ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Power Performance Efficiency of Symmetric Multiprocessors

Μάιος 2011

Ατομική Διπλωματική Εργασία

Power Performance Efficiency of Symmetric Multiprocessors

Στέλιος Πάλλης

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2011

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Power Performance Efficiency of Symmetric Multiprocessors

Στέλιος Πάλλης

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 2011

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή που κ. Pedro Trancoso για την υποστήριξη και καθοδήγηση που μου προσέφερε κατά την διάρκεια της υλοποίησης της παρούσας εργασίας. Μέσα από τις συμβουλές του καθώς και όλες τις γνώσεις του σαν εκπαιδευτικός, κατάφερα να ολοκληρώσω την παρούσα διπλωματική εργασία

Τέλος, θα ήθελα να ευχαριστήσω τους φίλους μου, την οικογένεια μου και ιδιαίτερα του γονείς μου Νίκο και Μαρία, για την αγάπη και την κατανόηση που επέδειξαν όλα αυτά τα χρόνια των σπουδών μου.

Περίληψη

Η κατανάλωση ενέργειας ενός επεξεργαστή ,εκτός από την επίδοση είναι σημαντικό θέμα που προβληματίζει τους σχεδιαστές των σύγχρονων επεξεργαστών. Τα τελευταία χρόνια έχουμε μία μεγάλη αύξηση στην επίδοση των επεξεργαστών, αλλά και μία ανάλογη ή και ακόμη μεγαλύτερη αύξηση στην κατανάλωση ενέργειας των επεξεργαστών.

Με την δεδομένη πλέον στις μέρες μας τη χρήση πολυπύρηνων επεξεργαστών και κατάργηση των μονοπύρηνων επεξεργαστών, στην παρούσα διπλωματική εργασία, προσπάθησα να συγκρίνω κάποια είδη πολυπύρηνων επεξεργαστών με διαφορετικά configurations και να καταλήξω σε κάποια συμπεράσματα σχετικά με την αποδοτικότητα τους σε σχέση με την κατανάλωση ενέργειας και την επίδοση τους. Οι προδιαγραφές που μελέτησα στα πειράματα είχαν να κάνουν με τον αριθμό των πυρήνων, το issue και την συχνότητα του επεξεργαστή, την in-order με out-of-order εκτέλεση και άλλες προδιαγραφές που σχετίζονται με την cache του επεξεργαστή. Οι μετρήσεις έγιναν με την βοήθεια του προσομοιωτή SESC και την σουίτα εφαρμογών SPLASH2.

Μέσα από πληροφορίες που μας δίνει το SESC για τον χρόνο εκτέλεσης και την κατανάλωση ενέργειας θα μετρήσουμε το Power-Performance-Efficiency για 6 benchmarks και θα δούμε ποία configurations επηρεάζουν λιγότερο και ποια περισσότερο το Power-Performance-Efficiency.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Πρόλογος

Η επίδοση ενός επεξεργαστή είναι ένα θέμα το οποίο απασχολεί τους σχεδιαστές από τις αρχές της δημιουργίας ηλεκτρονικών υπολογιστών και έχουν γίνει παρά πολλές έρευνες για αυτό το θέμα. Την τελευταία δεκαετία όμως με την ραγδαία αύξηση των υπολογιστών και της επιστήμης της πληροφορικής ένα άλλο θέμα με το οποίο έχουν καταπιαστεί οι ερευνητές είναι η κατανάλωση ενέργειας ενός επεξεργαστή κάτι εξίσου σημαντικό με την επίδοση αφού ο αριθμός των υπολογιστών και κατεπέκταση των επεξεργαστών έχει αυξηθεί δραματικά.

Η αύξηση της κατανάλωσης ενέργειας οδηγεί σε αύξηση της θερμοκρασίας του επεξεργαστή και σε μεγάλη κατανάλωση ενέργειας για να τρέξει μια εφαρμογή. Η μεγάλη κατανάλωση ενέργειας ενός επεξεργαστή μπορεί να μην είναι μεγάλο πρόβλημα όταν έχουμε ένα επεξεργαστή, όπως έχουμε σε desktop πλατφόρμες. Όταν όμως έχουμε πάρα πολλούς επεξεργαστές όπως έχουμε σε μεγάλους εξυπηρετητές και σε υπερυπολιστές τότε είναι πολύ σημαντικό να έχουμε όσο γίνεται πιο χαμηλή κατανάλωσης ενέργειας. Επίσης σε φορητές πλατφόρμες είναι επίσης πολύ σημαντικό να έχουμε μικρή κατανάλωσης ενέργειας του επεξεργαστή για να έχουμε περισσότερη διάρκεια στην ζωή της μπαταρίας.

Δεδομένη είναι πλέον η κατάργηση των μονολιθικών επεξεργαστών και η αντικατάσταση τους από τους πολυπύρηνους επεξεργαστές και κατεπέκταση την σειριακής επεξεργασία και η αντικατάσταση της από την παράλληλη επεξεργασία. Έτσι, η παρούσα διπλωματική εργασία έχει σαν αντικείμενο την μελέτη των διαφορετικών configuration πολυπύρηνων

επεξεργαστών και εύρεση των configurations που βελτιώνουν το Power-Performance-Efficiency των επεξεργαστών, καθώς και να την μελέτη των configurations ενός επεξεργαστή που επηρεάζουν περισσότερο το efficiency.

1.2 Μεθοδολογία

Στην ατομική διπλωματική μου εργασία γίνεται μία περιγραφή των μέτρων και των σχέσεων που έχουν χρησιμοποιηθεί στην ανάλυση του Power-Performance efficiency.Στην συνέχεια περιγράφεται η λειτουργία του προσομοιωτή που έχει χρησιμοποιηθεί και επίσης περιγράφονται οι εφαρμογές που έχουν χρησιμοποιηθεί για τα πειράματα. Για τις διάφορες μετρήσεις και την προσομοίωση των benchmarks, χρησιμοποίησα τον προσομοιωτή SESC [10] και τη σουίτα εφαρμογών SPLASH2 [11].

Τα configurations που δοκίμασα, είχαν να κάνουν με τον αριθμό των πυρήνων του επεξεργαστή, το issue του επεξεργαστή, την συχνότητα του επεξεργαστή, την in-order με out-of-order εκτέλεση από τους επεξεργαστές, το μέγεθος και το associativity της cache του δευτέρου επιπέδου, τον αριθμό των memory ports και το cache line size. Μέσα από συγκρίσεις των αποτελεσμάτων, προσπαθώ να καταλήξω στο ποια είναι τα ιδανικά configurations για πολυπύρηνους επεξεργαστές, σε θέμα επίδοσης και κατανάλωσης ενέργειας. Ποια είναι δηλαδή τα configurations που θα έχουμε καλύτερη επίδοση με την λιγότερη κατανάλωση ενέργειας. Με άλλα λόγια γίνετε μια ανάλυση του Power-Performance efficiency για να δούμε σε ποια configurations έχουμε τις μεγαλύτερες τιμές στο Power-Performance efficiency.

1.3 Σκοπός Ατομικής Διπλωματικής Εργασίας

Η διπλωματική μου εργασία έχει σαν στόχο την μελέτη της κατανάλωσης ενέργειας σε σχέση με την επίδοση των πολυπύρηνων επεξεργαστών. Στόχος είναι να βρεθούν τα configurations ενός πολυπύρηνου επεξεργαστή που επηρεάζουν περισσότερο και βελτιώνουν το Power-Performance efficiency.

Μέσα από τα πειράματα τα οποία έγιναν και την μέτρηση του Power-Performance efficiency για διαφορετικές εφαρμογές θα δούμε ποία είναι τα καλύτερα configurations σε μία αρχιτεκτονική ενός επεξεργαστή με τα οποία θα έχουμε ένα πολυπύρηνο επεξεργαστή με την χαμηλότερη κατανάλωση ενέργειας και την καλύτερη επίδοση.

Για τον λόγο αυτό τα διαφορετικά configurations τα οποία έχουν μελετηθεί έχουν σχέση με την δομή του επεξεργαστή, την συχνότητα του και διαφορετικές ρυθμίσεις στο δεύτερο επίπεδο της κρυφής μνήμης του επεξεργαστή. Για την καλύτερη και ακριβέστερη συλλογή αποτελεσμάτων χρησιμοποιήθηκαν συνολικά 6 benchmarks από την σουίτα εφαρμογών SPLASH2. Επίσης για τα πειράματα χρησιμοποιήθηκε ο αρκετά έγκυρος και αξιόπιστος προσομοιωτής SESC.

1.4 Δομή Ατομικής Διπλωματικής Εργασίας

Η παρούσα διπλωματική εργασία με θέμα την μελέτη της αποδοτικότητας πολυπύρηνων επεξεργαστών σε θέμα κατανάλωσης ενέργειας και επίδοσης, αποτελείται από 7 κεφάλαια.

Το πρώτο κεφάλαιο είναι η εισαγωγή στην διπλωματική μου εργασία, στο οποίο γίνεται μία περιγραφή της μεθοδολογίας του σκοπού και της δομής της εργασίας αυτής.

Στο δεύτερο κεφάλαιο παρουσιάζεται η σχετική δουλεία που έχει γίνει παλιότερα γύρω από το θέμα της διπλωματικής μου εργασίας

Στο τρίτο κεφάλαιο γίνεται μία επεξήγηση του τι είναι οι προσομοιωτές και πώς λειτουργούν και στην συνέχεια μία εισαγωγή και επεξήγηση του τρόπου λειτουργίας του προσομοιωτή SESC, ο οποίος χρησιμοποιήθηκε για τα πειράματα που έγιναν στα πλαίσια της διπλωματικής μου εργασίας.

Στο τέταρτο κεφάλαιο περιγράφονται τα μέτρα και οι σχέσεις οι οποίες χρησιμοποιήθηκαν για να μετρηθεί το Power-Performance efficiency.

Στο πέμπτο κεφάλαιο γίνεται μία περιγραφή και επεξήγηση του πώς λειτουργούν τα benchmarks τα οποία χρησιμοποιήθηκαν στην εργασία αυτή.

Στο έκτο κεφάλαιο παρουσιάζονται όλα τα πειραματικά αποτελέσματα, γραφικές παραστάσεις, σχολιασμός και αιτιολόγηση των αποτελεσμάτων, καθώς και συμπερασμάτων που προκύπτουν από αυτά.

Στο έβδομο και τελευταίο κεφάλαιο παρουσιάζονται τα συμπεράσματα και η μελλοντική δουλεία που μπορεί να γίνει.

Κεφάλαιο 2

Σχετική Δουλειά

Πολλοί είναι οι ερευνητές που έχουν ασχοληθεί με το θέμα της ανάλυσης του power-performance-efficiency και γενικότερα της κατανάλωσης ενέργειας των επεξεργαστών τα τελευταία χρόνια. Ο βασικό στόχος είναι να βρεθεί η ‘’χρυσή τομή’’ για να έχουμε την καλύτερη επίδοση με την λιγότερη κατανάλωση ενέργειας. Για να γίνουν αυτές οι μελέτες χρησιμοποιήθηκαν πολλές διαφορετικές μέθοδοι σε πολλές και διαφορετικές παραμέτρους ενός επεξεργαστή. Για αυτή την εργασία έχουν μελετηθεί διάφορα άρθρα τα οποία έχουν σχέση με το θέμα της παρούσας εργασίας. Τα άρθρα αυτά αναφέρονται στο τελευταίο τμήμα της έκθεσης αυτής, στην βιβλιογραφία. Στο κομμάτι αυτό αναφέρονται και σχολιάζονται μερικά σημαντικά συμπεράσματα από προηγούμενες εργασίες που έχουν γίνει σχετικά με το θέμα αυτό.

Συγκεκριμένα [1] ανάλυση του power-performance-efficiency στους παραδοσιακούς μονοπύρηνους επεξεργαστές έδειξαν ότι αυτά που επηρεάζουν περισσότερο το efficiency είναι το frequency και το issue width του επεξεργαστή . Το μεν frequency όσο αυξάνεται αυξάνει και την τιμή του efficiency ενώ το issue width αυξάνει το efficiency μέχρι σε ένα σημείο. Επίσης ένα ακόμα σημαντικό συμπέρασμα γύρω από αυτό το θέμα είναι ότι διαφορετικά είδη εφαρμογών έχουν διαφορετική συμπεριφορά όσων αφορά την μέτρηση του efficiency. Με αυτό το πόρισμα οι ερευνητές κατέληξαν σε διάφορους αλγόριθμους, οι οποίοι επιλέγουν τα καλύτερα configurations για ένα συγκεκριμένο είδος εφαρμογής, ώστε να έχουμε το καλύτερο δυνατό power-performance-efficiency από την εκτέλεση ενός συγκεκριμένου τύπου εφαρμογών.

Οι αρχιτεκτονικές επεξεργαστών μελετήθηκαν πάρα πολύ σε θέματα κατανάλωσης ενέργειας και επίδοσης. Σε μία έρευνα,[5] έχει συγκριθεί το energy-delay product για επεξεργαστές χωρίς pipelining, με pipelining και superscalar. Οι έρευνα αυτή έδειξε ότι οι επεξεργαστές χωρίς pipelining υστερούν σε σύγκριση με τις άλλες δύο κατηγορίες επεξεργαστών. Σε μία άλλη έρευνα [6] με στόχο να βρεθούν οι επιπτώσεις στο power-performance-efficiency, του pipelining σε SMT και CMP επεξεργαστές, αυτό το οποίο ισχύει είναι ότι αυξάνοντας τα στάδια του pipelining έχουμε καλύτερο efficiency και στους δύο τύπους επεξεργαστών. Από μία άλλη ερεύνα [7] θέλοντας να μελετηθεί το power-performance-efficiency για asymmetric επεξεργαστές φαίνεται ότι οι asymmetric

επεξεργαστές έχουν καλύτερες προοπτικές από τους smp επεξεργαστές σε θέμα power-performance-efficiency. Στο άρθρο [20], παρουσιάζεται μία έρευνα γύρω από RTL level VLIW-embedded επεξεργαστή, ο οποίος μπορεί να αλλάζει δυναμικά το issue-width του, και τα συμπεράσματα είναι ο συγκεκριμένος επεξεργαστής είναι πολύ καλύτερος σε θέματα κατανάλωσης ενέργειας και επίδοσης.

Διαφορετικά configurations του επεξεργαστή έχουν μελετηθεί από πολλούς ερευνητές. Σε μία έρευνα [22] στην όποια μετρήθηκε η κατανάλωση ενέργειας για διαφορετικά frequencies, τα συμπεράσματα είναι ότι σε συστήματα ψηλών επιδόσεων είναι πιο αποδοτικό να έχουμε ψηλό frequency. Σε μία άλλη έρευνα [19], στην οποία μελετήθηκαν διαφορετικά configurations τα αποτελέσματα δείχνουν ότι σε συστήματα χαμηλής κατανάλωσης ενέργειας σημαντικό ρόλο για το power-performance-efficiency έχει το μέγεθος του pipeliline για floating point εφαρμογές, ενώ για integer εφαρμογές το μέγεθος των integer alu units.

Πολλές είναι οι έρευνες που έγιναν και για την cache. Συγκεκριμένα [2] έρευνα για την L1 data και instruction cache έδειξε ότι η παράμετρος που επηρεάζει περισσότερο το energy-delay efficiency είναι το μέγεθος της. Ακόμα έρευνες [3] που έγιναν γύρω από την οργάνωση των cache έδειξαν ότι ο τρόπος με τον οποίο είναι οργανωμένη η cache είναι πολύ σημαντικός για την κατανάλωση ενέργειας. Η βελτίωση του τρόπου οργάνωσης μίας cache μπορεί να μείωση την κατανάλωση ενέργειας μέχρι και 30%. Τέλος σε μία άλλη έρευνα[4] αυτό το οποίο οι ερευνητές προσπαθούν να εξηγήσουν είναι ότι σημαντικό ρόλο στην κατανάλωση ενέργειας έχει η σειρά με την οποία γίνονται οι εντολές reads και τα writes σε ένα πρόγραμμα. Επίσης η κατανάλωση ενέργειας επηρεάζεται από το data toggle rate.

Τέλος σε μία έρευνα [21], που μελετήθηκε το scheduling σε πολυπύρηνους επεξεργαστές με στόχο την μείωση της θερμοκρασίας του επεξεργαστή, τα αποτελέσματα έδειξαν ότι λόγω ψηλών θερμοκρασιών έχουμε χειρότερες επιδόσεις. Επίσης σε πολυπύρηνους επεξεργαστές η δυνατότητα ψύξης των πυρήνων είναι μικρότερη από τους μονοπύρηνους επεξεργαστές. Για επίλυση αυτών των προβλημάτων οι ερευνητές παρουσιάζουν αλγόριθμους με τους οποίους μπορεί να γίνουν schedule τα processes στους πυρήνες με πιο αποδοτικό τρόπο.

Στην παρούσα ατομική διπλωματική εργασία θα αναλύσουμε το power-performance-efficiency για σημαντικές παραμέτρους smp επεξεργαστών για να δούμε ποίες παράμετροι του επεξεργαστή επηρεάζουν περισσότερο το power-performance-efficiency.

Κεφάλαιο 3

Προσομοιωτής SESC

3.1 Προσομοιωτές Γενικά

Ένας προσομοιωτής αρχιτεκτονικής είναι ένα μεγάλο και περίπλοκο κομμάτι λογισμικού το οποίο μοντελοποιεί μέρη του υπολογιστή όπως ο επεξεργαστής, η μνήμη και μονάδες εισόδου εξόδου ενός υπολογιστή. Οι προσομοιωτές περνούν σαν είσοδο κάποιες παραμέτρους που έχουν σχέση με την αρχιτεκτονική ενός μέρους του υπολογιστή και δοθέντος μίας εφαρμογής έχουν σαν στόχο να προσομοιώσουν την εκτέλεση της εφαρμογής και να μας δώσουν σαν έξοδο ακριβείς και σημαντικές πληροφορίες για την επίδοση, την κατανάλωση ενέργειας και άλλες σημαντικές πληροφορίες που σχετίζονται με τις συγκεκριμένες παραμέτρους και αρχιτεκτονική.

Οι προσομοιωτές έχουν κάποια μεγάλα πλεονεκτήματα και για τον λόγο αυτό χρησιμοποιούνται αρκετά. Το μεγαλύτερο πλεονέκτημα είναι ότι μπορεί να γίνει αξιολόγηση του υλικού χωρίς να χρειάζεται να κατασκευαστεί γεγονός που είναι οικονομικά πολύ συμφέρων και ακόμα μπορούμε να έχουμε πρόσβαση σε μέρη ενός υπολογιστή που δεν υπάρχουν καν. Επίσης οι προσομοιωτές παρέχουν πολύ λεπτομερή στοιχεία για διάφορα θέματα όπως την επίδοση, την κατανάλωση ενέργειας κ.α.

Δύο σημαντικά θέματα τα οποία είναι πολύ σημαντικό να λαμβάνουν υπόψη οι προσομοιωτές είναι η αξιοπιστία και η ταχύτητα. Όσο περισσότερο αξιόπιστα είναι τα αποτελέσματα ενός προσομοιωτή τόσο πιο αξιόπιστες θα είναι οι έρευνες. Ακόμα δεδομένου ότι η εκτέλεση μιας εφαρμογής γίνεται πολύ πιο γρήγορα σε πραγματική μηχανή, είναι καλό

οι προσομοιωτές να είναι όσο πιο γρήγοροι γίνεται για να εξοικονομείτε πολύτιμος χρόνος σε μία έρευνα.

Υπάρχουν δύο κατηγορίες προσομοιωτών σε σχέση με τον τρόπο που προσομοιώνουν την εκτέλεση μίας εφαρμογής.
Trace-driven: Αυτού του τύπου οι προσομοιωτές χρησιμοποιούν τα trace instructions των εφαρμογών για να προσομοιώσουν την εκτέλεση.
Execution-driven: Οι προσομοιωτές αυτοί εκτελούν πραγματικά τις εντολές της εφαρμογής με ένα εξομοιωτή, κατευθείαν από το binary της εφαρμογής. Οι execution-driven προσομοιωτές χρησιμοποιούνται περισσότερο γιατί είναι πιο αξιόπιστοι.

3.2 Προσομοιωτής WATTCH

Ο προσομοιωτής WATTCH [9], ο οποίος έχει χρησιμοποιηθεί παλιότερα για μία εργασία, είναι ένας αρκετά αξιόπιστος προσομοιωτής. Το power model του προσομοιωτή SESC είναι βασισμένο στον προσομοιωτή αυτό.

Το WATTCH είναι ένας προσομοιωτής που προσομοιώνει την συμπεριφορά μια μηχανής και εκτιμά την κατανάλωση ενέργειας του επεξεργαστή. Το μοντέλο του εργαλείου αυτού έχει υλοποιηθεί σε SimpleScalar [23] architectural simulator. Το SimpleScalar architectural simulator είναι ένας προσομοιωτής που κάνει γρήγορη και ακριβή προσομοίωση επεξεργαστών SimpleScalar αρχιτεκτονικής, η οποία είναι μία αρχιτεκτονική που μοιάζει πολύ με την MIPS αρχιτεκτονική. Το μοντέλο ενέργειας του WATTCH είναι βασισμένο στο CACTI ένα cache optimization tool. Σύμφωνα με τον David Brooks το ποσοστό λάθους του προσομοιωτή αυτού δεν ξεπερνά το 10%, κάτι που τον κάνει αρκετά ακριβή προσομοιωτή.

3.3 Εισαγωγή στον Προσομοιωτή SESC

Ο προσομοιωτής ο οποίος χρησιμοποίησα για την εκτέλεση των πειραμάτων ήταν ο SESC [10]. Ο προσομοιωτής αυτός είναι αρκετά γρήγορος και ακριβές και επίσης μπορεί να παρέχει πληροφορίες για την κατανάλωση ενέργειας και την επίδοση του επεξεργαστή και έτσι κρίθηκε κατάλληλος για να τον χρησιμοποιήσω.

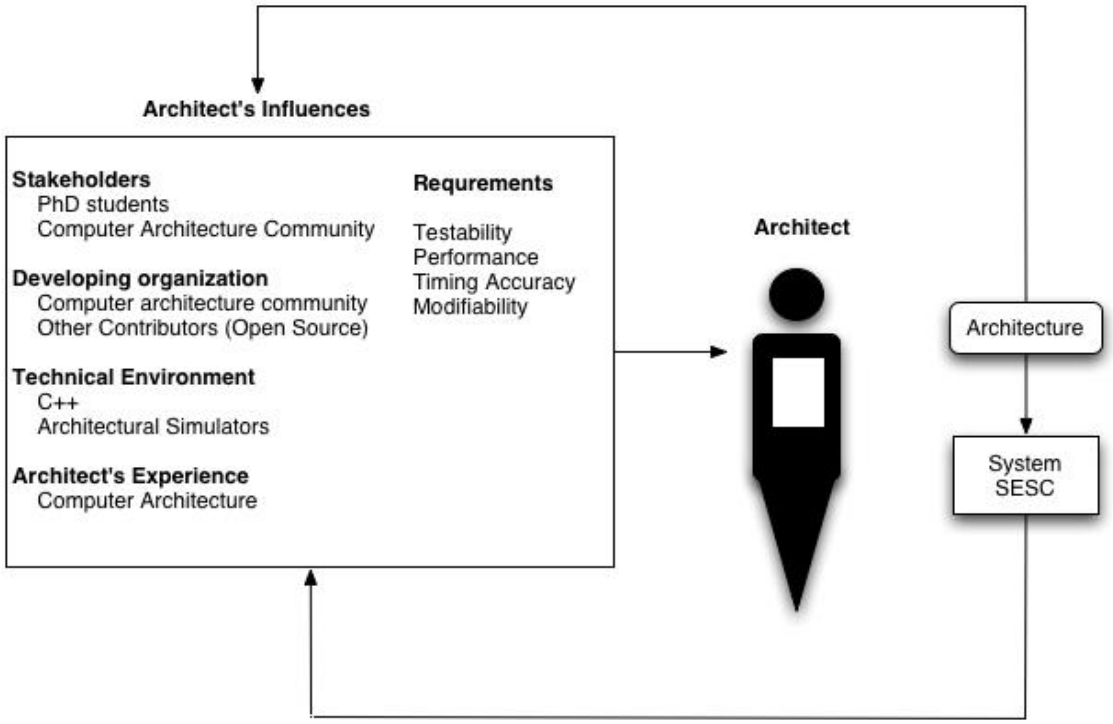
Ο SESC είναι ένας προσομοιωτής γραμμένος σε c++ και open source. Αναπτύχθηκε κυρίως από την ομάδα ερευνών i-acoma στο πανεπιστήμιο του Illinois στο Urbana-Champaign αλλά και από άλλες ομάδες σε διάφορα άλλα πανεπιστήμια. Μοντελοποιεί πολλές διαφορετικές

αρχιτεκτονικές επεξεργαστών όπως είναι οι single processors, οι chip multiprocessors και οι processors-in-memory. Επίσης ο SESC μοντελοποιεί όλα τα συστατικά που απαιτεί ένας συγχρόνως επεξεργαστής όπως ένα πλήρες out-of-order pipeline με branch prediction, caches, buses κα., κάνοντας τον έτσι πιο ακριβή στις προσομοιώσεις.

Η διαθεσιμότητα και η διαμόρφωση του πηγαίου κώδικα του SESC, τον κάνουν ένα σπουδαίο εργαλείο για την αξιολόγηση ερευνητικών προτάσεων. Ο προσομοιωτής αυτός έχει πολλά πλεονεκτήματα σε σχέση με άλλους προσομοιωτές για τον λόγο ότι αναπαριστά πολλές αρχιτεκτονικές όπως dynamic superscalar processors, CMPs, processor-in-memory και multithreading αρχιτεκτονικές. Ακόμα έναν άλλο πλεονέκτημα του προσομοιωτή είναι η ταχύτητα του. Είναι ικανός να εκτελεί 1,5 εκατομμύρια εντολές MIPS το δευτερόλεπτο σε επεξεργαστή Intel Pentium 4 στα 3GHz.

Ο SESC είναι event-driven προσομοιωτής δηλ. η προσομοίωση καθοδηγείται από events π.χ. ένα function call. Ο εξομοιωτής ο οποίος χρησιμοποιεί ο SESC έχει αναπτυχθεί από το MINT και εξομοιώνει έναν επεξεργαστή MIPS. Σε κάθε κύκλο του επεξεργαστή καλούνται πολλές λειτουργίες του πυρήνα του προσομοιωτή, ενώ άλλες λειτουργίες καλούνται μόνο όταν πρέπει, με τα events.

Πιο κάτω γίνεται μια περιγραφή του κύκλου λειτουργίας του προσομοιωτή SESC όπως αυτή φαίνετε στο σχήμα 3.1. Αυτοί οι οποίοι λοιπόν επηρεάζουν την αρχιτεκτονική είναι οι Stakeholders, η οργάνωση της ανάπτυξης, το τεχνικό περιβάλλον, η πείρα για την αρχιτεκτονική και τέλος οι απαιτήσεις. Στους Stakeholders μπορούμε να συναντήσουμε έρευνες μαθητών διδακτορικού και γενικά έρευνες στην αρχιτεκτονική υπολογιστών. Στην οργάνωση της ανάπτυξης βρίσκουμε τον βασικό δημιουργό του SESC που είναι ο Jose Renau και κάποιοι άλλοι μαθητές της ομάδας ερευνών i-acoma. Επειδή όμως όπως έχω αναφέρει, ο κώδικας του SESC είναι open source, ο καθένας μπορεί να συμβάλει στην ανάπτυξη και βελτίωση του προσομοιωτή. Στο τεχνικό περιβάλλον του SESC είναι η c++ , στην οποία έχει γραφτεί. Ο λόγος που έχει επιλεχθεί η γλώσσα αυτή είναι ότι είναι πιο γρήγορη από την java και υποστηρίζει αντικειμενοστραφή προγραμματισμό. Ο SESC μπορεί να τρέξει σε πολλά Unix συστήματα και σε big-endian και little-endian επεξεργαστές. Στην κατηγορία της πείρας έχουμε τους βασικούς δημιουργούς που ήταν εξοικειωμένοι με την χρήση και δημιουργία προσομοιωτών αρχιτεκτονικής .Τέλος στις απαιτήσεις συναντούμε ορολογίες όπως, επίδοση, ακριβείς χρονικές μετρήσεις και επεκτασιμότητα, πράγματα τα οποία χαρακτηρίζουν τον SESC.



Σχήμα 3.1*: Σχήμα στο οποίο φαίνετε ο κύκλος δημιουργίας του SESC.*

3.4 Superscalar Out-Of Order pipeline

Το superscalar out-of-order pipeline είναι μία τεχνική η οποία μοντελοποιείται στην προσομοίωση από τον προσομοιωτή SESC. Είναι σημαντικό να κατανοήσουμε την λειτουργία αυτής της τεχνικής, η οποία χρησιμοποιείται από τους σύγχρονους επεξεργαστές για την εκτέλεση των εντολών.

Καταρχάς, το pipelining είναι μια τεχνική των επεξεργαστών, που επιτρέπει σε πολλές εντολές να επικαλυφθούν, ενώ αυτές εκτελούνται στον επεξεργαστή. Η τεχνική αυτή δημιουργήθηκε, αρκετό καιρό πριν, όταν οι ερευνητές συνειδητοποίησαν ότι καμία εντολή δεν χρησιμοποιούσε τον επεξεργαστή όλη την ώρα. Ανακάλυψαν λοιπόν ότι ήταν δυνατό για έναν επεξεργαστή να χρησιμοποιεί ταυτόχρονα περισσότερες από μία εντολές, κάτι το οποίο επιτυγχάνεται με το pipelining. Υπάρχουν όμως αρκετές δυσκολίες και προκλήσεις σε αυτή την τεχνική όπως exception handling, branch misprediction κ.α. Παρόλα αυτά όμως το pipelining χρησιμοποιείται σε μεγάλο βαθμό στους σύγχρονους επεξεργαστές.

