Αρχιτεκτονική και Υλοποίηση σε Αναδιατασσόμενη Λογική του Αλγορίθμου T-Coffee για συνένωση κομματιών DNA

Λάκκα Ματίνα

Διπλωματική Εργασία

Πολυτεχνείο Κρήτης
Τμήμα Ηλεκτρονικών Μηχανικών
και Μηχανικών Ηλεκτρονικών Υπολογιστών
Εργαστήριο Μικροεπεξεργαστών και Υλικού

Επιτροπή
Δόλλας Απόστολος, Καθηγητής (Επιβλέπων)
Παπαευσταθίου Ιωάννης, Επίκουρος Καθηγητής
Πνευματικάτος Διονύσιος, Αναπληρωτής Καθηγητής

Χανιά 2009

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον καθηγητή μου κ. Δόλλα Απόστολο, επιβλέποντα της παρούσας διπλωματικής εργασίας, για την εποικοδομητική συνεργασία σε ότι αφορά τη διπλωματική μου, αλλά και για την ψυχολογική υποστήριξη σε δύσκολες στιγμές που παρουσιάστηκαν κατά τη διάρκεια αυτής. Επίσης, θα ήθελα να ευχαριστήσω τον Επίκουρο Καθηγητή κ. Παπαευσταθίου Ιωάννη και τον Αναπληρωτή Καθηγητή κ. Πνευματικάτο Διονύσιο για τη συμμετοχή τους ως μέλη της εξεταστικής επιτροπής.

Ένα μεγάλο ευχαριστώ στους διδακτορικούς φοιτητές Ευριπίδη Σωτηριάδη και Γρηγόρη Χρυσό για τη συνεργασία τους, την πολύτιμη βοήθεια τους και την τεράστια υπομονή τους για την περάτωση της διπλωματικής μου εργασίας.

Ευχαριστώ θερμά επίσης τον κ. Κιμιωνή Μάρκο, υπεύθυνο του εργαστηρίου Μικροεπεξεργαστών και Υλικού, για την ταχεία εξυπηρέτηση του όταν χρειάστηκε να εφοδιαστώ με τον κατάλληλο τεχνολογικό εξοπλισμό.

Επίσης, να ευχαριστήσω τον Dr. Cedric Notredame, συγγραφέα του αλγορίθμου T-Coffee, για την άμεση ηλεκτρονική επικοινωνία κατά το στάδιο της κατανόησης του αλγορίθμου για την μετέπειτα υλοποίηση του σε hardware.

Δεν γίνεται να παραλείψω να ευχαριστήσω τους φίλους μου, εντός και εκτός τους εργαστηρίου Μικροεπεξεργαστών και Υλικού, που μου συμπαραστάθηκαν όλο αυτό τον καιρό και γενικά για όλες τις ευχάριστες στιγμές που έχουμε ζήσει μαζί.

Πάνω απ’ όλους όμως θα ήθελα να ευχαριστήσω τους γονείς μου Νίκο και Εύη, και τον αδερφό μου Γιώργο που είναι πάντα δίπλα μου σε όποια απόφαση και αν πάρω.

Περίληψη

Η Πολλαπλή Ευθυγράμμιση Ακολουθιών είναι μακράν το πιο κοινό έργο στην βιοπληροφορική. Οι διαδικασίες που βασίζονται σε σύγκριση ακολουθιών είναι ποικίλες και εκτείνονται από έρευνες σε βάσεις δεδομένων για την πρόγνωση δευτερεύουσας δομής. Οι ακολουθίες μπορούν να συγκριθούν ανά δύο σε βάσεις δεδομένων για την εύρεση ομόλογων αλληλουχιών, ή μπορεί να γίνει πολλαπλή ευθυγράμμιση για να απεικονίσει το αποτέλεσμα της εξέλιξης σε μια ολόκληρη οικογένεια πρωτεϊνών.

Η Πολλαπλή Ευθυγράμμιση Ακολουθιών αποτελεί ουσιαστικό εργαλείο για την ανάλυση της πρωτεϊνικής δομής και πρόβλεψη λειτουργίας των πρωτεϊνών. Πρόσφατα αναπτύχθηκαν συστήματα που έχουν προχωρήσει στην εξέλιξη της τεχνολογίας σε ότι αφορά την ακρίβεια, την ικανότητα διαχείρισης χιλιάδων πρωτεϊνών και την ευελιξία ως προς τη σύγκριση πρωτεϊνών που δεν μοιράζονται τον ίδιο τομέα αρχιτεκτονικής. Προκειμένου να βελτιωθεί η ακρίβεια και η ταχύτητα της Πολλαπλής Ευθυγράμμισης Ακολουθιών έχουν αναπτυχθεί διάφοροι αλγόριθμοι με στόχο το αποτέλεσμα που προκύπτει να έχει βιολογική σημασία.

Στα πλαίσια της παρούσας διπλωματικής εργασίας μελετήθηκε, σχεδιάστηκε και υλοποιήθηκε σε αναδιατασσόμενη λογική ο αλγόριθμος T-Coffee που για μικρό αριθμό ακολουθιών εκτελεί ακριβή και ταχεία Πολλαπλή Ευθυγράμμιση Ακολουθιών.

Περιεχόμενα

1. Εισαγωγή ……. 11

1.1 Τι είναι πολλαπλή ευθυγράμμιση ακολουθιών (MSA)…..…….. 11

1.2 Γιατί να κάνουμε πολλαπλή ευθυγράμμιση ακολουθιών (MSA……..….. 14

2. Σχετική ΄Έρευνα ……...…….…… 15

2.1 Μέθοδοι MSA και σχετικοί αλγόριθμοι.……..…..…..…… 15

2.1.1 Ανάλυση μεθόδων για τη δημιουργία MSA ……..…...…… 16

2.1.2 Multiple Sequence Alignment Αλγόριθμοι…..……...…..…. 17

2.2 Multiple Sequence Alignment χρησιμοποιώντας FPGAs…….....…….. 22

3. Αλγόριθμος T-Coffee και Μοντελοποίηση ……...……..... 25

3.1 Ο αλγόριθμος T-Coffee …….. 25

3.2 Περιγραφή του αλγορίθμου T-Coffee …….……..... 27

3.3 Μοντελοποίηση του αλγορίθμου T-Coffee ……..….. 34

3.3.1 Ανάλυση software αλγορίθμου T-Coffee ……..… 34

3.3.2 Μοντελοποίηση συνάρτησης residue_pair_extended_list……..... 35

3.3.3 Μοντελοποίηση συνάρτησης residue_pair_extended_list……..... 38

4. Αρχιτεκτονική Σχεδίασης …….……..... 39

4.1 Περιγραφή της σχεδίασης και της υλοποίηση της συνάρτησης residue_pair_exetnded_list……..... 39

4.2 Περιγραφή της σχεδίασης και της υλοποίηση της συνάρτησης residue_pair_exetnded_list……..... 46

5. Αποτελέσματα-Μετρήσεις …… 55

5.1 Ταυτοποίηση λειτουργίας ……..…...…… 55
5.2 Απόδοση συστήματος ……..……...……..…… 56
5.3 Αποτίμηση Απόδοσης ……...…..…….. 58

6. Συμπεράσματα μελλοντικές επεκτάσεις και αλλαγές …….……....…… 63

6.1 Συμπεράσματα ……...……...…..…….... 63
6.2 Μελλοντικές επεκτάσεις ……..…..……..…… 63

7. Βιβλιογραφία ……..……. 65

Βασικοί Όροι

Alignment: Διαδικασία σύγκρισης 2 (pairwise) ή περισσότερων (multiple) ακολουθιών-αλληλουχιών ψάχνοντας για μια σειρά από όμοιους ή παρόμοιους χαρακτήρες (**residues**) στις ακολουθίες-αλληλουχίες

Multiple Sequence Alignment (MSA): Πολλαπλή ευθυγράμμιση αλληλουχιών είναι η ευθυγράμμιση των N> 2 αλληλουχιών που λαμβάνεται με την προσθήκη ("-") κενών (gaps) μεταξύ ακολουθιών ώστε οι ακολουθίες που προκύπτουν να έχουν όλες μήκος L και μπορεί να ρυθμιστεί σε ένα πίνακα (matrix) με M σειρές και L στήλες όπου κάθε στήλη αντιπροσωπεύει μία ομόλογη θέση (κάθε στήλη αντιστοιχεί σε ένα συγκεκριμένο αμινοξύ ή νουκλεοτίδιο (residue) στη "prototypical" protein)

Pairwise Alignment: Ευθυγράμμιση δύο ακολουθιών. Δύο αλληλουχίες συμπληρώνονται με κενά, για την επίτευξη ίδιου μήκους, και για την απεικόνιση maximum similarity/conservation on a character-by- character basis. Μία βέλτιστη Pairwise Alignment είναι μία ευθυγράμμιση που έχει το βέλτιστο score

Homology: Συσχέτιση ακολουθιών που είναι αποτέλεσμα απόκλισης από κοινό πρόγονο

Identity: Ακολυθίες ή υπο-ακολουθίες που είναι αμετάβλητες, δηλαδή ο αριθμός των όμοιων βάσεων ή αμινοξέων που ταιρίαζουν (match) μεταξύ δύο ευθυγραμμισμένων ακολουθιών

Percent Identity: Παράγεται από τη διαίρεση του προηγούμενου αριθμού με το συνολικό μήκος της ευθυγραμμισμένης ακολουθίας (aligned sequence) και πολλαπλασιάζοντας επί εκατό

Weight: Τιμή που αποδίδεται σε κάθε ζεύγος χαρακτήρων ανάλογα το identity

Similarity: Ακολυθίες ή υπο-ακολουθίες που σχετίζονται μεταξύ τους

Consensus: Η γραμμή στο αρχείο εξόδου με τους χαρακτήρες *** : .**
που δείχνει τη μέση τιμής αξιοπιστίας της κάθε στήλης στο τελικό alignment

Gap open: Ποινή για το πρώτο residue σε ένα κενό

Gap Extension: Ποινή για επιπλέον residue σε ένα κενό

Gap Penalty: Gap open, Gap Extension

Match: Πανομοιότυποι χαρακτήρες

Mismatch: Διαφορετικοί χαρακτήρες

Optimization: Η βελτιστοποίηση αναφέρεται στο πρόβλημα, στο οποίο κάποιος επιδιώκει να ελαχιστοποιήσει ή να μεγιστοποιήσει τη λεγόμενη objective (cost) function (OF) με τη συστηματική επιλογή των τιμών των ανεξάρτητων μεταβλητών της παρούσας λειτουργίας μέσα από ένα επιτρεπτό σύνολο.

Objective Function: Είναι το μαθηματικό μέτρο για την ποιότητα μιας MSA. Ιδανικά, όσο καλύτερο είναι το score της τόσο η MSA είναι βιολογικά ορθή

Κεφάλαιο 1

Εισαγωγή

Στο πρώτο κεφάλαιο της παρούσας διπλωματικής εργασίας γίνεται εισαγωγή στο θέμα με το οποίο αυτή πραγματεύεται, δηλαδή τι είναι και για ποιους λόγους χρησιμοποιείται η Πολλαπλή Ευθυγράμμιση Ακολουθιών (Multiple Sequence Alignment -MSA).

1.1 Τι είναι Multiple Sequence Alignment (MSA)

Για πολλά γονίδια μία έρευνα σε μία βάση δεδομένων θα αποκάλυπτε μια ολόκληρη σειρά ομόλογων ακολουθιών. Στη συνέχεια, κάποιος θα ήθελε να μάθει για την εξέλιξη και την διατήρηση της ακολουθίας σε μια τέτοια ομάδα. Το ζήτημα αυτό υπερβαίνει αυτό που μπορεί λογικά να επιτευχθεί με τις μεθόδους σύγκρισης ακολουθιών. Οι pairwise comparisons δεν δείχνουν άμεσα τις θέσεις που διατηρούνται από ένα σύνολο ακολουθιών και τείνουν να χάσουν λεπτές ομοιότητες που γίνονται ορατές όταν παρατηρούνται ταυτόχρονα μεταξύ πολλών ακολουθιών. Έτσι, κάποιος θα ήθελε να συγκρίνει ταυτόχρονα πολλές αλληλουχίες .

Πολλαπλή Ευθυγράμμιση Ακολουθιών (Multiple Sequence Alignment -MSA) μπορεί να θεωρηθεί ως μια γενίκευση της Pairwise Sequence Alignment - αντί για ευθυγράμμιση δύο ακολουθιών, N ακολουθίες έχουν ευθυγραμμιστεί ταυτόχρονα, όπου N είναι > 2 .

MSA εφαρμόζεται τόσο για νουκλεοτιδικές όσο και για ακολουθίες αμινοξέων. Για να κατασκευαστεί μια MSA, μπορεί να χρειάζεται η εισαγωγή κενών (gaps) σε ακολουθίες στις θέσεις όπου δεν υπάρχουν κενά στην αντίστοιχη pairwise alignment. Αυτό σημαίνει ότι πολλαπλές ευθυγραμμίσεις συνήθως περιέχουν περισσότερα κενά από οποιοδήποτε άλλο ζεύγος ευθυγραμμισμένων ακολουθιών.

Μια πολλαπλή ευθυγράμμιση (Multiple Sequence Alignment) οργανώνει μια σειρά ακολουθιών σε ένα σχήμα (scheme-matrix) όπου οι θέσεις που πιστεύεται ότι είναι ομόλογες είναι γραμμένες σε μια κοινή στήλη. Όπως για Pairwise Alignment έτσι και για Multiple Sequence Alignment, όταν μια ακολουθία δεν διαθέτει ένα αμινοξύ σε μια συγκεκριμένη

θέση αυτό δηλώνεται με μια παύλα. Επίσης όπως για Pairwise Alignment, υπάρχουν συμβάσεις σχετικά με την βαθμολόγηση (scoring) της πολλαπλής ευθυγράμμισης. Σε μια απλή προσέγγιση, προσθέτονται όλα τα αποτελέσματα από τις pairwise aligments που περιλαμβάνονται σε μία multiple alignment. Για ένα linear gap penalty αυτό έχει ως αποτέλεσμα να βαθμολογηθεί η κάθε στήλη της τελικής ευθυγράμμισης με το άθροισμα όλων των amino acid pair scores σε αυτή τη στήλη. Το αντίστοιχο score ονομάζεται sum of pairs (SP-) score.

Αν και θα είχε νόημα βιολογικά, οι διακρίσεις μεταξύ των διαφόρων μορφών ευθυγράμμισης σπάνια λαμβάνονται υπόψη κατά την πολλαπλή ευθυγράμμιση, γεγονός που οφείλεται στις υπολογιστικές δυσκολίες κατά τον υπολογισμό των Multiple Sequence Alignments.

Υπάρχουν διάφοροι τύποι ευθυγράμμισης αλλά στην παρούσα διπλωματική εργασία θα μας απασχολήσουν οι εξής:

1. **Global alignment** : Τρόπος ευθυγράμμισης που βασίζεται στην υπόθεση ότι δύο πρωτεΐνες είναι κατά βάση παρόμοιες σε όλο τους το μήκος. Ο τρόπος ευθυγράμμισης αυτός επιχειρεί να ταιριάξει (match) τις δύο πρωτεΐνες μεταξύ τους κατά το μήκος τους , ακόμη και αν κάποια μέρη της ευθυγράμμισης δεν είναι πολύ πειστικά.

Ένα μικρό παράδειγμα:

```
NLGPSTKDFGKISESREFDNQ
 |      ||||    |
QLNQLERSFGKINMRLEDALV
```

2. **Local alignment** : Τρόπος ευθυγράμμισης που ψάχνει για τμήματα (segments) των δύο αλληλουχιών που ταιριάζουν καλά. Δεν υπάρχει καμία προσπάθεια ώστε να εισέλθει ολόκληρη ακολουθία (sequence) στην ευθυγράμμιση, μόνο τα τμήματα εκείνα που φαίνεται να έχουν καλή ομοιότητα, σύμφωνα με κάποιο κριτήριο.

Με βάση τις ίδιες ακολουθίες όπως παραπάνω, θα μπορούσε κανείς να λάβει:

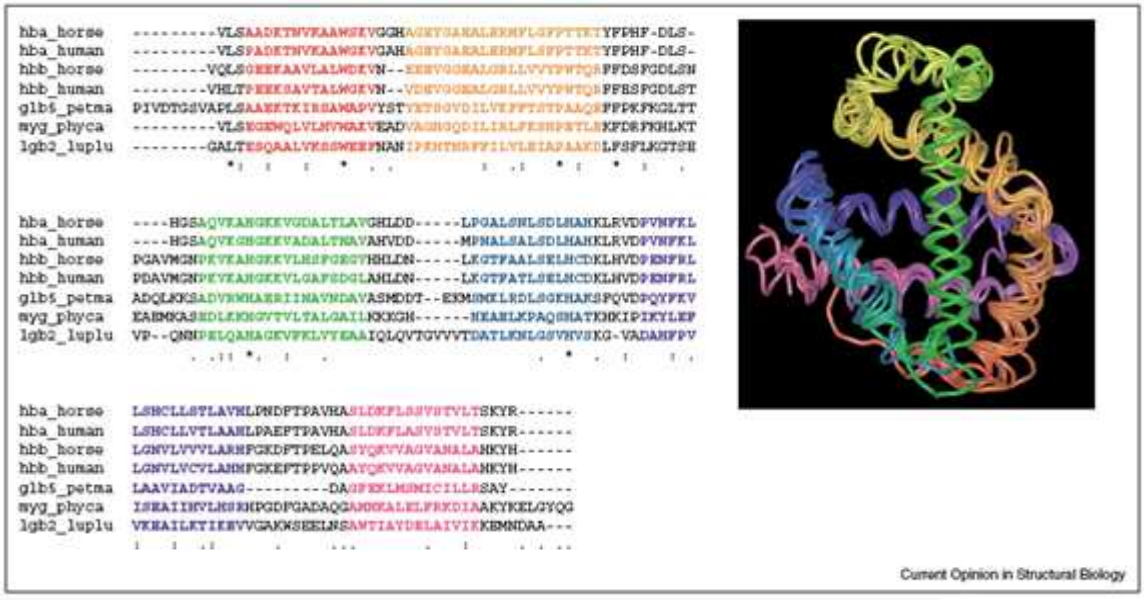
```
NLGPSTKDDFGKILGPSTKDDQ
         ||||
QNQLERSSNFGKINQLERSSNN
```

Μπορεί να φαίνεται ότι θα πρέπει πάντα κανείς να χρησιμοποιεί local alignment. Ωστόσο, μπορεί να είναι δύσκολο να εντοπιστεί μια συνολική ομοιότητα, σε αντίθεση με την domain-to-domain similarity, αν κάποιος χρησιμοποιεί μόνο local alignment. Έτσι η global alignment είναι χρήσιμη σε ορισμένες περιπτώσεις.

3. **Optimal alignment** : Για μια δεδομένη ομάδα ακολουθιών, δεν υπάρχει μόνο μία "σωστή" ευθυγράμμιση ("correct" alignment), μόνο η ευθυγράμμιση που είναι "βέλτιστη" ("optimal") σύμφωνα με ορισμένους υπολογισμούς. Ο καθορισμός για το ποια είναι η καλύτερη ευθυγράμμιση για ένα δεδομένο σύνολο ακολουθιών αφήνεται πραγματικά στην κρίση του ερευνητή.

Θεωρητικά, το να γίνει μια optimal alignment μεταξύ δύο ακολουθιών είναι υπολογιστικά απλό (**Smith-Waterman algorithm**) [14] , αλλά η ευθυγράμμιση ενός μεγάλου αριθμού ακολουθιών χρησιμοποιώντας την ίδια μέθοδο είναι σχεδόν αδύνατη. Το πρόβλημα αυξάνει εκθετικά με τον αριθμό των αλληλουχιών που συμμετέχουν για την εύρεση της MSA.

Ακολουθεί ένα παράδειγμα Πολλαπλής Ευθυγράμμισης Ακολουθιών στο οποίο φαίνεται το επίπεδο διατήρησης column by column χρησιμοποιώντας ClustalW notation ενός ‘*’ για completelely conserved column, ενός ‘:’ highly conserved column και ενός ‘.’ για less conserved column.



Σχήμα 1.1: Multiple Sequence Alignment

1.2 Γιατί να κάνουμε Multiple Sequence Alignments (MSAs)

Οι Multiple Sequence Alignments έχουν μεγάλη σημασία για τη βιολογική έρευνα. Επιπλέον, η ταχεία συσσώρευση DNA αλληλουχιών κατά τα τελευταία χρόνια έκανε την MSA απαραίτητο εργαλείο για την έρευνα.

Η MSA μπορεί να αποκαλύψει όμοια στοιχεία σε μια στήλη (conserved residues) που θα επιτρέπουν τον προσδιορισμό ενδεχόμενων σημαντικών περιοχών. Για παράδειγμα, conserved amino acid residues συνήθως εμπλέκονται σε πρωτεϊνική λειτουργία ή είναι υπεύθυνα για τη διαρθρωτική σταθερότητα των πρωτεϊνών. Σε αλληλουχίες DNA, conserved regions μπορούν να αντιπροσωπεύουν ένα ρυθμιστικό στοιχείο. Εκτός από τον εντοπισμό conserved residues, μια πιο σύνθετη προσέγγιση είναι να χρησιμοποιηθεί πληροφορία από την MSA χρησιμοποιώντας regions of residues με conserved properties για την κατασκευή ενός στατιστικού μοντέλου, όπως ένα Position Specific Scoring Matrix ή ίσως Hidden Markov Model. Τα μοντέλα αυτά χρησιμοποιούνται για τον εντοπισμό των conserved regions στα πρόσφατα sequenced γονιδιώματα, ή χρησιμοποιούνται για κατασκευή βάσεων δεδομένων, όπως PROSITE ή PFAM.

