα προσπαθήσω μέσω των προσομοιώσεων να δείξω αν με την αύξηση των επεξεργαστών και κάποιων άλλων παραμέτρων έχουμε βελτίωση της επίδοσης και της απόδοσης. Σύμφωνα με το νόμο του Amdahl, μια εφαρμογή που έχει παραλληλισμό 99%, περιορίζεται στο μέγιστο speedup του 100, ανεξαρτήτως από τους διαθέσιμους πόρους που δίνονται στην εφαρμογή. Το σειριακό κομμάτι, είναι αυτό που περιορίζει τη μέγιστη απόδοση της εφαρμογής και γι' αυτό οι ερευνητές οδηγήθηκαν σε κάποιες εναλλακτικές μεθόδους όσων αφορά την τεχνολογία CMP.

Μία πολύ καλή λύση, στην οποία οδηγήθηκαν οι ερευνητές, είναι οι ετερογενείς chip multi-cores, όπου επεξεργαστές με διαφορετικές επιδόσεις και σχεδιασμό θα προσαρμοστούν στο ίδιο die. Οι ετερογενείς multi-cores, μπορούν να παρέχουν, την καλύτερη απόδοση/watt για ένα thread, κατά τη διάρκεια όλων των φάσεων του, επειδή είναι σε θέση να ταιριάξει με τις απαιτήσεις των εφαρμογών, όσων αφορά τους διαθέσιμους πόρους. Τα ετερογενή multicore συστήματα, δε σημαίνει ότι θα πρέπει να αποτελούνται μόνο, από διαφορετικούς ISA πυρήνες. Θα μπορούσαν να αποτελούνται, από ίδιους ISA πυρήνες, αλλά με διαφορετικές ιδιότητες επεξεργασίας, όπως η συχνότητα του επεξεργαστή, γνωστό και ως asymmetric multicore. Πρέπει να αναφερθεί, ότι ένα ομογενές multi-core σύστημα, μπορεί να εξελιχθεί σε ένα asymmetric multi-core σύστημα λόγω ενός σφάλματος του hardware, όπως η υποβάθμιση συχνότητας των πυρήνων.

Άλλα ερευνητικά προγράμματα, έχουν προτείνει μια περισσότερο αποδοτική λύση από τα ετερογενή multi-core συστήματα. Τα δυναμικά ετερογενή multicore

συστήματα, έχουν τη δυνατότητα να αλλάζουν δυναμικά τις όποιες ρυθμίσεις τους, προσαρμόζοντας τες στις απαιτήσεις των εφαρμογών. Πιο συγκεκριμένα, όταν αρχίζει μια σειριακή εκτέλεση, το διαθέσιμο υλικό καθιστά υπεύθυνο γι' αυτή την εκτέλεση έναν πολύ δυνατό επεξεργαστή, ενώ όταν αρχίζει μια παράλληλη εκτέλεση, η εκτέλεση καθίσταται σε πολλούς πυρήνες που είναι κατάλληλοι για την παράλληλη εκτέλεση. [1]

3. **Κεφάλαιο 3**

Αναδρομή στους πολυπύρηνους επεξεργαστές

3.1 Εισαγωγή στους πολυπύρηνους επεξεργαστές 9
3.2 XEON Intel 10
3.3 IBM POWER 5 12
3.4 Intel 80-core terascale processor 14
3.5 LARRABEE 16
3.6 PIRANHA 18
3.7 AMD OPTERON 20
3.8 IBM PowerXCell 8i 21
3.9 Γενική περιγραφή 21

3.1 Εισαγωγή στους πολυπύρηνους επεξεργαστές

Οι πολυπύρηνοι επεξεργαστές προήλθαν από τις ανάγκη για δυνατότερους υπολογιστές χωρίς το μειονέκτημα τις υψηλής κατανάλωσης και τις υψηλής θερμοκρασίας.

Λίγο μετά το 2000 τα ερευνητικά εργαστήρια ανακάλυψαν ότι η αύξηση της συχνότητας των επεξεργαστών , που αποτελούσε μέχρι τότε το κύριο μέσο τις αύξησης τις επίδοσης, είχε περιορισμούς και δεν θα μπορούσε να συνεχίσει επ' άπειρο. Ο λόγος για αυτό το μεγάλο εμπόδιο ήταν η αύξηση της θερμοκρασίας όσο αυξανόταν η συχνότητα και η μεγάλη ανάγκη για κατανάλωση για την ψύξη αυτών των επεξεργαστών. Η λύση σε αυτό το πρόβλημα ήταν η δημιουργία επεξεργαστών που αντί για ένα δυνατό επεξεργαστή ενσωμάτωναν δύο ή και παραπάνω μικρότερης ισχύς. Αυτοί οι επεξεργαστές μπορούσαν να δουλεύουν παράλληλα βελτιώνοντας την συνολική απόδοση του συστήματος.

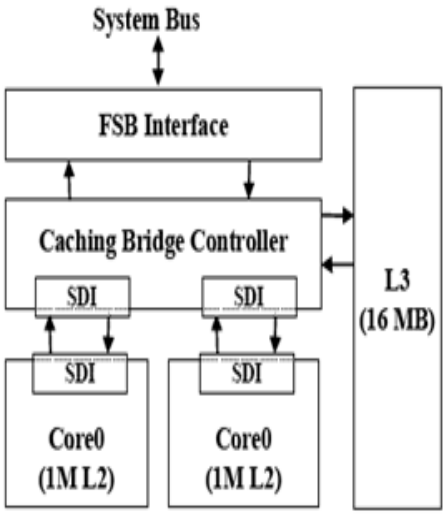
Στη συνέχεια, θα παρουσιαστούν μερικοί επεξεργαστές που έχουν ιδιαίτερο ενδιαφέρον ώστε να τους δούμε με κάποια παραπάνω λεπτομέρεια. Σε αυτούς περιλαμβάνονται σχέδια που υπάρχουν διαθέσιμα στην αγορά όπως ο XEON της Intel και ο Power5 της IBM. Επίσης, θα παρουσιαστούν κάποια ερευνητικά προγράμματα όπως το Piranha ,ο Terascale processor της Intel και το larrabee, ένας επεξεργαστής ο οποίος προορίζεται από την Intel να υπάρχει τόσο σαν κάρτα γραφικών όσο και σαν επεξεργαστής για εφαρμογές γενικής χρήσης.

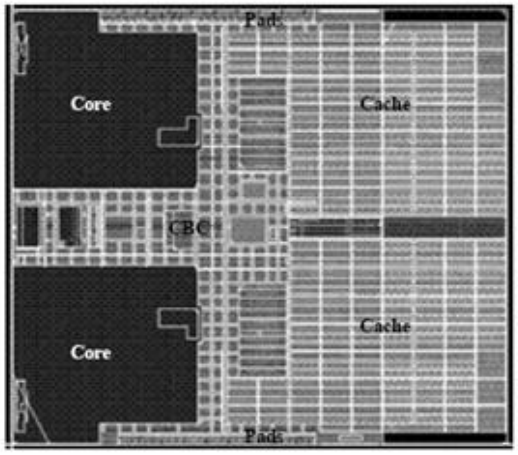
3.2 XEON Intel

Οι επεξεργαστές XEON [2] είναι η σειρά της Intel που ενσωματώνει σε servers.Οι πρώτοι XEON βγήκαν σε κυκλοφορία το 1998 σαν αντικαταστάτες του Pentium II pro και είχαν ένα μόνο πυρήνα. Σήμερα βγαίνουν με δύο και τέσσερις πυρήνες και λέγεται ότι ο διάδοχος τους θα περιέχει οχτώ πυρήνες. Στα πλαίσια τις μελέτης μας, είδαμε τον Multi threaded dual core 65nm Xeon, ένα διπύρηνο επεξεργαστή που βγήκε στην αγορά το 2006 και υποστηρίζει την τεχνολογία multithreading, με μικρές ανάγκες σε χώρο. Ο επεξεργαστής αυτός έχει δύο πυρήνες των 64-bit ο καθένας και μια ενοποιημένη κρυφή μνήμη τρίτου επιπέδου χωρητικότητας 16 MB (unified L3 cache). Ο κάθε πυρήνας υποστηρίζει δύο threads και 1 MB κρυφή μνήμη δευτέρου επιπέδου. Το

ιδιαίτερο χαρακτηριστικό αυτού του επεξεργαστή είναι ο τρόπος που υλοποιείται. Χρησιμοποιεί ένα caching bridge controller (CBC) για να στείλει τα δεδομένα από και προς την cache, τους πυρήνες και το front side bus (FSB). Για να καταφέρει να ενώσει τους δύο πυρήνες με το CBC χρησιμοποιεί ,υψηλής απόδοσης simple direct interface (SID). (*σχήμα 3.1*)

Στο *σχήμα 3.2* βλέπουμε μια γενική απεικόνιση του die.





Σχήμα 3.1: *Processor Block Diagram* [2] ***Σχήμα 3.2:*** *Γενική απεικόνιση die* [2]

Με την χρήση της αρχιτεκτονικής που περιγράψαμε και που φαίνεται από τις εικόνες, έγινε εφικτό να μειωθούν τα latencies για την προσέγγιση δεδομένων στην cache και στο external bus.

Για την δημιουργία του λογικού μέρους του CBC χρησιμοποιήθηκε η τεχνική βασισμένη στα cells.Η τεχνική αυτή είναι ιδιαίτερα αποδοτική σε σχέδια όπου ο χρόνος είναι το κύριο ζητούμενο ,για να επιτευχθεί καλύτερη επίδοση σε χρόνο μεγαλώνει το μέγεθος και δημιουργούνται μεγαλύτερες ανάγκες σε ισχύ. Η ανάγκη για ισχύ τελικά μειώθηκε με την χρήση ενός άλλου είδους transistors , τα Low Leakage transistors (LL-cells).

Ο επεξεργαστής αυτός κατάφερε τους στόχους του ,να είναι ισχυρός ,μικρός και χωρίς υψηλές ανάγκες σε ισχύ. Αναλυτικά σε εμβαδόν die 435 mm2 περιέχει 1.33B transistors, κάθε πυρήνας με 1,25V μπορεί να φτάσει στα 3.0GHz με χείριστη ανάγκη για ισχύ τα 150 W ,υπό φυσιολογικές για server συνθήκες χρειάζεται περίπου 100 W.

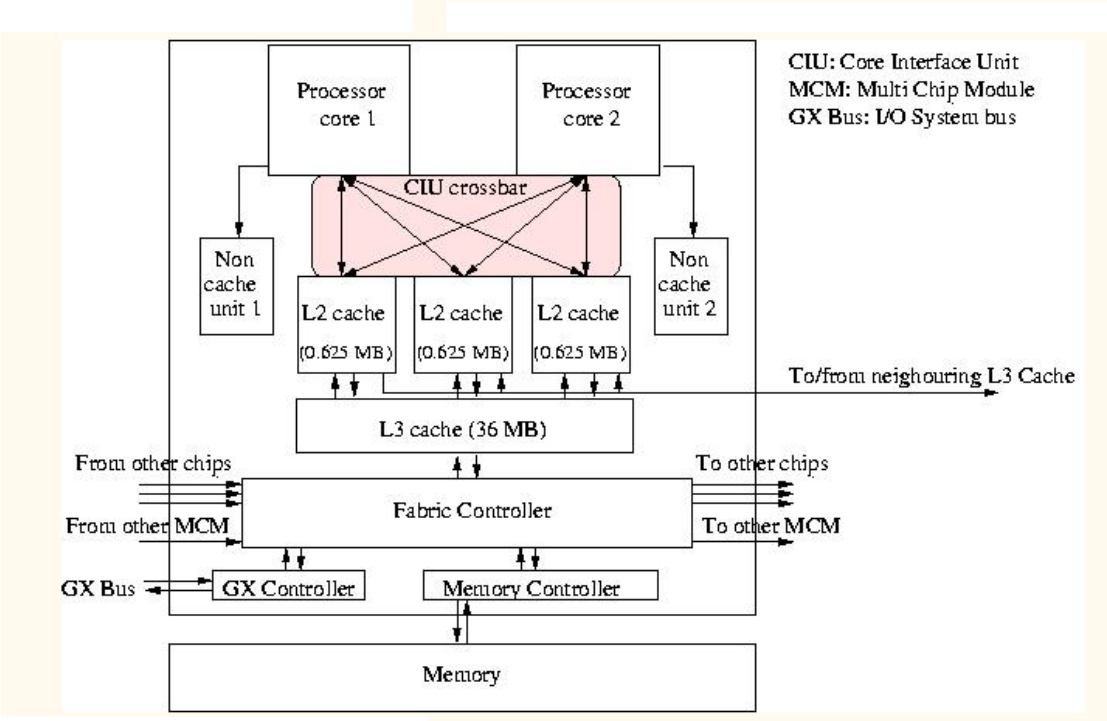
3.3 IBM POWER 5

Ο IBM POWER5 [3] είναι ένας dual-core επεξεργαστής, όπου ο κάθε core τρέχει 2 threads. Τα threads διαμοιράζονται πολλούς πόρους, όπου το Global Competition Table (GCT or reorder buffer), το Branch History Table(BHT) και το Translation Lookaside Buffer (TLB). Ο IBM POWER5 εξισοροπεί κατάλληλα τη χρήση των πόρων διαμέσου των threads με διάφορους μηχανισμούς στο hardware, κάτι που σημαίνει μεγαλύτερη απόδοση.

Το POWER5+ chip είναι μια ανακατασκευασμένη έκδοση του POWER5 χρησιμοποιώντας 90-nanometer (nm) processor technology αντί των 130 nm και δεν έχει ουσιαστικά κάποια νέα features εκτός του ότι το clock frequence μπορεί να γίνει ελαφρώς υψηλότερο κατά τη διάρκεια της ζωής του. Αυτή η τεχνολογία σμίκρυνσης, κατέστησε δυνατό στην IBM να τοποθετήσει δύο processor cores σε ένα chip. Το chip frequency του POWER5+ είναι στο διάστημα 1.5-1.9 GHz.

Ο POWER5+ processor core περιλαμβάνει private L1 instruction και data caches. Κάθε dual chip module αποτελείται (DMC) από ένα POWER5+ chip (dual-core με on-chip L2) και ένα L3 cache chip. Και η L2 και η L3 cache είναι κοινή και για τους 2 πυρήνες. Το L1 instruction cache έχει χωρητικότητα 64 KB και είναι 2-way set associative, ενώ το L1 data cache έχει 32 KB χωρητικότητα και είναι 4-way set associative. Το POWER5+ chip έχει 1,92 MB L2 cache που διαιρείται εξίσου σε 3 modules και 10-way set associative με cache line 128 bytes. Η off-chip L3 cache έχει χωρητικότητα 36 MB και είναι 12-way set associative με cache line 256 bytes. Η L3 cache είναι επίσης χωρισμένη σε 3 κομμάτια, που κάθε ένα από αυτά χρησιμεύει σαν

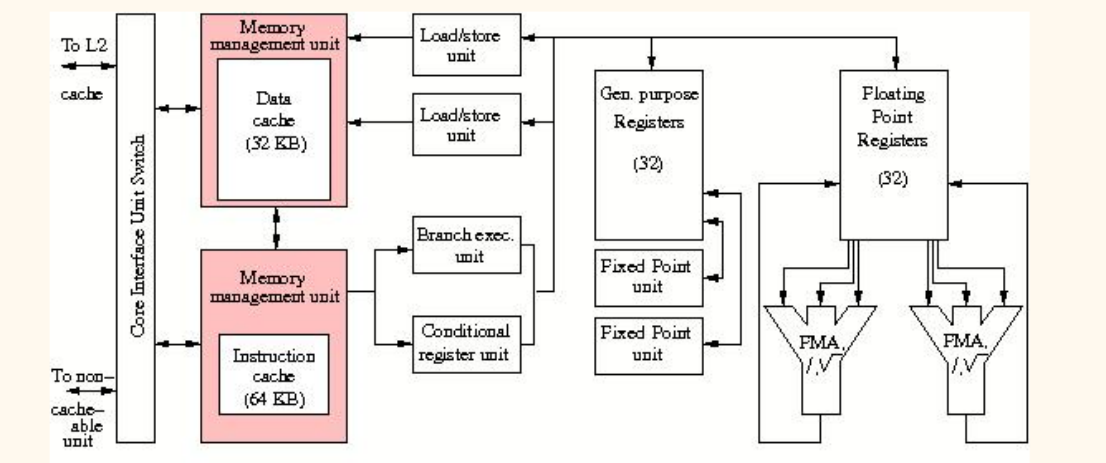
spill cache, δηλαδή σαν cache υπερχείλισης του αντίστοιχου L2 κομματιού. Τα L2 cache modules είναι συνδεδεμένα με τους cores μέσω του Core Interface Unit, ένα 2(cores) x 3(L2 modules) crossbar με μέγιστο bandwidth της τάξης 40 bytes/cycle, ανά port. Αυτό επιτρέπει τη μεταφορά 32 bytes είτε στο L1 instruction cache είτε στο L1 data cache του κάθε core, όπως επίσης και την αποθήκευση 8 bytes στη μνήμη την ίδια χρονική στιγμή. Η L3 cache έχει μια λανθάνουσα κατάσταση 80 κύκλων, σε αντιδιαστολή με την κύρια μνήμη που έχει μια λανθάνουσα κατάσταση 220 κύκλων.



Σχήμα 3.2: *Διάγραμμα του IBM POWER5+ chip layout* [3]

Υπάρχει ένα άλλο χαρακτηριστικό γνώρισμα του POWER5+ που δεν βοηθά για συχνά data access, αλλά που μπορεί να ωφελήσει για προγράμματα όπου το data access δεν είναι τόσο συχνό: το Simultaneus Multithreading (SMT). Τα POWER5 CPUs είναι σε θέση να κρατάνε δύο process threads εν ενεργεία, την ίδια χρονική στιγμή. Τα functional units παίρνουν instructions από οποιοδήποτε από τα δύο threads που είναι σε θέση να γεμίσουν ένα slot σε ένα instruction word, που θα γίνει issue στα functional units. Για τους πολύ συχνούς υπολογισμούς το single thread (ST) mode μπορεί να είναι καλύτερο και αυτό γιατί στο SMT τα 2 threads ανταγωνίζονται για τα entries στις caches

που αυτό μπορεί να οδηγήσει στην περίπτωση των συχνών data access. Εδώ πρέπει να σημειώσουμε ότι το SMT είναι κάπως διαφορετικό από το "normal" multithreading. Στην περίπτωση του "normal" multithreading, ένα thread που γίνεται stall, σταματά και αντικαθίσταται από κάποιο άλλο proccess thread που είναι ενεργό εκείνη τη στιγμή, κάτι που φυσικά κοστίζει σε χρόνο. Αυτό σημαίνει ότι το δεύτερο thread πρέπει να είναι ενεργό για ένα δίκαιο αριθμό κύκλων (κατά προτίμηση μερικές εκατοντάδες κύκλοι τουλάχιστον). Το SMT δεν έχει αυτό το μειονέκτημα, αλλά ο σχεδιασμός των instructions και των δύο threads είναι αρκετά περίπλοκος.



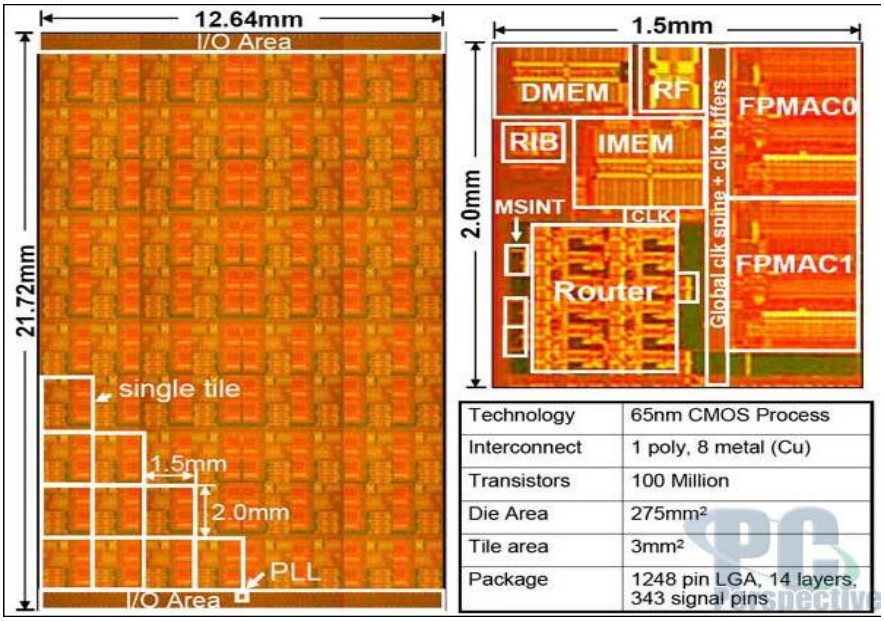
Σχήμα 3.3: *IBM POWER5+ processor core* [3]

3.4 Intel 80-core terascale processor

Ο Intel 80-core terascale [4] processor αυτός αποτελεί ένα ερευνητικό πρόγραμμα της Intel και είναι ο πρώτος generally programmable microprocessor που έσπασε το φράγμα του ενός Teraflop.Ο επεξεργαστής αυτός, όπως μαρτυράει και το όνομα του περιέχει 80 πυρήνες και κατασκευάστηκε με κύριο στόχο να μελετηθούν οι τρόποι για διαχείριση ενέργειας και να διερευνηθούν διαφορετικές προσεγγίσεις για την διασύνδεση των πυρήνων μέσα στο die. Δευτερεύων στόχος για την Intel, ήταν να αποδείξει ότι μπορεί να επιτευχθεί υψηλή επίδοση με τόσο μεγάλο αριθμό πυρήνων. Ο

επεξεργαστής αυτός, ήταν ο πρώτος που κατάφερε να ξεπεράσει τρέχοντας εφαρμογή kernel τις ένα τρισεκατομμύριο floating point πράξεις ανά δευτερόλεπτο.

Ο επεξεργαστής, αποτελείται από ένα «δυσδιάστατο πίνακα» 8 x10 όπου κάθε κελί είναι ένας πυρήνας (*σχήμα 3.4*).Συνολικά περιέχονται 100 εκατομμύρια transistors σε ένα die με εμβαδόν 275mm2. Κάθε ένας από τους 80 πυρήνες περιέχει ένα δρομολογητή με 5 Ports , δύο μονάδες floating point, 3KB instruction memory , 2KB data memory , και ένα 10 port 32 entry register file. Κάθε Floating Point Unit (FPU) μπορεί να έχει μέγιστη επίδοση 4 FLOP/cycle/core.



Σχήμα 3.4: *Η δομή του Intel 80 core terascale processor* [4]

Καθένας από τους 80 πυρήνες αποδίδει 4,27GHz με ζητούμενη ισχύ 1,07V. Συνολικά ο 80-core terascale processor μπορεί να αποδώσει μέγιστα 1,37 TFLOPS ανεβάζοντας θερμοκρασία 80 βαθμούς Κελσίου και χρειάζεται ισχύ 97 Watts.

Ο επεξεργαστής αυτός, καταφέρνει να επιτύχει καλύτερες επιδόσεις και από τον Tilera ,έναν επεξεργαστή με 64 πυρήνες, αλλά και από τον Intel Core 2 Duo Quad. Αναλυτικότερα, ο Tilera πετυχαίνει επίδοση 0.192 TFLOPS/s θέλοντας 35W ισχύ, ενώ ο 80-core terascale processor καταφέρνει να φτάσει τα 1,37 TFLOPS/s στα 97W πετυχαίνοντας επίδοση 2,5 φορές καλύτερη ανά μονάδα ισχύος (Watt). Αντίστοιχα ο

Intel Core 2 Duo Quad πετυχαίνει επίδοση 0,9 GFLOPS/Watt την στιγμή που ο 80-core terascale processor φτάνει τα 19,4 GFLOPS/Watt επίδοση πάνω από 20 φορές μεγαλύτερη από τον κοινό πολυπύρηνο επεξεργαστή που κυκλοφορεί στην αγορά.

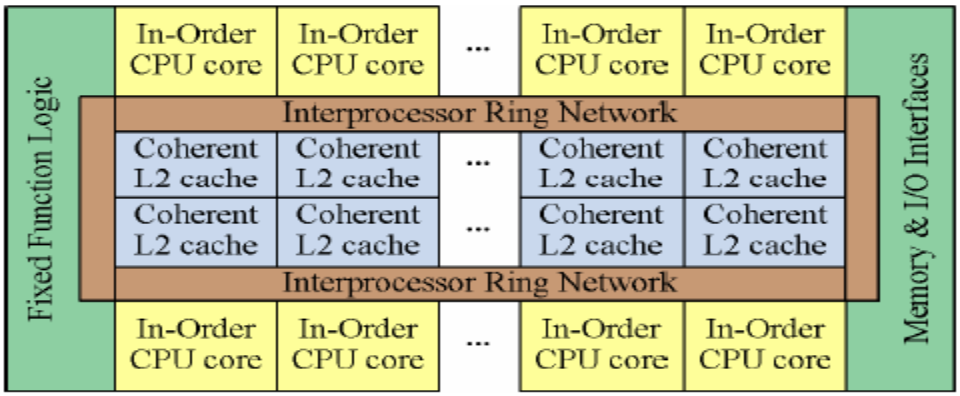
Παρόλο, που ο 80-core terascale processor είναι ιδιαίτερα ισχυρός έχει ένα μεγάλο μειονέκτημα. Σαν ερευνητικό πείραμα που είναι, έχουν γίνει οι κατάλληλες τροποποιήσεις έτσι ώστε να αυξηθεί η επίδοση. Στην περίπτωση αυτού του επεξεργαστή οι τροποποιήσεις είναι ιδιαίτερα σημαντικές αφού είναι κατάλληλος για μελέτη kernel. Βασικό χαρακτηριστικό που λείπει από τον επεξεργαστή αυτό, είναι το ελλιπές σύνολο εντολών assembly που μπορεί να εκτελέσει και δεν του επιτρέπει να τρέξει ολοκληρωμένες εφαρμογές.

Ο επεξεργαστής, χρησιμοποιεί επίσης λέξεις μήκους 96 bit, Very Long Instruction Word (VLIW). Με την υλοποίηση του VLIW καταφέρνει να εκτελεί μέχρι 4 πράξεις ανά κύκλο. Σημαντικός είναι και ο τρόπος, πάνω στον οποίο βασίστηκε το προγραμματιστικό μοντέλο του 80-core terascale processor που είναι το message passing για επικοινωνία και συγχρονισμό ανάμεσα στους πυρήνες. Σε αυτό το σημείο θα πρέπει να πούμε ότι το κάθε πρόγραμμα που γράφεται για τον 80-core terascale processor είναι γραμμένο σε assembly στο χέρι ειδικά για αυτόν ,με τον περιορισμό ενός μόνο loop και με συγκεκριμένη έκταση του offset στην μνήμη. Γενικότερα, θεωρείται ερευνητικό πρόγραμμα με μέλλον που περιμένουμε να δούμε επιδόσεις με ολοκληρωμένο instruction set.

3.5 LARRABEE

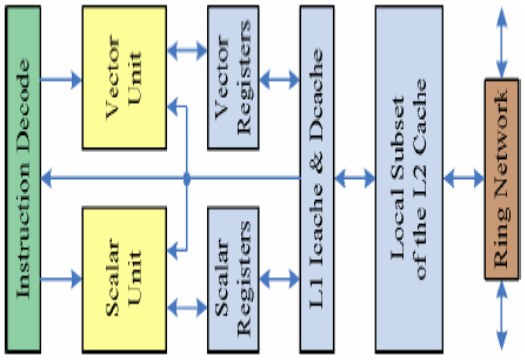
Ο Larrabee [5], αποτελεί ένα καινούριο σχέδιο της Intel που προσδοκά να καλύψει τόσο την αγορά των καρτών γραφικών, βραχυπρόθεσμα, όσο και την αγορά των επεξεργαστών, μακροπρόθεσμα . Είναι ένας επεξεργαστής που χρησιμοποιεί πολλούς in-order πυρήνες που επεξεργάζονται το x86 instruction set (*σχήμα 3.5*). Οι πυρήνες που

περιέχει ο Larrabee βασίζονται στους είδη υπάρχοντες in-order Pentium με την προσθήκη εντολών μήκους 64 bits, multithreading και ενός wide vector processor unit (VPU). Οι πυρήνες που ενσωματώνει ο επεξεργαστής έχουν 32KB instruction cache όπως επίσης και 32 KB data cache σε αντίθεση με τα 8KB Icache και 8KB Dcache που είχαν οι απλοί single-threaded Pentiums προκάτοχοι τους.



Σχήμα 3.5: *Η δομή του Larrabee* [5]

Ο κάθε πυρήνας επικοινωνεί με την L2 που του αντιστοιχεί (256 KB μέρος της συνολικής L2 cache), μέσω του δακτυλίου δύο κατευθύνσεων που περιβάλει την μνήμη. Επίσης ο δακτύλιος με την βοήθεια των fixed function logic αλλά και των memory & I/O interfaces δίνουν την δυνατότητα να επικοινωνούν μεταξύ τους τα cores. Κύριο ρόλο στην επίδοση παίζουν οι vectors που σε συνδυασμό με την in-order εκτέλεση των εντολών μας δίνουν υψηλή επίδοση ανά watt αλλά και σε σχέση με το μέγεθος που καταλαμβάνει ο πυρήνας. Παρακάτω βλέπουμε την δομή του κάθε πυρήνα (*σχήμα 3.*6) αλλά και τον πίνακα (*πίνακας 3.1*),όπου βλέπουμε τις διαφορές μεταξύ out-of order και in-order εκτέλεση εντολών.



# CPU cores:	2 out-of-order	10 in-order
Instruction issue:	4 per clock	2 per clock
VPU per core:	4-wide SSE	16-wide
L2 cache size:	4 MB	4 MB
Single-stream:	**4 per clock**	**2 per clock**
Vector throughput:	**8 per clock**	**160 per clock**

Σχήμα 3.6*:Δομή core στο Larrabee* [5] ***Πίνακας 3.1:*** *Πίνακας σύγκρισης in/out order* [5]

Όπως παρουσιάζει και ο πίνακας, όταν έχουμε in-order εκτέλεση το throughput των vectors αυξάνεται κατά πολύ, αν και μειώνονται τα issues, αποδίδοντας πολύ καλά για τον επεξεργαστή Larrabee. Πρέπει επίσης να σημειώσουμε ότι οι Pentium που χρησιμοποιούνται για το Larrabee διαφέρουν όχι μόνο στο cache αλλά και σε άλλα μέρη με τους πρόγονούς τους. Οι κυριότερες διαφορές που έχουν είναι ότι πλέον υποστηρίζουν 64 bit εντολές αλλά και multithreading. Ιδιαίτερα εξελιγμένο είναι και το prefetching που επιτρέπει στον επεξεργαστή να έχει περιορισμένα misses. Παρόλο τις αλλαγές που έχουν γίνει , οι πυρήνες αυτοί υποστηρίζουν ακόμα το x86 instruction set διατηρώντας backward compatibility.