Με αυτή την τεχνική μπορούμε να υποθέσουμε ότι ένας επεξεργαστής σε κάθε στάδιο του pipeline χρειαζόταν έναν κύκλο μηχανής. Σε αυτή την περίπτωση όμως ο επεξεργαστής είναι σαν μια ουρά αναμονής. Η πρώτη εντολή που προσκομίζεται είναι η πρώτη εντολή που δεσμεύεται. Ακόμα για παράδειγμα η εκτέλεση ενός πολλαπλασιασμού κινητής υποδιαστολής θα πάρει περισσότερο χρόνο από την σύγκριση ενός ακεραίου με το μηδέν, κάτι το οποίο προκαλεί καθυστερήσεις στο pipelining. Τα προβλήματα αυτά λύθηκαν με την υποστήριξη multicycle εντολών. Ο επεξεργαστής δεν ήταν πλέον μια ουρά αναμονής αφού μία εντολή που προσκομίζεται σε χρόνο t μπορεί να τελειώσει πριν από μία εντολή που προσκομίζεται σε χρόνο t-1. Οι επεξεργαστές με αυτή την τεχνική ονομάζονται out-of-order επεξεργαστές και για να έχουν επιτυχία πρέπει να λύσουν προβλήματα που δημιουργούνται κατά τον χειρισμό των εξαιρέσεων.

Σε τέλειες συνθήκες, μπορούμε να υποθέσουμε, ότι ένας out-of-order επεξεργαστής μπορεί να εκτελεί μία εντολή κάθε κύκλο μηχανής. Για να βελτιωθεί περισσότερο η απόδοση μπορεί να επιτραπεί η εκτέλεση πολλών εντολών σε ένα κύκλο μηχανής, γεγονός που μπορεί να επιτευχτεί με δύο τρόπους.

Ο πρώτος τρόπος είναι με πολύ μεγάλες instruction word architectures (VLIW). Ο επεξεργαστής με αυτή την μέθοδο κάνει issue έναν σταθερό αριθμό εντολών ανά κύκλο. Ο δεύτερος τρόπος που είναι αυτός που μας ενδιαφέρει είναι με superscalar επεξεργαστές. Οι επεξεργαστές αυτού του τύπου προσπαθούν να κάνουν issue περισσότερες από μία εντολές κάθε κύκλο μηχανής 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. Τεχνικές όπως static και dynamic scheduling χρησιμοποιούνται για να μεγιστοποιηθεί ο αριθμός των εντολών που γίνονται issue.

3.5 Διαχείριση Προσομοίωσης από τον SESC

Στον SESC οι εντολές εκτελούνται σε ένα module εξομοίωσης, που εξομοιώνει το MIPS Instruction Set Architecture (ISA). Ουσιαστικά 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 στην 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 pipeline.

Ο τρόπος που έχει περιγραφεί πιο πάνω, μπορεί να αιτιολογηθεί στο ότι είναι πολύ γρηγορότερο να εκτελεστεί μια εντολή σε έναν απλό εξομοιωτή και επίσης πολύ πιο εύκολο για το πρόγραμμα και την αποσφαλμάτωση 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 αποδεχτό.

3.6 Η Cache στον SESC

Σε ένα τυπικό μοντέρνο επεξεργαστή, υπάρχει το πρώτο επίπεδο cache για της εντολές, η I-cache και το πρώτο επίπεδο cache για τα δεδομένα, η D-Cache. Αυτές αναφέρονται και αλλιώς σαν L1 caches. Σε πιο χαμηλό επίπεδο βρίσκεται η πιο μεγάλη, πιο αργή L2 cache που αποτελεί το δεύτερο επίπεδο της cache. Σε πολλούς επεξεργαστές υπάρχει και τρίτο επίπεδο cache , η L3 cache που είναι πιο μεγάλη και πιο αργή από την L2 cache. Η cache έχει πολλές παραμέτρους. Μερικές από αυτές οι οποίες μοντελοποιούνται στον SESC είναι τα Sizes, τα Hit & Miss latencies, τα Replacement policies, τα Cache-line sizes τα associativities κ.α.

Η υλοποίηση της cache στον SESC είναι αρκετά περίπλοκη και χρησιμοποιεί πολλά event-driven callbacks. Αυτό είναι απαραίτητο για να μοντελοποιηθούν όλες οι καθυστερήσεις και οι συναλλαγές που γίνονται στην cache. Για παράδειγμα ένα read miss στην L1 cache μπορεί να προκαλέσει ένα dirty cache line στην L1 που θα πρέπει να γραφτεί στην L2, και μόνο όταν αυτό γίνει να μπορεί το L1 miss να σταλεί στην L2. Ακολούθως ένα read miss στην L2 πρέπει να ζητήσει access στο bus , μετά να το στείλει στην μνήμη και ούτω καθεξής.

Ο GProcessor (generic processor) είναι αντικείμενο το οποίο έχει σχέση με λειτουργίες οι οποίες εκτελούνται από τον επεξεργαστή. Κάθε ένα λοιπόν από αυτά τα αντικείμενα έχει ένα MemorySystem αντικείμενο, το οποίο δημιουργεί την ιεραρχία της cache και το οποίο δημιουργεί τον τρόπο με τον οποίο το GProcessor επικοινωνεί με το υψηλότερα επίπεδα της

cache. Το MemorySystem αντικείμενο δημιουργείται από το GProcessor όταν έχει την ανάγκη να επικοινωνήσει με το σύστημα της μνήμης. Υπάρχουν DMemoryRequest αντικείμενα και IMemoryRequests για δεδομένα και εντολές αντίστοιχα.

Το σημαντικότερο είναι ότι όλοι οι τύποι cache και των buses κληρονομούν από μία κοινή κλάση η οποία ορίζει ένα κοινό interface για της σημαντικότερες λειτουργίες της. Αυτό το κοινό interface επιτρέπει μία fully-configurable ιεραρχία στη cache. Ακόμα κάθε τύπος cache μπορεί να έχει διαφορετικό τρόπο να χειρίζεται τα requests ,φτάνει να λαμβάνει υπόψη τα interface της cache του ψηλότερου και χαμηλότερου επιπέδου.

3.7 Το power model του SESC

Ο SESC έχει την δυνατότητα να ρυθμιστεί με το power model και να μας δίνει σημαντικές πληροφορίες για την κατανάλωση ενέργειας του επεξεργαστή. Αυτές οι πληροφορίες όπως η ολική κατανάλωση ενέργειας (total power consumption) είναι απαραίτητες για την ολοκλήρωση της διπλωματικής μου εργασίας. Επίσης ο SESC έχει την δυνατότητα να ρυθμιστεί και με thermal model και να μας δίνει σημαντικές πληροφορίες για την θερμοκρασία του επεξεργαστή.

Το power model του SESC είναι βασισμένο στο SIM-WATTCH όπως έχει αναφερθεί.

Το SIM-WATTCH [9] μετρά την κατανάλωση ενέργειας με την σχέση Pd = CV ^2αf.
Το C είναι το load capacitance και υπολογίζεται στην προσομοίωση βάση του circuit και των transistor sizings. Το V και το f είναι το voltage και το frequency του επεξεργαστή και εξαρτώνται από την τεχνολογία του επεξεργαστή. Τέλος το α είναι μία τιμή από το 0 μέχρι 1 και υπολογίζεται από το πόσο συχνά τα clock ticks οδηγούν σε switching activity κατά μέσο όρο.

Κεφάλαιο 4

Μέτρα και Σχέση

4.1 Εισαγωγή

Για να γίνει μία καλή και σωστή μέτρηση του Power-Performance efficiency πολύ σημαντικό ρόλο σε αυτό, παίζει η σχέση με την οποία θα συνδυαστούν οι δύο αυτές έννοιες, της κατανάλωσης ενέργειας και της επίδοσης ενός επεξεργαστή. Στην σχέση αυτή βρίσκεται το κλειδί στο να αναλύσουμε σωστά το Power-Performance efficiency.

Η πιο κοινή και ίσως προφανές σχέση για να χαρακτηρίσουμε το Power-Performance efficiency ενός επεξεργαστή είναι μία απλή αναλογία, όπως MIPS (million instruction per second) δηλαδή πόσα εκατομμύρια εντολές εκτελούνται το δευτερόλεπτο διά watts(W) τα οποία καταναλώνονται MIPS/W. Αυτή η σχέση μετρά το efficiency προβάλλοντας την επίδοση που επιτυγχάνεται για κάθε watt το οποίο καταναλώνεται. Όσο πιο μεγάλος είναι ο αριθμός σε αυτή τη σχέση τόσο μεγαλύτερη η αποδοτικότητα της μηχανής.

Από την άλλη όμως υπάρχει και η ακραία περίπτωση όπου η κατανάλωση ενέργειας είναι ο πρωταρχικός μας σκοπός, όταν για παράδειγμα έχουμε φορητές πλατφόρμες η οποίες έχουν μπαταρία και θέλουμε η μπαταρία να έχει μεγαλύτερη διάρκεια ζωής. Σε αυτή την περίπτωση περισσότερο βάρος πρέπει να δώσουμε στην κατανάλωση ενέργειας, και η επίδοση να πάει σε δεύτερη μοίρα. Η σχέση MIPS/W ίσως να μην είναι η κατάλληλη και να χρειάζεται να την αλλάξουμε δίνοντας περισσότερο βάρος στην κατανάλωση ενέργειας.

Για μεγαλύτερα και πιο αποδοτικά συστήματα όπως είναι τα workstations περισσότερο βάρος χρειάζεται να δώσουμε στην επίδοση. Η σχέση (MIPS)^2/W περιγράφει με καλύτερο τρόπο το Power-Performance efficiency για αυτές της μηχανές. Για ακόμη μεγαλύτερα και πιο αποδοτικά συστήματα όπως είναι σε εξυπηρετητές η σχέση για το Power-Performance

efficiency πρέπει να δώσει ακόμη περισσότερη βαρύτητα στην επίδοση. Μια καλή σχέση ίσως να ήταν (MIPS)^3/W.

4.2 Επιλογή Σχέσης για Power-Performance efficiency

Για την μέτρηση του το Power-Performance efficiency η σχέση η οποία επιλέχτηκε να χρησιμοποιηθεί είναι η εξής:

Eff (s,a)=(MIPS(s,a))^3/W(s,a).

Όπου:

MIPS: Είναι τα εκατομμύρια εντολές που εκτελούνται το δευτερόλεπτο.

W: Η ολική κατανάλωση ενέργειας.

s: Συμβολίζει το σύστημα το οποίο διαφέρει σε μόνο μία παράμετρο από το s-baseline.

a: Συμβολίζει τις διαφορετικές εφαρμογές.

Η σχέση αυτή φαίνεται να δίνει πολύ μεγάλο βάρος στην επίδοση παρά στην κατανάλωση ενέργειας, όμως κάτι τέτοιο δεν ισχύει. Οι σύγχρονοι πολυπύρηνοι επεξεργαστές έχουν πολύ πιο μεγάλες επιδόσεις σε σχέση με τους παλιούς μονολιθικός επεξεργαστές. Με δεδομένη λοιπόν την χρήση ενός επεξεργαστή με 8 πυρήνες για baseline, η επίδοση ενός τέτοιου επεξεργαστή είναι αρκετά υψηλή και επιβάλλετε η χρήση μίας τέτοιας εξίσωσης για την μέτρηση του Power-Performance efficiency. Επίσης τα configurations του επεξεργαστή που έχουν επιλεγεί για baseline βασίζονται σε υπάρχον επεξεργαστή ο οποίος προορίζεται για μεγάλα συστήματα, όπως για εξυπηρετητές. Σύμφωνα λοιπόν με τον David M. Brooks [8] πιο κατάλληλη σχέση για να μετρήσουμε το Power-Performance efficiency σε τέτοιου είδους επεξεργαστές είναι η σχέση η οποία έχει επιλεχθεί.

Κεφάλαιο 5

Benchmarks

5.1 Επιλογή των benchmarks

Σημαντικό ρόλο στο να γίνει μία αξιόπιστη μέτρηση του Power-Performance efficiency, από την οποία θα έχουμε σωστά και αξιόπιστα αποτελέσματα, έχουν τα benchmarks που θα δώσουμε ως είσοδο στον προσομοιωτή. Για τον λόγο αυτό προτιμήθηκε να χρησιμοποιηθούν benchmarks από έτοιμη σουίτα εφαρμογών, τα οποία είναι δοκιμασμένα και αξιόπιστα. Επίσης αυτά τα benchmarks χρησιμοποιούν τον επεξεργαστή και την μνήμη με διαφορετικό τρόπο, κάτι το οποίο θέλουμε να μελετήσουμε. Να δούμε δηλαδή τι συμβαίνει όταν έχουμε διαφορετικού τύπου εφαρμογές. Για τα πειράματα λοιπόν χρησιμοποιήθηκαν benchmarks από την σουίτα εφαρμογών SPLASH2.

5.2 Η σουίτα εφαρμογών SPLASH2

Η σουίτα εφαρμογών SPLASH2 αποτελεί μία δεύτερη αναβαθμισμένη έκδοση της σουίτας εφαρμογών SPLASH. Έχει αναπτυχθεί στο Πανεπιστήμιο του Stanford και περιέχει παραλληλοποιημένα προγράμματα για Shared-Memory επεξεργαστές. Η σουίτα αυτή αποτελείται από 4 kernels: fft, colesky, lu, radix, κάτι το οποίο δεν υπήρχε στην προηγούμενη σουίτα, την SPLASH. Επίσης στην SPLASH2 υπάρχουν και 8 εφαρμογές : barnes, fmm, ocean, radiosity, raytrace, volrend, water-nsquared, water-spatial. Από αυτά έγινε μία επιλογή τριών kernels και τριών εφαρμογών για τα πειράματα τα οποία θα εκτελούνταν. Συγκεκριμένα επιλέχτηκαν τα fft, lu, radix και τα barnes, fmm, ocean.

5.3 Barnes

Το Barnes είναι μία εφαρμογή [16] η οποία προσομοιώνει ένα σύστημα από οργανισμούς σε τρείς διαστάσεις με έναν αριθμό από time-steps, χρησιμοποιώντας την μέθοδο Barnes-Hut hierarchical N-body. Στην εφαρμογή γίνεται μία αναπαράσταση της υπολογιστικής περιοχής με την δομή δεδομένων octree. Τα φύλλα περιέχουν τις πληροφορίες για κάθε σώμα και οι εσωτερικοί κόμβοι 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 την διανομή μορίων και είναι μη δεδομένη. Στην κύρια μνήμη, 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.

5.4 FMM

Το FMM είναι μία εφαρμογή [14] που όπως και το barnes προσομοιώνει ένα σύστημα από οργανισμούς με έναν αριθμό από time-steps. Ωστόσο, προσομοιώνει αλληλεπιδράσεις σε δύο διαστάσεις χρησιμοποιώντας μία διαφορετική μέθοδο, την adaptive Fast Multipole Method. Όπως και στο barnes οι κύριες δομές δεδομένων είναι τα body και τα tree cells, με πολλά particles σε κάθε φύλλο. Το FMM διαφέρει σε δύο σημαντικούς λόγους από το barnes: (ι) το δέντρο δεν είναι προσπελάσιμο once per body, αλλά είναι μόνο για πάνω ή κάτω πέρασμα (κάθε timestep) που υπολογίζει τις αλληλεπιδράσεις μεταξύ των κυττάρων και διαδίδει τις επιπτώσεις αυτών στα σώματα ,(ιι) η ακρίβεια δεν ελέγχεται από το πόσα κύτταρα έχει ένα σώμα η τα κύτταρα που αλληλεπιδρούν αλλά από το πόσο ακριβείς είναι η μοντελοποίηση μίας αλληλεπίδρασης. Στην κύρια μνήμη, δε γίνεται καμία προσπάθεια, ώστε να υπάρχει έξυπνη κατανομή από δεδομένα οργανισμών.

5.5 Ocean

Το ocean είναι μία εφαρμογή [15] η οποία εξομοιώνει τις μετακινήσεις, μεγάλης κλίμακας, του ωκεανού με βάση τους στροβίλους και τα όρια των ρεμάτων . Οι κύριες διαφορές με την αντίστοιχη εφαρμογή του SPLASH είναι:(i) κάνει partition τα grids σε τετράγωνα παρά σε γκρούπς με στήλες για να βελτιώσει την επικοινωνία σε αναλογία υπολογισμού, (ii) τα grids εννοιολογικά εκπροσωπούνται από πίνακες τεσσάρων διαστάσεων, με όλα τα subgrids να δεσμεύονται συνεχόμενα και τοπικά στα nodes που ανήκουν και (iii) χρησιμοποιεί μία

red-black Gauss-Seidel multigrid equation solver σε αντίθεση με την SOR solver.

5.6 FFT

Το FFT kernel [12] είναι μια σύνθετη μονοδιάστατη έκδοση 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 root 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 τρεία 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.

5.7 LU

Το lu kernel [17] είναι ένα πρόγραμμα το οποίο παραγοντοποιεί ένα πλέγμα στο γινόμενο ενός χαμηλότερου τριγωνικού και ενός υψηλότερου τριγωνικού πλέγματος. Το πυκνό nxn πλέγμα A, διαιρείται σε έναν NxN πίνακα από BxB blocks (ν=NB), ώστε να εκμεταλλευτεί τη χρονική τοποθεσία των submatrix στοιχείων. Για να μειωθεί η επικοινωνία, το block ownership ορίζεται να χρησιμοποιεί ένα δυσδιάστατο scatter αποσύνθεσης, με τα blocks να ενημερώνονται από τούς 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 να διατηρήσει την καλή ισορροπία των φορτίων. Τα αρκετά μικρά μεγέθη του block 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.

5.8 Radix

Το radix kernel [13] είναι ένα πρόγραμμα το οποίο υλοποιεί radix ταξινόμηση ακεραίων αριθμών. Ο αλγόριθμος που υλοποιεί είναι επαναληπτικός και εκτελεί 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 μέσο των writes, παρά μέσω των reads.

Στον πιο κάτω πίνακα φαίνονται σημαντικά στοιχεία για κάθε εφαρμογή [11].

Πίνακας 5.1*: Ενδεικτικά αποτελέσματα από εκτέλεση των benchmarks σε επεξεργαστή με 32 πυρήνες.*

Code	Problem Size	Total Instr (M)	Total FLOPS (M)	Total Reads (M)	Total Writes (M)	Shared Reads (M)	Shared Writes (M)
Barnes	16K particles	2002.79	239.24	406.85	313.29	225.05	93.23
FMM	16K particles	1250.02	423.88	226.23	38.58	217.84	30.10
Ocean	258 X258 ocean	379.93	101.54	81.89	18.93	80.26	17.27
FFT	64K points	34.79	6.36	4.07	2.88	4.05	2.87
LU	512x512 matrix,16 x 16 blocks	494.05	92.20	104.00	48.00	93.20	44.74
Radix	1M integers, radix 1024	50.99		12.06	7.03	7.03	10

Κεφάλαιο 6

Πειραματικά Αποτελέσματα

6.1 Εισαγωγή

Όπως έχει ήδη αναφερθεί ο προσομοιωτής ο οποίος επιλέχθηκε για να γίνουν τα πειράματα είναι ο SESC. Στο κεφάλαιο 3 γίνεται περιγραφή του τρόπου λειτουργίας του προσομοιωτή αυτού.

Ο SESC εγκαταστάθηκε σε λειτουργικό σύστημα Ubuntu 9.10. Η μηχανή στην οποία έγινε build ο SESC έχει επεξεργαστή τον Intel I7 920 με 4 core και 8 threads στα 2.66 GHz και 6GB main memory. Επίσης έγινε εγκατάσταση των Sescutils τα οποία είναι εργαλεία που χρειάζονται για να γίνει cross-compile ένα πρόγραμμα και να μπορεί να τρέξει στον προσομοιωτή.

Τα benchmarks τα οποία επιλέχθηκαν, όπως έχει ήδη αναφέρει, είναι από την σουίτα εφαρμογών SPLASH2, για να έχουμε πιο έγκυρα και αξιόπιστα αποτελέσματα. Επιλέχτηκαν λοιπόν 6 συνολικά benchmarks, 3 εφαρμογές και 3 kernels, barnes, fmm, ocean και fft, lu, radix αντίστοιχα. Επιλέξαμε τα 6 αυτά benchmarks για τον λόγο ότι με μερικά από τα άλλα αντιμετωπίσαμε πρόβλημα με την εκτέλεση τους. Επίσης τα 6 αυτά benchmarks καλύπτουν όλο το εύρος αριθμού εντολών των εφαρμογών που υπάρχουν στην σουίτα εφαρμογών SPLASH2. Στο κεφάλαιο 4 γίνεται περιγραφή του κάθε benchmark ξεχωριστά.

6.2 Επιλογή baseline και configurations για το πείραμα

Η επιλογή των baseline configurations του επεξεργαστή, δηλαδή των configurations που θα κρατούσαμε σταθερά σε κάθε πείραμα αλλάζοντας μόνο κάθε φορά το configuration που θα θέλαμε να ελέγξουμε, έγινε βάση υπάρχων επεξεργαστή. Συγκεκριμένα ο επεξεργαστής αυτός είναι ο Intel Xeon της οικογένειας επεξεργαστών ‘’Beckton’’. Ο επεξεργαστής αυτός προορίζεται για συστήματα υψηλών επιδόσεων, όπως για εξυπηρετητές. Στον πίνακα 6.1 φαίνονται οι ακριβείς ρυθμίσεις για baseline.

Η αρχιτεκτονική που θα χρησιμοποιούταν στην προσομοίωση ήταν η MIPS Instruction Set Architecture (ISA). Η αρχιτεκτονική η οποία εφαρμόζεται στον πολυπύρηνο επεξεργαστή είναι η SMP (symmetric multiprocessing) και στην περίπτωση αυτή οι πυρήνες του επεξεργαστή συμπεριφέρονται σαν διαφορετικοί επεξεργαστές.

Τα configurations του επεξεργαστή τα οποία εξετάστηκαν στα πειράματα είναι αριθμός των πυρήνων του επεξεργαστή, το issue του επεξεργαστή, το frequency του επεξεργαστή, In Order σε σύγκριση με την Out of Order εκτέλεση, το cache line size, το μέγεθος και το associativity της L2 cache και ο αριθμός των memory ports. Ο αριθμός των πυρήνων του επεξεργαστή κρίθηκε αναγκαίο να εξετασθεί για να δούμε πως επηρεάζεται το Power-Performance efficiency όταν αυξάνουμε τους πυρήνες του επεξεργαστή. Το issue του επεξεργαστή, που σύμφωνα με το configuration file του SESC σχετίζεται με πολλές παραμέτρους του επεξεργαστή, κρίθηκε επίσης αναγκαίο να μελετηθεί στα πειράματα για τον λόγο ότι αλλάζοντας το αλλάζει όλη η δομή του επεξεργαστή. Το frequency του επεξεργαστή, που στην περίπτωση αυτή είναι το ίδιο για όλους τους πυρήνες, είναι μία αρκετά σημαντική παράμετρος που ελέγχτηκε στα πειράματα για να δούμε κατά πόσο είναι καλό να έχουμε ψηλό ή χαμηλό frequency για να έχουμε το καλύτερο Power-Performance efficiency. Στα πειράματα μετρήσαμε ακόμα το Power-Performance efficiency για In-order εκτέλεση και Out-of-order εκτέλεση για να συγκριθούν οι δύο αυτοί τρόποι εκτέλεσης. Σε θέμα μνήμης σημαντικό configuration είναι το cache line size, το οποίο ισοδυναμεί με το block size όλων των επιπέδων της cache. Τέλος εξετάσθηκαν το μέγεθος και το associativity της L2 cache, καθώς και ο αριθμός των memory ports για να δούμε πως και αυτά τα configurations της cache επηρεάζουν θετικά ή αρνητικά το Power-Performance efficiency.

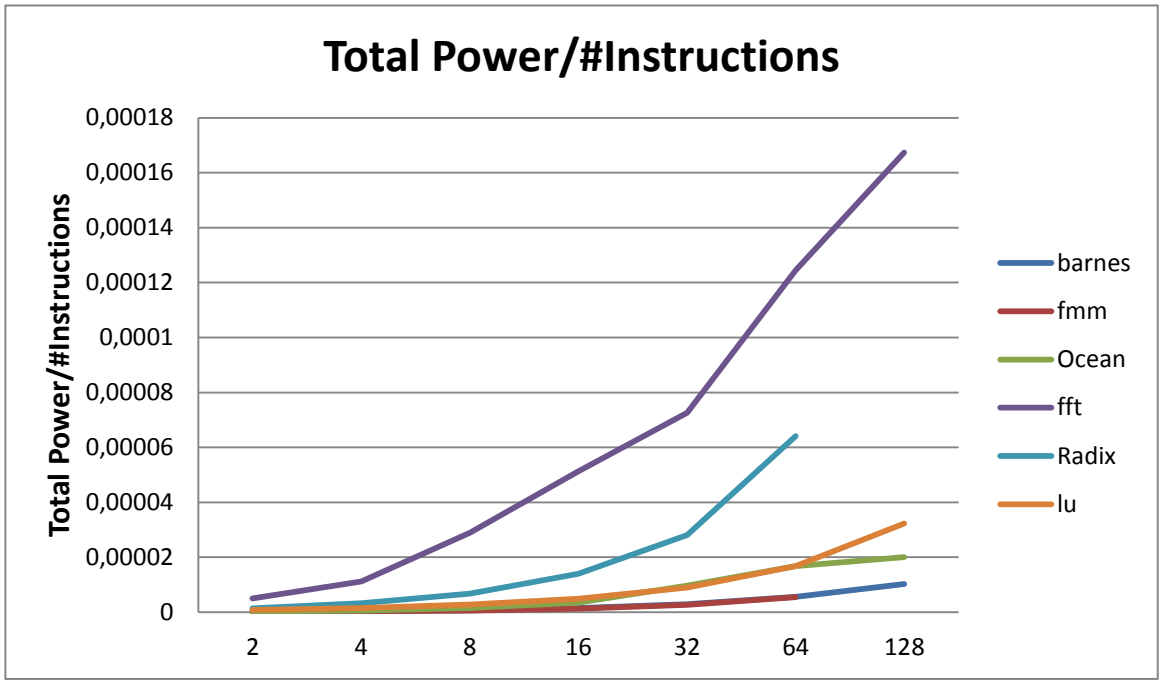
Στο πίνακα 6.1 φαίνονται οι ακριβείς ρυθμίσεις των configurations που εξετάσθηκαν στα πειράματα καθώς και το διάστημα των τιμών που πήραν τα συγκεκριμένα configurations.

Πίνακας 6.1*: Baseline ρυθμίσεις του επεξεργαστή και το εύρος των ρυθμίσεων προς εξέταση.*

	Baseline	**Range**
Frequency	2Ghz	0.5-3.5Ghz
Cores	8	2-128
Technology	45nm	-
CacheLineSize	32	8-64
L1 I-Cache / L1 D-Cache	32KB 2-way, Bsize= CacheLineSize, 1 cycle hit	-
L2 Cache	512KB 8-way, Bsize= CacheLineSize, 12 cycle hit	256KB-16MB, 1-64way,
Memory Bus width	8B	-
Memory Ports	2	1-32
Int -FP units	$(issue)/3+1 - $(issue)/2+1	
Execution	Out-Of-Order	In-order
issue	8	1-32
fetchWidth =$(issue) instQueueSize =2*$(issue) issueWidth =$(issue) retireWidth =$(issue)+1 areaFactor =($(issue)*$(issue)+0.1)/16		

6.3 Power-Performance efficiency για τον αριθμό των πυρήνων του επεξεργαστή

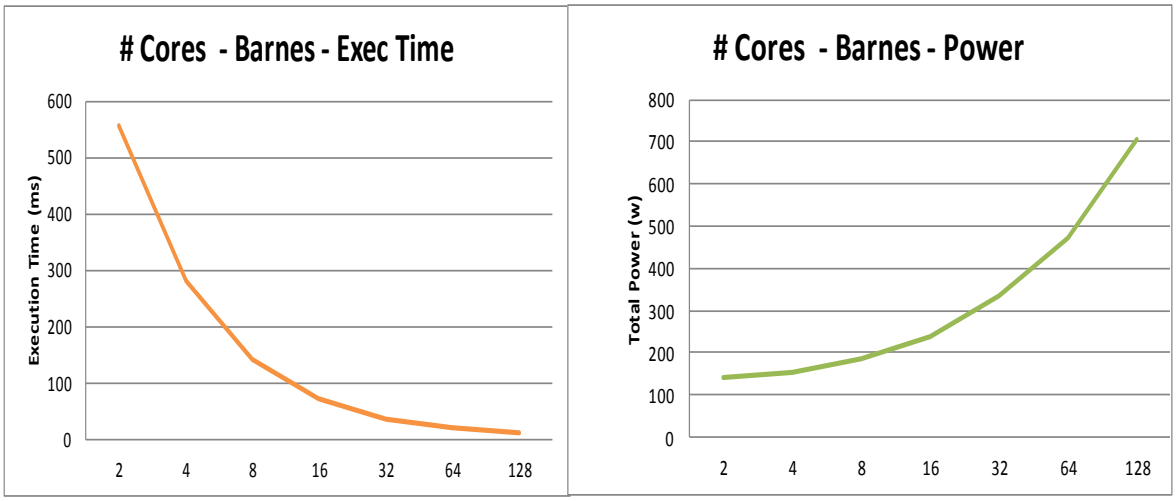
Το σημαντικότερο configuration του επεξεργαστή που έχει εξετασθεί, αποτελεί ο αριθμός των πυρήνων του επεξεργαστή. Για να δούμε πως η αύξηση στον αριθμό των πυρήνων του επεξεργαστή επηρεάζει το Power-Performance efficiency κρατήσαμε σταθερά τα υπόλοιπα configuration στο baseline και αλλάξαμε τον αριθμό των πυρήνων από 2 μέχρι 128 και για τα 6 benchmarks που έχουν επιλεχθεί. Για τις εφαρμογές FMM και Radix τα πειράματα έγιναν για μέχρι 64 πυρήνες λόγω αδυναμίας των εφαρμογών να εκτελεστούν με περισσότερους πυρήνες.



