ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ &
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ ΚΑΙ ΥΛΙΚΟΥ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Επιτάχυνση της συνάρτησης φυλογενετικής πιθανοφάνειας με χρήση **High Level Synthesis(HLS)** σε απομακρυσμένες πλατφόρμες αναδιατασσόμενης λογικής**(FPGA)**

ΣΟΥΦΛΑΚΗΣ ΕΜΜΑΝΟΥΗΛ

Εξεταστική Επιτροπή:
Καθηγητής κ. Δόλλας Απόστολος
Καθηγητής κ. Ζερβάκης Μιχαήλ
Επίκουρος Καθηγητής κ. Αλαχιώτης Νικόλαος (University of Twente)

Σεπτέμβριος 2020

Περίληψη

Μια από τις μεγαλύτερες προκλήσεις του 21ου αιώνα αποτελεί η ραγδαία και συνεχόμενη αύξηση των δεδομένων. Πολλοί είναι οι τομείς της επιστήμης που αντιμετωπίζουν προβλήματα στη διαχείριση και επεξεργασία των υπέρογκων δεδομένων, με τη βιολογία να αποτελεί έναν από αυτούς. Αναλυτικότερα, η επεξεργασία φυλογενετικών αναλύσεων DNA είναι μία χρονοβόρα διαδικασία και ο χρόνος που απαιτείται για τη διεκπεραίωσή της είναι ανάλογος του αριθμού των αλληλουχιών DNA-είδη προς μελέτη. Όσο αυξάνονται οι αλληλουχίες DNA, τόσο αυξάνονται και οι υπολογισμοί που απαιτούνται, καθώς και ο χρόνος εκτέλεσης αυτών. Έτσι, οι δυνατότητες της βιολογίας περιορίζονται λόγω έλλειψης υπολογιστικής δύναμης. Ακόμη και με τη χρήση γρήγορων υπολογιστών σύγχρονης τεχνολογίας το πρόβλημα των υπέρογκων υπολογισμών παραμένει, με αποτέλεσμα οι υπάρχουσες πλατφόρμες (π.χ. Personal Computer PC) να κρίνονται ανεπαρκείς ως προς την εξοικονόμηση χρόνου. Στην παρούσα διπλωματική εργασία μελετάται η μέθοδος της μέγιστης πιθανοφάνειας που αποτελεί έναν απ' τους αλγόριθμους για τη διεκπεραίωση φυλογενετικών αναλύσεων. Συγκεκριμένα, εξετάζεται η συμπεριφορά της μέγιστης πιθανοφάνειας , αρχικά, σε μια πλατφόρμα αναδιατασσόμενης λογικής (FPGA) και στη συνέχεια, σε περισσότερες απομακρυσμένες πλατφόρμες FPGA. Χρησιμοποιώντας τα οφέλη των FPGAs παρουσιάζεται η συμπεριφορά της μεθόδου της μέγιστης πιθανοφάνειας πάνω σε αρχιτεκτονική σχεδίαση ειδικού σκοπού. Έπειτα, πραγματοποιούνται προσεγγίσεις πλατφόρμων βασισμένες σε ένα πρότυπο κέντρου δεδομένων με απομακρυσμένους πόρους και FPGAs. Τέλος, η διπλωματική αυτή μελετά το πώς συμπεριφέρεται η επεκτασιμότητα (scalability) της αύξησης των αποκεντρωμένων FPGAs πάνω σε προσαρμοσμένη αρχιτεκτονική σχεδίαση με σκοπό την επιτάχυνση της μεθόδου της μέγιστης πιθανοφάνειας.

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα Καθηγητή κ. Δόλλα Απόστολο για την πολύτιμη βοήθειά του στην πορεία μου στο Πολυτεχνείο της Κρήτης, για την ειλικρίνεια και την άψογη συμπεριφορά που είχε απέναντί μου, καθώς και τη συνεργασία μας στα πλαίσια της παρούσας διπλωματικής εργασίας.

Επιπρόσθετα, θα ήθελα να ευχαριστήσω τον Επίκουρο Καθηγητή του Πανεπιστημίου του Twente κ. Αλαχιώτη Νικόλαο, για την επίσης πολύτιμη βοήθειά του σε αρκετά θέματα υλοποίησης της παρούσας διπλωματικής εργασίας, με ορθή καθοδήγηση και σκέψη. Εδραιώσαμε μία άψογη συνεργασία σε όλους τους τομείς.

Σειρά έχουν όλοι όσοι εργάζονταν ή εργάζονται στο εργαστήριο MHL του Πολυτεχνείου της Κρήτης. Η βοήθειά τους ήταν σημαντική, είτε άμεσα, είτε έμμεσα. Κάποιοι από αυτούς είναι ο Καθηγητής του Εθνικού Μετσόβιου Πολυτεχνείου κ. Πνευματικάτος Διονύσιος, ο Δρ. Σωτηριάδης Ευριπίδης, και φυσικά, ο κ. Κιμιωνής Μάρκος που πάντα με εφοδίαζε με ό,τι εξοπλισμό χρειαζόμουν και ήταν παρών σε κάθε μου πρόβλημα. Θα ήθελα να ευχαριστήσω, ιδιαίτερα, τον Δρ. κ. Θεοδωρόπουλο Δημήτρη για όλες τις χρήσιμες συμβουλές που μου έδινε καθ' όλη τη διάρκεια της διπλωματικής μου εργασίας.

Επίσης, θα ήθελα να ευχαριστήσω τον κ. Πισσαδάκη Μανώλη και τον κ. Μαλακωνάκη Παύλο που με βοήθησαν στην κατανόηση βασικών στοιχείων αυτής της διπλωματικής εργασίας τόσο σε γενικό, όσο και σε ειδικό επίπεδο.

Ακόμη, θα ήθελα να ευχαριστήσω όλα τα παιδιά από την αίθουσα των διπλωματικών του MHL για όλες τι συζητήσεις και τις απορίες που λύσαμε μαζί, στα πλαίσια της αρχιτεκτονικής υπολογιστών και μη.

Ένα επιπλέον ευχαριστώ σε όλους τους φίλους μου για την ευχάριστη παρέα και στήριξή τους όλα αυτά τα χρόνια.

Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου που με στήριξαν και συνεχίζουν να με στηρίζουν σε κάθε μου απόφαση και βήμα, καθώς και τα αδέρφια μου που μου στάθηκαν τόσο σε δύσκολες, όσο και σε ευχάριστες καταστάσεις. Ένα ξεχωριστό και μεγάλο ευχαριστώ, ωστόσο, στην αδερφή μου Ελίνα για την αδιανόητη υπομονή και στήριξη που μου προσέφερε σε όλη την πορεία μου μέχρι σήμερα.

Περιεχόμενα

Πίνακας εικόνων

Διαγράμματα

Πίνακες

Κεφάλαιο 1: Εισαγωγή

Το πρώτο κεφάλαιο παραθέτει πληροφορίες για την επιλογή του θέματος προς μελέτη, καθώς επίσης και τους στόχους της προκείμενης διπλωματικής εργασίας. Τέλος, περιέχει συνοπτικά την δομή της παρούσας εργασίας.

1.1 Εισαγωγή

Η βιολογία αποτελεί έναν από τους βασικότερους πυλώνες της επιστήμης και της ραγδαίας εξέλιξής της. Ο άνθρωπος έχει καταφέρει μέσω της βιολογίας να ανακαλύψει είδη, οργανισμούς, ασθένειες και άλλα, που συνέβαλαν στη διεύρυνση των γνώσεών του σχετικά με τον ίδιο και τη φύση του. Ένα από τα μεγαλύτερα κατορθώματα του ανθρώπου είναι η ανακάλυψη της καταγωγής του ανθρώπινου είδους από τα πρώτα χρόνια. Το ανθρώπινο DNA, καθώς και το DNA άλλων παρόμοιων και μη οργανισμών, συνδέονται μεταξύ τους δημιουργώντας ένα μεγάλο δέντρο της φύσης, ένα δέντρο της ζωής.

Ένα σημαντικό ζήτημα, ωστόσο, παρουσιάζεται στη διαχείριση όλων των ειδών προς μελέτη. Μπορεί το δέντρο της ζωής να διανθίζεται με την εισαγωγή καινούργιων ειδών που ανακαλύπτονται κάθε μέρα, η ερμηνεία του, ωστόσο, γίνεται πιο δύσκολη . Πρόβλημα αποτελούν τα μεγάλου όγκου δεδομένα προς επεξεργασία. Για παράδειγμα, στο facebook κάθε ώρα μεταφορτώνονται περισσότερες από 10 εκατομμύρια εικόνες [1]. Σε ένα γενικότερο πλαίσιο, το 2011 σε διάρκεια μόλις 2 ημερών δημιουργήθηκαν δεδομένα που άγγιζαν τα 1,8ZBytes [2]. Σύμφωνα, λοιπόν, με την παρούσα διπλωματική εργασία, ο βαθμός δυσκολίας του δέντρου της ζωής μεγαλώνει, όταν τα είδη προς μελέτη αυξάνονται, όταν, δηλαδή, τα δεδομένα προς επεξεργασία αυξάνονται.

Οι αλγόριθμοι που κατασκευάζουν το δέντρο της ζωής ποικίλουν σε είδη, τρόπους και μεταβλητές. Όλοι τους, ωστόσο, διαθέτουν ως κοινό παρονομαστή τις μεγάλου όγκου πράξεις που είναι απαραίτητο να υπολογιστούν για την ορθή κατασκευή του δέντρου. Οι 2 μεγαλύτερες εταιρίες σχεδίασης και κατασκευής Κεντρικών Επεξεργαστικών Κυκλωμάτων (CPU) Intel (https://www.intel.com/content/www/us/en/homepage.html) και AMD (https://www.amd.com/en), που συνεχίζουν να αναπτύσσονται διαρκώς, προσφέρουν μεγάλη υπολογιστική δύναμη αντάξια της τεχνολογικής πορείας και εξέλιξης των επιστημών. Ωστόσο, δεν είναι αρκετό. Ο λόγος έγκειται στην αρχιτεκτονική αυτών των υπολογιστικών κυκλωμάτων. Τα συγκεκριμένα υπολογιστικά κυκλώματα έχουν σχεδιαστεί και κατασκευαστεί από τις μεγάλες εταιρίες για να αποδίδουν σε ένα τεράστιο φάσμα εντολών. Οι αλγόριθμοι, όμως, που κατασκευάζουν το δέντρο της ζωής, απαιτούν τη συνεχή εκτέλεση συγκεκριμένων πράξεων όσο πιο γρήγορα και αποτελεσματικά γίνεται. Σε περίπτωση που οι πράξεις αυτές μετατραπούν σε εντολές ενός CPU, ο CPU θα εκτελέσει τις πράξεις με αρκετά περιττά στάδια ανάμεσά τους. Αναλυτικότερα, δεν είναι δυνατό να εκτελεστούν συγκεκριμένα κυκλώματα του CPU προς ικανοποίηση συγκεκριμένων απαιτήσεων ενός χρήστη, διότι οι CPU's έχουν κατασκευαστεί με τρόπο τέτοιο ώστε, να συμπεριλαμβάνουν όλες τις πιθανές απαιτήσεις στην εκτέλεσή τους, είτε αυτό είναι μία απλή πράξη, είτε ένα σύνολο πολύπλοκων εντολών. Για το λόγο αυτό, σε κάθε εκτέλεση πράξεων της CPU παρουσιάζεται καθυστέρηση.

Προς λύση του παραπάνω προβλήματος, στην παρούσα διπλωματική κατασκευάστηκε ένας επιταχυντής με την βοήθεια του High Level Synthesis (HLS) χρησιμοποιώντας το Vivado HLS πρόγραμμα της Xilinx [3] που αποσκοπεί στην ικανοποίηση των εκάστοτε απαιτήσεων ενός χρήστη. Ο επιταχυντής έχει σχεδιαστεί με τρόπο τέτοιο ώστε, οι πράξεις της μεθόδου να εκτελούνται γρήγορα και αποτελεσματικά χωρίς επιμέρους καθυστερήσεις και ελέγχους. Ένα τέτοιο κύκλωμα έχει τη δυνατότητα να ξεπεράσει σε απόδοση τους CPUs των μεγάλων εταιριών, θέτοντας νέα όρια στο χρόνο που απαιτείται για τον υπολογισμό των πράξεων της μεθόδου του δέντρου της ζωής.

Η κατασκευή του παραπάνω επιταχυντή οδηγεί σε μία γρήγορη και αποτελεσματική υλοποίηση της μεθόδου σχετικά με όλες τις πράξεις που πρέπει να υπολογιστούν. Ωστόσο, παρά τη σημαντική αύξηση απόδοσης με τη χρήση του νέου επιταχυντή, ο χρόνος υπολογισμού του δέντρου της ζωής για όλα τα γνωστά είδη παραμένει υψηλός.

Για την περαιτέρω βελτιστοποίηση του χρόνου υπολογισμού πραγματοποιείται προσέγγιση με βάση τον παραλληλισμό. Πιο συγκεκριμένα, οι πράξεις προς υπολογισμό μοιράζονται σε απομακρυσμένες πλατφόρμες αναδιατασσόμενης λογικής (FPGA) και εκτελούνται παράλληλα. Με αυτόν τον τρόπο διασπώνται τα δεδομένα προς επεξεργασία σε αποκεντρωμένα κυκλώματα ίδιας αρχιτεκτονικής σχεδίασης.

Με βάση τα παραπάνω, παρ' ότι τα οφέλη του παραλληλισμού γίνονται διακριτά, ακόμα και στην περίπτωση της χρήσης του παραλληλισμού υπάρχει κάποια δυσκολία. Αναλυτικότερα, όταν αυξάνεται ο αριθμός των FPGAs που χρησιμοποιούνται για παράλληλη επεξεργασία δεδομένων, η επιτάχυνση πραγματοποιεί, αρχικά, μία ανοδική πορεία, η οποία, όταν φτάσει σε κάποιο σημείο, συνεχίζει να ανεβαίνει απειροελάχιστα και στη συνέχεια καταλήγει σε καθοδική πορεία. Η απόδοση σε συνάρτηση με την επιτάχυνση ακολουθεί την ίδια πορεία. Είναι διαφορετικό να σταλούν δεδομένα σε 2 FPGAs από το να σταλούν σε 50 FPGAs. Όσο ανεβαίνει ο αριθμός των FPGAs, τόσο πιο πολύπλοκο, δύσκολο και χρονοβόρο γίνεται για τις FPGAs να συγχρονιστούν και να σταλούν δεδομένα σε όλες για εκτέλεση.

Παρ' όλο τον παραπάνω περιορισμό στον παραλληλισμό, η απόδοση ανεβαίνει αρκετά. Για το λόγο αυτό, στην παρούσα διπλωματική προσεγγίζεται παράλληλα η μέθοδος της μέγιστης πιθανοφάνειας με προσαρμοσμένη αρχιτεκτονική ειδικού σκοπού, χρησιμοποιώντας απομακρυσμένες πλατφόρμες αναδιατασσόμενης λογικής αποκεντρωμένων πόρων.

1.2Δομή της διπλωματικής εργασίας

Το δεύτερο κεφάλαιο εστιάζει στην κατανόηση του βιολογικού μέρους. Πιο συγκεκριμένα, μας εξηγεί τα διάφορα είδη φυλογενετικών δέντρων και τον τρόπο κατασκευής τους. Επίσης, περιλαμβάνει μία αναφορά πάνω σε διάφορους αλγόριθμους που χρησιμοποιούνται σήμερα για την επεξεργασία των αλληλουχιών DNA και την κατασκευή των φυλογενετικών δέντρων. Τέλος, περιγράφει συνοπτικά τον τρόπο λειτουργίας του προγράμματος RAxML που χρησιμοποιεί τον αλγόριθμο της μέγιστης πιθανοφάνειας για την κατασκευή φυλογενετικών δέντρων [4].

Το τρίτο κεφάλαιο αναλύει τον τρόπο λειτουργίας της μεθόδου της μέγιστης πιθανοφάνειας, τόσο μαθηματικά όσο και σχηματικά. Επιπρόσθετα, παρουσιάζει τις ιδιαιτερότητες του RAxML προγράμματος σχετικά με τον αλγόριθμο της μέγιστης πιθανοφάνειας για συγκεκριμένο πιθανο-θεωρητικό μοντέλο εξέλιξης αλληλουχιών DNA που χρησιμοποιείται στην παρούσα εργασία.

Το τέταρτο κεφάλαιο εισαγάγει την έννοια των αποκεντρωμένων πόρων και επεξηγεί την εφαρμογή της στον τομέα της μηχανικής.

Το πέμπτο κεφάλαιο ασχολείται με τη λεπτομερή ανάλυση της αρχιτεκτονικής της μεθόδου της μέγιστης πιθανοφάνειας που κατασκευάστηκε και χρησιμοποιείται με τη βοήθεια του High-Level-Synthesis(HLS).

Το έκτο κεφάλαιο περιέχει τη μελέτη των αποκεντρωμένων πόρων πάνω στην αρχιτεκτονική σχεδίαση της μεθόδου της μέγιστης πιθανοφάνειας που κατασκευάστηκε.

Το έβδομο κεφάλαιο περιλαμβάνει πολλαπλές υλοποιήσεις της παρούσας αρχιτεκτονικής σχεδίασης σε πλατφόρμες αναδιατασσόμενης λογικής. Πιο συγκεκριμένα, σε αυτό το κεφάλαιο αναλύονται οι περιορισμοί του υλικού. Παραδείγματα περιορισμών του υλικού αποτελούν το ρολόι, οι διαθέσιμοι υλικοί πόροι, το εύρος της μνήμης (Memory Bandwidth), τα πρωτόκολλα επικοινωνίας κ.α. Επιπλέον, γίνεται παρουσίαση και ανάλυση όλων των εργαλείων και των ιδιοτήτων τους που χρησιμοποιήθηκαν για την πραγματοποίηση των υλοποιήσεων.

Το όγδοο κεφάλαιο παραθέτει τα αποτελέσματα της υλοποίησης του έβδομου κεφαλαίου. Ο χρόνος ολοκλήρωσης, το Throughput και η απόδοση (%), είναι τα κύρια σημεία εστίασης. Επιπρόσθετα, το κεφάλαιο αυτό περιλαμβάνει προσεγγίσεις σε απομακρυσμένες πλατφόρμες αναδιατασσόμενης λογικής με αποκεντρωμένους πόρους.

Το ένατο και τελευταίο κεφάλαιο εστιάζει στα συμπεράσματα που διεξάγονται από την παρούσα μελέτη, καθώς επίσης και στις μελλοντικές βλέψεις και επεκτάσεις για τυχόν υλοποιήσεις σε πιο γρήγορες πλατφόρμες αναδιατασσόμενης λογικής με καλύτερο διαθέσιμο υλικό.

Κεφάλαιο 2: Βιολογικό υπόβαθρο

Το δεύτερο κεφάλαιο εστιάζει στο βιολογικό κομμάτι της παρούσας διπλωματικής και την κατανόηση των βιολογικών όρων που χρησιμοποιούνται. Επιπρόσθετα, γίνεται μία αναφορά στον τρόπο λειτουργίας του προγράμματος RAxML και τη χρήση του στη συγκεκριμένη μελέτη.

2.1 Εξελικτική διαδικασία-Αλληλουχίες DNA-Φυλογενετικές σχέσεις

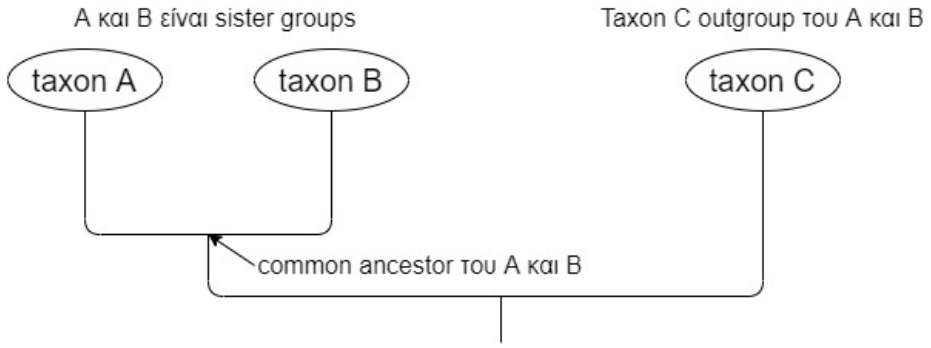
Από τη δεκαετία του '60 και ιδίως μετά την αστραπιαία εξέλιξη των υπολογιστικών συστημάτων (1971 μέχρι και σήμερα), η επιστήμη της βιολογίας άνθισε. Εφαρμόστηκαν αλλαγές σε επίπεδο εξοπλισμού (μικροσκόπια, H/Y, ειδικά διαμορφωμένα τεχνολογικά εργαστήρια) με την ανάπτυξη ορισμένων τομέων της τεχνολογίας, όπως είναι η οπτο-ηλεκτρονική και η πληροφορική. Αυτό το βήμα δημιούργησε νέες ευκαιρίες και επιλογές στον κλάδο της βιολογίας που εξελίσσεται πλέον με ραγδαίους ρυθμούς.

Μία από τις ποικίλες ευκαιρίες που γεννήθηκαν είναι η εξελικτική διαδικασία. Επί χρόνια οι βιολόγοι προσπαθούν να κατασκευάσουν ένα εξελικτικό δέντρο, -δέντρο της ζωής, όπως αποκαλείται-, το οποίο θα μπορεί να περιγράψει την εξέλιξη μεταξύ των διαφόρων ζωντανών και μη οργανισμών. Η ιδέα εδραιώθηκε όταν, τη δεκαετία του ’60 κέρδισε έδαφος η υπόθεση που υποστηρίζει ότι, συγκεκριμένες περιοχές του γενετικού υλικού μπορεί να περιέχουν σημαντικές πληροφορίες για την εξέλιξη του είδους. Στο παρελθόν, ωστόσο, μία τέτοια δενδρική κατασκευή των διαφόρων ειδών υλοποιούταν σύμφωνα με την παρουσία ή απουσία ιδιαίτερων χαρακτηριστικών. Ένα μείζον ζήτημα που παρουσιάστηκε, αφορά τη δενδρική τοποθέτηση, όταν 2 ή περισσότερα είδη δεν έχουν προφανή διαφορετικά ή ιδιαίτερα χαρακτηριστικά. Η λύση του ζητήματος της ορθά δενδρικής ταξινόμησης των ειδών-οργανισμών βρέθηκε στις νουκλεοτιδικές ομοιότητες και διαφορές που αξιοποιούνται στην βάση των τελευταίων [5].

2.2 Φυλογένεια και Φυλογενετικό Δέντρο

Κάθε ζωντανός οργανισμός που υπάρχει πάνω στη γη εδώ και δισεκατομμύρια χρόνια, όπως και αυτοί που έχουν εξαφανιστεί (λ.χ. δεινόσαυροι), διαφέρουν μεταξύ τους ανάλογα με το περιβάλλον που υπάρχουν και εξελίσσονται. Ωστόσο, παρά τις όποιες διαφορές τους, η εξελικτική τους ιστορία πιθανώς ομοιάζει σε κάποια σημεία, καθώς όλοι αυτοί οι οργανισμοί αποτελούν τμήμα του ίδιου δέντρου της ζωής, -φυλογενετικό δέντρο, όπως αποκαλείται στο χώρο της επιστήμης-.

Το φυλογενετικό δέντρο αναπαριστά τις εξελικτικές σχέσεις των οργανισμών και παρέχει πληροφορίες για τις αλλαγές που έχουν πραγματοποιηθεί στις γενεαλογικές σχέσεις των οργανισμών στο πέρας του χρόνου. Πιο αναλυτικά, οι ομάδες των οργανισμών που περιλαμβάνονται στο εξελικτικό δέντρο ονομάζονται taxa (ενικός: taxon) και προβάλλονται ως άκρες του δέντρου, όπως διακρίνεται στην Εικόνα 2. 1. Οι κόμβοι στο δέντρο απεικονίζουν τους κοινούς προγόνους (common ancestors) των taxa. Δύο ή περισσότερα taxa που περιλαμβάνονται στον ίδιο κόμβο ονομάζονται αδελφές ομάδες (sister groups). Επιπρόσθετα με τις αδελφές ομάδες, στα φυλογενετικά δέντρα περιλαμβάνονται και οι λεγόμενες εξω-ομάδες (outgroups), οι οποίες μπορεί να έχουν κάποιο κοινό πρόγονο με κάποια sister groups, δίχως να περιλαμβάνονται στο ίδιο sister group με τα υπόλοιπα.



Εικόνα 2. 1 Εξελικτικό δέντρο που περιέχει τις έννοιες αδερφικές ομάδες, κοινός πρόγονος και εξω-ομάδα.

2.3 Τύποι Φυλογενετικών Δέντρων

Υπάρχουν 2 διαφορετικοί τύποι φυλογενετικών δέντρων: τα δέντρα με ρίζα (rooted) και τα δέντρα χωρίς ρίζα. Αναλυτικότερα, τα δέντρα με ρίζα είναι ο πιο συνήθης τύπος φυλογενετικών δέντρων που απεικονίζει τον κοινό πρόγονο και την κατεύθυνση της εξέλιξης των ειδών (taxa). Τα δέντρα χωρίς ρίζα δεν παρέχουν πληροφορίες για τη ρίζα και την κατεύθυνση της εξελικτικής πορείας των ειδών. Όταν εξετάζονται περισσότερα από 2 είδη, ο αριθμός των πιθανών δέντρων που προκύπτει αυξάνεται παραγοντικά. Οι μαθηματικοί τύποι που υπολογίζουν τον αριθμό των πιθανών δέντρων είναι ο εξής:

1.Tree(Δέντρο) με ρίζα: $\text{Τρ} = \frac{(2n-3)!}{2^{n-2}(n-3)!}$

2.Tree χωρίς ρίζα: $\text{Τχρ} = \frac{(2n-5)!}{2^{n-3}(n-3)!}$

Στον Πίνακας 2. 1 απεικονίζεται η αύξηση του αριθμού των πιθανών δέντρων με ρίζα και χωρίς ρίζα, ως αποτέλεσμα της αύξησης του αριθμού των ειδών.

Είδη	**Πιθανά δέντρα με ρίζα**	**Πιθανά δέντρα χωρίς ρίζα**
3	3	1
4	15	3
5	105	15
6	945	105
7	10395	945
8	135135	10395
9	2027025	135135
10	34459425	2027025
20	8,20079E+21	2,21643E+20
30	4,9518E+38	8,68736E+36
40	1,00985E+57	1,31149E+55
50	2,75292E+76	2,83806E+74

Πίνακας 2. 1 Όλα τα πιθανά δέντρα με και χωρίς ρίζα που προκύπτουν με συντελεστή τα είδη

Διαπιστώνεται πως, καθώς ο αριθμός των ειδών αυξάνεται, πληθαίνουν αισθητά τα πιθανά δέντρα, καθιστώντας δύσκολη την αναγνώριση του αυθεντικού φυλογενετικού δέντρου που περιέχει όλα τα γνωστά μέχρι στιγμής είδη.

2.4 Μέθοδοι φυλογενετικών σχέσεων και προγράμματα υλοποίησης αυτών

Στις μέρες μας υπάρχουν πολλά προγράμματα που εκτελούν ποικίλες μεθόδους για τον υπολογισμό των φυλογενετικών σχέσεων πολλαπλών αλληλουχιών DNA. Πολλές από αυτές τις μεθόδους βρίσκονται σε διάφορες ιστοσελίδες, όπως το Department of Genome Sciences του τμήματος Ιατρικής του Πανεπιστημίου της Ουάσιγκτον. (http://evolution.genetics.washington.edu/phylip/software.html)

Στη διπλωματική αυτή γίνεται ανάλυση της μεθόδου της μέγιστης πιθανοφάνειας. Η συγκεκριμένη μέθοδος προτιμάται λόγω της ικανοποιητικής απόδοσής της στην εξελικτική διαδικασία. Πολλά προγράμματα, όπως το PHYML [6] και το GARLI [7], χρησιμοποιούν την μέθοδο

της μέγιστης πιθανοφάνειας. Η διπλωματική αυτή κάνει χρήση του προγράμματος Randomized Axelerated Maximum Likelihood (RAxML) [4], [8].

2.5 Τρόπος λειτουργείας του RAxML

Το RAxML είναι ένα πρόγραμμα που χρησιμοποιεί τη μέθοδο της μέγιστης πιθανοφάνειας για τον υπολογισμό του πιθανότερου φυλογενετικού δέντρου ειδών σε φυλογενετικές αναλύσεις μεγάλης κλίμακας. Το πρόγραμμα ξεκινά κάνοντας ευθυγράμμιση των αλληλουχιών DNA προς μελέτη. Στην συνέχεια, πραγματοποιείται μια σειρά από υποθέσεις για την ευθυγράμμιση των αλληλουχιών, καθώς και η επιλογή ενός δέντρου αναφοράς. Κατ' αυτόν τον τρόπο, αρχίζει η αναδρομική διαδικασία της μέγιστης πιθανοφάνειας για όλες τις ευθυγραμμίσεις των αλληλουχιών. Αφού τελειώσει η αναδρομική διαδικασία, -δηλαδή, αφού εκτελεστεί η μέθοδος της μέγιστης πιθανοφάνειας για κάθε μια θέση της ευθυγραμμισμένης αλληλουχίας-, το RAxML υποθέτει εκ νέου, θέτοντας διαφορετικό δέντρο αναφοράς. Η διαδικασία αυτή εκτελείται αρκετές φορές και κάθε φορά αποθηκεύεται το score του κάθε δέντρου, σύμφωνα με το εκάστοτε υποτιθέμενο δέντρο αναφοράς. Μετά την ολοκλήρωση αυτής της διαδικασίας δημιουργούνται πολλά αρχεία με τα score των δέντρων για όλα τα δέντρα αναφοράς που χρησιμοποιήθηκαν, καθώς και ένα αρχείο που περιέχει το καλύτερο δέντρο (best tree). Το αρχείο αυτό περιλαμβάνει το δέντρο με τη μεγαλύτερη πιθανοφάνεια όλων των υπόλοιπων δέντρων (Stamatsakis et al., 2005; Stamatakis, 2006a).

Ύψιστης σημασίας είναι να γίνει κατανοητό ότι, ο υπολογιστικός όγκος της μέγιστης πιθανοφάνειας αυξάνεται κατά τις διαστάσεις ειδών και μήκους αλληλουχιών. Η διπλωματική αυτή εστιάζει στην καλύτερη διαχείριση των μηκών αλληλουχιών των ειδών.

Κεφάλαιο 3: Μέθοδος Μέγιστης Πιθανοφάνειας

Το τρίτο κεφάλαιο περιλαμβάνει την ανάλυση της μεθόδου της μέγιστης πιθανοφάνειας καθώς και των σημαντικότερων μοντέλων εξέλιξης των αλληλουχιών DNA που την πλαισιώνουν. Επιπρόσθετα, αναλύει τη συνάρτηση της μέγιστης πιθανοφάνειας μαθηματικά και σχηματικά.

3.1Μέγιστη πιθανοφάνεια

Ο αλγόριθμος της μέγιστης πιθανοφάνειας είναι μια μέθοδος μέσω της οποίας γίνεται η επιλογή ενός δέντρου, το οποίο επακόλουθα αυξάνει την πιθανοφάνεια των δεδομένων που μελετώνται. Για την επιλογή του δέντρου και τη μεγιστοποίηση της πιθανότητας πραγματοποιείται μία σειρά υποθέσεων, η οποία και συνεχίζεται με απώτερο σκοπό την καλύτερη δυνατή εξασφάλιση της αξιοπιστίας των δεδομένων. Οι υποθέσεις αυτές ποικίλλουν σε αριθμό και είδος. Παραδείγματα υποθέσεων αποτελούν ορισμένες ειδικές παράμετροι συμπεριλαμβανομένων ακόμη και των ίδιων των δέντρων. Πιο συγκεκριμένα, όσον αφορά την εξελικτική διαδικασία που αναλύεται στην παρούσα εργασία, η μέθοδος της μέγιστης πιθανοφάνειας επιλέγει το δέντρο με την μεγαλύτερη πιθανοφάνεια που καθιστά πιο πιθανή την ιστορική πορεία και εξέλιξη των ειδών προς μελέτη [9].

3.2 Πιθανότητα-Πιθανοφάνεια: Η διαφορά τους και πως η πιθανοφάνεια συνδέεται με την εξελικτική διαδικασία

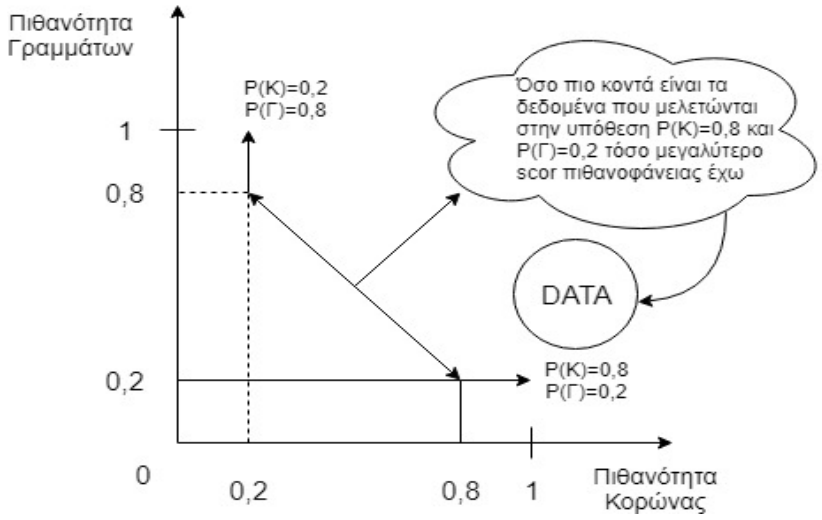
Για να γίνει πλήρως κατανοητή η έννοια της "πιθανοφάνειας", ύψιστης σημασίας είναι να διαχωριστεί ο όρος "πιθανοφάνεια" από τον όρο "πιθανότητα". Ο διαχωρισμός των 2 προαναφερθέντων όρων γίνεται ευδιάκριτος με το ακόλουθο παράδειγμα:

Έστω ένα νόμισμα. Ξέρουμε ότι το νόμισμα είναι δίκαιο εξαρχής. Αυτό πρακτικά σημαίνει ότι, η πιθανότητα η μια ρίψη του νομίσματος να είναι είτε κορώνα είτε γράμματα είναι ίδια και ισούται με 0.5(50%).

Εάν μας ζητείται η πιθανότητα να φέρουμε 2 κορώνες στη σειρά, η πιθανότητα είναι:
P (Να φέρουμε 2 κορώνες, Δεδομένου ότι, Το νόμισμα είναι δίκαιο) ή
P (ΚΚ|ρκ,ργ=0.5)=0.5x0.5=0.25 (25%).

Με βάση το προηγούμενο παράδειγμα δεν λαμβάνουμε ως δεδομένο ότι το νόμισμα είναι δίκαιο. Αντίθετα μας ζητείται να βρούμε την πιθανότητα του νομίσματος για κορώνα και γράμματα αντίστοιχα. Το δεδομένο μας στην προκειμένη είναι πως, στις 100 ρίψεις οι 80 είναι κορώνα. Δε μπορούμε να υπολογίσουμε την πιθανότητα που μας ζητείται, αν δε γνωρίζουμε όλες τις πιθανές υποθέσεις. Μπορούμε, ωστόσο, να υπολογίσουμε την πιθανοφάνεια της δικαιότητας του νομίσματος. Με άλλα λόγια, αν έχω σε 100 ρίψεις 80 κορώνες, είναι πιθανότερο το νόμισμά μου να μην είναι δίκαιο. Συνεπώς, η πιθανοφάνεια παρατηρεί τα δεδομένα και δίνει μια εκτίμηση που τα καθιστούν πιο πιθανά. Πιο συγκεκριμένα:
L (Πόση είναι η πιθανότητα της κορώνας και των γραμμάτων, δεδομένου ότι, σε 100 ρίψεις είχαμε 80 κορώνες)

Όπως παρατηρείται παραπάνω, τα δεδομένα μας μένουν σταθερά, ενώ η πιθανότητα της κορώνας και των γραμμάτων γίνεται μεταβλητή. Συμπεραίνουμε, λοιπόν, ότι, η πιθανότητα της κορώνας και των γραμμάτων που ταιριάζουν περισσότερο στα δεδομένα μας είναι ρκ=0.8 και ργ=0.2 αντίστοιχα. Με βάση αυτήν την υπόθεση, παίρνουμε την μεγαλύτερη πιθανή πιθανοφάνεια. Αν θέσουμε τις πιθανότητες αντίστροφα, (ρκ=0.2 και ργ=0.8) η πιθανοφάνεια θα είναι πολύ μικρή, γεγονός που συνάδει με τη βάση των δεδομένων που παρατηρούμε. Η παρακάτω εικόνα δείχνει σχηματικά το παραπάνω παράδειγμα.



Εικόνα 3. 1 Παράδειγμα παρατήρησης δεδομένων σε σχέση με τις διάφορες υποθέσεις

Συνοψίζοντας, όταν έχουμε μια σταθερή υπόθεση, ανάλογα με τα δεδομένα που δίνονται, βρίσκουμε την πιθανότητα των δεδομένων μας πάνω στην σταθερή αυτή υπόθεση. Σε αντίθεση, όταν μας δίνονται κάποια σταθερά δεδομένα, βρίσκουμε την πιθανοφάνεια όπου, μια ή περισσότερες υποθέσεις κάνουν τα δεδομένα μας πιο αληθινά.

Αντίστοιχα με το προαναφερθέν παράδειγμα, πραγματοποιείται και η μέθοδος της μέγιστης πιθανοφάνειας σε μια εξελικτική διαδικασία. Αναλυτικότερα, αντί να βρούμε τα δεδομένα (αλληλουχίες DNA) που δημιουργούν το καλύτερο δέντρο, βρίσκουμε το δέντρο που μεγιστοποιεί την πιθανότητα του ότι τα δεδομένα μας είναι πιο αξιόπιστα και πιθανά. Με μαθηματική διατύπωση: P(data|tree) έναντι P(tree|data).

Εφόσον λοιπόν αναλύθηκε η διαφοροποίηση των όρων "πιθανότητα" και "πιθανοφάνεια" και η επεξήγηση του τελευταίου, το επόμενο βήμα είναι η περαιτέρω ανάλυση της "πιθανοφάνειας" ως μία μέθοδος που χρησιμοποιεί αλληλουχίες DNA για την κατασκευή φυλογενετικών δέντρων. Για τον υπολογισμό της πιθανοφάνειας των δέντρων, η μέθοδος της μέγιστης πιθανοφάνειας χρειάζεται κάποιο πιθανοθεωρητικό μαθηματικό μοντέλο της εξέλιξης των αλληλουχιών DNA των ειδών, το οποίο αναλύεται στη συνέχεια.

3.3Πιθανοθεωρητικά μοντέλα εξέλιξης αλληλουχιών DNA

Μοντέλο εξέλιξης αλληλουχιών DNA καλείται ένας 4x4 πίνακας που παρουσιάζει την πιθανότητα του ρυθμού υποκατάστασης των τεσσάρων νουκλεοτιδίων των DNA αλληλουχιών: Αδενίνη (A), Γουανίνη (G), Θυμίνη (T) και Κυτοσίνη (C), σε ένα απειροελάχιστο χρονικό διάστημα dt. Αναλυτικότερα, κάθε ένα στοιχείο του πίνακα περιέχει το ρυθμό με τον οποίο ένα νουκλεοτίδιο αντικαθίσταται από κάποιο άλλο. Τα νουκλεοτίδια χωρίζονται σε 2 ομάδες: την πουρίνη (A ή G) και την πυριμιδίνη (C ή T) [10]. Προτού αναλυθεί το γενικότερο μοντέλο εξέλιξης, είναι ύψιστης σημασίας να τονισθεί ότι τα εξελικτικά μοντέλα που θα συναντήσουμε υπακούουν στην ανέλιξη Markov (Μαρκοβιανά μοντέλα, όπως ονομάζονται) [11]. Όταν ένα μοντέλο εξέλιξης υπακούει στην ανέλιξη Markov, το μοντέλο μπορεί να αποκτήσει έναν ή περισσότερους περιορισμούς. Κάποιοι από αυτούς τους περιορισμούς είναι:

- Χρονική ομογένεια: οι πιθανότητες μεταβάσεων είναι ανεξάρτητες του χρόνου.
- Στασιμότητα: κάθε χρονική στιγμή η κατανομή των βάσεων είναι αυτή της κατάστασης ισορροπίας.
- Αντιστρεπτότητα των πιθανοτήτων μετάβασης: οι πιθανότητες είναι ίδιες και για τις αντίστροφες μεταβάσεις.

Αποτελεί γενική παραδοχή το ότι, τα μοντέλα εξέλιξης υπακούουν στους 3 παραπάνω περιορισμούς για λόγους υπολογιστικής απλότητας. Ο γενικός πίνακας 4x4 (χωρίς περιορισμούς),

που αφορά τους ρυθμούς υποκατάστασης των νουκλεοτιδίων των αλληλουχιών DNA, είναι ο Qij, ο οποίος μαθηματικά εκφράζεται ως εξής:

Γενικός πίνακας Q

$$\begin{pmatrix} -\mu(a\pi_C + b\pi_G + c\pi_T) & \mu a\pi_C & \mu b\pi_G & \mu c\pi_T \\ \mu g\pi_A & -\mu(g\pi_C + d\pi_G + e\pi_T) & \mu d\pi_G & \mu e\pi_T \\ \mu h\pi_A & \mu j\pi_C & -\mu(h\pi_A + j\pi_G + f\pi_T) & \mu f\pi_T \\ \mu i\pi_A & \mu k\pi_C & \mu l\pi_G & -\mu(i\pi_A + k\pi_C + l\pi_G) \end{pmatrix}$$

Κάθε γραμμή του παραπάνω πίνακα (i), όπως και κάθε στήλη του (j), αντιστοιχεί στα 4 νουκλεοτίδια, Αδενίνη (A), Κυτοσίνη (C) , Γουανίνη (G) και Θυμίνη (T). Οι ειδικοί παράμετροι a, b, c, d, e, f, g, h, i, j, k και l, αποτελούν το μετασχηματισμό από μια βάση σε κάποια άλλη, ενώ οι παράμετροι πa, πc, πg και πt, είναι οι συχνότητες των βάσεων κάθε νουκλεοτιδίου αντίστοιχα. Τέλος, η παράμετρος μ είναι η μέση στιγμιαία τιμή των ρυθμών υποκατάστασης των νουκλεοτιδίων.

Εφαρμόζοντας τα Μακροβιανά μοντέλα, ο γενικός πίνακας Q τροποποιείται και το γενικό μοντέλο υλοποιείται με τον περιορισμό της χρονικής αντιστρεπτότητας. Πιο συγκεκριμένα, χρονικά αντιστρεπτό (Time-Reversible) λέγεται ένα μοντέλο όταν ο αριθμός των αλλαγών από μία βάση i σε μια βάση j είναι ίδιος με τον ρυθμό αλλαγών από μία βάση j σε μία βάση i, σε ένα περιορισμένο χρονικό διάστημα dt. Ως αποτέλεσμα, οι ειδικοί παράμετροι του πίνακα Q περιορίζονται, ώστε να τηρηθεί η υπόθεση της χρονικής αντιστρεπτότητας. Συνεπώς, οι ειδικοί παράμετροι των βάσεων διαμορφώνονται ως εξής: a=g, b=h, c=i, d=j, e=k και f=l.

Πίνακας Q με GTR

$$\begin{pmatrix} -\mu(a\pi_C + b\pi_G + c\pi_T) & \mu a\pi_C & \mu b\pi_G & \mu c\pi_T \\ \mu a\pi_A & -\mu(a\pi_C + d\pi_G + e\pi_T) & \mu d\pi_G & \mu e\pi_T \\ \mu b\pi_A & \mu d\pi_C & -\mu(b\pi_A + d\pi_G + f\pi_T) & \mu f\pi_T \\ \mu c\pi_A & \mu e\pi_C & \mu f\pi_G & -\mu(c\pi_A + e\pi_C + f\pi_G) \end{pmatrix}$$

Το γενικό μοντέλο Q μετονομάζεται σε Γενικό Χρονικά Αντιστρεπτό Μοντέλο (General Time-Reversible Model ή GTR Model) μετά την εφαρμογή του περιορισμού της χρονικής αντιστρεπτότητας. Το μεγαλύτερο ποσοστό βιολόγων χρησιμοποιεί το GTR μοντέλο για τις περισσότερες φυλογενετικές αναλύσεις, καθώς είναι το μόνο μοντέλο που έχει τον περιορισμό της χρονικής αντιστρεπτότητας και συνάμα περιέχει τις περισσότερες δυνατές παραμέτρους. Λόγω αυτής της λειτουργικής του ιδιαιτερότητας, οι ρυθμοί αντικατάστασης των νουκλεοτιδίων τείνουν να είναι πιο ακριβείς. Όσο μεγαλώνει ο αριθμός των παραμέτρων, τόσο αυξάνεται η υπολογιστική πολυπλοκότητα της φυλογενετικής ανάλυσης, αλλά και η ακρίβεια των αποτελεσμάτων [12].