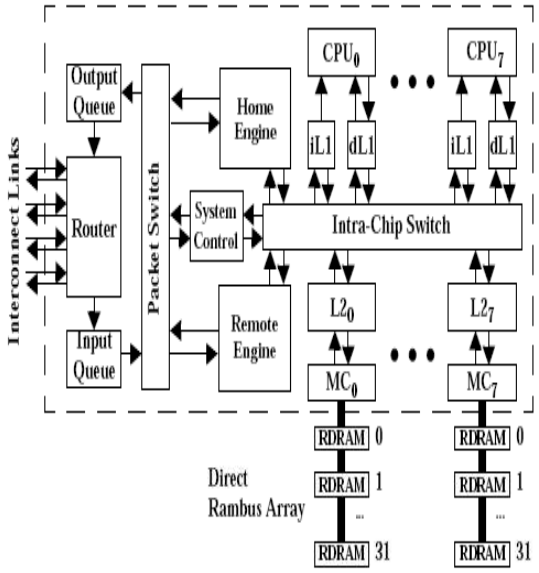
Ο Larrabee χρησιμοποιεί dual–issue decoder με υψηλό multi issue rate όπως είδαμε και από την ερευνά μας. Το κύριο βάρος της επεξεργασίας των εντολών γίνεται μέσο ενός primary pipeline ενώ το δεύτερο είναι βοηθητικό και εκτελεί απλές πράξεις. Επιπροσθέτως, ο επεξεργαστής που εξετάζουμε διαθέτει τέσσερα threads εκτέλεσης. Η εναλλαγή τους μας βοηθάει να γλιτώσουμε καθυστερήσεις που μπορεί να οφείλονται είτε στην αδυναμία του μεταγλωττιστή να κάνει σωστή χρονοδρομολόγηση ώστε να μην έχουμε stall, είτε στην αδυναμία να γίνουν τα σωστά δεδομένα prefetched στην ώρα τους. Τέλος, το κύριο πλεονέκτημα του larrabee είναι ότι μπορεί να αποδώσει ιδιαίτερα καλά τόσο σε εφαρμογές γραφικών, όσο και σε εφαρμογές γενικής χρήσης.

3.6 PIRANHA

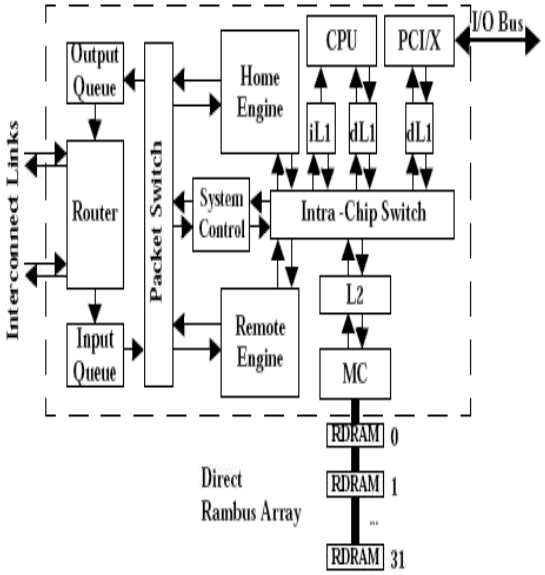
Το Piranha [6] είναι ένα ερευνητικό πρόγραμμα που ανέπτυξε η Compaq το 2000 με σκοπό να δημιουργήσουν σε λίγο χρόνο και με μικρό κόστος ένα σύστημα που μπορεί να επεξεργάζεται on-line transactions αποδοτικά.

Το κύριο κομμάτι του Piranha είναι ένας processing node με 8 απλούς πυρήνες Alpha με ξεχωριστή L1 cache για instructions και data για κάθε πυρήνα , shared L2

cache ,οχτώ ελεγκτές μνήμης,2 protocol engines και ένα network router όλα σε ένα die (Σχήμα 3.7).



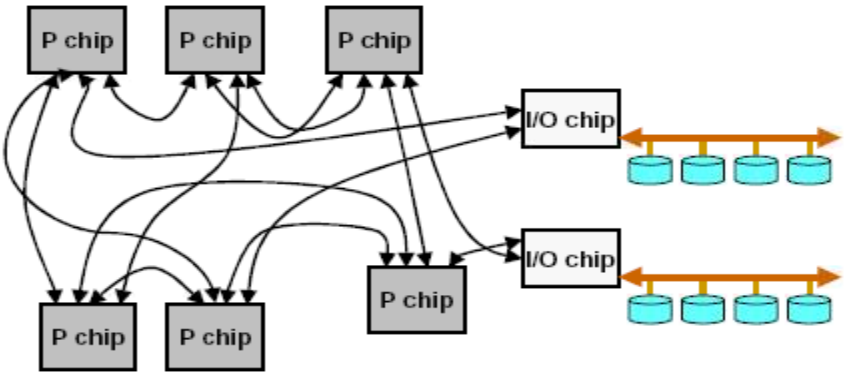
Σχήμα 3.7: Η δομή του processing chip [6]



Σχήμα 3.8: *Η δομή του I/O chip* [6]

Το chip που περιγράψαμε, δεν έχει την δυνατότητα για I/O ενέργειες. Οι ενέργειες αυτές γίνονται στο I/O chip (*σχήμα 3.8*) το οποίο είναι συγκριτικά μικρό σε σχέση με αυτό που χρησιμοποιείται για τις κυρίως πράξεις. Η κύρια διαφορά του με το processing chip είναι ότι περιέχει μόνο ένα πυρήνα και συνεπώς μόνο μια L2 cache και ένα memory controller.Τα δύο αυτά chips ενωμένα μπορούν να μας δώσουν συστήματα με μεγάλη απόδοση.

Τα modules αυτά όπως αναφέραμε και πριν αποτελούνται από επεξεργαστές Alpha. Τα cores αυτά έχουν συχνότητα 500MHz και pipeline 8 σταδίων. Επίσης κάνουν issue μία εντολή ανά κύκλο και τα issues γίνονται in order. Έχουν ξεχωριστή L1 cache για τα instructions και τα data που η καθεμία είναι 64KB με δομή two-way set associative. Η L2 cache που χρησιμοποιείται ενιαία και από τους οχτώ πυρήνες είναι 1MB χωρισμένη σε οχτώ banks, ένα για κάθε core. Κάθε bank είναι 8-way set associative, χρησιμοποιεί round robin replacement policy έχει την δυνατότητα να επικοινωνήσει μέσω του κεντρικού interface με τα άλλα modules στο chip.



Σχήμα 3.9: *Piranha σύστημα με έξι processing και δύο I/O chip* [6]

Μεγάλο πλεονέκτημα του σχεδίου αυτού της Compaq είναι ότι κάθε σύστημα μπορεί να έχει διαφορετικό αριθμό processing nodes δίνοντας κάθε φορά στο σχέδιο την ισχύ και την επίδοση που χρειάζεται. Στην *σχήμα 3.9* μπορούμε να δούμε ένα σύστημα piranha με οχτώ processing nodes και δύο I/O nodes.Με τον τρόπο αυτό μπορούν να επεξεργαστούν στον ίδιο χρόνο πολλά παραπάνω δεδομένα επιτυγχάνοντας καλύτερη επίδοση σε συστήματα που μπορούν να μας δώσουν τον απαιτούμενο παραλληλισμό όπως οι on-line transactions με διαφορετικές βάσεις δεδομένων.

Στη συνέχεια. θα γίνει μία σύντομη παρουσίαση για τους επεξεργαστές που αποτελούν το γρηγορότερο supercomputer αυτή τη στιγμή , τον Roadrunner. Οι παρακάτω επεξεργαστές συνδυασμένοι στον Roadrunner φτάνουν το 1,38 Pflop/s peak performance.

3.7 AMD OPTERON

Η βασική υπολογιστική μονάδα του Roadrunner είναι ο επεξεργαστής της AMD, OPTERON [7]. Ο επεξεργαστής αυτός περιέχει δύο πυρήνες και πετυχαίνει επίδοση 1,8GHz. Επίσης κάθε πυρήνας έχει 64 KB data ,64 KB instruction L1 cache και 2 MB L2 unified cache.Ο καθένας προσφέρει επίδοση 7,2Gflops/s χρησιμοποιώντας double precision (DP) και 14,4Gflops/s χρησιμοποιώντας single precision (SP).

3.8 IBM PowerXCell 8i

Ο IBM PowerXCell8i [7] είναι μια παραλλαγή του Cell BE που χρησιμοποιήθηκε στον Roadrunner λόγω των χαμηλών επιδόσεων του αρχικού μοντέλου στις double precision πράξεις. Ο επεξεργαστής αυτός χρονίζεται στα 3,2 GHz και περιέχει συνολικά εννιά πυρήνες. Οι οχτώ απο αυτούς έχουν βοηθητική λειτουργία και για αυτό λέγονται Synergistic Processing Elements (SPE).Κάθε SPE περιέχει μια SIMD (single instruction multiple data) unit που μπορεί να κάνει issue συνολικά 4 double precision floating point ή 8 single point floating point πράξεις τον κύκλο. Ο κύριος πυρήνας λέγεται Power Processing Element (PPE) και μπορεί να κάνει issue 2 double precision floating point πράξεις τον κύκλο. Επίσης ο κύριος πυρήνας δομείται με ιεραρχία cache. Αναλυτικά περιέχει 32 KB L1 data cache και 32 KB L1 instruction cache όπως επίσης και 512 KB L2 cache. Επιπρόσθετα ο επεξεργαστής αυτός υποστηρίζει DDR2 μνήμη μέχρι και 32 GB. Τέλος είναι σημαντικό να πούμε είναι σημαντικό να πούμε ότι οι επεξεργαστές PowerXcell 8i παίζουν ρόλο accelerator στο σύστημα και προσφέρουν το 95 % της συνολικής επίδοσης του supercomputer.

3.9 Γενική περιγραφή

Σε αυτό το κεφάλαιο παρουσιάστηκαν κάποια μοντέλα πολυπύρηνων επεξεργαστών. Αναλυτικότερα, παρουσιάστηκαν κάποιοι επεξεργαστές που υπάρχουν στην αγορά, όπως επίσης και κάποια ερευνητικά project, που αποτελούν τον προάγγελο, όσων θα ακολουθήσουν στο μέλλον.

Πιο συγκεκριμένα, παρουσιάστηκε ο ***Xeon*** της Intel όπου είναι η σειρά της Intel που ενσωματώνει σε servers και βγαίνουν με δύο και τέσσερις επεξεργαστές. Ο ***Power5*** της IBM, όπου είναι ένας dual-core επεξεργαστής, όπου το κάθε core τρέχει 2 threads. Ο **Intel 80-core terascale processor**, όπου είναι ένα επεξεργαστής 80 πυρύνων και

κατασκευάστηκε με κύριο στόχο να μελετηθούν οι τρόποι για διαχείριση ενέργειας και να διερευνηθούν διαφορετικές προσεγγίσεις για την διασύνδεση των πυρήνων μέσα στο die. Ο επαναστατικός ***Larrabee*** της Intel, που προσδοκά να καλύψει τόσο την αγορά των καρτών γραφικών, βραχυπρόθεσμα, όσο και την αγορά των επεξεργαστών, μακροπρόθεσμα και χρησιμοποιεί inorder πυρήνες. Ο ***Piranha***, όπου είναι ένας επεξεργαστής με 8 πυρήνες, και αποτελεί ένα ερευνητικό πρόγραμμα που ανέπτυξε η Compaq το 2000 με σκοπό να δημιουργήσουν σε λίγο χρόνο και με μικρό κόστος ένα σύστημα που μπορεί να επεξεργάζεται on-line transactions αποδοτικά. Ο ***AMD OPTERON***, που είναι ένας επεξεργαστής δύο πυρήνων και πετυχαίνει επίδοση 1,8GHz και ο **IBM PowerXCell 8i**, που χρονίζεται στα 3,2 GHz και περιέχει συνολικά εννιά πυρήνες. Οι δύο τελευταίοι επεξεργαστές, χρησιμοποιούνται στον πιο γρήγορο supercomputer αυτή τη στιγμή, τον ***Roadrunner***.

Η εκτενής αναφορά στους παραπάνω επεξεργαστές, είχε σαν σκοπό να παρουσιάσει την κατάσταση η οποία επικρατεί στους πολυπύρηνους επεξεργαστές σήμερα και να προσπαθήσει να επεξηγήσει τα κύρια σημεία της αρχιτεκτονικής τους.

4. ΚΕΦΑΛΑΙΟ 4

Προσομοιωτής SESC

4.1 Τι είναι γενικά ο προσομοιωτής....23
4.2 Παλιότερη εμπειρία με προσομοιωτή....24
4.3 Εισαγωγή για τον προσομοιωτή SESC....25
4.4 Τι είναι το superscalar out-of-order pipeline....27
4.5 Πως το SESC διαχειρίζεται την προσομοίωση....29

4.1 Τι είναι γενικά ο προσομοιωτής

Στην έρευνα της microarchitecture , οι ερευνητές έχουν κάποιο είδος πρότασης για έναν μικροεπεξεργαστή που θα είναι καλύτερος από την τρέχουσα κατάσταση προόδου. Αυτός ο μικροεπεξεργαστής θα πρέπει να είναι γρηγορότερος, να χρησιμοποιεί λιγότερη ενέργεια και να είναι πιο αξιόπιστος.

Σε κάθε περίπτωση, δεδομένου ότι ένα τέτοιο εγχείρημα θα ήταν ακριβό να σχεδιαστεί και να κατασκευαστεί, οι ερευνητές δημιούργησαν τους προσομοιωτές μικροεπεξεργαστών. Υπάρχουν διαφορετικές προσεγγίσεις σε αυτό. Είναι οι trace-driven, δηλ., αυτοί που χρησιμοποιούν τα trace instructions των εφαρμογών, όπως επίσης και οι

execution driven όπου είναι και οι περισσότεροι σήμερα, δηλ., αυτοί που εκτελούν πραγματικά την εφαρμογή που προσομοιώνεται.

Πολλοί προσομοιωτές, διαιρούν επίσης την προσομοίωση σε έναν εξομοιωτή, ο οποίος πραγματικά εκτελεί την εφαρμογή που προσομοιώνεται, και έναν προσομοιωτή συγχρονισμού, ο οποίος διαμορφώνει το συγχρονισμό και την ενέργεια της εφαρμογής που προσομοιώνεται. Επίσης, θα πρέπει να υπάρξει μέρος του προσομοιωτή, το οποίο είναι event-driven που αυτό σημαίνει ότι, μέρη του προσομοιωτή θα μπορούν να κάνουν schedule ένα γεγονός, δηλ., ένα function call που μπορεί να εμφανιστεί κάποια στιγμή στο μέλλον.

Οι σύγχρονοι προσομοιωτές μικροεπεξεργαστών, είναι ιδιαίτερα σύνθετα συστήματα λογισμικού. Έχουν αυστηρές απαιτήσεις συγχρονισμού και πρέπει να διατηρήσουν μια υψηλή πιστότητα ακρίβειας της προσομοίωσης. Αυτή η πιστότητα πρέπει να είναι η βασική απαίτηση οποιουδήποτε προσομοιωτή. Ταυτόχρονα, δε θα πρέπει να υπάρχει κανένα σημείο σε έναν προσομοιωτή, που δεν απεικονίζει με ακρίβεια τη συμπεριφορά μικροεπεξεργαστών.

Ένα δεύτερο πολύ σημαντικό στοιχείο των προσομοιωτών είναι η απόδοση. Δεδομένου, ότι ένας προσομοιωτής εκτελεί τις εντολές ένα εκατομμύριο φορές πιο αργά από έναν πραγματικό επεξεργαστή, η ταχύτητα εκτέλεσης είναι σημαντική. Ένας γρηγορότερος προσομοιωτής, θα είναι σε θέση να εκτελέσει τις συγκριτικές μετρήσεις επιδόσεων(benchmarks) γρηγορότερα. Μια ερευνητική εργασία θα απαιτήσει συχνά χιλιάδες ώρες του χρόνου προσομοίωσης, και η μείωση αυτού είναι σημαντική. [8]

4.2 Παλιότερη εμπειρία με προσομοιωτή

Στα πλαίσια του μαθήματος της αρχιτεκτονικής υπολογιστών, είχα χρησιμοποιήσει τον προσομοιωτή M5 [9] και τη σουίτα εφαρμογών SPLASH [10], κάτι

το οποίο με βοήθησε να εμφυσήσω με μεγαλύτερη ευκολία, τα στοιχεία του νέου προσομοιωτή που θα χρησιμοποιούσα.

Το M5 δημιουργήθηκε απο την National Science Foundation (NSF) με την βοήθεια διάφορων κολοσσών στον χώρο της πληροφορικής, όπως η Hewlett-Packard,η Intel, η IBM και η Sun και είναι μια πλατφόρμα για έρευνα πάνω στην αρχιτεκτονική υπολογιστών. Χαρακτηριστικό του M5, είναι ότι περιέχει system-level αρχιτεκτονική, όπως και αρχιτεκτονική μικροεπεξεργαστών.

Το M5 λειτουργεί στα περισσότερα λειτουργικά συστήματα (Linux, MacOS X ,Solaris, OpenBSD ,Cygwin), βασικά χαρακτηριστικά του είναι ότι υποστηρίζει ένα αριθμό απο ISA (Instruction Set Architecture) και λειτουργικά συστήματα για full system simulation (booting an entire operating system), όπως και για Syscall emulation (running one or more applications by emulating system calls).

4.3 Εισαγωγή για τον προσομοιωτή SESC

Με την πάροδο της διπλωματικής, αποφασίσαμε να χρησιμοποιήσω (ύστερα από προτροπή του επιβλέποντα καθηγητή μου κ. Trancoso) έναν άλλο προσομοιωτή που ονομάζεται SESC [8], [11]. Αυτό συνέβη, γιατί πιστέψαμε πως με αυτόν τον προσομοιωτή θα υλοποιηθούν οι στόχοι μου πιο αποδοτικά και με μεγαλύτερη επιτυχία. Πρόκειται για έναν προσομοιωτή αρκετά γρήγορο, που υποστηρίζει την εκτέλεση out-of-order επεξεργαστών.

Το SESC, είναι ένας αρχιτεκτονικός προσομοιωτής μικροεπεξεργαστών που αναπτύχθηκε από την ερευνητική ομάδα i-acoma στο UIUC και διάφορες ομάδες σε άλλα πανεπιστήμια και προσαρμόζει διαφορετικές αρχιτεκτονικές επεξεργαστών, όπως οι single processors, οι chip multiprocessors και οι processors-in-memory. Επίσης, προσαρμόζει ένα πλήρες out-of-order pipeline με branch prediction, caches, buses και

οποιοδήποτε άλλο συστατικό που χρειάζεται ένας σύγχρονος επεξεργαστής, για ακριβή προσομοίωση.

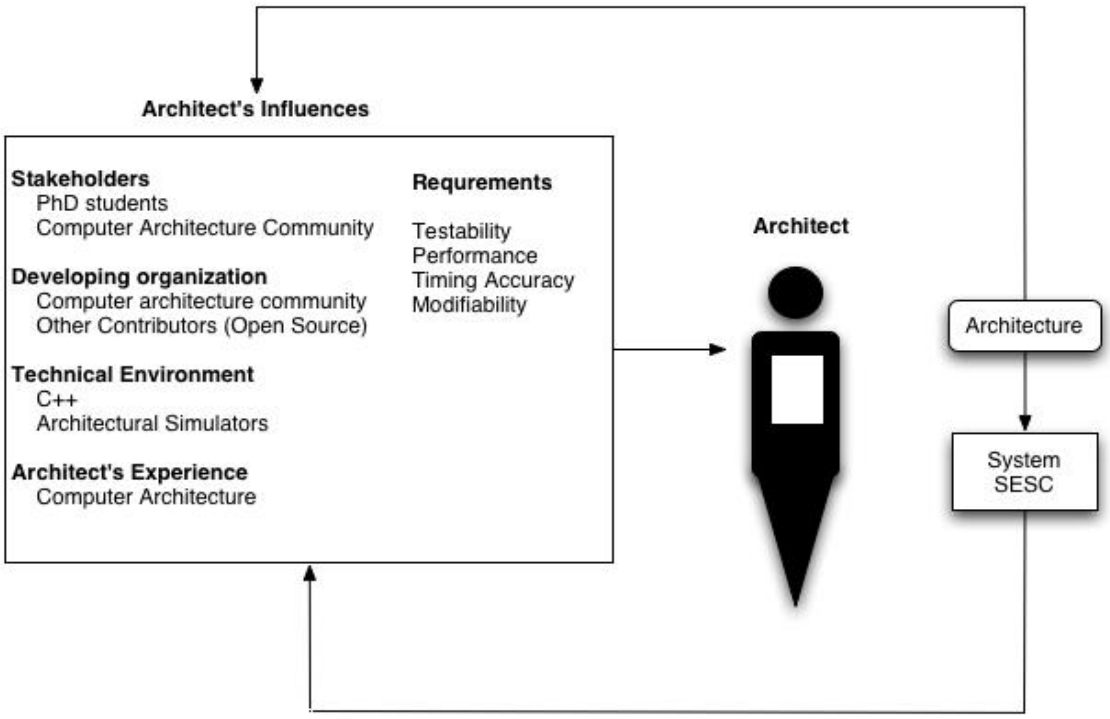
Το SESC, ανήκει στην κατηγορία του event-driven προσομοιωτή. Έχει έναν εξομοιωτή που είναι φτιαγμένος από το MINT , ένα παλιό project που εξομοιώνει έναν επεξεργαστή MIPS. Πολλές λειτουργίες στον πυρήνα του προσομοιωτή, καλούνται κάθε κύκλο του επεξεργαστή, αλλά πολλές άλλες καλούνται μόνο όταν χρειάζεται, χρησιμοποιώντας τα events.

Η διαμόρφωση και η διαθεσιμότητα του πηγαίου κώδικα, κάνουν το SESC ένα εξαιρετικό εργαλείο για την αποτίμηση διαφόρων ερευνητικών προτάσεων. Έχει πάρα πολλά πλεονεκτήματα σε σχέση με άλλους προσομοιωτές, όπως το Simplescalar. Το SESC, αναπαριστά ένα πλήθος από αρχιτεκτονικές, συμπεριλαμβάνοντας δυναμικούς superscalar επεξεργαστές, CMPs, processor-in-memory και multithreading αρχιτεκτονικές. Το Simplescalar εστιάζει μόνο σε μονοπύρηνους επεξεργαστές, ενώ το SESC προσομοιώνει και πολυπύρηνους επεξεργαστές και επιπλέον είναι πολύ γρήγορο, ικανό να εκτελεί 1.5 εκατομμύριο εντολές το δευτερόλεπτο σε έναν Pentium4, 3GHz επεξεργαστή.

Ο βασικός developer του SESC είναι ο Jose Renau, τον οποίο και θα ήθελα να ευχαριστήσω, αφού με βοήθησε πολύ σε όποιες δυσκολίες συνάντησα κατά τη χρησιμοποίηση του SESC, μέσα από το mailing list του προσομοιωτή.

Το SESC είναι γραμμένο σε C++ και είναι open source, δίνοντας τη δυνατότητα σε όλους μας να τον επεκτείνουμε. Επίσης, τρέχει σε πολλά UNIX λειτουργικά συστήματα όπως Linux, Darwin/MacOS X και επίσης τρέχει σε .

Στο παρακάτω σχήμα (*σχήμα 4.1*), μπορούμε να δούμε ένα σχεδιάγραμμα που αναπαριστά τον κύκλο δημιουργίας του SESC που περιγράψαμε πιο πάνω:



Σχήμα 4.1: *Κύκλος δημιουργίας του SESC* [8]

4.4 Τι είναι το superscalar out-of-order pipeline

Μπορούμε να φανταστούμε έναν απλό επεξεργαστή σαν ένα μαύρο κουτί, που εκτελεί μια εντολή κάθε κύκλο μηχανής. Οι πρώτοι μικροεπεξεργαστές, ακολούθησαν αυτή την προσέγγιση, όμως οι ερευνητές κατάλαβαν ότι καμία εντολή δε χρησιμοποιεί τον επεξεργαστή όλη την ώρα και θα μπορούσε να είναι δυνατό για έναν επεξεργαστή, να χρησιμοποιεί ταυτόχρονα περισσότερες από μία εντολές(αυτό επιτυγχάνεται με το pipeline). Το pipelining, είναι μια τεχνική των μικροεπεξεργαστών, που επιτρέπει σε πολλές εντολές να επικαλυφθούν, ενώ αυτές εκτελούνται στον επεξεργαστή. Γι' αυτό το λόγο το pipeline δεν είναι καθόλου εύκολο και εισάγει πολλές προκλήσεις(exception handling, branch misprediction). Ωστόσο, στις μέρες μας, οι περισσότεροι επεξεργαστές χρησιμοποιούν το pipelining και μάλιστα σε πολύ μεγάλο βαθμό. Ένας Intel Pentium έχει πάνω από 30 στάδια pipelining.

Μπορούμε να φανταστούμε έναν επεξεργαστή όπου κάθε στάδιο του pipeline απαιτεί έναν κύκλο. Σε αυτή την περίπτωση, μπορούμε να δούμε τον επεξεργαστή σαν μια ουρά αναμονής: η πρώτη εντολή που προσκομίζεται(fetch) είναι η πρώτη εντολή που δεσμεύεται(commit). Επίσης, μπορούμε να υποθέσουμε ότι η εκτέλεση ενός πολλαπλασιασμού κινητής υποδιαστολής, θα πάρει περισσότερο χρόνο απ' ότι αν συγκρίναμε έναν ακέραιο αριθμό με το μηδέν. Ωστόσο, η υποστήριξη multicycle εντολών , σταματά να καθιστά το pipeline σαν μια ουρά αναμονής, δεδομένου ότι μια εντολή που προσκομίζεται σε χρόνο t μπορεί να τελειώσει πριν από μια εντολή που προσκομίζεται σε χρόνο t-1. Σε αυτή την περίπτωση όμως, οι out-of-order επεξεργαστές, θα πρέπει να εξετάσουν πολλά προβλήματα κατά το χειρισμό των εξαιρέσεων, αφού για να γίνει κάτι τέτοιο θα πρέπει να τηρούνται οι κατάλληλες προϋποθέσεις.

Κάτω από τέλειες συνθήκες, μπορούμε να υποθέσουμε, ότι ένας out-of-order επεξεργαστής μπορεί να εκτελεί μία εντολή κάθε κύκλο μηχανής. Για να βελτιώσουμε την απόδοση περαιτέρω θα μπορούσαμε να επιτρέψουμε την εκτέλεση πολλών εντολών σε ένα κύκλο μηχανής. Αυτό μπορεί να επιτευχθεί με δύο τρόπους:

- **Με πολύ μεγάλες instruction word architectures (VLIW):** ο επεξεργαστής κάνει issue έναν σταθερό αριθμό εντολών ανά κύκλο.
- **Superscalar processors:** ο επεξεργαστής προσπαθεί να κάνει issue περισσότερες από μία εντολές κάθε κύκλο ώστε να κρατηθούν όλες οι λειτουργικές μονάδες πολυάσχολες. Βεβαίως, είναι λογικό σε αυτή την περίπτωση να υπάρξουν περιορισμοί στα παράλληλα issues, όπως το να μην έχουμε περισσότερη από μία εντολή μνήμης, κάθε κύκλο ρολογιού.

Προκειμένου, να μεγιστοποιηθεί ο αριθμός των εντολών που γίνονται issue, χρησιμοποιούνται τεχνικές static και dynamic scheduling. Οι static scheduled processors χρησιμοποιούν in-order εκτέλεση, ενώ οι dynamic scheduled processors χρησιμοποιούν out-of-order εκτέλεση.

4.5 Πως το SESC διαχειρίζεται την προσομοίωση

Στο SESC, οι εντολές εκτελούνται σε ένα module εξομοίωσης, που εξομοιώνει το MIPS Instruction Set Architecture (ISA). Ουσιαστικά δηλαδή, εξομοιώνει τις εντολές που βρίσκονται στο binary της εφαρμογής.

Ο εξομοιωτής λοιπόν, επιστρέφει instruction objects στο SESC που χρησιμοποιούνται έπειτα για τον timing simulator. Αυτά τα instruction objects περιέχουν όλες τις πληροφορίες που είναι απαραίτητες για τον ακριβή συγχρονισμό. Δηλαδή περιλαμβάνει τη διεύθυνση της εντολής, τις διευθύνσεις οποιωνδήποτε store και loads που γίνονται στη μνήμη, τους source και destination registers και όλα τα functional units που χρησιμοποιούνται από την εντολή. Ο προσομοιωτής χρησιμοποιεί αυτές τις πληροφορίες για να υπολογίσει πόσο χρόνο χρειάζεται, να εκτελεστεί η εντολή μέσω του pipeline.

Ο τρόπος που περιγράψαμε παραπάνω, μπορεί να αιτιολογηθεί στο ότι είναι πολύ γρηγορότερο να εκτελεστεί μια εντολή σε έναν απλό εξομοιωτή και επίσης είναι πολύ πιο εύκολο για το πρόγραμμα και για το debug όταν η εκτέλεση και ο συγχρονισμός είναι διαχωρισμένα. Ο timing simulator, είναι πολύ σύνθετος και αν δεν έχει επιπτώσεις στον υπολογισμό των εντολών, δε χρειάζεται να είναι 100% ακριβής. Αν δηλαδή, ένα σφάλμα έχει πιθανότητα να εμφανιστεί 0,1%, δε θα μας ενοχλήσει και θα το κάνουμε αποδεκτό.

5. **Κεφάλαιο 5**

Πειραματικά Αποτελέσματα

5.1 Εισαγωγή 31
5.2 Επιλογή των benchmarks 32
5.3 Περιγραφή configurations των benchmarks 35
5.4 Execution time vs Simulation time 36
5.5 Ανάλυση αποτελεσμάτων για το FFT benchmark 37
5.5.1 FFT-execution time 37
5.5.2 FFT-speedup 40
5.5.3 FFT-Υπολογισμός χώρου του chip 43
5.6 Ανάλυση αποτελεσμάτων για το CHOLESKY benchmark 44
5.6.1 CHOLESKY-execution time 44
5.6.2 CHOLESKY-speedup 47
5.6.3 CHOLESKY-Υπολογισμός χώρου του chip 49
5.7 Ανάλυση αποτελεσμάτων για το LU benchmark 51
5.7.1 LU-execution time 51
5.7.2 LU-speedup 54
5.7.3 LU-Υπολογισμός χώρου του chip 56
5.8 Ανάλυση αποτελεσμάτων για το RADIX benchmark 58
5.8.1 RADIX-execution time 58
5.8.2 RADIX-speedup 61
5.8.3 RADIX-Υπολογισμός χώρου του chip 63
5.9 Ανάλυση αποτελεσμάτων για το OCEAN benchmark 64
5.9.1 OCEAN-execution time 65
5.9.2 OCEAN-speedup 68
5.9.3 OCEAN-Υπολογισμός χώρου του chip 71
5.10 Ανάλυση αποτελεσμάτων για το BARNES benchmark 72

5.10.1 BARNES-execution time 73
5.10.2 BARNES-speedup 75
5.10.3 BARNES-Υπολογισμός χώρου του chip 77
5.11 Περίληψη και σύνοψη των πειραματικών αποτελεσμάτων 79

5.1 Εισαγωγή

Αποφασίζοντας λοιπόν, να δουλέψω με το SESC [8], έπρεπε να μελετήσω τον τρόπο λειτουργίας του προσομοιωτή που περιέγραψα παραπάνω. Αυτό θα με διευκόλυνε στη συνέχεια ώστε να καταλάβω σε βάθος τον προσομοιωτή και να μπορέσω να ανταπεξέλθω στις όποιες δυσκολίες μπορεί να συναντούσα. Συν τοις άλλοις, μπόρεσα να κατανοήσω το πώς δουλεύει ο προσομοιωτής στον οποίο μου ανατέθηκε να υλοποιήσω τη διπλωματική μου, ώστε αν χρειαζόταν να τον επεκτείνω ή να αλλάξω κάποιες ρυθμίσεις, να ήταν πιο εύκολο.

