"Σχεδίαση και Υλοποίηση σε Γλώσσα Περιγραφής Υλικού VHDL του Αλγορίθμου BLASTp"

από τον
Σπύρο Μαρκόπουλο

για τη μερική εκπλήρωση των απαιτήσεων για το
δίπλωμα του
Ηλεκτρονικού Μηχανικού και Μηχανικού Υπολογιστών

στο Πολυτεχνείο της Κρήτης
Τμήμα Ηλεκτρονικών Μηχανικών και Μηχανικών
Υπολογιστών
January 31, 2010

Εξεταστική Επιτροπή:
Καθηγητής Πνευματικάτος Διονύσιος (Επιβλέπων)
Καθηγητής Δόλλας Απόστολος
Επίκουρος Καθηγητής Παπαευσταθίου Ιωάννης

Γενικά

Πολλές φορές στο χώρο της Βιολογίας υπάρχει η ανάγκη να εξακριβωθεί εάν μία ακολουθία από "σκόρπια" αμινοξέα συνθέτει μία πρωτεΐνη, ένα κομμάτι DNA ή είναι απλά μία άχρηστη ακολουθία. Δημιουργούνται λοιπόν νέα ερωτήματα που αφορούν στις μεθόδους εξακρίβωσης που πρέπει να ακολουθηθούν, αλλά και ποια εργαλεία είναι απαραίτητο να χρησιμοποιηθούν.

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω τον καθηγητή μου Διονύσιο Πνευματικάτο για την πολύτιμη καθοδήγηση του, καθώς και την εξεταστική επιτροπή που συναποτελείται από τους κ. Απόστολο Δόλλα και κ. Ιώαννη Παπευσταυθίου.

Επίσης θα ήθελα να ευχαριστήσω τον διδακτορικό βοηθό Ευριπιδή Σωτηριάδη που ήταν πάντα διαθέσιμος για μένα. Λόγω της εμπειρίας του στον αλγόριθμο blast μου έλυσε πολλές απορίες. Θα ήθελα επίσης να πω ευχαριστώ στα παιδιά του εργαστηρίου, που όποτε χρειάστηκε, με βοήθησαν. Ιδιαίτερα θα ήθελα να ευχαριστήσω τον Ιωάννη Μακρή για την παρέα και τις συμβουλές του, που χωρίς αυτές η περάττωση της διπλωματικές θα είχε λάβει μεγαλύτερο χρονικό διάστημα.

Τέλος θα ήθελα να ευχαριστήσω την οικογενειά μου για την στήριξη κατά τη διάρκεια της διπλωματικής εργασίας αλλά και την υπομόνη που είχαν όλα αυτά τα χρόνια.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Εισαγωγή στο Πρόβλημα

Πολλές φορές στο χώρο της Βιολογίας υπάρχει η ανάγκη να εξακριβωθεί εάν μία ακολουθία από "σκόρπια αμινοξέα" συνθέτει μία πρωτεΐνη, ένα κομμάτι DNA ή είναι απλά μία αυθαίρετη ακολουθία. Δημιουργούνται λοιπόν με τη σειρά τους νέα υποπροβλήματα που αφορούν κυρίως στις τεχνικές που πρέπει να ακολουθηθούν ώστε να λάβουν οι βιολόγοι τα αποτελέσματα που αναζητούν. Τέτοια ερωτήματα αφορούν στη στοίχιση ακολουθιών από πρωτεΐνες ή DNA. Είναι ένα πολύ σημαντικό κομμάτι για την επιστήμη της βιοπληροφορικής, αφού ο τρόπος στοίχισης δύο ακολουθιών οδηγεί συνήθως και στην ανάλογη χρήση μεθόδου αναζήτησης ομοιότητας. Πέρα όμως από τη σύγκριση μίας υπό εξέταση ακολουθίας με μία υπάρχουσα βάση δεδομένων, υπάρχει η ανάγκη να πραγματοποιηθεί αυτή με ένα γρήγορο αλλά και ακριβή αλγόριθμο. Η επιλογή του αλγορίθμου είναι μείζονος σημασίας, από τη στιγμή που κάθε αλγόριθμος έχει τα δικά του χαρακτηριστικά και πλεονεκτήματα. Συνεπώς ανάλογα με το είδος και τα χαρακτηριστικά των ακολουθιών που είναι υπό εξέταση - ταυτοποίηση, χρησιμοποιείται και ο κατάλληλος αλγόριθμος. Για την εξακρίβωση της ομοιότητας μεταξύ των ακολουθιών χρησιμοποιείται ένα είδος score, το οποίο είναι θετικό σε περίπτωση που υπάρχει ομοιότητα μεταξύ δύο αμινοξέων ή είναι αρνητικό όταν δεν υπάρχει μεγάλη ομοιότητα. Ο τρόπος με τον οποίο εξετάζουμε το ποσοστό ομοιότητας θα καθορίσει και τα αποτελέσματα που θα ληφθούν. Έχουν κατασκευαστεί αρκετοί τέτοιοι πίνακες, οι οποίοι έχουν προκύψει από τα χαρακτηριστικά των πρωτεϊνών. Πιο αναλυτικά ανάλογα με το πόσο διαφορετικές είναι δύο ακολουθίες πρωτεϊνών γίνεται διαφορετική επιλογή πίνακα scoring. Επίσης σημαντικό ρόλο έχει και η ευαισθησία που θέλουμε να υπάρχει στην αναζήτηση και είναι άμεση συνέπεια συνδυασμού των παραπάνω θεμάτων. Συνεπώς όλα αυτά αποτελούν ζητήματα,

τα οποία επιλύονται με την κατάλληλη επιλογή αλγορίθμου καθώς και του πίνακα score. Τέλος όλα τα παραπάνω ερωτήματα εμπίπτουν σε ένα πρόβλημα βελτιστοποίησης της αναζήτησης ομοιότητας μεταξύ μίας ακολουθίας και μίας βάσης δεδομένων.

Η συγκεκριμένη διπλωματική εργασία εξετάζει τον αλγόριθμο **BlastP**, ο οποίος αφορά στη σύγκριση μίας ακολουθίας με μία βάση δεδομένων από ακολουθίες. Για τον σκοπό αυτό έχει γίνει η σχεδίαση και η υλοποίηση του σε αναδιατασσόμενη λογική. Πιο συγκεκριμένα έχει δοθεί μεγάλο βάρος στο βήμα της επέκτασης του αλγορίθμου, σε επίπεδο hardware. Επίσης η λογική της επέκτασης οποίο αποτελεί μία πρώτη προσπάθεια διαφορετικής προσέγγισης του εν λόγω βήματος για τον αλγόριθμο BlastP.

1.2 Δομή Διπλωματικής

Στα επόμενα κεφάλαια θα παρουσιαστεί με δομημένο τρόπο το σκεπτικό της διπλωματικής, παραθέτοντας σε κάθε κεφάλαιο τις απαιτούμενες πληροφορίες για την κατανόηση του επόμενου κεφαλαίου. Επομένως:

Στο *κεφάλαιο 2* Αρχικά αναφέρονται κάποια ιστορικά γεγονότα, τα οποία αφορούν στις προσπάθειες που είχαν γίνει για την ανακάλυψη των πρωτεϊνών. Επίσης αναφέρεται ο βιολογικός ρόλος τους στην καθημερινότητα, ο οποίος δικαιολογεί και τη σημασία των ερευνών που έχουν τις ως αντικείμενο. Ακολουθεί μία περιγραφή των μεθόδων που βοηθούν στο να εξεταστούν δύο ή περισσότερες ακολουθίες πρωτεϊνών, καθώς και το τι πλεονεκτήματα έχει η κάθε μία μέθοδος. Οι τεχνικές αυτές υποστηρίζονται από κάποιους αλγορίθμους, όπως ο FASTA και ο BLASTP , που εξετάζεται στην παρούσα διπλωματική εργασία. Όλοι οι αλγόριθμοι αναφέρονται και παρουσιάζονται με σαφή τρόπο, ώστε να γίνουν αντιληπτές οι διαφορές μεταξύ τους.

Στη συνέχεια γίνεται μία λεπτομερής περιγραφή του αλγορίθμου BLAST, με έμφαση στη έκδοση BLASTP, όπου και αναφέρονται τα βήματα του αλγορίθμου και αναλύεται το σκεπτικό τους. Τέλος γίνεται μία εισαγωγή στους πίνακες scoring, όπου αναφέρεται ο τρόπος που προέκυψαν καθώς και ο ρόλος τους στη διαδικασία της αναζήτησης ομοιότητας μεταξύ δύο ακολουθιών. Πράγμα πολύ σημαντικό αφού είναι αλληλένδετο με το βήμα της επέκτασης του αλγορίθμου BLASTP.

Στο *κεφάλαιο 3* παρουσιάζεται η σχεδίαση και η υλοποίηση της αρχιτεκτονικής των βημάτων του αλγορίθμου σε επίπεδο αναδιατασσόμενης λογικής. Γίνεται μία προσπάθεια να παρουσιαστεί όσο το δυνατόν πιο απλά και κατανοητά η διαδικασία που ακολουθήθηκε, καθώς και ο λόγος που έγιναν κάποιες παραδοχές ή συμβάσεις κατά τη διάρκεια υλοποίησης. Όλα αυτά οδηγούν στην ολοκλήρωση των βημάτων της αρχιτεκτονικής και παρουσίαση των απο-

τελεσμάτων στο επόμενο κεφάλαιο.

Στο *κεφάλαιο 4* παρουσιάζονται τα αποτελέσματα των μετρήσεων, που προκύπτουν από την αρχιτεκτονική του συστήματος. Αρχικά περιγράφεται ο τρόπος που πραγματοποιήθηκε ο έλεγχος λειτουργίας της αρχιτεκτονικής, ενώ στη συνέχεια αναφέρονται τα σύνολο ακολουθιών που χρησιμοποιήθηκαν για την καταγραφή των απαιτούμενων μετρήσεων.

Στο *κεφάλαιο 5* καταγράφονται τα συμπεράσματα και τυχόν μελλοντικές επεκτάσεις. Ύστερα από την παρουσίαση της αρχιτεκτονικής υλοποίησης των βημάτων του αλγορίθμου καθώς και των αποτελεσμάτων λειτουργίας της, αναφέρονται τα συμπεράσματα που προκύπτουν. Αυτά αφορούν στο κατά πόσο καλύτερη είναι η σχεδίαση από κάποιες υπάρχουσες, αλλά και τι χρονισμούς παρουσιάζει. Επίσης χρησιμοποιείται η εμπειρία της συγκεκριμένης διπλωματικής και των μετρήσεων που ελήφθησαν ώστε να γίνουν κάποιες προτάσεις για μελλοντικές επεκτάσεις ή αλλαγές προσέγγισης σε κάποια ζητήματα, που μπορεί να βελτιώσουν την υπάρχουσα σχεδίαση ή ακόμα και να οδηγήσουν σε μία νέα διαφορετική σχεδίαση.

1.3 Συνεισφορά Διπλωματικής Εργασίας

Η παρούσα διπλωματική εργασία προσφέρει τα εξής παρακάτω:

- Υλοποίηση επεξεργαστή - επιταχυντή του αλγορίθμου BLASTp που κάνει string matching γρηγορότερα από ότι το software
- Υλοποίηση σε hardware της έκδοσης BLASTp του αλγορίθμου BLAST, που αφορά στην αναζήτηση ομοιότητας μεταξύ πρωτεϊνικών ακολουθιών
- Υλοποίηση της υπάρχουσας αρχιτεκτονικής σχεδίαση για το δεύτερο βήμα, πραγματοποιώντας αλλαγές στις δομικές μονάδες που είναι απαραίτητο να γίνουν
- Νέα αρχιτεκτονική σχεδίαση για το τρίτο βήμα του αλγορίθμου η οποία αφορά στη διαδικασία της επέκτασης και πώς μπορεί να γίνει γρήγορα
- Πραγματοποιείται τριπλή επέκταση ανά κύκλο ρολογιού
- Υλοποίηση της νέας αρχιτεκτονικής σε hardware.
- Ενσωμάτωση νέου Scoring Scheme, στο οποίο λαμβάνεται υπόψη η εξελικτική σχέση των αμινοξέων - χρήση πίνακα scoring BLOSUM62.

Κεφάλαιο 2

Πρωτεΐνες, Ακολουθίες και Μέθοδοι Σύγκρισης τους

2.1 Πρωτεΐνες

2.1.1 Γενικά

Οι πρωτεΐνες είναι οργανικές ενώσεις που κατασκευάζονται από αμινοξέα διατεταγμένα σε γραμμική αλυσίδα και αναδιπλωμένα σε σφαιρικό σχήμα[17, 27, ?]. Τα αμινοξέα σε μία αλυσίδα πολυμερών ενώνονται με πεπτιδικούς δεσμούς μεταξύ των καρβοξυλικών και των αμυνομάδων σε γειτονικά υπολείμματα των αμινοξέων. Η αλληλουχία των αμινοξέων σε μία πρωτεΐνη καθορίζεται από την αλληλουχία του γονιδίου, το οποίο κωδικοποιείται στον γενετικό κώδικα. Γενικά ο γενετικός κώδικας καθορίζει 20 πρότυπα αμινοξέων, αλλά σε ορισμένους οργανισμούς ο γενετικός κώδικας μπορεί να περιλαμβάνει την selenocysteine (ένα αμινοξύ που παρατηρείται σε διάφορα ένζυμα, όπως το glutathione peroxidase) και την pyrrolysine (ένα αμινοξύ που είναι παρατηρείται σε μονοκυτταρικά μικρόβια). Αμέσως μετά ή και κατά τη διάρκεια της σύνθεσης, τα υπολείμματα σε μία πρωτεΐνη είναι συχνά χημικά τροποποιημένα από την μετά-μεταφραστική τροποποίηση, η οποία μεταβάλλει τις φυσικές και χημικές ιδιότητες, την αναδίπλωση, τη σταθερότητα, τη δραστηριότητα και, τελικά, τη λειτουργία των πρωτεϊνών. Οι πρωτεΐνες μπορούν να συνεργαστούν μεταξύ τους για την επίτευξη μιας συγκεκριμένης λειτουργίας, και συχνά σχηματίζουν σταθερά σύμπλοκα.

2.1.2 Ιστορικά

Οι πρωτεΐνες αναγνωρίστηκαν ως μία χωριστή κατηγορία βιολογικών μορίων το δέκατο όγδοο αιώνα. Καταγεγραμμένα παραδείγματα [19] του καιρού αυ-

τού, περιλαμβάνουν τη λευκωματή από ασπράδια αυγού, ινώδες και γλουτένη σιταριού. Ο Ολλανδός χημικός Gerhardus Johannes Mulder πραγματοποίησε στοιχειακή ανάλυση κοινών πρωτεϊνών και διαπίστωσε ότι σχεδόν όλες οι πρωτεΐνες είχαν τον ίδιο εμπειρικό τύπο: $C_{400}H_{620}N_{100}O_{120}P_1S1$ Όμως έφτασε στο λανθασμένο συμπέρασμα ότι μπορεί να αποτελούνται από ένα μόνο είδος (πολύ μεγάλο) μόριο. Ο όρος "πρωτεΐνη" για να περιγράψει αυτά τα μόρια, προτάθηκε από έναν συγγενή του Mulder. Το όνομα πρωτεΐνη προέρχεται από την ελληνική λέξη πρωτεΐος που σημαίνει πρωτογενές. Ο Mulder προσπάθησε να αναγνωρίσει τα προϊόντα της αποδόμησης μιας πρωτεΐνης, όπως το αμινοξύ: λευκίνη, για το οποίο βρήκε ένα μοριακό βάρος (σχεδόν σωστό) 131 Da.

Η δυσκολία καθαρισμού πρωτεϊνών σε μεγάλες ποσότητες έκανε δύσκολη τη μελέτη τους για τους πρώτους βιοχημικούς. Ως εκ τούτου, οι πρώτες μελέτες επικεντρώθηκαν στις πρωτεΐνες που θα μπορούσαν να καθαριστούν σε μεγάλες ποσότητες, όπως εκείνες του αίματος, του αυγού το ασπράδι, διάφορες τοξίνες, και πεπτικά / μεταβολικά ένζυμα που προέρχονταν από τα σφαγεία. Τη δεκαετία του 1950, εταιρεία Armur Hot Dog Co. καθάρισε ένα κιλό καθαρής ριβονουκλεάσης Α από πάγκρεας βοδινού και την κατέστησε διαθέσιμη στους επιστήμονες. Η χειρονομία αυτή είχε ως αποτέλεσμα η ριβονουκλεάση Α να καταστεί η μείζονος ουσία προς ανάλυση και μελέτη για τις επόμενες δεκαετίες.

Ο Linus Pauling πιστώθηκε την επιτυχή πρόβλεψη της κανονικής δευτεροταγούς δομής που βασίζεται στους δεσμούς υδρογόνου, μία ιδέα που είχε προταθεί πρώτη φορά από τον William Astbury το 1933. Αργότερα το έργο του Walter Kauzmann αναφορικά με τη μετουσίωση (διαδικασία κατά την οποία οι πρωτεΐνες χάνουν την δευτεροταγή δομή τους λόγω εξωτερικών παραγόντων,όπως η θέρμανση), που βασίζεται εν μέρει σε προηγούμενες μελέτες του Kaj Linderstrorm-Lang, συνέβαλλε στην κατανόηση της αναδίπλωσης και της δομής των πρωτεϊνών μελετώντας τις υδροφοβικές αλληλεπιδράσεις. Το 1949 ο Fred Sanger προσδιόρισε σωστά την αλληλουχία αμινοξέων της ινσουλίνης, αποδεικνύοντας ότι οι πρωτεΐνες αποτελούνται από γραμμικά πολυμερή αμινοξέων και όχι διακλαδισμένων αλυσίδων ή κολλοειδών. Το 1960 πραγματοποιήθηκε η πρώτη ατομική ανάλυση πρωτεΐνης μέσω της κρυσταλλογραφίας χρησιμοποιώντας ακτίνες - Χ και το 1980 μέσω RMN. Σήμερα (2009) η Protein Data Bank έχει πάνω από 55.0000 ατομικές δομικές αναλύσεις πρωτεϊνών. Τέλος η χρήση του CRYO - ηλεκτρονικού μικροσκοπίου - για μεγάλες ομάδες μακρομορίων και η υπολογιστική μέθοδος πρόβλεψης της δομής μιας πρωτεΐνης για μικρά τμήματα πρωτεϊνών είναι δύο μέθοδοι που προσεγγίζουν την ατομική ανάλυση.

2.1.3 Βιολογικός Ρόλος

Οι διάφορες λειτουργίες που παρατηρούνται στους οργανισμούς γίνονται χάρη στις πρωτεΐνες. Ο βιολογικός τους ρόλος καθορίζεται κάθε φορά από την τρισδιάστατη δομή τους που είναι συνέπεια της αλληλουχίας των αμινοξέων, η οποία και ξεκινά από την πρωτοταγή δομή.

Όπως άλλα βιολογικά μακρομόρια, όπως οι πολυσακχαρίτες, τα λιπίδια και τα νουκλεϊκά οξέα, οι πρωτεΐνες είναι απαραίτητες για όλους τους ζωντανούς οργανισμούς και συμμετέχουν σε κάθε διαδικασία μέσα στα κύτταρα. Πολλές πρωτεΐνες δρουν ως ένζυμα που καταλύουν βιοχημικές αντιδράσεις, και είναι ζωτικής σημασίας στο μεταβολισμό. Άλλες πρωτεΐνες έχουν δομικές ή μηχανικές λειτουργίες, όπως οι πρωτεΐνες του κυτταρικού σκελετού, οι οποίες συμβάλουν στη διατήρηση της μορφής των κυττάρων. Οι πρωτεΐνες είναι επίσης σημαντικές στη διακυτταρική επικοινωνία, τη δράση του ανοσοποιητικού συστήματος, το σχηματισμό κυτταρικών ιστών και τον κυτταρικό κύκλο.

Οι πρωτεΐνες είναι απαραίτητα συστατικά στη διατροφή του ανθρώπου, δεδομένου ότι τα ζώα δεν μπορούν να συνθέσουν όλα τα αμινοξέα, αλλά πρέπει να τα λάβουν από τα τρόφιμα. Μέσω της διαδικασίας της πέψης, τα ζώα αποικοδομούν την πρωτεΐνη σε ελεύθερα αμινοξέα που μπορούν να χρησιμοποιηθούν για πρωτεϊνική σύνθεση.

2.1.4 Βιοχημική Ανάλυση

Οι πρωτεΐνες είναι γραμμικά πολυμερή κατασκευασμένες από σειρές μήκους έως 20 διαφορετικών αμινοξέων. Όλα τα αμινοξέα έχουν κοινά δομικά χαρακτηριστικά, συμπεριλαμβανομένου ενός α-άνθρακα που είναι ενωμένος σε μία αμινομάδα, καρβοξυλομάδα, και μία μεταβλητή πλευρική αλυσίδα. Μόνο η προλίνη διαφέρει από αυτή τη βασική δομή που περιέχει έναν ασυνήθιστο δακτύλιο μεταξύ του αζώτου και της αμινομάδας, η οποία αναγκάζει τη CO-NH αμίδιο ρίζα σε μία συγκεκριμένη διάπλαση. Οι πλευρικές αλυσίδες των αμινοξέων, που παρουσιάζονται σε μία λεπτομερή λίστα αμινοξέων, έχουν μια μεγάλη ποικιλία από δομικές και χημικές ιδιότητες. Το συνδυασμένο αποτέλεσμα όλων των πλευρικών αλυσίδων των αμινοξέων σε μια πρωτεΐνη είναι αυτό που καθορίζει απόλυτα την τρισδιάστατη δομή και τη χημική συμπεριφορά της.

Τα αμινοξέα σε μία πολυπεπτιδική αλυσίδα είναι συνδεδεμένα με πεπτιδικούς δεσμούς. Μόλις ένα μεμονωμένο αμινοξύ συνδεθεί στην πρωτεϊνική αλυσίδα, αυτό ονομάζεται κατάλοιπο (residue), ενώ οι αλυσίδες από άνθρακα, άζωτο και άτομα οξυγόνου που συνδέονται, είναι γνωστή ως κύρια αλυσίδα ή πρωτεϊνική σπονδυλική στήλη. Ο πεπτιδικός δεσμός έχει δύο μορφές συντονισμού που συμβάλλουν με ομοιοπολικού χαρακτήρα δεσμούς και αναστέλλουν την περιστροφή γύρω από τον άξονα της, ώστε οι α-άνθρακες να είναι στο

ίδιο περίπου επίπεδο. Οι άλλες δύο διεδρικές γωνίες στον πεπτιδικό δεσμό καθορίζουν το τοπικό σχήμα της πρωτεϊνικής σπονδυλικής στήλης. Το τέλος μιας πρωτεΐνης με μία ελεύθερη καρβοξυλομάδα είναι γνωστό ως C-τελικό (C-terminus - carboxy terminus), ενώ το τέλος με μία ελεύθερη αμινομάδα είναι γνωστή ως N-τελικό (N-terminus - amino terminus).

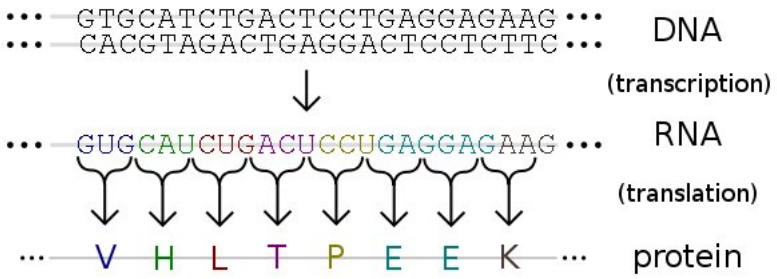
Οι λέξεις πρωτεΐνη, πολυπεπτίδιο και πεπτίδιο είναι λιγάκι ασαφείς και μπορεί να επικαλύπτονται σε νόημα. Η λέξη *Πρωτεΐνη* χρησιμοποιείται γενικά στην αναφορά ενός πλήρους βιολογικού μορίου σε μία σταθερή διάπλαση, ενώ η λέξη *πεπτίδιο* χρησιμοποιείται μικρά ολιγομερή αμινοξέα που συνήθως δεν έχουν σταθερή τρισδιάστατη δομή. Ωστόσο, το όριο μεταξύ των δύο δεν είναι σαφώς καθορισμένο και συνήθως βρίσκεται κοντά σε 20 - 30 κατάλοιπα. Η λέξη *πολυπεπτίδιο* μπορεί να αναφέρεται σε οποιαδήποτε απλή γραμμική αλυσίδα αμινοξέων, ανεξαρτήτου μήκους, αλλά συχνά συνεπάγεται η απουσία καθορισμένης διάπλασης.

2.1.5 Σύνθεση

Οι πρωτεΐνες είναι μεγάλα σύνθετα βιομόρια, με μοριακό βάρος από 10.000 μέχρι πάνω από 1 εκατομμύριο, αποτελούμενα από αμινοξέα, τα οποία ενώνονται μεταξύ τους με πεπτιδικούς δεσμούς, σχηματίζοντας μια γραμμική αλυσίδα, καλούμενη αλυσίδα πολυπεπτιδίων. Όλες οι πρωτεΐνες περιέχουν άνθρακα, οξυγόνο και άζωτο και οι περισσότερες εξ αυτών και θείο. Η ακολουθία αμινοξέων σε μια πρωτεΐνη καθορίζεται από ένα γονίδιο και κωδικοποιείται κατά το γενετικό κώδικα DNA. Παρόλο που ο γενετικός κώδικας κωδικοποιεί 20 αμινοξέα, τα αμινοξέα που συνιστούν την πρωτεΐνη συχνά υφίστανται χημικές αλλαγές κατά τη μετά-μεταγραφική τροποποίηση: είτε προτού να μπορέσει η πρωτεΐνη να λειτουργήσει, είτε ως τμήμα των μηχανισμών ελέγχου. Περισσότερες από μία πρωτεΐνες συχνά λειτουργούν μαζί για να επιτύχουν κάποια συγκεκριμένη λειτουργία, ή μπορεί ακόμα και να συσσωματωθούν για να διαμορφώσουν σταθερά σύμπλοκα.

Οι πρωτεΐνες αποτελούνται από αμινοξέα χρησιμοποιώντας πληροφορία που είναι κωδικοποιημένη στα γονίδια. Κάθε πρωτεΐνη έχει μοναδική ακολουθία αμινοξέων, που καθορίζεται από την νουκλεοτιδική ακολουθία του γονιδιώματος που κωδικοποιεί την εκάστοτε πρωτεΐνη. Ο γενετικός κώδικας είναι ένα σύνολο από 3 νουκλεϊτιδικά σύνολα, καλούμενα κωδικόνια και κάθε συνδυασμός 3 νουκλεοτιδίων συνιστούν ένα αμινοξύ, για παράδειγμα AUG (Αδενίνη - Ουρακίλη - Γουανίνη) είναι ο κώδικας για την μεθιόνη (methionine). Επειδή το DNA περιλαμβάνει 4 νουκλεοτίδια, ο συνολικός αριθμός των πιθανών κωδικονίων είναι 64. Επομένως, υπάρχει ένα περίσσευμα στον γενετικό κώδικα, με μερικά αμινοξέα να καθορίζονται από περισσότερα του ενός κωδικονίου. Γονίδια κωδικοποιημένα στο DNA, αρχικά μεταγράφονται στο προ-αγγελιοφόρο

RNA (pre-messenger RNA - mRNA) από πρωτεΐνες όπως η RNA πολυμεράση. Οι περισσότεροι οργανισμοί επεξεργάζονται το προ-αγγελιοφόρο RNA (γνωστό και ως πρωτεύον αντίγραφο) χρησιμοποιώντας διάφορες μορφές μετά-μεταγραφικής τροποποίησης για να σχηματίσουν το ώριμο mRNA, το οποίο στη συνέχεια χρησιμοποιείται ως πρότυπο για την σύνθεση πρωτεϊνών από το ριβόσωμα. Στα προκαρυωτικά κύτταρα το mRNA μπορεί είτε να χρησιμοποιηθεί από τη στιγμή που παράγεται, είτε να δεσμευτεί από ένα ριβόσωμα, αφότου έχει απομακρυνθεί από το νουκλεοτίδιο. Σε αντίθεση, στα ευκαρυωτικά κατασκευάζεται το mRNA στον κυτταρικό πυρήνα και ύστερα μεταφέρεται μέσω της πυρηνικής μεμβράνης στο κυτταρόπλασμα, όπου πραγματοποιείται η σύνθεση της πρωτεΐνης. Ο ρυθμός σύνθεσης μιας πρωτεΐνης είναι υψηλότερος σε ένα ευκαρυωτικό κύτταρο από ότι στα προκαρυωτικά και μπορεί να φτάσει έως και τα 20 αμινοξέα το δευτερόλεπτο. Η παραπάνω διαδικασία φαίνεται στο σχήμα 2.1.