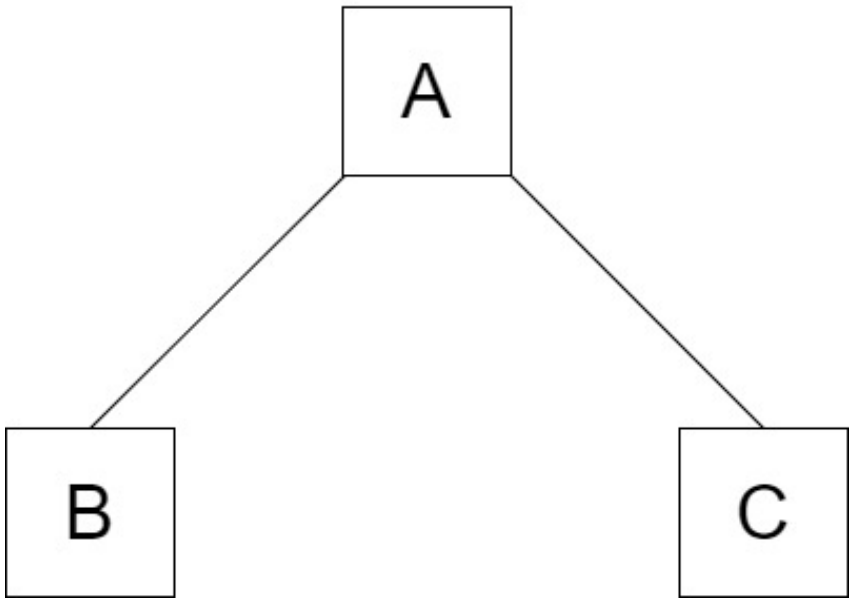
Η διπλωματική αυτή χρησιμοποιεί το GTR μοντέλο για τη διεκπεραίωση της μεθόδου της μέγιστης πιθανοφάνειας. Με βάση το μοντέλο εξέλιξης των αλληλουχιών GTR μπορούν να υπολογιστούν οι πιθανότητες υποκατάστασης των ακολουθιών, σύμφωνα με τον τύπο $P(s) = e^{Qs}$. Η συγκεκριμένη εξίσωση υπολογίζει την πιθανότητα αλλαγής μιας κατάστασης εκ των τεσσάρων νουκλεοτιδίων A, C, G, ή T σε οποιαδήποτε άλλη κατάσταση, κατά μήκος ενός κλαδιού μήκους s με ρυθμό υποκατάστασης Q. Τα αποτελέσματα των υπολογισμών δημιουργούν έναν πίνακα υποκατάστασης [13].

3.4Υπολογισμός πιθανοφάνειας του δέντρου

Όπως έχει προαναφερθεί, η μέθοδος της μέγιστης πιθανοφάνειας παραθέτει μια εκτίμηση, (ένα scor), για κάθε υπόθεση, με βάση τα δεδομένα που μελετώνται. Η μέθοδος που μελετάται λειτουργεί σύμφωνα με την αρχή του Felsenstein ή όπως αλλιώς λέγεται Felsenstein's Pruning Algorithm (FPA) [14], [15]. Ο FPA χρησιμοποιείται διότι προσφέρει έναν πρακτικό και γρήγορο τρόπο υπολογισμού της μεθόδου της μέγιστης πιθανοφάνειας, ο οποίος εξηγείται στην συνέχεια.

Για κάθε μία θέση N της αλληλουχίας DNA μελετάται ένα υποτιθέμενο δέντρο με ρίζα A, τριών διανυσμάτων A, B και C, όπως φαίνεται στην Εικόνα 3. 2. Ο ρόλος της μεθόδου της μέγιστης πιθανοφάνειας χρησιμοποιώντας την μέθοδο FPA, είναι να εκτιμήσει κατά πόσο το υποτιθέμενο

δέντρο ανταποκρίνεται στα δεδομένα που μελετώνται, πόσο, δηλαδή, αξιόπιστο είναι το υποτιθέμενο δέντρο που μελετάται.

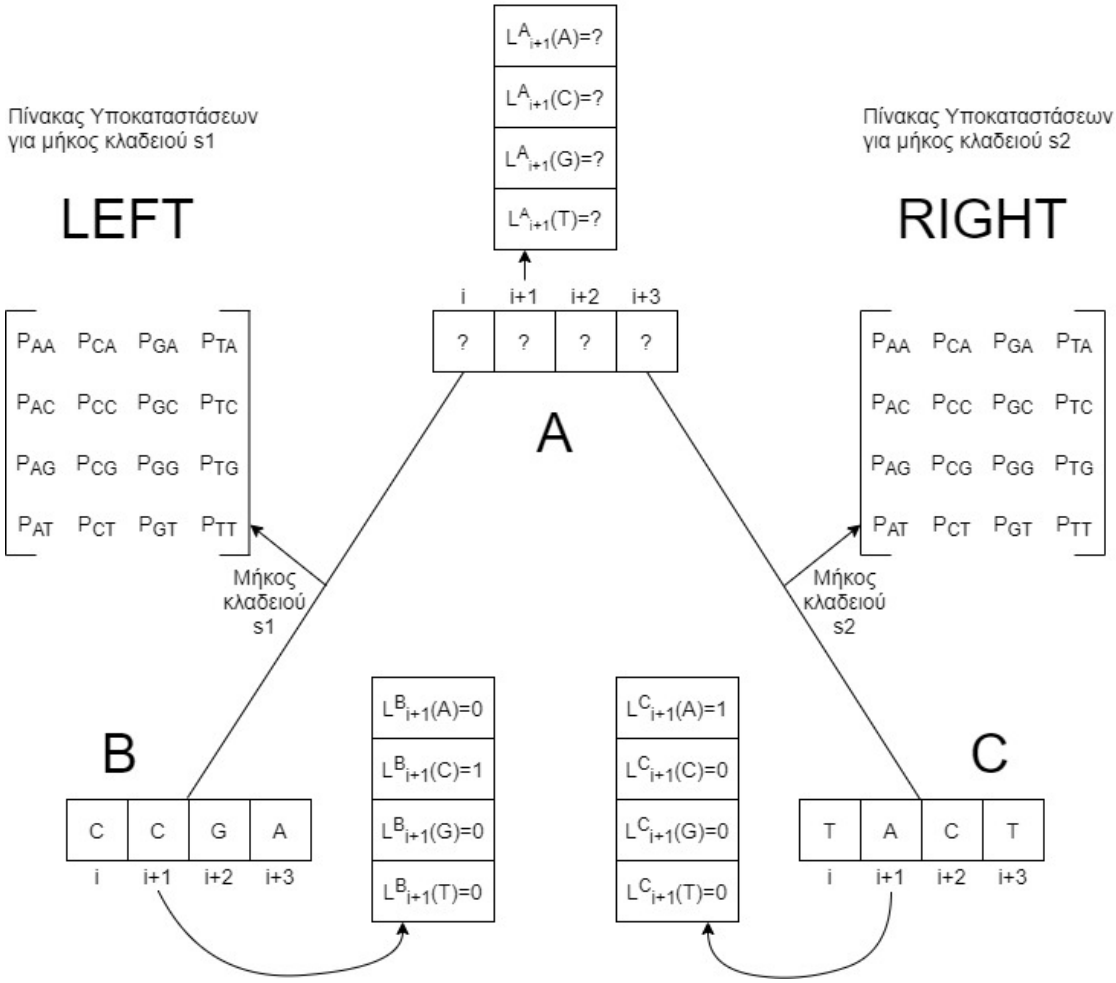


Εικόνα 3. 2 A-B-C Υποτιθέμενο δέντρο τριών ακολουθιών

Σύμφωνα με την Εικόνα 3. 2, πραγματοποιείται η υπόθεση πως τα B και C, που είναι γνωστά, έχουν κοινό πρόγονο τον A. Για να βρεθεί κατά το πόσο η υπόθεση που έγινε ανταποκρίνεται καλύτερα στα δεδομένα πρέπει να υπολογιστεί ο βαθμός πιθανοφάνειας του A, υποθετικού προγόνου του B και C. Ο υπολογισμός της υποτιθέμενης ρίζας πραγματοποιείται βρίσκοντας για κάθε μια θέση N των διανυσμάτων B και C την πιθανοφάνεια της θέσης του διανύσματος A αντίστοιχα. Ο παρακάτω τύπος εξηγεί τη διαδικασία της FPA μεθόδου:

$$L(X_{Aj} = i) = [\sum_{z} P_{iz}\,(S_{AB}) L(X_{Bj} = z)][\sum_{z} P_{iz}(S_{AC}) L(X_{Cj} = z) \quad (1)$$

Η εξίσωση της σχέσης (1) περιλαμβάνει αρκετές παραμέτρους και όρους. Με X συμβολίζεται μία ακολουθία νουκλεοτιδίων, δηλαδή ένα διάνυσμα. Η παράμετρος j αντιπροσωπεύει τη θέση της ακολουθίας X και οι παράμετροι i και z τις καταστάσεις των τεσσάρων νουκλεοτιδίων A, C, G και T. Ο όρος P συμβολίζει έναν πίνακα υποκατάστασης από μία κατάσταση i σε μία κατάσταση z, για συγκεκριμένο μήκος διαστήματος S. Ο όρος L συμβολίζει την πιθανοφάνεια μιας ακολουθίας X στη θέση j, για όλες τις z και i καταστάσεις. Όταν μία ακολουθία X είναι γνωστή, όπως οι B και C ακολουθίες, τότε η πιθανοφάνεια ισούται με 1 στην περίπτωση που η κατάσταση της θέσης j της ακολουθίας είναι ίδια με την κατάσταση z που μελετάται. Διαφορετικά, εάν οι καταστάσεις j και z δε συμπίπτουν, η πιθανοφάνεια ισούται με 0. Οι ακολουθίες A, B και C χωρίζονται σε N*16 διανύσματα έκαστη, για κάθε μια θέση N των αλληλουχιών DNA. Κάθε ένα διάνυσμα μιας θέσης N αποτελείται από 4 θέσεις. Κάθε μία από τις 4 θέσεις του διανύσματος περιέχει 4 πιθανοφάνειες των καταστάσεων A, C, G, και T. Οι πιθανοφάνειες των N διανυσμάτων σχηματίζουν συνολικά έναν 4x4 πίνακα για κάθε μια θέση N των αλληλουχιών. Η Εικόνα 3. 3 εξηγεί σχηματικά τα παραπάνω.



Εικόνα 3. 3 Υπολογισμός Α ακολουθίας κοινού υποτιθέμενου προγόνου των ακολουθιών Β και C

Όπως διακρίνεται στην Εικόνα 3. 3, επιλέχτηκε τυχαία η θέση i+1 των A, B και C πινάκων πιθανοφάνειας, ως παράδειγμα. Κάθε μια θέση περιέχει 4 πιθανοφάνειες των τεσσάρων καταστάσεων των νουκλεοτιδίων A, C, G και T. Στην περίπτωση του πίνακα B, η θέση i+1 περιέχει την κατάσταση C. Συνεπώς, οι πιθανοφάνειες των τεσσάρων καταστάσεων A, C, G και T ισούνται με 0, εκτός της πιθανοφάνειας της κατάστασης C, καθώς η C είναι η κατάσταση που μελετάται στη θέση i+1. Με αντίστοιχο τρόπο, οι πιθανοφάνειες της θέσης i+1 του πίνακα C ισούνται με 0, εκτός της πιθανοφάνειας A που μελετάται στην i+1 θέση. Ο πίνακας υποκατάστασης της B ακολουθίας ονομάζεται Left και περιέχει δεδομένα για την πιθανότητα υποκατάστασης όλων των πιθανοφανειών των θέσεων του πίνακα πιθανοφάνειας B, για μήκος κλαδιού s1. Αντίστοιχα, ο πίνακας υποκατάστασης για την C ακολουθία ονομάζεται Right με μήκος κλαδιού s2. Σύμφωνα με τον μαθηματικό τύπο (1), οι πράξεις για τον υπολογισμό της θέσης i+1 της ζητούμενης A ακολουθίας γίνονται με τον εξής τρόπο.

$$L^A_{i+1}(A) = [P_{AA}(s1)L^B_{i+1}(A)+ P_{AC}(s1)L^B_{i+1}(C)+ P_{AG}(s1)L^B_{i+1}(G)+ P_{AT}(s1)L^B_{i+1}(T)] \times [P_{AA}(s2)L^C_{i+1}(A)+ P_{AC}(s2)L^C_{i+1}(C)+ P_{AG}(s2)L^C_{i+1}(G)+ P_{AT}(s2)L^C_{i+1}(T)]$$

$$L^A_{i+1}(C) = [P_{CA}(s1)L^B_{i+1}(A)+ P_{CC}(s1)L^B_{i+1}(C)+ P_{CG}(s1)L^B_{i+1}(G)+ P_{CT}(s1)L^B_{i+1}(T)] \times [P_{CA}(s2)L^C_{i+1}(A)+ P_{CC}(s2)L^C_{i+1}(C)+ P_{CG}(s2)L^C_{i+1}(G)+ P_{CT}(s2)L^C_{i+1}(T)]$$

$$L^A_{i+1}(G) = [P_{GA}(s1)L^B_{i+1}(A)+ P_{GC}(s1)L^B_{i+1}(C)+ P_{GG}(s1)L^B_{i+1}(G)+ P_{GT}(s1)L^B_{i+1}(T)] \times [P_{GA}(s2)L^C_{i+1}(A)+ P_{GC}(s2)L^C_{i+1}(C)+ P_{GG}(s2)L^C_{i+1}(G)+ P_{GT}(s2)L^C_{i+1}(T)]$$

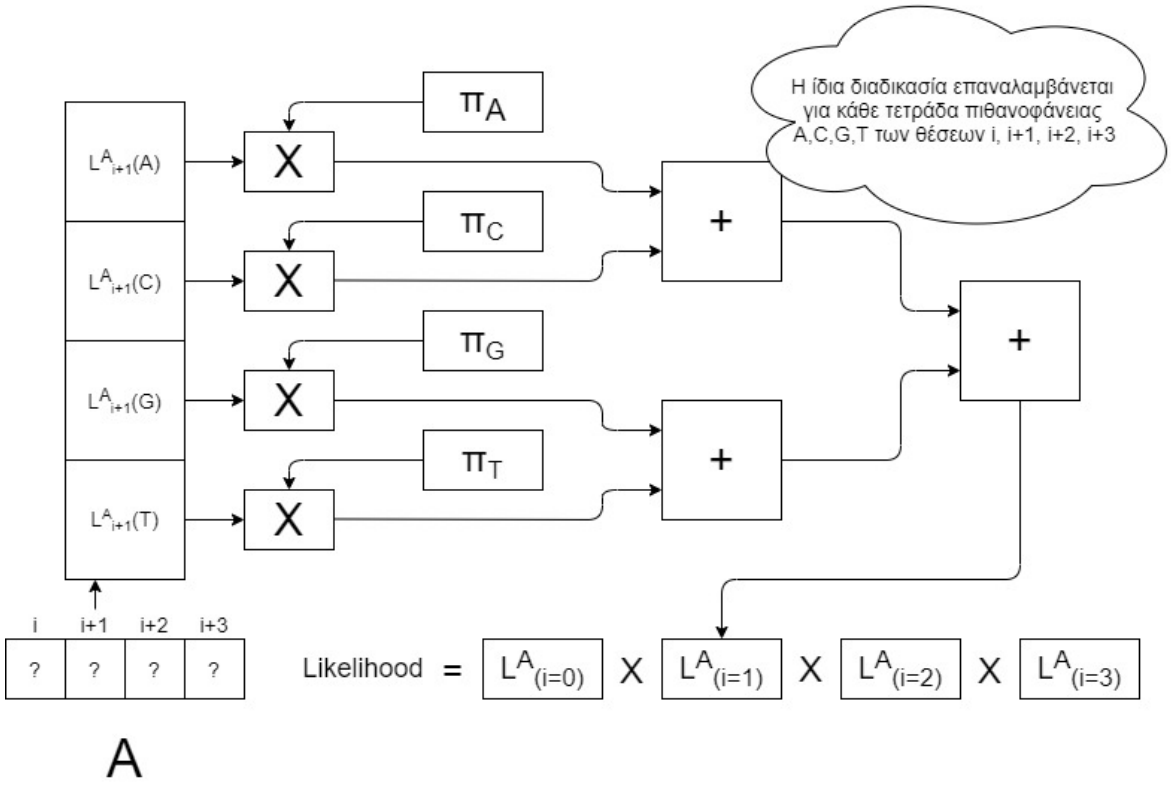
$$L^A_{i+1}(T) = [P_{TA}(s1)L^B_{i+1}(A)+ P_{TC}(s1)L^B_{i+1}(C)+ P_{TG}(s1)L^B_{i+1}(G)+ P_{TT}(s1)L^B_{i+1}(T)]$$

x $[P_{TA}(s2)L^{C}_{i+1}(A)+ P_{TC}(s2)L^{C}_{i+1}(C)+ P_{TG}(s2)L^{C}_{i+1}(G)+ P_{TT}(s2)L^{C}_{i+1}(T)]$

Μετά την ολοκλήρωση της παραπάνω διαδικασίας, πολλαπλασιάζονται οι πιθανοφάνειες του κοινού προγόνου A με τις συχνότητες βάσεων π των νουκλεοτιδίων A, C, G και T. Τέλος η συνολική πιθανοφάνεια του δέντρου υπολογίζεται πολλαπλασιάζοντας τις πιθανοφάνειες των θέσεων i του κοινού προγόνου A μεταξύ τους. Η μαθηματική εξίσωση της διαδικασίας υπολογισμού μεταξύ των πιθανοφανειών και των συχνοτήτων βάσεων είναι η εξής:

$$L(i) = \sum_{j} \pi_j \, L(X_i^A = j) \text{, όπου } j = A, C, G \text{ και } T \quad (2)$$

Η Εικόνα 3. 4 υποδεικνύει πώς λειτουργεί η σχέση (2) σχηματικά.



Εικόνα 3. 4 Υπολογισμός τελικής πιθανοφάνειας του κοινού προγόνου A

Αφού εφαρμοστεί η σχέση (2), η τελική ζητούμενη πιθανοφάνεια του προγόνου A υπολογίζεται πολλαπλασιάζοντας τις 4 πιθανοφάνειες των i θέσεων, όπως διακρίνεται στην Εικόνα 3. 4. Η μαθηματική εξίσωση υπολογισμού είναι η:

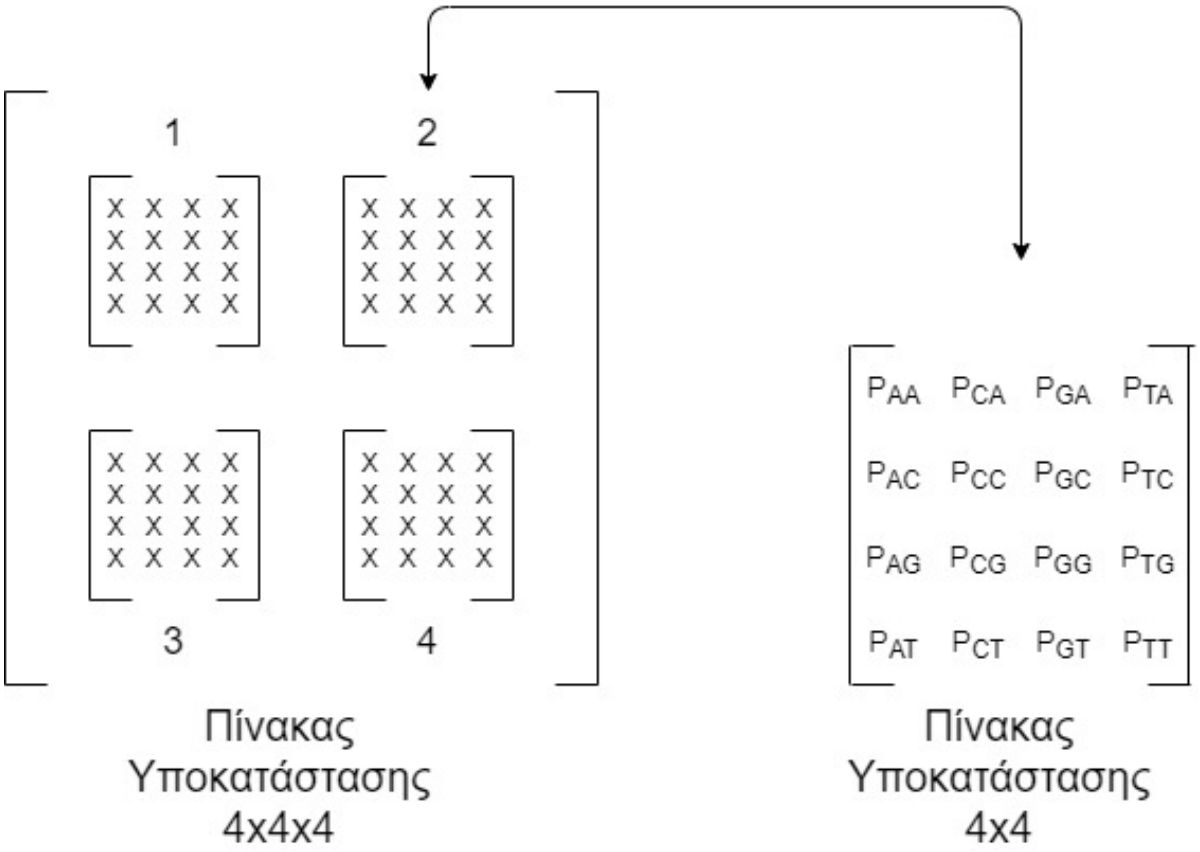
$$L = \prod_{i} L(i) \quad (3)$$

Ωστόσο, ο υπολογισμός της πιθανοφάνειας της σχέσης (3) δίνει συνήθως ένα αποτέλεσμα πολύ μικρό αριθμητικά, το οποίο μπορεί να επιφέρει αριθμητικές υπερχειλίσεις. Για την αντιμετώπιση αυτού του προβλήματος, συνηθίζεται να χρησιμοποιείται λογαριθμική προσέγγιση έναντι των πολλαπλασιασμών που πραγματεύεται η σχέση (3) [13]. Η μαθηματική εξίσωση του λογαριθμικού υπολογισμού του προγόνου A είναι η εξής:

$$L = \sum_{i} \log(L(i)) \quad (4)$$

3.5 Προσαρμογές RAxML

Όπως αναλύθηκε προηγουμένως, οι πίνακες υποκατάστασης Left και Right είναι μεγέθους 4x4. Κάθε φορά, ωστόσο, που μελετάται η πιθανοφάνεια ενός δέντρου μεταξύ τριών κόμβων, οι πίνακες ρυθμών υποκατάστασης αλλάζουν, λόγω ετερογένειας. Αυτό πρακτικά σημαίνει ότι, κάθε διάνυσμα έχει διαφορετικούς ρυθμούς με τους οποίους ένα νουκλεοτίδιο αντικαθίσταται από κάποιο άλλο σε πεπερασμένο χρονικό διάστημα dt. Αναλυτικότερα, κάθε διάνυσμα Ν διαφέρει στον ρυθμό υποκατάστασης των νουκλεοτιδίων του από τα υπόλοιπα διανύσματα, με αποτέλεσμα ο όγκος σε πραγματικά δεδομένα να είναι μεγάλος. Λόγω του μεγάλου όγκου, ο Yang [16] πρότεινε ένα μοντέλο που ονομάζεται διακεκριμένο Γ μοντέλο (Discrete Gamma Model). Το discrete Γ μοντέλο χρησιμοποιεί έναν διαφορετικό πίνακα υποκατάστασης μεγέθους 4x4x4, για όλο το μήκος Ν των αλληλουχιών. Ο πίνακας 4x4x4 είναι το αποτέλεσμα μιας διαδικασίας ομαλοποίησης όλων των ρυθμών υποκατάστασης, για κάθε μια θέση Ν των αλληλουχιών. Το αποτέλεσμα της ομαλοποίησης είναι η αναφορά διαφορετικού πίνακα υποκατάστασης για κάθε μία θέση i των Ν διανυσμάτων. Πρόκειται για 4 πίνακες υποκατάστασης που προσεγγίζουν όλα τα διανύσματα Ν, όπως φαίνεται στην Εικόνα 3. 5. Η διπλωματική αυτή χρησιμοποιεί το διακεκριμένο Γ μοντέλο μαζί με το GTR μοντέλο μέσω του RAxML, δηλαδή το GTR-GAMMA μοντέλο [16].

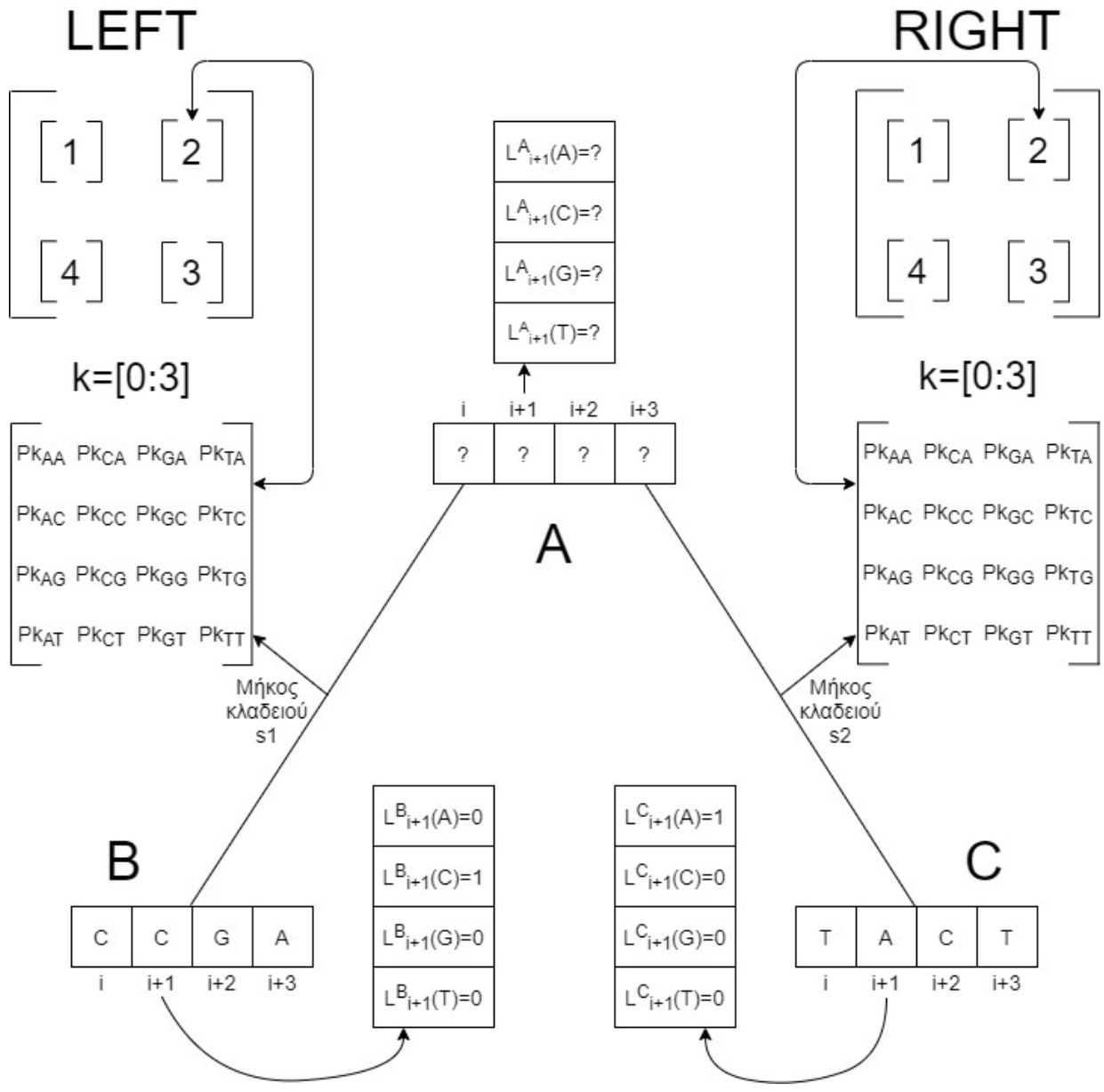


Εικόνα 3. 5 Πίνακας υποκατάστασης 4x4x4 διακεκριμένου Γ μοντέλου

Σύμφωνα με την Εικόνα 3. 5, ο νέος πίνακας υποκατάστασης του Γ μοντέλου αποτελείται από 4 πίνακες υποκατάστασης για μήκος κλαδιού S. Η εισαγωγή του νέου πίνακα υποκατάστασης μειώνει αισθητά τον όγκο των δεδομένων προς επεξεργασία, δίχως να προσθέτει παραπάνω όγκο πράξεων προς εκτέλεση. Αναλυτικότερα, κάθε Ν διάνυσμα δεν αντιπροσωπεύεται από το δικό του ξεχωριστό πίνακα υποκατάστασης, αλλά από μόλις έναν πίνακα υποκατάστασης για όλα τα Ν διανύσματα. Η παρακάτω μαθηματική εξίσωση εξηγεί πως επιτυγχάνεται αυτό.

$$L(X_{Aj} = i) = [\sum_{z} P_L k_{iz}(S_{AB}) L(X_{Bj} = z)][\sum_{z} P_R k_{iz}(S_{AC}) L(X_{Cj} = z)] \quad (5)$$

Όπως παρατηρείται, η μαθηματική σχέση (5) είναι πανομοιότυπη με την (1). Η διαφορά τους έγκειται στο ότι, κάθε φορά που μελετάται μία ξεχωριστή θέση πιθανοφάνειας i της ζητούμενης ακολουθίας A, οι πιθανοφάνειες των δύο ακολουθιών B και C πολλαπλασιάζονται με διαφορετικούς πίνακες υποκατάστασης k. Η Εικόνα 3. 6 εξηγεί σχηματικά την παραπάνω μαθηματική εξίσωση (5).



Εικόνα 3. 6 Υπολογισμός A κοινού προγόνου B και C, εφαρμόζοντας τις ιδιαιτερότητες του RAxML

Όπως διακρίνεται στην Εικόνα 3. 6, χρησιμοποιούνται διαφορετικοί πίνακες $P_{Left}k(s1)$ και $P_{Right}k(s2)$, για κάθε τετράδα πιθανοφάνειας θέσης i, στις B και C ακολουθίες αντίστοιχα. Οι παρακάτω πράξεις βασίζονται στην σχέση (5) και επιδεικνύουν πώς λειτουργεί ο αλγόριθμος της μέγιστης πιθανοφάνειας με τους δύο νέους πίνακες Left και Right.

$$L^A_{i+1}(A) = [P_L2_{AA}(s1)L^B_{i+1}(A) + P_L2_{AC}(s1)L^B_{i+1}(C) + P_L2_{AG}(s1)L^B_{i+1}(G) + P_L2_{AT}(s1)L^B_{i+1}(T)]$$
$$\text{x} \quad [P_R2_{AA}(s2)L^C_{i+1}(A) + P_R2_{AC}(s2)L^C_{i+1}(C) + P_R2_{AG}(s2)L^C_{i+1}(G) + P_R2_{AT}(s2)L^C_{i+1}(T)]$$

$$L^{A}_{i+1}(C) = [P_L2_{CA}(s1)L^{B}_{i+1}(A) + P_L2_{CC}(s1)L^{B}_{i+1}(C) + P_L2_{CG}(s1)L^{B}_{i+1}(G) + P_L2_{CT}(s1)L^{B}_{i+1}(T)] \times [P_R2_{CA}(s2)L^{C}_{i+1}(A) + P_R2_{CC}(s2)L^{C}_{i+1}(C) + P_R2_{CG}(s2)L^{C}_{i+1}(G) + P_R2_{CT}(s2)L^{C}_{i+1}(T)]$$

$$L^{A}_{i+1}(G) = [P_L2_{GA}(s1)L^{B}_{i+1}(A) + P_L2_{GC}(s1)L^{B}_{i+1}(C) + P_L2_{GG}(s1)L^{B}_{i+1}(G) + P_L2_{GT}(s1)L^{B}_{i+1}(T)] \times [P_R2_{GA}(s2)L^{C}_{i+1}(A) + P_R2_{GC}(s2)L^{C}_{i+1}(C) + P_R2_{GG}(s2)L^{C}_{i+1}(G) + P_R2_{GT}(s2)L^{C}_{i+1}(T)]$$

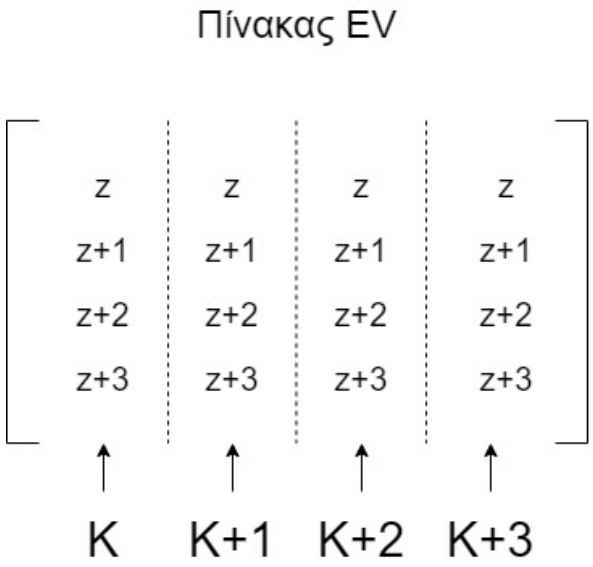
$$L^{A}_{i+1}(T) = [P_L2_{TA}(s1)L^{B}_{i+1}(A) + P_L2_{TC}(s1)L^{B}_{i+1}(C) + P_L2_{TG}(s1)L^{B}_{i+1}(G) + P_L2_{TT}(s1)L^{B}_{i+1}(T)] \times [P_R2_{TA}(s2)L^{C}_{i+1}(A) + P_R2_{TC}(s2)L^{C}_{i+1}(C) + P_R2_{TG}(s2)L^{C}_{i+1}(G) + P_R2_{TT}(s2)L^{C}_{i+1}(T)]$$

Παρατηρείται ότι, σε κάθε θέση i των ακολουθιών B και C αντιστοιχούν διαφορετικοί πίνακες υποκατάστασης k=[1:4]. Πιο συγκεκριμένα, όταν μελετάται η θέση i των ακολουθιών, για την ακολουθία B ο πίνακας υποκατάστασης Left είναι ο P_L1. Αντίστοιχα, για την ακολουθία C ο πίνακας υποκατάστασης Right είναι ο P_R1. Για την i+1 θέση, οι πίνακες υποκατάστασης είναι οι P_L2 και P_R2 για B και C αντίστοιχα, κ.ο.κ.

Επιπρόσθετα, αφού υπολογιστούν οι πιθανοφάνειες των i θέσεων, το RAxML πραγματεύεται μια σειρά πράξεων μεταξύ ενός πίνακα 4x4, που ονομάζεται EV, και των τετράδων πιθανοφανειών του κοινού προγόνου A [17]. Αναλυτικότερα, κάθε τετράδα πιθανοφάνειας θέσης i του A διανύσματος πολλαπλασιάζεται και προσθέτεται με όλο τον πίνακα EV. Ο πίνακας EV (Eigen Vector), όπως υποδεικνύουν και τα αρχικά του, αποτελεί τα ιδιοδιανύσματα της σχέσης $P(s) = e^{Qs}$. Ο λόγος για τον οποίο το RAxML πραγματοποιεί αυτές τις πρόσθετες πράξεις στο GTR-GAMMA μοντέλο έγκειται στο να απλοποιηθούν οι υπολογισμοί στη ρίζα A του δέντρου καθώς και η διάρκεια της βελτιστοποίησης του μήκους των κλαδιών του δέντρου. Η μαθηματική εξίσωση που περιγράφει τις πράξεις που γίνονται είναι η εξής:

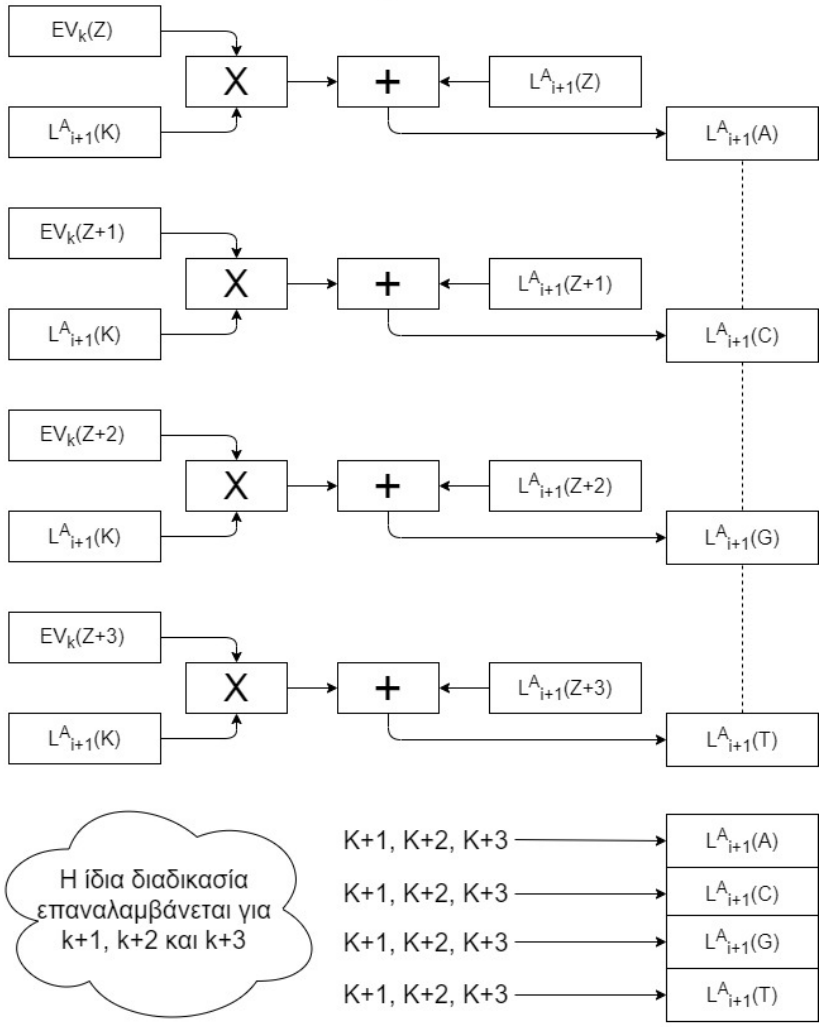
$$L(X_{Aj} = i) = \sum_{K} \sum_{z} EV_K(z) L(X_{Aj} = K) \quad (6)$$

Η παράμετρος K της σχέσης (6) συμβολίζει τις 4 πιθανοφάνειες μιας θέσης i του διανύσματος A και συνάμα μία τετράδα ιδιοδιανυσμάτων από τον πίνακα EV. Όπως διακρίνεται στην σχέση (6), κάθε μία πιθανοφάνεια θέσης i του A διανύσματος πολλαπλασιάζεται με μία τετράδα ιδιοδιανυσμάτων του πίνακα EV και στη συνέχεια το αποτέλεσμα προστίθεται αθροιστικά, δημιουργώντας τις τελικές πιθανοφάνειες του A διανύσματος. Πιο συγκεκριμένα, ο πίνακας ιδιοδιανυσμάτων EV είναι ο εξής:



Εικόνα 3. 7 Πίνακας ιδιοδιανυσμάτων EV

Όπως παρατηρείται στην Εικόνα 3. 7, σε κάθε μία τετράδα του πίνακα EV αντιστοιχεί διαφορετικό K. Σχηματικά, η διαδικασία της σχέσης (6) παρατίθεται στην Εικόνα 3. 8.



Εικόνα 3. 8 Πράξεις μεταξύ ιδιοδιανυσμάτων και πιθανοφανειών

Αναλυτικότερα, αν γίνει ανάπτυξη της σχέσης (6), -για μία τετράδα πιθανοφάνειας μιας θέσης i του A διανύσματος-, οι πράξεις είναι οι εξής:

$$L^{A}_{i+1}(A) = [EV_k(Z)\, L^{A}_{i+1}(K) + EV_{k+1}(Z)\, L^{A}_{i+1}(K+1) + EV_{k+2}(Z)\, L^{A}_{i+1}(K+2) + EV_{k+3}(Z) L^{A}_{i+1}(K+3)]$$

$$L^{A}_{i+1}(C) = [EV_k(Z+1)\, L^{A}_{i+1}(K) + EV_{k+1}(Z+1)\, L^{A}_{i+1}(K+1) + EV_{k+2}(Z+1)\, L^{A}_{i+1}(K+2) + EV_{k+3}(Z+1) L^{A}_{i+1}(K+3)]$$

$$L^{A}_{i+1}(A) = [EV_k(Z+2)\, L^{A}_{i+1}(K) + EV_{k+1}(Z+2)\, L^{A}_{i+1}(K+1) + EV_{k+2}(Z+2)\, L^{A}_{i+1}(K+2) + EV_{k+3}(Z+2)\, L^{A}_{i+1}(K+3)]$$

$$L^{A}_{i+1}(A) = [EV_k(Z+3)\, L^{A}_{i+1}(K) + EV_{k+1}(Z+3)\, L^{A}_{i+1}(K+1) + EV_{k+2}(Z+3)\, L^{A}_{i+1}(K+2) + EV_{k+3}(Z+3)\, L^{A}_{i+1}(K+3)]$$

Αφού ολοκληρωθεί και η παραπάνω διαδικασία, το RAxML συνεχίζει τον υπολογισμό της συνολικής πιθανοφάνειας του δέντρου, όπως αναλύθηκε στην παράγραφο 3.4.

Κεφάλαιο 4: Αποκεντρωμένοι πόροι

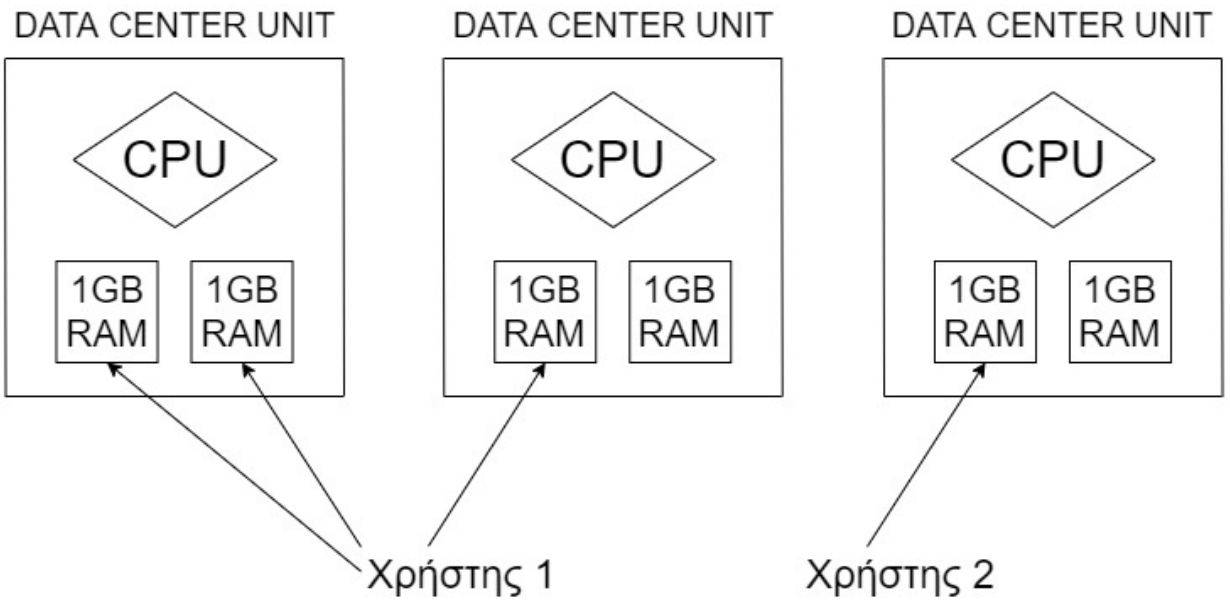
Το τέταρτο κεφάλαιο πραγματεύεται την έννοια των αποκεντρωμένων πόρων. Σε αυτό το κεφάλαιο αναλύονται τα οφέλη ενός συστήματος αποκεντρωμένων πόρων συγκριτικά με τα απλά συστήματα.

