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

Υλοποίηση συστήματος βιοπληροφορικής αποτελούμενο από ηλεκτρονικό υπολογιστή και αναδιατασσόμενη Λογική για επιτάχυνση του Αλγορίθμου GlimmerHMM

Χρυσάνθου Ναυσικά

Διπλωματική Εργασία

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

Χανιά 2010

Ευχαριστίες

Θα ήθελα να ευχαριστήσω πρώτα απ' όλα τον επιβλέποντα καθηγητή μου κ. Παπαευσταθίου Ιωάννη για την εποικοδομητική συνεργασία σε ότι αφορά τη διπλωματική μου. Επίσης, θα ήθελα να ευχαριστήσω τον Καθηγητή κ. Δόλλα Απόστολο και τον *Αναπληρωτή Καθηγητή κ. Πνευματικάτο Διονύσιο,* για τη συμμετοχή τους ως μέλη της εξεταστικής επιτροπής.

Ένα μεγάλο ευχαριστώ, στον διδακτορικό φοιτητή Γρηγόρη Χρυσό, για την πολύ μεγάλη βοήθεια του και την καθοδήγηση του σε κάθε βήμα της πορείας από την αρχή μέχρι και την ολοκλήρωση αυτής της εργασίας. Επίσης θέλω να ευχαριστήσω τον διδακτορικό φοιτητή Ευριπίδη Σωτηριάδη, αλλά και τον πτυχιούχο πλέον Γεωργιάδη Νικόδημο για τη συνεργασία μας.

Τις ευχαριστίες μου θα ήθελα να εκφράσω και στον υπεύθυνο του εργαστηρίου Μικροεπεξεργαστών και Υλικού κ. Μάρκο Κιμιωνή, καθώς και σε όλα τα μέλη του εργαστηρίου για την βοήθεια που μου παρείχαν και για το άψογο κλίμα που υπήρχε στις μεταξύ μας σχέσεις.

Για τη φιλία και τη στήριξη τους θα ήθελα να ευχαριστήσω και όλους τους φίλους και συναδέλφους που ήταν δίπλα μου.

Πάνω απ' όλους και απ' όλα όμως θα ήθελα κυρίως να ευχαριστήσω την μαμά μου Μαρία και την αδερφή μου Κυπρούλα, που είναι πάντα δίπλα μου και με στηρίζουν στα όνειρα και τις αποφάσεις μου σε όλες τις φάσεις της ζωής μου.

Περίληψη

Οι μοριακές ακολουθίες συντίθενται από έναν αριθμό από στοιχεία τα οποία ονομάζονται βάσεις. Οι ακολουθίες DNA αποτελούνται από τέσσερα είδη νουκλεϊκών οξέων, ενώ οι πρωτεϊνικές ακολουθίες από είκοσι διαφορετικά είδη αμινοξέων. Στην εξελικτική διαδικασία οι μοριακές αυτές ακολουθίες υφίσταται μεταλλάξεις όπως αντιγραφή, διαγραφή και αντικατάσταση. Αυτά τα γεγονότα μετάλλαξης έχουν ως αποτέλεσμα την ποικιλόμορφη διαφοροποίηση ανάμεσα στις ακολουθίες. Προκειμένου να βρεθούν ποιά μέρη της ακολουθίας είναι όμοια, γίνεται η εύρεση γονιδίων.

Η εύρεση γονιδίων 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 εύρεσης γονιδίων έχουν αναπτυχθεί διάφοροι αλγόριθμοι με στόχο το αποτέλεσμα που προκύπτει να έχει βιολογική σημασία.

Στα πλαίσια της παρούσας διπλωματικής εργασίας επιλέχθηκε ο αλγόριθμος GlimmerHMM ο οποίος μελετήθηκε και στην συνέχεια σχεδιάστηκε και υλοποιήθηκε σε αναδιατασσόμενη λογική ένα βασικό μέρος του επιδιώκοντας κάποιο κέρδος όσο αφορά το χρόνο εκτέλεσης του. Στη συνέχεια με τη χρήση του οδηγού του PCI-express υλοποιήθηκε ένα ολοκληρωμένο σύστημα που υλοποιεί τον αλγόριθμο.

Περιεχόμενα

Κατάλογος σχημάτων…….. 9

Κατάλογος πινάκων…….. 11

Γλωσσάρι…….. 13

1. *Εισαγωγή…… 15*

1.1 *Επεξήγηση νουκλεϊκών οξέων…… 15*

1.2 *Τι είναι η εύρεση γονιδίων (gene finding)……. 16*

1.3 *Σε τι μας χρησιμεύει η εύρεση γονιδίων (gene finding)…… 19*

2. Σχετική έρευνα…… 21

2.1 *Hidden Markov model (HMM)…… 21*

2.2 *Αλγόριθμοι εύρεσης γονιδίου (gene finding)…… 23*

2.2-1 Αλγόριθμος HMMgene…… 23

2.2-2 Αλγόριθμος TWINSCAN…… 24

2.2-3 Αλγόριθμος GenScan…….. 24

2.2-4 Αλγόριθμος GENIE…… 26

2.2-5 Αλγόριθμος FGENESV…… 27

2.2-6 *Αλγόριθμος* Glimmer……. 28

2.3 Αλγόριθμοι και FPGA's…….. 29

2.3-1 Αλγόριθμος HMMsearch σε αναδιατασσόμενη λογική…… 31

2.3-2 Αλγόριθμος ClawSearch σε αναδιατασσόμενη λογική…… 31

2.3-3 Αλγόριθμος Glimmer σε αναδιατασσόμενη λογική……. 32

2.4 Σύγκριση αλγορίθμων…… 32

3. Αλγόριθμος GlimmerHMM και μοντελοποίηση…… 35

3.1 Αλγόριθμος GlimmerHMM …….. 36

3.2 Περιγραφή αλγορίθμου GlimmerHMM……. 39

3.3 Μοντελοποίηση Αλγορίθμου GlimmerHMM…… 39

3.3-1 Ανάλυση Software Αλγορίθμου GlimmerHMM…… 40

3.3-2 Μοντελοποίηση Συνάρτησης get_prob_of_window1……. 43

4. Αρχιτεκτονική Σχεδίασης……. 47

4.1 Περιγραφή της σχεδίασης και της υλοποίησης της συνάρτησης get_prob_of_window1…….. 47

4.1-1 Περιγραφή της σχεδίασης και της υλοποίησης της συνάρτησης ISE.. 48

4.1-2 Περιγραφή συστήματος ……. 53

4.2 Περιγραφή της σχεδίασης και της υλοποίησης της συνάρτησης EDK.. 55

5. Αποτελέσματα-Μετρήσεις…… 63

5.1 Ταυτοποίηση λειτουργίας …… 63

5.2 Απόδοση συστήματος…… 63

5.3 Αποτίμηση Απόδοσης……. 64

6. Συμπεράσματα και Μελλοντικές Επεκτάσεις…….. 69
6.1 Μελλοντικές επεκτάσεις ……. 69

7.* Παραπομπές ……. *71

Κατάλογος σχημάτων

1.1-1: Η δομή των νουκλεϊκών οξέων…….. 15

1.1-2: *Λεπτομερής περιγραφή (a) του DNA, (b) διπλή έλικα DNA.……* *16*

1.2-1*:* *Πρωτεϊνική σύνθεση γονιδίου……* *17*

1.2-2*:* *Αναλυτική σύγκριση πολλών ειδών……* *18*

1.2-3: *Γονιδιακή δομή ευκαρυωτικών (α) και προκαρυωτικών (b) γονιδίων……* *19*

2.1-1: *Σύστημα HMM που δείχνει μία ακολουθία καταστάσεων, η οποία δημιουργεί μία ακολουθία παρατηρήσεων……* *22*

2.1-2: *HMM για unspliced γονίδια……* *22*

2.1-3: *Τρία διαφορετικά πλαίσια splicing με τη χρήση τριών ιντρονίων……* *22*

2.2-3: *HMM στον αλγόριθμο GenScan……* *25*

2.2-2: *Δυσδιάστατη αναπαράσταση του Clustering χρησιμοποιώντας τον αλγόριθμο Genie……* *27*

2.2-3: *(a) Απόδοση διαφόρων αλγορίθμων εύρεσης γονιδίων έχοντας σαν είσοδο τη γονιδιακή ακολουθία ρυζιού, (b) Απόδοση τρων ιδίων αλγορίθμων σε 42 σχεδόν τεχνικές γονιδιακές ακολουθίες περιέχοντας 178 γνωστές γονιδιακές ακολουθίες ανθρώπου …… ……* *28*

2.2-4*:* *Υποψήφια γονίδια (start/ stop codons)……* *29*

2.2-5: *Αναπαράσταση του Glimmer score……* *29*

2.3-1: Πλακέτα FPGA……. 30

2.3-2 *Εύρεση γονιδίου χρησιμοποιώντας FPGA……* *31*

3.1-1: *Διάγραμμα του GHMM για GlimmerHMM……* *36*

3.2-1: *Δέντρο μοντελοποίησης αλγορίθμου GlimmerHMM……* *37*

3.2-2: *Επικάλυψη υποψήφιων γονιδίων……* *38*

3.2-3: *Flow chart του αλγορίθμου GlimmerHMM.……* 39

3.3.1-1: *Ενδεικτική μορφή αρχείου εισόδου FASTA στο σύστημα……* *42*

3.3.1-2: *Ενδεικτική έξοδος αλγορίθμου GlimmerHMM……* *42*

3.3-4: *Flow chart συνάρτησης get_prob_of_window1……* *45*

4.1-1: *Block diagram* του περιεχομένου του κάθε δένδρου…… 49

4.1-2 *: Τα δύο πρώτα επίπεδα της αρχιτεκτονικής μας…….* *51*

4.1-3: *Τα δύο τελευταία επίπεδα της αρχιτεκτονικής μας……* *52*

4.1-4: *Block diagram της αρχιτεκτονικής μας……* *54*

4.1-5: Block diagram της σχεδίασης μας……. 56

4.1-5: *Μονάδα ελέχγου της δικιάς μας FIFO……* *57*

4.2-1: *Block diagram του συστήματος μας……* *59*

4.2-2: *Μονάδα ελέγχου του συστήματος (control_my_black_box)……* *60*

4.2-3: *Block diagram αρχιτεκτονικής συστήματος……* *61*

Κατάλογος πινάκων

2.3.3-1: Αποτελέσματα του Glimmer αλγορίθμου SW-HW **…….. 32**

2.3.3-2: Αποτελέσματα του Glimmer αλγορίθμου SW vs. HW-SW co-design…….. 32

2.4-1: Τύποι αλγορίθμων εύρεσης γονιδίου σε DNA ακολουθίες……. 33

3.3-1: *Απεικόνιση αποτελεσμάτων συνάρτησης με τη χρήση του V-Tune Performance Analyzer…….. 40*

3.3-2: *Απεικόνιση εισόδων…… 41*

3.3-3: *Απεικόνιση αποτελεσμάτων χρόνου στο software…… 41*

4.2-1: *Σήματα εισόδου my_black_box…….. 58*

4.2-2: *Σήματα εξόδου my_black_box…… 58*

5.2-1: *Χρήση πόρων FPGA αρχιτεκτονική σχεδίασης……* 64

5.2-2: *Χρήση πόρων FPGA στην υλοποίηση του συστήματος……. 64*

5.3-1: *Συχνότητα της σχεδίασης και του συστήματος μας…… 64*

5.3-2: *Απεικόνιση εισόδων…… 65*

5.3-3: *Χρόνος και speedup εκτέλεσης συνάρτησης ανά βάση……. 65*

5.3-4: Χρόνος και speedup εκτέλεσης συνάρτησης ……. 66

5.3-5: *Χρόνος και speedup εκτέλεσης αλγορίθμου……. 66*

Γλωσσάρι

Coding region: Στη γενετική coding περιοχές, είτε αναφερόμαστε σε DNA είτε σε RNA, είναι οι περιοχές του οργανισμού οι οποίες κωδικοποιούνται σε πρωτεϊνικές ακολουθίες. Οι διάφορες λειτουργίες που παρατηρούνται στους οργανισμούς γίνονται χάρη στις πρωτεΐνες.

Codon: *Ο γενετικός κώδικας αποτελείται από 64 τριπλέτες νουκλεοτιδίων. Η κάθε τριπλέτα ονομάζεται codon.*

Clustering*: μπορεί να θεωρηθεί από τα πιο σημαντικά μαθησιακά προβλήματα. Εξετάζει την εύρεση μιας δομής σε μια συλλογή από σκορπισμένα στοιχεία.*

CUPS: Ενημερώσεις κυττάρων ανά δευτερόλεπτο.

Exon *ή* ***εξάνια****: είναι ένα μέρος του RNA που δεν κωδικοποιεί εντολές για σύνθεση πρωτεϊνών.*

FASTA: *στη βιοπληροφορική, FASTA format (γνωστό και σαν Pearson format) είναι ένα κείμενο το οποίο εμπεριέχει είτε νουκλεοτιδικές ή πεπτικές ακολουθίες, στις οποίες ζεύγη βάσεων ή αμινοξέων εκπροσωπούνται από ένα κωδικοποιημένο γράμμα. Επίσης, η μορφή επιτρέπει ονόματα ακολουθιών και σχόλια πάνω σε αυτές.*

Forward and reverse strand: Με τη διπλή αλυσίδα DNA υπάρχουνε έξι πιθανά πλαίσια ανάγνωσης (reading frames - codons). Τρία με προσανατολισμό προς τα εμπρός για ένα από τα σκέλη του DNA και άλλα τρία που είναι αντίστροφα στο αντίθετο σκέλος του.

Intergenic*: μια intergenic περιοχή (**IGR**) είναι μια έκταση από DNA ακολουθίες που βρίσκονται μεταξύ γονιδίων και περιέχουν λίγα ή και καθόλου γονίδια.*

Interrupts: η συσκευή μας ειδοποιεί όταν είναι έτοιμη, ενώ το πρόγραμμά μας θα μπορεί να συνεχίζει να κάνει κάτι άλλο στο ενδιάμεσο.

Intron *ή* ***ιντρόνια****: είναι ένα μέρος του DNA που δεν κωδικοποιεί εντολές για σύνθεση πρωτεϊνών.*

Lookup table: είναι ένας πίνακας αναζήτησης που χρησιμοποιείται συχνά για να αντικαταστήσει τον παλιό πίνακα με έναν απλουστευμένο πίνακα. Η εξοικονόμηση από την άποψη της επεξεργασίας του χρόνου μπορεί να είναι σημαντική, δεδομένου ότι ανακτώντας μια τιμή από τη μνήμη είναι συχνά πιο γρήγορο από την έκθεσή του σε ακριβούς υπολογισμούς ή με τη λειτουργία εισόδου / εξόδου.

Μονάδα μέτρησης nt: nt είναι μια κοινή μονάδα μήκους για τα μονής αλυσίδας νουκλεοτίδια (RNA).

Non-coding region: Στη γενετική non-coding περιοχές, είτε αναφερόμαστε σε DNA είτε σε RNA, είναι οι περιοχές του οργανισμού οι οποίες δεν κωδικοποιούνται σε πρωτεινικές ακολουθίες.

Poly-A**:* **Polyadenylation** *είναι η συμπληρωμένη λέξη που αναφέρεται στο* ***poly (A) tail *σε ένα RNA μόριο. Το poly(A) tail αποτελείται μόνο από βάσεις Αδενίνης.*

Polling: στο polling το πρόγραμμά μας πρέπει να ρωτάει συνέχεια τη συσκευή με την οποία επικοινωνεί αν είναι διαθέσιμη.

Pribnow: *επίσης γνωστό και σαν* ***Pribnow-Schaller box,*** *όπου είναι μια ακολουθία TATAAT 6 νουκλεοτιδίων.*

Promoter: *Είναι μια περιοχή του DNA που διευκολύνει τη σύνθεση του RNA ενός συγκεκριμένου γονιδίου.*

***ORF**:* *Στη μοριακή γενετική, ένα ανοικτό πλαίσιο ανάγνωσης (ORF) είναι μια αλληλουχία DNA που θα μπορούσε να κωδικοποιήσει μια πρωτεΐνη, επειδή στερείται από τερματικά codons. Τα σημεία εκκίνησης και τα τελικά σημεία μιας συγκεκριμένης ORF δεν είναι ισοδύναμα προς τα άκρα του αγγελιοφόρου RNA (mRNA), αλλά τα άκρα του ORF συνήθως περιέχονται σε ακολουθία του mRNA του. Σε ένα γονίδιο, ORFs βρίσκονται μεταξύ της ακολουθίας κωδικού εκκίνησης (initiation codon) και της ακολουθίας τερματικού κωδικού (termination codon). Τα ORFs συνήθως αποκωδικοποιούν κατά τη μετακίνηση τους μέσα από τα κομμάτια του DNA, ενώ προσπαθούν να εντοπίσουν ένα γονίδιο. Δεδομένου ότι υπάρχουν διαφορές στην ακολουθία εκκίνησης κώδικα των οργανισμών με τροποποίηση γενετικό κώδικα, το ORF θα προσδιοριστεί με διαφορετικό τρόπο. Ένα τυπικό ανιχνευτικό ORF απασχολεί αλγορίθμους με βάση τους υπάρχοντες γενετικούς κώδικες, καθώς και όλα τα πιθανά πλαίσια ανάγνωσης.*

***Splice sites**:* *στη μοριακή βιολογία splicing είναι η τροποποίηση του RNA μετά τη μεταγραφή (transcription), στην οποία τα ιντρόνια αφαιρούνται και τα εξάνια προστίθενται.*

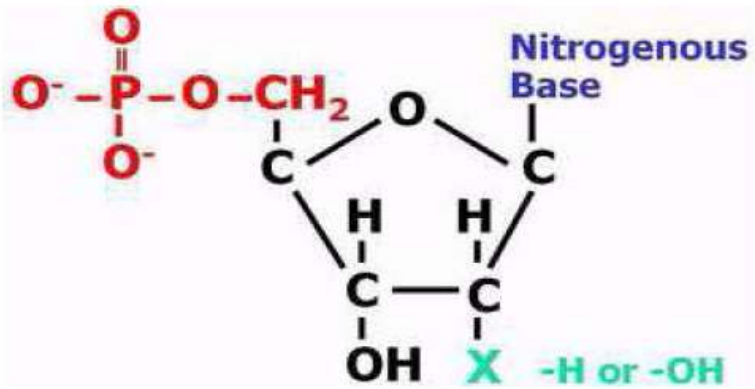
Κεφάλαιο 1

Εισαγωγή

Στο πρώτο κεφάλαιο της παρούσας διπλωματικής εργασίας γίνεται εισαγωγή στο θέμα της εύρεσης γονιδίων (gene finding) καθώς και τι είναι και για ποιους λόγους χρησιμοποιείται η εύρεση.

1.1 Επεξήγηση νουκλεϊκών οξέων

Το **DNA** και το **RNA** είναι τα δύο είδη νουκλεονικών οξέων που σχετίζονται πολύ στενά με την γενετική πληροφορία. Είναι δηλαδή, τα στοιχεία εκείνα που έχουν να κάνουν με την κληρονομικότητα και τα ιδιαίτερα χαρακτηριστικά κάθε ατόμου. Η δομική μονάδα των νουκλεονικών οξέων είναι τα νουκλεοτίδια τα οποία αποτελούνται από μια πεντόζη που είναι ενωμένη με μία φωσφορική ομάδα και μία αζωτούχο βάση (Σχήμα 1.1-1). Στο μόνο που διαφέρει το ένα νουκλεοτίδιο με το άλλο είναι η αζωτούχα βάση με την οποία είναι συνδεδεμένο. Στα νουκλεοτίδια του DNA η αζωτούχος βάση μπορεί να είναι: Αδενίνη (A - Adenine), Γουανίνη (G - Guanine), Θυμίνη (T - Thymine), Κυτοσίνη (C - Cytosine). Στο RNA μπορεί να συναντήσει κανείς όλες τις προηγούμενες, έκτος από τη Θυμίνη που τη θέση της καταλαμβάνει η Ουρακίλη (U - Uracil) .

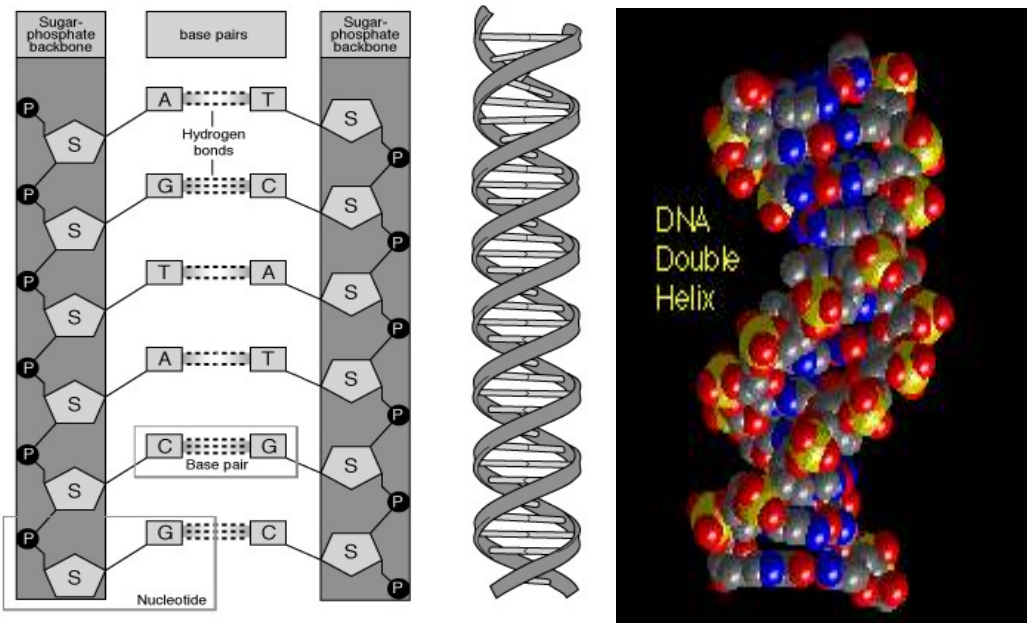


Σχήμα 1.1-1: Η δομή των νουκλεϊκών οξέων

Τα νουκλεοτίδια συνδέονται μεταξύ τους με ομοιοπολικό δεσμό για να σχηματίσουν το μακρομόριο και μπορούν να εμφανίζονται στην αλυσίδα του DNA με οποιαδήποτε σειρά. Το σύμπλεγμα του σακχάρου και των φωσφορικών οξέων αποτελεί τη ραχοκοκαλιά του μορίου του DNA. Επειδή μία μόνο αλυσίδα του μορίου του DNA είναι αρκετά εύθραυστη συνήθως το DNA εμφανίζεται να σχηματίζει διπλές αλυσίδες. Ο κανόνας για τον σχηματισμό των αλυσίδων αυτών είναι η τήρηση της συμπληρωματικότητας των βάσεων. Τα επιτρεπτά ζεύγη είναι A-T και C-G. Στην περίπτωση σχηματισμού συμπληρωματικής αλυσίδας RNA, οι συνδυασμοί αυτοί γίνονται A-U και C-G, καθώς όπου

υπάρχει θυμίνη στο DNA υπάρχει ουρακίλη στο RNA. Η συμπληρωματικότητα είναι το χαρακτηριστικό εκείνο του DNA που του παρέχει τη δυνατότητα του αυτοδιπλασιασμού (αντιγραφή του DNA).

Μια ακόμα διαφορά στη δομή των DNA και RNA είναι ότι ενώ το RNA αποτελείται από μόνο μια αλυσίδα νουκλεοτιδίων, το DNA είναι δίκλωνο (Σχήμα 1.1-2 (α) και Σχήμα 1.1-2(b)). Οι σημαντικότερες διαδικασίες που παρατηρούνται στο DNA είναι η αντιγραφή, η μεταγραφή, η μετάφραση και η μετάλλαξη. Το DNA αποτελεί το γενετικό υλικό όλων των κυττάρων και των περισσοτέρων ιών. Το γενετικό υλικό ενός κυττάρου αποτελεί το γονιδίωμα ή αλλιώς γένωμα του. Μια σειρά νουκλεοτιδίων του DNA αποτελούν ένα γονίδιο. Τα γονίδια περιέχουν τον κωδικό για την παραγωγή των πρωτεϊνών, που είναι τα μακρομόρια εκείνα που διεκπεραιώνουν τις περισσότερες εργασίες στο κύτταρο.