Σχήμα 2.1: *Η DNA ακολουθία από ένα γονίδιο κωδικοποιεί την ακολουθία αμινοξέων μίας πρωτεΐνης*

Η διαδικασία σύνθεσης μίας πρωτεΐνης από ένα πρότυπο mRNA είναι γνωστή ως μετάφραση[15, 3, 5, 1]. Η μετάφραση συμβαίνει στο κυτταρόπλασμα όπου βρίσκονται τα ριβοσώματα. Τα ριβοσώματα είναι μικρές και μεγάλες υπομονάδες που περιβάλλουν το mRNA. Κατά τη διάρκεια της μετάφρασης το mRNA αποκωδικοποιείται για να παράγει ένα συγκεκριμένο πολυπεπτίδιο, σύμφωνα με τους κανόνες που καθορίζονται από τον τρινουκλεϊκό γενετικό κώδικα. Χρησιμοποιείται η ακολουθία του mRNA ως πρότυπο για την καθοδή-

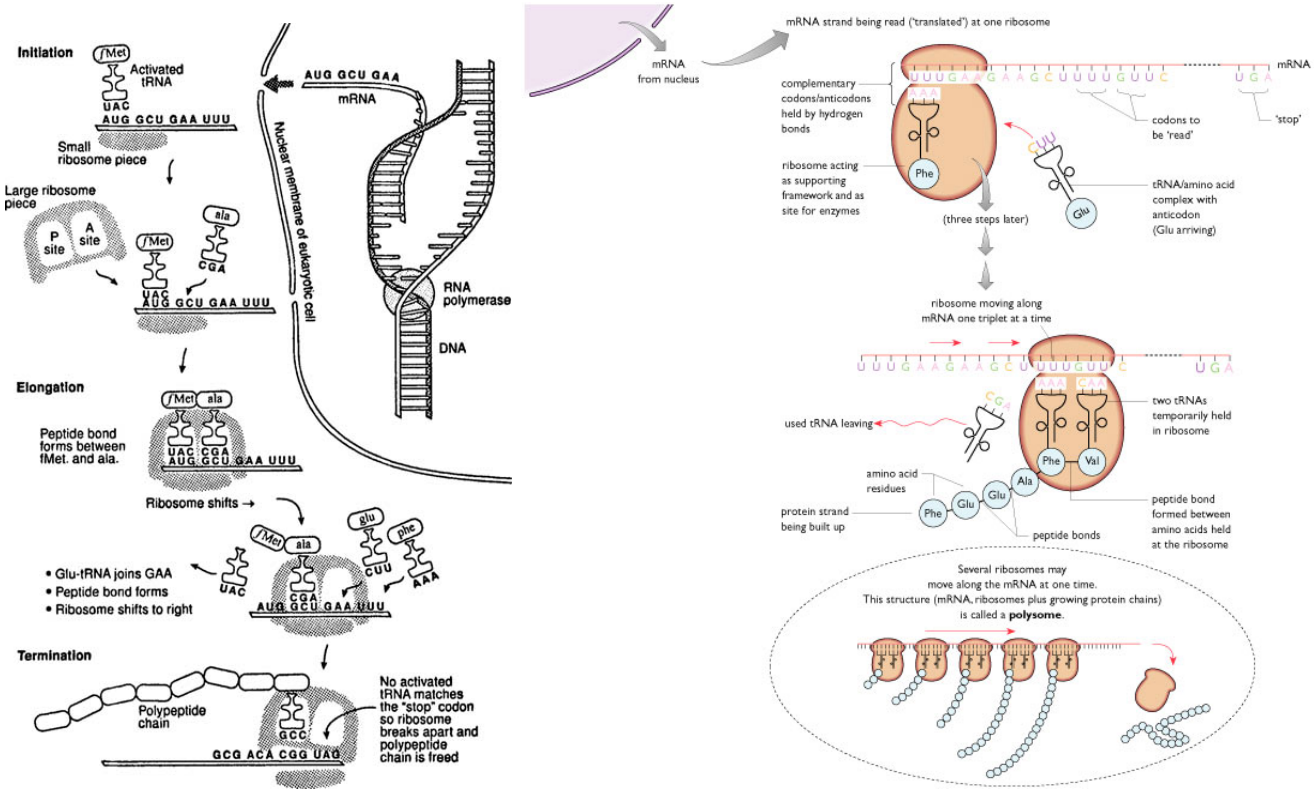
γηση της σύνθεσης μίας αλυσίδας αμινοξέων που σχηματίζει μία πρωτεΐνη. Η διαδικασία της μετάφρασης χωρίζεται σε τέσσερις φάσεις: *ενεργοποίηση* (*activation*), *έναρξη* (*initiation*), *επιμήκυνση* (*elongation*) και *λήξη* (*termination*), περιγράφοντας όλη τη φάση ανάπτυξης του προϊόντος της μετάφρασης, μίας αλυσίδας αμινοξέων ή ενός πολυπεπτιδίου. Επίσης συχνά το παραγόμενο πολυπεπτίδιο αναφέρεται και ως *εκκολαπτόμενη αλυσίδα*.

Κατά την ενεργοποίηση, το σωστό αμινοξύ (AA) συνδέεται με το σωστό μεταφορικό RNA, tRNA. Αν και αυτό το βήμα δεν είναι τεχνικά ένα βήμα της μετάφρασης, είναι αναγκαίο για την διαδικασία της μετάφρασης. Το αμινοξύ AA ενώνεται με μία καρβοξυλική ομάδα στο σημείο 3'OH του tRNA μέσω δεσμού εστέρα. Όταν σε ένα tRNA ενώνεται ένα αμινοξύ, τότε λέγεται ότι είναι *ενεργοποιημένο*. Κατά την έναρξη η μικρή υπομονάδα του ριβοσώματος συνδέεται με το 5' τέλος του mRNA με τη βοήθεια παραγόντων έναρξης, initation factors (IF), οι οποίοι είναι άλλες πρωτεΐνες που βοηθούν στη διαδικασία. Η επιμήκυνση πραγματοποιείται όταν το επόμενο στη σειρά αμινοξεϊκό - tRNA (*ενεργοποιημένο* tRNA) συνδέεται στο ριβόσωμα με το πουρινικό νουκλεοτίδιο τριφοσφωρική-5'-γουανοσίνη, Guanosine-5'-triphosphate (GTP) και έναν παράγοντα επιμήκυνσης. Η λήξη ενός πολυπεπτιδίου συμβαίνει όταν η A πλευρά ενός ριβοσώματος συναντά ένα κωδικώνιο παύσης (UAA, UAG, UGA). Όταν συμβεί αυτό το tRNA δεν μπορεί να το αναγνωρίσει, αλλά κάποιοι παράγοντες απελευθέρωσης μπορούν να αναγνωρίσουν τα συγκεκριμένα κωδικώνια, οπότε προκαλούν την απελευθέρωση πολυπεπτιδικής αλυσίδας. Μία απεικόνιση των βημάτων φαίνεται στο σχήμα 2.2. Η ικανότητα απενεργοποίησης ή αναστολής μετάφρασης της βιοσύνθεσης πρωτεϊνών χρησιμοποιείται από τα αντιβιοτικά, όπως: ανισομυκίνη (anisomycin), χλωροαμφινεκόλη (chloramphenicol), τετρακυλίνη (tetracycline), στρεπτομυκίνη (streptomycin), ερυθρομυκίνη (erythromycin), πυρομυκίνη (puromycin).

1. Έναρξη - Ένα ριβόσωμα συνδέεται στο mRNA και ξεκινάει να κωδικοποιεί στο κωδικόνιο FMet, συνήθως AUG, μερικές φορές GUG ή UUG.

2. Επιμήκυνση - Το tRNA φέρνει το αντίστοιχο αμινοξύ (το οποίο έχει ένα αντικωδικόνιο που προσδιορίζει το αμινοξύ ως το αντίστοιχο μόριο για ένα κωδικόνιο) σε κάθε κωδικόνιο όσο το ριβόσωμα κινείται κάτω από το νήμα του mRNA.

3. Λήξη - Ανάγνωση του τελικού κωδικονίου στο mRNA (κωδικόνιο παύσης), το οποίο σταματάει την διαδικασία σύνθεσης της πεπτιδικής αλυσίδας και την απελευθερώνει.



Σχήμα 2.2: *Δύο γραφικές αναπαραστάσεις για την διαδικασία σύνθεσης - μετάφρασης μίας πρωτεΐνης*

Χημική Σύνθεση

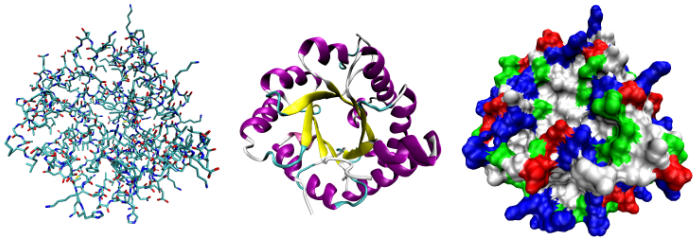
Μικρές πρωτεΐνες μπορούν επίσης να παραχθούν χημικά μέσω μιας οικογένειας μεθόδων γνωστής και ως *πεπτιδική σύνθεση*, οι οποίες στηρίζονται σε τεχνικές οργανικής σύνθεσης όπως η *chemical ligation* για να παράγουν πεπτίδια σε υψηλή απόδοση. Η χημική σύνθεση επιτρέπει την εισαγωγή μη φυσικών αμινοξέων στις πολυπεπτιδικές αλυσίδες, όπως την προσθήκη φθοριζουσών ριζών στις πλευρικές αλυσίδες των αμινοξέων. Αυτές οι μέθοδοι είναι πολύ χρήσιμες στην εργαστηριακή βιοχημεία και στην κυτταρική βιολογία, αλλά όχι για εμπορικές εφαρμογές. Όμως η χημική σύνθεση είναι αναποτελεσματική για πολυπεπτιδικές αλυσίδες μήκους μεγαλύτερου των 300 αμινοξέων, με αποτέλεσμα να μην μπορούμε να διαπιστώσουμε στην αρχική τριτογενή δομή των παραγόμενων πρωτεϊνών. Τέλος οι περισσότερες χημικές συνθέσεις διενεργούνται μεταξύ του αμινοξέος C-terminus και του αμινοξέος N-terminus, αντίθετα με την βιολογική διεργασία.

2.1.6 Δομή Πρωτεϊνών

Οι περισσότερες πρωτεΐνες αναδιπλώνονται σε μοναδικές τρισδιάστατες δομές. Το σχήμα στο οποίο μια πρωτεΐνη αναδιπλώνεται είναι γνωστό ως *φυσική διάπλαση*. Παρόλο που πολλές πρωτεΐνες μπορούν να αναδιπλωθούν, χωρίς βοήθεια, μέσω των χημικών ιδιοτήτων των αμινοξέων τους, υπάρχουν ορισμένες που απαιτούν τη συνδρομή των ειδικών μοριακών πρωτεϊνών *chaperones*, για να αναδιπλωθούν στη φυσική τους κατάσταση. Μπορούν να διακριθούν τέσσερις ξεχωριστές πτυχές της δομής μιας πρωτεΐνης.

- Πρωτοταγής δομή: η ακολουθία των αμινοξέων
- Δευτεροταγής δομή: τακτικά επαναλαμβανόμενες τοπικές δομές που σταθεροποιούνται με δεσμούς υδρογόνου μεταξύ των καρβοξυλομάδων και των αμινομάδων των αμινοξέων. Τα πιο γνωστά παραδείγματα είναι η α-έλικα (alpha helix), δεξιόστροφη, όπου οι σπείρες διατηρούνται στη θέση τους με δεσμούς υδρογόνου μεταξύ των καρβοξυ- και αμινομάδων. Μία άλλη είναι η β-πτυχωτή επιφάνεια (β-sheet), όπου στη περίπτωση αυτή διασταυρώνονται παράλληλες αλυσίδες πολυπεπτιδίων που ενώνονται στις διασταυρώσεις με δεσμούς υδρογόνου σχηματίζοντας έτσι μια εξαιρετικά σφιχτή δομή, όπως στο μετάξι. Ακόμα υπάρχουν οι turns. Επειδή οι δευτεροταγείς δομές είναι τοπικές, πολλές περιοχές από διαφορετικές δευτεροταγείς δομές μπορούν να επαναλαμβάνονται στο ίδιο μόριο πρωτεΐνης.
- Τριτοταγής δομή: Το συνολικό σχήμα ενός απλού μορίου πρωτεΐνης. Η χωρική σχέση μίας δευτεροταγούς δομής με μία άλλη. Γενικά οι τριτοταγείς δομές σταθεροποιούνται από μη τοπικές αλληλεπιδράσεις, με συνηθέστερο το σχηματισμό ενός υδροφοβικού πυρήνα, αλλά και ιονικών ομολόγων (salt bridges), δεσμών υδρογόνου, δισουλφιδικών δεσμών και ακόμα και μετά-μεταφραστικών τροποποιήσεων. Συνήθως ο ορισμός "τριτοταγής δομή " χρησιμοποιείται ως συνώνυμο του όρου αναδίπλωσης. Η τριτογενής δομή ελέγχει τη βασική λειτουργία μιας πρωτεΐνης. Παρακάτω παρουσιάζονται διάφορες πρωτεΐνες στη μορφή αυτή 2.3
- Τεταρτοταγής δομή: Η δομή που αποτελείται από πολλά μόρια πρωτεϊνών (πολυπεπτιδικές αλυσίδες), που συνήθως ονομάζονται υπομονάδες πρωτεϊνών σε αυτό το πλαίσιο, τα οποία λειτουργούν ως ενιαίο σύμπλεγμα πρωτεϊνών. (Παράδειγμα: αιμοσφαιρίνη)

Οι πρωτεΐνες δεν είναι εντελώς άκαμπτα μόρια. Εκτός από τα δεδομένα επίπεδα της δομής, οι πρωτεΐνες μπορούν να μετακινηθούν μεταξύ των διαφόρων συναφών δομών, ενώ εκτελούν τις λειτουργίες τους. Στο πλαίσιο των



Σχήμα 2.3: *Τρεις πιθανές αναπαραστάσεις μιας τρισδιάστατης δομής της πρωτεΐνης triose phosphate isomerase. Αριστερά: Ατομική αναπαράσταση ανάλογα με τον ατομικό τύπο. Μέση: Απλοποιημένη χρωματική αναπαράσταση καταδεικνύοντας τη διάπλαση της πρωτεϊνικής σπονδυλικής στήλης, Δεξιά: Διαλυτή αντιπροσωπευτική επιφάνεια χρωματισμένη ανάλογα με τον τύπο κατάλοιπου (όξινο: κόκκινο, βασικό: μπλε, πολικά: πράσινο, μη πολικά: λευκό)*

λειτουργικών αυτών ανακατατάξεων, η τριτοταγής και τεταρτοταγής δομή συνήθως ονομάζονται και διαπλάσεις (*conformations*), και οι μεταβάσεις μεταξύ τους ονομάζονται αλλαγές διάπλασης. Τέτοιες αλλαγές προκαλούνται συχνά από τη σύνδεση του υποστρώματος ενός μορίου στην ενεργή πλευρά ενός ενζύμου, ή στην φυσική περιοχή της πρωτεΐνης που συμμετέχει στη χημική κατάλυση. Επίσης οι πρωτεΐνες μπορούν να υποβληθούν σε αλλαγή της δομής τους μέσων θερμικής δόνησης και σύγκρουσης με άλλα μόρια.

Ανεπίσημα οι πρωτεΐνες μπορούν να διαχωριστούν σε τρεις κύριες κατηγορίες, οι οποίες συσχετίζονται με την τυπική τριτοταγή *δομή*: σφαιρικές πρωτεΐνες, ινώδεις πρωτεΐνες και μεμβρανώδεις πρωτεΐνες. Σχεδόν όλες οι σφαιρικές πρωτεΐνες είναι διαλυτές και είναι ένζυμα. Οι ινώδεις πρωτεΐνες είναι συχνά δομικές, όπως το κολλαγόνο, το κύριο συστατικό του συνδετικού ιστού, ή η κερατίνη, η συνιστώσα πρωτεΐνης στα μαλλιά και τα νύχια. Οι μεμβρανώδεις πρωτεΐνες συχνά λειτουργούν ως υποδοχείς ή αναθέτουν σε μόρια να περάσουν μέσα από την κυτταρική μεμβράνη. Ένας άλλος διαχωρισμός μπορεί να γίνει με κριτήριο τη *σύνθεση* τους, όπου και διακρίνονται σε απλές, όταν αποτελούνται μόνο από αμινοξέα και σε σύνθετες όταν στο μόριο τους περιλαμβάνονται και μη πρωτεϊνικά τμήματα όπως μέταλλα, σάκχαρα και λίπη. Επίσης με κριτήριο τη *λειτουργία* τους διακρίνονται σε δομικές όταν αποτελούν τα δομικά υλικά του κυττάρου και λειτουργικές όταν συμβάλουν σε κάποιες λειτουργίες. Τέλος μια ειδική περίπτωση από ενδομοριακούς δεσμούς υδρογόνου, που ελάχιστα προστατεύονται από το νερό και για αυτό το λόγο προωθούν την αφυδάτωσή τους, ονομάζεται dehydrons.

2.2 Ακολουθίες - Ευριστικοί Αλγόριθμοι

Για τις ακολουθίες αμινοξέων, που σχηματίζουν τις πρωτεΐνες, υπάρχει η ανάγκη ευθυγράμμισής τους. Η *ευθυγράμμιση ακολουθιών* βοηθά στον εντοπισμό των περιοχών που παρουσιάζουν ομοιότητα και η οποία ενδέχεται να είναι αποτέλεσμα λειτουργικής, διαρθρωτικής και εξελικτικής σχέσης μεταξύ των ακολουθιών αυτών[14, **?**, 16]. Έχουν αναπτυχθεί δύο τύποι ευθυγράμμισης ακολουθίας: η *τοπική ευθυγράμμιση (local alignment)* και η *ολική ευθυγράμμιση (global alignment)*. Η ολική ευθυγράμμιση σχετίζεται με την ευθυγράμμιση μίας υπό ακολουθίας που εκτείνεται κατά μήκος όλης της ακολουθίας. Αντίθετα η τοπική ευθυγράμμιση, αφορά στον εντοπισμό περιοχών ομοιότητας σε μεγάλες ακολουθίες, οι οποίες είναι συνολικά πολύ διαφορετικές. Πιο αναλυτικά η ολική ευθυγράμμιση, που επιχειρεί να ευθυγραμμίσει κάθε κατάλοιπο σε κάθε ακολουθία, είναι πιο χρήσιμη όταν οι υπό εξέταση ακολουθίες είναι παρόμοιες και περίπου ίσου μήκους. (Αυτό βέβαια σημαίνει ότι μία ολική ευθυγράμμιση δεν μπορεί να τελειώνει με κενό). Μία τεχνική ολικής ευθυγράμμισης που χρησιμοποιείται ευρέως, είναι ο αλγόριθμος των *Needleman - Wunsch*, που βασίζεται στη μέθοδο του δυναμικού προγραμματισμού. Αντίθετα η τοπική ευθυγράμμιση είναι πιο χρήσιμη για ανόμοιες ακολουθίες, για τις οποίες υπάρχει η υπόνοια ότι περιέχουν περιοχές ομοιότητας ή όμοιες υπακολουθίες στο ευρύτερο περιεχόμενο των ακολουθιών τους. Ο αλγόριθμος *Smith - Waterman* είναι μία γενική μέθοδος τοπικής ευθυγράμμισης, βασισμένος επίσης στον δυναμικό προγραμματισμό. Με δύο αρκετά όμοιες ακολουθίες, δεν υπάρχει διαφορά μεταξύ ολικής και τοπικής ευθυγράμμισης. Ένα παράδειγμα ολικής και τοπικής ευθυγράμμισης φαίνεται στο σχήμα 2.4

```
Global FTFTALILLAVAV
       F--TAL-LLA-AV

Local  FTFTALILL-AVAV
       --FTAL-LLAAV--
```

Σχήμα 2.4: Αναπαράσταση ολικής και τοπικής ευθυγράμμισης

Υβριδικές μέθοδοι, γνωστές και ως ημιολικές ή "ολικές" μέθοδοι, προσπαθούν να εντοπίσουν την καλύτερη δυνατή ευθυγράμμιση, που περιλαμβάνει την αρχή και το τέλος της μίας ή της άλλης ακολουθίας. Αυτό είναι ιδιαίτερα χρήσιμο όταν το τέλος του κάτω τμήματος της μίας ακολουθίας συμπίπτει με το πάνω τμήμα μίας άλλης ακολουθίας. Στην περίπτωση αυτή, ούτε η ολική ούτε η

τοπική ευθυγράμμιση είναι εντελώς κατάλληλη: η ολική ευθυγράμμιση θα επιχειρήσει να επεκτείνει την ευθυγράμμιση και πέρα της περιοχής επικάλυψης, ενώ η τοπική ενδέχεται να μην καλύψει πλήρως το τμήμα της επικάλυψης.

2.2.1 Pairwise sequence alignment

Η ευθυγράμμιση ακολουθιών κατά ζεύγη χρησιμοποιείται για να βρεθεί η καλύτερη αντιστοίχιση ανά ζεύγος (τοπική) ή ολική ευθυγράμμιση για δύο ακολουθίες. Οι ευθυγραμμίσεις ανά ζεύγος μπορούν να χρησιμοποιηθούν μόνο σε δύο ακολουθίες τη φορά, αλλά είναι αποτελεσματικές στον υπολογισμό και χρησιμοποιούνται συχνά από μεθόδους που δεν απαιτούν εξαιρετική ακρίβεια (όπως η αναζήτηση μίας βάσης δεδομένων για ακολουθίες με υψηλή ομοιότητα με μία ακολουθία που είναι υπό εξέταση). Υπάρχουν τρεις βασικές μέθοδοι που παράγουν ευθυγράμμιση ανά ζεύγος: μέθοδοι *dot-matrix*, *δυναμικός προγραμματισμός* και μέθοδοι *λέξης*.

Η *dot-matrix* προσέγγιση αποτελεί ένα είδος τοπικής ευθυγράμμισης, η οποία έμμεσα παράγει μία οικογένεια από στοιχίσεις για ξεχωριστές περιοχές της ακολουθίας, είναι ποιοτική και απλή, αλλά πολύ χρονοβόρα για να χρησιμοποιηθεί σε μεγάλη κλίμακα. Είναι πολύ εύκολο να εντοπιστούν οπτικά ορισμένα χαρακτηριστικά της ακολουθίας - όπως καταχωρήσεις, διαγραφές, επαναλήψεις, ή ανεστραμμένες επαναλήψεις - σε μία αποτύπωση της dot-matrix.

Η τεχνική του *δυναμικού προγραμματισμού* μπορεί να εφαρμοστεί για τοπικές ευθυγραμμίσεις μέσω του αλγορίθμου των *Smith - Waterman*. Στην τυπική περίπτωση, οι ευθυγραμμίσεις πρωτεϊνών χρησιμοποιούν έναν πίνακα υποκατάστασης για να δώσει σκορ στις αντιστοιχίες και αναντιστοιχίες μεταξύ αμινοξέων, καθώς και ποινή κενού για αντιστοίχιση ενός αμινοξέος της μίας ακολουθίας με ένα κενό της άλλης ακολουθίας. Οι ακολουθίες DNA και RNA μπορούν να χρησιμοποιούν έναν πίνακα σκοραρίσματος, αλλά συνήθως απλά κρατάνε ένα θετικό σκορ ταιριάσματος, ένα αρνητικό σκορ αναντιστοιχίας και ένα αρνητικό σκορ για ποινή κενού. Επομένως η μέθοδος του δυναμικού προγραμματισμού εγγυάται ότι θα βρεθεί η βέλτιστη ευθυγράμμιση δοθέντος μίας συγκεκριμένης συνάρτησης σκοραρίσματος. Ωστόσο, ο προσδιορισμός μίας καλής συνάρτησης σκοραρίσματος συχνά αποτελεί εμπειρικό, και όχι θεωρητικό ζήτημα, Παρόλο που ο δυναμικός προγραμματισμός είναι επεκτάσιμος σε περισσότερες των δύο ακολουθιών, είναι απαγορευτικά αργός για εξαιρετικά μεγάλες ακολουθίες ή για μεγάλο αριθμό ακολουθιών.

Οι μέθοδοι *λέξης*, γνωστές και ως k-tuple μέθοδοι, είναι ευριστικές μέθοδοι που δεν εγγυώνται ότι θα βρουν τη βέλτιστη ευθυγράμμιση, αλλά είναι πολύ πιο αποτελεσματικές σε σχέση με τη μέθοδο του δυναμικού προγραμματισμού. Οι μέθοδοι αυτές είναι εξαιρετικά χρήσιμες για αναζητήσεις σε μεγάλης κλίμακας βάσεις δεδομένων, όπου είναι αναμενόμενο ότι μεγάλος αριθμός των

υποψήφιων ακολουθιών δε θα έχουν σημαντική αντιστοιχία με την υπό εξέταση ακολουθία. Οι μέθοδοι λέξης είναι γνωστές για την εφαρμογή τους σε εργαλεία αναζήτησης βάσεων δεδομένων, όπως η *FASTA* και η οικογένεια *BLAST* Οι μέθοδοι λέξης εντοπίζουν μία σειρά από σύντομες, μη επικαλύπτουσες υπακολουθίες - "λέξεις" στην υπό εξέταση ακολουθία, που μετά αντιστοιχίζονται με τις υποψήφιες ακολουθίες της βάσης δεδομένων. Οι σχετικές θέσεις των λέξεων στις υπό σύγκριση ακολουθίες αφαιρείται για να διατηρείται ένα μέτρο αντιστάθμισης (offset). Επομένως υποδηλώνεται μία περιοχή ευθυγράμμισης εάν πολλές διαφορετικές λέξεις παράγουν το ίδιο offset.

Στη μέθοδο *FASTA* ο χρήστης ορίζει μία τιμή k, η οποία θα χρησιμοποιηθεί ως μήκος λέξης με το οποίο θα πραγματοποιηθεί η αναζήτηση στη βάση δεδομένων. Η μέθοδος είναι αργότερη αλλά πιο ευαίσθητη για μικρές τιμές του k, οι οποίες είναι προτιμότερες για αναζητήσεις που το μήκος της υπό εξέταση ακολουθία είναι πολύ μικρό. Οι μέθοδοι αναζήτησης της οικογένειας *BLAST* παρέχουν έναν αριθμό από βελτιστοποιημένους αλγορίθμους, για συγκεκριμένους τύπους υπό εξέταση ακολουθιών, όπως εύρεση αντιστοιχίας ακολουθιών με μακρινή συγγένεια. Από την οικογένεια BLAST, στην παρούσα διπλωματική επικεντρώνεται το ενδιαφέρον στον αλγόριθμο *Blast P*, που είναι κατάλληλος για αναζήτηση ακολουθιών πρωτεϊνών.

2.3 Αλγόριθμος Blast

Ο αλγόριθμος **Basic Local Alignment Search Tool - BLAST** είναι ένας από τους πιο ευρέως χρησιμοποιούμενους αλγορίθμους στη Βιοπληροφορική[20, 37, 34, 31, 29]. Είναι μία ευριστική μέθοδος, η οποία χρησιμοποιείται για την σύγκριση μίας ακολουθίας με μία βάση δεδομένων ακολουθιών, και αναγνωρίζει τις ακολουθίες της βάσης δεδομένων που έχουν ομοιότητα με την υπό εξέταση ακολουθία πάνω από ένα συγκεκριμένο κατώφλι. Το πρόγραμμα BLAST σχεδιάστηκε από τους Eugene Myers, Stephen Altschul, Warren Gish, David J. Lipman και Webb Miller στο NHI και δημοσιεύτηκε το 1990.Το γεγονός ότι είναι αλγόριθμος βελτιστοποιημένης ταχύτητας τον καθιστά πολύ πρακτικό για τεράστιες βάσεις δεδομένων γονιδιώματος. Συγκριτικά με τους αλγορίθμους δυναμικού προγραμματισμού, που αναφέρθηκαν στο προηγούμενο εδάφιο, ο BLAST είναι έως και 50 φορές γρηγορότερος, παρόλο που δεν εγγυάται τη βέλτιστη ευθυγράμμιση μεταξύ της υπό εξέταση ακολουθίας και των ακολουθιών της βάσης δεδομένων, όπως συμβαίνει μέσω της χρήσης δυναμικού προγραμματισμού. Επίσης ο BLAST είναι χρονικά πιο αποτελεσματικός και από τον FASTA, αφού πραγματοποιεί αναζήτηση μόνο για τα πιο σημαντικά δείγματα στις ακολουθίες, χρησιμοποιώντας παράλληλα συγκριτική ευαισθησία.