Μετά τη μελέτη λοιπόν του προσομοιωτή, έπρεπε να τον εγκαταστήσω, κάτι που έγινε με επιτυχία. Το SESC έγινε build στη μηχανή thales και μέσω αυτής έτρεχα τα διάφορα benchmarks. Η μηχανή thales, είναι μία μηχανή του Πανεπιστημίου Κύπρου, αποτελούμενη από 4 Dual Core AMD Opteron επεξεργαστές, συγχρονισμένοι στα 1,8 GHz και 4 GB main memory.

Μετά την εγκατάσταση του προσομοιωτή, ήθελα να τρέξω ένα δικό μου απλό πρόγραμμα πάνω στο SESC για να βεβαιωθώ ότι δουλεύει. Έτρεξα λοιπόν, ένα απλό hello world, εγκαθιστώντας έναν cross-compiler και το SESC ανταποκρίθηκε απόλυτα.

5.2 Επιλογή των benchmarks

Στη συνέχεια, έπρεπε να αποφασίσω ποια θα ήταν τα benchmarks τα οποία θα προσομοίωνα και θα έπαιρνα τα αποτελέσματά μου. Αρχικά, σκέφτηκα να κάνω μια δική μου εφαρμογή, η οποία θα έκανε διάφορες πράξεις ώστε να αύξανε την πολυπλοκότητα της προσομοίωσης. Ύστερα όμως, από παρότρυνση του καθηγητή μου κ. Trancoso, αποφάσισα να χρησιμοποιήσω μια έτοιμη σουίτα εφαρμογών η οποία θα περιλάμβανε πολλά benchmarks, πολλών διαφορετικών τύπων. Κάτι τέτοιο, θα ήταν πολύ πιο ασφαλές για τα αποτελέσματά μου, αφού τα διάφορα configurations θα δοκιμάζονταν σε πολλές και διαφορετικές συνθήκες.

Έτσι, αποφάσισα να χρησιμοποιήσω τη σουίτα εφαρμογών SPLASH2 [12], μια πιο εξελιγμένη σουίτα του SPLASH [10] που είχα χρησιμοποιήσει για τον προσομοιωτή M5. Για να είναι κάτι τέτοιο εφικτό, έπρεπε να ελέγξω αν υπήρχε συμβατότητα της συγκεκριμένης σουίτας με το SESC. Πράγματι, υπήρχε απόλυτη συμβατότητα με το SESC και το μόνο που έπρεπε να κάνω ήταν να κατεβάσω τα pro-compiled binaries [13] για SESC. Επίσης, έπρεπε να κατεβάσω το SPLASH2 package [13] που περιέχει τα input files που χρησιμοποιεί το SPLASH2 και να δημιουργήσω soft-links για τα binaries. Το SPLASH2 περιέχει 4 kernels και 8 applications, οπότε είχα πληθώρα επιλογών για τη διεκπεραίωση των πειραμάτων μου.

Για τα πειράματά μου, χρησιμοποίησα τα 4 kernels και 2 applications. Συγκεκριμένα τα 4 ***kernels*** ήταν: το *fft,* το *cholesky,* το *lu* και το *radix* και τα 2 **applications** ήταν: το *ocean* και το *barnes.*

Μετά την επιλογή των εφαρμογών, έπρεπε να μελετήσω το configuration file που στο συγκεκριμένο προσομοιωτή ήταν το smp.conf (*σχήμα 5.1*). Δεδομένου ότι η δική μου μελέτη είναι πάνω στους επεξεργαστές, έπρεπε να εστιάσω την προσοχή μου στα configurations που είχαν άμεση σχέση με την αλλαγή των ρυθμίσεων του επεξεργαστή. Συζητώντας λοιπόν και με τον Παναγιώτη Πετρίδη που με βοήθησε πολύ στο να βγει σε πέρας η διπλωματική εργασία, αποφάσισα να αλλάξω το issue των επεξεργαστών.

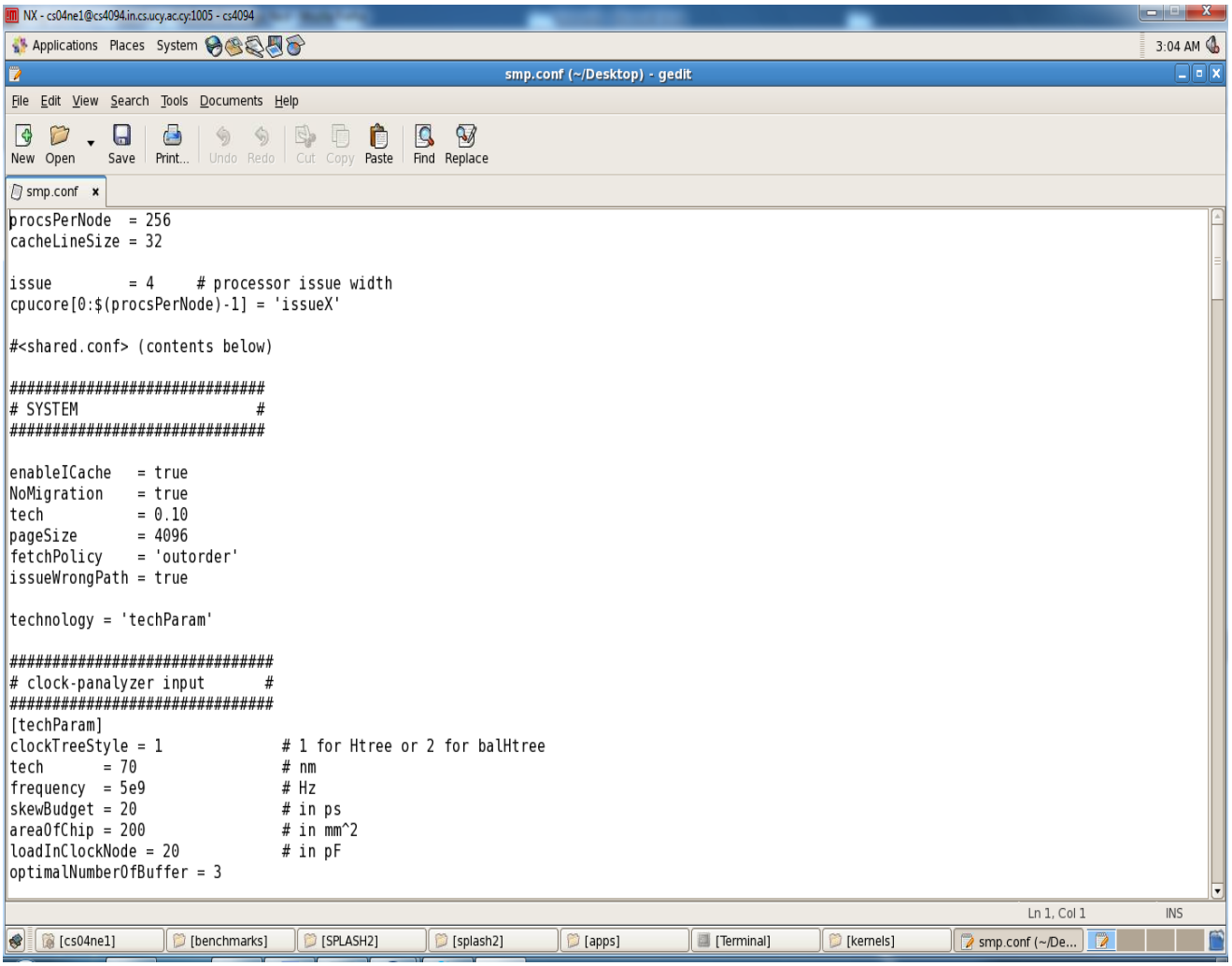
Αλλάζοντας το issue, ουσιαστικά άλλαζα όλη τη δομή του επεξεργαστή, αφού το issue ήταν βασική παράμετρος του επεξεργαστή και για πολλά άλλα κομμάτια του.
Στον παρακάτω πίνακα (*πίνακας 5.1*) φαίνονται οι αλλαγές που γινόντουσαν με την αλλαγή του issue:

Τύπος μεταβλητής	**Ισοδυναμία μεταβλητής**
areaFactor	($(issue)*$(issue)+0.1)/16
fetchWidth	$(issue)
instQueueSize	2*$(issue)
issueWidth	$(issue)
retireWidth	$(issue)+1
maxBranches	16*$(issue)
maxLoads	10*$(issue)+16
maxStores	10*$(issue)+16
robSize	36*$(issue)+32
intRegs	32+16*$(issue)
fpRegs	32+12*$(issue)

Πίνακας 5.1: *Αλλαγές στη δομή του επεξεργαστή με την αλλαγή του issue*

Παρατηρούμε, ότι όλες οι παραπάνω παράμετροι επηρεάζονται άμεσα από το issue, αφού τη συγκεκριμένη μεταβλητή, την παίρνουν σαν παράμετρο. Έχοντας λοιπόν, σαν βάση την αλλαγή του issue, έπρεπε να αποφασίσω τις τιμές που θα δώσω στο issue. Επειδή ήθελα τα configurations μου, να προσεγγίζουν πραγματικές τιμές, διάλεξα σαν μέγιστο όριο του issue στο 32. Έτσι, οι τιμές που του έδωσα για τα πειράματά μου ήταν: 4, 8, 16 και 32 για out-of-order επεξεργαστές. Επίσης ένα άλλο configuration που προσομοίωσα ήταν αυτό για την εκτέλεση σε in-order επεξεργαστές. Θεώρησα, ότι είναι

ένα πολύ χρήσιμο configuration για να δείξω τη διαφορά σε χρόνους σε σχέση με την εκτέλεση out-of-order επεξεργαστών.



Σχήμα 5.1: *Ενδεικτική αναπαράσταση του configuration file*

5.3 Περιγραφή configurations των benchmarks

Για κάθε ένα από τα benchmarks, χρησιμοποίησα είτε λόγω συνθηκών είτε λόγω αδυναμίας του benchmark να τρέξει σε κάποιες συγκεκριμένες ρυθμίσεις, διαφορετικά configurations. Συγκεκριμένα:

- Για το **fft**, το **lu** και το **ocean** χρησιμοποίησα:
  - 4 issue, out-of-order σε 2, 4, 8, 16, 32, 64, 128 και 256 επεξεργαστές.
  - 8 issue, out-of-order σε 2, 4, 8, 16, 32, 64, 128 και 256 επεξεργαστές.
  - 16 issue, out-of-order σε 2, 4, 8, 16, 32, 64, 128 και 256 επεξεργαστές.
  - 32 issue, out-of-order σε 2, 4, 8, 16, 32, 64, 128 και 256 επεξεργαστές.
  - in-order σε 2, 4, 8, 16, 32, 64, 128 και 256 επεξεργαστές.

  Δε χρησιμοποίησα παραπάνω επεξεργαστές, λόγω αδυναμίας των συγκεκριμένων benchmarks, να ανταπεξέλθουν στη συγκεκριμένη ρύθμιση.

- Για το **cholesky** και **barnes** χρησιμοποίησα:
  - 4 issue, out-of-order σε 2, 4, 8, 16, 32, 64 και 128 επεξεργαστές.
  - 8 issue, out-of-order σε 2, 4, 8, 16, 32, 64 και 128 επεξεργαστές.
  - 16 issue, out-of-order σε 2, 4, 8, 16, 32, 64 και 128 επεξεργαστές.
  - 32 issue, out-of-order σε 2, 4, 8, 16, 32, 64 και 128 επεξεργαστές.
  - in-order σε 2, 4, 8, 16, 32, 64 και 128 επεξεργαστές.

  Δε χρησιμοποίησα παραπάνω επεξεργαστές, λόγω αδυναμίας των συγκεκριμένων benchmarks, να ανταπεξέλθουν στη συγκεκριμένη ρύθμιση.

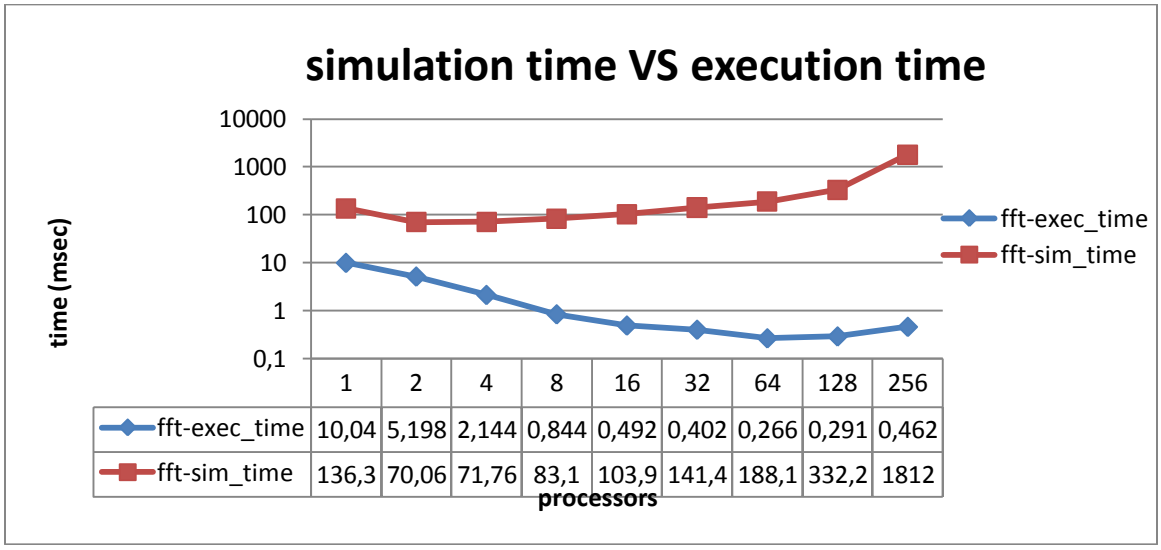
- Για το **radix**, χρησιμοποίησα:
  - 4 issue, out-of-order σε 2, 4, 8, 16, 32, 64 επεξεργαστές.
  - 8 issue, out-of-order σε 2, 4, 8, 16, 32, 64 επεξεργαστές.
  - 16 issue, out-of-order σε 2, 4, 8, 16, 32, 64 επεξεργαστές.
  - 32 issue, out-of-order σε 2, 4, 8, 16, 32, 64 επεξεργαστές.
  - in-order σε 2, 4, 8, 16, 32, 64επεξεργαστές.

Δε χρησιμοποίησα παραπάνω επεξεργαστές, λόγω αδυναμίας του συγκεκριμένου benchmark, να ανταπεξέλθει στη συγκεκριμένη ρύθμιση.

Κάθε επεξεργαστής έχει ξεχωριστή μνήμη cache επιπέδου ένα που χωρίζεται σε δυο μέρη, την L1 cache για instruction και την L1 cache για Data. Το κάθε ένα από τα δυο αυτά κομμάτια της L1 έχει χωρητικότητα 32 KB και associativity 2 ενώ ακολουθεί πολιτική Write Through (WT), κατά την οποία η πληροφορία γράφεται και στο block στην μνήμη cache αλλά και στην μνήμη χαμηλότερου επιπέδου.
Επίσης η κρυφή μνήμη επιπέδου ένα ακολουθεί πολιτική αντικατάστασης Least Recently Used (LRU) κατά την οποία το block μνήμης που αντικαθίσταται είναι αυτό που έχει χρησιμοποιηθεί παλαιότερα.

5.4 Execution time vs Simulation time

Στην παρακάτω γραφική (*σχήμα 5.2*) θα παρουσιάσω ενδεικτικά, μια σύγκριση μεταξύ του Execution Time και του Simulation Time. Το execution time, είναι ο χρόνος που χρειάζεται ένα benchmark για να εκτελεστεί, ενώ το simulation time είναι ο χρόνος που χρειάζεται για να ολοκληρωθεί η προσομοίωση. Η σύγκριση θα γίνει σε ένα μόνο benchmark (το *fft*), αφού αυτή είναι αρκετή για να μας δείξει τη διαφορά στους χρόνους αυτών των 2 στοιχείων.



	1	2	4	8	16	32	64	128	256
fft-exec_time	10,04	5,198	2,144	0,844	0,492	0,402	0,266	0,291	0,462
fft-sim_time	136,3	70,06	71,76	83,1	103,9	141,4	188,1	332,2	1812

Σχήμα 5.2: *Σύγκριση μεταξύ Execution time και Simulation time*

Από την παραπάνω γραφική, παρατηρούμε ότι ενώ το execution time, μειώνεται όσο αυξάνονται οι επεξεργαστές, το simulation time αυξάνεται. Όσο λοιπόν αυξάνονταν οι επεξεργαστές και το benchmark εκτελούνταν σε καλύτερους χρόνους, τόσο ο χρόνος προσομοίωσης μεγάλωνε ώστε τελικά να εξάγουμε τα αποτελέσματα. Η γραφική έχει γίνει με logarithmic scale, αφού η διαφορά στους χρόνους ανάμεσα στο execution time και στο simulation time ήταν πολύ μεγάλη.

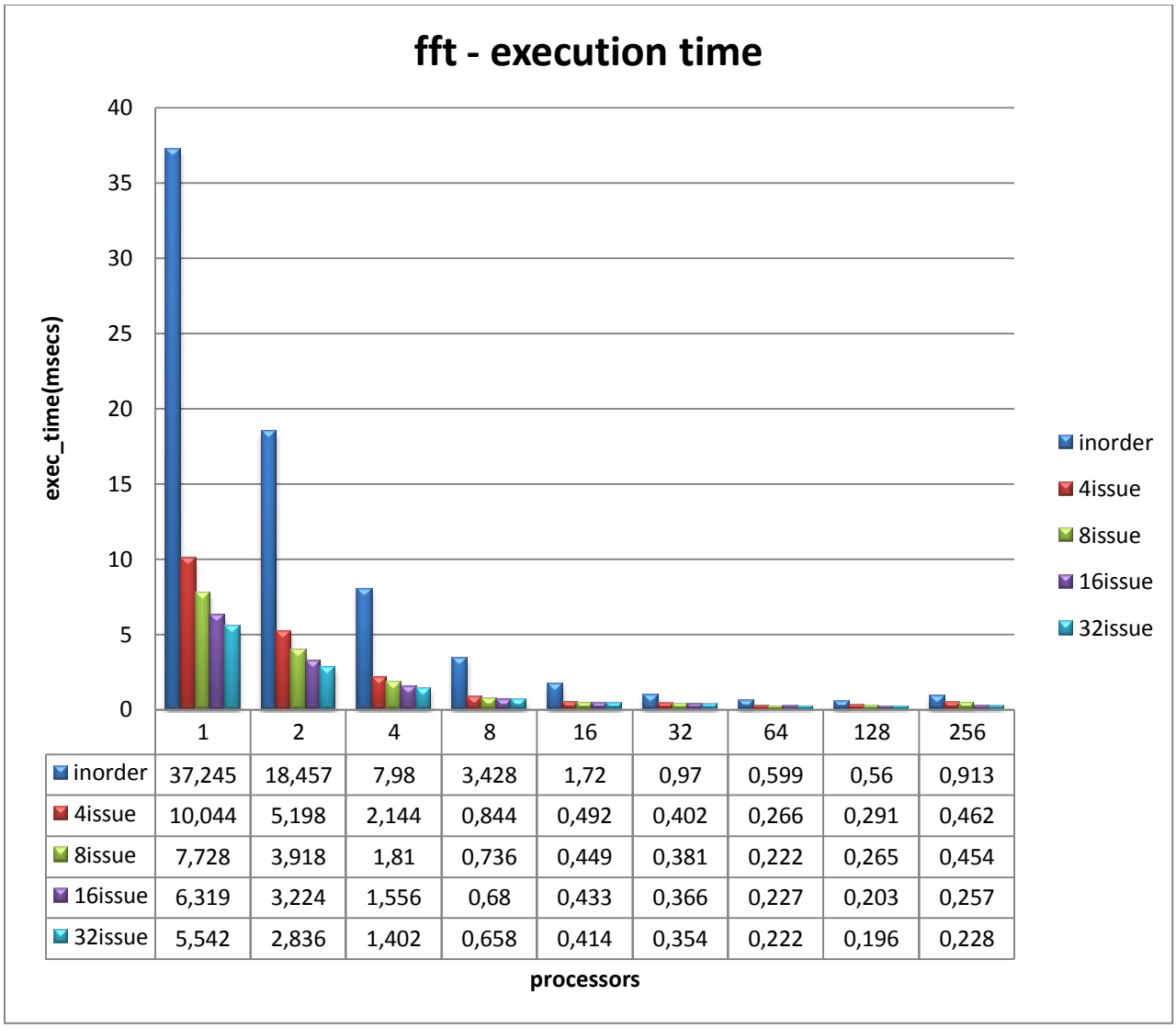
5.5 Ανάλυση αποτελεσμάτων για το FFT benchmark

Το fft kernel [14] είναι μία σύνθετη έκδοση 1-D του radix-√n sixstep FFT algorithm, ο οποίος βελτιστοποιείται για να ελαχιστοποιήσει την interprocessor επικοινωνία. Το data set, αποτελείται από n σύνθετα data points που μετασχηματίζονται και ένα άλλο n σύνθετο data point που αναφέρεται ως roots of unity. Και τα 2 sets of data, οργανώνονται σαν √n x √n που χωρίζονται, έτσι ώστε σε κάθε επεξεργαστή να ανατίθεται, ένα συνεχόμενο set σειρών, που δεσμεύονται στην τοπική μνήμη του.

Η επικοινωνία εμφανίζεται σε τρία matrix transpose steps που απαιτούνται για την all-to-all interprocessor επικοινωνία. Κάθε επεξεργαστής, μεταθέτει ένα συνεχόμενο submatrix √n/p x √n/p, από κάθε άλλο επεξεργαστή και μεταθέτει ένα submatrix τοπικά. Οι μεταθέσεις γίνονται block για να εκμεταλλευτεί την επαναχρησιμοποίηση του cache line. Για να αποφευχθεί το hot-spotting memory, ο επεξεργαστής i, πρώτα μεταθέτει ένα submatrix από τον επεξεργαστή i+1, μετά ένα από τον επεξεργαστή i+2 κλπ.

5.5.1 FFT-execution time

Στο παρακάτω σχήμα (*σχήμα 5.3*), βλέπουμε τη γραφική για τα 5 configurations(4 issue, 8 issue, 16 issue, 32 issue, inorder) όσον αφορά το execution time .



	1	2	4	8	16	32	64	128	256
inorder	37,245	18,457	7,98	3,428	1,72	0,97	0,599	0,56	0,913
4issue	10,044	5,198	2,144	0,844	0,492	0,402	0,266	0,291	0,462
8issue	7,728	3,918	1,81	0,736	0,449	0,381	0,222	0,265	0,454
16issue	6,319	3,224	1,556	0,68	0,433	0,366	0,227	0,203	0,257
32issue	5,542	2,836	1,402	0,658	0,414	0,354	0,222	0,196	0,228

Σχήμα 5.3: *Γραφική που αναπαριστά το fft execution time για μέχρι και 256 επεξεργαστές*

Από την παραπάνω γραφική παρατηρούμε αρχικά, ότι για την inorder εκτέλεση εντολών, έχουμε πολύ μεγάλες τιμές, συγκριτικά με τις υπόλοιπες. Αυτό, είναι κάτι που αναμέναμε, αφού είναι απολύτως λογικό να χρειάζεται μια εφαρμογή περισσότερο χρόνο, όταν είναι αναγκασμένη να εκτελεί τις εντολές της με τη σειρά. Αυτό έχει σαν αποτέλεσμα να, υπάρχουν καθυστερήσεις λόγω των πολλών εξαρτήσεων.

Για ένα επεξεργαστή στην inorder εκτέλεση, παρατηρούμε έναν χρόνο της τάξεως του 37.245 msecs ενώ για την αντίστοιχη 4issue out-of-order εκτέλεση ένα χρόνο του 10.044 msecs. Η διαφορά λοιπόν, είναι πολύ μεγάλη και βλέπουμε πόσο σημαντική

είναι η out-of-order εκτέλεση. Βλέπουμε επίσης, ότι αυξάνοντας το issue, πετυχαίνουμε όλο και καλύτερο χρόνο και καταφέρνουμε να φτάσουμε για ένα επεξεργαστή στην τιμή του 5,542 msecs με 32 issue.

Αυτή τη βελτίωση των χρόνων, τη συναντάμε και αυξάνοντας τους επεξεργαστές. Παρατηρούμε ότι αυξάνοντας τον αριθμό των επεξεργαστών έχουμε μείωση στους χρόνους και για την inorder εκτέλεση και για το μεταβλητό issue. Η βελτίωση στους χρόνους είναι αισθητή μέχρι και τους 16 επεξεργαστές όπου ο χρόνος είναι 0,433 msecs για 32 issue. Βεβαίως και με την περεταίρω αύξηση των επεξεργαστών έχουμε πάλι βελτίωση χρόνων, απλά προσεγγιστικά οι τιμές αυτές είναι πολύ κοντά με τις προηγούμενες, χωρίς όμως αυτό να σημαίνει πως είναι αμελητέες.

Στους 64 επεξεργαστές, παρατηρούμε μια παράξενη συμπεριφορά, αφού για 8 και 32 issue έχουμε την ίδια τιμή 0,222 msecs ενώ για 16 issue, η τιμή είναι 0.227 msecs. Οι τιμές όμως, είναι πραγματικά πολύ μικρές και αυτή η παρατήρηση, δεν μπορεί σε καμία περίπτωση να χαρακτηριστεί ως κανόνας.

Για 128 επεξεργαστές παρατηρούμε ότι για 4 και 8 issue, οι χρόνοι είναι χειρότεροι σε σχέση με τους αντίστοιχους για 64 επεξεργαστές. Συγκεκριμένα, για 4 issue σε 64 επεξεργαστές έχουμε 0,266 msecs, ενώ για 128 συναντάμε 0,291 msecs. Επίσης, για 8 issue σε 64 επεξεργαστές έχουμε 0,222 msecs, ενώ για 128 συναντάμε 0,265 msecs. Όμως, για 16 issue, 32 issue και inorder εκτέλεση πετυχαίνουμε καλύτερους χρόνους σε σχέση με τους 64 επεξεργαστές. Συγκεκριμένα, για 16 issue σε 64 επεξεργαστές έχουμε 0,227 msecs, ενώ για 128 συναντάμε 0,203 msecs. Για 32 issue σε 64 επεξεργαστές έχουμε 0,222 msecs, ενώ για 128 συναντάμε 0,196 msecs. Επίσης, για inorder σε 64 επεξεργαστές έχουμε 0,599 msecs ενώ για 128 συναντάμε 0,56 msecs.

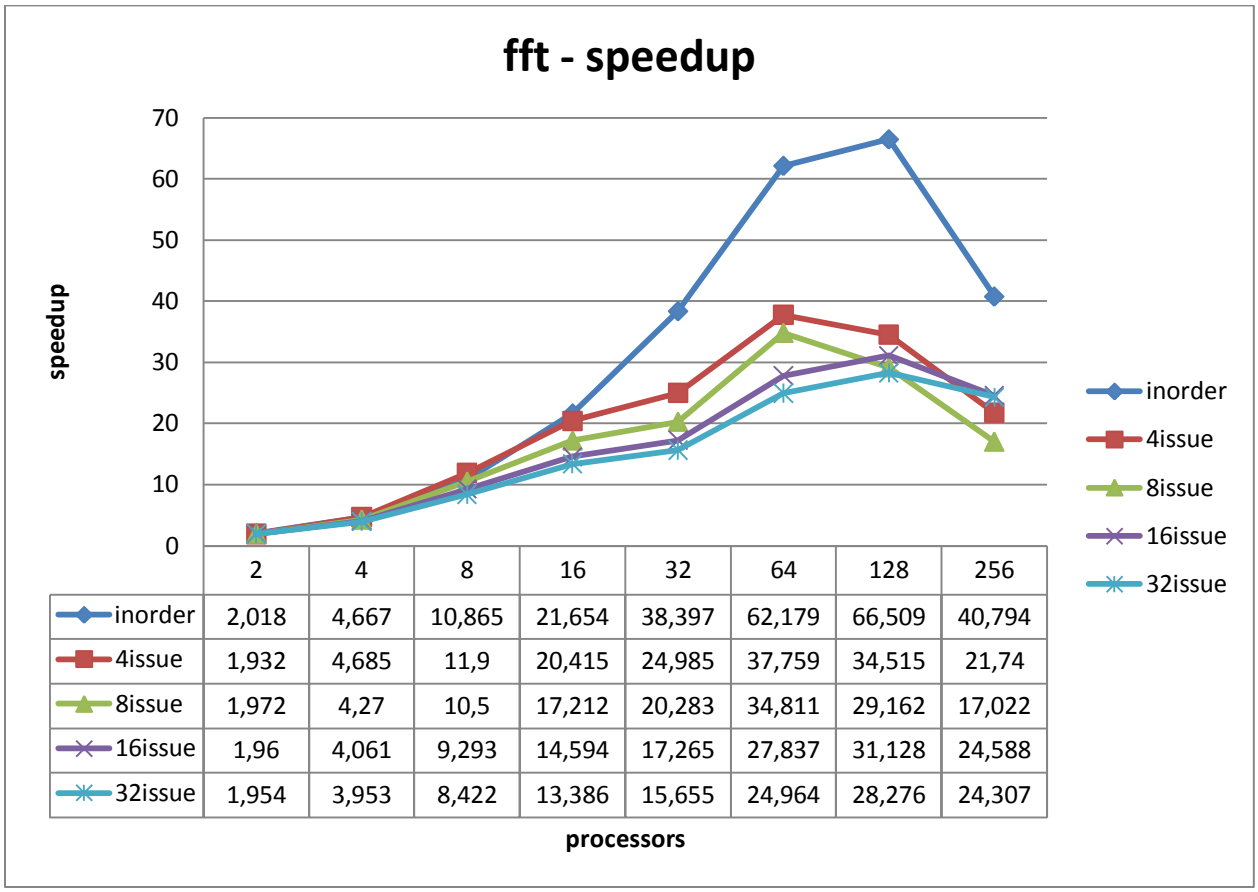
Τέλος, παρατηρούμε ότι για 256 επεξεργαστές, οι χρόνοι που πετυχαίνουμε είναι αρκετά μεγάλοι. Για 4 και 8 issue οι χρόνοι είναι χειρότεροι ακόμα από τους 32 επεξεργαστές για το αντίστοιχο issue. Συγκεκριμένα για 4 issue στους 256 επεξεργαστές έχουμε 0.462 msecs ενώ για 32 επεξεργαστές έχουμε 0,402 msecs. Για 8 issue στους 256

επεξεργαστές έχουμε 0,454 msecs ενώ για 32 συναντάμε 0,381 msecs. Για 16, 32 issue και inorder εκτέλεση οι χρόνοι είναι χειρότεροι από τους αντίστοιχους των 64 επεξεργαστών. Συγκεκριμένα, για 16 issue σε 64 επεξεργαστές έχουμε 0,227 msecs, ενώ για 256 συναντάμε 0,257 msecs. Για 32 issue σε 64 επεξεργαστές έχουμε 0,222 msecs, ενώ για 256 συναντάμε 0,228 msecs. Επίσης, για inorder σε 64 επεξεργαστές έχουμε 0,599 msecs ενώ για 256 συναντάμε 0,913 msecs. Βλέπουμε λοιπόν, ότι στους 256 επεξεργαστές έχουμε αρκετά αργούς χρόνους και αυτό οφείλεται στην αυξημένη πολυπλοκότητα που συναντάμε, με την παρουσία τόσων επεξεργαστών. Σίγουρα, με τόσους επεξεργαστές υπάρχουν επιπλέον καθυστερήσεις, που αφορούν την επικοινωνία των επεξεργαστών μεταξύ τους και το πώς αυτοί μπορούν να διαχειριστούν, τη συγκεκριμένη παράλληλη εφαρμογή.