Σχήμα 1.1-2: Λεπτομερής περιγραφή του DNA (α), διπλή έλικα DNA (b)

1.2 Τι είναι η εύρεση γονιδίων (gene finding)

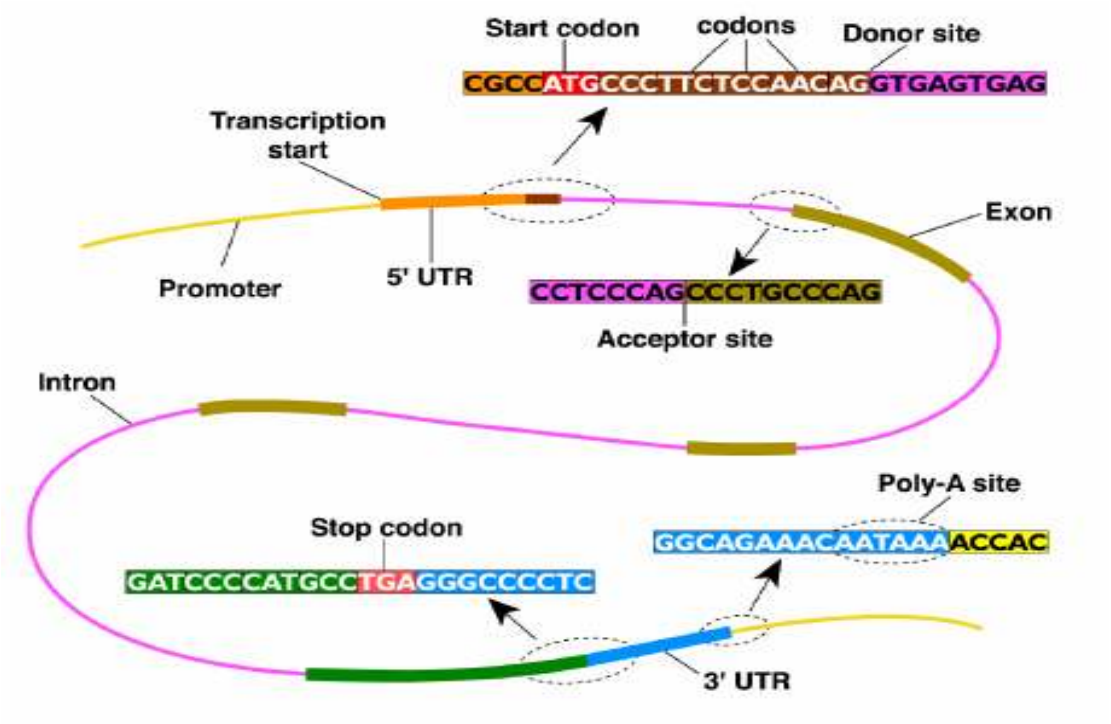
Η εύρεση γονιδίων αναφέρεται στον προσδιορισμό αναπτυγμάτων των γονιδιακών ακολουθιών στο DNA, οι οποίες είναι βιολογικά λειτουργικές. Η εύρεση γονιδίων είναι κρίσιμη στη κατανόηση του γονιδιώματος ενός είδους. Το σχήμα 1.2-1 παρουσιάζει τη δομή μιας πρωτεϊνικής σύνθεσης γονιδίου σε ευκαριωτικά.

Ο γονιδιακός προσδιορισμός είναι πολύ σημαντικός έχοντας αυξανόμενο ποσοστό ερευνών στο γονιδίωμα όλων των διαφορετικών οργανισμών. Είναι χαρακτηριστικό ότι τα τελευταία χρόνια έχουν εμφανιστεί πολλά νέα και αρχαία βακτηριακά γονιδιώματα. Ένα από τα κύρια προβλήματα για τους βιολόγους είναι ο χωρισμός των γενετικών πληροφοριών από το γονιδίωμα του οργανισμού. Οι βιολόγοι ενδιαφέρονται πολύ για τη συγκεκριμένη πληροφορία, καθώς αποτελεί τη βάση για περαιτέρω έρευνα. Υπάρχουν πολλές διαφορετικές μέθοδοι, όπως η μικρό-ακολουθιακή ανάλυση, η οποία ή δεν έχει υψηλή ακρίβεια, ή ο αναγκαίος χρόνος επεξεργασίας είναι τεράστιος.

Έχουνε προταθεί πολλές προσεγγίσεις για την εύρεση είτε ευκαρυωτικών, είτε προκαρυωτικών γονιδίων. Αυτές οι προσεγγίσεις εμπίπτουν κυρίως στις ακόλουθες τρείς κατηγορίες: προσεγγίσεις βασισμένες στις ομοιότητες (homology-based approaches), συγκριτικές γονιδιακές προσεγγίσεις (comparative genomics approaches), και τέλος Ab Initio προσεγγίσεις (όπου ***ab initio*** είναι λατινικός όρος και σημαίνει από την αρχή). Παρακάτω γίνεται μια σύντομη περιγραφή των τριών κατηγοριών εύρεσης γονιδίων.

1. Προσεγγίσεις βασισμένες στις ομοιότητες (homology-based approaches)

Η πρώτη κατηγορία αφορά προσεγγίσεις οι οποίες είναι βασισμένες στις ομοιότητες. Λαμβάνοντας υπόψη μια βιβλιοθήκη από ακολουθίες άλλων οργανισμών, ψάχνουμε την ακολουθία που μας ενδιαφέρει σε αυτή τη βιβλιοθήκη και αναγνωρίζουμε μια βιβλιοθήκη ακολουθιών (γνωστά γονίδια) η οποία είναι όμοια με αυτή. Επίσης, μπορούμε να συγκρίνουμε τη δεδομένη ακολουθία με σαφής ακολουθία ετικετών (expressed sequence tags - ETSs) του ιδίου οργανισμού για να ταυτίσουμε περιοχές που αντιστοιχούν στο επεξεργασμένο mRNA. Εάν οι αναγνωρισμένες ακολουθίες είναι γονίδια τότε πιθανώς η δεδομένη ακολουθία να είναι ένα γονίδιο. Αυτού του τύπου προσεγγίσεις βρίσκουν βιολογικά σχετικά γονίδια. Εντούτοις, δε μπορούν να αναγνωρίσουν γονίδια τα οποία είναι κωδικοποιημένα για πρωτεΐνες, οι οποίες δεν βρίσκονται ακόμα στη βιβλιοθήκη. Ο BLAST (Basic Local Alignment Search Tool) είναι ένα πολύ γνωστό εργαλείο αναζήτησης, που ανήκει σε αυτή τη κατηγορία.



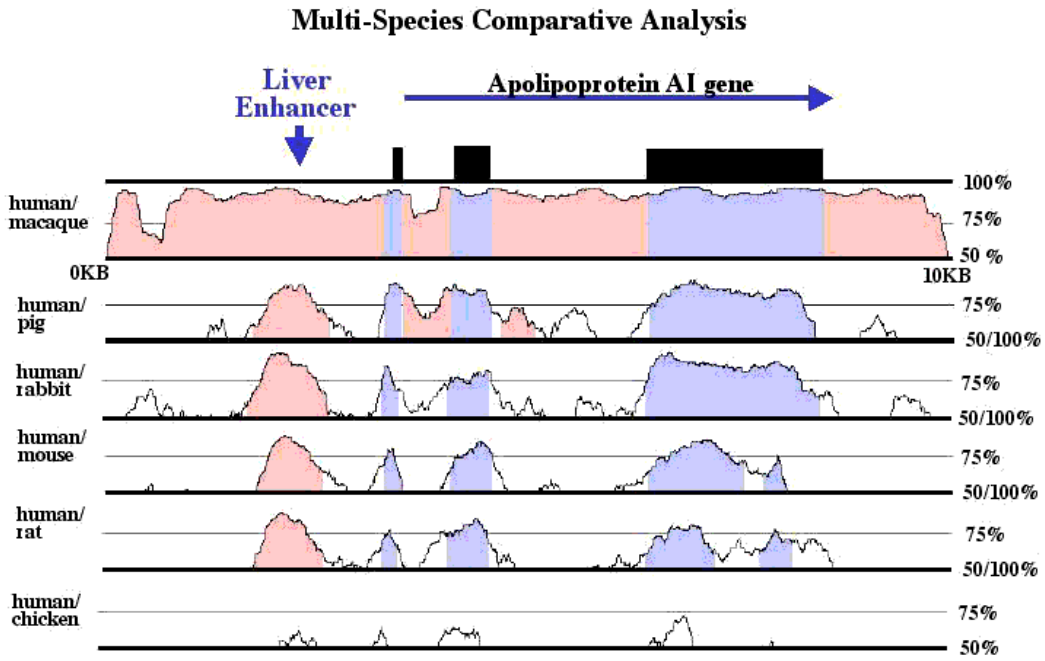
Σχήμα 1.2-1: Πρωτεϊνική σύνθεση γονιδίου

2. Συγκριτικές γονιδιακές προσεγγίσεις (comparative genomics approaches)

Οι συγκριτικές γονιδιακές προσεγγίσεις για την εύρεση γονιδίων είναι βασισμένες στην παρατήρηση που η δύναμη της φυσικής επιλογής κάνει τα γονίδια και άλλα λειτουργικά στοιχεία να αλλοιώνονται σε πιο αργό ποσοστό από άλλα μέρη ενός γονιδιώματος. Με περισσότερες γονιδιακές ακολουθίες σε σχετικά είδη, τα γονίδια μπορούν να προσδιοριστούν με τη σύγκριση αυτών των γονιδίων για να ανιχνεύσουν αυτή τη συντήρηση (σχήμα 1.2-2).

3. Ab Initio προσεγγίσεις

Σε αυτή την κατηγορία το ***Ab initio*** γονίδιο έρευνας, ψάχνει για συγκεκριμένα σήματα από κωδικοποιημένες πρωτεΐνες γονιδίων. Υπάρχουν δύο τύποι οργανισμών: Προκαρυωτικοί και ευκαρυωτικοί. Οι *προκαρυωτικοί* (σχήμα 1.2-3(b)) έχουν μικρά γονιδιώματα (0.5 ~ 10,000,000 bp) και υψηλή πυκνότητα σύνθεσης (>90%). Στα προκαρυωτικά δεν υπάρχουν ***ιντρόνια***. Η εύρεση γονιδίου για τα προκαρυωτικά είναι σχετικά εύκολη δεδομένου ότι τα προκαρυωτικά γονίδια έχουν συγκεκριμένα σήματα όπως ο μεταγραφικός συντελεστής δεσμευτικού χώρου και το κιβώτιο **Pribnow** τα οποία είναι διευκολύνουν στην αναγνώριση. Επιπλέον, κωδικοποιημένη πρωτεϊνική ακολουθία είναι ένα παρακείμενο ανοικτό πλαίσιο αναγνώρισης (ORF), αρχίζοντας από το αρχικό **codon** (ATG) και τερματίζοντας με το τελικό **codon** (TAG/TGA/TAA).

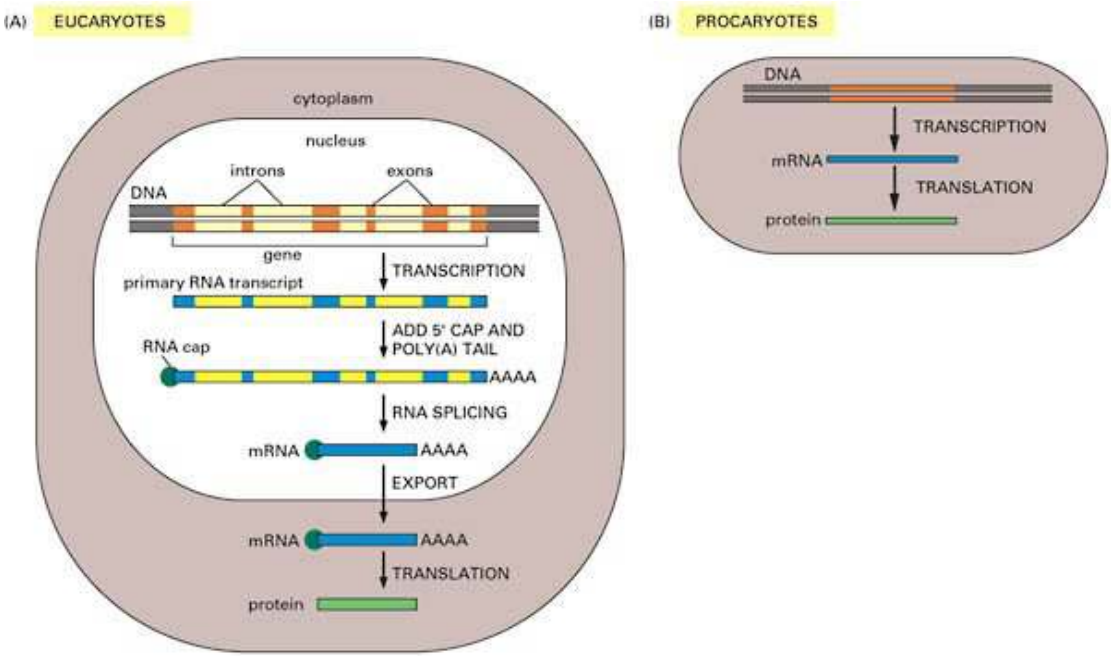


Σχήμα 1.2-2: Αναλυτική σύγκριση πολλών ειδών

Εντούτοις, για τα *ευκαριωτικά* γονίδια (σχήμα 1.2-3(α)), το Ab Initio γονίδιο έρευνας είναι δυσκολότερο για τους εξής λόγους. Κατ 'αρχάς, τα γονίδια χωρίζονται από μεγάλες

intergenic περιοχές. Δεύτερον, ένα γονίδιο δεν είναι συνεχόμενο. Το γονίδιο είναι χωρισμένο σε **εξάνια** και **s** με μηχανισμούς σύνδεσης μέσα στο ευκαρυωτικό κύτταρο. Λόγο αυτής της διάσπασης καθιστάτε δύσκολο ο καθορισμός του ORF. Προφανώς από τα παραπάνω φαίνεται ότι τα σήματα είναι δυσκολότερο να προσδιοριστούν από ότι στα προκαρυωτικά, αφού αυτά τα σήματα είναι πιο σύνθετα και απροσδιόριστα.

Το HMM (Hidden Markov Model) έχει ευρέως χρήση για την εύρεση προκαρυωτικών και ευκαρυωτικών γονιδίων. Δίνοντας ένα γένωμα G μήκους L, το αποτέλεσμα του HMM που ακολουθεί το «κρυμμένο στάδιο μονοπατιού» S παράγει το παρατηρούμενο γονιδίωμα G χρησιμοποιώντας τον αλγόριθμο Viterbi. Η πιθανότητα από το «κρυμμένο στάδιο» ακολουθία S, δεδομένου του G υπολογίζεται από τον κανόνα του Bayes : ***P{S|G} =P{S,G}/Σ P{S',G},*** όπου το ***S'*** είναι ένα σύνολο όλων των κρυμμένων πιθανών μονοπατιών του μήκους L. Παραδείγματα αλγορίθμων οι οποίοι αντιστοιχούν σε αυτή την κατηγορία είναι οι: ***Glimmerhmm***, *GenScan (Chris Burge 1997), FGENESH (Solovyev 1997), HMMgene (Krogh, 1997), GENMARKER (Borodovsky &Mclninch 1993) και VEIL (Henderson, Salzberg & Fasman 1997).*



Σχήμα 1.2-3: Γονιδιακή δομή ευκαρυωτικών (α) και προκαρυωτικών(b)

1.3 Σε τι μας χρησιμεύει η εύρεση γονιδίων (gene finding)

Από τότε που ο J. D. Watson και ο F. H. C. Crick ανακάλυψαν την χημική δομή του DNA το 1953, η μοριακή βιολογία έχει σημειώσει σημαντική πρόοδο. Μεγάλος αριθμός δεδομένων μοριακών ακολουθιών εξάγονται από τα εργαστήρια και αποθηκεύονται σε βάσεις δεδομένων. Επιπλέον, μεγάλες ανθρώπινες ακολουθίες DNA έχουν παραχθεί κάτω από τις προσπάθειες του Human Genome Project το οποίο έχει ως σκοπό την παραγωγή ολόκληρης της ανθρώπινης ακολουθίας DNA η οποία υπολογίζεται περίπου στις τρία

εκατομμύρια βάσεις. Ο ρυθμός αύξησης της ποσότητας των βιολογικών ακολουθιών είναι ταχύς. Γι αυτό, υπάρχει μεγάλη ανάγκη να υπάρχουν αποδοτικοί αλγόριθμοι που θα αποσπούν και θα αναλύουν την υπάρχουσα πληροφορία από τις μοριακές ακολουθίες.

Το φαινόμενο της ζωής μπορεί να εξερευνηθεί διαμέσου του κωδικοποιημένου γενετικού υλικού, δεοξυριβονουλεϊκό οξύ (DNA). Αναφερόμενοι σε ακολουθίες, εννοούμε ακολουθίες νουκλεοτιδίων (που περιέχονται στο DNA ή RNA). Από έρευνες προκύπτει ότι δεν είναι όλα τα τμήματα μίας ακολουθίας ύψιστης σημασίας, αλλά μόνο τμήματα αυτής κουβαλούν γενετική πληροφορία, η οποία μπορεί να μεταφραστεί σε αμινοξέα πρωτεϊνικών ακολουθιών κάτω από ορισμένες συνθήκες. Από την άλλη για την ανακάλυψη του μυστηρίου της κυριαρχίας της ζωής οι άνθρωποι δεν διαβάζουν μόνο τις ακολουθίες DNA αλλά επιπλέον, τις αναλύουν με σκοπό να αποσπάσουν την λειτουργικότητα και την περίπτωση της ενεργοποίησης κάθε γονιδίου. Τότε τα αποτελέσματα της ανάλυσης αυτής μπορούν να χρησιμοποιηθούν για να βοηθήσουν τους ανθρώπους να προλαμβάνουν τις γενετικές αρρώστιες, την εκ γενετής εγκληματικότητα κτλ.

Η εύρεση γονιδίων έχει μεγάλη σημασία για τη βιολογική έρευνα. Είναι ένα από τα πρώτα και πιο σημαντικά βήματα για τη κατανόηση του γενώματος ενός είδους, εφόσον αυτό έχει τοποθετηθεί διαδοχικά. Αρχικά η εύρεση γονιδίων βασίστηκε στον προσεκτικό πειραματισμό πάνω σε ζωντανά κύτταρα και οργανισμούς. Στατιστική ανάλυση των ποσοστών ομόλογου επανασυνδυασμού διάφορων διαφορετικών γονιδίων μπορεί να καθορίσει τη κατάταξη τους σε ένα συγκεκριμένο χρωματόσωμα, και οι πληροφορίες από πολλά τέτοια πειράματα θα μπορούσαν να συνδυαστούν για να δημιουργήσουν έναν γενετικό χάρτη διευκρινίζοντας τη θέση των γνωστών γονιδίων στη μεταξύ τους σχέση. Έτσι γίνεται όλο και πιο πιθανό να προβλέψουμε τη λειτουργία ενός γονιδίου βασιζόμενοι μόνο στην ακολουθία που έχει.

Κεφάλαιο 2

Σχετική Έρευνα

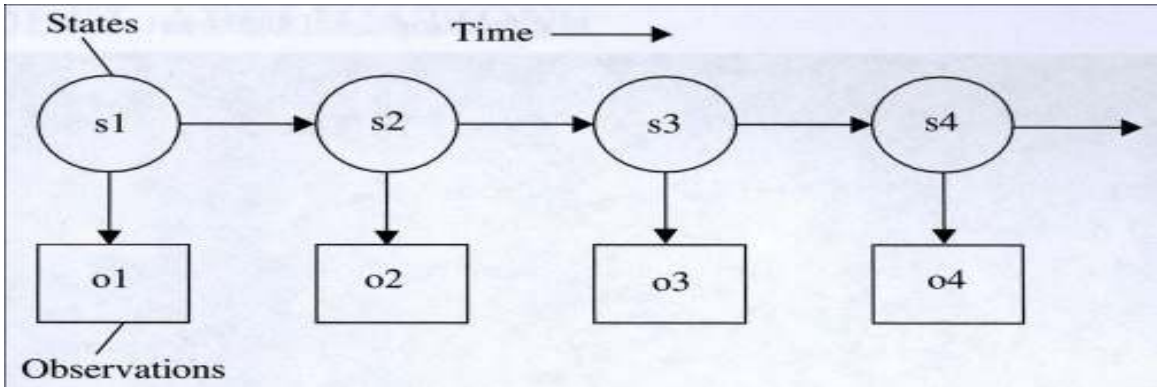
Η αναγνώριση γονιδίων είναι ένα από τα σημαντικότερα προβλήματα βιοπληροφορικής. Το γεγονός ότι όλο και περισσότερες ολοκληρωμένες γονιδιακές ακολουθίες είναι διαθέσιμες σε δημόσιες βάσεις δεδομένων, είναι ιδιαίτερα απαραίτητο για το σχολιασμό γονιδίου. Τα ζητήματα της εύρεσης γονιδίου έχουν αντιμετωπιστεί από πολλές ομάδες από τη δεκαετία του ’80. Από τότε, με τη βοήθεια που προσφέρεται μέσω υπολογιστών, έχει σημειωθεί μεγάλη πρόοδος. Λόγο αυτής της ραγδαίας προόδου που έχει σημειωθεί τα τελευταία χρόνια στα διάφορα προγράμματα αλληλουχιών, μας υποκινεί το ενδιαφέρον για την αναζήτηση καλών αλγορίθμων γονίδιο-εύρεσης.

Έτσι σε αυτό το κεφάλαιο παρουσιάζουμε ένα σύνολο αλγορίθμων γενικής βελτιστοποίησης (general optimization algorithms) που χρησιμοποιούνται ευρέως στην επιστήμη υπολογιστών και έχουν εφαρμοστεί για την επίλυση του προβλήματος της έρευνας γονιδίων (gene finding), 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, με διαφορετικό τρόπο. Κριτήριο επιλογής του καταλληλότερου αλγορίθμου για να επιτευχθεί η έρευνα γονιδίων θεωρείται 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. Εμείς θα ασχοληθούμε με αλγορίθμους που ανήκουν στη τρίτη κατηγορία εύρεσης γονιδίων (ab initio). Το κοινό χαρακτηριστικό όλων των παρακάτω αλγορίθμων είναι ότι βασίζονται στο hidden Markov μοντέλο (HMM). Παρακάτω γίνεται μια περιγραφή του μοντέλου και στη συνέχεια των αλγορίθμων.