Ο BLAST, χρησιμοποιώντας μία ευριστική μέθοδο, βρίσκει ομόλογες ακολουθίες, χωρίς να συγκρίνει καμία ακολουθία στο σύνολό της, αλλά εντοπίζοντας μικρές αντιστοιχίες μεταξύ των δύο ακολουθιών. Η διαδικασία εύρεσης αρχικών λέξεων λέγεται *seed*. Εφόσον ο BLAST βρει μία πρώτη αντιστοιχία, ξεκινά να πραγματοποιεί τοπικές ευθυγραμμίσεις. Ενόσω προσπαθεί να βρει αντιστοιχία μεταξύ των ακολουθιών, τα σύνολα κοινών γραμμάτων, γνωστά ως λέξη, είναι πολύ σημαντικά. Συνεπώς ο ευριστικός αλγόριθμος BLAST εντοπίζει όλες τις κοινές λέξεις μεταξύ της ακολουθίας που εξετάζουμε και όλων των ακολουθιών μίας βάσης δεδομένων. Τα αποτελέσματα χρησιμοποιούνται στη συνέχεια για την κατασκευή της ευθυγράμμισης, εξετάζοντας τις γειτονικές λέξεις της υπό εξέταση ακολουθίας. Οι λέξεις αυτές πρέπει να έχουν σκορ το οποίο να ικανοποιεί μία συνθήκη κατώτατου κατωφλίου T, όταν συγκρίνονται με βάση ένα πίνακα σκοραρίσματος. Το κατώφλι σκοραρίσματος T, καθορίζει εάν μία λέξη θα συμπεριληφθεί στην ευθυγράμμιση ή όχι. Εφόσον έχουν βρεθεί οι αρχικές λέξεις (seeding complete), η ευθυγράμμιση, μήκους μόνο 3 κατάλοιπων, επεκτείνεται προς τις δύο κατευθύνσεις. Η κάθε επέκταση αυξάνει ή μειώνει το σκορ, επομένως όσο αυτό το σκορ παραμένει μεγαλύτερο του προκαθορισμένου κατωφλίου T, η συγκεκριμένη επέκταση θα περιλαμβάνεται στα αποτελέσματα εξόδου του BLAST. Αντίθετα άμα το σκορ προκύψει χαμηλότερο από το προκαθορισμένο κατώφλι T, η ευθυγράμμιση θα σταματήσει να επεκτείνεται, ώστε να μην συμπεριληφθούν περιοχές φτωχής ευθυγράμμισης στα αποτελέσματα εξόδου του BLAST.

2.3.1 Βήματα αλγορίθμου BLAST

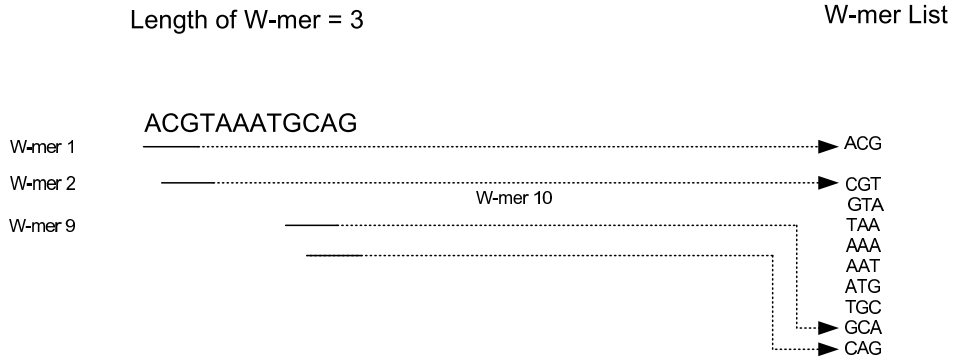
Αρχικά η εκτέλεση του αλγορίθμου BLAST, απαιτεί μία ακολουθία που είναι υπό εξέταση (query sequence) και μία ακολουθία αναφοράς (target sequence) ή βάση δεδομένων από ακολουθίες, σύμφωνα με την οποία θα γίνει η εξέταση. Σε μία τυπική χρήση, η εξεταζόμενη ακολουθία είναι πολύ μικρότερη συγκριτικά με τη βάση δεδομένων.

Η βασική ιδέα του BLAST είναι ότι σε μία στατιστικά σημαντική ακολουθία περιέχονται συνήθως ζεύγη περιοχών υψηλού σκοραρίσματος, *High-Scoring Segment Pairs*. Έτσι ο BLAST αναζητά ακολουθίες ευθυγράμμισης υψηλού σκοραρίσματος μεταξύ της εξεταζόμενης ακολουθίας και των ακολουθιών της βάσης δεδομένων, χρησιμοποιώντας μία ευριστική προσέγγιση που ομοιάζει με τον αλγόριθμο των Smith - Waterman. Επειδή η εξαντλητική προσέγγιση των Smith - Waterman είναι πολύ αργή για αναζήτηση σε μεγάλες βάσεις δεδομένων γονιδιώματος, ο BLAST χρησιμοποιεί μια προσέγγιση που είναι λιγότερο ακριβής συγκριτικά με αυτή των Smith - Waterman, αλλά πάνω από 50 φορές γρηγορότερη. Παρακάτω περιγράφονται τα βήματα του αλγορίθμου **BLASTP**, ο οποίος είναι και η έκδοση του αλγορίθμου BLAST που αναλύεται

στην παρούσα διπλωματική:

1. Απομάκρυνση από την υπό εξέταση ακολουθία, υπακολουθιών που επαναλαμβάνονται. Τέτοιες ακολουθίες μπορεί να δίνουν υψηλό σκορ, γεγονός που μπερδεύει τον αλγόριθμο να βρει τις πραγματικά σημαντικές ακολουθίες στη βάση δεδομένων. Ουσιαστικά πρόκειται για *φιλτράρισμα*

2. Δημιουργία μίας λίστας λέξεων k- γραμμάτων, τα οποία λέγονται *w-mers* της εξεταζόμενης ακολουθίας. Δηλαδή κατασκευάζονται ακολουθιακά λέξεις με μήκος k = 3 για την ακολουθία εισόδου, έως ότου και το τελευταίο γράμμα συμπεριληφθεί. Στο σχήμα 2.5 παρουσιάζεται ένα παράδειγμα:



Σχήμα 2.5: Πρώτο βήμα αλγορίθμου BlastP

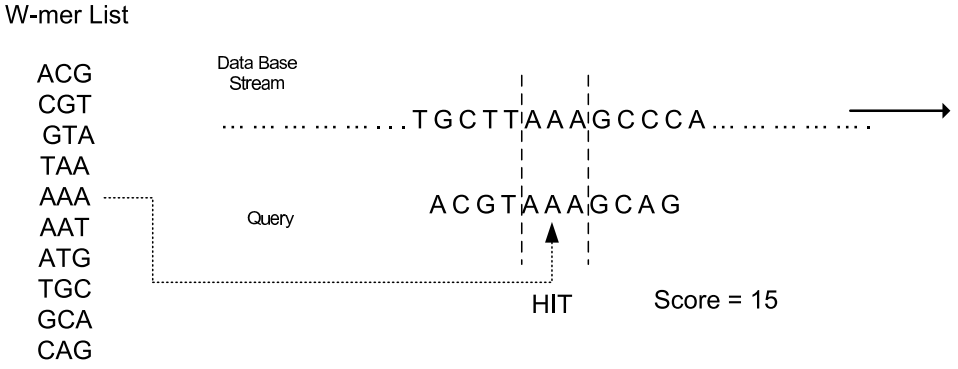
3. Δημιουργία λίστας όλων των πιθανών λέξεων που ταιριάζουν. Στο βήμα αυτό ο BLASTP εξετάζει μόνο τις λέξεις που αποδίδουν υψηλό σκορ. Το σκορ δημιουργείται συγκρίνοντας την εκάστοτε λέξη του βήματος (2) με όλες τις λέξεις k γραμμάτων. Χρησιμοποιείται ένας πίνακας υποκατάστασης (substitution matrix), για να εντοπίζεται το σκορ ανάλογα με τη σύγκριση του κάθε ζευγαριού. Υπάρχουν 20^k πιθανά σκορ ταιριάσματος για κάθε λέξη k γραμμάτων. Στη συνέχεια, το κατώφλι σκοραρίσματος T μίας γειτονικής λέξης χρησιμοποιείται για να μειωθεί ο αριθμός των λέξεων που ταιριάζουν. Οι λέξεις που έχουν σκορ μεγαλύτερο του κατωφλίου T, παραμένουν στη λίστα, ενώ οι υπόλοιπες απορρίπτονται.

4. Οργάνωση των εναπομεινάντων λέξεων υψηλού σκορ σε ένα αποτελεσματικό δέντρο αναζήτησης. Αυτό το βήμα βοηθά στο να γίνει γρήγορα η σύγκριση λέξεων υψηλού σκορ με τις ακολουθίες της βάσης δεδομένων.

5. Επανάληψη όλων των παραπάνω βημάτων για κάθε λέξη k = 3 γραμμάτων της εξεταζόμενης ακολουθίας.

6. Σάρωση των ακολουθιών της βάσης δεδομένων για ακριβές ταίριασμα με τις εναπομένουσες λέξεις υψηλού σκορ. Εάν βρεθεί ακριβές ταίριασμα, χρησιμοποιείται για να τροφοδοτήσει μία πιθανή συνεχής ευθυγράμμιση μεταξύ της εξεταζόμενης ακολουθίας και των ακολουθιών της βάσης δεδομένων.

7. Επέκταση των ακριβών ταιριασμάτων σε high - scoring segments (HSP). Ο BLASTP εκτείνει μία μακρύτερη ευθυγράμμιση μεταξύ της εξεταζόμενης ακολουθίας και της ακολουθίας της βάσης δεδομένων τόσο προς τη δεξιά όσο και προς την αριστερή κατεύθυνση, από το σημείο που παρατηρήθηκε ακριβές ταίριασμα. Η επέκταση δε σταματά έως ότου το συνολικό σκορ των HSP αρχίσει να μειώνεται. Σχηματικά αυτό φαίνεται στο παρακάτω σχήμα 2.6.



Σχήμα 2.6: Η διαδικασία επέκτασης όταν έχουμε ακριβές ταίριασμα

8. Δημιουργία λίστας με όλα τα HSP στη βάση δεδομένων, των οποίων το σκορ είναι αρκετά υψηλό και λαμβάνεται υπόψη. Το σκορ αυτό είναι μεγαλύτερο από το σκορ αποκοπής (cutoff score), το οποίο ορίζεται εμπειρικά. Εξετάζοντας την κατανομή των σκορ ευθυγράμμισης, η οποία προκύπτει από τη σύγκριση των τυχαίων ακολουθιών, μπορεί να προσδιοριστεί ένα σκορ αποκοπής S, με τιμή τέτοια που να εγγυάται τη σημασία των εναπομεινάντων HSP.

9. Υπολογισμός της σημασίας του σκορ των HSP . Ο BLASTP εκτιμά την στατιστική σημασία κάθε HSP σκορ, αξιοποιώντας την κατανομή

ακραίας τιμής του Gumbel *(Extreme Value Distribution - EVD)*. Σύμφωνα με την EVD του Gumbel, η πιθανότητα παρατήρησης p ενός σκορ S ίσου ή μεγαλύτερου του x δίνεται από την εξίσωση:

$p(S \geq x) = 1 - \exp(-e^{-\lambda(x-\mu)})$,

όπου $\mu = \frac{[\log(Km'n')]}{\lambda}$.

Οι στατιστικοί παράγοντες λ και K εξαρτώνται από τον πίνακα υποκατάστασης, από τις ποινές για κενά και τη σύνθεση της ακολουθίας (συχνότητα γραμμάτων). Τα m' και n' είναι τα αποτελεσματικά μήκη για την υπό εξέταση ακολουθία και την ακολουθία της βάσης δεδομένων αντίστοιχα. Πρέπει να σημειωθεί ότι ο BLASTP κάνει έναν ειδικό υπολογισμό για τα μήκη της ακολουθίας εισόδου αλλά και των ακολουθιών της βάσης δεδομένων, και τα ελαττώνει κατά ένα ποσοστό, ώστε να αποφύγει το *edge effect*. Τα μήκη αυτά υπολογίζονται από τις:

$m' \approx m - \frac{(\ln Kmn)}{H}$

και

$n' \approx n - \frac{(\ln Kmn)}{H'}$,

όπου H είναι ο μέσος όρος εκτιμώμενου σκορ για κάθε ευθυγραμμισμένο ζευγάρι κατάλοιπων σε μία ευθυγράμμιση δύο τυχαίων ακολουθιών. Οι Altschul και Gish πρότειναν τις τιμές: $\lambda = 0,318$, K = 0.13, και H = 0,40, συνεχή ευθυγράμμιση (χωρίς κενά), χρησιμοποιώντας ως πίνακα υποκατάστασης τον BLOSUM62. Το αναμενόμενο σκορ E ταιριάσματος μίας βάσης δεδομένων είναι ο αριθμός των φορών που μία ασυσχέτιστη ακολουθία θα αποκτούσε σκορ S υψηλότερο από x κατά τύχη. Η αναμενόμενη τιμή E που λαμβάνεται σε μία αναζήτηση για μία βάση δεδομένων με D ακολουθίες δίνεται από τον τύπο: $E \approx 1 - e{-}p(s > x)D$. Επίσης όταν $p < 0.1$, το E μπορεί να προσεγγιστεί από την Poison κατανομή ως

$E \approx pD$.

10. Κατασκευή μίας μακρύτερης ευθυγράμμισης με χρήση δύο ή περισσοτέρων περιοχών HSP. Αυτό προσφέρει επιπρόσθετη απόδειξη της σχέσης μεταξύ μίας ακολουθίας εισόδου και μίας ακολουθίας της βάσης δεδομένων. Ο BLASTP χρησιμοποιεί τη μέθοδο *Poison*, για εκτίμηση της σημασίας των περιοχών HSP, που συνδυάστηκαν πιο πρόσφατα. Με την υπόθεση ότι υπάρχουν δύο συνδυασμένες περιοχές HSP με σετ σκορ (65, 40) και (52, 45) αντίστοιχα, η μέθοδος Poison δίνει μεγαλύτερο βάρος στο σετ με το μικρότερο σκορ ($45 > 40$).

11. Αναφορά των αποτελεσμάτων των οποίων το σκορ είναι χαμηλότερο του κατωφλίου της παραμέτρου E.

Συμπερασματικά στην περίπτωση που είναι k = 3, δηλαδή για τον BLASTP, η αρχική αναζήτηση θα γίνεται για λέξεις μήκους 3 γραμμάτων που δίνει αποτελέσματα για τουλάχιστον "T", όταν συγκρίνεται με τη ζητούμενη ακολουθία, χρησιμοποιώντας έναν δεδομένο πίνακα υποκατάστασης. Οι λέξεις που έχουν σκορ T ή μεγαλύτερο επεκτείνονται και προς τις δύο κατευθύνσεις αποσκοπώντας να δημιουργηθούν ευθυγραμμίσεις που να επιτυγχάνουν σκορ μεγαλύτερο του προκαθορισμένου κατωφλίου S. Να σημειωθεί ότι η παράμετρος T επηρεάζει την ταχύτητα και την ευαισθησία της αναζήτησης.

2.3.2 Παραλλαγές του αλγορίθμου Blast

Λόγω της μεγάλης δημοφιλίας του αλγορίθμου BLAST, έχουν παραχθεί πολλές παραλλαγές του που εκτελούν διαφορετικές εργασίες. Οι παραλλαγές αυτές εξαρτώνται από την ακολουθία εισόδου καθώς και τη βάση δεδομένων που θα εξεταστεί, καθώς και τι θα συγκριθεί. Παρακάτω αναφέρονται οι πιο γνωστές[42]:

- **blastn**: Συγκρίνει νουκλεοτιδική ακολουθία (DNA) με μία βάση νουκλεοτιδικών ακολουθιών (DNA). Η αναζήτηση γίνεται και στις δύο ακολουθίες. Είναι ένα πρόγραμμα βελτιστοποιημένης ταχύτητας, όχι όμως και ευαισθησίας.

- **blastp**: Συγκρίνει τη ζητούμενη αμινοξεϊκή ακολουθία μία βάση πρωτεϊνικών ακολουθιών (σύγκριση πρωτεΐνης με πρωτεΐνες).

- **blastx**: Συγκρίνει μία άγνωστη νουκλεοτιδική ακολουθία (DNA), μεταφρασμένη σε όλα τα έξι δυνατά πλαίσια ανάγνωσης (reading frames), με μία βάση πρωτεϊνικών ακολουθιών του NCBI. Χρησιμοποιείται για την εύρεση πιθανών μεταφρασμένων πρωτεϊνικών προϊόντων μίας άγνωστης νουκλεοτιδικής ακολουθίας.

- **tblastn**: Συγκρίνει τη ζητούμενη πρωτεϊνική ακολουθία με μία βάση νουκλεοτιδικών ακολουθιών του NCBI που μεταφράζεται δυναμικά σε όλα τα έξι δυνατά πλαίσια ανάγνωσης.

- **tblastx**: Μετατρέπει μία νουκλεοτιδική ακολουθία (DNA) σε μία πρωτεϊνική ακολουθία σε όλα τα έξι δυνατά πλαίσια ανάγνωσης και την συγκρίνει με μία βάση νουκλεοτιδικών ακολουθιών NCBI, η οποία έχει επίσης μεταφραστεί σε όλα τα έξι δυνατά πλαίσια ανάγνωσης. Ο σκοπός του tblastx είναι να βρει τις πιο μακρινές σχέσεις μεταξύ νουκλεοτιδικών ακολουθιών. Τέλος είναι η πιο αργή παραλλαγή από την οικογένεια BLAST.

- **BLAST2**: Ονομάζεται εξελιγμένο (advanced) BLAST. Εκτελεί στοιχίσεις που περιέχουν κενά (gapped alignments).
- **MEGABLAST**: Είναι ένα πρόγραμμα που χρησιμοποιεί έναν άπληστο (greedy) αλγόριθμο (MIller *et al*, 2000), για αναζήτηση στοίχισης νουκλεοτιδικών ακολουθιών. Χρησιμοποιείται για στοίχιση ακολουθιών με μικρές διαφορές και είναι δέκα φορές γρηγορότερο από παρόμοια προγράμματα. Ενδείκνυται για σύγκριση μεγάλων ακολουθιών.
- **PSI - BLAST**: Position Specific Interated BLAST. Χρησιμοποιεί σταθερή αναζήτηση, στην οποία οι ακολουθίες που θα βρεθούν στον πρώτο γύρο αναζητήσεων χρησιμοποιούνται για να χτίσουν ένα αποτελεσματικό μοντέλο για τους επόμενους κύκλους αναζήτησης.
- **PHI - BLAST**: Pattern Hit Initiated BLAST. Συνδυάζει το ταίριασμα ενός πρότυπου φυσιολογικής έκφρασης με μία συγκεκριμένη θέση που επαναλαμβάνεται στην πρωτεϊνική ακολουθία.
- **RPS - BLAST**: Συγκρίνει μία πρωτεϊνική ακολουθία ως προς τη βάση Conserved Domain Database (CD - Search).

2.4 Πίνακες Υποκατάστασης - Συστήματα Score

Για την επίτευξη της στοίχισης είναι αναγκαία η χρήση κάποιου συστήματος αναφοράς ως προς το σκορ, που μετράει την ομοιότητα μεταξύ των αμινοξέων. Υπάρχουν διάφορα είδη σκορ, από τα πλέον απλοϊκά έως πιο σύνθετα. Η πιο απλή προσέγγιση περιγράφεται ως εξής. Για κάθε όμοιο χαρακτήρα προστίθεται στο τρέχον σκορ (+5), ενώ για κάθε ανόμοιο χαρακτήρα προστίθεται (-4). Είναι μία προσέγγιση η οποία δε λαμβάνει υπόψη καθόλου την εξελικτική διαδικασία των αμινοξέων και των μεταλλάξεων που μπορούν να πραγματοποιηθούν. Αυτό επηρεάζει την ευαισθησία του εκάστοτε αλγορίθμου και κατ' επέκταση τα αποτελέσματα που λαμβάνονται. Συνεπώς γίνεται κατανοητό ότι ανεξάρτητα ποιος αλγόριθμος θα χρησιμοποιηθεί (*BLAST*, *FASTA*) το σημαντικότερο ρόλο στη διαδικασία της στοίχισης παίζει το σκορ. Οι συνεισφορές στο σκορ μπορεί να είναι είτε θετικές είτε αρνητικές, όπως για παράδειγμα η προσθήκη κενών αλλά και η ποινή που επιφέρει τυχόν επέκταση τους ώστε να πραγματοποιηθεί η στοίχιση. Επίσης λαμβάνονται υπόψη και οι εξελικτικές αλλά και λειτουργικές σχέσεις των αμινοξέων, χρησιμοποιώντας τους λεγόμενους *πίνακες υποκατάστασης* (*substitution matrices*)[2, 28, 13]. Οι πίνακες αυτοί περιγράφουν τη συχνότητα να μεταλλαχθεί ένα γράμμα μίας ακολουθίας σε ένα διαφορετικό κατά τη διάρκεια του χρόνου. Αναπαριστούν δηλαδή τη χρονική απόκλιση και τα ποσοστά υποκατάστασης μεταξύ των αμινοξέων.

Υπάρχουν πολλοί διαφορετικοί πίνακες υποκατάστασης, με πιο γνωστούς τους **PAM**, Point Accepted Mutation, και ο **BLOSUM**, BLOcks SUbstitution Matrix, με διαστάσεις $20x20$. Οι πίνακες αυτοί περιγράφουν σε ένα γενικό πλαίσιο την εξελικτική σχέση των είκοσι αμινοξέων. Το γεγονός αυτό καθιστά την επιλογή του πίνακα υποκατάστασης πολύ σημαντική καθώς κάθε πίνακας έχει τις ιδιαιτερότητες του. Πιο αναλυτικά ο εκάστοτε πίνακας είναι με τέτοιον τρόπο κατασκευασμένος, ώστε να υλοποιεί τις καλύτερες στοιχίσεις σε συνάρτηση με την εξελικτική σχέση των πρωτεϊνών. Δηλαδή κάποιος πίνακας είναι "κατάλληλος" για να χρησιμοποιηθεί στη στοίχιση δύο πρωτεϊνών που είναι ομόλογες και κάποιος άλλος για δύο πρωτεΐνες που εξελικτικά είναι απομακρυσμένες.

Παρακάτω περιγράφονται οι δύο πιο γνωστοί πίνακες υποκατάστασης. Αρχικά ο *PAM* και στη συνέχεια ο *BLOSUM*.

2.4.1 Πίνακας Υποκατάστασης PAM

PAM: Point Accepted Mutation ή Percent Accepted Mutation[12]. Είναι ένα σύνολο πινάκων που χρησιμοποιούνται για το σκορ στοίχισης μίας ακολουθίας. Ο πρώτος πίνακας δημοσιεύτηκε το 1978 από τον *Margarett Dayhoff*. Κατασκευάστηκε χρησιμοποιώντας γενική στοίχιση ομόλογων - σχετιζόμενων ακολουθιών, οι οποίες είχαν ομοιότητα 85%. Υπάρχουν πολλές εκδόσεις PAM, ανάλογα με τις ακολουθίες, όπως το PAM1, το οποίο είναι "κατάλληλο" για σκορ στοίχισης μεταξύ ακολουθιών που έχουν μεγάλη ομοιότητα. Στον PAM1 δίνεται η πιθανότητα ένα αμινοξύ από τα 100 να μεταλλαχθεί σε ένα δεδομένο χρονικό διάστημα. Αντίθετα στον PAM256 δίνεται η πιθανότητα να γίνουν 256 μεταλλάξεις σε 100 αμινοξέα. Προκύπτει, επομένως, ότι δεν υπάρχει ο πιο "σωστός" πίνακας, αλλά ο πίνακας που επιλέγεται ανάλογα με τα χαρακτηριστικά των ακολουθιών προς στοίχιση.

Ουσιαστικά ο πίνακας PAM1 αποδέχεται ένα τρόπο μετάλλαξης των πρωτεϊνών με βάση το μοντέλο αλυσίδας του Markov. Επίσης, σύμφωνα με τον Dayhoff περίπου το 55% tryptophans, 52% cysteines και το 27% των glycines θα παραμείνουν αμετάβλητες. 'Αλλα αμινοξέα, όπως τα alanine, aspartic ocid, glutamic ocid, glyucine, lycine και serine είναι πιο πιθανό να "πάρουν" τη θέση ενός αρχικού αμινοξέος asparagine σε μία ακολουθία, από ότι το ίδιο το αμινοξύ asparagine σε μία μελλοντική εξέλιξη της ακολουθίας αυτής.

Κάθε τέτοιος πίνακας είναι $20x20$, επειδή υπάρχουν 20 αμινοξέα. Η τιμή την οποία λαμβάνει κάθε θέση του πίνακα αναπαριστά το λογάριθμο της πιθανότητας μετάλλαξης ενός δοθέντος αμινοξέος σε ένα άλλο. Δηλαδή για κάθε συγκεκριμένο ζευγάρι (A_iA_j) δύο αμινοξέων στη θέση (i, j) στον PAM πίνακα που αναπαριστά τη συχνότητα με την οποία το A_i αναμένεται να αντικαταστήσει το A_j μεταξύ των δύο ακολουθιών.

Με μία πιο μαθηματική γραφή, θα δειχθεί πώς προκύπτει ο scoring matrix PAM250, από τον αρχικό πίνακα PAM1. Για τον PAM1 ισχύουν τα παρακάτω:

- Υπάρχει ένα σύνολο αποδεκτών μεταλλάξεων.
- Η πιθανότητα μετάλλαξης p_a για κάθε αμινοξύ α.

Το σύνολο αυτό εξήχθη από τον Dayhoff και με βάση αυτό μπορεί να υπολογιστεί ο αριθμός των φορών που παρατηρήθηκε η μετάλλαξη $a \leftrightarrow b$, f_{ab}. Επίσης ισχύει $f_{ab} = f_{ba}$, αφού οι μεταλλάξεις δεν έχουν κατεύθυνση . Επίσης χρειάζεται ο συνολικός αριθμός μεταλλάξεων, στις οποίες το αμινοξύ a υπήρξε:

$$f = \sum_a f_a, \tag{2.1}$$

καθώς και το συνολικό αριθμό που τα αμινοξέα εμφανίστηκαν σε μετάλλαξη:

$$f_a = \sum_{b \neq a} f_{ab} \tag{2.2}$$

M_{aa} είναι η πιθανότητα ένα αμινοξύ να παραμείνει αμετάβλητο στη διάρκεια ενός εξελικτικού διαστήματος. Η σχετική μετάλλαξη είναι η πιθανότητα να αλλάξει ένα αμινοξύ a στην εξελικτική διάρκεια που εξετάζουμε, η τιμή της δίνεται από τον τύπο: $m_a = \frac{f_a}{100 f p_a}$.

Επομένως η πιθανότητα ένα αμινοξύ να παραμείνει αμετάβλητο είναι η συμπληρωματική πιθανότητα:

$M_{aa} = 1 - m_a$

Αντίθετα η πιθανότητα ένα αμινοξύ a να μεταλλαχθεί σε ένα αμινοξύ b, μπορεί να συμβολιστεί ως M_{ab} και η πιθανότητα της μπορεί να υπολογιστεί ως το γινόμενο της δεσμευμένης πιθανότητας του a να μεταλλαχθεί στο β, δοθέντος ότι το a μεταλλάχθηκε, επί την πιθανότητα του a να μεταλλαχθεί:

$$\begin{aligned} M_{ab} &= P(a \longrightarrow b) \\ &= P(a \longrightarrow b | a\ changed) P(a\ changed) \\ &= \frac{f_{ab}}{f_a} m_a \end{aligned} \tag{2.3}$$

.