4.1 Η έννοια των αποκεντρωμένων πόρων

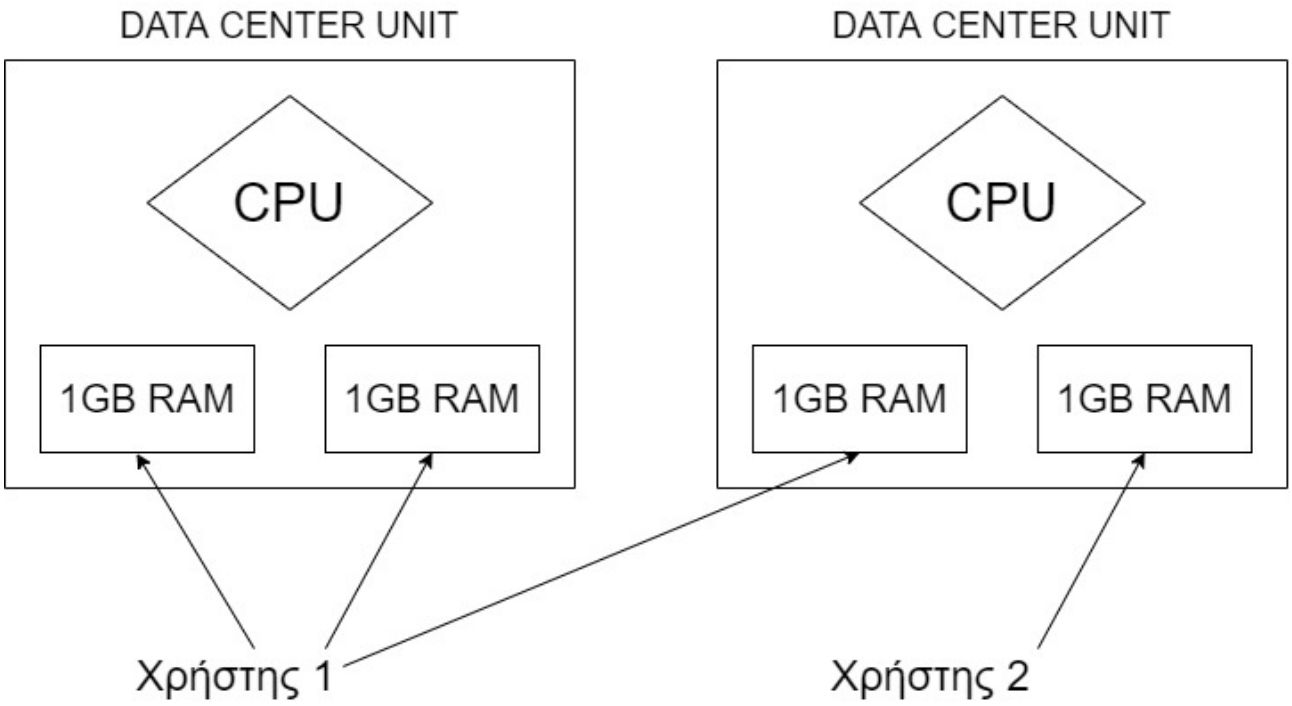
Οι αποκεντρωμένοι πόροι, ως έννοια, αφορούν το διαμοιρασμό των εργασιών και διεργασιών σε 2 ή περισσότερα μεμονωμένα και απομακρυσμένα μέρη. Αναλυτικότερα, πρόκειται για το διαμερισμό των διαφόρων εργασιών σε μικρότερες, με στόχο την καλύτερη απόδοση στο λιγότερο δυνατό χρόνο [18], [19].

Στα πλαίσια της αρχιτεκτονικής υπολογιστών, οι αποκεντρωμένοι πόροι παίζουν σημαντικό ρόλο, καθώς εξασφαλίζουν εξοικονόμηση πόρων, ικανοποιητική τοπικότητα και πιο ευνοϊκές συνθήκες για την αποδοτική εφαρμογή του παραλληλισμού, δίχως πολλούς και χρονοβόρους ελέγχους.

Λόγου χάρη, έστω ένα Data Center (DC) με διαθέσιμους εξυπηρετητές 2GBytes RAM ο καθένας, όπως φαίνεται στην Εικόνα 4. 1. Κάποια χρονική στιγμή, 2 χρήστες κάνουν αίτηση για να χρησιμοποιήσουν τους εξυπηρετητές. Ο πρώτος χρήστης (χρήστης 1) απαιτεί από τον εξυπηρετητή 3GByte RAM μνήμης, ενώ ο δεύτερος χρήστης (χρήστης 2) απαιτεί 1GByte μνήμης RAM. Στην περίπτωση που το DC δεν είναι κατασκευασμένο αρχιτεκτονικά με την λογική των αποκεντρωμένων πόρων, χρειάζονται 3 εξυπηρετητές για την ικανοποίηση των απαιτήσεων των χρηστών 1 και 2, όπως φαίνεται στην Εικόνα 4. 1. Αντίθετα, σε ένα DC αποκεντρωμένων πόρων επιτυγχάνεται καλύτερη τοπικότητα, εξοικονόμηση πόρων και ενέργειας, αφού χρειάζονται μόλις 2 εξυπηρετητές για την κάλυψη των απαιτήσεων και των 2 χρηστών. Το σενάριο ενός DC αποκεντρωμένων πόρων φαίνεται στην Εικόνα 4. 2.



Εικόνα 4. 1 Παράδειγμα εξυπηρέτησης χρηστών από εξυπηρετητές ενός DC χωρίς αποκεντρωμένους πόρους



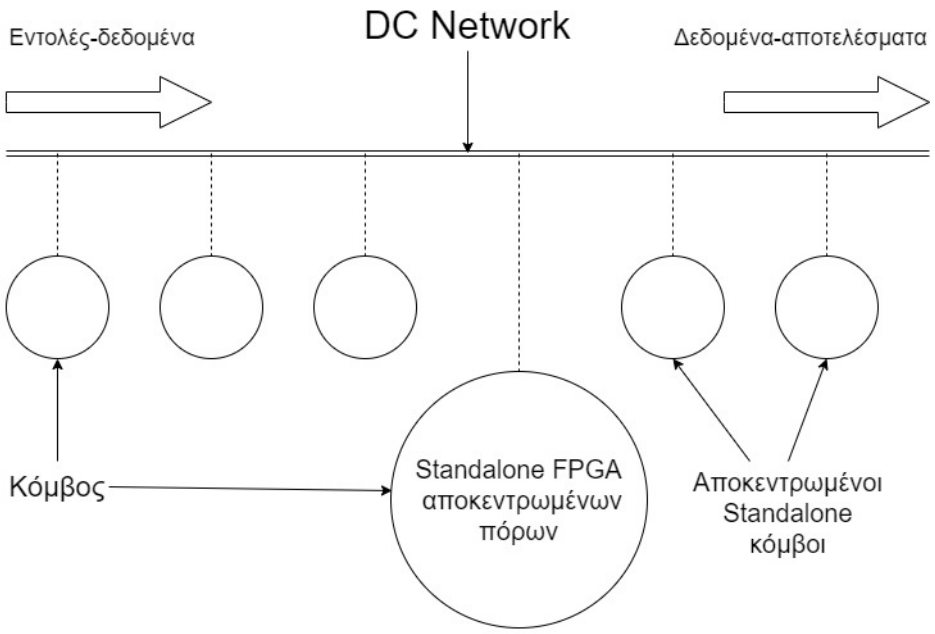
Εικόνα 4. 2 Παράδειγμα εξυπηρέτησης χρηστών από εξυπηρετητές ενός DC αποκεντρωμένων πόρων

Σε συσχετισμό με το παράδειγμα της Εικόνα 4. 2, η διάσπαση παίζει σημαντικό ρόλο στον κλάδο της αρχιτεκτονικής των υπολογιστικών συστημάτων, τόσο στα DC, όπως αναλύθηκε παραπάνω, όσο και στις FPGAs στα πλαίσια της διπλωματικής εργασίας.

4.2 Data Centers & FPGAs

Η ανάγκη για μεγάλης ταχύτητας DCs, λόγω της ραγδαίας αύξησης των δεδομένων, έφερε τα κατανεμημένα συστήματα. Σήμερα όλα τα συστήματα που περιέχουν εξυπηρετητές, συστήματα ονοματοδοσίας, δρομολογητές κ.α., είναι σχεδιασμένα ως κατανεμημένα συστήματα. Τα συστήματα αυτά διασπούν το φόρτο εργασίας των εντολών τους σε μικρότερες διεργασίες και τις δρομολογούν στα DCs για παράλληλη εκτέλεση. Η όλη διαδικασία ελέγχεται από CPU-Hosts των κατανεμημένων συστημάτων που συγχρονίζουν την διαδικασία αποστολής και λήψης των δρομολογηθέντων διεργασιών στα DCs. Μόλις ολοκληρωθεί η άφιξη των διεργασιών στα DCs, ξεκινάει η εκτέλεση αυτών παράλληλα και κατανεμημένα. Ωστόσο η ταχύτητα εκτέλεσης των διεργασιών δεν είναι αρκετή για την κάλυψη των αναγκών της επεξεργασίας πολλών δεδομένων. Το κύριο πρόβλημα είναι πως οι πόρ,οι των DCs, όπως μνήμη, υπολογιστική δύναμη και αποθηκευτικός χώρος, δεν είναι αποκεντρωμένοι αναμεταξύ τους. Η λύση του προβλήματος έγκειται στην κατασκευή ανεξάρτητων (standalone) αποκεντρωμένων και απομακρυσμένων πόρων.

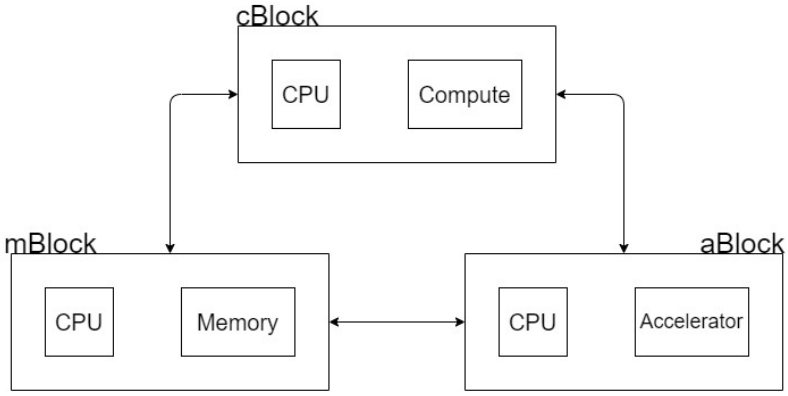
Η εισαγωγή των FPGAs στα DCs μπορούν να προσφέρουν τη λειτουργία standalone πόρων για όλες τις διεργασίες που πρέπει να εκτελεστούν. Αναλυτικότερα, κάθε FPGA διακρίνεται σε standalone μνήμη, επιταχυντή και υπολογιστική μονάδα (CPU) για συγχρονισμό δημιουργώντας, με αυτόν τον τρόπο, ένα σύνολο αποκεντρωμένων και standalone πόρων. Υπάρχει, δηλαδή, πλήρης ανεξαρτητοποίηση των πόρων των standalone FPGAs αναμεταξύ τους και κυρίως αποξένωση μεταξύ των πόρων των FPGA και των πόρων των DCs. Το αποτέλεσμα αυτής της αποκεντρωμένης μεθόδου είναι η δημιουργία κόμβων για τη σύνδεση των αποκεντρωμένων FPGAs με τα DCs, όπως φαίνεται στην Εικόνα 4. 3 [18], [20], [21].



Εικόνα 4. 3 Δίκτυο DC με standalone FPGAs με αποκεντρωμένους πόρους

4.3 FPGA πλατφόρμες αποκεντρωμένων πόρων

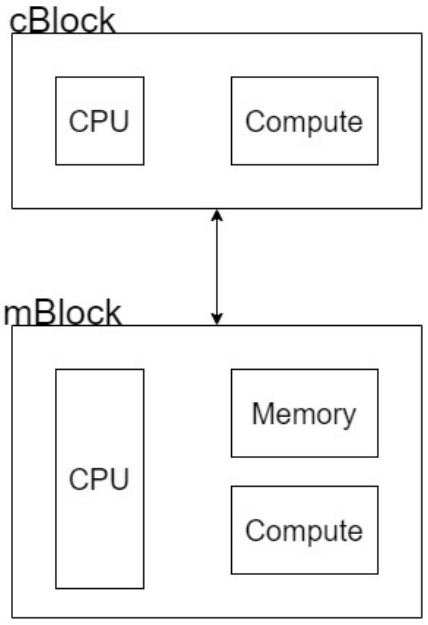
Όπως αναφέρθηκε στην παράγραφο 4.2, οι FPGAs μπορούν να λειτουργήσουν απομονωμένοι αναμεταξύ τους και σε ένα δίκτυο, δημιουργώντας μία πλατφόρμα από αποκεντρωμένους πόρους. Ωστόσο οι FPGAs μπορούν να χρησιμοποιηθούν με διάφορους τρόπους για την κατασκευή μιας πλατφόρμας αποκεντρωμένων πόρων, δίχως την ανάγκη ύπαρξης κάποιου εξωτερικού και διαφορετικού πόρου, όπως ένα DC. Μία τέτοια προσέγγιση, βασισμένη μόνο σε FPGAs(FPGA-based), αποτελεί το Project του Theodoropoulos, et al. [22]. Το συγκεκριμένο Project χρησιμοποιεί 3 απομακρυσμένες FPGAs αποκεντρωμένων πόρων. Κάθε FPGA λειτουργεί ανεξάρτητα από τις υπόλοιπες, δίχως να χρειάζεται κάποιος έλεγχος από τις άλλες FPGAs. Μία από τις 3 FPGAs λειτουργεί ως Host, μία άλλη ως Memory και η τρίτη ως επιταχυντής. Η FPGA Host (cBlock) αποτελεί τον εγκέφαλο όλου του Project, καθώς είναι υπεύθυνη για την μεταφορά δεδομένων, διεργασιών και το συγχρονισμό των FPGAs. Η FPGA Memory (mBlock) περιέχει πολλούς πόρους μνήμης και η FPGA επιταχυντής (aBlock) περιέχει επιταχυντικά κυκλώματα. Άπαντες FPGAs εργάζονται με την δικιά τους τοπική επεξεργαστική μονάδα (PU) για το συγχρονισμό των τοπικών πόρων τους και είναι ικανές για Block-to-Block υψηλής ταχύτητας μεταφοράς δεδομένων χρησιμοποιώντας απλά ή προσαρμοσμένα κυκλώματα εισόδου-εξόδου (I/O). Η Εικόνα 4. 4 εξηγεί σχηματικά την σύνδεση των FPGAs του Project [22].



Εικόνα 4. 4 Εικονική αναπαράσταση και σύνδεση των τριών Blocks του Project

Το Project του Theodoropoulos, et al. [22], παρουσιάζει ένα παράδειγμα πολλαπλασιασμού πινάκων για την αξιολόγηση των αποκεντρωμένων πόρων που υλοποιεί. Για μικρού μεγέθους πίνακες, η χρήση του cBlock πραγματοποιεί τον πιο γρήγορο υπολογισμό σε συσχετισμό με την χρήση των 2 υπολοίπων αποκεντρωμένων mBlock και aBlock. Ο λόγος που οι τελευταίες αποκεντρωμένες FPGAs δεν έχουν καλή απόδοση για λίγα δεδομένα έγκειται στο ότι, ο περισσότερος χρόνος σπαταλάται στις Block-to-Block μεταφορές δεδομένων και μηνυμάτων συγχρονισμού και ο υπολειπόμενος χρόνος που χρησιμοποιείται για την ολοκλήρωση των υπολογισμών είναι λίγος. Ωστόσο, όσο αυξάνονται τα μεγέθη των πινάκων, τόσο κερδίζει έδαφος το αποκεντρωμένο aBlock που περιέχει επιταχυντικά κυκλώματα. Αυτό συμβαίνει καθώς, τα δεδομένα είναι πολλά και χρειάζεται παραπάνω χρόνος για τον υπολογισμό των πράξεων απ' ότι για τη μεταφορά των δεδομένων και το συγχρονισμό των Blocks. Συνοψίζοντας, συγκριτικά με τα απλά συστήματα και τις πλατφόρμες, η απόδοση των αποκεντρωμένων πόρων αυξάνεται αναλογικά με την αύξηση του υπολογιστικού όγκου.

Μια ακόμη προσέγγιση της προσφοράς των αποκεντρωμένων πόρων γίνεται διακριτή με το Project REMAP (Remote mEmory Manager for disAggregated Platforms) του Theodoropoulos, et al. [23]. Το REMAP προσεγγίζει μια υλοποίηση αποκεντρωμένων πόρων για την απομακρυσμένη διαχείριση της μνήμης. Η λογική του REMAP είναι παρόμοια με το Project του Theodoropoulos, et al. [22]. Για το REMAP χρησιμοποιούνται 2 FPGAs συνδεδεμένες αναμεταξύ τους. Μία FPGA παίρνει τον ρόλο της FPGA-HOST (cBlock) και η άλλη της FPGA-MEM (mBlock). Το cBlock είναι ο εγκέφαλος του Project, καθώς δρομολογεί τις συναλλαγές των μηνυμάτων και των δεδομένων μεταξύ των 2 Blocks. Όταν γίνεται μια αίτηση από το cBlock στο mBlock, μέσω υψηλής απόδοσης I/O, πραγματοποιούνται συναλλαγές μεταξύ των 2 FPGAs με σκοπό την απομακρυσμένη πρόσβαση της φυσικής διεύθυνσης της αποκεντρωμένης μνήμης του mBlock. Σύμφωνα με τα αποτελέσματα του REMAP, μία εφαρμογή της τοπικής υπολογιστικής μονάδας (local Application Processing Unit-APU) μπορεί να πραγματοποιήσει δυναμική δέσμευση απομακρυσμένων και αποκεντρωμένων πόρων μνήμης, δίχως την αναγκαιότητα πρόσθετων βιβλιοθηκών ή αλλαγές στην APU. Η συνολική επιτάχυνση της αρχιτεκτονικής του REMAP, σε σύγκριση με την κλασσική διαμόρφωση CPU-Memory, κυμαίνεται έως και 1,3 φορές πιο γρήγορη. Η Εικόνα 4. 5 επιδεικνύει τη διάταξη των FPGAs του REMAP [23].

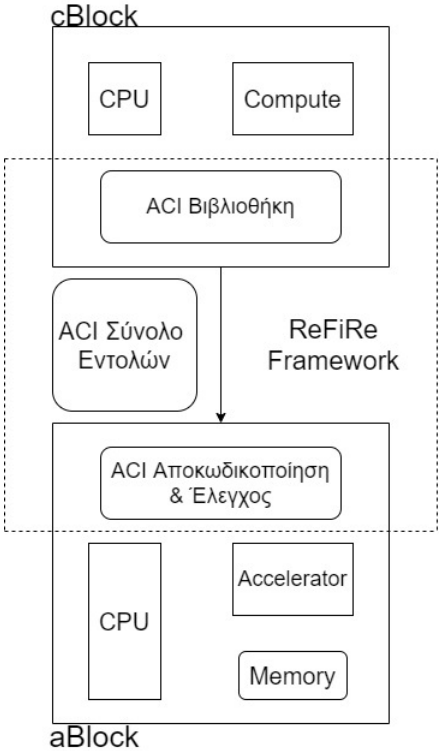


Εικόνα 4. 5 Εικονική αναπαράσταση και σύνδεση των Compute και Memory Blocks του Project REMAP

Η παραπάνω αναφορά των Projects του Theodoropoulos, et al. [22] και REMAP επεξηγεί το πώς οι αποκεντρωμένοι πόροι έχουν την ικανότητα να ανεβάσουν τις αποδόσεις σε σύγκριση με απλές πλατφόρμες. Ένα ακόμη Project, που αξίζει να αναλυθεί και στο οποίο βασίζεται η αρχιτεκτονική σχεδίαση των αποκεντρωμένων πόρων της παρούσας εργασίας, είναι το Project ReFiRe (Remote Fine-grained Reconfigurable acceleration) των Pissadakis et al. [24]. Το ReFiRe, όπως και το REMAP, είναι μία υλοποίηση με 2 απομακρυσμένες και αποκεντρωμένες FPGAs. Η μία FPGA λειτουργεί ως FPGA-HOST και η άλλη ως FPGA επιταχυντής. Το cBlock, όπως αναφέρθηκε και στα 2 προηγούμενα Projects του Theodoropoulos, et al. [22] και REMAP, είναι υπεύθυνο για τη μεταφορά δεδομένων, το συντονισμό και το συγχρονισμό των 2 Blocks. Το aBlock, όπως αναφέρθηκε στο Project του Theodoropoulos, et al. [22], περιέχει επιταχυντικά κυκλώματα. Συγκεκριμένα, το aBlock του ReFiRe περιέχει κόμβους στους οποίους συνδέονται επιταχυντές. Οι επιταχυντές αυτοί χρησιμοποιούν τη δικιά τους μνήμη, καθώς επίσης και την κεντρική DDR μνήμη του aBlock μέσω ειδικού I/O κυκλώματος. Όπως γίνεται διακριτό, η σχέση του cBlock και aBlock του ReFiRe μοιάζει αρκετά με τη σχέση των αντίστοιχων 2 Blocks του Project του Theodoropoulos, et al. [22]. Η διαφορά τους έγκειται στον τρόπο με τον οποίο τα Blocks του ReFiRe επικοινωνούν μεταξύ τους. Η επικοινωνία των Blocks πραγματοποιείται από το ReFiRe Πλαίσιο-Δομή (Framework), το οποίο λειτουργεί σε 3 στάδια:

- Advanced Co-processor Instruction (ACI): Το ACI αποτελεί ένα σύνολο προσαρμοσμένων και πολύπλοκων εντολών για την επικοινωνία μεταξύ των 2 Blocks, το συγχρονισμό και τον απομακρυσμένο έλεγχο των επιταχυντών.
- Βιβλιοθήκη υποστήριξης ACI : Η συγκεκριμένη βιβλιοθήκη βοηθά το χρήστη σε επίπεδο εφαρμογής προγραμματιστικής διεπαφής (Application Programming Interface-API) με σκοπό την ομαλή μετατροπή των εντολών των εφαρμογών λογισμικού σε εντολές με τη φόρμα ACI.
- Έλεγχος αποκωδικοποίησης εντολών ACI: Αφορά μία εφαρμογή λογισμικού μιας μηχανής πεπερασμένων καταστάσεων (FSM) που αποκωδικοποιεί τις ACI εντολές και ελέγχει την ανασυγκροτήσιμη περιοχή.

Τα 3 παραπάνω στάδια πλαισιώνουν τη δομή της επικοινωνίας των Blocks αναμεταξύ τους μέσω του ReFiRe Framework, όπως φαίνεται στην Εικόνα 4. 6 [24].



Εικόνα 4. 6 Εικονική αναπαράσταση του ReFiRe Framework για την επικοινωνία του cBlock με το aBlock

Κεφάλαιο 5: Αρχιτεκτονική μεθόδου μέγιστης πιθανοφάνειας

Το κεφάλαιο αυτό εστιάζει στην επιλογή και ανάλυση της αρχιτεκτονικής σχεδίασης του επιταχυντή που κατασκευάστηκε για το πέρας της μεθόδου της μέγιστης πιθανοφάνειας.

5.1 Αρχιτεκτονική του επιταχυντή της μέγιστης πιθανοφάνειας

Κάθε hardware φέρει κάποιους περιορισμούς, όπως γίνονται γνωστοί από τον κατασκευαστή. Κατά συνέπεια, η αρχιτεκτονική σχεδίαση του κυκλώματος-επιταχυντή, που κατασκευάστηκε για τις ανάγκες της παρούσας μελέτης, ανταποκρίνεται στους περιορισμούς του hardware που χρησιμοποιήθηκε. Στο τρέχον κεφάλαιο, γίνεται γνωστή η αρχιτεκτονική σχεδίαση που επιλέχθηκε. Το κομμάτι της υλοποίησής της, ωστόσο, παρατίθεται στο κεφάλαιο 7. Η μέθοδος της μέγιστης πιθανοφάνειας για το GTR-GAMMA μοντέλο από το RAxML φαίνεται στην Εικόνα 5. 1 με τη μορφή ψευδο-κώδικα.

```
Για s από 0 σε N //Διανύσματα από τις θέσεις N
    x1[16] ← B(s) διάνυσμα
    x2[16] ← C(s) διάνυσμα
    x3[16] ← A(s) διάνυσμα

    //i παράμετρος: Υπολογισμός απόστασης μιας τετράδας στοιχείων x1 και x2
    //διανυσμάτων για συγκεκριμένη k θέση του πίνακα x1px2

    //k παράμετρος (πρώτη εμφάνιση): Υπολογισμός όλων των αποστάσεων και
    //πολ/σμός αυτωνών για τον υπολογισμό του πίνακα x1px2 για συγκεκριμένη j
    //τετράδα στοιχείων

    //j παράμετρος: Αλλαγή τετράδας στοιχείων x1 και x2 διανυσμάτων προς
    //επεξεργασία για την δημιουργία και των τεσσάρων x1px2 πινάκων

    //k παράμετρος (δεύτερη εμφάνιση), 1 παράμετρος: Υπολογισμός τελικού x3
    //διανύσματος για συγκεκριμένο πίνακα x1px2 ανά j τετράδες.

    Για j από 0 σε 3 //Υπολογισμός τεσσάρων τετράδων
        Για k από 0 σε 3 //Υπολογισμός πίνακα x1px2 για j
            Για i από 0 σε 3 //Υπολογισμός απόστασης για k
                //Υπολογισμός αριστερής απόστασης (B σε A)
                Temp_x=x1 ← x1[j*4+i] * Left[j*16+k*4+i]
                Ump_x=x1 ← Σ_{i=0}^{3} Temp_{x=x1}

                //Υπολογισμός δεξιάς απόστασης (C σε A)
                Temp_x=x2 ← x2[j*4+i] * Right[j*16+k*4+i]
                Ump_x=x1 ← Σ_{i=0}^{3} Temp_{x=x2}
            Τέλος

            //Πολ/σμός αποστάσεων αριστερού και δεξιού παιδιού
            x1px2_j[k] ← Ump_x=x1 * Ump_x=x2
        Τέλος
        Για k από 0 σε 3
            Για l από 0 σε 3
                //Υπολογισμός j τετράδας στοιχείων x3 διανύσματος
                X3[j*4+l] ← EV[k*4+l] * x1px2_j[k]
            Τέλος
        Τέλος
    Τέλος
Τέλος
```

Εικόνα 5. 1 Μέθοδος της Μέγιστης Πιθανοφάνειας

Η Εικόνα 5. 1, αντιπροσωπεύει τον αλγόριθμο της μέγιστης πιθανοφάνειας γραμμένο σε ψευδο-κώδικα. Σύμφωνα με την παράγραφο 3.4, οι 2 γνωστοί πίνακες πιθανοφάνειας B και C, όπως διακρίνεται στην Εικόνα 5. 1, καθώς και ο ζητούμενος πίνακας A αποτελούν τα διανύσματα N θέσεων των αλληλουχιών DNA προς μελέτη. Το μέγεθος των διανυσμάτων αντιστοιχεί σε 16 στοιχεία για κάθε θέση N μιας αλληλουχίας. Επομένως, πρόκειται για πίνακες A, B και C μεγέθους N*16. Επιπλέον, στην Εικόνα 5. 1 διακρίνονται και οι πίνακες Left, Right και EV. Οι πίνακες Left και Right είναι οι πίνακες υποκατάστασης του GTR-GAMMA μοντέλου, όπως αναλύθηκε εκτενώς στην παραγράφο 3.5. Ο πίνακας EV αποτελεί τα ιδιοδιανύσματα του μοντέλου εξέλιξης GTR.

Ο όγκος των δεδομένων και οι επαναλήψεις για όλο το μήκος N των αλληλουχιών κατέχουν σημαντικό ρόλο στη διεκπεραίωση της μεθόδου της μέγιστης πιθανοφάνειας. Πιο συγκεκριμένα, όσο αυξάνεται το N, τόσο αυξάνονται οι επαναλήψεις, ο όγκος των δεδομένων προς μελέτη, καθώς και ο όγκος των αποτελεσμάτων. Ωστόσο, παρ' όλο που το μέγεθος του N επηρεάζει το συνολικό όγκο των δεδομένων, δεν επηρεάζει το χρόνο που απαιτείται για κάθε μια ξεχωριστή επανάληψη της διαδικασίας. Συνεπώς, το αρχικό κέντρο εστίασης είναι μια αρχιτεκτονική που βελτιστοποιεί τη μια επανάληψη της όλης διαδικασίας, δηλαδή την επανάληψη για N=1.

5.2 Στρατηγική σχεδίασης αρχιτεκτονικής κυκλώματος για τη διεκπεραίωση της μεθόδου της μέγιστης πιθανοφάνειας

Η στρατηγική που εφαρμόστηκε για την πιο αποδοτική διεκπεραίωση της μεθόδου

Η στρατηγική για την κατασκευή της αρχιτεκτονικής που εφαρμόζεται στη διπλωματική αυτή είναι από μέσα προς τα έξω. Συγκεκριμένα, για αριθμό θέσεων N=1, ο αλγόριθμος της μέγιστης πιθανοφάνειας περιέχει 5 βρόχους επανάληψης, όπως φαίνεται στην Εικόνα 5. 1. Εφαρμόζοντας τη στρατηγική αυτή, τα loops της μεθόδου μελετήθηκαν ως διαφορετικά μέρη. Αρχικά, απομονώθηκε και χρησιμοποιήθηκε το πιο εσωτερικό loop, όπως φαίνεται στην Εικόνα 5. 1. Στη συνέχεια, ενσωματώθηκαν και τα υπόλοιπα loops στην αρχιτεκτονική, ένα τη φορά με τη σειρά.

Διαχωρισμός της μεθόδου

Προτού γίνει μετάβαση στη λεπτομερή ανάλυση της αρχιτεκτονικής σχεδίασης του κυκλώματος, πρέπει να επισημανθεί μια σημαντική παρατήρηση για τον αλγόριθμο της μέγιστης πιθανοφάνειας με το GTR-GAMMA μοντέλο. Οι 2 επαναλήψεις δεν είναι ανεξάρτητες από τις υπόλοιπες επαναλήψεις. Ο λόγος έγκειται στο ότι, οι πράξεις μέσα σε αυτές τις επαναλήψεις περιλαμβάνουν και τον x1px2 πίνακα, ο οποίος πρέπει να υπολογιστεί πριν εκτελεστεί οποιαδήποτε άλλη πράξη που αποτελείται από τον τελευταίο πίνακα. Λόγω αυτού, η παράλληλη εκτέλεση της μεθόδου δεν είναι εφικτή. Προς διευκόλυνση της διαδικασίας εκτέλεσης, τα 2 loops που υπολογίζουν τον x1px2 πίνακα ονομάστηκαν Likelihood Calculate. Αντίστοιχα, τα 2 άλλα loops που υπολογίζουν τον τελικό πίνακα x3, ονομάστηκαν EV Final Calculate. Με βάση αυτή τη διάκριση, το EV Final Calculate δεν μπορεί να εκτελεστεί, εάν πρώτα δεν ολοκληρωθεί η εκτέλεση του Likelihood Calculate.

5.2.1 Likelihood Calculate κύκλωμα της μεθόδου

Στην Εικόνα 5. 1, δημιουργήθηκε, αρχικά, αρχιτεκτονική για τους πολλαπλασιασμούς που πραγματοποιούνται στον αλγόριθμο μεταξύ των 2 διανυσμάτων από τους πίνακες X1 και X2 και των πινάκων ρυθμών υποκατάστασης Left και Right αντίστοιχα.

5.2.1.α Αρχιτεκτονική UMP κυκλώματος

Όλοι οι πολλαπλασιασμοί μπορούν να γίνουν παράλληλα στο χρόνο εκτέλεσης ενός πολλαπλασιασμού. Η επανάληψη που πραγματοποιείται υπολογίζει το άθροισμα των γινομένων τεσσάρων στοιχείων των διανυσμάτων X1 και X2. Στην πρώτη επανάληψη, λόγου χάριν, είναι:

x1[0] * left[0] και x2[0] * right[0]

Ωστόσο, στην Εικόνα 5. 1, παρατηρείται πως τα πρώτα 4 στοιχεία του X1 και X2 πίνακα πολλαπλασιάζονται με τα πρώτα 16 στοιχεία των πινάκων Left και Right αντίστοιχα. Αυτή η παρατήρηση επάγεται στο ότι, η αρχιτεκτονική του UMP κυκλώματος βασίζεται στην ορθή αναδιάταξη, στο συγχρονισμό, καθώς και στην ορθή και γρήγορη ροή των δεδομένων και τον υπολογισμό των αποτελεσμάτων.

Οι πολλαπλασιασμοί κάθε πίνακα πιθανοτήτων υποκατάστασης X και πίνακα ρυθμού υποκατάστασης Ypok για k=[0:3], -δηλαδή, τα 4 πρώτα στοιχεία(i=[0:3]) του πίνακα X-, υπολογίζονται ως εξής:

$UmpTemp_{i,k}[0]=X[i] * Ypok[k*4+i]$
$UmpTemp_{i,k}[0]=X[i] * Ypok[k*4+i]$
$UmpTemp_{i,k}[0]=X[i] * Ypok[k*4+i]$
$UmpTemp_{i,k}[0]=X[i] * Ypok[k*4+i]$

Οι παραπάνω πολλαπλασιασμοί για τον υπολογισμό των τεσσάρων πρώτων στοιχείων του πίνακα X επαναλαμβάνονται 4 φορές, όσες είναι και το k.

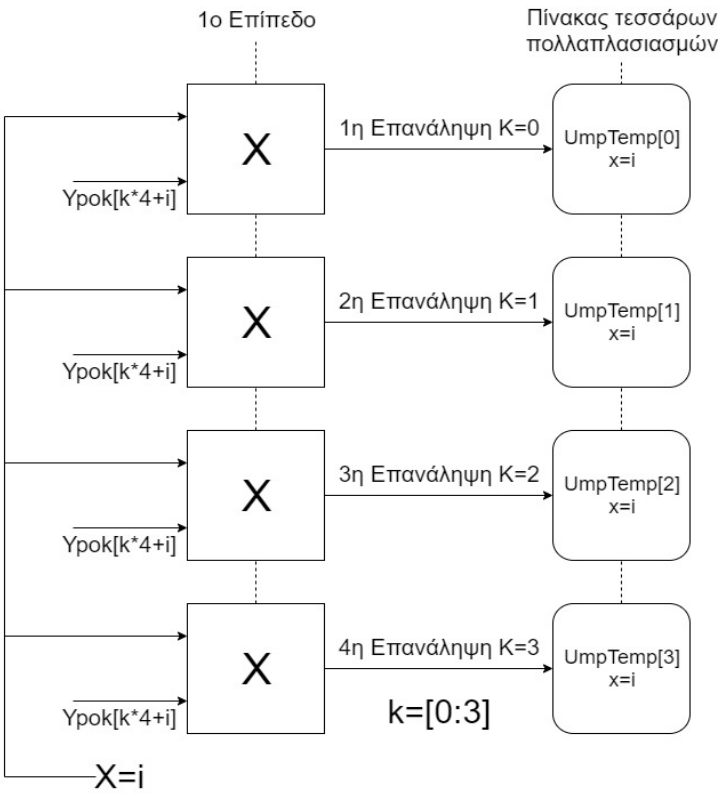
1η Τετράδα για k=0 και i=[0:3]
2η Τετράδα για k=1 και i=[0:3]
3η Τετράδα για k=2 και i=[0:3]
4η Τετράδα για k=3 και i=[0:3]

Οι πολλαπλασιασμοί, λοιπόν, στην προκειμένη είναι 16. Κάθε ένα στοιχείο του πίνακα X πολλαπλασιάζεται με μία τετράδα στοιχείων του πίνακα Ypok. Αυτή η τετράδα στοιχείων, όμως, δεν είναι σειριακή, αλλά πολλαπλάσια του K. Με βάση αυτό, μπορεί να πραγματοποιηθεί αναδιάταξη των πράξεων χωρίς να επηρεαστεί το τελικό αποτέλεσμα, με τον εξής τρόπο:

Nοστή τετράδα	N=[0:3]
$UmpTemp_{i,k}[0]=X[i] * Ypok[k*4+i]$	1η Τετράδα για i=0 και k=[0:3]
$UmpTemp_{i,k}[0]=X[i] * Ypok[k*4+i]$	2η Τετράδα για i=1 και k=[0:3]
$UmpTemp_{i,k}[0]=X[i] * Ypok[k*4+i]$	3η Τετράδα για i=2 και k=[0:3]
$UmpTemp_{i,k}[0]=X[i] * Ypok[k*4+i]$	4η Τετράδα για i=3 και k=[0:3]

Ο λόγος για τον οποίο επιλέχθηκε το παραπάνω μονοπάτι υπολογισμών κάνοντας ορθή αναδιάταξη των πράξεων, εξηγείται εκτενώς στο κεφάλαιο 7.

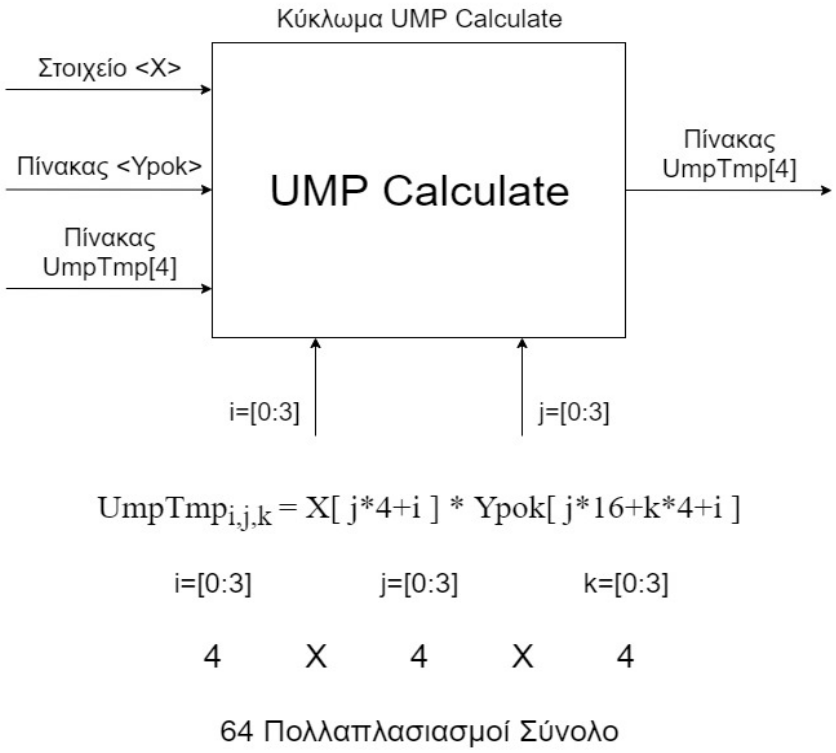
Έχοντας δομήσει την εικόνα για τον τρόπο με τον οποίο θα εκτελεστούν οι πράξεις της μεθόδου, επόμενο βήμα αποτελεί η κατασκευή μιας αρχιτεκτονικής σχεδίασης για τον παράλληλο υπολογισμό όλων των παραπάνω απαιτούμενων πράξεων, όπως εμφανίζεται στην Εικόνα 5. 2 παρακάτω.



Εικόνα 5. 2 UMP Calculate κύκλωμα

Η παραπάνω αρχιτεκτονική σχεδίαση αφορά την κατασκευή ενός κυκλώματος κατάλληλου για τον παράλληλο υπολογισμό τεσσάρων γινομένων ενός στοιχείου του πίνακα X. Όπως φαίνεται στην Εικόνα 5. 2 , το κύκλωμα με όνομα UMP Calculate λαμβάνει ως εισόδους ένα στοιχείο x ενός πίνακα υποκατάστασης X, καθώς και έναν πίνακα ρυθμού υποκατάστασης Ypok για να υπολογίσει τα 4 γινόμενα του στοιχείου x.

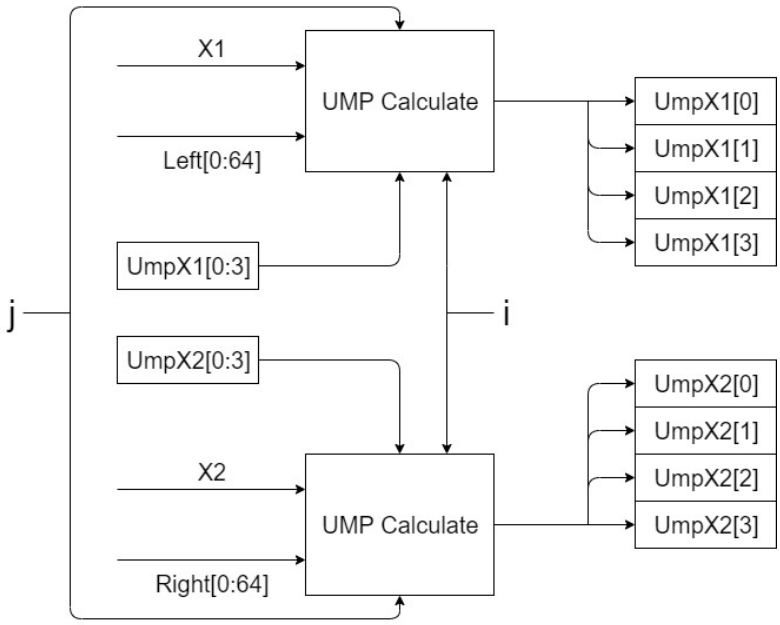
Προτού, ωστόσο, πραγματοποιηθεί η τελική κατασκευή του Ump Calculate κυκλώματος, αναγκαίο βήμα αποτελεί ο συλλογισμός για το πώς θα υπολογιστούν όχι μόνο οι 4, αλλά όλοι οι πολλαπλασιασμοί που απαιτούνται για την διεκπεραίωση της μεθόδου της μέγιστης πιθανοφάνειας, χρησιμοποιώντας μόνο μία σχεδίαση κυκλώματος Ump Calculate. Αναλυτικότερα, κάθε ένα στοιχείο από τα 16 του πίνακα X πολλαπλασιάζεται 4 φορές με 4 διαφορετικά στοιχεία του πίνακα Ypok. Χρειάζεται, λοιπόν, να υπολογιστούν 64 πολλαπλασιασμοί για το πέρας και των 16 στοιχείων του πίνακα X. Η προσωρινή σχεδίαση στην Εικόνα 5. 2, υπολογίζει όλα τα γινόμενα για ένα μόλις στοιχείο του πίνακα X. Συνεπώς, το Ump Calculate κύκλωμα πρέπει να εκτελεστεί 16 φορές, όσα, δηλαδή, είναι και τα στοιχεία του πίνακα X. Η τελική αρχιτεκτονική σχεδίαση του Ump Calculate κυκλώματος που παρουσιάζεται στην Εικόνα 5. 3, περιλαμβάνει όλες τις απαραίτητες παραμέτρους για το πέρας και των 64 πολλαπλασιασμών που απαιτούνται για τον υπολογισμό ενός πίνακα X.



Εικόνα 5. 3 Είσοδοι και έξοδοι UMP Calculate κυκλώματος

Η τελική αρχιτεκτονική σχεδίαση του UMP Calculate κυκλώματος, όπως φαίνεται στην Εικόνα 5. 3, έχει 3 παραμέτρους. Η πρώτη παράμετρος k, είναι η εσωτερική (τοπική) παράμετρος του Ump Calculate κυκλώματος. Χρησιμοποιείται για την εκτέλεση των τεσσάρων πολλαπλασιασμών που αντιστοιχεί σε ένα στοιχείο του πίνακα X. Η δεύτερη παράμετρος i, προσφέρει καθοδήγηση στο να υπολογιστούν οι πολλαπλασιασμοί μίας τετράδας στοιχείων του πίνακα X. Η ρόλος της τρίτης και τελευταίας παραμέτρου j, έγκειται στο να υπολογιστούν οι υπόλοιπες 3 τετράδες στοιχείων του πίνακα X. Με αυτόν τον τρόπο, το σύνολο των πολλαπλασιασμών γίνεται 64, όσοι είναι και οι πολλαπλασιασμοί που απαιτούνται για το πέρας των υπολογισμών ενός πίνακα X.

Μετά την κατασκευή του Ump Calculate κυκλώματος, σειρά έχει η χρήση του για το πέρας όλων των στοιχείων και των 2 πινάκων υποκατάστασης X1 και X2. Η παρακάτω αρχιτεκτονική σχεδίαση, Εικόνα 5. 4, παρουσιάζει τον τρόπο επίτευξης των πολλαπλασιασμών και των 2 πινάκων υποκατάστασης X1 και X2.