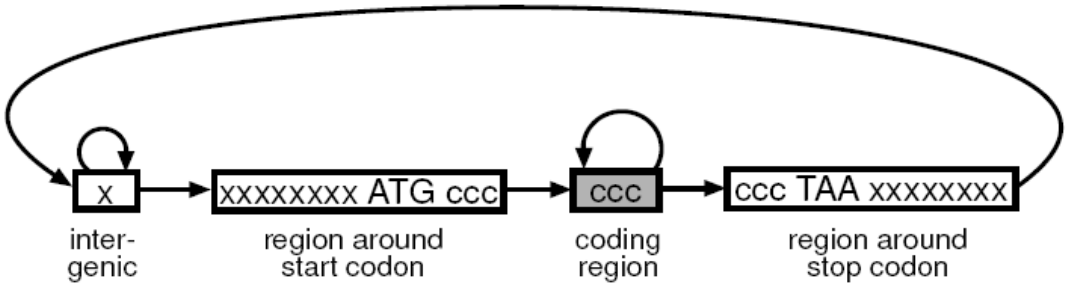
2.1 Hidden Markov model (HMM)

Το μοντέλο Markov (Hidden Markov Model - HMM) είναι ένα στατιστικό μοντέλο κατά το οποίο το σύστημα θεωρείται ως μια διαδικασία Markov, αλλά με άγνωστες παραμέτρους. Πιο συγκεκριμένα είναι μια πεπερασμένη σειρά καταστάσεων, καθεμία από τις οποίες σχετίζονται με μία (γενικά πολυδιάστατη) κατανομή πιθανοτήτων. Σε ένα κανονικό μοντέλο Markov, οι καταστάσεις είναι άμεσα ορατές από τον παρατηρητή, και ως εκ τούτου οι μόνες παράμετροι που μας χρειάζονται είναι οι πιθανότητες μετάβασης μεταξύ των καταστάσεων (state transition probabilities). Το σχήμα 2.1-1 δείχνει το διάγραμμα ροής για ένα HMM. Οι καταστάσεις καλούνται *κρυμμένες* (hidden), διότι η ακολουθία των καταστάσεων είναι άγνωστη, αλλά τα δεδομένα που παράγονται από τις καταστάσεις είναι γνωστά. Τα δεδομένα που μπορούν να παρατηρηθούν πρέπει να χρησιμοποιηθούν για να συμπεράνουμε την θέση και την έκταση των καταστάσεων στον χώρο των χαρακτηριστικών, καθώς και τις παραμέτρους για την κατανομή της αρχικής

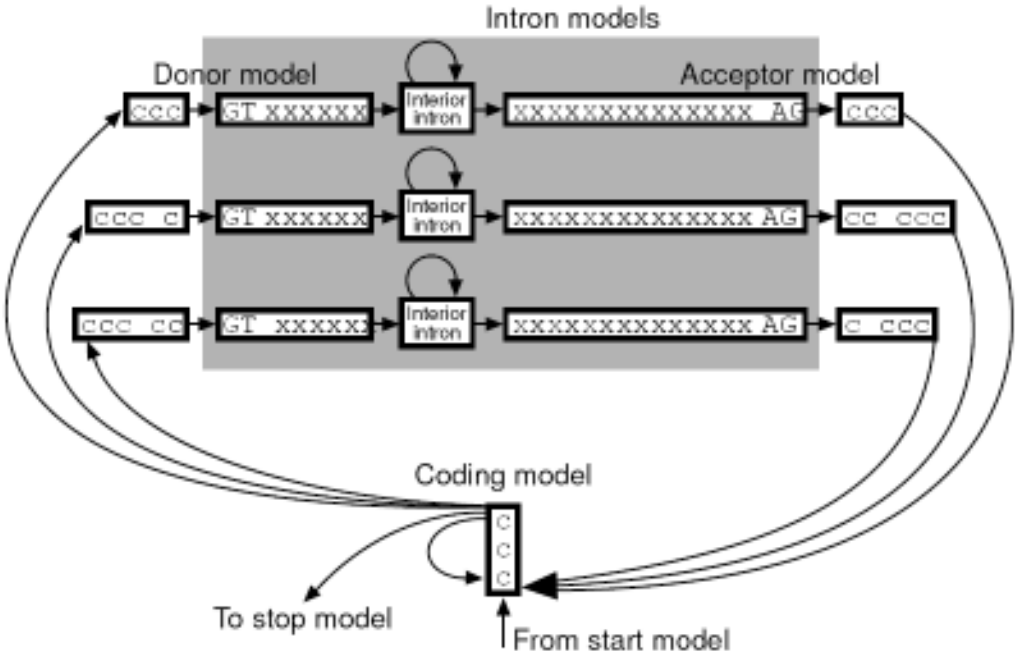
κατάστασης και των μεταβάσεων. Από τη στιγμή που θα γίνουν γνωστές οι παράμετροι, ένα HMM μπορεί να χρησιμοποιηθεί για να μετατρέψουμε μία ακολουθία από διανύσματα χαρακτηριστικών σε μία καταλληλότερη ακολουθία καταστάσεων.



Σχήμα 2.1-1: Ένα σύστημα HMM που δείχνει μία ακολουθία καταστάσεων και η οποία δημιουργεί μία ακολουθία παρατηρήσεων.



Σχήμα 2.1-2: HMM για uspliced γονίδια. 'x' είναι το στάδιο του non-coding DNA και 'c' το στάδιο του coding DNA.



Σχήμα 2.1-3: Για να επιτραπεί το splicing σε τρία διαφορετικά πλαίσια τρία ιντρόνια χρειάζονται.

Τα μοντέλα αποκτούνται από στατιστική ανάλυση των δεδομένων εκπαίδευσης. Η διαδικασία αυτή καλείται εξαγωγή μοντέλου (model inference). Το πρώτο, και πιο σημαντικό, βήμα στην εκπαίδευση ενός HMM είναι η συλλογή ενός αντιπροσωπευτικού συνόλου από υποδείγματα για μία δοθείσα κλάση γονιδίων, για παράδειγμα, ρύζι. Για καλύτερα αποτελέσματα, πρέπει να συλλεχθεί ένας αριθμός από παραδείγματα ακολουθιών από ρύζι που έχουν παρθεί σε διαφορετικά περιβάλλοντα και σε πολλές διαφορετικές συνθήκες. Στην ιδανική περίπτωση, το εκπαιδευτικό σετ θα πρέπει να περιέχει όλες τις διαφοροποιήσεις των γονιδιακών ακολουθιών. Ένα σύνολο από 100 γονίδια είναι ένας καλός οδηγός για την εκπαίδευση μιας γονιδιακής ακολουθίας. Εντούτοις, πολλοί τύποι γονιδιωμάτων παρουσιάζουν μεγαλύτερη διαφοροποίηση, όπως το γονιδίωμα του ανθρώπου και γι' αυτό το λόγο χρειάζονται περισσότερα δείγματα εκπαίδευσης ώστε να δημιουργηθεί ένα επιτυχές μοντέλο.

Κατά τη διάρκεια της εκπαίδευσης ενός HMM, πρέπει να εκτιμηθούν όλες οι παράμετροι, από τα διανύσματα χαρακτηριστικών του σετ εκπαίδευσης. Ειδικότερα, οι παράμετροι αυτές είναι ο αριθμός των καταστάσεων, το μέσο διάνυσμα και ο πίνακας συνδιασποράς για κάθε μία από τις καταστάσεις. Η κατανομή της αρχικής κατάστασης και ο πίνακας μετάβασης των καταστάσεων. Κατά συνθήκη, αυτές οι παράμετροι λαμβάνονται με τη χρήση του αλγορίθμου Baum – Welch. Η διαδικασία ξεκινά με τυχαίες αρχικές τιμές για όλες τις παραμέτρους και τις βελτιστοποιεί μέσω επαναληπτικής εκτίμησης. Κάθε επανάληψη τρέχει όλο το σύνολο των δεδομένων εκπαίδευσης, και για αυτό γίνεται συνεχώς μέχρι το μοντέλο να συγκλίνει σε ικανοποιητικές τιμές.

2.2 Αλγόριθμοι εύρεσης γονιδίου (gene finding)

*Σε αυτή την ενότητα περιγράφονται διάφοροι αλγόριθμοι εύρεσης γονιδίου που έχουν υλοποιηθεί σε software*Εργαλεία για την ανάλυση και σχολιασμό των γονιδιακών DNA ακολουθιών και μοντέλα οργανισμών χρησιμοποιούνται ευρέως για την αναγνώριση κωδικοποιημένων περιοχών (coding), έτσι ώστε να εξαχθεί η δομή των γονιδίων και των πρωτεϊνών που προκύπτουν.

2.2-1 Αλγόριθμος HMMgene:

Η μέθοδος μπορεί να προβλέψει όλη ή ένα μέρος της ακολουθία των γονιδίων και έτσι να μπορεί να χρησιμοποιηθεί σε μεγάλες ακολουθίες. Επίσης, ο HMMgene μπορεί να προβλέψει splice sites και start/stop codons. Εάν μερικά χαρακτηριστικά μιας ακολουθίας είναι γνωστά, όπως οι πρωτεΐνες ή επαναλαμβανόμενα στοιχεία κτλ., τότε αυτές οι περιοχές μετατρέπονται σε coding ή non-coding και τότε το πρόγραμμα θα βρει την καλύτερη δομή γονιδίων υπό το πλαίσιο αυτών των περιορισμών. Ο αλγόριθμος είναι βασισμένος σε Hidden Markov Model (HMM), το οποίο είναι ένα μοντέλο με πιθανότητες όσων αφορά τη γονιδιακή δομή. Αυτό σημαίνει πως η κάθε πρόβλεψη αντιστοιχεί σε μια πιθανότητα η οποία μας δείχνει πόσο αξιόπιστη είναι η πρόβλεψη αυτή μιας περιοχής ως γονίδιου. Ο HMMgene μπορεί να βρει τα N καλύτερα γονίδια για μια ακολουθία εισόδου. Αυτό είναι χρήσιμο στην περίπτωση που υπάρχουν αρκετές ίσες γονιδιακές δομές ώστε να μπορούμε να βρούμε και εναλλακτικό splicing. Σαν είσοδο δέχεται ένα αρχείο με μία η περισσότερες DNA ακολουθίες σε μορφή FASTA. Σαν έξοδο έχουμε τη πρόβλεψη ενός μέρους ή ολόκληρης της γονιδιακής ακολουθίας. Στην έξοδο εμπεριέχονται οι τοποθεσίες

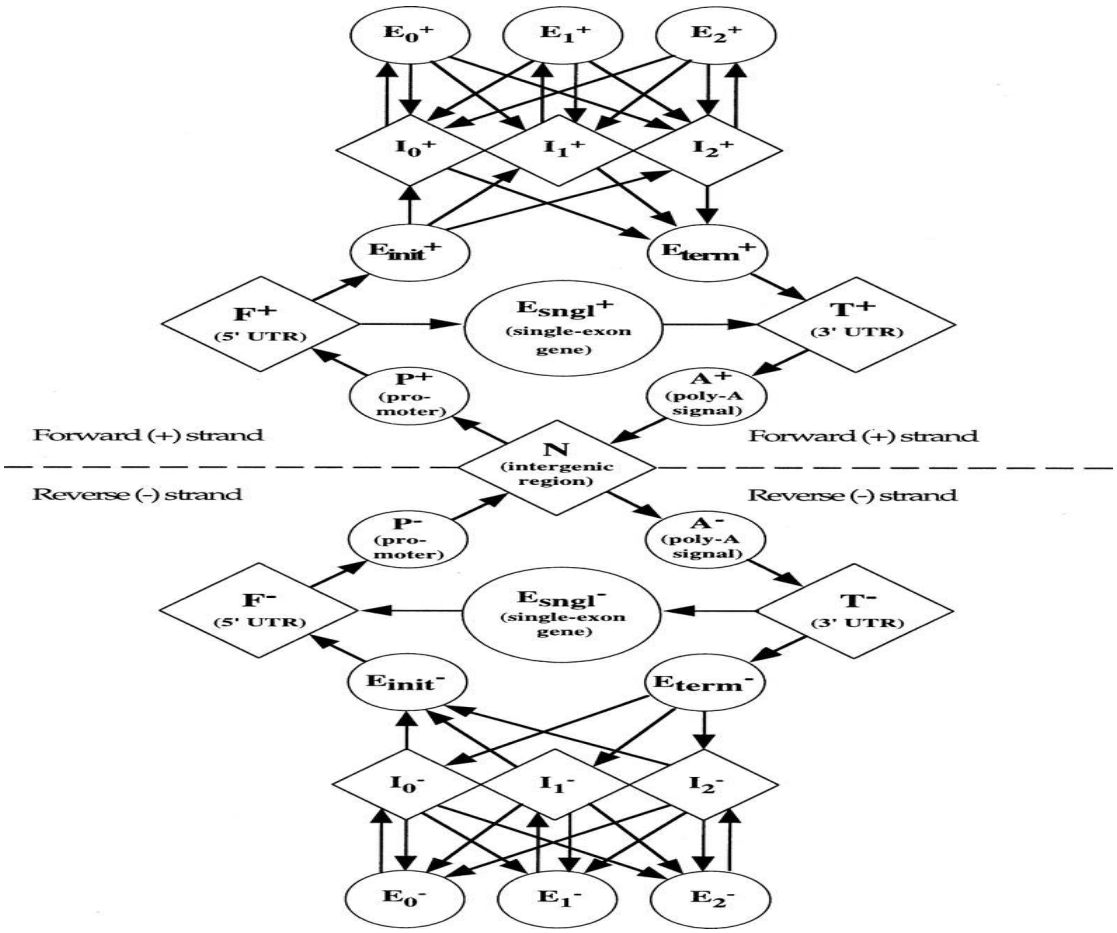
των προβλεπόμενων γονιδίων και οι coding τους περιοχές καθώς επίσης, τα score όλων των γονιδίων, αλλά και των εξανίων.

2.2-2 Αλγόριθμος TWINSCAN:

Το πρώτο βήμα του αλγορίθμου είναι η παραγωγή μιας υπάρχουσας ακολουθίας, η οποία αντικαθιστά τα νουκλεοτίδια μιας άλλης στοχευόμενης, με ένα από τα τρία σύμβολα δείχνοντας είτε ομοιότητα, ανομοιότητα ή μη ευθυγράμμιση καθώς συγκρίνεται με τη κορυφή τεσσάρων ομολόγων στη βάση της ακολουθίας από το σχετικό γονιδίωμα. Η πιθανότητα της ήδη υπάρχουσας ακολουθίας υπολογίζεται με ένα μοντέλο Markov, το οποίο έχει συγκεκριμένη σειρά με πιθανότητες καθορισμένες για τα τρία στάδια συμβόλων, αντί για το νουκλεοτιδικό «αλφάβητο». Ο αλγόριθμος αποδεικνύει επιτυχώς τη τιμή ανθρώπινου (κτλ) γενώματος για την ακριβή δημιουργία υπολογιστικών σχολιασμών. Όσο το αποτέλεσμα των σχολιασμών αποτελείται μόνο από 25,622 γονίδια ο αλγόριθμος θεωρείται λίγο καλύτερος από τον GENSCAN.

2.2-3 Αλγόριθμος GenScan:

Η μέθοδος GenScan θεωρείται από της πιο ακριβής αλγορίθμους όσον αφορά το γονιδίωμα του ανθρώπου. Το σχήμα 2.2-3 δείχνει το HMM για τον αλγόριθμο GenScan.Το κάθε διαμάντι ή κύκλος είναι ένα στάδιο της γονιδιακής περιοχής που αντιστοιχεί σε μια βασική λειτουργική μονάδα από ένα ευκαρυωτικό γονίδιο, π.χ., **intergenic** περιοχή, **εξάνια**, **ιντρόνια** κτλ.



Σχήμα 2.2-3: HMM στον αλγόριθμο GenScan

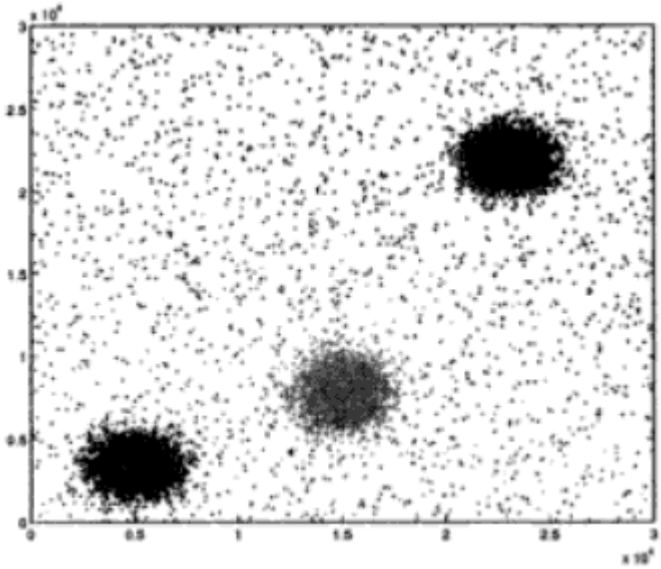
Το πάνω μισό στο Σχήμα 2.2-3 παρουσιάζει τις καταστάσεις ενός γονιδίου στην ορθή αλυσίδα(+), ενώ το κάτω μέρος παρουσιάζει τα στάδια ενός γονιδίου στην ανάστροφη αλυσίδα (-). Αυτό συμβαίνει για να μπορεί να εξετάζει και τα δυο σκέλη παράλληλα. Το N είναι μια **intergeric** περιοχή, ενώ ο P είναι οργανωτής και ο F είναι πέντε περιοχές που δεν έχουν ερμηνευτεί από την αρχή της μεταγραφής, μέχρι το σήμα έναρξης μεταφράσεων. Το T είναι τρείς περιοχές που δεν έχουν ερμηνευτεί αμέσως μετά το αρχικό **codon** μέχρι το σήμα **poly-A.** Το Esngl είναι προκαρυωτικό γονίδιο, όπου η μετάφραση ξεκινά στο τερματικό **codon,** ενώ το Einit είναι του αρχικού **εξανίου**. Το Ek (0<=k<=2) είναι η φάση k εσωτερικού **εξανίου**, και το Ik είναι μια φάση από **ιντρόνια.** Τέλος το Eterm είναι τερματικό **εξάνιο**. Για ευκολία ο GenScan αλγόριθμος μεταχειρίζεται τις περιοχές χορηγών και αποδεχτών, τη μετάφραση έναρξης/λήξης σημάτων ως μια υπορουτίνα των συνεργαζομένων **εξανίων**.Ο GenScan αλγόριθμος ακολουθεί «φάσεις» από **ιντρόνια** και **εξάνια**. Τα εσωτερικά **ιντρόνια** και **εξάνια** διαιρούνται σύμφωνα με τη κάθε «φάση». I0 είναι ένα **ιντρόνιο** που βρίσκεται μεταξύ **codons** βρίσκεται στη «φάση» 0. I1 είναι ένα **ιντρόνιο** το οποίο είναι μετά τη πρώτη βάση του **codon** και βρίσκεται στη «φάση» 1 και τέλος I2 είναι ένα **ιντρόνιο** το οποίο βρίσκεται μετά τη δεύτερη βάση του **codon** και βρίσκεται στη «φάση» 2. Αυτό ισχύει και για τα εσωτερικά **εξάνια** και αντίστοιχα είναι E0, E1, E2.

Τα πλεονεκτήματα του αλγορίθμου είναι: (1)μοντελοποιεί από ένα μέχρι και πολλά **εξάνια** γονίδια, (2) διαμορφώνει το μήκος των **εξάνια** και **ιντρόνια, (3)** μοντελοποιεί **promoters, poly-A** σήματα και **intergenic** περιοχές, (4)επιτρέπει τα προωθημένα και

αντίστροφα γονίδια, (5) είναι σε θέση να βρει ένα μέρος του γονιδίου ή και ολόκληρο. Τα μειονεκτήματα του είναι (1)δε μπορεί να χειριστεί επικαλυπτόμενες μονάδες μεταγραφής και (2) δεν υπάρχει εναλλακτικός τρόπος διαίρεσης.

2.2-4 *Αλγόριθμος* GENIE:

Ο αλγόριθμος Genie βασίζεται στη θεωρία του clustering. Ένας στοιχειώδης clustering αλγόριθμος περιέχει μια λίστα από k κέντρα. Κάθε καινούργιο σημείο που παρουσιάζεται είτε αντιστοιχεί σε έναν από τους clusters, ή χρησιμοποιείται για να ξεκινήσει καινούριο cluster ενώ γίνεται η συγχώνευση δυο άλλων clusters. Όταν λέμε γενικευμένος στοιχειώδης αλγόριθμος, εννοούμε ένα στοιχειώδη αλγόριθμο που να μετακινεί ένα κέντρο σε μια λίστα από κέντρα, χρησιμοποιώντας το βάρος των ήδη υπάρχουσων κέντρων και παρουσιάζοντας ένα καινούριο σημείο. Αρχικά ο αλγόριθμος διαλέγει παραμέτρους. Δηλαδή διορθώνει τον αριθμό των κέντρων k, τον αριθμό των αρχικών σημείων m και ακολούθως το μέγεθος των παραγόμενων n. Στο δεύτερο στάδιο γίνεται η αρχικοποίηση διαλέγοντας σημεία πχ. m, c1, …, cm τα οποία θα είναι αρχικά προτεινόμενα κέντρα. Στη συνέχεια προσδιορίζει βάρος wi = 1 για καθ' ένα από αυτά τα προτεινόμενα κέντρα. Στο τρίτο στάδιο γίνεται το στοιχειώδες clustering. Για κάθε υπό ακόλουθο σημείο p δεδομένων πρέπει να βρίσκει το πλησιέστερο προτεινόμενο κέντρο ci στο σημείο p, μετακινεί το πλησιέστερο υποψήφιο κέντρο χρησιμοποιώντας την εξίσωση $c_i = \frac{(w_i * c_i + p)}{(w_i + 1)}$, και αυξάνει το αντίστοιχο βάρος wi = wi + 1. Τέλος όταν ισχύει η συνθήκη $Count \equiv 0 \ mod \ n$ τότε ακολουθεί το τέταρτο βήμα όπου γίνεται η ενημέρωση γενεών για τα υποψήφια κέντρα. Όταν Count= n, 2n,3n,… για κάθε κέντρο ci στη λίστα L από κέντρα τότε: υπολογίζεται τη πιθανότητα για επιβίωση χρησιμοποιώντας την εξίσωση $p_i = \frac{w_i}{\sum_{i=1}^{m} w_i}$. Έπειτα διαλέγει τυχαίο αριθμό στο διάστημα δ [0,1] και εάν pi> δ, τότε διατηρεί το κέντρο ci στη λίστα L και το χρησιμοποιεί στην επόμενη παραγωγή των n σημείων. Εάν pi<δ, τότε γίνεται αντικατάσταση του κέντρου ci με ένα άλλο που επιλέγεται τυχαία μέσα από τη λίστα L. Μετά γίνεται και πάλι αρχικοποίηση του βάρους wi=1, παρόλο που μερικά σημεία τώρα είναι εγγεγραμμένα σε άλλα κέντρα και γίνεται επαναφορά στο τρίτο βήμα για να συνεχιστεί η διαδικασία. Στο τελευταίο βήμα υπολογίζονται οι τελευταίοι clusters. Η λίστα L περιέχει m κέντρα που μπορούν να ομαδοποιηθούν στα τελικά k κέντρα βασισμένα στην Ευκλείδεια απόσταση.

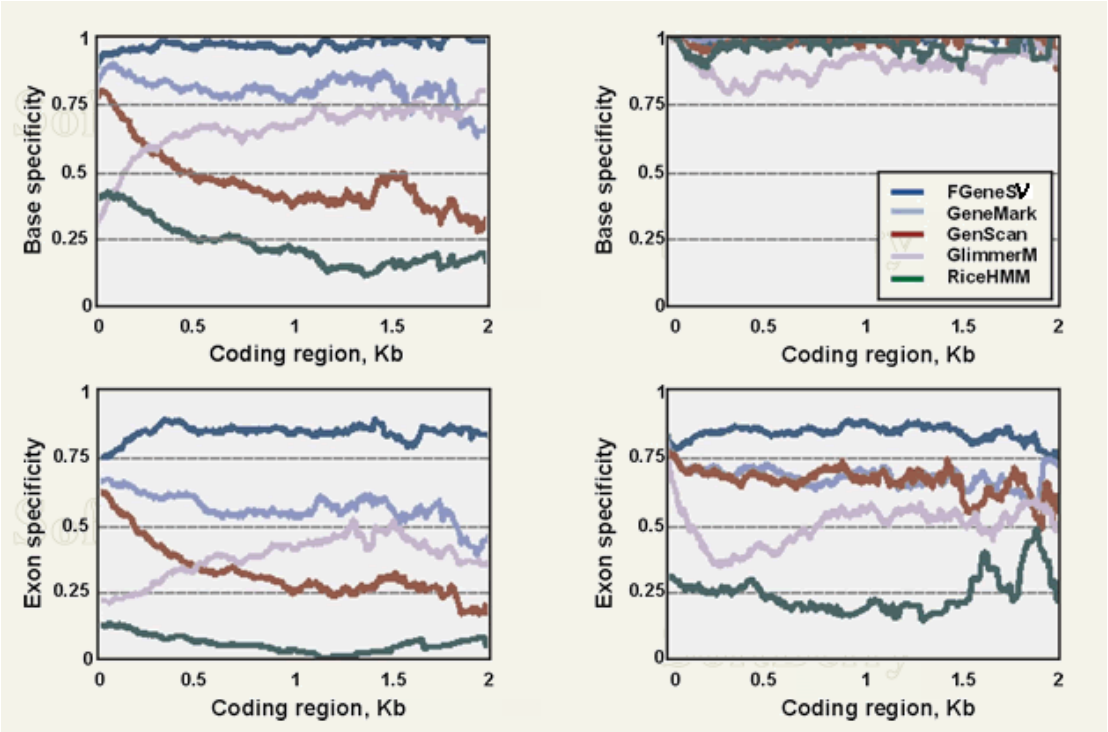


Σχήμα 2.2-2: Δυσδιάστατη αναπαράσταση του Clustering χρησιμοποιώντας τον αλγόριθμο Genie

2.2-5 Αλγόριθμος FGENESV:

Ο αλγόριθμος FGENESV είναι βασισμένος σε αναγνώριση προτύπων διαφορετικών σημάτων και σε αλυσίδες Markov για την κωδικοποιημένη περιοχή. Ο συνδυασμός αυτών των δύο και μιας δοσμένης ακολουθίας, κατασκευάζει με δυναμικό προγραμματισμό ένα σύνολο γονιδίων. Είναι ο πιο γρήγορος διαθέσιμος αλγόριθμος όσον αφορά την τρίτη κατηγορία αλγορίθμων γονίδιο-εύρεσης, (ab initio). Υπάρχουν δυο εκδοχές στο πρόγραμμα.