Συνοψίζοντας, μπορούμε να πούμε ότι αυξάνοντας τους επεξεργαστές ναι μεν πετυχαίναμε καλύτερο execution time , αλλά όταν ο αριθμός των επεξεργαστών μεγάλωνε και συγκεκριμένα από τους 64 επεξεργαστές, αρχίσαμε να βλέπουμε τις πρώτες παράξενες συμπεριφορές ώσπου να φτάσουμε στους 256 όπου είχαμε χειρότερους χρόνους εν μέρει και από τους 64 επεξεργαστές. Όσον αφορά την αλλαγή του issue, πετύχαμε σταθερά καλύτερους χρόνους εκτός από τους 64 επεξεργαστές που περιγράψαμε πιο πάνω.

5.5.2 FFT-speedup

Στο παρακάτω σχήμα (*σχήμα 5.4*), βλέπουμε τη γραφική για τα 5 configurations(4 issue, 8 issue, 16 issue, 32 issue, inorder) όσον αφορά το speedup. Το speedup θα μας δείξει αναλογικά πόση "επιτάχυνση" πετυχαίνουμε συγκριτικά με το χρόνο προσομοίωσης για ένα επεξεργαστή. Ο τύπος που χρησιμοποιήθηκε για τον υπολογισμό του speedup είναι: ***Sp = T1 / Tp*** όπου το p είναι ο αριθμός των επεξεργαστών, το T1 είναι το *execution time* του σειριακού αλγορίθμου και το Tp είναι το execution time του παράλληλου αλγορίθμου με p επεξεργαστές.



	2	4	8	16	32	64	128	256
inorder	2,018	4,667	10,865	21,654	38,397	62,179	66,509	40,794
4issue	1,932	4,685	11,9	20,415	24,985	37,759	34,515	21,74
8issue	1,972	4,27	10,5	17,212	20,283	34,811	29,162	17,022
16issue	1,96	4,061	9,293	14,594	17,265	27,837	31,128	24,588
32issue	1,954	3,953	8,422	13,386	15,655	24,964	28,276	24,307

Σχήμα 5.4: *Γραφική που αναπαριστά το fft speedup για μέχρι και 256 επεξεργαστές*

Από την παραπάνω γραφική, παρατηρούμε αρχικά, ότι το μεγαλύτερο speedup το πετυχαίνουμε για την εκτέλεση των inorder εντολών. Βλέπουμε ότι το speedup σε inorder, φτάνει μέχρι και 66,509 για 128 επεξεργαστές, όταν το αμέσως επόμενο speedup για out of order επεξεργαστές είναι 37,759 για 64 επεξεργαστές σε 4 issue.

Για την inorder εκτέλεση, παρατηρούμε ότι έχουμε ραγδαία αύξηση μέχρι τους 64 επεξεργαστές. Στους 128 επεξεργαστές, συναντάμε πάλι αύξηση του speedup σε σχέση με τους 64, χωρίς όμως μεγάλες διαφορές που θα μπορούσαν να δικαιολογήσουν την παρουσία τόσων επεξεργαστών. Συγκεκριμένα στους 64 έχουμε speedup 62,179 ενώ στους 128 έχουμε speedup 66,509. Στους 256 επεξεργαστές το speedup πέφτει απότομα στο 40,794 λίγο μεγαλύτερο δηλαδή από το speedup στους 32 επεξεργαστές που ήταν

38,387. Προφανώς, όπως αναφέραμε και παραπάνω η παρουσία τόσων επεξεργαστών δε βοηθά στη γρηγορότερη εκτέλεση της εφαρμογής, αλλά αντίθετα την επιβαρύνει.

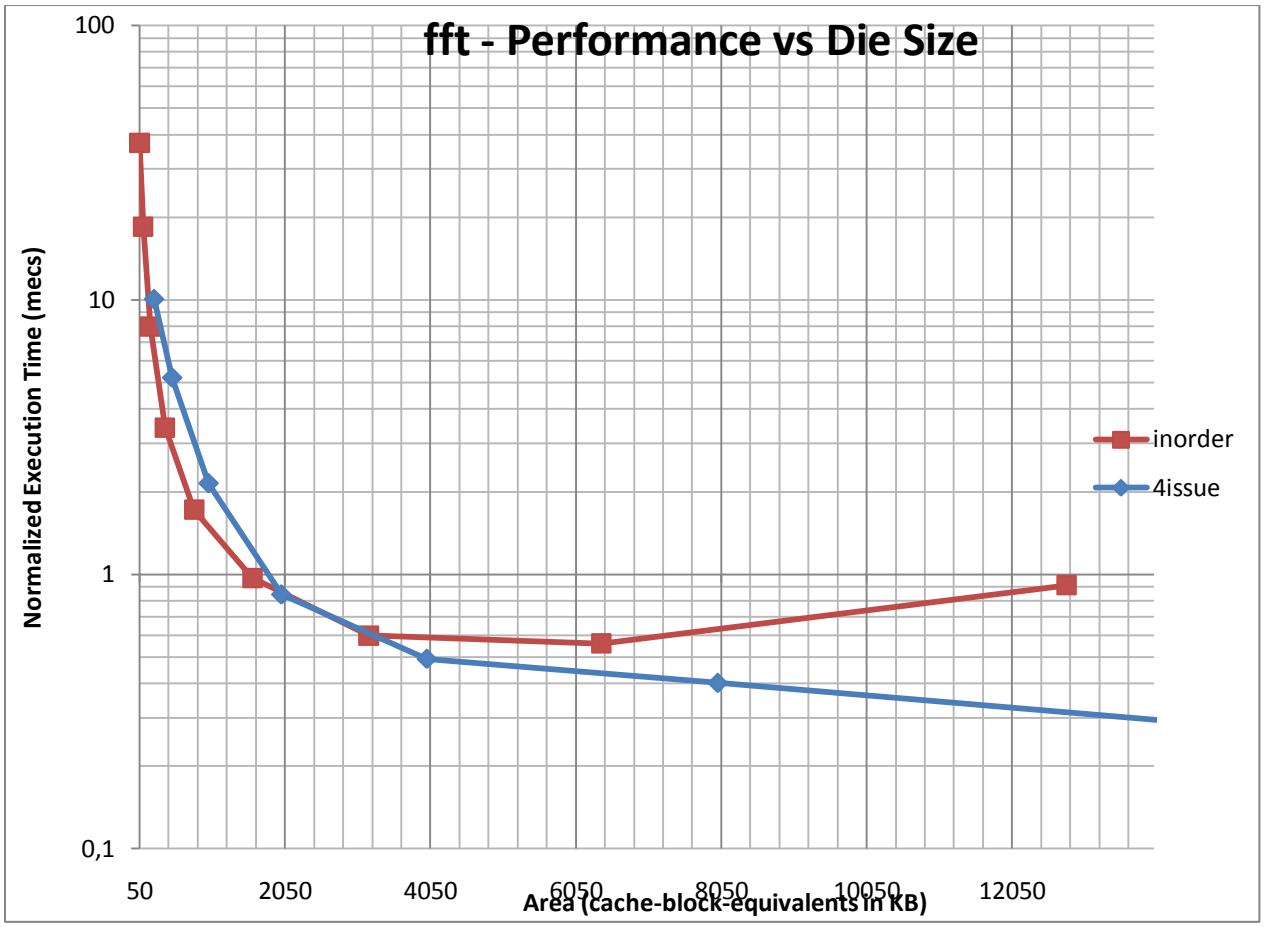
Όσον αφορά την out-of-order εκτέλεση, το μεγαλύτερο speedup το συναντάμε στο 4 issue εκτός από τους 2 και τους 256 επεξεργαστές. Το μεγαλύτερο speedup, συγκριτικά με τα άλλα issue, είναι στους 64 επεξεργαστές. Στο 4issue των 64 επεξεργαστών έχουμε speedup 37,759, τη στιγμή που για τους ίδιους επεξεργαστές στο 32 issue, έχουμε 24,964 δηλαδή υπάρχει μια διαφορά 12,795. Συγκριτικά, μπορούμε να δούμε ότι όσο μικρότερο είναι το issue, το speedup είναι αναλογικά μεγαλύτερο.

Μπορούμε να δούμε ότι το speedup είναι πολύ ικανοποιητικό, μέχρι και τους 16 επεξεργαστές όπου έχουμε για 4 issue speedup 20,415, για 8 issue speedup 17,212, για 16 issue speedup 14,594, για 32 issue 13,386 και για inorder 21,654. Από εκεί και έπειτα το speedup, ναι μεν είναι καλό, αλλά δεν είναι ανάλογο της αύξησης του αριθμού των επεξεργαστών. Στους 64 επεξεργαστές, βλέπουμε ότι το speedup είναι πολύ μικρό σε σχέση με τους επεξεργαστές μας. Για να ήμασταν ικανοποιημένοι θα θέλαμε να δούμε ένα speedup κοντά στο 64 και όχι για παράδειγμα ένα 37,759 που συναντάμε στο 4 issue. Το ίδιο ισχύει και για τους 128 και τους 256 επεξεργαστές όπου το speedup μας είναι επίσης χαμηλό συγκριτικά με τους επεξεργαστές μας. Μόνο στην inorder εκτέλεση βλέπουμε ότι speedup μας, τουλάχιστον μέχρι τους 64 επεξεργαστές είναι ικανοποιητικό αφού έχουμε 62,179.

Συνοψίζοντας, μπορούμε να πούμε, ότι το speedup μας είναι ανάλογο των επεξεργαστών μας, μέχρι και τους 16 επεξεργαστές και ότι από τους 64 και μετά ναι μεν το speedup είναι μεγαλύτερο από τα προηγούμενα, αλλά δεν είναι ανάλογο των επεξεργαστών μας. Στους 256 επεξεργαστές, το speedup είναι απογοητευτικό και μπορούμε να πούμε ότι μεταξύ 64 και 128 επεξεργαστών είναι στα όρια του μέγιστου βαθμού παραλληλισμού της.

5.5.3 FFT-Υπολογισμός χώρου του chip

Στο παρακάτω σχήμα (*σχήμα 5.5*), θα προσπαθήσουμε να συγκρίνουμε την αποδοτικότητα των inorder και των out-of-order επεξεργαστών με βάση το χώρο του chip. Αυτό θα γίνει σύμφωνα με το cache-byte-equivalents (CBE) [15], [16], με τον inorder επεξεργαστή να έχει area 50 KB CBE και τον out-of-order να έχει area 250 KB CBE.



Σχήμα 5.5: *Γραφική που αναπαριστά το chip area για μέχρι και 256 επεξεργαστές.*

Από την παραπάνω γραφική, μπορούμε να συγκρίνουμε τους inorder με τους out-of-order επεξεργαστές. Αρχικά, παρατηρούμε ότι στους inorder επεξεργαστές η διαφορά στο execution time είναι εμφανής. Για το δεδομένο die area, μπορούμε να δούμε ότι χρησιμοποιώντας 4 inorder επεξεργαστές πετυχαίνουμε καλύτερο execution time απότι χρησιμοποιώντας 1 out-of-order επεξεργαστή. Αυτό σε χώρο μεταφράζεται σε 250 KB

CBE για τον out-of-order και 200 KB CBE σε inorder, άρα έχουμε μια εξοικόνομηση του χώρου 20%.

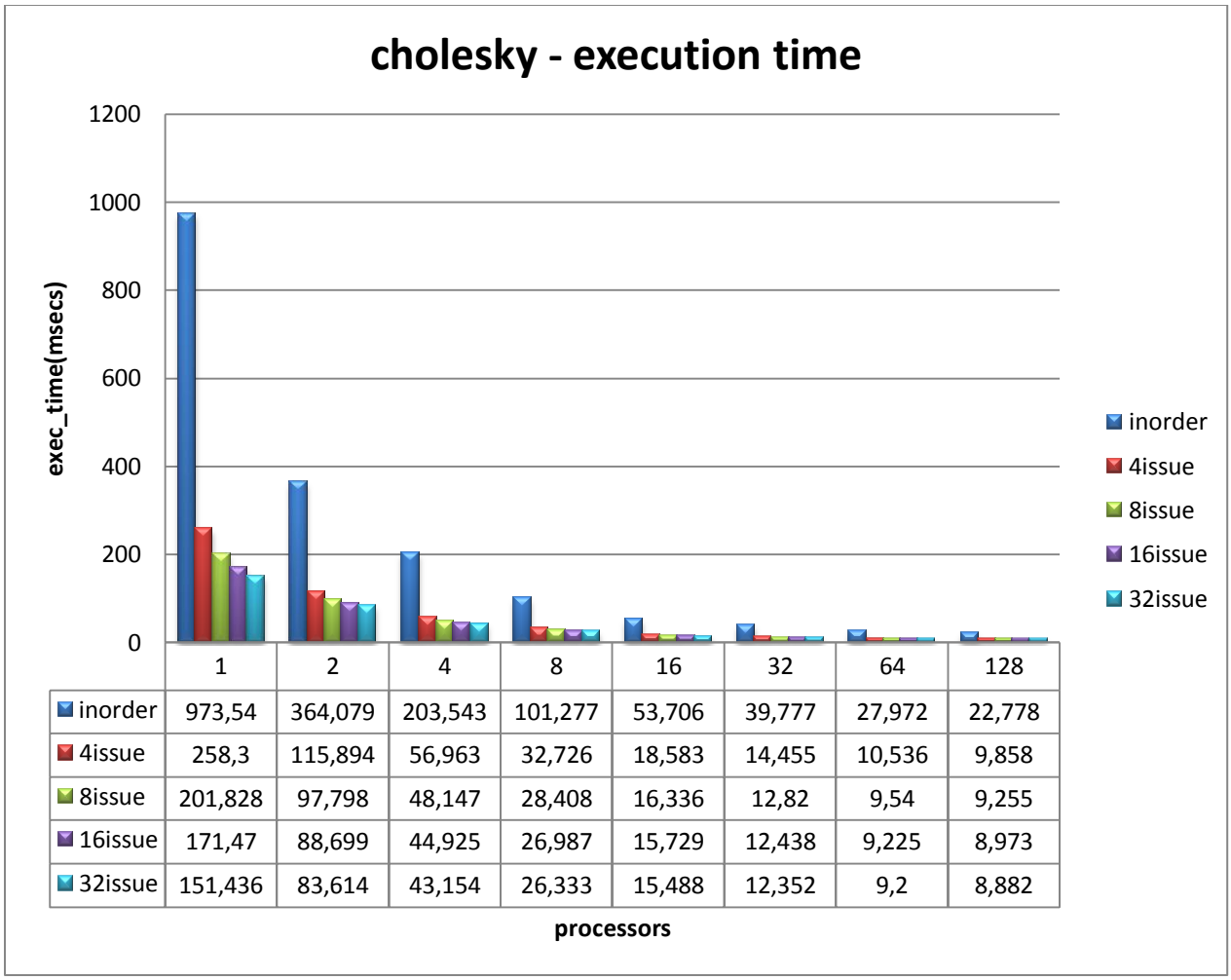
Βλέποντας πιο προσεκτικά τη γραφική, μπορούμε να διακρίνουμε τα κομμάτια τα οποία κάνουν πιο αποδοτικούς, τους inorder επεξεργαστές. Συγκεκριμένα βλέπουμε, ότι από τα 50 KB CBE έως τα 2050 KB CBE, πιο αποδοτικοί είναι οι inorder επεξεργαστές. Από τα 2050 KB CBE και μετά, οι out-of-order επεξεργαστές είναι εμφανώς καλύτεροι, αφού το execution time, μικραίνει αρκετά, εν συγκρίσει με τους inorder επεξεργαστές.

5.6 Ανάλυση αποτελεσμάτων για το CHOLESKY benchmark

Το cholesky kernel [14] είναι ένα benchmark που είναι παρόμοιο στη δομή και στο partitioning με το kernel LU factorization, αλλά έχει δύο μεγάλες διαφορές: i) λειτουργεί σε αραιά πλέγματα, που έχουν μεγαλύτερη επικοινωνία στον υπολογισμό του ratio, για συγκρίσιμα μεγέθη προβλήματος και ii) δεν είναι γενικά συγχρονισμένο μεταξύ των βημάτων.

5.6.1 CHOLESKY-execution time

Στο παρακάτω σχήμα (*σχήμα 5.6*), βλέπουμε τη γραφική για τα 5 configurations(4 issue, 8 issue, 16 issue, 32 issue, inorder) όσον αφορά το execution time .



	1	2	4	8	16	32	64	128
inorder	973,54	364,079	203,543	101,277	53,706	39,777	27,972	22,778
4issue	258,3	115,894	56,963	32,726	18,583	14,455	10,536	9,858
8issue	201,828	97,798	48,147	28,408	16,336	12,82	9,54	9,255
16issue	171,47	88,699	44,925	26,987	15,729	12,438	9,225	8,973
32issue	151,436	83,614	43,154	26,333	15,488	12,352	9,2	8,882

Σχήμα 5.6: *Γραφική που αναπαριστά το cholesky execution time για μέχρι και 256 επεξεργαστές*

Από την παραπάνω γραφική παρατηρούμε αρχικά, ότι για την inorder εκτέλεση εντολών, έχουμε πολύ μεγάλες τιμές, συγκριτικά με τις υπόλοιπες. Αυτό, είναι κάτι που παρατηρήσαμε και στα παραπάνω benchmarks και όπως επισημάναμε οφείλεται, στο ότι μια εφαρμογή χρειάζεται περισσότερο χρόνο, όταν είναι αναγκασμένη να εκτελεί τις εντολές της με τη σειρά.

Για ένα επεξεργαστή στην inorder εκτέλεση, παρατηρούμε έναν χρόνο της τάξεως του 973,54 msecs ενώ για την αντίστοιχη 4issue εκτέλεση ένα χρόνο του 258,3 msecs. Βλέπουμε επίσης, ότι αυξάνοντας το issue, πετυχαίνουμε όλο και καλύτερο χρόνο

και καταφέρνουμε να φτάσουμε για ένα επεξεργαστή στην τιμή του 151,436 msecs με 32 issue.

Αυτή τη βελτίωση των χρόνων, τη συναντάμε και αυξάνοντας τους επεξεργαστές. Παρατηρούμε ότι αυξάνοντας τον αριθμό των επεξεργαστών έχουμε μείωση στους χρόνους και για την inorder εκτέλεση και για το μεταβλητό issue. Η βελτίωση στους χρόνους είναι συνεχής μέχρι και τους 128 επεξεργαστές όπου ο χρόνος είναι 8,882 msecs για 32 issue.

Από τους 32 επεξεργαστές και μετά, παρατηρούμε ότι ναι μεν έχουμε βελτίωση των χρόνων, αλλά αυτή η βελτίωση δεν είναι τόσο αισθητή αν αναλογιστούμε τον αριθμό των επεξεργαστών μας. Βλέπουμε όμως ότι αλλάζοντας το issue, πετυχαίνουμε αρκετά καλύτερους χρόνους. Συγκεκριμένα, για ένα επεξεργαστή το execution time στο 4 issue είναι 258,3 msecs, ενώ για 32 issue είναι 151,436 msecs. Στους 2 επεξεργαστές για 4 issue είναι 115,894 msecs, ενώ για 32 issue είναι 83,614 msecs. Στους 4 επεξεργαστές για 4 issue είναι 56,963 msecs, ενώ για 32 issue είναι 43,154 msecs. Στους 8 επεξεργαστές για 4 issue είναι 32,726 msecs, ενώ για 32 issue είναι 26,333 msecs. Από τους 16 επεξεργαστές και μετά πάλι έχουμε βελτίωση των χρόνων αλλάζοντας το issue, όμως αυτές οι διαφορές δεν είναι και τόσο μεγάλες.

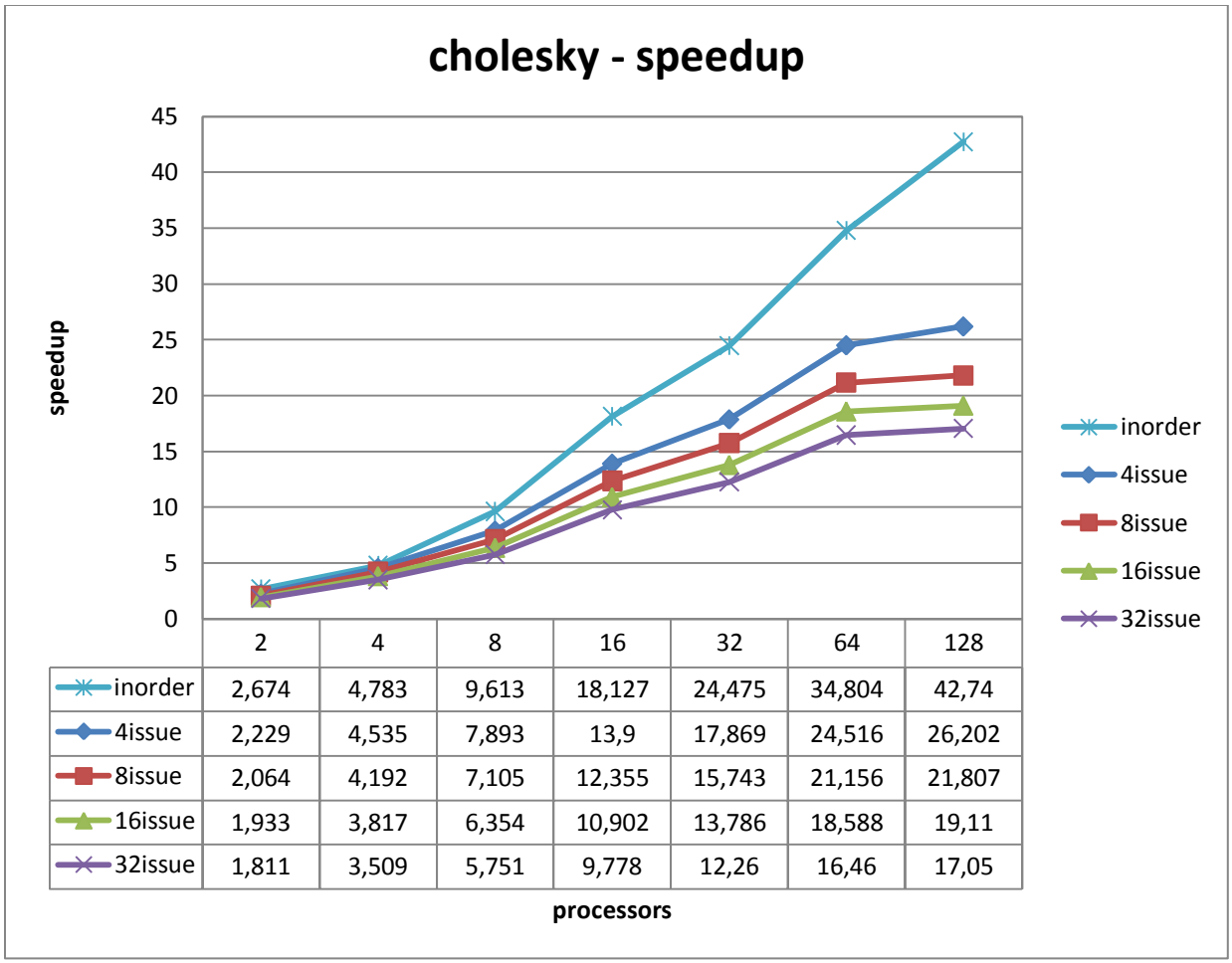
Στους inorder επεξεργαστές, έχουμε και εκεί βελτίωση των χρόνων, όμως οι χρόνοι που εκτελείται το benchmark είναι πολύ μεγαλύτεροι. Για ένα επεξεργαστή στην inorder εκτέλεση, το execution time είναι 973,54 msecs και για 128 επεξεργαστές το execution time είναι 22,778 msecs. Με την αλλαγή των επεξεργαστών, υπάρχει κάθε φορά μια αρκετά μεγάλη μείωση στους χρόνους. Μόνο από τους 64 επεξεργαστές στους 128 η διαφορά αυτή δεν είναι και τόσο μεγάλη, αφού στους 64 επεξεργαστές το execution time είναι 27,972 msecs και στους 128 το execution time είναι 22,778 msecs.

Συνοψίζοντας, μπορούμε να πούμε ότι αυξάνοντας τους επεξεργαστές πετύχαμε καλύτερο execution time , αλλά από τους 32 επεξεργαστές και μετά οι διαφορές δεν είναι

και πολύ μεγάλες, παρά μόνο στην inorder εκτέλεση. Όσον αφορά την αλλαγή του issue, πετύχαμε σταθερά καλύτερους χρόνους σε όλα τα στάδια της προσομοίωσής μας.

5.6.2 CHOLESKY-speedup

Στο παρακάτω σχήμα (*σχήμα 5.7*), βλέπουμε τη γραφική για τα 5 configurations(4 issue, 8 issue, 16 issue, 32 issue, inorder) όσον αφορά το speedup. Το speedup θα μας δείξει αναλογικά πόση "επιτάχυνση" πετυχαίνουμε συγκριτικά με το χρόνο προσομοίωσης για ένα επεξεργαστή. Ο τύπος που χρησιμοποιήθηκε για τον υπολογισμό του speedup είναι: ***Sp = T1 / Tp*** όπου το p είναι ο αριθμός των επεξεργαστών, το T1 είναι το *execution time* του σειριακού αλγορίθμου και το Tp είναι το execution time του παράλληλου αλγορίθμου με p επεξεργαστές.



	2	4	8	16	32	64	128
inorder	2,674	4,783	9,613	18,127	24,475	34,804	42,74
4issue	2,229	4,535	7,893	13,9	17,869	24,516	26,202
8issue	2,064	4,192	7,105	12,355	15,743	21,156	21,807
16issue	1,933	3,817	6,354	10,902	13,786	18,588	19,11
32issue	1,811	3,509	5,751	9,778	12,26	16,46	17,05

Σχήμα 5.7: *Γραφική που αναπαριστά το cholesky speedup για μέχρι και 256 επεξεργαστές*

Από την παραπάνω γραφική, παρατηρούμε αρχικά, ότι το μεγαλύτερο speedup, όπως και στα προηγούμενα το πετυχαίνουμε για την εκτέλεση των inorder εντολών. Βλέπουμε ότι το speedup σε inorder, φτάνει μέχρι και 42,74 για 128 επεξεργαστές, όταν το αμέσως επόμενο speedup για out of order επεξεργαστές είναι 26,202 για 128 επεξεργαστές σε 32 issue.

Για την inorder εκτέλεση, παρατηρούμε ότι έχουμε αύξηση μέχρι τους 128 επεξεργαστές. Η αύξηση που παρατηρούμε είναι μεγάλη και μέχρι τους 16 επεξεργαστές το speedup είναι ανάλογο του αριθμού των επεξεργαστών. Από τους 32 επεξεργαστές και μετά έχουμε μεν αύξηση, αλλά αυτή η αύξηση δεν είναι ανάλογη του αριθμού των

επεξεργαστών. Για 32 επεξεργαστές, το speedup είναι 24,475, για 64 επεξεργαστές το speedup είναι 34,804 και για 128 επεξεργαστές είναι 42,74.

Όσον αφορά την out-of-order εκτέλεση, το μεγαλύτερο speedup το συναντάμε στο 4 issue. Το μεγαλύτερο speedup, συγκριτικά με τα άλλα issue, είναι στους 128 επεξεργαστές. Στο 4issue των 128 επεξεργαστών έχουμε speedup 26,202 το οποίο όμως είναι πολύ μικρό αν αναλογιστούμε τους 128 επεξεργαστές. Συγκριτικά, μπορούμε να δούμε ότι όσο μικρότερο είναι το issue, το speedup είναι αναλογικά μεγαλύτερο.