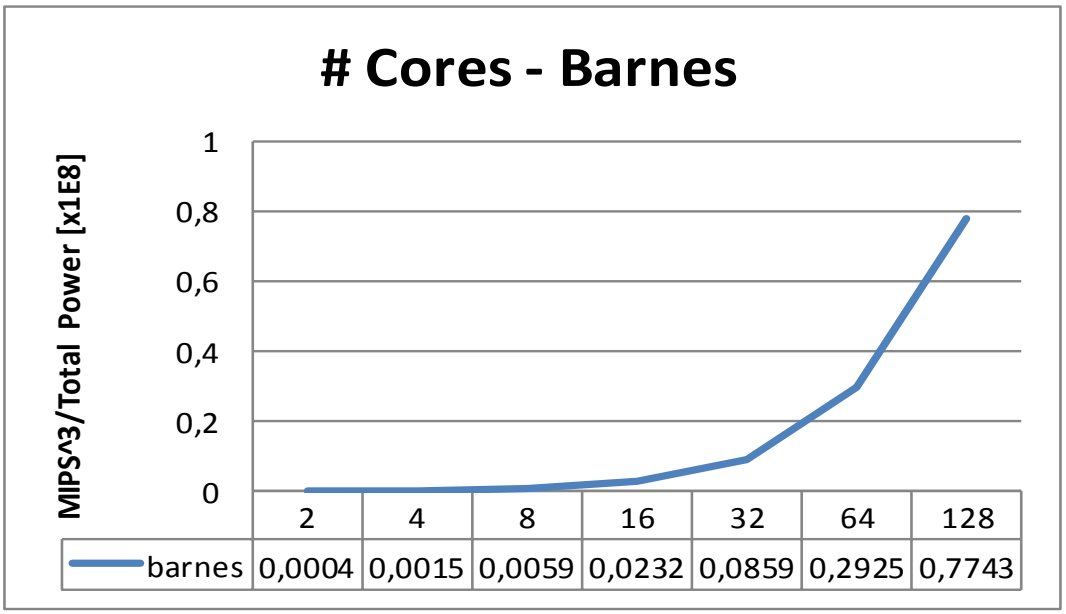
Σχήμα 6.1: *Γραφική παράσταση που αναπαριστά την ολική κατανάλωση ενέργειας διά τον αριθμό των εντολών για το Barnes, το FMM , το Ocean, το FFT, το Radix και το Lu για μέχρι και 128 πυρήνες*

Η πιο πάνω γραφική παράσταση μας δείχνει την ολική κατανάλωση ενέργειας δια τον αριθμό των εντολών για κάθε εφαρμογή από 2 μέχρι 128 πυρήνες. Από την πιο πάνω γραφική παράσταση μπορούμε να συμπεράνουμε ότι η εφαρμογή η οποία καταπονεί περισσότερο τον επεξεργαστή σε θέμα κατανάλωσης ενέργειας είναι η FFT και λιγότερο η εφαρμογή Barnes. Οι εφαρμογές αυτές έχουν τον χαμηλότερο αριθμό εντολών και τον ψηλότερο αντίστοιχα. Μία εφαρμογή η οποία έχει μεγαλύτερο αριθμό εντολών από μία άλλη μπορεί να έχει μεγαλύτερη ολική κατανάλωση από μία άλλη, αλλά όταν έχουμε πολλούς πυρήνες σε ένα

επεξεργαστή η εφαρμογή η οποία έχει μεγαλύτερο αριθμό εντολών θα παραλληλοποιηθεί σε καλύτερο βαθμό με αποτέλεσμα οι πολλοί πυρήνες να μένουν απασχολημένοι και να μην ξοδεύεται ενέργεια άσκοπα. Αυτός είναι και ο λόγος που παρατηρούνται αυτά τα αποτελέσματα στην πιο πάνω γραφική παράσταση, δηλαδή εφαρμογές με λιγότερο αριθμό εντολών τείνουν να έχουν μεγαλύτερα ποσοστά κατανάλωσης ενέργειας σε σχέση με τον αριθμό εντολών τους από εφαρμογές οι οποίες έχουν περισσότερες εντολές.



Σχήμα 6.2: *Γραφική παράσταση που αναπαριστά τους χρόνους εκτέλεσης και την κατανάλωση ενέργειας για το Barnes για μέχρι και 128 πυρήνες.*



Σχήμα 6.3: *Γραφική παράσταση που αναπαριστά το power-performance efficiency για το Barnes για μέχρι και 128 πυρήνες.*

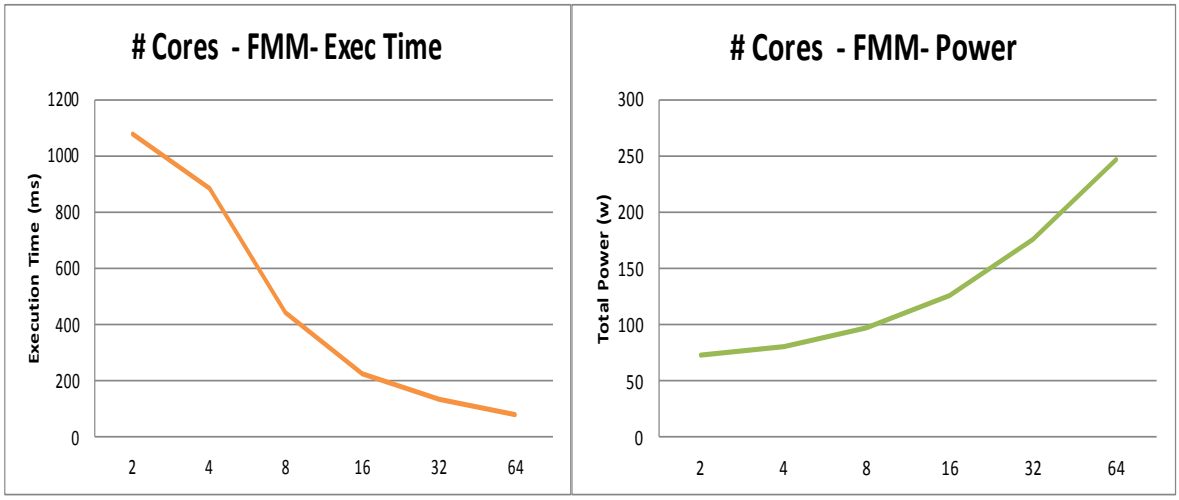
Από την πιο πάνω γραφική παράσταση, σχήμα 6.3, παρατηρείται ότι όσο αυξάνουμε τον αριθμό των επεξεργαστών αυξάνεται και το Power-Performance efficiency για την εφαρμογή Barnes. Συγκεκριμένα αυξάνοντας με εκθετικό ρυθμό τους πυρήνες του επεξεργαστή έχουμε ανάλογη εκθετική αύξηση στο Power-Performance efficiency.

Σε θέμα κατανάλωσης ενέργειας, σχήμα 6.2, αυξάνοντας των αριθμό των πυρήνων του επεξεργαστή με εκθετικό ρυθμό παρατηρείται αρκετά μεγάλη αύξηση στην ολική κατανάλωση ενέργειας. Αυτό επηρεάζει αρνητικά το Power-Performance efficiency.

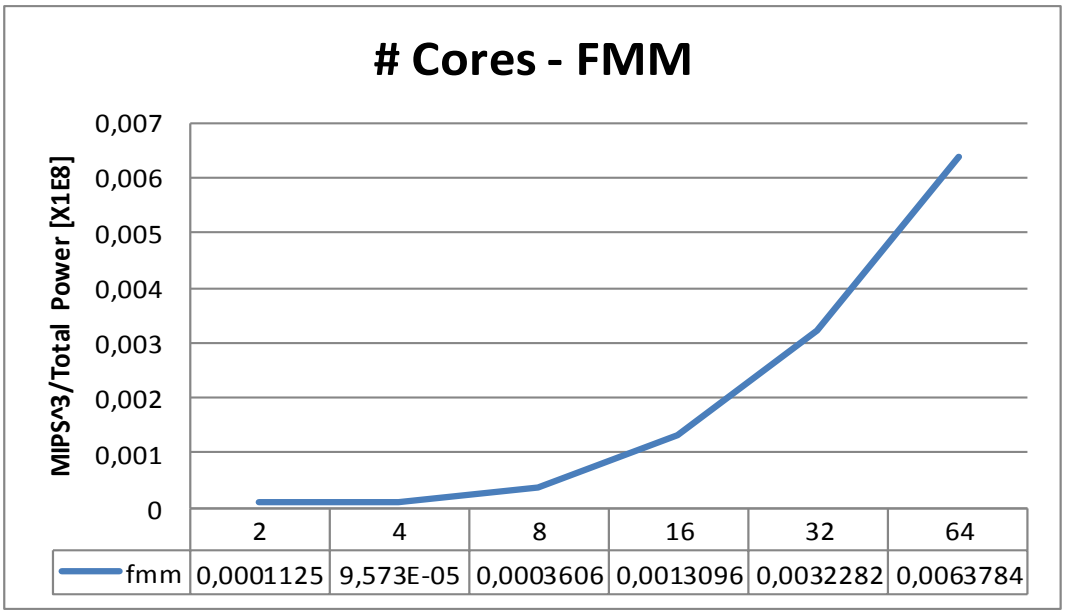
Σε θέμα επίδοσης, σχήμα 6.2, όσο αυξάνουμε τον αριθμό των πυρήνων του επεξεργαστή παρατηρείται μία μείωση στους χρόνους εκτέλεσης με εκθετικό ρυθμό. Οι χρόνοι εκτέλεσης της εφαρμογής Barnes συνεχίζουν να μειώνονται με μεγάλο ρυθμό για μέχρι και 128 πυρήνες γεγονός που οφείλεται στη δομή της και στο ότι η συγκεκριμένη εφαρμογή έχει τον μεγαλύτερο αριθμό εντολών από τις άλλες και έτσι μπορεί να παραλληλοποιηθεί σε μεγάλο βαθμό. Επίσης το speedup της εφαρμογής αυτής όσο αυξάνονται οι πυρήνες είναι πολύ κοντά στο ιδανικό [11].

Σαν αποτέλεσμα έχουμε μεγάλη μείωση του Power-Performance efficiency. Συγκεκριμένα από το baseline configuration στους 8 πυρήνες με χρόνο εκτέλεσης 140.82 ms, μέχρι και τον μεγαλύτερο αριθμό πυρήνων του πειράματος, τους 128 πυρήνες με χρόνο εκτέλεσης 11.51 ms , έχουμε Power-Performance efficiency 0.0059 και 0.77 αντίστοιχα. Παρατηρείται δηλαδή μία αύξηση στο Power-Performance efficiency 130,5 φορές. Στο κατώτατο όριο που ήταν οι 2 πυρήνες ο χρόνος εκτέλεσης ήταν 557,04 ms, και η τιμή του Power-Performance efficiency είναι 0.0004. Από αυτό το σημείο μέχρι και το baseline configuration στους 8 πυρήνες παρατηρείται αύξηση στο Power-Performance efficiency 14.75 φορές.

Ο μέσος όρος αύξησης του Power-Performance efficiency συγκριτικά, ανάμεσα σε κάθε αύξηση στον αριθμό των πυρήνων είναι 3.55 φορές. Με άλλα λόγια αυξάνοντας των αριθμό των πυρήνων στο διπλάσιο έχουμε 3.55 φορές μεγαλύτερο Power-Performance efficiency. Επίσης το σημείο στο οποίο έχουμε συγκριτικά την μεγαλύτερη αύξηση του Power-Performance efficiency είναι από τους 4 πυρήνες σε 8 πυρήνες 3.93 φορές καλύτερο στους 8 πυρήνες, ενώ το σημείο με την χειρότερη συγκριτικά αύξηση είναι από τους 64 πυρήνες σε 128 πυρήνες 2.66 φορές καλύτερο στους 128 πυρήνες. Το γεγονός αυτό οφείλετε στην πολύ μεγάλη ολική κατανάλωση ενέργειας που παρατηρείται στο σημείο των 128 πυρήνων.



Σχήμα 6.4: *Γραφική παράσταση που αναπαριστά τους χρόνους εκτέλεσης και την κατανάλωση ενέργειας για το fmm για μέχρι και 64 πυρήνες λόγω αδυναμίας της εφαρμογής να τρέξει για 128 πυρήνες.*



Σχήμα 6.5: *Γραφική παράσταση που αναπαριστά το power-performance efficiency για το fmm για μέχρι και 64 πυρήνες λόγω αδυναμίας της εφαρμογής να τρέξει για 128 πυρήνες.*

Από την πιο πάνω γραφική παράσταση, σχήμα 6.5, παρατηρείται ότι όσο αυξάνουμε τον αριθμό των επεξεργαστών αυξάνεται και το Power-Performance efficiency για την εφαρμογή FMM. Συγκεκριμένα αυξάνοντας με εκθετικό ρυθμό τους πυρήνες του επεξεργαστή έχουμε ανάλογη εκθετική αύξηση στο Power-Performance efficiency.

Σε θέμα κατανάλωσης ενέργειας, σχήμα 6.4, αυξάνοντας των αριθμό των πυρήνων του επεξεργαστή με εκθετικό ρυθμό παρατηρείται μία μεγάλη αύξηση της ολικής κατανάλωσης

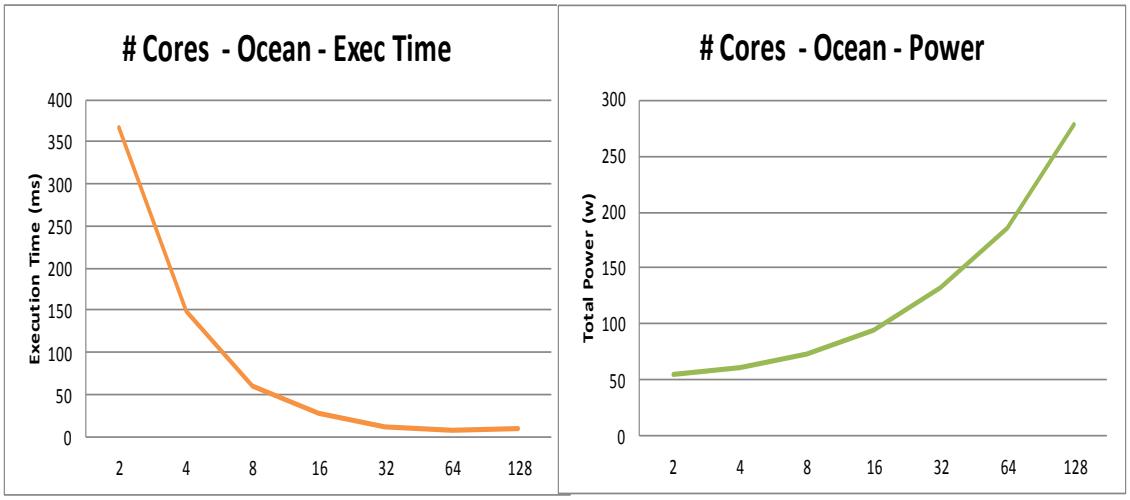
ενέργειας. Για την εφαρμογή αυτή έχουμε μικρότερη αύξηση στην ολική κατανάλωση ενέργειας απ' ότι για την εφαρμογή Barnes.

Σε θέμα επίδοσης, σχήμα 6.4, όσο αυξάνουμε τον αριθμό των πυρήνων του επεξεργαστή παρατηρείται μία μείωση στους χρόνους εκτέλεσης με εκθετικό ρυθμό. Οι χρόνοι εκτέλεσης της εφαρμογής FMM συνεχίζουν να μειώνονται με μεγάλο ρυθμό για μέχρι και 64 πυρήνες που είναι ο μέγιστος αριθμός πυρήνων με τους οποίους μπορούμε να εκτελέσουμε την εφαρμογή. Η εφαρμογή αυτή είναι η δεύτερη εφαρμογή σε αριθμό εντολών, από όλες τις άλλες. Επίσης η εφαρμογή FMM χρησιμοποιεί τον επεξεργαστή σε αρκετά μεγάλο βαθμό. Η μείωση των χρόνων εκτέλεσης λοιπόν, όσο αυξάνουμε τον αριθμό των επεξεργαστών είναι δικαιολογημένη γιατί εκτός από την μεγάλη υπολογιστική ισχύ που χρειάζεται η εφαρμογή, λόγω της δομής της μπορεί και να παραλληλοποιηθεί σε μεγάλο βαθμό.

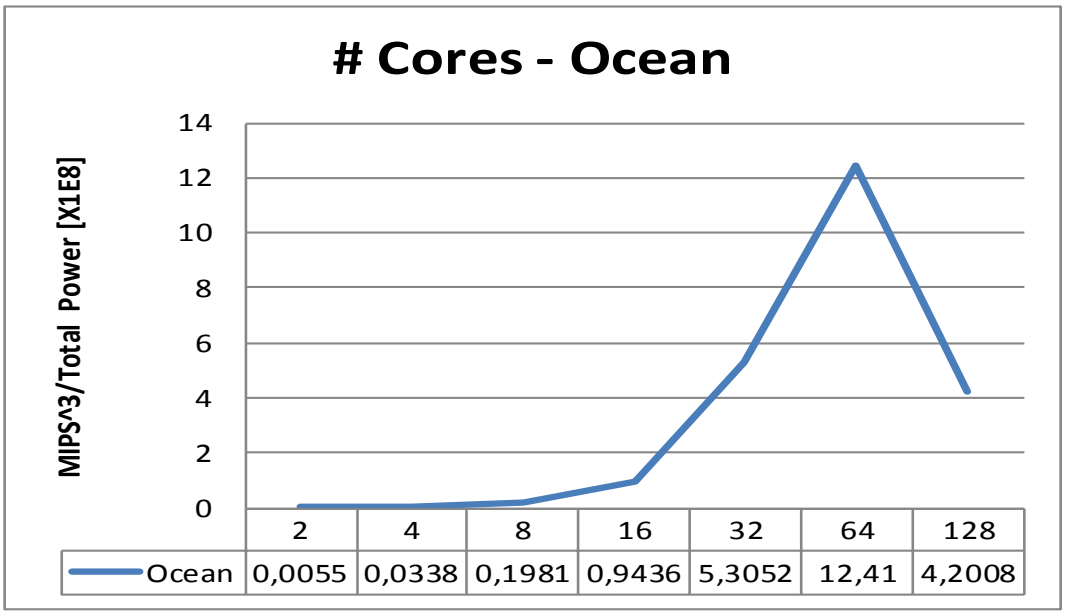
Στο σημείο από 2 πυρήνες μέχρι 4 πυρήνες παρατηρείται ότι το Power-Performance efficiency μειώνεται. Το γεγονός αυτό οφείλετε στην μη ικανοποιητική βελτίωση της επίδοσης. Οι χρόνοι εκτέλεσης της εφαρμογής είναι 1073.74 για 2 πυρήνες και 881.65 για 4 πυρήνες αντίστοιχα. Στην συνέχεια το Power-Performance efficiency αυξάνεται με μεγάλο ρυθμό όπως έχει αναφερθεί.

Συγκεκριμένα από το baseline configuration στους 8 πυρήνες με χρόνο εκτέλεσης 443.69 ms, μέχρι και τον μεγαλύτερο αριθμό πυρήνων του πειράματος, τους 64 πυρήνες με χρόνο εκτέλεσης 82.17 ms , έχουμε Power-Performance efficiency 0.00036 και 0.0063 αντίστοιχα. Παρατηρείται δηλαδή μία αύξηση στο Power-Performance efficiency 17,5 φορές. Στο κατώτατο όριο που ήταν οι 2 πυρήνες η τιμή του Power-Performance efficiency είναι 0.00011. Από αυτό το σημείο μέχρι και το baseline configuration στους 8 πυρήνες παρατηρείται αύξηση στο Power-Performance efficiency 3.27 φορές.

Ο μέσος όρος αύξησης του Power-Performance efficiency συγκριτικά, ανάμεσα σε κάθε αύξηση στον αριθμό των πυρήνων είναι 2.53 φορές, δηλαδή αυξάνοντας των αριθμό των πυρήνων στο διπλάσιο έχουμε 2.53 φορές μεγαλύτερο Power-Performance efficiency. Επίσης το σημείο στο οποίο έχουμε συγκριτικά την μεγαλύτερη αύξηση του Power-Performance efficiency είναι και πάλι από τους 4 πυρήνες σε 8 πυρήνες 3.78 φορές καλύτερο στους 8 πυρήνες. Στο σημείο από τους 2 πυρήνες σε 4 πυρήνες παρατηρείται μείωση του Power-Performance efficiency 0.86 φορές χειρότερο στους 4 πυρήνες.



Σχήμα 6.6: Γραφική παράσταση που αναπαριστά τους χρόνους εκτέλεσης και την κατανάλωση ενέργειας για το Ocean για μέχρι και 128 πυρήνες.



***Σχήμα 6.7**: Γραφική παράσταση που αναπαριστά το power-performance efficiency για το Ocean για μέχρι και 128 πυρήνες.*

Από την πιο πάνω γραφική παράσταση, σχήμα 6.7, παρατηρείται ότι όσο αυξάνουμε τον αριθμό των επεξεργαστών αυξάνεται και το Power-Performance efficiency για την εφαρμογή Ocean μέχρι και το σημείο των 64 πυρήνων σε αντίθεση με τις δύο προηγούμενες εφαρμογές. Στην συνέχεια για 128 πυρήνες παρατηρείται μία μεγάλη μείωση του Power-Performance efficiency.

Σε θέμα κατανάλωσης ενέργειας, σχήμα 6.6, αυξάνοντας των αριθμό των πυρήνων του επεξεργαστή με εκθετικό ρυθμό παρατηρείται μία μεγάλη αύξηση της ολικής κατανάλωσης ενέργειας. Η ολική κατανάλωση ενέργειας αυξάνεται με μεγάλο ρυθμό μέχρι το σημείο των

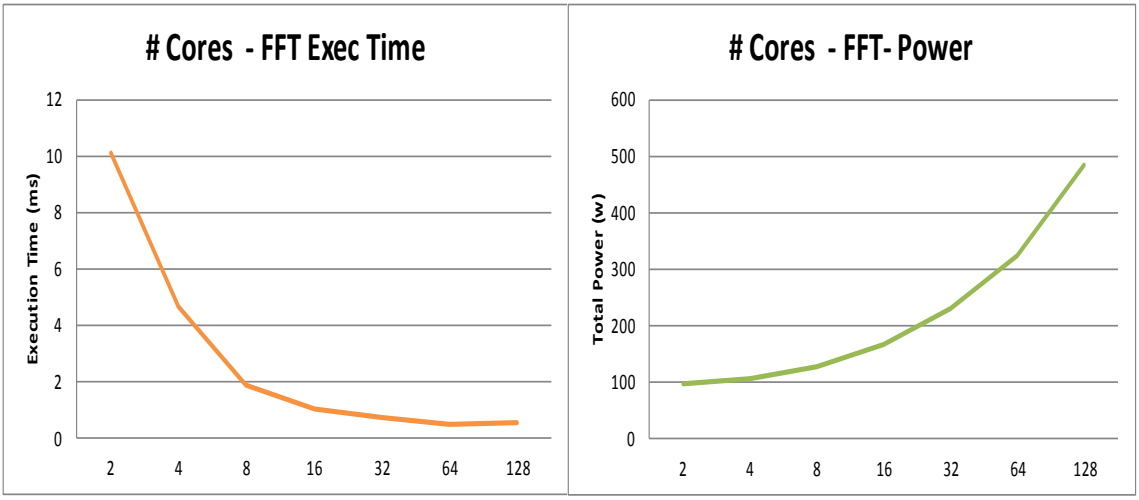
64 πυρήνων και ακολούθως για 128 πυρήνες η ολική κατανάλωση ενέργειας δεν αυξάνεται τόσο. Αυτό είναι αντίθετο με το Power-Performance efficiency, το οποίο παρατηρείται να μειώνεται σημαντικά στο σημείο των 128 πυρήνων.

Σε θέμα επίδοσης, σχήμα 6.6, όσο αυξάνουμε τον αριθμό των πυρήνων του επεξεργαστή παρατηρείται μία μείωση στους χρόνους εκτέλεσης με εκθετικό ρυθμό. Οι χρόνοι εκτέλεσης της εφαρμογής Ocean συνεχίζουν να μειώνονται με μεγάλο ρυθμό για μέχρι και 64 πυρήνες . Στην συνέχεια όμως για 128 πυρήνες παρατηρείται αύξηση του χρόνου εκτέλεσης από 6.87 ms για 64 πυρήνες, σε 9.3 ms για 128 πυρήνες γεγονός το οποίο δικαιολογεί την πτώση του Power-Performance efficiency. Αυτό οφείλετε στο γεγονός ότι η εφαρμογή Ocean δεν μπορεί να παραλληλοποιηθεί περισσότερο λόγω της δομής της εφαρμογής. Με άλλα λόγια ο χρόνος που χρειάζεται για να παραλληλοποιηθεί η εφαρμογή για 128 πυρήνες, το overhead αυτό δηλαδή κοστίζει περισσότερο από το να τρέχαμε την εφαρμογή σε λιγότερους πυρήνες.

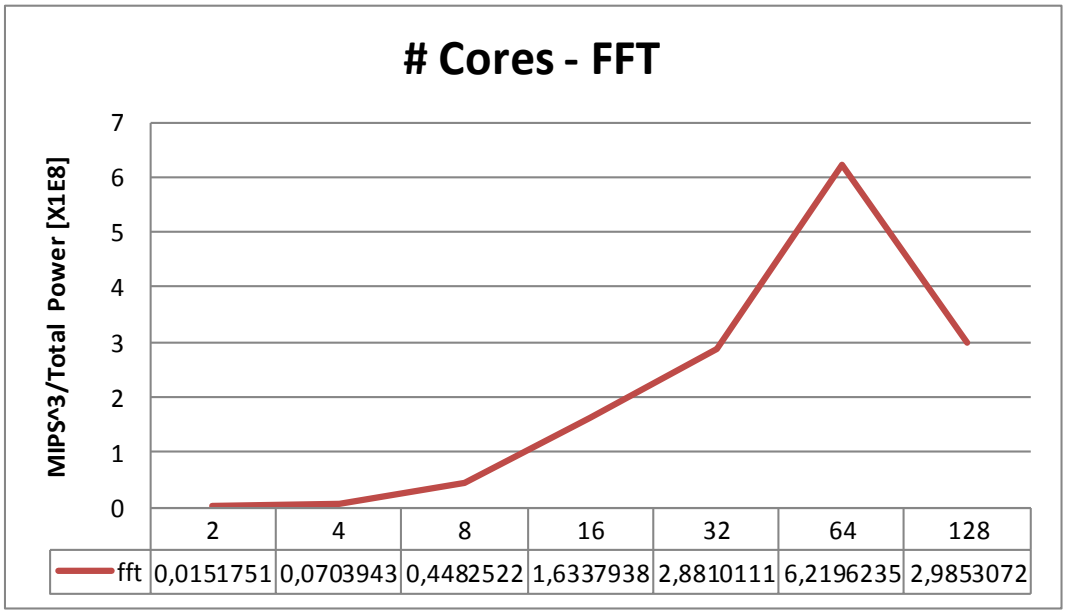
Στο σημείο λοιπόν από 64 πυρήνες μέχρι 128 πυρήνες παρατηρείται ότι το Power-Performance efficiency μειώνεται από 12.41 σε 4.2, 0.33 φορές. Από 2 πυρήνες μέχρι και 64 πυρήνες παρατηρείται αύξηση του Power-Performance efficiency.

Συγκεκριμένα από το baseline configuration στους 8 πυρήνες με χρόνο εκτέλεσης 59.5 ms, μέχρι και τον αριθμό πυρήνων του πειράματος με το καλύτερο Power-Performance efficiency , τους 64 πυρήνες έχουμε Power-Performance efficiency 0.19 και 12.41 αντίστοιχα. Παρατηρείται δηλαδή μία αύξηση στο Power-Performance efficiency 65,3 φορές. Στο κατώτατο όριο που ήταν οι 2 πυρήνες η τιμή του Power-Performance efficiency είναι 0.0055. Από αυτό το σημείο μέχρι και το baseline configuration στους 8 πυρήνες παρατηρείται αύξηση στο Power-Performance efficiency 34.5 φορές.

Ο μέσος όρος αύξησης του Power-Performance efficiency συγκριτικά, ανάμεσα σε κάθε αύξηση στον αριθμό των πυρήνων είναι 4.16 φορές, δηλαδή αυξάνοντας των αριθμό των πυρήνων στο διπλάσιο έχουμε 4.16 φορές μεγαλύτερο Power-Performance efficiency. Επίσης το σημείο στο οποίο έχουμε συγκριτικά την μεγαλύτερη αύξηση του Power-Performance efficiency είναι από τους 2 πυρήνες σε 4 πυρήνες 6 φορές καλύτερο στους 8 πυρήνες.



***Σχήμα 6.8**: Γραφική παράσταση που αναπαριστά τους χρόνους εκτέλεσης και την κατανάλωση ενέργειας για το fft για μέχρι και 128 πυρήνες.*



***Σχήμα 6.9**: Γραφική παράσταση που αναπαριστά το power-performance efficiency για το fft για μέχρι και 128 πυρήνες.*

Από την πιο πάνω γραφική παράσταση , σχήμα 6.9, παρατηρείται ότι όσο αυξάνουμε τον αριθμό των επεξεργαστών αυξάνεται και το Power-Performance efficiency για την εφαρμογή FFT μέχρι και το σημείο των 64 πυρήνων όπως και στην εφαρμογή Ocean. Στην συνέχεια για 128 πυρήνες παρατηρείται μία μεγάλη μείωση του Power-Performance efficiency.