1) FGENESV0 το οποίο είναι κατάλληλο για μικρό αριθμό γονιδιωμάτων (<10kb), και χρησιμοποιεί γενικές παραμέτρους από κωδικοποιημένες περιοχές.
2) FGENESV χρησιμοποιεί σαν είσοδο γονιδιακές ακολουθίες και έχει σαν έξοδο συγκεκριμένες παραμέτρους γονιδιωμάτων. Ο αλγόριθμος προβλέπει όλα τα γονίδια τα οποία δεν περιέχουν ιντρόνια και χρησιμοποιείται και για την εύρεση μικρής ομάδας γονιδίων τα οποία περιέχουν ιντρόνια.



Σχήμα 2.2-5: Στις δυο πάνω γραφικές βλέπουμε την απόδοση διαφόρων αλγορίθμων εύρεσης γονιδίων έχοντας σαν είσοδο τη γονιδιακή ακολουθία ρυζιού (οι οποίοι αναγράφονται μέσα στην εικόνα) . Στις δυο κάτω βλέπουμε την απόδοση τρων ιδίων αλγορίθμων σε 42 σχεδόν τεχνικές γονιδιακές ακολουθίες περιέχοντας 178 γνωστές γονιδιακές ακολουθίες ανθρώπου (δηλαδή 900 e)

2.2-6 *Αλγόριθμος* Glimmer:

Ο αλγόριθμος Glimmer χρησιμοποιείται **μόνο** για προκαρυωτικούς οργανισμούς. Το κύριο χαρακτηριστικό του είναι η χρήση των Interpolated Markov Models (IMMs) για την εύρεση κωδικοποιημένων περιοχών. Η βασική διαφορά μεταξύ Markov Models και Interpolated Context Markov Models (ICMM) είναι ο αριθμός των προηγούμενων βάσεων που χρησιμοποιούνται για τον υπολογισμό της τελικής πιθανότητας. Το μειονέκτημα των fixed Markov Models είναι ότι ο υπολογισμός της πιθανότητας για k+1 βάσεις εξαρτάται από το σύνολο των προηγούμενων βάσεων, ενώ στην περίπτωση των ICMM ο αριθμός αυτός είναι μεταβλητός όταν η είσοδός μας είναι μικρή. Ο αλγόριθμος αποτελείται από τρία στάδια: στάδιο εκπαίδευσης (Training phase), αναγνώρισης (identification phase) και επίλυσης (resolving overlap).

1. *Στάδιο εκπαίδευσης (*Training *phase):*

Σε αυτή τη φάση «χτίζουμε» ένα δέντρο χρησιμοποιώντας τις κωδικοποιημένες περιοχές των γνωστών γονιδίων. ORF είναι ένα μέρος του DNA με αρχή τα start codons και τέλος τα end codons. Για κάθε μια από τις τρεις θέσεις ανάγνωσης, είναι όλα χωρισμένα σε k+1 περιοχές για τον υπολογισμό της αμοιβαίας πληροφορίας για κάθε θέση έναντι της k-th βάσης. Η θέση με την υψηλότερη αμοιβαία πληροφορία επιλέγεται και χρησιμοποιείται για τη κατασκευή του δέντρου. Η παραπάνω διαδικασία επαναλαμβάνεται για κάθε κόμβο του δέντρου. Τέλος, «χτίζεται» ένα δέντρο με οχτώ επίπεδα και ο κάθε κόμβος περιλαμβάνει

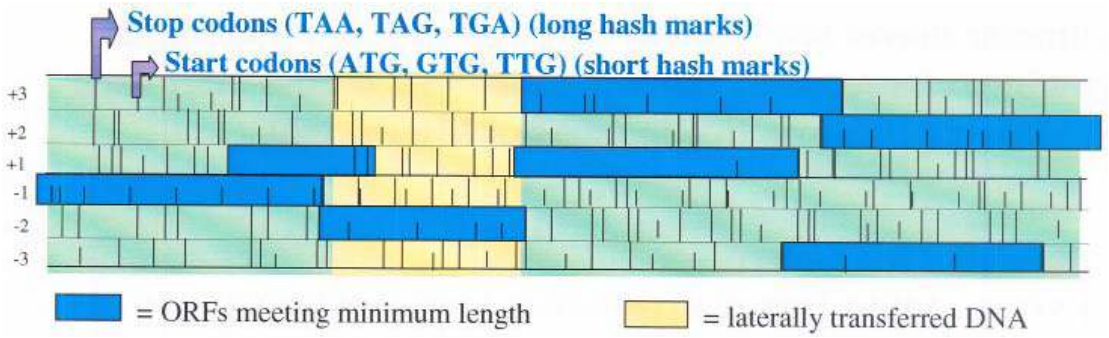
πληροφορία για τις πιθανότητες που φέρουν οι τέσσερις βάσεις (A, C, T, G) και τη θέση με την υψηλότερη αμοιβαία πληροφορία.

2. *Στάδιο αναγνώρισης (identification phase)*:

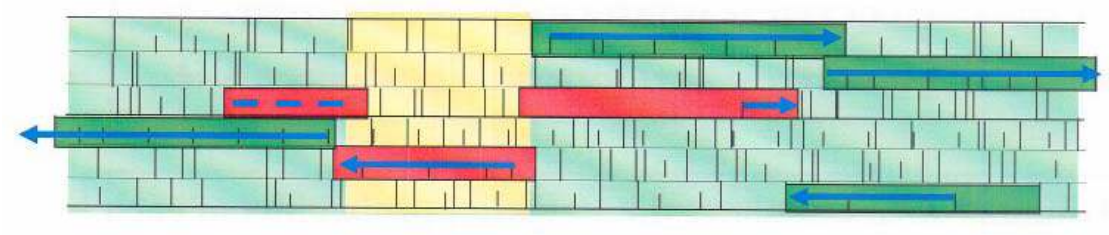
Η δεύτερη φάση του αλγορίθμου υπολογίζει το σκορ του ORF που δείχνει τη πιθανότητα για την οποία το ORF είναι ένα γονίδιο. Τα βήματα είναι τα εξής: Γίνεται επιλογή του σωστού δέντρου για το σωστό αναγνωστικό πλαίσιο. Για κάθε ένα k+1 από τα διαφορετικά περιεχόμενα του ORF διασχίζουμε το δέντρο μέχρι να φτάσουμε στα φύλλα. Διαβάζει τη πιθανότητα και υπολογίζει το λογάριθμο της τιμής. Προσθέτει τη πιθανότητα στον προηγούμενο λογάριθμο. Γίνεται η μετάθεση στην επόμενη θέση του ORF και επαναλαμβάνεται η διαδικασία.

3. Στάδιο επίλυσης (resolving overlap):

Στην τελική φάση ο Glimmer επιλύει το θέμα των επικαλυπτόμενων ακολουθιών. Η κύρια διαδικασία για την επίλυση του προβλήματος είναι η συντόμευση του επικαλυπτόμενου ORF επιλέγοντας διαφορετικό start/stop codon. Τέλος εάν δε μπορεί να βρεθεί τρόπος αντιμετώπισης της επικάλυψης τότε τη θέση του θα πάρει το ORF με το μεγαλύτερο σκορ.



***Σχήμα 2.2-3**: Υποψήφια γονίδια (start/ stop codons)*



Σχήμα 2.2-4: Glimmer score: η κόκκινη ζώνη περιέχει τα χαμηλά σκορ, η πράσινη τα υψηλά σκορ και τέλος το υπόλοιπο μέρος δεν έχει start codon

2.3 Αλγόριθμοι HMM και FPGA's

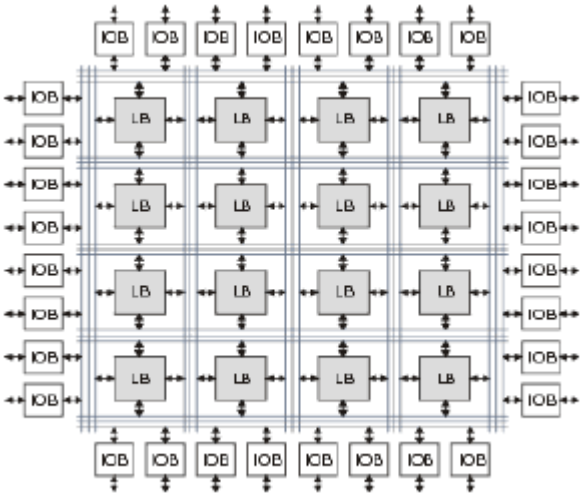
Οι σύγχρονες FPGAs (Field *Programmable* Gate Arrays) έχουν αυξηθεί σημαντικά ως προς το μέγεθος, την ικανότητα και την ταχύτητα. Αρκετές επιχειρήσεις σήμερα, φτιάχνουν FPGAs πλακέτες και τις εφοδιάζουν με διάφορους τρόπους ώστε να βοηθούν την

CPU σαν συνεπεξεργαστές στην αντιμετώπιση υπολογισμού κάποιου αλγορίθμου. Σε γενικές γραμμές, πάντως, δεν είναι εύκολο να προγραμματιστεί μια FPGA.

Γενικά μιλώντας δεν υπάρχει ένας compiler που θα μεταφράσει ένα πρόγραμμα σε μια προγραμματιστική γλώσσα υψηλού επιπέδου σε αποδοτικό κώδικα για την εκτέλεση μιας FPGA. Οι σχεδιαστές λογισμικού οι οποίοι προγραμματίζουν τις παραδοσιακές CPUs δεν ενθαρρύνονται να χρησιμοποιούν μηχανήματα με FPGAs, επειδή είναι δύσκολο να γίνει αυτό χωρίς τη γνώση του hardware. Για αυτό το λόγο οι FPGAs, δεν χρησιμοποιούνται σε πολλές επιστημονικές εφαρμογές των οποίων οι επιδόσεις θα μπορούσαν να βελτιώσουν σε μεγάλο βαθμό με τη χρήση αυτών.

Προγραμματιζόμενοι λογικοί μηχανισμοί (*Programmable* Modules) είναι ενότητες που έχουν αναπτυχθεί για να είναι προσαρμοσμένες για συγκεκριμένες λειτουργίες στο τελευταίο στάδιο της κατασκευής. Έχουν γίνει αρκετά δημοφιλής λόγω των πλεονεκτημάτων που έχουν αποκτηθεί μέσω της ευελιξία τους. Προγραμματιζόμενες ενότητες ή PDLs είναι επωφελής, λόγω του γεγονότος ότι το κόστος ενός chip που περιέχει μια προγραμματιζόμενη μονάδα είναι ανεξάρτητο από τον αριθμό των πυλών που βρίσκονται σε αυτό. Οι FPGAs είναι μια ενότητα στη VLSI που μπορεί να προγραμματιστεί για να καταστεί δυνατή η υλοποίηση των πολυεπιπέδων δικτύων και για τα πολύπλοκα συστήματα σε ένα ενιαίο chip.

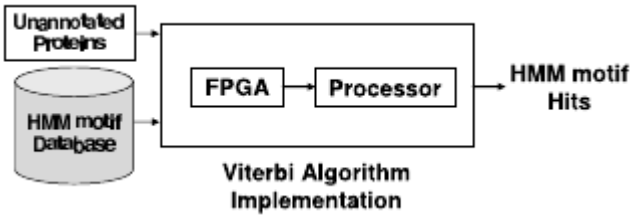
Μια πλακέτα FPGA (σχήμα 2.3-1)είναι ένας πίνακας που αποτελείται από: Logic Blocks (συνδυαστικά ή διαδοχικά), σημεία διασύνδεσης (διακόπτες) μεταξύ των μπλοκ λογικής, και Input / Output Blocks (IOB). Τα Logic Blocks μπορεί να προγραμματιστούν για την υλοποίηση πολλών λειτουργιών, και συνήθως υλοποιούνται σαν *Look-Up Table* (LUT), όπου ένα LUT με k εισόδους είναι προγραμματισμένος ως ένας πίνακας αλήθειας μιας τερματικής λειτουργίας μέχρι k μεταβλητές. Το σημείο διασύνδεσης των μπλόκς εισόδου / εξόδου μπορεί να προγραμματιστεί για την επίτευξη των απαιτούμενων συνδέσεων.



Σχήμα 2.3-1: FPGA: φαίνονται τα λογικά μπλοκς (logic blocks - LB), τα μπλοκ εισόδου/εξόδου (input/output blocks - IOB), και τα σημεία διασύνδεσης

Όπως φαίνεται από τα παραπάνω με τη χρήση των FPGAs μπορούμε να κάνουμε έναν αλγόριθμο γρηγορότερο. Παρακάτω ακολουθεί μια σύντομη περιγραφή των αλγορίθμων

Grail και GeneFinder. Στο σχήμα 2.3-2 φαίνεται η εφαρμογή ενός HMM αλγορίθμου σε αναδιατασσόμενη λογική.



Σχήμα 2.3-2: Εύρεση γονιδίου χρησιμοποιώντας FPGA

2.3-1 Αλγόριθμος Grail σε αναδιατασσόμενη λογική

Ο σχεδιασμός υλικού έγινε σε Verilog και η FPGA που χρησιμοποιήθηκε είναι η Xilinx Spastan-3 XC3S1500. Επίσης το εργαλείο που χρησιμοποιήθηκε είναι η Xilinx ISE 8.2i. Το μήκος ενός PE είναι 451 λογικοί πόροι. Το ποσοστό μνήμης που απαιτείται είναι 50 RAM εισόδου για κάθε στάδιο του Grail αλγορίθμου που περιλαμβάνει 42 εκπομπές και 8 μεταβάσεις. Περαιτέρω, υπάρχουν ακόμα 3 είσοδοι ανά στάδιο του αλγορίθμου για κάθε PE's IVS. Για αυτό το μέγεθος και των δύο (αλγόριθμος και PE) που μπορεί να υποστηριχτεί είναι περιορισμένο από τον αριθμό των Block RAM μέσα στην FPGA. Χρησιμοποιώντας και τα 32 Block RAMs που έχει η XC3S1500, είναι σε θέση να φιλοξενήσει 10 PEs και να υποστηρίξει τον Grail αλγόριθμο για μέχρι 256 στάδια. Η απόδοση του συστήματος που χρησιμοποιείται συχνά στην βιοπληροφορική είναι η ενημερώσεις κυττάρων ανά δευτερόλεπτο (cell updates per second - CUPS). Άρα έχοντας συχνότητα ρολογιού στα 70 MHz * 10 PE's = 700 Mega CUPS, σε μέγιστη επιτρεπόμενη ταχύτητα ρολογιού των 34 MHz.

2.3-2 Αλγόριθμος GeneFinder σε αναδιατασσόμενη λογική

Χρησιμοποιώντας μια FPGA Xilinx Virtex II XC2V6000, η οποία 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. Μια σειρά δοκιμών αξιολόγησης των επιδόσεων έχουν διεξαχθεί χρησιμοποιώντας διαφορετικό αριθμό ακολουθιών για την αξιολόγηση του χρόνου επεξεργασίας των FPGA-accelerated implementations έναντι στον ακολουθιακό κώδικα GeneFinder. Κατά τις δοκιμές χρησιμοποιείται PCI βασισμένο σε ADP-WRC-II board από Alpha-Data με μία Xilinx XC2V6000 FPGA. Η εφαρμογή GeneFinder είναι υποδειγματική σε Intel Pentium IV 3 GHz επεξεργαστή με 1GB RAM. Η FPGA 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 τον αλγόριθμο GeneFinder. Μια σύγκριση αυτών των δύο προσεγγίσεων παραλληλοποιήσης 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.

2.3-3 Αλγόριθμος Glimmer σε αναδιατασσόμενη λογική

Ο σχεδιασμός υλικού έγινε σε Vhdl και η FPGA που χρησιμοποιήθηκε είναι η Xilinx Virtex5 SX240T. Η ταχύτητα ρολογιού είναι στα 150 MHz. Ο πίνακας 2.3.3-1 μας δείχνει την απόδοση του συστήματος, καθώς επίσης και τη σύγκριση μεταξύ του συστήματος και του software στον υπολογισμό του σκορ. Το αντίκτυπο της βαθμολόγησης στη φάση του υπολογισμού του σκορ για το συνολικό χρόνο εκτέλεσης της συνάρτησης φαίνεται στον πίνακα 2.3.3-2. Σε αυτό τον πίνακα υπάρχει επίσης και το ποσοστό του συνολικού χρόνου εκτέλεσης με τη χρήση μόνο της CPU, καθώς και μία CPU με συνεπεξεργαστή FPGA και η προβολή του Speedup για το πλήρες σύστημα. Η μεγαλύτερη απαιτούμενη απόδοση είναι 600 Mbps με μέσο όρο 300 Mbps.

Database	Number of bases for the scoring ORFs	Execution Time(CPU 2.66 GHz, 1 GB RAM)	Execution Time(FPGA @150MHz)	CPU Throughput (Bases/sec)	FPGA Throughput (Bases/sec)	SpeedUp FPGA vs. CPU
NC_004061	2385968	0.399	0.016	5981518	149997673	25.07 x
NC_003062	135542442	30.082	0.903	4505765	149999980	33.29 x
NC_004463	524562162	114.514	3.497	4580768	150000000	32.74 x

Πίνακας 2.3.3-1: Αποτελέσματα του Glimmer αλγορίθμου SW-HW

Database	CPU % of Scoring phase execution time	Total Execution Time (CPU 2.66 GHz. 1 GB RAM)	New Execution Time Using CPU+FPGA	Total SpeedUp CPU+FPGA vs. CPU
NC_004061	12.39%	3.22	2.84	1.14 x
NC_003062	57.24%	52.55	23.38	2.25 x
NC_004463	59.59%	192.17	81.15	2.37 x

Πίνακας 2.3.3-2: Αποτελέσματα του Glimmer αλγορίθμου SW vs. HW-SW co-design

2.4 Σύγκριση αλγορίθμων

Σε αυτό το υποκεφάλαιο γίνεται μια σύντομη αναφορά στις ομοιότητες καθώς και στις διαφορές των αλγορίθμων που έχουμε αναφέρει παραπάνω. Πρώτη ομοιότητα είναι ότι ο σκοπός όλων των αλγορίθμων είναι η εύρεση γονιδίων. Επίσης όλοι οι αλγόριθμοι δέχονται σαν είσοδο ένα αρχείο με μία η περισσότερες DNA ακολουθίες σε μορφή FASTA. Σαν έξοδο έχουμε τη πρόβλεψη ενός μέρους ή ολόκληρης της γονιδιακής ακολουθίας. Στην έξοδο εμπεριέχονται οι τοποθεσίες των προβλεπόμενων γονιδίων και οι coding τους περιοχές καθώς επίσης, τα score όλων των γονιδίων, αλλά και των εξανίων.

Συνεχίζουμε στις διαφορές, οι οποίες οφείλονται κυρίως στην επιλογή μεθόδου για την εύρεση κωδικοποιημένων περιοχών. *Οι αλγόριθμοι HMMgene, TWINSCAN, GenScan, FGENESV και* Glimmer χρησιμοποιούν μοντέλα Markov για την εύρεση κωδικοποιημένων

περιοχών, ενώ ο αλγόριθμος GENIE βασίζεται στη θεωρία του clustering. Τέλος οι αλγόριθμοι Grail και GeneFinder *χρησιμοποιούν για αυτό τον σκοπό τον αλγόριθμο Viterbi.*

Ο HMMgene μπορεί να βρει τα N καλύτερα γονίδια για μια ακολουθία εισόδου. Αυτό είναι χρήσιμο στην περίπτωση που υπάρχουν αρκετές ίσες γονιδιακές δομές ώστε να μπορούμε να βρούμε και εναλλακτικό splicing. Στον αλγόριθμο *TWINSCAN* όσο το αποτέλεσμα των σχολιασμών αποτελείται μόνο από 25,622 γονίδια ο αλγόριθμος θεωρείται λίγο καλύτερος, όσον αφορά την ακρίβεια και την ταχύτητα, από τον GENSCAN ο οποίος θεωρείται ο πιο ακριβής αλγόριθμος όσον αφορά το γονιδίωμα του ανθρώπου. Ακολουθεί ο πίνακας 2.4-1 ο οποίος χωρίζει τους αλγορίθμους μας σε 3 κατηγορίες ανάλογα τον τύπο αλγορίθμου στον οποίο είναι βασισμένοι.

Τύποι αλγορίθμων	**Στοιχεία**	**Παραδείγματα Αλγορίθμων**
Νευρικό σύστημα **(Neural Network – Uberacher & Mural 1991)**	Αναλυτικές τεχνικές πρότυπου όσον αφορά τις διαδικασίες εκμάθησης στα γνωστικά συστήματα. Χρησιμοποιεί τα σύνολα δεδομένων εκπαίδευσης, για την κατασκευή των κανόνων που μπορούν να κάνουν προβλέψεις ή ταξινομήσεις σε δεδομένα.	Grail
Σύστημα βασισμένο σε κανόνες **(Rule-based system)**	Χρησιμοποιεί μια σαφή δέσμη κανόνων για τη λήψη αποφάσεων.	GeneFinder
Hidden Markov Model **(Burge & Karlin 1997)**	Αντιπροσωπεύει ένα σύστημα ως ένα σύνολο διακριτών περιοχών και στις μεταβάσεις μεταξύ των εν λόγω περιοχών. Τα μοντέλα Markov είναι «κρυφά», όταν μια ή περισσότερες από τις περιοχές δεν μπορούν να παρατηρηθούν άμεσα. Κάθε μετάβαση έχει μια σχετική πιθανότητα. Επίσης έχει το πλεονέκτημα της ρητής μοντελοποίησης πως για κάθε μια πιθανότητα από μια ακολουθία χαρακτηριστικών συνδυάζονται σε μια εκτίμηση πιθανότητας για το σύνολο του γονιδίου.	*HMMgene* *TWINSCAN* *GenScan* Genie *FGENESV* Glimmer GlimmerHMM

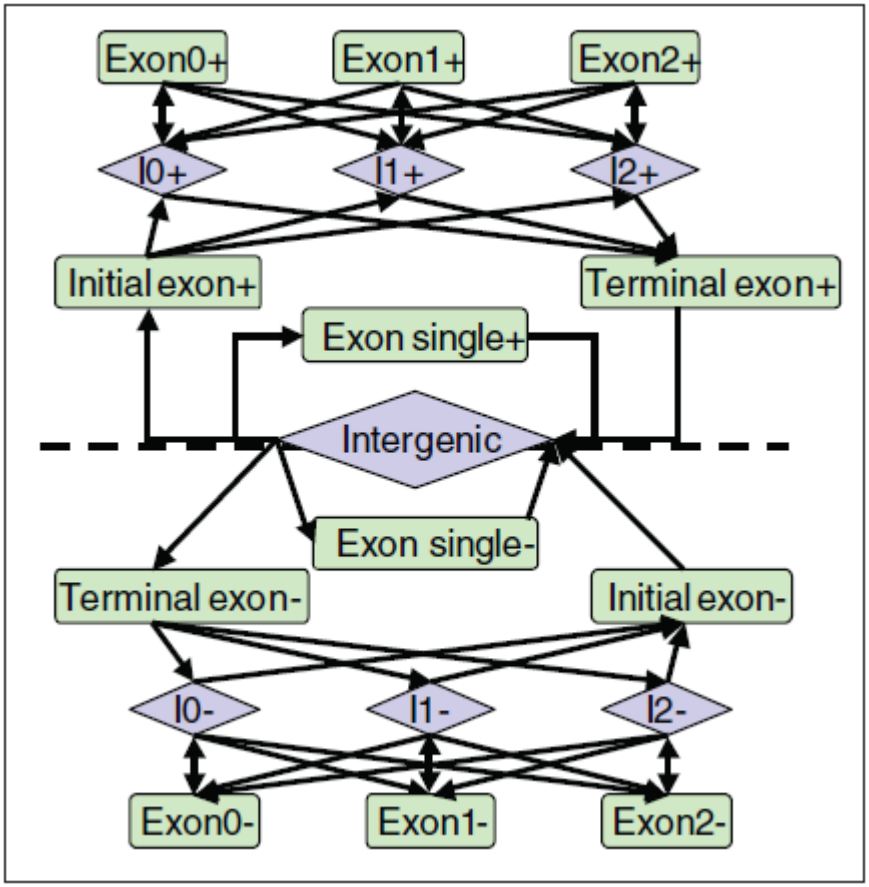
Πίνακας 2.4-1: Τύποι αλγορίθμων εύρεσης γονιδίου σε DNA ακολουθίες.