Εικόνα 5. 4 Παράλληλη εκτέλεση 2 UMP Calculate κυκλωμάτων για X1 και X2 πίνακες

Η Εικόνα 5. 4 περιλαμβάνει 2 πανομοιότυπα Ump Calculate κυκλώματα. Το ένα αφορά τον πίνακα X1 και το άλλο τον πίνακα X2. Τα 2 αυτά Ump Calculate κυκλώματα εκτελούνται παράλληλα με κοινές παραμέτρους τις i και j. Το κάθε κύκλωμα δέχεται ως εισόδους κάθε φορά ένα στοιχείο από τους πίνακες υποκατάστασης X1 ή X2, καθώς και έναν από τους πίνακες ρυθμού υποκατάστασης Ypok, Left ή Right . Πιο συγκεκριμένα, το πρώτο κύκλωμα, σύμφωνα με την Εικόνα 5. 4, δέχεται ένα στοιχείο του πίνακα X1 και τον πίνακα Left, ενώ το δεύτερο κύκλωμα δέχεται ένα στοιχείο του πίνακα X2 και τον πίνακα Right. Τέλος, το κάθε κύκλωμα λαμβάνει το δικό του πίνακα $UmpX_i$, διότι κάθε πολλαπλασιασμός είναι μοναδικός, είτε προέρχεται από τον πίνακα X1, είτε από τον πίνακα X2.

5.2.1.β Αρχιτεκτονική κυκλώματος υπολογισμού του Likelihood Calculate κυκλώματος της μεθόδου

Με τον τρόπο που έχει σχεδιαστεί το κύκλωμα μέχρι στιγμής, υπολογίζονται συνολικά 8 γινόμενα, 4 γινόμενα για ένα στοιχείο του πίνακα X1 και 4 γινόμενα για ένα στοιχείο του πίνακα X2. Ο αλγόριθμος, ωστόσο, πρέπει να υπολογίσει 32 γινόμενα συνολικά, ώστε να κατασκευαστεί ο πίνακας x1px2 της πρώτης τετράδας στοιχείων των 2 πινάκων υποκατάστασης X1 και X2. Για το λόγο αυτό, σχεδιάστηκε ο αλγόριθμος με τέτοιο τρόπο ώστε, να επεξεργάζεται τετράδες στοιχείων του πίνακα X (βλ. παράμετρος i, Εικόνα 5. 3).

Υπολογισμός του πίνακα Ump X

Έχοντας αναλύσει το απαραίτητο υπόβαθρο της μεθόδου, το επόμενο βήμα αφορά την πρόσθεση των 16 γινομένων μιας τετράδας στοιχείων του πίνακα X. Σύμφωνα με τον αλγόριθμο της μέγιστης πιθανοφάνειας, ο υπολογισμός του πίνακα αθροισμάτων των γινομένων μιας τετράδας στοιχείων πραγματοποιείται ως εξής:

$Temp_{i,k=0}[0]+ Temp_{i,k=0}[1]+ Temp_{i,k=0}[2]+ Temp_{i,k=0}[3] = UmpX[k]$, όπου i=[0:3] και k=0

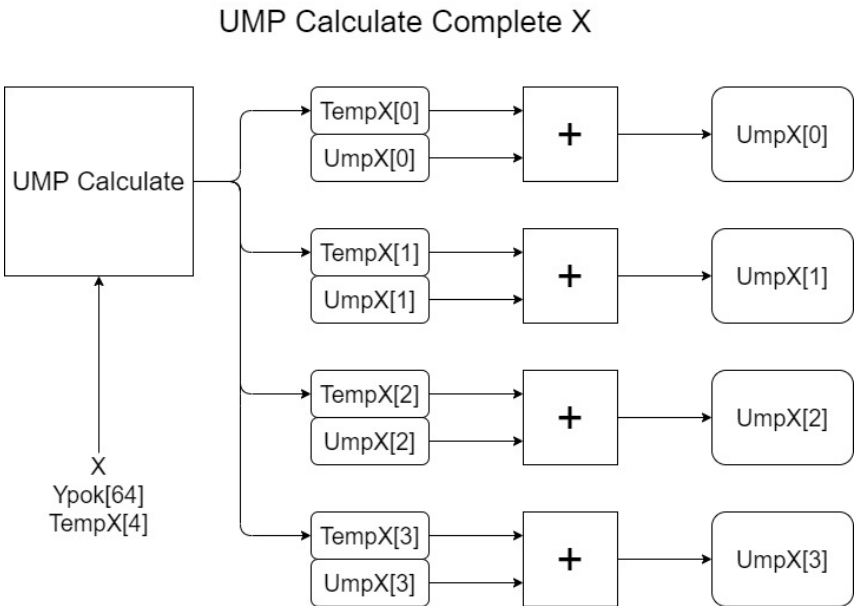
Το παραπάνω σύνολο πράξεων εκτελείται k φορές, όπου k=[0:3]

Ακολούθως, πραγματοποιώντας αναδιάταξη των πολλαπλασιασμών στον αλγόριθμο της μέγιστης πιθανοφάνειας, πρέπει να εφαρμοστεί μία διαφορετική προσέγγιση και στις προσθέσεις των γινομένων Temp[0:3] αναμεταξύ τους. Ο UmpX πίνακας που χρησιμοποιήθηκε και παραπάνω

συμβολίζει έναν τοπικό πίνακα τεσσάρων θέσεων που συγκρατεί τα αθροίσματα των γινομένων που υπολογίζονται από το UMP Calculate κύκλωμα. Έτσι, κάθε μια θέση του πίνακα Ump X περιέχει το άθροισμα όλων των απαραίτητων γινομένων που πλαισιώνουν την συγκεκριμένη θέση, για συγκεκριμένο i. Οι παρακάτω πράξεις παρουσιάζουν τον τρόπο επίτευξης της αναδιάταξης των προσθέσεων, σύμφωνα με τον τρόπο που μόλις εξηγήθηκε.

$$Temp_{i=0,k=0}[0] + UmpX[0] = UmpX[0]$$
$$Temp_{i=1,k=0}[1] + UmpX[1] = UmpX[1]$$
$$Temp_{i=2,k=0}[2] + UmpX[2] = UmpX[2]$$
$$Temp_{i=3,k=0}[3] + UmpX[3] = UmpX[3]$$

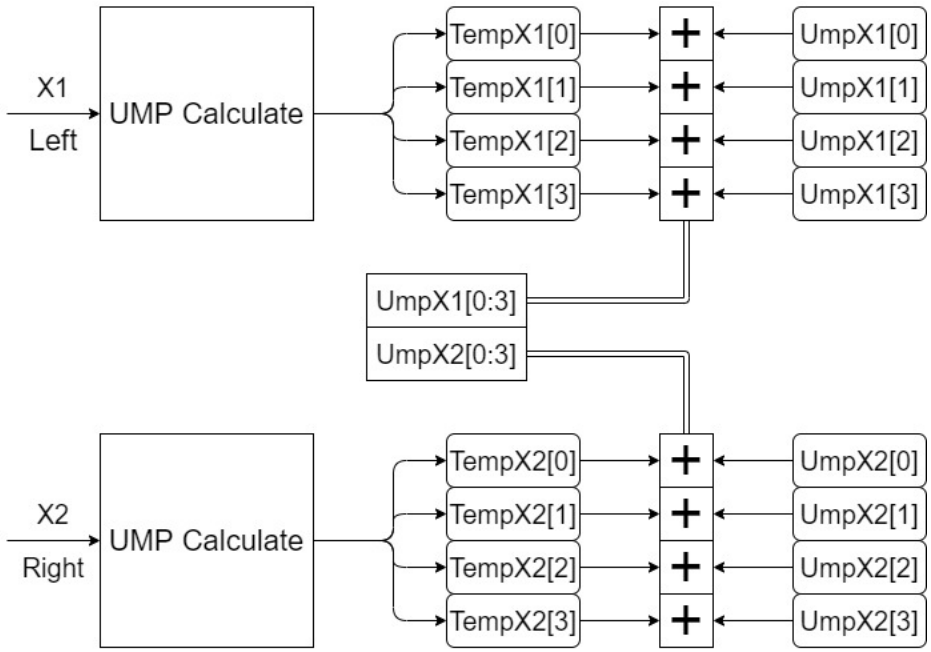
Όπως παρατηρείται στις παραπάνω πράξεις, κάθε ένα από τα 4 γινόμενα Temp, που υπολογίζονται για ένα στοιχείο του πίνακα X από το UMP Calculate κύκλωμα, προστίθεται με μία από τις θέσεις του πίνακα UmpX σε αντιστοιχία με την κάθε θέση του πίνακα. Αναλυτικότερα, το πρώτο γινόμενο Temp του πρώτου στοιχείου προστίθεται στην πρώτη θέση του πίνακα UmpX, το δεύτερο γινόμενο Temp στη δεύτερη θέση του πίνακα UmpX κ.ο.κ. Η αρχιτεκτονική σχεδίαση των παραπάνω προσθέσεων και ο υπολογισμός των γινομένων του Ump Calculate κυκλώματος φαίνεται στην Εικόνα 5. 5.



Εικόνα 5. 5 Αρχιτεκτονική UMP Calculate Complete X κυκλώματος

Σύμφωνα με την Εικόνα 5. 5, κάθε ένα από τα γινόμενα των στοιχείων μιας τετράδας του πίνακα X προστίθεται αθροιστικά σε αντιστοιχία μεταξύ του TempX και UmpX πίνακα. Στο παραπάνω κύκλωμα έχει δοθεί η ονομασία UMP Calculate Complete X.

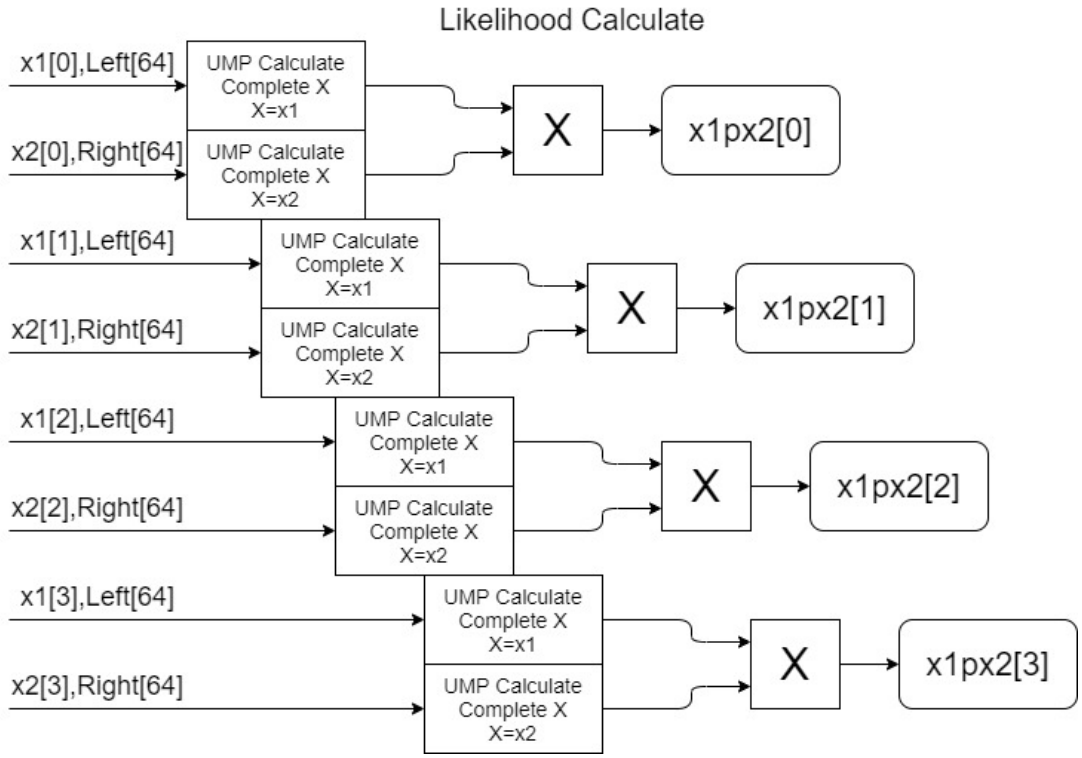
Σειρά έχει η εφαρμογή της UMP Calculate Complete X αρχιτεκτονικής για τον υπολογισμό των πράξεων και των 2 πινάκων υποκατάστασης X1 και X2. Η Εικόνα 5. 6 απεικονίζει την απαιτούμενη αρχιτεκτονική για τους μέχρι στιγμής υπολογισμούς.



Εικόνα 5. 6 Αρχιτεκτονική παράλληλης εκτέλεσης 2 UMP Calculate Complete X κυκλωμάτων για X1 και X2 πίνακες.

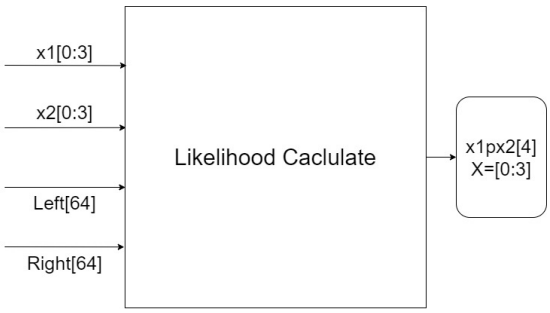
Όπως είναι ήδη γνωστό, θα χρησιμοποιηθούν παράλληλα 2 UMP Calculate κυκλώματα. Επομένως, οι προσθέσεις με εισόδους τις X1 και Left θα εκτελεστούν παράλληλα με τις προσθέσεις που έχουν εισόδους τις X2 και Right, όπως παρουσιάζεται στην Εικόνα 5. 6. Η συγκεκριμένη αρχιτεκτονική σχεδίαση υπολογίζει όλα τα απαραίτητα αθροίσματα γινομένων για τον υπολογισμό ενός στοιχείου από τους πίνακες X1 και X2 αντίστοιχα.

Στην συνέχεια, για να υπολογιστεί ο ζητούμενος πίνακας x1px2 για μία τετράδα στοιχείων των πινάκων X1 και X2, πρέπει να εκτελεστεί 4 φορές η αρχιτεκτονική σχεδίαση της Εικόνα 5. 6. Η αρχιτεκτονική, επομένως, πραγματοποιείται ως εξής:



Εικόνα 5. 7 Αρχιτεκτονική του Likelihood Calculate κυκλώματος της μέγιστης πιθανοφάνειας

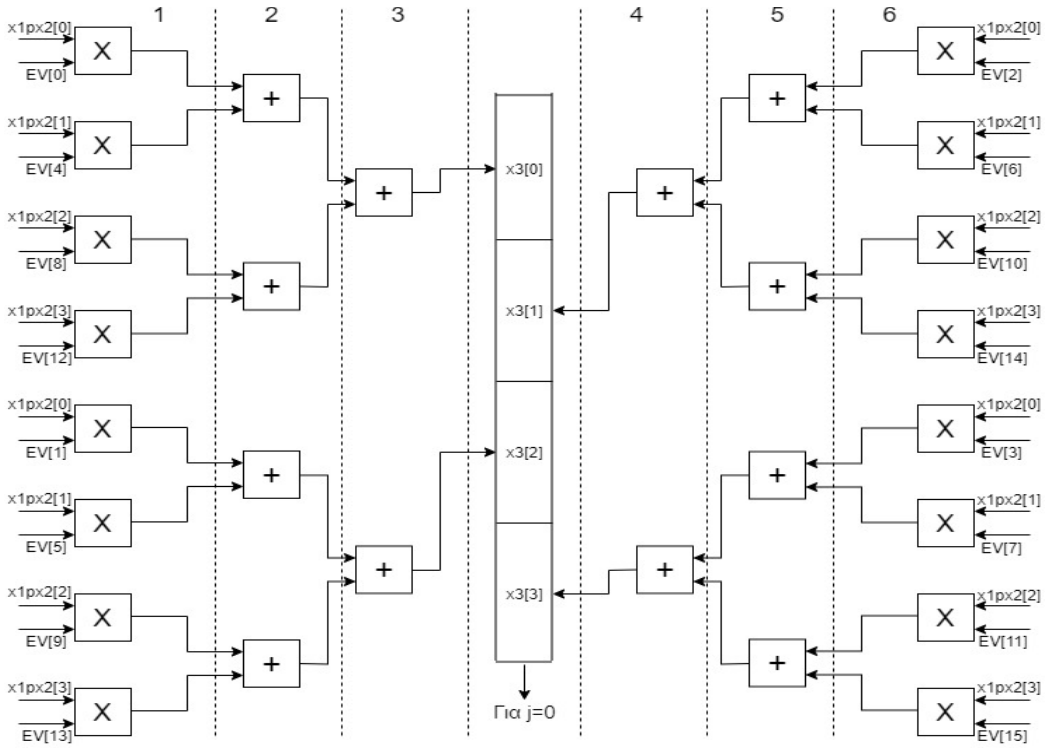
Η αρχιτεκτονική που εμφανίζεται στην Εικόνα 5. 7 ονομάζεται Likelihood Calculate. Όπως παρατηρείται, στην αρχιτεκτονική αυτή επεξεργάζονται κάθε φορά διαφορετικά στοιχεία από τις 2 τετράδες των X1 και X2 πινάκων, δηλαδή X1[0:3] και X2[0:3]. Στο τέλος της συγκεκριμένης αρχιτεκτονικής, μετά από κάθε παράλληλη εκτέλεση των 2 UMP Calculate Complete X κυκλωμάτων, πολλαπλασιάζονται οι 2 πίνακες UmpX για να υπολογιστεί η τελική θέση του πίνακα x1px2. Η Εικόνα 5. 8 παρουσιάζει τις εισόδους και εξόδους του Likelihood Calculate κυκλώματος.



Εικόνα 5. 8 Είσοδοι και έξοδοι του Likelihood Calculate κυκλώματος της μεθόδου της μέγιστης πιθανοφάνειας

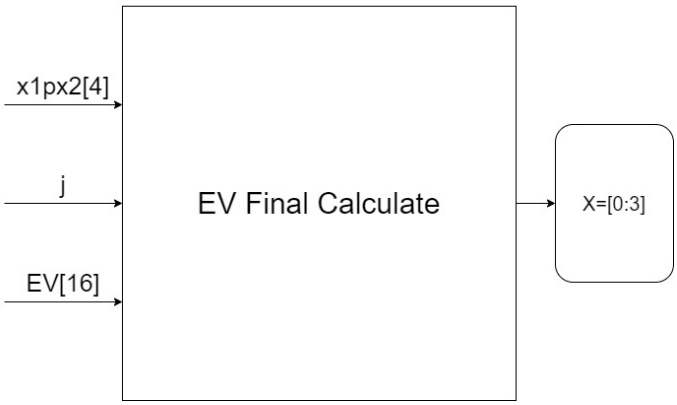
5.2.2 EV Final Calculate κύκλωμα της μεθόδου

Η Εικόνα 5. 1, περιλαμβάνει το EV Final Calculate κύκλωμα της μεθόδου που περιέχει 2 loops. Τα 2 αυτά loops πολλαπλασιάζουν τον πίνακα EV με τον πίνακα x1px2 που υπολογίζεται στο Likelihood Calculate. Ακόμη, στο EV Final Calculate κύκλωμα πραγματοποιείται αναδιάταξη των πράξεων παρόμοια με την αναδιάταξη του Likelihood Calculate κυκλώματος. Η αρχιτεκτονική που κατασκευάστηκε για το EV Final Calculate και περιλαμβάνει τις αναδιατάξεις των πράξεων απεικονίζεται στην Εικόνα 5. 9.



Εικόνα 5. 9 Αρχιτεκτονική σχεδίαση του του EV Final Calculate κυκλώματος της μεθόδου της μέγιστης πιθανοφάνειας

Στην Εικόνα 5. 9, οι διακεκομμένες κάθετες γραμμές οριοθετούν τις αριθμημένες περιοχές. Οι πράξεις των αριθμημένων περιοχών εκτελούνται παράλληλα σε ζευγάρια. Συγκεκριμένα, πρώτα εκτελείται παράλληλα το 1,6 ζευγάρι των περιοχών, έπειτα το 2,5 και τέλος το 3,4. Με τον παραπάνω τρόπο, πραγματοποιείται παραλληλισμός στο μέγιστο δυνατό βαθμό, εκτελώντας, ταυτόχρονα, όλες τις απαραίτητες πράξεις, σύμφωνα με την αναδιάταξη που εφαρμόζεται. Η Εικόνα 5. 10 περιλαμβάνει τις εισόδους και εξόδους του EV Final Calculate κυκλώματος της μεθόδου της μέγιστης πιθανοφάνειας.



Εικόνα 5. 10 Είσοδοι και έξοδοι του EV Final Calculate κυκλώματος της μεθόδου της μέγιστης πιθανοφάνειας

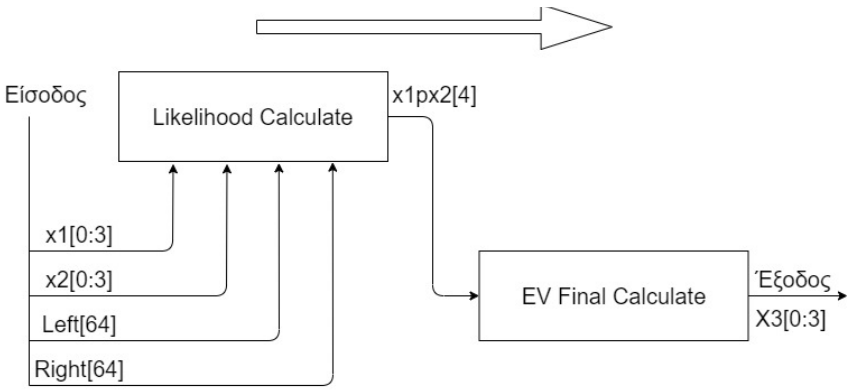
Όπως παρατηρείται στην Εικόνα 5. 10 το EV Final Calculate κύκλωμα της μεθόδου δέχεται ως εισόδους τον πίνακα x1px2 για συγκεκριμένο j εύρους [0:3], το ίδιο το j, καθώς και τον πίνακα ρυθμού συχνοτήτων των νουκλεοτιδίων EV. Μετά το πέρας της εκτέλεσης των αναδιαταγμένων πράξεων, υπολογίζεται η έξοδος που περιλαμβάνει τα 4 από τα 16 στοιχεία του ζητούμενου πίνακα X3.

5.3 Ολοκληρωμένη αρχιτεκτονική σχεδίαση της μεθόδου της μέγιστης πιθανοφάνειας.

Έχοντας αναλύσει διεξοδικά την αρχιτεκτονική σχεδίαση του Likelihood Calculate και του EV Final Calculate κυκλώματος, επόμενο βήμα αποτελεί η ένωσή τους, ώστε να υπολογιστεί ο ζητούμενος πίνακας X3.

Υπολογισμός μίας τετράδας στοιχείων του πίνακα X3.

Σύμφωνα με τα παραπάνω, το Likelihood Calculate υπολογίζει έναν από τους 4 πίνακες x1px2. Ακολούθως, το EV Final Calculate δέχεται ως είσοδο έναν από τους 4 πίνακες x1px2 του Likelihood Calculate και υπολογίζει μία τετράδα στοιχείων του ζητούμενου πίνακα X3. Η Εικόνα 5. 11 περιγράφει σχηματικά τα παραπάνω:

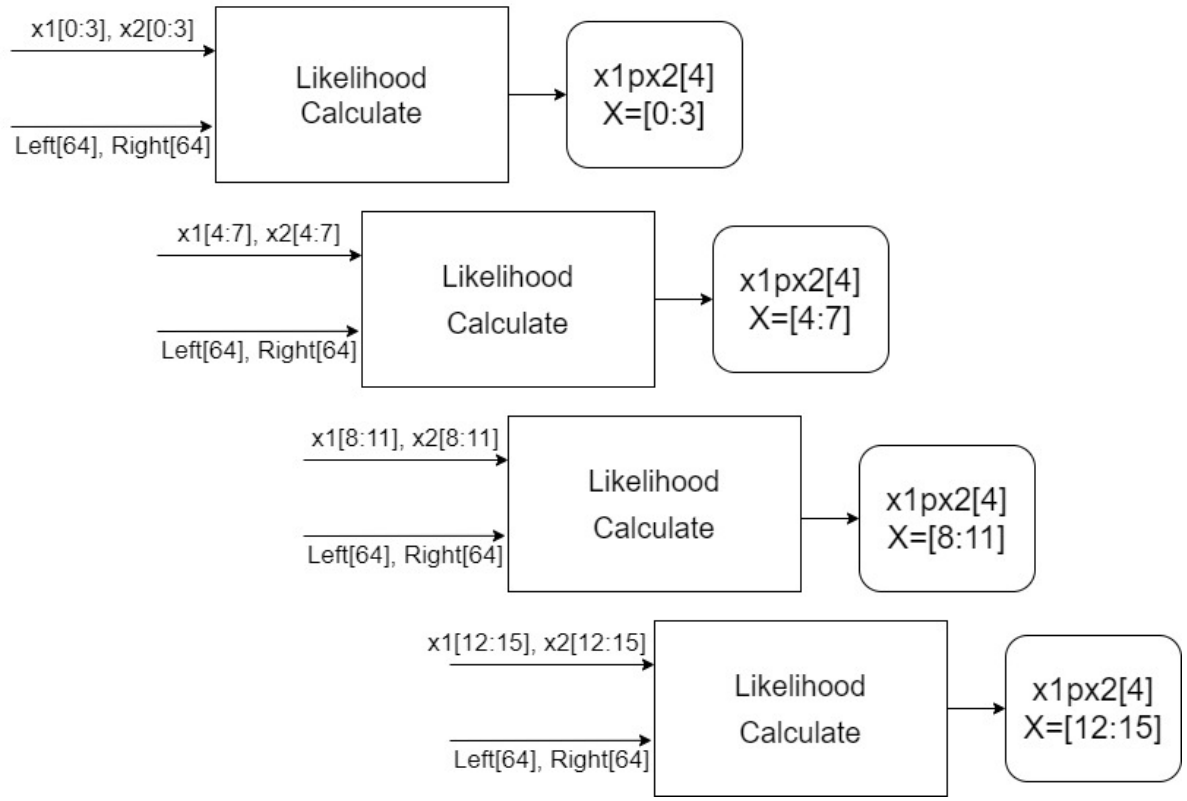


Εικόνα 5. 11 Εκτέλεση της μεθόδου της μέγιστης πιθανοφάνειας για μία τετράδα στοιχείων

Όπως έχει προαναφερθεί, είναι αδύνατο να εκτελεστεί το Likelihood Calculate και το EV Final Calculate κύκλωμα παράλληλα, καθώς για τον υπολογισμό του τελευταίου προ-απαιτείται ο υπολογισμός του πίνακα x1px2 που πραγματοποιείται στο Likelihood Calculate κύκλωμα. Στην Εικόνα 5. 11, η αρχιτεκτονική σχεδίαση υπολογίζει μόλις 4 στοιχεία του ζητούμενου πίνακα X3.

Υπολογισμός των τεσσάρων πινάκων x1px2.

Το πρώτο βήμα για τον υπολογισμό όλων των στοιχείων του ζητούμενου πίνακα X3 είναι να υπολογιστούν όλοι οι πίνακες x1px2. Για την επίτευξη αυτού, πρέπει να εκτελεστεί το Likelihood Calculate κύκλωμα 4 φορές με διαφορετικές τετράδες στοιχείων των πινάκων X1 και X2 αντίστοιχα. Η Εικόνα 5. 12 εξηγεί σχηματικά τα παραπάνω.

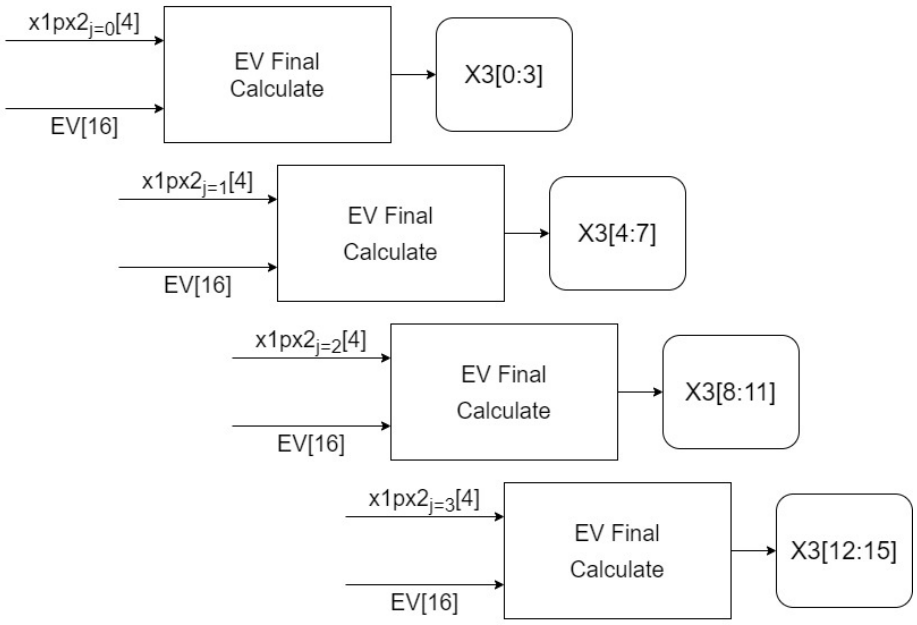


Εικόνα 5. 12 Εκτέλεση του Likelihood Calculate κυκλώματος της μεθόδου της μέγιστης πιθανοφάνειας για τις 4 τετράδες των στοιχείων των πινάκων X1 και X2.

Όπως παρατηρείται στην Εικόνα 5. 12, το κύκλωμα του Likelihood Calculate κυκλώματος δέχεται ως είσοδο 2 τετράδες στοιχείων. Η μία τετράδα προέρχεται από τον πίνακα X1 και η άλλη από τον πίνακα X2. Το κύκλωμα έχει σχεδιαστεί με τέτοιο τρόπο ώστε, κάθε φορά να εκτελούνται πράξεις από διαφορετικές τετράδες των πινάκων X1 και X2.

Υπολογισμός του ζητούμενου πίνακα X3.

Σύμφωνα με την Εικόνα 5. 12, υπολογίζονται και οι 4 πίνακες x1px2 που προ-απαιτούνται ως είσοδοι στο EV Final Calculate κύκλωμα. Κατά αυτόν τον τρόπο, η εκτέλεση του EV Final Calculate γίνεται δυνατή και οδηγεί στον υπολογισμό του πίνακα X3, όπως φαίνεται στην Εικόνα 5. 13.

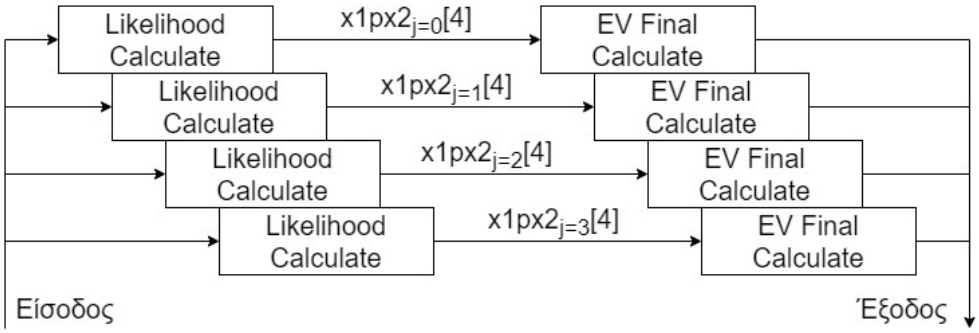


Εικόνα 5. 13 Εκτέλεση του EV Final Calculate κυκλώματος της μεθόδου της μέγιστης πιθανοφάνειας για τον υπολογισμό των τεσσάρων τετράδων των στοιχείων των πινάκων X3.

Όπως παρατηρείται στην Εικόνα 5. 13, η αρχιτεκτονική της εκτέλεσης του EV Final Calculate κυκλώματος της μεθόδου για τον υπολογισμό του πίνακα X3 είναι παρόμοια με την αρχιτεκτονική εκτέλεσης του Likelihood Calculate κυκλώματος, Εικόνα 5. 12. Αναλυτικότερα, κάθε φορά που το Likelihood Calculate υπολογίζει έναν πίνακα x1px2, για συγκεκριμένο j, το EV Final Calculate ξεκινά την εκτέλεση των υπολογισμών μιας τετράδας στοιχείων του πίνακα X3, για συγκεκριμένο j.

Σύνοψη της αρχιτεκτονικής σχεδίασης της μεθόδου της μέγιστης πιθανοφάνειας.

Τελικό βήμα για την ολοκλήρωση της αρχιτεκτονικής σχεδίασης της μεθόδου της μέγιστης πιθανοφάνειας αποτελεί η εκτέλεση των κυκλωμάτων Likelihood Calculate και EV Final Calculate για το πέρας όλων των στοιχείων του πίνακα X1 και X2 και τον τελικό υπολογισμό του ζητούμενου πίνακα X3. Η Εικόνα 5. 14 απεικονίζει την επίτευξη αυτού.



Εικόνα 5. 14 Εκτέλεση της μεθόδου της μέγιστης πιθανοφάνειας για όλα τα στοιχεία των πινάκων προς επεξεργασία.

5.4 Συμπεράσματα αρχιτεκτονικής σχεδίασης της μεθόδου της μέγιστης πιθανοφάνειας

Είναι σημαντικό να υπογραμμιστεί πως, η αρχιτεκτονική σχεδίαση που επιλέχθηκε προς κατασκευή αποσκοπεί στην ομαλή μετάβαση της θεωρίας στην πράξη. Συγκεκριμένα, ο υπολογισμός της μεθόδου της μέγιστης πιθανοφάνειας προσαρμόζεται στους περιορισμούς του hardware που αναλύονται στο Κεφάλαιο 7 παρακάτω.

Κεφάλαιο 6: Αρχιτεκτονική πλατφόρμας αποκεντρωμένων πόρων

Το έκτο κεφάλαιο παραθέτει την αρχιτεκτονική σχεδίαση της πλατφόρμας με FPGAs αποκεντρωμένων πόρων για την εφαρμογή του επιταχυντή που υλοποιείται στην παρούσα διπλωματική εργασία.

6.1 Επιλογή βάσης για την κατασκευή αρχιτεκτονικής σχεδίασης πλατφόρμας αποκεντρωμένων πόρων

Όπως αναλύθηκε στο κεφάλαιο 4, οι αποκεντρωμένοι πόροι μπορούν να αποβούν ωφέλιμοι στην επεξεργασία πολλών δεδομένων. Ιδιαίτερη σημασία δίνεται στο Project ReFiRe του Pissadakis et al. [24]. Αρχικά, Το ReFiRe Framework καταφέρνει να μειώσει το κόστος του χρόνου που απαιτείται για τον συγχρονισμό των aBlocks χρησιμοποιώντας εντολές σε μορφή ACI. Το cBlock στέλνει διεργασίες (Tasks) στα aBlocks χρησιμοποιώντας την βιβλιοθήκη υποστήριξης ACI. Κατ' αυτόν τον τρόπο, ενεργοποιεί την απομακρυσμένη χρήση ενός ή περισσοτέρων κόμβων επιταχυντών σπαταλώντας λιγότερο χρόνο για το συγχρονισμό των FPGAs αναμεταξύ τους. Η μείωση του κόστους του συγχρονισμού των αποκεντρωμένων FPGAs είναι σημαντική για τη λειτουργία της αρχιτεκτονικής σχεδίασης του κεφαλαίου αυτού, καθώς, όσο αυξάνονται σε αριθμό τα αποκεντρωμένα aBlocks, ο συγχρονισμός γίνεται ολοένα και πιο δύσκολος και χρονοβόρος δημιουργώντας πρόβλημα στην απόδοση του συστήματος [24].

6.2 Αρχιτεκτονική σχεδίαση FPGAs πλατφόρμας αποκεντρωμένων πόρων

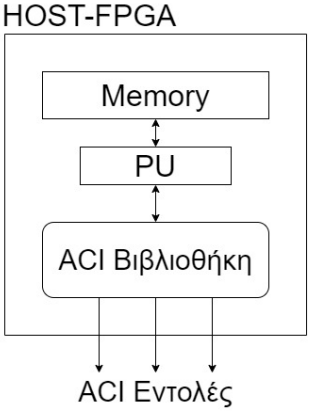
Η αρχιτεκτονική μιας πλατφόρμας που αποτελείται από FPGAs με αποκεντρωμένους πόρους κατασκευάζεται σε 3 στάδια.

- Στάδιο 1ο: Κατασκευή αρχιτεκτονικής του cBlock και του/των aBlock/-s.
- Στάδιο 2ο: Κατασκευή αρχιτεκτονικής της διάταξης της μνήμης (Memory Layout).
- Στάδιο 3ο: Επιλογή μοντέλου συγχρονισμού όλων των Blocks αναμεταξύ τους.

6.2.1 Στάδιο 1ο: Κατασκευή αρχιτεκτονικής του cBlock και του/των aBlock/-s.

Αρχιτεκτονική σχεδίαση cBlock

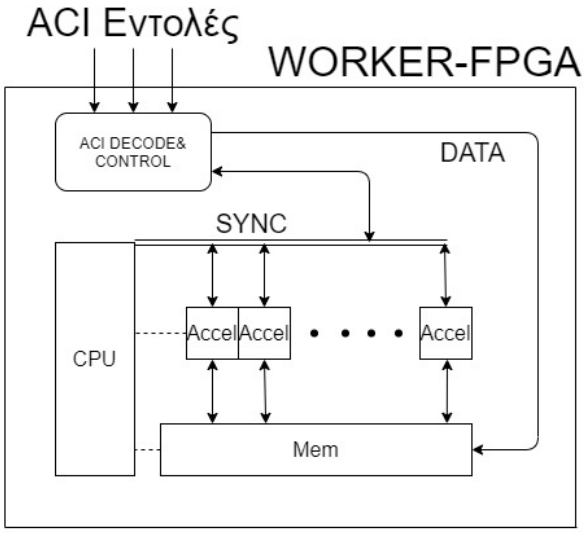
Το cBlock, όντας ο εγκέφαλος της πλατφόρμας των αποκεντρωμένων πόρων, χρησιμοποιεί την ίδια αρχιτεκτονική σχεδίαση του ReFiRe και ονομάζεται HOST. Η HOST-FPGA χρησιμοποιεί την ACI βιβλιοθήκη, μέσω του PU της, για τη μετατροπή των εντολών σε ACI εντολές, με σκοπό την απομακρυσμένη χρήση κόμβων-επιταχυντών του/των aBlock/-s. Η Εικόνα 6. 1 παραθέτει σχηματικά την αρχιτεκτονική σχεδίαση του cBlock.



Εικόνα 6. 1 Αρχιτεκτονική HOST-FPGA

Αρχιτεκτονική σχεδίαση aBlock

Το aBlock αποτελεί την FPGA με τα επιταχυντικά κυκλώματα και ονομάζεται WORKER. Η WORKER-FPGA αποκωδικοποιεί τις ACI εντολές και ελέγχει τη δρομολόγηση της αποκωδικοποιημένης εντολής στον/στους κόμβο/-ους του/των επιταχυντή/-ών. Η διαδικασία αποκωδικοποίησης, ο έλεγχος, ο συγχρονισμός του/των επιταχυντή/-ων και η σύνδεσή τους με τις δικές τους μνήμες, καθώς και τις περιφερειακές μνήμες της WORKER-FPGA, πραγματοποιούνται από το PU του κάθε WORKER ξεχωριστά, ανεξάρτητα και αποκεντρωμένα. Η Εικόνα 6. 2 εξηγεί τη σχεδίαση του WORKER σχηματικά.

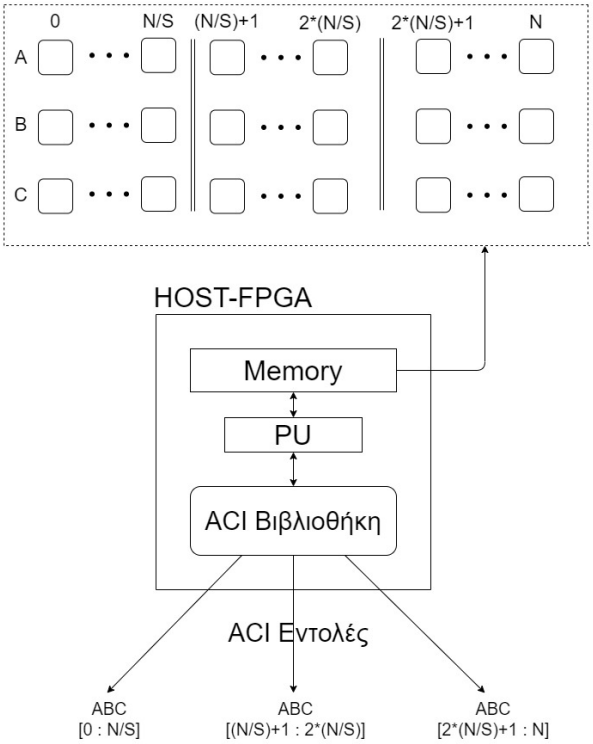


Εικόνα 6. 2 Αρχιτεκτονική WORKER-FPGA

6.2.2 Στάδιο 2ο: Κατασκευή αρχιτεκτονικής της διάταξης της μνήμης.

Η διάταξη της μνήμης αποτελεί βασική προϋπόθεση για την ομαλή λειτουργία και το σωστό διαμερισμό των δεδομένων σε περισσότερους από έναν επιταχυντή ή WORKERS ή και τα 2. Στην περίπτωση της χρήσης ενός μόλις επιταχυντή ανά WORKER και ταυτόχρονα 2 ή περισσότερους WORKERS η διάταξη της μνήμης για τους πίνακες X1, X2 και X3, μεγέθους N έκαστος, (βλέπε Κεφάλαιο 5) αλλάζει ως εξής:

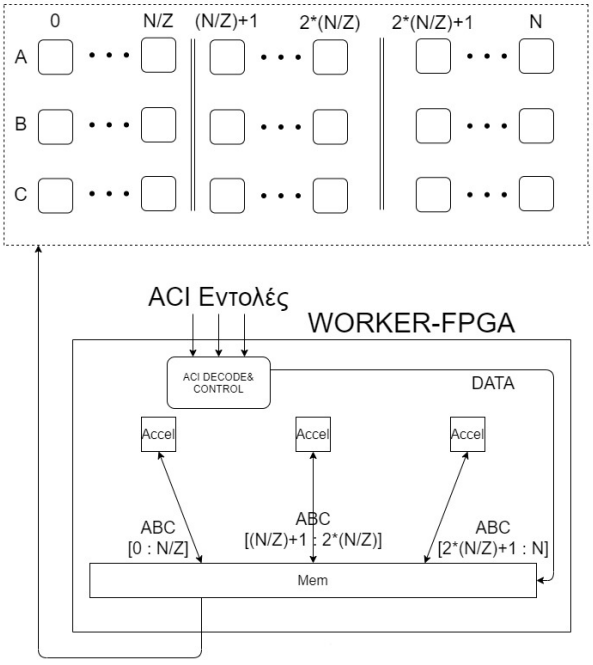
Έστω S ο αριθμός των διαθέσιμων WORKERS. Η διάταξη της μνήμης του HOST διασπάται από το PU του, N/S φορές, όπως φαίνεται στην Εικόνα 6. 3.



Εικόνα 6. 3 Διάταξη μνήμης HOST-FPGA για την περίπτωση των πολλών WORKERS με έναν επιταχυντή

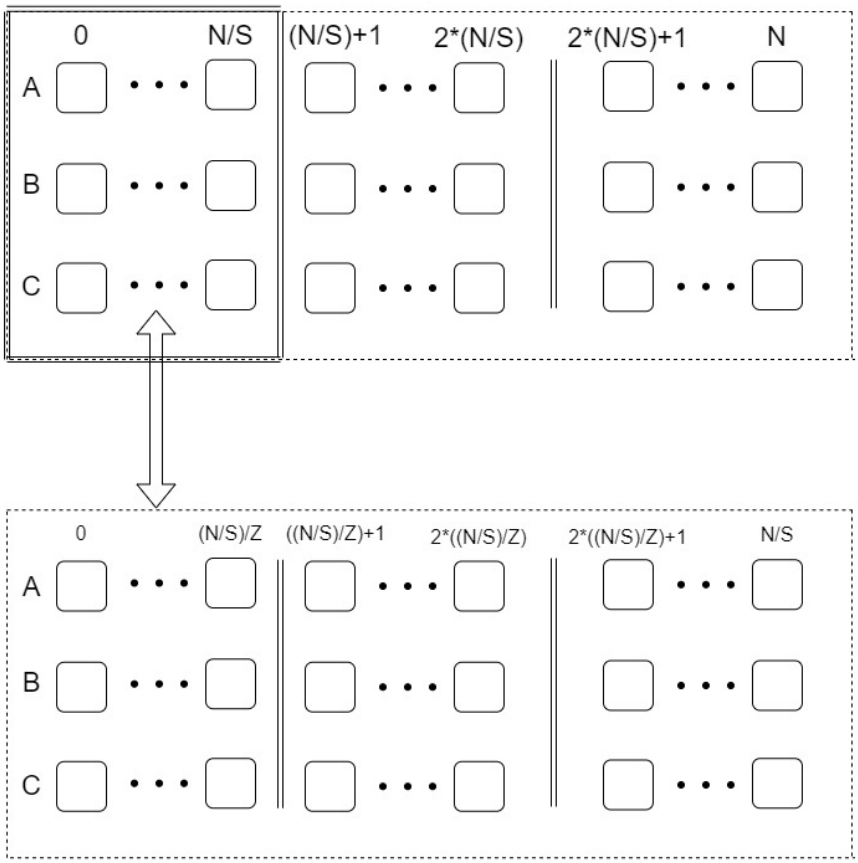
Όπως διακρίνεται στην Εικόνα 6. 3, το N διασπάται S φορές στη μνήμη του HOST και δρομολογούνται S σύνολα εντολών ACI σε όλους τους WORKERS. Σε αντίθετη περίπτωση, στην περίπτωση, δηλαδή, που γίνεται χρήση ενός WORKER 2 ή περισσότερων επιταχυντών, η διάταξη της μνήμης διαφοροποιείται ανάλογα, με τον εξής τρόπο:

Έστω Z ο αριθμός των διαθέσιμων επιταχυντών ενός WORKER. Η διάταξη της μνήμης του WORKER διασπάται από το PU του, N/Z φορές, όπως φαίνεται στην Εικόνα 6. 4.