Σε θέμα κατανάλωσης ενέργειας, σχήμα 6.8, αυξάνοντας των αριθμό των πυρήνων του επεξεργαστή με εκθετικό ρυθμό παρατηρείται μία μεγάλη αύξηση της ολικής κατανάλωσης

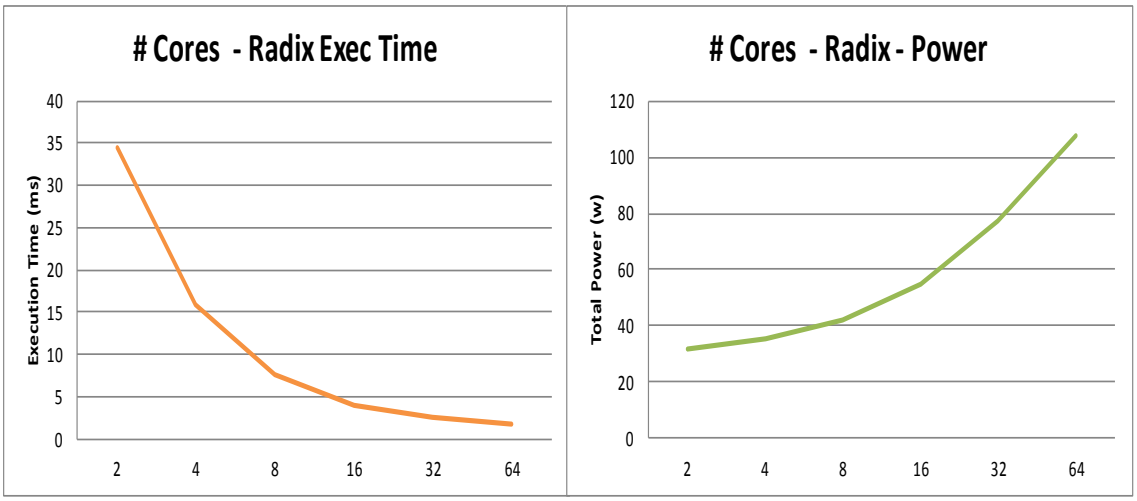
ενέργειας. Η ολική κατανάλωση ενέργειας στην περίπτωση αυτής της εφαρμογής αυξάνεται με πιο μικρό ρυθμό από ότι στις προηγούμενες εφαρμογές.

Σε θέμα επίδοσης, σχήμα 6.8, όσο αυξάνουμε τον αριθμό των πυρήνων του επεξεργαστή παρατηρείται μία μείωση στους χρόνους εκτέλεσης με εκθετικό ρυθμό. Οι χρόνοι εκτέλεσης της εφαρμογής FFT συνεχίζουν να μειώνονται με μεγάλο ρυθμό για μέχρι και 64 πυρήνες . Στην συνέχεια όμως για 128 πυρήνες παρατηρείται αύξηση του χρόνου εκτέλεσης από 0.46 ms για 64 πυρήνες, σε 0.541 ms για 128 πυρήνες γεγονός το οποίο δικαιολογεί την απότομη μείωση του Power-Performance efficiency σε εκείνο το σημείο. Αυτό οφείλετε στο γεγονός ότι η εφαρμογή FFT έφτασε σε σημείο που η παραλληλοποίηση που γίνεται για να τρέξει η εφαρμογή σε 128 πυρήνες έχει χειρότερα αποτελέσματα από ότι αν έτρεχε η εφαρμογή σε λιγότερους πυρήνες. Το overhead για να τρέξει η εφαρμογή παράλληλα σε 128 πυρήνες κοστίζει, με αποτέλεσμα να έχουμε χειρότερο χρόνο εκτέλεσης.

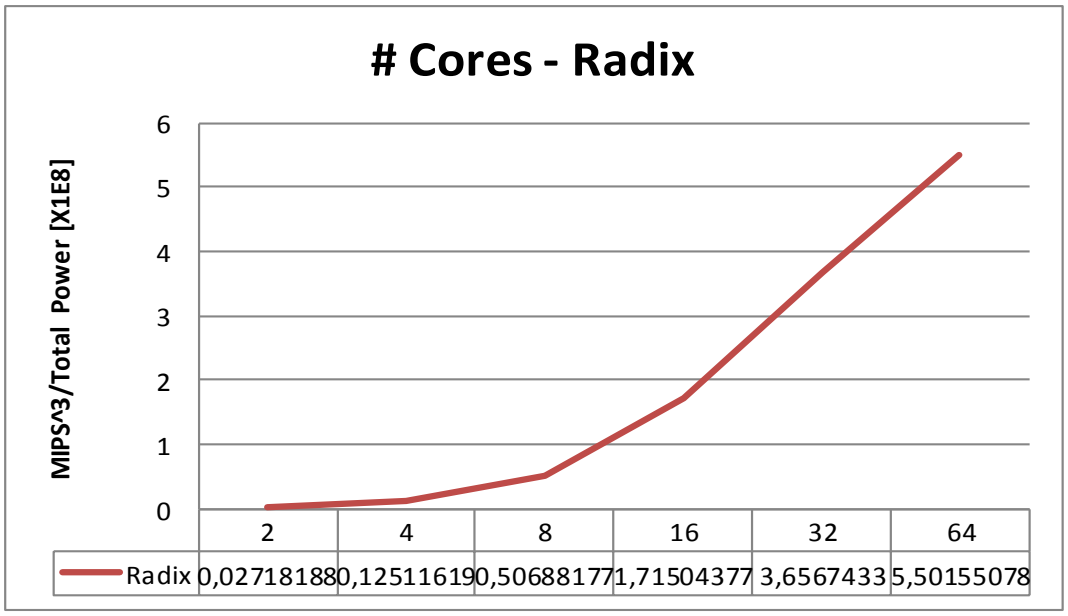
Στο σημείο λοιπόν από 64 πυρήνες μέχρι 128 πυρήνες παρατηρείται ότι το Power-Performance efficiency μειώνεται από 54515 σε 26166 0.47 φορές. Από 2 πυρήνες μέχρι και 64 πυρήνες παρατηρείται αύξηση του Power-Performance efficiency.

Συγκεκριμένα από το baseline configuration στους 8 πυρήνες με χρόνο εκτέλεσης 1.8 ms, μέχρι και τον αριθμό πυρήνων του πειράματος με το καλύτερο Power-Performance efficiency, τους 64 πυρήνες με χρόνο εκτέλεσης 0.46 ms έχουμε Power-Performance efficiency 3928.9 και 54515 αντίστοιχα. Παρατηρείται δηλαδή μία αύξηση στο Power-Performance efficiency 13.87 φορές. Στο κατώτατο όριο που ήταν οι 2 πυρήνες η τιμή του Power-Performance efficiency είναι 133. Από αυτό το σημείο μέχρι και το baseline configuration στους 8 πυρήνες παρατηρείται αύξηση στο Power-Performance efficiency 29.5 φορές.

Ο μέσος όρος αύξησης του Power-Performance efficiency συγκριτικά, ανάμεσα σε κάθε αύξηση στον αριθμό των πυρήνων είναι 3.16 φορές, δηλαδή αυξάνοντας των αριθμό των πυρήνων στο διπλάσιο έχουμε 3.16 φορές μεγαλύτερο Power-Performance efficiency. Επίσης το σημείο στο οποίο έχουμε συγκριτικά την μεγαλύτερη αύξηση του Power-Performance efficiency είναι από τους 4 πυρήνες σε 8 πυρήνες 6.36 φορές καλύτερο στους 8 πυρήνες.



***Σχήμα 6.10**: Γραφική παράσταση που αναπαριστά τους χρόνους εκτέλεσης και την κατανάλωση ενέργειας για το Radix για μέχρι και 128 πυρήνες.*



***Σχήμα 6.11**: Γραφική παράσταση που αναπαριστά το power-performance efficiency για το Radix για μέχρι και 64 πυρήνες λόγω αδυναμίας της εφαρμογής να τρέξει για 128 πυρήνες.*

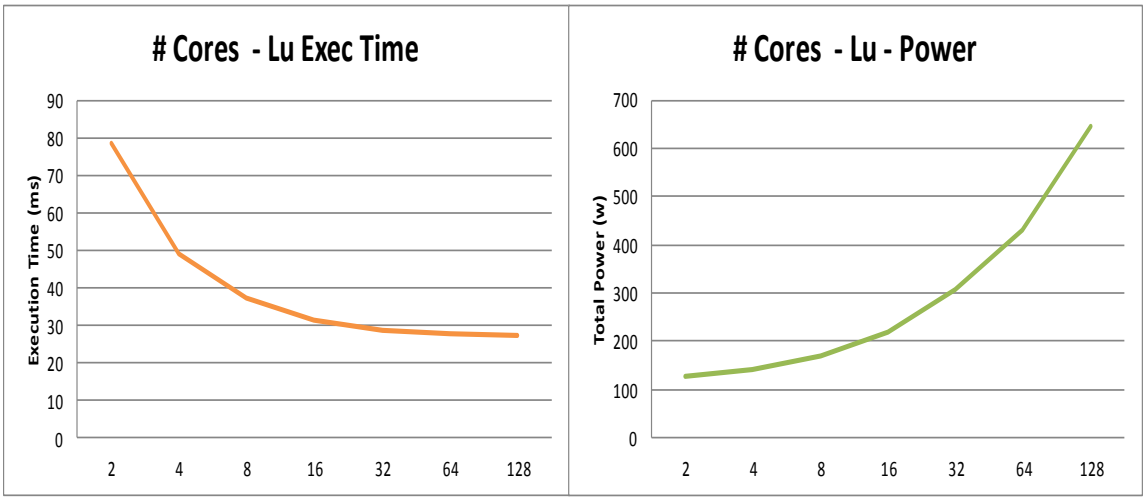
Από την πιο πάνω γραφική παράσταση, σχήμα 6.11, παρατηρείται ότι όσο αυξάνουμε τον αριθμό των επεξεργαστών αυξάνεται και το Power-Performance efficiency για την εφαρμογή Radix. Συγκεκριμένα αυξάνοντας με εκθετικό ρυθμό τους πυρήνες του επεξεργαστή έχουμε ανάλογη εκθετική αύξηση στο Power-Performance efficiency.

Σε θέμα κατανάλωσης ενέργειας , σχήμα 6.10, αυξάνοντας των αριθμό των πυρήνων του επεξεργαστή με εκθετικό ρυθμό παρατηρείται μία μεγάλη αύξηση της ολικής κατανάλωσης ενέργειας.

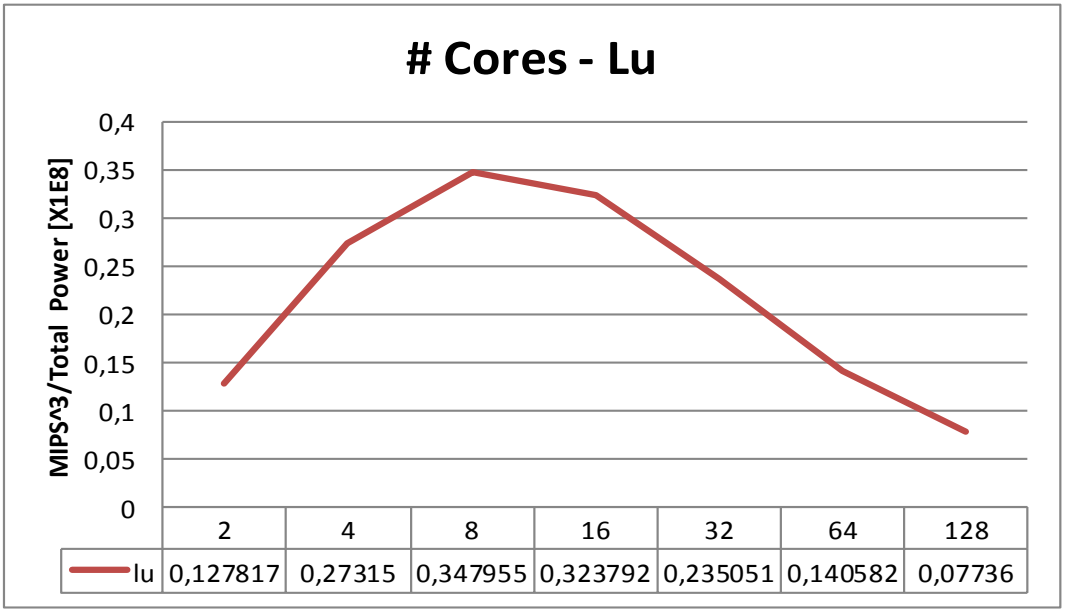
Σε θέμα επίδοσης, σχήμα 6.10, όσο αυξάνουμε τον αριθμό των πυρήνων του επεξεργαστή παρατηρείται μία μείωση στους χρόνους εκτέλεσης με εκθετικό ρυθμό. Οι χρόνοι εκτέλεσης της εφαρμογής Radix συνεχίζουν να μειώνονται με μεγάλο ρυθμό για μέχρι και 64 πυρήνες που είναι ο μέγιστος αριθμός πυρήνων με τους οποίους μπορούμε να εκτελέσουμε την εφαρμογή. Το γεγονός ότι η εφαρμογή δεν τρέχει για περισσότερους από 64 πυρήνες είναι μία πληροφορία που σε συνδυασμό με τα προηγούμενα αποτελέσματα μας λέει ότι η εφαρμογή δεν είναι αποδοτικό να παραλληλοποιηθεί περισσότερο.

Συγκεκριμένα από το baseline configuration στους 8 πυρήνες με χρόνο εκτέλεσης 7.6 ms, μέχρι και τον μεγαλύτερο αριθμό πυρήνων του πειράματος, τους 64 πυρήνες με χρόνο εκτέλεσης 1.65 ms , έχουμε Power-Performance efficiency 162.7 και 1765.9 αντίστοιχα. Παρατηρείται δηλαδή μία αύξηση στο Power-Performance efficiency 10.8 φορές. Στο κατώτατο όριο που ήταν οι 2 πυρήνες ο χρόνος εκτέλεσης ήταν 34.5 ms, και η τιμή του Power-Performance efficiency είναι 8.72 . Από αυτό το σημείο μέχρι και το baseline configuration στους 8 πυρήνες παρατηρείται αύξηση στο Power-Performance efficiency 18.65 φορές.

Ο μέσος όρος αύξησης του Power-Performance efficiency συγκριτικά, ανάμεσα σε κάθε αύξηση στον αριθμό των πυρήνων είναι 3.13 φορές. Με άλλα λόγια αυξάνοντας των αριθμό των πυρήνων στο διπλάσιο έχουμε 3.13 φορές μεγαλύτερο Power-Performance efficiency. Επίσης το σημείο στο οποίο έχουμε συγκριτικά την μεγαλύτερη αύξηση του Power-Performance efficiency είναι από τους 2 πυρήνες σε 4 πυρήνες 4.6 φορές καλύτερο στους 4 πυρήνες, ενώ το σημείο με την χειρότερη συγκριτικά αύξηση είναι από τους 32 πυρήνες σε 64 πυρήνες 1.5 φορές καλύτερο στους 64 πυρήνες.



***Σχήμα 6.12**: Γραφική παράσταση που αναπαριστά τους χρόνους εκτέλεσης και την κατανάλωση ενέργειας για το Lu για μέχρι και 128 πυρήνες.*



***Σχήμα 6.13**: Γραφική παράσταση που αναπαριστά το power-performance efficiency για το Lu για μέχρι και 128 πυρήνες.*

Από την πιο πάνω γραφική παράσταση, σχήμα 6.13, παρατηρείται ότι όσο αυξάνουμε τον αριθμό των επεξεργαστών αυξάνεται και το Power-Performance efficiency για την εφαρμογή Lu για μέχρι και το baseline configuration που είναι 8 πυρήνες . Στην συνέχεια από 8 πυρήνες για μέχρι και 128 πυρήνες παρατηρείται μία μείωση του Power-Performance efficiency.

Σε θέμα κατανάλωσης ενέργειας , σχήμα 6.12, αυξάνοντας των αριθμό των πυρήνων του επεξεργαστή με εκθετικό ρυθμό παρατηρείται μία μεγάλη αύξηση της ολικής κατανάλωσης

ενέργειας. Η ολική κατανάλωση ενέργειας σε αυτή την εφαρμογή έχει τον δεύτερο μεγαλύτερο ρυθμό αύξησης από όλες τις εφαρμογές.

Σε θέμα επίδοσης, σχήμα 6.12, όσο αυξάνουμε τον αριθμό των πυρήνων του επεξεργαστή παρατηρείται μία μείωση στους χρόνους εκτέλεσης. Αυτό το οποίο αλλάζει είναι ο ρυθμός μειώσεις των χρόνων εκτέλεσης της εφαρμογής όσο αυξάνουμε τον αριθμό των πυρήνων του επεξεργαστή. Από το σημείο των 2 πυρήνων μέχρι και τους 8 πυρήνες ο ρυθμός μειώσεις των χρόνων εκτέλεσης είναι εκθετικός. Συγκεκριμένα για 2 πυρήνες, 4 πυρήνες και 8 πυρήνες οι αντίστοιχοι χρόνοι είναι 78.9 ms 48.8 ms και 37.2 ms. Η εφαρμογή λοιπόν παραλληλοποιείται αρκετά καλά σε αυτό το διάστημα με αποτέλεσμα να έχουμε βελτίωση του Power-Performance efficiency. Στην συνέχεια ο ρυθμός μειώσεις των χρόνων εκτέλεσης γίνεται γραμμικός. Αυτό οφείλεται στο γεγονός ότι η εφαρμογή αυτή έχει τα μεγαλύτερα ποσοστά χρόνου εκτέλεσης που χρησιμοποιούνται για synchronization. Συγκεκριμένα για 16 πυρήνες, 32 πυρήνες 64 πυρήνες και 128 πυρήνες οι αντίστοιχοι χρόνοι είναι 31.4 ms 28.62 ms 27.55 ms και 27 ms.

Συγκεκριμένα από το baseline configuration στους 8 πυρήνες μέχρι και τον αριθμό πυρήνων του πειράματος με το χαμηλότερο Power-Performance efficiency, τους 128 πυρήνες έχουμε Power-Performance efficiency 0.34 και 0.077 αντίστοιχα. Παρατηρείται δηλαδή μία μείωση στο Power-Performance efficiency 0.22 φορές. Στο κατώτατο όριο που ήταν οι 2 πυρήνες η τιμή του Power-Performance efficiency είναι 0.12. Από αυτό το σημείο μέχρι και το baseline configuration στους 8 πυρήνες παρατηρείται αύξηση στο Power-Performance efficiency 2.83 φορές.

Ο μέσος όρος αύξησης του Power-Performance efficiency συγκριτικά, ανάμεσα σε κάθε αύξηση στον αριθμό των πυρήνων είναι 1.05 φορές. Με άλλα λόγια αυξάνοντας των αριθμό των πυρήνων στο διπλάσιο έχουμε 1.05 φορές μεγαλύτερο Power-Performance efficiency. Επίσης το σημείο στο οποίο έχουμε συγκριτικά την μεγαλύτερη αύξηση του Power-Performance efficiency είναι από τους 2 πυρήνες σε 4 πυρήνες 2.25 φορές καλύτερο στους 4 πυρήνες, ενώ το σημείο με την χειρότερη συγκριτικά μείωση είναι από τους 64 πυρήνες σε 128 πυρήνες 0.55 φορές χειρότερο στους 128 πυρήνες.

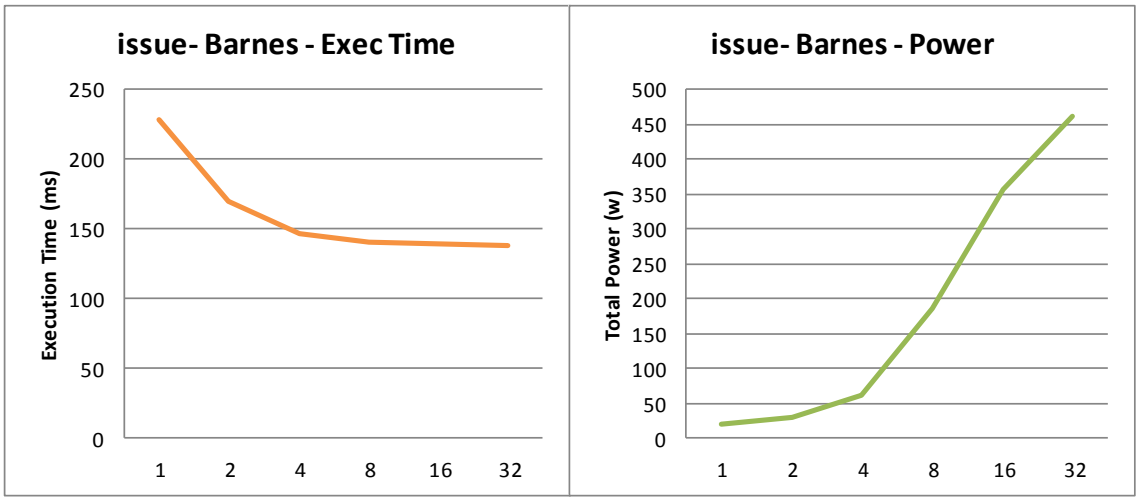
6.4 Power-Performance efficiency για το issue του επεξεργαστή

Ένα σημαντικό configuration του επεξεργαστή που έχει εξετασθεί, αποτελεί το issue του επεξεργαστή. Για να δούμε πως η αλλαγή στο μέγεθος του issue επηρεάζει το Power-Performance efficiency κρατήσαμε σταθερά τα υπόλοιπα configuration στο baseline και αλλάξαμε το issue με τιμές από 1 μέχρι 32 και για τα 6 benchmarks που έχουν επιλεχθεί.

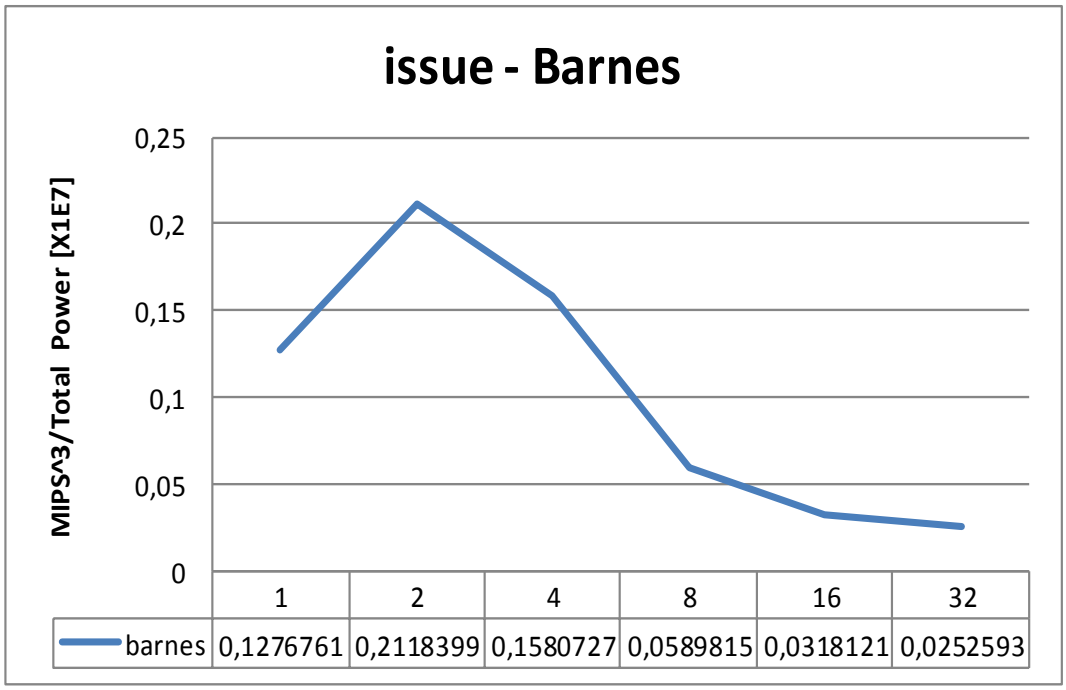
Το issue του επεξεργαστή είναι μία πολύ σημαντική παράμετρος που έχει εξετασθεί για τον λόγω ότι σχετίζεται με πολλές παραμέτρους του επεξεργαστή. Μπορούμε να πούμε ότι αλλάζοντας το issue του επεξεργαστή αλλάζει ολόκληρη η δομή του επεξεργαστή. Συγκεκριμένα οι παράμετροι με τις οποίες σχετίζεται το issue φαίνονται στο πίνακα 6.2.

Πίνακας 6.2*: Ισοδυναμία issue με άλλες παραμέτρους του επεξεργαστή*

Παράμετροι	**Ισοδυναμία**
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)



***Σχήμα 6.14**: Γραφική παράσταση που αναπαριστά τους χρόνους εκτέλεσης και την κατανάλωση ενέργειας για το Barnes για μέχρι και issue 32.*



***Σχήμα 6.15**: Γραφική παράσταση που αναπαριστά το power-performance efficiency για το Barnes για μέχρι και issue 32.*

Από την πιο πάνω γραφική παράσταση, σχήμα 6.15, παρατηρείται ότι όσο αυξάνουμε το μέγεθος του issue του επεξεργαστή αυξάνεται και το Power-Performance efficiency για την εφαρμογή Barnes για μέχρι και μέγεθος issue 2 . Στην συνέχεια από 2 issue για μέχρι και 32 issue παρατηρείται μία μείωση του Power-Performance efficiency.

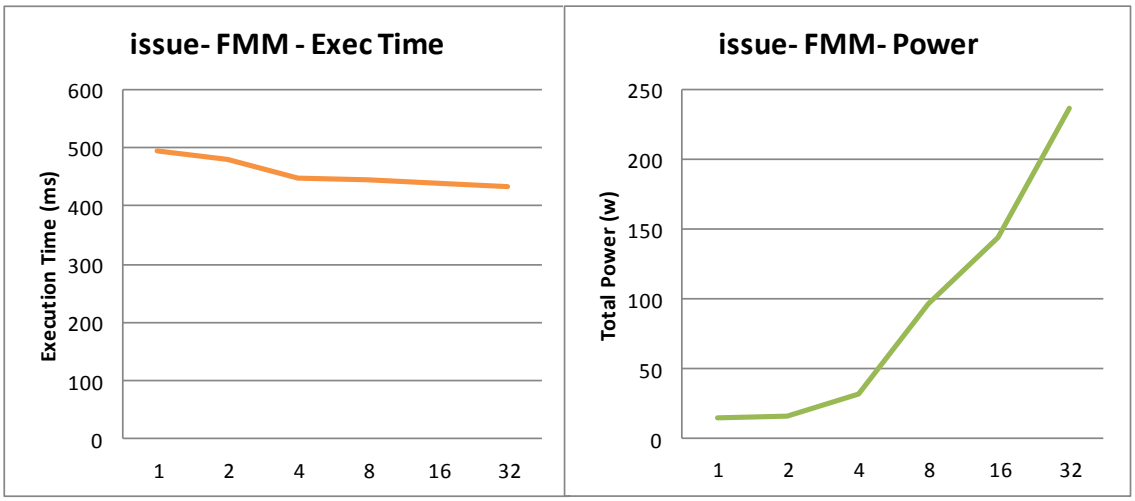
Σε θέμα κατανάλωσης ενέργειας, σχήμα 6.14, όσο το μέγεθος του issue του επεξεργαστή μεγαλώνει, η κατανάλωση ενέργειας αυξάνεται. Ο ρυθμός αύξησης όμως διαφέρει από

σημείο σε σημείο διαφορετικού μεγέθους του issue, κάτι που επηρεάζει σε μεγάλο βαθμό το Power-Performance efficiency. Για την εφαρμογή Barnes παρατηρείται ότι από issue 1 μέχρι και issue 2 που ο ρυθμός αύξησης της ολικής κατανάλωση ενέργειας είναι αρκετά χαμηλός, το Power-Performance efficiency αυξάνεται. Στην συνέχεια παρατηρείται μείωση του Power-Performance efficiency μέχρι και issue 32. Ο τρόπος με τον οποίο μειώνεται το Power-Performance efficiency είναι αντίστροφος ανάλογος του ρυθμού αύξησης της ολικής κατανάλωση ενέργειας. Από αυτή την παρατήρηση συμπεραίνουμε ότι η τιμές του Power-Performance efficiency έχουν άμεση σχέση με την κατανάλωση ενέργειας και ότι η επίδοση δεν επηρεάζει σημαντικά το Power-Performance efficiency όταν αυξάνουμε το issue του επεξεργαστή. Το issue όπως φαίνεται και στον πίνακα 6.2, σχετίζεται με το area factor του επεξεργαστή. Η ισοδυναμία του area factor με το τετράγωνο του issue είναι ένας σημαντικός λόγος για τον οποίο αυξάνεται η ολική κατανάλωση ενέργειας όσο αυξάνεται το issue του επεξεργαστή. Επίσης αυξάνοντας το issue αυξάνουμε τις μονάδες του επεξεργαστή, γεγονός που αυξάνει την κατανάλωση ενέργειας.

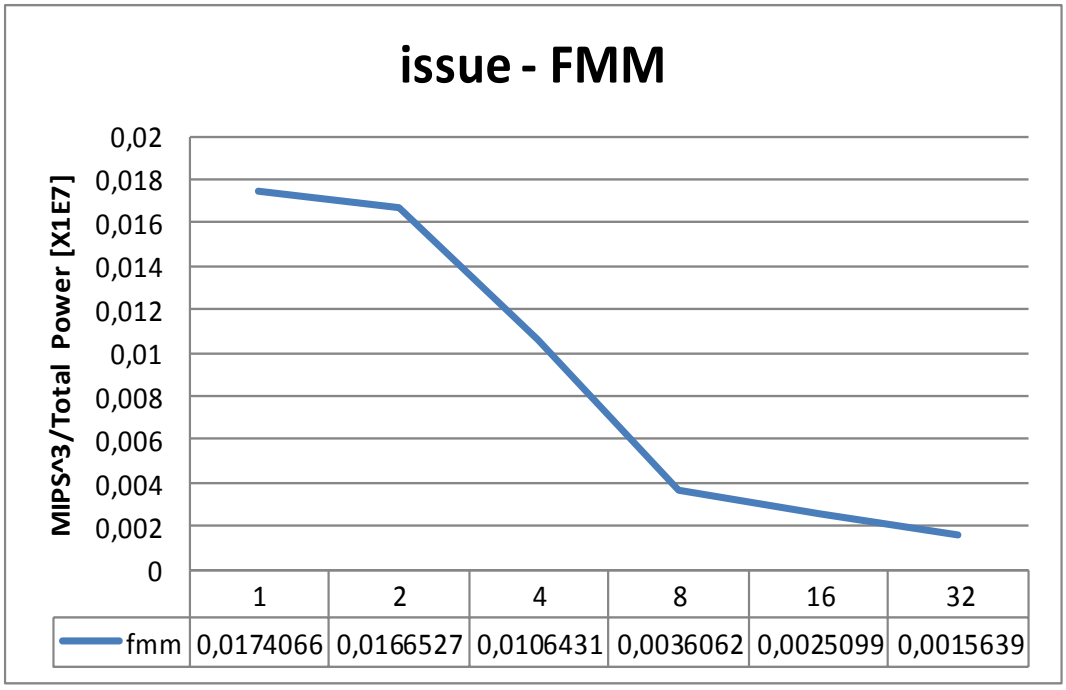
Σε θέμα επίδοσης, σχήμα 6.14, όσο το μέγεθος του issue του επεξεργαστή μεγαλώνει παρατηρείται μία μείωση στους χρόνους εκτέλεσης, όχι όμως σε τόσο μεγάλο βαθμό που να επηρεάζει σημαντικά το Power-Performance efficiency. Η μόνη περίπτωση στην οποία οι χρόνοι εκτέλεσης βελτιώνονται σημαντικά είναι από issue 1 σε issue 2, 228.46 ms και 169.49 ms αντίστοιχα, γεγονός που δικαιολογεί την αύξηση του Power-Performance efficiency σε εκείνο το σημείο.