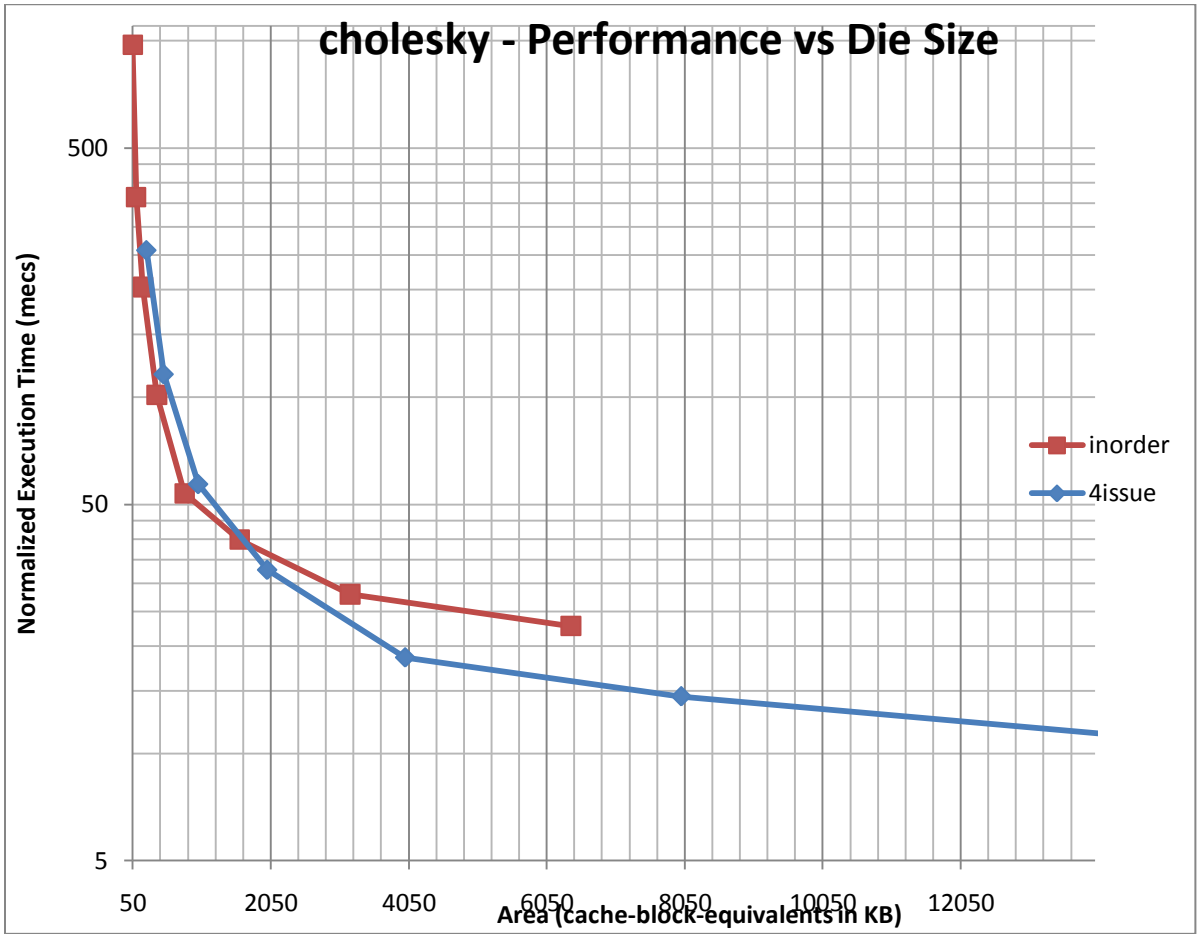
Παρατηρώντας, τη γραφική, μπορούμε να δούμε ότι το speedup που πετυχαίνουμε δεν είναι καθόλου καλό. Ναι μεν, έχουμε συνεχής αύξηση, αλλά μετά τους 16 επεξεργαστές το speedup, δε δικαιολογεί τον αριθμό των επεξεργαστών. Για 4 issue στους 32 επεξεργαστές το speedup είναι 17,869, για 64 επεξεργαστές το speedup είναι 24,516 και για 128 επεξεργαστές το speedup είναι 26,202. Στους 128 επεξεργαστές το speedup που πετυχαίνουμε και στην inorder και στην out-of-order εκτέλεση είναι πολύ μικρό συγκριτικά με τον αριθμό των επεξεργαστών. Φαίνεται ότι το συγκεκριμένο benchmark ο βαθμός παραλληλισμού του περιορίζεται στους 16 επεξεργαστές, αφού από τους 32 επεξεργαστές και μετά δεν έχουμε το ανάλογο speedup.

Συνοψίζοντας, μπορούμε να πούμε, ότι το speedup παρουσιάζει μια συνεχή αύξηση μέχρι τους 128 επεξεργαστές, αλλά δεν είναι ανάλογο του αριθμού των επεξεργαστών μας, όσον αφορά τους 32 επεξεργαστές και μετά. Στην inorder εκτέλεση το speedup που παρατηρούμε είναι σε καλύτερα επίπεδα από την out-of-order εκτέλεση, αλλά δεν παύει και αυτό να υστερεί συγκριτικά με τον αριθμό των επεξεργαστών.

5.6.3 CHOLESKY-Υπολογισμός χώρου του chip

Στο παρακάτω σχήμα (*σχήμα 5.8*), θα προσπαθήσουμε να συγκρίνουμε την αποδοτικότητα των inorder και των out-of-order επεξεργαστών με βάση το χώρο του

chip. Αυτό θα γίνει σύμφωνα με το cache-byte-equivalents (CBE) [15], [16], με τον inorder επεξεργαστή να έχει area 50 KB CBE και τον out-of-order να έχει area 250 KB CBE.



Σχήμα 5.8: *Γραφική που αναπαριστά το chip area για μέχρι και 128 επεξεργαστές*

Από την παραπάνω γραφική, παρατηρούμε και εδώ ότι η διαφορά στο execution time, ανάμεσα στους out-of-order και στους inorder επεξεργαστές, είναι εμφανής. Για το δεδομένο die area, συναντάμε ότι ακριβώς και στο fft, δηλαδή ότι χρησιμοποιώντας 4 inorder επεξεργαστές πετυχαίνουμε καλύτερο execution time απότι χρησιμοποιώντας 1 out-of-order επεξεργαστή. Άρα και εδώ έχουμε μια εξοικονόμηση του χώρου της τάξεως του 20%. Αυτή η αναλογία συνεχίζεται και με την περεταίρω αύξηση των επεξεργαστών.

Σε αυτή τη γραφική, βλέπουμε ότι οι inorder επεξεργαστές είναι πιο αποδοτικοί από τα 50 KB CBE έως τα 800 KB CBE. Από τα 800 KB CBE και μετά βλέπουμε ότι οι

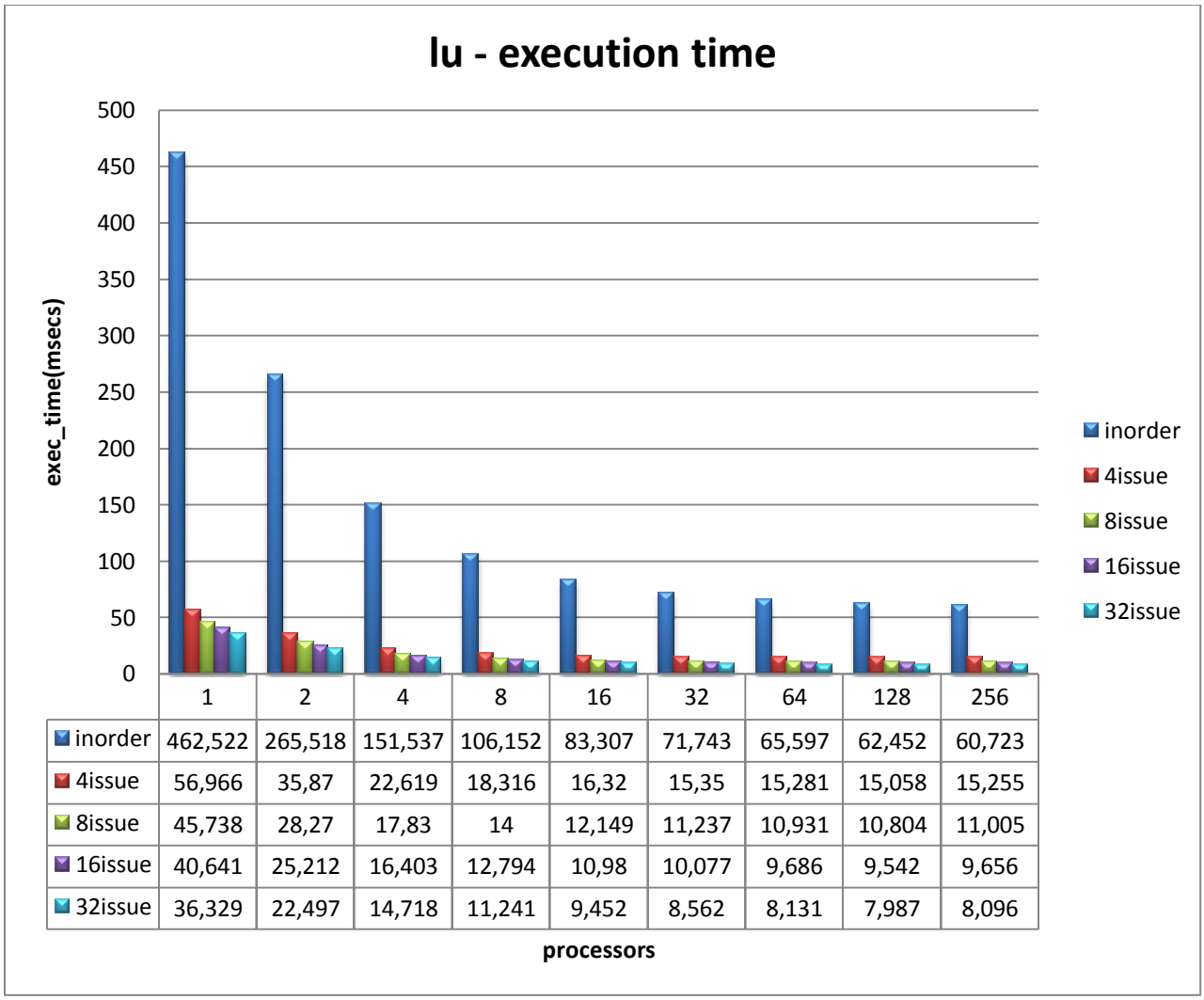
out-of-order επεξεργαστές είναι πιο αποδοτικοί και βλέπουμε ότι το execution time έχει καθοδική πορεία και φτάνει σε επίπεδα που οι inorder επεξεργαστές δεν μπορούν να φτάσουν.

5.7 Ανάλυση αποτελεσμάτων για το LU benchmark

Το LU kernel [14] παραγοντοποιεί ένα πυκνό πλέγμα στο γινόμενο ενός χαμηλότερου τριγωνικού και ενός υψηλότερου τριγωνικού πλέγματος. Το πυκνό nxn πλέγμα A, διαιρείται σε έναν NxN πίνακα, από BxB blocks (n=NB), ώστε να εκμεταλλευτεί τη χρονική τοποθεσία των submatrix στοιχείων. Για να μειωθεί η επικοινωνία, το block ownership ορίζεται να χρησιμοποιεί ένα 2-D scatter αποσύνθεσης, με τα blocks να ενημερώνονται από τους επεξεργαστές στους οποίους ανατίθενται. Το μέγεθος του block B πρέπει να είναι αρκετά μεγάλο, ώστε να κρατήσει χαμηλό το cache miss rate και ταυτόχρονα αρκετά μικρό ώστε να μπορεί διατηρήσει την καλή ισορροπία των φορτίων. Τα αρκετά μικρά μεγέθη του block, (B=8 ή B=16) αποτελούν στην πράξη, μια πολύ καλή λύση. Τα στοιχεία μέσα σε ένα block, δεσμεύονται συνεχόμενα, για να βελτιώσουν τα πλεονεκτήματα του spatial locality και τα blocks δεσμεύονται τοπικά στους επεξεργαστές στους οποίους ανατίθενται.

5.7.1 LU-execution time

Στο παρακάτω σχήμα (*σχήμα 5.9*), βλέπουμε τη γραφική για τα 5 configurations(4 issue, 8 issue, 16 issue, 32 issue, inorder) όσον αφορά το execution time .



	1	2	4	8	16	32	64	128	256
inorder	462,522	265,518	151,537	106,152	83,307	71,743	65,597	62,452	60,723
4issue	56,966	35,87	22,619	18,316	16,32	15,35	15,281	15,058	15,255
8issue	45,738	28,27	17,83	14	12,149	11,237	10,931	10,804	11,005
16issue	40,641	25,212	16,403	12,794	10,98	10,077	9,686	9,542	9,656
32issue	36,329	22,497	14,718	11,241	9,452	8,562	8,131	7,987	8,096

Σχήμα 5.9: *Γραφική που αναπαριστά το lu execution time για μέχρι και 256 επεξεργαστές*

Από την παραπάνω γραφική παρατηρούμε αρχικά, ότι για την inorder εκτέλεση εντολών, έχουμε πολύ μεγάλες τιμές, συγκριτικά με τις υπόλοιπες. Αυτό, είναι κάτι που παρατηρήσαμε και στα παραπάνω benchmarks και όπως επισημάναμε οφείλεται, στο ότι μια εφαρμογή χρειάζεται περισσότερο χρόνο, όταν είναι αναγκασμένη να εκτελεί τις εντολές της με τη σειρά. Αυτό έχει σαν αποτέλεσμα να μη χρησιμοποιείται αποδοτικά το pipelining και να υπάρχουν καθυστερήσεις λόγω των πολλών εξαρτήσεων.

Για ένα επεξεργαστή στην inorder εκτέλεση, παρατηρούμε έναν χρόνο της τάξεως του 462,522 msecs ενώ για την αντίστοιχη 4issue εκτέλεση ένα χρόνο του 56,966 msecs. Εδώ η διαφορά παρατηρούμε, ότι είναι ακόμα μεγαλύτερη από τα προηγούμενα

benchmarks και είναι ακόμα πιο σαφής, η σημαντικότητα της out-of-order εκτέλεσης. Βλέπουμε επίσης, ότι αυξάνοντας το issue, πετυχαίνουμε όλο και καλύτερο χρόνο και καταφέρνουμε να φτάσουμε για ένα επεξεργαστή στην τιμή του 36,329 msecs με 32 issue.

Αυτή τη βελτίωση των χρόνων, τη συναντάμε και αυξάνοντας τους επεξεργαστές. Παρατηρούμε ότι αυξάνοντας τον αριθμό των επεξεργαστών έχουμε μείωση στους χρόνους και για την inorder εκτέλεση και για το μεταβλητό issue. Η βελτίωση στους χρόνους είναι αισθητή μέχρι και τους 128 επεξεργαστές όπου ο χρόνος είναι 7,987 msecs για 32 issue. Μέχρι τους 128 επεξεργαστές λοιπόν, η βελτίωση είναι συνεχής και σε ότι έχει να κάνει με την αύξηση του issue και σε ότι έχει να κάνει με την αύξηση των επεξεργαστών.

Για 256 επεξεργαστές παρατηρούμε ότι για 8 issue, οι χρόνοι είναι χειρότεροι σε σχέση με τους αντίστοιχους για 64 επεξεργαστές. Συγκεκριμένα, για 8 issue σε 64 επεξεργαστές έχουμε 10,931 msecs, ενώ για 256 συναντάμε 11,005 msecs. Για 4, 16 και 32 issue, οι χρόνοι είναι χειρότεροι σε σχέση με τους αντίστοιχους για 128 επεξεργαστές. Συγκεκριμένα, για 4 issue σε 128 επεξεργαστές έχουμε 15,058 msecs, ενώ για 256 συναντάμε 15,255 msecs. Για 16 issue σε 128 επεξεργαστές έχουμε 9,542 msecs, ενώ για 256 συναντάμε 9,656 msecs. Τέλος για 32 issue σε 128 επεξεργαστές έχουμε 7,987 msecs, ενώ για 256 συναντάμε 8,096 msecs. Αντίθετα, στην inorder εκτέλεση έχουμε συνεχής βελτίωση ακόμα και από τους 128 στους 256 επεξεργαστές. Συγκεκριμένα, για inorder σε 128 επεξεργαστές έχουμε 62,452 msecs, ενώ για 256 συναντάμε 60,723 msecs.

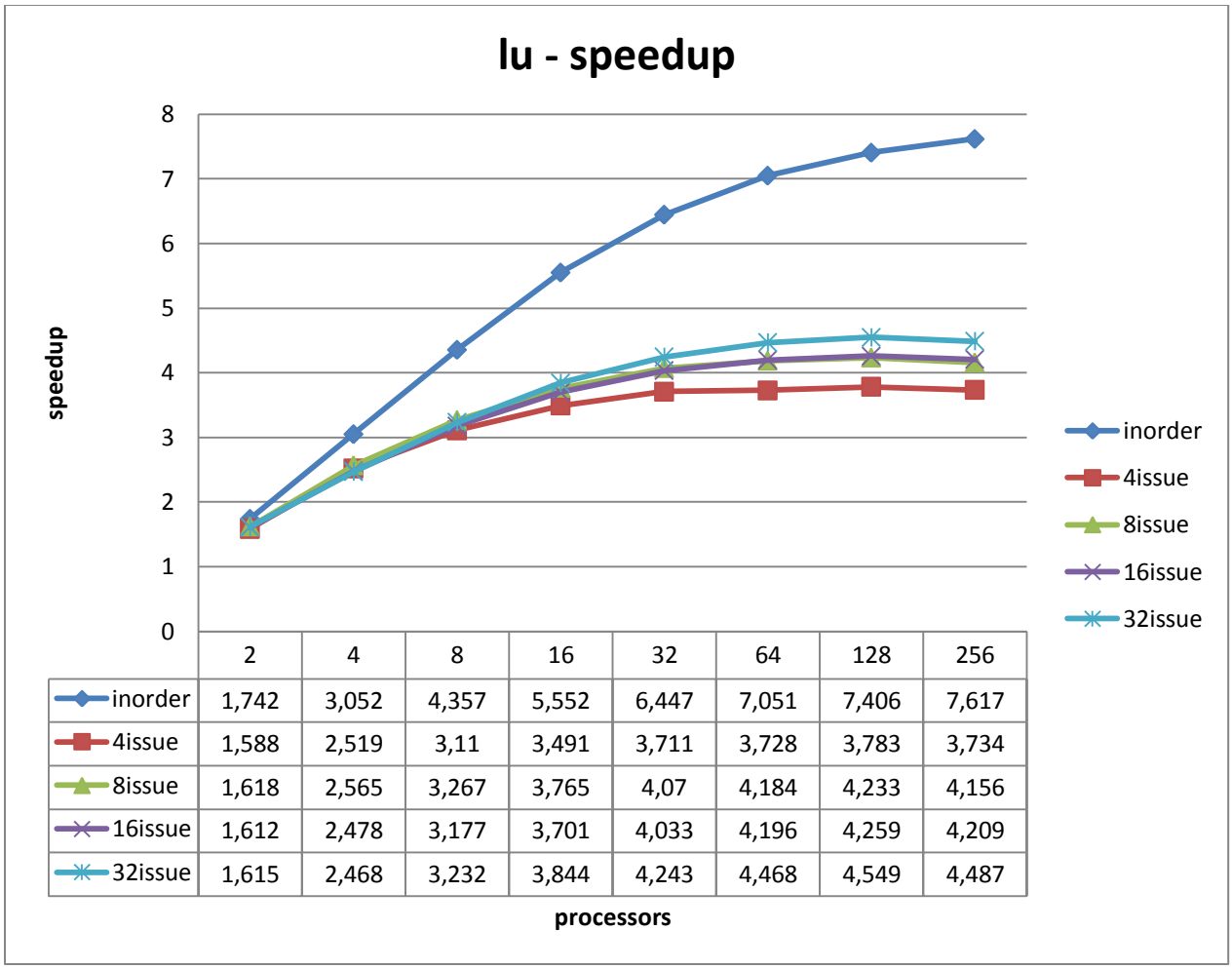
Όπως και να χει, αυξάνοντας τους επεξεργαστές, οι διαφορές δεν είναι πολύ μεγάλες με τους χρόνους των λιγότερων επεξεργαστών. Ναι μεν έχουμε βελτίωση, αλλά αυτή η βελτίωση δεν είναι τόσο αισθητή ώστε να δικαιολογεί το νούμερο των επεξεργαστών.

Συνοψίζοντας, μπορούμε να πούμε ότι αυξάνοντας τους επεξεργαστές πετυχαίναμε καλύτερο execution time , αλλά όταν ο αριθμός των επεξεργαστών έφτασε

στους 256 επεξεργαστές είδαμε κάποιους χειρότερους χρόνους. Όσον αφορά την αλλαγή του issue, πετύχαμε σταθερά καλύτερους χρόνους σε όλα τα στάδια της προσομοίωσής μας.

5.7.2 LU-speedup

Στο παρακάτω σχήμα (*σχήμα 5.10*), βλέπουμε τη γραφική για τα 5 configurations(4 issue, 8 issue, 16 issue, 32 issue, inorder) όσον αφορά το speedup. Το speedup θα μας δείξει αναλογικά πόση "επιτάχυνση" πετυχαίνουμε συγκριτικά με το χρόνο προσομοίωσης για ένα επεξεργαστή. Ο τύπος που χρησιμοποιήθηκε για τον υπολογισμό του speedup είναι: ***Sp = T1 / Tp*** όπου το p είναι ο αριθμός των επεξεργαστών, το T1 είναι το *execution time* του σειριακού αλγορίθμου και το Tp είναι το execution time του παράλληλου αλγορίθμου με p επεξεργαστές.



	2	4	8	16	32	64	128	256
inorder	1,742	3,052	4,357	5,552	6,447	7,051	7,406	7,617
4issue	1,588	2,519	3,11	3,491	3,711	3,728	3,783	3,734
8issue	1,618	2,565	3,267	3,765	4,07	4,184	4,233	4,156
16issue	1,612	2,478	3,177	3,701	4,033	4,196	4,259	4,209
32issue	1,615	2,468	3,232	3,844	4,243	4,468	4,549	4,487

Σχήμα 5.10: *Γραφική που αναπαριστά το lu speedup για μέχρι και 256 επεξεργαστές*

Από την παραπάνω γραφική, παρατηρούμε αρχικά, ότι το μεγαλύτερο speedup, όπως και στα προηγούμενα το πετυχαίνουμε για την εκτέλεση των inorder εντολών. Βλέπουμε ότι το speedup σε inorder, φτάνει μέχρι και 7,617 για 128 επεξεργαστές, όταν το αμέσως επόμενο speedup για out of order επεξεργαστές είναι 4,569 για 128 επεξεργαστές σε 32 issue.

Για την inorder εκτέλεση, παρατηρούμε ότι έχουμε αύξηση μέχρι τους 256 επεξεργαστές. Βεβαίως η αύξηση που παρατηρούμε δεν είναι μεγάλη, αλλά δεν παύει να ισχύει. Από τους 32 επεξεργαστές και μετά η αύξηση είναι ολο και μικρότερη. Ενώ για 2 επεξεργαστές είχαμε ένα speedup 1,742 και για 4 είχαμε 3,052, από τους 32 στους 64 η διαφορά είναι πολύ μικρή, αφού για 32 είχαμε speedup 6,447 και για 64 είχαμε 7,051.

Αυτή τη μικρή αύξηση, τη συναντάμε και στις υπόλοιπες προσομοιώσεις για 128 και 256 επεξεργαστές.

Όσον αφορά την out-of-order εκτέλεση, το μεγαλύτερο speedup το συναντάμε στο 32 issue. Το μεγαλύτερο speedup, συγκριτικά με τα άλλα issue, είναι στους 128 επεξεργαστές. Στο 4issue των 128 επεξεργαστών έχουμε speedup 3,783, τη στιγμή που για τους ίδιους επεξεργαστές στο 32 issue, έχουμε 4,549. Συγκριτικά, μπορούμε να δούμε ότι όσο μικρότερο είναι το issue, το speedup είναι αναλογικά μεγαλύτερο.

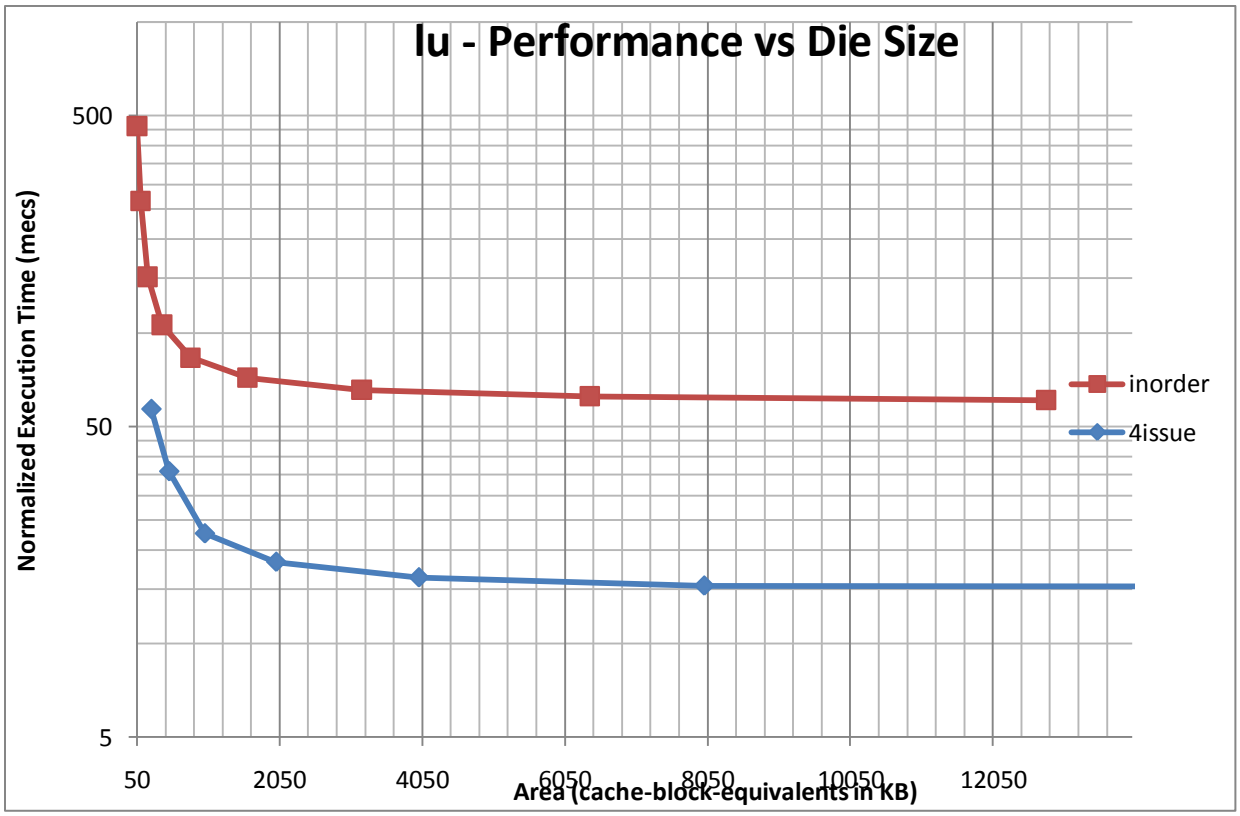
Παρατηρώντας, τη γραφική, μπορούμε να δούμε ότι το speedup που πετυχαίνουμε δεν είναι καθόλου καλό. Ναι μεν, έχουμε συνεχής αύξηση, αλλά μετά τους 4 επεξεργαστές το speedup, δε δικαιολογεί τον αριθμό των επεξεργαστών. Ακόμα και στους 4 επεξεργαστές το speedup, που βλέπουμε δεν είναι πολύ καλό αφού το καλύτερο speedup που πετυχαίνουμε στην out-of-order εκτέλεση είναι 2,478 για 16 issue και για την inorder 3,052. Το μέγιστο speedup του 7,617 για την inorder και του 4,549 για την out-of-order, είναι ελάχιστο συγκριτικά με τους 256 και 128 επεξεργαστές αντίστοιχα. Φαίνεται ότι το συγκεκριμένο benchmark δεν έχει μεγάλο βαθμό παραλληλισμού, με αποτέλεσμα να μην έχουμε και το ανάλογο speedup.

Συνοψίζοντας, μπορούμε να πούμε, ότι το speedup παρουσιάζει μια συνεχή αύξηση μέχρι τους 128 επεξεργαστές, αλλά δεν είναι ανάλογο του αριθμού των επεξεργαστών μας. Συγκεκριμένα, το speedup παρουσιάζεται απογοητευτικό μετά τους 4 επεξεργαστές και αυτό οφείλεται, στο χαμηλό βαθμό παραλληλισμού της συγκεκριμένης εφαρμογής.

5.7.3 LU-Υπολογισμός χώρου του chip

Στο παρακάτω σχήμα (*σχήμα 5.11*), θα προσπαθήσουμε να συγκρίνουμε την αποδοτικότητα των inorder και των out-of-order επεξεργαστών με βάση το χώρο του

chip. Αυτό θα γίνει σύμφωνα με το cache-byte-equivalents (CBE) [15], [16], με τον inorder επεξεργαστή να έχει area 50 KB CBE και τον out-of-order να έχει area 250 KB CBE.



Σχήμα 5.11: *Γραφική που αναπαριστά το chip area για μέχρι και 256 επεξεργαστές.*

Από την παραπάνω γραφική, παρατηρούμε ότι η διαφορά στους inorder και στους out-of-order επεξεργαστές είναι πολύ μεγάλη. Συγκεκριμένα, βλέπουμε ότι οι inorder επεξεργαστές είναι πολύ πιο αργοί σε όλα τα διαστήματα της γραφικής.

Στο μόνο σημείο που θα μπορούσαμε να πούμε, ότι οι inorder επεξεργαστές είναι πιο αποδοτικοί είναι από τα 50 KB CBE έως τα 250 KB CBE και αυτό, δεδομένου ότι θα μας περιοριζόταν ο χώρος σε αυτά τα επίπεδα και άρα δε θα μπορούσαμε να χρησιμοποιήσουμε out-of-order επεξεργαστές αφού ο ελάχιστος χώρος που απαιτείται γι'αυτούς είναι 250 KB CBE.

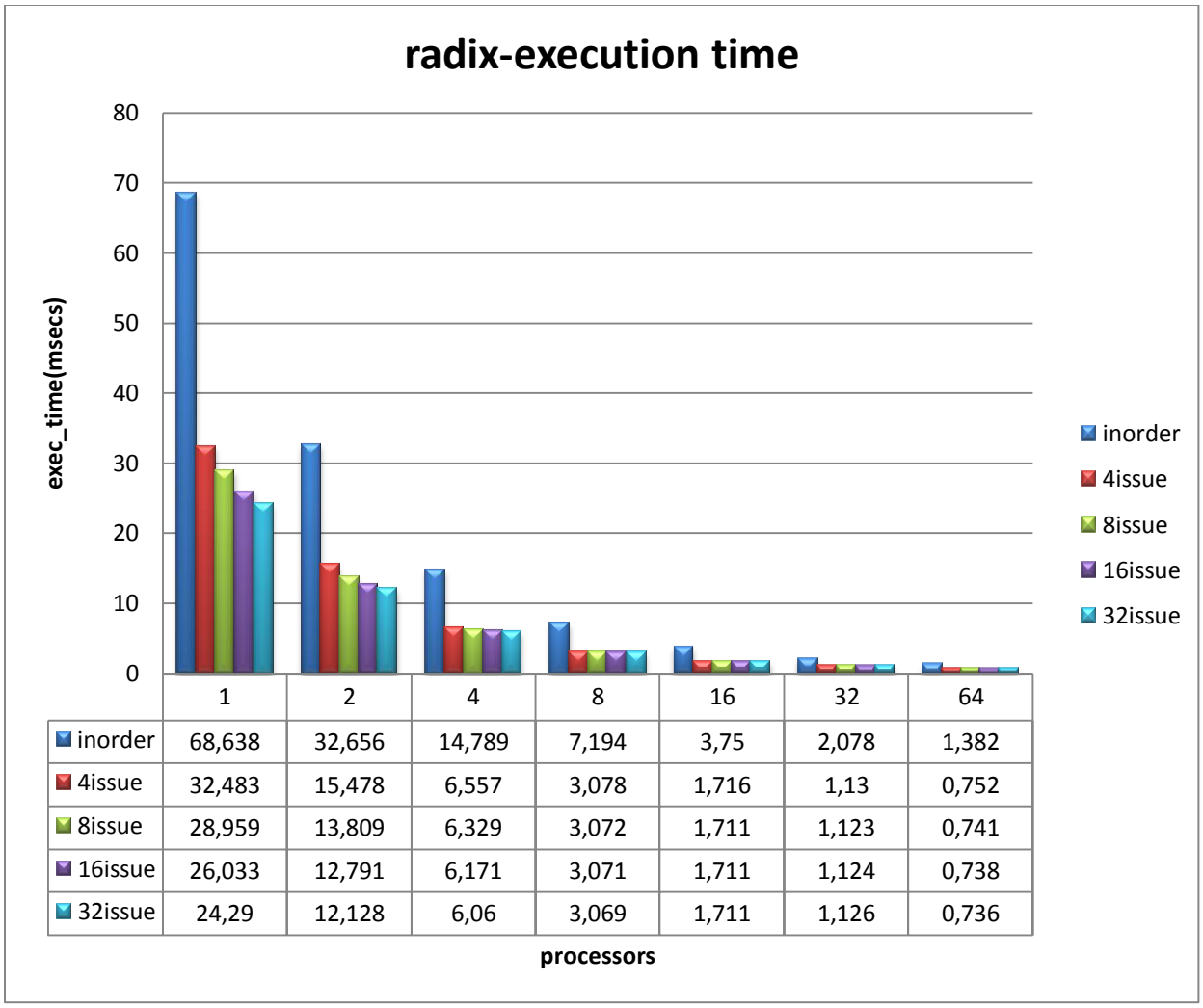
5.8 Ανάλυση αποτελεσμάτων για το RADIX benchmark

Το RADIX kernel [14], είναι ένα benchmark που έχει να κάνει με ταξινόμηση ακεραίων. Ο αλγόριθμος είναι επαναληπτικός, εκτελώντας μια επανάληψη για κάθε ψηφίο radix r των κλειδιών. Σε κάθε επανάληψη, ο επεξεργαστής δημιουργεί ένα τοπικό ιστόγραμμα. Τα τοπικά ιστογράμματα συσσωρεύονται έπειτα σε ένα γενικό ιστόγραμμα.