Εικόνα 6. 4 Διάταξη μνήμης για την περίπτωση ενός WORKER με πολλούς επιταχυντές

Όπως διακρίνεται στην Εικόνα 6. 4 το N διασπάται Z φορές και δρομολογούνται τα Z φορές διασπασμένα δεδομένα στους επιταχυντές αντίστοιχα. Στην περίπτωση που συμβαίνουν και τα 2, δηλαδή χρήση 2 ή περισσοτέρων WORKERS και ταυτόχρονα 2 ή περισσοτέρων επιταχυντών για κάθε WORKER, η διάταξη της μνήμης αλλάζει και στον HOST και στον WORKER σύμφωνα με τις 2 παραπάνω περιπτώσεις. Η Εικόνα 6. 5 εξηγεί το τελικό σενάριο σχηματικά.

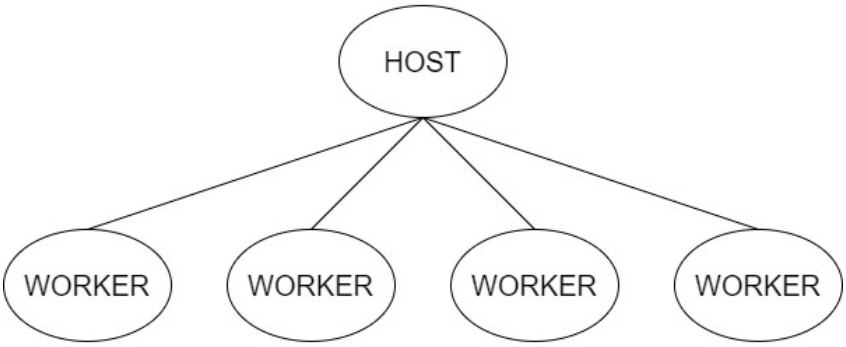


Εικόνα 6. 5 Διάταξη μνήμης για την περίπτωση των πολλών WORKERS με πολλούς επιταχυντές έκαστος

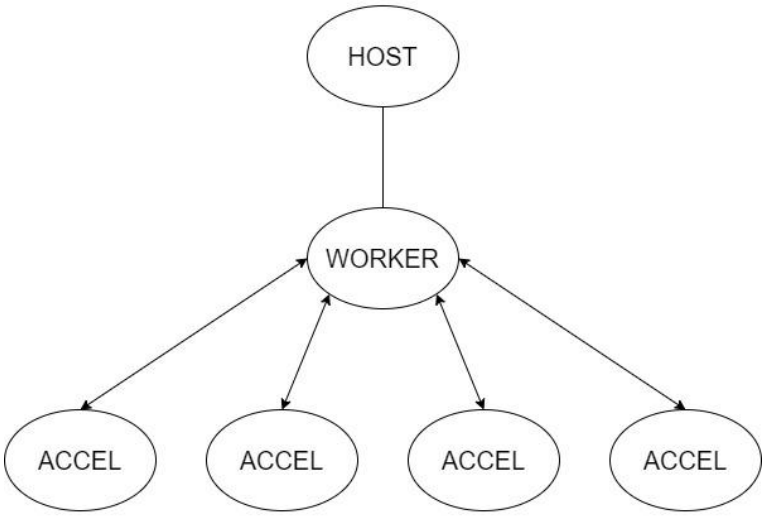
Παρατηρώντας την αρχιτεκτονική σχεδίαση της διάταξης της μνήμης του HOST και των WORKERS στην Εικόνα 6. 5, τα οφέλη των αποκεντρωμένων πόρων γίνονται πιο κατανοητά και διακριτά. Λόγω του ότι οι FPGAs είναι αποκεντρωμένες αναμεταξύ τους, δημιουργείται η ευχέρεια σε κάθε FPGA να ασχολείται μόνο με τη διαχείριση των δικών της πόρων, δίχως να απαιτείται κάποιος έλεγχος από κάποια άλλη FPGA. Με άλλα λόγια, το PU της κάθε FPGA είναι υπεύθυνο μόνο για την διάταξη και τη λειτουργία των δικών του αποκεντρωμένων πόρων, εστιάζοντας, με αυτόν τον τρόπο, την εργασία του σε τοπικό επίπεδο.

6.2.3 Στάδιο 3ο: Επιλογή μοντέλου συγχρονισμού των Blocks αναμεταξύ τους.

Ο συγχρονισμός αποτελεί την πιο σημαντική μεταβλητή της εξίσωσης κατασκευής μιας πλατφόρμας αποκεντρωμένων πόρων. Με τη μορφή εμποδίων (Barriers) πραγματοποιείται ο συγχρονισμός των Blocks αναμεταξύ τους. Τα Barriers ποικίλουν σε τρόπους που συγχρονίζουν μια πλατφόρμα με FPGAs αποκεντρωμένων πόρων. Για την περίπτωση πολλών WORKERS με έναν επιταχυντή έκαστος (Εικόνα 6. 6), αλλά και για την περίπτωση ενός WORKER με πολλούς επιταχυντές (Εικόνα 6. 7), χρησιμοποιήθηκε η πιο απλή μορφή Barrier για το συγχρονισμό και των 2 περιπτώσεων.



Εικόνα 6. 6 Συγχρονισμός σεναρίου πολλών WORKERS με έναν επιταχυντή έκαστος



Εικόνα 6. 7 Συγχρονισμός σεναρίου ενός WORKER με πολλούς επιταχυντές έκαστος

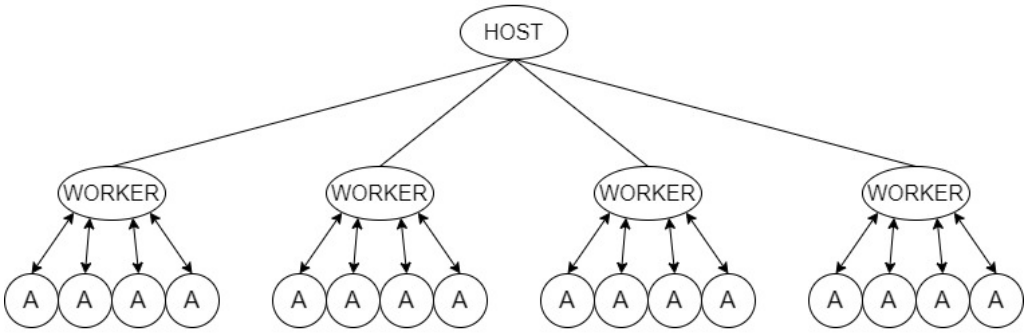
Η μορφή του Barrier των Εικόνα 6. 6 και Εικόνα 6. 7, ονομάζεται Centralized ή Counter Barrier και η λεπτομερή λειτουργία του γίνεται διακριτή από τον ψευδο-κώδικα της Εικόνα 6. 8

```
//C κοινόχρηστη μεταβλητή
//S για WORKERS, Z για επιταχυντές
C ← S ή Z

Mutex Lock    //Κλείδωμα διεργασίας
        //Μείωση κοινόχρηστου μετρητή C
        C = C – 1
Mutex Unlock //Ξεκλείδωμα διεργασίας

Αν C = 0 τότε
        //Εφτασαν όλοι οι WORKERS/επιταχυντές
        //Ξανά αρχικοποίηση μετρητή C
        C ← S ή Z
Διαφορετικά
        Όσο C≠0 τότε
                Spin
        Τέλος
Τέλος
```

Εικόνα 6. 8 Ψευδο-κώδικας ενός απλού Barrier

Παρατηρείται στην Εικόνα 6. 8 πως, ο αλγόριθμος ξεκινάει με την αρχικοποίηση μιας κοινόχρηστης μεταβλητής C σε S WORKERS ή σε Z επιταχυντές, ανάλογα με την παρούσα περίπτωση. Η C μεταβλητή είναι προσβάσιμη από όλους τους WORKERS ή επιταχυντές. Όταν ένας WORKER ή επιταχυντής ολοκληρώσει την εργασία του, -φτάσει, δηλαδή, στο Barrier-, τότε πραγματοποιείται: α) κλείδωμα διεργασίας (Mutex Lock), β) μείωση της κοινόχρηστης μεταβλητής C κατά 1, και γ) ξεκλείδωμα διεργασίας (Mutex Unlock). Έπειτα γίνεται έλεγχος για την ολοκλήρωση της διαδικασίας του Barrier παρατηρώντας την τιμή της μεταβλητής C. Αν η μεταβλητή C έχει την τιμή 0, τότε όλοι οι WORKERS ή επιταχυντές έχουν φτάσει στο Barrier, οπότε η διαδικασία ολοκληρώνεται και η μεταβλητή C αρχικοποιείται πάλι. Στην περίπτωση που η διαδικασία δεν έχει ολοκληρωθεί, (C≠0), τότε εκτελείται μία ατέρμονη επανάληψη, ένα Spin.

Στην περίπτωση που η πλατφόρμα αποκεντρωμένων πόρων αποτελείται από πολλούς WORKERS πολλών επιταχυντών έκαστος, ο συγχρονισμός λειτουργεί με μία πιο σύνθετη μορφή Barrier, όπως διακρίνεται σχηματικά στην Εικόνα 6. 9.



Εικόνα 6. 9 Συγχρονισμός σεναρίου πολλών WORKERS με πολλούς επιταχυντές έκαστος

Το Barrier της Εικόνα 6. 9 ονομάζεται Tree Barrier και η λειτουργία του γίνεται γνωστή με τον ψευδο-κώδικα της Εικόνα 6. 10.

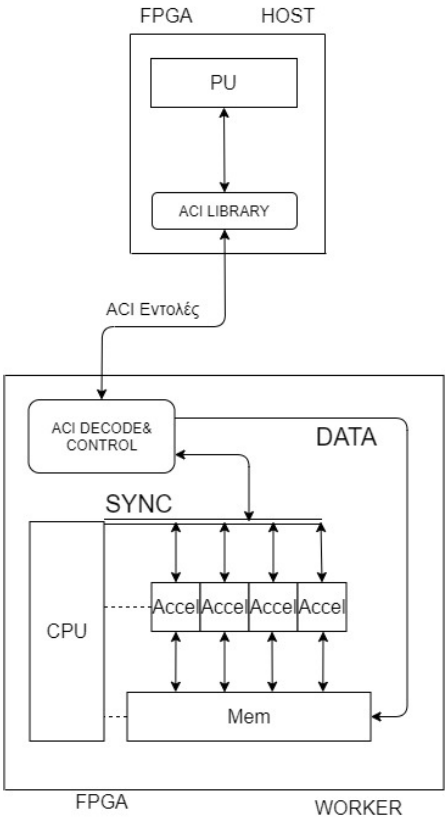
```
//C0 κοινόχρηστη μεταβλητή
C0 ← S

//C Τοπική μεταβλητή
C  ← 0

Όσο C0 ≠ 0 τότε
        Mutex Lock    //Κλείδωμα διεργασίας

        //Αύξηση τοπικού μετρητή C
        C = C + 1

        Αν C = Z τότε
                //Τελείωσαν όλοι οι επιταχυντές
                //Μείωση κοινόχρηστης μεταβλητής C0
                C0 = C0 – 1
        Τέλος
        Mutex Unlock //Ξεκλείδωμα διεργασίας

        Όσο C≠0 τότε
                Spin
        Τέλος
Τέλος
```

Εικόνα 6. 10 Ψευδο-κώδικας ενός Tree Barrier

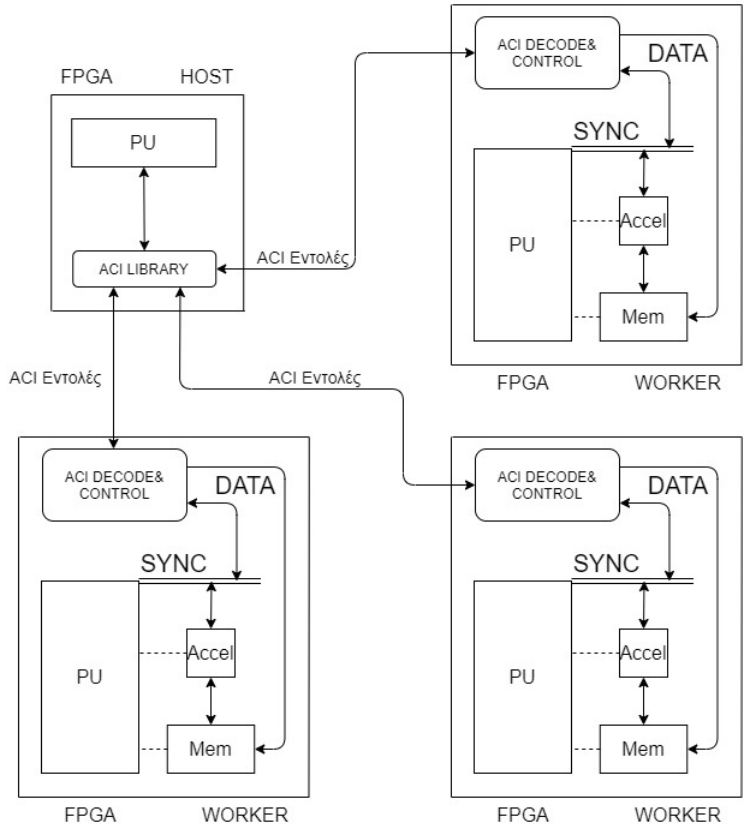
Όπως παρατηρείται στον ψευδο-κώδικα της Εικόνα 6. 10, γίνεται, αρχικά, αρχικοποίηση 2 μεταβλητών. Η μία μεταβλητή είναι η C0, η οποία είναι κοινόχρηστη και χρησιμοποιείται από τον HOST για το συγχρονισμό των WORKERS. Η άλλη μεταβλητή είναι η C, η οποία είναι τοπική μεταβλητή του κάθε WORKER και χρησιμοποιείται από τον κάθε WORKER ξεχωριστά για τον συγχρονισμό των επιταχυντών του. Η λειτουργία του συγχρονισμού κυμαίνεται σε 2 επίπεδα. Το πρώτο επίπεδο είναι πανομοιότυπο με το σενάριο των πολλών WORKERS ενός επιταχυντή (Εικόνα 6. 6). Μόλις ένας WORKER φέρει εις πέρας την εργασία που του ανατέθηκε από τον HOST, μειώνεται κατά 1 η C0 μεταβλητή, όπως διακρίνεται στην Εικόνα 6. 10. Όταν η C0 μεταβλητή πάρει την τιμή 0, δηλαδή, όταν όλοι οι WORKERS τελειώσουν την εκτέλεση των διεργασιών που τους ανατέθηκαν, ολοκληρώνεται ο συγχρονισμός από τον HOST. Το δεύτερο επίπεδο συγχρονισμού πραγματοποιείται στον κάθε WORKER ξεχωριστά. Ο τρόπος που λειτουργεί είναι ίδιος με το σενάριο του ενός WORKER με πολλούς επιταχυντές (Εικόνα 6. 7). Είναι μείζονος σημασίας να γίνει κατανοητό πως, οι WORKERS είναι αποκεντρωμένοι αναμεταξύ τους. Κάθε WORKER είναι υπεύθυνος για τον συγχρονισμό μόνο των δικών του επιταχυντών, δίχως να χρειάζεται κάποιον επιπλέον έλεγχο ή συγχρονισμό από τις υπόλοιπες FPGAs. Αυτός είναι και ο λόγος που η C μεταβλητή του ψευδο-κώδικα της Εικόνα 6. 10 είναι τοπική σε κάθε WORKER. Η C τοπική μεταβλητή αρχικοποιείται στο 0 και τη στιγμή που πάρει την τιμή Z η διαδικασία συγχρονισμού του WORKER ολοκληρώνεται. Όπως γίνεται διακριτό, τα 2 επίπεδα συγχρονισμού αλληλοεπιδρούν αναμεταξύ τους από κάτω προς τα πάνω σχηματίζοντας ένα δέντρο (Εικόνα 6. 9).

6.3 Ολοκληρωμένη αρχιτεκτονική σχεδίαση πλατφόρμων αποκεντρωμένων πόρων

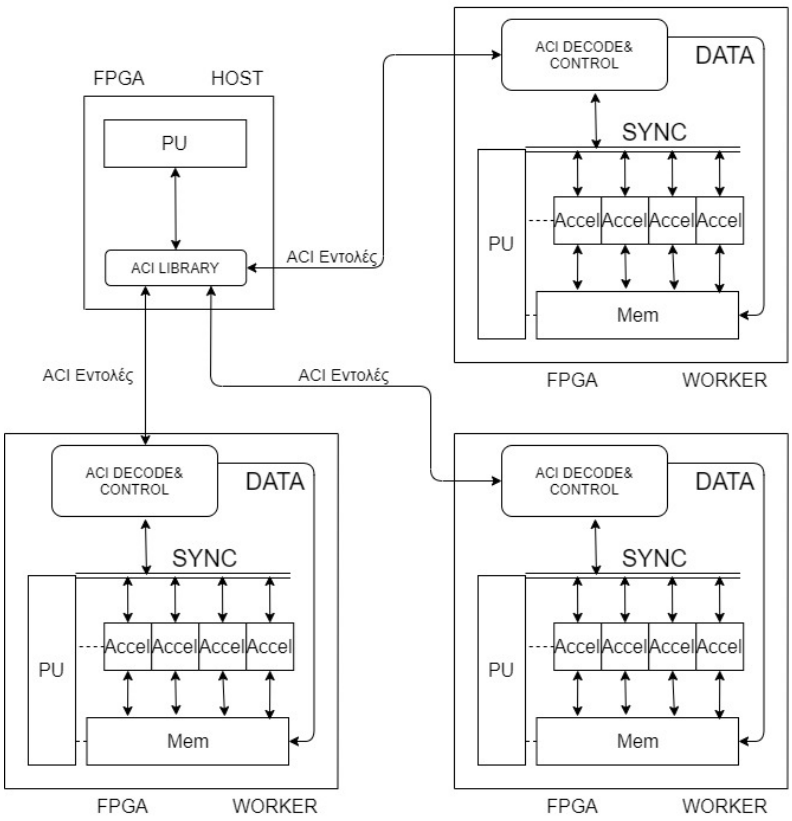
Σε αυτή την παράγραφο παρατίθενται παραδείγματα ολοκληρωμένων πλατφόρμων αποκεντρωμένων πόρων, σύμφωνα με τα 3 στάδια που αναλύθηκαν παραπάνω. Στα παρακάτω παραδείγματα οι WORKERS (S) είναι από [1:3] και οι επιταχυντές (Z) από [1:4]. Η διάταξη της μνήμης και ο συγχρονισμός των FPGAs ποικίλουν στα παραδείγματα αυτά, ανάλογα με τον αριθμό των WORKERS και των επιταχυντών, όπως αναλύθηκαν στα 3 στάδια παραπάνω.



Εικόνα 6. 11 Παράδειγμα μιας πλατφόρμας αποκεντρωμένων πόρων με έναν WORKER που περιέχει 4 επιταχυντές



Εικόνα 6. 12 Παράδειγμα μιας πλατφόρμας αποκεντρωμένων πόρων με 3 WORKERS του ενός επιταχυντή έκαστος



Εικόνα 6. 13 Παράδειγμα μιας πλατφόρμας αποκεντρωμένων πόρων με 3 WORKERS των τεσσάρων επιταχυντών έκαστος

Κεφάλαιο 7: Υλοποίηση

Το έβδομο κεφάλαιο εστιάζει στις διάφορες υλοποιήσεις του επιταχυντή του κεφαλαίου 5 που πραγματοποιήθηκαν, σύμφωνα με τους περιορισμούς του hardware στην πράξη.

7.1VIVADO-HLS

Το HLS είναι ένα εργαλείο που πραγματοποιεί εύκολα την κατασκευή hardware για συγκεκριμένες αρχιτεκτονικές, σύμφωνα με τις προτιμήσεις και τις ιδέες του χρήστη. Χρησιμοποιεί υψηλές γλώσσες προγραμματισμού για την περιγραφή υλικού, όπως η C, C++ και SystemC, δίχως να απαιτείται η ασχολία με χαμηλότερες και πιο δύσκολες γλώσσες περιγραφής υλικού, όπως είναι η VHDL ή η Verilog. Όλες οι γλώσσες περιγραφής υλικού κατασκευάζουν κυκλώματα. Η διαφορά των υψηλών και χαμηλών γλωσσών έγκειται στον τρόπο με τον οποίο επιτυγχάνεται η κατασκευή των κυκλωμάτων. Συγκεκριμένα, οι γλώσσες χαμηλού επιπέδου περιγραφής υλικού, όπως η VHDL, είναι πολύπλοκες και σε κάποιες περιπτώσεις χρονοβόρες (π.χ. Port Maps, Ανάθεση σημάτων, Δημιουργία και διαχείριση FSM κ.α.). Το HLS δίνει τη δυνατότητα μεταγλώττισης κώδικα σε γλώσσες χαμηλού επιπέδου περιγραφής υλικού χρησιμοποιώντας γλώσσες υψηλού επιπέδου περιγραφής υλικού, όπως είναι η C. Πρόκειται στην ουσία για ένα μεταγλωττιστή της εκάστοτε αρχιτεκτονικής. Λόγου χάριν, μπορεί η αρχιτεκτονική να γραφεί σε κώδικα C και το HLS να μετατρέψει τον κώδικα C σε κώδικα VHDL ή Verilog ή και τους 2 και έτσι ο χρήστης να αποφύγει την πολύπλοκη και χρονοβόρα διαδικασία των γλωσσών χαμηλού επιπέδου. Η διαδικασία μεταγλώττισης ονομάζεται Synthesis.

Το HLS, ωστόσο, δεν μπορεί να υλοποιήσει την επιθυμητή αρχιτεκτονική, εάν πρώτα δεν του υποδειχτούν οι προτιμήσεις της αρχιτεκτονικής σχεδίασης. Για το λόγο αυτό, το HLS διαθέτει Directives που συμβάλλουν στην επίτευξη της ιδανικής αρχιτεκτονικής, σύμφωνα με τις απαιτήσεις του χρήστη [3].

7.2Επιταχυντής-Accelerator

Χρησιμοποιώντας τα Directives του HLS, είναι εφικτό να βρεθούν μια ή περισσότερες αρχιτεκτονικές που επιταχύνουν το κύκλωμα προς υλοποίηση. Το τελικό επιθυμητό κύκλωμα, -υλοποιημένο με ένα ή περισσότερα Directives-, που τρέχει πιο γρήγορα από το απλό κύκλωμα ονομάζεται Επιταχυντής (Accelerator). Επιταχυντής χρησιμοποιείται και στη συγκεκριμένη διπλωματική εργασία, με σκοπό την ολοκλήρωση της διαδικασίας της μέγιστης πιθανοφάνειας στο λιγότερο δυνατό χρονικό διάστημα, δεδομένου ότι υπάρχουν διαθέσιμοι πεπερασμένοι υλικοί πόροι.

7.3 Directives που χρησιμοποιήθηκαν στην αρχιτεκτονική σχεδίαση του Κεφαλαίου 5

Στην αρχιτεκτονική σχεδίαση του Κεφαλαίου 5 χρησιμοποιήθηκαν 3 Directives. Το πρώτο και κύριο Directive που εφαρμόστηκε είναι το Pipeline Directive. Το δεύτερο Directive που υλοποιήθηκε είναι το Array Partition Directive και το τρίτο το Allocation Directive.

7.3.1 Pipeline Directive

Το Pipeline Directive αποτελεί ένα από τα πιο σημαντικά Directives του Vivado HLS. Το συγκεκριμένο Directive δίνει την δυνατότητα στο χρήστη να εκτελέσει ένα κύκλωμα χρησιμοποιώντας διοχέτευση (Pipeline). Το Pipeline μετατρέπει ένα κύκλωμα με τέτοιο τρόπο ώστε, πριν ολοκληρωθεί η εκτέλεση του κυκλώματος να υπάρχει η δυνατότητα να εκτελεστούν άλλα δεδομένα. Με άλλα λόγια, απορροφά τα δεδομένα που δίνονται στο κύκλωμα, δίχως να χρειάζεται η αποθήκευση των δεδομένων εισόδου πριν και μετά την είσοδο των δεδομένων στο Pipelined κύκλωμα. Η παράμετρος που ελέγχει κάθε πότε και πόσο συχνά εισέρχονται καινούργια δεδομένα προς επεξεργασία στο Pipelined κύκλωμα ονομάζεται Interval. Η παράμετρος Interval είναι ιδιαίτερα σημαντική, καθώς η αρχιτεκτονική του Κεφαλαίου 5 δε μπορεί να εκτελεστεί δίχως τη σωστή τιμή της παραμέτρου Interval του Pipeline Directive.

7.3.2 Array Partition Directive

Ο ρόλος του Array Partition Directive έγκειται στο να διασπά έναν πίνακα σε επιμέρους υπο-πίνακες (Array Parts). Όμοια με τη φιλοσοφία του Pipeline Directive όπου υπάρχει η παράμετρος Interval, στο Array Partition Directive υπάρχει η παράμετρος factor που ελέγχει πόσες φορές θα διασπαστεί ένας πίνακας. Επιπρόσθετα, στο Array Partition Directive υπάρχουν 3 διαφορετικοί τρόποι για να διασπαστεί ένας πίνακας.

1ος τρόπος διάσπασης: Block

Ο πρώτος τρόπος διάσπασης Block, διασπά έναν πίνακα σε factor υπο-πίνακες που δε μεταβάλλουν τη σειρά του πίνακα. Για παράδειγμα, για τη διάσπαση ενός πίνακα Α 1000 θέσεων με factor 4, η διαδικασία είναι η ακόλουθη:

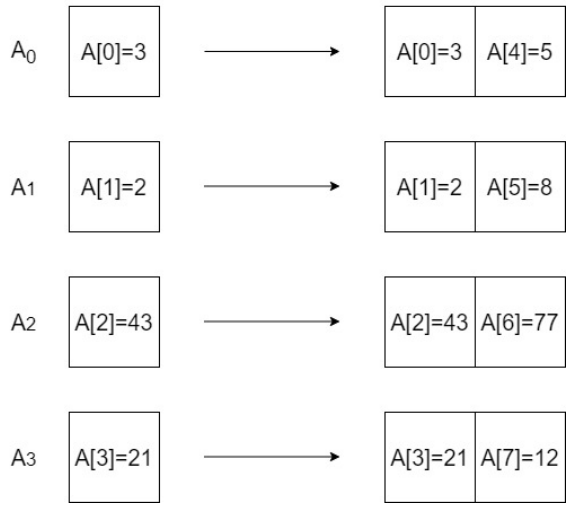
A_0[0:249] A_1[250:499] A_2[500:749] A_3[750:999]

Όπως παρατηρείται, ο πίνακας Α περιλαμβάνει 1000 θέσεις τις οποίες η παράμετρος factor διασπά σε υπο-πίνακες 250 θέσεων ο καθένας. Με αφετηρία την πρώτη θέση του πίνακα Α[0], το Block Array Partition Directive απαριθμεί σειριακά 250 θέσεις για τον πρώτο υπο-πίνακα, 250 θέσεις για το δεύτερο υπο-πίνακα, κ.ο.κ.

2ος τρόπος διάσπασης: Cyclic

Ο Cyclic διασπά έναν πίνακα Α σε factor υπο-πίνακες, όμοια με τον τρόπο διάσπασης Block. Η διαφορά έγκειται στο ότι, στον Cyclic οι υπο-πίνακες δεν περιέχουν τα στοιχεία του πίνακα Α σειριακά. Τα στοιχεία των υπο-πινάκων αποθηκεύονται με τον εξής τρόπο:

Έστω A[7]={3,2,43,21,5,8,77,12}, Cyclic με Factor=4



Εικόνα 7. 1 Παράδειγμα διάσπασης πίνακα με Cyclic Array Partition και factor 4

Όπως παρατηρείται στο παράδειγμα της Εικόνα 7. 1, τα δεδομένα αποθηκεύονται στους $A_{0:3}$ υπο-πίνακες κυκλικά. Με τον τρόπο αυτό, επιτυγχάνεται μια πιο γρήγορη πρόσβαση στα στοιχεία του πίνακα Α, όταν απαιτούνται στοιχεία στη σειρά, για παράδειγμα τα στοιχεία από τη θέση 200 έως τη θέση 300.

3ος τρόπος διάσπασης: Complete

Ο τρίτος τρόπος διάσπασης πινάκων είναι το Complete Array Partition. Στο συγκεκριμένο τρόπο διάσπασης δεν υπάρχει παράμετρος factor. Συγκεκριμένα, το Complete διασπά έναν πίνακα

τόσες φορές όσες είναι και το μέγεθός του, κατασκευάζοντας μεμονωμένους Registers για κάθε στοιχείο του πίνακα A ξεχωριστά. Σύμφωνα με τα παραπάνω, ο Complete τρόπος είναι ο πιο γρήγορος και αποδοτικός από τους 3 τρόπους διάσπασης, καθώς δίνει τη δυνατότητα να διαβαστεί οποιοσδήποτε πίνακας, ανεξαρτήτου μεγέθους, σε ένα μόλις χρόνο. Παρά το πλεονέκτημα του Complete Array Partition έναντι στους άλλους 2 τρόπους, ο συγκεκριμένος τρόπος διάσπασης είναι ακριβός σε υλικούς πόρους για την κατασκευή του. Οι υλικοί πόροι Flip Flops και Look Up Tables που απαιτούνται για τους Registers και τη λογική σχεδίαση των μεμονωμένων κελιών ενός πίνακα έχουν μεγάλο κόστος, με αποτέλεσμα ο συγκεκριμένος τρόπος να αποφεύγεται για πίνακες μεγάλου μεγέθους.

Λαμβάνοντας υπόψη τα παραπάνω, για την αρχιτεκτονική σχεδίαση της παρούσας διπλωματικής επιλέχθηκε ο τρόπος διάσπασης Complete Array Partition. Αυτό διότι, όλοι οι πίνακες, εκτός του X1, X2 και X3, είναι μικρού μεγέθους. Έτσι, εφαρμόστηκε Complete Array Partition στους πίνακες Left, Right και EV, καθώς επίσης και σε όλους τους απαραίτητους προσωρινούς (Temporary) πίνακες που κατασκευάστηκαν μέσα στον επιταχυντή για την εφαρμογή της αρχιτεκτονικής σχεδίασης του κεφαλαίου 5.

7.3.3 Allocation Directive

Το τρίτο Directive που εφαρμόστηκε είναι το Allocation Directive. Όταν σε ένα κύκλωμα εφαρμόζεται το Pipeline Directive το κύκλωμα κατασκευάζεται με τέτοιο τρόπο ώστε, σε κάθε interval κύκλο να εισάγονται νέα δεδομένα για εκτέλεση. Ωστόσο, όπως αναλύθηκε στο Κεφάλαιο 5, πρέπει, κάθε φορά, να εκτελούνται παράλληλα 2 UMP Calculate κυκλώματα. Για να εκτελεστεί ένα συγκεκριμένο κομμάτι του Pipelined κυκλώματος παράλληλα, πρέπει να γίνει χρήση του Allocation Directive. Το συγκεκριμένο Directive περιλαμβάνει μια παράμετρο που ονομάζεται limit. Αυτή η παράμετρος καθορίζει πόσες φορές εκτελείται παράλληλα ένα κύκλωμα. Στην περίπτωση της αρχιτεκτονικής σχεδίασης του Κεφαλαίου 5, πρέπει να εκτελεστεί παράλληλα 2 φορές το UMP Calculate κύκλωμα, οπότε η τιμή της παραμέτρου limit για το UMP Calculate κύκλωμα ανέρχεται σε 2.

7.4 Περιορισμοί στην αρχιτεκτονική του επιταχυντή

7.4.1 Περιορισμοί των υλικών πόρων των FPGAs

Όπως ειπώθηκε στο Κεφάλαιο 5, μπορούσε να κατασκευαστεί ένα κύκλωμα που εκτελεί τα πάντα παράλληλα. Για τις ανάγκες της πλήρης παράλληλης εκτέλεσης της μεθόδου της μέγιστης πιθανοφάνειας απαιτούνται πολλοί υλικοί πόροι. Οι FPGAs, ωστόσο, δε διαθέτουν άπειρους υλικούς πόρους.

7.4.2 Περιορισμοί της περιφερειακής μνήμης

Οι περιορισμοί της μνήμης διακρίνονται σε 2 διαστάσεις: α) το εύρος ζώνης (Bandwidth) της μνήμης και β) το πλάτος των θυρών της μνήμης (Memory Ports).

7.4.2.α Memory Bandwidth

Τα δεδομένα διαβάζονται από τη μνήμη και αποθηκεύονται στη μνήμη. Αναλυτικότερα, ο επιταχυντής δέχεται δεδομένα, αφού διαβαστούν από τη μνήμη. Ακόμη, ο επιταχυντής στέλνει δεδομένα στη μνήμη για αποθήκευση, μετά το πέρας των πράξεων που έχει κατασκευαστεί να εκτελεί. Για να υλοποιηθεί η πλήρης παράλληλη αρχιτεκτονική της παραγράφου 7.4.1, προϋποτίθεται ότι, έχουν διαβαστεί (read) όλοι οι πίνακες από τη μνήμη στον πρώτο μόλις κύκλο, προτού εκτελεστούν οι πράξεις της μεθόδου. Για να πραγματοποιηθούν περισσότερα από 100 reads σε έναν κύκλο, απαιτείται μια μνήμη με πολύ μεγάλο Bandwidth. Πιο συγκεκριμένα, για να διαβαστούν, λόγου χάριν, 100 διπλής ακρίβειας (double) τιμές, ταυτόχρονα, σε χρόνο ενός κύκλου 10ns, απαιτούνται 74,5

Gbytes/sec Bandwidth μνήμης. Ωστόσο, κάθε μνήμη έχει συγκεκριμένες προδιαγραφές από τον κατασκευαστή της. Συνεπώς, το επόμενο βήμα είναι να περιοριστούν οι απαιτήσεις του χρήστη, έτσι ώστε το ζητούμενο Bandwidth να μπορέσει να ανταποκριθεί σε αυτές. Για να επιτευχθεί αυτό, πρέπει να μειωθούν τα reads των πινάκων τόσο ώστε, να είναι όσο πιο κοντά γίνεται στο Bandwidth που δίνει ο κατασκευαστής. Με άλλα λόγια, παρ' ότι μπορεί να κατασκευαστεί μια αρκετά πιο γρήγορη αρχιτεκτονική σχεδίαση από αυτή του Κεφαλαίου 5, το Bandwidth της μνήμης περιορίζει αυτήν την προσέγγιση. Κάθε FPGA περιέχει ένα πεπερασμένο αριθμό από Memory Ports που εργάζονται ταυτόχρονα, για read και write, από και προς τη μνήμη, αντίστοιχα.

7.4.2.β Πλάτος των Memory Ports

Στην παράγραφο 7.4.2.α αναλύθηκε ο περιορισμός του Bandwidth της μνήμης που έχει ως αποτέλεσμα τη διαθεσιμότητα πεπερασμένων ενεργών Memory Ports ταυτόχρονα. Ωστόσο, υπάρχει ένας ακόμη περιορισμός που αφορά το πλάτος των Memory Ports. Πιο συγκεκριμένα, το πόση πληροφορία μπορεί να διαβάσει ή να γράψει ένα Memory Port από και προς την μνήμη σε έναν μόλις κύκλο ρολογιού. Το πλάτος των Memory Ports αποτελεί έναν από τους σημαντικότερους περιοριστικούς παράγοντες της αρχιτεκτονικής σχεδίασης του επιταχυντή της παρούσας διπλωματικής εργασίας. Η σημασία του πλάτους των Memory Ports γίνεται διακριτή με τα ακόλουθα σενάρια:

Είναι γνωστό πως, η συνάρτηση της μέγιστης πιθανοφάνειας περιέχει 3 διανύσματα μεγέθους N*16 έκαστος. Κάθε ένα διάνυσμα, δηλαδή, αποτελείται από έναν πίνακα 16 στοιχείων διπλής ακρίβειας (double). Στην περίπτωση που το πλάτος των Memory Ports ισούται με το μέγεθος ενός αριθμού double, -είναι, δηλαδή, 64bits-, τότε ο επιταχυντής μπορεί να διαβάσει από τη μνήμη, μέσω των Memory Ports, ένα double στοιχείο από κάθε Memory Port έκαστος. Ωστόσο, κάθε ένα διάνυσμα δεν αποτελείται από ένα double στοιχείο, αλλά από 16. Αυτό σημαίνει πως, ο επιταχυντής είναι ικανός να έχει στη διάθεσή του ένα ολόκληρο διάνυσμα, με ελάχιστο χρόνο τους 16 κύκλους. Επομένως, δεχόμενος καινούργιο διάνυσμα σε κάθε 16 κύκλους, ο επιταχυντής περιορίζεται σε μία pipelined αρχιτεκτονική σχεδίαση με ελάχιστο interval το 16. Σε ένα διαφορετικό σενάριο με μεγαλύτερο πλάτος των Memory Ports, η κατάσταση διαφέρει αρκετά. Αναλυτικότερα, στην ιδανική περίπτωση που το πλάτος έχει μέγεθος 1024bits (όσο είναι και το μέγεθος ενός διανύσματος), για παράδειγμα, τότε ο επιταχυντής μπορεί να έχει διαθέσιμο όλο το διάνυσμα σε έναν μόλις κύκλο. Σε ένα τέτοιο ιδανικό σενάριο μπορεί να εφαρμοστεί στον επιταχυντή μια pipelined αρχιτεκτονική σχεδίαση με interval 1.

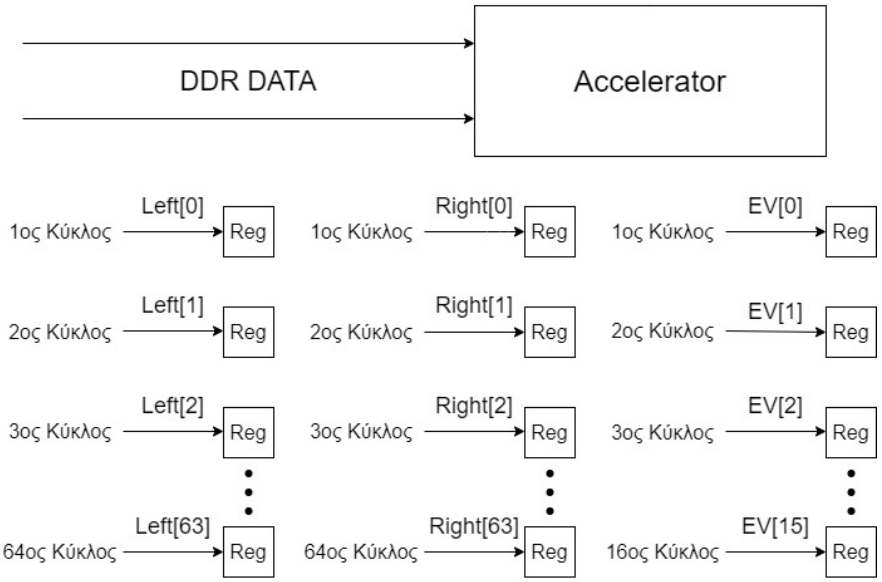
Κατ' αυτόν τον τρόπο κατασκευάστηκε το βέλτιστα αργό κύκλωμα, αυξάνοντας το interval τόσο ώστε, να ανταποκρίνεται στις προδιαγραφές της μνήμης. Στην επόμενη παράγραφο 7.5 αναλύονται διεξοδικά τα προβλήματα του Bandwidth της μνήμης και του πλάτους των Memory Ports, καθώς και ο λόγος που επιλέχθηκε η συγκεκριμένη αρχιτεκτονική σχεδίαση του κεφαλαίου 5.

7.5 Σχεδίαση αρχιτεκτονικής για το πέρας όλων των δεδομένων ανεξαρτήτου μεγέθους.

Μέχρι τώρα, αναλύθηκε εκτενώς η βέλτιστη αρχιτεκτονική, για N=1. Στόχος αποτελεί, ωστόσο, η διεκπεραίωση της μεθόδου της μέγιστης πιθανοφάνειας για μέγεθος N. Αυτό πρακτικά σημαίνει ότι, οι πίνακες x1,x2 και x3 είναι μεγέθους N*16 και όχι 16.

7.5.1 Σταθεροί πίνακες Left,Right και EV με σταθερό μέγεθος

Οι πίνακες left, right και EV παραμένουν σταθεροί για όλα τα N. Για το λόγο αυτό, η μεταφορά τους στον επιταχυντή πραγματοποιείται μια μόλις φορά. Συνεπώς, δεν απαιτείται ούτε υψηλός αριθμός ενεργών Memory Ports, ούτε υψηλό μέγεθος πλάτους αυτωνών. Η αρχιτεκτονική που εφαρμόστηκε για τους 3 σταθερούς πίνακες Left, Right και EV περιλαμβάνεται στην Εικόνα 7. 2.



Εικόνα 7. 2 Αρχιτεκτονική σχεδίαση πριν την έναρξη της εκτέλεσης της μεθόδου της μέγιστης πιθανοφάνειας

Όπως φαίνεται στην Εικόνα 7. 2, προτού ξεκινήσει ο υπολογισμός όλων των N, ο επιταχυντής δέχεται ως είσοδο από τη μνήμη τους σταθερούς και απαράλλακτους, μέχρι το πέρας της μεθόδου, πίνακες left, right και EV. Οι τιμές που διαβάζονται από την μνήμη, μέσω των Memory Ports, αποθηκεύονται σε προσωρινούς (temporary) πίνακες που δημιουργούνται εντός του επιταχυντή. Εφαρμόζοντας το array partition directive με complete partition σε αυτούς τους προσωρινούς πίνακες, επιτυγχάνεται η αποθήκευση των τριών σταθερών πινάκων left, right και EV σε μεμονωμένους καταχωρητές μέσα στην αρχιτεκτονική του επιταχυντή. Με λίγους παραπάνω υλικούς πόρους και κάποιους παραπάνω κύκλους, μηδενίζονται οι περιορισμοί της μνήμης για τους συγκεκριμένους πίνακες. Μπορούν, δηλαδή, να διαβαστούν 64 τιμές από τον πίνακα left, 64 τιμές από τον πίνακα right και 16 τιμές από τον πίνακα EV ταυτόχρονα σε ένα μόλις κύκλο.

7.5.2 Πρωτόκολλα επικοινωνίας, διαβάσματος και μεταφοράς δεδομένων για γνωστό και μη μέγεθος πινάκων

Το μέγεθος ενός πίνακα αποτελεί απαραίτητο στοιχείο για όλες σχεδόν τις αρχιτεκτονικές σχεδίασης υλικού. Το πρόγραμμα Vivado HLS -και κάθε πρόγραμμα που χρησιμοποιείται για την αρχιτεκτονική και κατασκευή υλικού-, πρέπει να γνωρίζει το ακριβές μέγεθος όλων των πινάκων που επεξεργάζεται.

Τα σταθερά μεγέθη πινάκων είναι ήδη γνωστά. Πρόκειται για τους πίνακες Left,Right και EV. Οι πίνακες, ωστόσο, με άγνωστο το μέγεθός τους κατά την κατασκευή υλικού, είναι οι x1,x2 και x3. Οι συγκεκριμένοι πίνακες είναι μεγέθους N*16. Σε αυτό το σημείο, η αρχιτεκτονική μπορεί να πάρει 2 κατευθύνσεις, όπως αναλύεται στη συνέχεια.

7.5.2.α Πρωτόκολλο επικοινωνίας διαβάσματος πινάκων x1,x2 και x3 χρησιμοποιώντας BRAM κελιά μνήμης

Το πρωτόκολλο επικοινωνίας και η λειτουργία των BRAM κελιών μνήμης της FPGA

Τα BRAM κελιά μνήμης αποτελούν μια αρκετά γρήγορη μνήμη εντός της FPGA. Χρησιμοποιούνται, κυρίως, για την αποθήκευση και συχνή χρήση πινάκων γνωστού μεγέθους από την DDR μνήμη. Ωστόσο παρουσιάζουν 2 σημαντικά μειονεκτήματα.