Από στοιχεία πιθανοτήτων προκύπτουν τα παρακάτω αθροίσματα.

$$\sum_b M_{ab} = 1 \tag{2.4}$$

$$\sum_a p_a M_{aa} = 0.99 \tag{2.5}$$

.

Οπότε έχει κατασκευαστεί ο πίνακας μεταλλάξεων, ο οποίος έχει κανονικοποιηθεί για να αναπαριστά το γεγονός ότι κατά μέσο όρο 1 στα 100 αμινοξέα θα μεταλλαχθεί - έχει ληφθεί υπόψη και η λίστα με τις αποδεκτές μεταλλάξεις. Οπότε τώρα μπορεί να οριστεί ο πίνακας υποκατάστασης. Οι εισαγωγές σε αυτό τον πίνακα έχουν σχέση με το λόγο των δύο πιθανοτήτων, δηλαδή, την πιθανότητα να έχει συμβεί ένα ζεύγος μετάλλαξης από το να είναι απλή εμφάνιση. Αυτό ονομάζεται λόγος απόδοσης (likelihood) $\frac{M_{ab}}{p_b}$. Κάθε εισαγωγή (λογαριθμική πιθανότητα) στον πίνακα S υπολογίζεται

$$S_k ab = 10 \log_{10} \frac{M_{ab}^k}{p_b} \tag{2.6}$$

Συμπερασματικά ο πίνακας υποκατάστασης PAM250, φαίνεται στο σχήμα 2.7. Τέλος να σημειωθεί ότι υπάρχουν κάποιοι περιορισμοί που διέπουν τους πίνακες PAM. Το σύνολο δεδομένων με το οποίο κατασκευάστηκε ο αρχικός πίνακας είναι αρκετά παλιό. Επίσης ένα σημαντικό χαρακτηριστικό είναι ότι σε όλους γίνεται η υπόθεση ότι όλες οι μεταλλάξεις συμβαίνουν με την ίδια συχνότητα, το οποίο βιολογικά δεν είναι αποδεκτό, αν και μαθηματικά είναι βολικό.

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	2	-2	0	0	-2	0	0	1	-1	-1	-2	-1	-1	-3	1	1	1	-6	-3	0
R	-2	6	0	-1	-4	1	-1	-3	2	-2	-3	3	0	-4	0	0	-1	2	-4	-2
N	0	0	2	2	-4	1	1	0	2	-2	-3	1	-2	-3	0	1	0	-4	-2	-2
D	0	-1	2	4	-5	2	3	1	1	-2	-4	0	-3	-6	-1	0	0	-7	-4	-2
C	-2	-4	-4	-5	12	-5	-5	-3	-3	-2	-6	-5	-5	-4	-3	0	-2	-8	0	-2
Q	0	1	1	2	-5	4	2	-1	3	-2	-2	1	-1	-5	0	-1	-1	-5	-4	-2
E	0	-1	1	3	-5	2	4	0	1	-2	-3	0	-2	-5	-1	0	0	-7	-4	-2
G	1	-3	0	1	-3	-1	0	5	-2	-3	-4	-2	-3	-5	0	1	0	-7	-5	-1
H	-1	2	2	1	-3	3	1	-2	6	-2	-2	0	-2	-2	0	-1	-1	-3	0	-2
I	-1	-2	-2	-2	-2	-2	-2	-3	-2	5	2	-2	2	1	-2	-1	0	-5	-1	4
L	-2	-3	-3	-4	-6	-2	-3	-4	-2	2	6	-3	4	2	-3	-3	-2	-2	-1	2
K	-1	3	1	0	-5	1	0	-2	0	-2	-3	5	0	-5	-1	0	0	-3	-4	-2
M	-1	0	-2	-3	-5	-1	-2	-3	-2	2	4	0	6	0	-2	-2	-1	-4	-2	2
F	-3	-4	-3	-6	-4	-5	-5	-5	-2	1	2	-5	0	9	-5	-3	-3	0	7	-1
P	1	0	0	-1	-3	0	-1	0	0	-2	-3	-1	-2	-5	6	1	0	-6	-5	-1
S	1	0	1	0	0	-1	0	1	-1	-1	-3	0	-2	-3	1	2	1	-2	-3	-1
T	1	-1	0	0	-2	-1	0	0	-1	0	-2	0	-1	-3	0	1	3	-5	-3	0
W	-6	2	-4	-7	-8	-5	-7	-7	-3	-5	-2	-3	-4	0	-6	-2	-5	17	0	-6
Y	-3	-4	-2	-4	0	-4	-4	-5	0	-1	-1	-4	-2	7	-5	-3	-3	0	10	-2
V	0	-2	-2	-2	-2	-2	-2	-1	-2	4	2	-2	2	-1	-1	-1	0	-6	-2	4

Σχήμα 2.7: Scoring matrix PAM250

2.4.2 Πίνακας Υποκατάστασης BLOSUM

BLOSUM: BLOcks of Amino Acid SUbstitution Matrix[7, 35][36]. Το 1992, οι *Steve* και *Jorgia Henikoff* ήθελαν να μοντελοποιήσουν πρωτεΐνες που ήταν αρκετά αποκλίνουσες. Το σκεπτικό τους ήταν να κατασκευάσουν έναν πίνακα από τις ακολουθίες που είχαν μακρινή συγγένεια, αντί να κατασκευάσουν τον πίνακα ως αποτέλεσμα της ύψωσης σε μία δύναμη του PAM1. Για το λόγο αυτό χρησιμοποίησαν τοπική στοίχιση ακολουθιών, όπου καμία ακολουθία δεν είχε ομοιότητα λιγότερη του 62%. Πιο συγκεκριμένα έγινε χρήση μίας μεγάλης βάσης δεδομένων, από πιστοποιημένες στοιχίσεις (Block Database), για πολύ "conserved" περιοχές πρωτεϊνικών οικογενειών, που είναι δηλαδή χωρίς κενά στην ακολουθία στοίχισης. Έτσι προέκυψε και ο όρος BLOSUM από το BLOcks SUbstitution Matrix. Η επιλογή των blocks έγινε λόγω της ανάγκης για πολλαπλή στοίχιση. Αυτό έχει ως αποτέλεσμα ευκολότερη ευθυγράμμιση για όμοιες ακολουθίες. Επίσης δεν είναι επιθυμητή η εισαγωγή ή η διαγραφή, επειδή καθιστούν πιο περίπλοκο τον υπολογισμό των πιθανοτήτων υποκατάστασης. Τέλος υπάρχει μεγαλύτερο ενδιαφέρον για ανίχνευση conserved περιοχών στις πρωτεϊνικές ακολουθίες, οπότε η προσοχή στρέφεται σε αυτές τις περιοχές, όταν υπολογίζεται ο πίνακας υποκατάστασης.

Διαφορετικοί τύποι πινάκων υποκατάστασης BLOSUM μπορούν να κατασκευαστούν από τη στάθμιση του διαφορετικού βαθμού ομοιότητας μεταξύ των ακολουθιών. Για παράδειγμα ο πίνακας BLOSUM62 υπολογίζεται από τα πρωτεϊνικά blocks, τέτοια ώστε εάν δύο ακολουθίες έχουν παραπάνω από 62% ομοιότητα, τότε η συνεισφορά αυτών των ακολουθιών σταθμίζεται στην άθροιση τους. Με αυτόν τον τρόπο η συνεισφορά των πολλαπλών εγγραφών από "συγγενικές" ακολουθίες είναι μειωμένη. Για την κατασκευή των πινάκων BLOSUM θα υπολογιστεί, όπως και στους πίνακες PAM το σκορ ως ο λογάριθμος του λόγου της παρατηρημένης πιθανότητας μετάλλαξης ενός αμινοξέος σε ένα άλλο, διαιρούμενη με την τυχαία πιθανότητα εμφάνισης της μετάλλαξης. Οι πίνακες BLOSUM βασίζονται σε παρατηρημένες στοιχίσεις και όχι σε αποτελέσματα που έχουν προκύψει από συγκρίσεις πρωτεϊνών, οι οποίες έχουν υψηλή ομοιότητα, όπως συμβαίνει στους PAM πίνακες. Στο σχήμα 2.8 παρουσιάζεται ο BLOSUM62, που προκύπτει από τους παρακάτω υπολογισμούς. Αρχικά υπολογίζεται ο αριθμητής:

1. Μέτρηση της συχνότητας εμφάνισης $c_{ij}^{(k)}$ ζευγαριού για κάθε ζευγάρι αμινοξέων i, j, για κάθε στήλη k του κάθε block. Οι τιμές αυτές υπολογίζονται σύμφωνα με τους τύπους:

$$c_{ii}^{(k)} = \binom{n_i}{2} \tag{2.7}$$

και

$$c_{ij}^{(k)} = n_i n_j \tag{2.8}$$

, όπου n_i = ο αριθμός που το i κατάλοιπο παρατηρήθηκε στην στήλη.

2. 'Αθροιση του σκορ για κάθε στήλη:

$$c_i j = \sum_k c_{ij}^{(k)} \tag{2.9}$$

3. Κανονικοποίηση των ζευγών συχνότητας ώστε να αθροίζουν 1:

$$T = \sum_{i \geq j} c_{ij} = w \frac{n(n-1)}{2} \tag{2.10}$$

όπου w: ο αριθμός των στηλών και n: ο αριθμός των ακολουθιών.

$$q_{ij} = \frac{c_{ij}}{T}. \tag{2.11}$$

Για τον υπολογισμό του παρανομαστή του λόγου ακολουθούνται τα παρακάτω βήματα:

1. Υπολογισμός της αναμενόμενης πιθανότητας να εμφανιστεί το i κατάλοιπο σε ένα ζευγάρι (i, j):

$$p_i = q_{ii} + \sum_{j \neq i} \frac{q_{ij}}{2} \tag{2.12}$$

.

2. Ο επιθυμητός παρανομαστής είναι η αναμενόμενη συχνότητα για κάθε ζεύγος, υποθέτοντας ότι έχουμε στατιστική ανεξαρτησία.

$$e_{ii} = p_i^2 \tag{2.13}$$

και

$$e_{ij} = 2 p_i p_j \tag{2.14}$$

για $i \neq j$.

3. Οπότε κάθε εισαγωγή (i, j) στον λογαριθμικό πίνακα αποδόσεων είναι ίση με $\frac{q_{ij}}{e_{ij}}$.

4. Ο λόγος της λογαριθμικής απόδοσης είναι:

$$s_{ij} = \log_2 \frac{q_{ij}}{e_{ij}} \tag{2.15}$$

Αφού αποθηκευτούν οι τιμές των λόγων στον πίνακα προκύπτει ο ζητούμενος BLOSUM πίνακας. Οι θετικές τιμές σκορ αναφέρονται σε πιο πιθανές να συμβούν υποκαταστάσεις, ενώ οι αρνητικές για λιγότερο πιθανές υποκαταστάσεις. Όταν ο αριθμός πίνακα BLOSUMx , όπου ο αριθμός x δηλώνει το βαθμό

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

Σχήμα 2.8: Scoring matrix BLOSUM62

ομοιότητας μεταξύ των ακολουθιών, είναι χαμηλός, ακολουθίες με υψηλή συγγένεια συμπυκνώνονται και δίνεται έμφαση στις πιο ποικίλες ακολουθίες του πίνακα. Εάν ο αριθμός του πίνακα BLOSUMx είναι υψηλός, μόνο οι σχεδόν πανομοιότυπες ακολουθίες συμπυκνώνονται και δίνεται έμφαση στις ακολουθίες που έχουν μικρότερη ποικιλία. Εμπειρικά οι πίνακες BLOSUM έχουν συμπεριφερθεί πολύ καλά, ενώ ο BLOSUM62 έχει γίνει το *de facto* πρότυπο για πολλά προγράμματα στοίχισης πρωτεϊνών, όπως ο BlastP.

2.4.3 Σύγκριση πίνακα PAM - πίνακα BLOSUM

Συγκεντρωτικά[4] μία σύγκριση μεταξύ των δύο τύπου πινάκων υποκατάστασης θα δείξει ότι οι PAM πίνακες βασίζονται σε ένα ξεκάθαρο εξελικτικό

μοντέλο, χρησιμοποιώντας φυλογενετικά δέντρα. Σύμφωνα με αυτό, οι πιο μακρινές συγγενικά μεταλλάξεις, είναι η αντανάκλαση των τμηματικά μικρών μεταλλάξεων που επαναλαμβάνονται, και για το λόγο αυτό μπορούν να δουλέψουν για ένα μεγάλο εύρος ποικιλομορφίας. Όμως οι υποθέσεις για το μοντέλο παραβιάζονται ξεκάθαρα, επειδή η κάθε θέση είναι εξαρτώμενη από τα συμφραζόμενα (γενικό πλαίσιο), ενώ οι πιο σπάνιες μεταλλάξεις είναι περισσότερο υποκείμενες σε δειγματικό λάθος, ειδικά όσο ο αριθμός n του PAMn είναι μεγάλο.

Οι BLOSUM πίνακες δε βασίζονται σε κάποιο εξελικτικό μοντέλο, αλλά έχουν προκύψει τελείως εμπειρικά. Βασίζονται σε συγκρίσεις ακολουθιών καλύπτοντας ένα ευρύ φάσμα ποικιλότητας. Οι BLOSUM πίνακες είναι πιο ευαίσθητοι σε ακολουθίες με χαμηλή ομοιότητα σε σχέση με τους PAM πίνακες. Παρόλα αυτά είναι περιορισμένοι σε ένα υποσύνολο conserved τομέων. Τέλος υπονοείται ένα μοντέλο εξέλιξης "star- tree", το οποίο αγνοεί την κλειστότητα της σχέσης.

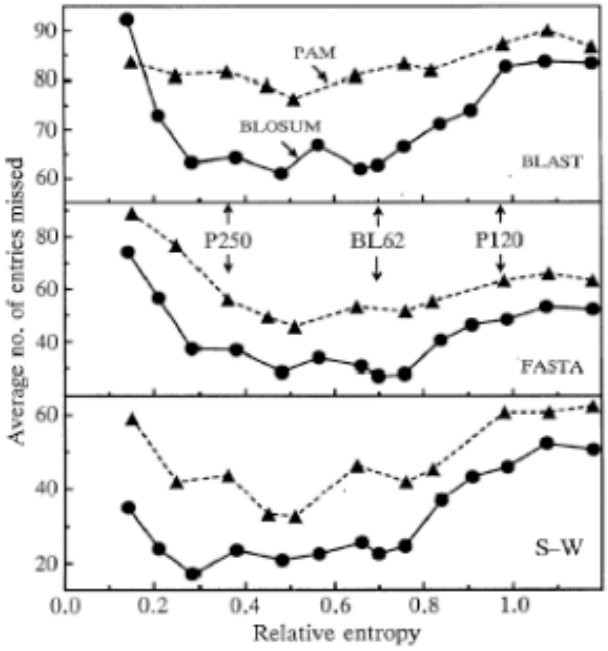
Οι διαφορές αυτές οδηγούν στους εξής κανόνες:

- Χρήση χαμηλού βαθμού ομοιότητας πίνακα PAM ή υψηλού βαθμού ομοιότητας πίνακα BLOSUM, όταν η εξελικτική απόσταση είναι μικρή και οι ακολουθίες έχουν μικρή απόκλιση.

- Χρήση υψηλού βαθμού ομοιότητας πίνακα PAM ή χαμηλού βαθμού ομοιότητας πίνακα BLOSUM, όταν η εξελικτική απόσταση είναι μεγάλη και οι ακολουθίες έχουν μεγάλη απόκλιση.

Ένα συγκεντρωτικό διάγραμμα σύγκρισης των δύο παραπάνω πινάκων υποκατάστασης φαίνεται στο σχήμα 2.9, που η σύγκριση πραγματοποιείται τόσο για τον αλγόριθμο BLAST, όσο και για τους FASTA και Smith - Waterman. Το κριτήριο που χρησιμοποιείται είναι η *relative entropy*, η οποία μπορεί να χρησιμοποιηθεί για σύγκριση διαφορετικών πινάκων υποκατάστασης. Η εντροπία καταδεικνύει την ικανότητα του συστήματος να διακρίνει τον "θόρυβο". Η relative εντροπία μίας τυχαίας ευθυγράμμισης πρέπει να είναι αρνητική. Η τιμή της δίνεται από τον τύπο:

$$H = \sum_{i=1}^{20} \sum_{j=1}^{i} p_i p_j (i,j) \tag{2.16}$$



Σχήμα 2.9: Σύγκριση πινάκων PAM και BLOSUM τριών διαφορετικών βαθμών ομοιότητας, x =250, 62 , 120, με κριτήριο την relative εντροπία, κατηγοριοποιημένα για τις μεθόδους BLAST, FASTA και Smith - Waterman.

2.5 Προηγούμενες προσεγγίσεις υλοποίησης του αλγορίθμου BLAST

Λόγω της μεγάλης δημοφιλίας του προβλήματος αναζήτησης ομοιότητας μεταξύ δύο ακολουθιών έχουν πραγματοποιηθεί αρκετές προσεγγίσεις υλοποίησης του αλγορίθμου BLAST. Επειδή αυτό το πρόβλημα είναι υπολογιστικά πολύ απαιτητικό, υπήρξε η επιθυμία να παρουσιαστεί ένας τρόπος επίλυσης του χρησιμοποιώντας FPGA's. Ιστορικά η πρώτη απόπειρα πραγματοποιήθηκε στις αρχές του 1990, όταν χρησιμοποιήθηκε η πλατφόρμα Splash 2 από τον Hoang[23] για να λυθεί το πρόβλημα με βάση τον αλγόριθμο των Smith - Waterman. Αργότερα ο Guccuine[21] χρησιμοποίησε τεχνολογία Jbits, ενώ το Virginia Tech Configurable Computing Labratory[33] μαζί με το Nayang Technological University[32] χρησιμοποίησαν χρονική εκτέλεση αναδιάταξης (run time reconfiguration) για το ίδιο πρόβλημα, υλοποιώντας επίσης τον αλγόριθμο Smith - Waterman.

'Εως σήμερα έχουν πραγματοποιηθεί λίγες ακαδημαϊκές προσεγγίσεις για τον αλγόριθμο BLAST. Η πρώτη απόπειρα ήταν ο RC-BLAST project[30],

στην οποία οι σχεδιαστές υλοποίησαν πλήρως τον αλγόριθμο NCBI BLAST, αλλά η απόδοση εκτιμήθηκε χαμηλότερη ακόμα και από τα αντίστοιχα προγράμματα υλοποίησης σε software. Μία πιο πρόσφατη ακαδημαϊκή προσπάθεια για ταυτοποίηση DNA ακολουθιών και αναζήτηση σε βάση δεδομένων παρουσιάστηκε το 2005 από το εργαστήριο CAAD του πανεπιστημίου της Βοστώνης[22]. Υλοποίησαν τον αλγόριθμο BLAST για μικρές ακολουθίες (queries) με έως και 800 δεδομένα, επιτυγχάνοντας σημαντική επιτάχυνση συγκριτικά με τις αντίστοιχες υλοποιήσεις σε software. Το έργο αυτό ονομάστηκε TreeBLAST. Η αρχιτεκτονική parallel Mercury BLAST[24, 25] προτάθηκε από το πανεπιστήμιο της Ουάσινγκτον, υλοποιώντας τον BLASTn και προσφέροντας μία καλή επιτάχυνση σε σχέση με τα αντίστοιχα software προγράμματα, τα οποία έτρεχαν σε υπολογιστές γενικού σκοπού. Η πιο πρόσφατη υλοποίηση BLAST χρησιμοποιώντας αναδιατασσόμενη λογική παρουσιάστηκε από την IRISA, CNRS στη Γαλλία και το τεχνολογικό ινστιτούτο του Πεκίνου στην Κίνα[26]. Είναι μία πλακέτα με αναδιατασσόμενη λογική συνδυασμένη με μία μνήμη 64 GB FLASH και χρησιμοποιήθηκε για την υλοποίηση του αλγορίθμου BLASTx/TBLASTn/TBLASTx.

Παράλληλα με τις ακαδημαϊκές προσπάθειες πραγματοποιήθηκαν και εμπορικές που είχαν ως στόχο να τρέξουν αποδοτικά τον αλγόριθμο BLAST. Η Timelogic Inc. αναφέρει για το σύστημα DeCrypher πολύ εντυπωσιακά αποτελέσματα για την υλοποίηση του BLAST, χωρίς όμως να δίνει λεπτομέρειες ούτε για τα δεδομένα εισόδου που χρησιμοποιήθηκαν για να τρέξουν τα πειράματα ούτε για τον αριθμό των chip του συστήματος αλλά ούτε και την στρατηγική I/O που ακολουθήθηκε. Για τους λόγους αυτούς τα αποτελέσματα δεν μπορούν να συγκριθούν με τα υπόλοιπα. Το πανεπιστήμιο της Καλιφόρνια στο Berkeley κατασκεύασε μία μηχανή BEE ως διατασσόμενο υπερυπολογιστή και μία εφαρμογή του έργου ήταν ο αλγόριθμος BLAST[18]. Η μηχανή BEE-2 αναφέρθηκε να είναι δύο φορές πιο γρήγορη από το DeCrypher, αλλά χωρίς να δίνονται λεπτομέρειες απόδοσης, οπότε ούτε αυτά τα αποτελέσματα μπορούν να συγκριθούν με τα υπόλοιπα.

Τέλος έχουν γίνει προσπάθειες και στο Πολυτεχνείο Κρήτης από μεταπτυχιακούς και προπτυχιακούς φοιτητές. Η υλοποίηση μίας μηχανής κατάλληλης για όλες τις εκδόσεις του αλγορίθμου BLAST και για κάθε σετ δεδομένων είναι η διδακτορική διατριβή που υλοποιεί ο κ. Σωτηριάδης. Ο κ. Κοζανίτης στη διπλωματική του εργασία είχε επίσης υλοποιήσει δύο διαφορετικές γενιές μηχανών που υλοποιούσαν τον αλγόριθμο BLASTn[40, 38, 39]. Ο κ. Αφράτης είχε επίσης μελετήσει τις ιδιότητες του αλγορίθμου και είχε προτείνει τη δημιουργία ενός φίλτρου δεδομένων για τον αλγόριθμο[11, 10]. ενώ ο κ. Γαλανάκης είχε υλοποιήσει στα πλαίσια της διπλωματικής του εργασίας ένα φίλτρο Bloom για το σύστημα σύμφωνα με τη μελέτη του κ. Αφράτη[9]. Τέλος ο κ. Μπλεμένος στη διπλωματική του εργασία είχε χρησιμοποιήσει τον αλγόριθμο

BLAST σαν benchmark στον πολυεπεξεργαστή MPLEM που είχε υλοποιήσει.

Οι παραπάνω υλοποιήσεις και μελέτες στις διπλωματικές εργασίες αφορούν όλες την έκδοση BLASTn του αλγορίθμου και έχουν δώσει έμφαση στο δεύτερο (βασικό) βήμα του. Στην εργασία που παρουσιάζεται έχει υλοποιηθεί η έκδοση BLASTp του αλγορίθμου έχοντας δοθεί έμφαση στο τρίτο βήμα του (βήμα επέκτασης).

2.6 " Τι καινούργιο προσφέρει η παρούσα διπλωματική εργασία? "

Η συγκεκριμένη διπλωματική εργασία υλοποιεί σε hardware την έκδοση BLASTp του αλγορίθμου BLAST, που αφορά στην αναζήτηση ομοιότητας μεταξύ πρωτεϊνικών ακολουθιών. Πιο αναλυτικά υλοποιείται αρχικά το δεύτερο βήμα του αλγορίθμου, σύμφωνα με την υπάρχουσα αρχιτεκτονική σχεδίαση, πραγματοποιώντας αλλαγές στις δομικές μονάδες που είναι απαραίτητο να γίνουν. Για το τρίτο βήμα του αλγορίθμου πραγματοποιείται νέα αρχιτεκτονική σχεδίαση, η οποία αφορά στη διαδικασία της επέκτασης και πώς μπορεί να γίνει γρήγορα. Στη συνέχεια η αρχιτεκτονική υλοποιείται σε hardware. Επίσης έχει ενσωματωθεί νέο Scoring Scheme, το οποίο λαμβάνεται υπόψη η εξελικτική διαδικασία των αμινοξέων στη διάρκεια του χρόνου. Έτσι κάθε ζεύγος αμινοξέων αντιστοιχεί σε συγκεκριμένο score (αρνητικό ή θετικό), το οποίο προστίθεται στο τρέχων score της επέκτασης. Έτσι κατά την επέκταση δε χρησιμοποιείται ο γενικός κανόνας scoring (όπου γινόταν πρόσθεση (+5) σε περίπτωση ομοιότητας και πρόσθεση (-4) σε περίπτωση μη ομοιότητας μεταξύ δύο αμινοξέων), αλλά χρησιμοποιείται ένας πίνακας scoring, ο οποίος αποδίδει συγκεκριμένη τιμή για κάθε ζεύγος αμινοξέων. Το γεγονός αυτό είναι ιδιαιτέρως σημαντικό, καθώς συνιστά πιο ορθή επιλογή από βιολογικής πλευράς. Συνεπώς έχει χρησιμοποιηθεί ο scoring matrix BLOSUM62, ο οποίος αποτελεί μία πολύ επιλογή για ακολουθίες που έχουν ομοιότητα 62% και πάνω, γεγονός που τον καθιστά de facto επιλογή για πολλές εφαρμογές. Επίσης η συγκεκριμένη διαδικασία της επέκτασης που έχει υλοποιηθεί, πραγματοποιεί τριπλή επέκταση ανά φορά βήματος του αλγορίθμου και όχι μίας απλής επέκτασης που πραγματοποιούταν σε προηγούμενες υλοποιήσεις. Αυτό έχει ως πλεονέκτημα να πραγματοποιείται πολύ γρήγορα η επέκταση, από τη στιγμή που γίνονται 3 extensions ανά κύκλο ρολογιού.