Τέλος, κάθε επεξεργαστής χρησιμοποιεί το γενικό ιστόγραμμα για να μεταθέσει τα κλειδιά του σε ένα νέο πίνακα, για την επόμενη επανάληψη. Το βήμα της μετάθεσης απαιτεί all-to-all επικοινωνία. Η μετάθεση είναι καθορισμένη από τον αποστολέα, έτσι τα κλειδιά επικοινωνούν μέσω των writes, παρά μέσω των reads.

5.8.1 RADIX-execution time

Στο παρακάτω σχήμα (*σχήμα 5.12*), βλέπουμε τη γραφική για τα 5 configurations(4 issue, 8 issue, 16 issue, 32 issue, inorder) όσον αφορά το execution time.



	1	2	4	8	16	32	64
inorder	68,638	32,656	14,789	7,194	3,75	2,078	1,382
4issue	32,483	15,478	6,557	3,078	1,716	1,13	0,752
8issue	28,959	13,809	6,329	3,072	1,711	1,123	0,741
16issue	26,033	12,791	6,171	3,071	1,711	1,124	0,738
32issue	24,29	12,128	6,06	3,069	1,711	1,126	0,736

Σχήμα 5.12: *Γραφική που αναπαριστά το radix execution time για μέχρι και 64 επεξεργαστές*

Από την παραπάνω γραφική παρατηρούμε όπως και στα προηγούμενα benchmarks, ότι για την inorder εκτέλεση εντολών, έχουμε πολύ μεγάλες τιμές, συγκριτικά με τις υπόλοιπες. Αυτό, είναι κάτι που αναμέναμε, αφού είναι απολύτως λογικό να χρειάζεται μια εφαρμογή περισσότερο χρόνο, όταν είναι αναγκασμένη να εκτελεί τις εντολές της με τη σειρά. Αυτό έχει σαν αποτέλεσμα να μη χρησιμοποιείται αποδοτικά το pipelining και να υπάρχουν καθυστερήσεις λόγω των πολλών εξαρτήσεων.

Για ένα επεξεργαστή στην out-of-order εκτέλεση, παρατηρούμε έναν χρόνο της τάξεως του 68.638 msecs ενώ για την αντίστοιχη 4issue εκτέλεση ένα χρόνο του 32.483

msecs. Η διαφορά λοιπόν, είναι πολύ μεγάλη και βλέπουμε πόσο σημαντική είναι η out-of-order εκτέλεση. Βλέπουμε επίσης, ότι αυξάνοντας το issue, πετυχαίνουμε όλο και καλύτερο χρόνο και καταφέρνουμε να φτάσουμε για ένα επεξεργαστή στην τιμή του 24,29 msecs με 32 issue.

Στον 1 επεξεργαστή βλέπουμε τη μεγαλύτερη διαφορά από το 4 issue στο 32 issue. Η διαφορά που πετυχαίνουμε είναι σημαντική αφού στο 4 issue έχει χρόνο 32,483 msecs ενώ στο 32 issue έχει χρόνο 24,29 msescs. Επίσης και στους 2 επεξεργαστές πετυχαίνουμε αρκετά σημαντική διαφορά, αφού στο 4 issue έχει χρόνο 15,478 msecs, ενώ στο 32 έχει χρόνο 12,128 msescs. Μόνο στους 32 επεξεργαστές, βλέπουμε ότι, έχουμε ελάχιστα χειρότερους χρόνους όσον αφορά το 16 και 32 issue. Στο 8 issue έχουμε χρόνο 1,123 msecs ενώ στο 16 issue έχουμε χρόνο 1,124 msecs, ελάχιστα δηλαδή χειρότερο. Επίσης στο 32 issue έχουμε πάλι ελάχιστο χειρότερο χρόνο από το 16 issue, με προσομοίωση στα 1,126 msecs.

Αυτή τη βελτίωση των χρόνων, τη συναντάμε και αυξάνοντας τους επεξεργαστές. Παρατηρούμε ότι αυξάνοντας τον αριθμό των επεξεργαστών έχουμε μείωση στους χρόνους και για την inorder εκτέλεση και για το μεταβλητό issue. Η βελτίωση στους χρόνους είναι αισθητή μέχρι και τους 64 επεξεργαστές όπου ο χρόνος είναι 0,736 msecs για 32 issue.

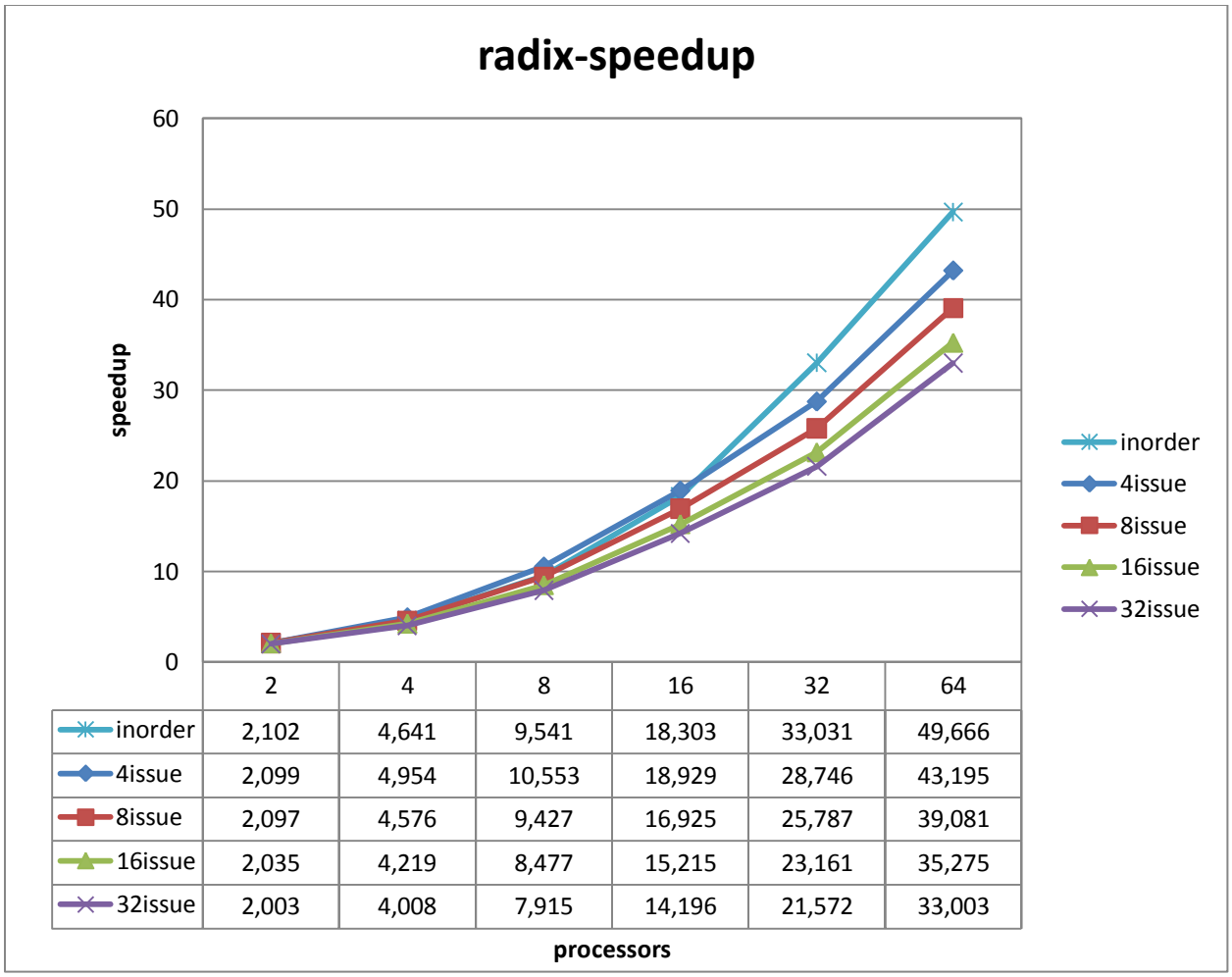
Σε αντίθεση με τα άλλα benchmarks, η συγκεκριμένη εφαρμογή έχει μεγάλο βαθμό παραλληλισμού και αυτό φαίνεται από τους πολύ μικρούς χρόνους που πετυχαίνουμε αυξάνοντας τους επεξεργαστές. Ακόμα και στην inorder εκτέλεση οι χρόνοι μειώνονται πάρα πολύ και για 64 επεξεργαστές πετυχαίνει χρόνο 1,382 msecs δηλαδή προσεγγιστικά έναν χρόνο αρκετά κοντά με την out-of-order εκτέλεση.

Συνοψίζοντας, μπορούμε να πούμε ότι αυξάνοντας τους επεξεργαστές ναι μεν πετυχαίναμε καλύτερο execution time . Στο συγκεκριμένο benchmark δε βλέπουμε κάποια παράξενα αποτελέσματα και αυτό οφείλεται στο μεγάλο βαθμό παραλληλισμού

του. Όσον αφορά την αλλαγή του issue, πετύχαμε σταθερά καλύτερους χρόνους εκτός από τους 32 επεξεργαστές που περιγράψαμε πιο πάνω.

5.8.2 RADIX-speedup

Στο παρακάτω σχήμα (*σχήμα 5.13*), βλέπουμε τη γραφική για τα 5 configurations(4 issue, 8 issue, 16 issue, 32 issue, inorder) όσον αφορά το speedup. Το speedup θα μας δείξει αναλογικά πόση "επιτάχυνση" πετυχαίνουμε συγκριτικά με το χρόνο προσομοίωσης για ένα επεξεργαστή. Ο τύπος που χρησιμοποιήθηκε για τον υπολογισμό του speedup είναι: ***Sp = T1 / Tp*** όπου το p είναι ο αριθμός των επεξεργαστών, το T1 είναι το *execution time* του σειριακού αλγορίθμου και το Tp είναι το execution time του παράλληλου αλγορίθμου με p επεξεργαστές.



	2	4	8	16	32	64
inorder	2,102	4,641	9,541	18,303	33,031	49,666
4issue	2,099	4,954	10,553	18,929	28,746	43,195
8issue	2,097	4,576	9,427	16,925	25,787	39,081
16issue	2,035	4,219	8,477	15,215	23,161	35,275
32issue	2,003	4,008	7,915	14,196	21,572	33,003

Σχήμα 5.13: *Γραφική που αναπαριστά το radix speedup για μέχρι και 64 επεξεργαστές*

Από την παραπάνω γραφική, παρατηρούμε αρχικά, ότι το μεγαλύτερο speedup, όπως και στα προηγούμενα το πετυχαίνουμε για την εκτέλεση των inorder εντολών. Βλέπουμε ότι το speedup σε inorder, φτάνει μέχρι και 49,666 για 64 επεξεργαστές, όταν το αμέσως επόμενο speedup για out of order επεξεργαστές είναι 43,195 για 64 επεξεργαστές σε 4 issue.

Για την inorder εκτέλεση, παρατηρούμε ότι έχουμε συνεχής αύξηση μέχρι τους 64 επεξεργαστές. Το speedup που πετυχαίνουμε είναι συνεχώς ανάλογο του αριθμού των επεξεργαστών. Μόνο στους 64 επεξεργαστές μπορούμε ότι το speedup που πετυχαίνει δεν είναι τόσο κοντά στον αριθμό των επεξεργαστών, αλλά και το 49,666 που έχει δεν

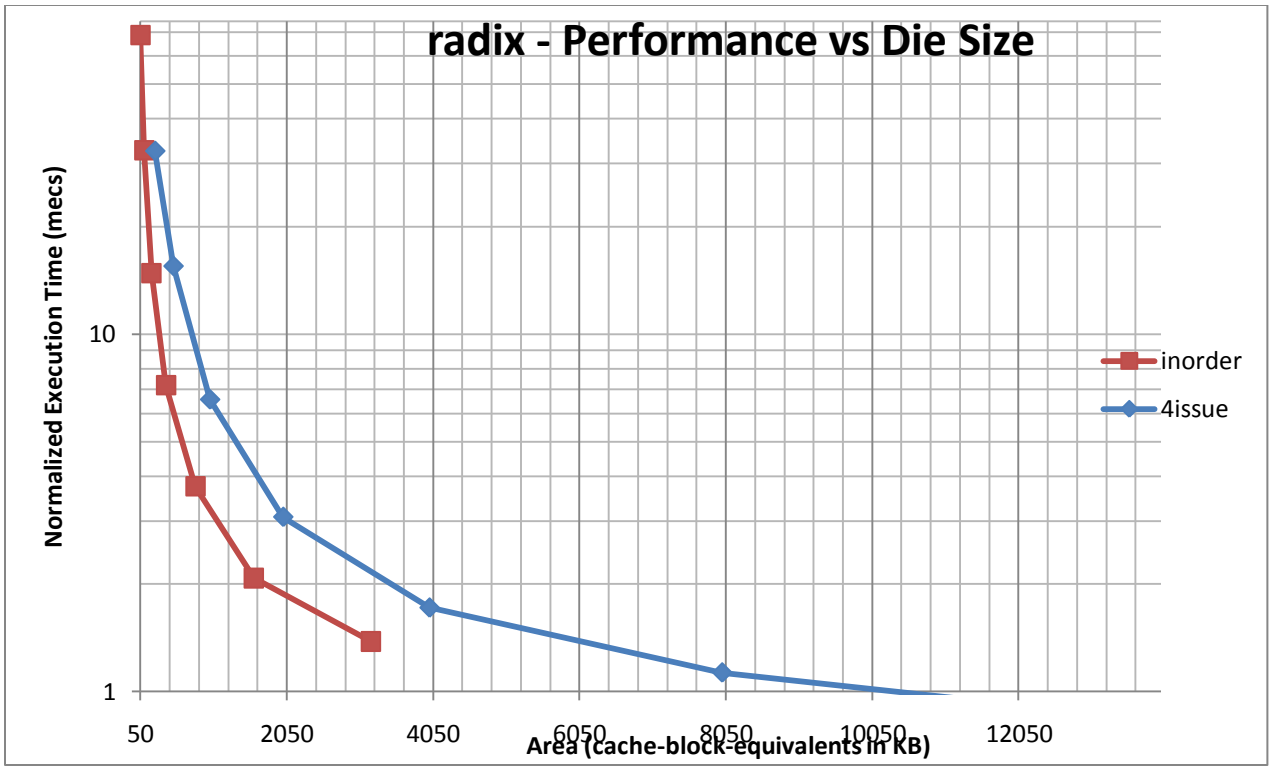
είναι καθόλου άσχημο, ειδικά αν αναλογιστούμε τη συμπεριφορά των προηγούμενων benchmarks.

Όσον αφορά την out-of-order εκτέλεση, το μεγαλύτερο speedup το συναντάμε στο 4 issue. Το μεγαλύτερο speedup, συγκριτικά με τα άλλα issue, είναι στους 64 επεξεργαστές. Στο 4issue των 64 επεξεργαστών έχουμε speedup 43,195, τη στιγμή που για τους ίδιους επεξεργαστές στο 32 issue, έχουμε 33,003. Συγκριτικά και σε αυτό το benchmark μπορούμε να δούμε ότι όσο μικρότερο είναι το issue, το speedup είναι αναλογικά μεγαλύτερο. Το speedup 43,195 μπορούμε να πούμε ότι είναι πάρα πολύ ικανοποιητικό όπως επίσης και το 33,003 για το 32 issue αν συγκρίνουμε με το speedup των προηγούμενων benchmarks.

Συνοψίζοντας, μπορούμε να πούμε, ότι το speedup παρουσιάζει μια συνεχή αύξηση μέχρι και τους 64 επεξεργαστές. Είναι αρκετά ικανοποιητικό και αν στους 64 επεξεργαστές είχαμε λίγο καλύτερους χρόνους θα αγγίζαμε το τέλειο. Παρόλη αυτή τη μικρή απόκλιση στους 64 επεξεργαστές, το speedup είναι σε πάρα πολύ καλά επίπεδα και αυτό οφείλεται στο μεγάλο βαθμό παραλληλισμού της συγκεκριμένης εφαρμογής.

5.8.3 RADIX-Υπολογισμός χώρου του chip

Στο παρακάτω σχήμα (*σχήμα 5.14*), θα προσπαθήσουμε να συγκρίνουμε την αποδοτικότητα των inorder και των out-of-order επεξεργαστών με βάση το χώρο του chip. Αυτό θα γίνει σύμφωνα με το cache-byte-equivalents (CBE) [15], [16], με τον inorder επεξεργαστή να έχει area 50 KB CBE και τον out-of-order να έχει area 250 KB CBE.



Σχήμα 5.14: *Γραφική που αναπαριστά το chip area για μέχρι και 64 επεξεργαστές.*

Από την παραπάνω γραφική, παρατηρούμε και εδώ ότι η διαφορά στο execution time, ανάμεσα στους out-of-order και στους inorder επεξεργαστές, είναι μικρότερη. Για το δεδομένο die area, βλέπουμε ότι χρησιμοποιώντας 2 inorder επεξεργαστές πετυχαίνουμε ανάλογο execution time με τη χρησιμοποίηση 1 out-of-order επεξεργαστή. Άρα εδώ, έχουμε μια εξοικόνομηση του χώρου της τάξεως του 60% για την ίδια απόδοση.

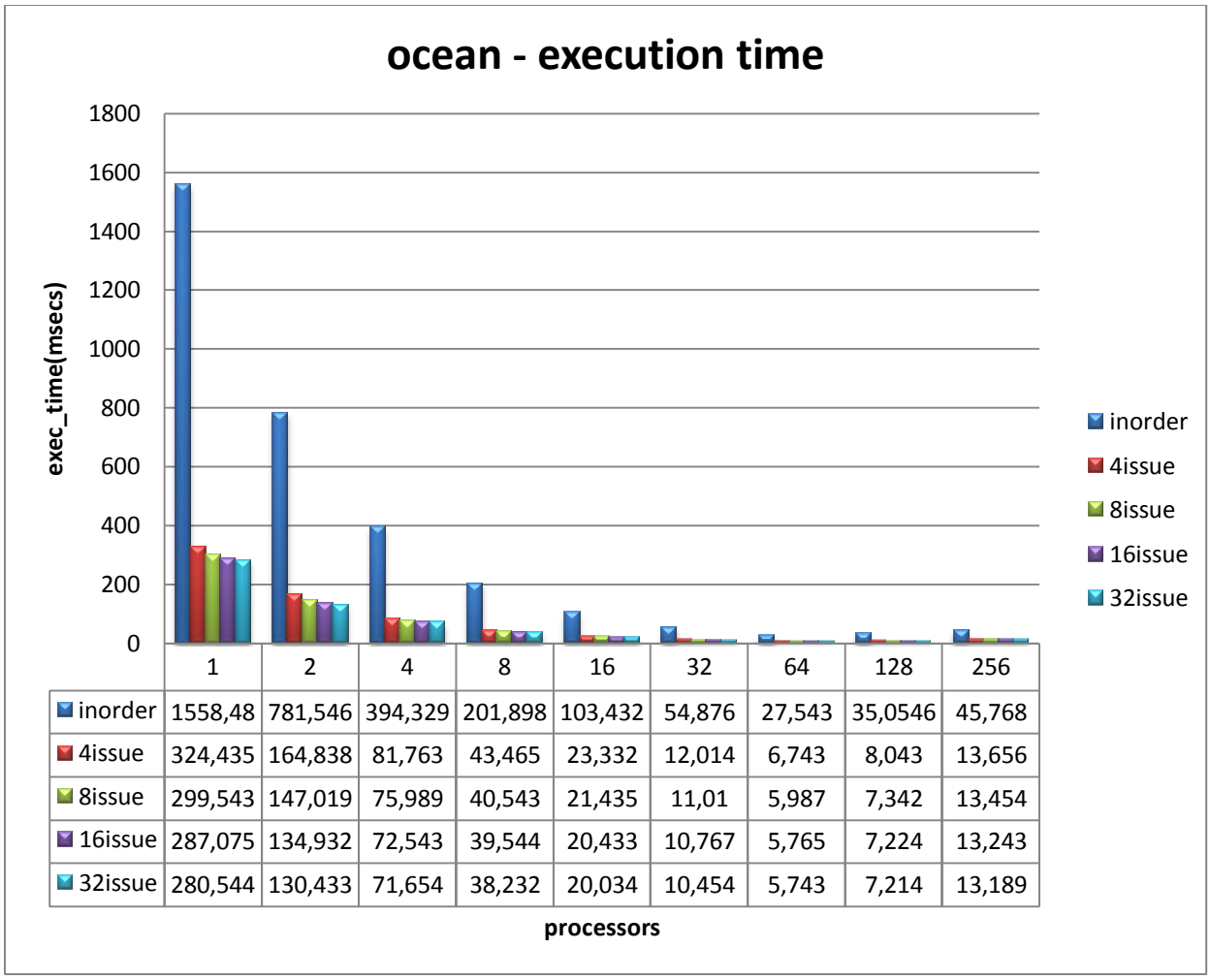
Παρατηρώντας τη γραφική, μπορούμε να δούμε ότι οι inorder επεξεργαστές είναι πιο αποδοτικοί για area από 50 KB CBE, μέχρι 3200 KB CBE. Από εκεί και μετά οι out-of-order επεξεργαστές, φτάνουν σε πολύ χαμηλότερο execution time, χωρίς όμως να μπορούμε να προσομοιώσουμε τους inorder επεξεργαστές, για μεγαλύτερο die size, λόγω αδυναμίας του συγκεκριμένου benchmark.

5.9 Ανάλυση αποτελεσμάτων για το OCEAN benchmark

Το OCEAN application [14], μελετά τις μεγάλης κλίμακας μετακινήσεις του ωκεανού με βάση τους στροβίλους και τα όρια των ρευμάτων και είναι μια βελτιωμένη έκδοση του OCEAN program στο SPLASH. Οι σημαντικότερς διαφορές είναι: i) χωρίζει τα πλέγματα σε υποπλέγματα, αντί σε ομάδες στηλών για να βελτιώσει την επικοινωνία στη συχνότητα του υπολογισμού, ii) τα πλέγματα αντιπροσωπεύονται εννοιολογικά σαν 4-D πίνακες, με όλα τα υποπλέγματα να δεσμεύονται συνεχόμενα και τοπικά στους κόμβους που ανατίθενται και iii) χρησιμοποιεί ένα red-black Gauss-Seidel multigrid λύτη εξίσωσης, αντί για έναν SOR λύτη.

5.9.1 OCEAN-execution time

Στο παρακάτω σχήμα (*σχήμα 5.15*), βλέπουμε τη γραφική για τα 5 configurations(4 issue, 8 issue, 16 issue, 32 issue, inorder) όσον αφορά το execution time .



	1	2	4	8	16	32	64	128	256
inorder	1558,48	781,546	394,329	201,898	103,432	54,876	27,543	35,0546	45,768
4issue	324,435	164,838	81,763	43,465	23,332	12,014	6,743	8,043	13,656
8issue	299,543	147,019	75,989	40,543	21,435	11,01	5,987	7,342	13,454
16issue	287,075	134,932	72,543	39,544	20,433	10,767	5,765	7,224	13,243
32issue	280,544	130,433	71,654	38,232	20,034	10,454	5,743	7,214	13,189

Σχήμα 5.15: *Γραφική που αναπαριστά το ocean execution time για μέχρι και 256 επεξεργαστές*

Από την παραπάνω γραφική παρατηρούμε αρχικά, ότι για την inorder εκτέλεση εντολών, έχουμε πολύ μεγάλες τιμές, συγκριτικά με τις υπόλοιπες. Αυτό, είναι κάτι που παρατηρήσαμε και στα παραπάνω benchmarks και όπως επισημάναμε οφείλεται, στο ότι μια εφαρμογή χρειάζεται περισσότερο χρόνο, όταν είναι αναγκασμένη να εκτελεί τις εντολές της με τη σειρά. Αυτό έχει σαν αποτέλεσμα να έχουμε μεγάλες καθυστερήσεις, λόγω των πολλών εξαρτήσεων αφού δεν μπορεί να γίνει issue περισσότερο από μία εντολή.

Για ένα επεξεργαστή στην inorder εκτέλεση, παρατηρούμε έναν χρόνο της τάξεως του 1558,484 msecs, ενώ για την αντίστοιχη 4issue εκτέλεση ένα χρόνο του

324,435 msecs. Βλέπουμε ότι πλέον, συναντάμε πολύ μεγάλες τιμές αφού προσομοιώνουμε application και μέσα από αυτή τη διαφορά τιμών φαίνεται ξεκάθαρα η σημαντικότητα της out-of-order εκτέλεσης. Βλέπουμε επίσης, ότι αυξάνοντας το issue, πετυχαίνουμε όλο και καλύτερο χρόνο και καταφέρνουμε να φτάσουμε για ένα επεξεργαστή στην τιμή του 280,544 msecs με 32 issue.

Αυτή τη βελτίωση των χρόνων, τη συναντάμε και αυξάνοντας τους επεξεργαστές. Παρατηρούμε ότι αυξάνοντας τον αριθμό των επεξεργαστών έχουμε μείωση στους χρόνους και για την inorder εκτέλεση και για το μεταβλητό issue. Η βελτίωση στους χρόνους είναι αισθητή μέχρι και τους 64 επεξεργαστές όπου ο χρόνος είναι 5,743 msecs για 32 issue. Συγκρίνοντας με προηγούμενα benchmarks, βλέπουμε ότι η συγκεκριμένη εφαρμογή παρουσιάζει πολύ μεγάλο παραλληλισμό αφού από αρχική τιμή σε 4 issue με 324,435 msecs, καταφέρνουμε και τη φτάνουμε σε μια τιμή της τάξης του 5.743 msecs για 32 issue.

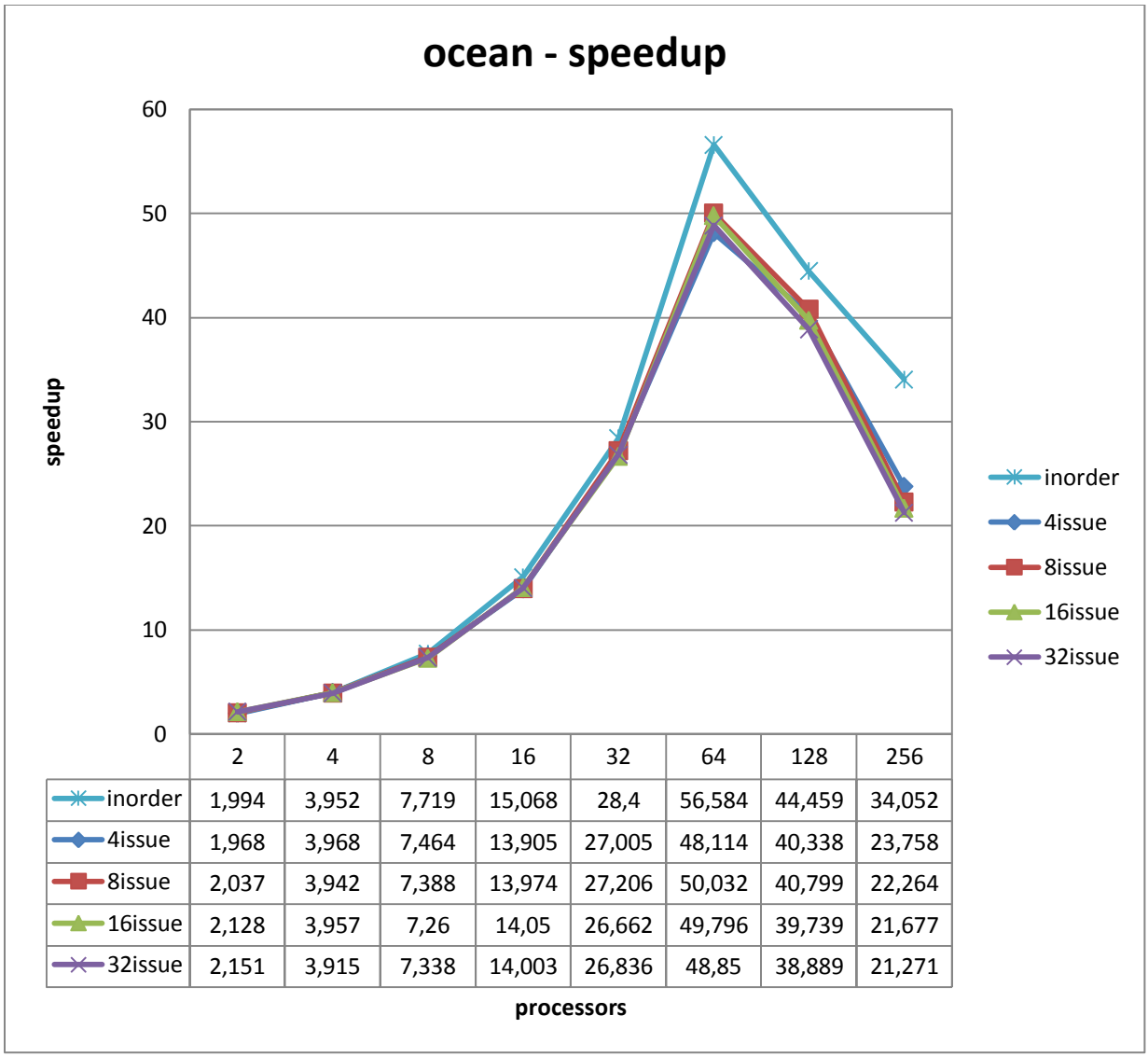
Στους 128 και 256 επεξαργαστές παρατηρούμε, ότι δεν πετυχαίνουμε καλύτερο execution time αφού οι χρόνοι είναι λίγο μεγαλύτεροι. Στο 4 issue για 64 επεξεργαστές, έχουμε execution time 6,743 msecs, ενώ στους 128 επεξεργαστές 8,043 msecs και στους 256 έχουμε 13,656 msecs. Ο βαθμός παραλληλισμού της εφαρμογής είναι μεν υψηλός, αλλά όχι τόσο υψηλός ώστε να μπορεί να διαχειριστεί τους 128 και 256 επεξεργαστές.