Μειονεκτήματα πρωτοκόλλου επικοινωνίας σε BRAM κελιά για x1, x2 και x3 πίνακες

Πρώτο μειονέκτημα αποτελεί το γεγονός ότι, παρ' όλο που τα BRAM κελιά αποτελούν μια πολύ γρήγορη μνήμη, υπάρχει πεπερασμένος αριθμός αυτών των κελιών στην FPGA. Εάν πραγματοποιηθεί προσέγγιση σε πραγματικά δεδομένα, το μέγεθος του N είναι μεταξύ κάτι εκατομμυρίων. Είναι, λοιπόν, αδύνατον να χωρέσουν GigaBytes δεδομένων σε MegaBytes BRAM Κελιά μνήμης. Ως δεύτερο μειονέκτημα παρουσιάζεται ο χρόνος που απαιτείται για να σταλούν και να αποθηκευτούν τα δεδομένα στα BRAM κελιά μνήμης της FPGA.

Προβλήματα που προκύπτουν από τα 2 παραπάνω μειονεκτήματα

Μετά την εφαρμογή της παραπάνω αρχιτεκτονικής, δημιουργήθηκαν πολλά προβλήματα που αφορούν την κακή σχεδίαση για την απόδοση του συνολικού χρόνου εκτέλεσης της όλης διαδικασίας.

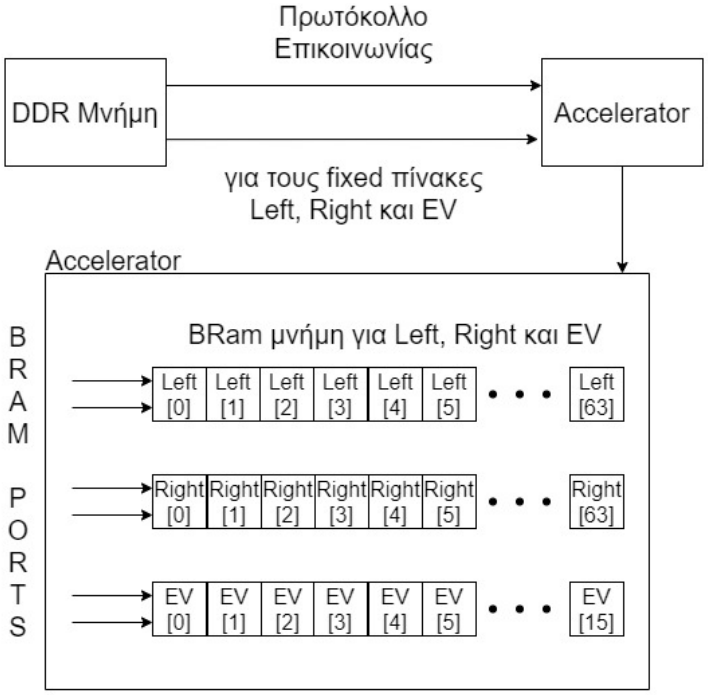
Ένα πρόβλημα αποτέλεσε η αλόγιστη χρήση πόρων BRAM κελιά μνήμης, σε μια προσπάθεια να χωρέσουν όσο το δυνατόν περισσότερα δεδομένα σε αυτά τα λίγα διαθέσιμα κελιά μνήμης. Σκοπός της αλόγιστης χρήσης BRAM κελιών μνήμης ήταν η μείωση, όσο το δυνατόν περισσότερο, των κλήσεων του επιταχυντή για το πέρας όλων των δεδομένων των πινάκων x1, x2 και x3.

Ένα ακόμη μείζον πρόβλημα ήταν ο χρόνος που απαιτείται για την μεταφορά όλων των δεδομένων με BRAM πρωτόκολλο. Συγκεκριμένα, μετά τη μεταφορά όλων των δεδομένων προς επεξεργασία, πρέπει να πραγματοποιηθεί επιπρόσθετο διάβασμα των δεδομένων από τα BRAM κελιά μνήμης.

Συμπέρασμα

Συνοψίζοντας, οι συνεχείς κλήσεις του επιταχυντή, η αλόγιστη χρήση πόρων, ο συνολικός χρόνος μεταφοράς δεδομένων στα BRAM κελιά και οι περιορισμένοι διαθέσιμοι πόροι από BRAM κελιά, καθιστούν τη χρήση των BRAM κελιών μνήμης, για τους πίνακες x1, x2 και x3, ακατάλληλους σχεδιαστικά.

Από την άλλη, εφόσον τα BRAM κελιά χρησιμοποιηθούν για την αποθήκευση των τριών, μικρών, σταθερών και απαράλλακτων, -μέχρι το πέρας της όλης διαδικασίας-, πινάκων Left, Right και EV, προσφέρουν ικανοποιητική αποδοτικότητα. Καθώς το μέγεθος των τριών σταθερών πινάκων είναι γνωστό και σχετικά μικρό, τα BRAM κελιά αποτελούν μια αποδοτική λύση για τη διαχείριση αυτών. Η Εικόνα 7. 3 απεικονίζει το πρωτόκολλο επικοινωνίας για τους 3 σταθερούς πίνακες Left, Right και EV στα BRAM κελιά μνήμης.

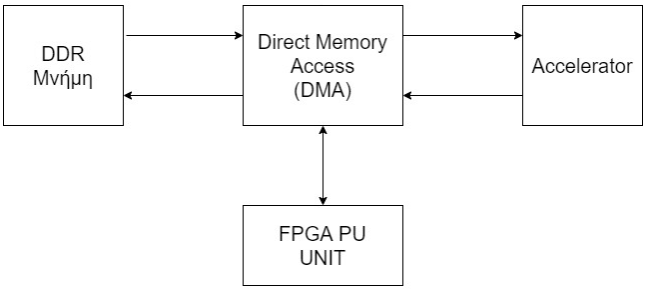


Εικόνα 7. 3 Πρωτόκολλο επικοινωνίας μεταξύ DDR μνήμης και BRAM κελιά για τους πίνακες Left, Right και EV

7.5.2.β Πρωτόκολλο επικοινωνίας διαβάσματος πινάκων x1,x2 και x3 με Streaming-FIFO πρωτόκολλο

Το Streaming-FIFO πρωτόκολλο επικοινωνίας και η λειτουργεία του

Το Streaming-FIFO πρωτόκολλο επικοινωνίας είναι ένα πρωτόκολλο που λειτουργεί πραγματοποιώντας μεταφορά δεδομένων με Streaming. Πιο συγκεκριμένα, δημιουργούνται Streams τα οποία αποτελούν ομοιότυπα κελιά (Shells) που περιέχουν ένα μόλις δεδομένο. Ο τρόπος που δρομολογούνται τα δεδομένα είναι ο First In First Out (FIFO). Σε επίπεδο Hardware χρησιμοποιείται το κύκλωμα Direct Memory Access (DMA), όπως φαίνεται στην Εικόνα 7. 4.



Εικόνα 7. 4 Block diagram από την DDR μνήμη στον επιταχυντή μέσω του DMA κυκλώματος

Το DMA κύκλωμα έχει τη δυνατότητα να διαβάσει δεδομένα από τη μνήμη και να τα δρομολογήσει σε επιταχυντές και το αντίστροφο. Ο εγκέφαλος, όμως, της διαδικασίας δρομολόγησης, όπως γίνεται διακριτό στην Εικόνα 7. 4, ελέγχεται πλήρως από τον επεξεργαστή της FPGA, τόσο στο διάβασμα (read), όσο και στο γράψιμο (write), από και προς τη DDR μνήμη.

Πλεονεκτήματα Streaming-FIFO πρωτόκολλου

Αρκετά είναι τα πλεονεκτήματα του Streaming-FIFO πρωτοκόλλου, τόσο για την αρχιτεκτονική του κεφαλαίου 5, όσο και για τη συνολική αποδοτικότητα της μεθόδου.

Το πρώτο και κυριότερο πλεονέκτημα είναι ότι, σε αντίθεση με τα BRAM κελιά και το πρωτόκολλο επικοινωνίας τους, το Streaming-FIFO πρωτόκολλο δε χρησιμοποιεί αλόγιστους

πόρους από την FPGA για να κρατήσει τα δεδομένα. Αναλυτικότερα, κατασκευάζεται ένα κελί που περιέχει ένα δεδομένο, το οποίο διαβάζεται από τη μνήμη μέσω DMA. Με τον τρόπο αυτό, είναι εφικτό να εξαλειφθεί ο περιορισμός του υλικού , σύμφωνα με τον οποίο κάθε πίνακας πρέπει να έχει σταθερό (fixed) μέγεθος κατά τη διάρκεια εκτέλεσης του επιταχυντή. Από την άλλη, υπάρχει ένας δείκτης (pointer) που παρουσιάζει τη διεύθυνση του ίδιου κελιού του Stream. Υπάρχει και σε αυτήν την περίπτωση fixed Hardware. Η διαφορά, ωστόσο, έγκειται στο ότι, αντί να δεσμευτεί σε πόρους ένας τεράστιος πίνακας, δεσμεύεται μόλις ένα κελί. Ακολούθως, σε αυτό το κελί-δείκτη δρομολογείται διαφορετικό δεδομένο σε κάθε κύκλο.

Το δεύτερο πλεονέκτημα είναι ότι, ο pipelined επιταχυντής, που κατασκευάστηκε και αναλύθηκε εκτενώς στο κεφάλαιο 5, μπορεί να δεχθεί εισόδους από ένα κελί-δείκτη και να επεξεργαστεί με αυτόν τον τρόπο όλους τους x1, x2 και x3 πίνακες ανεξαρτήτου μεγέθους του N, καλώντας τον επιταχυντή μία μόλις φορά. Αναλυτικότερα, ένα pipelined κύκλωμα απορροφά συνεχώς καινούργια δεδομένα σε κάθε κύκλο, δίχως να χρειάζεται κάποια πρόσθετη μνήμη αποθήκευσης των δεδομένων-εισόδων προτού και αφού απορροφηθούν από το pipelined κύκλωμα.

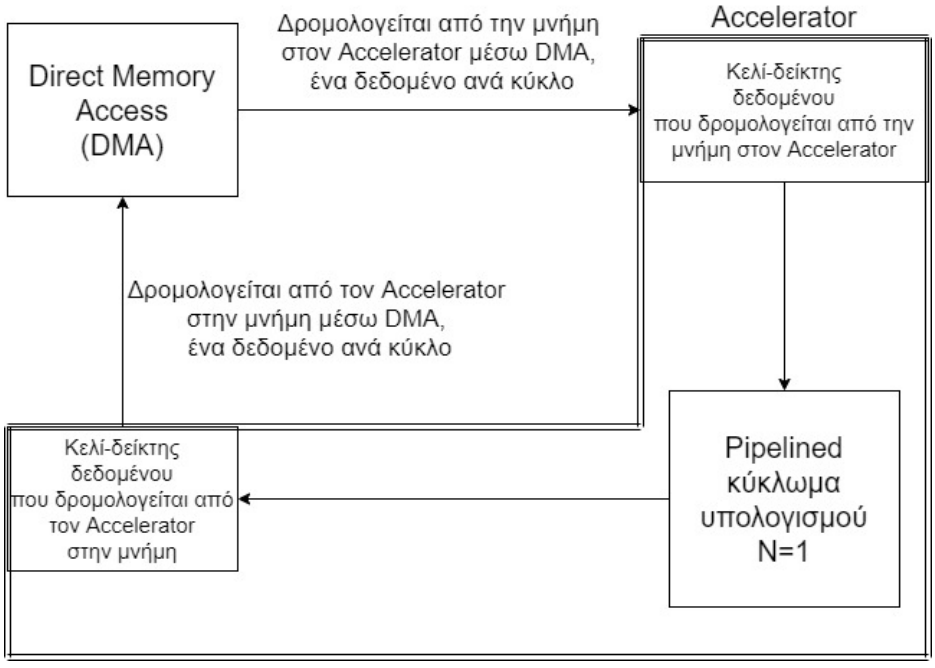
Μειονεκτήματα Streaming-FIFO πρωτόκολλου

Το μοναδικό μειονέκτημα του συγκεκριμένου πρωτοκόλλου είναι ότι, σε κάθε κύκλο δρομολογείται ένας πεπερασμένος αριθμός από bits, μέσω ενός DMA κυκλώματος. Αυτός ο περιορισμός οδηγεί σε ειδική προσαρμογή των απαιτήσεων του χρήστη που αφορά το μέγεθος της μεταφοράς δεδομένων. Αναλυτικότερα, αν το DMA κύκλωμα υποστηρίζει μεταφορά δεδομένων μεγέθους 1024bits σε κάθε κύκλο, για παράδειγμα, τότε θα μπορούσε να πραγματοποιηθεί η μεταφορά μιας δομής ενός πίνακα 16 στοιχείων διπλής ακρίβειας (που είναι το ακριβές μέγεθος ενός διανύσματος N) σε ένα μόλις κύκλο. Ωστόσο, λόγω περιορισμού του πλάτους των Memory Ports, τα DMA κυκλώματα δε διαθέτουν τόσο υψηλό νούμερο μεταφοράς bits σε έναν κύκλο. Ακόμη, λόγω του περιορισμού του Bandwidth της μνήμης, ο αριθμός των ταυτόχρονα ενεργών DMA κυκλωμάτων είναι περιορισμένος.

BRAM έναντι σε Streaming-FIFO

Λαμβάνοντας υπόψη τα μειονεκτήματα και πλεονεκτήματα και των 2 τρόπων που αναλύθηκαν παραπάνω, υπάρχουν αρκετές και σημαντικές διαφορές μεταξύ τους. Πρέπει να επιτευχθεί εκτίμηση αυτών, αναλύοντας το βάρος των ιδιαίτερων ιδιοτήτων που έχει το κάθε ένα.

Έστω το Streaming FIFO πρωτόκολλο. Η Εικόνα 7. 5 εξηγεί τον τρόπο που πραγματοποιούνται τα reads και writes από και προς τον επιταχυντή με Streaming-FIFO πρωτόκολλο.



Εικόνα 7. 5 Επίδειξη διαδικασίας ροής δεδομένων με Streaming-FIFO πρωτόκολλο επικοινωνίας

Έστω ότι το κελί-δείκτης δεδομένου της Εικόνα 7. 5 περιέχει έναν αριθμό διπλής ακρίβειας. Σε αυτήν την περίπτωση, θα διαβαστεί ένα N, 16, δηλαδή, τιμές διπλής ακρίβειας σε 16 κύκλους. Επομένως, το interval τίθεται στο νούμερο 16. Ωστόσο, αν ληφθεί υπόψη ο χρόνος που απαιτείται για τη μεταφορά δεδομένων στα BRAM κελιά, καθώς και οι συνεχείς κλήσεις του επιταχυντή, ο συνολικός χρόνος είναι πολύ μεγαλύτερος σε σύγκριση με το Streaming-FIFO πρωτόκολλο επικοινωνίας. Επίσης, τα δεδομένα από τους πίνακες X1, X2 και X3 χρειάζεται να διαβαστούν μία μόνο φορά, ώστε να ξεκινήσουν οι πράξεις για όλα τα N. Ο τρόπος με τον οποίο λειτουργεί το Streaming-FIFO πρωτόκολλο (διαφορετικό δεδομένο κάθε κύκλο), συνεισφέρει στην απορρόφηση των δεδομένων από το pipelined κύκλωμα, δίχως την αναγκαιότητα αποθήκευσης των δεδομένων εισόδου.

Συμπεράσματα

Στο Streaming-FIFO πρωτόκολλο, ακόμα και στην περίπτωση μεταφοράς μιας τιμής διπλής ακρίβειας από το DMA κύκλωμα, εξαλείφονται τα μειονεκτήματα της BRAM προσέγγισης. Αυτά είναι, ο πεπερασμένος αριθμός πόρων σε BRAM κελιά, ο συνολικός χρόνος διαβάσματος και μεταφοράς δεδομένων σε αυτά τα κελιά και οι συνεχείς κλήσεις του επιταχυντή. Βάση των παραπάνω λόγων, επιλέχθηκε το Streaming-FIFO πρωτόκολλο επικοινωνίας για τους πίνακες X1, X2 και X3.

7.6 SDSoC: Εργαλείο υλοποίησης και εφαρμογής των επιταχυντών

Το εργαλείο που χρησιμοποιήθηκε για την υλοποίηση του επιταχυντή στην FPGA, συγχρονίζοντας όλα τα απαραίτητα περιφερειακά, -όπως η μνήμη-, είναι το SDSoC. Το SDSoC είναι ένα εργαλείο της Xilinx που χρησιμοποιεί το Vivado HLS για τη δημιουργία επιταχυντών και το Vivado για την υλοποίηση των επιταχυντών στην FPGA, συγχρονίζοντας όλα τα απαραίτητα περιφερειακά.

Το εργαλείο Vivado είναι ένας τρόπος για να υλοποιηθούν οι επιταχυντές στην FPGA με συγκεκριμένα πρωτόκολλα επικοινωνίας, συγκεκριμένες μνήμες και συγκεκριμένη σχεδίαση, με ότι τυχόν περιφερειακά χρειάζεται ή επιθυμεί ο χρήστης για να υλοποιηθεί ένας επιταχυντής. Το SDSoC χρησιμοποιεί Vivado και Vivado HLS, παρέχοντας, ωστόσο, τα δικά του Directives για την υλοποίηση των επιταχυντών στην FPGA [25].

7.6.1 Directives του SDSoC που χρησιμοποιήθηκαν για την υλοποίηση του επιταχυντή του κεφαλαίου 5

Το Vivado, Vivado HLS και το SDSoC χρησιμοποιούν προκαθορισμένες ρυθμίσεις για την κατασκευή και υλοποίηση των επιταχυντών. Στην παράγραφο 7.3 αναλύθηκαν εκτενώς τα Directives του Vivado HLS που χρησιμοποιήθηκαν για την κατασκευή του επιταχυντή. Για να πραγματοποιηθούν, ωστόσο, οι προτιμήσεις υλοποίησης του επιταχυντή στην FPGA από το Vivado μέσω του SDSoC, πρέπει να γίνει χρήση των Directives του SDSoC.

7.6.1.α Copy & Access_pattern Directives

Το Copy Directive αντιγράφει όλα τα δεδομένα ενός πίνακα μεταξύ της μνήμης και του επιταχυντή. Το συγκεκριμένο Directive πρέπει να γνωρίζει τον πίνακα και το μέγεθός του για να εφαρμοστεί.

Το Access_pattern Directive χρησιμοποιείται για να επιλεγεί ένα πρωτόκολλο επικοινωνίας μιας μεταβλητής ή ενός πίνακα. Οι επιλογές των πρωτοκόλλων επικοινωνίας είναι 2: Streaming-FIFO πρωτόκολλο και BRAM.

Τα 2 παραπάνω directives πρέπει να χρησιμοποιηθούν μαζί σε έναν πίνακα, ώστε να υλοποιηθεί με Streaming-FIFO πρωτόκολλο επικοινωνίας. Επομένως, εφόσον η αρχιτεκτονική σχεδίαση του επιταχυντή του κεφαλαίου 5 απαιτεί να υλοποιηθούν οι πίνακες X1, X2 και X3 με Streaming-FIFO πρωτόκολλο επικοινωνίας, έγινε χρήση των Copy και Access_pattern Directives από το SDSoC για τους συγκεκριμένους πίνακες.

7.6.1.β Mem_attribute Directive

Το Mem_attribute Directive χρησιμοποιείται για τη διαμόρφωση της μνήμης, όπως ακριβώς επιθυμεί ο χρήστης. Υπάρχουν 2 επιλογές δόμησης της μνήμης. Η PHYSICAL_CONTIGUOUS και η NON_ PHYSICAL_CONTIGUOUS επιλογή.

NON_ PHYSICAL_CONTIGUOUS

Η επιλογή NON_ PHYSICAL_CONTIGUOUS παραπέμπει το SDSoC να διαβάσει και να επεξεργαστεί έναν πίνακα που έχει δεσμευτεί στη μνήμη με τη χρήση της malloc συνάρτησης της stdlib.h βιβλιοθήκης.

PHYSICAL_CONTIGUOUS

Η επιλογή PHYSICAL_CONTIGUOUS παραπέμπει το SDSoC να διαβάσει και να επεξεργαστεί έναν πίνακα που έχει δεσμευτεί στη μνήμη με τη χρήση της sds_alloc συνάρτησης της sds_lib.h βιβλιοθήκης.

Προτιμήθηκε η επιλογή της PHYSICAL_CONTIGUOUS μνήμης για τους πίνακες X1, X2 και X3, καθώς παρατηρήθηκε πως είναι ταχύτερη από τη NON_ PHYSICAL_CONTIGUOUS.

7.6.1.γ Data_mover Directive

Το Data_mover Directive χρησιμοποιείται για να προσδιοριστεί ο τρόπος με τον οποίο ένας πίνακας θα μεταφερθεί μέσω DMA. Ακόμη, υπάρχει η δυνατότητα να προσδιοριστούν πόσα DMA κυκλώματα θα χρησιμοποιηθούν, αλλά και ποιο DMA θα χρησιμοποιηθεί για συγκεκριμένο/-ους πίνακα/-ες. Για τις υλοποιήσεις του επιταχυντή του κεφαλαίου 5 χρησιμοποιήθηκαν 3 ξεχωριστά DMA κυκλώματα με Scatter & Gather για τους πίνακες X1, X2 και X3 αντίστοιχα.

7.6.1.δ Resource Directive

Το Resource Directive χρησιμοποιείται για παράλληλη εκτέλεση πολλών πανομοιότυπων επιταχυντών. Αναλυτικότερα, δημιουργούνται αντίγραφα κυκλώματα ενός επιταχυντή με διαφορετικό ID το καθένα για παράλληλη εκτέλεση. Ο συγχρονισμός όλων των πανομοιότυπων επιταχυντών γίνεται αυτόματα από το Resource Directive, χρησιμοποιώντας την πιο απλή μορφή Barrier.

7.7 Υλοποίηση του επιταχυντή της μέγιστης πιθανοφάνειας στην FPGA ZCU102

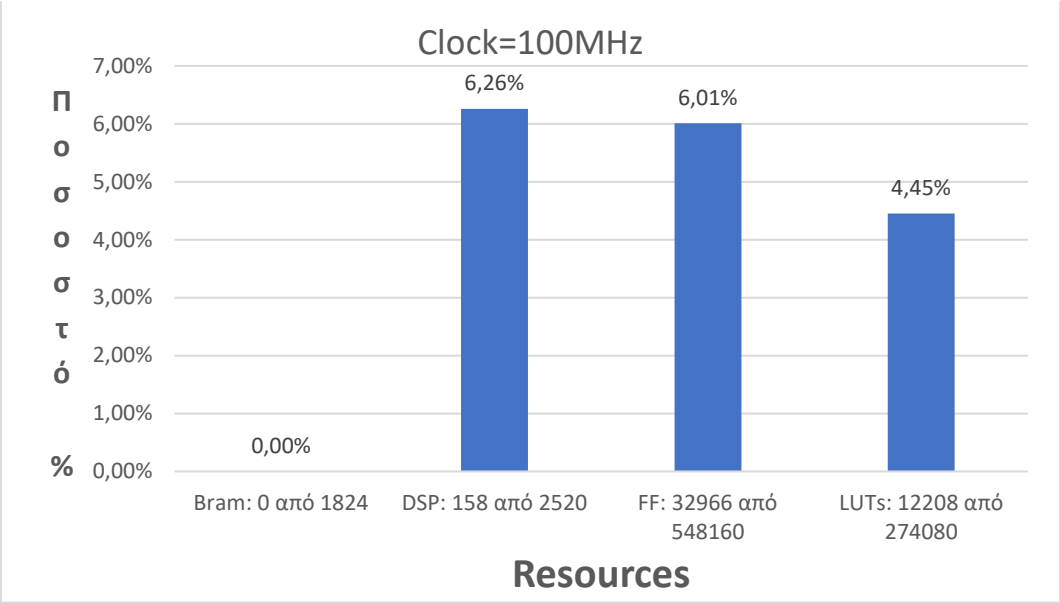
Η υλοποίηση του επιταχυντή της μέγιστης πιθανοφάνειας πραγματοποιήθηκε στην FPGA ZCU102 Ultrascale MPSoC. Η ZCU102 είναι αρκετά μεγάλη και γρήγορη FPGA με DDR4 μνήμη, ικανοποιητικό Bandwidth μνήμης, μεγάλης συχνότητας επεξεργαστή και αρκετά εμπλουτισμένη σε υλικούς πόρους (resources). Η υλοποίηση του επιταχυντή πλαισιώνεται από 3 διαφορετικούς παράγοντες της FPGA.

Interval του Pipelined κυκλώματος

Θεωρητικά, το μέγιστο πλάτος της πληροφορίας από και προς τη μνήμη, μέσω ενός DMA κυκλώματος μιας ZCU102 MPSoC FPGA, είναι τα 128bits. Ωστόσο, δε συμβαίνει το ίδιο και στην πράξη, καθώς δεν υπάρχει υλικό χωρίς απώλειες, έστω και σε ελάχιστο βαθμό. Συνεπώς, υλοποιήθηκε η μεταφορά ενός δεδομένου διπλής ακρίβειας (64bits) από τους DMAs. Ένα δεδομένο ανά κύκλο σημαίνει πως, το interval του επιταχυντή τίθεται στο 16. Η fixed τιμή του interval στο 16 είναι ο λόγος που επιλέχθηκε η συγκεκριμένη αρχιτεκτονική σχεδίαση του επιταχυντή του κεφαλαίου 5.

Υλικοί πόροι της ZCU102 (Resourses)

Τα resources της αρχιτεκτονικής με 100MHz προκαθορισμένο ρολόι, είναι τα εξής:



Εικόνα 7. 6 Διάγραμμα χρησιμοποίησης υλικών πόρων για 100MHz ρολόι

Όπως φαίνεται στην Εικόνα 7. 6, τα Resources που απαιτούνται για τις ανάγκες της υλοποίησης του επιταχυντή είναι ελάχιστα. Αν υπήρχε η δυνατότητα μείωσης του interval σε 8, 4, 2 ή και μόλις 1, τα Resources που θα απαιτούνταν θα αυξάνονταν ραγδαία. Για την παρούσα διπλωματική εργασία με interval 16 και ρολόι 100MHz, τα απαιτούμενα Resources δεν ξεπερνούν το 6,26%. Η χρήση των απαιτούμενων Resources παραμένουν σε τόσο χαμηλό ποσοστό εξαιτίας του interval. Με interval 16 απαιτείται η εκτέλεση ενός συγκριμένου αριθμού παράλληλων πράξεων και η δέσμευση συγκεκριμένης μνήμης για την υλοποίηση του pipeline στο κύκλωμα. Το Vivado HLS στο Synthesis δεσμεύει Resources τόσα ώστε, να ανταποκρίνονται στον μέγιστο αριθμό των παράλληλων πράξεων και της απαιτούμενης μνήμης. Στην περίπτωση που μειώνεται το interval, χρειάζονται παραπάνω Resources, διότι απαιτείται μεγαλύτερος αποθηκευτικός χώρος και πρέπει να εκτελεστούν περισσότερες πράξεις παράλληλα.

Ρολόι, ταχύτητα της ZCU102 και συνολικό βάθος Pipeline (σε κύκλους)

Το ρολόι της FPGA σε αυτή την πρώτη υλοποίηση του επιταχυντή παρέμεινε στην προκαθορισμένη τιμή του στα 100MHz. Όταν εφαρμόστηκε η υλοποίηση με 100MHz ρολόι, το ρολόι του επιταχυντή για έναν κύκλο τέθηκε στα 10ns, ως στόχος του ρολογιού (Target Clock). Το εκτιμώμενο ρολόι (Estimated Clock) τέθηκε στα 7,44ns με αβεβαιότητα (Uncertainty) στα 2,7ns. Τέλος, το βάθος του pipeline με interval 16 στα 100MHz ρολόι ορίστηκε στους 59 κύκλους.

7.7.1 Υλοποιήσεις στην ZCU102 διαφοροποιώντας παραμέτρους

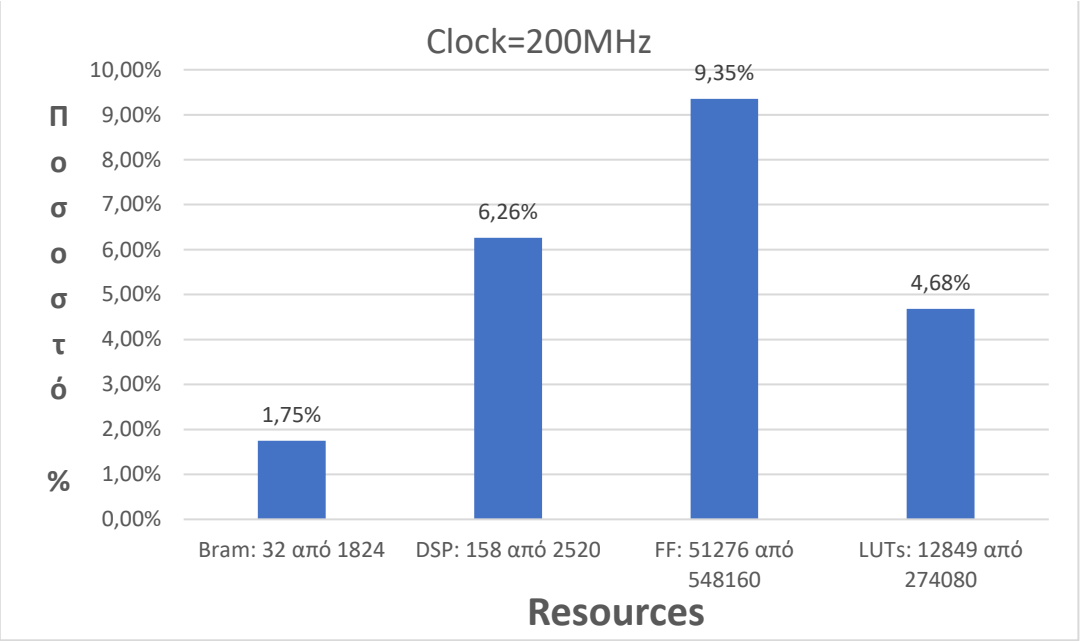
Στην παράγραφο 7.7 αναλύθηκαν διεξοδικά οι 3 διαφορετικοί παράγοντες της ZCU102. Ωστόσο, πραγματοποιήθηκε τροποποίηση αυτών των παραμέτρων για να επιτευχθεί μια πιο γρήγορη προσέγγιση του επιταχυντή της μέγιστης πιθανοφάνειας. Στο κεφάλαιο 7.4 έγινε γνωστό πως, με τους περιορισμούς των πρωτόκολλων επικοινωνίας και του Bandwidth της μνήμης είναι αδύνατο να τροποποιηθεί η παράμετρος Interval του Pipelined επιταχυντή. Οι άλλες, όμως, 2 παράμετροι μπορούν να τροποποιηθούν με τέτοιο τρόπο ώστε, να εκτελεστεί ο επιταχυντής πιο γρήγορα.

Ρολόι, ταχύτητα της ZCU102 και συνολικό βάθος Pipeline (σε κύκλους)

Το ρολόι της FPGA στο κεφάλαιο 7.6 έλαβε την προκαθορισμένη τιμή στα 100MHz. Σε αυτή την ιδιαίτερη υλοποίηση, το ρολόι τέθηκε στα 200MHz, διπλασιάστηκε. Αφού εφαρμόστηκε η υλοποίηση αυτή, το ρολόι του επιταχυντή για ένα κύκλο μειώθηκε στις καινούργιες τιμές του Target Clock 5ns, Estimated Clock 3,65ns και Uncertainty 1,35ns. Η μείωση του ρολογιού του επιταχυντή είναι αναμενόμενη, καθώς το ρολόι της FPGA διπλασιάστηκε. Ωστόσο, το βάθος του pipeline αυξήθηκε στους 96 κύκλους από τους 56 της υλοποίησης στα 100MHz ρολόι. Η αύξηση της ταχύτητας του ρολογιού της FPGA είναι ένας από τους πιο συνηθισμένους τρόπους βελτίωσης της απόδοσης των επιταχυντών.

Υλικοί πόροι της ZCU102 (Resourses)

Τα resources της αρχιτεκτονικής με διπλάσιο ρολόι 200MHz είναι τα εξής:



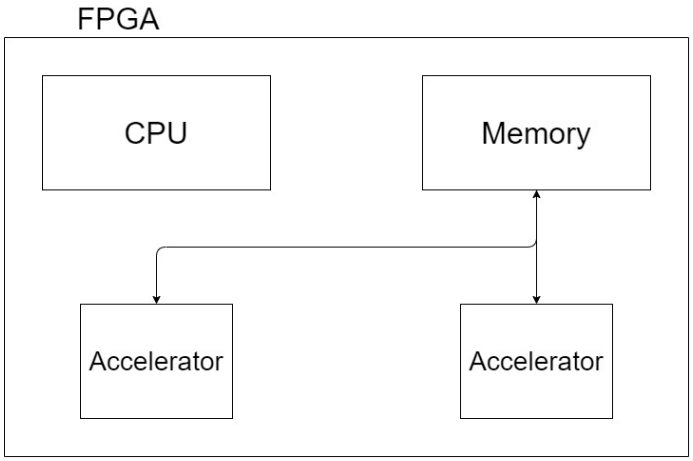
Εικόνα 7. 7 Διάγραμμα χρησιμοποίησης υλικών πόρων για 200MHz ρολόι

Όπως παρατηρείται στην Εικόνα 7. 7, χρειάζονται λίγα περισσότερα resources για να υλοποιηθεί το κύκλωμα με 200MHz ρολόι. Τα resources δεν είναι ο μόνος παράγοντας που αλλάζει. Όπως αναφέρθηκε παραπάνω, το pipelined κύκλωμα έχει γίνει μεγαλύτερο σε βάθος (περισσότεροι κύκλοι ρολογιού), συγκριτικά με το 100MHz ρολόι που ήταν μικρότερο. Αυτές οι 2 αλλαγές πραγματοποιήθηκαν από το Vivado HLS για να μπορέσει να εκτελεστεί ο επιταχυντής με 200MHz ρολόι.

7.7.2 Πολλαπλές υλοποιήσεις του επιταχυντή στην ZCU102 προσεγγίζοντας παράλληλη αρχιτεκτονική σχεδίαση

Κάνοντας χρήση του Resource Directive (βλέπε παράγραφος 7.6.1.δ), προσεγγίζεται μια αρχιτεκτονική σχεδίαση με πολλαπλούς επιταχυντές ίδιας κατασκευής, με στόχο την παράλληλη εκτέλεση και επεξεργασία των δεδομένων. Εκτός από τις 2 υλοποιήσεις των 7.7 και 7.7.1 παραγραφών, κατασκευάστηκαν άλλες 4 υλοποιήσεις στην πλατφόρμα ZCU102.

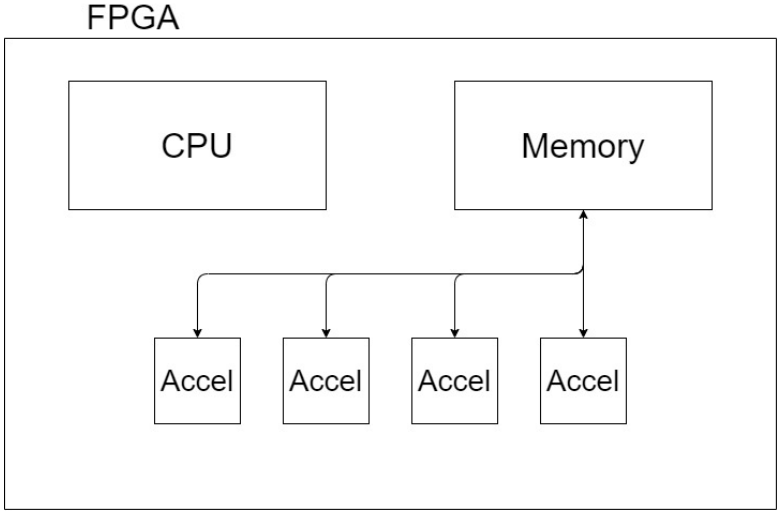
7.7.2.α Υλοποίηση αρχιτεκτονικής σχεδίασης με 2 ίδιας κατασκευής επιταχυντές



Εικόνα 7. 8 Χρήση δύο ίδιων επιταχυντών για παράλληλη εκτέλεση δεδομένων

Όπως φαίνεται στην Εικόνα 7. 8, 2 ίδιοι επιταχυντές εκτελούν παράλληλα διαφορετικά δεδομένα από τη μνήμη, ώστε να μοιραστεί ο όγκος των δεδομένων προς επεξεργασία. Με αυτόν τον τρόπο, επιτυγχάνεται ένας διαμοιρασμός των δεδομένων που έχει ως συνέπεια τη μείωση του συνολικού χρόνου που απαιτείται για την επεξεργασία όλων των δεδομένων.

7.7.2.β Υλοποίηση αρχιτεκτονικής σχεδίασης με 4 ίδιας κατασκευής επιταχυντές



Εικόνα 7. 9 Χρήση τεσσάρων ίδιων επιταχυντών για παράλληλη εκτέλεση δεδομένων

Η λογική της υλοποίησης της Εικόνα 7. 9 είναι πανομοιότυπη με την λογική της υλοποίησης της 7.7.2.α παραγράφου. Μόνη τους διαφορά αποτελεί το ότι στην υλοποίηση αυτής της παραγράφου πραγματοποιείται παραλληλισμός με 4 πανομοιότυπους επιταχυντές, διαμοιράζοντας και εδώ τα δεδομένα προς επεξεργασία στον κάθε επιταχυντή.

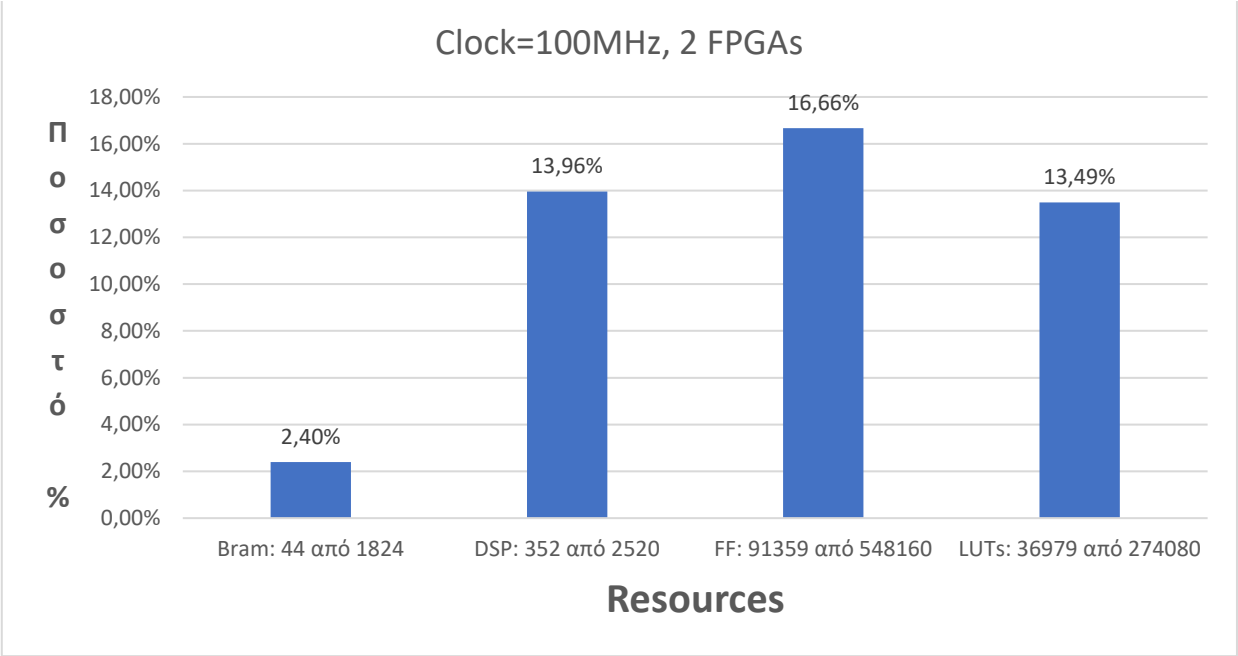
Ρολόι, ταχύτητα της ZCU102 και συνολικό βάθος Pipeline (σε κύκλους)

Το ρολόι που εφαρμόστηκε, αρχικά, για τις υλοποιήσεις των 7.7.2.α και 7.7.2.β παραγράφων, είναι τα 100MHz. Καθώς πρόκειται για πανομοιότυπους επιταχυντές, το Clock Target, Estimated Clock και Uncertainty παρέμειναν τα ίδια με την υλοποίηση της 7.7 παραγράφου για τα 100MHz. Έγινε προσπάθεια εκτέλεσης των 7.7.2.α και 7.7.2.β παραγραφών στα 200MHz, όπως έγινε στην 7.7.1 παράγραφο. Το εργαλείο SDSoC, ωστόσο, δε μπόρεσε να πραγματοποιήσει τις επιθυμητές υλοποιήσεις. Σε αντίθεση, πραγματοποιήθηκε Synthesis και Place and Route από το SDSoC με

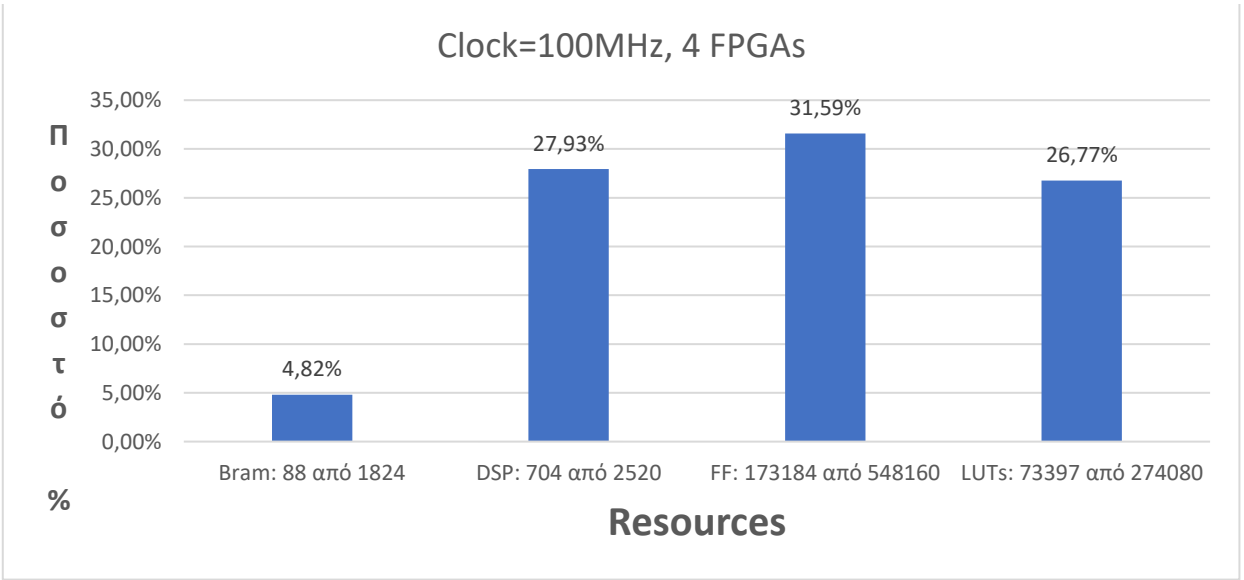
150MHz ρολόι, τόσο για τις υλοποιήσεις της παραγράφου 7.7.2.α, όσο και για την παράγραφο 7.7.2.β. Το Target Clock για τα 150MHz τέθηκε στα 6,67ns, το Estimated Clock στα 4.99ns και το Uncertainty στα 1,8ns. Επιπρόσθετα, το βάθος του pipeline τέθηκε στους 85 κύκλους, ανάμεσα στις δύο προηγούμενες υλοποιήσεις των 100MHz και 200MHz (59 < 85 < 94).

Υλικοί πόροι της ZCU102 (Resources)

Τα resources που απαιτούνται για τις υλοποιήσεις των δύο και τεσσάρων επιταχυντών παράλληλα στα 100MHz ρολόι γίνονται διακριτά στην Εικόνα 7. 10 και Εικόνα 7. 11 αντίστοιχα.