Το καλύτερο Power-Performance efficiency έχουμε όταν ο αριθμός των issue είναι 2 στο 0.21, με μία διαφορά 3.6 φορές καλύτερο από το baseline configuration που το Power-Performance efficiency είναι 0.058. Το χειρότερο Power-Performance efficiency έχουμε όταν ο αριθμός των issue είναι 32 στο 0.025, με μία διαφορά 0.43 φορές χειρότερο από το baseline configuration .

Συνοψίζοντας, όσο αυξάνουμε το μέγεθος του issue του επεξεργαστή έχουμε σταδιακά αρκετά μεγάλη αύξηση της κατανάλωσης ενέργειας. Από την άλλη σε θέμα επίδοσης οι χρόνοι εκτέλεσης της εφαρμογής βελτιώνονται, όχι όμως σε μεγάλο βαθμό. Συμπέρασμα λοιπόν είναι, όταν θέλουμε να έχουμε καλό Power-Performance efficiency καλό θα είναι να έχουμε χαμηλό issue του επεξεργαστή. Το ιδανικότερο issue για την εφαρμογή Barnes είναι 2.



***Σχήμα 6.16**: Γραφική παράσταση που αναπαριστά τους χρόνους εκτέλεσης και την κατανάλωση ενέργειας για το fmm για μέχρι και issue 32.*



***Σχήμα 6.17**: Γραφική παράσταση που αναπαριστά το power-performance efficiency για το fmm για μέχρι και issue 32.*

Από την πιο πάνω γραφική παράσταση, σχήμα 6.17, παρατηρείται ότι όσο αυξάνουμε το μέγεθος του issue του επεξεργαστή μειώνεται το Power-Performance efficiency για την εφαρμογή FMM για μέχρι και μέγεθος issue 32 .

Η κατανάλωσης ενέργειας, σχήμα 6.16, όσο το μέγεθος του issue του επεξεργαστή μεγαλώνει, η ολική κατανάλωση ενέργειας αυξάνεται. Ο ρυθμός αύξησης όμως διαφέρει από σημείο σε σημείο διαφορετικού μεγέθους του issue, κάτι που επηρεάζει σε μεγάλο βαθμό το

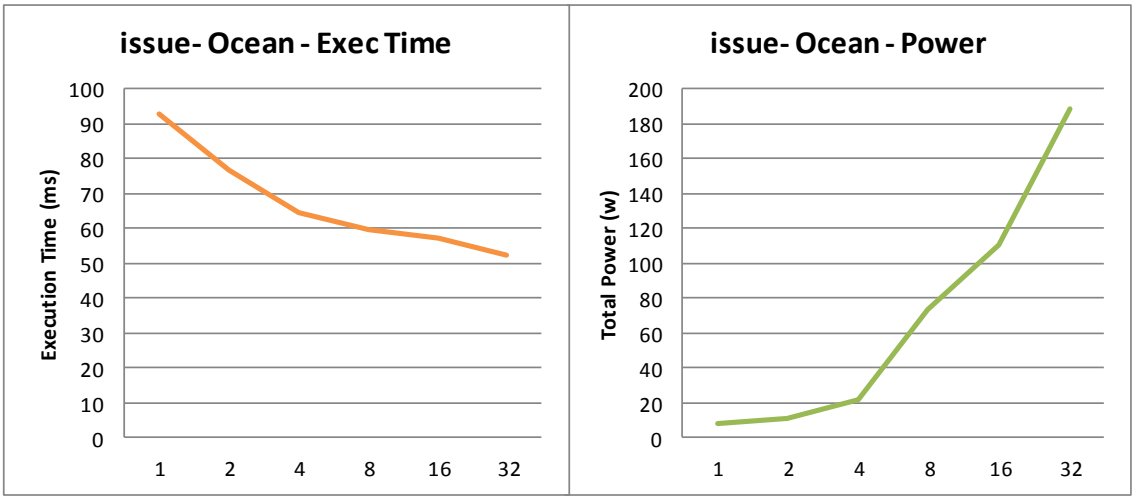
Power-Performance efficiency. Για την εφαρμογή FMM , παρατηρείται ότι από issue 1 μέχρι και issue 2 που ο ρυθμός αύξησης της ολικής κατανάλωση ενέργειας είναι αρκετά χαμηλός, το Power-Performance efficiency μειώνεται ελάχιστα. Στην συνέχεια από issue 2 μέχρι και issue 4 ο ρυθμός αύξησης της ολικής κατανάλωση ενέργειας μεγαλώνει κάτι που επηρεάζει αρνητικά το Power-Performance efficiency. Ακολούθως από issue 4 μέχρι και issue 8 ο ρυθμός αύξησης της ολικής κατανάλωση ενέργειας μεγαλώνει ακόμη περισσότερο αλλά το Power-Performance efficiency εξακολουθεί να μειώνεται με τον ίδιο ρυθμό. Μετά από issue 8 μέχρι και issue 16 παρατηρείται ελάχιστη μείωση στον ρυθμό αύξησης της ολικής κατανάλωση ενέργειας επηρεάζοντας θετικά τον ρυθμό μείωσης του Power-Performance efficiency.Τέλος από issue 16 μέχρι και issue 32 το Power-Performance efficiency εξακολουθεί να μειώνεται με τον ίδιο ρυθμό με πριν ενώ αντίθετα ο ρυθμός αύξησης της ολικής κατανάλωση ενέργειας μεγαλώνει.

Σε θέμα επίδοσης, σχήμα 6.16, όσο το μέγεθος του issue του επεξεργαστή μεγαλώνει παρατηρείται μία μείωση στους χρόνους εκτέλεσης, όχι όμως σε τόσο μεγάλο βαθμό που να επηρεάζει σημαντικά το Power-Performance efficiency. Για issue 1, issue 2, issue 4, issue 8, issue 16 και issue 32 οι χρόνοι εκτέλεσης είναι 495.7 ms, 481.6 ms, 449.3 ms, 443.6 ms, 437.9 ms και 434.6 ms αντίστοιχα. Οι βελτιώσεις αυτές σε χρόνους εκτέλεσης δικαιολογούν τους ρυθμούς μείωσης του Power-Performance efficiency που δεν συμβάδιζαν με τους ρυθμός αύξησης της ολικής κατανάλωση ενέργειας. Γενικά όμως δεν παρατηρείται πολύ μεγάλη βελτίωση της επίδοσης.

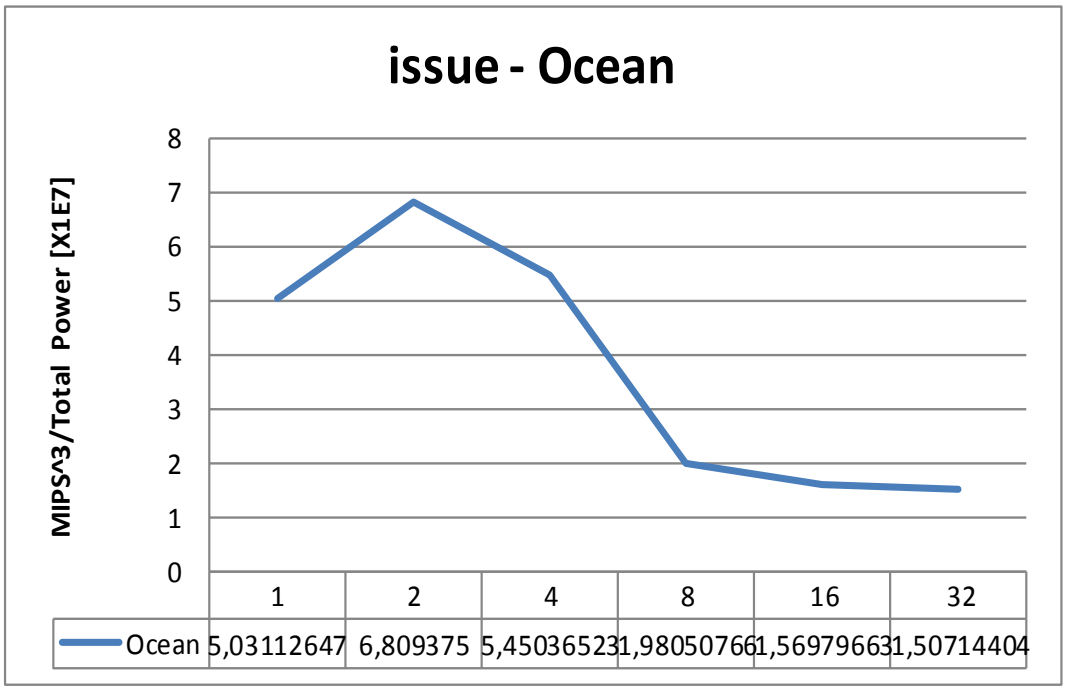
Όπως έχει αναφερθεί και όπως φαίνεται και στον πίνακα 6.2 το issue σχετίζεται με τον αριθμό των μονάδων του επεξεργαστή. Η εφαρμογή FMM λόγω εξαρτήσεων στον κώδικα της, όσο αυξάνουμε το issue δεν βελτιώνεται σε μεγάλο βαθμό η επίδοση. Επίσης οι καθυστερήσεις που δημιουργούνται όταν το issue είναι μικρό για την εφαρμογή FMM δεν φτάνουν σε ψηλά επίπεδα, γεγονός που δικαιολογεί τους χαμηλούς ρυθμούς βελτίωσης των χρόνων εκτέλεσης.

Το καλύτερο Power-Performance efficiency έχουμε όταν ο αριθμός των issue είναι 1 στο 0.017, με μία διαφορά 4.72 φορές καλύτερο από το baseline configuration που το Power-Performance efficiency είναι 0.0036. Το χειρότερο Power-Performance efficiency έχουμε όταν ο αριθμός των issue είναι 32 στο 0.0015, με μία διαφορά 0.41 φορές χειρότερο από το baseline configuration .

Συνοψίζοντας, όσο αυξάνουμε το μέγεθος του issue του επεξεργαστή έχουμε σταδιακά αρκετά μεγάλη αύξηση της κατανάλωσης ενέργειας ενώ σε θέμα επίδοσης οι χρόνοι εκτέλεσης της εφαρμογής βελτιώνονται σημαντικά όχι όμως σε τόσο μεγάλο βαθμό που να βελτιώνουν το Power-Performance efficiency. Συμπέρασμα λοιπόν είναι, όταν θέλουμε να έχουμε καλό Power-Performance efficiency καλό θα είναι να έχουμε χαμηλό issue του επεξεργαστή. Το ιδανικότερο issue για την εφαρμογή FMM είναι 1.



***Σχήμα 6.18**: Γραφική παράσταση που αναπαριστά τους χρόνους εκτέλεσης και την κατανάλωση ενέργειας για το ocean για μέχρι και issue 32.*



***Σχήμα 6.19**: Γραφική παράσταση που αναπαριστά το power-performance efficiency για το ocean για μέχρι και issue 32.*

Από την πιο πάνω γραφική παράσταση, σχήμα 6.19, παρατηρείται ότι όσο αυξάνουμε το μέγεθος του issue του επεξεργαστή αυξάνεται και το Power-Performance efficiency για την εφαρμογή Ocean για μέχρι και μέγεθος issue 2 . Στην συνέχεια από 2 issue για μέχρι και 32 issue παρατηρείται μία μείωση του Power-Performance efficiency.

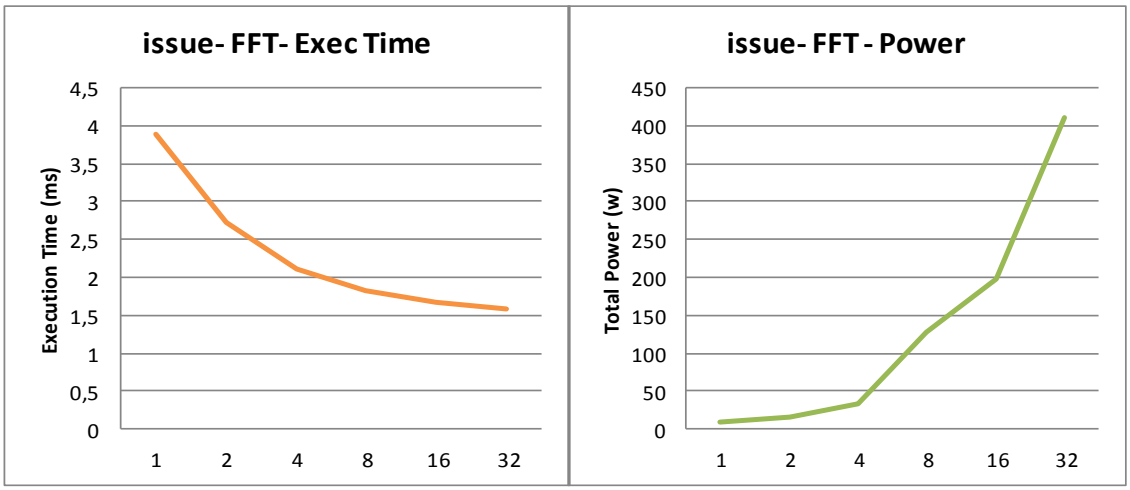
Σε θέμα κατανάλωσης ενέργειας, σχήμα 6.18, όσο το μέγεθος του issue του επεξεργαστή μεγαλώνει, η κατανάλωση ενέργειας αυξάνεται. Ο ρυθμός αύξησης όμως διαφέρει από σημείο σε σημείο διαφορετικού μεγέθους του issue, κάτι που επηρεάζει σε μεγάλο βαθμό το Power-Performance efficiency. Για την εφαρμογή Ocean, παρατηρείται ότι από issue 1 μέχρι και issue 2 που ο ρυθμός αύξησης της ολικής κατανάλωση ενέργειας είναι αρκετά χαμηλός, το Power-Performance efficiency αυξάνεται. Στην συνέχεια από issue 2 μέχρι και issue 4 το Power-Performance efficiency μειώνεται γιατί ο ρυθμός αύξησης της ολικής κατανάλωση ενέργειας αυξάνεται. Ακολούθως από issue 4 μέχρι και issue 8 ο ρυθμός αύξησης της ολικής κατανάλωση ενέργειας αυξάνεται σε μεγάλο βαθμό και αυτό επηρεάζει τον ρυθμό μείωσης του Power-Performance efficiency, ο οποίος επίσης αυξάνεται αρκετά. Μετά από issue 8 μέχρι και issue 16 παρατηρείται ελάχιστη μείωση στον ρυθμό αύξησης της ολικής κατανάλωση ενέργειας επηρεάζοντας θετικά τον ρυθμό μείωσης του Power-Performance efficiency. Τέλος από issue 16 μέχρι και issue 32 το Power-Performance efficiency εξακολουθεί να μειώνεται με ελάχιστα καλύτερο ρυθμό από πριν ενώ αντίθετα ο ρυθμός αύξησης της ολικής κατανάλωση ενέργειας μεγαλώνει ελάχιστα.

Σε θέμα επίδοσης, σχήμα 6.18, όσο το μέγεθος του issue του επεξεργαστή μεγαλώνει παρατηρείται μία μείωση στους χρόνους εκτέλεσης, όχι όμως σε τόσο μεγάλο βαθμό που να επηρεάζει σημαντικά το Power-Performance efficiency. Η μόνη περίπτωση στην οποία οι χρόνοι εκτέλεσης βελτιώνονται σημαντικά είναι από issue 1 σε issue 2, 92.67 ms και 76.48 ms αντίστοιχα, γεγονός που δικαιολογεί την αύξηση του Power-Performance efficiency σε εκείνο το σημείο. Επίσης οι χρόνοι εκτέλεσης βελτιώνονται περισσότερο από ότι σε άλλες εφαρμογές από issue 16 σε issue 32, γεγονός που δικαιολογεί τον ελάχιστα καλύτερο ρυθμό μείωσης του Power-Performance efficiency σε εκείνο το σημείο.

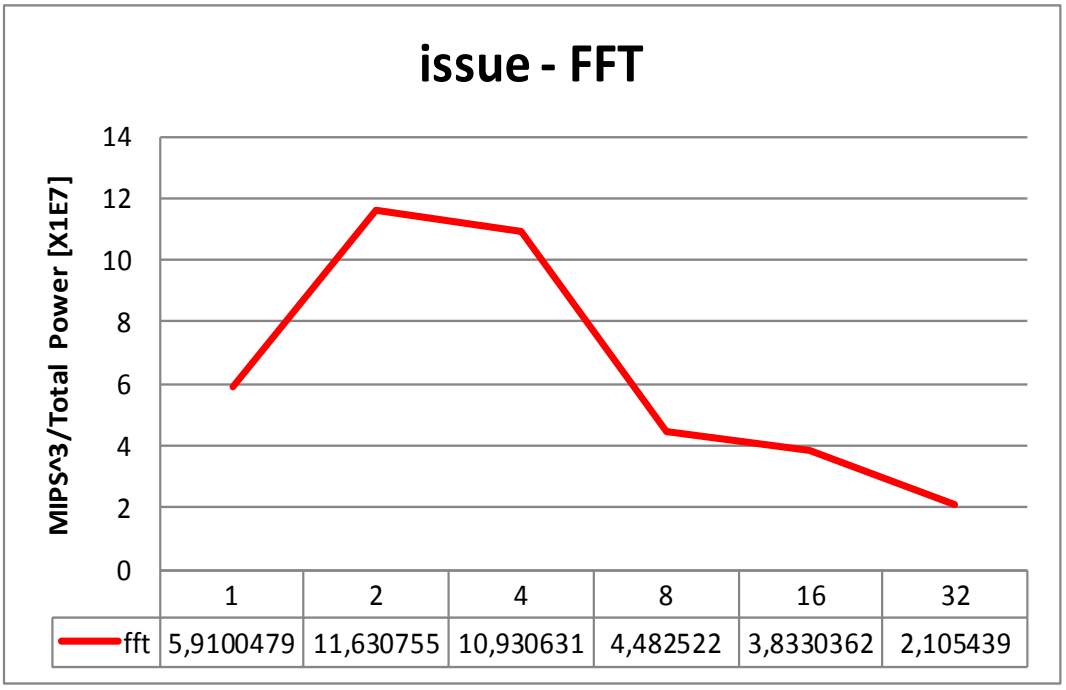
Όπως έχει αναφερθεί και όπως φαίνεται και στον πίνακα 6.2 το issue σχετίζεται με τον αριθμό των μονάδων του επεξεργαστή. Για την εφαρμογή Ocean όταν αυξάνουμε το issue από 1 σε 2 έχουμε αρκετά μεγάλη μείωση στις καθυστερήσεις που δημιουργούνται και αυξάνεται η επίδοση. Για μεγαλύτερα issue οι καθυστερήσεις αυτές δεν μειώνονται σε μεγάλο βαθμό και γι' αυτό η επίδοση δεν βελτιώνεται αρκετά.

Το καλύτερο Power-Performance efficiency έχουμε όταν ο αριθμός των issue είναι 2 στο 6.8, με μία διαφορά 3.4 φορές καλύτερο από το baseline configuration που το Power-Performance efficiency είναι 1.98. Το χειρότερο Power-Performance efficiency έχουμε όταν ο αριθμός των issue είναι 32 στο 1.5, με μία διαφορά 0.75 φορές χειρότερο από το baseline configuration .

Συνοψίζοντας, όσο αυξάνουμε το μέγεθος του issue του επεξεργαστή οι χρόνοι εκτέλεσης της εφαρμογής βελτιώνονται, αλλά η σταδιακή μεγάλη αύξηση της κατανάλωσης ενέργειας επηρεάζει αρνητικά το Power-Performance efficiency. Όταν θέλουμε λοιπόν να έχουμε καλό Power-Performance efficiency πρέπει να κρατούμε χαμηλό το issue του επεξεργαστή. Το ιδανικότερο issue για την εφαρμογή Ocean είναι 2.



Σχήμα 6.20*: Γραφική παράσταση που αναπαριστά τους χρόνους εκτέλεσης και την κατανάλωση ενέργειας για το fft για μέχρι και issue 32.*



Σχήμα 6.21*: Γραφική παράσταση που αναπαριστά το power-performance efficiency για το fft για μέχρι και issue 32.*

Από την πιο πάνω γραφική παράσταση, σχήμα 6.21, παρατηρείται ότι όσο αυξάνουμε το μέγεθος του issue του επεξεργαστή αυξάνεται και το Power-Performance efficiency για την εφαρμογή FFT για μέχρι και μέγεθος issue 2 . Στην συνέχεια από 2 issue για μέχρι και 32 issue παρατηρείται μία μείωση του Power-Performance efficiency.

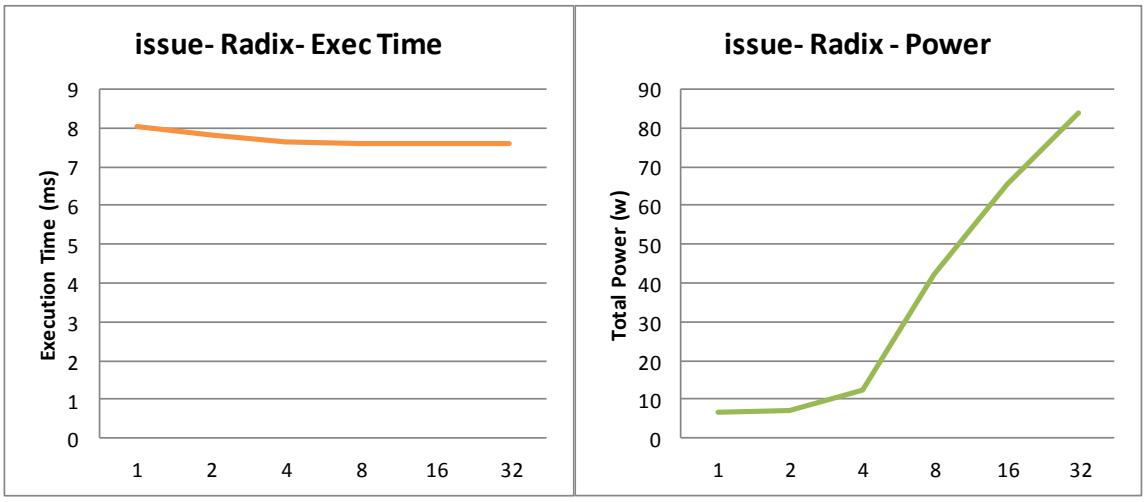
Σε θέμα κατανάλωσης ενέργειας, σχήμα 6.20, όσο το μέγεθος του issue του επεξεργαστή μεγαλώνει, η κατανάλωση ενέργειας αυξάνεται αλλά με διαφορετικό ρυθμό από σημείο σε σημείο διαφορετικού μεγέθους του issue. Για την εφαρμογή FFT, παρατηρείται ότι από issue 1 μέχρι και issue 2 που ο ρυθμός αύξησης της ολικής κατανάλωση ενέργειας είναι αρκετά χαμηλός, το Power-Performance efficiency αυξάνεται αρκετά. Στην συνέχεια από issue 2 μέχρι και issue 4 ο ρυθμός αύξησης της ολικής κατανάλωση ενέργειας αυξάνεται κάτι που επηρεάζει το Power-Performance efficiency, το οποίο μειώνεται αλλά ελάχιστα. Ακολούθως από issue 4 μέχρι και issue 8 ο ρυθμός αύξησης της ολικής κατανάλωση ενέργειας μεγαλώνει ακόμη περισσότερο και αυτό επηρεάζει τον ρυθμό μείωσης του Power-Performance efficiency, ο οποίος επίσης αυξάνεται αρκετά. Μετά από issue 8 μέχρι και issue 16 παρατηρείται ελάχιστη μείωση στον ρυθμό αύξησης της ολικής κατανάλωση ενέργειας σε αντίθεση με τον ρυθμό μείωσης του Power-Performance efficiency που μειώνεται αρκετά. Τέλος από issue 16 μέχρι και issue 32 το Power-Performance efficiency μειώνεται με ελάχιστα χειρότερο ρυθμό από πριν , που σε αυτό συμβάλει η ελάχιστη αύξηση του ρυθμού αύξησης της ολικής κατανάλωση ενέργειας.

Σε θέμα επίδοσης, σχήμα 6.20, όσο το μέγεθος του issue του επεξεργαστή μεγαλώνει παρατηρείται μείωση στους χρόνους εκτέλεσης. . Για issue 1, issue 2, issue 4, issue 8, issue 16 και issue 32 οι χρόνοι εκτέλεσης είναι 3.8 ms, 2.7 ms, 2.1 ms, 1.8 ms, 1.6 ms και 1.5 ms αντίστοιχα. Η μόνη περίπτωση στην οποία οι χρόνοι εκτέλεσης βελτιώνονται σημαντικά είναι από issue 1 σε issue 2, γεγονός που δικαιολογεί την αύξηση του Power-Performance efficiency σε εκείνο το σημείο. Ακόμα οι χρόνοι εκτέλεσης βελτιώνονται αρκετά από issue 2 σε issue 4 αλλά η μεγάλη αύξηση της κατανάλωσης ενέργειας όπως έχει αναφερθεί επηρεάζει αρνητικά το Power-Performance efficiency.

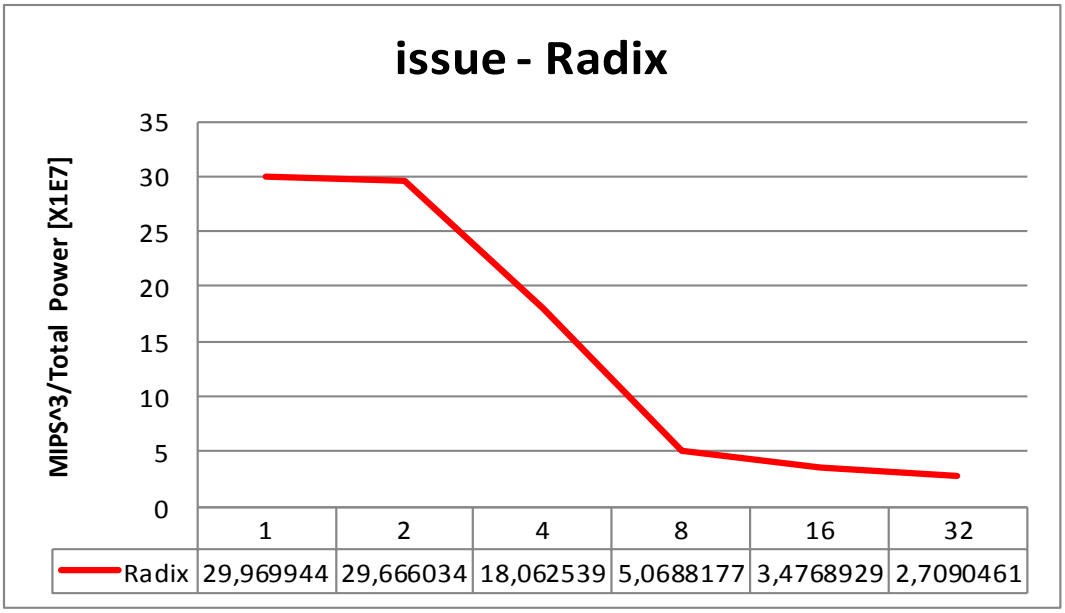
Όπως έχει αναφερθεί και όπως φαίνεται και στον πίνακα 6.2 το issue σχετίζεται με τον αριθμό των μονάδων του επεξεργαστή. Για την εφαρμογή FFT όταν αυξάνουμε το issue από 1 σε 2 έχουμε αρκετά μεγάλη μείωση στις καθυστερήσεις που δημιουργούνται και αυξάνεται η επίδοση αρκετά. Ακόμα συμβαίνει και μέχρι issue 4. Για μεγαλύτερα issue οι καθυστερήσεις αυτές δεν μειώνονται σε μεγάλο βαθμό και γι’ αυτό η επίδοση δεν βελτιώνεται αρκετά.

Το καλύτερο Power-Performance efficiency έχουμε όταν ο αριθμός των issue είναι 2 στο 101943 με μία διαφορά 2.6 φορές καλύτερο από το baseline configuration που το Power-Performance efficiency είναι 39289. Το χειρότερο Power-Performance efficiency έχουμε όταν ο αριθμός των issue είναι 32 στο 18454, με μία διαφορά 0.46 φορές χειρότερο από το baseline configuration .

Συνοψίζοντας, όσο αυξάνουμε το μέγεθος του issue του επεξεργαστή οι χρόνοι εκτέλεσης της εφαρμογής βελτιώνονται συγκριτικά περισσότερο από τις άλλες εφαρμογές , αλλά η σταδιακή μεγάλη αύξηση της κατανάλωσης ενέργειας επηρεάζει αρνητικά το Power-Performance efficiency. Όταν στόχος μας είναι το καλύτερο Power-Performance efficiency πρέπει να κρατούμε χαμηλό το issue του επεξεργαστή. Το ιδανικότερο issue για την εφαρμογή FFT είναι 2.



***Σχήμα 6.22**: Γραφική παράσταση που αναπαριστά τους χρόνους εκτέλεσης και την κατανάλωση ενέργειας για το radix για μέχρι και issue 32.*



***Σχήμα 6.23**: Γραφική παράσταση που αναπαριστά το power-performance efficiency για το radix για μέχρι και issue 32.*

Από την πιο πάνω γραφική παράσταση, σχήμα 6.23, παρατηρείται ότι όσο αυξάνουμε το μέγεθος του issue του επεξεργαστή μειώνεται το Power-Performance efficiency για την εφαρμογή Radix για μέχρι και μέγεθος issue 32.