Η προσπάθεια για να κάνει κανείς sequencing ενός ολόκληρου γονιδιώματος είναι ένα δύσκολο έργο, ιδίως όταν λαμβάνονται υπόψη μεγάλα ευκαρυωτικά γονιδιώματα. Ωστόσο, μία από τις κύριες δυσκολίες είχε τεθεί από τη χρήση των MSAs. Αν και το

sequencing ενός μικρού τμήματος DNA είναι μια ρουτίνα για πολλά μοριακά εργαστήρια, είναι πρακτικά αδύνατο για μεγαλύτερες ακολουθίες. Η λύση είναι να κόψουμε την ακολουθία σε μικρά τυχαία τμήματα και να κάνουμε sequencing αυτών. Η ανασυγκρότηση ολόκληρου χρωμοσώματος γίνεται in silico, με την ευθυγράμμιση των τμημάτων και την εύρεση των overlaps.

Η κατασκευή των MSAs συνδέεται στενά με φυλογενετική ανάλυση. Ένα φυλογενετικό δέντρο μπορεί να συνταχθεί από μια MSA. Η μελέτη της μοριακής εξέλιξης είναι ένας τομέας όπου MSAs χρησιμοποιούνται σε μεγάλο βαθμό.

Μια άλλη πολύ σημαντική εφαρμογή των MSAs είναι η ενσωμάτωσή τους σε πολλές μεθόδους πρόβλεψης της δομής ή της λειτουργίας από ακολουθία. Αυτές οι μέθοδοι είναι μια σημαντική συμβολή της βιοπληροφορικής για την πειραματική έρευνα. Το ποσοστό των γνωστών ακολουθιών αυξάνεται, ενώ άλλες πληροφορίες όπως είναι η λειτουργία τους έχει καθυστερήσει. Πολλές μέθοδοι ενσωματώνουν πληροφορίες από τις MSAs για να βελτιώσουν τις προβλέψεις τους . Τέτοιες εφαρμογές είναι οι pairwise sequence alignments που κάνουν χρήση διαρθρωτικών δεδομένων, πρόσφατες μέθοδοι για την πρόβλεψη της δευτεροταγούς δομής πρωτεΐνης και της πρόβλεψης γονιδίου από τη σύγκριση των sequenced γονιδιωμάτων.

Κεφάλαιο 2

Σχετική Έρευνα

Στο κεφάλαιο αυτό παρουσιάζεται ένα σύνολο αλγορίθμων γενικής βελτιστοποίησης (general optimization algorithms) που χρησιμοποιούνται ευρέως στην επιστήμη υπολογιστών και έχουν εφαρμοστεί για την επίλυση του προβλήματος της Πολλαπλής Ευθυγράμμισης Ακολουθιών (Multiple Sequence Alignment problem), τόσο σε επίπεδο software όσο και σε hardware. Κάθε ένας από αυτούς τους αλγορίθμους επιχειρεί να δώσει μία γρήγορη και ακριβή λύση σε αυτό το πρόβλημα, καθένας με το δικό του τρόπο. Κριτήριο επιλογής του καταλληλότερου αλγορίθμου για να επιτευχθεί μια MSA μπορεί να θεωρηθεί η διαθεσιμότητα του αλγορίθμου, η ακρίβεια του, το πόσο εξοικειωμένος είναι ο χρήστης με το συγκεκριμένο αλγόριθμο και γενικά ποιον θεωρεί ο χρήστης καλύτερο για το πρόβλημα που θέλει να λύσει.

2.1 Μέθοδοι Multiple Sequence Alignment και Σχετικοί Αλγόριθμοι

Η Πολλαπλή Ευθυγράμμιση Ακολουθιών είναι υπολογιστικά δύσκολο να παραχθεί και οι περισσότερες εφαρμογές για την επίλυση του προβλήματος αυτού έχουν οδηγήσει σε NP-complete προβλήματα συνδυαστικής βελτιστοποίησης. Ωστόσο, η χρησιμότητα των εν λόγω alignments στην βιοπληροφορική οδήγησε στην ανάπτυξη διάφορων μεθόδων κατάλληλες για ευθυγράμμιση (aligning) τριών ή περισσότερων ακολουθιών.

Υπάρχουν τρεις μέθοδοι αλγορίθμων που σχετίζονται με την εύρεση της MSA και οι αλγόριθμοι που πρόκειται να αναπτυχθούν στην παρούσα ενότητα χρησιμοποιούν μία από τις μεθόδους αυτές προκειμένου να επιλύσουν το πρόβλημα της MSA [2].

- **Progressive:** ClustalW (1994), Muscle (2004)
- **Iterative:** MAFFT, Praline, IterAlign
- **Consistency Based:** Gotoh (1990), Martin Vingron (1991), Dialign (1996), T-Coffee (2000), Probcons (2004)

2.1.1 Ανάλυση Μεθόδων για τη δημιουργία Multiple Sequence Alignment

Progressive methods

Η Προοδευτική, επίσης γνωστή ως ιεραρχική ή μέθοδος δέντρου (tree method), δημιουργεί μια πολλαπλή ευθυγράμμιση ακολουθιών ευθυγραμμίζοντας πρώτα τις πιο παρόμοιες ακολουθίες και στη συνέχεια προσθέτοντας διαδοχικά τις λιγότερο συνδεόμενες αλληλουχίες ή ομάδες αλληλουχιών για την ευθυγράμμιση μέχρι ολόκληρο το query set να έχει ενσωματωθεί στη λύση. Το αρχικό δέντρο που περιγράφει την συγγένεια των ακολουθιών, βασίζεται σε pair-wise συγκρίσεις. Progressive τεχνικές συνεπώς κατασκευάζουν αυτόματα ένα φυλογενετικό δέντρο, καθώς και μια ευθυγράμμιση. Τα αποτελέσματα της Progressive Alignment εξαρτώνται από την επιλογή των "πιο σχετικών" ακολουθιών και ως εκ τούτου μπορεί να είναι ευαίσθητες σε τυχόν ανακρίβειες στις αρχικές pair-wise alignments. Επιπλέον οι περισσότερες Progressive Multiple Sequence Alignment μέθοδοι αποδίδουν ένα βάρος στις αλληλουχίες του query set σύμφωνα με τη συγγένεια τους, η οποία μειώνει την πιθανότητα να έχει γίνει μια κακή επιλογή των αρχικών ακολουθιών και ως εκ τούτου βελτιώνει την ακρίβεια της ευθυγράμμισης. Επειδή οι progressive μέθοδοι δεν εγγυώνται να συγκλίνουν σε μία global optimum alignment , η ποιότητα της ευθυγράμμισης μπορεί να είναι δύσκολο να αξιολογηθεί και η πραγματική τους βιολογική σημασία μπορεί να είναι ασαφής.

Iterative method

Η Επαναληπτική μέθοδος προσπαθεί να βελτιώσει το αδύνατο σημείο της προοδευτικής μεθόδου, τη μεγάλη εξάρτηση της ακρίβειας των αρχικών pair-wise alignments. Οι Επαναληπτικές μέθοδοι βελτιστοποιούν μία Objective Function που βασίζεται σε μία επιλεγμένη μέθοδο βαθμολόγησης μίας ευθυγράμμισης με την ανάθεση μιας πρώτης global ευθυγράμμισης και στη συνέχεια επαναπροσαρμόζοντας υποσύνολα ακολουθιών. Τα επανευθυγραμμισμένα υποσύνολα ευθυγραμμίζονται ώστε να παράγουν την MSA της επόμενης επανάληψης. Μπορούν να σχεδιαστούν διάφοροι τρόποι για την επιλογή των ακολουθιών- υποομάδων και της Objective Function

Consistency Based method

Ξεκινά με την αναγνώριση των τμημάτων που παρουσιάζουν υψηλή ομοιότητα ανά ζεύγη. Κάθε ζεύγος είναι σταθμισμένο με ένα βάρος και ένα δεύτερο score με βάση τη συμβατότητά του με το πλήρες σύνολο του τμήματος των ζευγών. Στη συνέχεια η MSA συναρμολογείται σταδιακά με την προσθήκη των ζευγαριών του τμήματος που εξετάζεται ανάλογα με το βάρος τους. Οι μέθοδοι αυτού του τύπου χρησιμοποιούν μια βάση δεδομένων με local high-scoring alignments και long-range global alignments για να δημιουργήσουν την τελική ευθυγράμμιση (alignment). Δεδομένου ενός συνόλου ακολουθιών η optimal MSA ορίζεται ως αυτή που συμφωνεί περισσότερο με όλες τις πιθανές optimal pair-wise alignments. Διακρίνονται ως πολύ γρήγορες και ακριβείς μέθοδοι για την εύρεση της MSA.

2.1.2 Multiple Sequence Alignment Αλγόριθμοι

CLUSTALW

Η ευαισθησία της πιο γνωστής progressive multiple sequence alignment μεθόδου έχει βελτιωθεί σε μεγάλο βαθμό για την ευθυγράμμιση διαφορετικών πρωτεϊνικών ακολουθιών .Πρώτον, συντελεστές στάθμισης ανατίθενται σε κάθε ακολουθία σε μία μερική ευθυγράμμιση με σκοπό να υπο-σταθμιστούν ακολουθίες σχεδόν αντίγραφα και να υπερ-σταθμιστούν οι περισσότερο αποκλίνουσες. Δεύτερον, πίνακες αντικατάστασης αμινοξέων (amino acid substitution matrices) ποικίλλουν σε διαφορετικά στάδια της ευθυγράμμισης

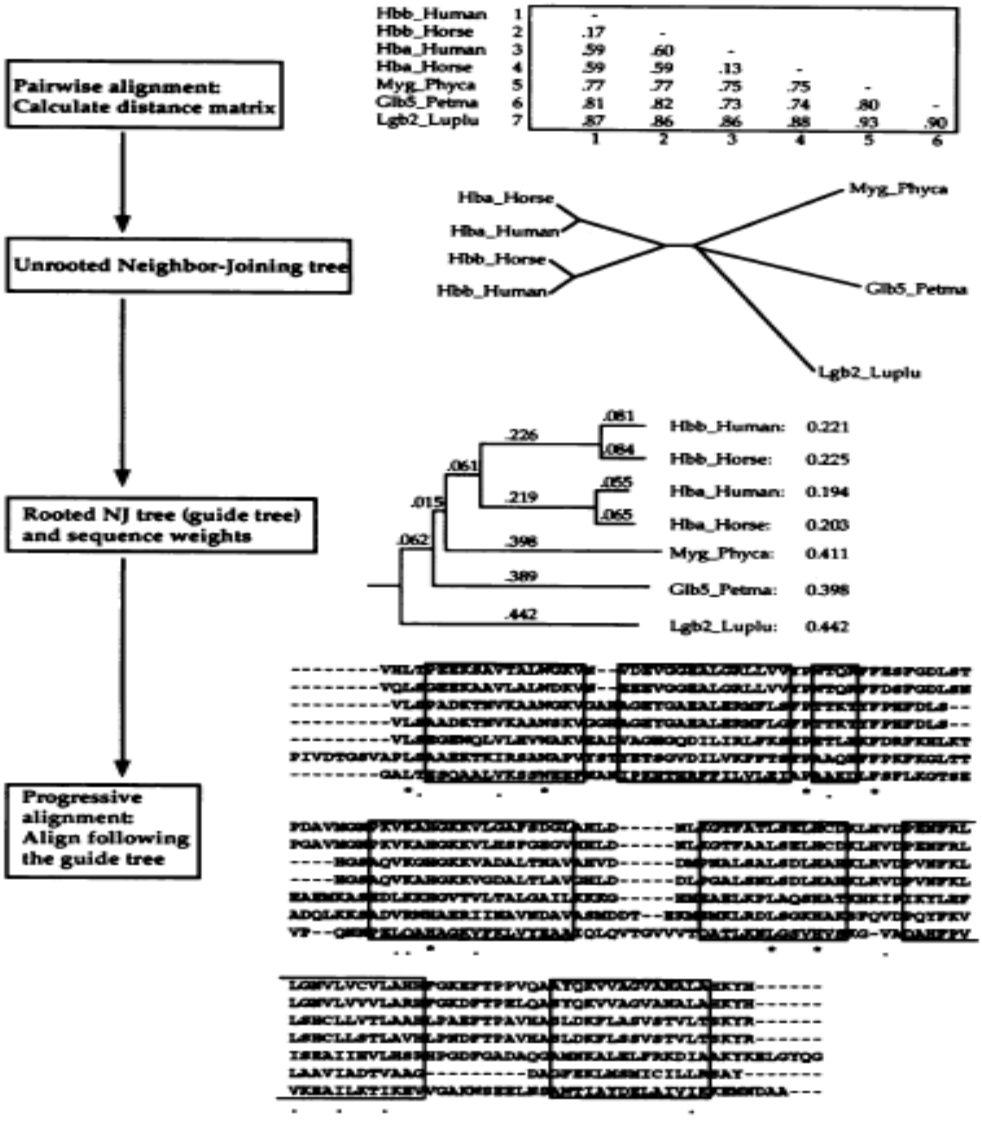
σύμφωνα με την απόκλιση των ακολουθιών που πρόκειται να ευθυγραμμιστούν. Τρίτον, residue-specific gap penalty και τοπικά μειωμένα gap-penalties στις υδρόφιλες περιοχές ενθαρρύνουν νέα κενά (gaps) σε ενδεχόμενες περιοχές βρόχων παρά στην κανονική δευτερεύουσα δομή. Τέταρτον, θέσεις σε αρχικές ευθυγραμμίσεις όπου έχουν τοποθετηθεί κενά, λαμβάνουν τοπικά μειωμένο gap-penalty για να ενθαρρύνουν την τοποθέτηση νέων κενών σε αυτές τις θέσεις. Αυτές οι τροποποιήσεις που έχουν ενσωματωθεί σε ένα νέο πρόγραμμα, CLUSTAL W [6] .

Ο αλγόριθμος ClustalW αποτελείται από 3 βήματα:

1) Δημιουργούνται pair-wise alignments μεταξύ όλων των ακολουθιών στο σύνολο που πρόκειται να ευθυγραμμιστεί. Alignment scores χρησιμοποιούνται για τη δημιουργία ενός πίνακα αποστάσεων
(distance matrix). Κατά τον υπολογισμό αυτού του πίνακα, ο αλγόριθμος λαμβάνει υπόψη την απόκλιση των ακολουθιών.

2) Δημιουργείται ένα φυλογενετικό δέντρο οδηγός (guide phylogenetic tree) χρησιμοποιώντας τη μέθοδο του κοντινότερου γείτονα (Neighbour-Joining method). Το δέντρο έχει παρακλάδια σε διάφορα μήκη. Το μήκος τους είναι ανάλογο στην εκτιμώμενη απόκλιση για το κάθε παρακλάδι.

3) Γίνεται προοδευτική ευθυγράμμιση (progressive alignment) των ακολουθιών, ακολουθώντας τη σειρά καθορίζει το δέντρο. Οι ακολουθίες ευθυγραμμίζονται από τα φύλλα προς τη ρίζα του δέντρου και βάση των φυλογενετικών σχέσεων που υποδεικνύονται από το δέντρο.



Σχήμα 2.1: Η βασική διαδικασία progressive alignment, χρησιμοποιώντας ένα σύνολο από 7 globins τριτογενούς δομής

Σε κάθε στάδιο της progressive alignment εφαρμόζεται δυναμικός προγραμματισμός χρησιμοποιώντας ένα scoring matrix. Ο αλγόριθμος υπολογίζει τα βάρη των ακολουθιών σύμφωνα με το δέντρο και επιλέγει το αντίστοιχο scoring matrix (σύμφωνα με την απόκλιση των συγκρινόμενων ακολουθιών).

Ο αλγόριθμος αυτός έχει ορισμένα μειονεκτήματα:

1) Δεν εγγυάται ότι θα βρεθεί η βέλτιστη global λύση.

2) Λάθη που μπορεί να συμβούν στα αρχικά στάδια της ευθυγράμμισης δεν μπορούν να διορθωθούν στη συνέχεια καθώς νέα πληροφορία επόμενων ακολουθιών προστίθεται.

3) Το αρχικό φυλογενετικό δέντρο προέρχεται από ένα πίνακα αποστάσεων μεταξύ χωριστά ευθυγραμμισμένων ζευγαριών ακολουθιών και είναι λιγότερο αξιόπιστο από τα δέντρα που προέρχονται από ολοκληρωμένη πολλαπλή ευθυγράμμιση.

4) Όταν υπάρχει μεγάλη απόκλιση μεταξύ των ακολουθιών (λιγότερο από 25-30 % ομοιότητα μεταξύ ζευγαριών) αυτή η progressive μέθοδος γίνεται λιγότερο αξιόπιστη.

MAFFT

Η δημιουργία του αλγορίθμου MAFFT [7] συνέβαλλε δραματικά στη μείωση του χρόνου της CPU συγκριτικά με άλλες μεθόδους. 1) Ομόλογες περιοχές αναγνωρίζονται ταχύτατα από τον γρήγορο Μετασχηματισμό Fourier (fast Fourier Transform) κατά τον οποίο μία ακολουθία αμινοξέων μετατρέπεται σε μια ακολουθία που αποτελείται από τον όγκο και τις τιμές πολικότητας κάθε αμινοξέος. 2) Προτείνεται ένα απλοποιημένο scoring σύστημα που αποδίδει καλά στη μείωση του χρόνου της CPU και αυξάνει την ακρίβεια ευθυγραμμίσεων τόσο για ακολουθίες με πολλές εισαγωγές ή προεκτάσεις όσο και για ακολουθίες που είναι σχετικές σε μικρό βαθμό παρόμοιου μεγέθους . Δύο τεχνικές, progressive method (FFT-NS-2) και iterative refinement method (FFT-NS-1), υλοποιούνται στον αλγόριθμο MAFFT. Οι τεχνικές αυτές συγκρίθηκαν με άλλες μεθόδους σε computer simulations και benchmark tests. Ο χρόνος CPU για τον FFT-NS-2 μειώθηκε δραματικά σε σύγκριση με τον ClustalW με συγκρίσιμη ακρίβεια. Ο FFT-NS-1 είναι 100 φορές πιο γρήγορος από τον T-Coffee, όταν ο αριθμός των ακολουθιών ξεπερνά τις 60, χωρίς να θυσιάζεται η ακρίβεια.

Χαρακτηριστικά αλγορίθμου MAFFT:

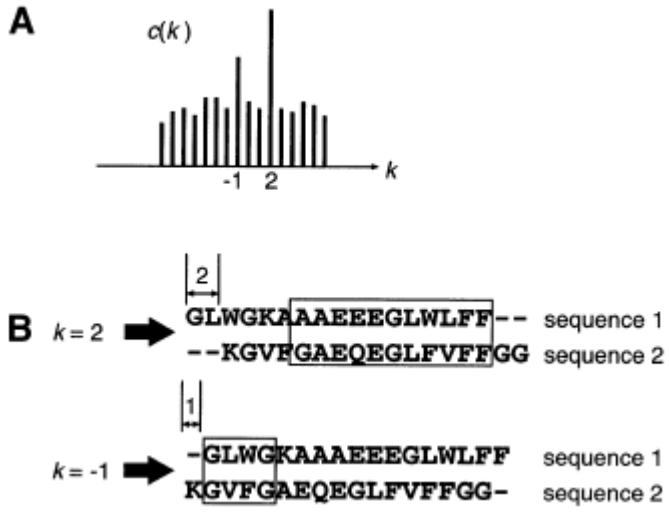
1) Χρησιμοποιεί Fast Fourier Transform να επιταχύνει profile alignments

2) Χρησιμοποιεί γρήγορη μέθοδο δύο σταδίων για ευθυγραμμίσεις κτιρίου χρησιμοποιώντας συχνότητες k-mer

3) Προσφέρει πολλές διαφορετικές τεχνικές βαθμολόγησης και ευθυγράμμισης

4) Ένα από τα πιο ακριβή προγράμματα διαθέσιμα

5) Πολλές μορφές εξόδου συμπεριλαμβανομένων των διαδραστικών φυλογενετικών δέντρων



Σχήμα 2.2: (Α) Αποτέλεσμα FFT ανάλυσης, (Β) Διενεργείται συρόμενη ανάλυση παραθύρου και προσδιορίζονται οι θέσεις των ομόλογων blocks

DIALIGN

Ο αλγόριθμος DIALIGN [8] είναι μια νέα μέθοδος τόσο για pairwise όσο και για multiple sequence alignment νουκλεϊκών και πρωτεϊνικών ακολουθιών. Ενώ τα τυποποιημένα προγράμματα ευθυγράμμισης βασίζονται στη σύγκριση μόνο των residues και στην επιβολή gap penalties, ο αλγόριθμος DIALIGN κατασκευάζει ευθυγραμμίσεις συγκρίνοντας ολόκληρα τμήματα των ακολουθιών. Δεν υπάρχουν gap penalties. Αυτή η άποψη είναι ιδιαίτερα κατάλληλη, εφόσον οι αλληλουχίες δεν συνδέονται globally, αλλά εμφανίζουν μόνο local ομοιότητες , όπως συμβαίνει σε γονιδιακό DNA και σε πολλές οικογένειες πρωτεϊνών. Με τέσσερα διαφορετικά σύνολα δεδομένων, αποδεικνύεται ότι ο αλγόριθμος DIALIGN είναι σε θέση να ευθυγραμμίσει σωστά διατηρούμενα μοτίβα σε πρωτεϊνικές ακολουθίες. Ευθυγραμμίσεις που παράχθηκαν από DIALIGN έχουν συγκριθεί συστηματικά με τα αποτελέσματα πέντε άλλων προγραμμάτων ευθυγράμμισης .

Ενδεικτικά για τον αλγόριθμο Dialign:

1) Ευθυγράμμιση ολόκληρων τμημάτων και όχι μεμονωμένων αμινοξέων (βάσεων).

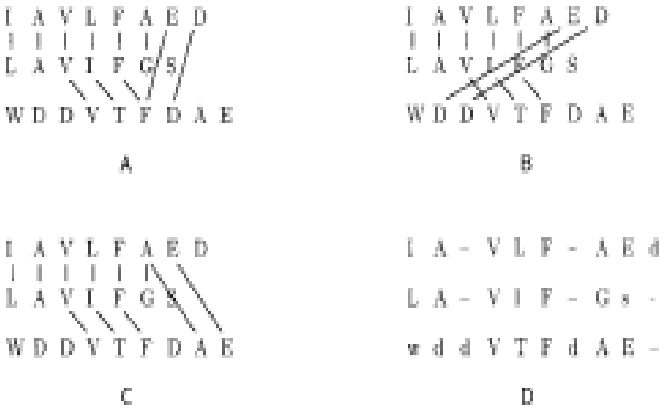
2) Pairwise συγκρίσεις > τμήματα ζευγαριών (segment pairs) (διαγώνιες) , εκπροσωπεί local alignments.