ΚΕΦΑΛΑΙΟ 3

3.1 Αλγόριθμος GlimmerHMM

Ο GlimmerHMM είναι ένας αλγόριθμος βασισμένος στο Generalized Hidden Markov Model (GHMM) και αναπτύχθηκε το 1998 από τον Steven Salzberg. Ο αλγόριθμος είναι αρκετά διαδεδομένος και δίνει μία γρήγορη και ακριβής λύση στο πρόβλημα εύρεσης γονιδίων. Η αρχική του ονομασία ήταν Glimmer. Στο πέρασμα των χρόνων λόγω της βελτίωσης του αλγορίθμου πήρε τα ονόματα Glimmer1 ακολούθως Glimmer2, GlimmerM και τέλος GlimmerHMM. Η διαφορά του προτελευταίου με τον τελευταίο αλγόριθμο είναι απλά ότι ο GlimmerM είναι εκπαιδευμένος μόνο για προκαρυωτικά γονίδια, ενώ ο GlimmerHMM για ευκαρυωτικά με τι αντίστοιχες διαφοροποιήσεις στο software.

Ο GlimmerHMM ανήκει στη τρίτη κατηγορία εύρεσης γονιδίων (ab-initio) που αναλύσαμε στο πρώτο κεφάλαιο. Μοιάζει αρκετά με τον αλγόριθμο Genscan, τον οποίο και περιγράψαμε στο Κεφάλαιο 2. Αν και η εύρεση γονιδίου είναι κατασκευασμένη σύμφωνα με το γενικό μαθηματικό μοντέλο μιας GHMM, επιπροσθέτα ενσωματώνει τα splice site models, κομμάτι παρμένο από τον αλγόριθμο GeneSplicer και την απόφαση δέντρου από τον αλγόριθμο GlimmerM. Επίσης, ο υπολογισμός των coding και non-coding περιοχών γίνεται με τη χρήση των Interpolated Markov Models. Η δομή GHMM περιλαμβάνει ιντρόνια για κάθε φάση, διαγονιδιακές περιοχές και τέσσερα ήδη εξανίων (initial, interval, final και single). Η παραπάνω περιγραφή φαίνεται στο σχήμα 3.1-1.



Σχήμα 3.1-1: Διάγραμμα του GHMM για GlimmerHMM. Η διακεκομμένη γραμμή στη μέση χωρίζει την ευθεία και ανάστροφη αλυσίδα του μοντέλου. Κάθε στάδιο μέσα στο GHMM υλοποιείται σαν ξεχωριστό υπό μοντέλο.

3.2 Περιγραφή αλγορίθμου GlimmerHMM

Ο αλγόριθμος GlimmerHMM χωρίζεται σε τρία στάδια. Τα τρία αυτά στάδια είναι: στάδιο εκπαίδευσης (training phase), αναγνώρισης (identification phase) και επίλυσης (resolving overlap).

Στάδιο εκπαίδευσης (training phase):

Ο αλγόριθμος χρησιμοποιεί μεγάλα ORF's και ακολουθίες παρόμοιες με γνωστά γονίδια από άλλους οργανισμούς σαν στάδιο εκπαίδευσης του μοντέλου. Είναι σημαντικό να αναφέρουμε ότι η εκπαίδευση του αλγορίθμου γίνεται ***μόνο*** με πέντε διαφορετικά γονιδιώματα οργανισμών (training data sets) τα οποία είναι: 1. Arabidopsis, 2. Celegans, 3. Human, 4. Rice, 5. zebrafish

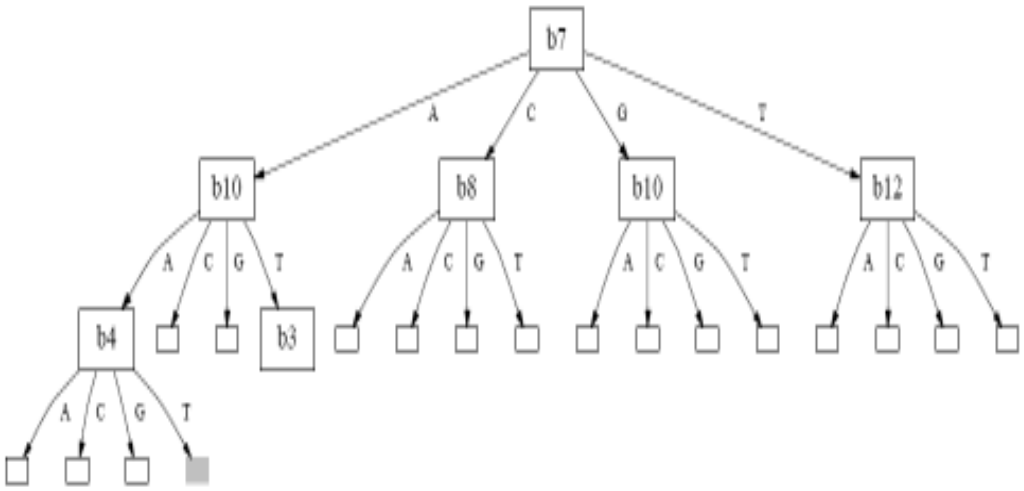
Κατά το training stage, ο αλγόριθμος δέχεται ως είσοδο τα γονίδια ενός από των πέντε παραπάνω οργανισμών και κατασκευάζει ένα HMM σε δενδρική δομή. Το δέντρο αποτελείται από οχτώ επίπεδα κόμβων, ενώ ο κάθε κόμβος περιλαμβάνει τέσσερις

πιθανότητες (μια για κάθε δυνατή βάση A, C, G, T). Τα δεδομένα των δέντρων που υπολογίζονται στον αλγόριθμο αλλάζουν **μόνο** ανάλογα με το πιο γονιδίωμα θα γίνει το training. Η διαδικασία εκπαίδευσης του αλγορίθμου είναι αυτή που περιγράφεται παρακάτω.

Η εισερχόμενη αλληλουχία εκπαίδευσης του αλγορίθμου διαμελίζεται σε επικαλυπτόμενες δωδεκάδες βάσεων . Αυτές οι δωδεκάδες καταμετρώνται και στη συνέχεια χρησιμοποιώντας μόνο τις οχτώ από τις δώδεκα βάσεις προκύπτει μία πιθανότητα εμφάνισης για όλες τις δυνατές οκτάδες βάσεων. Τελικά, σχηματίζεται μια δενδρική δομή οχτώ επιπέδων της οποίας η διάσχιση από τη ρίζα μέχρι τα φύλλα απεικονίζει μια οκτάδα βάσεων ενώ στα φύλλα αποθηκεύονται οι τέσσερις πιθανότητες εμφάνισης στην τελευταία θέση της κάθε δωδεκάδας μιας από τις τέσσερις δυνατές γενετικές βάσεις.

Επειδή για κάθε codon έχουμε τρεις θέσεις πλαισίου ανάγνωσης τελικά προκύπτουν τρείς δενδρικές δομές, όπως αυτές περιγράφηκαν παραπάνω , μία για κάθε θέση πλαισίου του γονιδίου του οργανισμού εκπαίδευσης.

Ο GlimmerHMM χτίζει ένα δέντρο όπως αυτό φαίνεται στο σχήμα 3.2-1.



Σχήμα 3.2-1: Δέντρο μοντελοποίησης

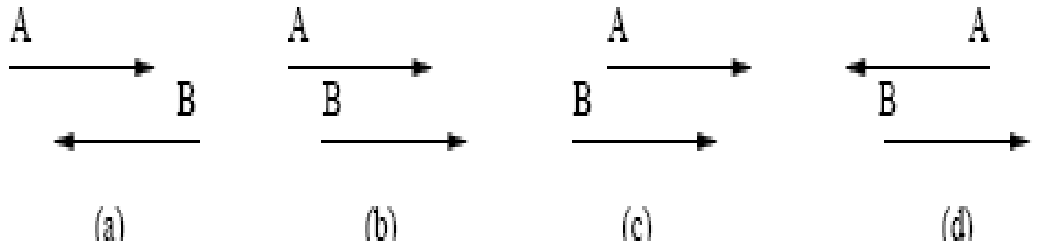
Στάδιο αναγνώρισης (identification phase):

Εφόσον έχει ολοκληρωθεί το στάδιο εκπαίδευσης, περνάμε στο στάδιο εντοπισμού. Δεδομένου ενός υποψηφίου ORF ο αλγόριθμος διαμελίζει το ORF σε όλες τις επικαλυπτόμενες δωδεκάδες. Στη συνέχεια για κάθε μια από αυτές και ακολουθώντας το κατάλληλο μονοπάτι στο δέντρο υπολογίζεται η πιθανότητα εμφάνισης του. Πρώτα απ 'όλα θα πρέπει να σημειωθεί ότι η ενδελεχής συλλογή από μια σειρά καλών στοιχείων εκπαίδευσης, είναι ένα βήμα που πρέπει να προηγηθεί για τη τυχόν εύρεση γονιδίων. Αυτό το βήμα είναι πολύ σημαντικό, καθώς η ποιότητα των στοιχείων που παρουσιάστηκαν για

την εκπαίδευση είναι ευθέως ανάλογη με την ακρίβεια που είχε το αποτέλεσμα εύρεσης γονιδίων.

Στάδιο επίλυσης (resolving overlap):

Στο τελευταίο στάδιο, ο GlimmerHMM, επιλύει τις υποψήφιες γονιδιακές αλληλουχίες που αλληλεπικαλύπτονται. Η κύρια ευελιξία στην επίλυση επικάλυψης είναι η δυνατότητα μείωσης ενός ORF, επιλέγοντας ένα διαφορετικό codon έναρξης. Ας υποθέσουμε ότι οι ακολουθίες A και B επικαλύπτονται, και ότι η A έχει μεγαλύτερο σκορ. Υπάρχουνε τέσσερις πιθανότητες για τον τρόπο με τον οποίο επικαλύπτονται, όπως φαίνεται στο σχήμα 3.2-2. Κάθε βέλος στο σχήμα δείχνει στο τρίτο τέλος της ακολουθίας.



Σχήμα 3.2-2: Επικάλυψη υποψήφιων γονιδίων.

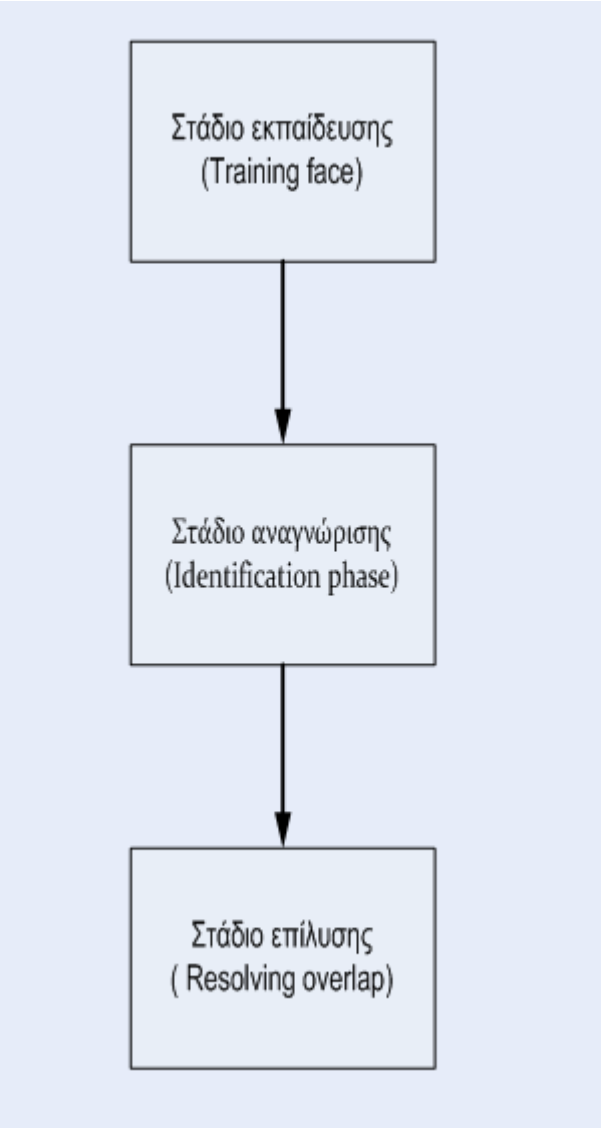
1. *Ας υποθέσουμε ότι η A και B επικαλύπτονται όπως το σχήμα 3.2-2(a). Μετακινώντας το codon έναρξης δε μπορούμε να ελαχιστοποιήσουμε την επικάλυψη σε αυτή τη περίπτωση. Εάν η A είναι αρκετά πιο μεγάλη από τη B τότε απορρίπτουμε την B. Αλλιώς, δεχόμαστε και τις δυο με το σχόλιο ότι έχουν επικάλυψη.*

2. *Ας υποθέσουμε ότι η A και B επικαλύπτονται όπως φαίνεται στο σχήμα 3.2-2(b). Εάν μετακινήσουμε στη B το codon έναρξης επιλύουμε την επικάλυψη με τέτοιο τρόπο ώστε η B να έχει ακόμα αρκετό σκορ και μήκος ώστε να δεχτούμε και τις δύο. Εάν αυτό δε γίνεται τότε συνεχίζουμε στη κατάσταση (3).*

3. *Ας υποθέσουμε ότι η A και B επικαλύπτονται όπως το σχήμα 3.2-2(c). Εάν το σκορ της επικάλυψης είναι ένα μικρό κλάσμα του σκορ της ακολουθίας A, και κινείται το codon έναρξης του A με τέτοιο τρόπο ώστε στην A να παραμένει μεγάλο το μήκος τότε δεχόμαστε και τις δυο. Αλλιώς, απορρίπτουμε τη B.*

4. *Ας υποθέσουμε ότι η A και B επικαλύπτονται όπως το σχήμα 3.2-2(d). Μετακινούμε το codon έναρξης της B μέχρι το σκορ επικάλυψης να είναι μεγαλύτερο από αυτό της A. Τότε μετακινούμε και το codon έναρξης της A μέχρι το σκορ επικάλυψης να είναι μεγαλύτερο από αυτό της B. Επαναλαμβάνουμε μέχρι να μην υπάρχει άλλη επικάλυψη και τότε δεχόμαστε και τα δύο.*

Εάν υπάρχουν περισσότερες από δυο ακολουθίες, τότε εφαρμόζουμε τις παραπάνω διεργασίες ξεκινώντας με την αντιμετώπιση των αλληλοεπικαλύψεων κατά φθίνουσα σειρά βαθμολογίας. Με αυτό τον τρόπο, εάν η A επικαλύπτει τη B και η B επικαλύπτει τη C, και η A έχει το μεγαλύτερο σκορ, τότε μπορεί να απορριφτεί η B πριν επικαλυφθεί με τη C, γιατί αυτό θα σημαίνει ότι απορρίπτεται και η C.



Σχήμα 3.2-3: Flow chart του αλγορίθμου GlimmerHMM.

3.2 Μοντελοποίηση Αλγορίθμου GlimmerHMM

Σε αυτή την ενότητα θα αναλυθούν τα διάφορα μέρη του software του αλγορίθμου όσον αφορά το χρόνο εκτέλεσής του. Επίσης θα αναλυθούν τα μέρη του αλγορίθμου που

κρίθηκαν σημαντικό να υλοποιηθούν σε hardware και θα διευκρινιστούν οι λόγοι που αυτά επιλέχθηκαν στα πλαίσια της παρούσας διπλωματικής εργασίας .

3.3.1 Ανάλυση Software Αλγορίθμου GlimmerHMM

Ο αλγόριθμος *GlimmerHMM* εκτελείται σε UNIX ή LINUX command line εισάγοντας τις σωστές παραμέτρους .

Για την ανάλυση των επιδόσεων του αλγορίθμου *GlimmerHMM* και για την εκτίμηση της συνάρτησης του αλγορίθμου που θεωρείται υπολογιστικά βαριά, χρησιμοποιήθηκε η εφαρμογή Intel V-Tune Performance Analyzer for Linux. Κατά την εκτέλεση του αλγορίθμου με τα πέντε διαφορετικά γονίδια εκπαίδευσης και με την ίδια αλληλουχία εισόδου παρατηρήθηκε ότι ο χρόνος εκτέλεσης διαφοροποιείται όπως φαίνεται στον πίνακα 3.3-1. Επίσης, εκτελώντας τον αλγόριθμο για διάφορα training datasets χρησιμοποιώντας την ίδια είσοδο (Leishmania braziliensis 2012684 bp βάσεις), τα αποτελέσματα του V-Tune Performance Analyzer έδειξαν ότι η συνάρτηση του αλγορίθμου που παρουσίασε ενδιαφέρον όσον αφορά τις επιδόσεις της είναι η get_prob_of_window1. Κατά συνέπεια επιλέξαμε αυτήν για την υλοποίηση της σε hardware.

A/A	Training Data Set	% execution time get_prob_of_window1
1	Arabidopsis	28.28%
2	Celegans	31.07%
3	Human	28.51%
4	Rice	23.79%
5	Zebrafish	29.84%

Πίνακας 3.3-1: Απεικόνιση αποτελεσμάτων συνάρτησης με τη χρήση του V-Tune Performance Analyzer.

Στον πίνακα 3.3-2 που ακολουθεί βλέπουμε τις εισόδους που χρησιμοποιήσαμε για όλες τις χρονικές μετρήσεις, και στον πίνακα 3.3-3 βλέπουμε το συνολικό χρόνο του αλγορίθμου και τον χρόνο εκτέλεσης της συνάρτησης get_prob_of_window1, για έξι διαφορετικές εισόδους (Πίνακας 3.3-2)και τα πέντε training data sets.

Number	Input's Name	Input's Length
(1)	Gallus Gallus	1024
(2)	Ostreococus lucimarinus	149386
(3)	Leishmania braziliensis	2012684
(4)	Populus Trichocarpa	35571569
(5)	Homo Sapiens	90908613
(6)	Monodelphis	192966877

Πίνακας 3.3-2: Απεικόνιση εισόδων

INPUT	TRAINING DATASET									
	Arabitopsis		Rice		Celegans		Zebrafish		Human	
	Total time (sec)	Function Time (sec)	Total time (sec)	Function Time (sec)	Total time (sec)	Function Time (sec)	Total time (sec)	Function Time (sec)	Total time (sec)	Function Time (sec)
(1)	0.04	<0.01	0.14	<0.01	0.25	<0.01	0.06	<0.01	0.17	<0.01
(2)	0.86	0.44	1.16	0.44	2.05	0.41	0.88	0.46	1.46	0.32
(3)	12.02	6.07	15.54	6.18	29.36	6.09	12.29	6.12	21.95	4.93
(4)	222.71	104.59	247.28	105.48	347.14	104.76	206.49	105.10	229.50	106.78
(5)	631.74	149.79	707.14	146.17	1016.45	145.19	604.35	145.72	608.66	115.25
(6)	1319.61	585.47	1468.09	589.16	2120.53	586.22	1223.88	587.51	1327.95	586.67

Πίνακας 3.3-3: Απεικόνιση αποτελεσμάτων χρόνου στο software

Ο αλγόριθμος δέχεται ένα αρχείο της μορφής fasta όπως φαίνεται στο σχήμα 3.3.1-1. Όλοι οι είσοδοι που έχουν χρησιμοποιηθεί σε αυτή τη διπλωματική εργασία είναι από το επίσημο site του NCBI (National Center for Biotechnology Information - [24]). Τα αρχικά γράμματα A, C, G, T αντιστοιχούν στις: Αδενίνη (A - Adenine), Γουανίνη (G - Guanine), Θυμίνη (T - Thymine), Κυτοσίνη (C - Cytosine). Στο σχήμα 3.3.1-3 απεικονίζεται ένα μικρό δείγμα εξόδου από το σύστημα μας.

```
>AB013110
AGTCTATAAGAAGATTGACAGCCAAGAACACCACCACAATGAAGACCGCCGCTCTTGCACCGCTCT
TCTTCCTCCCCTCTGCCCTCGCCACGACGGTCTATCTCG
>X89714
TACCTCGCCAGTTACCTCTCCGCGACAGTGGTTAACGACGCGGTCGCGGGCCGCAGCGCAAAAAA
AAAAAAAAAAAAAAAAAAAAAAA
>Y08867
CTCAAAAAAGSCAGTAATGACTTGACTCAATGATGGTCAATAGTGGAATCTGGCCATGTGGTGATT
GCCGAGTTGTTRACAGGCTGSCCCCTTTCTCATCTTTCCT
>AY013315
GAGGACCGGCTGGTCCGGTGCAATGGCCATCGCCATCAATTGCTGCTGTGCAAG
```

Σχήμα 3.3.1-1: Ενδεικτική μορφή αρχείου εισόδου FASTA στο σύστημα. Το σύμβολο ‘ > ’ δηλώνει το όνομα της κάθε ακολουθίας (sequence) και ακολουθούν τα στοιχεία (residues) αυτής.

```
GlimmerHMM
Sequence name: ref|NW_001581855.1|Mdm1_WGA3_2:1-21256363
Sequence length: 21256363 bp

Predicted genes/exons

Gene Exon Strand  Exon            Exon Range      Exon
   #    #         Type                            Length

   1    1  +  Initial         543        575        33
   1    2  +  Internal        598        792       195
   1    3  +  Terminal        801       1025       225

   2    1  +  Single         1214       1363       150

   3    1  +  Single         1445       1927       483

   4    1  -  Terminal       2396       2469        74
   4    2  -  Internal       2493       2547        55
   4    3  -  Initial        2724       2885       162

   5    1  -  Terminal       3760       3978       219
   5    2  -  Internal       4485       4664       180
   5    3  -  Initial        4747       4758        12

   6    1  +  Initial        4955       5120       166
   6    2  +  Internal       5224       5322        99
   6    3  +  Terminal       5687       5820       134

   7    1  -  Single         5845       6072       228

   8    1  -  Terminal       7202       7303       102
   8    2  -  Initial        7401       7529       129

   9    1  -  Terminal       7855       7938        84
   9    2  -  Initial        8388       8426        39

  10    1  -  Single         8653       9240       588
```

Σχήμα 3.3.1-2: Ενδεικτική έξοδος

3.3.2 Μοντελοποίηση Συνάρτησης get_prob_of_window1

Η συνάρτηση έχει το πρωτότυπο ***double get_prob_of_window1 (int start, tModel * Delta, char *orf)*** *και καλείται από την graph.c*. Δέχεται ως εισόδους:

- 1 ακέραια μεταβλητή τιμή (start) η οποία δείχνει τη θέση ενός χαρακτήρα στην είσοδο μας. Η συνάρτηση αυτή καλείται για όλα τα στοιχεία της ακολουθίας εισόδου (sequence).
- Το Delta όπου είναι δείκτης σε μια από τις τέσσερις δενδρικές δομές που δημιουργήθηκαν κατά την εκπαίδευση.
- Και τέλος δέχεται σαν είσοδό της το **orf** που είναι η αλληλουχία εισόδου.