Σε θέμα κατανάλωσης ενέργειας, σχήμα 6.22, όσο το μέγεθος του issue του επεξεργαστή μεγαλώνει, η κατανάλωση ενέργειας αυξάνεται αλλά με διαφορετικό ρυθμό από σημείο σε σημείο διαφορετικού μεγέθους του issue. Για την εφαρμογή Radix, παρατηρείται ότι από

issue 1 μέχρι και issue 2 που ο ρυθμός αύξησης της ολικής κατανάλωση ενέργειας είναι αρκετά χαμηλός, το Power-Performance efficiency μειώνεται ελάχιστα. Στην συνέχεια από issue 2 μέχρι και issue 4 ο ρυθμός αύξησης της ολικής κατανάλωση ενέργειας αυξάνεται και ο ρυθμός μείωσης του Power-Performance efficiency αυξάνεται σε μεγάλο βαθμό. Ακολούθως από issue 4 μέχρι και issue 8 ο ρυθμός αύξησης της ολικής κατανάλωση ενέργειας μεγαλώνει ακόμη περισσότερο αλλά το Power-Performance efficiency εξακολουθεί να μειώνεται με τον ίδιο ρυθμό. Μετά από issue 8 μέχρι και issue 16 παρατηρείται ελάχιστη μείωση στον ρυθμό αύξησης της ολικής κατανάλωση ενέργειας ενώ ο ρυθμός μείωσης του Power-Performance efficiency βελτιώνεται σημαντικά. Τέλος από issue 16 μέχρι και issue 32 το Power-Performance efficiency εξακολουθεί να μειώνεται με τον ίδιο ρυθμό όπως συμβαίνει και στον ρυθμό αύξησης της ολικής κατανάλωση ενέργειας.

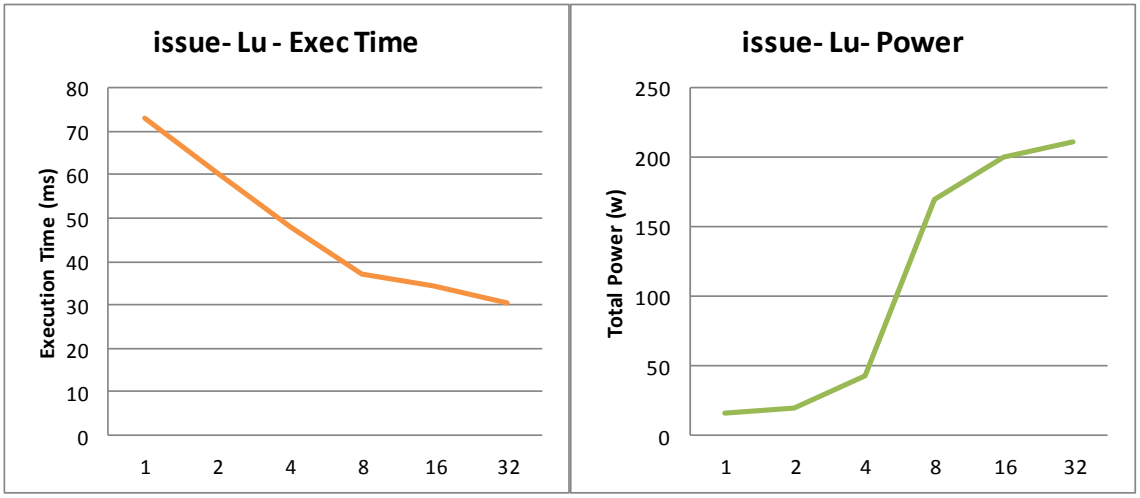
Σε θέμα επίδοσης, σχήμα 6.22, όσο το μέγεθος του issue του επεξεργαστή μεγαλώνει παρατηρείται μία μείωση στους χρόνους εκτέλεσης, όχι όμως σε τόσο μεγάλο βαθμό που να επηρεάζει σημαντικά το Power-Performance efficiency. Για issue 1, issue 2, issue 4, issue 8, issue 16 και issue 32 οι χρόνοι εκτέλεσης είναι 8 ms, 7.8 ms, 7.65 ms, 7.61 ms, 7.605 ms 7.600 ms αντίστοιχα. Οι βελτιώσεις αυτές σε χρόνους εκτέλεσης δικαιολογούν τους ρυθμούς μείωσης του Power-Performance efficiency που δεν συμβάδιζαν με τους ρυθμός αύξησης της ολικής κατανάλωση ενέργειας. Γενικά όμως δεν έχουμε μεγάλη βελτίωση της επίδοσης και γι’ αυτό το Power-Performance efficiency μειώνεται.

Όπως έχει αναφερθεί και όπως φαίνεται και στον πίνακα 6.2 το issue σχετίζεται με τον αριθμό των μονάδων του επεξεργαστή. Η εφαρμογή Radix λόγω εξαρτήσεων στον κώδικα της, όσο αυξάνουμε το issue δεν βελτιώνεται σε μεγάλο βαθμό η επίδοση. Επίσης οι καθυστερήσεις που δημιουργούνται όταν το issue είναι μικρό για την εφαρμογή Radix δεν φτάνουν σε ψηλά επίπεδα, γεγονός που δικαιολογεί τους χαμηλούς ρυθμούς βελτίωσης των χρόνων εκτέλεσης.

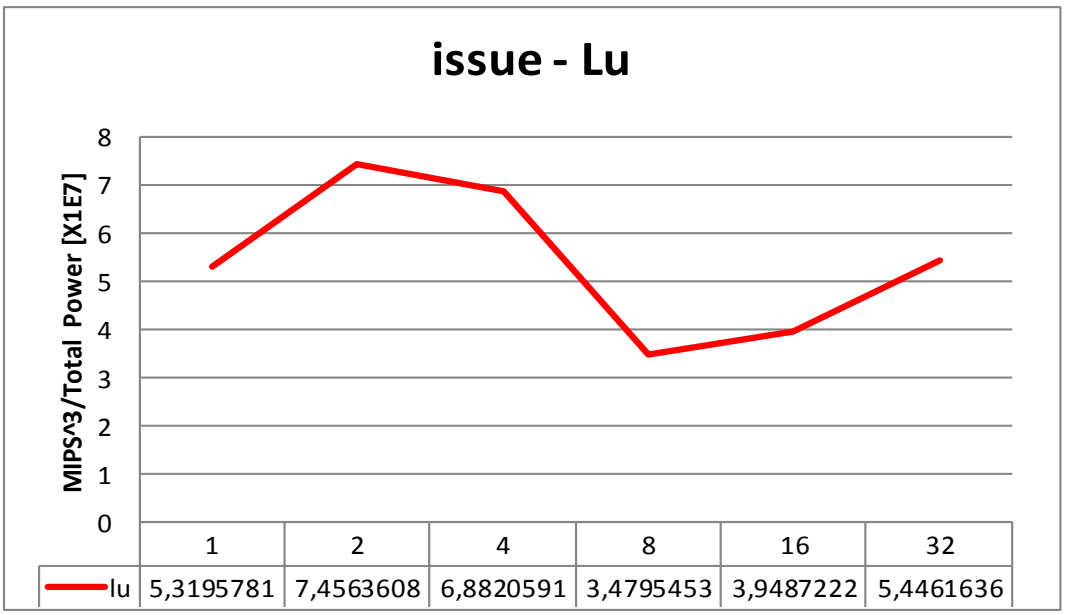
Το καλύτερο Power-Performance efficiency έχουμε όταν ο αριθμός των issue είναι 1 στο 9620 με μία διαφορά 5.9 φορές καλύτερο από το baseline configuration που το Power-Performance efficiency είναι 1627. Το χειρότερο Power-Performance efficiency έχουμε όταν ο αριθμός των issue είναι 32 στο 869 με μία διαφορά 0.77 φορές χειρότερο από το baseline configuration .

Συνοψίζοντας, όσο αυξάνουμε το μέγεθος του issue του επεξεργαστή έχουμε σταδιακά

αρκετά μεγάλη αύξηση της κατανάλωσης ενέργειας ενώ σε θέμα επίδοσης οι χρόνοι εκτέλεσης της εφαρμογής δεν βελτιώνονται σημαντικά . Συμπέρασμα λοιπόν είναι, όταν στοχεύουμε σε καλό Power-Performance efficiency θα πρέπει να έχουμε χαμηλό issue του επεξεργαστή. Το ιδανικότερο issue για την εφαρμογή Radix είναι 1.



***Σχήμα6.24**: Γραφική παράσταση που αναπαριστά τους χρόνους εκτέλεσης και την κατανάλωση ενέργειας για το lu για μέχρι και issue 32.*



***Σχήμα 6.25**: Γραφική παράσταση που αναπαριστά το power-performance efficiency για το lu για μέχρι και issue 32.*

Από την πιο πάνω γραφική παράσταση, σχήμα 6.25, παρατηρείται ότι όσο αυξάνουμε το μέγεθος του issue του επεξεργαστή αυξάνεται και το Power-Performance efficiency για την

εφαρμογή Lu για μέχρι και μέγεθος issue 2 . Στην συνέχεια από 2 issue για μέχρι και 8 issue παρατηρείται μία μείωση του Power-Performance efficiency. Ακολούθως από 8 issue για μέχρι και 32 issue το Power-Performance efficiency αυξάνεται και πάλι.

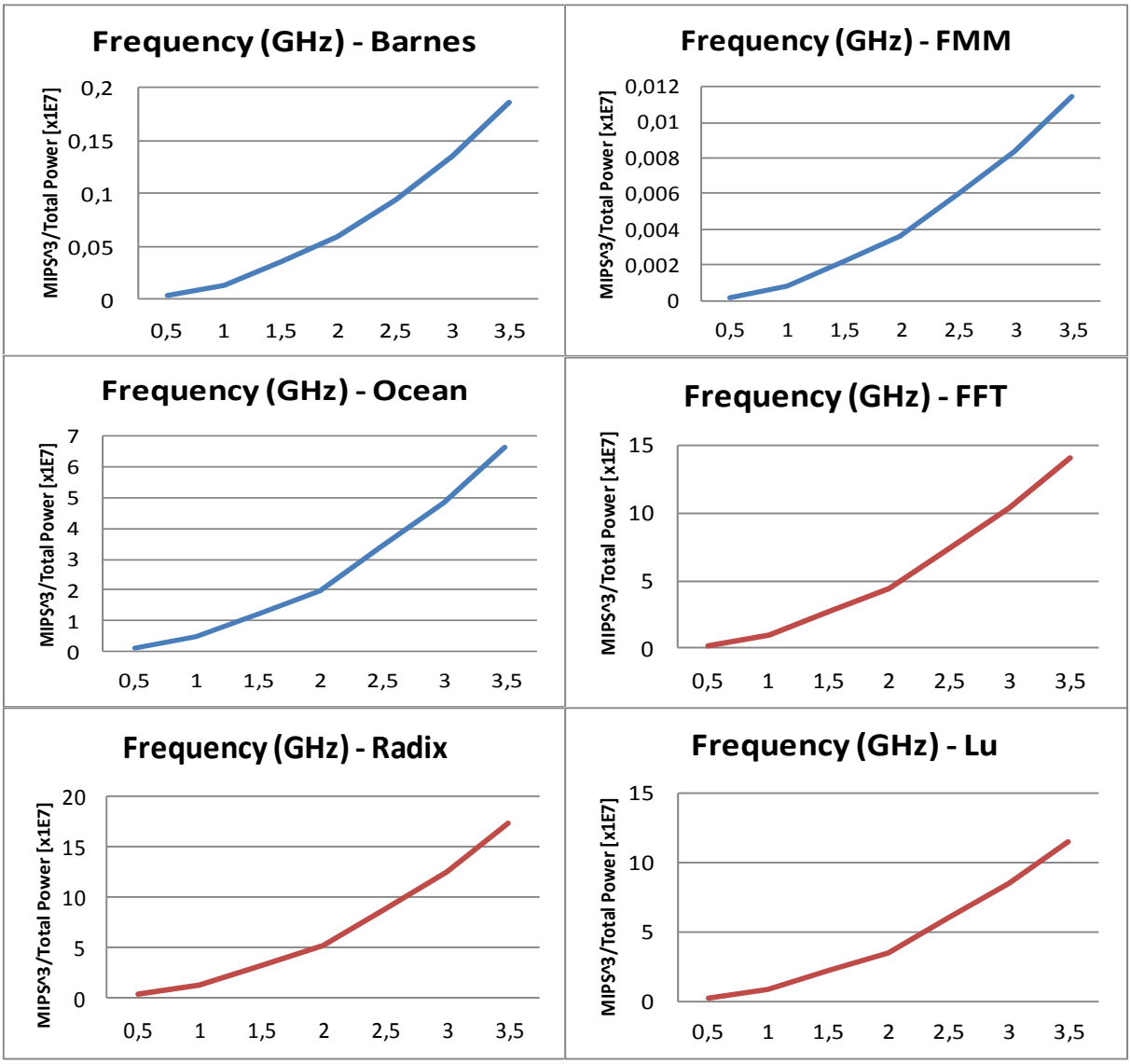
Σε θέμα κατανάλωσης ενέργειας, σχήμα 6.24, όσο το μέγεθος του issue του επεξεργαστή μεγαλώνει, η κατανάλωση ενέργειας αυξάνεται αλλά με διαφορετικό ρυθμό από σημείο σε σημείο διαφορετικού μεγέθους του issue. Για την εφαρμογή Lu, παρατηρείται ότι από issue 1 μέχρι και issue 2 που ο ρυθμός αύξησης της ολικής κατανάλωση ενέργειας είναι αρκετά χαμηλός, το Power-Performance efficiency αυξάνεται. Στην συνέχεια από issue 2 μέχρι και issue 4 ο ρυθμός αύξησης της ολικής κατανάλωση ενέργειας αυξάνεται κάτι που επηρεάζει το Power-Performance efficiency, το οποίο μειώνεται. Ακολούθως από issue 4 μέχρι και issue 8 ο ρυθμός αύξησης της ολικής κατανάλωση ενέργειας μεγαλώνει ακόμη περισσότερο και αυτό επηρεάζει τον ρυθμό μείωσης του Power-Performance efficiency, ο οποίος επίσης αυξάνεται αρκετά. Μετά από issue 8 μέχρι και issue 16 παρατηρείται μεγάλη μείωση στον ρυθμό αύξησης της ολικής κατανάλωση ενέργειας και αυτό φαίνεται να επηρεάζει θετικά το Power-Performance efficiency που αρχίζει να αυξάνεται. Τέλος από issue 16 μέχρι και issue 32 το Power-Performance efficiency αυξάνεται με μεγαλύτερο ρυθμό από πριν , κάτι που οφείλετε στην περαιτέρω μείωση του ρυθμού αύξησης της ολικής κατανάλωση ενέργειας.

Σε θέμα επίδοσης, σχήμα 6.24, όσο το μέγεθος του issue του επεξεργαστή μεγαλώνει παρατηρείται μία μείωση στους χρόνους εκτέλεσης. . Για issue 1, issue 2, issue 4, issue 8, issue 16 και issue 32 οι χρόνοι εκτέλεσης είναι 72.85 ms, 60.4 ms, 47.94 ms, 37.25 ms, 34.39 ms και 30.35 ms αντίστοιχα. Στην εφαρμογή αυτή παρατηρείται η μεγαλύτερη βελτίωση χρόνων από κάθε άλλη εφαρμογή. Αυτό όμως από μόνο του δεν είναι αρκετό στο να βελτιώσει το Power-Performance efficiency. Σε συνδυασμό λοιπόν της επίδοσης και της κατανάλωσης ενέργειας, η οποία όπως έχει αναφερθεί δεν αυξάνεται συνεχώς με τον ίδιο ρυθμό παρατηρείται άλλοτε το Power-Performance efficiency να αυξάνεται και άλλοτε να μειώνεται.

Το καλύτερο Power-Performance efficiency έχουμε όταν ο αριθμός των issue είναι 2 στο 7.4 με μία διαφορά 2.13 φορές καλύτερο από το baseline configuration που το Power-Performance efficiency είναι 3.47. Το χειρότερο Power-Performance efficiency έχουμε στο baseline configuration όταν δηλαδή ο αριθμός των issue είναι 8 , σε αντίθεση με όλες τις άλλες εφαρμογές.

Συνοψίζοντας, όσο αυξάνουμε το μέγεθος του issue του επεξεργαστή οι χρόνοι εκτέλεσης της εφαρμογής βελτιώνονται συγκριτικά περισσότερο από όλες τις άλλες εφαρμογές . Επίσης σε θέμα κατανάλωσης ενέργειας έχουμε και πάλι διαφορετική συμπεριφορά από όλες τις άλλες εφαρμογές. Στην αρχή ο ρυθμό αύξησης της ολικής κατανάλωση ενέργειας αυξάνεται και στην συνέχεια μειώνεται. Από αυτή λοιπόν την εφαρμογή δεν μπορούμε να εξάγουμε ασφαλή συμπεράσματα για το Power-Performance efficiency . Το ιδανικότερο issue όμως για την εφαρμογή Lu είναι 2, στο οποίο υπάρχει συμφωνία από τις περισσότερες εφαρμογές.

6.5 Power-Performance efficiency για το frequency του επεξεργαστή



Σχήμα 6.26*: Γραφικές παραστάσεις που αναπαριστούν το power-performance efficiency για το όλες τις εφαρμογές από 0.5GHz μέχρι 3.5GHz*

Το frequency του επεξεργαστή είναι ένα configuration του επεξεργαστή το οποίο συνδέεται άμεσα με την κατανάλωση ενέργειας και την επίδοση του επεξεργαστή. Για να δούμε πως η αλλαγή στο frequency του επεξεργαστή επηρεάζει το Power-Performance efficiency κρατήσαμε σταθερά τα υπόλοιπα configurations στο baseline και αλλάξαμε το frequency με τιμές από 0.5GHz μέχρι 3.5GHz και για τα 6 benchmarks που έχουν επιλεχθεί.

Από τις πιο πάνω γραφικές παραστάσεις, σχήμα 6.15, παρατηρείται ότι όσο αυξάνουμε το frequency του επεξεργαστή, (που είναι το ίδιο για όλους τους πυρήνες) αυξάνεται και το Power-Performance efficiency. Γενικά φαίνεται ότι όλες οι εφαρμογές συμφωνούν και δεν υπάρχουν μεγάλες διαφορές στον τρόπο με τον οποίο αυξάνεται το Power-Performance efficiency σε κάθε μία εφαρμογή ξεχωριστά. Αυτό είναι κάτι το οποίο αναμέναμε, αφού είναι γενικά αποδεκτό ότι με την αύξηση του frequency του επεξεργαστή μειώνεται ο χρόνος εκτέλεσης, δηλαδή έχουμε καλύτερες επιδόσεις. Επίσης με την αύξηση του frequency του επεξεργαστή αυξάνεται και η κατανάλωση ενέργειας.

Η κατανάλωση ενέργειας είναι ανάλογη με το frequency του επεξεργαστή. Δηλαδή αυξάνοντας το frequency του επεξεργαστή αυξάνεται ανάλογα και η κατανάλωση ενέργειας ενώ μειώνοντας το frequency του επεξεργαστή μειώνεται ανάλογα και η κατανάλωση ενέργειας. Ενδεικτικά η κατανάλωση ενέργειας για την εφαρμογή Barnes για 0.5GHz, 1GHz, 2GHz, 3GHz και 3.5GHZ είναι 54w, 98w, 184w, 271w, και 315w αντίστοιχα. Ανάλογες είναι και οι αυξήσεις στην κατανάλωση ενέργειας και στις υπόλοιπες εφαρμογές.

Αυτό το οποίο επηρεάζει θετικά το Power-Performance efficiency, το οποίο παρατηρείται να αυξάνεται είναι η επίδοση. Η επίδοση αυξάνεται αρκετά καθώς αυξάνεται το frequency του επεξεργαστή, κάτι το οποίο δικαιολογεί την συνεχής αύξηση του Power-Performance efficiency. Ενδεικτικά οι χρόνοι εκτέλεσης για την εφαρμογή Barnes για 0.5GHz, 1GHz, 2GHz, 3GHz και 3.5GHZ είναι 563.28 ms, 281.64 ms, 140.82 ms, 93.88 ms, και 80.46 ms αντίστοιχα. Ανάλογες είναι και οι βελτιώσεις στους χρόνους εκτέλεσης των υπόλοιπων εφαρμογών.

Η βελτίωση που παρατηρείται στο Power-Performance efficiency από το baseline που είναι στα 2GHz μέχρι και το πιο μεγάλο frequency στα 3.5GHz είναι της τάξης του 310-340%, και με την πιο μεγάλη βελτίωση να γίνεται στην εφαρμογή Radix με 3.41 φορές καλύτερο Power-Performance efficiency στα 3.5GHz από τα 2GHz. Από την άλλη η πιο μικρή βελτίωση του efficiency γίνεται στην εφαρμογή Barnes με 3.13 φορές καλύτερο Power-

Performance efficiency στα 3.5GHz από τα 2GHz. Η εφαρμογή Barnes όπως φαίνεται και από το πίνακα 5.1, είναι η εφαρμογή η οποία χρειάζεται περισσότερο την μνήμη και λιγότερο τον επεξεργαστή. Σε αντίθεση η εφαρμογή Radix δεν χρειάζεται πολύ την μνήμη αλλά περισσότερο την υπολογιστική ισχύ. Αυτός είναι και ο λόγος για τον οποίο παρατηρούνται αυτά τα αποτελέσματα για το Power-Performance efficiency. Επίσης η εφαρμογή FFT έχει χαμηλό βαθμό αύξησης του Power-Performance efficiency όσο αυξάνεται το frequency για τον λόγο ότι καταπονεί περισσότερο τον επεξεργαστή σε θέμα κατανάλωσης ενέργειας όπως φαίνεται και στο σχήμα 6.1.

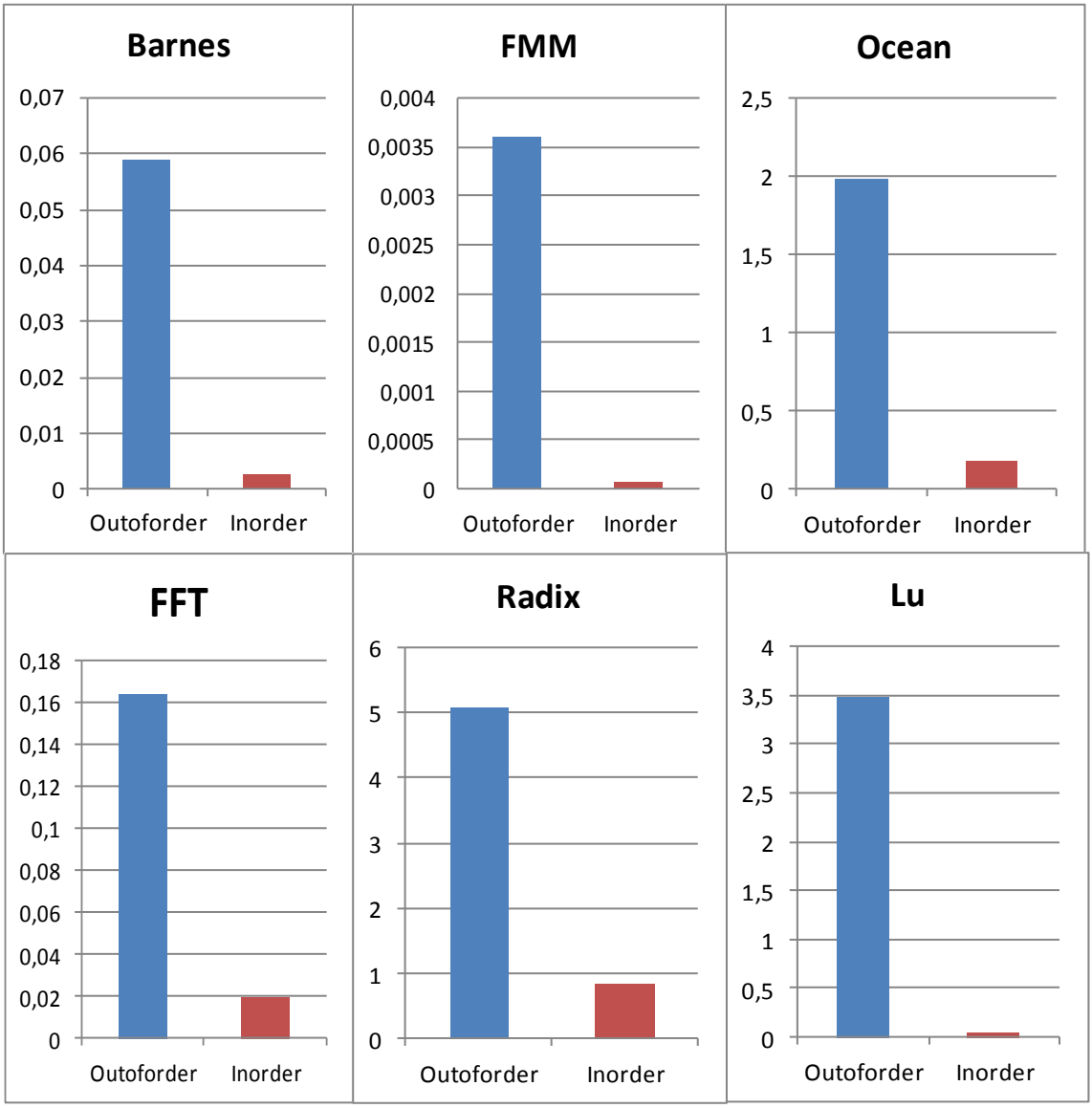
Γενικά εφαρμογές οι οποίες χρειάζονται περισσότερο την μνήμη παρά την υπολογιστική ισχύ του επεξεργαστή έχουν μικρότερη αύξηση στο Power-Performance efficiency από τις άλλες όσο αυξάνεται το frequency του επεξεργαστή.

Η μείωση που παρατηρείται στο efficiency από το baseline που είναι στα 2GHz μέχρι και το πιο μικρό frequency στα 0.5GHz είναι μεταξύ 16 και 21 φορές μικρότερη στα 0.5GHz από τα 2GHz, με την μεγαλύτερη μείωση να γίνεται στην εφαρμογή Radix και την μικρότερη μείωση στην εφαρμογή Lu .

Το διάστημα στο οποίο έχουμε την μεγαλύτερη αύξηση του Power-Performance efficiency είναι το διάστημα από 0.5GHz σε 1GHz σε μέσο όρο από όλες τις εφαρμογές. Αυτό συμβαίνει γιατί η σχέση με την οποία μετρούμε το Power-Performance efficiency βασίζεται σε σύγχρονους επεξεργαστές, (μετρώντας την επίδοση με MIPS^3) και το frequency στα 0.5GHz περιορίζει τον επεξεργαστή σε πολύ χαμηλές επιδόσεις.

Ο μέσος όρος αύξησης του Power-Performance efficiency ανά 0.1GHz είναι 1.93 για το Barnes, 2 για το FMM για το Lu και για το FFT , 2.21 για το Ocean και 2.4 για το Radix και . Με άλλα λόγια αυξάνοντας το frequency του επεξεργαστή κατά 0.1GHz έχουμε μέσο όρο 2 φορές καλύτερο Power-Performance efficiency. Άρα είναι σημαντικό οι επεξεργαστές να έχουν ψηλό frequency αν στοχεύουν στο να έχουν το καλύτερο Power-Performance efficiency.

6.6 Power-Performance efficiency για Out-of-order και In-order εκτέλεση



Σχήμα 6.27*: Γραφικές παραστάσεις που αναπαριστούν το power-performance efficiency για out-of-order και in-order εκτέλεση για το όλες τις εφαρμογές.*

Η In-order εκτέλεση και η Out-of-Order εκτέλεση αποτελούν δύο τρόπους με τους οποίους ένας επεξεργαστής μπορεί να εκτελεί της εντολές. Out-of-order εκτέλεση έχουμε όταν ο επεξεργαστής εκτελεί τις εντολές χωρίς να παίζει ρόλο η σειρά με την οποία έρχονται η εντολές, σε αντίθεση με την Out-of-order, στην οποία οι εντολές πρέπει να εκτελούνται με την σειρά. Για να δούμε την διαφορά του Power-Performance efficiency μεταξύ In-order και Out-of-order εκτέλεση κρατήσαμε σταθερά τα υπόλοιπα configurations στο baseline και αλλάξαμε το Execution και για τα 6 benchmarks που έχουν επιλεχθεί.

Από τις πιο πάνω γραφικές παραστάσεις, σχήμα 6.27, παρατηρείται ότι υπάρχει μεγάλη διαφορά στο Power-Performance efficiency μεταξύ Out-of-order και In-order εκτέλεσης. Σε όλες τις εφαρμογές μπορεί να γίνει αυτή η παρατήρηση με , σε μερικές η διαφορά να είναι πολύ μεγάλη και σε άλλες πάρα πολύ μεγάλη. Η διαφορά αυτή οφείλετε στην μεγάλη διαφορά που υπάρχει στον χρόνο εκτέλεσης μεταξύ Out-of-order και In-order εκτέλεσης, κάτι απόλυτα λογικό , να χρειάζεται δηλαδή μια εφαρμογή περισσότερο χρόνο όταν είναι αναγκασμένη να εκτελεί τις εντολές της με την σειρά, αφού όταν γίνεται αυτό γίνονται πολλές καθυστερήσεις λόγω των πολλών εξαρτήσεων.

Σε θέμα κατανάλωσης ενέργειας έχουμε μια σημαντική διαφορά μεταξύ των δύο, με την In-order εκτέλεση να έχει λιγότερη κατανάλωση ενέργειας από την Out-of-order. Αυτό συμβαίνει γιατί στην περίπτωση της In-order εκτέλεσης ο επεξεργαστής χρησιμοποιεί in-order pipelining ενώ για Out-of-order το αντίστοιχο Out-of-order pipelining. Η διαφορά μεταξύ τους είναι ότι το μεν In-order pipelining είναι λιγότερο περίπλοκο να υλοποιηθεί σαν κύκλωμα στο chip και χρειάζεται πολύ λιγότερο χώρο από ότι το Out-of-order pipelining που είναι πολύ περίπλοκο.