Κεφάλαιο 3

Περιγραφή Αρχιτεκτονικής

3.1 Εισαγωγή στην Αρχιτεκτονική

Τα βήματα που σχεδιάστηκαν και υλοποιήθηκαν στην παρούσα διπλωματική είναι τα βήματα που παρουσιάστηκαν και στο προηγούμενο κεφάλαιο. Η σχεδίαση έχει χωριστεί σε δύο stages. Το σκεπτικό της υλοποίησης έχει "σπάσει" τη λειτουργία του αλγορίθμου σε δύο βασικά κομμάτια. Στο πρώτο κομμάτι, που περιγράφεται από την πρώτη βαθμίδα, αφορά παρασκευαστικά βήματα του αλγορίθμου BLASTp, δηλαδή την δημιουργία των λιστών με τα wmers και την αναζήτηση ταυτοποίησης ενός wmer της υπό εξέταση ακολουθίας με ένα της βάσης δεδομένων. Η αναζήτηση συνεχίζει για όλα τα wmers της βάσης δεδομένων μέχρι να βρεθεί matching. Στην περίπτωση που δεν βρεθεί matching το wmers που συγκρίθηκε με το αντίστοιχο στης υπό εξέταση ακολουθίας κρατιέται σε μία τρίτη λίστα, να υπάρχει διαθέσιμο στην περίπτωση που χρειαστεί κατά τη διάρκεια της επέκτασης (ενώ έχει γίνει matching σε κάποιο άλλο γειτονικό του wmer). Στην περίπτωση που βρεθεί matching σταματάει η λειτουργία της πρώτης βαθμίδας και ξεκινάει η λειτουργία του δεύτερου κομματιού. Το δεύτερο κομμάτι περιγράφεται από τη δεύτερη βαθμίδα, στην οποία πραγματοποιείται το βήμα της επέκτασης του αλγορίθμου, ενώ ελέγχεται και η εγκυρότητά της. 'Οσο η επέκταση είναι έγκυρη, συνεχίζεται, διαφορετικά σταματάει και αδρανοποιείται η δεύτερη βαθμίδα επιστρέφοντας παράλληλα τη λειτουργία στην πρώτη. Τα παραπάνω εκτελούνται μέχρι να εξαντληθεί η βάση δεδομένων ή να γίνει πλήρης ταυτοποίηση της υπό εξέταση ακολουθίας. Παρακάτω παρουσιάζεται σε μορφή ψευδοκώδικα οι ενέργειες που εκτελούνται στη συνολική σχεδίαση:

```
// λειτουργία αλγορίθμου
 main()
```

```
{
    // η είσοδος μπορεί να είναι είτε query είτε database
    input mode();
    if (query)
    {
        // κατασκευή των wmers
        create wmers();
        // δημιουργία της λίστας με τα wmer
        do ListOfWmers();
    }
    else
    {
      if(database)
        {
          // κατασκευή των λέξεων
          create words();
          // δημιουργία της λίστας λέξεων
          do ListOfWords();
          // κατασκευή των database wmers
          readWordbyChar();
          check(*readWordbyChar, *ListOfWmers)
          // βρέθηκε matching
          if (true)
          {
              // απενεργοποιείται η πρώτη βαθμίδα
              pause_stage1();
              // ενεργοποιείται η δεύτερη βαθμίδα
              enable_stage2();
              // πραγματοποιείται επέκταση κατά τρεις χαρακτήρες
              extention();
              // Χρήση score από τον πίνακα BLOSUM62
              blosum62Score();
              checkValidextention();
              // η επέκταση είναι έγκυρη, όποτε συνεχίζεται ..
              if (true)
                  extention();
              // η επέκταση δεν είναι έγκυρη, οπότε σταματάει ..
              else
              {
                 // ενεργοποιείται η πρώτη βαθμίδα
                 enable_stage1();
```

```
                    // απενεργοποιείται η δεύτερη βαθμίδα
                    pause_stage2();
                }
            // δε βρέθηκε matching
            else
                // συνέχισε την κατασκευή λέξεων
                continue();
            }
        }
    }
}
```

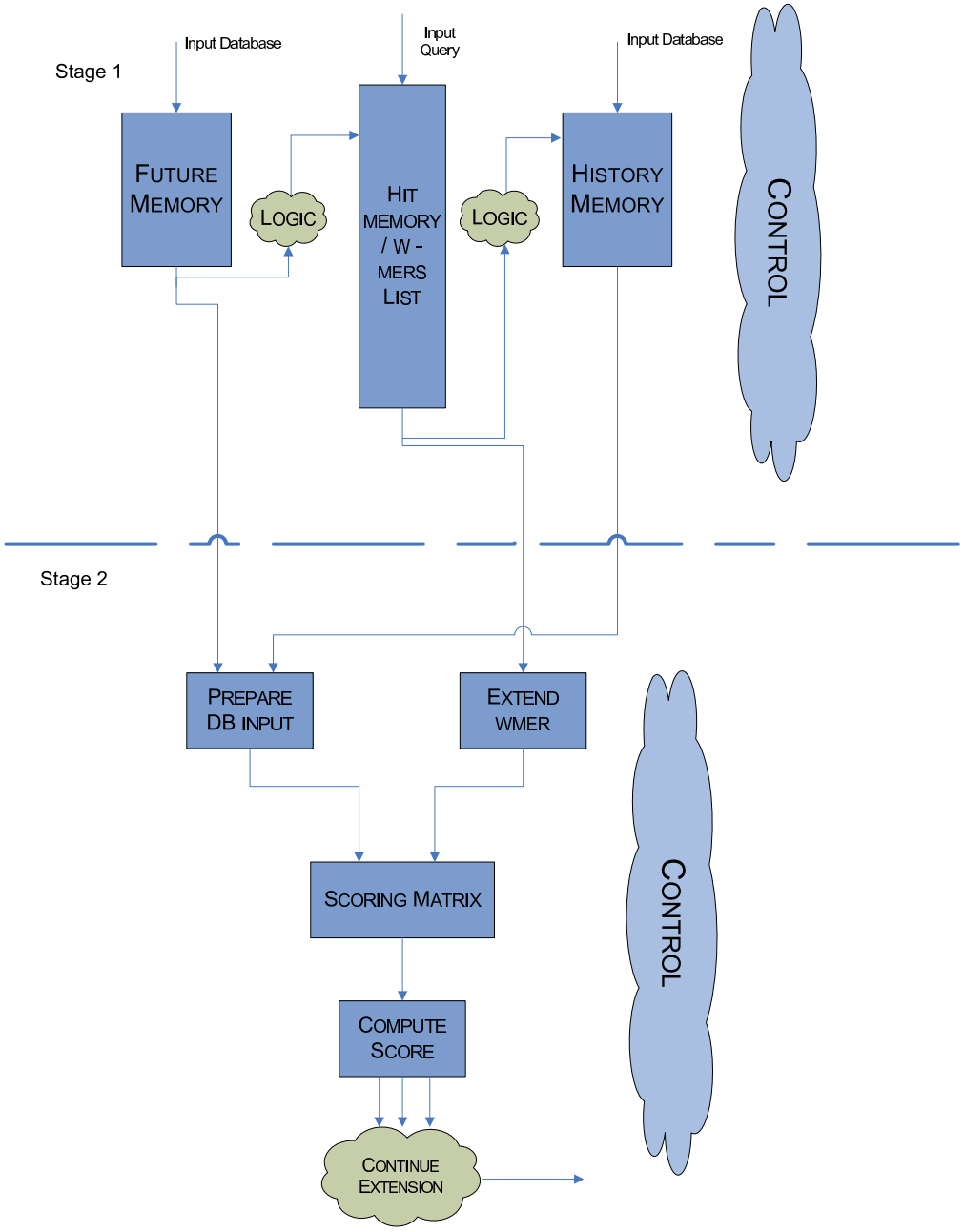
Από την παραπάνω περιγραφή σε ψευδοκώδικα γίνεται αντιληπτό ότι εάν δουλεύει η πρώτη βαθμίδα, δε μπορεί να δουλεύει η δεύτερη βαθμίδα. Γενικά ελέγχεται μία έξοδος της πρώτης βαθμίδας, τα δεδομένα της hitmemory, και ανάλογα την τιμή τους προκαλείται η ενεργοποίηση ή όχι της δεύτερης βαθμίδας. Επομένως οι δύο βαθμίδες λειτουργούν συμπληρωματικά η μία προς την άλλη. Παρακάτω θα περιγραφούν οι δύο βαθμίδες ξεχωριστά, καθώς και το σκεπτικό υλοποίησης του καθενός, αλλά και των δομικών μονάδων που εμπεριέχονται σε αυτά. Αρχικά δίνεται ένα σχηματικό - συνολικό διάγραμμα της συνολικής σχεδίασης 3.1

3.2 Δεδομένα Εισόδου

Αρχικά θα πρέπει να αναφερθεί ποια είναι τα δεδομένα μας και η μορφή στην οποία χρησιμοποιούνται. Ο αλγόριθμος Blastp αφορά ακολουθίες πρωτεϊνών, συνεπώς τα δεδομένα μας είναι συνδυασμός των 20 γνωστών αμινοξέων. Κάθε γράμμα - αμινοξύ της ακολουθίας αντιστοιχίζεται σε μία μοναδική δυαδική μορφή. Η αναπαράσταση που χρησιμοποιείται φαίνεται στον παρακάτω πίνακα 3.1.

3.3 Περιγραφή Πρώτης Βαθμίδας

Οι δομικές μονάδες που αποτελούν το πρώτο stage είναι λειτουργικές μονάδες, οι οποίες χρησιμοποιούνται για να εκτελούν συγκεκριμένες διεργασίες ή να αποθηκεύουν δεδομένα. Ο πιο λογικός τρόπος για την σχεδίαση μίας διαδρομής δεδομένων είναι η εξέταση των κύριων δομικών μονάδων που απαιτούνται για την εκτέλεση των ζητούμενων βημάτων του αλγορίθμου Blastp. Σύμφωνα με τις απαιτήσεις των βημάτων που εκτελούνται στη βαθμίδα αυτή



Σχήμα 3.1: Γενική άποψη των δύο βαθμίδων

χρησιμοποιούνται μνήμες δεδομένων και τροποποιημένοι καταχωρητές. Στο σχήμα 3.2 φαίνεται η πρώτη βαθμίδα.

Η περιγραφή της εκάστοτε δομικής μονάδας της διαδρομής δεδομένων στη

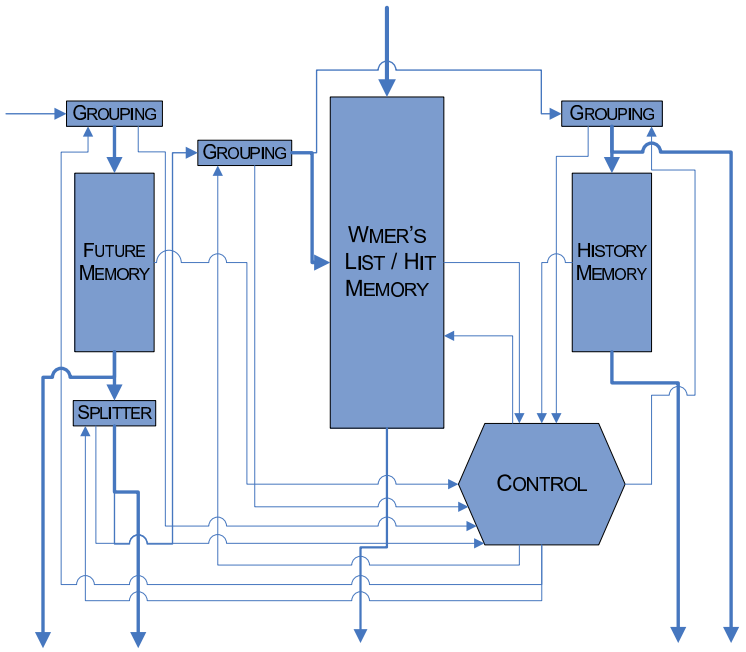
Αμινοξύ		Δυαδική Τιμή
Alanine	A	00000
Arginine	R	00001
Asparagine	N	00010
Aspartic Acid	D	00011
Cysteine	C	00100
Glutamic Acid	E	00110
Glutamine	Q	00101
Glycine	G	00111
Histidine	H	01000
Isoleucine	I	01001
Leucine	L	01010
Lysine	K	01011
Methionine	M	01100
Phenylalanine	F	01101
Proline	P	01110
Serine	S	01111
Threonine	T	10000
Tryptophan	W	10001
Tyrosine	Y	10010
Valine	V	10011

Table 3.1: Αντιστοίχιση χαρακτήρων αμινοξέων στη δυαδική μορφή

βαθμίδα, θα γίνει με βάση τις ενέργειες που είναι αναγκαίο να εκτελεστούν στο κάθε βήμα του αλγορίθμου Blastp. Ο στόχος της πρώτης βαθμίδας είναι να εξετάσει εάν υπάρχει ταίριασμα μεταξύ μίας λέξης που προκύπτει από μία ακολουθία εισόδου, που προέρχεται από τη βάση δεδομένων με μία λέξη, που έχει σχηματιστεί από την υπό εξέταση ακολουθία.

3.3.1 Δεδομένα της υπό εξέταση ακολουθίας

Η απαιτούμενη λίστα λέξεων της υπό εξέταση ακολουθίας που θα συγκριθεί με την αντίστοιχη της βάσης δεδομένων έχει κατασκευαστεί με τη βοήθεια μίας single port μνήμης. Η μνήμη ονομάζεται hitmemory και ως χαρακτηριστικά έχει πλάτος 19 bits και βάθος 32768 θέσεις. Το βάθος προκύπτει ως αποτέλεσμα του 2^{15}, που συμβολίζει το σύνολο των διευθύνσεων που προκύπτουν από την χρήση 15 bits ως πλάτος λέξης. Η επιλογή του πλάτους της μνήμης έχει το σκεπτικό ότι η πρώτη θέση (1 bit) δηλώνει άμα σε μία x διεύθυνση, έχει γραφτεί κάποια λέξη της υπό εξέταση ακολουθίας. Επομένως άμα



Σχήμα 3.2: Η διαδρομή δεδομένων για την πρώτη βαθμίδα της αρχιτεκτονικής

η πρώτη θέση περιέχει valid bit τότε η διεύθυνση έχει μία λέξη, διαφορετικά δεν περιέχει και τα δεδομένα της μνήμης στη συγκεκριμένη διεύθυνση δεν λαμβάνονται υπόψη. Οι επόμενες 17 θέσεις αναπαριστούν τον αριθμό λέξης, που προκύπτει σε σχέση με τη θέση της λέξης μέσα στην εξεταζόμενη ακολουθία. Το τελευταίο bit χρησιμεύει ως ένδειξη πολλαπλότητας, δηλαδή άμα μία λέξη έχει δημιουργηθεί παραπάνω από μία φορά. Σχηματικά η παραπάνω περιγραφή παρουσιάζεται στο σχήμα 3.2

Δεδομένα μίας γραμμής της hitmemory		
– 1 bit –	——— 19 bits ———	– 1 bit –
Valid 'V'	Θέση wmer στην ακολουθία	πολλαπλότητα εμφάνισης

Table 3.2: Αναπαράσταση μίας γραμμής της μνήμης hitmemory

3.3.2 Δεδομένα των ακολουθιών της βάσης δεδομένων

Σύμφωνα με τον αλγόριθμο πρέπει να δημιουργηθεί η λίστα λέξεων 3-γραμμάτων για τις ακολουθίες της βάσης δεδομένων. Κάθε γράμμα αντιστοιχίζεται με 5 bits, επομένως κάθε λέξη αποτελείται από 15 bits. Κάθε γράμμα από το σύνολο

του βιολογικού αλφάβητου δίνεται ως είσοδος με σκοπό τη δημιουργία μίας νέας λέξης κάθε φορά. Για τον σκοπό αυτό χρησιμοποιείται ένας παραμετρικός ως προς το μέγεθος καταχωρητής ολίσθησης - *shiftregister*. Ο καταχωρητής αυτός δέχεται είσοδο των 5 bits και παράγει έξοδο των 15 bits, μία λέξη δηλαδή αποτελούμενη από 3 γράμματα. Επειδή η ροή δεδομένων στην είσοδο μπορεί να μην είναι συνεχής αλλά διακοπτόμενη, δημιουργείται η ανάγκη να γίνεται παύση στη λειτουργία του καταχωρητή ολίσθησης (αυτό επηρεάζεται από την τιμή του σήματος *dbvaliddata*). Η παύση αυτή ισοδυναμεί με συνολική παύση της βαθμίδας, επειδή η διακοπή δημιουργίας νέων λέξεων προκαλεί την αδρανοποίηση των υπόλοιπων δομικών μονάδων της βαθμίδας. Για το σκοπό αυτό ο καταχωρητής υποστηρίζει ένα σήμα *enable*, το οποίο ενεργοποιεί τη λειτουργία του, ανάλογα με το αν τα δεδομένα στην είσοδο είναι έγκυρα ή όχι. Επομένως υπάρχει η δυνατότητα να "παγώνει" όλη η διαδρομή δεδομένων της πρώτης βαθμίδας, γεγονός αναγκαίο για την συνολική υλοποίηση του αλγορίθμου Blastp.

Επίσης ο καταχωρητής ολίσθησης, εκτός της κύριας εξόδου, η οποία είναι τα "πακεταρισμένα" δεδομένα με τη μορφή λέξεων, έχει άλλες δύο εξόδους πολύ σημαντικές. Η πρώτη αναφέρεται στο σήμα *regflag*, που αποτελεί την ένδειξη ότι μία νέα λέξη δημιουργήθηκε. Το εν λόγω σήμα ελέγχεται από έναν τοπικό μετρητή, που το ενεργοποιεί ανάλογα με τον αριθμό ολισθήσεων που πραγματοποιούνται κάθε φορά (*assertion on every 3 shifts*). Τέλος το σήμα αυτό οδηγείται ως είσοδος στη μονάδα ελέγχου της βαθμίδας. Η δεύτερη έξοδος, *datacount*, έχει βοηθητικό χαρακτήρα και χρησιμοποιείται ως είσοδος στη δεύτερη βαθμίδα. Η έξοδος αυτή θα αναλυθεί στην περιγραφή της δεύτερης βαθμίδας.

Από τη στιγμή που έχει δημιουργηθεί μία νέα λέξη, θα πρέπει να προστεθεί στην υπάρχουσα λίστα λέξεων. Για το λόγο αυτό χρησιμοποιείται μία *dual port block* μνήμη όπου εγγράφεται κάθε λέξη που δημιουργείται από τον αρχικό καταχωρητή ολίσθησης. Η χρησιμοποιούμενη μνήμη, με ονομασία *futurememory*, έχει πλάτος 15 bits και βάθος 1024 θέσεις. Η μνήμη επιλέχθηκε να είναι *dual port*, καθώς χρειάζεται να γράφεται και να διαβάζεται η ίδια θέση μνήμης όποτε είναι απαραίτητο. Η πρώτη πόρτα εισόδου της μνήμης χρησιμοποιείται για τα δεδομένα εγγραφής, ενώ η δεύτερη πόρτα για τα δεδομένα ανάγνωσης.

Στη συνέχεια θα πρέπει να γίνει έλεγχος μεταξύ των λέξεων που υπάρχουν στη λίστα λέξεων που αφορά τις ακολουθίες της βάσης δεδομένων και των λέξεων που περιέχονται στη λίστα λέξεων (hitmemory)που έχει προκύψει από την υπό εξέταση ακολουθία. Επειδή ο έλεγχος θα πρέπει να γίνει για όλες τις πιθανές λέξεις της υπό εξέταση ακολουθίας, θα πρέπει να επεξεργαστούν οι λέξεις που διαβάζονται από την μνήμη futurememory πριν οδηγηθούν προς εξέταση ταιριάσματος (matching - hit). Δηλαδή μεταξύ της μνήμης futurememory και της μνήμης hitmemory παρεμβάλλονται δύο δομικές μο-

νάδες. Η πρώτη μονάδα έχει ως λειτουργία το "σπάσιμο" των λέξεων που διαβάζονται από τη μνήμη futurememory. Ο *splitregister* λαμβάνει κάθε λέξη και την σπάει στους τρεις χαρακτήρες της. Στη συνέχεια κάθε χαρακτήρας μπαίνει ως είσοδος σε έναν δεύτερο καταχωρητή ολίσθησης, ο οποίος ονομάζεται *mergeregister*, όπου ανακατασκευάζεται κάθε "σπασμένη" λέξη. Η νέα δημιουργημένη λέξη χρησιμοποιείται ως διεύθυνση για την μνήμη hitmemory. Με τον τρόπο αυτό εξετάζεται εάν υπάρχει ταίριασμα για κάθε λέξη της εξεταζόμενης ακολουθίας. Εάν υπάρξει ταίριασμα διαβάζεται το πρώτο bit της θέσης στη διεύθυνση όπου έγινε και εμφανίζεται στην έξοδο της μνήμης. Η έξοδος αυτή ονομάζεται *ramout* και χρησιμοποιείται ως είσοδος στη δεύτερη βαθμίδα. Όμως τα 5 most significant bit της εξόδου του mergeregister χρησιμοποιούνται ως είσοδος σε έναν τρίτο καταχωρητή ολίσθησης, τον *reconstructregister*. Η λειτουργία του είναι όμοια με αυτή που περιγράφθηκε για τον shiftregister, και απλά δημιουργεί εκ νέου τις λέξεις που διαβάστηκαν από την futurememory και συγκρίθηκαν στην hitmemory ώστε να γραφτούν σε μία τρίτη μνήμη, την *historymemory*. Όμοια με τον shiftregister, ο reconstructregister έχει ως έξοδο, το σήμα *reconstructed*, το οποίο δηλώνει εάν έχει ανακατασκευαστεί ή όχι μία λέξη. Τέλος η historymemory είναι όπως και η futurememory μία dual port block memory, βάθους 1024 θέσεων και πλάτους 15 bits, που η πρώτη πόρτα εισόδου χρησιμοποιείται για την εγγραφή δεδομένων και η δεύτερη για την ανάγνωση δεδομένων.

Ο λόγος χρήσης δύο πανομοιότυπων μνημών οφείλεται στο γεγονός ότι υπάρχει η ανάγκη να διατηρείται διαθέσιμος ένας ικανοποιητικός αριθμός λέξεων της λίστας λέξεων της βάσης δεδομένων. Πιο αναλυτικά, σε ένα πιθανό ταίριασμα θα πρέπει να γίνει, σύμφωνα με τον αλγόριθμο, επέκταση στην εξεταζόμενη ακολουθία (query) με σκοπό να εξεταστούν και οι γειτονικοί χαρακτήρες. Επομένως θα πρέπει να υπάρχουν διαθέσιμες όλες οι λέξεις που σχηματίζουν την ακολουθία ή τις ακολουθίες της βάσης δεδομένων. Η χρήση των μνημών futuremory και historymemory εξασφαλίζει τη διαθεσιμότητα των λέξεων που χρειάζονται για εξέταση ταιριάσματος κατά τη διαδικασία της επέκτασης (βαθμίδα 2). Ο λόγος χρησιμοποίησης των δύο καταχωρητών ολίσθησης, splitregister και mergeregister είναι ο τρόπος που γίνεται το matching. Το matching γίνεται εξετάζοντας τρεις χαρακτήρες τις υπό εξέταση ακολουθίας με τρεις χαρακτήρες της βάσης δεδομένων. Δύο γειτονικά wmers της εξεταζόμενης ακολουθίας έχουν διαφορά ένα χαρακτήρα. Οι λέξεις που δημιουργούνται από τον αρχικό καταχωρητή (shiftregister) ολίσθησης έχουν διαφορά τρεις χαρακτήρες. Αυτό οδηγεί στην ανάγκη να υπάρχουν διαθέσιμες και οι ενδιάμεσες λέξεις της βάσης, τα οποία αποτελούν τα απαιτούμενα wmers της βάσης, με τα οποία γίνεται η αναζήτηση ταυτοποίησης με τα αντίστοιχα της εξεταζόμενης ακολουθίας. Συνεπώς η χρήση των δύο καταχωρητών επιτρέπει στο "κόψιμο" και "ράψιμο" των λέξεων που διαβάζονται από την future

memory. Επίσης οι παραπάνω καταχωρητές διασφαλίζουν ότι θα υπάρχει η δυνατότητα σε πιθανό matching να είναι διαθέσιμοι οι πιο κοντινοί χαρακτήρες που ενδεχομένως να μην περιέχονται σε κάποια λέξη.

3.3.3 Περιγραφή της μονάδας ελέγχου πρώτης βαθμίδας

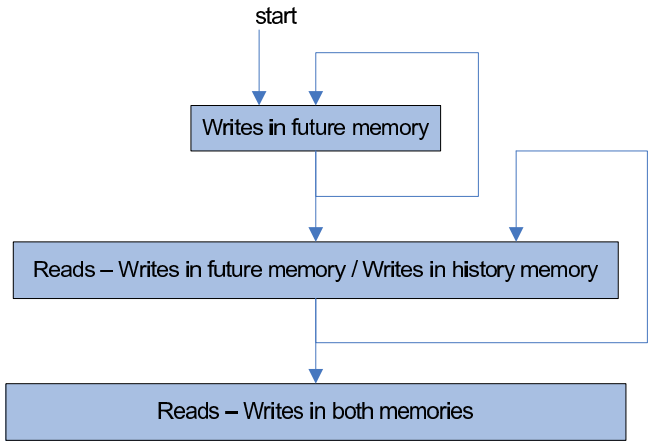
Για την σύνδεση όλων των παραπάνω δομικών μονάδων και για την επιθυμητή λειτουργία της πρώτης βαθμίδας απαιτείται μία μονάδα ελέγχου. Η κατασκευή της μονάδας ελέγχου αποτελεί και τη μεγαλύτερη πρόκληση, καθώς πρέπει να περιγραφεί με τέτοιον τρόπο, ώστε να μπορεί να χειριστεί και την πιο ακραία περίπτωση λειτουργίας της διαδρομής δεδομένων της βαθμίδας. Για την κατασκευή μίας μονάδας ελέγχου, συνήθως χρησιμοποιούνται μηχανές πεπερασμένων καταστάσεων, *finite state machines*. Μία δεύτερη τεχνική, η οποία ονομάζεται μικροπρογραμματισμός, *microprogramming*, χρησιμοποιεί μία αναπαράσταση προγραμματισμού για τη μονάδα ελέγχου.

Στη σχεδίαση της μονάδας ελέγχου της πρώτης βαθμίδας χρησιμοποιήθηκε η πρώτη μέθοδος, αυτή της πεπερασμένης μηχανής κατάστασης (Finite State Machine - FSM). Μία μηχανή πεπερασμένης κατάστασης αποτελείται από ένα σύνολο καταστάσεων και οδηγιών για τον τρόπο (εν)αλλαγής καταστάσεων. Οι οδηγίες ορίζονται από μία συνάρτηση επόμενης κατάστασης, η οποία απεικονίζει την τρέχουσα κατάσταση και τις εισόδους σε μία νέα κατάσταση. Όταν χρησιμοποιείται μία μηχανή πεπερασμένης κατάστασης για τη μονάδα ελέγχου, κάθε κατάσταση επίσης καθορίζει ένα σύνολο εξόδων που ενεργοποιούνται όταν η μηχανή είναι σε αυτή την κατάσταση. Η υλοποίηση μίας τέτοιας μηχανής συνήθως υποθέτει ότι όλες οι έξοδοι, που δεν είναι ρητά ενεργοποιημένες, είναι απενεργοποιημένες.

Με δεδομένη τη διαδρομή δεδομένων του σχήματος 3.2 πρέπει τώρα να παρουσιαστεί τι συμβαίνει κατά τη διάρκεια της εκτέλεσης, εφόσον αυτό θα καθορίσει τι επιπλέον σήματα ελέγχου θα χρειαστούν, όπως και την ανάθεση των σημάτων ελέγχου. Η μονάδα ελέγχου πεπερασμένης κατάστασης αντιστοιχεί στα βήματα εκτέλεσης. Μία συνολική εικόνα της μονάδας έλεγχου για όλα τα βήματα εκτέλεσης παρουσιάζεται στο σχήμα 3.3

Η μηχανή πεπερασμένης κατάστασης αποτελείται από αρκετά τμήματα. Η ομαλή εκτέλεση των βημάτων μπορεί να διακοπεί μόνο από τον έξω κόσμο (dbvalidsignal, pausestage1), οπότε μεταβαίνει στην ανάλογη κατάσταση αντιμετώπισης. Παρακάτω γίνεται μία παρουσίαση των πιθανών βημάτων εκτέλεσης καθώς και οι ενέργειες τους.

1. **Βήμα προετοιμασίας**. Υποδηλώνει ότι έχει ενεργοποιηθεί η είσοδος *dbstart* και ότι τα δεδομένα που θα ακολουθήσουν θα αφορούν τη βάση δεδομένων. Στο βήμα αυτό δεν ενεργοποιείται κανένα σήμα ελέγχου,



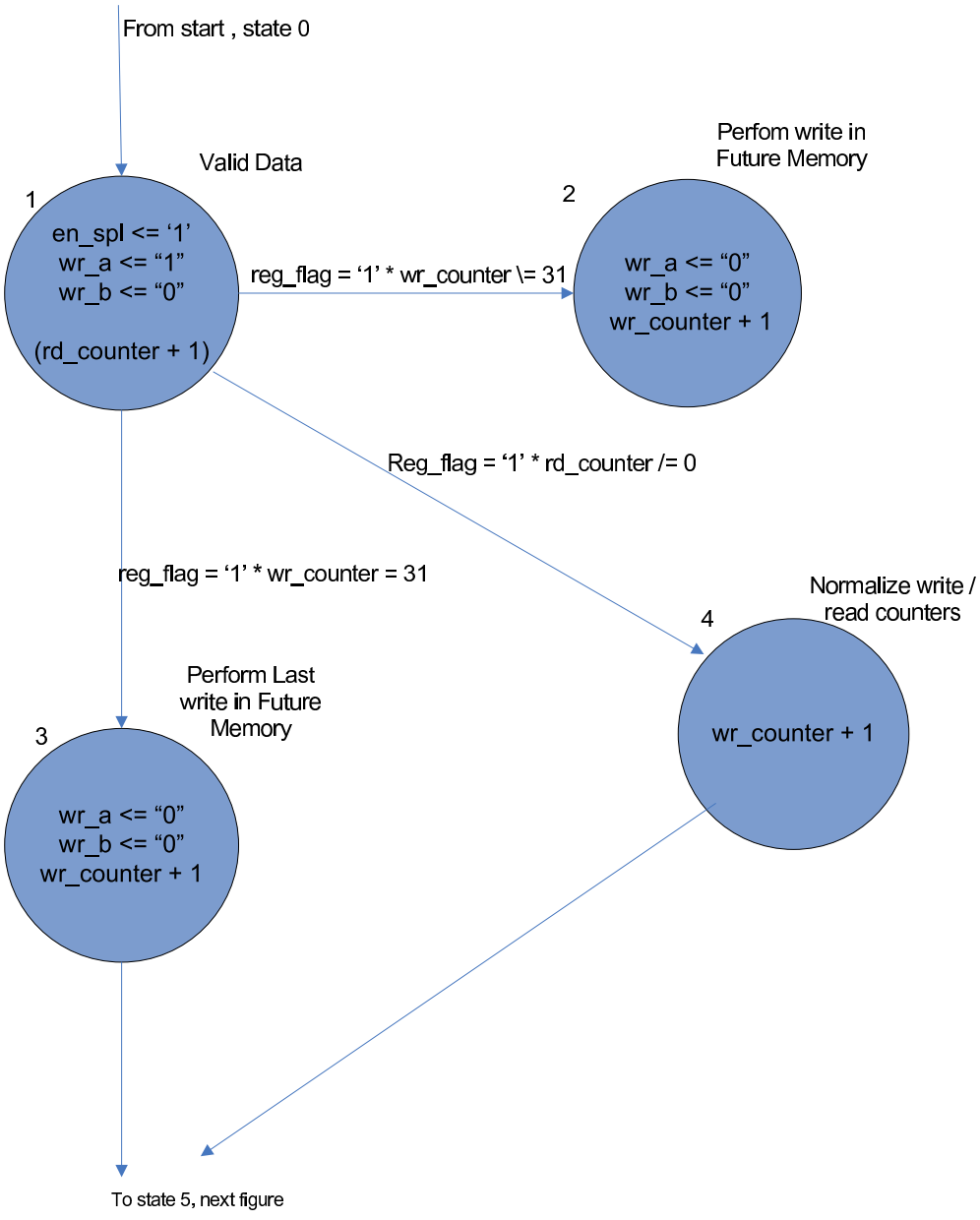
Σχήμα 3.3: Συνολική απεικόνιση της μονάδας ελέγχου για την πρώτη βαθμίδα

είναι ένα προπαρασκευαστικό βήμα.