Η συνάρτηση αυτή έχει ως σκοπό την επεξεργασία εισόδου με τελικό σκοπό τον υπολογισμό της πιθανότητας κάθε δωδεκάδας βάσεων διασχίζοντας την δενδρική δομή ανάλογα με την είσοδό μας ανά κλήση. Για την εξαγωγή αυτής της τελικής πιθανότητας η διαδικασία που εκτελείται από τη συνάρτηση αναλύεται παρακάτω:

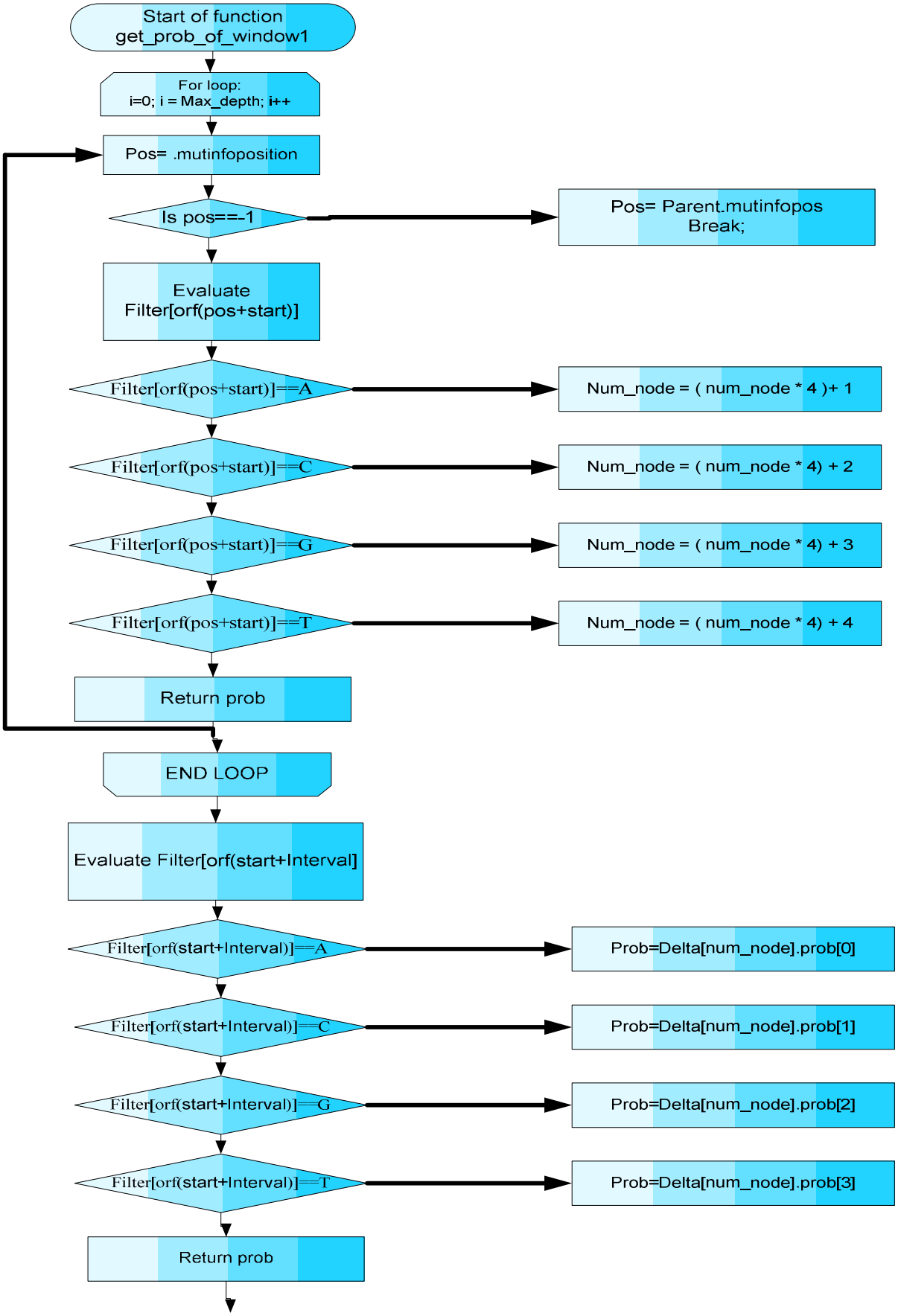
Η συνάρτησή μας δέχεται σαν είσοδο ένα δέντρο ανά κλήση και για την ακρίβεια ένα από τα τέσσερα δέντρα (NMODEL, MODEL[0], MODEL[1], MODEL[2]) που δημιουργούνται κατά το training phase, όπως περιγράφηκε προηγουμένως. Η αναπαραγωγή ενός codon γίνεται ανά τρείς βάσεις, για αυτό έχουμε τρία πλαίσια ανάγνωσης, όπου σε κάθε ένα αντιστοιχεί και ένα από τα MODEL[0], MODEL[1], MODEL[2]. Οι περιοχές αυτές ονομάζονται coding. Το δέντρο NMODEL είναι για τη περίπτωση που οι περιοχές του οργανισμού δεν κωδικοποιούνται σε πρωτεϊνικές ακολουθίες. Οι περιοχές αυτές ονομάζονται non-coding. Πιο συγκεκριμένα τα δέντρα αυτά είναι οχτώ επιπέδων και συνολικά αποτελούνται από 21845 κόμβους. Στον κάθε κόμβο αποθηκεύονται οι τέσσερις πιθανότητες εμφάνισης στη τελευταία θέση της κάθε δωδεκάδας (mutual information) μιας από τις τέσσερις δυνατές γενετικές (A, C, G, T), ανάλογα με την επιλογή που θα γίνει στο training data set. Αξίζει να σημειωθεί ότι τα δέντρα αυτά εξαρτώνται μόνο από το training data set. Δηλαδή σε περίπτωση αλλαγής εισόδου ακολουθιών (sequence) παραμένουν τα ίδια.

Για κάθε orf που δέχεται η συνάρτηση γίνεται η διάσχιση του δέντρου μέχρι τον υπολογισμό της τελικής πιθανότητας ανά κλήση. Ξεκινώντας από τον πρώτο κόμβο του δέντρου ανάλογα με την είσοδό μας γίνεται η διάσχιση αυτού. Αυτό επαναλαμβάνεται μέχρι το τέλος της ακολουθίας βάσεων. Εάν μέσα στο δέντρο υπάρχει η τιμή -1 και η

ακολουθία διασχίσει αυτό τον κόμβο τότε επιστρέφεται η πιθανότητα του κόμβου του πατέρα και η συνάρτηση τερματίζεται με αυτό σαν τελικό σκορ. Αυτό σημαίνει ότι δεν υπάρχουν άλλες έγκυρες διαθέσιμες τιμές.

Όπως αναφέραμε στην αρχή της υποενότητας η συνάρτηση καλείται από την graph.c. Η συνάρτηση κάθε φορά καλείται για μία δωδεκάδα βάσεων και επιστρέφει την πιθανότητα της. Μέσα στην graph και για την ίδια δωδεκάδα βάσεων γίνεται η κλήση της συνάρτησης για κάθε ένα από τα πλαίσια ανάγνωσης, αλλά και για το non-coding μοντέλο. Η πιθανότητα επιστρέφεται και λογαριθμίζεται τόσο στη non-coding περιοχή όσο και στις coding περιοχές. Τέλος ο λογάριθμος της πιθανότητας της non-coding αφαιρείται από το λογάριθμο της πιθανότητας της coding περιοχής και στη συνέχεια αποθηκεύεται σε ένα πίνακα. Έτσι υλοποιήσαμε και το κομμάτι επιστροφής. Τα αποτελέσματα της κάθε κλήσης της συνάρτησης είναι τα coding και non-coding.

TCod[m][j][i]=PDom[m][j][i]+TCod[m][j][i-1]+log2(coding)-log2(noncoding). Όπου το i είναι το μήκος της ακολουθίας βάσεων και το j κυμαίνεται από το μηδέν μέχρι και το δυο, αφού αφορά το codon του γενετικού κώδικα. Δηλαδή το j εκπροσωπεί την ομαδοποίηση των τριών νουκλεοτιδίων ή αλλιώς της τριπλέτας. Το m είναι ο αριθμός των training data sets που μπορούν να μπούνε παράλληλα για τη σύγκριση τους. Για τώρα, μέχρι την περαιτέρω αναβάθμιση του αλγορίθμου, είναι ίσο με το μηδέν. Ακολουθεί το flow chart της συνάρτησης.



Σχήμα 3.3-4: Flow chart συνάρτησης get_prob_of_window1

Κεφάλαιο 4

Αρχιτεκτονική Σχεδίασης

Στο κεφάλαιο αυτό περιγράφεται και αναλύεται η σχεδίαση και η υλοποίηση του συστήματος μας. Αρχικά περιγράφεται η αρχιτεκτονική της σχεδίασης που υλοποιεί τη διάσχιση της δενδρικής δομής και την εξαγωγή της πιθανότητας για μια δωδεκάδα βάσεων, όπως υλοποιείται από τη συνάρτηση get_prob_of_window1 στο software. Η υλοποίηση της αρχιτεκτονικής σχεδίασης έγινε με τα εργαλεία XILINX ISE Design Suite 10.1 και XILINX EDK Design Suite 10.1.

Στη συνέχεια γίνεται μια περιγραφή της διασύνδεσης της παραπάνω αρχιτεκτονικής με το module του PCI-express, ώστε να μπορέσουμε να έχουμε επιστροφή των αποτελεσμάτων στο PC. Τέλος αναλύεται η υλοποίηση του τελικού συστήματος όπου περιλαμβάνει την αποστολή δεδομένων από το PC στην σχεδίαση μέσω PCI-express, και στην επιστροφή των αποτελεσμάτων στο PC πάλι μέσω PCI-express.

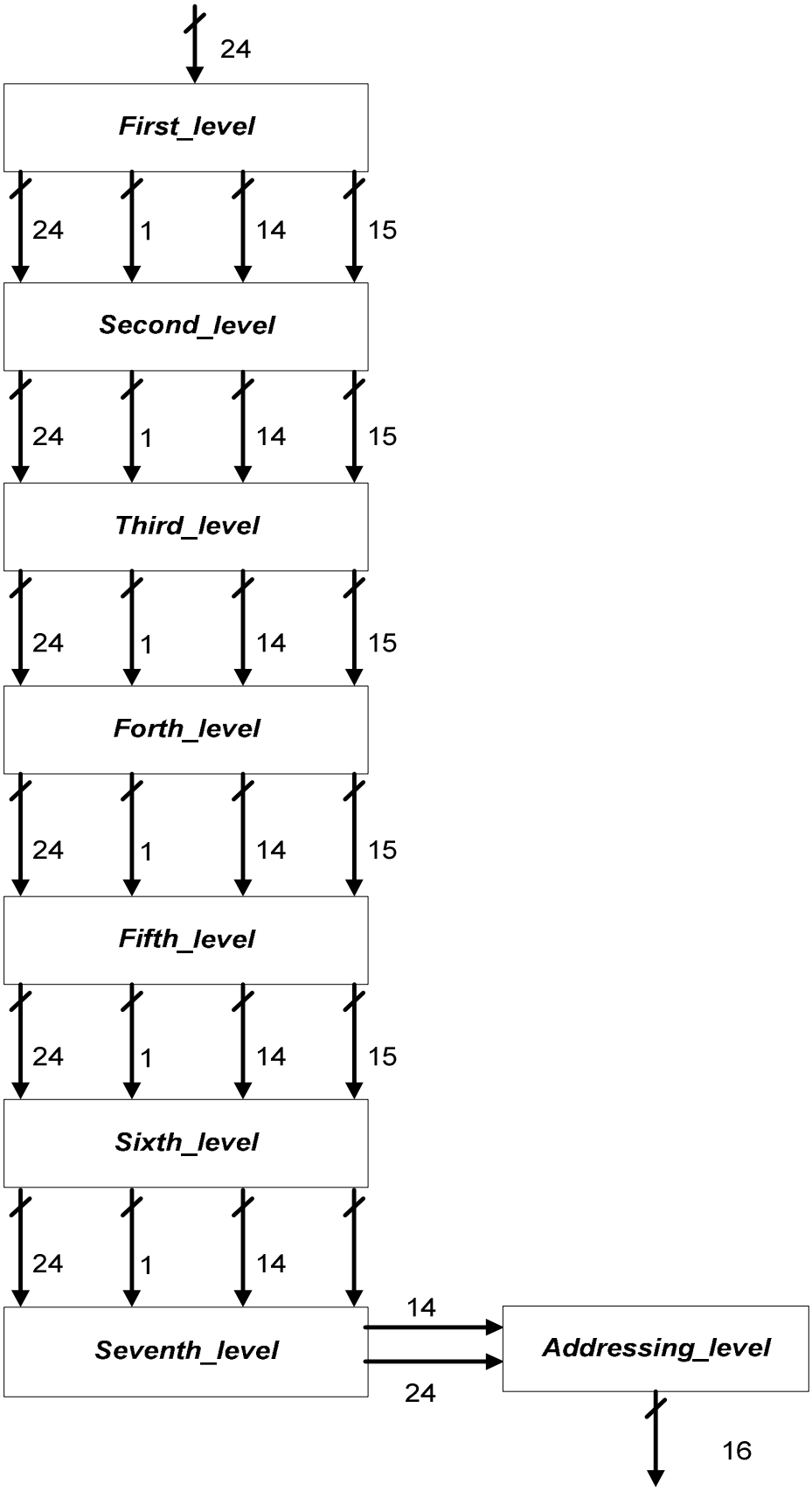
4.1 Περιγραφή της σχεδίασης και της υλοποίησης της συνάρτησης get_prob_of_window1

Αρχικά, θα γίνει περιγραφή της αρχιτεκτονικής σε hardware που υλοποιεί τη συνάρτηση του software get_prob_of_window1. Η συνάρτηση, όπως αναφέρθηκε στο προηγούμενο κεφάλαιο, δέχεται ως είσοδο μια δωδεκάδα βάσεων και διασχίζοντας μια δενδρική δομή επιστρέφει μια τιμή πιθανότητας η οποία στη συνέχεια λογαριθμίζεται. Για την υλοποίηση σε hardware έγινε μια αντιστοιχία των βάσεων γενετικού υλικού στις αντίστοιχες τιμές, όπως φαίνεται παρακάτω: A => “00”, C => “01”, G => “10”, T => “11”. Αρχικά θα γίνει εκτενής περιγραφή για την υλοποίηση της αρχιτεκτονικής μας στο ISE και θα ακολουθήσει η εφαρμογή της στον οδηγό του PCI-express μέσω EDK.

4.1-1 Περιγραφή της σχεδίασης και της υλοποίησης της συνάρτησης στο ISE

Όπως έχει αναφερθεί παραπάνω, το σύστημα μας είναι υλοποιημένο σε pipeline. Στο σχήμα 4.1-1 παρουσιάζεται το block diagram του περιεχομένου του κάθε δέντρου (TREE MODEL). Στα σχήματα που ακολουθούν παρουσιάζεται κάθε φορά ένα μέρος του διαγράμματος της σχεδίασης της αρχιτεκτονικής μας (block diagram). Πιο συγκεκριμένα: στο σχήμα 4.1-2 φαίνονται τα πρώτα δυο επίπεδα, στο σχήμα 4.1-3 φαίνεται το τελευταίο επίπεδο.

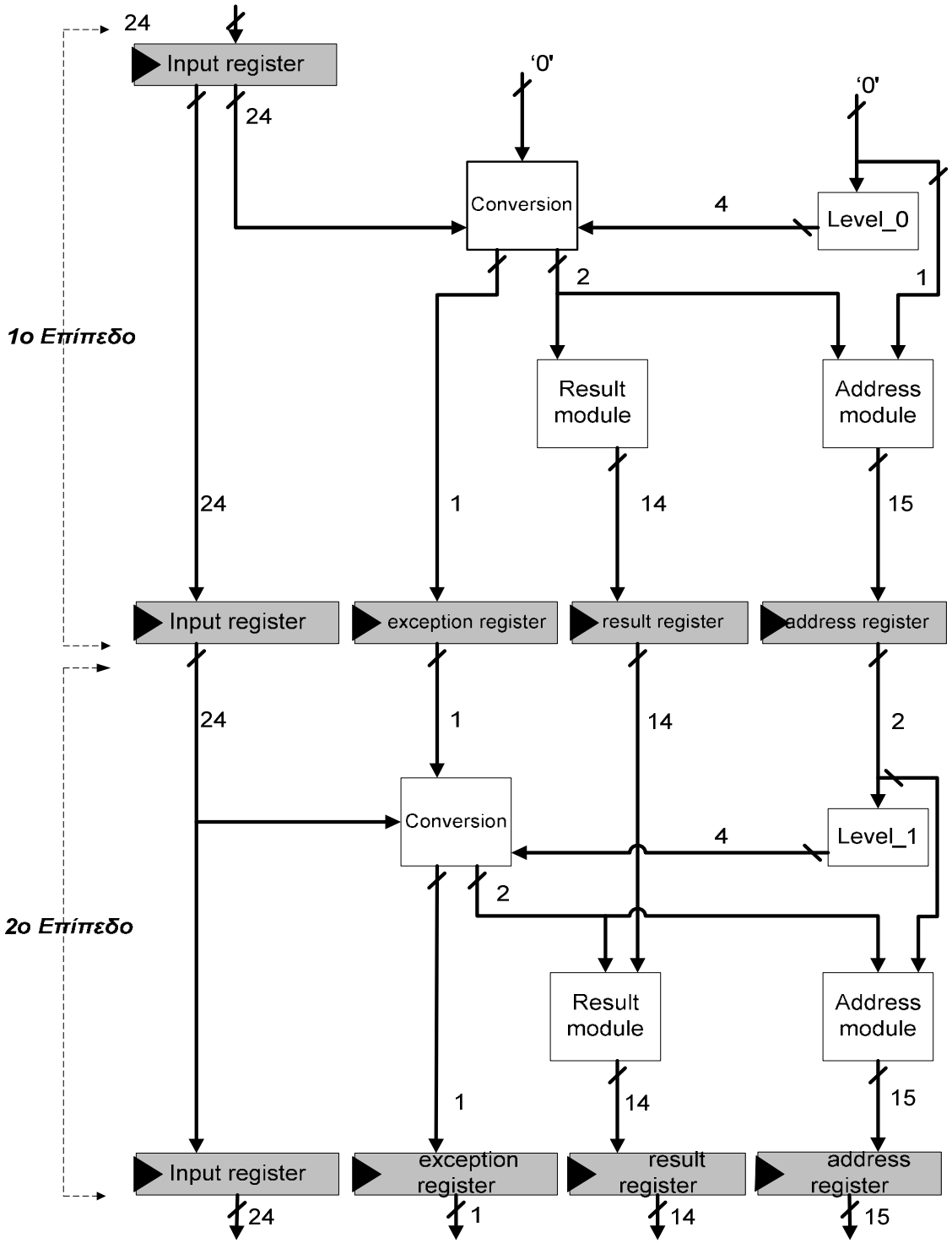
Μνήμες: Τα δένδρα στο software είναι οχτώ επιπέδων. Αποτελούνται από 21465 κόμβους και ο καθένας αποθηκεύει τις τέσσερις πιθανότητες εμφάνισης στη τελευταία θέση της κάθε δωδεκάδας (mutual information) μιας από τις τέσσερις δυνατές γενετικές βάσεις (A, C, G, T). Για να μειώσουμε το χώρο που καταλαμβάνει το δέντρο στη σχεδίαση μας δεν περιλάβαμε τις πιθανότητες που έχουν οι εσωτερικοί κόμβοι και επίσης δεν περιλάβαμε την πληροφορία του τελευταίου επιπέδου, αφού αυτά δε μας χρειάζονται. Η μορφή της σχεδίασης μας είναι pipeline δέντρο εφτά επιπέδων το οποίο δέχεται ως είσοδο 24 bits (12 βάσεις * 2 bits της κάθε γενετικής βάσης).



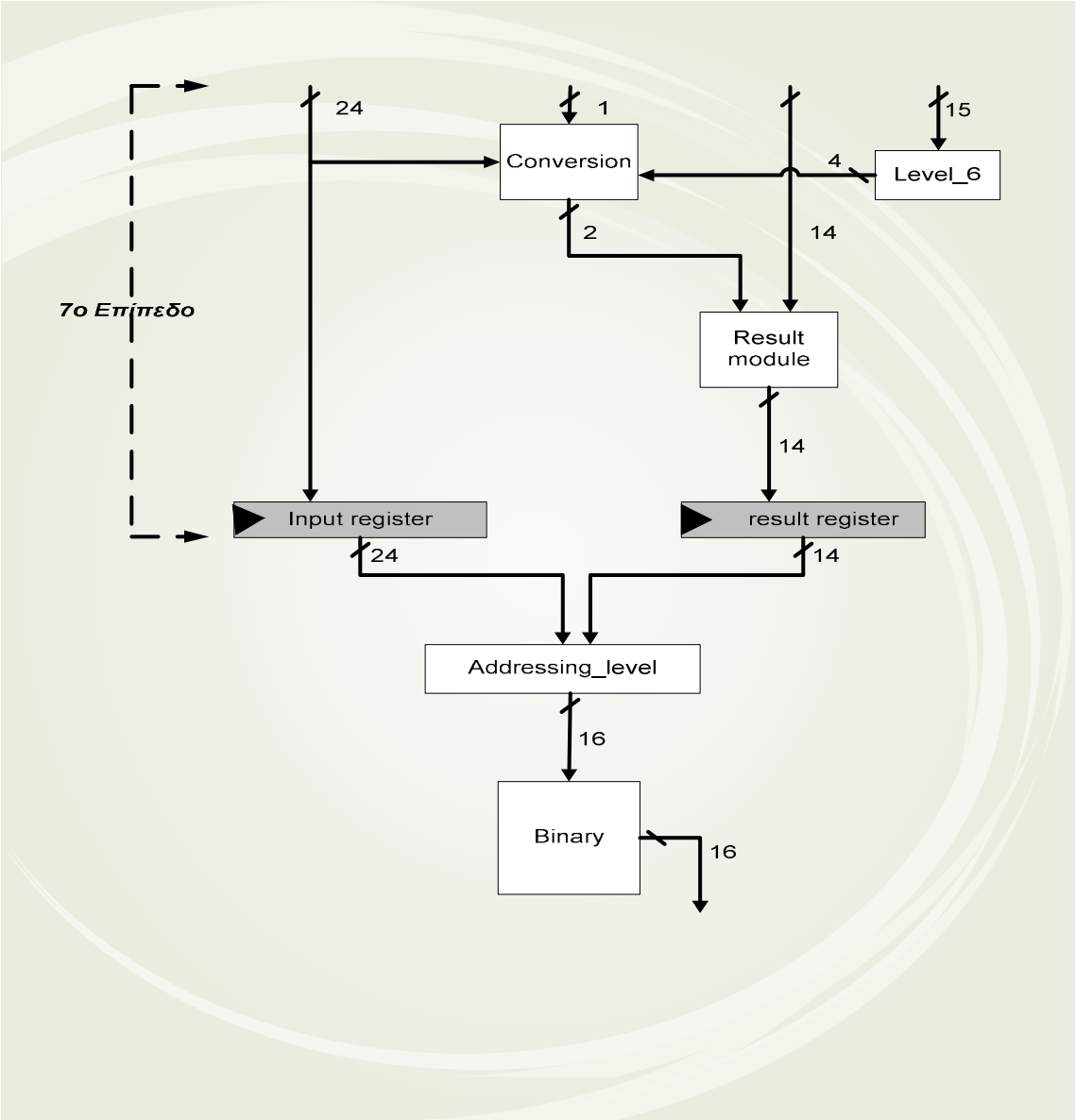
Σχήμα 4.1-1: Block diagram του περιεχομένου του κάθε δένδρου.

Λόγω του γεγονότος ότι οι μνήμες είχαν δύο θύρες ανάγνωσης και για εκμετάλλευση της παραλληλίας σε κάθε δενδρική δομή δίνονται δυο δωδεκάδες βάσεων σε κάθε κύκλο. Έτσι το κάθε top_level δέχεται μια είσοδο των 28 bits. Από το (27 down to 4) είναι είσοδος για τον υπολογισμό της μιας πιθανότητας, ενώ τα (25 down to 2) bits είναι για τον υπολογισμό της δεύτερης πιθανότητας. Ο λόγος για τον οποίο χρησιμοποιήσαμε 28 bits shift_register και όχι 26 bits είναι γιατί η είσοδος μας είναι των 4 bits και με αυτό τον τρόπο δε χάνουμε δεδομένα εισόδου στο orf, το οποίο είναι 24 bits, αφού η κάθε βάση αντιπροσωπεύεται από 2 bits. Οι δυο αυτοί υπολογισμοί λειτουργούν παράλληλα και ακολουθούν τα ίδια βήματα. Θα ακολουθήσει η περιγραφή ενός εκ των δυο συστημάτων.

Input _ register: Τα 24 bits που προέρχονται από τον *shift_register* αρχικά αποθηκεύονται σε έναν pipeline καταχωρητή *(Input_register*).



Σχήμα 4.1-2 : Τα δύο πρώτα επίπεδα της αρχιτεκτονικής μας.



Σχήμα 4.1-3: Τα δύο τελευταία επίπεδα της αρχιτεκτονικής μας.

Conversion: Στη συνέχεια αυτά τα δεδομένα μπαίνουν σαν είσοδο στο *conversion*. Είσοδοι σε αυτό το component είναι και τα περιεχόμενα της μνήμης, καθώς επίσης και ένα σήμα 1 bit το οποίο παίρνει την τιμή '1' όταν τα περιεχόμενα της μνήμης είναι invalid. Η πρώτη διεύθυνση της μνήμης και το πρώτο σήμα exception είναι αρχικοποιημένα στο μηδέν σε ένα αρχικό σήμα. Έτσι η μνήμη level_0 επιστρέφει τα δεδομένα στα οποία δείχνει η πρώτη διεύθυνση και το exception είναι ίσο με '0'. Μέσα στο conversion καθορίζονται ποιά 2 bits (valid_bits), ανάλογα με το περιεχόμενο της μνήμης, θα χρησιμοποιηθούν από το orf. Στην περίπτωση όπου τα περιεχόμενα της μνήμης είναι Invalid, δηλαδή είναι ίσα με -1, τότε τα δύο bits και όλα όσα ακολουθούν στους επόμενους κύκλους παίρνουν τη τιμή "00". Αυτό γίνεται γιατί δεν υπάρχει κάποια χρήσιμη πληροφορία στα επόμενα δέντρα στις αντίστοιχες διευθύνσεις μνήμης. Αυτά είναι η έξοδος του conversion, αλλά και η είσοδος του address_module και result_module.