3) Οι διαγώνιοι σταθμίζονται για την εύρεση πιθανότητας.

4) Η ευθυγράμμιση δημιουργήθηκε από συνεπής διαγώνιες .

5) Δεν υπάρχουν gap penalties.

6) Δεν εξαρτάται από την σειρά των ακολουθιών



Σχήμα 2.3: (A) και (B) αντιπροσωπεύουν μη συνεπή συλλογές των διαγωνίων, Στο (A), το ' F ' στην τρίτη σειρά έχει εκχωρηθεί ταυτόχρονα σε δύο διαφορετικά residues της πρώτης σειράς. Στο (B), υπάρχει μία «cross-over» εκχώρηση των residues. Αντιθέτως, (C) είναι συνεπής συλλογή των διαγωνίων. Είναι δυνατόν να εισαχθούν κενά σε τέτοιες αλληλουχίες που τα residues οποία συνδέονται με τις διαγώνιες είναι στην ίδια στήλη της προκύπτουσας ευθυγράμμισης (D). Residues που δεν συμμετέχουν σε καμία από τις τρεις διαγώνιες είναι τυπωμένα με πεζά γράμματα.Δεν θεωρείται ότι θα ευθυγραμμιστούν.

2.2 Multiple Sequence Alignment χρησιμοποιώντας FPGA΄s

Επειδή η Πολλαπλή Ευθυγράμμιση Ακολουθιών είναι ένα θεμελιώδες πρόβλημα και αποτελεί πρόκληση στην υπολογιστική μοριακή βιολογία, ενδιαφέρον στην επιστημονική κοινότητα προκάλεσε η ανάπτυξη εφαρμογών MSA, πέρα από το software , σε αναδιατασσόμενη λογική με σκοπό την σύγκριση αυτών καθώς για πολύ μεγάλα datasets με εκατοντάδες ακολουθίες οι εφαρμογές MSA σε software τείνουν να γίνουν πολύ αργές .Ακολουθεί σύντομη περιγραφή ανάπτυξης των αλγορίθμων ClustalW, Dialign καθώς και του T-coffee, που αποτελεί και το θέμα της παρούσας διπλωματικής εργασίας, σε αναδιατασσόμενη λογική.

ClustalW

Για τον αλγόριθμο του ClustalW που θεωρείται το πιο ευρέως χρησιμοποιούμενο λογισμικό για MSA αναφέρονται δύο υλοποιήσεις σε αναδιατασσόμενη λογική.

- Η μία υλοποίηση [9] περιγράφει το σχεδιασμό υλικού σε Verilog. Χρησιμοποιώντας μια Xilinx Virtex II XC2V6000, είναι σε θέση να φιλοξενήσει 92 PEs σε μέγιστη επιτρεπόμενη ταχύτητα ρολογιού των 34 MHz. Η συγκεκριμένη εφαρμογή επιτυγχάνει μια σταθερή απόδοση (συμπεριλαμβανομένης της μεταφοράς όλων των δεδομένων) της ~ 1 GCUPS (1δις ενημερώσεις κυττάρων ανά δευτερόλεπτο), ένα μέτρο που χρησιμοποιείται συνήθως για τη σύγκριση parallelized Smith-Waterman υλοποιήσεις. Μια σειρά δοκιμών αξιολόγησης των επιδόσεων έχουν διεξαχθεί με τη χρήση διαφορετικού αριθμού globin ακολουθιών για την αξιολόγηση του χρόνου επεξεργασίας των FPGA-accelerated implementations έναντι στον ακολουθιακό κώδικα ClustalW. Κατά τις δοκιμές χρησιμοποιείται PCI βασισμένο σε ADP-WRC-II board από Alpha-Data με μία Xilinx XC2V6000 FPGA. Η εφαρμογή ClustalW είναι υποδειγματική σε Intel Pentium IV 3 GHz επεξεργαστή με 1GB RAM. Η FPGA-accelerated pairwise alignment επιτυγχάνει speedups μεταξύ 45 και 50. Τουλάχιστον ο ίδιος αριθμός των υπολογιστών που συνδέονται με ένα γρήγορο switch απαιτείται για να επιτευχθεί ένα παρόμοιο speedup χρησιμοποιώντας το ClustalW-MPI code from Li (2003). Μια σύγκριση αυτών των δύο προσεγγίσεις παραλληλοποιήσεων δείχνει ότι αναδιατασσόμενη επιτάχυνση (reconfigurable hardware accelaration) είναι ανώτερη από άποψη τιμής / απόδοσης. Η συγκεκριμένη λύση είναι επίσης εύκολα

επεκτάσιμη σε FPGAs επόμενης γενιάς, όπως σε οικογένεια Virtex-4 με την απλή αύξηση του αριθμού των PEs στο συγκεκριμένο σχεδιασμό.

- Η δεύτερη εφαρμογή του αλγορίθμου ClustalW σε αναδιατασσόμενη λογική [10]είχε ως αναφορά ένα Xeon 2.8GHz επεξεργαστή με 4 GB RAM. Η FPGA-accelerated pairwise alignment επιτυγχάνει speedups περίπου 1600. Το μέγιστο συνολικό speedup είναι περίπου 34. Τουλάχιστον ο ίδιος αριθμός των υπολογιστών που συνδέονται με ένα γρήγορο switch απαιτείται για να επιτευχθεί ένα παρόμοιο speedup χρησιμοποιώντας το ClustalW-MPI . Χρησιμοποιούμε τα ίδια στοιχεία δοκιμών για την αξιολόγηση του παράλληλου κώδικα ClustalWMPI στο σύστημα Dawning 4000H. Το σύστημα αυτό ένα ειδικό μηχάνημα για βιοπληροφορικής. Εκτός από τους γενικούς επεξεργαστές, περιλαμβάνει δέκα ειδικούς επιταχυντές για sequence alignment αλγορίθμους. Επιπλέον περιέχει 40 υπολογιστικούς κόμβοι (computing nodes) και 5 κόμβους διακομιστή (server node). Οι κόμβοι υπολογιστών είναι blades, καθένας από τους οποίους περιέχει Dual Xeon 2.8GHz επεξεργαστή και 2GB RAM. Συνδέονται μεταξύ τους με ένα δίκτυο ενός GB.

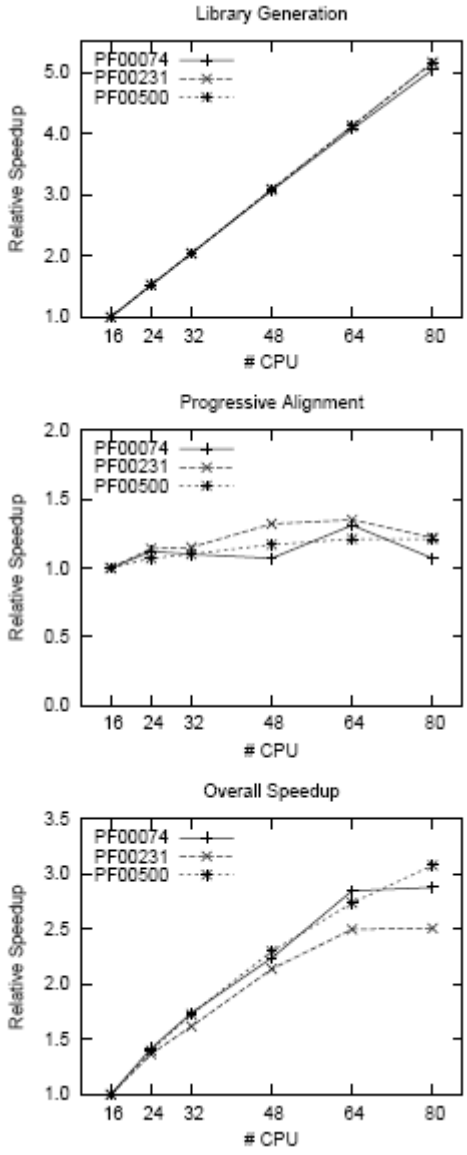
DIALIGN

Η αρχιτεκτονική αυτή σχεδιάστηκε για να επιταχύνει τα pair-wise βήματα του αλγορίθμου DIALIGN [11] που είναι το πιο υπολογιστικά βαρύ μέρος αυτού του αλγορίθμου για MSA. Η προτεινόμενη αρχιτεκτονική για αυτόν τον αλγόριθμο έχει σχεδιαστεί για να χειριστεί μεγάλες ακολουθίες χωρίζοντας το dataset σε blocks των 200 ακολουθιών. ΄Έγινε επιτυχημένα σύνθεση στο εργαλείο Altera FPGA STRATIX 2 EP2S180F1508I4. Για αποτελέσματα πραγματικών αλληλουχιών DNA των μεγεθών 121 Kbp και 169 Kbp, υπήρξε speedup των 383,41 ενάντια σε μια βελτιστοποιημένη εφαρμογή C, αποδεικνύοντας ότι μπορεί να είναι πολύ χρήσιμο να επιταχυνθεί το πρόβλημα της πολλαπλής ευθυγράμμισης ακολουθιών. Τα speedups επιτεύχθηκαν με 3 πολύ διαφορετικά μεγέθη ακολουθιών που ήταν μεταξύ 343 και 383 και αποδείχθηκε ότι τα speedups που επιτεύχθηκαν δεν εξαρτώνται από το μέγεθος των ακολουθιών.

T-COFFEE

Για να επικυρωθεί το λογισμικό πραγματοποιήθηκε μια σειρά πειραμάτων με πρωτεϊνικά δεδομένα από τη βάση δεδομένων Pfam [20]. Τα πειράματα που πραγματοποιήθηκαν σε σύμπλεγμα (cluster) που αποτελείται από Dual Intel Xeon 3GHz κόμβους. Κάθε κόμβος είναι εξοπλισμένος με 2GB μνήμης RAM, και τρέχει Linux. Το

σύμπλεγμα συνδέεται από ένα δίκτυο FastEthernet. Σε πειράματα μας χρησιμοποιείται η mpich2-1.0.4p1, μία γνωστή και αποτελεσματική εφαρμογή του προτύπου MPI. Το λογισμικό αυτό συντάχθηκε με το GCC-3.4.4 σύνολο των μεταγλωττιστών. Τα πειράματα εκτελέστηκαν σε 16 με 80 επεξεργαστές. Σε κάθε πείραμα, κάθε επεξεργαστής θα μπορούσε να χρησιμοποιήσει 768MB της κύριας μνήμη ως cache storage. Ο σκοπός του έργου αυτού είναι να παρέχει σταθερή και αποτελεσματική εφαρμογή του αλγορίθμου T-Coffee ότι να διατηρηθεί η λειτουργικότητα του αρχικού λογισμικού και να ξεπεράσει τους κύριους περιορισμούς. Η εφαρμογή αυτή έχει δοκιμαστεί σε όλες τις σημαντικές πλατφόρμες, συμπεριλαμβανομένων των συστημάτων 64-bit.



Σχήμα 2.4: Σχετικό speedup Parallel T-Coffee, υπολογισμένο σε σχέση με το χρόνο εκτέλεσης για 16 CPU

Κεφάλαιο 3

Αλγόριθμος T-Coffee και Μοντελοποίηση

Στο κεφάλαιο αυτό περιγράφονται αναλυτικά τα βασικά βήματα του αλγορίθμου T-Coffee καθώς επίσης και των συναρτήσεων που υλοποιήθηκαν.

3.1 Αλγόριθμος T-Coffee

Ο αλγόριθμος T-Coffee (**T**ree-based **C**onsistensy **O**bjective **F**unction **F**or alignm**E**nt **E**valuation) [1] είναι μια προσπάθεια να ελαχιστοποιήσει την επίδραση της απληστίας (greedy algorithms) από την οποία υποφέρουν οι περισσότερες μέθοδοι που σχετίζονται με MSAs, παρά το γεγονός ότι η στρατηγική που προτείνεται είναι επίσης μια άπληστη προοδευτική μέθοδο, που επιτρέπει την πολύ καλύτερη χρήση της πληροφορίας από τα πρώτα κιόλας στάδια.

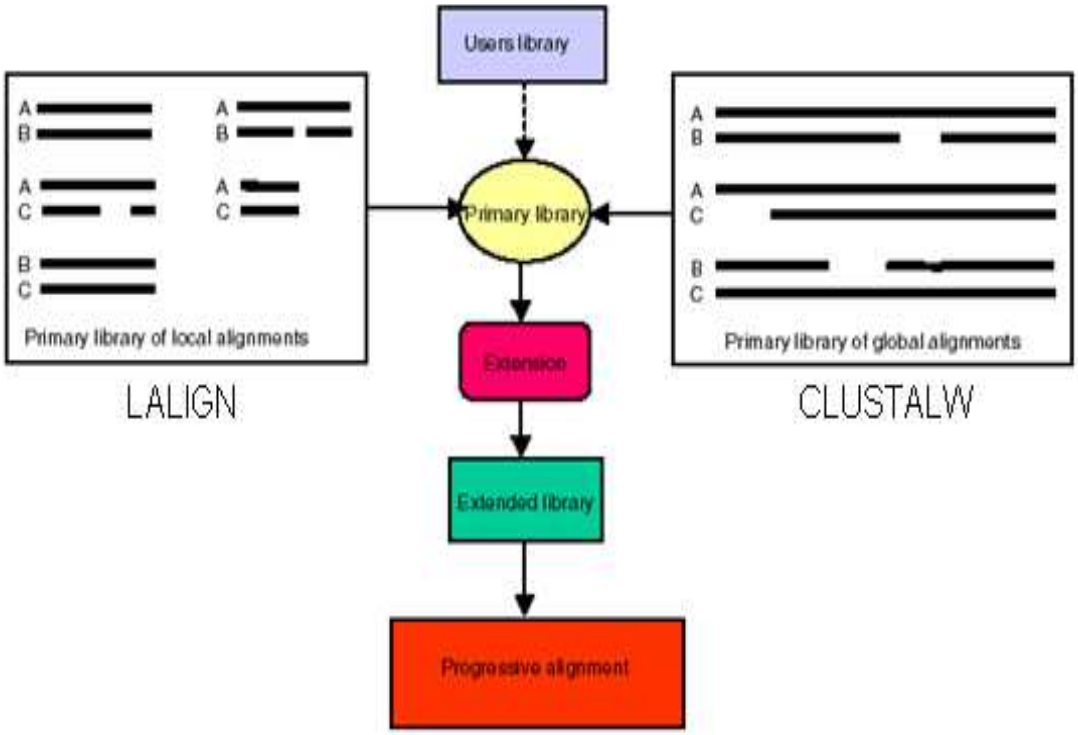
Η κύρια εναλλακτική λύση για progressive alignment είναι η ταυτόχρονη ευθυγράμμιση (simultaneous alignment) όλων των ακολουθιών και αυτό επιτυγχάνεται από μεθόδους που επιχειρούν να πραγματοποιήσουν global ή local alignments.Μία μέθοδος που να μπορεί να συνδυάζει τις καλύτερες ιδιότητες από global και local alignment μπορεί να είναι πολύ ισχυρή. Αυτό είναι το δεύτερο κίνητρο του αλγορίθμου T-Coffee: ο σχεδιασμός μιας μεθόδου που παρέχει μια απλή, ελαστική και, το πιο σημαντικό, ακριβή λύση του προβλήματος του πώς να συνδυαστούν πληροφορίες αυτού του είδους.

Ο αλγόριθμος T-Coffee έχει δύο κύρια χαρακτηριστικά. Πρώτον, παρέχει ένα απλό και ελαστικό μέσο για τη δημιουργία multiple alignments, χρησιμοποιώντας ετερογενείς πηγές δεδομένων. Τα δεδομένα από αυτές τις πηγές παρέχονται στον αλγόριθμο T-Coffee, μέσω μίας βιβλιοθήκης από pair-wise alignments. Εδώ αποδεικνύεται η δύναμη του αλγορίθμου T-Coffee υπολογίζοντας multiple alignments χρησιμοποιώντας μια βιβλιοθήκη που δημιουργήθηκε με τη χρήση μείξης global και local pair-wise alignments.

Η βιβλιοθήκη μπορεί να είναι ελαστική γιατί μπορεί να δημιουργηθεί χρησιμοποιώντας διαφορετικές πηγές πληροφορίας:

i. Global και 10 καλύτερες local non-intersecting pair-wise alignments
ii. Διαρθρωτική (structural) πληροφορία σχετικά με aligned πρωτεΐνες
iii. Δεδομένα που έχουν δημιουργηθεί από άλλα λογισμικά (other MSA software)

Το δεύτερο χαρακτηριστικό του αλγορίθμου T-Coffee είναι η μέθοδος βελτιστοποίησης (optimization method), η οποία χρησιμοποιείται για την εύρεση της multiple alignment που ταιριάζει καλύτερα με τις pair-wise alignments στην βιβλιοθήκης η οποία αποτελεί είσοδο στον αλγόριθμο. Χρησιμοποιείται η λεγόμενη προοδευτική στρατηγική (progressive strategy) (**Feng & Doolittle, 1987;Taylor, 1988; Thompson et al., 1994**) [13] που είναι παρόμοια με εκείνη που χρησιμοποιείται για τον αλγόριθμο ClustalW (**JulieD.Thomson**) [6]. Αυτό έχει το πλεονέκτημα να είναι σχετικά γρήγορη και ισχυρή. Με τον αλγόριθμο T-Coffee, ωστόσο, γίνεται χρήση των πληροφοριών στη βιβλιοθήκη για την πραγματοποίηση progressive alignment κατά τέτοιο τρόπο που να μας επιτρέπει να εξετάσουμε τις ευθυγραμμίσεις (alignments) μεταξύ όλων των ζευγών ενώ εκτελούμε κάθε βήμα της progressive multiple alignment. Αυτό έχει ως αποτέλεσμα progressive alignment, με όλα τα πλεονεκτήματα της ταχύτητας και της απλότητας, αλλά με πολύ μικρότερη τάση να γίνουν λάθη στα πρώτα στάδια της MSA. Ο αλγόριθμος T-Coffee είναι μία μέθοδος για progressive alignment με την ικανότητα του να εξετάζει πληροφορίες από ολόκληρο το σύνολο των ακολουθιών κατά τη διάρκεια κάθε σταδίου για την ολοκλήρωση της ευθυγράμμισης , και όχι μόνο των ακολουθιών που πρόκειται να ευθυγραμμιστούν σε εκείνο το στάδιο.



Σχήμα 3.1: T-coffee's Layout

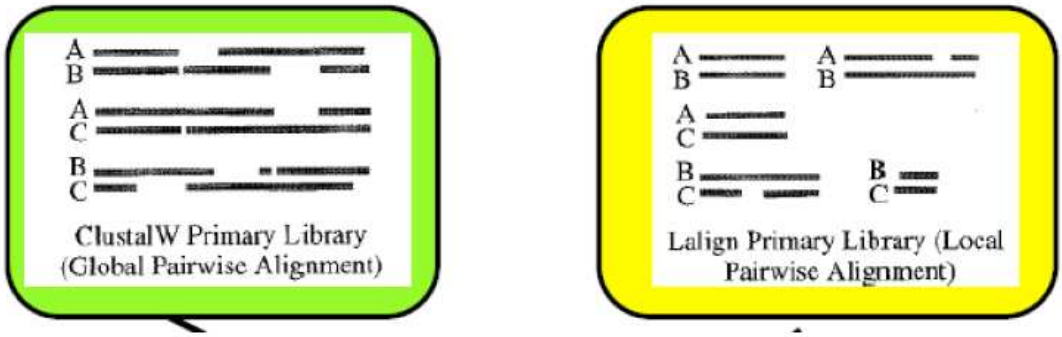
3.2 Περιγραφή Αλγορίθμου T-Coffee

Ο αλγόριθμος T-Coffee αποτελείται από τρία βήματα :

1ο βήμα: Generating a Primary Library of alignments

Η Primary Library περιλαμβάνει ένα σύνολο από pair-wise alignments μεταξύ όλων των ακολουθιών που πρέπει να εναρμονιστούν. Στη βιβλιοθήκη, θα περιλαμβάνεται πληροφορία για κάθε ένα από τα N(N-1)/2 ζεύγη ακολουθιών, όπου N είναι ο αριθμός των ακολουθιών. Χρησιμοποιούνται δύο πηγές για να γίνει alignment για κάθε ζεύγος ακολουθιών, ένα local και ένα global. Οι global alignments προκύπτουν χρησιμοποιώντας τον αλγόριθμο ClustalW για τις ακολουθίες, δύο κάθε φορά. Αυτό χρησιμοποιείται για να δώσει μία full-length alignment μεταξύ κάθε ζεύγους ακολουθιών. Οι local alignments είναι οι δέκα topscoring non-intersecting local alignments , μεταξύ κάθε ζεύγους ακολουθιών, που

συγκεντρώθηκαν με τη χρήση του προγράμματος Lalign του πακέτου FASTA με τις προεπιλεγμένες παραμέτρους.

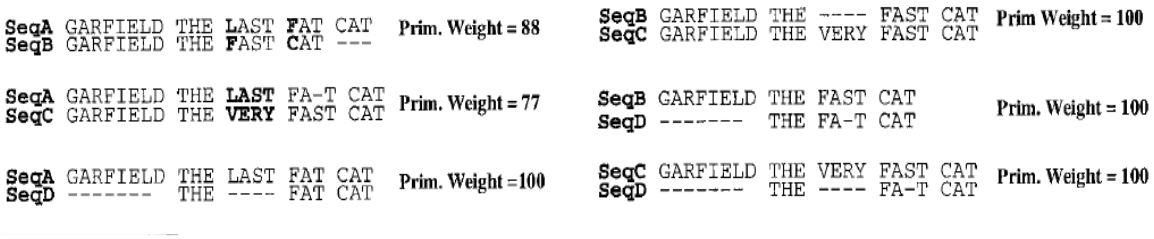


Σχήμα 3.2: Global και Local Pairwise Alignments

Στη βιβλιοθήκη, κάθε ευθυγράμμιση εκπροσωπείται ως μία λίστα από pair-wise residue matches (e.g. residue x of sequence A is aligned with residue y of sequence B). Κάθε ένα από αυτά τα ζεύγη είναι ένα constraint. Δεν είναι όλα τα constaints εξίσου σημαντικά . Μερικά μπορεί να προέρχονται από τμήματα ευθυγραμμίσεων που έχουν περισσότερες πιθανότητες να είναι σωστά.