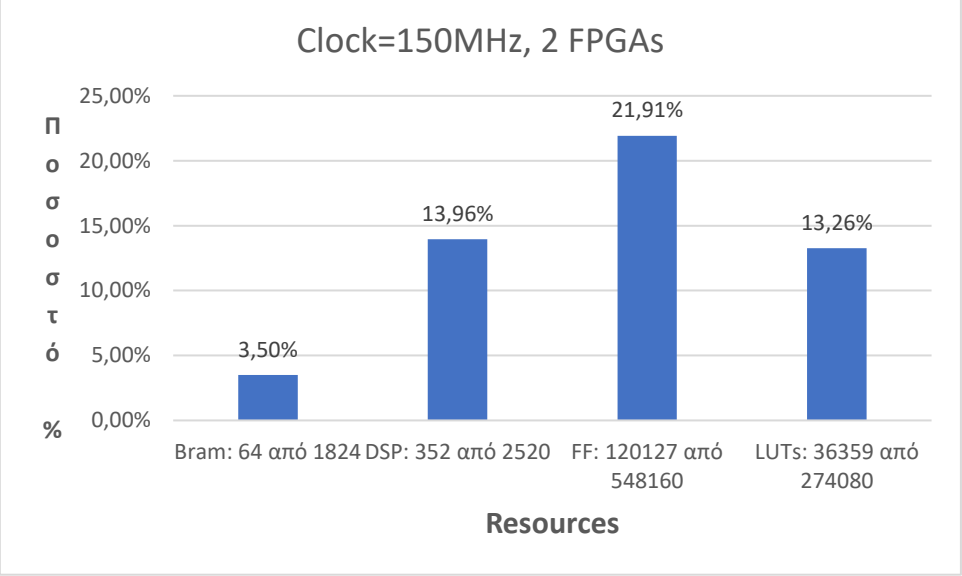


Εικόνα 7. 10 Διάγραμμα χρήσης υλικών πόρων για την υλοποίηση των δύο επιταχυντών στα 100MHz ρολόι

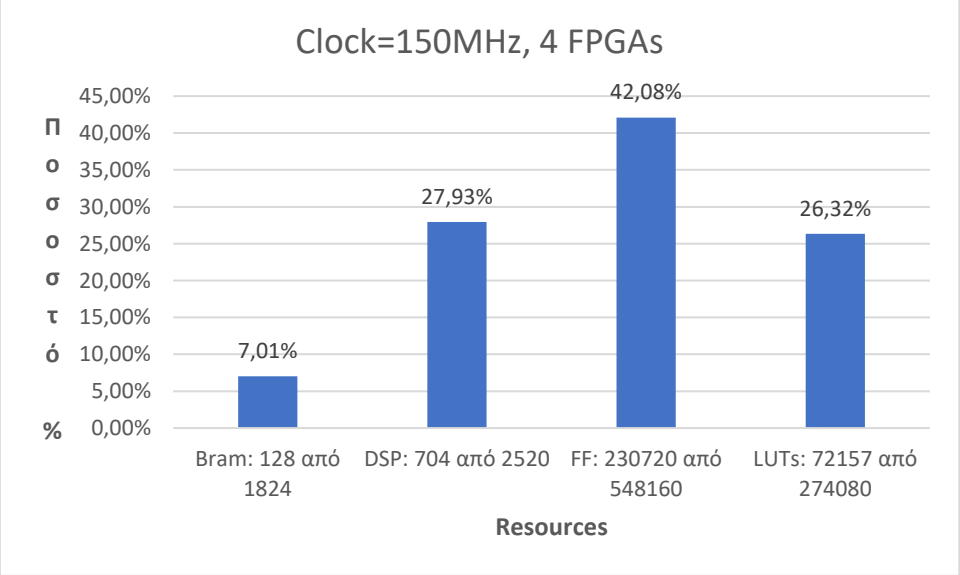


Εικόνα 7. 11 Διάγραμμα χρήσης υλικών πόρων για την υλοποίηση των τεσσάρων επιταχυντών στα 100MHz ρολόι

Όπως γίνεται διακριτό από τα διαγράμματα των Εικόνα 7. 10 και Εικόνα 7. 11, το Vivado HLS δεσμεύει περισσότερα ή λιγότερα resources για τις ανάγκες της παράλληλης εκτέλεσης των επιταχυντών. Αυτό συμβαίνει και στις 2 υλοποιήσεις των δύο ή τεσσάρων επιταχυντών παράλληλα. Τα resources δεν ξεπερνούν το 32% για τις υλοποιήσεις στα 100MHz ρολόι. Στα 150MHz τα resources των υλοποιήσεων με 2 και 4 επιταχυντές παράλληλα διαμορφώνονται σύμφωνα με την Εικόνα 7. 12 και Εικόνα 7. 13 αντίστοιχα.



Εικόνα 7. 12 Διάγραμμα χρήσης υλικών πόρων για την υλοποίηση των δύο επιταχυντών στα 150MHz ρολόι



Εικόνα 7. 13 Διάγραμμα χρήσης υλικών πόρων για την υλοποίηση των τεσσάρων επιταχυντών στα 150MHz ρολόι

Πραγματοποιώντας συγκρίσεις των resources με ίδιο αριθμό επιταχυντών, στις υλοποιήσεις με 100MHz και 150MHz ρολόι, παρατηρείται μία αύξηση των BRAMs και των FFs, καθώς και μία μείωση στα resources των LUTs. Όλες αυτές οι αλλαγές στα resources των υλοποιήσεων πραγματοποιούνται από το Vivado HLS, με απώτερο σκοπό την εκτέλεση των υλοποιήσεων με 2 ή 4 επιταχυντές παράλληλα, αυξάνοντας το ρολόι κατά 50MHz.

Τέλος, στην Εικόνα 7. 14 παρατίθεται ένα συγκεντρωτικό ποσοστιαίο διάγραμμα που περιέχει τα resources όλων των υλοποιήσεων που έγιναν.

	Bram	DSP	FF	LUTs
100MHz	0,00%	6,26%	6,01%	4,45%
200MHz	1,75%	6,26%	9,35%	4,68%
100MHz, 2-Accel	2,40%	13,96%	16,66%	13,49%
150MHz, 2-Accel	3,50%	13,96%	21,91%	13,26%
100MHz, 4-Accel	4,82%	27,93%	31,59%	26,77%
150MHz, 4-Accel	7,01%	27,93%	42,08%	26,32%

Εικόνα 7. 14 Συγκεντρωτικό ποσοστιαίο διάγραμμα των resources όλων των υλοποιήσεων

Κεφάλαιο 8: Αξιολόγηση- Αποτελέσματα

Το όγδοο κεφάλαιο παραθέτει τα αποτελέσματα των υλοποιήσεων του hardware του κεφαλαίου 7, σε σύγκριση με τα αποτελέσματα υλοποίησης της μεθόδου της μέγιστης πιθανοφάνειας σε software. Επιπρόσθετα, το κεφάλαιο αυτό πραγματεύεται την υλοποίηση της αρχιτεκτονικής σχεδίασης του κεφαλαίου 6, με βάση τη μέθοδο της μέγιστης πιθανοφάνειας, προσεγγίζοντας διάφορες απομακρυσμένες πλατφόρμες αναδιατασσόμενης λογικής με αποκεντρωμένους πόρους.

8.1 Αποτελέσματα υλοποίησης της μεθόδου της μέγιστης πιθανοφάνειας σε software.

Το hardware που επιλέχθηκε είναι ένα laptop με επεξεργαστή Intel i5-7200U των 2 πυρήνων, με Multi-Threading (2 Thread για κάθε πυρήνα) στα 2.5GHz ταχύτητα πυρήνων. Πραγματοποιήθηκαν 2 τρόποι εκτέλεσης της συνάρτησης της μέγιστης πιθανοφάνειας από το RAxML:

- Σειριακή (Sequential)
- AVX

Τόσο στη σειριακή, όσο και στην AVX εκτέλεση της μεθόδου της μέγιστης πιθανοφάνειας, πάρθηκαν 1000 μετρήσεις του αλγόριθμου, μέσω του RAxML για συγκεκριμένο μήκος αλληλουχίας N. Έπειτα, τυπώθηκε ο μέσος όρος των 1000 μετρήσεων, έτσι ώστε να υπάρξει μια πιο δικαία εικόνα του υπολογισμού του χρόνου, για συγκεκριμένο N. Επιπρόσθετα, βάση του χρόνου που υπολογίστηκε σε όλες τις υλοποιήσεις του software και όλες τις υλοποιήσεις των FPGA του κεφαλαίου 7, παρατίθεται η διεκπεραιωτική ικανότητα (Throughput) των υλοποιήσεων, διαιρώντας το μέγεθος του N με τον αντίστοιχο χρόνο εκτέλεσης (N/sec). Τα αποτελέσματα της σειριακής και AVX εκτέλεσης της μεθόδου της μέγιστης πιθανοφάνειας στο laptop, με τον επεξεργαστή που αναγράφεται παραπάνω, γίνονται διακριτά στον Πίνακας 8. 1.

N	**Χρόνος SeQ (sec)**	**Χρόνος AVX (sec)**	**Throughput SeQ (N/Sec)**	**Throughput AVX (N/Sec)**	**Απόδοση Throughput AVX/SeQ %**
1000	0,000075	0,000039	13333333,33	25641025,64	92,30%
10000	0,000803	0,000448	12453300,12	22321428,57	79.24%
100000	0,008277	0,004949	12081672,1	20206102,24	67,24%
1000000	0,082046	0,047684	12188284,62	20971395,02	72,06%
2000000	0,165035	0,099054	12118641,5	20191006,93	66,61%

Πίνακας 8. 1 Πίνακας χρόνου-throughput και αύξηση απόδοσης(%) μεταξύ σειριακής και AVX υλοποίησης

Όπως παρατηρείται στον Πίνακας 8. 1, η εκτέλεση της μεθόδου της μέγιστης πιθανοφάνειας με την AVX μέθοδο προσφέρει μεγαλύτερο throughput, συγκριτικά με τη σειριακή εκτέλεση (SeQ). Η τελευταία στήλη του Πίνακας 8. 1 περιλαμβάνει το ποσοστό (%) της περισσότερης απόδοσης που δίνει στο χρήστη η AVX εκτέλεση έναντι της σειριακής. Πιο συγκεκριμένα, στην πρώτη σειρά του Πίνακας 8. 1, για παράδειγμα, για N=1000, το Throughput του AVX είναι 1,923 (92,3%) φορές περισσότερο από το Throughput του SeQ, όπως δίνεται από το πηλίκο της διαίρεσης Throughput AVX/SeQ. Ομοίως, προκύπτει και το ποσοστό της απόδοσης για τις υπόλοιπες υλοποιήσεις. Όπως γίνεται αντιληπτό, η απόδοση για όλα τα N ξεπερνάει το 66% και φτάνει μέχρι και 92,3%.

8.2 Αποτελέσματα των υλοποιήσεων του κεφαλαίου 7

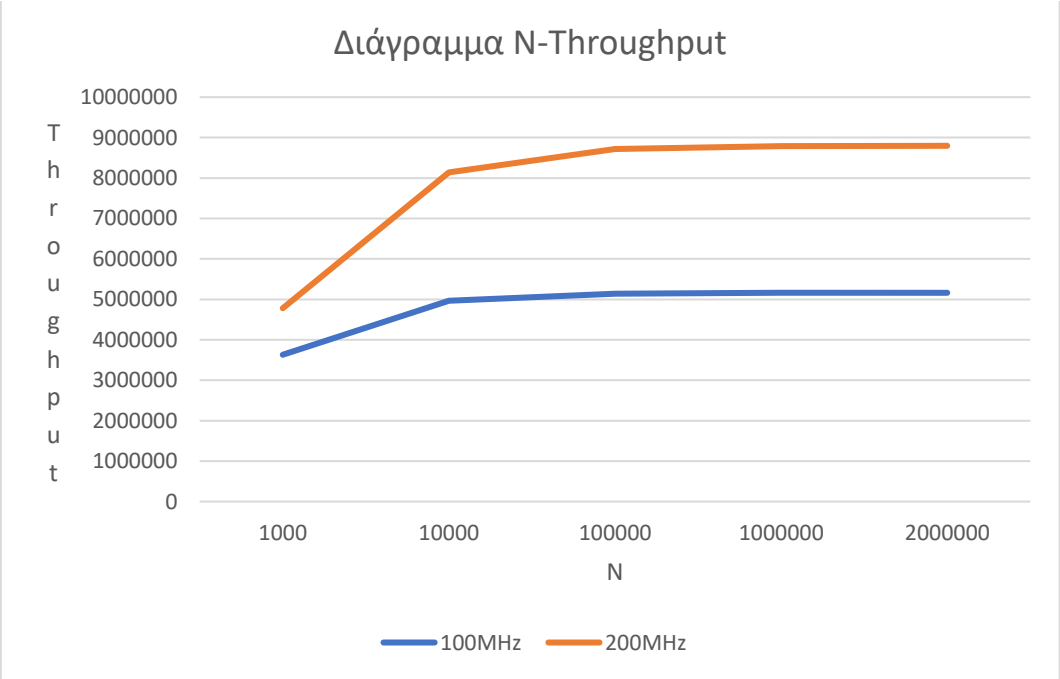
Στην παράγραφο 7.7 του κεφαλαίου 7, πραγματοποιήθηκαν πολλαπλές υλοποιήσεις του επιταχυντή πάνω στην FPGA ZCU102 MPSoC. Όπως αναλύθηκε, το interval παραμένει σταθερά 16 για όλες τις υλοποιήσεις. Οι μετρήσεις που έγιναν αφορούν το μέγεθος του N, το οποίο κυμαίνεται

στις τιμές 1.000, 10.000, 100.000. 1.000.000 και 2.000.000 . Για τις ανάγκες της υψηλής ακρίβειας των μετρήσεων, χρησιμοποιήθηκε η κλάση Perf_counter του SDSoC. Η συγκεκριμένη κλάση έχει τη δυνατότητα να μετρήσει τους κύκλους από ένα επιθυμητό διάστημα του χρήστη. Για το λόγο αυτό, πριν την έναρξη οποιασδήποτε υλοποίησης του επιταχυντή, ο μετρητής των κύκλων της κλάσης Perf_counter αρχικοποιείται στο 0 και μόλις ο/οι επιταχυντής/-ές τελειώσει/-ουν την εκτέλεση των πράξεών του/τους, εκτυπώνονται στην οθόνη οι συνολικοί κύκλοι που απαιτούνται για την εκτέλεση της υλοποίησης του επιταχυντή/-ών. Διαιρώντας τους κύκλους που απαιτούνται, για συγκεκριμένο N, με τη συχνότητα του PU της ZCU102, υπολογίστηκε ο συνολικός χρόνος που απαιτείται για την εκτέλεση των υλοποιήσεων σε δευτερόλεπτα (seconds). Ο Πίνακας 8. 2 δείχνει τα αποτελέσματα σε seconds και throughput για συγκεκριμένες τιμές N, σύμφωνα με τις υλοποιήσεις ενός επιταχυντή στα 100MHz ρολόι και ενός επιταχυντή στα 200MHz ρολόι του κεφαλαίου 7.7 και 7.7.1 αντίστοιχα.

N	**Χρόνος (sec) 100MHz**	**Χρόνος (sec) 200MHz**	**Throughput 100MHz (N/Sec)**	**Throughput 200MHz (N/Sec)**	**Αύξηση απόδοσης (%)**
1000	0,000275468	0,000209201	3630191,705	4780093,715	31,67%
10000	0,002013521	0,001227953	4966423,635	8143635,547	63,97%
100000	0,019459566	0,011468607	5138860,732	8719454,904	68,67%
1000000	0,193731773	0,113730213	5161775,928	8792738,331	70,34%
2000000	0,387428552	0,227354395	5162242,153	8796838,974	70,40%

Πίνακας 8. 2 Πίνακας χρόνου-throughput και αύξηση απόδοσης(%) μεταξύ των υλοποιήσεων των 100 και 200MHz για έναν επιταχυντή.

Όπως γίνεται διακριτό στον Πίνακας 8. 2, διπλασιάζοντας το ρολόι, η απόδοση αυξάνεται έως και 70,4% σε σύγκριση με την υλοποίηση των 100MHz. Το Διάγραμμα 8. 1 παραθέτει την πορεία του throughput για τις 2 υλοποιήσεις των 100 και 200MHz.



Διάγραμμα 8. 1 Διάγραμμα N-Throughput των υλοποιήσεων του ενός επιταχυντή στα 100 και 200MHz.

Όπως γίνεται διακριτό στο Διάγραμμα 8. 1, υπάρχει διαφορά στο throughput μεταξύ μικρού N (1000) και μεγαλύτερων N (>10000). Αυτό συμβαίνει ακόμα και στην πιο απλή περίπτωση του ενός επιταχυντή στα 100MHz ρολόι, διότι απαιτείται συγχρονισμός του επιταχυντή με τα περιφερειακά της FPGA από το PU της. Τα δεδομένα για N=1000 είναι λίγα, οπότε ο επιταχυντής

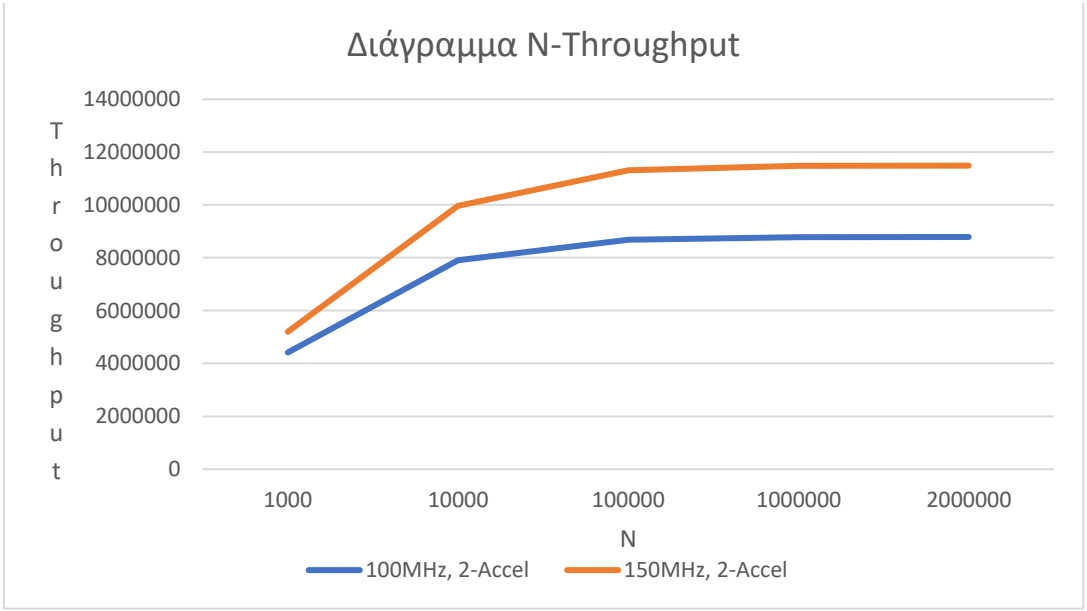
δεν προλαβαίνει να αξιοποιήσει τις δυνατότητές του. Ωστόσο, μόλις αυξηθούν τα δεδομένα προς επεξεργασία, το throughput ανεβαίνει, όπως διακρίνεται στην καμπύλη του Διάγραμμα 8. 1. Έπειτα, ανεβαίνει απειροελάχιστα, όσο μεγαλώνει το N. Αναλυτικότερα, από το N=10.000 και μετά, διακρίνεται η αξιοποίηση των δυνατοτήτων της υλοποίησης του επιταχυντή στα 100MHz ρολόι. Επιπρόσθετα, η καμπύλη N-Throughput της υλοποίησης των 200MHz μοιάζει αρκετά με την καμπύλη των 100MHz. Ωστόσο, η καμπύλη των 200MHz υστερεί στην ομαλοποίηση του throughput από N=10.000 με N=100.000 σε σύγκριση με την ταχύτητα ομαλοποίησης του throughput της καμπύλης των 100MHz. Επίσης, το throughput αυξάνεται σε σύγκριση με την υλοποίηση των 100MHz. Καθώς διπλασιάζεται το ρολόι, θα περίμενε κανείς διπλασιασμό και στο throughput. Το throughput, ωστόσο, δεν παρουσιάζει διπλασιασμό της απόδοσης, διότι όπως έχει προαναφερθεί, η πράξη απέχει από την θεωρία.

Ακολούθως, ιδιαίτερο ενδιαφέρον παρουσιάζουν τα αποτελέσματα των υλοποιήσεων της παραγράφου 7.7.2.α. Οι υλοποιήσεις αυτές αφορούν τη χρήση 2 πανομοιότυπων επιταχυντών παράλληλα στα 100 και 150MHz ρολόι. Τα αποτελέσματα των υλοποιήσεων χρόνος-throughput-απόδοση παρατίθενται στον Πίνακας 8. 3.

N	**Χρόνος (sec) 100MHz**	**Χρόνος (sec) 150MHz**	**Throughput 100MHz (N/Sec)**	**Throughput 150MHz (N/Sec)**	**Αύξηση απόδοσης (%)**
1000	0,000226463	0,000192359	4415739,283	5198606,651	17,72%
10000	0,001265377	0,00100345	7902786,037	9965615,222	26,10%
100000	0,011524792	0,008839054	8676945,855	11313428,04	30,38%
1000000	0,113859276	0,087130743	8782771,497	11477005,34	30,67%
2000000	0,227633953	0,174117551	8786035,549	11486492,81	30,73%

Πίνακας 8. 3 Πίνακας χρόνου-throughput και αύξηση απόδοσης(%) μεταξύ των υλοποιήσεων των δύο επιταχυντών με 100 και 150MHz ρολόι.

Όπως διακρίνεται στον Πίνακας 8. 3, το throughput παρουσιάζει ελάχιστη διαφορά σε σύγκριση με την υλοποίηση του ενός επιταχυντή στα 200MHz (Πίνακας 8. 2). Ο λόγος έγκειται στο ότι, οι 2 υλοποιήσεις πραγματεύονται το διπλασιασμό της απόδοσης με διαφορετικό τρόπο. Ωστόσο, παρόλο που ο τρόπος επίτευξης διαφέρει, τα αποτελέσματα είναι πανομοιότυπα. Ακόμη, αυξάνοντας το ρολόι της υλοποίησης των 2 επιταχυντών κατά 50MHz, παρατηρείται αύξηση της απόδοσης από 17,72% για μικρό N, έως και 30,73% για μεγάλο N. Το Διάγραμμα 8. 2 δείχνει τις καμπύλες N-Throughput των 2 υλοποιήσεων των 2 επιταχυντών με 100 και 150MHz ρολόι αντίστοιχα.



Διάγραμμα 8. 2 Διάγραμμα N-Throughput των υλοποιήσεων των δύο επιταχυντών στα 100 και 150MHz.

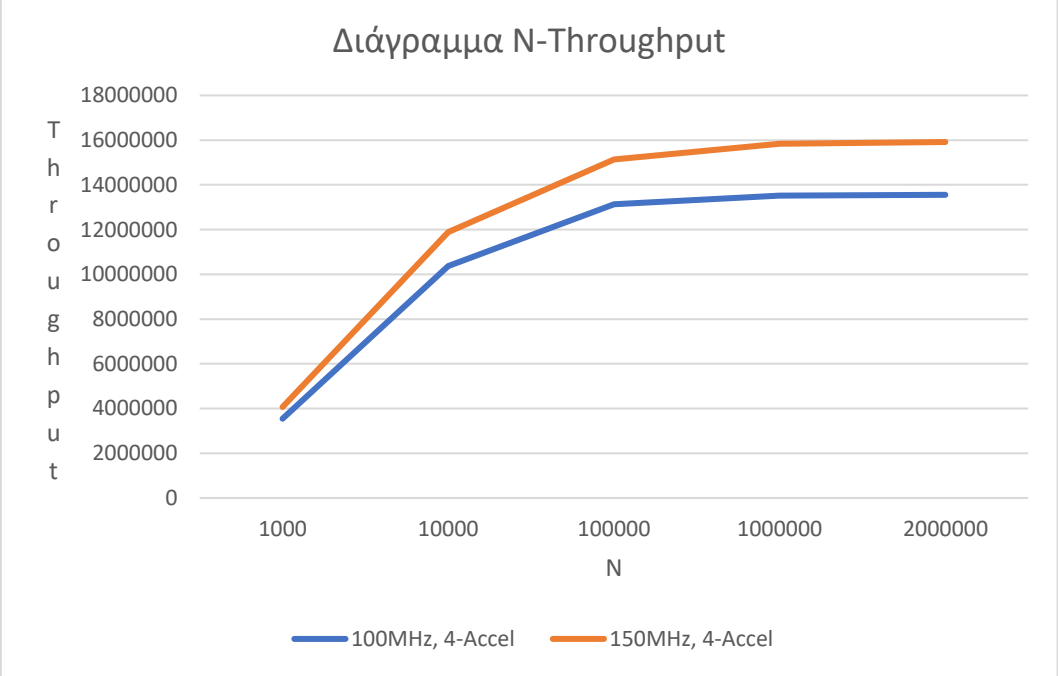
Όπως φαίνεται στο Διάγραμμα 8. 2, διακρίνεται και εδώ μία μερικώς αυξημένη καθυστέρηση της ομαλοποίησης του throughput από το N=10.000 έως το N=100.000 της υλοποίησης των 150MHz σε σύγκριση με την ομαλοποίηση της καμπύλης της υλοποίησης των 100MHz.

Μία ακόμη διαφορετική προσέγγιση, είναι οι υλοποιήσεις της παραγράφου 7.7.2.β των τεσσάρων επιταχυντών παράλληλα στα 100 και 150MHz ρολόι. Τα αποτελέσματα της υλοποίησης αυτής γίνονται διακριτά στον Πίνακας 8. 4.

N	**Χρόνος (sec) 100MHz**	**Χρόνος (sec) 150MHz**	**Throughput 100MHz (N/Sec)**	**Throughput 150MHz (N/Sec)**	**Αύξηση απόδοσης (%)**
1000	0,000282298	0,000245925	3542353,078	4066287,122	14,79%
10000	0,000964676	0,000840754	10366169,77	11894084,52	14,73%
100000	0,007614491	0,006607571	13132853,45	15134155,05	15,23%
1000000	0,073945604	0,063147645	13523454,29	15835903,41	17,09%
2000000	0,147538603	0,125652635	13555774,25	15916896,63	17,41%

Πίνακας 8. 4 Πίνακας χρόνου-throughput και αύξηση απόδοσης(%) μεταξύ των υλοποιήσεων των τεσσάρων επιταχυντών με 100 και 150MHz ρολόι.

Όπως διακρίνεται στον Πίνακας 8. 4, υπάρχει αύξηση του throughput σε σχέση με όλες τις προηγούμενες υλοποιήσεις (Πίνακας 8. 2 και Πίνακας 8. 3), εκτός της περίπτωσης των δεδομένων για N=1000. Στην περίπτωση των λίγων δεδομένων, υπάρχει υστέρηση στην απόδοση (μειωμένο throughput) ακόμα και στην απλή υλοποίηση του ενός επιταχυντή με προκαθορισμένο ρολόι στα 100MHz. Επίσης, παρατηρείται και εδώ αύξηση της απόδοσης της υλοποίησης των τεσσάρων επιταχυντών στα 150MHz σε σύγκριση με την υλοποίηση των τεσσάρων επιταχυντών στα 100MHz ρολόι. Η απόδοση κυμαίνεται από 14,73% έως 17,41%. Στο Διάγραμμα 8. 3 απεικονίζονται οι καμπύλες N-Throughput των 2 υλοποιήσεων.



Διάγραμμα 8. 3 Διάγραμμα N-Throughput των υλοποιήσεων των τεσσάρων επιταχυντών στα 100 και 150MHz.

Όπως φαίνεται στο Διάγραμμα 8. 3, η καθυστέρηση της ομαλοποίησης του throughput από το N=10.000 έως το N=1.000.000 για τα 150MHz είναι αυξημένη, σε σύγκριση με την υλοποίηση

των 100MHz. Ο Πίνακας 8. 5 παρουσιάζει την απόδοση (%) του συστήματος των τεσσάρων επιταχυντών με προκαθορισμένο ρολόι στα 100MHz, σε σύγκριση με τις υλοποιήσεις του ενός και δύο επιταχυντών στα 100MHz.

N	**1 Επιταχυντής – 100MHz**	**2 Επιταχυντές - 100MHz**
1000	-2,42%	-19,78%
10000	108,72%	31,17%
100000	155,55%	51,35%
1000000	161,99%	53,97%
2000000	162,59%	54,28%

Πίνακας 8. 5 Αυξομείωση απόδοσης μεταξύ της υλοποίησης των τεσσάρων επιταχυντών στα 100MHz με 1 επιταχυντή στα 100MHz και 2 επιταχυντές στα 100MHz.

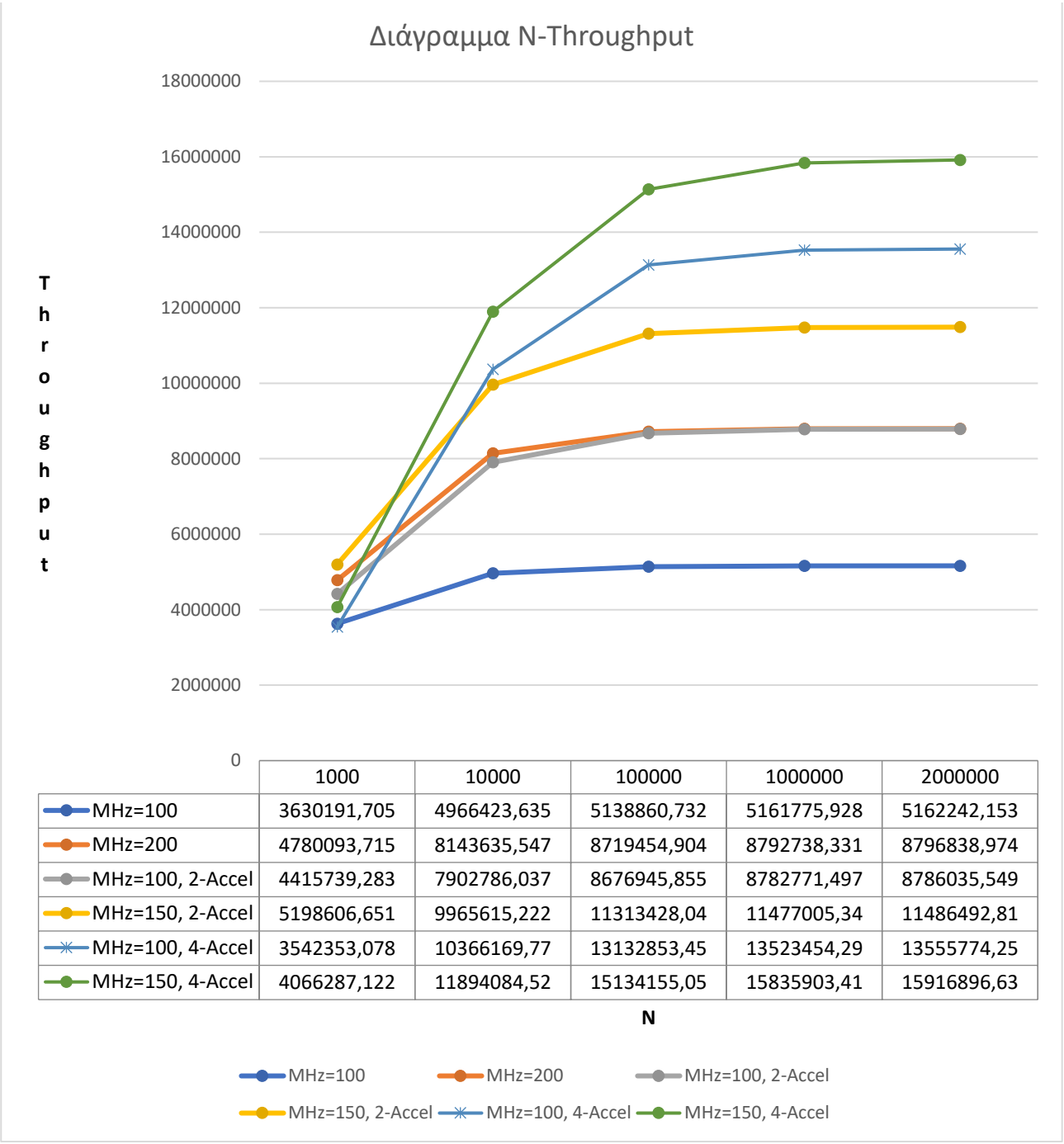
Όπως γίνεται διακριτό στον Πίνακας 8. 5, υπάρχει και στις 2 περιπτώσεις αύξηση της απόδοσης έως και 162,59% για την υλοποίηση των τεσσάρων επιταχυντών στα 100MHz για όλα τα N, εκτός της περίπτωσης των λίγων δεδομένων για N=1000 όπου παρατηρείται μείωση της απόδοσης. Η μείωση της απόδοσης για μικρά N είναι αναμενόμενη, διότι όπως έχει αναφερθεί, απαιτείται πρόσθετος χρόνος για το συγχρονισμό των επιταχυντών αναμεταξύ τους από το PU της FPGA. Παρ' ότι εκτελούνται δεδομένα παράλληλα σε 4 πανομοιότυπους επιταχυντές, η καθυστέρηση, λόγω συγχρονισμού, έχει ως αποτέλεσμα την αργοπορημένη έναρξη των πράξεων των επιταχυντών που καταλήγει σε μειωμένη απόδοση της υλοποίησης για μικρού όγκου δεδομένα. Στον Πίνακας 8. 6 φαίνεται η μέγιστη αύξηση της απόδοσης της υλοποίησης με το μεγαλύτερο throughput των τεσσάρων επιταχυντών στα 150MHz, σε σύγκριση με την πιο απλή υλοποίηση τους ενός επιταχυντή στα 100MHz.

N	**Απόδοση(%)**
1000	12,01%
10000	139,48%
100000	194,50%
1000000	206,79%
2000000	208,33%

Πίνακας 8. 6 Αύξηση της απόδοσης μεταξύ της γρηγορότερης υλοποίησης των τεσσάρων επιταχυντών στα 150MHz με την πιο απλή υλοποίηση του ενός επιταχυντή στα 100MHz.

Όπως διακρίνεται στον Πίνακας 8. 6, η απόδοση παρουσιάζει μικρή αύξηση για μικρό N. Ωστόσο, όσο αυξάνεται το N η απόδοση αυξάνεται φτάνοντας έως και 208,33%.

Επίσης παρατίθεται ένα συγκεντρωτικό διάγραμμα N-Throughput όλων των αποτελεσμάτων των υλοποιήσεων, ώστε να γίνει διακριτή η απόδοση των υλοποιήσεων αναμεταξύ τους. Το συγκεντρωτικό διάγραμμα είναι το Διάγραμμα 8. 4.



	1000	10000	100000	1000000	2000000
MHz=100	3630191,705	4966423,635	5138860,732	5161775,928	5162242,153
MHz=200	4780093,715	8143635,547	8719454,904	8792738,331	8796838,974
MHz=100, 2-Accel	4415739,283	7902786,037	8676945,855	8782771,497	8786035,549
MHz=150, 2-Accel	5198606,651	9965615,222	11313428,04	11477005,34	11486492,81
MHz=100, 4-Accel	3542353,078	10366169,77	13132853,45	13523454,29	13555774,25
MHz=150, 4-Accel	4066287,122	11894084,52	15134155,05	15835903,41	15916896,63

Διάγραμμα 8. 4 Συγκεντρωτικό διάγραμμα N-Throughput όλων των αποτελεσμάτων υλοποιήσεων του κεφαλαίου 7 μαζί με τις τιμές του Throughput για κάθε μία υλοποίηση για όλα τα N.

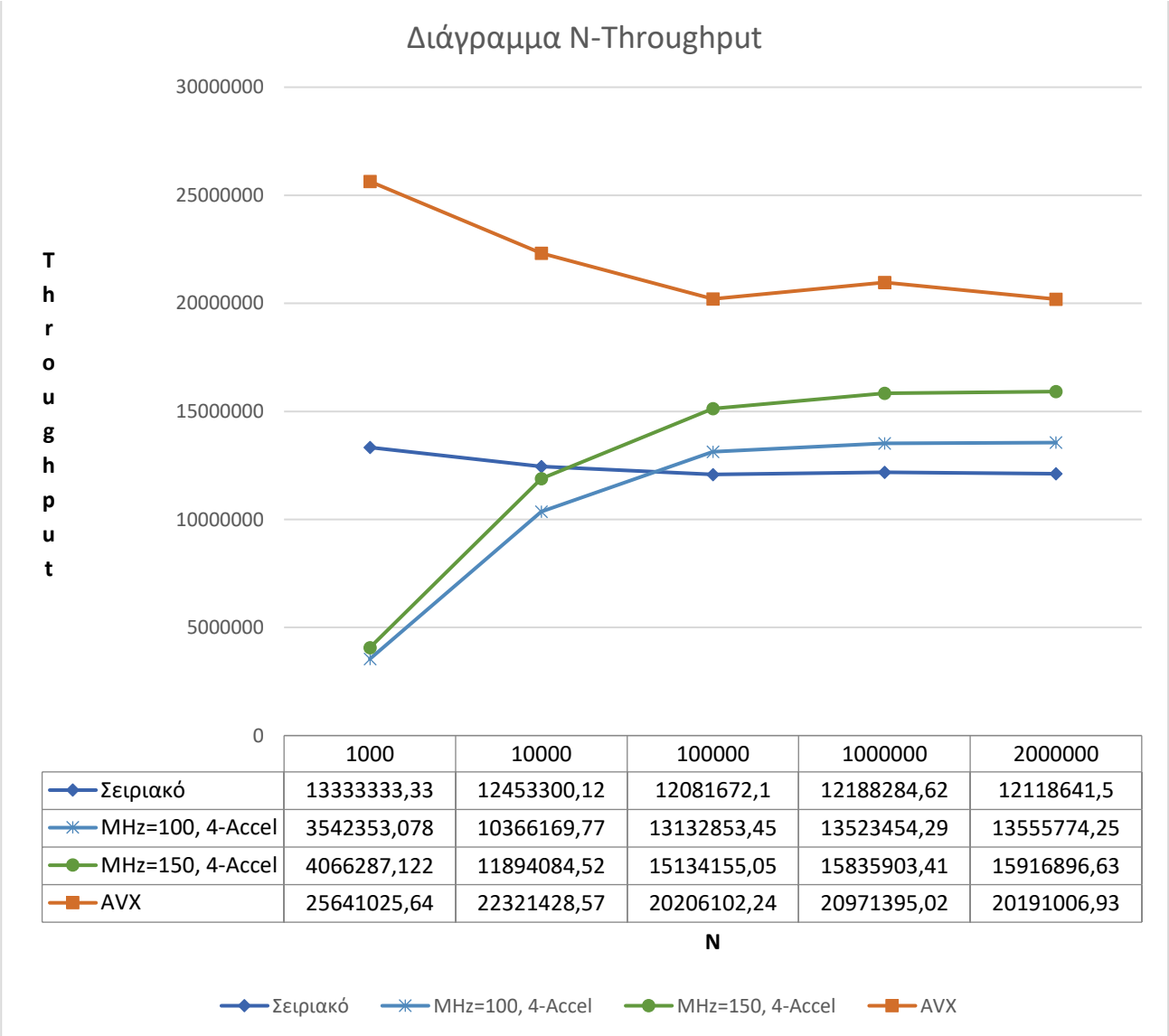
Τέλος, παρατίθεται ένας πίνακας που υπολογίζει το ποσοστό (%) του throughput για 100MHz ρολόι στην πράξη, σε σύγκριση με τη θεωρία. Η θεωρητική (ιδανική) τιμή του throughput, για μία υλοποίηση, υπολογίζεται από το ρολόι (σε Hz) διά την τιμή του interval (100 MHz/16 interval=6250000 Throughput N/Sec). Ο Πίνακας 8. 7 παρουσιάζει το ποσοστό (%) του πόσο κοντά φτάνει το throughput στην πράξη, για όλα τα N, σε σύγκριση με το throughput της θεωρίας για ρολόι 100MHz και interval 16.

N	Απόδοση (%)
1000	58,08%
10000	79,46%
100000	82,21%
1000000	82,58%
2000000	82,59%

Πίνακας 8. 7 Πίνακας απόδοσης ποσοστού (%) της θεωρητικής τιμής του Throughpu, στην πράξη.

8.3 Σύγκριση των αποτελεσμάτων των υλοποιήσεων του κεφαλαίου 7 με τα αποτελέσματα των υλοποιήσεων της παραγράφου 8.1.

Όπως γίνεται διακριτό από τον Πίνακας 8. 1 της παραγράφου 8.1 που περιέχει τα αποτελέσματα των υλοποιήσεων σε software, το throughput είναι αρκετά μεγάλο. Στην συνέχεια παρατίθεται το διάγραμμα Throughput-N των αποτελεσμάτων των υλοποιήσεων του software και των υλοποιήσεων των τεσσάρων επιταχυντών στα 100 και 200MHz (Διάγραμμα 8. 5).



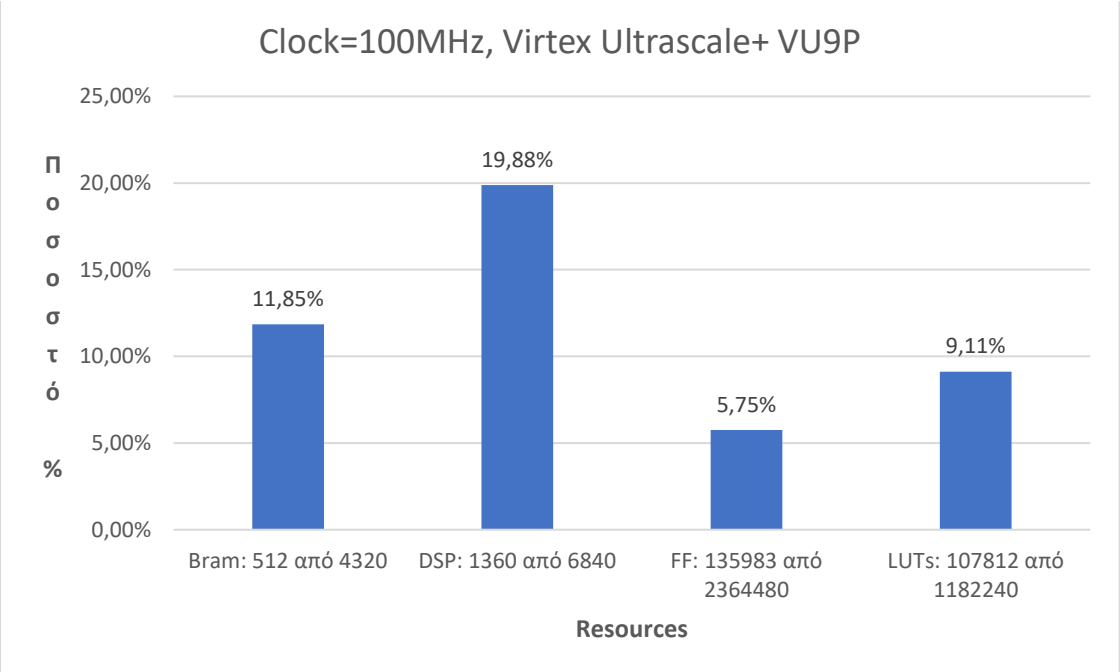
Διάγραμμα 8. 5 Διάγραμμα N-Throughput με τις υλοποιήσεις των τεσσάρων επιταχυντών στα 100 και 150MHz ρολόι και τις software υλοποιήσεις.

Στο Διάγραμμα 8. 5 παρατηρείται πως, μόνο οι δύο υλοποιήσεις των τεσσάρων επιταχυντών με 100 και 150MHz καταφέρνουν να υπερβούν σε απόδοση την σειριακή υλοποίηση του software. Καμία υλοποίηση του κεφαλαίου 7 δεν μπορεί να ξεπεράσει το throughput της AVX software υλοποίησης. Το κύριο πρόβλημα, για το οποίο οι υλοποιήσεις της FPGA υστερούν σε απόδοση, είναι ο περιορισμός του πλάτους των Memory Ports, όπως αναλύθηκε στην παράγραφο 7.4.2.β. Όπως, επίσης, έγινε γνωστό στο κεφάλαιο 7, λόγω περιορισμού του πλάτους των Memory Ports, η τιμή του interval περιορίζεται στο 16. Σε ένα διαφορετικό σενάριο, όπου το πλάτος των Memory Ports είναι αρκετά μεγαλύτερο των 64bits, το interval θα είχε την δυνατότητα να μειωθεί. Μία τέτοια προσέγγιση μπορεί να γίνει εφικτή χρησιμοποιώντας τις Virtex Ultrascale+ FPGAs της Xillinx που είναι διαθέσιμες στο F1 Amazon. Οι συγκεκριμένες FPGAs διακρίνονται για τα πολλά resources τους, το υψηλό Memory Bandwidth και το πλάτος των Memory Ports τους που είναι τα 1024bits, θεωρητικά. Θεωρητικό πλάτος 1024bits σημαίνει πως, υπάρχει η δυνατότητα μεγέθους πλάτους των 512bits στην πράξη. Αυτό σημαίνει πως, το interval μπορεί να μειωθεί από το νούμερο 16 στο νούμερο 2. Το θεωρητικό throughput με ρολόι 100MHz και interval 2 είναι τα 50.000.000 N/Sec. Ακόμη και στην περίπτωση που το throughput στην πράξη πιάσει το 80% του θεωρητικού throughput (40.000.000 N/Sec), η διαφορά στην απόδοση είναι αρκετά μεγάλη. Στον Πίνακας 8. 8 παρουσιάζεται η αυξημένη απόδοση (%) του 80% του θεωρητικού throughput της Virtex Ultrascale+ FPGA, σε σύγκριση με τα throughputs της software υλοποίησης.