Όμως, μέχρι τους 64 επεξεργαστές, βλέπουμε ότι κάθε φορά που αυξάνουμε τους επεξεργαστές το execution time μειώνεται περίπου στο μισό. Για το configuration με το 4 issue βλέπουμε ότι για ένα επεξεργαστή το execution time είναι 324,435 msecs, για 2 επεξεργαστές είναι 164,838 msecs, για 4 επεξεργαστές είναι 81,763 msecs, για 8 επεξεργαστές είναι 43,465 msecs, για 16 επεξεργαστές είναι 23,332 msecs, για 32 επεξεργαστές είναι 12,014 msecs και για 64 επεξεργαστές είναι 6,743. Το ίδιο ισχύει και για την inorder εκτέλεση αφού για ένα επεξεργαστή το execution time είναι 1558,484 msecs, για 2 επεξεργαστές είναι 781,546 msecs, για 4 επεξεργαστές είναι 394,329 msecs, για 8 επεξεργαστές είναι 201,898 msecs, για 16 επεξεργαστές είναι 103,432 msecs, για 32 επεξεργαστές είναι 54,876 msecs και για 64 επεξεργαστές είναι 27,543 msecs.

Συνοψίζοντας, μπορούμε να πούμε ότι η συγκεκριμένη εφαρμογή παρουσιάζει έναν πολύ καλό βαθμό παραλληλισμού μέχρι τους 64 επεξεργαστές αφού διπλασιάζοντας τους επεξεργαστές μειώναμε το execution time περίπου στο μισό και στην inorder και στην out-of-order εκτέλεση. Στους 128 και 256 επεξεργαστές το execution time πετύχαμε λίγο χειρότερους χρόνους, αφού η εφαρμογή μας δεν μπορούσε να διαχειριστεί το πλήθος των επεξεργαστών. Παρ' όλα αυτά, μπορούμε να πούμε ότι μείναμε ικανοποιημένοι αφού μέχρι τους 64 επεξεργαστές η διαφορά στους χρόνους με την αύξηση των επεξεργαστών ήταν η ιδανική. Όσον αφορά την αλλαγή του issue, πετύχαμε σταθερά καλύτερους χρόνους σε όλα τα στάδια της προσομοίωσής μας.

5.9.2 OCEAN-speedup

Στο παρακάτω σχήμα (*σχήμα 5.16*), βλέπουμε τη γραφική για τα 5 configurations(4 issue, 8 issue, 16 issue, 32 issue, inorder) όσον αφορά το speedup. Το speedup θα μας δείξει αναλογικά πόση "επιτάχυνση" πετυχαίνουμε συγκριτικά με το χρόνο προσομοίωσης για ένα επεξεργαστή. Ο τύπος που χρησιμοποιήθηκε για τον υπολογισμό του speedup είναι: ***Sp = T1 / Tp*** όπου το p είναι ο αριθμός των επεξεργαστών, το T1 είναι το *execution time* του σειριακού αλγορίθμου και το Tp είναι το execution time του παράλληλου αλγορίθμου με p επεξεργαστές.



	2	4	8	16	32	64	128	256
inorder	1,994	3,952	7,719	15,068	28,4	56,584	44,459	34,052
4issue	1,968	3,968	7,464	13,905	27,005	48,114	40,338	23,758
8issue	2,037	3,942	7,388	13,974	27,206	50,032	40,799	22,264
16issue	2,128	3,957	7,26	14,05	26,662	49,796	39,739	21,677
32issue	2,151	3,915	7,338	14,003	26,836	48,85	38,889	21,271

Σχήμα 5.16: *Γραφική που αναπαριστά το ocean speedup για μέχρι και 256 επεξεργαστές*

Από την παραπάνω γραφική, παρατηρούμε αρχικά, ότι το μεγαλύτερο speedup, όπως και στα προηγούμενα benchmarks το πετυχαίνουμε για την εκτέλεση των inorder εντολών. Βλέπουμε ότι το speedup σε inorder, φτάνει μέχρι και 56,584 για 64 επεξεργαστές, όταν το αμέσως επόμενο speedup για out-of-order επεξεργαστές είναι 50,032 για 64 επεξεργαστές σε 8 issue.

Βλέπουμε, ότι λόγω του μεγάλου βαθμού παραλληλισμού της εφαρμογής, οι τιμές της γραφικής κινούνται πάνω κάτω στα ίδια επίπεδα με την inorder εκτέλεση να ξεχωρίζει στους 64 επεξεργαστές.

Οι τιμές του speedup για όλα τα configurations είναι ανάλογες με τον αριθμό των επεξεργαστών και αν μη τι άλλο είναι πολύ ικανοποιητικές Αυτό όμως ισχύει για μέχρι τους 64 επεξεργαστές, αφού στους 128 και 256 επεξεργαστές το speedup κινείται σε πολύ χαμηλά επίπεδα Συγκεκριμένα, στους 128 επεξεργαστές για 4 issue, έχουμε speedup 40,338 και στους 256 έχουμε 23,758, ενώ για το αντίστοιχο issue στους 64 επεξεργαστές έχουμε speedup 48,114. Βλέπουμε λοιπόν ότι ο παραλληλισμός της εφαρμογής φτάνει στα όρια του στους 64 επεξεργαστές, αφού σε περισσότερους από τους 64 επεξεργαστές, όχι μόνο δε βλέπουμε βελτίωση του speedup αλλά και επιπλέον μείωση.

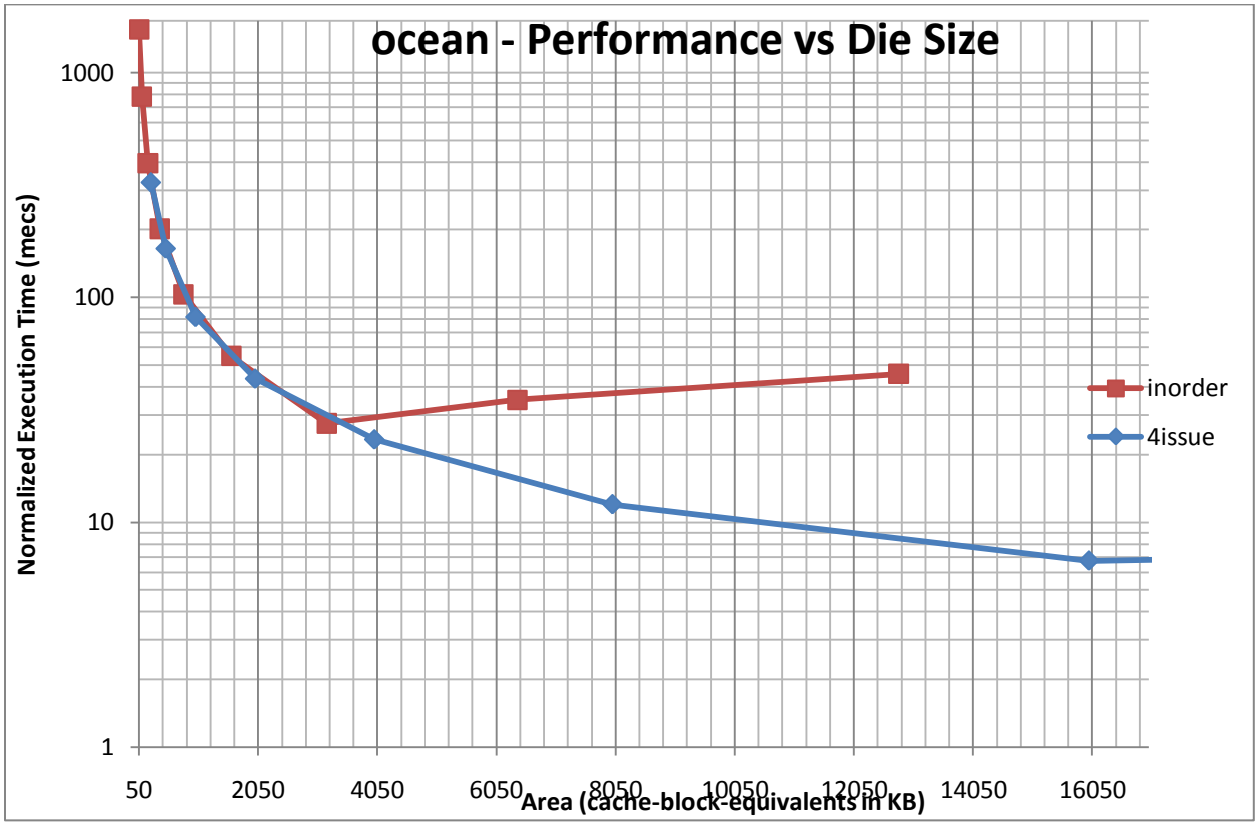
Οι διαφορές βέβαια του speedup μεταξύ του κάθε issue είναι από μικρές έως ελάχιστες. Μέχρι και τους 64 επεξεργαστές το speedup είναι αρκετά κοντά με τον αριθμό των επεξεργαστών για όλα τα configurations. Συγκεκριμένα, στο 4 issue για 64 επεξεργαστές, έχουμε speedup 48,114, στο 8 issue έχουμε speedup 50,032, στο 16 issue έχουμε speedup 49,796, στο 32 issue έχουμε speedup 48,86 και στην inorder εκτέλεση έχουμε speedup 56,584.

Για την inorder εκτέλεση, παρατηρούμε και εδώ ότι έχουμε αύξηση ανάλογη με τον αριθμό των επεξεργαστών μέχρι και τους 64 επεξεργαστές. Από τους 128 επεξεργαστές έχουμε και εδώ μείωση του speedup, αλλά όχι τόσο μεγάλη όσο στην out-of-order εκτέλεση. Συγκεκριμένα, στους 128 επεξεργαστές για inorder, έχουμε speedup 44,459 και στους 256 έχουμε 34,052, ενώ στους 64 επεξεργαστές έχουμε speedup 56,584.

Συνοψίζοντας, μπορούμε να πούμε, ότι το speedup είναι πάρα πολύ ικανοποιητικό μέχρι και τους 64 επεξεργαστές. Μετά τους 64 επεξεργαστές, παρατηρούμε μείωση του speedup και φαίνεται η συγκεκριμένη εφαρμογή δεν μπορεί να διαχειριστεί τον αριθμό τόσων επεξεργαστών.

5.9.3 OCEAN-Υπολογισμός χώρου του chip

Στο παρακάτω σχήμα, θα προσπαθήσουμε να συγκρίνουμε την αποδοτικότητα των inorder και των out-of-order επεξεργαστών με βάση το χώρο του chip. Αυτό θα γίνει σύμφωνα με το cache-byte-equivalents (CBE) [15], [16], με τον inorder επεξεργαστή να έχει area 50 KB CBE και τον out-of-order να έχει area 250 KB CBE.



Σχήμα 5.17: *Γραφική που αναπαριστά το chip area για μέχρι και 256 επεξεργαστές.*

Από την παραπάνω γραφική, παρατηρούμε ότι στον 1 και 2 επεξεργαστές η διαφορά στο execution time, ανάμεσα στους out-of-order και στους inorder επεξεργαστές, είναι αισθητή. Για το δεδομένο die area, βλέπουμε ότι ισχύει σε γενικές γραμμές, ότι συναντήσαμε και στο fft και στο cholesky. Δηλαδή, χρησιμοποιώντας 4 inorder επεξεργαστές πετυχαίνουμε ανάλογο execution time με τη χρησιμοποίηση 1 out-of-order επεξεργαστή. Άρα και εδώ, έχουμε μια εξοικόνομηση του χώρου της τάξεως του 20% για την ίδια απόδοση.

Στο θέμα αποδοτικότητας, ανάμεσα στους inorder και τους out-of-order επεξεργαστές, μπορούμε να δούμε ότι στο συγκεκριμένο benchmark, δεν μπορούμε να καταλήξουμε σε κάποια πιο συμφέρουσα επιλογή. Μέχρι το area των 3200 KB CBE, βλέπουμε ότι δεν υπάρχει διαφορά αφού και οι inorder και οι out-of-order κινούνται στην ίδια καμπύλη.

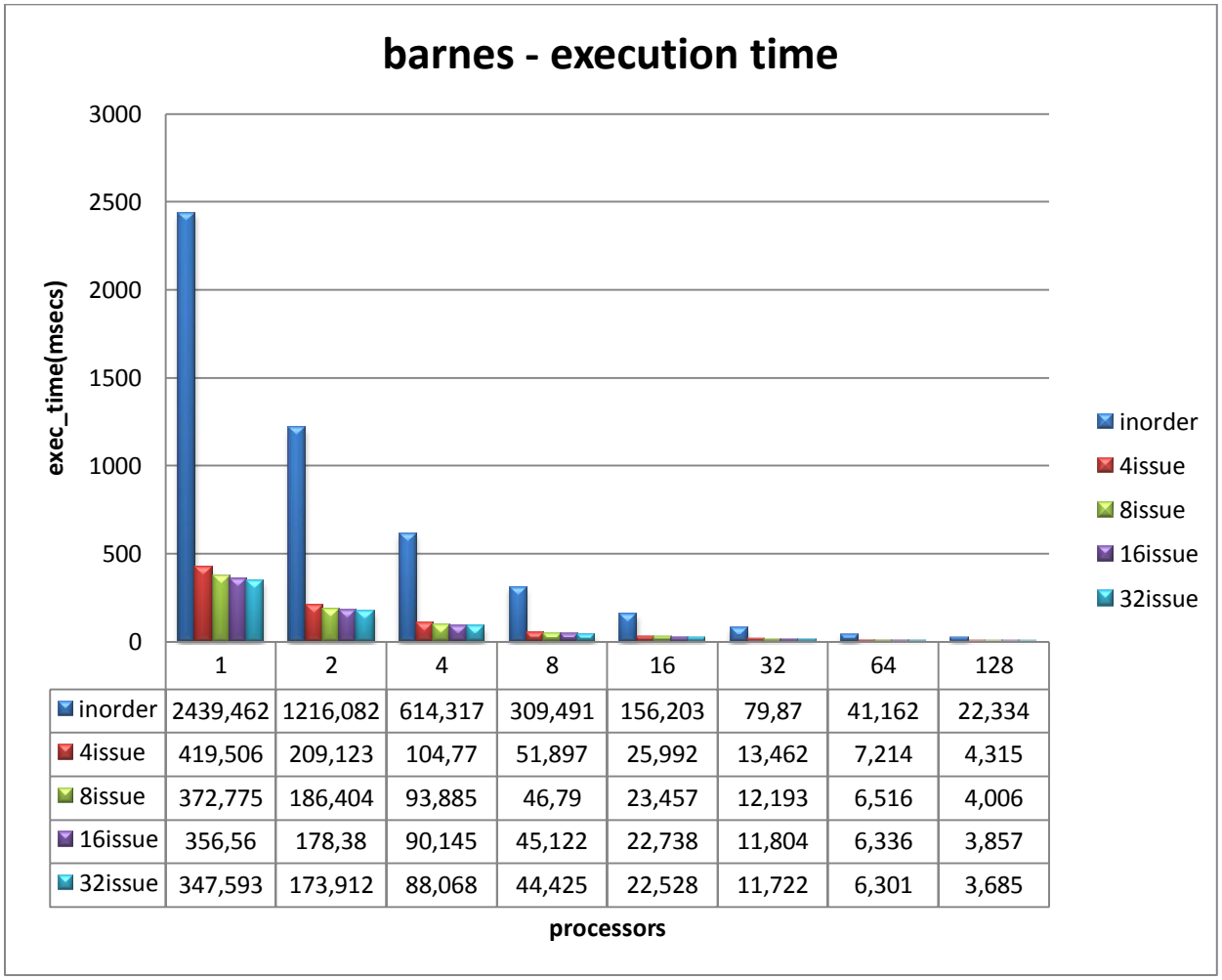
Μετά τα 3200 KB CBE οι out-of-order, είναι φανερά πιο αποδοτικοί, αφού το execution time πέφτει αρκετά σε σχέση με τους inorder επεξεργαστές. Στο κομμάτι που θα μπορούσαμε να πούμε ότι οι inorder επεξεργαστές θα ήταν πιο αποδοτικοι, είναι από τα 50 KB CBE έως τα 250 KB CBE και αυτό, δεδομένου ότι θα μας περιοριζόταν ο χώρος σε αυτά τα επίπεδα και άρα δε θα μπορούσαμε να χρησιμοποιήσουμε out-of-order επεξεργαστές αφού ο ελάχιστος χώρος που απαιτείται γι'αυτούς είναι 250 KB CBE.

5.10 Ανάλυση αποτελεσμάτων για το BARNES benchmark

Το BARNES application [14], προσομοιώνει ένα σύστημα από οργανισμούς(πχ. γαλαξίες ή μόρια) σε τρεις διαστάσεις με έναν αριθμό από time-steps, χρησιμοποιώντας τη μέθοδο Barnes-Hut hierarchical N-body. Διαφέρει σε δύο πράγματα από την έκδοση του SPLASH: i) επιτρέπει πολλαπλά μόρια ανά κύτταρο φύλλων και ii) εφαρμόζει τις δομές δεδομένων κυττάρων διαφορετικά, για καλύτερη τοποθέτηση στοιχείων. Όπως στο SPLASH application, αναπαριστά την υπολογιστική περιοχή σαν octree, με τα φύλλα να περιέχουν τις πληροφορίες για κάθε σώμα και εσωτερικούς κόμβους που αντιπροσωπεύουν τα space cells. Τις περισσότερες φορές, ξοδεύεται σε μερικά traversals του octree (ένα traversal ανά οργανισμό) για να υπολογίσει τις δυνάμεις στους μεμονωμένους οργανισμούς. Η επικοινωνία εξαρτάται από τη διανομή μορίων και είναι μη δομημένη. Στη main memory, δε γίνεται καμία προσπάθεια, ώστε να υπάρχει έξυπνη κατανομή από δεδομένα οργανισμών, δεδομένου ότι αυτό είναι δύσκολο στο page granularity και όχι τόσο σημαντικό στην απόδοση.

5.10.1 BARNES-execution time

Στο παρακάτω σχήμα (*σχήμα 5.18*), βλέπουμε τη γραφική για τα 5 configurations(4 issue, 8 issue, 16 issue, 32 issue, inorder) όσον αφορά το execution time .



	1	2	4	8	16	32	64	128
inorder	2439,462	1216,082	614,317	309,491	156,203	79,87	41,162	22,334
4issue	419,506	209,123	104,77	51,897	25,992	13,462	7,214	4,315
8issue	372,775	186,404	93,885	46,79	23,457	12,193	6,516	4,006
16issue	356,56	178,38	90,145	45,122	22,738	11,804	6,336	3,857
32issue	347,593	173,912	88,068	44,425	22,528	11,722	6,301	3,685

***Σχήμα 5.18**: Γραφική που αναπαριστά το barnes execution time για μέχρι και 128 επεξεργαστές*

Από την παραπάνω γραφική παρατηρούμε αρχικά, ότι για την inorder εκτέλεση εντολών, έχουμε πολύ μεγάλες τιμές, συγκριτικά με τις υπόλοιπες. Αυτό, είναι κάτι που παρατηρήσαμε και στα παραπάνω benchmarks και όπως επισημάναμε οφείλεται, στο ότι

μια εφαρμογή χρειάζεται περισσότερο χρόνο, όταν είναι αναγκασμένη να εκτελεί τις εντολές της με τη σειρά. Αυτό έχει σαν αποτέλεσμα να έχουμε μεγάλες καθυστερήσεις, λόγω των πολλών εξαρτήσεων αφού δεν μπορεί να γίνει issue περισσότερο από μία εντολή.

Για ένα επεξεργαστή στην inorder εκτέλεση, παρατηρούμε έναν χρόνο της τάξεως του 2439,462 msecs ενώ για την αντίστοιχη 4issue εκτέλεση ένα χρόνο του 419,506 msecs. Βλέπουμε ότι πλέον, συναντάμε πολύ μεγάλες τιμές αφού προσομοιώνουμε application και μέσα από αυτή τη διαφορά τιμών φαίνεται ξεκάθαρα η σημαντικότητα της out-of-order εκτέλεσης. Βλέπουμε επίσης, ότι αυξάνοντας το issue, πετυχαίνουμε όλο και καλύτερο χρόνο και καταφέρνουμε να φτάσουμε για ένα επεξεργαστή στην τιμή του 347,593 msecs με 32 issue.

Αυτή τη βελτίωση των χρόνων, τη συναντάμε και αυξάνοντας τους επεξεργαστές. Παρατηρούμε ότι αυξάνοντας τον αριθμό των επεξεργαστών έχουμε μείωση στους χρόνους και για την inorder εκτέλεση και για το μεταβλητό issue. Η βελτίωση στους χρόνους είναι αισθητή μέχρι και τους 128 επεξεργαστές όπου ο χρόνος είναι 3,685 msecs για 32 issue. Συγκρίνοντας με προηγούμενα benchmarks, βλέπουμε ότι η συγκεκριμένη εφαρμογή παρουσιάζει πολύ μεγάλο παραλληλισμό αφού από αρχική τιμή σε 4 issue με 419,506 msecs, καταφέρνουμε και τη φτάνουμε σε μια τιμή της τάξης του 3.685 msecs για 32 issue. Πραγματικά, η διαφορά είναι πάρα πολύ μεγάλη και η εφαρμογή φαίνεται να χρησιμοποιεί στο έπακρο όλους τους πυρήνες του επεξεργαστή.

Ενδεικτικά, μπορούμε να δούμε, ότι κάθε φορά που αυξάνουμε τους επεξεργαστές το execution time μειώνεται περίπου στο μισό. Για το configuration με το 4 issue βλέπουμε ότι για ένα επεξεργαστή το execution time είναι 419,506 msecs, για 2 επεξεργαστές είναι 209,123 msecs, για 4 επεξεργαστές είναι 104,77 msecs, για 8 επεξεργαστές είναι 51,897 msecs, για 16 επεξεργαστές είναι 25,992 msecs, για 32 επεξεργαστές είναι 13,462 msecs, για 64 επεξεργαστές είναι 7,214 msecs και για 128 επεξεργαστές είναι 4,315 msecs. Το ίδιο ισχύει και για την inorder εκτέλεση αφού για ένα επεξεργαστή το execution time είναι 2439,462 msecs, για 2 επεξεργαστές είναι

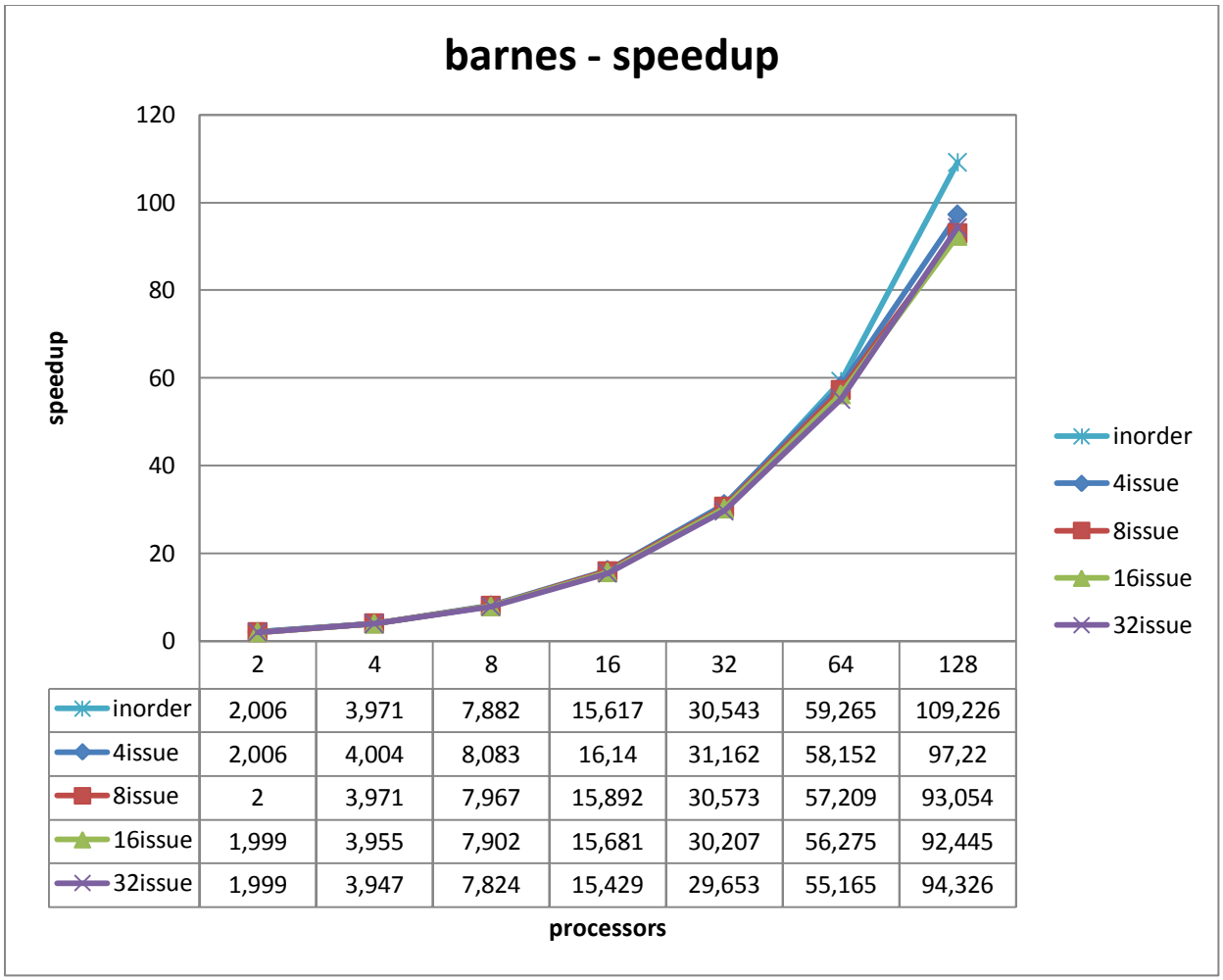
1216,082 msecs, για 4 επεξεργαστές είναι 614,317 msecs, για 8 επεξεργαστές είναι 309,491 msecs, για 16 επεξεργαστές είναι 156,203 msecs, για 32 επεξεργαστές είναι 79,87 msecs, για 64 επεξεργαστές είναι 41,162 msecs και για 128 επεξεργαστές είναι 22,314 msecs

Με την αλλαγή των issue, είχαμε σε όλες τις προσομοιώσεις καλύτερους χρόνους, με τις μεγαλύτερες διαφορές να τις παρατηρούμε από το 4 issue στο 8 issue. Στο 16 και 32 issue είχαμε πάλι καλύτερους χρόνους, απλά οι διαφορές μεταξύ τους ήταν σχετικά πιο μικρές. Ενδεικτικά, για 4 επεξεργαστές στο 4 issue, το execution time είναι 104,77 msecs, στο 8 issue είναι 93,885 msecs, στο 16 issue είναι 90,145 msecs και στο 32 issue είναι 88,068 msecs.

Συνοψίζοντας, μπορούμε να πούμε ότι η συγκεκριμένη εφαρμογή παρουσιάζει πάρα πολύ μεγάλο παραλληλισμό αφού διπλασιάζοντας τους επεξεργαστές μειώναμε το execution time περίπου στο μισό και στην inorder και στην out-of-order εκτέλεση. Όσον αφορά την αλλαγή του issue, πετύχαμε σταθερά καλύτερους χρόνους σε όλα τα στάδια της προσομοίωσής μας.

5.10.2 BARNES-speedup

Στο παρακάτω σχήμα (*σχήμα 5.19*), βλέπουμε τη γραφική για τα 5 configurations(4 issue, 8 issue, 16 issue, 32 issue, inorder) όσον αφορά το speedup. Το speedup θα μας δείξει αναλογικά πόση "επιτάχυνση" πετυχαίνουμε συγκριτικά με το χρόνο προσομοίωσης για ένα επεξεργαστή. Ο τύπος που χρησιμοποιήθηκε για τον υπολογισμό του speedup είναι: ***Sp = T1 / Tp*** όπου το p είναι ο αριθμός των επεξεργαστών, το T1 είναι το *execution time* του σειριακού αλγορίθμου και το Tp είναι το execution time του παράλληλου αλγορίθμου με p επεξεργαστές.



	2	4	8	16	32	64	128
inorder	2,006	3,971	7,882	15,617	30,543	59,265	109,226
4issue	2,006	4,004	8,083	16,14	31,162	58,152	97,22
8issue	2	3,971	7,967	15,892	30,573	57,209	93,054
16issue	1,999	3,955	7,902	15,681	30,207	56,275	92,445
32issue	1,999	3,947	7,824	15,429	29,653	55,165	94,326

processors

Σχήμα 5.19: *Γραφική που αναπαριστά το barnes speedup για μέχρι και 128 επεξεργαστές*

Από την παραπάνω γραφική, παρατηρούμε αρχικά, ότι το μεγαλύτερο speedup, όπως και στα προηγούμενα benchmarks το πετυχαίνουμε για την εκτέλεση των inorder εντολών. Βλέπουμε ότι το speedup σε inorder, φτάνει μέχρι και 109,226 για 128 επεξεργαστές, όταν το αμέσως επόμενο speedup για out-of-order επεξεργαστές είναι 94,326 για 128 επεξεργαστές σε 32 issue.

Βλέπουμε, ότι λόγω του μεγάλου βαθμού παραλληλισμού της εφαρμογής, οι τιμές της γραφικής κινούνται πάνω κάτω στα ίδια επίπεδα με την inorder εκτέλεση να ξεχωρίζει στους 128 επεξεργαστές.