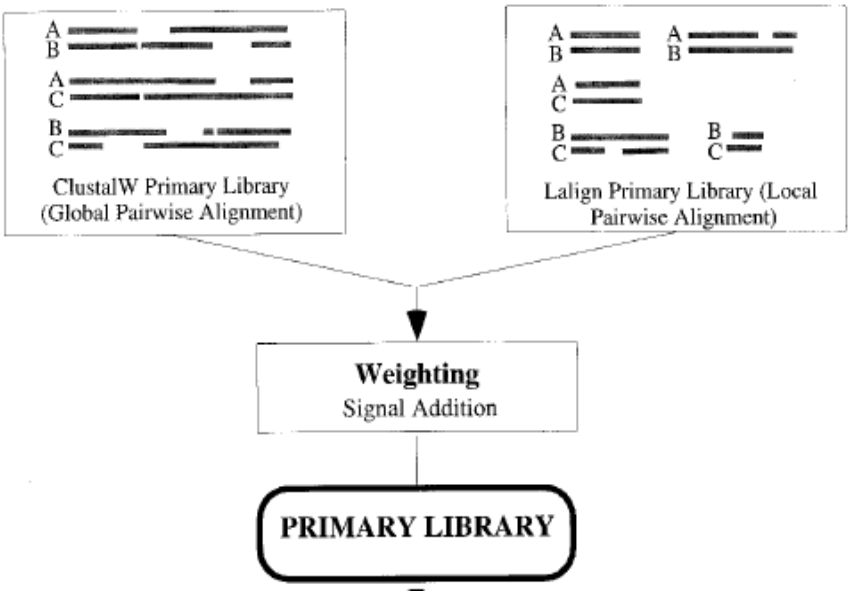
Αυτό θα ληφθεί υπόψη κατά τον υπολογισμό της πολλαπλής ευθυγράμμισης και θα δοθεί προτεραιότητα στια πλέον αξιόπιστα residue pairs. Αυτό είναι επιτευχθεί με τη χρήση ενός συστήματος στάθμισης (weighting scheme).

Ο αλγόριθμος αναθέτει ένα βάρος σε κάθε ζεύγος aligned residues στη βιβλιοθήκη. Ένα ιδανικό primary βάρος θα χαρακτηρίσει την ορθότητα ενός constraint. Χρησιμοποιείται sequence identity, η οποία είναι γνωστή να αποτελεί εύλογη ένδειξη της ακρίβειας, όταν ευθυγραμμίζονται ακολουθίες με περισσότερο από 30% ομοιότητα (**Sander & Schneider, 1991**) [5].Αυτό το σύστημα στάθμισης (weighting scheme) έχει αποδειχθεί πολύ αποτελεσματική για προηγούμενη consistency-based objective function (**Notredame et al.,1998**) [3]. Επίσης έχει το μεγάλο πλεονέκτημα της απλότητας. Οι βιβλιοθήκες είναι λίστες weighted pair-wise constraints. Κάθε constraint δέχεται βάρος ίσο προς το επί τοις εκατό ποσοστό ομοιότητας ως προς το pair-wise alignment από το οποίο προέρχεται. Για κάθε σύνολο ακολουθιών, δύο primary libraries υπολογίζονται μαζί με τα βάρη, μία χρησιμοποιώντας ClustalW (global alignments) και μία δεύτερη χρησιμοποιώντας Lalign (local alignments).



Σχήμα 3.3: Primary Library

Στόχος είναι ο αποτελεσματικός συνδυασμός των local και global alignments.Αυτό επιτυγχάνεται συγκεντρώνοντας τις ClustalW και Lalign βιβλιοθήκες σε μια απλή διαδικασία της προσθήκης. Αν κάποιο ζεύγος είναι διπλό μεταξύ των δύο βιβλιοθηκών, τότε αυτό συγχωνεύεται σε μία ενιαία καταχώριση που έχει βάρος ίσο με το άθροισμα των δύο βαρών (global και local). Σε αντίθετη περίπτωση, μια νέα καταχώρηση δημιουργείται για το ζεύγος που θα εξεταστεί. Residue pairs που δεν πραγματοποιήθηκαν δεν εκπροσωπούνται (by default they will be considered to have a weight of zero).



Σχήμα 3.4: Δημιουργία Primary Library

Η primary library μπορεί να χρησιμοποιηθεί άμεσα για τον υπολογισμό της MSA . Θα μπορούσε να βρεθεί μια ευθυγράμμιση που ταιριάζει όσο το δυνατόν καλύτερα με τα weighted pairs
των residues. Ωστόσο, αυξάνεται σε πολύ μεγάλο βαθμό η αξία της πληροφορίας στη βιβλιοθήκη εξετάζοντας τη συνοχή από κάθε pair of residues με residue pairs από όλες τις

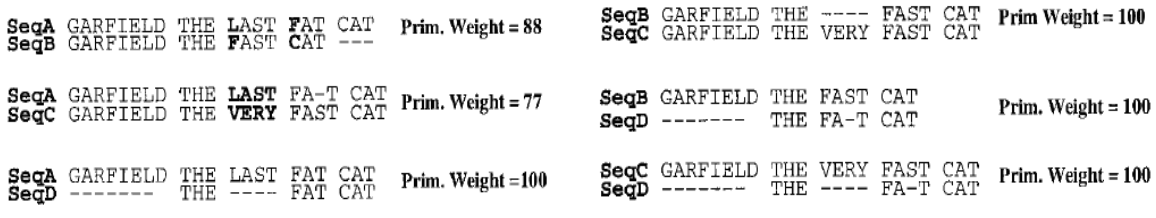
ευθυγραμμίσεις. Για κάθε ζεύγους ευθυγραμμισμένων residues στη βιβλιοθήκη, μπορεί να ανατεθεί ένα βάρος που αντικατοπτρίζει το βαθμό κατά τον οποίο αυτά τα residues ευθυγραμμίζονται με συνοχή με τα residues από όλες τις ευθυγραμμίσεις. Η διαδικασία αυτή ονομάζεται library extension.

2ο βήμα: Extending the Primary Library

Η γενική ιδέα είναι να συνδυαστεί πληροφορία κατά τέτοιο τρόπο ώστε το τελικό βάρος για κάθε pair of residues, να αντικατοπτρίζει ορισμένη από την πληροφορία που περιέχεται στη βιβλιοθήκη. Για να γίνει αυτό, μία προσέγγιση τριπλέτας (triplet aproach) χρησιμοποιείται.

Η στρατηγική έχει κάποιες ομοιότητες με τη γενική ιδέα των overlapping weights που αναπτύσσονται στο Dialign2 (**Morgenstern, 1999**) [15] ή τη μέθοδο των intermediate sequences (**Neuwald et al. (1997)**) [16] για την αναζήτηση βάσεων δεδομένων. Βασίζεται στη λήψη κάθε aligned residue pair της βιβλιοθήκης και τον έλεγχο της ευθυγράμμισης των δύο residues με τα residues από τις υπόλοιπες ακολουθίες .

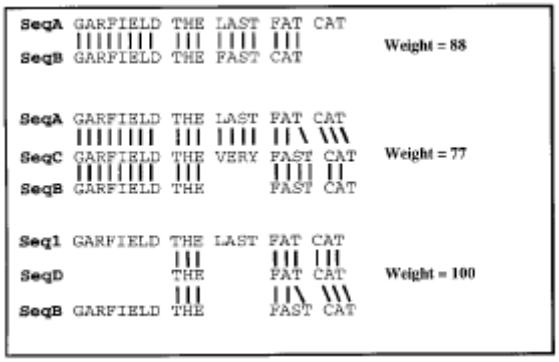
Για παράδειγμα, ας θεωρήσουμε ξανά τις τέσσερις σειρές A, B, C και D του παρακάτω σχήματος . Ας ονομάσουμε A (G) το G του Garfield στην ακολουθία A, B (G), το αντίστοιχο G στην ακολουθία B και W (A (Z), B (Z)) το βάρος που σχετίζεται με ζεύγος αυτών των residues στην primary βιβλιοθήκη. Στο alignment των A και B, μεταξύ A (G) και B (G) υπάρχει match. Συνεπώς, το αρχικό βάρος, για αυτό το ζευγάρι residues μπορεί να οριστεί σε 88 (primary weight την ευθυγράμμιση των ακολουθιών A και B, που είναι το επί τοις εκατό ποσοστό ομοιότητας (percent identity) αυτού του ζεύγους).



Primary Library

Αν τώρα εξετάσουμε την ευθυγράμμιση της ακολουθίας A με την ακολουθία B μέσω της ακολουθίας C, μπορούμε να δούμε ότι η A(G) και C(G) έχουν ευθυγραμμιστεί, όπως και C(G) και B(G). Συμπεραίνουμε ότι υπάρχει alignment της A(G) με την B(G) μέσω της ακολουθίας C. Συνδέουμε την ευθυγράμμιση με ένα βάρος ίσο προς το ελάχιστο του

W1(A(G), C(G)) και W2 (C(G), B(G)). Αφού W1=77 και W2=100, το βάρος που προκύπτει είναι ίσο με 77. Στην extended library, η νέα αυτή τιμή προστίθεται στον προηγούμενο βάρος για να δώσει το συνολικό βάρος των 165 (δηλαδή 77 Β 88) για το ζεύγος A(G), B(G).



Σχήμα 3.5: Extended Library

Η πλήρης επέκταση θα απαιτήσει την εξέταση όλων των υπόλοιπων τριπλετών. Δεν φέρουν όλες οι τριπλέτες πληροφορία. Για παράδειγμα, η ευθυγράμμιση των Α και Β μέσω της ακολουθίας D δεν περιέχει καμία πληροφορία σε σχέση με το A(G) ή B(G), και, κατά συνέπεια, δεν έχει καμία επίπτωση στο βάρος που συνδέει τα A(G) και B(G). Συνοπτικά, το βάρος που συνδέεται με ένα ζευγάρι residues θα είναι το άθροισμα όλων των βαρών που συγκεντρώνονται μέσω της εξέτασης όλων των τριπλετών που φέρουν πληροφορία για το συγκεκριμένο ζεύγος.

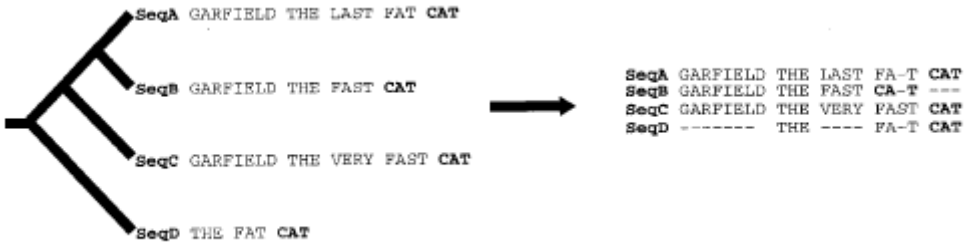
Όσο περισσότερες ενδιάμεσες ακολουθίες υποστηρίζουν την ευθυγράμμιση του εν λόγω ζεύγους, τόσο μεγαλύτερο θα είναι και το βάρος του. Extension θα διεξαχθεί σε κάθε ζεύγος των residue Α και Β. Όταν η διαδικασία έχει ολοκληρωθεί, το ζευγάρι των ακολουθιών Α και Β θα έχουν συγκεντρώσει πληροφορία από όλες τις άλλες ακολουθίες που συμμετέχουν στην MSA και βρίσκοντα στη βιβλιοθήκη.. Αυτό το σενάριο είναι επαναλαμβάνεται για κάθε ζεύγος (AC, AD, BC, BD, CD) από τις υπόλοιπες ακολουθίες. Το πλήρες σύνολο ζευγών αποτελεί την extended library.

Ζεύγη residues που δεν έχουν συμβεί ποτέ έχουν μηδενικά βάρη (αυτό θα ισχύει για την πλειοψηφία των pair residues). Σε αντίθετη περίπτωση, το βάρος θα αντικατοπτρίζει ένα συνδυασμό της ομοιότητας μεταξύ των ζευγών ακολουθιών ή των τμημάτων ακολουθιών όπου το residue pair προέρχεται από τη συνοχή του εν λόγω residue pair με όλα τα άλλα residue pairs στην extended library. Αυτά τα scores μπορούν να χρησιμοποιηθούν για την ευθυγράμμιση κάθε δύο ακολουθιών από το data set με τη χρήση συμβατικού δυναμικού προγραμματισμού (**Gotoh, 1982**) [4].

Όταν κανονικά γίνεται ευθυγράμμιση σε ένα ζευγάρι ακολουθιών, γίνεται χρήση ενός συνόλου scores που προέρχονται από ορισμένους γενικούς πίνακες από amino acid weights όπως Blosum matrix (**Henikoff & Henikoff, 1992**) [17]. Στην περίπτωση του αλγορίθμου T-Coffee, μπορεί να αντικατασταθεί αυτός ο πίνακας με ένα σύνολο από scores που είναι συγκεκριμένα για κάθε δυνατό pair of residues στις δύο ακολουθίες. Αυτό θα επιτρέψει την ευθυγράμμιση που θα πραγματοποιηθεί να λάβει υπόψη ειδικά residues, αλλά και να κατευθυνθεί ως προς τη συνοχή με όλες τις άλλες ακολουθίες στο dataset. Αυτό είναι μία ισχυρή ικανότητα και μπορεί να χρησιμοποιηθεί για την πραγματοποίηση της προοδευτικής ευθυγράμμισης (progressive alignment) αποφεύγοντας πολλές από τα local-minimum προβλήματα συνήθως που συνδέονται με την εν λόγω τεχνική.

3ο βήμα: Progressive alignment strategy

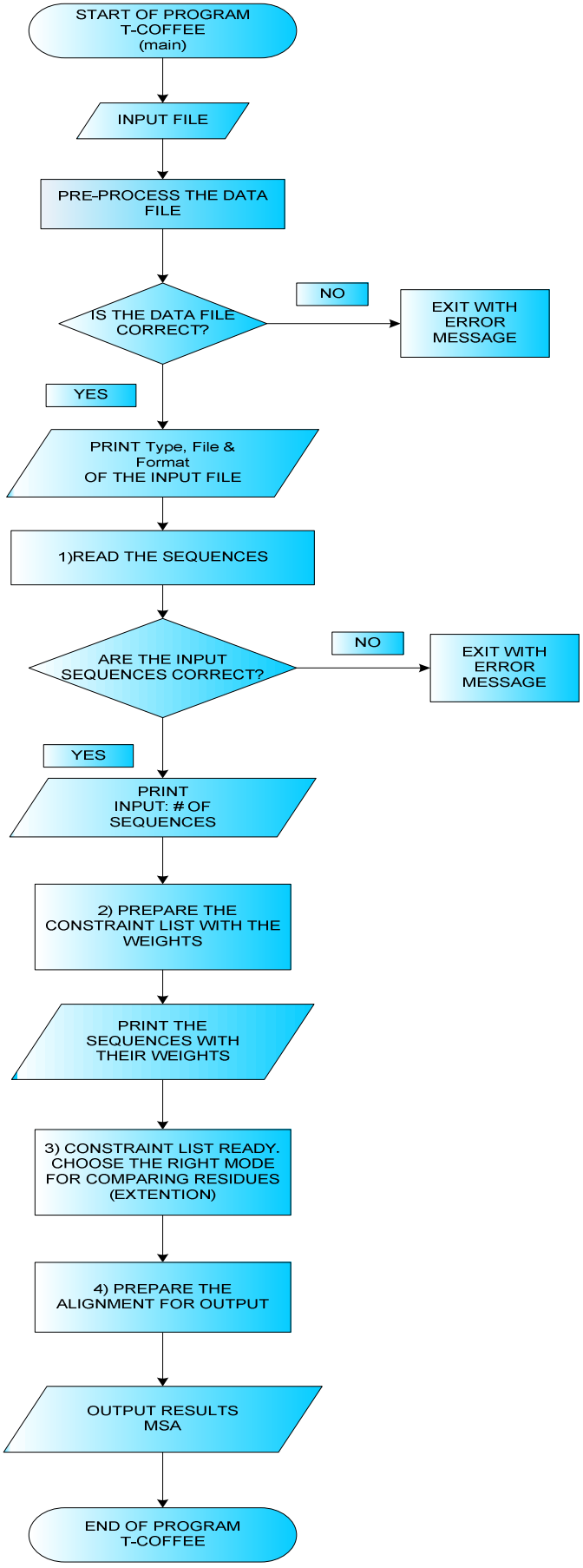
Κατά την progressive alignment (**Thompson et al., 1994**) [6], σε πρώτη φάση δημιουργούντα pair-wise alignments για την παραγωγή ενός πίνακα αποστάσεων (distance matrix) ανάμεσα σε όλες τις ακολουθίες, η οποία με τη σειρά του χρησιμοποιείται για την παραγωγή ενός guide-tree με τη χρήση της μεθόδου neighbor-joining (**Saitou & NEI, 1987**) [18]. Αυτό είναι ένα φυλογενετικό δένδρο, το οποίο χρησιμοποιείται για την άμεση ομαδοποίηση των ακολουθιών κατά τη διάρκεια της MSA. Οι κοντινότερες δύο αλληλουχίες στο δέντρο ευθυγραμμίζονται πρώτες με τη χρήση δυναμικού προγραμματισμού. Αυτή η ευθυγράμμιση χρησιμοποιεί τα βάρη που προήλθαν από την extended library για την ευθυγράμμιση των residues σε δύο ακολουθίες. Αυτό το ζεύγος ακολουθιών “τακτοποιείται” και τα κενά (gaps) που έχουν έχει εισαχθεί δεν μπορούν να μετατεθούν αργότερα. Στη συνέχεια ευθυγραμμίζονται οι δύο επόμενες πλησιέστερες ακολουθίες ή προστίθεται μία ακολουθία στην υπάρχουσα ευθυγράμμιση των δύο πρώτων ακολουθιών, ανάλογα με το τι προτείνεται από τον οδηγό δέντρο (guide-tree). Οι επόμενες δύο πλησιέστερες ακολουθίες της προ-ευθυγραμμισμένης ομάδας ακολουθιών είναι πάντα συνδεδεμένες. Αυτό συνεχίζεται μέχρι όλες οι ακολουθίες να ευθυγραμμισθούν. Για την ευθυγράμμιση των δύο ομάδων προ-ευθυγραμμισμένων ακολουθιών χρησιμοποιείται το score που προήλθε από την extended library, όπως και πριν, αλλά λαμβάνεται ο μέσος όρος από τα scores στην βιβλιοθήκη για κάθε στήλη της υφιστάμενης ευθυγράμμισης .



Σχήμα 3.6: Regular Progressive Alignment Strategy

Όπως χρησιμοποιείται εδώ, η διαδικασία δεν απαιτεί τυχόν πρόσθετες παραμέτρους όπως gap penalties. Αυτό οφείλεται, εν μέρει, στο γεγονός ότι οι τιμές προς αντικατάσταση (substitution values(the library weights)) είχαν υπολογιστεί σε ευθυγραμμίσεις που τέτοιου είδους penalties είχαν ήδη εφαρμοστεί. Πρακτικά, αυτό σημαίνει ότι κατά τη διάρκεια της προοδευτικής φάσης (progressive phase) χρησιμοποιείται ένας αλγόριθμος δυναμικού-προγραμματισμού (**Gotoh, 1982**) [4] με τα gap-opening και gap-extension penalties να τίθενται στο μηδέν για την ευθυγράμμιση δύο αλληλουχιών ή δύο ομάδων προ-ευθυγραμμισμένων ακολουθιών.

Ακολουθεί το flow chart του αλγορίθμου T-Coffee .



Σχήμα 3.7: T-coffee's Flow Chart

3.3 Μοντελοποίηση Αλγορίθμου T-Coffee

Στην ενότητα αυτή θα αναλυθεί το μέρος του software του αλγορίθμου που κρίθηκε σημαντικό να υλοποιηθεί σε hardware και θα διευκρινιστούν οι λόγοι που αυτό επιλέχθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας .

3.3.1 Ανάλυση Software Αλγορίθμου T-Coffee

Ο αλγόριθμος T-Coffee δεν μπορεί να τρέξει σε λειτουργικό Microsoft Windows. Τρέχει σε UNIX ή LINUX command line εισάγοντας τις σωστές παραμέτρους . Για την ανάλυση των επιδόσεων του αλγορίθμου T-Coffee και για την εκτίμηση των κομματιών του αλγορίθμου που θεωρούνται υπολογιστικά βαριά χρησιμοποιήθηκε η εφαρμογή Intel VTune Performance Analyzer for Linux. Μετά την εισαγωγή διαφόρων datasets , οι συναρτήσεις του αλγορίθμου που παρουσίασαν ενδιαφέρον ως προς τις επιδόσεις τους και κατά συνέπεια ως προς την υλοποίηση τους σε hardware είναι η residue_pair_extended_list και η fast_get_dp_cost.