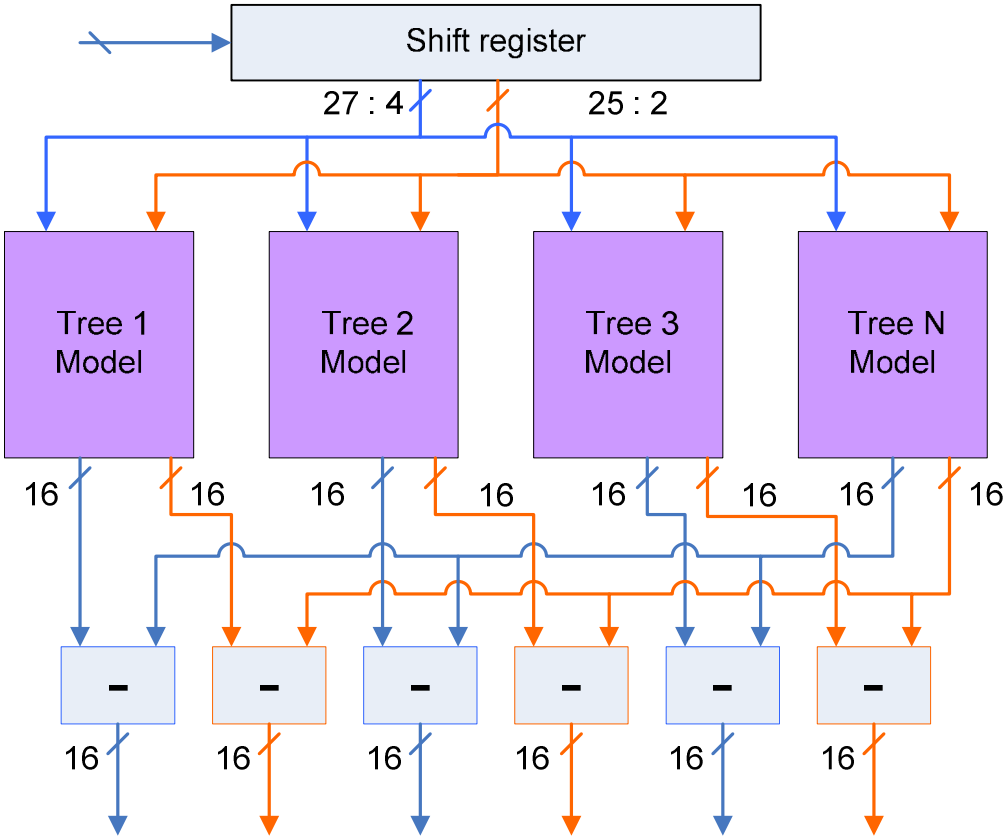
Address_module: Το address_module υπολογίζει τη καινούρια διεύθυνση μνήμης για το επόμενο επίπεδο έχοντας σαν είσοδο την προηγούμενη διεύθυνση που έχει υπολογιστεί από το αντίστοιχο component του προηγούμενου επιπέδου. Το αποτέλεσμά της αποθηκεύεται σε έναν pipeline καταχωρητή (*address_register*).

Result _ module: Το result_module με ολίσθηση κάθε φορά των 2 valid bits, που προήλθαν από το conversion, υπολογίζει τη διεύθυνση της τελευταίας μνήμης (*binary*). Το αποτέλεσμα αυτού του component και πάλι αποθηκεύεται σε έναν pipeline καταχωρητή (result_register).

Όταν η τιμή φθάσει στο τελικό επίπεδο περιέχει την τελική διεύθυνση της binary μνήμης που περιλαμβάνει την τελική πιθανότητα. Η παραπάνω διαδικασία επαναλαμβάνεται σε κάθε ένα από τα pipeline stages μέχρι να υπολογιστεί η τελευταία διεύθυνση των 14 bits. Όπως φαίνεται στο σχήμα 4.1-2 το τελευταίο επίπεδο είναι λίγο τροποποιημένο. Οι έξοδοι του προέρχονται από τον result_register και τον input_register είναι είσοδοι στο component addressing level. Το 14 bits σήμα που είναι έξοδος από τον result_register το ενώνουμε με τα 2 τελευταία bits του εισερχόμενου ORF. Τα 16 αυτά bits διευθυνσιοδοτούν τη binary μνήμη. Η μνήμη αυτή στην ουσία είναι ένα lookup table το οποίο έχουμε γεμίσει με τις λογαριθμισμένες πιθανότητες δρομολόγησης του δέντρου, ανάλογα με την ακολουθία εισόδου. Έτσι γλυτώνουμε χρόνο συγκριτικά με το software που λογαριθμίζει τις πιθανότητες. Ο λόγος για τον οποίο δεν αφήσαμε τις τιμές μας σε 32 bits floating point ακρίβειας σε αυτή τη μνήμη, αλλά τους έχουμε μετατρέψει σε 16 bits floating point ακρίβειας, είναι γιατί δεν επαρκούσαν οι πόροι της FPGA και συγκεκριμένα δεν ήταν αρκετά τα διαθέσιμα block ram του αναπτυξιακού (FPGA).

4.1-2 Περιγραφή συστήματος

Σαν είσοδο το σύστημα μας δέχεται 4 bits τη φορά τα οποία είναι είσοδος σε έναν shift_register των 28 bits. Τα 4 Tree Models όπως φαίνονται και στο σχήμα λειτουργούν παράλληλα. Τα τρία πρώτα Tree Models (Tree_Model_1, Tree_Model_2, Tree_Model_3) αφορούν τις coding περιοχές, ενώ το Tree_Model_N αφορά τη non-coding περιοχή (σχήμα 4.1-4). Ουσιαστικά η μόνη διαφορά του κάθε Tree_Model είναι τα δέντρα εκπαίδευσης που έχουν εσωτερικά σαν μνήμες. Το κάθε Tree_Model έχει 2 εξόδους των 16 bits. Για να γίνει πιο κατανοητή η σχεδίαση θα αναφερθούμε στο Tree_Model_1 και Tree_Model_N. Για το Tree_Model_1 έχουμε τις εξόδους Result_1a και Result_1b και για το Tree_Model_N έχουμε αντίστοιχα τις Result_Na και Result_Nb. Τα a και b έχουν να κάνουν με τις δύο διαφορετικές δωδεκάδες βάσεων που έχουμε ως είσοδο στο σύστημα μας ανά κύκλο.



Σχήμα 4.1-4: Block diagram της αρχιτεκτονικής μας.

Όπως έχουμε πει στο κεφάλαιο 3 η συνάρτηση get_prob_of_window1 στο software επιστρέφει ανά κλήση το αποτέλεσμα της πιθανότητας δρομολόγησης στο δέντρο ανάλογα το orf εισόδου. Για να γίνει αυτό στο hardware, επιστρέφουμε τιμή των 16 bits floating point ακρίβειας. Ο λογάριθμος της πιθανότητας της non-coding περιοχής αφαιρείται από το λογάριθμο της πιθανότητας της coding περιοχής και το αποτέλεσμα αυτών αποθηκεύεται σε έναν πίνακα. Έτσι το Result_Na αφαιρείται από το Result_1a και το Result_Nb αντίστοιχα αφαιρείται από το Result_1b. Αυτό γίνεται μέσα στους αφαιρέτες του σχήματος. Οι αφαιρέτες έχουν δημιουργηθεί από το εργαλείο coregen της Xilinx.

4.1-3 GlimmerHMM αρχιτεκτονική vs. Glimmer αρχιτεκτονική

Η υλοποίηση του αλγορίθμου Glimmer για προκαρυωτικούς οργανισμούς έχει υλοποιηθεί σε αναδιατασσόμενη λογική και δημοσιεύθηκε σε διεθνές συνέδριο [αναφορά] ->14, όπως αναφέρθηκε και στο Κεφάλαιο 2. Σε αυτή την παράγραφο θα γίνει μία σύγκριση των δύο αρχιτεκτονικών σημειώνοντας τις ομοιότητες και τις διαφορές τους. Η βασική ομοιότητα μεταξύ των δύο υλοποιήσεων αφορά την δομή του δέντρου σε hardware. Δηλαδή, το ότι και στις δύο υλοποιήσεις η απεικόνιση των δενδρικών δομών είναι pipeline ανά επίπεδο δένδρου. Οι διαφορές μεταξύ των δυο αρχιτεκτονικών είναι:

1. Στην υλοποίηση του Glimmer στους ενδιάμεσους κόμβους εκτός από το mutual information (γενετική βάση), αποθηκεύονται και οι πιθανότητες που αντιστοιχούν σε αυτή. Επομένως, η πληροφορία που απαιτείτο να αποθηκευτεί ήταν πολύ περισσότερη σε σχέση με τη δική μας σχεδίαση.

2. Επίσης, στην αρχική σχεδίαση υπολογίζεται κάθε φορά ο λογάριθμος της πιθανότητας χρησιμοποιώντας δύο lookup tables. Αντίθετα, στη δικιά μας αρχιτεκτονική έχουμε δημιουργήσει ένα lookup table στο τελευταίο επίπεδο που κρατάει όλες τις ήδη λογαριθμισμένες πιθανότητες ξεχωριστά και διευθυνσιοδοτείται ανάλογα με την δρομολόγηση του δέντρου που γίνεται σύμφωνα με την είσοδό μας.

3. Στη δικιά μας αρχιτεκτονική η ακρίβεια της κάθε πιθανότητας είναι 16 bits floating point μονής ακρίβειας, ενώ στην πρώτη υλοποίηση χρησιμοποιήθηκαν 32 bits floating point μονής ακρίβειας. Παρότι έχουμε μικρότερη ακρίβεια από την πρώτη σχεδίαση και κατά συνέπεια μικρή απώλεια σφάλματος εντούτοις η δική μας αρχιτεκτονική υλοποιήθηκε σε μία μεσαία μεγέθους FPGA σε αντίθεση με την αρχική σχεδίαση που απαιτούσε τη χρήση της μεγαλύτερης FPGA της οικογένειας Virtex 5.

4. Τέλος, στην πρώτη σχεδίαση το τελικό αποτέλεσμα είναι αυτό που απαιτούνταν, ενώ στη δική μας αρχιτεκτονική παίρνουμε ένα αποτέλεσμα ανά κύκλο.

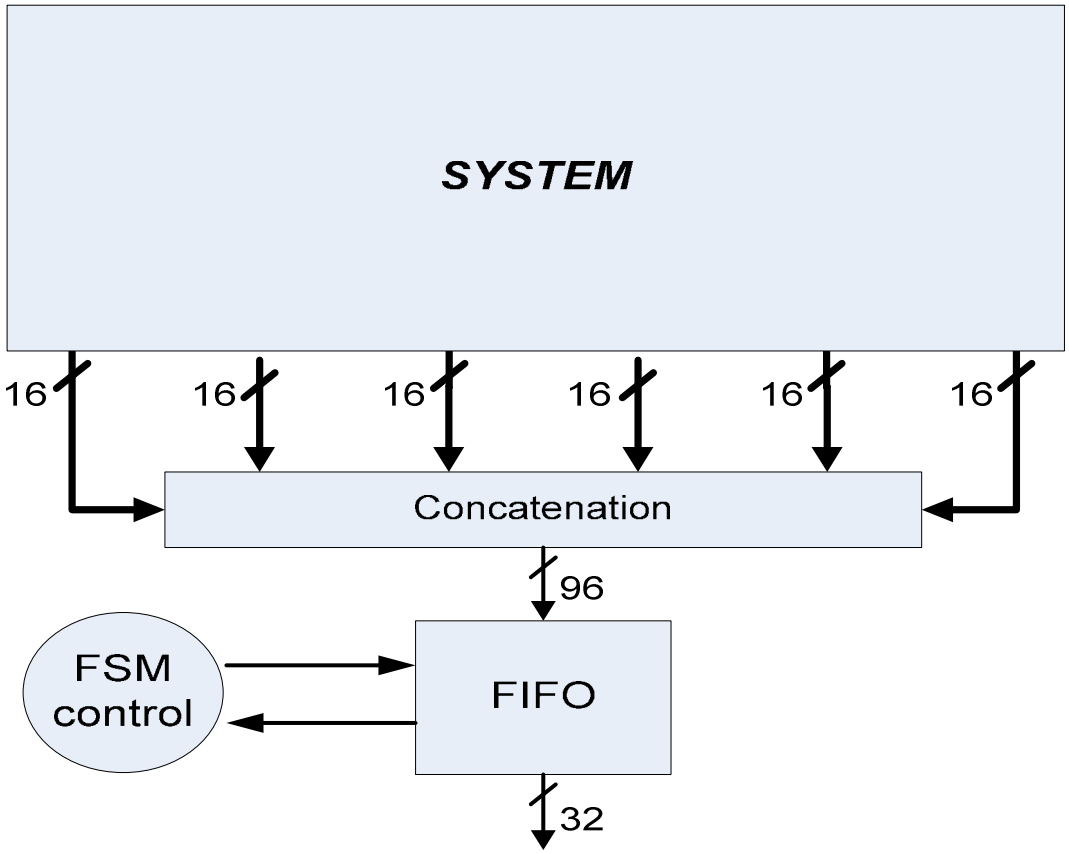
4.2 Περιγραφή της σχεδίασης και της υλοποίησης της συνάρτησης στο EDK

Ο τελικός στόχος ήταν η ένωση της παραπάνω αρχιτεκτονικής μέσω του PCI-express με το PC και εν συνεχεία με το υπόλοιπο software του αλγορίθμου. Ο οδηγός του PCI-express και το εργαλείο που χρησιμοποιήσαμε για χρήση του οδηγού του PCI-express έχουν υλοποιηθεί από προπτυχιακούς φοιτητές στις διπλωματικές τους εργασίες (Καρτσωνάκης, Γεωργιάδης).

Στην αρχιτεκτονική του PCI-express υπάρχει υλοποιημένη μια διεπαφή επικοινωνίας με σχεδιάσεις των χρηστών φέροντας το όνομα my_black_box. Τα σήματα εισόδου εξόδου καθώς και η σκοπιμότητα τους φαίνονται στους πίνακες 4.2-1 και 4.2-2 αντίστοιχα. Τα δεδομένα εισόδου καθώς και εξόδου πρέπει να είναι 32 bits, αφού αυτό

ορίζει ο οδηγός του PCI-express. Έτσι στην είσοδο εμείς ως δεδομένα βάζουμε 4 bits τη φορά, δηλαδή δύο νέες γενετικές βάσεις κάθε κύκλο.

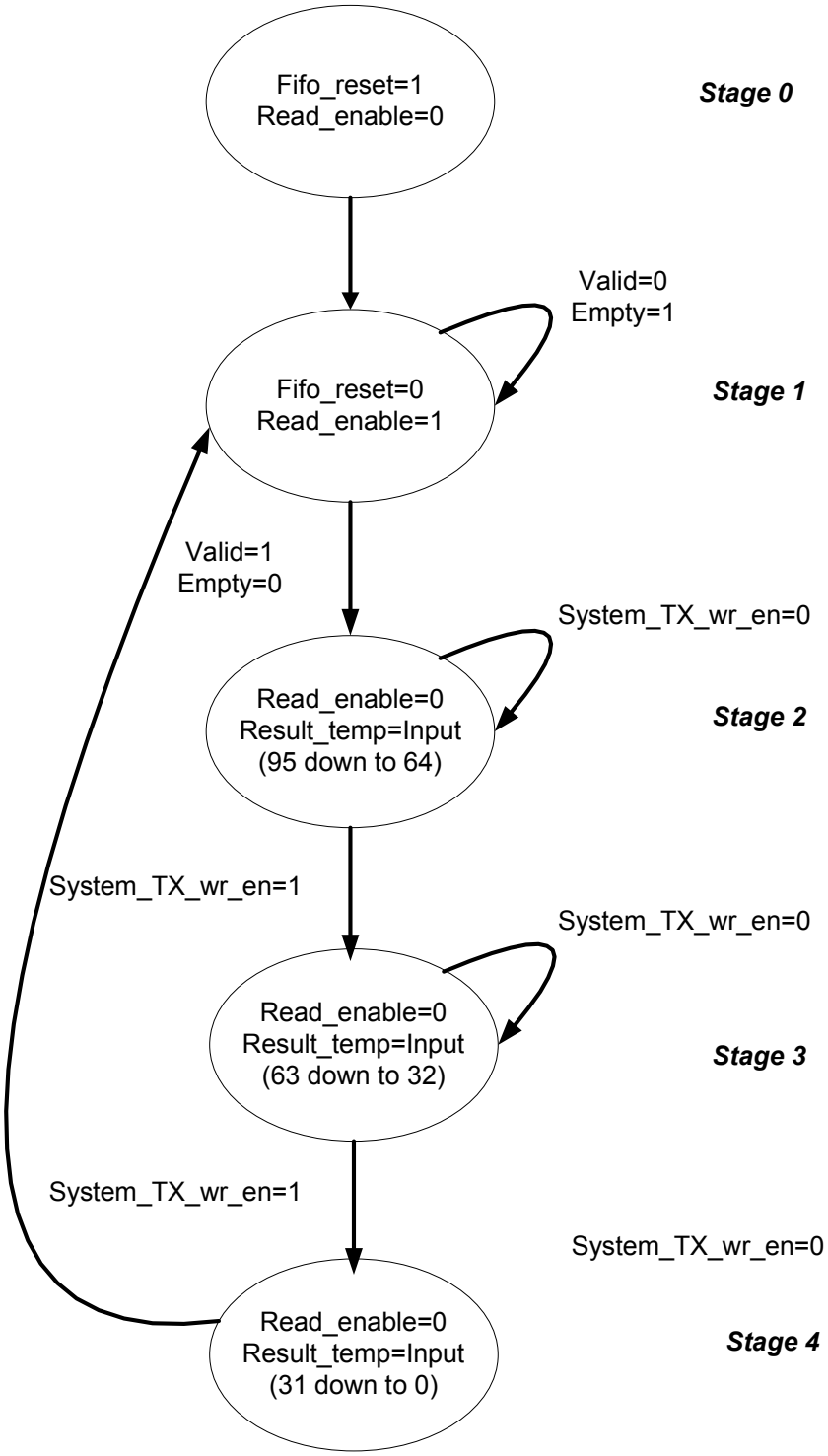
Η έξοδος της αρχιτεκτονικής μας πριν τη προσαρμογή για χρήση του οδηγού PCI-express ήταν έξι τιμές, των 16 bits η κάθε μία, οι οποίες προέρχονται από τους αφαιρέτες (Σχήμα 4.1-4). Για να γίνει προσαρμογή έπρεπε να προστεθεί μια FIFO στην οποία θα αποθηκεύονται τα δεδομένα που βγαίνουν από το σύστημα μας ανά κύκλο. Οι έξι αυτές τιμές (sub_1a & sub_1b & sub_2a & sub_2b & sub_3a & sub_3b) έχουν ενωθεί σε ένα εσωτερικό σήμα και είναι είσοδος στην τελική μας FIFO που έχει μήκος 96 bits, στη σχεδίαση της αρχιτεκτονικής μας (σχήμα 4.1-5). Το πια πρέπει να είναι η έξοδος από τη FIFO καθορίζεται από την μονάδα ελέγχου (fsm control) που φαίνεται στο σχήμα 4.1-6.



Σχήμα 4.1-5: Block diagram της σχεδίασης μας

Τα χαρακτηριστικά της FIFO είναι ότι είναι DRAM (distributed RAM) με κοινό ρολόι για εγγραφή και ανάγνωση. Επίσης η ανάγνωση από αυτήν γίνεται στον ίδιο κύκλο στον οποίο ζητήθηκε (First-Word Fall-Through). Το πλάτος της όπως αναφέρθηκε παραπάνω είναι 96 bits και το βάθος της 4096 bits. Τέλος έγινε η επιλογή του σήματος valid για έλεγχο στη μονάδα ελέγχου και επιλέχθηκε η χρήση επιπλέον λογικής για πιο ακριβή μέτρηση

δεδομένων (use extra logic for more accurate data counts). Η επιλογή αυτής της μεθόδου έγινε επειδή έχουμε επιλέξει First-Word Fall-Through, άρα έχουμε latency μηδέν και με την επιλογή αυτή μπορούμε να έχουμε πιο ακριβή δεδομένα. Η FIFO αυτή δέχεται και επιστρέφει στον ίδιο κύκλο 96 bits, που στην ουσία είναι οι έξι 16bits ποσότητες που έχουν υπολογιστεί παραπάνω. Ανά τρείς κύκλους διαβάζουμε μια νέα ποσότητα των 96 bits, που είναι αποθηκευμένη στη FIFO της σχεδίασης, την οποία σπάμε σε τρία 32bit και ανά κύκλο τις γράφουμε στη FIFO εξόδου (TX_FIFO) της αρχιτεκτονικής χρήσης του PCI-express. Επίσης στο software του PCIE κάνουμε τις αντίστοιχες μετατροπές οι οποίες θα αναφερθούν στη συνέχεια. Να σημειωθεί ότι η RX_FIFO είναι η FIFO εισόδου στο σύστημά μας, ενώ η TX_FIFO είναι η FIFO εξόδου.



Σχήμα 4.1-6: Μονάδα ελέγχου της δικιάς μας FIFO.

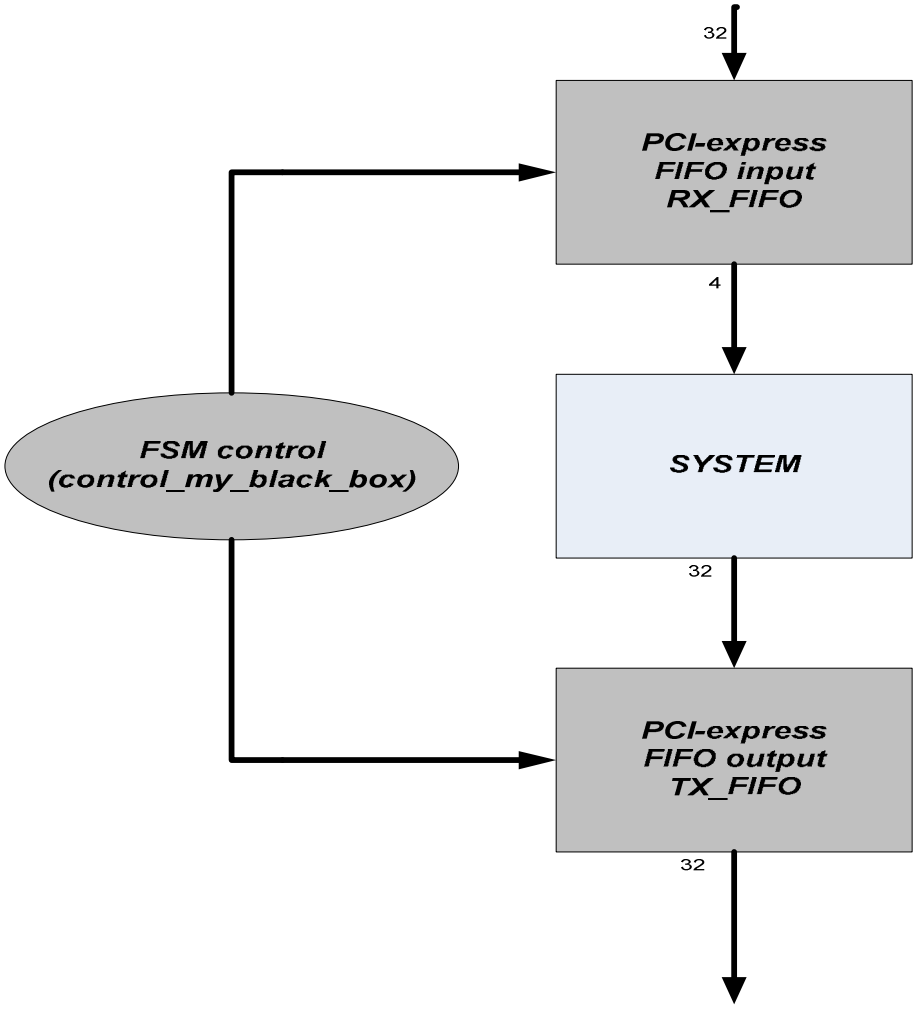
Σήματα εισόδου my_black_box	Χρησιμότητα σημάτων
PCIE_CLK	Ρολόι του PCI-express στα 62,5 MHz
RST	Reset συστήματος το οποίο έχει θετική λογική
FSL_Rst	Επέμβαση χρήστη για αρχικοποίηση συστήματος εν ώρα λειτουργίας
FSL_CLK	Ρολόι του microblaze
RX_engine_data	Δεδομένα εισόδου των 32 bits που λαμβάνουμε από το PCI-express
RX_engine_WR_EN	Σήμα ενεργοποίησης εγγραφής στην RX_FIFO
TX_engine_RD_PCIE	Σήμα ενεργοποίησης ανάγνωσης από το PCI-express
TX_fifo_full	Σήμα ελέγχου για το εάν η TX_fifo είναι γεμάτη
TX_fifo_empty	Σήμα ελέγχου για το εάν η TX_fifo είναι άδεια
RX_fifo_full	Σήμα ελέγχου για το εάν η RX_fifo είναι γεμάτη
RX_fifo_empty	**Σήμα ελέγχου για το εάν η RX_fifo είναι άδεια**