Η μεγαλύτερη αύξηση του Power-Performance efficiency παρατηρείται στην εφαρμογή Lu , 69.4 φορές μεγαλύτερο στην out-of-order εκτέλεση από ότι στην in-order . Αυτό συμβαίνει διότι υπάρχει η μεγαλύτερη διαφορά σε χρόνο εκτέλεσης μεταξύ out-of-order και in-order εκτέλεσης. Ο λόγος είναι ότι η εφαρμογή Lu έχει τον κώδικα με τις περισσότερες εξαρτήσεις, κάτι το οποίο κάνει την in-order εκτέλεση πολύ αργή. Επίσης αυτό το φαινόμενο παρατηρείται και στην εφαρμογή FMM. Το αντίθετο γίνεται με την εφαρμογή Radix η οποία έχει την μικρότερη αύξηση του efficiency, 6 φορές μεγαλύτερο στην out-of-order εκτέλεση, γιατί το πρόγραμμα αυτό δεν έχει πολλές εξαρτήσεις, λόγω του ότι υλοποιεί έναν επαναληπτικό αλγόριθμο ταξινόμησης

Η Out-of-order εκτέλεση είναι πολύ καλύτερη σε θέμα επίδοσης από την in-order εκτέλεση. Αυτό έχει σαν αποτέλεσμα να έχει και πολύ καλύτερο Power-Performance efficiency. Άρα η Out-of-order εκτέλεση είναι πολύ καλύτερη όταν στοχεύουμε σε καλύτερο Power-Performance efficiency από την In-order.

6.7 Power-Performance efficiency για το Cache line size

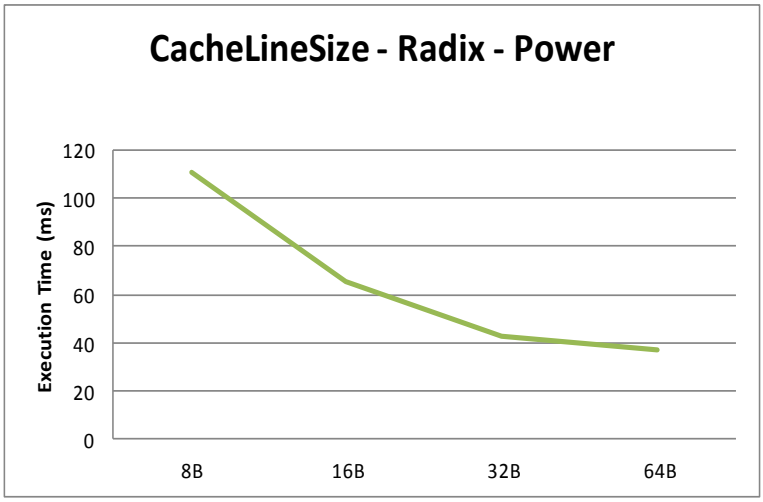
Το Cache line size είναι ένα configuration το οποίο συνδέεται άμεσα με την cache του επεξεργαστή. Για να δούμε πως η αλλαγή στο Cache line size επηρεάζει το Power-Performance efficiency κρατήσαμε σταθερά τα υπόλοιπα configurations στο baseline και αλλάξαμε το Cache line size με τιμές από 8 μέχρι 64 και για τα 6 benchmarks που έχουν επιλεχθεί.

Το Cache line size είναι μία σημαντική παράμετρος που έχει μελετηθεί για τον λόγω ότι σχετίζεται με άλλες παραμέτρους της cache του επεξεργαστή. Συγκεκριμένα οι παράμετροι με τις οποίες σχετίζεται το Cache line size φαίνονται στο πίνακα 6.3.

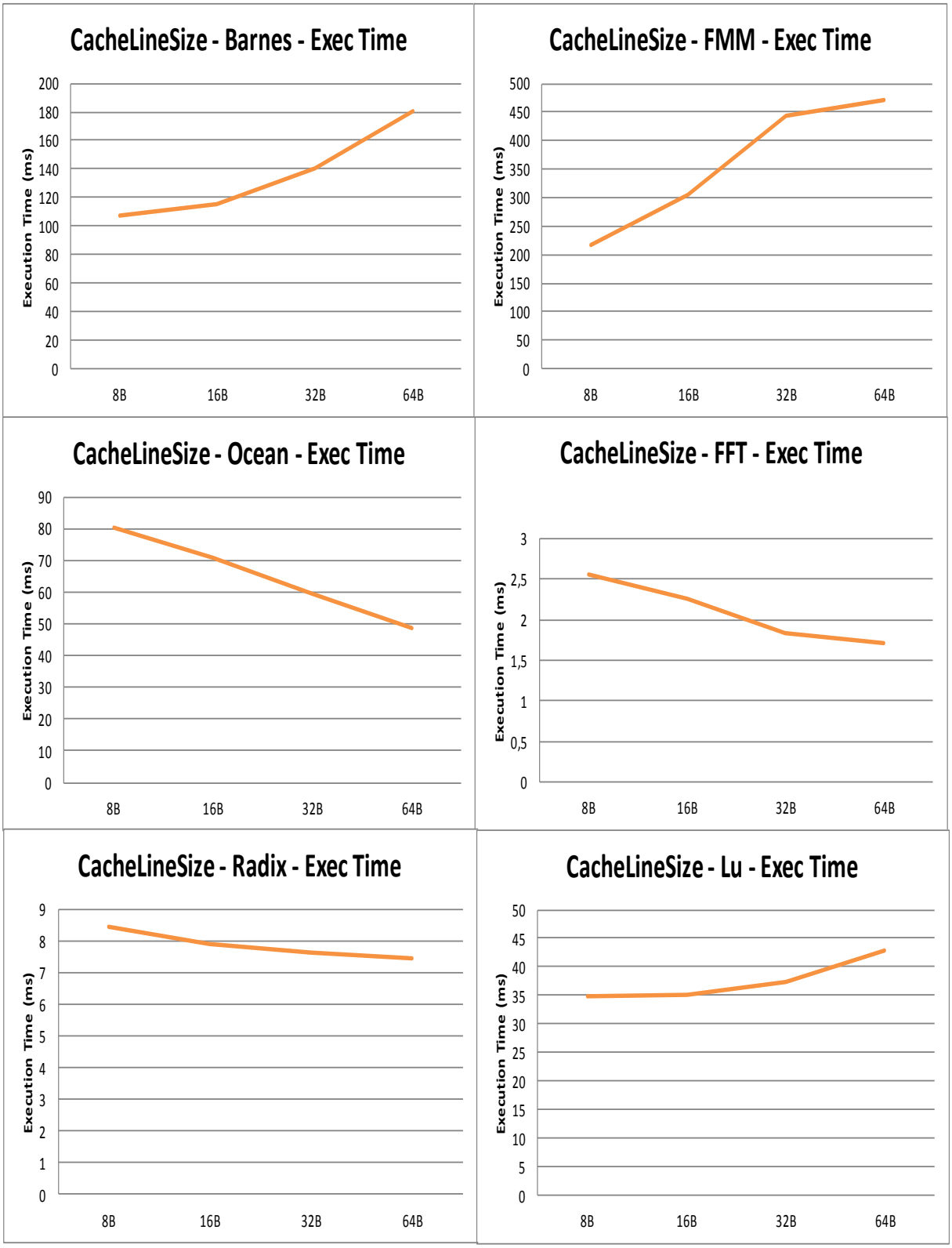
***Πίνακας 6.3**: Ισοδυναμία* CacheLineSize *με άλλες παραμέτρους.*

Παράμετροι	**Ισοδυναμία**
IL1 bsize	$(CacheLineSize)
DL1 bsize	$(CacheLineSize)
L2 bsize	$(CacheLineSize)
[MemoryBus] portOccp	$(CacheLineSize) / 4

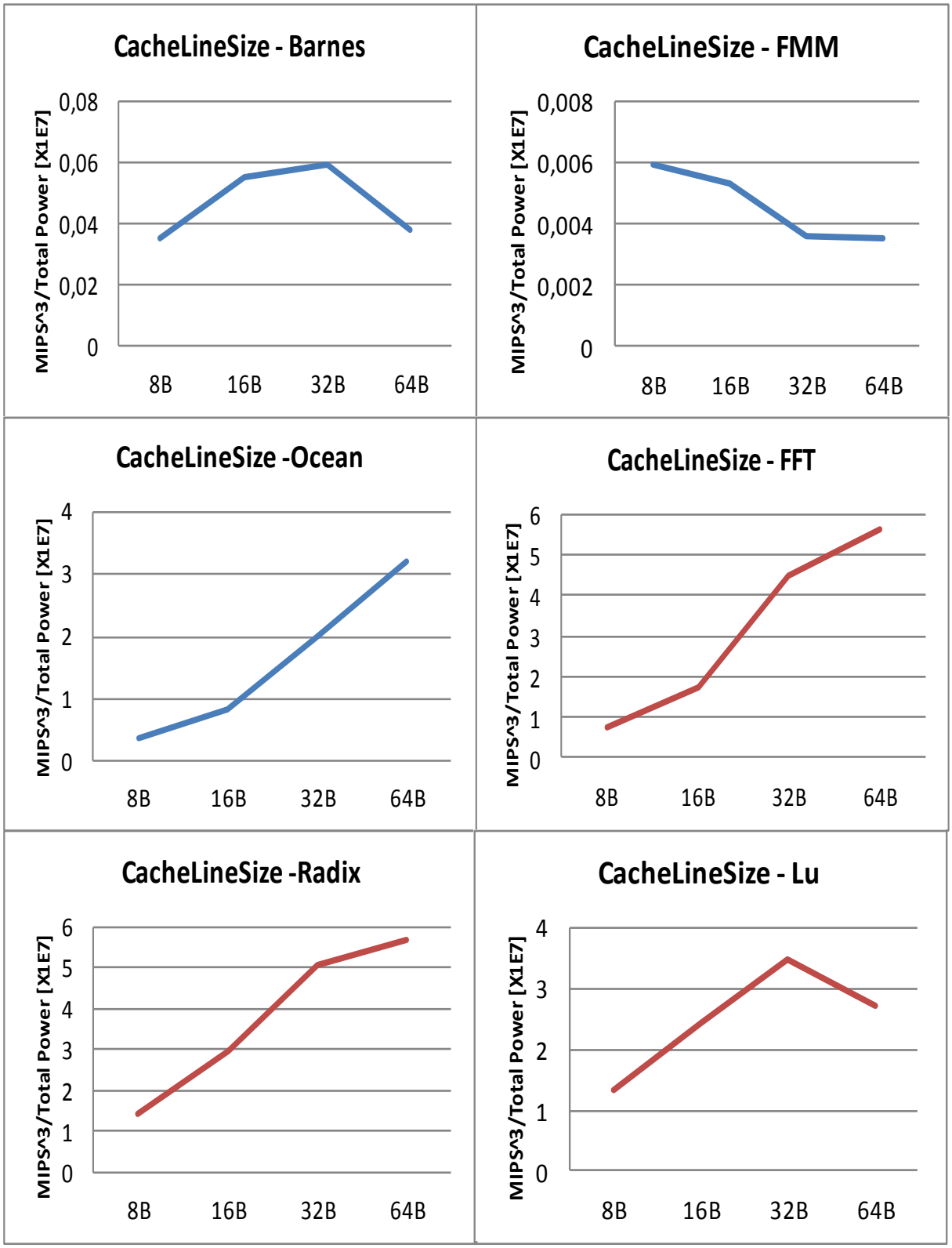
Όπως φαίνεται στο πίνακα 6.3 τα block size του πρώτου επιπέδου της cache και του δευτέρου είναι ίσα και ισοδύναμα με το Cache line size. Επίσης με το Cache line size σχετίζεται και η παράμετρος portOccp.



***Σχήμα 6.28:** Γραφική παράσταση που αναπαριστά την κατανάλωση ενέργειας για την εφαρμογή Radix για Cache line size μέχρι και 64.*



Σχήμα 6.29: *Γραφικές παραστάσεις που αναπαριστούν τους χρόνους εκτέλεσης για όλες τις εφαρμογές για Cache line size μέχρι και 64.*



Σχήμα 6.30: *Γραφικές παραστάσεις που αναπαριστούν το power-performance efficiency για όλες τις εφαρμογές για Cache line size μέχρι και 64.*

Από τις πιο πάνω γραφικές παραστάσεις, σχήμα 6.30, παρατηρείται ότι οι διάφορες εφαρμογές έχουν διαφορετική συμπεριφορά, όσο αφορά το Power-Performance efficiency, όταν αλλάζει το μέγεθος του Cache line size. Σε μερικές εφαρμογές παρατηρείται συνεχείς αύξηση όσο μεγαλώνει το μέγεθος του Cache line size, ενώ σε άλλες παρατηρείται αύξηση μέχρι σε ένα σημείο και στην συνέχεια έχουμε μείωση. Υπάρχουν και εφαρμογές που βλέπουμε συνεχείς μείωση.

Σε θέμα κατανάλωσης ενέργειας, σχήμα 6.28, όσο αυξάνουμε το μέγεθος του Cache line size παρατηρείται αρκετά μεγάλη μείωση στην ολική κατανάλωση ενέργειας. Στο σχήμα 6.28 φαίνεται η κατανάλωση ενέργειας για την εφαρμογή Radix, ενώ με παρόμοιο τρόπο η κατανάλωση μειώνεται και στις άλλες εφαρμογές. Από το σημείο του Cache line size 8 μέχρι και το Cache line size 16 η μείωση αυτή είναι αρκετά μεγάλη. Συγκεκριμένα για μέγεθος 16 παρατηρείται 1.8 μέχρι 2.6 φορές μικρότερη ολική κατανάλωση ενέργειας από ότι σε μέγεθος 8. Στην συνέχεια από 16 μέχρι 32 Cache line size η μείωση στην ολική κατανάλωση ενέργειας είναι πιο μικρή. Για μέγεθος 32 παρατηρείται 1.7 μέχρι 1.2 φορές μικρότερη ολική κατανάλωση ενέργειας από ότι σε μέγεθος 16. Τέλος από 32 μέχρι 64 Cache line size η μείωση στην ολική κατανάλωση ενέργειας μικραίνει ακόμη περισσότερο.

Σε θέμα επίδοσης, σχήμα 6.29, σε μερικές εφαρμογές παρατηρείται μείωση στους χρόνους εκτέλεσης, όσο αυξάνεται το μέγεθος του Cache line size, ενώ σε άλλες γίνεται ακριβώς το αντίθετο δηλ. παρατηρείται αύξηση στους χρόνους εκτέλεσης. Υπάρχουν και ενδιάμεσες περιπτώσεις, όπου δηλαδή έχουμε μείωση και αύξηση στους χρόνους εκτέλεσης για μία εφαρμογή. Συγκεκριμένα για την εφαρμογή Barnes για 8,16, 32 και 64 Cache line size παρατηρούνται οι χρόνοι εκτέλεσης 106.88, 114.98, 140.82 και 180.33 αντίστοιχα. Οι χρόνοι εκτέλεσης αυξάνονται αλλά το Power-Performance efficiency βελτιώνεται μέχρι και Cache line size 32 λόγω της μεγάλης μείωσης κατανάλωσης ενέργειας όπως έχει αναφερθεί. Για Cache line size 64 παρατηρείται μείωση του efficiency για τον λόγο ότι η επίδοση χειροτερεύει αρκετά και σε συνδυασμό με την μειωμένη μείωση της κατανάλωσης ενέργειας επηρεάζεται αρνητικά το efficiency. Για τις εφαρμογές Ocean, FFT και Radix όσο αυξάνεται το μέγεθος του Cache line size οι χρόνοι εκτέλεσης μειώνονται, γεγονός που δικαιολογεί την συνεχής αύξηση του efficiency. Για την εφαρμογή FMM 8,16, 32 και 64 Cache line size παρατηρούνται οι χρόνοι εκτέλεσης 217.43, 305.03, 443.69 και 657.98 αντίστοιχα. Δηλαδή οι χρόνοι εκτέλεσης αυξάνονται αρκετά γεγονός που επηρεάζει αρνητικά το efficiency, το οποίο μειώνεται συνεχώς, παρά τις μεγάλες μειώσεις τις κατανάλωσης ενέργειας. Τέλος για την εφαρμογή Lu οι χρόνοι εκτέλεσης αυξάνονται ελάχιστα εκτός από 32 μέχρι 64 Cache

line size όπου παρατηρείται αρκετά μεγάλη αύξηση γι' αυτό και σε εκείνο το σημείο το Power-Performance efficiency μειώνεται.

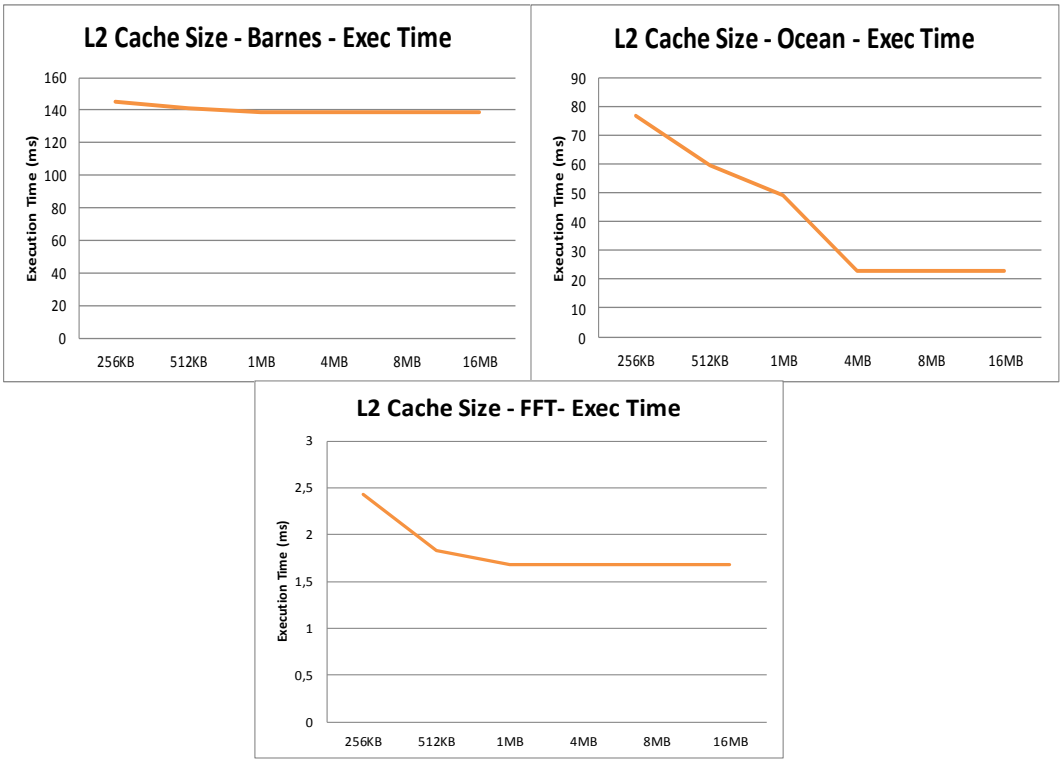
Στις εφαρμογές Ocean, FFT και Radix παρατηρείται μία συνεχείς αύξηση στο Power-Performance efficiency όπως έχει ήδη αναφερθεί. Ο μέσος όρος αύξησης του efficiency για τις εφαρμογές αυτές συγκριτικά, ανάμεσα σε κάθε αύξηση στο μέγεθος του Cache line size είναι 2.27 φορές. Με άλλα λόγια αυξάνοντας το μέγεθος του Cache line size στο διπλάσιο έχουμε 2.27 φορές μεγαλύτερο Power-Performance efficiency για τις εφαρμογές αυτές. Αυτό γίνεται γιατί αυτές οι εφαρμογές έχουν μεγάλα miss rates. Όσο αυξάνεται το Cache line size τα ποσοστά αυτά μειώνονται και η επίδοση αυξάνεται.

Στις εφαρμογές Barnes και Lu παρατηρείται μία αύξηση στο Power-Performance efficiency μέχρι και Cache line size 32 ενώ στην συνέχεια παρατηρείται μείωση στο Power-Performance efficiency, όπως έχει ήδη αναφερθεί. Για την εφαρμογή Lu ο μέσος όρος αύξησης του efficiency συγκριτικά, ανάμεσα σε κάθε αύξηση στο μέγεθος του Cache line size είναι 1.1 φορές, δηλαδή αυξάνοντας το μέγεθος του Cache line size στο διπλάσιο έχουμε 1.1 φορές μεγαλύτερο Power-Performance efficiency. Για την εφαρμογή Barnes ο μέσος όρος αύξησης του efficiency συγκριτικά, ανάμεσα σε κάθε αύξηση στο μέγεθος του Cache line size είναι 0.7 φορές, δηλαδή αυξάνοντας το μέγεθος του Cache line size στο διπλάσιο έχουμε 0.7 φορές χειρότερο Power-Performance efficiency, ενώ για την εφαρμογή FMM η τιμή αυτή είναι ακόμα πιο χαμηλή. Οι εφαρμογές αυτές έχουν χαμηλά miss rates και μετά από ένα συγκεκριμένο Cache line size δεν χαμηλώνουν άλλο.

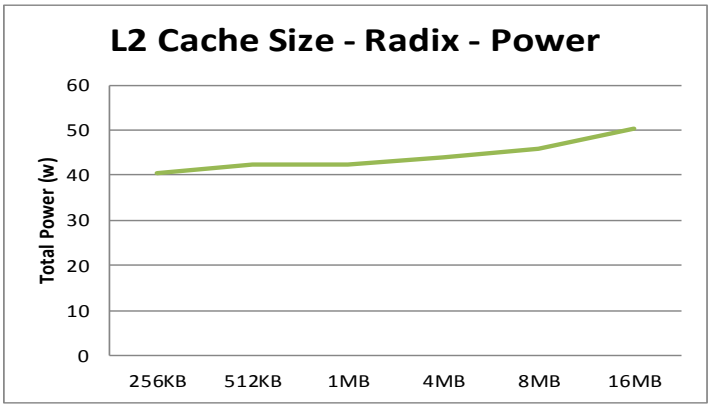
Συνοψίζοντας, όσο αυξάνουμε το Cache line size η κατανάλωση ενέργειας μειώνεται, ενώ σε θέμα επίδοσης άλλοτε αυξάνεται και άλλοτε μειώνεται για τον λόγο ότι η επίδοση εξαρτάτε από την δομή της εφαρμογής και από την χρήση της μνήμης που κάνει μία εφαρμογή. Επίσης εφαρμογές με υψηλό miss rate είναι καλό να εκτελούνται με μεγάλο Cache line size για τον λόγο ότι μειώνει το miss rate. Από τα πιο πάνω λοιπόν και το σχήμα 6.30, για τις περισσότερες εφαρμογές είναι καλό έχουμε μεγάλο Cache line size αν στοχεύουμε στο καλύτερο Power-Performance efficiency.

6.8 Power-Performance efficiency για την cache δευτέρου επιπέδου

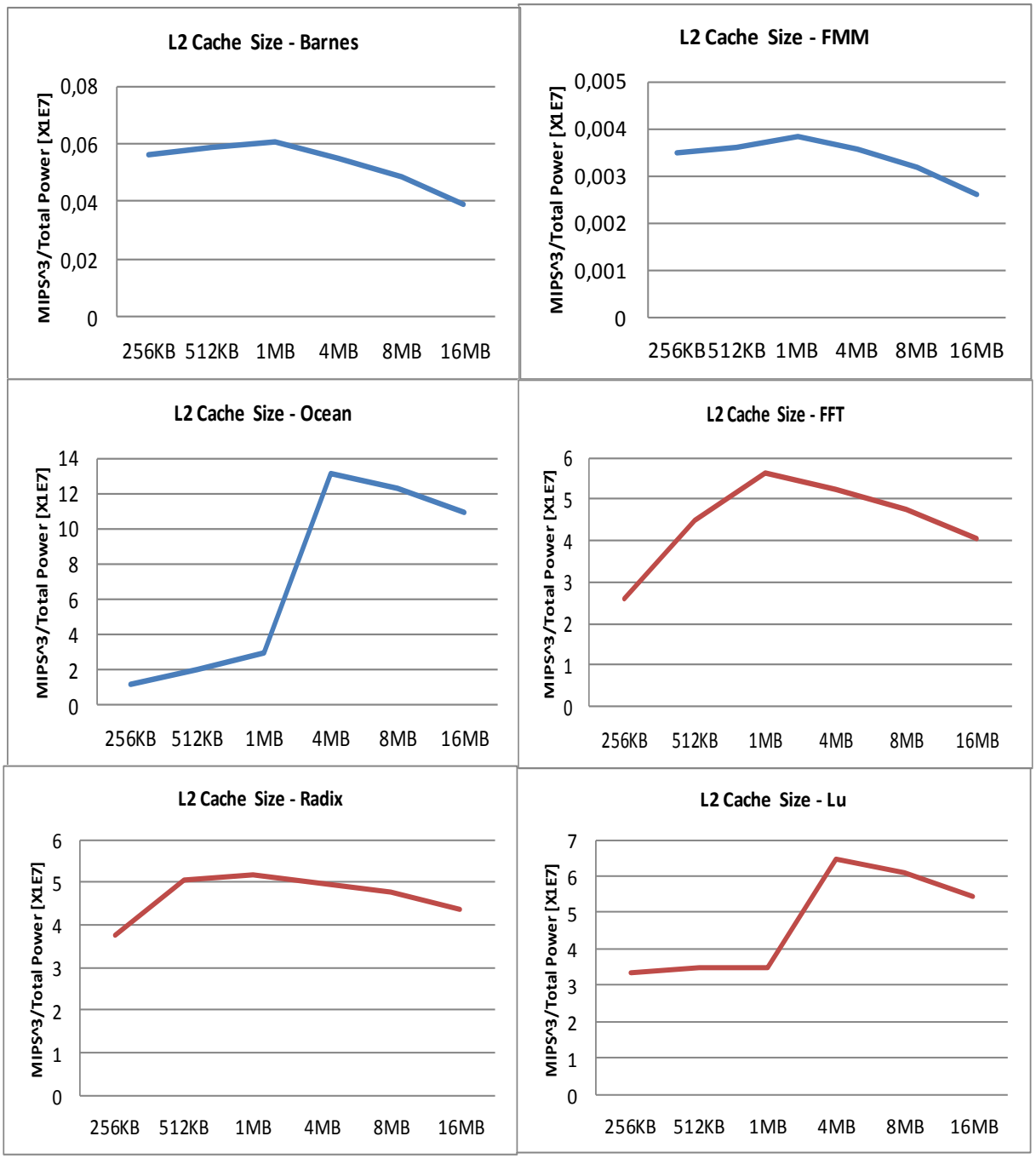
Για το δεύτερο επίπεδο της cache τα configurations που εξετάστηκαν στο πείραμα είναι το μέγεθος και το associativity. Για να δούμε πως η αλλαγή στο μέγεθος και το associativity της L2 cache επηρεάζει το Power-Performance efficiency κρατήσαμε σταθερά τα υπόλοιπα configurations στο baseline και αλλάξαμε το μέγεθος με τιμές από 256KB μέχρι 16MB και το associativity με τιμές από 1 μέχρι 32 και για τα 6 benchmarks που έχουν επιλεχθεί.



Σχήμα 6.31: *Γραφικές παραστάσεις που αναπαριστούν τους χρόνους εκτέλεσης για τις εφαρμογές Barnes, Ocean και fft για μέγεθος της L2 cache μέχρι και 16MB.*



Σχήμα 6.32: *Γραφική παράσταση που αναπαριστά την κατανάλωση ενέργειας για την εφαρμογή Radix για μέγεθος της L2 cache μέχρι και 16MB.*



Σχήμα 6.33*: Γραφικές παραστάσεις που αναπαριστούν το power-performance efficiency για όλες τις εφαρμογές για μέγεθος της L2 cache μέχρι και 16MB.*

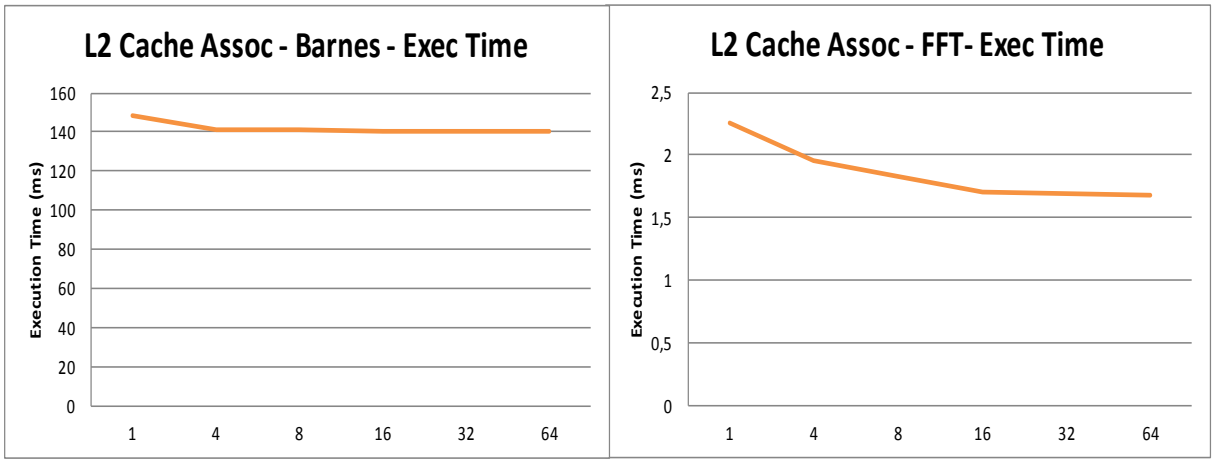
Από τις πιο πάνω γραφικές παραστάσεις, σχήμα 6.33, παρατηρείται ότι οι διάφορες εφαρμογές έχουν διαφορετική συμπεριφορά όταν αλλάζει το μέγεθος της L2 cache, κάτι που είναι φυσιολογικό αφού μια εφαρμογή μπορεί να χρησιμοποιεί περισσότερο την μνήμη από μια άλλη. Παρατηρείται ότι σε μερικές εφαρμογές το Power-Performance efficiency δεν αλλάζει καθώς αυξάνουμε το μέγεθος της L2 cache, ενώ σε άλλες παρατηρούνται

αξιοσημείωτες αλλαγές. Όσο αυξάνουμε το μέγεθος της L2 cache έχουμε μια λογική αύξηση στην κατανάλωση ενέργειας, σχήμα 6.32,. Αυτή η αύξηση στην ολική κατανάλωση ενέργειας δεν είναι πολύ μεγάλη, σε βαθμό που να επηρεάζει σε μεγάλο βαθμό την σχέση με την οποία μετρούμε το Power-Performance efficiency. Αυτό λοιπόν που επηρεάζει πάρα πολύ το Power-Performance efficiency είναι η επίδοση, δηλαδή κατά πόσο θα έχουμε μείωση στον χρόνο εκτέλεσης μίας εφαρμογής όταν αλλάζουμε το μέγεθος της L2 cache.