2. **Βήμα έναρξης δεδομένων**. Υποδηλώνει ότι έχει ενεργοποιηθεί η είσοδος *dbvalidadata* και ότι τα δεδομένα που λαμβάνονται ως είσοδος είναι έγκυρα. Για την υλοποίηση του βήματος χρειάζεται να ενεργοποιηθούν το σήμα *wrenable* του πρώτου καταχωρητή ολίσθησης.

3. **Βήμα εγγραφής δεδομένων στην πρώτη μνήμη**. Στο βήμα αυτό εξετάζεται αρχικά, το σήμα *regflag* του αρχικού καταχωρητή ολίσθησης. Εφόσον είναι ενεργοποιημένο, ενεργοποιείται το σήμα εγγραφής στη μνήμη *wra* ενώ αυξάνουμε και την τιμή του δείκτη εγγραφής της μνήμης *wraddr*. Τα πρώτα τρία βήματα φαίνονται συνολικά στο σχήμα 3.4

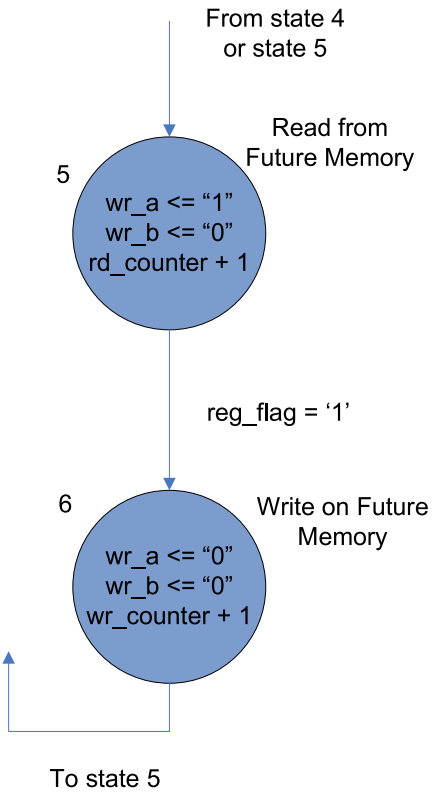
4. **Βήμα ανάγνωσης και εγγραφής δεδομένων για την πρώτη μνήμη**. Εάν έχουν πραγματοποιηθεί εγγραφές σε όλες τις θέσεις της futurememory ($wra = 1023$), για την εγγραφή μίας επόμενης νέας λέξης θα πρέπει αρχικά να διαβαστεί η λέξη που είχε γραφτεί πρώτα και στη συνέχεια να γίνει η εγγραφή της νέας λέξης στην τρέχουσα θέση μνήμης. Στο βήμα αυτό ενεργοποιούνται τα σήματα ανάγνωσης της μνήμης *wrb* (ουσιαστικά το σήμα εγγραφής της δεύτερης πόρτας της μνήμης διατηρείται σταθερά απενεργοποιημένο) και στην συνέχεια ενεργοποιείται το σήμα εγγραφής της πρώτης πόρτας *wra*. Επίσης ακολουθεί η ανάλογη αύξηση των δεικτών εγγραφής και ανάγνωσης της μνήμης *wraddr* και *rdaddr*. Να σημειωθεί ότι ο δείκτης ανάγνωσης διατηρείται σε δύο σήματα. Το πρώτο σήμα χρησιμοποιείται όπως περιγράφθηκε προηγουμένως, ενώ το δεύτερο θα χρησιμοποιηθεί στη δεύτερη βαθμίδα. Οι ενέργειες του τρέ-



Σχήμα 3.4: Εγγραφές στην πρώτη μνήμη δεδομένων

χοντος βήματος παρουσιάζονται στο σχήμα 3.5

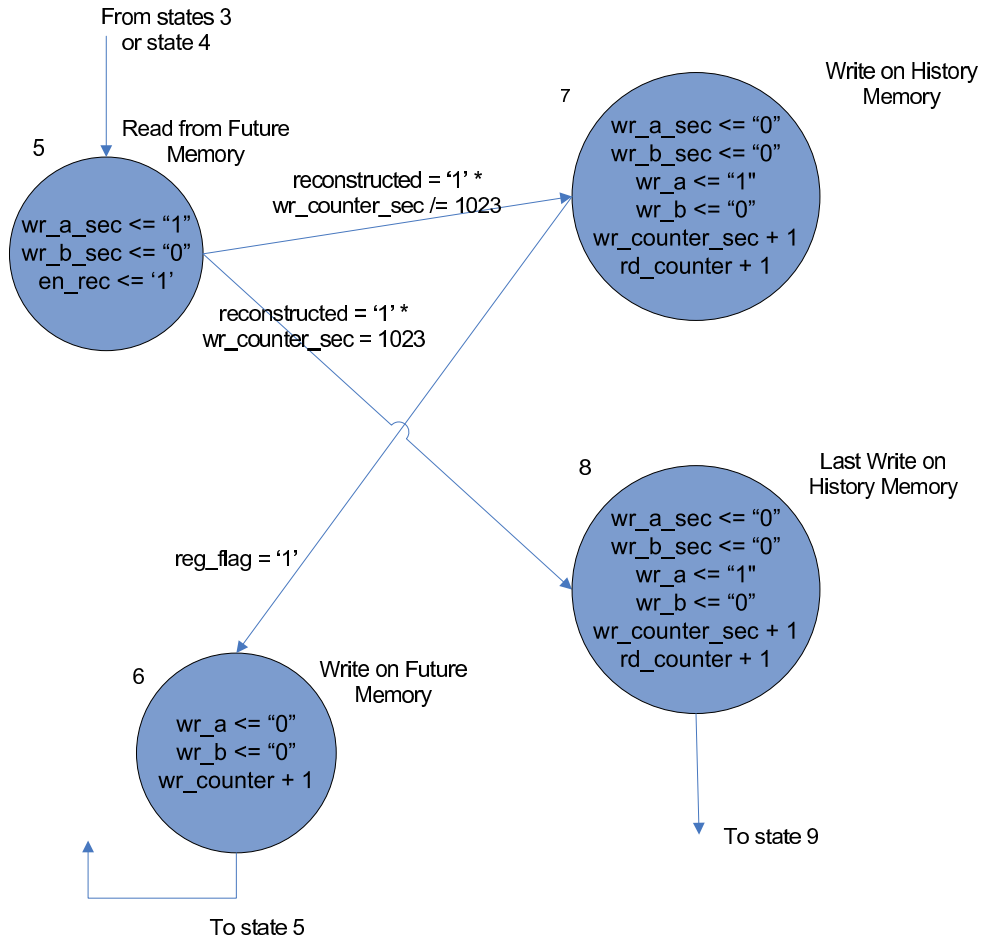
5. **Βήμα εγγραφής δεδομένων στη δεύτερη μνήμη**. Από τη στιγμή που διαβαστεί μία λέξη από την πρώτη μνήμη χρειάζεται 3 κύκλους ρολογιού για να φτάσει στη δεύτερη μνήμη - historymemory. Αυτό γίνεται αντι-



Σχήμα 3.5: Ανάγνωση - Εγγραφή στην πρώτη μνήμη δεδομένων

ληπτό από τη μονάδα ελέγχου, μέσω του σήματος *reconstructed*, που - ενεργοποιημένο - δηλώνει ότι η λέξη έχει ανακατασκευαστεί και είναι έτοιμη να εγγραφεί στην δεύτερη μνήμη. Επομένως ενεργοποιείται το σήμα εγγραφής για την πρώτη πόρτα της δεύτερης μνήμης, *wrasec*, ενώ αυξάνεται και η τιμή του δείκτη εγγραφής που αφορά στη δεύτερη μνήμη. Παράλληλα η μονάδα ελέγχου εξετάζει εάν κατά τη διάρκεια εγγραφής στη historymemory, ενεργοποιείται το σήμα *regflag* του shiftregister. Στην περίπτωση που συμβαίνει αυτό, αφού ολοκληρωθεί το τρέχων βήμα, η εκτέλεση μεταβαίνει στο προηγούμενο βήμα, ανάγνωσης και εγγραφής δεδομένων στην futurememory. Σε διαφορετική περίπτωση περιμένει την ενεργοποίηση του σήματος *regflag*. Πρακτικά αυτό δε συμβαίνει ποτέ, επειδή η ενεργοποίηση των σημάτων regflag και reconstructed συμβαίνει με διαφορά ενός κύκλου. Στο σχήμα 3.6 παρουσιάζονται όσα περιγράφηκαν.

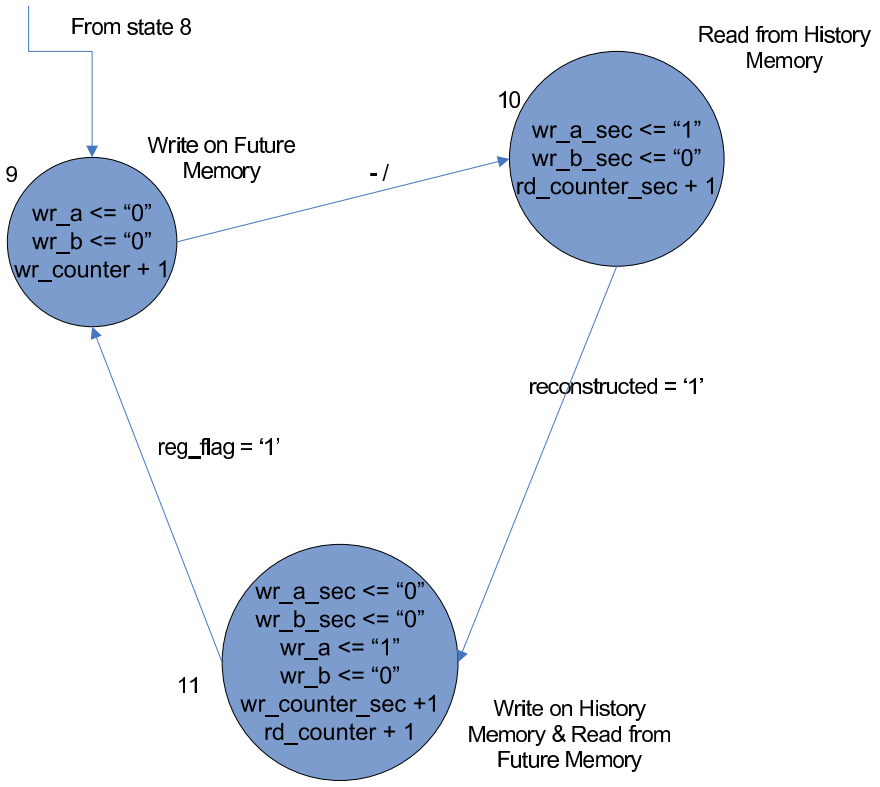
6. **Βήμα ανάγνωσης και εγγραφής δεδομένων για τη δεύτερη μνήμη.** Στο βήμα αυτό μεταβαίνει η μονάδα ελέγχου όταν έχει γίνει εγγραφή σε κάθε



Σχήμα 3.6: Ανάγνωση - Εγγραφή στην πρώτη μνήμη δεδομένων και εγγραφές στη δεύτερη μνήμη δεδομένων

θέση της historymemory, ($wrasec = 1023$). Αυτό έχει ως αποτέλεσμα να πρέπει να διαβαστεί πρώτα η λέξη που είναι ήδη γραμμένη στην τρέχουσα διεύθυνση και ύστερα να εγγραφεί η νέα λέξη. Σε πλήρη αντιστοιχία με την futurememory, το σήμα εγγραφής για τη δεύτερη πόρτα *wrbsec* παραμένει απενεργοποιημένο, ενώ ακολούθως ενεργοποιείται το σήμα εγγραφής για την πρώτη πόρτα, *wrasec*. Οι τιμές των δεικτών ανάγνωσης και εγγραφής αυξάνονται κατά ένα ώστε δείχνουν στις κατάλληλες θέσεις μνήμης. Τέλος χρησιμοποιείται ένα ακόμη σήμα για να διατηρείται η διεύθυνση ανάγνωσης για τη δεύτερη μνήμη, που είναι αναγκαίο για τη λειτουργία της δεύτερης βαθμίδας. Αφού ολοκληρωθεί το βήμα αυτό η εκτέλεση μεταβαίνει στο βήμα 4. Στο σχήμα 3.7 παρουσιάζονται τα

βήματα ανάγνωσης - εγγραφής για τις δύο μνήμες δεδομένων.



Σχήμα 3.7: Εγγραφή - Ανάγνωση και στις δύο μνήμες δεδομένων

7. **Βήμα αντιμετώπισης μη διαρκούς ροής έγκυρων δεδομένων εισόδου**. Όπως γίνεται κατανοητό από την ονομασία του, είναι ένα πολύ σημαντικό βήμα, που αφορά στον τρόπο εκτέλεσης της βαθμίδας, εάν υπάρξει διακοπή στην ακολουθία δεδομένων στην είσοδο του shiftregister ($dbvalidadata = '0'$). Διακρίνονται δύο περιπτώσεις. Η πρώτη περίπτωση αφορά όταν πραγματοποιούνται μόνο εγγραφές στην future-memory, όπου ο χειρισμός της είναι απλό πάγωμα των εγγραφών. Η δεύτερη περικλείει όλα τα υπόλοιπα πιθανά ενδεχόμενα. Ανάλογα με τις τιμές των σημάτων regflag και reconstructed, πριν διακοπεί η έγκυρη ροή δεδομένων εισόδου, εκτελούνται εγγραφές ή/και αναγνώσεις λέξεων εκτός σειράς. Με τον όρο εκτός σειράς, υποδηλώνεται ότι η εγγραφή / ανάγνωση προβλεπόταν να γίνει αδιάφορα άμα το σήμα dbvalidata απενεργοποιήθηκε. Συνεπώς αφού ολοκληρωθούν οι απαραίτητες ενέργειες (π.χ. ρύθμιση των τιμών δεικτών διεύθυνσης εγγραφής και ανάγνωσης

για τις δύο μνήμες), το σύστημα παραμένει ανενεργό μέχρι τα δεδομένα εισόδου να είναι και πάλι έγκυρα.

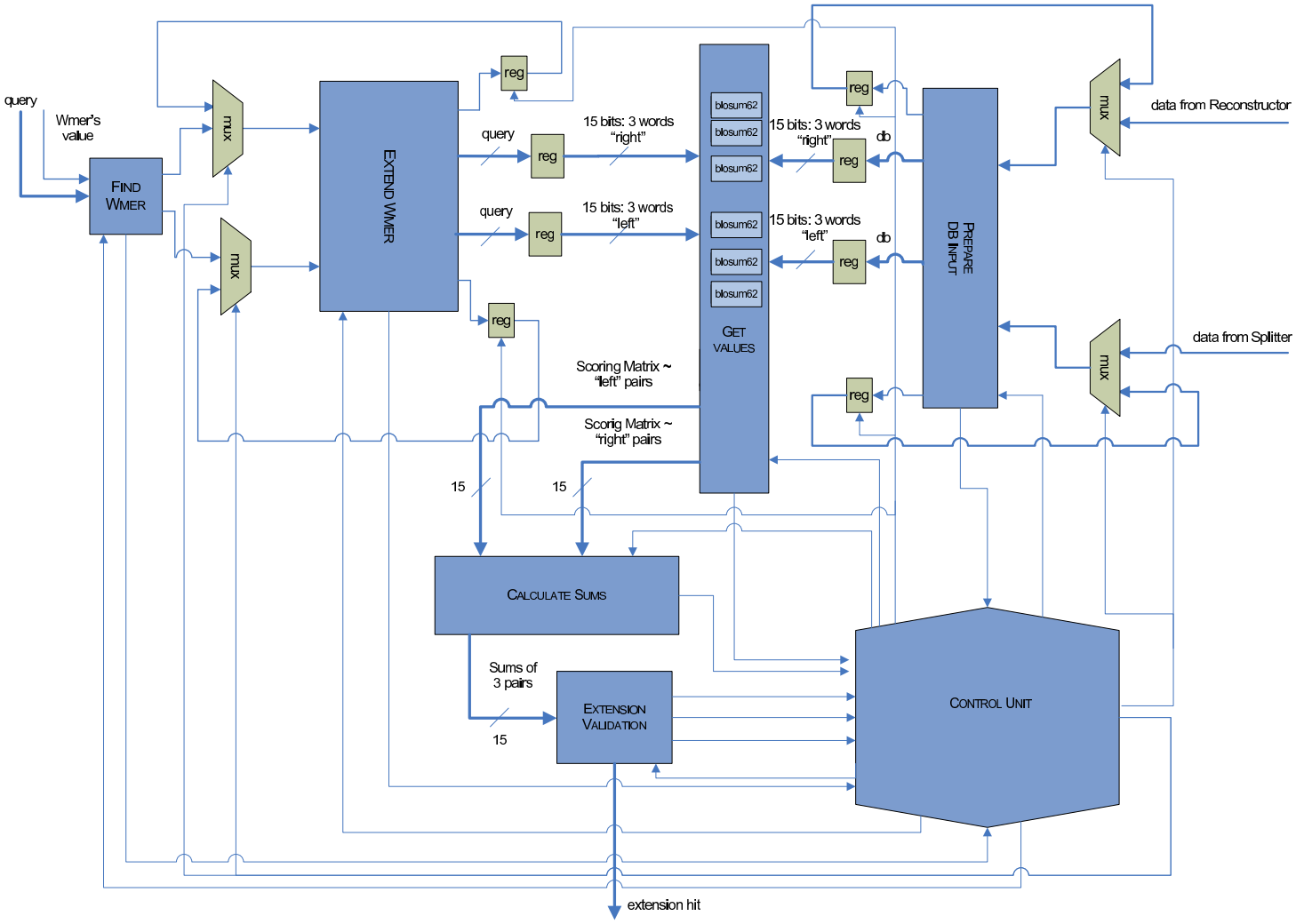
Τα παραπάνω βήματα δίνουν μία συνολική περιγραφή της απαιτούμενης λογικής που χρησιμοποιήθηκε για την κατασκευή της μονάδας ελέγχου. Στην περίπτωση φόρτωσης - αρχικοποίησης της μνήμης hitmemory, η μόνη διαφορά με τα παραπάνω βήματα είναι ότι ενεργοποιείται το σήμα εγγραφής της μνήμης, *wr_ram*.

3.4 Περιγραφή Δεύτερης Βαθμίδας

Κατά τη διάρκεια της αναζήτησης ταιριάσματος μεταξύ της υπό εξέταση ακολουθίας και της βάσης δεδομένων, η hitmemory εμφανίζει στην έξοδο της το πρώτο bit της θέσης της τρέχουσας διεύθυνσης. Αυτό το bit οδηγείται ως είσοδος *hit* στη δεύτερη βαθμίδα, όπως και τα διπλανά 17 bits που δηλώνουν τη θέση της λέξης στην εξεταζόμενη ακολουθία. Η δεύτερη βαθμίδα έχει επίσης, ως εισόδους τα δεδομένα των καταχωρητών splitregister και reconstructregister, αλλά και το σήμα εξόδου του shiftregister *datacount*, όπως είχε αναφερθεί και στην προηγούμενη ενότητα. Από τη μονάδα ελέγχου της πρώτης βαθμίδας οδηγούνται στη δεύτερη βαθμίδα τα δύο διπλά σήματα, *rdpoint* και *rdpointsec*,που κρατούν την τιμή θέσης ανάγνωσης για τις μνήμες futurememory και historymemory. Τέλος η υπό εξέταση ακολουθία αποτελεί μία ακόμα είσοδό της. Επομένως, στην περίπτωση που υπάρξει *matching*, γίνεται παύση της λειτουργίας της πρώτης βαθμίδας και έναρξη της δεύτερης. Η δεύτερη βαθμίδα, που σχηματικά φαίνεται στο σχήμα 3.8, αποτελείται από κάποιες δομικές μονάδες με συγκεκριμένο ρόλο η κάθε μία.

3.4.1 Εύρεση τρέχοντος wmer και επέκτασή του

Το πρώτο πρόβλημα αποτελεί ο προσδιορισμός της θέσης της λέξης που υπήρξε το matching μέσα στην εξεταζόμενη ακολουθία. Για τον σκοπό αυτό χρησιμοποιείται η μονάδα *findwmer*, που λαμβάνει ως εισόδους την εξεταζόμενη ακολουθία και τα 17 bits, που δηλώνουν τη θέση της λέξης που έγινε το hit. Η εύρεση της είναι αναγκαία, επειδή θα πρέπει να είναι διαθέσιμες όλες οι γειτονικές της λέξεις για την διαδικασία της επέκτασης. Εφόσον η θέση βρεθεί, το most significant bit, δηλαδή η έναρξη του wmer, και το least significant bit, δηλαδή η λήξη του wmer, οδηγούνται σε δύο καταχωρητές και ακολούθως σε δύο πολυπλέκτες 2 προς 1 πριν οδηγηθούν στη μονάδα *extendwmer*. Η λειτουργία της μονάδας αυτής είναι η επέκταση του wmer και προς τις δύο κατευθύνσεις κατά τρεις χαρακτήρες. Επομένως στην έξοδο εμφανίζονται οι δύο γειτονικές του λέξεις, σε μορφή χαρακτήρων. Επίσης υπολογίζονται το νέο



Σχήμα 3.8: Συνολική απεικόνιση της μονάδας ελέγχου για τη δεύτερη βαθμίδα

bit έναρξης και λήξης της διευρυμένης υπακολουθίας. Οι δύο αυτές τιμές οδηγούνται αρχικά σε καταχωρητές και στη συνέχεια στους πολυπλέκτες 2 προς 1, που βρίσκονται πριν τη μονάδα extendwmer. Στην αρχική περίπτωση που υπήρξε hit στην πρώτη βαθμίδα επιλέγονται ως είσοδοι της extendwmer, οι έξοδοι της findwmer μονάδας, ενώ όταν συνεχίζεται η διαδικασία της επέκτασης επιλέγονται οι νέες υπολογισμένες τιμές έναρξης και λήξης της διευρυμένης υπακολουθίας.

3.4.2 Προετοιμασία δεδομένων της βάσης δεδομένων

Παράλληλα με τη λειτουργία της μονάδας extendwmer, πραγματοποιείται και η προετοιμασία των δεδομένων της βάσης δεδομένων που θα συγκριθούν με τις γειτονικές λέξεις του wmer στη μονάδα *preparedbinput*. Η μονάδα αυτή δημιουργεί τις αντίστοιχες γειτονικές λέξεις, χρησιμοποιώντας τις μνήμες της πρώτης βαθμίδας, που θα συγκριθούν με αυτές που υπολογίζονται στη μονάδα extendwmer. Οι είσοδοι της μονάδας είναι το σήμα *datacount*, τα περιεχόμενα των καταχωρητών splitregister και recontructregister και τα δεδομένα των μνημών futuremory και historymemory. Πιο αναλυτικά ανάλογα σε ποιο wmer παρατηρήθηκε το matching, θα πρέπει να αποκλειστούν κάποιοι χαρακτήρες από τα δεδομένα των δύο καταχωρητών, επειδή μπορεί να περιλαμβάνονται στο wmer αυτό. Αυτός ακριβώς είναι ο ρόλος του σήματος datacount. Οι τιμές που λαμβάνει είναι τρεις ('0', '1', '2') και υποδηλώνει πόσους χαρακτήρες από τα δεδομένα των καταχωρητών θα χρησιμοποιηθούν ως μέρος των εισόδων της preparedbiput. Το υπόλοιπο μέρος της εισόδων συμπληρώνεται από τα δεδομένα ή μέρος τους που βρίσκονται στις θέσεις των μνημών, που υποδηλώνονται από τις τιμές των δεικτών θέσης ανάγνωσης. Στην έξοδο της μονάδας οδηγούνται δύο νέες δημιουργημένες λέξεις προς σύγκριση καθώς και μέρος των δεδομένων από τις θέσεις των μνημών που διαβάστηκαν, τα οποία θα χρησιμοποιηθούν στην περίπτωση της συνέχισης της επέκτασης. Η επιλογή τους θα γίνει πάλι με χρήση πολυπλεκτών 2 προς 1, όπου στην αρχική περίπτωση του hit θα επιλέγονται τα δεδομένα των δύο καταχωρητών, ενώ στη συνέχεια τα δεδομένα των μνημών που δε χρησιμοποιήθηκαν για τη δημιουργία των λέξεων.

Παρακάτω παρουσιάζονται οι πίνακες 3.3 και 3.4 που συγκεντρώνουν τους πιθανούς συνδυασμούς δεδομένων καταχωρητών και δεδομένων μνημών για την δημιουργία των απαιτούμενων λέξεων που θα προωθηθούν για σύγκριση με τις αντίστοιχες λέξεις της εξεταζόμενης ακολουθίας. Επομένως ανάλογα με την τιμή του datacount προκύπτει:

Datacount	Width Bits	Data left	Temporart data left
"00"	14 - 5 4 - 0	data splitter(9-0) future memory(14-10)	future memory(9-0) "00000"
"01"	14 - 10 9 - 0	data splitter(14-10) future memory(14 - 5)	future memory(4-0) "0000000000"
"10"	14 - 0	datasplitter	future memory

Table 3.3: Δημιουργία "αριστερών" δεδομένων ανάλογα με την τιμή του σήματος *datacount*

Datacount	Width Bits	Data right	Temporary data right
"00"	14 - 10 9 - 0	data reconstructor(4-0) history memory(14-5)	history memory(4-0) "0000000000"
"01"	14 - 5 4 - 0	data reconstructor(14-5) history memory(14-10)	history memory(9-0) "00000"
"10"	14 - 0	data reconstructor	history memory

Table 3.4: Δημιουργία "δεξιών" δεδομένων ανάλογα με την τιμή του σήματος *datacount*

Θα πρέπει να παρατηρηθεί ότι όλες οι έξοδοι της μονάδας preparedbinput οδηγούνται σε καταχωρητές, σε πλήρη αντιστοιχία με τα δεδομένα εξόδου της μονάδας extendwmer.