Να σημειωθεί ότι ο συγκεκριμένος αλγόριθμος χρησιμοποιείται τόσο για την ευθυγράμμιση πρωτεϊνικών όσο και νουκλεοτιδικών ακολουθιών , αν και είναι καταλληλότερος για πρωτεϊνικές αλληλουχίες . Για το λόγο αυτό και τα αρχεία εισόδου περιέχουν μόνο πρωτεϊνικές αλληλουχίες. Επίσης χρειάζεται να επισημανθεί ότι μεγάλα datasets μπορεί να δημιουργήσουν πρόβλημα στη λειτουργία του αλγορίθμου , διότι είναι δύσκολο να αναλυθούν. Το πρόβλημα είναι ότι όταν υπάρχουν πάρα πολλές ακολουθίες , MSA προγράμματα τείνουν να γίνουν πολύ αργά και ανακριβή. Επιπλέον μεγάλα datasets είναι δύσκολο να εμφανιστούν και να αναλυθούν. Με λίγα λόγια , το καλύτερο μέγεθος για ένα σύνολο δεδομένων MSA είναι μεταξύ 20 και 40 ακολουθίες. Ο αλγόριθμος T-Coffee περιορίζεται σε ανώτατο όριο 50 ακολουθιών (sequences) και μήκος 2000 χαρακτήρων (residues). Πάνω από αυτό τον αριθμό , το πρόγραμμα αλλάζει αυτόματα σε μια άλλη λειτουργία (heuristic mode), με την επωνυμία DPA, όπου DPA σημαίνει Double Progressive Alignment.

Παρατίθεται ο πίνακας των αποτελεσμάτων από τη χρήση του VTune Performance Analyzer .

Input	residue_pair_extended_list (clockticks %)	fast_get_dp_cost (clockticks %)
Input1: 1amk_ref1	16.35 %	-
Input2: 1idy_ref2	52.83 %	14.49 %
Input3: 1wit_ref2	41.61 %	18.55 %
Input4: 1ag8_ref4	48.01 %	16.08 %
Input5:2scw_1drw	35.71 %	39.96 %

3.3.2 Μοντελοποίηση Συνάρτησης residue_pair_extended_list

Η συνάρτηση αυτή δέχεται ως εισόδους 4 ακέραιες τιμές s1, s2, r1, r2 που αντιστοιχούν σε 2 αριθμούς ακολουθιών (sequences) και 2 στοιχεία (residues) των ακολουθιών αυτών που θέλουμε να επεξεργαστούμε για την εξαγωγή ενός τελικού score, βάσει ενός συγκεκριμένου αρχείου εισόδου (Σχήμα 3.8). Επιπλέον δέχεται ως είσοδο κάποιες σταθερές , τις normalize και max_ext_value .

Για την εξαγωγή αυτού του τελικού score η διαδικασία που εκτελείται από τη συνάρτηση είναι η εξής :

Υπάρχει ένα τρισδιάστατου lookup table (Σχήμα 3.9 : s1 s2 r1 r2 weight) , όπου κρατείται πληροφορία για την σχέση όλων των ακολουθιών και όλων των στοιχείων αυτών, δηλαδή τα matches, ανάλογα με το αρχείο εισόδου , και μέσω του οποίου γίνεται ο έλεγχος της ευθυγράμμισης των δύο residues που έχουμε εισάγει με τα residues από τις υπόλοιπες ακολουθίες .

Match σημαίνει ότι τα ζεύγη sequence - residue (s1 - r1 , s2 - r2) που έχουμε εισάγει υπάρχουν στο lookup table είτε απευθείας μέσω της εισόδου (direct match), είτε μέσω ενός άλλου ζευγαριού sequence - residue (indirect match) . Το ζευγάρι αυτό μέσω του οποίου ψάχνουμε να βρούμε match μεταξύ των ζευγαριών που έχουμε εισάγει καλείται target_sequence (t_s) - target_residue (t_r). Σε κάθε match αντιστοιχεί ένα βάρος (weight). Αν παρατηρήσουμε ότι υπάρχει ευθυγράμμιση μεταξύ s1(r1) και t_s (t_r) όπως επίσης και μεταξύ s2(r2) και t_s (t_r) ,τότε το βάρος που προκύπτει θα είναι ίσο προς το ελάχιστο του W1(s1(r1), t_s (t_r)) και W2 (s2(r2), t_s (t_r)).

Το βάρος που συνδέεται με το ζευγάρι residues που έχουμε εισάγει θα είναι το άθροισμα όλων των βαρών που συγκεντρώνονται μέσω της εξέτασης με το t_s - t_r που φέρουν πληροφορία για το συγκεκριμένο ζεύγος. Το άθροισμα όλων αυτών των βαρών αποτελεί το score που προκύπτει από τα matches που υπάρχουν μεταξύ της εισόδου s1, s2, r1, r2 και του τρισδιάστατου lookup table. Για την εξαγωγή του τελικού αποτελέσματος , αν η

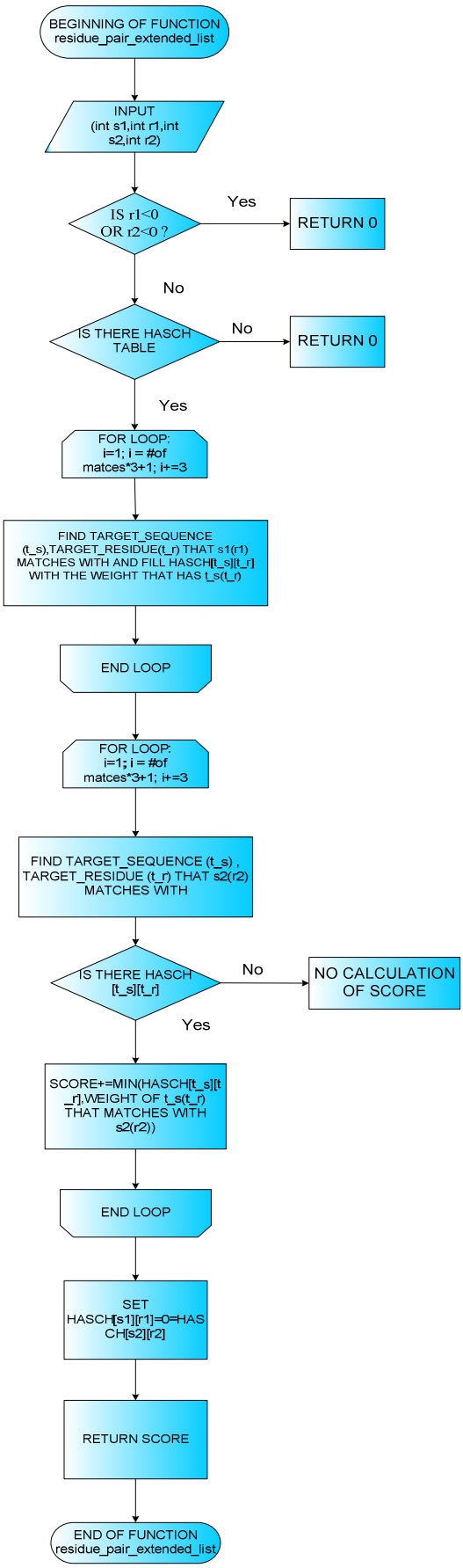
σταθερά εισόδου normalize φέρει θετική τιμή , το score που βρέθηκε πολλαπλασιάζεται με τη σταθερά εισόδου normalize και το γινόμενο αυτό διαιρείται στη συνέχεια με τη σταθερά εισόδου max_ext_value. Το πηλίκο αυτό αποτελεί το τελικό score. Αν η σταθερά normalize δεν φέρει θετική τιμή τότε ως τελικός score ορίζεται το score που προέκυψε από το άθροισμα των βαρών.(Σχήμα 3.10 : flow chart)

```
>hmgl_trybr
kkdsnapkramtsfmffssdfrskhsdlsivemskaagaawkelgpeerk
vyeemaekdkerykrem
>hmgt_mouse
kpkrprsayniyvsesfqeakddsaqgklklvneawknlspeekqayiql
akddrirydnemksweeqmae
>hmgb_chite
adkpkrplsaymlwlnsaresikrenpdfkvtevakkggelwrglkdkse
weakaatakqnyiralqeyerngg
>hmgl_wheat
dpnkpkrapsaffvfmgefreefkqknpknksvaavgkaagerwkslses
ekapyvakanklkgeynkaiaaynkgesa
```

Σχήμα 3.8: Ενδεικτική μορφή αρχείου εισόδου FASTA στο σύστημα. Με το σύμβολο ‘ > ’ δηλώνεται το όνομα της κάθε ακολουθίας (sequence) και ακολουθούν τα στοιχεία (residues) αυτής.

```
0    1    2    1    27
0    1    3    2    27
0    1    4    3    27
0    1    5    4    27

          .
          .
          .

0    2    1    1    19
0    2    2    2    19
0    2    3    3    19
0    2    4    4    19

          .
          .
          .

3    4    41   53   12
3    4    42   54   12
3    4    43   55   12
3    4    44   56   12
3    4    45   57   12
```

Σχήμα 3.9: Ενδεικτική μορφή αρχείου τρισδιάστατου lookup table



Σχήμα 3.10 : residue_pair_extended_list's Flow Chart

3.3.3 Μοντελοποίηση Συνάρτησης fast_get_dp_cost

Η συνάρτηση αυτή δέχεται ως εισόδους δύο ακέραιες τιμές ns1 και ns2 που αντιστοιχούν στον αριθμό ακολουθιών που υπάρχουν σε δύο λίστες (list1 και list2) τις οποίες επεξεργάζεται και οι οποίες περιέχουν ζευγάρια sequence - residue. Επιπλέον δέχεται ως είσοδο κάποιες σταθερές, όπως και η προηγούμενη συνάρτηση, τις normalize, max_ext_value και score_K . Η συνάρτηση αυτή υπολογίζει επίσης ένα τελικό score βάσει των στοιχείων που υπάρχουν στις δύο λίστες . Οι δύο λίστες μπορεί να μην περιέχουν τον ίδιο αριθμό στοιχείων. Ο μέγιστος αριθμός ακολουθιών που μπορεί να υπάρχει σε μία λίστα είναι 25, δηλαδή η μεγαλύτερη τιμή που μπορεί να δοθεί στις μεταβλητές ns1 και ns2 είναι ίση με 25.

Σαρώνοντας τη λίστα αυτή βρίσκουμε όλα τα matches που υπάρχουν μεταξύ των στοιχείων αυτής και του τρισδιάστατου lookup table όπως αναφέρθηκε και στην περιγραφή της συνάρτησης residue_pair_extended_list. Προφανώς γίνεται έλεγχος για την ύπαρξη match μεταξύ κάθε στοιχείου της λίστας με όλες τις ακολουθίες που υπάρχουν στο αρχείο εισόδου. Ομοίως και εδώ η κάθε ακολουθία που ελέγχεται για την ύπαρξη match με το εκάστοτε στοιχείο της λίστας καλείται t_s (target_sequence) και το στοιχείο αυτής με το οποίο θα υπάρξει match καλείται t_r (target_residue).

Μόλις βρεθεί το πρώτο ζεύγος t_s (t_r) , δηλαδή το πρώτο match του $1^{ου}$ στοιχείου της λίστας, γίνεται έλεγχος αν υπάρχει match μεταξύ του t_s (t_r) που βρέθηκε και του πρώτου στοιχείου της δεύτερης λίστας. Στη συνέχεια ελέγχεται παράλληλα αν υπάρχει match μεταξύ του t_s (t_r) και του $2^{ου}$ στοιχείου της λίστας που επεξεργαζόμαστε καθώς επίσης και μεταξύ του t_s (t_r) και του $2^{ου}$ στοιχείου της δεύτερης λίστας , κ.ο.κ. Η διαδικασία αυτή συνεχίζεται μέχρι και τα τελευταία στοιχεία των δύο λιστών.

Μόλις ολοκληρωθεί η διαδικασία αυτή για το $1^{ο}$ t_s (t_r) του $1^{ου}$ στοιχείου της list2 που βρέθηκε, ακολουθείται η ίδια διαδικασία για το $2^{ο}$ match που υπάρχει μεταξύ του $1^{ου}$ στοιχείου της list2 και του look - table.

¨Όταν ολοκληρωθεί η εύρεση όλων των matches του $1^{ου}$ στοιχείου της list2 τότε προχωράμε στην εύρεση των matches του επόμενου στοιχείου αυτής της λίστας και παράλληλα ελέγχονται και τα στοιχεία της δεύτερης λίστας .

Αν βρεθεί match τότε κρατείται το αντίστοιχο βάρος από το lookup table. Matches που βρίσκονται μέσω του ίδιου t_s (t_r) έχουν ως αποτέλεσμα το άθροισμα των αντίστοιχων βαρών. Αυτό σε ότι αφορά τη μία λίστα. Για την δεύτερη λίστα κρατείται διαφορετικό άθροισμα ακόμη και αν αυτό προκύπτει για το ίδιο t_s (t_r).

Μετά τον έλεγχο ενός t_s (t_r) με όλα τα στοιχεία των list2 και list1 και πριν την εύρεση νέου υπολογίζεται ένα score ανάλογα με κάποιες προϋποθέσεις και τέλος αφού ολοκληρωθεί η αναζήτηση των matches για όλα τα στοιχεία της list2 βρίσκουμε το τελικό score, στον υπολογισμό του οποίου συμβάλουν το score που υπολογίζεται για ένα συγκεκριμένο t_s (t_r) καθώς επίσης και οι σταθερές που εισάγονται ως είσοδοι στο συνολικό σύστημα.

Κεφάλαιο 4

Αρχιτεκτονική Σχεδίασης

Στο κεφάλαιο αυτό περιγράφεται και αναλύεται η σχεδίαση και η υλοποίηση των δύο συναρτήσεων residue_pair_extended_list και fast_get_dp_cost του αλγορίθμου που θεωρούνται υπολογιστικά οι πιο βαριές. Η υλοποίηση της αρχιτεκτονικής σχεδίασης έγινε στο εργαλείο XILINX ISE Design Suite 10.1 Virtex-4 FPGA Family Member Device XC4VFX140.

4.1 Περιγραφή της σχεδίασης και της υλοποίησης της συνάρτησης residue_pair_extended_list

Το lookup table που χρησιμοποιεί η συγκεκριμένη συνάρτηση, σε επίπεδο αρχιτεκτονικής αναπαραστάθηκε με δύο μνήμες BRAM,συγκεκριμένα δύο dual port ROM , την address_matrix και τη residue_matrix αντίστοιχα οι οποίες αρχικοποιούνται offload και όχι κατά τη διάρκεια της εκτέλεσης ανάλογα το αρχείο εισόδου και το look - table που προκύπτει από αυτό.

Η μνήμη residue_matrix περιέχει τα residues των ακολουθιών, δηλαδή την $3^{η}$ και $4^{η}$ στήλη του αρχείου που αντιστοιχεί στο lookup table (Σχήμα 3.9). Το μεγαλύτερο dataset που μπορεί να υποστηριχτεί από τη συγκεκριμένη σχεδίαση δέχεται αρχείο εισόδου με αριθμό ακολουθιών ίσο με 50 και μέγιστο αριθμό residues ανά ακολουθία ίσο με 1000. Το μέγεθος του lookup table καθορίζεται από τη φύση των δεδομένων. Για το λόγο η μνήμη residue_matrix δεν έχει συγκεκριμένο μέγεθος αλλά διαφέρει ανάλογα το αρχείο εισόδου. Το πλάτος της μνήμης επιλέχθηκε ίσο με 20 γιατί έχει οριστεί από τη σχεδίαση ότι ο μέγιστος αριθμός residues που μπορεί να υπάρχει σε μία ακολουθία είναι 1000 , οπότε χρειάζονται 10 bits για την αναπαράσταση του κάθε residue.

Η μνήμη address_matrix περιέχει τον αριθμό των matches μεταξύ 2 ακολουθιών του αρχείου ανάλογα το αρχείο του lookup table, δηλαδή τις διευθύνσεις start - end , όπου

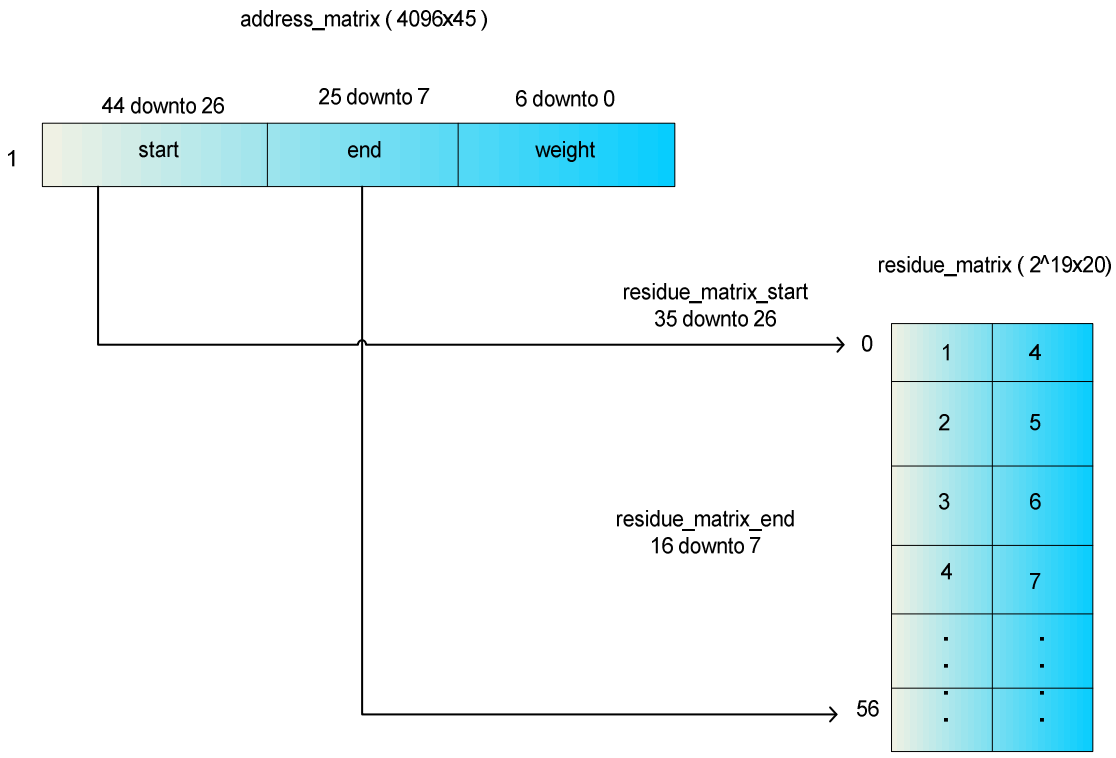
ξεκινάνε και τελειώνουν τα matches μεταξύ δύο ακολουθιών καθώς επίσης και το βάρος τους. Αν θεωρήσουμε το lookup table που παρατίθεται (Σχήμα 3.9) , η μνήμη address_matrix περιέχει τις διευθύνσεις 0 (start) έως 56 (end) για τις ακολουθίες 0 και 1 που αντιστοιχούν στις 2 πρώτες στήλες καθώς επίσης και το αντίστοιχο βάρος που βρίσκεται στην $5^{η}$ στήλη, 57 (start) έως 104 (end) για τις ακολουθίες 0 και 2 με το αντίστοιχο βάρος , κ.ο.κ. Για προφανείς λόγους δεν παρατίθεται ολόκληρο το αρχείο του look - up table αλλά μέρος αυτού. Η μνήμη address_matrix έχει διαστάσεις [4096 * 45] , καθώς οι δυνατοί συνδυασμοί των 50 ακολουθιών είναι 50x50 = 2500 και τα περιεχόμενα αυτής είναι ουσιαστικά 3 πεδία , που αναπαριστώνται από 19 bits τα 2 πρώτα πεδία και 7 bits το $3^{ο}$ πεδίο (Σχήμα 4.1).

Η διευθυνσιοδότηση των μνημών γίνεται μέσω μίας μονάδας ελέγχου , η οποία είναι μια μηχανή πεπερασμένων καταστάσεων (control - fsm) , ως εξής :

Για κάθε ζευγάρι εισόδου s1, s2, r1, r2 βρίσκουμε τη σχέση s1 και s2. Αν δηλαδή έχουμε είσοδο το ζεύγος ακολουθιών 0 - 1, τότε η διεύθυνση του address_matrix για την αποθήκευση των διευθύνσεων start - end των matches αυτών των δύο ακολουθιών είναι s1 x 64 + s2, δηλαδή 0 x 64 + 1 = 1, δηλαδή η διεύθυνση 1 του address_matrix περιέχει τις διευθύνσεις του look - up table όπου ξεκινούν και τελειώνουν τα matches των ακολουθιών 0 και 1 .

Επειδή όπως προαναφέρθηκε ο μέγιστος αριθμός ακολουθιών που μπορεί να υπάρξει είναι 50 (από 0 έως 49) και να επιτευχθεί η σωστή διευθυνσιοδότηση του address_matrix ανάλογα με το ζευγάρι ακολουθιών που θα εισάγουμε ως είσοδο. Επομένως οι διευθύνσεις του address_matrix θα είναι από 1 (για s0-s1) έως 3.121(για s48-s49).

Η διευθυνσιοδότηση του residue_matrix γίνεται ουσιαστικά από τη μηχανή πεπερασμένων καταστάσεων έχοντας ως όρια τις διευθύνσεις start – end του address_matrix και περιέχει τα matches του look - up table ($3^{η}$ & $4^{η}$ στήλη) για τις συγκεκριμένες ακολουθίες.



Σχήμα 4 .1 : Διευθυνσιοδότηση residue_matrix μέσω address_matrix

Η εύρεση match , τόσο direct όσο και indirect όπως αναλύθηκε στην υποενότητα 3.3.2 , για τις εισόδους s1, s2, r1, r2 γίνεται και αυτή επίσης μέσω της μονάδας ελέγχου (control - fsm).

Αρχικά ελέγχεται η ύπαρξη direct match ,αν δηλαδή υπάρχει στο look - up table η είσοδος s1, s2, r1, r2 . Αν υπάρχει τότε το αντίστοιχο βάρος κρατείται σε έναν καταχωρητή με το όνομα score. Αν δεν υπάρχει τότε ο καταχωρητής έχει τιμή μηδέν.

Στη συνέχεια γίνεται έλεγχος για το αν υπάρχει indirect match , αν δηλαδή υπάρχει match μεταξύ s1, r1 και s2,r2 μέσω κάποιου άλλου ζευγαριού sequence - residue (t_s - t_r) , ελέγχεται δηλαδή παράλληλα αν υπάρχει match μεταξύ s1 - t_s και s2 - t_s.