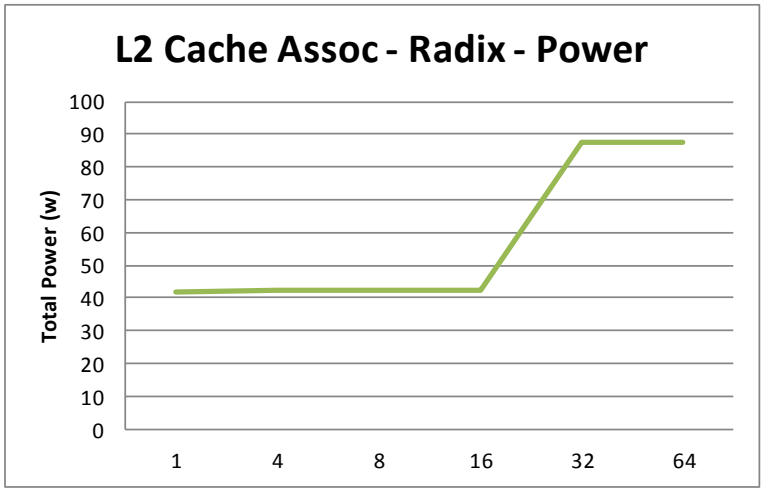
Για την εφαρμογή Barnes δεν έχουμε μεγάλες μεταβολές στο Power-Performance efficiency. Όσο αυξάνουμε το μέγεθος της L2 cache έχουμε μία σταθερά μικρή μείωση στον χρόνο εκτέλεσης, σχήμα 6.31, της εφαρμογής, κάτι που γίνεται μέχρι και το 1MB. Στην συνέχεια οι χρόνοι εκτέλεσης δεν βελτιώνονται και έτσι παρατηρείται μία μείωση του Power-Performance efficiency για μέχρι και 16MB. Αυτή η συμπεριφορά παρατηρείται στις εφαρμογές FMM, FFT και Radix. Στην εφαρμογή FFT παρατηρείται η πιο απότομη βελτίωση του efficiency γιατί χρειάζεται το πιο μεγάλο μέγεθος μνήμης από της άλλες. Γενικά όμως, οι εφαρμογές αυτές δεν χρειάζονται μεγάλο μέγεθος για την μνήμη για τον λόγω ότι δεν δεσμεύουν μεγάλο μέγεθος μνήμης και επίσης δεν δεσμεύουν συνεχόμενη μνήμη.

Για τις εφαρμογές Ocean και Lu βλέπουμε μία διαφορετική αλλά παρόμοια συμπεριφορά. Όπως και στις άλλες εφαρμογές έχουμε μία σταθερή μικρή βελτίωση του Power-Performance efficiency μέχρι και το 1MB. Στην συνέχεια μέχρι τα 4MB έχουμε μία απότομη μείωση του χρόνου εκτέλεσης, σχήμα 6.31, της εφαρμογής κάτι που αυξάνει κατά πολύ το Power-Performance efficiency. Αυτή η βελτίωση της επίδοσης οφείλετε στο γεγονός ότι η εφαρμογές αυτές χρειάζεται να δεσμεύσουν συνεχόμενη μνήμη κάτι το οποίο μπορεί να γίνει χωρίς πολλές καθυστερήσεις όταν η μνήμη είναι αρκετά μεγάλη. Ακολούθως από 4MB μέχρι 16MB παρατηρείται μία σταθερή μείωση στο Power-Performance efficiency για τον λόγο ότι δεν έχουμε βελτίωση στην επίδοση. Σε σημείο των 4MB έχουμε το μέγιστο μέγεθος μνήμης που χρειάζονται οι πιο πάνω εφαρμογές για να εκτελεστούν χωρίς καθυστερήσεις.

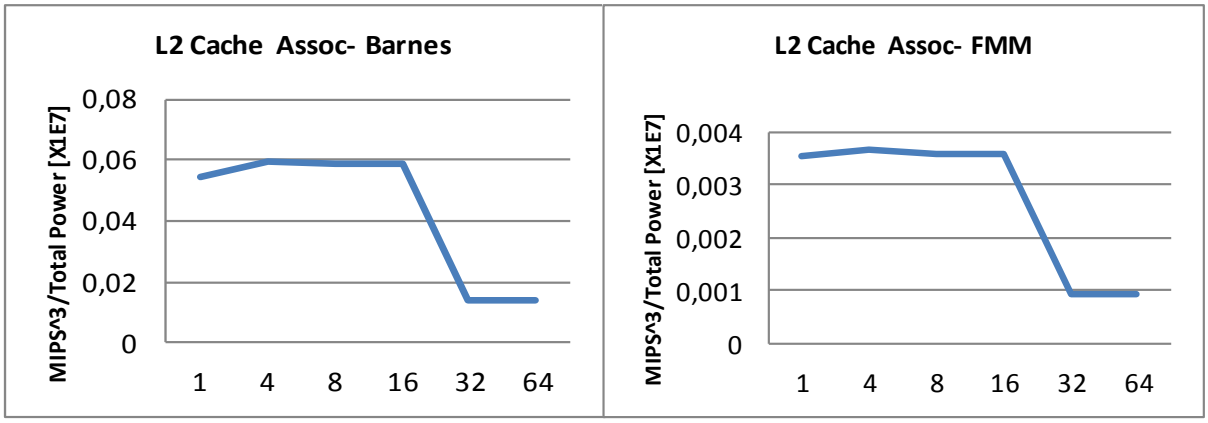
Συνοψίζοντας, όσο αυξάνουμε το size της L2 Cache η κατανάλωση ενέργειας αυξάνεται με χαμηλό ρυθμό , ενώ σε θέμα επίδοσης παρατηρείται βελτίωση μέχρι σε ένα σημείο, πράγμα απόλυτα φυσιολογικό. Από τα πιο πάνω και το σχήμα 6.33, μπορούμε να συμπεράνουμε ότι το καλύτερο μέγεθος για L2 cache κυμαίνεται από 1MB μέχρι 4MB.

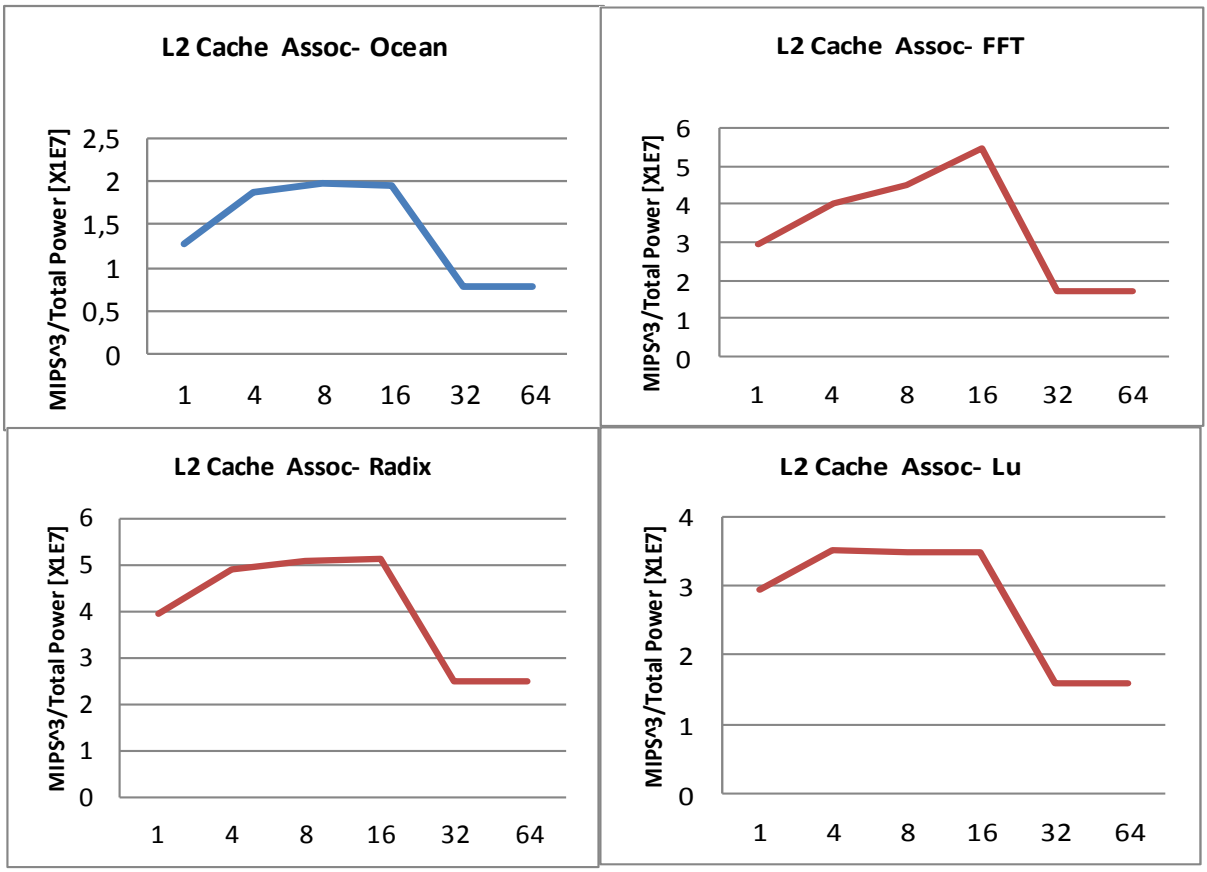


Σχήμα 6.34: *Γραφικές παραστάσεις που αναπαριστούν τους χρόνους εκτέλεσης για τις εφαρμογές Barnes και fft για associativity της L2 cache μέχρι 64.*



Σχήμα 6.35: *Γραφική παράσταση που αναπαριστά την κατανάλωση ενέργειας για την εφαρμογή Radix για associativity της L2 cache μέχρι 64.*





Σχήμα 6.36: *Γραφικές παραστάσεις που αναπαριστούν το power-performance efficiency για όλες τις εφαρμογές για associativity της L2 cache μέχρι 64.*

Από τις πιο πάνω γραφικές παραστάσεις, σχήμα 6.36, παρατηρείται ότι οι διάφορες εφαρμογές έχουν μία παρόμοια συμπεριφορά όσο αφορά το Power-Performance efficiency ,όταν αλλάζει το associativity της L2 cache. Όσο αυξάνουμε το associativity της L2 cache έχουμε μια λογική αύξηση στην κατανάλωση ενέργειας. Η ολική κατανάλωση ενέργειας αυξάνεται με ένα σταθερά μικρό βαθμό μέχρι και associativity 16. Από 16 μέχρι και 32 έχουμε μία μεγάλη αύξηση της ολική κατανάλωση ενέργειας, γεγονός που μειώνει αρκετά το Power-Performance efficiency.

Σε θέμα επίδοσης, σχήμα 6.34, η μεγαλύτερη μείωση στον χρόνο εκτέλεσης των εφαρμογών παρατηρείται στο σημείο μεταξύ associativity 1 και associativity 4, με αποτέλεσμα να έχουμε μία αύξηση στο Power-Performance efficiency όλων των εφαρμογών σε αυτό το σημείο. Στην συνέχεια οι χρόνοι εκτέλεσης των εφαρμογών βελτιώνονται, αλλά σε όχι τόσο μεγάλο βαθμό που να επηρεάζουν την σχέση του Power-Performance efficiency.

Συγκεκριμένα στις εφαρμογές Barnes, FMM, Radix και Lu έχουμε την ίδια συμπεριφορά. Το Power-Performance efficiency αυξάνεται αισθητά στο σημείο μεταξύ associativity 1 και associativity 4, για τον λόγο που έχει αναφερθεί. Ακολούθως μέχρι και associativity 16 παρατηρείται μία σταθερά μικρή μείωση του Power-Performance efficiency, ενώ στην συνέχεια έχουμε μια μεγάλη μείωση μέχρι και το associativity 32. Τέλος από 32 μέχρι 64 το Power-Performance efficiency είναι σχεδόν σταθερό. Η μεγαλύτερη αύξηση του efficiency γίνεται στην εφαρμογή radix από associativity 1 μέχρι associativity 4 και είναι της τάξης του 1,24X.

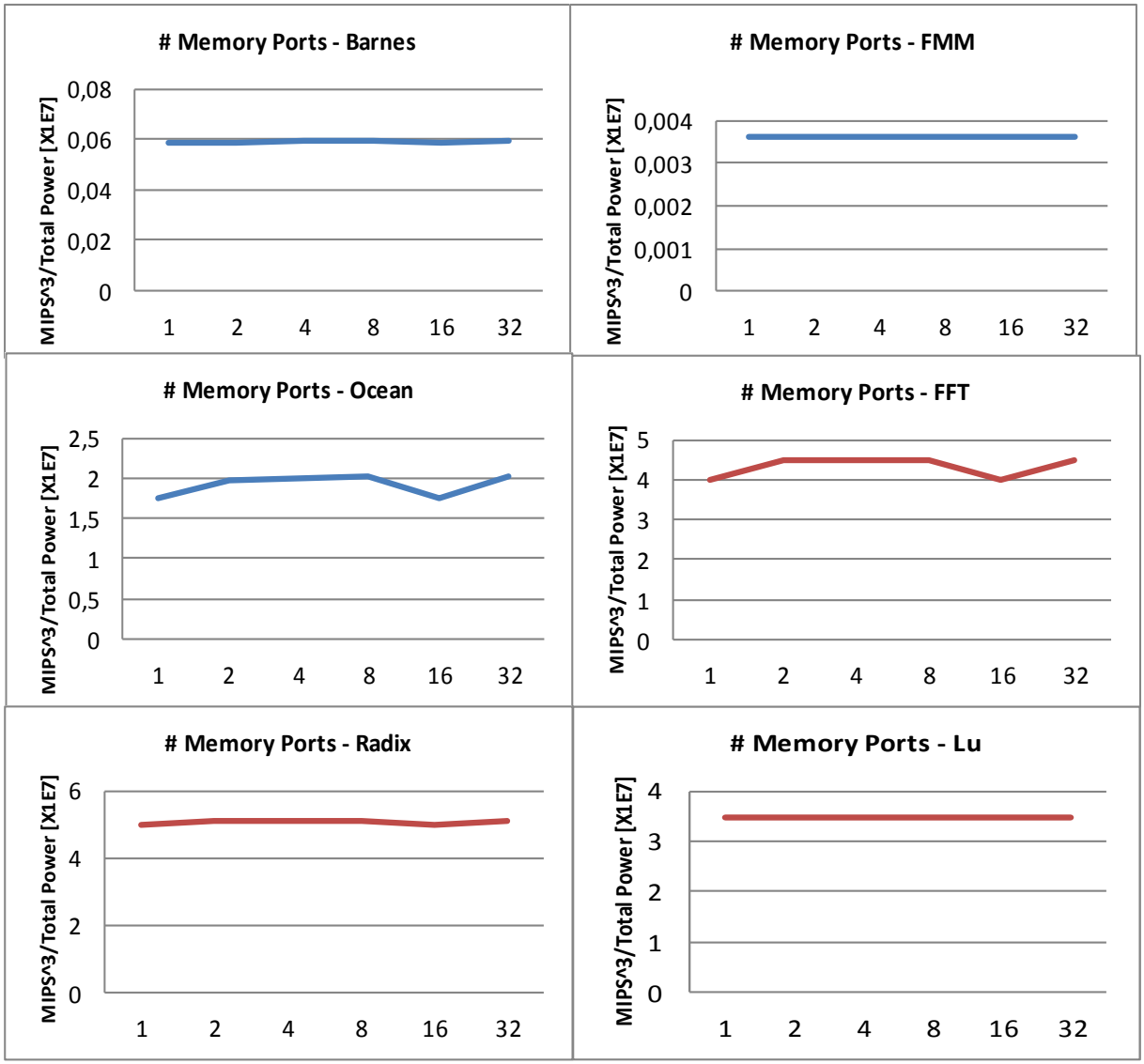
Στην εφαρμογή Ocean έχουμε την ίδια συμπεριφορά με της προηγούμενες, με την μόνη διαφορά ότι στο σημείο από associativity 4 μέχρι associativity 8 δεν παρατηρείται μείωση του Power-Performance efficiency, αλλά αύξηση. Το γεγονός αυτό οφείλεται στο γεγονός ότι ο χρόνος εκτέλεσης της εφαρμογής μειώνεται σε βαθμό που να υπερνικά, την αύξηση της ολικής κατανάλωσης ενέργειας, κάτι το οποίο δεν γίνεται στις άλλες εφαρμογές.

Στην εφαρμογή fft έχουμε μία αρκετά διαφορετική συμπεριφορά. Από associativity 1 μέχρι και associativity 16 το Power-Performance efficiency αυξάνεται αρκετά. Συγκεκριμένα από 1 μέχρι και 4 έχουμε μεγάλη αύξηση, κάτι που ισχύει για όλες τις εφαρμογές. Ακολούθως από 4 μέχρι 8 υπάρχει αύξηση αλλά με μικρότερο ρυθμό. Μετά από 8 μέχρι 16 ο ρυθμός αύξησης αυξάνεται και πάλι, κάτι που βλέπουμε να γίνεται μόνο σε αυτή την εφαρμογή, και ο λόγος είναι η μείωση των χρόνων εκτέλεσης, σχήμα 6.34. Στην συνέχεια έχουμε την ίδια συμπεριφορά με της άλλες εφαρμογές λόγω της απότομης αύξησης της ολικής κατανάλωσης ενέργειας.

Συνοψίζοντας, όσο αυξάνουμε το associativity της L2 Cache η κατανάλωση ενέργειας αυξάνεται με χαμηλό ρυθμό , εκτός από το σημείο 16 -32 , ενώ σε θέμα επίδοσης παρατηρείται μικρή βελτίωση, εκτός από το σημείο 1 -4 . Από τα πιο πάνω και το σχήμα 6.36 μπορούμε να συμπεράνουμε ότι το καλύτερο associativity για L2 cache κυμαίνεται από 4 μέχρι 16.

6.9 Power-Performance efficiency για τον αριθμό των memory ports

Το τελευταίο configuration που εξετάσθηκε στα πειράματα ήταν ο αριθμός των memory ports. Για να δούμε πως η αλλαγή στον αριθμό των memory ports επηρεάζει το Power-Performance efficiency κρατήσαμε σταθερά τα υπόλοιπα configurations στο baseline και αλλάξαμε τον αριθμό των memory ports από 1 μέχρι 32 και για τα 6 benchmarks που έχουν επιλεχθεί.



Σχήμα 6.37: *Γραφικές παραστάσεις που αναπαριστούν το power-performance efficiency για όλες τις εφαρμογές για αριθμό memory ports μέχρι και 32 .*

Από τις πιο πάνω γραφικές παραστάσεις, σχήμα 6.37, παρατηρείται ότι στην εκτέλεση των διάφορων εφαρμογών δεν έχουμε μεγάλες αλλαγές στο Power-Performance efficiency όταν αλλάζουμε τον αριθμό των memory ports. Όσο αφορά την κατανάλωση ενέργειας η αλλαγή της ολικής κατανάλωσης ενέργειας είναι αμελητέα όσο αυξάνονται τα memory ports. Το ίδιο συμβαίνει και με τους χρόνους εκτέλεσης, με αποτέλεσμα να μην έχουμε μεγάλες αλλαγές στο efficiency.

Οι εφαρμογές με την πιο ενδιαφέρον συμπεριφορά είναι η FFT και η Ocean που όπως φαίνεται και στο σχήμα 6.37 έχουν την ίδια συμπεριφορά. Στην αρχή από 1 μέχρι 2 memory ports παρατηρείται αύξηση του efficiency γιατί μειώνονται οι χρόνοι εκτέλεσης των εφαρμογών, κάτι που συμβαίνει και στο σημείο μεταξύ 16 και 32 memory ports. Στο σημείο μάλιστα των 32 memory ports έχουμε την πιο μεγάλη τιμή στο efficiency. Σε αντίθεση στο σημείο από 8 μέχρι 16 memory ports έχουμε μείωση του efficiency για τον αντίστροφο λόγο. Από το σημείο των 2 μέχρι και 8 memory ports το efficiency είναι σταθερό.

Γενικά ο αριθμός των memory ports δεν επηρεάζει σημαντικά το Power-Performance efficiency αφού η μεγαλύτερη διαφορά είναι στο σημείο 1 και 32 memory ports για την εφαρμογή ocean με 1.15 φορές καλύτερο efficiency για 32 memory ports από ότι για 1 memory port .

Κεφάλαιο 7

Συμπεράσματα και Μελλοντικές Επεκτάσεις

7.1 Συμπεράσματα

Στην εργασία αυτή παρουσιάστηκε μία ανάλυση του Power-Performance efficiency για διαφορετικά configurations σε πολυπύρηνο επεξεργαστή, με διαφορετικές εφαρμογές. Τα αποτελέσματα από τα πειράματα οδήγησαν σε κάποια συμπεράσματα για το ποια είναι τα configurations ενός πολυπύρηνου επεξεργαστή που επηρεάζουν θετικά ή αρνητικά το Power-Performance efficiency. Επίσης διαφορετικές εφαρμογές δείχνουν διαφορετικά αποτελέσματα ως προς το Power-Performance efficiency.

Σε θέμα αριθμού των πυρήνων του πολυπύρηνο επεξεργαστή είδαμε ότι γενικά η αύξηση των πυρήνων συμβάλει θετικά στο Power-Performance efficiency. Υπάρχουν όμως και εξαιρέσεις, όταν δηλαδή μία εφαρμογή δεν μπορεί λόγω της δομής της να παραλληλοποιηθεί σε μεγάλο βαθμό, αυξάνοντας τον αριθμό των πυρήνων μειώνεται η επίδοση και οδηγεί σε μείωση του Power-Performance efficiency.

Το issue του επεξεργαστή όπως έχουμε δει είναι καλό σε πολυπύρηνους επεξεργαστές να είναι μικρό, όταν στοχεύουμε στο καλύτερο Power-Performance efficiency. Στις περισσότερες εφαρμογές ιδανικότερο θα ήταν να έχει την τιμή 2, ενώ σε άλλες με issue 1 είχαμε ακόμα πιο ψηλό Power-Performance efficiency.

Για το frequency του επεξεργαστή όλες οι εφαρμογές δείχνουν ότι με ψηλό frequency έχουμε καλύτερο Power-Performance efficiency. Εφαρμογές οι οποίες χρειάζονται μεγαλύτερη υπολογιστική ισχύ έχουν ακόμη μεγαλύτερα ποσοστά αύξησης της τιμής του Power-Performance efficiency.

Η σύγκριση της in-order με out-of-order εκτέλεσης έδειξε ότι η δεύτερη είναι κατά πολύ καλύτερη και σε θέμα Power-Performance efficiency εκτός από την επίδοση. Εφαρμογές οι οποίες έχουν εξαρτήσεις στις εντολές τους έδειξαν ακόμη περισσότερη βελτίωση σε out-of-order εκτέλεση.

Σε configurations που έχουν σχέση με την cache του επεξεργαστή, οι εφαρμογές δείχνουν διαφορετικά αποτελέσματα. Το Cache line size, από τα αποτελέσματα των περισσότερων εφαρμογών φαίνεται ότι το Power-Performance efficiency βελτιώνεται όσο το αυξάνουμε. Υπάρχουν όμως και εφαρμογές που δείχνουν το αντίθετο. Γενικά όταν μία εφαρμογή έχει ψηλό miss rate, αυξάνοντας το Cache line size βελτιώνεται η επίδοση επηρεάζοντας θετικά το Power-Performance efficiency. Είδαμε επίσης αποτελέσματα για το μέγεθος της cache του δευτέρου επιπέδου. Οι περισσότερες εφαρμογές έδειξαν ότι το βέλτιστο μέγεθος για L2 cache σε θέμα Power-Performance efficiency κυμαίνεται από 1MB μέχρι 4MB. Γενικά για μέγεθος της L2 cache υπάρχει ένα συγκεκριμένο μέγεθος για μία εφαρμογή, που μετά από αυτό δεν έχουμε βελτίωση της επίδοσης και παρατηρείται μείωση του Power-Performance efficiency λόγω της αυξημένης κατανάλωσης ενέργειας. Για το associativity της L2 cache από τα αποτελέσματα μπορούμε να συμπεράνουμε ότι για καλύτερο Power-Performance efficiency οι ιδανικότερες τιμές θα ήταν από 4 μέχρι 16.

Τέλος για τον αριθμό των memory ports είδαμε ότι δεν επηρεάζει σε σημαντικό βαθμό το Power-Performance efficiency πολυπύρηνου επεξεργαστή.

7.2 Μελλοντική Εργασία

Όπως έχει είδη αναφερθεί μία εφαρμογή έχει διαφορετικά ιδανικά configurations που βελτιώνουν στο μέγιστο το Power-Performance efficiency ενός πολυπύρηνου επεξεργαστή. Εξαρτάτε λοιπόν από την ίδια την εφαρμογή η οποία θα εκτελεστεί για το ποια είναι τα κατάλληλα configurations που χρειάζεται να έχει ο επεξεργαστής. Ενδιαφέρον λοιπόν θα ήταν μία μελλοντική εργασία οι οποία θα είχε ως θέμα την εύρεση του πως οι πολυπύρηνοι επεξεργαστές θα μπορούσαν να προσαρμοστούν στις απαιτήσεις της εφαρμογής προς εκτέλεση, για να πετύχουμε το καλύτερο δυνατό Power-Performance efficiency.

Αυτό θα μπορούσε να γίνει με την χρήση δυναμικών ετερογενών πολυπύρηνων επεξεργαστών, οι οποίοι αποτελούν το μέλλον των πολυπύρηνων επεξεργαστών.

Βιβλιογραφία

[1] Watt Matters Most? Desing Space Exploration of High Performance Microprocessors for Power-Performance-Efficiency. P. Trancoso, June 2007

[2] W Fornaciari, Donatella Sciuto, Cristina Silvano,Vittorio Zaccaria, A Design Framework to Efficiently Explore Energy-Delay Tradeoffs, 2001

[3] Milind B. Kamble and Kanad Ghose, Analytical Energy Dissipation Models For Low Power Caches, ISLPED 1997

[4] Uming Ko1, Poras T. Balsara2, and Ashwini K. Nanda1, Energy Optimization of Multi-Level Processor Cache Architectures, IEEE 1996

[5] Ricardo Gonzalez and Mark Horowitz,Energy Dissipation In General Purpose Microprocessors, September 1996

[6] Benjamin Lee and David Brooks, Effects of Pipeline Complexity on SMT/CMP Power-Performance Efficiency

[7] Ryan E. Grant Ahmad Afsahi, Power-Performance Efficiency of Asymmetric Multiprocessors for Multi-threaded Scientific Applications

[8] David M. Brook, Pradip Bose, Stanley E. Schuster, Hans Jacobson, Prabhakar N. Kudva, Alper Buyuktosunoglu , John-David Wellman, Victor Zyuban , POWER-AWARE MICROARCHITECTURE: Design and Modeling Challenges for Next-Generation Microprocessors

[9] Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. David Brooks

[10] SESC: SuperESCalar Simulator, Pablo Montesinos Ortego Paul Sack December 20, 2004

[11] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. (1995, June). The SPLASH-2 Programs: Characterization and Methodological Considerations.

[12] Bailey, D. H. FFTs in External or Hierarchical Memory. Journal of Supercomputing, 4(1):23-35, March 1990.

[13] Blelloch, G. E., et. al. A Comparison of Sorting Algorithms for the Connection Machine CM-2. In Symposium on Parallel Algorithms and Architectures, pp. 3-16, July 1991.

[14] Singh, J. P. Parallel Hierarchical N-body Methods and Their Implications for Multiprocessors. PhD Thesis, Stanford University, February 1993.

[15] Brandt, A. Multi-Level Adaptive Solutions to Boundary-Value Problems. Mathematics of Computation, 31(138):333-390, April 1977.

[16] Holt, C. and Singh, J. P.Hierarchical N-Body Methods on Shared Address Space Multiprocessors. SIAM Conference on Parallel Processing for Scientific Computing, Feb 1995, to appear.

[17] Woo, S. C., Singh, J. P., and Hennessy, J. L. The Performance Advantages of Integrating Block Data Transfer in Cache-Coherent Multiprocessors. Proceedings of the 6th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VI), October 1994.

[18] Mircea R. Stan Kevin Skadron University of Virginia, Power-Aware Computing

[19] T. M. Conte, K.N. Menezes, S. W. Sathaye, and M.C. Toburen, System-Level Power Consumption Modeling and Tradeoff Analysis Techniques for Superscalar Processor Design, IEEE Trans. on VLSI Systems, vol 8, no. 2, April 2000.

[20] Balaji V. Iyer Jesse G. Beu Thomas M. Conte, Length Adaptive Processors: A Solution for the Energy/Performance Dilemma in Embedded Systems, Georgia Institute of Technology, Atlanta, GA.

[21] Kyriakos Stavrou, Pedro Trancoso, Thermal-Aware Scheduling for Future Chip Multiprocessors, Department of Computer Science, University of Cyprus, December 2006.

[22] A. Miyoshi, C. Lefurgy, E. V. Hensbergen, R. Rajamony, R, Rajkumar, Critical Power Slope: Understanding the Runtime Effects of Frequency Scaling In Proceedings of the 16th Annual ACM International Conference on Supercomputing, 2002

[23] SimpleScalar: http://www.simplescalar.com/