N	Απόδοση(%) SeQ	Απόδοση(%) AVX
1000	200%	56%
10000	221%	79%
100000	231%	98%
1000000	282%	90%
2000000	230%	98%

Πίνακας 8. 8 Πίνακας αύξησης απόδοσης μεταξύ του throughput της Virtex FPGA με τις υλοποιήσεις του software (Σειριακή και AVX).

Όπως διακρίνεται στον Πίνακας 8. 8, η χρήση της Virtex FPGA παρουσιάζει σχεδόν διπλάσιο throughput από το throughput της AVX υλοποίησης, ακόμα και στην περίπτωση του 80% του θεωρητικού throughput. Επιπρόσθετα, κατασκευάστηκε ένας ακόμη επιταχυντής στο Vivado HLS 2017.4, χρησιμοποιώντας ως FPGA τη Virtex Ultrascale+ VU9P. Η αρχιτεκτονική σχεδίαση του επιταχυντή τροποποιήθηκε για να επιτευχθεί interval ίσο με 2. Τα resources της αρχιτεκτονικής αυτής γίνονται διακριτά στο Εικόνα 8. 1.



Εικόνα 8. 1 Resources της Virtex Ultrascale+ VU9P για αρχιτεκτονική σχεδίαση με interval ίσο με 2.

Όπως διακρίνεται στην Εικόνα 8. 1, τα resources δεν ξεπερνούν το 20%. Ωστόσο παρουσιάζεται μία σημαντική αύξηση των resources στα DSPs. Η ενέργεια αυτή είναι λογική, διότι πρέπει να εκτελεστούν παραπάνω πράξεις παράλληλα με interval 2, σε σύγκριση με τις πράξεις που πρέπει να εκτελεστούν παράλληλα με interval 16. Τέλος, το βάθος της pipelined αρχιτεκτονικής σχεδίασης με Interval 2 είναι οι 150 κύκλοι.

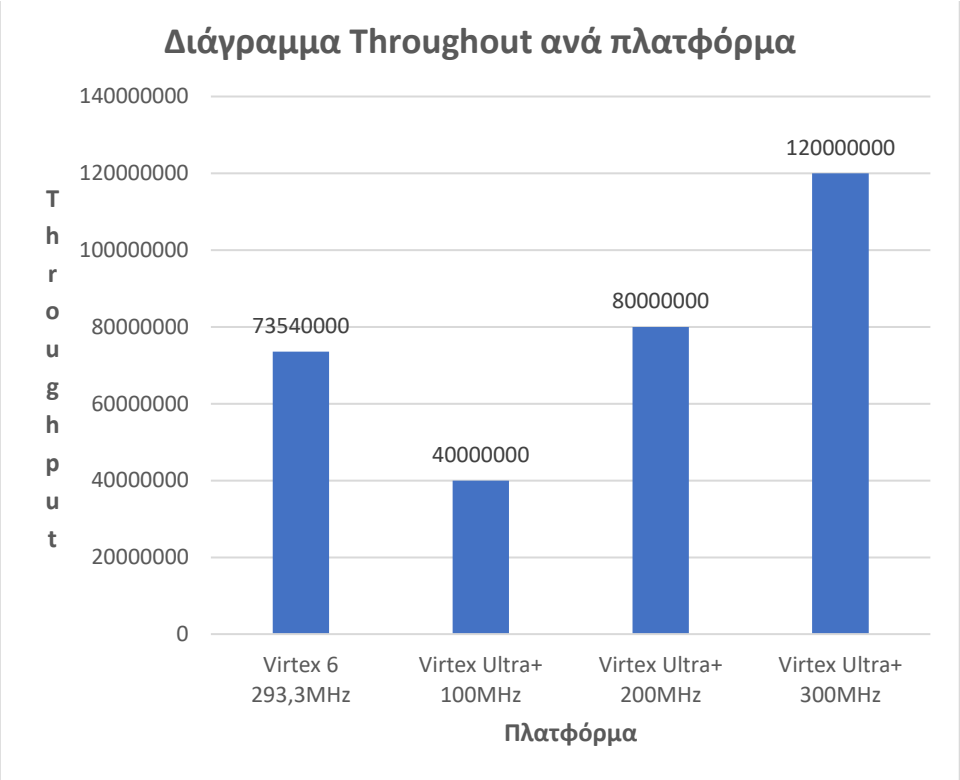
8.3.1 Σχετική μελέτη πάνω στην επιτάχυνση της συνάρτησης φυλογενετικής πιθανοφάνειας και σύγκριση αποτελεσμάτων

Η σχετική μελέτη των περισσότερων επιταχύνσεων της συνάρτησης της φυλογενετικής πιθανοφάνειας ή αλλιώς Phylogenetic Likelihood Function (PLF), αφορούν κυρίως υλοποιήσεις πάνω στην οικογένεια Virtex της Xilinx, χρησιμοποιώντας FPGAs. Όπως έχει αναφερθεί, οι Virtex FPGAs διαθέτουν μεγάλο πλάτος των Memory Ports, γεγονός που αυξάνει το Throughput και συνάμα την απόδοση και το Speed-up των υλοποιήσεων, σε σύγκριση με γρήγορες επεξεργαστικές μονάδες ηλεκτρονικών υπολογιστών τελευταίας τεχνολογίας.

Χρησιμοποιώντας προσαρμοσμένη αρχιτεκτονική σχεδίαση πάνω στη Virtex 5 SX240T FPGA της Xilinx με ρολόι τα 101MHz, οι Alachiotis et al. [26] μείωσαν, σε σύγκριση με βελτιστοποιημένες υλοποιήσεις, τον συνολικό χρόνο εκτέλεσης του PLF σε πολλαπλά Threads ενός υπολογιστικού συστήματος υψηλής απόδοσης με 8 διπύρηνους AMD Opteron επεξεργαστές στα 2.6GHz και 64GB κύριας μνήμης. Η προσαρμοσμένη υλοποίηση της FPGA χρειάστηκε το λιγότερο χρόνο. Ξεπέρασε, δηλαδή, σε απόδοση ένα υπολογιστικό σύστημα μεγάλης κλίμακας με βελτιστοποιημένες υλοποιήσεις του προγράμματος υπολογισμού του PLF [26].

Μία άλλη προσέγγιση με διαφορετική αρχιτεκτονική σχεδίαση πάνω στη Virtex 5 SX95T-2 FPGA της Xilinx με ρολόι 293.3MHz, πραγματοποίησαν οι Berger et al. [27] Η σύγκριση αποτελεσμάτων έγινε με την εκτέλεση του RAxML με AVX πάνω στον Intel i7 2600 επεξεργαστή στα 3.4 GHz. Τα αποτελέσματα έδειξαν πως, η AVX υλοποίηση παρουσίασε ελαφρά μεγαλύτερο Throughput από την υλοποίηση στην FPGA (78.830.000N/sec έναντι 73.540.000N/sec). Ωστόσο, με την εισαγωγή περισσότερων πανομοιότυπων επιταχυντών για παράλληλη εκτέλεση, το Throughput για την FPGA μπορεί να παρουσιάσει περεταίρω βελτίωση. Η συγκεκριμένη έρευνα δείχνει πως, σε καινούργιες και τεχνολογικά εξελιγμένες FPGAs μπορούν να υπάρξουν μέχρι και 8 επιταχυντές-πυρήνες για παράλληλη εκτέλεση με ρολόι τα 167.78MHz. Εκτιμάται πως, η μέγιστη απόδοση εκτέλεσης 8 παράλληλων επιταχυντών-πυρήνων σε μία Virtex 6 FPGA μπορεί να φτάσει έως και τα 335.570.000N/sec (42.000.000N/sec ο κάθε επιταχυντής) [27].

Οι παραπάνω μελέτες αποδεικνύουν πως, ο περιορισμός του υλικού και ιδιαίτερα της περιφερειακής μνήμης (Bandwidth και πλάτος των Memory Ports), παίζουν σημαντικό ρόλο στη συνολική απόδοση των πλατφόρμων που πραγματεύονται επιταχυντικά κυκλώματα. Χαρακτηριστικό παράδειγμα αποτελεί η μοντελοποίηση της Virtex Ultrascale+ VU9P FPGA που πραγματεύεται η παρούσα διπλωματική εργασία. Η συγκεκριμένη FPGA μοντελοποιήθηκε στην πιο απλή περίπτωση του ενός επιταχυντή στα 100MHz ρολόι. Η ικανότητα της Virtex Ultrascale+ VU9P FPGA για Throughput περίπου στα 40.000.000N/sec σε σύγκριση με το Throughput πάνω στην παλαιότερης τεχνολογίας Virtex 5 FPGA με σχεδόν 3x ρολόι, είναι διακριτή, όπως έδειξε η μελέτη των Berger et al [27]. Στο διάγραμμα της Εικόνα 8. 2 παρατίθεται η σύγκριση των διαφόρων προσεγγίσεων, καθώς και ορισμένα σενάρια πάνω στη μοντελοποίηση της Virtex Ultrascale+ VU9P FPGA.



Εικόνα 8. 2 Διάγραμμα Throughput-Πλατφόρμα χρησιμοποιώντας Virtex FPGAs και πραγματοποιώντας διάφορες προσεγγίσεις

Όπως παρατηρείται στο διάγραμμα της Εικόνα 8. 2, η καινούργια και εξελιγμένης τεχνολογίας Virtex Ultrascale+ VU9P FPGA παρουσιάζει αύξηση του Throughput και συνάμα της απόδοσης για την επιτάχυνση της μεθόδου της μέγιστης πιθανοφάνειας, σε σύγκριση με την παλαιότερης τεχνολογίας Virtex 6 FPGA. Χαρακτηριστικό παράδειγμα βελτίωσης της απόδοσης είναι πως, με διπλασιασμό και τριπλασιασμό του ρολογιού της Virtex Ultrascale+ VU9P FPGA (σε ιδανικές συνθήκεςη Virtex 6 FPGA υστερεί στο Throughput N/sec.

8.4 Ο επιταχυντής της μεθόδου της μέγιστης πιθανοφάνειας με τη χρήση του ReFiRe Framework για την κλήση απομακρυσμένων επιταχυντών.

Για να πραγματοποιηθούν οι προσεγγίσεις πρέπει πρώτα να οριστούν οι παράμετροι αυτών, ώστε να υπάρξει ευκαμψία, με απώτερο σκοπό την αναζήτηση της βέλτιστης μοντελοποίησης που παρουσιάζει τη μεγαλύτερη απόδοση του συστήματος. Οι 3 παράμετροι είναι οι εξής:

- Ο αριθμός των WORKER-FPGA που θα χρησιμοποιηθούν
- Ο αριθμός των ενεργών επιταχυντών σε κάθε WORKER-FPGA για παράλληλη εκτέλεση
- Το μέγεθος των δεδομένων προς επεξεργασία, δηλαδή το N.

Οι 3 παραπάνω παράμετροι σχετίζονται άμεσα με την επεκτασιμότητα (scalability) [28] της πλατφόρμας των FPGAs με αποκεντρωμένους πόρους. Με άλλα λόγια, η ικανότητα της πλατφόρμας για επέκταση εξαρτάται από τις παραμέτρους της ίδιας της πλατφόρμας. Η παράμετρος N τίθεται στα 2.000.000.000 για όλες τις μοντελοποιήσεις των πλατφορμών. Το ρολόι λαμβάνει την προκαθορισμένη τιμή του στα 100MHz και οι FPGAs δεν ξεπερνούν τις 1000. Στον Πίνακας 8. 9 δίνεται η μοντελοποίηση για την απλή περίπτωση του ενός επιταχυντή ανά WORKER-FPGA. Ο αριθμός των WORKER-FPGA ποικίλει σε εύρος από 1 έως 1000, με κάθε σενάριο να χρειάζεται διαφορετικό χρόνο για συγχρονισμό (SynQ) με την HOST-FPGA.

Workers	Χρόνος για SynQ (Sec)	N	Χρόνος για N (Sec)	Συνολικό Throughput (N/Sec)
1	4,56228E-06	2000000000	387,4285516	5162242,092
10	2,54175E-05	200000000	38,74285516	51622387,66
50	0,000126727	40000000	7,748571031	258107886,3
100	0,000258869	20000000	3,874285516	516189724,9
200	0,000510683	10000000	1,937142758	1032176321
300	0,000767623	6666667	1,291428505	1547752664
500	0,001278562	4000000	0,774857103	2576869082
1000	0,002606114	2000000	0,387428552	5127749347

Πίνακας 8. 9 Αποτελέσματα συνολικού χρόνου και Throughput για το σενάριο του ενός επιταχυντή ανά FPGA, με φόρτο εργασίας 2.000.000.000.

Όπως διακρίνεται στον Πίνακας 8. 9, η στήλη Workers αντιπροσωπεύει τον αριθμό των WORKER-FPGAs που μοντελοποιούνται. Η στήλη Χρόνος για SynQ, είναι ο χρόνος που απαιτείται για συγχρονισμό των Workers με τη HOST-FPGA. Το N είναι το μέγεθος των N που διαχειρίζεται ο κάθε Worker απομακρυσμένα και αποκεντρωμένα. Η στήλη Χρόνος για N, αντιπροσωπεύει το χρόνο που απαιτείται για την ολοκλήρωση των πράξεων του συνολικού φόρτου εργασίας (N=2.000.000.000), σύμφωνα με τον αριθμό των διαθέσιμων Workers. Η τελευταία στήλη υπολογίζει το συνολικό Throughput (N/Sec) για όλα τα σενάρια των Workers, συμπεριλαμβάνοντας το χρόνο για συγχρονισμό και το χρόνο για την ολοκλήρωση του N. Παρατηρείται πως η επεκτασιμότητα αυξάνεται όταν προθέτονται περισσότεροι Workers για την εκτέλεση του συνολικού φόρτου εργασίας. Το ReFiRe Framework παρουσιάζει χαμηλούς χρόνους για το συγχρονισμό των Workers με την HOST-FPGA. Χαρακτηριστικό παράδειγμα αποτελεί η περίπτωση του χρόνου για συγχρονισμό 1000 Workers που είναι μόλις τα 2,6 mSec. Η χρήση του ReFiRe Framework, η διάταξη της μνήμης και ο συγχρονισμός του κεφαλαίου 6, επιφέρουν ικανοποιητική επεκτασιμότητα στην αντιμετώπιση του συνολικού φόρτου εργασίας. Όπως φαίνεται στον Πίνακας 8. 9, το συνολικό Throughput για το συνολικό φόρτο εργασίας φτάνει έως και 5,12 δις-N το δευτερόλεπτο. Στον Πίνακας 8. 10, παρατίθεται η μοντελοποίηση με παραμέτρους ίδιες με τη μοντελοποίηση του Πίνακας 8. 9. Η μόνη διαφορά τους είναι πως, στη μοντελοποίηση του Πίνακας 8. 10 κάθε Worker περιέχει δύο επιταχυντές αντί για έναν.

Workers	Χρόνος για SynQ (Sec)	N	Χρόνος για N (Sec)	Συνολικό Throughput (N/Sec)
1	4,56228E-06	2000000000	227,6339526	8786035,373
10	2,54175E-05	200000000	22,76339526	87860257,38
50	0,000126727	40000000	4,552679053	439289549,5
100	0,000258869	20000000	2,276339526	878503650
200	0,000510683	10000000	1,138169763	1756419026
300	0,000767623	6666667	0,758779842	2633146829
500	0,001278562	4000000	0,455267905	4380715090
1000	0,002606114	2000000	0,227633953	8686585408

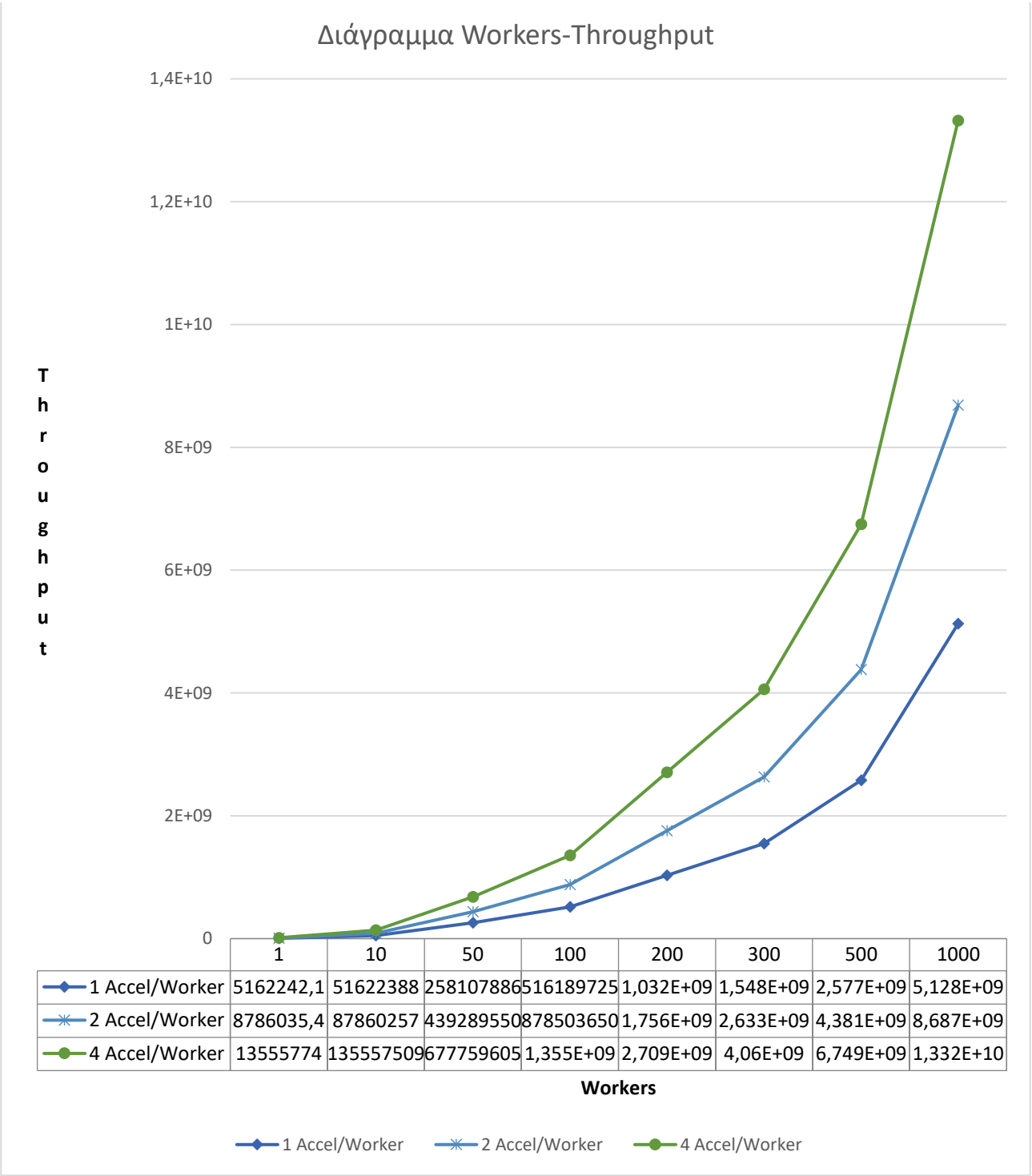
Πίνακας 8. 10 Αποτελέσματα συνολικού χρόνου και Throughput για το σενάριο των δύο επιταχυντών ανά FPGA, με φόρτο εργασίας 2.000.000.000.

Όπως διακρίνεται στον Πίνακας 8. 10, ο χρόνος που απαιτείται για την ολοκλήρωση του συνολικού φόρτου εργασίας, συμπεριλαμβανομένου και του χρόνου για συγχρονισμό των 2 επιταχυντών από τον κάθε Worker ξεχωριστά, μειώνεται σε σύγκριση με το χρόνο της μοντελοποίησης με έναν επιταχυντή του Πίνακας 8. 9. Αυτό συμβαίνει, λόγω της διάταξης της μνήμης και το συγχρονισμό των επιταχυντών του κάθε Worker που αναλύθηκαν εκτενώς στο κεφάλαιο 6. Ακόμη, το συνολικό Throughput του συνολικού φόρου εργασίας αυξάνεται έως και 70% σε σύγκριση με το συνολικό Throughput της μοντελοποίησης του Πίνακας 8. 9. Στον Πίνακας 8. 11, παρουσιάζεται η ίδια μοντελοποίηση των προηγουμένων 2, με 4, ωστόσο, επιταχυντές ανά Worker.

Workers	**Χρόνος για SynQ (Sec)**	**N**	**Χρόνος για N (Sec)**	**Συνολικό Throughput (N/Sec)**
1	4,56228E-06	2000000000	147,5386034	13555773,83
10	2,54175E-05	200000000	14,75386034	135557508,9
50	0,000126727	40000000	2,950772067	677759604,6
100	0,000258869	20000000	1,475386034	1355339619
200	0,000510683	10000000	0,737693017	2709279295
300	0,000767623	6666667	0,491795345	4060394571
500	0,001278562	4000000	0,295077207	6748645407
1000	0,002606114	2000000	0,147538603	13320481998

Πίνακας 8. 11 Αποτελέσματα συνολικού χρόνου και Throughput για το σενάριο των τεσσάρων επιταχυντών ανά FPGA, με φόρτο εργασίας 2.000.000.000.

Όπως διακρίνεται από την μοντελοποίηση του Πίνακας 8. 11, ο συνολικός χρόνος για όλο το φόρτο εργασίας μειώνεται περεταίρω, σε σύγκριση με τους χρόνους των μοντελοποιήσεων των πινάκων Πίνακας 8. 9 και Πίνακας 8. 10. Επιπρόσθετα, αυξάνεται και το συνολικό Throughput που κυμαίνεται, πλέον, έως και 13,32 δις-N το δευτερόλεπτο. Ακόμη, παρατίθεται ένα συγκεντρωτικό διάγραμμα Throughput-Workers όλων των μοντελοποιήσεων (Διάγραμμα 8. 6), ώστε να γίνει πιο διακριτή η αύξηση της επεκτασιμότητας πάνω στην αρχιτεκτονική σχεδίαση του κεφαλαίου 6.



	1	10	50	100	200	300	500	1000
1 Accel/Worker	5162242,1	51622388	258107886	516189725	1,032E+09	1,548E+09	2,577E+09	5,128E+09
2 Accel/Worker	8786035,4	87860257	439289550	878503650	1,756E+09	2,633E+09	4,381E+09	8,687E+09
4 Accel/Worker	13555774	135557509	677759605	1,355E+09	2,709E+09	4,06E+09	6,749E+09	1,332E+10

Διάγραμμα 8. 6 Συγκεντρωτικό διάγραμμα Throughput-Workers για όλα τα σενάρια του ενός, δύο και τεσσάρων επιταχυντών ανά Worker, με φόρτο εργασίας 2.000.000.000.

Όπως διακρίνεται στο Διάγραμμα 8. 6, το Throughput και συνάμα η απόδοση των μοντελοποιήσεων αυξάνονται διαρκώς για όλα τα σενάρια των διαθέσιμων Workers. Ωστόσο, το μέγεθος του φόρτου εργασίας έχει εξίσου σημαντικό ρόλο στη συνολική απόδοση και επεκτασιμότητα των μοντελοποιήσεων. Ο φόρτος εργασίας των παραπάνω μοντελοποιήσεων τέθηκε στα 2.000.000.000 N, όπως έχει αναφερθεί,. Στην περίπτωση όπου ο φόρτος εργασίας είναι μικρότερος, η απόδοση και η επεκτασιμότητα της μοντελοποιημένης πλατφόρμας αλλάζουν. Στους πίνακες Πίνακας 8. 12, Πίνακας 8. 13 και Πίνακας 8. 14, διακρίνεται η μοντελοποιημένη εκδοχή του ενός, δύο και τεσσάρων επιταχυντών ανά Worker με συνολικό φόρτο εργασίας N=1.000.000.

Workers	Χρόνος για SynQ (Sec)	N	Χρόνος για N (Sec)	Συνολικό Throughput (N/Sec)
1	4,56228E-06	1000000	0,193731773	5161654,374
10	2,54175E-05	100000	0,019459566	51321573,23
50	0,000126727	20000	0,004027043	240745181,6
100	0,000258869	10000	0,002013521	440065219,4
200	0,000510683	5000	0,001163249	597395970,7
300	0,000767623	3333	0,000918225	593173177,2
500	0,001278562	2000	0,000550935	546598186,4
1000	0,002606114	1000	0,000275468	347031693,9

Πίνακας 8. 12 Αποτελέσματα συνολικού χρόνου και Throughput για το σενάριο του ενός επιταχυντή ανά FPGA, με φόρτο εργασίας 1.000.000.

Workers	Χρόνος για SynQ (Sec)	N	Χρόνος για N (Sec)	Συνολικό Throughput (N/Sec)
1	4,56228E-06	1000000	0,113859276	8782419,591
10	2,54175E-05	100000	0,011524792	86578513,17
50	0,000126727	20000	0,002530753	376296354,1
100	0,000258869	10000	0,001265377	656062237
200	0,000510683	5000	0,000811785	756161631,8
300	0,000767623	3333	0,000754875	656815034,5
500	0,001278562	2000	0,000452925	577538042,4
1000	0,002606114	1000	0,000226463	353035505,2

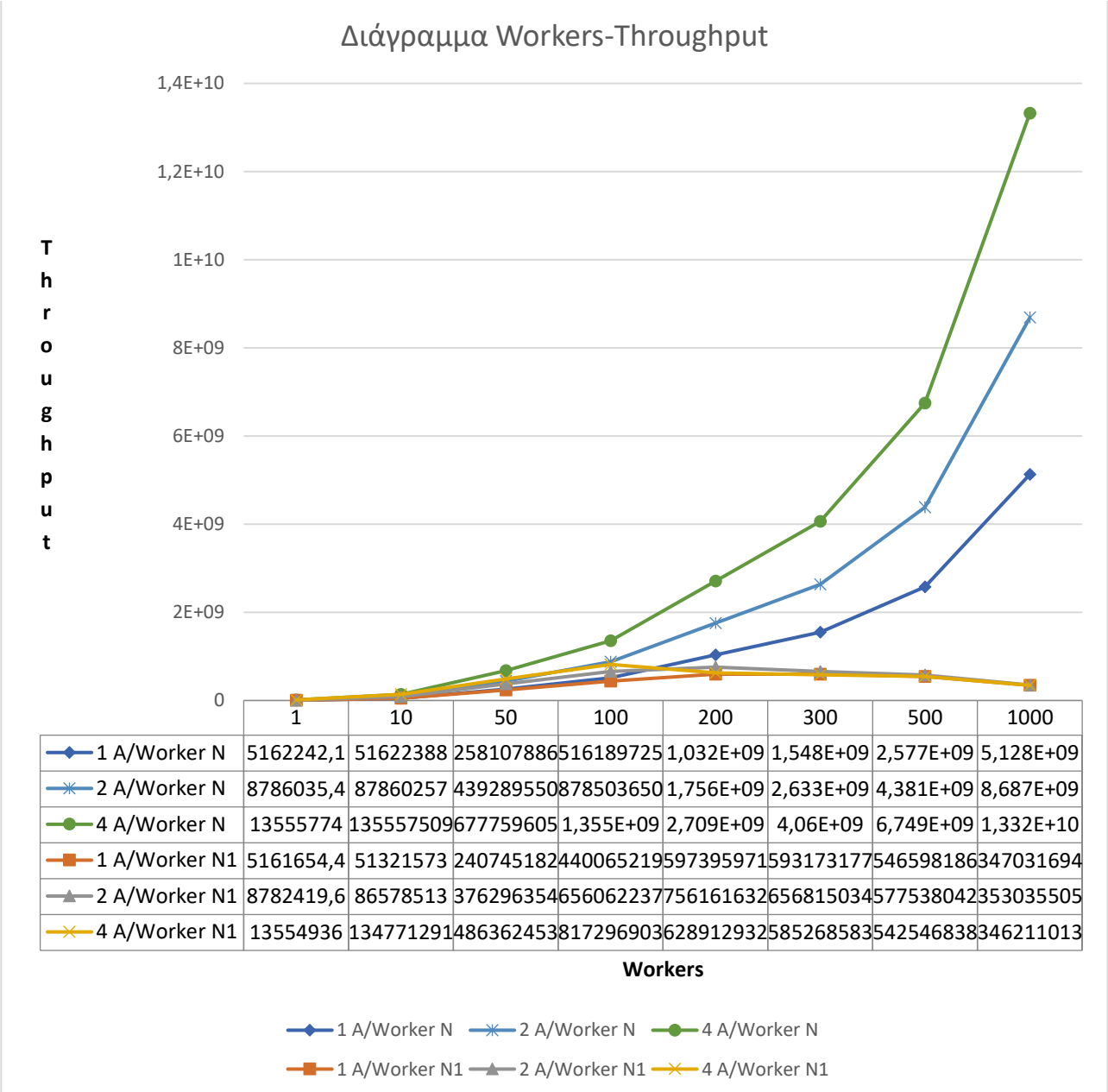
Πίνακας 8. 13 Αποτελέσματα συνολικού χρόνου και Throughput για το σενάριο των δύο επιταχυντών ανά FPGA, με φόρτο εργασίας 1.000.000.

Workers	Χρόνος για SynQ (Sec)	N	Χρόνος για N (Sec)	Συνολικό Throughput (N/Sec)
1	4,56228E-06	1000000	0,073769302	13554935,94
10	2,54175E-05	100000	0,00739456	134771291,1
50	0,000126727	20000	0,001929353	486362453
100	0,000258869	10000	0,000964676	817296902,6
200	0,000510683	5000	0,001079362	628912932,2
300	0,000767623	3333	0,000940994	585268582,7
500	0,001278562	2000	0,000564596	542546838,3
1000	0,002606114	1000	0,000282298	346211013,3

Πίνακας 8. 14 Αποτελέσματα συνολικού χρόνου και Throughput για το σενάριο των τεσσάρων επιταχυντών ανά FPGA, με φόρτο εργασίας 1.000.000.

Όπως διακρίνεται στις παραπάνω μοντελοποιήσεις με φόρτο εργασίας N=1.000.000, παρατηρείται, αρχικά, αύξηση του συνολικού Throughput ως ένα σημείο. Έπειτα, υπάρχει μείωση του συνολικού Throughput, πτώση, δηλαδή, της απόδοσης. Με την εισαγωγή περισσότερων Workers στο σημείο όπου, παρατηρείται για πρώτη φορά μείωση της απόδοσης και μετά, η απόδοση μειώνεται περαιτέρω. Στην περίπτωση του Πίνακας 8. 12, το συνολικό Throughput παρουσιάζει τη μέγιστη τιμή του, μοντελοποιώντας 200 Workers με N=5.000 ο καθένας. Επίσης, παρατηρείται πως, μέγιστη τιμή

Throughput παρουσιάζουν οι μοντελοποιήσεις του Πίνακας 8. 13 και Πίνακας 8. 14 μοντελοποιώντας 200 Workers με N=5.000 και 100 Workers με N=10.000 αντίστοιχα. Ο λόγος για τον οποίο παρουσιάζεται μείωση της απόδοσης, είναι η ανάθεση μικρού φόρτου εργασίας ανά Worker. Όπως αναλύθηκε στα αποτελέσματα των υλοποιήσεων στο κεφάλαιο 8.2, όλα τα Throughput ομαλοποιούνται όταν ο φόρτος εργασίας είναι μεγάλος. Αναλυτικότερα, όπως φαίνεται στο Διάγραμμα 8. 1, Διάγραμμα 8. 2 και Διάγραμμα 8. 3, το Throughput είναι χαμηλό για μικρό φόρτο εργασίας. Όσο αυξάνεται ο φόρτος εργασίας αυξάνεται και το Throughput, και από κάποιο σημείο και μετά παρουσιάζει μία σταθερή τιμή, ομαλοποιείται. Στις μοντελοποιήσεις με συνολικό φόρτο εργασίας 1.000.000, όσο αυξάνονται οι Workers, ανατίθεται όλο και μικρότερος φόρτος εργασίας ανά Worker. Μικρός φόρτος εργασίας ανά Worker σημαίνει πως, κάθε Worker προσφέρει χαμηλό Throughput στο συνολικό σύστημα κι έτσι μειώνεται το συνολικό Throughput της μοντελοποιημένης πλατφόρμας. Στο παρακάτω συγκεντρωτικό διάγραμμα (Διάγραμμα 8. 7), παρατίθενται τα συνολικά Throughputs για τις 3 μοντελοποιήσεις με φόρτο εργασίας N=2.000.000.000 και τις 3 μοντελοποιήσεις με φόρτο εργασίας N1=1.000.000, για συγκεκριμένους μοντελοποιημένους Workers.



	1	10	50	100	200	300	500	1000
1 A/Worker N	5162242,1	51622388	258107886	516189725	1,032E+09	1,548E+09	2,577E+09	5,128E+09
2 A/Worker N	8786035,4	87860257	439289550	878503650	1,756E+09	2,633E+09	4,381E+09	8,687E+09
4 A/Worker N	13555774	135557509	677759605	1,355E+09	2,709E+09	4,06E+09	6,749E+09	1,332E+10
1 A/Worker N1	5161654,4	51321573	240745182	440065219	597395971	593173177	546598186	347031694
2 A/Worker N1	8782419,6	86578513	376296354	656062237	756161632	656815034	577538042	353035505
4 A/Worker N1	13554936	134771291	486362453	817296903	628912932	585268583	542546838	346211013

Διάγραμμα 8. 7 Συγκεντρωτικό διάγραμμα Throughput-Workers για όλα τα σενάρια του ενός, δύο και τεσσάρων επιταχυντών ανά Worker, με φόρτο εργασίας 2.000.000.000 και 1.000.000.

Όπως παρατηρείται στο Διάγραμμα 8. 7, όσο μοντελοποιούνται περισσότεροι Workers, το συνολικό Throughput για συνολικό φόρτο εργασίας N=2.000.000.000 αυξάνεται σημαντικά. Αντίθετα, για φόρτο εργασίας N1=1.000.000 παρατηρείται μία μικρή αύξηση του συνολικού Throughput μέχρι ένα σημείο, απ' όπου και αρχίζει να μειώνεται με την είσοδο περισσότερων μοντελοποιημένων Workers. Γίνεται έτσι διακριτή η μεγάλη διαφορά της επεκτασιμότητας μεταξύ των 2 μοντελοποιημένων φόρτων εργασίας N και N1 πάνω στις ίδιες πλατφόρμες.

Κεφάλαιο 9: Συμπεράσματα και Μελλοντικές επεκτάσεις

Το ένατο κεφάλαιο παραθέτει τα συμπεράσματα της παρούσας διπλωματικής εργασίας, καθώς και πιθανές μελλοντικές επεκτάσεις αυτής.

9.1 Συμπεράσματα

Η παρούσα διπλωματική εργασία παρουσιάζει την κατασκευή ενός επιταχυντή με τη χρήση του Vivado HLS για την επιτάχυνση της μεθόδου της μέγιστης πιθανοφάνειας. Πραγματοποιώντας διάφορες υλοποιήσεις με τη χρήση του επιταχυντή (βλέπε Κεφάλαιο 7), τα αποτελέσματα των μετρήσεων (βλέπε Κεφάλαιο 8) έδειξαν το πόσο μεγάλο αντίκτυπο έχουν οι περιορισμοί του υλικού στην αποδοτικότητα των υλοποιήσεων. Ο μεγαλύτερος περιορισμός είναι το Bandwidth της μνήμης και ιδιαίτερα το πλάτος των Memory Ports στην FPGA. Πραγματοποιώντας μία μοντελοποίηση της Virtex Ultrascale+ VU9P FPGA που έχει Memory Ports με αισθητά μεγαλύτερο πλάτος από την ZCU102 Ultrascale MPSoC FPGA, παρατηρείται αύξηση της απόδοσης που ξεπερνά έως και στο διπλάσιο την απόδοση της AVX Software υλοποίησης του κεφαλαίου 8.1. Τέλος, γίνεται διακριτή η επεκτασιμότητα της μοντελοποίησης πλατφόρμων FPGAs με αποκεντρωμένους πόρους, χρησιμοποιώντας το ReFiRe Framework, τη διάταξη της μνήμης και το μηχανισμό συγχρονισμού, σύμφωνα με την αρχιτεκτονική σχεδίαση του κεφαλαίου 6.

9.2 Μελλοντικές επεκτάσεις

Όπως αναλύθηκε στην παρούσα διπλωματική εργασία, οι περιορισμοί του υλικού και ιδιαίτερα της μνήμης, εμποδίζουν την επεκτασιμότητα στην απόδοση, καθώς, λόγω του περιορισμένου Bandwidth της μνήμης και του πλάτους των Memory Ports, το interval παραμένει εγκλωβισμένο σε συγκεκριμένες τιμές. Ωστόσο, χρησιμοποιώντας τη Virtex Ultrascale+ VU9P FPGA που διαθέτει Memory Ports με μεγαλύτερο πλάτος, το interval καταφέρνει να περιοριστεί στο νούμερο 2. Η μείωση του interval από 16 σε 2 παρουσιάζει μεγάλη αύξηση του Throughput και συνάμα της αποδοτικότητας του επιταχυντή. Επομένως, μία από τις μεγαλύτερες επεκτάσεις που μπορούν να πραγματοποιηθούν για την επιτάχυνση της μεθόδου της μέγιστης πιθανοφάνειας είναι η καλύτερη δυνατή μείωση του interval.

Επιπρόσθετα, η αύξηση του ρολογιού, καθώς και των πανομοιότυπων επιταχυντών που εκτελούνται παράλληλα στην FPGA, μπορούν να συμβάλουν περεταίρω στην αύξηση της απόδοσης της επιτάχυνσης της μεθόδου της μέγιστης πιθανοφάνειας. Ακολούθως, παρατίθενται διάφορα σενάρια, ως παραδείγματα, των όσων ειπώθηκαν παραπάνω.

Παράδειγμα 1:

1 επιταχυντής, 100MHz ρολόι, interval=1 Throughput=100.000.000 N/Sec

Παράδειγμα 2:

1 επιταχυντής, 200MHz ρολόι, interval=1 Throughput=200.000.000 N/Sec

Παράδειγμα 3:

2 επιταχυντές, 100MHz ρολόι, interval=1 Throughput=200.000.000 N/Sec

Παράδειγμα 4:

2 επιταχυντές, 200MHz ρολόι, interval=1 Throughput=400.000.000 N/Sec

Σύμφωνα με τα παραπάνω ενδεικτικά παραδείγματα, γίνεται κατανοητή η ανάγκη για μικρό interval, μεγαλύτερο ρολόι, καθώς και περισσότεροι πανομοιότυποι επιταχυντές που εκτελούνται παράλληλα.

Η παραπάνω προσέγγιση θα μπορούσε να εφαρμοστεί σε μία πλατφόρμα με FPGAs από αποκεντρωμένους πόρους, όπως αναλύθηκε στην αρχιτεκτονική του κεφαλαίου 6. Μία τέτοια προσέγγιση θα μπορούσε να αυξήσει αρκετά την απόδοση της επιτάχυνσης της μεθόδου της μέγιστης πιθανοφάνειας, καθώς σε ένα τέτοιο σενάριο η επεκτασιμότητα θα είναι υψηλή, σύμφωνα με την αρχιτεκτονική σχεδίαση του κεφαλαίου 6.

Βιβλιογραφία

[1] V. Mayer-Schönberger and K. Cukier, *Big data: A revolution that will transform how we live, work, and think*. Houghton Mifflin Harcourt, 2013.

[2] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mob. networks Appl.*, vol. 19, no. 2, pp. 171–209, 2014.

[3] V.-H. Xilinx, "Vivado design suite user guide-high-level synthesis." UG902, 2014.

[4] A. Stamatakis, T. Ludwig, and H. Meier, "RAxML-III: a fast program for maximum likelihood-based inference of large phylogenetic trees," *Bioinformatics*, vol. 21, no. 4, pp. 456–463, 2005.

[5] D. R. Frost *et al.*, "The amphibian tree of life," *Bull. Am. Museum Nat. Hist.*, vol. 2006, no. 297, pp. 1–291, 2006.

[6] S. Guindon, J.-F. Dufayard, V. Lefort, M. Anisimova, W. Hordijk, and O. Gascuel, "New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of PhyML 3.0," *Syst. Biol.*, vol. 59, no. 3, pp. 307–321, 2010.

[7] D. J. Zwickl, "GARLI. Genetic algorithm approaches for the phylogenetic analysis of large biological sequence datasets under the maximum likelihood criterion," *Austin Univ. Texas Austin*, 2006.

[8] A. Stamatakis, "RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models," *Bioinformatics*, vol. 22, no. 21, pp. 2688–2690, 2006.

[9] I. J. Myung, "Tutorial on maximum likelihood estimation," *J. Math. Psychol.*, vol. 47, no. 1, pp. 90–100, 2003.

[10] "Nucleotide - Wikipedia." https://en.wikipedia.org/wiki/Nucleotide (accessed Jun. 11, 2020).

[11] W. R. Gilks, "M arkov Chain M onte C arlo," *Encycl. Biostat.*, vol. 4, 2005.

[12] P. J. Waddell and M. A. Steel, "General time reversible distances with unequal rates across sites," 1996.

[13] M. Ott, "Inference of Large Phylogenetic Trees on Parallel Architectures," Technische Universität München, 2010.

[14] J. Felsenstein, "Evolutionary trees from gene frequencies and quantitative characters: finding maximum likelihood estimates," *Evolution (N. Y).*, vol. 35, no. 6, pp. 1229–1242, 1981.

[15] J. Felsenstein, "Evolutionary trees from DNA sequences: a maximum likelihood approach," *J. Mol. Evol.*, vol. 17, no. 6, pp. 368–376, 1981.

[16] Z. Yang, "Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods," *J. Mol. Evol.*, vol. 39, no. 3, pp. 306–314, 1994.

[17] A. Stamatakis and M. Ott, "Exploiting fine-grained parallelism in the phylogenetic likelihood function with MPI, Pthreads, and OpenMP: A performance study," in *IAPR International Conference on Pattern Recognition in Bioinformatics*, 2008, pp. 424–435.

[18] N. Alachiotis, D. Theodoropoulos, and D. Pnevmatikatos, "Versatile deployment of FPGA accelerators in disaggregated data centers: A bioinformatics case study," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, 2017, pp. 1–4.

[19] K. Katrinis *et al.*, "Rack-scale disaggregated cloud data centers: The dReDBox project vision," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2016, pp. 690–695.

[20] J. Weerasinghe, F. Abel, C. Hagleitner, and A. Herkersdorf, "Disaggregated fpgas: Network performance comparison against bare-metal servers, virtual machines and linux containers," in *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2016, pp. 9–17.

[21] G. M. Saridis *et al.*, "EVROS: All-optical programmable disaggregated data centre interconnect utilizing hollow-core bandgap fibre," in *2015 European Conference on Optical Communication (ECOC)*, 2015, pp. 1–3.

[22] D. Theodoropoulos, N. Alachiotis, and D. Pnevmatikatos, “FPGA-based evaluation platform for disaggregated computing,” in *Proceedings - 2017 17th International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation, SAMOS 2017*, Apr. 2018, vol. 2018-Janua, pp. 129–136, doi: 10.1109/SAMOS.2017.8344620.
[23] D. Theodoropoulos, A. Reale, D. Syrivelis, M. Bielski, N. Alachiotis, and D. Pnevmatikatos, “REMAP: Remote mEmory Manager for disAggregated Platforms,” in *Proceedings of the International Conference on Application-Specific Systems, Architectures and Processors*, Aug. 2018, vol. 2018-July, doi: 10.1109/ASAP.2018.8445095.
[24] E. Pissadakis, N. Alachiotis, P. Skrimponis, D. Theodoropoulos, T. Korakis, and D. Pnevmatikatos, “ReFiRe: Efficient Deployment of Remote Fine-Grained Reconfigurable Accelerators,” in *2018 International Conference on Field-Programmable Technology (FPT)*, 2018, pp. 322–325.
[25] E. Xilinx, “SDSoC Environment UserGuide, Jan. 2018.” .
[26] N. Alachiotis, A. Stamatakis, E. Sotiriades, and A. Dollas, “A reconfigurable architecture for the phylogenetic likelihood function,” in *2009 International Conference on Field Programmable Logic and Applications*, 2009, pp. 674–678.
[27] S. A. Berger, N. Alachiotis, and A. Stamatakis, “An optimized reconfigurable system for computing the phylogenetic likelihood function on dna data,” in *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, 2012, pp. 352–359.
[28] “Scalability - Wikipedia.” https://en.wikipedia.org/wiki/Scalability (accessed Jun. 15, 2020).