Αν υπάρχει τότε στον καταχωρήτη score προστίθεται το ελάχιστο βάρος των W1 (s1(r1) , t_s(t_r)) και W2 (s2(r2) , t_s(t_r)) .Τέλος προστίθενται όλα τα βάρη που έχουν βρεθεί για την είσοδο s1, s2, r1, r2 της συνάρτησης αναλόγως των matches που έχουν βρεθεί μεταξύ της συγκεκριμένης εισόδου και του αρχείου εισόδου στο σύστημα.

Πρώτα βρίσκουμε τη σχέση των s1 - t_s και s2 - t_s ώστε να διευθυνσιοδοτηθεί σωστά η μνήμη address_matrix.

Μόλις βρεθούν οι σωστές διευθύνσεις για τη μνήμη address_matrix , από τα περιεχόμενα τους διευθυνσιοδοτείται η μνήμη residue_matrix ώστε να ελεγχθεί αν υπάρχουν

matches μεταξύ των residues r1 , r2 των ακολουθιών s1 , s2 και κάποιου residue t_r του t_s που εξετάζεται κάθε φορά.

Ανάλογα τώρα με το αν το t_s είναι μικρότερο του s1, ή μεγαλύτερο του s2, ή είναι μεταξύ των s1, s2 υπάρχουν οι αντίστοιχες περιπτώσεις για την εύρεση match στη residue_matrix. Ενδεικτικά:

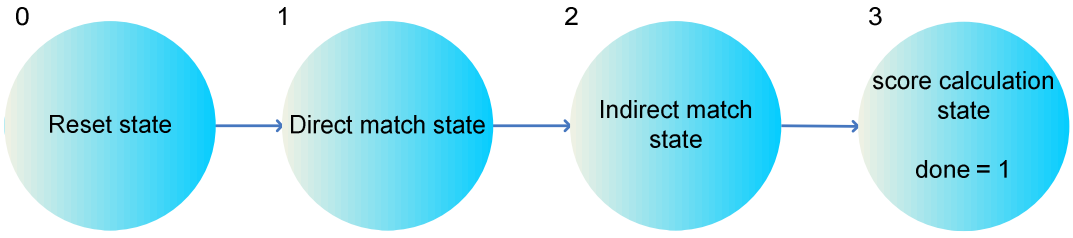
- s1 = 1s2 = 2 t_s = 0 => 0 - 1 0 - 2
- s1 = 1s2 = 2 t_s = 3 => 1 - 3 2 - 3
- s1 = 1s2 = 3 t_s = 2 => 1 - 2 2 - 3

Αν υπάρχει match μεταξύ των r1 και r2 των s1 - t_s και s2 - t_s με το ίδιο t_r τότε προστίθεται στο score το βάρος του ζευγαριού που έχει την μικρότερη τιμή. Αν δεν βρεθεί ίδιο t_r τότε γίνεται έλεγχος του επόμενου t_s – t_r.

Αν το t_s είναι ίσο με κάποιο από τα s1 ή s2 τότε προχωράμε αμέσως σε επόμενο t_s . Το t_s είναι ουσιαστικά ένας counter που παίρνει τιμή από μηδέν έως (ns – 1), όπου ns είναι ο αριθμός των ακολουθιών που υπάρχουν στο αρχείο εισόδου . Μόλις ολοκληρωθεί η αναζήτηση για match για ένα συγκεκριμένο t_s τότε προχωράμε σε επόμενο. Όταν το t_s πάρει τιμή μεγαλύτερη από τον αριθμό των ακολουθιών που υπάρχουν στο αρχείο εισόδου αυτό σημαίνει ότι έχουν ελεγχθεί όλες οι δυνατές περιπτώσεις για την εύρεση match της εισόδου s1, s2, r1, r2, τόσο για direct όσο και για indirect , και τότε υπολογίζεται το score από το άθροισμα των βαρών για την είσοδο s1, s2, r1, r2.

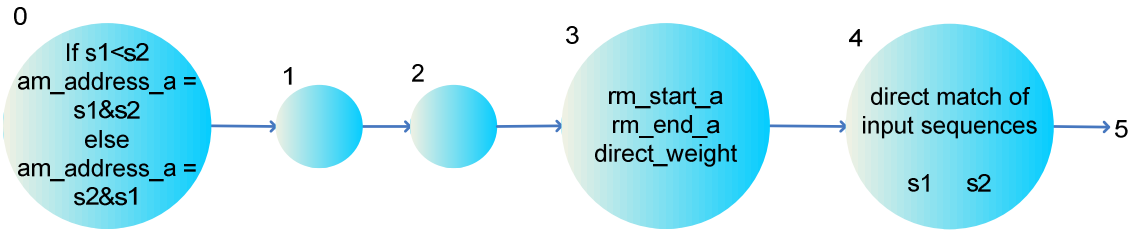
Για να γίνει εύκολα αντιληπτό αυτό υπάρχει ένα σήμα done που παίρνει την τιμή 1 και δηλώνει ότι έχει ολοκληρωθεί η αναζήτηση για την εύρεση των matches μεταξύ της εισόδου που έχουμε εισάγει και όλων των υπόλοιπων ακολουθιών που υπάρχουν στο αρχείο εισόδου.

Παρακάτω φαίνεται μία γενική fsm που παρουσιάζει τις βασικές λειτουργίες της συνάρτησης residue_pair_extended_list.



Σχήμα 4 .2 : Γενική Μηχανή Πεπερασμένων Καταστάσεων της μονάδας ελέγχου της συνάρτησης residue_pair_extended_list

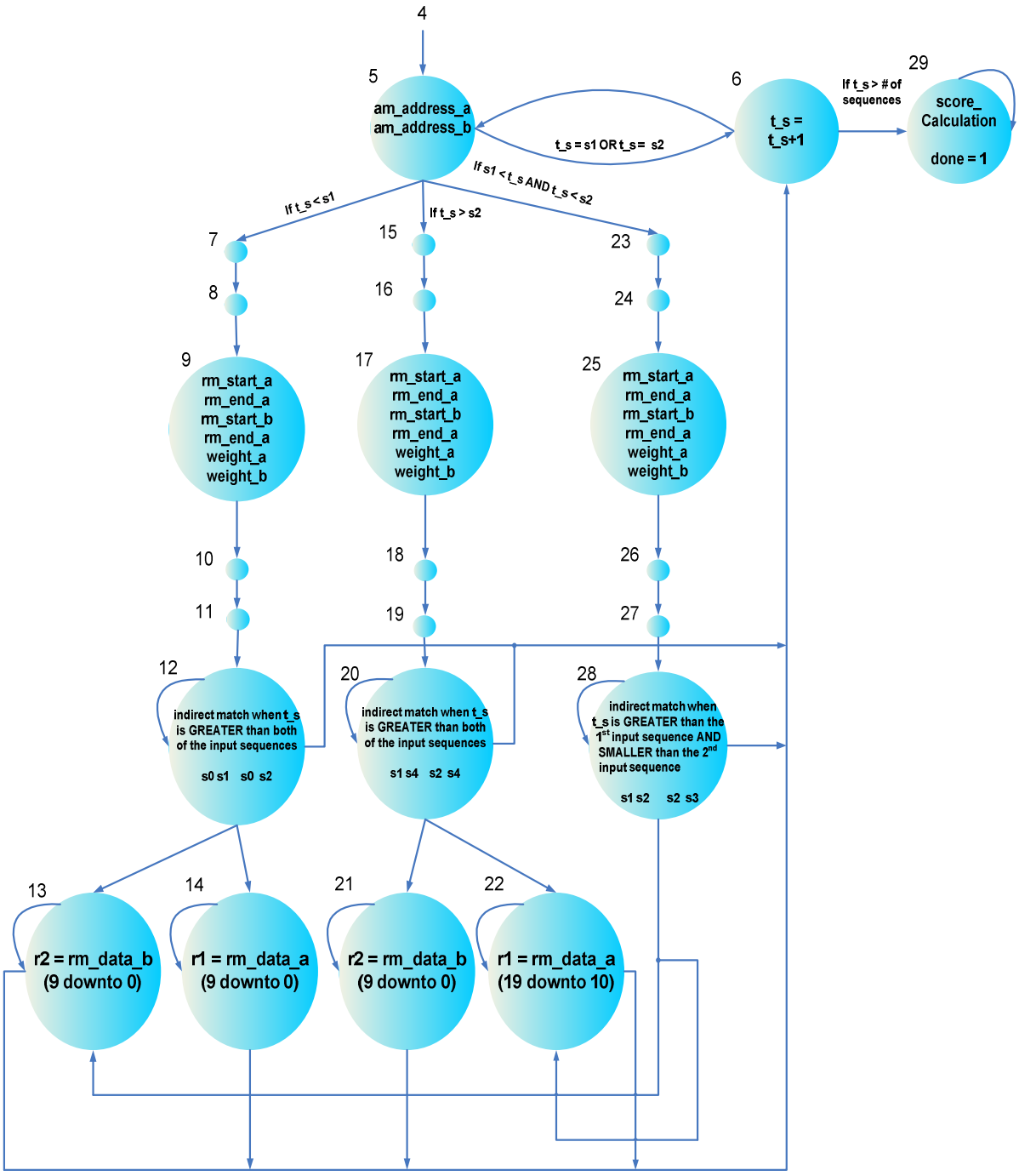
Οι καταστάσεις της fsm για την εύρεση του direct match φαίνονται αναλυτικά στην παρακάτω fsm .



Σχήμα 4 .3 : Μηχανή Πεπερασμένων Καταστάσεων για την εύρεση του direct match

Να σημειωθεί ότι οι κενές καταστάσεις παρεμβάλλονται για λόγους σωστής λειτουργίας της fsm ώστε να συγχρονίζονται τα δεδομένα των μνημών address_matrix και residue_matrix . Λόγω κατασκευής τους , οι μνήμες παράγουν δεδομένα κάθε δύο κύκλους γι' αυτό και μετά από κάθε διευθυνσιοδότηση τους απαιτούνται δύο κενές καταστάσεις ώστε να διαβαστούν τα σωστά δεδομένα από τη συγκεκριμένη διεύθυνση.

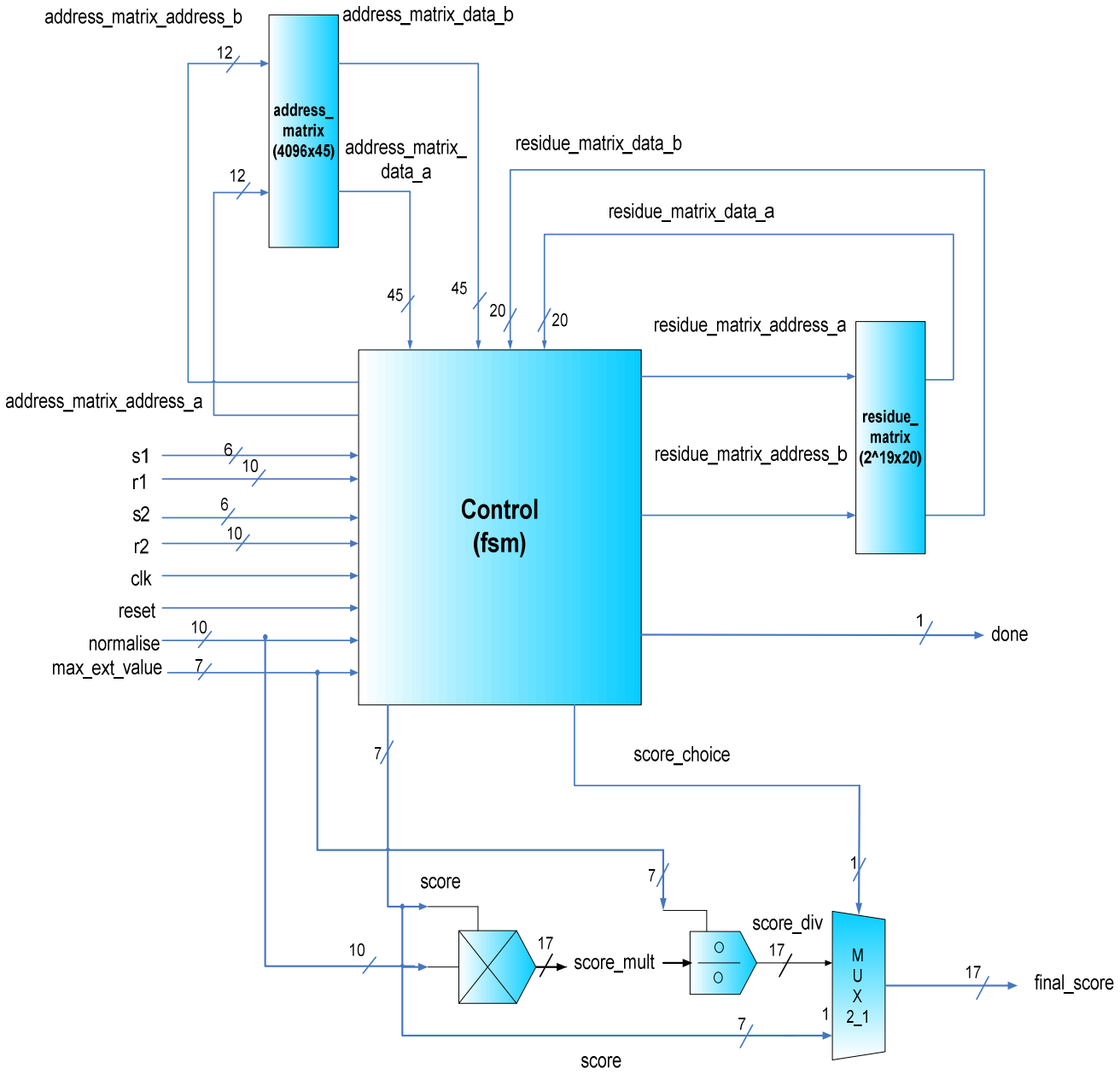
Οι καταστάσεις της fsm για την εύρεση του indirect match καθώς και του υπολογισμού του score από τα matches που προέκυψαν φαίνονται αναλυτικά στην παρακάτω fsm



Σχήμα 4 .4 : Μηχανή Πεπερασμένων Καταστάσεων για την εύρεση του indirect match και τον υπολογισμό του score

am => address_matrix
rm => residue_matrix

Ακολουθεί το συνολικό block diagram της συνάρτησης residue_pair_extended_list



Σχήμα 4 .5 : Block diagram της αρχιτεκτονικής για τη συνάρτηση residue_pair_extended_list

Στο κάτω μέρος του block diagram φαίνεται η διαδικασία για την εξαγωγή του τελικού score. Το score που προκύπτει από το άθροισμα των βαρών μεταξύ της εισόδου s1 , s2 , r1 , r2 και των matches που έχουν βρεθεί στο look - up table , πολλαπλασιάζεται με την σταθερά normalize , το γινόμενο που προκύπτει διαιρείται με τη σταθερά max_ext_value και αναλόγως την τιμή του σήματος ελέγχου για το αν η σταθερά normalize έχει ή όχι θετική τιμή, περνάει από τον πολυπλέκτη ως έξοδος που αντιστοιχεί στο τελικό score το πηλίκο της διαίρεσης ή το score που βρέθηκε από το άθροισμα των βαρών.

4.2 Περιγραφή της σχεδίασης και της υλοποίησης της συνάρτησης fast_get_dp_cost

Για την αναπαράσταση του look - up table που χρειάζεται για την εύρεση των matches μεταξύ των στοιχείων της list2 και των ακολουθιών που υπάρχουν στο αρχείο εισόδου , κατά την υλοποίηση της συνάρτησης αυτής χρησιμοποιήθηκαν και εδώ οι μνήμες address_matrix και residue_matrix όπως αυτές περιγράφηκαν στη συνάρτηση residue_pair_extended_list.

Οι δύο λίστες που χρησιμοποιούνται εδώ είναι και αυτές 2 μνήμες BRAM , επίσης dual port ROM και η διευθυνσιοδότηση καθώς και η επεξεργασία τους ως προς τα matches που υπάρχουν για τα στοιχεία αυτών γίνεται και σε αυτή την περίπτωση μέσω μια μονάδας ελέγχου (fsm) . Οι διαστάσεις που έχουν οι 2 λίστες είναι 32x17. Το βάθος τους είναι 32 γιατί όπως διευκρινίστηκε στην υποενότητα 3.3.3 ο μέγιστος αριθμός ακολουθιών και κατά συνέπεια στοιχείων σε κάθε λίστα είναι 25. Το πλάτος τους είναι 17 γιατί κάθε στοιχείο της λίστας είναι ένα ζεύγος sequence –pair, οπότε χρειάζονται 6 bits για την αναπαράσταση μία εκ των 50 ακολουθιών και 11 bits για την αναπαράσταση ενός εκ των 1000 residues ακόμη και σε περίπτωση που υπάρξει στη λίστα αρνητικό residue.

Η διευθυνσιοδότηση των μνημών address_matrix και residue_matrix γίνεται και εδώ όπως ακριβώς περιγράφηκε και στη συνάρτηση residue_pair_extended_list , μόνο που εδώ η διευθυνσιοδότηση δεν γίνεται μέσω ακολουθιών που δίνονται ως είσοδο αλλά μέσω των ακολουθιών που υπάρχουν στις δύο λίστες .

Αρχικά διαβάζονται τα περιεχόμενα της πρώτης διεύθυνσης της λίστας που έχει τα λιγότερα στοιχεία , που ουσιαστικά είναι ένα ζεύγος sequence - residue. Αν τύχει το στοιχείο residue να είναι αρνητικό τότε εξετάζεται αμέσως η επόμενη διεύθυνση. Υπάρχει ένας counter n_gap2 που μετράει πόσα αρνητικά στοιχεία residues υπάρχουν στη list2. Ομοίως υπάρχει ένας counter n_gap1 για τη list1.

Κατά την υλοποίηση αυτής της συνάρτησης υπάρχει και εδώ ο ίδιος counter όπως και στην προηγούμενη συνάρτηση που ουσιαστικά δείχνει με ποια ακολουθία (t_s) από το

σύνολο που υπάρχει στο αρχείο εισόδου θέλουμε να εξετάσουμε αν υπάρχει ή όχι match μεταξύ του residue που υπάρχει στη λίστα και κάποιου από την ακολουθία που δείχνει ο counter.Προφανώς ελέγχονται και πάλι όλες οι ακολουθίες που βρίσκονται στο αρχείο εισόδου. Γίνεται σύγκριση των δύο ακολουθιών , αυτής που βρίσκεται στη διεύθυνση που δείχνει κάθε φορά στη λίστα και αυτής που δείχνει ο counter και με αυτόν τρόπο διευθυνσιοδοτείται η μνήμη address_matrixκαι ανάλογα τα περιεχόμενα αυτής της διεύθυνσης , διευθυνσιοδοτείται και η μνήμη residue_matrix.

Ανάλογα και πάλι με το αν το t_s είναι μικρότερο ή μεγαλύτερο του sequence που υπάρχει στη λίστα υπάρχουν οι αντίστοιχες περιπτώσεις για την εύρεση match στη residue_matrix .

Αν δεν βρεθεί match (t_r) μεταξύ του sequence που βρίσκεται στην $1^{η}$ διεύθυνση και της πρώτης ακολουθίας του αρχείου εισόδου (s0) τότε εξετάζεται επόμενο t_s (s1) κ.ο.κ. Αν βρεθεί match , τότε κρατείται το βάρος που υπάρχει look - up table σε ένα καταχωρητή hasch2 και υπάρχει και ένας counter n_hasch2 που κρατάει τον αριθμό των matches που υπάρχουν μεταξύ ενός t_s (t_r) και όλων των στοιχείων της list2.

Ακολουθεί έλεγχος για το αν υπάρχει match μεταξύ του t_s (t_r) που βρέθηκε και του $1^{ου}$ στοιχείου της δεύτερης λίστας. Έτσι διευθυνσιοδοτείται η δεύτερη λίστα. Έπειτα διευθυνσιοδοτείται ξανά η address_matrix , τώρα μέσω του t_s και της ακολουθίας που βρίσκεται στη $1^{η}$ διεύθυνση της δεύτερης λίστας και αναλόγως τη σχέση των ακολουθιών , υπάρχουν οι αντίστοιχες περιπτώσεις για την εύρεση match στη residue_matrix .

Αν βρεθεί match , κρατείται και σε αυτή την περίπτωση το βάρος που υπάρχει look - up table σε ένα καταχωρητή hasch1 και υπάρχει και ένας ακόμη counter n_hasch1 που κρατάει τον αριθμό των matches που υπάρχουν μεταξύ ενός t_s (t_r) και όλων των στοιχείων της list1.

Στη συνέχεια ελέγχεται παράλληλα αν υπάρχει match μεταξύ του t_s (t_r) και του $2^{ου}$ στοιχείου της λίστας που επεξεργαζόμαστε καθώς επίσης και μεταξύ του t_s (t_r) και του $2^{ου}$ στοιχείου της δεύτερης λίστας , κ.ο.κ. Αν προκύψουν και άλλα matches με το συγκεκριμένο t_s (t_r) ,το αντίστοιχο βάρος προστίθεται στους καταχωρητές hasch2 και hasch1 και αυξάνονται αντίστοιχα οι counters n_hasch2 και n_hasch1 για να δηλώσουν την εύρεση νέου match μεταξύ του t_s (t_r) και των στοιχείων που βρίσκονται στις λίστες .

Η διαδικασία αυτή θα συνεχιστεί για όλα τα στοιχεία των δύο λιστών και μόλις ολοκληρωθεί , η μηχανή πεπερασμένων καταστάσεων ξεκινάει την ίδια διαδικασία για να βρεθεί το $2^{ο}$ match μεταξύ του $1^{ου}$ στοιχείου της list2 και του τρισδιάστατου πίνακα.