3.4.3 Σύγκριση λέξεων σύμφωνα με τον BLOSUM62

Στο στάδιο αυτό έχουν υπολογιστεί οι γειτονικές λέξεις τόσο για το wmer, όπου παρατηρήθηκε matching, όσο και για τη αντίστοιχη λέξη της βάσης δεδομένων. Τα δεδομένα που προέρχονται από τη μονάδα extendwmer διακρίνονται στα σήματα: query_r0, query_r1, query_r2, που αφορούν τους τρεις χαρακτήρες που βρίσκονται δεξιά του τρέχοντος wmer, ενώ τα query_l0, query_l1, query_l2 αφορούν τους χαρακτήρες που βρίσκονται αριστερά του τρέχοντος wmer. Επιπλέον τα σήματα *db_data_left* και *db_data_right*, που προέρχονται από τη μονάδα preparedbinput αφορούν την αριστερή και δεξιά λέξη από την εξεταζόμενη λέξη της βάσης δεδομένων αντίστοιχα. Για τη σύγκριση των τεσσάρων λέξεων θα δημιουργηθούν έξι ζεύγη των δύο γραμμάτων. Κάθε ζεύγος αντιπροσωπεύει μία διεύθυνση, για τον πίνακα υποκατάστασης. Πιο αναλυτικά σε ό,τι αφορά τον πίνακα υποκατάστασης θα πρέπει να σημειωθεί ότι έχει επιλεχθεί ο πίνακας υποκατάστασης BLOSUM62. Με βάση την αντιστοίχιση των χαρακτήρων σε δυαδική μορφή και τον πίνακα 2.8 που παρουσιάστηκε στο προηγούμενο κεφάλαιο δημιουργούμε ένα αρχείο .coe με το οποίο αρχικοποιούμε μία μνήμη *single port*, η οποία έχει βάθος 1024 θέσεις και πλάτος 5 bits. Η μνήμη αυτή αντιπροσωπεύει τον πίνακα BLOSUM62.

Το κάθε ζεύγος χαρακτήρων σχηματίζει μία διεύθυνση (των 10 bits) με την οποία προσπελαύνεται η μνήμη *blosum62_matrix* και λαμβάνεται στην έξοδο της η τιμή που αντιστοιχεί στο εξεταζόμενο ζεύγος γραμμάτων. Εφόσον πρέπει να συγκριθούν έξι ζεύγη χαρακτήρων, χρησιμοποιούνται έξι στιγμιότυπα της μνήμης blosum62_matrix. Τα έξι στιγμιότυπα της μνήμης περιέχονται στη μονάδα *getvalues*. Στη μονάδα αυτή λαμβάνονται ως είσοδοι τα δεδομένα που προέρχονται από τις μονάδες extendwmer και preparedbinput και σχηματί-

ζονται τα απαιτούμενα ζεύγη. Ως έξοδοι προκύπτουν τα δεδομένα των έξι μνημών τα οποία οδηγούνται σε επόμενη μονάδα.

Θα πρέπει να αναφερθεί ότι λόγω του τρόπου με τον οποίο αποθηκεύονται οι χαρακτήρες στους καταχωρητές splitregister και reconstructregister, ο σχηματισμός των ζευγών πραγματοποιείται ο εξής: db_data_r0 *vs* query_r0, db_data_r1 *vs* query_r1, db_data_r2 *vs* query_r2, db_data_l0 *vs* query_l0, db_data_l1 *vs* query_l1 και db_data_l2 *vs* query_l2.

3.4.4 Υπολογισμός αθροισμάτων ζευγών

Η μονάδα *computesums* λαμβάνει ως εισόδους τις εξόδους των μνημών - πινάκων υποκατάστασης blosum62 και ως λειτουργία έχει να αθροίσει ανά δύο τα ζεύγη ως προς το εξεταζόμενο wmer. Οπότε προκύπτουν τα παρακάτω τρία αθροίσματα:

Θέσεις DB2-R0: $sum0 = (db\_data\_r0 vs query\_r0) + (db\_data\_l0 vs query\_l0)$
Θέσεις DB1-R1: $sum1 = (db\_data\_r1 vs query\_r1) + (db\_data\_l1 vs query\_l1)$
Θέσεις DB0-R2: $sum2 = (db\_data\_r2 vs query\_r2) + (db\_data\_l2 vs query\_l2)$

Έλεγχος για την εγκυρότητα της Επέκτασης

Ο ρόλος της μονάδας *extensionvalidation* είναι να ελέγξει κατά πόσο η τριπλή επέκταση (για τρεις χαρακτήρες) που πραγματοποιήθηκε είναι έγκυρη. Ο έλεγχος πραγματοποιείται εξετάζοντας το most significant bit των τριών αθροισμάτων που λαμβάνονται ως είσοδος στη μονάδα. Εάν το bit αυτό είναι '1' αντιπροσωπεύει το αρνητικό πρόσημο - αναπαράσταση σε two's complement[8] του αθροίσματος. Επομένως εκτελείται έλεγχος πρόσημου για τα τρία αθροίσματα, και διακρίνονται οι παρακάτω περιπτώσεις:

- Αρνητικό πρόσημο για sum0: Αυτό σημαίνει ότι κακώς πραγματοποιήθηκε η επέκταση και δεν λαμβάνεται υπόψη κανένα από τα τρία αθροίσματα.

- Θετικό πρόσημο για sum0 και αρνητικό πρόσημο για sum1: Εξάγεται το συμπέρασμα ότι μόνο η επέκταση κατά ένα χαρακτήρα εκατέρωθεν του εξεταζόμενου wmer είναι έγκυρη. Στο συνολικό τελικό score που αποτελεί έξοδο της δεύτερης βαθμίδας υπολογίζεται μόνο το άθροισμα sum0.

- Θετικά πρόσημα για sum0 και sum1 αλλά αρνητικό πρόσημο για sum2: Εξάγεται το συμπέρασμα ότι μόνο η επέκταση κατά δύο χαρακτήρες εκατέρωθεν του εξεταζόμενου wmer είναι έγκυρη, και στο συνολικό τελικό score υπολογίζονται τα αθροίσματα sum0 και sum1.

- Όλα τα πρόσημα αθροισμάτων θετικά: Στην περίπτωση αυτή εξετάζεται εάν τα αθροίσματα είναι μηδενικά ή όχι. Εάν προκύψουν όλα μηδενικά δεν πληρούνται οι προϋποθέσεις του αλγορίθμου συνεπώς η επέκταση δεν είναι έγκυρη και στο συνολικό τελικό score δεν λογίζεται κανένα άθροισμα. Αντίθετα εάν είναι μη μηδενικά η επέκταση είναι έγκυρη με αποτέλεσμα να συνεχιστεί η επέκταση για άλλους τρεις χαρακτήρες εκατέρωθεν του εξεταζόμενου wmer, ενώ στο συνολικό τελικό score υπολογίζονται και τα τρία επιμέρους αθροίσματα.

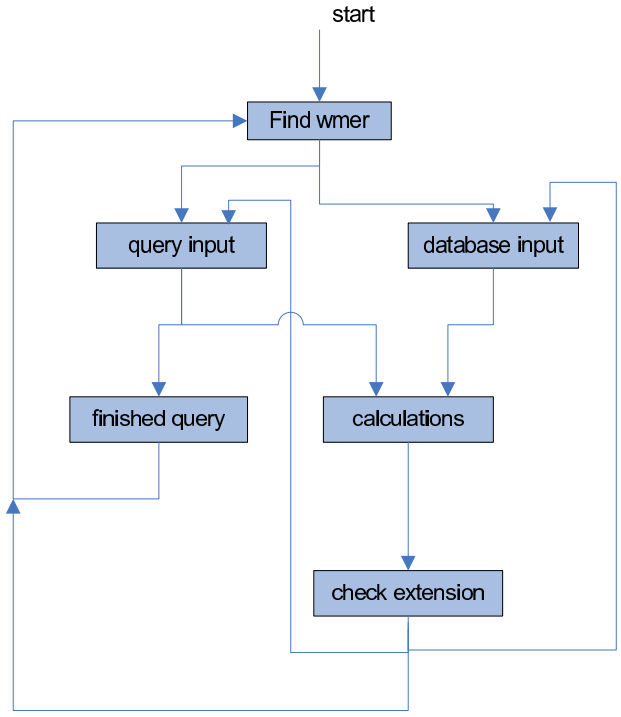
3.4.5 Περιγραφή της μονάδας ελέγχου δεύτερης βαθμίδας

Όπως και στην πρώτη βαθμίδα η μονάδα έλεγχου της δεύτερης υλοποιείται με χρήση μηχανής πεπερασμένης κατάστασης. Το σκεπτικό για την υλοποίηση της προκύπτει από τη διαδρομή δεδομένων της βαθμίδας που παρουσιάζεται στο σχήμα 3.8. Η λογική με την οποία είναι κατασκευασμένη η διαδρομή δεδομένων επιτρέπει την ξεχωριστή ενεργοποίηση κάθε δομικής μονάδας γεγονός που μειώνει την πολυπλοκότητα της μονάδας ελέγχου. Η γενική άποψη της μονάδας έλεγχου στο σχήμα 3.9 οπτικοποιεί το πώς προωθούνται τα δεδομένα μεταξύ των δομικών μονάδων κατά μήκος της διαδρομής δεδομένων.

Η μηχανή πεπερασμένης κατάστασης αποτελείται από αρκετά τμήματα. Η εκτέλεση των βημάτων του αλγορίθμου ενεργοποιείται όταν να υπάρξει matching στην πρώτη βαθμίδα. Όταν η έξοδος της hitmemory ενεργοποιηθεί, η μονάδα ελέγχου της δεύτερης βαθμίδας "παγώνει" την πρώτη και ενεργοποιεί τη διαδρομή δεδομένων της δεύτερης. Τα βήματα εκτέλεσης καθώς και οι ενέργειες που λαμβάνουν χώρα παρουσιάζονται παρακάτω.

1. **Απενεργοποίηση πρώτης βαθμίδας**. Στο βήμα αυτό ελέγχεται η έξοδος της μνήμης hitmemory και εάν είναι ενεργοποιημένη, ενεργοποιείται το σήμα *pause_stage1*. Επίσης ενεργοποιούνται τα σήματα *en_fwmer* και *en_regfwmer* που αφορούν στη μονάδα findwmer και στους καταχωρητές (υποδέχονται τις εξόδους της findwmer) αντίστοιχα.

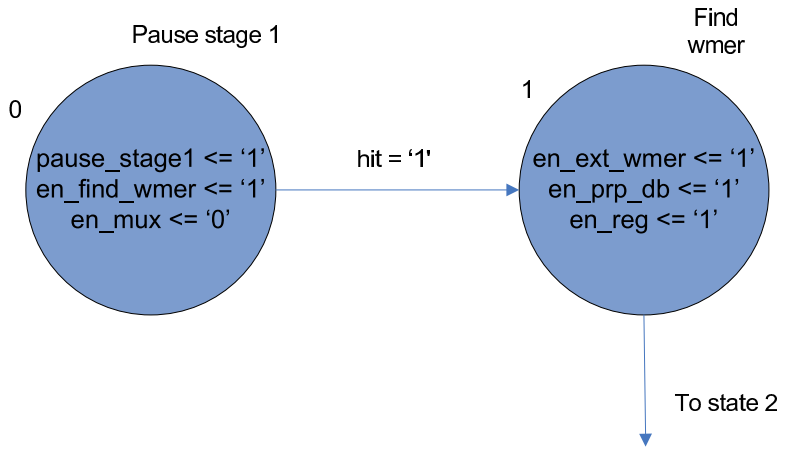
2. **Εύρεση της θέσης του εξεταζόμενου wmer**. Στο βήμα αυτό ελέγχεται μέσω του σήματος *found_wmer*, εάν έχει βρεθεί η θέση του εξεταζόμενου wmer στην υπό εξέταση ακολουθία. Εάν είναι ενεργοποιημένο το σήμα, ενεργοποιούνται τα σήματα *en_extwmer*, *en_dbprp*, *en_reg* και *en_regext*. Τα δύο πρώτα σήματα ενεργοποιούν τις μονάδες extendwmer και preparedbinput. Τα άλλα δύο σήματα ενεργοποιούν τους καταχωρητές που υποδέχονται τις εξόδους των τρεχόντων δομικών μονάδων. Επίσης τα σήματα en_fwmer και en_regfmwer απενεργοποιούνται, αφού η λειτουργία τους έχει ολοκληρωθεί. Εάν το σήμα *found_wmer* δεν



Σχήμα 3.9: Συνολική απεικόνιση της μονάδας ελέγχου για τη δεύτερη βαθμίδα

είναι ενεργοποιημένο, παραμένουμε στο βήμα αυτό έως να ενεργοποιηθεί. Σχηματικά τα δύο πρώτα βήματα εκτέλεσης παρουσιάζονται στο σχήμα 3.10

3. **Επέκταση των γειτονικών λέξεων τόσο για το wmer όσο και για τη λέξη της βάσης δεδομένων**. Στο βήμα αυτό ελέγχεται εάν έχει πραγματοποιηθεί η επέκταση των κατάλληλων δεδομένων στις μονάδες extendwmer και preparedbinput. Εφόσον τα σήματα πληροφόρησης - flags - ενεργοποιηθούν, ενεργοποιείται το σήμα *en_getval*, το οποίο αφορά στη μονάδα getvalues, στην οποία εκτελείται η σύγκριση μεταξύ των δημιουργημένων ζευγαριών με βάση τον πίνακα υποκατάστασης BLOSUM62. Τέλος τα σήματα που αφορούν τις μονάδες επέκτασης δεδομένων, καθώς και των καταχωρητών που υποδέχονται τις εξόδους των πρώτων απενεργοποιούνται. Θα πρέπει να αναφερθεί ότι κατά τη διάρκεια επέκτασης του wmer στη μονάδα extendwmer, ελέγχεται εάν έχουμε περίπτωση ακριανής επέκτασης. Δηλαδή το wmer είναι είτε στην αρχή είτε στο τέλος της εξεταζόμενης ακολουθίας ή η επέκταση βγαίνει εκτός ορίων της εξεταζόμενης ακολουθίας. Στην περίπτωση αυτή στα-



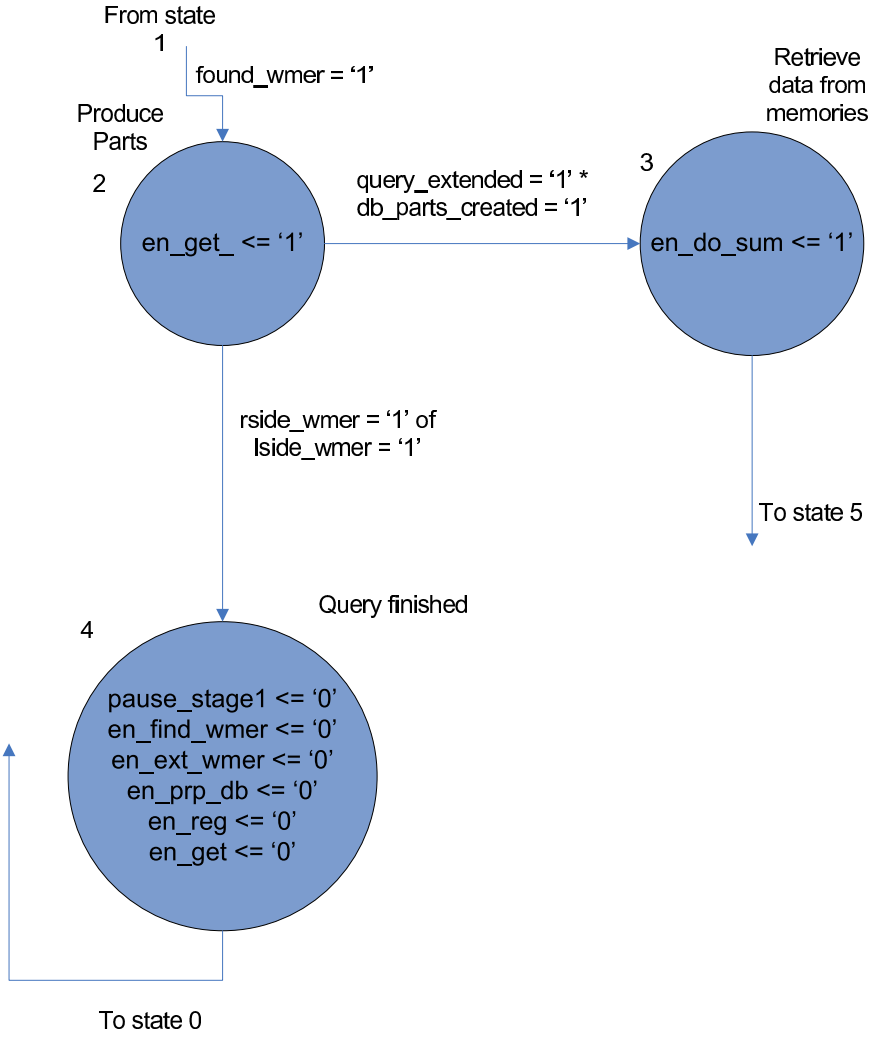
Σχήμα 3.10: Βήματα εύρεσης της θέσης του τρέχοντος wmer στην εξεταζόμενη ακολουθία

ματάει η λειτουργία της δεύτερης βαθμίδας και ενεργοποιείται εκ νέου η πρώτη βαθμίδα για την αναζήτηση νέου matching.

4. **Λήψη score ζευγών**. Στο βήμα αυτό λαμβάνονται τα score που αφορούν στα έξι σχηματισμένα ζεύγη χαρακτήρων. Εφόσον ληφθούν (ενεργοποιείται το flag της μονάδας) θα πρέπει να ενεργοποιηθεί το σήμα *en_calc* για τη μονάδα computesums, για να υπολογιστούν τα αθροίσματα (ανά δύο) ζευγών, ενώ παράλληλα να απενεργοποιηθεί η μονάδα getvalues. Σε περίπτωση που δεν έχουν ληφθεί τα απαιτούμενα score, η μονάδα getvalues παραμένει ενεργοποιημένη έως ότου ληφθούν. Τα παραπάνω δύο βήματα εμφανίζονται στο σχήμα 3.11

5. **Υπολογισμός αθροισμάτων**. Εάν υπολογιστούν τα ζητούμενα αθροίσματα, ενεργοποιείται η μονάδα extensionvalidation μέσω του σήματος *en_extval*. Η μονάδα computesums απενεργοποιείται.

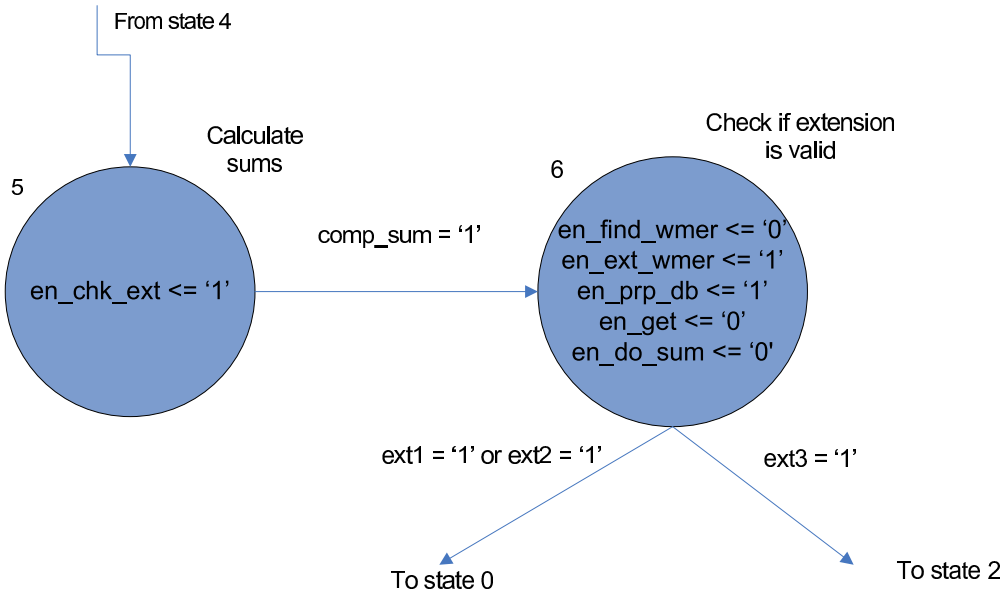
6. **Έλεγχος έγκυρης επέκτασης**. Το βήμα αυτό καθορίζει εάν θα συνεχιστεί η επέκταση του wmer ή θα σταματήσει και θα ενεργοποιηθεί η πρώτη βαθμίδα πάλι. Η απόφαση αυτή προκύπτει από τις τιμές των σημάτων *no_extension*, *extend_byone*, *extend_bytwo* και *complete_extension*. Η επέκταση συνεχίζεται μόνο στην περίπτωση που ενεργοποιηθεί το σήμα complete_extension. Σε όλες τις άλλες περιπτώσεις η επέκταση σταματάει, υπολογίζεται το κατάλληλο score και ενεργοποιείται η πρώτη βαθμίδα. Στην περίπτωση που η επέκταση θα πρέπει να συνεχιστεί, η εκτέλεση οδηγείται στο βήμα 2, απενεργοποιώντας παράλληλα τη μονάδα



Σχήμα 3.11: Βήματα επέκτασης δεδομένων και υπολογισμός score ζευγών χαρακτήρων

extensionvalidation. Τέλος ενεργοποιείται η επιλογή *en_sel* των πολυπλεκτών ώστε στη συνέχιση της επέκτασης να χρησιμοποιηθούν τα δεδομένα που παράχθηκαν κατά την εκτέλεση του βήματος 2, για παράδειγμα η νέα αρχή και λήξη της εξεταζόμενης ακολουθίας. Το σχήμα 3.12 παρουσιάζει τα δύο τελευταία βήματα εκτέλεσης.

Η χρήση των καταχωρητών ύστερα από τις δομικές μονάδες findwmer, extendwmer και preparedbinput έχει τον σκοπό να διατηρούμε τις παραγόμενες τιμές δεδομένων, χωρίς να διατηρούμε τις δομικές μονάδες ενεργοποιημένες.



Σχήμα 3.12: Βήμα ελέγχου εγκυρότητας της επέκτασης

Αυτή η επιλογή έγινε, επειδή παρατηρήθηκε μη ομαλή λειτουργία της βαθμίδας. Πιο συγκεκριμένα γινόταν η λανθασμένη επιλογή - χρήση τιμών δεδομένων στις επόμενες δομικές μονάδες, γεγονός που οδηγούσε σε απρόβλεπτη εκτέλεση. Επομένως με τους καταχωρητές λύνεται το πρόβλημα αυτό.

3.5 Παρατηρήσεις

Στο σημείο αυτό πρέπει να παρατηρηθεί ότι η συγκεκριμένη υλοποίηση του αλγορίθμου BLASTp δεν υποστηρίζει πολλαπλά εμφανιζόμενα wmers. Εάν συμβεί να εμφανιστεί ένα wmer (π.χ. το AGC) παραπάνω από μία φορά στην υπό εξέταση ακολουθία, θα καταγραφεί ως wmer με πολλαπλότητα παραπάνω της μίας, αλλά κατά τη διαδικασία της επέκτασης θα ληφθεί χρησιμοποιηθεί μόνο το wmer, στο οποίο παρατηρήθηκε πρώτα το hit. Παρόλα αυτά μπορεί να αντιμετωπιστεί το πρόβλημα αυτό με πολλαπλή εφαρμογή της δεύτερης βαθμίδας. Η υλοποίηση του τρίτου βήματος δεν αντιμετωπίζει κανένα πρόβλημα στην εξυπηρέτηση πολλών ίδιων wmer, από τη στιγμή που στο βήμα αυτό πραγματοποιείται: αρχικά η επέκταση, στη συνέχεια ελέγχεται η εγκυρότητα της και τέλος γίνεται η εκτίμηση του score. Συνεπώς σε περίπτωση εμφάνισης πολλαπλού wmer σε ένα query, μπορεί να γίνει χρήση παραπάνω module της δεύτερης βαθμίδας και να εξυπηρετηθούν όλες οι "εμφανίσεις" του wmer.

Κεφάλαιο 4

Αποτελέσματα - Μετρήσεις

4.1 Γενικά

Παρακάτω περιγράφεται η διαδικασία που ακολουθήθηκε για την εκτέλεση της σχεδίασης των δύο stages, ενώ παρουσιάζονται τα αποτελέσματα της απόδοσης σε σύγκριση με ένα απλό προσωπικό υπολογιστή.

Για την υλοποίηση της σχεδίασης χρησιμοποιήθηκε το πρόγραμμα ISE 10.1, ενώ για την προσομοίωση της λειτουργίας των δύο stages ο ISE Simulator. Κατά τη διάρκεια υλοποίησης των εκάστοτε δομικών μονάδων, πραγματοποιήθηκαν διάφορες προσομοιώσεις, ώστε να διαπιστωθεί η ορθή και επιθυμητή λειτουργία τους. Στη συνέχεια αφού ενώθηκαν τα δύο stages, πραγματοποιήθηκε μια σειρά από προσομοιώσεις ώστε να συμπεριληφθούν όλα τα πιθανά σενάρια δεδομένων εισόδου και να διαπιστωθεί η εύρυθμη λειτουργία της σχεδίασης. Επίσης υπήρξε σύγκριση των αποτελεσμάτων των προσομοιώσεων με τα αποτελέσματα που προκύπτουν από την εκτέλεση του προγράμματος Blastp - software - στην έκδοση που είναι διαθέσιμη από το NCBI. Τέλος να παρατηρηθεί ότι λόγο του μεγάλου όγκου δεδομένων και επομένως του μεγάλου χρόνου που απαιτείτο για την εκτέλεση των προσομοιώσεων, χρησιμοποιήθηκε query - ακολουθία εισόδου μεγέθους 50 χαρακτήρων.

4.2 Δεδομένα Εισόδου - Απόδοση

Σύμφωνα με τον αλγόριθμο, απαιτείται μία ακολουθία εισόδου, η οποία αποτελεί την υπό εξέταση ακολουθία, και ένα σύνολο από ακολουθίες, οι οποίες αποτελούν τη βάση δεδομένων. Η υπό εξέταση ακολουθία έχει μήκος 50 χαρακτήρες, ενώ η βάση δεδομένων μέχρι 10.000 χαρακτήρες.

Αναφορικά με την υλοποίηση της σχεδίασης χρησιμοποιήθηκε η FPGA xc4vsx25 της οικογένειας Virtex 4, ταχύτητας -12 και το πακέτο ff668. Από

τη Synthesis και Implementation - Place & Route της σχεδίασης προέκυψε συχνότητα λειτουργίας 80 Mhz. Στον παρακάτω πίνακα 4.1 παρουσιάζονται διάφορες πληροφορίες που αφορούν στην υλοποίηση της σχεδίασης.

Logic Utilization	**Used**	**Available**	**Utilization**
Number of 4 input LUTs	2.832	20.480	13%
Number of occupied Slices	2.089	10.240	20%
Number of slices containg only related logic	2.089	2.089	100%
Number of BUFG/BUFGCTRLs	1	32	3%
Number of FIFO16/RAMB16s	18	128	14%

Table 4.1: Αναφορά Σύνθεσης

4.3 Επιβεβαίωση ορθής λειτουργίας της σχεδίασης

Είναι απαραίτητο να πιστοποιηθεί ότι η υλοποίηση των δύο βαθμίδων λειτουργεί και παράγει τα αναμενόμενα αποτελέσματα συγκριτικά με τα αποτελέσματα που προκύπτουν από την εκτέλεση του software προγράμματος BLASTp του NCBI. Για το σκοπό διεξήχθησαν μία σειρά από πειράματα για να εξακριβωθεί αρχικά η ορθή λειτουργία της σχεδίασης και στη συνέχεια να εξεταστεί εάν τα παραγόμενα αποτελέσματα - score για κάθε ακολουθία που εξετάζεται - είναι τα ίδια ή όχι με τα αντίστοιχα που παράγονται από το software implementation του BLASTp που υπάρχει στο NCBI.

Η διαδικασία εξακρίβωσης της ορθής λειτουργίας της σχεδίασης πραγματοποιήθηκε σταδιακά. Πιο αναλυτικά, στο πρώτο στάδιο χρησιμοποιήθηκαν query και database ακολουθίες που είχαν κατασκευαστεί με κριτήριο ποια λειτουργία της σχεδίασης ήταν επιθυμητό να ελεγχθεί. Είναι σημαντικό να σταματάει ο αλγόριθμος όταν τελειώνει η εξεταζόμενη ακολουθία ή η βάση δεδομένων. Για το λόγο αυτό γράφτηκαν συγκεκριμένα testbenches τα οποία παρήγαγαν query και database sequences, που ήλεγχαν εάν θα σταματήσει ο αλγόριθμος όταν συμβεί ένα από τα παρακάτω ενδεχόμενα:

- Η θέση του query στο wmer είναι πλευρική. Δηλαδή όταν το hit πραγματοποιηθεί είτε στην αρχή είτε στο τέλος της εξεταζόμενης ακολουθίας.