Οι τιμές του speedup για όλα τα configurations είναι ανάλογες με τον αριθμό των επεξεργαστών και αν μη τι άλλο είναι πολύ ικανοποιητικές. Θεωρητικά, στην out-of-order εκτέλεση, το καλύτερο speedup για κάθε αριθμό των επεξεργαστών το συναντάμε στο configuration με το 4 issue, μόνο στους 128 επεξεργαστές το καλύτερο speedup το συναντάμε στο 32 issue. Οι διαφορές βέβαια του speedup μεταξύ του κάθε issue είναι από μικρές έως ελάχιστες. Μέχρι και τους 64 επεξεργαστές το speedup είναι περίπου το ίδιο με τον αριθμό των επεξεργαστών για όλα τα configurations. Συγκεκριμένα στο 4 issue έχουμε speedup 58,152, στο 8 issue έχουμε speedup 57,209, στο 16 issue έχουμε speedup 56,275, στο 32 issue έχουμε speedup 55,165 και στην inorder εκτέλεση έχουμε speedup 59,265. Στους 128 επεξεργαστές το speedup που συναντάμε μπορεί να μην είναι ανάλογο των επεξεργαστών μας αλλά δεν παύει να είναι αρκετά ικανοποιητικό. Στο 4 issue έχουμε speedup 97,22, στο 8 issue έχουμε speedup 93,054, στο 16 issue έχουμε speedup 92,445, στο 32 issue έχουμε speedup 94,326 και στην inorder εκτέλεση έχουμε speedup 109,226.

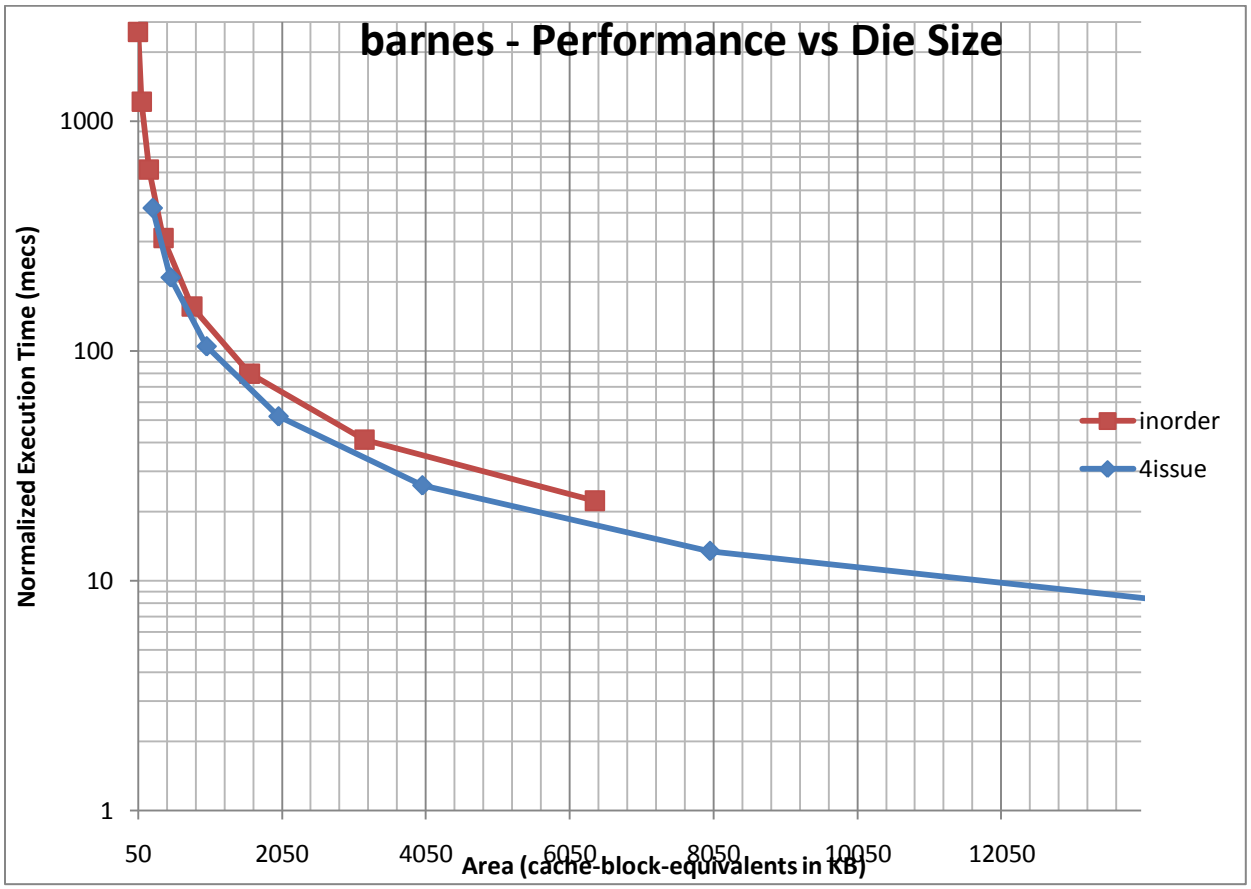
Για την inorder εκτέλεση, παρατηρούμε ότι έχουμε αύξηση μέχρι τους 256 επεξεργαστές. Η αύξηση, είναι και αυτή ανάλογη του αριθμού των επεξεργαστών, μέχρι και τους 128 επεξεργαστές. Συγκριτικά, μπορούμε να δούμε ότι όσο μικρότερο είναι το issue, το speedup είναι σχετικά μεγαλύτερο.

Συνοψίζοντας, μπορούμε να πούμε, ότι το speedup είναι πάρα πολύ ικανοποιητικό και αυτό οφείλεται στο μεγάλο βαθμό παραλληλισμού της εφαρμογής αφού πετυχαίνουμε τις μεγαλύτερες τιμές σε speedup, αν τις συγκρίνουμε με τα υπόλοιπα benchmarks.

5.10.3 BARNES-Υπολογισμός χώρου του chip

Στο παρακάτω σχήμα (*σχήμα 5.20*), θα προσπαθήσουμε να συγκρίνουμε την αποδοτικότητα των inorder και των out-of-order επεξεργαστών με βάση το χώρο του chip. Αυτό θα γίνει σύμφωνα με το cache-byte-equivalents (CBE) [15], [16], με τον

inorder επεξεργαστή να έχει area 50 KB CBE και τον out-of-order να έχει area 250 KB CBE.



Σχήμα 5.20: *Γραφική που αναπαριστά το chip area για μέχρι και 128 επεξεργαστές.*

Από την παραπάνω γραφική, παρατηρούμε ότι χρησιμοποιώντας 8 inorder επεξεργαστές πετυχαίνουμε ανάλογο execution time με τη χρησιμοποίηση 1 out-of-order επεξεργαστή. Αυτό σημαίνει ότι για 8 inorder επεξεργαστές, χρειαζόμαστε 400 KB CBE, ενώ για 1 out-of-order επεξεργαστή χρειαζόμαστε 250 KB CBE. Είναι λοιπόν ξεκάθαρο, πως για το συγκεκριμένο benchmark, η χρησιμοποίηση των inorder επεξεργαστών δεν αποσκοπεί σε κανένα κέρδος. Αντίθετα μάλιστα, όχι μόνο δεν έχουμε βελτίωση του χώρου, αλλά και επιπλέον επιβάρυνση.

Από θέμα απόδοσης, βλέπουμε ότι οι out-of-order επεξεργαστές κινούνται σε χαμηλότερο execution time, από τους inorder επεξεργαστές, σε όλα τα πλαίσια της

προσομοίωσης. Όπως στο lu και στο ocean, θα μπορούσαμε να πούμε πιο αποδοτικούς τους inorder επεξεργαστές από τα 50 KB CBE έως τα 250 KB CBE δεδομένου, ότι θα μας περιοριζόταν ο χώρος σε αυτά τα επίπεδα και άρα δε θα μπορούσαμε να χρησιμοποιήσουμε out-of-order επεξεργαστές αφού ο ελάχιστος χώρος που απαιτείται γι'αυτούς είναι 250 KB CBE.

5.11 Περίληψη και σύνοψη των πειραματικών αποτελεσμάτων

Μετά τη λεπτομερή ανάλυση των αποτελεσμάτων, ακολουθεί μια γενική σύνοψη για το κάθε benchmark ξεχωριστά.

Για το **fft**: είδαμε ότι πρόκειται για ένα σχετικά μικρό kernel και αυτό μπορούμε να το δούμε από τους χαμηλούς χρόνους προσομοίωσης. Το execution time βελτιωνόταν συνεχώς μέχρι και τους 64 επεξεργαστές. Στους 128 επεξεργαστές, είχαμε βελτίωση μόνο στο 16 issue, 32 issue και inorder επεξεργαστές. Στους 256 επεξεργαστές πετύχαμε χειρότερους χρόνους από τους 64 επεξεργαστές και στο 4 και 8 issue είχαμε χειρότερους χρόνους από τους 32 επεξεργαστές.

Σε γενικές γραμμές, είδαμε ότι αυξάνοντας το issue, πετυχαίναμε συνεχώς καλύτερους χρόνους και ότι ο παραλληλισμός της εφαρμογής, άρχισε να περιορίζεται από τους 64 επεξεργαστές και μετά. Στην inorder εκτέλεση και στα χαμηλότερα issue ναι μεν πετυχαίναμε καλύτερο speedup αλλά χειρότερα execution time , ειδικά στην inorder εκτέλεση, οι διαφορές στο execution time ήταν πολύ μεγάλες.

Όσο αφορά την απόδοση συγκριτικά με το die area, είδαμε ότι με inorder επεξεργαστές, έχουμε μια εξοικονόμηση του χώρου 20% και ότι από τα 50 KB CBE μέχρι τα 2050 CBE, οι inorder επεξαργαστές είναι πιο αποδοτικοί και ότι μετά τα 2050 KB CBE, οι out-of-order είναι πιο αποδοτικοί.

Για το **cholesky**: είναι ένα σχετικά μεγάλο kernel και αυτό φάνηκε από τους υψηλούς χρόνους του execution time. Είδαμε ότι παρουσίασε ένα αρκετά μεγάλο βαθμό

παραλληλισμού ειδικά μέχρι και τους 16 επεξεργαστές. Από εκεί και μετά πετύχαμε συνεχώς καλύτερους χρόνους, χωρίς όμως το speedup να δικαιολογεί το μεγάλο αριθμό των επεξεργαστών.

Είδαμε πάλι ότι αυξάνοντας το issue πετυχαίναμε καλύτερους χρόνους και ειδικά μέχρι και τους 8 επεξεργαστές, οι διαφορές ήταν σημαντικές. Και σε αυτό το benchmark είδαμε, ότι αυξάνοντας το issue ναι μεν πετυχαίνουμε καλύτερο execution time, όμως το speedup είναι μικρότερο. Το μεγαλύτερο speedup, το συναντήσαμε στην inorder εκτέλεση.

Όσο αφορά την απόδοση, συγκριτικά με το die area, είδαμε και σε αυτό το benchmark, ότι με inorder επεξεργαστές, έχουμε μια εξοικονόμηση του χώρου 20% και ότι από τα 50 KB CBE μέχρι τα 800 KB CBE, οι inorder επεξαργαστές είναι πιο αποδοτικοί και ότι μετά τα 800 KB CBE, οι out-of-order είναι πιο αποδοτικοί.

Για το **lu**: πρόκειται και αυτό για ένα σχετικά μεγάλο kernel, με το πιο χαμηλό όμως βαθμό παραλληλισμού. Είδαμε ότι από τους 8 επεξεργαστές και μετά το speedup αρχίζει και γίνεται απογοητευτικό. Στην inorder εκτέλεση, παρατηρούμε βελτιωμένο speedup, χωρίς όμως αυτό να σημαίνει ότι είναι ικανοποιητικό.

Μέχρι και τους 128 επεξεργαστές παρατηρούμε βελτίωση του execution time, με πολύ όμως μικρές διαφορές. Στους 256 επεξεργαστές συναντάμε χειρότερους χρόνους απότι στους 128, παρά μόνο στην inorder εκτέλεση, βλέπουμε μια μικρή βελτίωση.

Όσο αφορά την απόδοση, συγκριτικά με το die area, είδαμε ότι οι inorder επεξεργαστές, είναι ασύμφοροι για να τους χρησιμοποιήσουμε και στο μόνο area που θα μπορούσαν να είναι αποδοτικοί, είναι από 50 KB έως 250 KB CBE, δεδομένου ότι θα μας είχε περιοριστεί ο χώρος σε αυτά τα επίπεδα και άρα δε θα μπορούσαμε να χρησιμοποιήσουμε out-of-order επεξεργαστές, αφού ο ελάχιστος χώρος γι'αυτούς είναι 250 KB CBE.

Για το **radix**: είναι ένα kernel με αρκετά μεγάλο βαθμό παραλληλισμού. Είδαμε ότι μέχρι και τους 64 επεξεργαστές, που μας επέτρεπε να προσομοιώσουμε το συγκεκριμένο benchmark, πετύχαμε συνεχώς καλύτερο execution time και μάλιστα με σημαντικές διαφορές. Είδαμε, ότι αυξάνοντας το issue, πετύχαμε καλύτερους χρόνους με εξαίρεση τους 32 επεξεργαστές, με τις διαφορές όμως να είναι πολύ μικρές.

Και εδώ είδαμε, ότι αυξάνοντας το issue ναι μεν πετυχαίνουμε καλύτερο execution time, όμως το speedup είναι μικρότερο και το μεγαλύτερο speedup, το συναντάμε στην inorder εκτέλεση. Μέχρι τους 32 επεξεργαστές, το speedup είναι παρα πολύ καλό. Στους 64 επεξεργαστές, το speedup μπορεί να μην είναι ανάλογο των επεξεργαστών, όμως δεν παύει να είναι ικανοποιητικό.

Όσον αφορά την απόδοση, συγκριτικά με το die area, είδαμε ότι με inorder επεξεργαστές, έχουμε μια εξοικονόμηση του χώρου 60% και ότι από τα 50 KB CBE μέχρι τα 3200 KB CBE, οι inorder επεξαργαστές είναι πιο αποδοτικοί. Μετά τα 3200 KB CBE οι out-of-order επεξεργαστές, φτάνουν σε πολύ χαμηλότερο execution time, χωρίς όμως να μπορούμε να προσομοιώσουμε τους inorder επεξεργαστές, για μεγαλύτερο die size, λόγω αδυναμίας του συγκεκριμένου benchmark.

Για το **ocean**: πρόκειται για ένα application, με υψηλό βαθμό παραλληλισμού. Μέχρι και τους 64 επεξεργαστές, συναντάμε συνεχή και σημαντική βελτίωση στους χρόνους. Στους 128 και 256 επεξεργαστές, το execution time χειροτερεύει.

Αυξάνοντας το issue, πετυχαίνουμε καλύτερους χρόνους χωρίς όμως σημαντικές διαφορές. Το speedup, μέχρι και τους 32 επεξεργαστές μπορεί να θεωρηθεί παρά πολύ καλό, όμως και στους 64 επεξεργαστές δεν παύει να είναι ικανοποιητικό. Και εδώ στην inorder εκτέλεση, συναντάμε το μεγαλύτερο speedup, χωρίς όμως οι διαφορές να είναι τόσο μεγάλες, όσο

Όσον αφορά την απόδοση, συγκριτικά με το die area, είδαμε όπως και στο fft και το cholesky, ότι με inorder επεξεργαστές, έχουμε μια εξοικονόμηση του χώρου 20% και ότι μέχρι τα 3200 KB CBE δεν υπάρχει διαφορά, αφού οι inorder και οι out-of-order

επεξεργαστές κινούνται στην ίδια καμπύλη. Μετά τα 3200 KB CBE οι out-of-order επεξεργαστές είναι πιο αποδοτικοί. Οι inorder επεξεργαστές στο μόνο area που θα μπορούσαν να είναι αποδοτικοί είναι απο 50 KB CBE μέχρι 250 KB CBE για τους λόγους που περιγράψαμε παραπάνω.

Για το **barnes**: πρόκειται και αυτό για ένα application, με υψηλό βαθμό παραλληλισμού. Το execution time μειώνεται σημαντικά, με την αύξηση των επεξεργαστών και μάλιστα μέχρι και τους 128 επεξεργαστές. Μέχρι και τους 64 επεξεργαστές το speedup είναι σχεδόν ανάλογο με τον αριθμό των επεξεργαστών, όμως και στους 128 επεξεργαστές το speedup μπορεί να χαρακτηριστεί υψηλό.

Και εδώ το μεγαλύτερο speedup, το συναντάμε στην inorder εκτέλεση, με πολύ μικρές όμως διαφορές. Επίσης, βλέπουμε ότι αυξάνοντας το issue πετυχαίνουμε καλύτερο execution time, αλλά το speedup είναι μικρότερο. Η συγκεκριμένη εφαρμογή παρουσίασε το μεγαλύτερο βαθμό παραλληλισμού συγκριτικά με τις υπόλοιπες.

Όσο αφορά την απόδοση, συγκριτικά με το die area, είδαμε ότι σε αυτό το benchmark δεν έχουμε καθόλου εξοικόνομηση χώρου με inorder επεξεργαστές. Γενικότερα, είδαμε ότι οι out-of-order επεξεργαστές είναι πιο αποδοτικοί από τους inorder, αφού κινούνται σε χαμηλότερο execution time σε όλα τα πλαίσια της προσομοίωσης. και σε αυτό το benchmark, ότι με inorder επεξεργαστές, έχουμε μια εξοικονόμηση του χώρου 20% και ότι από τα 50 KB CBE μέχρι τα 800 KB CBE, οι inorder επεξαργαστές είναι πιο αποδοτικοί και ότι μετά τα 800 KB CBE, οι out-of-order είναι πιο αποδοτικοί. Όπως στο lu και στο ocean, οι inorder επεξεργαστές, στο μόνο area που θα μπορούσαν να είναι αποδοτικοί είναι απο 50 KB CBE μέχρι 250 KB CBE για τους λόγους που περιγράψαμε παραπάνω.

6. **Κεφάλαιο 6**

Συμπεράσματα και Μελλοντική εργασία

6.1 Γενικά συμπεράσματα 83
6.2 Μελλοντική εργασία 86

6.1 Γενικά συμπεράσματα

Η παρούσα ατομική διπλωματική εργασία έχει παρουσιάσει, μια προσπάθεια για καλύτερη επίδοση και απόδοση στο θέμα των πολυπύρηνων επεξεργαστών. Μέσα από πειράματα και αλλαγές των configurations των benchmarks, προσπάθησα να καταλήξω σε κάποια συμπεράσματα, τα οποία και θα παρουσιάσω σε αυτό το κεφάλαιο.

Η επίδοση και η βέλτιστη απόδοση είναι κάποιες από τις σημαντικότερες παραμέτρους στο χώρο των ηλεκτρονικών υπολογιστών. Μέσα από αυτή τη διπλωματική εργασία, εστίασα στη λειτουργία των πολυπύρηνων επεξεργαστών και είδα ότι κάθε λύση έχει το δικό της κόστος πάνω στον τομέα απόδοση.

Μέσα από την προσομοίωση των benchmarks, είδα ότι τα αποτελέσματα εξαρτώνται άμεσα από την εφαρμογή. Η κάθε εφαρμογή, είχε το δικό της βαθμό παραλληλισμού και ανταποκρινόταν διαφορετικά με την ύπαρξη πολλών πυρήνων. Με

τη χρησιμοποίηση out-of-order επεξεργαστών, είδαμε ότι το execution time βελτιωνόταν σημαντικά σε σχέση με τους inorder επεξεργαστές.

Τα benchmarks, που παρουσίασαν συνεχή μείωση του execution time με την αύξηση των επεξεργαστών ήταν το *cholesky*, το *radix* και το *barnes*. Το *fft* και το *lu* στους 256 πυρήνες, παρουσιάζουν χειρότερους χρόνους απ' ότι με 128 πυρήνες και δείχνουν να μην μπορούν να διαχειριστούν ένα τόσο μεγάλο αριθμό επεξεργαστών. Στο *ocean*, οι χρόνοι χειροτερεύουν από τους 128 επεξεργαστές δείγμα των μεγάλων καθυστερήσεων που μπορεί να προκύψουν έχοντας τόσο πολλούς πυρήνες.

Το benchmark με το πιο χαμηλό βαθμό παραλληλισμού ήταν το *lu*, αφού το μέγιστο speedup που συναντήσαμε ήταν στους inorder επεξεργαστές με 7,617 χρησιμοποιώντας 256 επεξεργαστές. Τον υψηλότερο βαθμό παραλληλισμού τον συναντήσαμε στο application *barnes*, όπου είχε 109,226 με inorder επεξεργαστές για 128 επεξεργαστές. Επίσης, μεγάλο βαθμό παραλληλισμού είχε και το application *ocean*, όπου είχε 56,584 με inorder επεξεργαστές για 64 επεξεργαστές.

Για κάθε benchmark, η καλύτερη αναλογία speedup με αριθμό επεξεργαστών, είναι η εξής: το *fft* στους 16 επεξεργαστές, το *cholesky* στους 16 επεξεργαστές, το *lu* στους 2 επεξεργαστές, το *radix* στους 32 επεξεργαστές, το *ocean* στους 32 επεξεργαστές και το *barnes* στους 64 επεξεργαστές.

Με την αύξηση του issue, είδαμε σε όλα τα benchmarks, ότι πετύχαμε μείωση του execution time. Όσο αυξάνονταν οι επεξεργαστές, είδαμε ότι οι διαφορές στο execution time μεταξύ των issues δεν ήταν τόσο μεγάλες, χωρίς αυτό να σημαίνει όμως ότι δεν ήταν σημαντικές. Στους λίγους επεξεργαστές, οι διαφορές μεταξύ των issues ήταν αρκετά μεγάλες και μάλιστα στον 1 επεξεργαστή η βελτίωση του χρόνου ξεκινούσε από 20% και έφτανε μέχρι και 50%.

Σε όλα τα configurations, είδαμε ότι όσο μικρότερο ήταν το issue, τόσο μεγαλύτερο ήταν το speedup. Δηλαδή ουσιαστικά, ναι μεν πετυχαίναμε καλύτερο

execution time, στα μεγαλύτερα issues όμως το speedup τους ήταν μικρότερο. Βλέπουμε λοιπόν, ότι υπήρχε μια σύγκρουση μεταξύ της επίδοσης και της αποδοτικότητας, αφού ναι μεν ένα σύνολο επεξεργαστών 32 issue είχαν καλύτερη επίδοση, όμως ένα σύνολο επεξεργαστών 4 issue είχαν καλύτερο speedup και άρα καλύτερη απόδοση. Το ίδιο μπορούμε να πούμε και για τους inorder επεξεργαστές, αφού για όλα τα configurations πέτυχαν το καλύτερο speedup, με τους χειρότερους όμως χρόνους.

Μια ενδιαφέρουσα σύγκριση που έγινε μεταξύ των inorder και των out-of-order επεξεργαστών ήταν αυτή του performance vs die size. Αυτή η σύγκριση, έγινε μέσω του cache byte equivalent area(CBE), όπου είναι το unit area για ένα byte του cache. Οι out-of-order επεξεργαστές μπορεί να έχουν καλύτερες επιδόσεις από τους inorder, όμως απαιτούν μεγαλύτερο power και area και ταυτόχρονα έχουν μεγαλύτερη πολυπλοκότητα. Γι' αυτό, το CBE για έναν inorder επεξεργαστή είναι 50 KB, ενώ για έναν out-of-order επεξεργαστή 250 KB. Έχοντας αυτό σαν κριτήριο, είδαμε ότι σε ορισμένα benchmarks και για ένα συγκεκριμένο chip area, οι inorder επεξεργαστές ήταν πιο αποδοτικοί από τους out-of-order. Παρατηρήσαμε ότι αν έχουμε ένα limitation μέχρι λίγο λιγότερο από τα 250 KB για ένα chip, σίγουρα μας συμφέρει η χρησιμοποίηση inorder επεξεργαστών, αφού το μέγεθος των out-of-order επεξεργαστών ξεκινά από τα 250 KB.

Στο *fft* είδαμε, ότι είχαμε μια εξοικονόμηση του χώρου 20% με τη χρησιμοποίηση inorder επεξεργαστών και ότι από τα 50 KB CBE μέχρι τα 2050 CBE ήταν πιο αποδοτικοί. Στο *cholesky*, είδαμε ότι και εκεί είχαμε μια εξοικονόμηση του χώρου 20% και ότι από τα 50 KB CBE μέχρι τα 800 KB CBE οι inorder επεξεργαστές είναι πιο αποδοτικοί. Στο *lu*, παρατηρήσαμε ότι οι inorder επεξεργαστές ήταν εντελώς ασύμφοροι παρά μόνο στην περίπτωση του limitation που περιγράψαμε πιο πάνω. Το *radix*, ήταν το benchmark με τη μεγαλύτερη αποδοτικότητα, αφού με τη χρησιμοποίηση inorder επεξεργαστών, είχαμε μια εξοικονόμηση χώρου 60% και επίσης ήταν πιο αποδοτικοί στο area μεταξύ 50 KB CBE και 3200 KB CBE, χωρίς όμως να μπορούμε να προσομοιώσουμε τους inorder επεξεργαστές, για μεγαλύτερο die size, λόγω αδυναμίας του συγκεκριμένου benchmark. Στο *ocean* είδαμε, ότι μπορεί με τους inorder επεξεργαστές να είχαμε μια εξοικονόμηση χώρου 20%, όμως μέχρι τα 3200 KB CBE, δε

μπορούσαμε να τους χαρακτηρίσουμε πιο αποδοτικούς αφού κινούνταν στην ίδια καμπύλη με τους out-of-order. Μετά τα 3200 KB CBE οι out-of-order ήταν εμφανώς πιο αποδοτικοί. Τέλος, στο *barnes,* είδαμε όπως και στο lu ότι οι inorder επεξεργαστές ήταν εντελώς ασύμφοροι παρά μόνο στην περίπτωση του limitation που έχει περιγραφεί.
Είναι εύκολο να δούμε, ότι για ένα συγκεκριμένο die size, οι inorder επεξεργαστές ήταν πιο αποδοτικοί, χάρη στο μικρότερο die size και τη χαμηλότερη πολυπλοκότητα.

Γενικά, είδαμε ότι τα αποτελέσματα εξαρτώνται άμεσα από την εφαρμογή που προσομοιωνόταν και το βαθμό παραλληλισμού της. Σίγουρα, μπορεί να μην υπάρχει μια σαφή λύση για βέλτιστη επίδοση και απόδοση, όμως μελετώντας προσεκτικά τις απαιτήσεις και προδιαγραφές του αντικειμένου, μπορεί να βρεθεί μια λύση με το μικρότερο δυνατό κόστος και τη βέλτιστη επίδοση και απόδοση.

6.2 Μελλοντική εργασία

Βλέποντας τη μεγάλη εξάρτηση των πειραματικών αποτελεσμάτων από τα benchmarks τα οποία προσομοιώθηκαν, θα μπορούσε μια μελλοντική εργασία να εστιάσει στο πως οι πολυπύρηνοι επεξεργαστές, θα μπορούσαν να προσαρμοστούν στις ανάγκες μια εφαρμογής.

Αυτό θα μπορούσε να επιτευχθεί με τη χρησιμοποίηση δυναμικών ετερογενών επεξεργαστών. Αυτή η προσέγγιση, θα μπορούσε να παράξει χρήσιμες πληροφορίες, σχετικά με το scheduling, προκειμένου να ικανοποιηθούν οι απαιτήσεις των εφαρμογών σύμφωνα με τις διαθέσιμους πόρους. Οι διαφορετικές πολιτικές του scheduling θα εξεταστούν, με σκοπό να βρεθεί η καλύτερη λύση για απόδοση, αποδοτική ισχύ και θερμικό σχεδιασμό.

Μετά τη διεξαγωγή των πειραμάτων, ήταν εμφανές πως κάθε εφαρμογή είχε τις δικές της απαιτήσεις, με αποτέλεσμα να μην μπορεί να βρεθεί το κατάλληλο configuration που θα μπορούσε να ικανοποιήσεις όλες τις εφαρμογές. Όμως, τα δυναμικά

ετερογενή multicore συστήματα, μπορούν να αλλάξουν τα configurations, σύμφωνα με τις απαιτήσεις των εφαρμογών. Επίσης, θα μπορούσε να μελετηθεί, πως ένα δυναμικό ετερογενές multicore σύστημα, είναι ικανό να αλλάξει όχι μόνο τα configurations του processing, αλλά επίσης και την ιεραρχία της μνήμης του.

Σίγουρα, τα δυναμικά ετερογενή multicore συστήματα, είναι το μέλλον αφού θα είναι άμεσα προσαρμοζόμενα με την εφαρμογή την οποία θα πρέπει να εκτελέσουν [1]. Ένας συνδυασμός πολλών και διαφορετικών επεξεργαστών που ο καθένας θα εκτελεί το δικό του υπολογιστικό κομμάτι, άμεσα προσαρμοσμένο στην εφαρμογή, θα μπορούσε να φτάσει την επίδοση και απόδοση των multicore συστημάτων, σε πολύ υψηλό επίπεδο.

7. References

[1] P. Petrides. Processor Parallelism: From Pipelining to Heterogeneous Multi-core.

[2] R. Varada, M. Sriram, K. Chou and J. Guzoo. (2006). Design and Integration Methods for a Multi-threaded Dual Core 65nm Xeon Processor. Available: www.acm.org

[3] R. Boneti, J. Cazorla, R. Gioiosa, A. Buyuktosunoglu, Ch. Cher and M. Valero. (2008). Software-Controlled Priority Characterization of POWER5 Processor. Available: www.acm.org

[4] T. Mattson, Wijnagaart and M. Rvd. Frumkin. (2007). Programming the Intel 80-core network-on-a-chip Terascale Processor. Available: www.acm.org

[5] L. Seiler et al. (2008) Larrabee: A Many-core x86 Architecture for Visual Computing. Available: www.acm.org

[6] La. Barroso et al. (2000). Piranha: A scalable Architecture Based on Single-Chip Multiprocessing. Available: www.acm.org

[7] K. Barker et al. Entering the Penataflop Era: The Architecture and Performance of Roadrunner. Available: www.acm.org

[8] P. M. Ortego and P. Sack. (2004, December 20). SESC: SuperESCalar Simulator. Available: http://sesc.sourceforge.net/sescdoc.pdf

[9] The M5 Simulator System. Available: http://www.m5sim.org

[10] Stanford Parallel Applications for Shared Memory. Available: http://www-flash.stanford.edu/apps/SPLASH

[11] SESC: cycle accurate architectural simulator. Available: http://sesc.sourceforge.net/index.html

[12] The Modified SPLASH-2 Home Page. Available: http://www.capsl.udel.edu/splash/

[13] Archer: SESCspec Available: http://www.gridappliance.org/wiki/index.php/Archer:SESCspec

[14] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. (1995, June). The SPLASH-2 Programs: Characterization and Methodological Considerations. Available: http://www.csd.uoc.gr/~hy527/papers/splash2_isca.pdf

[15] A. Holloway and M. Allen, "Exploring Core Designs for Chip Multiprocessors".

[16] J. Huh, D. Burger and S. W. Keckler, "Exploring the Design Space of Future CMPs", Computer Architecture and Technology Laboratory, Department of Computer Sciences, The University of Texas at Austin.

[17] John L. Henessy and David A. Patterson, *Computer Architectur a Quantitative Approach*, 4th edition.