Πριν την εύρεση όμως νέου t_s (t_r) , υπολογίζεται ένα score για κάθε t_s (t_r) αρκεί να πληρούνται κάποιες προϋποθέσεις. Υπάρχουν δύο μεταβλητές is_in_col1 και is_in_col2 που όταν πάρουν την τιμή 1 δηλώνουν ότι το match t_s (t_r) που βρέθηκε είναι και στοιχείο της list1 και list2 αντίστοιχα. Το score επομένως μπορεί να μία από τις παρακάτω τιμές :

1. Αν υπάρχει τιμή στον καταχωρητή hasch1 και αν is_in_col1 = 1 και is_in_col2 = 1 τότε :

   score + = MIN ((hasch2 * n_hasch1) , (hasch1 * n_hasch2))

2. Αν ισχύει μόνο is_in_col1 = 1 τότε :

   score + = hasch2 * (n_hasch1 + 1)

3. Αν υπάρχει τιμή στον καταχωρητή hasch1 και αν is_in_col2 = 1 τότε :

   score + = hasch1 * (n_hasch2 + 1)

Αν δεν ισχύει καμία από τις παραπάνω συνθήκες τότε το score ισούται με μηδέν.

Θεωρείται ότι ο καταχωρητής hasch2 έχει πάντα τιμή αφού πρώτα πρέπει να βρεθεί match με κάποιο στοιχείο της list2 και μετά να γίνει έλεγχος για το αν υπάρχει match του ίδιου t_s (t_r) με κάποιο στοιχείο της list1.

Για τον υπολογισμό του score σε μία από τις παραπάνω περιπτώσεις χρησιμοποιήθηκαν 4 πολλαπλασιαστές που υπολογίζουν τα γινόμενα (hasch2 * n_hasch1) , (hasch1 * n_hasch2) , hasch2 * (n_hasch1 + 1) και hasch1 * (n_hasch2 + 1) αντίστοιχα , δύο αθροιστές που προσθέτουν τα n_hasch2 και n_hasch1 με τη μονάδα , ένας συγκριτής για να επιλέγει τη μικρότερη ποσότητα κατά τον υπολογισμό του score στην $1^{η}$ περίπτωση , ένας πολυπλέκτης 4 σε 1 για να επιλέγεται αντίστοιχα ο κατάλληλος υπολογισμός του score αναλόγως των συνθηκών που ισχύουν και την τιμή του σήματος ελέγχου που προέρχεται από την fsm , ένας αθροιστής για να προσθέτει το νέο score που υπολογίζεται στο προηγούμενο και ένας καταχωρητής που κρατάει το τελικό score που προκύπτει μετά την εύρεση ενός t_s (t_r).

Μόλις ολοκληρωθεί η εύρεση όλων των matches του $1^{ου}$ στοιχείου της list2 τότε προχωράμε στην εύρεση των matches του $2^{ου}$ στοιχείου αυτής της λίστας και παράλληλα ελέγχονται και τα στοιχεία της δεύτερης λίστας.

Όταν βρεθούν τα matches όλων των στοιχείων της list2 , δηλαδή όταν η διεύθυνση της list2 γίνει ίση με ns2, τότε υπολογίζεται το τελικό score για όλα τα t_s (t_r) που βρέθηκαν για τα στοιχεία της list2 σύμφωνα με τον παρακάτω τύπο :

score = (score * SCORE_K) / ((ns1 - n_gap1) * (ns2 - n_gap2))

score = (normalise) ? ((score * normalise) / max_ext_value) : score

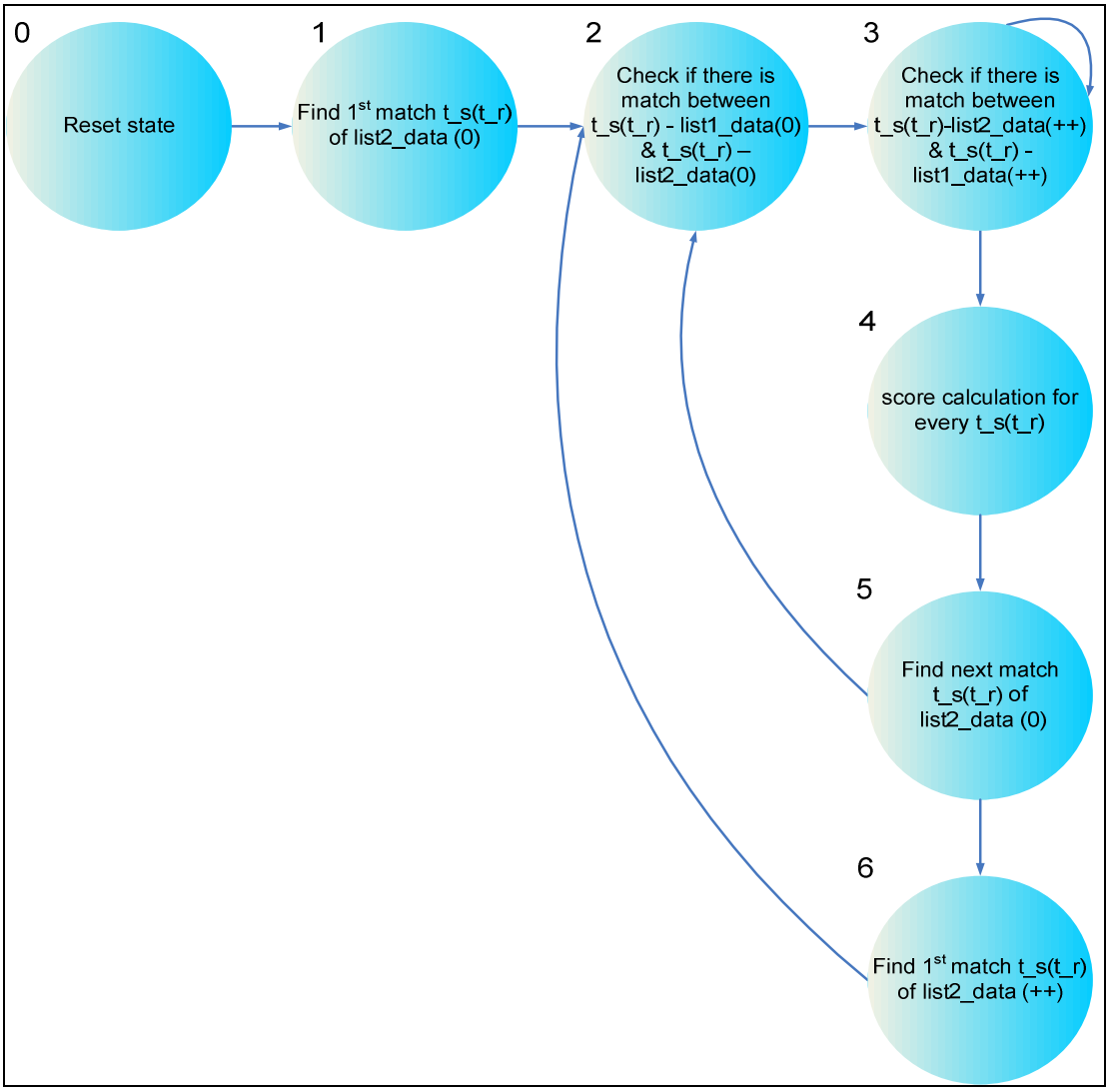
Για τον υπολογισμό του τελικού score χρησιμοποιήθηκαν 3 πολλαπλασιαστές για τον υπολογισμό των γινομένων (score * SCORE_K) , (ns1 - n_gap1) * (ns2 - n_gap2) , (score * normalise) , 2 αφαιρέτες για τον υπολογισμό των ποσοτήτων (ns1 - n_gap1) και (ns2 - n_gap2) , 2 διαιρέτες για τον υπολογισμό των πηλίκων και τέλος ένας πολυπλέκτης 2 σε 1 για την επιλογή του τελικού score.

Αν έχει δοθεί θετική τιμή στη μεταβλητή normalize, τότε το τελικό score υπολογίζεται ως εξής :

η τιμή του score, όπως αυτή προέκυψε από την πρώτη διαίρεση, πολλαπλασιάζεται με τη μεταβλητή normalize και στη συνέχεια το αποτέλεσμα του πολλαπλασιασμού διαιρείται με την τιμή της μεταβλητής max_ext_value , όπως και κατά τον υπολογισμό του τελικού score στη συνάρτηση residue_pair_extended_list. Το αποτέλεσμα της διαίρεσης ορίζεται ως έξοδος του πολυπλέκτη και αποτελεί το τελικό score.

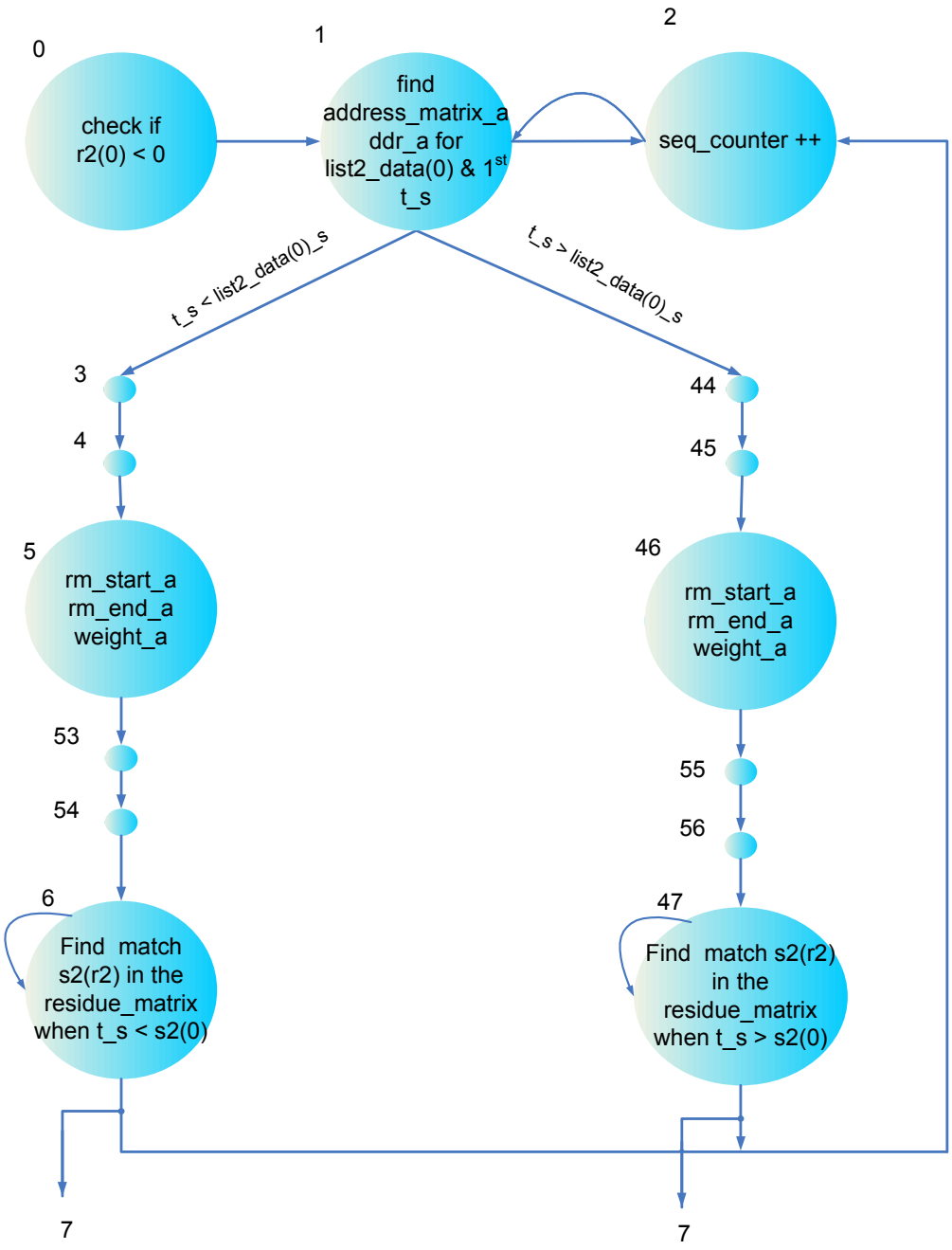
Αν δεν έχει δοθεί θετική τιμή στη μεταβλητή normalize, τότε ως έξοδος του πολυπλέκτη ορίζεται η τιμή που προέκυψε από την πρώτη διαίρεση και αυτή θεωρείται ως το τελικό score.

Παρακάτω φαίνεται μία γενική fsm που παρουσιάζει τα βασικά βήματα της συνάρτησης fast_get_dp_cost.



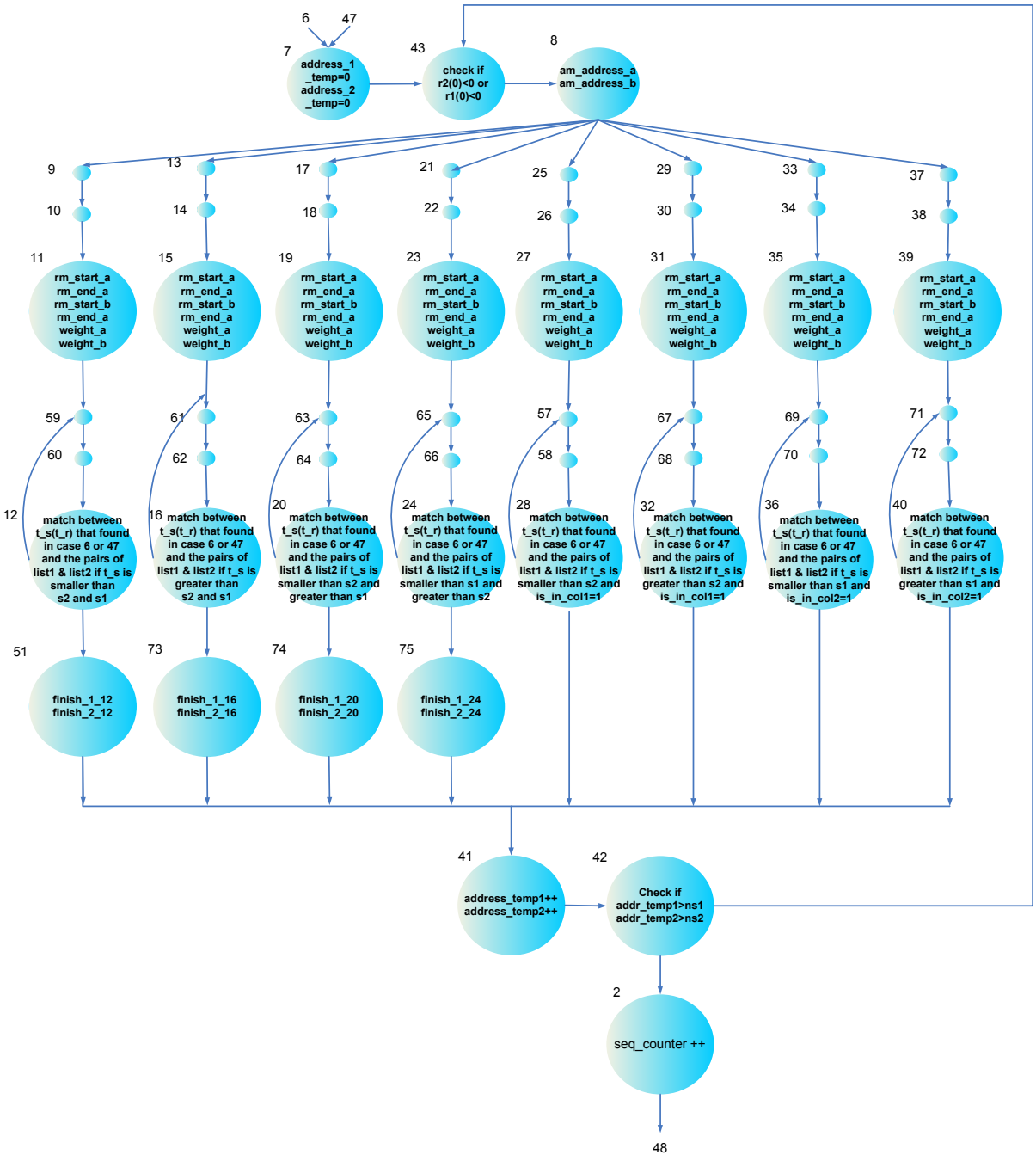
Σχήμα 4 .6 : Γενική Μηχανή Πεπερασμένων Καταστάσεων της μονάδας ελέγχου της συνάρτησης fast_get_dp_cost

Βήμα 1:



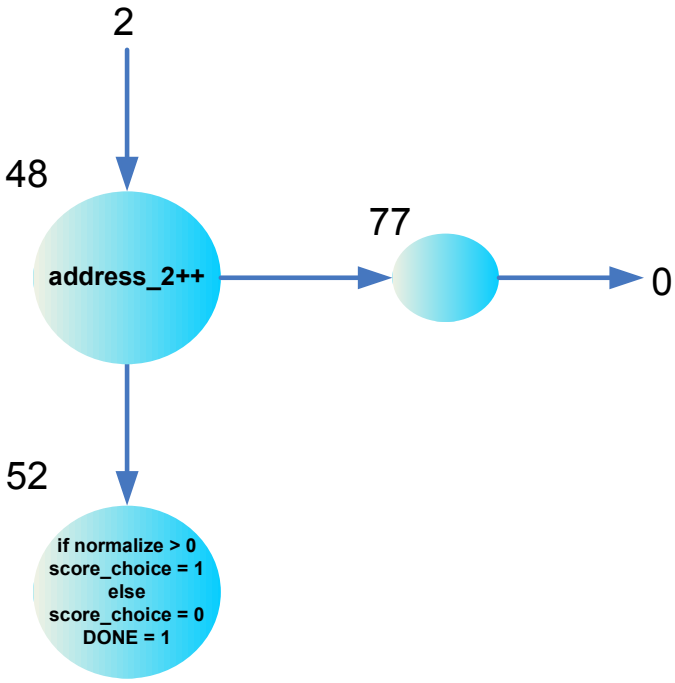
Σχήμα 4 .7 : Μηχανή Πεπερασμένων Καταστάσεων του 1ου βήματος της μονάδας ελέγχου της συνάρτησης fast_get_dp_cost

Βήματα 2 , 3 & 4:



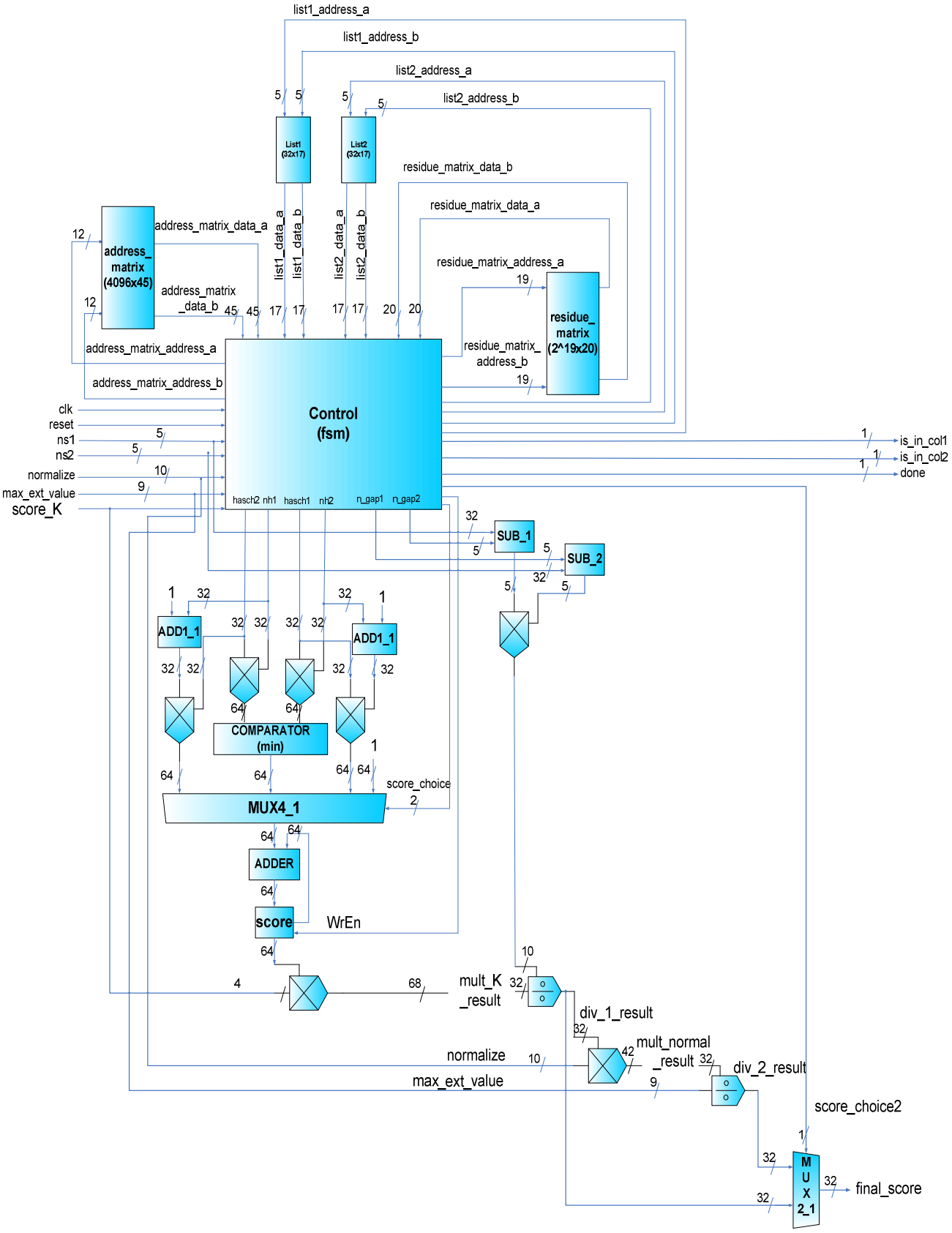
Σχήμα 4 .8 : Μηχανή Πεπερασμένων Καταστάσεων $2^{ου}$, $3^{ου}$, $4^{ου}$ βήματος της μονάδας ελέγχου της συνάρτησης fast_get_dp_cost

Βήμα 6:



Σχήμα 4 .9 : Μηχανή Πεπερασμένων Καταστάσεων 6^{ου} βήματος της μονάδας ελέγχου της συνάρτησης fast_get_dp_cost

Ακολουθεί το συνολικό block diagram της συνάρτησης fast_get_dp_cost



Σχήμα 4 .9: Block diagram της συνάρτησης fast_get_dp_cost

Κεφάλαιο 5

Επιβεβαίωση Λειτουργίας και Απόδοση Συστήματος

Στο κεφάλαιο αυτό περιγράφεται η διαδικασία επιβεβαίωσης λειτουργίας των συστημάτων που συνθέτουν την αρχιτεκτονική και παρουσιάζονται τα αποτελέσματα της απόδοσης σε σύγκριση με το PC.