Επομένως αφού ανιχνευτεί το matching μεταξύ ενός wmer της λίστας του query και του αντίστοιχου wmer της λίστας της βάσης δεδομένων θα πρέπει κατά τη διαδικασία της επέκτασης να "καταλάβει" ο αλγόριθμος ότι η επέκταση είναι έγκυρη μόνο από τη μία πλευρά (π.χ. εάν το wmer είναι το τελευταίο στην υπό εξέταση ακολουθία η επέκταση μπορεί να πραγματοποιηθεί μόνο προς τα αριστερά).

- Ύστερα από μία σειρά έγκυρων επεκτάσεων, η τρέχουσα επέκταση φθάνει σε πλευρικό όριο. Δηλαδή είτε από αριστερά είτε από δεξιά δεν μπορεί να πραγματοποιηθεί τριπλή επέκταση (π.χ. έχουν απομείνει μόνο 2 χαρακτήρες διαθέσιμοι για την επέκταση από την αριστερή πλευρά).

- Περίπτωση να τελειώσει η βάση δεδομένων. Αυτός ο έλεγχος επιτυγχάνεται με την προσθήκη μίας stoping word (hex: "FOF"), την οποία αναγνωρίζει το component που δημιουργεί τις επεκτάσεις για τα δεδομένα της βάσης δεδομένων.

Οι παραπάνω περιπτώσεις εξετάστηκαν για διαφορετικές ακολουθίες εισόδου και βάσεις δεδομένων, ώστε να επιτευχθούν όλες οι ακραίες περιπτώσεις λειτουργίας και να παρατηρηθεί εάν η συμπεριφορά του κυκλώματος είναι η αναμενόμενη κάθε φορά και σύμφωνη με τις προδιαγραφές που προκύπτουν από τα βήματα του αλγορίθμου BLASTp. Στα συγκεκριμένα πειράματα καταγραφόταν πάντα το τρέχον score ώστε να διαπιστωθεί εάν μέχρι τη λήξη λειτουργίας και στα προηγούμενα βήματα επέκτασης - εφόσον υπήρχαν - είχε προκύψει το σωστό score. Ο έλεγχος αυτός πραγματοποιούταν με το χέρι και έχοντας πρώτα θεωρητικά υπολογιστεί ο αριθμός των επεκτάσεων που πρέπει να γίνουν καθώς και τι score προκύπτει για κάθε ζεύγος αμινοξέων που πρόκειται να συγκριθεί με βάση τον scoring πίνακα BLOSUM62.

Η παραπάνω πειραματική διαδικασία αποτελεί το πρώτο κριτήριο ότι η υλοποίηση που πραγματοποιήθηκε έχει την συμπεριφορά που πρέπει σε περίπτωση που οι είσοδοι για query και βάση δεδομένων εμπίπτουν σε μία - ή και περισσότερες - από τις παραπάνω περιπτώσεις που περιγράφηκαν. Στο δεύτερο στάδιο για τον έλεγχο της σωστής λειτουργίας είναι η άμεση σύγκριση των αποτελεσμάτων της υλοποίησης σε hardware με τα αντίστοιχα της software υλοποίησης του BLASTp του NCBI. Για το λόγο αυτό χρησιμοποιήσαμε ως δεδομένα εισόδου για την εξεταζόμενη ακολουθία και για τη βάση δεδομένων πραγματικά δεδομένα από το NCBI. Συγκεκριμένα χρησιμοποιήθηκε το αρχείο drosoph.aa, dataset από το NCBI - το οποίο περιλαμβάνει όλα τα γονίδια για το Drosophila, που είναι ένα γένος από μικρές μύγες, που ανήκουν στην οικογένεια Drosophilidae, της οποίας τα μέλη συχνά αποκαλούνται "μύγες φρούτων". Αρχικά στα πειράματα χρησιμοποιήθηκαν queries μήκους 50 χαρακτήρων στα όποια όλα τα wmer εμφανιζόντουσαν μία φορά. Μετά το

τέλος της εκτέλεσης των υλοποιήσεων τόσο σε hardware όσο και σε software του αλγορίθμου BLASTp, εξετάστηκαν τα τελικά score τα οποία προκύψαν ίδια. Ο αριθμός των queries που χρησιμοποιήθηκαν ήταν μικρός - 4, λόγω της μεγάλης απαίτησης σε χρόνο. Ως δεύτερο dataset χρησιμοποιήθηκαν queries μήκους επίσης 50 χαρακτήρων, πάλι από το αρχείο drosoph.aa, με τη διαφορά ότι περιείχαν κάποιο wmer περισσότερες της μία φοράς. Μετά το πέρας της εκτέλεσης των δύο υλοποιήσεων προέκυψαν διαφορετικά score. Ο λόγος για την απόκλιση που παρατηρήθηκε στο score έγγυται στο γεγονός ότι ο αλγόριθμος NCBI BLASTp πραγματοποιεί επεκτάσεις για όλες τις εμφανίσεις ενός wmer σε μία ακολουθία. Αντίθετα η υλοποίηση του αλγορίθμου BLASTp που περιγράφεται στην παρούσα διπλωματική πραγματοποιεί επεκτάσεις μόνο για το wmer το οποίο ταυτοποιείται πρώτο. Σε μελλοντικές επόμενες ταυτοποιήσεις για το ίδιο wmer, το οποίο εμφανίζεται σε άλλη θέση στην ακολουθία δεν πραγματοποιούνται επεκτάσεις. Αυτό έχει ως συνέπεια να χάνονται κάποια score τα οποία αναπόφευκτα μπορεί να οδηγήσουν σε λάθος απόφαση για το ποσοστό ομοιότητας μεταξύ της ακολουθίας και της βάσης δεδομένων. Τέλος να σημειωθεί ότι ο μικρός αριθμός του μήκους του query προέκυψε αναγκαστικά, επειδή μεγαλύτερες τιμές του query οδήγησαν σε crashing του υπολογιστή.

4.4 Μετρήσεις

Για την εκτέλεση του software προγράμματος Blastp, χρησιμοποιήθηκε ένας υπολογιστής Pentium 4 με συχνότητα λειτουργίας 3Ghz και μνήμη 1GB, έχοντας λειτουργικό σύστημα WindowsXP. Στον παρακάτω πίνακα 4.2 παρουσιάζονται συνοπτικά οι μετρήσεις που ελήφθησαν για query 50 χαρακτήρων, επειδή η διαδικασία είναι αρκετά χρονοβόρα, αλλά και λόγω της μικρής μνήμης του συστήματος δεν ήταν δυνατός ο έλεγχος για μεγαλύτερο μήκος ακολουθίας. Για την καλύτερη παρουσίαση της απόδοσης της σχεδίασης, χρησιμοποιήθηκαν συσκευές fpga δύο οικογενειών, μία της οικογενείας Virtex4 με συχνότητα λειτουργίας στα 80Mhz και μία της οικογένειας Virtex5 με συχνότητα λειτουργίας στα 100Mhz. Θα πρέπει να σημειωθεί ότι το *throughput* για τον αριθμό των χαρακτήρων που μπορούν να εξυπηρετηθούν ανά δευτερόλεπτο, μετρήθηκε για διάφορες τιμές μήκους της ακολουθίας της βάσης δεδομένων. Πιο αναλυτικά χρησιμοποιήθηκαν ακολουθίες μήκους 100 χαρακτήρων - περισσότερο για εξέταση της ορθής λειτουργίας του συστήματος - και στη συνέχεια αυξανόταν σταδιακά το μήκος της εκάστοτε ακολουθίας που χρησιμοποιούταν, αποσκοπώντας σε καλύτερες και λεπτερομέραστατες προσομοιώσεις. Κατά τη διάρκεια των προσομοιώσεων συμπεριλήφθησαν πολλές περιπτώσεις, σε ό,τι αφορά τον αριθμό των matching - hits που εντοπίζονται. Συνεπώς υπήρξε

testbench για μόνο ένα match, και υποπεριπτώσεις του να συμβεί αυτό το match στην αρχή ή στο τέλος της ακολουθίας. Στη συνέχεια χρησιμοποιήθηκαν testbenches για ένα match στην αρχή και ένα στο τέλος της ακολουθίας, ύστερα testbenches με διαδοχικά matching ή σποραδικά, ενώ εξετάστηκε και το worst case senario, στο οποίο η υπό εξέταση ακολουθία έχει πολύ υψηλή ομοιότητα με τις αντίστοιχες ακολουθίες της βάσης δεδομένων. Ο λόγος που πραγματοποιήθηκαν αρκετά testbenches με διαφορετικό αριθμό matching αλλά και πυκνότητάς τους - συχνότητα που εμφανίζονται στην ακολουθία - είχε ως σκοπό να συμπεριληφθούν όσο το δυνατόν περισσότερες περιπτώσεις προσομοίωσης λειτουργίας του αλγορίθμου Blastp.

Device	**Throughput**
Virtex4	$40{\cdot}10^6$
Virtex5	$50{\cdot}10^6$
Pentium4@3	$19{,}80{\cdot}10^6$

Table 4.2: Throughput ανάλογα με την συσκευή

Από τα παραπάνω που έχουν αναφερθεί, προκύπτει ο παρακάτω πίνακας 4.3, ο οποίος παρουσιάζει τις τιμές που επιτυγχάνονται αναφορικά με το speed up που λαμβάνεται ως προς τον Pentium4.

Speedup	**Device**
$x2,020$	xc4vsx25
$x2,525$	xc5vlx50

Table 4.3: Τιμές Speedup

4.5 Εκτίμηση Αποτελεσμάτων

Για τον υπολογισμό δυνατοτήτων της σχεδίασης έγινε προβολή των αποτελεσμάτων που ελήφθησαν για μεγαλύτερα ολοκληρωμένα κυκλώματα. Πιο συγκεκριμένα για Virtex 4 SX 55, Virtex 5 LX330T και Virtex 6 HX565T. Η επιλογή των παραπάνω κυκλωμάτων έγινε με κριτήριο τη χωρητικότητα τους για τον κρίσιμο πόρο που στην παρούσα σχεδίαση είναι η λογική. Στον πίνακα 4.4

Τέλος εάν υποτεθεί ότι οι ταχύτητες ρολογιού για τα συγκεκριμένα chip με τις συγκεκριμένες παράλληλες μηχανές είναι 60 Mhz, 80 Mhz και 100Mhz αντίστοιχα, τότε η ταχύτητα υπολογισμού και η επιτάχυνση σε σχέση με έναν επεξεργαστή γενικού σκοπού προκύπτουν όπως φαίνεται στον πίνακα 4.5

Device	**Αριθμός Παράλληλων μηχανών**
Virtex 4 SX 55	12
Virtex 5 LX330T	25
Virtex 6 HX565T	44

Table 4.4: Χωρητική αναπαράσταση της σχεδίασης ανάλογα με τη χρησιμοποιούμενη συσκευή

Devive	**Throughput Char 10^6 / sec**	**SpeedUp**
Virtex 4 SX 55	360	18.18
Virtex 5 LX30T	1.000	50.50
Virtex 6 HX565T	2.200	111.11

Table 4.5: Εκτιμώμενες τιμές για Throughput και SpeedUp ανάλογα τη χρησιμοποιούμενη συσκευή

Κεφάλαιο 5

Συμπεράσματα - Μελλοντικές επεκτάσεις

5.1 Συμπεράσματα

Στην παρούσα διπλωματική πραγματοποιήθηκε μία υλοποίηση του αλγορίθμου Blastp, χρησιμοποιώντας αναδιατασσόμενη λογική, έχοντας ως στόχο να μελετηθεί η απόδοση της σχεδίασης αυτής, αλλά και να συγκριθούν τα αποτελέσματά της με αυτά που προκύπτουν από έναν προσωπικό υπολογιστή Pentium4 με συχνότητα λειτουργίας 3.00Ghz. Όπως παρατηρήθηκε στις μετρήσεις στο προηγούμενο κεφάλαιο, μπορεί κάποιος να πει ότι η υλοποίηση που περιγράφθηκε στο κεφάλαιο 3, είναι ικανοποιητική και αποδοτικότερη συγκριτικά με έναν Pentium4.

5.2 Μελλοντικές Επεκτάσεις

Η αρχιτεκτονική που σχεδιάστηκε και υλοποιήθηκε στα παραπάνω κεφάλαιο επιδέχεται αλλαγές που ενδεχομένως να οδηγήσουν σε βελτιστοποιήσεις. Αρχικά μπορεί να γίνει προσπάθεια για μεγαλύτερο παραλληλισμό της σχεδίασης, ώστε να γίνονται ταυτόχρονα κάποιες χρονοβόρες διεργασίες. Εάν ληφθεί υπόψη ότι η συγκεκριμένη υλοποίηση του αλγορίθμου καταλαμβάνει το 20% της fpga, τότε εύλογα προκύπτει το ερώτημα για εκμετάλλευση όλου του χώρου της. Επομένως εάν γίνει χρήση τέσσερις ή πέντε φορές της ίδιας σχεδίασης, συνδυασμένη με έναν controller για τη μεταξύ τους επικοινωνία θα μπορούσαν να προκύψουν ακόμα καλύτερα αποτελέσματα. Πιο αναλυτικά να πραγματοποιηθεί κατακερματισμός των ακολουθιών της βάσης με τέτοιο τρόπο ώστε να εξετάζονται πολλές ακολουθίες ταυτόχρονα. Ο controller θα ρυθμίζει το κατακερματισμό αυτό, το πώς θα εξυπηρετούνται τα matching - ένα είδος

arbitration, αλλά και πώς θα υπολογίζεται το score, ώστε να αποφεύγονται διπλές καταχωρήσεις του ίδιου score. Επίσης πιθανή αλλαγή στην προσέγγιση χρήσης των μνημών θα μπορούσε να οδηγήσει σε καλύτερα αποτελέσματά.

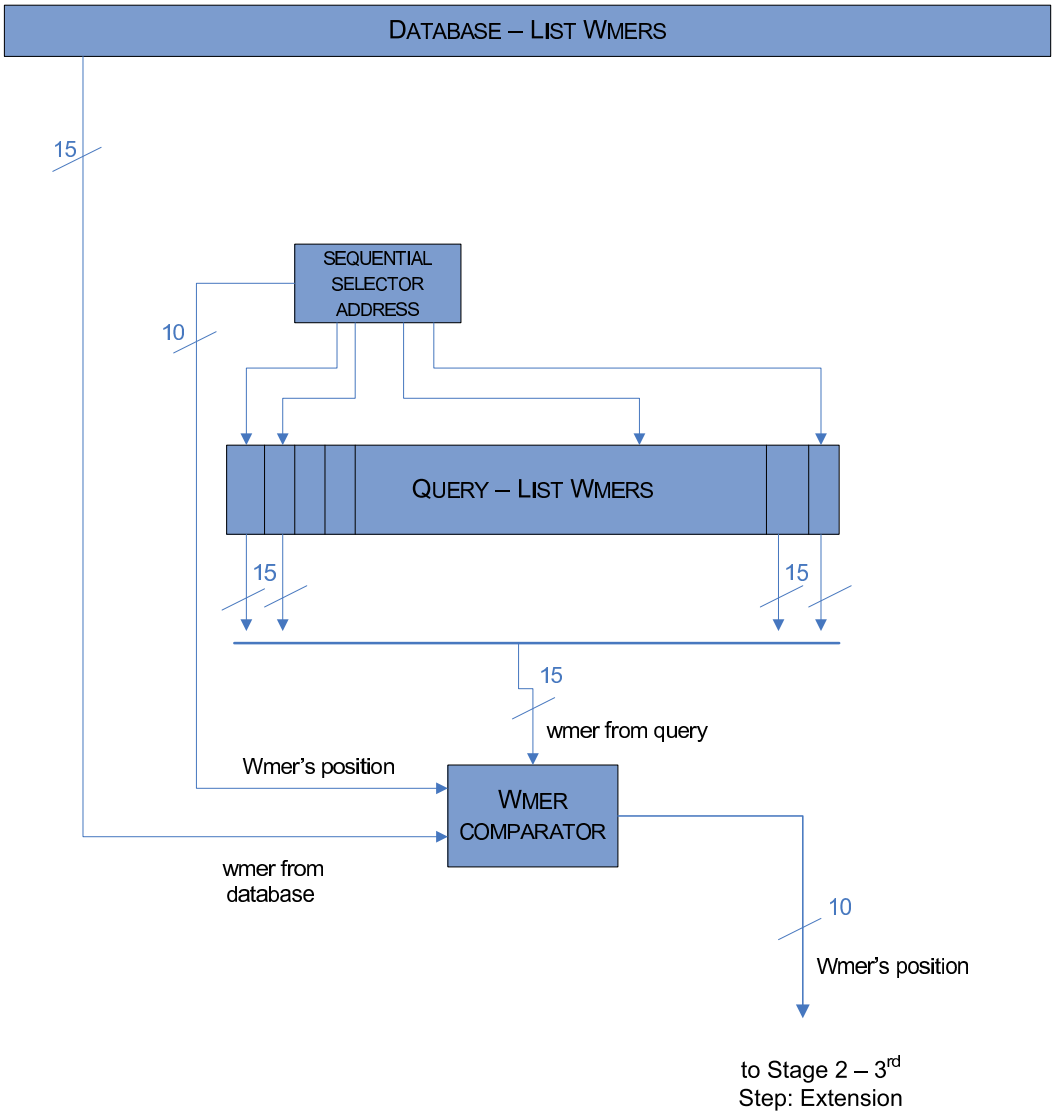
Τέλος η υλοποίηση μίας linked list για την εξυπηρέτηση των πολλαπλών εμφανίσεων ενός wmer στην υπό εξέταση ακολουθία είναι μία εφαρμογή που είναι απαραίτητη με την υπάρχουσα αρχιτεκτονική. Η χρήση της συνδεδεμένης λίστας θα δίνει τη δυνατότητα να διατηρούνται όλες οι θέσεις του wmer στο query, που εμφανίζει πολλαπλότητα. Με τον τρόπο αυτό θα είναι διαθέσιμες όλες οι θέσεις της πολλαπλότητας και θα μπορεί να πραγματοποιηθεί η προβλεπόμενη επέκταση για κάθε εμφάνιση του wmer. Επομένως θα είναι δυνατή η λήψη πιο λεπτομερών (σε πλήρη σύμπνοια με τον NCBI BLASTp) αποτελεσμάτων και στην περίπτωση που η εξεταζόμενη ακολουθία έχει πολλές κοινές περιοχές.

5.3 Προτεινόμενη λύση - Πολλαπλότητα Wmer

Δεδομένης της δυναμικότητας που παρουσιάζουν οι συνδεδεμένες λίστες (linked - lists) σε απαιτήσεις χώρου είναι δύσκολη η εφαρμογή τους σε FPGAs. Το σημαντικότερο μειονέκτημά τους είναι ότι επιβάλλεται χωρικός περιορισμός λόγω μνημών πεπερασμένου μεγέθους. Εκτιμάται λοιπόν ότι η βέλτιστη λύση θα είναι ως εξής:

Αρχικά πρέπει να γίνει μια διαφορετική προσέγγιση στην αρχιτεκτονική του δεύτερου βήματος. Στο παρακάτω σχήμα 5.1 παρουσιάζεται μία νέα αρχιτεκτονική για το βήμα αυτό. Η εξεταζόμενη ακολουθία φορτώνεται σε μία μνήμη πλάτους 15 bits και βάθους ίσου με το μήκος της ακολουθίας. Τα δεδομένα της βάσης δεδομένων θα έρχονται όπως και στην προηγούμενη σχεδίαση μέσω I/O. Επίσης θα υπάρχει ένας ακολουθιακός επιλογέας διεύθυνσης, ο οποίος θα χρησιμοποιείται για να "δείχνει" στην επόμενη θέση της μνήμης του query. Για κάθε νέο wmer που προκύπτει από τη βάση δεδομένων θα πραγματοποιείται έλεγχος με όλα τα wmers της query, μέσω ενός συγκριτή wmer. Στην περίπτωση που γίνει κάποιο hit, η τρέχουσα θέση του wmer που προκύπτει από τον ακολουθιακό επιλογέα διεύθυνσης θα οδηγείται στη βαθμίδα 2, η οποία είναι αυτή που σχεδιάστηκε για το βήμα της επέκτασης στην παρούσα διπλωματική. Για λόγους απλότητας το query έχει φορτωθεί σε μία δεύτερη μνήμη πανομοιότυπη με την πρώτη, ώστε να πραγματοποιούνται οι επεκτάσεις. Όταν ολοκληρωθεί η διαδικασία της επέκτασης για το τρέχων wmer και έχει υπολογιστεί το ανάλογο score, θα συνεχίζεται η αναζήτηση νέου hit, έως ότου ο ακολουθιακός επιλογέας διεύθυνσης δώσει και το τελευταίο wmer του query. Με τον τρόπο αυτό υποστηρίζεται πλήρως η πολλαπλότητα του wmer, αφού μετά την ολοκλήρωση της διαδικασίας επέκτασης για την πρώτη εμφάνιση του

wmer, συνεχίζεται η αναζήτηση για νέο hit (για το τρέχων wmer της βάσης δεδομένων) έως ότου τελειώσει το query, οπότε εάν εμφανιστεί ξανά το ίδιο wmer στο query πραγματοποιείται και πάλι το βήμα της επέκτασης του αλγορίθμου. Όταν ο ακολουθιακός επιλογέας διεύθυνσης φτάσει στο τέλος του query, τότε εκτελείται η ίδια διαδικασία για το επόμενο wmer της βάσης δεδομένων.



Σχήμα 5.1: Γενική άποψη προτεινόμενης αρχιτεκτονικής

Βιβλιογραφία

[1] http://www.accessexcellence.org/rc/vl/gg/proteinsynthesis.php.

[2] http://www.ncbi.nlm.nih.gov/education/blastinfo/scoring2.html.

[3] http://www.scienceaid.co.uk/biology/genetics/proteinsynthesis.html.

[4] Matrices comparison: www.cs.iastate.edu/ cs544/lectures/blosum_matrices.ppt.

[5] Ribosomes and transfer rna: http://advice.tutors-connect.com/265/protein-synthesis-emphasis-on-ribosomes-and-trna/.

[6] www.calstatela.edu/faculty/jmomand/alignmentmethods_03.ppt.

[7] www.cs.iastate.edu/ cs544/lectures/blosum_matrices.ppt.

[8] *Signed integers are two's complement binary values that can be used to represent both positive and negative integer values*, volume 1, chapter 4.2.1. Intel, November 2006.

[9] P. Afratis, C. Galankis, E. Sotiriades, G. Mplemenos, G. Chrysos, I. Papaefstathiou, and D. Pnevmatikatos. "design and implementation of a database filter for blast acceleration". Procceedings Design, Automation and Test Europe, April 2009.

[10] P. Afratis, E. Sotiriades, G. Chrysos, S. Fytraki, and D. Pnevmatikatos. "a rate-based prefiltering approach to blast acceleration". International Conference on Field Programmable Logic and Applications FPL, 2008.

[11] P. Afratis, E. Sotiriades, G. Chrysos, S. Fytraki, and D. Pnevmatikatos. "preprocessor for blast algorithm data". page 12. 2nd Greek ECE Student Conference, 2008.

[12] Churbanov Alexander. Pam matrix for blast algorithm. *PAM matrix for BLAST algorithm*, April 2002.

[13] Stephen F. Altschul. *Amino acid substitution matrices from a information theoretic perspective*, volume 3, pages 555–565. Jurnal of molecular biology, 1991.

[14] STEPHEN F. ALTSCHUL*t and JOHN D. KECECIOGLUt DAVID J. LIPMAN*t. Sequence alignments, March 1989.

[15] Alberts Bruce. *Molecular biology of the cell.* Garland Science, 2002.

[16] Michael Brudno, Sanket Malde, NewAuthor1, Poliakov Alexander, Chuong B. Do, Olivier Couronne, Inna Dubchak, and Serafim Batzoglou. Glocal alignment:finding rearrengements during alignmet, Junary 2003.

[17] Branden C. and Tooze J. *Introduction to Protein Structure.* Garland Pub, 1999.

[18] C. Chang. "blast implementation on bee2". Technical report, University of California at Berkeley, 2004.

[19] Hugh Chisholm, editor. *Encyclopædia Britannica*, volume 22, pages –. Horace Everett Hooper,, eleventh edition edition, 1910–1911.

[20] Sotiriadis Euripides and Apostolos Dollas. "a general reconfigurable architecture for the blast algorithm". Technical Report 3, Department of Electronic and Computer Engineering, Technical University of Crete, September 2007.

[21] S. Guccione and Eric Keller. *"Gene Matching Using Jbits"*, volume 2438, pages 1168–1171. Procceedings of the 12th International Conference on Field-Programmable Logic and Applications, Lecture Notes in Computer Science, 2002.

[22] M. C. Herbordt, J. Model, Y. Gu, B. Sukhwani, and T. VanCourt. *"Single Pass, BLAST-Like, Approxing String Matching on FPGAs"*, pages 217–226. 14th IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'06), 2006.

[23] D. Hoang. *"FPGA Implementation of Systolic Sequence Alingment"*, volume 705, pages 183–191. Procceedings of the 2nd International Workshop on Field-Programmable Logic and Applications, Lecture Notes in Computer Science, 1992.

[24] P. Krishnamurthy, J. Buhler, R. Chamberlain, M. Franklin, k. Gyang, and J. Lancaster. *"Biosequence Similarity Search on the Mercury System"*, pages 365–375. Proc. of the IEEE 15th Int'l Conf. on Application-Specific Systems, Architectures and Proccesors, September 2004.

[25] J. Lancaster, J. Buhler, and R. Chamberlain. *"Acceleration of Ungapped Extension in Mercury BLAST"*, page . 17th workshop on media and streaming processors, November 2005.

[26] D. Lavenier, L. Xinchun, and G. Georges. "seed-based genomic sequence comparison using fpga/flash accelerator". In *Procceedings of IEEE International Conference on Field Programmable Technology (FPT'06)*, pages 41–48, 2006.

[27] Ridley M. *Genome*. Harper Perennial, 2006.

[28] Dayhoff M.O., Schwartz R.M., and Orcutt B.C. *A modelof evolutionary change in proteins*, volume 5, pages 345–352. Atlas of Protein Sequence and Structure, 1978.

[29] D.W. Mount. *Bioinformatics: Sequence and Genome Analysis*. Cold Spring Harbor Press, 2004.

[30] K. Muriki, K. Underwood, and R. Sass. "rc-blast". In *Procceedings of the International Workshop on High Perfomance Computational Biology*, 2005.

[31] NCBI. Ncbi- blast tutorial.

[32] T. Oliver, B. Schmidt, and D. Maskel. *"Reconfigrable Architectures for Bio-sequence Database Scanning on FPGA's"*, volume 52, pages 851–855. IEEE Trans. Circuits Syst. II, 2005.

[33] K. Puttegowda. "a run time reconfigurable system for gene-sequence searching". In *In Procceedings 16th International Conference on VLSI Design*, volume -, pages 561–565, 2003.

[34] Casey RM. Blast sequences aid genomics and proteomics, 2005.

[35] Henikoff S. and Henikoff J.G. *Amino acid Substitution Matrices from Protein Blocks*, volume 89, pages 10915–10919. PNAS, 1992.

[36] Sean R. Eddy. Where did the blosum62 aligment score matrix come from? *Nature Biotechnology*, 22(8):1035,1036, August 2004.

[37] E. Sotiriades and A. Dollas. "design space exploration for the blast algorithm implementation". pages 323–326. Procceedings of the 15th IEEE International Symposium on Field -Programmable Custom Computing Machines (FCCM 2007), April 2007.

[38] E. Sotiriades, C. Kozanitis, G. Chrysos, and A. Dollas. "rapid prototyping of a system-on-a-chip for the blast algorithm implementation". pages 223–229. Procceedings 17th International IEEE Workshop on Rapid Prototyping (RSP), June 2006.

[39] E. Sotiriades, C. Kozanitis, and A. Dollas. "fpga based architecture for dna sequence comparison and database search". Procceedings of the 13th Reconfigurable Architecture Workshop (RAW) (on-line procceedings under the IPDPS conference), April 2006.

[40] E. Sotiriades, C. Kozanitis, and A. Dollas. "some initial results on hardware blast acceleration with a reconfigurable architecture". Procceedings of the 8th Workshop on High Perfomance Computational Biology (HiCOMB)(on-line procceedings under the IPDPS conference), April 2006.

[41] Jorg von Hagen. *Proteomics*. VCH-Wiley, 2008.

[42] Zwi Litoy, Pantelis Mpagos, and Stavros I. Xamodrakas. *Versions Of BLAST Algorithm*. PhD thesis, Department of Cell Biology & Biophysics, Department of Biology, University of Athens, February 2002.