5.1 Ταυτοποίηση λειτουργίας

Η λειτουργία της συγκεκριμένης αρχιτεκτονικής προσομοιώθηκε με το εργαλείο της Xilinx ISE Simulator 10.1. Κατά την διαδικασία της υλοποίησης της αρχιτεκτονικής, προσομοιώθηκε η λειτουργία των επιμέρους τμημάτων των δύο συναρτήσεων και διαπιστώθηκε η σωστή λειτουργία τους. Έπειτα, αφού ενώθηκαν όλα τα τμήματα προσομοιώθηκε το συνολικό σύστημα ώστε να διαπιστωθεί η σωστή λειτουργία του για κάθε συνάρτηση ξεχωριστά.

Τα αποτελέσματα της αρχιτεκτονικής συγκρίθηκαν με τα αποτελέσματα του αλγορίθμου T-Coffee κατά την εισαγωγή 4 αρχείων και διαπιστώθηκε η σωστή λειτουργία της σχεδίασης. Τα δεδομένα που εισήχθησαν φαίνονται αναλυτικά στον παρακάτω πίνακα και συγκεκριμένα το ποσοστό χρόνου εκτέλεσης της κάθε συνάρτησης που υπολογίστηκε για κάθε είσοδο από το Vtune analyzer, ο χρόνος εκτέλεσης του προγράμματος T-Coffee για κάθε είσοδο, ο αριθμός ακολουθιών (Seq No), το μήκος της μεγαλύτερης ακολουθίας (Length) για τη συγκεκριμένη είσοδο καθώς επίσης και ο αριθμός όλων των στοιχείων (res No) που υπάρχουν αντίστοιχα για κάθε αρχείο εισόδου.

Inputs	Seq No	Seq Length	Library (res No)	% residue_pair_ extended_list	% fast_get_ dp_cost	T-Coffee Execution Time (sec)
1amk_ref1	5	257	2,437	16.35 %	-	0.345
1idy_ref2	22	65	12.355	52.83 %	14.49 %	4.876
1wit_ref2	22	117	20,368	41.61 %	18.55 %	7.490
1ag8_ref4	19	549	78,176	48.01 %	16.08 %	26.932

Πίνακας 5-1: Τα πειράματα που εισήχθησαν για την ταυτοποίηση της αρχιτεκτονικής

5.2 Απόδοση Συστήματος

Η συγκεκριμένη αρχιτεκτονική υλοποιήθηκε σε Virtex-4 FPGA Family Member Device XC4VFX140, η οποία επιλέχθηκε λόγω του ότι έχει τον μεγαλύτερο αριθμό block RAMs (552 BRAMs) από την οικογένεια αυτή. Το Post Place and Route της σχεδίασης πραγματοποιήθηκε με την χρήση του εργαλείου Xilinx 10.1 και έτσι προέκυψαν η συχνότητα ρολογιού και οι πόροι της αρχιτεκτονικής. Πραγματοποιήθηκε Place & Route Simulation για να επιβεβαιωθεί η λειτουργία του συστήματος στη συγκεκριμένη FPGA.

Οι πόροι της FPGA που καταλαμβάνονται για τη συνάρτηση residue_pair_extended_list φαίνονται αναλυτικά στον παρακάτω πίνακα.

Inputs	**Πόροι FPGA**			**Parallel Computing Units**	**Clock Frequency**
	BRAMs	**Slices**	**DSPs**		
1amk_ref1	2% (14 out of 552)	1 % (553 out of 63168)	1% (1 out of 192)	39	160MHz
1idy_ref2	4% (25 out of 552)	1 % (698 out of 63168)	1% (1 out of 192)	22	160MHz
1wit_ref2	5% (33 out of 552)	1 % (742 out of 63168)	1% (1 out of 192)	16	144MHz
1ag8_ref4	17% (97 out of 552)	2 % (1495 out of 63168)	1% (1 out of 192)	5	147MHz

Πίνακας 5-2: Χρησιμοποίηση πόρων της FPGA για τη συνάρτηση residue_pair_extended_list

Οι πόροι της FPGA που καταλαμβάνονται για τη συνάρτηση fast_get_dp_cost φαίνονται αναλυτικά στον παρακάτω πίνακα.

Inputs	Πόροι FPGA			Parallel Computing Units	Clock Frequency
	BRAMs	Slices	DSPs		
1amk_ref1	-	-	-	-	-
1idy_ref2	4% (27 out of 552)	2% (1842 out of 63168)	9% (19 out of 192)	10	146MHz
1wit_ref2	6% (35 out of 552)	2% (1863 out of 63168)	9% (19 out of 192)	10	142MHz
1ag8_ref4	17% (99 out of 552)	4 % (2633 out of 63168)	9% (19 out of 192)	5	131MHz

Πίνακας 5-3: Χρησιμοποίηση πόρων της FPGA για τη συνάρτηση fast_get_dp_cost

Όπως φαίνεται και από τα αποτελέσματα υπάρχει μεγάλο μέρος αχρησιμοποίητων πόρων για αυτό μπορούν να χρησιμοποιηθούν παράλληλες μονάδες επεξεργασίας ανάλογα με το αρχείο εισόδου και το χώρο που καταλαμβάνει το Library της κάθε εισόδου (είναι ουσιαστικά η μνήμη residue_matrix) ώστε να επιτευχθεί το ανάλογο speedup.

5.3 Αποτίμηση Συστήματος

Ο αλγόριθμος T-Coffee εκτελέστηκε σε υπολογιστή με επεξεργαστή Intel Pentium 4 στα 2,66 GHz , με 1 GB RAM και λειτουργικό σύστημα ubudu 8.04.2. Για κάθε αρχείο εισόδου το πρόγραμμα εκτελέστηκε 5 φορές. Ο μέσος χρόνος εκτέλεσης στον υπολογιστή για κάθε διαφορετική είσοδο καθώς και ο χρόνος εκτέλεσης του αλγορίθμου σε αναδιατασσόμενη λογική φαίνεται στους παρακάτω πίνακες.

Συνάρτηση residue_pair_extended_list:

Συνολικές κλήσεις συνάρτησης residue_pair_extended_list για την είσοδο 1amk_ref1: 106,497

Input: 1amk_ref1		**Sequences No: 5 Length(residues): 257**		**Parallel Computing Units: 39**
Execution Time T-Coffee: 0.345 sec			Function Execution Time: 0.06sec	
Residues Input		Time SW (μsec)	Time HW (μsec)	Speedup (parallel computing units)
3	1	0.39	0.2	74 x
9	7	0.38	0.24	61 x
28	26	0.39	0.37	41 x
61	59	0.39	0.73	21 x
203	200	0.38	3.73	4 x

Πίνακας 5-4: Χρόνος εκτέλεσης SW vs HW για την είσοδο 1amk_ref1 για τη συνάρτηση residue_pair_extended_list

Συνολικές κλήσεις συνάρτησης residue_pair_extended_list για την είσοδο 1idy_ref2: 1,180,057

Input: 1idy_ref2		**Sequences No: 22 Length(residues): 65**		**Parallel Computing Units: 22**
Execution Time T-Coffee: 4.876 sec			Function Execution Time: 2.59 sec	
Residues Input		Time SW (μsec)	Time HW (μsec)	Speedup (parallel computing units)
1	1	1.55	1.06	32 x
2	2	1.67	1.08	34 x
10	10	1.66	1.3	28 x
33	33	1.69	2.83	13 x
57	57	1.50	8.4	4 x

Πίνακας 5-5: Χρόνος εκτέλεσης SW vs HW για την είσοδο 1idy_ref2 για τη συνάρτηση residue_pair_extended_list

Συνολικές κλήσεις συνάρτησης residue_pair_extended_list για την είσοδο 1ag8_ref41: 1,495,768

Input: 1wit_ref2		**Sequences No: 22 Length(residues): 117**		**Parallel Computing Units: 16**
Execution Time T-Coffee: 7.490 sec			Function Execution Time: 3.15 sec	
Residues Input		Time SW (μsec)	Time HW (μsec)	Speedup (parallel computing units)
2	2	1.28	1.197	17 x
5	5	1.49	1,21	19 x
20	20	1.58	1.52	16 x
58	59	1.64	4.19	6 x
91	92	1.61	8.1	3 x

Πίνακας 5-6: Χρόνος εκτέλεσης SW vs HW για την είσοδο 1wit_ref2 για τη συνάρτηση residue_pair_extended_list

Συνολικές κλήσεις συνάρτησης residue_pair_extended_list για την είσοδο 1idy_ref2: 8,347,622

Input: 1ag8_ref4		**Sequences No: 19 Length(residues): 549**		**Parallel Computing Units: 5**
Execution Time T-Coffee: 26.932 sec			Function Execution Time: 12.92 sec	
Residues Input		Time SW (μsec)	Time HW (μsec)	Speedup (parallel computing units)
1	5	0.46	1.064	2 x
25	29	1.41	3.018	2 x
136	140	1.1	16.11	0.3 x
278	278	1.52	32.95	0.2 x
483	482	1.48	57.63	0.13 x

Πίνακας 5-7: Χρόνος εκτέλεσης SW vs HW για την είσοδο 1ag8_ref4 για τη συνάρτηση residue_pair_extended_list

Από τους πίνακες φαίνεται πώς όσο μεγαλύτερο είναι το residue για το οποίο γίνεται η αναζήτηση τόσο μειώνεται η επιτάχυνση του συστήματος ακόμη και όταν υπάρχει παραλληλία. Αυτό οφείλεται στο γεγονός ότι για μεγάλα residues το σύστημα μας απαιτεί την διαπέραση της μνήμης residue_matrix σε μεγάλο βάθος(επομένως πολλές προσβάσεις στην μνήμη) αντίθετα με το software όπου η πληροφορία είναι αποθηκευμένη σε ένα τρισδιάστατο lookup table, όπως αναφέρθηκε στο κεφάλαιο 3. Ένα άλλο χαρακτηριστικό των αποτελεσμάτων είναι ότι το σύστημα μας επιτυγχάνει ικανοποιητική επιτάχυνση της εκτέλεσης της συνάρτησης για εισόδους με σχετικά μικρό μήκος των sequences ενώ για μεγάλες αλληλουχίες η επιτάχυνση μειώνεται και σε ορισμένες περιπτώσεις το σύστημα μας είναι πιο αργό. Ο κύριος λόγος της κακής απόδοσης του συστήματος για μεγάλες αλληλουχίες είναι η αύξηση απαιτήσεων σε μνήμη όπου περιοριζόμαστε από τα resources της FPGA με αποτέλεσμα να μην υπάρχει μεγάλη παραλληλία ανεξάρτητων υπολογιστικών μονάδων.

Συνάρτηση fast_get_dp_cost:

Συνολικές κλήσεις συνάρτησης fast_get_dp_cost για την είσοδο 1idy_ref2: 28,542

Input: 1idy_ref2	**Sequences No: 22 Length(residues): 65**		**Parallel Computing Units: 10**
Execution Time T-Coffee: 4.876 sec			Function Execution Time: 0.7 sec
Lists Input	Time SW (μsec)	Time HW (μsec)	Speedup (parallel computing units)
List2 List1	10.74	17.29	6 x
List2 List1	11.46	54.09	2 x
List2 List1	11.64	55.56	2 x
List2 List1	11.93	57.17	2 x
List2 List1	11.90	57.46	2 x

Πίνακας** Σφάλμα! Δεν υπάρχει κείμενο καθορισμένου στυλ στο έγγραφο.**-8: Χρόνος εκτέλεσης SW vs HW για την είσοδο 1ag8_ref4 για τη συνάρτηση fast_get_dp_cost

Συνολικές κλήσεις συνάρτησης fast_get_dp_cost για την είσοδο1wit_ref2: 54,618

Input: 1wit_ref2		**Sequences No: 22 Length(residues): 117**	**Parallel Computing Units: 10**
Execution Time T-Coffee: 7.490 sec			Function Execution Time: 1.08 sec
Lists Input	Time SW (μsec)	Time HW (μsec)	Speedup (parallel computing units)
List2 List1	4.67	50.67	1x
List2 List1	5.09	58.82	0.8x
List2 List1	6.18	59.23	1x
List2 List1	6.13	59.98	1x
List2 List1	5.69	42.88	1x

Πίνακας5-9: Χρόνος εκτέλεσης SW vs HW για την είσοδο 1ag8_ref4 για τη συνάρτηση fast_get_dp_cost

Συνολικές κλήσεις συνάρτησης fast_get_dp_cost για την είσοδο 1ag8_ref4: 271,905

Input: 1ag8_ref4		**Sequences No: 19 Length(residues): 549**	**Parallel Computing Units: 5**
Execution Time T-Coffee: 26.932 sec			Function Execution Time: 4.29 sec
Lists Input	Time SW (μsec)	Time HW (μsec)	Speedup (parallel computing units)
List2 List1	2.41	23.98	0.5x
List2 List1	2.8	26.52	0.5x
List2 List1	4.21	29.31	0.7x
List2 List1	4.29	34.52	0.6 x
List2 List1	4.32	36.71	0.5 x

Πίνακας 5-10: Χρόνος εκτέλεσης SW vs HW για την είσοδο 1ag8_ref4 για τη συνάρτηση fast_get_dp_cost

Στην συνάρτηση αυτή βλέπουμε ότι τα αποτελέσματα για μικρές εισόδους είναι αρκετά καλά αλλά για μεσαίες και μεγάλες εισόδους παρατηρούμε ότι τα αποτελέσματα μας είναι χειρότερα από το software. Ο λόγος είναι πάλι το γεγονός ότι κάθε είσοδος της συνάρτησης αποτελείται από πολλά στοιχεία residues με αποτέλεσμα να απαιτείται πολλές φορές η διαπέραση της εσωτερικής μνήμης. Το γεγονός αυτό προσθέτει ένα συνολικό overhead στην εκτέλεση της συνάρτησης με αποτέλεσμα συνολικά η συνάρτηση αυτή στο hardware να μην δίνει καλά αποτελέσματα για όλες τις εισόδους.

Στη συνέχεια παρατίθενται πίνακες που δείχνουν την επιτάχυνση του συστήματος για την καλύτερη και χειρότερη περίπτωση εισόδου όταν χρησιμοποιηθεί η ίδια FPGA και για τις δύο συναρτήσεις. Από τα παρακάτω αποτελέσματα προκύπτει ότι για μικρές και μεσαίες (ως προς το μήκος) αλληλουχίες επιτυγχάνουμε επιτάχυνση της εκτέλεσης και των δύο συναρτήσεων σε σχέση με το software. Είναι σημαντικό να σημειωθεί ότι για μεγάλες εισόδους δίνονται καλύτερα αποτελέσματα όταν δεν υπάρχει συνδυασμός των δύο συστημάτων.

Input: 1idy_ref2		**Sequences No: 22 Length(residues): 65**
Execution Time T-Coffee: 4.876 sec		Function Execution Time: 3.59 sec
Inputs	Parallel Computing Units	Speedup (parallel computing units)
Best Case	13 (res) / 7(fast)	10 x
Worst Case	18 (res) / 3 (fast)	1.6 x

Πίνακας 5-11: Speedup SW vs HW για την είσοδο 1idy_ref4 και για τις δύο συναρτήσεις

Input: 1wit_ref2		**Sequences No: 22 Length(residues): 117**
Execution Time T-Coffee: 7.490 sec		Function Execution Time: 4.23 sec
Inputs	Parallel Computing Units	Speedup (parallel computing units)
Best Case	2 (res) / 10 (fast)	1.7 x
Worst Case	9 (res) / 6 (fast)	1.13 x

Πίνακας 5-12: Speedup SW vs HW για την είσοδο 1wit_ref2 και για τις δύο συναρτήσεις

Input: 1ag8_ref4		Sequences No: 19 Length(residues): 549
Execution Time T-Coffee: 26.932 sec		Function Execution Time: 17.21 sec
Inputs	Parallel Computing Units	Speedup (parallel computing units)
Best Case	3 (res) / 2 (fast)	0.62 x
Worst Case	4 (res) / 1 (fast)	0.12 x

Πίνακας 5-13: Speedup SW vs HW για την είσοδο 1ag8_ref4 και για τις δύο συναρτήσεις

Κεφάλαιο 6

Συμπεράσματα και Μελλοντικές Επεκτάσεις

Στο κεφάλαιο αυτό αναφέρονται τα συμπεράσματα και οι μελλοντικές επεκτάσεις της παρούσας αρχιτεκτονικής σχεδίασης.

6.1 Συμπεράσματα

Η παρούσα διπλωματική εργασία παρουσίασε την μελέτη του αλγορίθμου T-coffee και την υλοποίηση των υπολογιστικά βαριών συναρτήσεων αυτού σε αναδιατασσόμενη λογική για την εύρεση μιας MSA. Στόχος ήταν το σύστημα αυτό να είναι αποδοτικότερο σε σύγκριση με έναν προσωπικό υπολογιστή.

Από τα αποτελέσματα μπορούμε να συμπεράνουμε ότι ο στόχος ικανοποιείται σε ικανοποιητικό βαθμό ειδικά όταν η παραλληλία που επιτρέπεται είναι αρκετά μεγάλη. Το μειονέκτημα είναι ο περιορισμός πόρων και για αυτό δεν έχουμε σε όλες τις περιπτώσεις ικανοποιητικά αποτελέσματα

6.2 Μελλοντικές Επεκτάσεις

Η αρχιτεκτονική που υλοποιήθηκε σε αυτή τη διπλωματική εργασία, μπορεί να υποστεί αλλαγές με σκοπό τη βελτίωση της απόδοσης, αλλά και να επεκταθεί ώστε να είναι δυνατή μία πιο ολοκληρωμένη προσέγγιση του αλγορίθμου T-Coffee.

Σημαντικές επεκτάσεις και αλλαγές που μπορούν να πραγματοποιηθούν είναι:

i. Πλήρης υλοποίηση του αλγορίθμου σε αναδιατασσόμενη λογική.

ii. Χρήση εξωτερικής μνήμης και κατασκευή του τρισδιάστατου lookup table ώστε η εύρεση match να γίνεται πιο γρήγορα και να μην χρειάζεται κάθε φορά να σαρώνεται όλη η μνήμη residue_matrix.

iii. Σύστημα software/hardware co-design.

iv. Υλοποίηση του συστήματος σε Virtex 5

Βιβλιογραφία

[1] Notredame, C., Higgins, D., Heringa, J.: T–Coffee: A novel method for fast and accurate multiple sequence alignment. J. Mol. Biol. **302**(1) (2000) 205–217

[2] Notredame, C.: Recent progress in multiple sequence alignment: A survey. Pharmacogenomics **3**(1) (2002) 131–144

[3] Notredame, C., Holm, L., Higgins, D.: COFFEE: an objective function for multiple sequence alignments.
Bioinformatics **14**(5) (1998) 407–422

[4] Gotoh, O.: An improved algorithm for matching biological sequences. J. Mol. Biol. **162**(3) (1982) 705–708

[5] Sander, C. & Schneider, R. (1991). Database of homology-derived protein structures and the structural meaning of sequence alignment. Proteins: Struct. Funct. Genet. 9, 56-68

[6] Thompson, J., Higgins, D. & Gibson, T. (1994).ClustalW: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-speci®c gap penalties and weight matrix choice. Nucl. Acids Res. 22, 4673-4690

[7] Kazutaka Katoh, Kazuharu Misawa, Kei-ichi Kuma & Takashi Miyata (2002). MAFFT: a novel method for raprid multiple sequence alignment based on fast Fourier Transform.Nucleic Acids Research, 2002, Vol 30 No. 14 3059-3066

[8] Buckard Morgenster, Kornelie French, Andreas Dress & Thomas Werner. DIalign: Finding local similirities by multiple sequrnce alignment. Bioinformatics Vol. 14 no.3 1998, pages 290-294

[9] Tim Oliver, Bertil Schmidt, Darran Nathan,ralf Clemens & Douglas Maskel.Using reconfigurable hardware to accelerate multiple sequence alignment with ClustalW.Bioinformatics Vol. 21 no. 16500, pages 3431-3132

[10] Xu Lin, Zhang Peiheng, Bu Dongbo, Feng Shengzhong, Sun Ninghui. To accelerate Multiple Sequence Alignment using FPGAs. National Research Center For Intelligence Computing Systems, Institute of Computing Technoligy, Chenese Academy of Science

[11] Azzadine Boukerche, Jan Mendonca Correa, Alba Christina Magalhaes Alves de Melo, Ricardo Pezzuol Jacobi & Andon Ferreira Rocha. An FPGA-Based Accelerator for Multiple Biological Sequence Alignment eith DIALIG

[12] Jaroslow Zola, Xiao Yang, Andrian Rospondek, Srinivas Aluru. Parallel-TCoffee: A parallel Multiple Sequence Aligner

[13] Feng, D.-F. & Doolittle, R. F. (1987). Progressive sequence alignment as a prerequisite to correct phylogenetic trees. J. Mol. Evol. 25, 351-360

[14] Smith, T. F. & Waterman, M. S. (1981). Identi®cation of common molecular subsequences. J. Mol. Biol. 147, 195-197.

[15] Morgenstern, B. (1999). Dialign2: improvement of the segment-to-segment approach to multiple sequence alignment. Bioinformatics, 15, 211-218

[16] Neuwald, A. F., Liu, J. S., Lipman, D. J. & Lawrence, C. B. (1997). Extracting protein alignment models from the sequence database. Nucl. Acids Res. 25, 1665-1677

[17] Henikoff, S. & Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. Proc. Natl Acad.Sci. USA, 89, 10915-10919.

[18] Saitou, N. & Nei, M. (1987). The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol. Biol. Evol. 4, 406-125.