Πίνακας 4.2-1: Σήματα εισόδου my_black_box

Σήματα εξόδου my_black_box	Χρησιμότητα σημάτων
RX_engine_CLK	Ρολόι ανάγνωσης της RX-FIFO από το PCI-express
RX_engine_RD_EN	Σήμα λήψης ενεργοποίησης για ανάγνωση από την RX-FIFO
TX_engine_CLK	Ρολόι εγγραφής της TX-FIFO από το PCI-express
TX_engine_data	Δεδομένα αποστολής των 32 bits προς το PCI-express
TX_engine_WR_EN	Σήμα ενεργοποίησης εγγραφής από την TX_FIFO
TX_engine_RD_EN	Σήμα ενεργοποίησης ανάγνωσης την TX_FIFO
FIFO_RST	Αρχικοποίηση των FIFO του συστήματος

Πίνακας 4.2-2: Σήματα εισόδου my_black_box

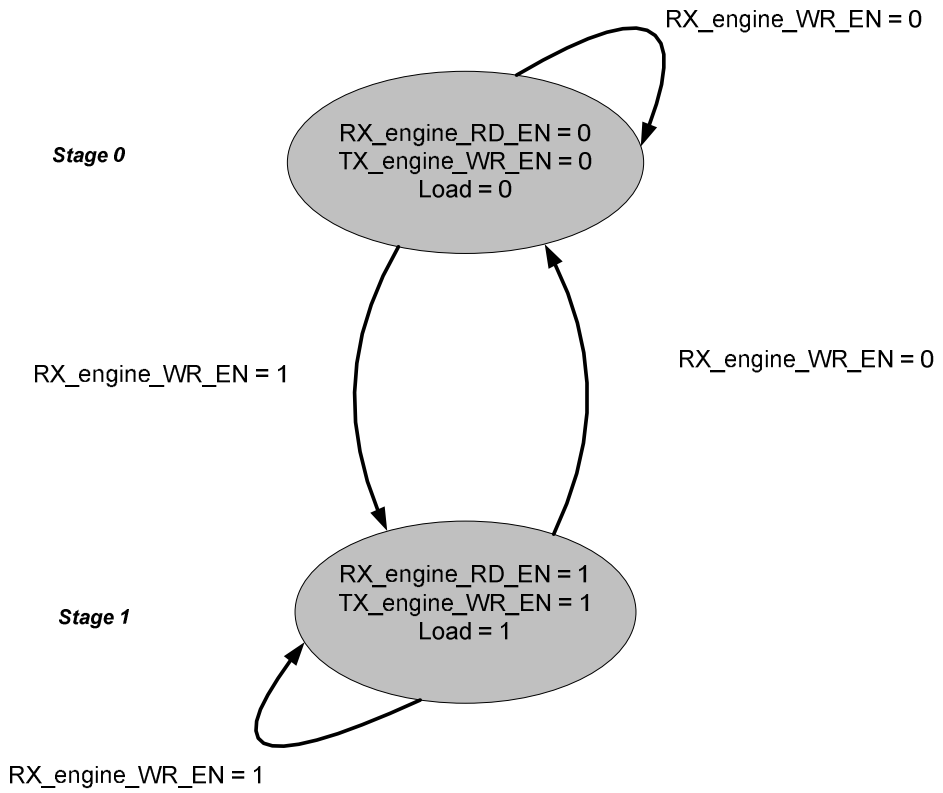
Όλο το σύστημα μας (σχήμα 4.2-1) καθώς και οι TX_FIFO και RX_FIFO έχουν συγχρονιστεί με το PCIE_CLK (RX_engine_CLK<= PCIE_CLK , TX_engine_CLK<= PCIE_CLK). Το ρολόι όπως φαίνεται και στον *Πίνακα 4.2-1* είναι στα 62,5 MHz. Αυτό σημαίνει ότι έχουμε περιορισμό στην απόδοση του συστήματος μας. Η εγγραφή και η ανάγνωση δεδομένων από τη FIFO και από προς τον «έξω κόσμο» γίνεται στα 62,5 MHz που είναι η συχνότητα του ρολογιού του PCI-express.



Σχήμα 4.2-1: Block diagram του συστήματος

Το control_my_black_box είναι η μονάδα ελέγχου του my_black_box. Δέχεται σαν εισόδους CRL, reset, RX_engine_WR_EN, TX_engine_RD_PCIE, TX_fifo_full, RX_fifo_empty, TX_fifo_empty και τέλος empty_fifo, ενώ σαν σήματα εξόδου έχουμε RX_engine_RD_EN, TX_engine_WR_EN, TX_engine_RD_EN και τέλος το load. Τα σήματα empty_fifo και load

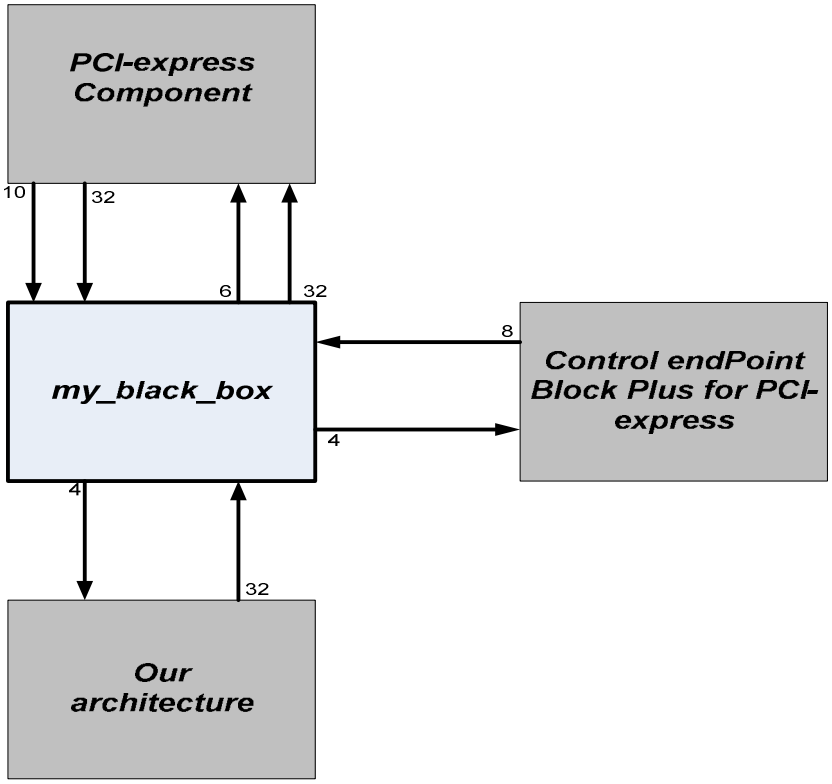
που αναφέραμε έχουν προστεθεί από εμάς με σκοπό τη διευκόλυνσή μας για τον έλεγχο της δικιάς μας FIFO και για την ενεργοποίηση της αρχιτεκτονικής μας αντίστοιχα. Στο σχήμα 4.2-2 φαίνεται η λειτουργικότητα της μονάδας ελέγχου. Όταν το σήμα εισόδου TX_engine_RD_PCIE είναι ίσο με το μηδέν τότε και το σήμα TX_engine_RD_EN παίρνει την ίδια μηδενική τιμή. Το ίδιο ισχύει και όταν παίρνει τη τιμή ένα. Εάν το σήμα RX_engine_WD_EN είναι ίσο με μηδέν τότε αυτό σημαίνει ότι δεν υπάρχουν δεδομένα για εγγραφή στην FIFO εισόδου και άρα παραμένουμε στην ίδια κατάσταση με τα σήματα TX_engine_WR_EN και RX_engine_RD_EN να είναι επίσης ίσα με το μηδέν. Εάν ενεργοποιηθεί τότε πάμε στη δεύτερη κατάσταση και μέσα σε αυτήν ενεργοποιούμε τα παραπάνω σήματα εξόδου. Εάν το RX_engine_WD_EN γίνει και πάλι ίσο με το μηδέν τότε γυρνάμε στην αρχική μας κατάσταση. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να προσπελαστεί η όλη είσοδος μας (σχήμα 4.2-3).



Σχήμα 4.2-2: Μονάδα ελέγχου του συστήματος (control_my_black_box)

Κάτι εξίσου σημαντικό το οποίο πρέπει να αναφερθεί είναι ότι υπάρχει ένας περιοριστικός παράγοντας στη TX-FIFO (fifo εξόδου). Το βάθος της είναι 4096 θέσεις των 32 bits. Αυτό σημαίνει ότι επειδή εμείς για κάθε νέο δεδομένο εισόδου θα πρέπει να επιστρέφουμε 3 τιμές των 16 bits, επομένως για κάθε κύκλο που έχουμε δύο δεδομένα θα πρέπει να επιστρέφονται στο PC 6 τιμές των 16 bits άρα 3 τιμές των 32 bits περιορίζουμε

την είσοδο μας στα 1365 δεδομένα. Έτσι σπάμε την είσοδό μας σε τόσα δεδομένα μέσο του software κώδικα.



Σχήμα 4.2-3: Block diagram αρχιτεκτονικής συστήματος

Κεφάλαιο 5

Επιβεβαίωση Λειτουργίας και Απόδοση Συστήματος

Στο κεφάλαιο αυτό περιγράφεται η διαδικασία επιβεβαίωσης λειτουργίας του συστήματος μας καθώς και η απόδοση του. Η υλοποίηση της αρχιτεκτονικής σχεδίασης έγινε με τα εργαλεία XILINX ISE Design Suite 10.1 και XILINX EDK Design Suite 10.1. Το τελικό σύστημα προγραμματίστηκε σε Virtex-5 FPGA Family Member Device XC5VLX110T-1ff1136 ενώ τα αποτελέσματα του συγκρίθηκαν με τα αντίστοιχα του software.

5.1 Ταυτοποίηση λειτουργίας

Κατά την διαδικασία της υλοποίησης της αρχιτεκτονικής, προσομοιώθηκε η λειτουργία της επιμέρους συνάρτησης και διαπιστώθηκε η σωστή λειτουργία της. Έπειτα, αφού ενώθηκε το software με το hardware (μέσο του οδηγού του PCI-express) προσομοιώθηκε το συνολικό σύστημα ώστε να διαπιστωθεί η σωστή λειτουργία του. Για την επιβεβαίωση της λειτουργίας του συστήματος χρησιμοποιήσαμε 6 διαφορετικές εισόδους (Πίνακας 3.3-2) με κάθε ένα από τα 5 διαφορετικά training datasets. Τα αποτελέσματα που πήραμε από το hardware τα συγκρίναμε με αυτά του software και καταλήξαμε ότι έχουμε κατά μέσο όρο απώλεια ακρίβειας της τάξεως του 5*10^(-4), ενώ η μέγιστη είναι της τάξεως του για κάθε δεδομένο. Αυτό οφείλεται στη χρήση floating point αριθμών 16 bits. Επίσης, οφείλεται και στους αφαιρέτες λόγο στρογγυλοποίησης του αποτελέσματος.

5.2 Απόδοση Συστήματος

Στους πίνακες 5.2-1 και 5.2-2 βλέπουμε πόσους πόρους καταλαμβάνει η αρχιτεκτονική μας και το σύστημά μας αντίστοιχα. Οι επιπλέον πόροι που καταλαμβάνονται οφείλονται στις αλλαγές που έγιναν στην αρχιτεκτονικής μας (χρήση αφαιρετών SUB και FIFO) για να είναι εφικτή και λειτουργική η χρήση του οδηγού του PCI-express. Αξίζει να σημειωθεί ότι το PCI-express σαν αρχιτεκτονική στο EDK χρησιμοποιεί 16 από τις 148 διαθέσιμες BRAM/FIFO που σημαίνει ότι από μόνο του καταλαμβάνει περίπου ένα ποσοστό της τάξεως του 10% της FPGA.

Device utilization summary	Used	Available	Utilization
Number of Slice Registers	*4,440*	*69,120*	6%
Number of Slice LUTs	4,617	69,120	6%
Number of Occupied Slices	2,506	17,280	14%
Number of bonded IOBs	103	640	16%
Number of Block RAM/FIFO	132	148	89%
Number of BUFG/BUFGCTRLs	5	32	15%

Πίνακας 5.2-1: Χρήση πόρων FPGA αρχιτεκτονική σχεδίασης.

Device utilization summary	Used	Available	Utilization
Number of Slice Registers	8,006	69,120	11%
Number of Slice LUTs	21,936	69,120	31%
Number of Occupied Slices	8,270	17,280	47%
Number of bonded IOBs	4	640	1%
Number of Block RAM/FIFO	147	148	99%
Number of BUFG/BUFGCTRLs	10	32	31%

Πίνακας 5.2-2: Χρήση πόρων FPGA στην υλοποίηση του συστήματος.

5.3 Αποτίμηση Συστήματος

Στον πίνακα 5.3-1 βλέπουμε τη συχνότητα της σχεδίασης και του συστήματος μας. Η συχνότητα του συστήματος είναι αναγκαστικά η ίδια με το ρολόι του PCI-express. Η σχεδίαση μας έχει επηρεαστεί από τις distributed μνήμες, αλλά και τους αφαιρέτες που προσθέσαμε. Άρα ο κύκλος ρολογιού μας είναι στα 0,0086 * (10^(-6)) sec με throughput=2 (στην έξοδό μας λαμβάνουμε 2 αποτελέσματα ανά κύκλο). Το software του αλγορίθμου GlimmerHMM εκτελέστηκε στον server του πολυτεχνείου με επεξεργαστή 2 Quad core Xeon στα 2,66 GHz , με 12 GB RAM και λειτουργικό σύστημα ubundu 8.1.

	Συχνότητα
Σχεδίαση (Post Place and Route)	***126,16 MHz***
Σύστημα (Σχεδίαση και PCI-express)	***62,5 MHz***

Πίνακας 5.3-1: Συχνότητα της σχεδίασης και του συστήματος μας.

Ο μέσος χρόνος εκτέλεσης της συνάρτησης ανά βάση στον υπολογιστή για κάθε διαφορετική είσοδο, ο χρόνος εκτέλεσης του αλγορίθμου σε αναδιατασσόμενη λογική καθώς και ο χρόνος software – hardware φαίνεται στους παρακάτω πίνακες.

Number	*Input's Name*	*Input's Length*
(1)	*Gallus Gallus*	*2056*
(2)	*Ostreococus lucimarinus*	*298772*
(3)	*Leishmania braziliensis*	*4418688*
(4)	*Populus Trichocarpa*	*78889342*
(5)	*Homo Sapiens*	*201756634*
(6)	*Monodelphis*	*428681600*

Πίνακας 5.3-2: Απεικόνιση εισόδων

Input	**Function Execution Time and Speed-Up per Base**				
	Software	Hardware		Software – Hardware co-design	
	Time (sec)	Time (sec)	Speed-up	Time (sec)	Speed-up
(1)	4.86*10^(-6)	0.0043*10^(-6)	1130 x	5.739*10^(-6)	0.84 x
(2)	1.54*10^(-6)	0.0043*10^(-6)	358.14 x	5.705*10^(-6)	0.27 x
(3)	1.40*10^(-6)	0.0043*10^(-6)	325.58 x	5.753*10^(-6)	0.24 x
(4)	1.35*10^(-6)	0.0043*10^(-6)	313.95 x	5.742*10^(-6)	0.23 x
(5)	1.37*10^(-6)	0.0043*10^(-6)	318.60 x	5.751*10^(-6)	0.23 x
(6)	1.41*10^(-6)	0.0043*10^(-6)	327.91 x	5.775*10^(-6)	0.24 x

Πίνακας 5.3-3: Χρόνος και speedup εκτέλεσης συνάρτησης ανά βάση

Input	Function Execution Time and Speed-up				
	Software	Hardware		Software – Hardware co-design	
	Time (sec)	Time (sec)	Speed-up	Time (sec)	Speed-up
(1)	0.0100	8.84*10^(-6)	1131.22 x	0.1179	0.8475 x
(2)	0.4600	0.0013	358.05 x	1.7044	0.2699 x
(3)	6.1900	0.0190	325.79 x	25.4207	0.2435 x
(4)	106.5000	0.3390	313.95 x	452.9826	0.2351 x
(5)	587.2900	1.8430	318.60 x	2465.3478	0.2382 x
(6)	284.4800	0.8670	327.91 x	1165.1445	0.2442 x

Πίνακας 5.3-4: Χρόνος και speedup εκτέλεσης συνάρτησης

Training Data Set	Input	SW	HW		HW – SW codesign	
		Time (sec)	Time (sec)	Speedup	Time (sec)	Speedup
Arabidopsis	***(1)***	0.0439	0.0300	1.3333 x	0.0417	0.9570 x
	(2)	0.8606	0.4213	2.0476 x	2.12449	0.4048 x
	(3)	12.0272	5.9690	2.0202 x	31.3707	0.3832 x
	(4)	222.7182	118.4590	1.8855 x	571.1026	0.3900 x
	(5)	631.7458	483.7930	1.3108 x	2947.2978	0.2143 x
	(6)	1319.6911	731.0070	1.8073 x	1895.2845	0.6963 x
	(1)	0.2592	0.2400	1.0417 x	0.2517	0.9929 x
	(2)	2.0554	1.6467	1.2500 x	3.3444	0.6129 x

Celegans	***(3)***	29.3679	23.2947	1.2617 x	48.6907	0.6030 x
	(4)	347.1423	242.2834	1.4322 x	695.3626	0.4992 x
	(5)	1016.4534	871.2686	1.1666 x	3336.6078	0.3046 x
	(6)	2120.5573	1534.3123	1.3821 x	2699.4545	0.7855 x
Rice	***(1)***	0.1443	0.1365	1.0769 x	0.1417	0.9873 x
	(2)	1.1605	0.7286	1.6111 x	2.4244	0.4785 x
	(3)	15.5497	9.3634	1.6603 x	34.7807	0.4468 x
	(4)	247.2835	141.8042	1.7439 x	594.7826	0.4157 x
	(5)	707.1425	560.9704	1.2606 x	3026.3178	0.2337 x
	(6)	1468.0239	878.9342	1.6703 x	2044.0745	0.7182 x
Zebrafish	***(1)***	0.0635	0.0598	1.2000 x	0.0617	0.9709 x
	(2)	0.8886	0.4206	2.0952 x	2.1244	0.4142 x
	(3)	12.2963	6.1756	1.9919 x	31.5907	0.3890 x
	(4)	206.4902	101.3989	2.0366 x	554.3726	0.3725 x
	(5)	604.3573	458.634	1.3177 x	2923.9778	0.2067 x
	(6)	1223.8823	636.3745	1.9232 x	1801.5145	0.6794 x
Human	***(1)***	0.1715	0.1697	1.0625 x	0.1717	0.9895 x
	(2)	1.4674	1.1423	1.2807 x	2.8444	0.5133 x
	(3)	21.9524	17.0275	1.2897 x	42.4407	0.5172 x
	(4)	229.4975	122.7276	1.8701 x	575.7026	0.3986 x
	(5)	608.6655	493.4108	1.2336 x	2958.7578	0.2057 x
	(6)	1327.9534	741.2833	1.7914 x	1906.4245	0.6966 x

Πίνακας 5.3-4: Χρόνος και speedup εκτέλεσης αλγορίθμου

Από τα παραπάνω βλέπουμε ότι η υλοποίηση σε hardware είναι επιτυχής. Η σχεδίαση μας είναι κατά μέσο όρο 300 φορές πιο γρήγορη από ότι στο software. Αυτό οφείλεται στο ότι:

1. Έχουμε παραλληλία 8 στο σύστημα καθώς παίρνουμε είσοδο 2 χαρακτήρων ανά κύκλο (throughput=2), και για κάθε έναν από αυτούς υπολογίζουμε τις 4 πιθανότητες των διαφορετικών μοντέλων (3 coding regions και 1 non-coding) παράλληλα.
2. Επίσης, έχουμε έξι floating point αφαιρέτες οι οποίοι λειτουργούν παράλληλα (αφαίρεση των 8 πιθανοτήτων που βγαίνουν από το σύστημα) για τον υπολογισμό των τελικών αποτελεσμάτων που επιστρέφονται στο software.
3. Οι πιθανότητες που βρίσκονται στην τελική μνήμη στη σχεδίαση μας για κάθε δέντρο είναι ήδη λογαριθμισμένες.

Όσον αφορά τη λειτουργία του software με το hardware τα αποτελέσματα οφείλονται στο ότι:

1. Η συχνότητα του PCI-express είναι στα 62,5 MHz. Σε άλλη περίπτωση θα είχαμε πιο γρήγορο σύστημα.
2. Ο οδηγός του PCI-express λειτουργεί με Polling και όχι με Interrupts στην επικοινωνία μεταξύ εισόδου/εξόδου. Δηλαδή το πρόγραμμά μας κάθε φορά κάνει ανάγνωση μνήμης από το PC και αυτό μας καθυστερεί πολύ.
3. Ο driver του PCI-express είναι υλοποιημένος ώστε να δέχεται μόνο 16Kbyte σαν είσοδο. Αυτό μας περιορίζει πολύ λόγω του ότι δεν μπορούμε να εισάγουμε μεγάλες βάσεις στο σύστημα μας.
4. Τέλος κάθε φορά μέσο του PCI-express γυρνάμε 32 bits δεδομένα. Αυτό έχει ως αποτέλεσμα να πρέπει να γίνεται κάθε φορά η επεξεργασία των δύο 16 bits αποτελεσμάτων στο software, άρα έχουμε κάποια απώλεια χρόνου σε αυτό το μέρος της υλοποίησης.

Κεφάλαιο 6

Συμπεράσματα και Μελλοντικές Επεκτάσεις

Η παρούσα διπλωματική εργασία παρουσίασε την μελέτη του αλγορίθμου GlimmerHMM και την υλοποίηση της υπολογιστικά βαριάς συνάρτησης αυτού σε αναδιατασσόμενη λογική. Στόχος ήταν το σύστημα αυτό να είναι αποδοτικότερο σε σύγκριση με έναν προσωπικό υπολογιστή.

Από τα αποτελέσματα μπορούμε να συμπεράνουμε ότι ο στόχος ικανοποιείται, ειδικά όταν η παραλληλία που επιτρέπεται είναι αρκετά μεγάλη. Τα μειονεκτήματα είναι ο περιορισμός των πόρων που παρέχει η FPGA, αλλά και ο οδηγός του PCI-express για τους λόγους που αναφέραμε στο προηγούμενο κεφάλαιο.

6.1 Μελλοντικές Επεκτάσεις

Το σύστημα 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 GlimmerHMM.

Σημαντικές επεκτάσεις και αλλαγές που μπορούν να πραγματοποιηθούν είναι:

i. Αλλαγές όσον αφορά τον οδηγό του PCI-express ως προς τον τρόπο λειτουργίας του driver (αύξηση της συχνότητας, χρήση Interrupts, αύξηση των δεδομένων για upload).

ii. Υλοποίηση του συστήματος σε μεγαλύτερη Virtex 5 για εκμετάλλευση περισσότερων πόρων κυρίως όσον αφορά τα Block RAM/FIFO. Έτσι θα μπορέσουν να αφαιρεθούν οι distributed μνήμες που κάνουν το σύστημά μας πιο αργό, αφού πέφτει η συχνότητα. Επίσης με την αλλαγή της Virtex θα μπορούμε να εκμεταλλευτούμε τους περισσότερους πόρους, ώστε οι binary μνήμες της αρχιτεκτονικής μας να γίνουν 32 bits floating point ακρίβειας και πάνω σε αυτή θα

μπορούν να υπάρξουν περισσότεροι παραλληλισμοί και μεγαλύτερη ακρίβεια στο τελικό αποτέλεσμα.

Παραπομπές

Διαδίκτυο

[1] http://www.xilinx.com

[2] http://www.cbcb.umd.edu/software/GlimmerHMM

[3] http://www.ncbi.nlm.nih.gov/genomes/genlist.cgi?taxid=2759&type=0&name=Complete%20Eukaryota

[4] http://www.cs.toronto.edu/~brudno/csc2427/Lec12Notes.pdf

[5] http://genes.mit.edu

[6] http://books.google.gr/books?id=zx3UcGsVaPsC&pg=PA135&dq=description+of+TwinScan+algo&cd=2#v=onepage&q=glimmer&f=false

[7] http://genome.cshlp.org/content/10/4/523.full

[8] http://linux1.softberry.com/berry.phtml?topic=fgenesh&group=help&subgroup=gfind

[9] http://www.molquest.com/molquest.phtml?group=index&topic=gfind

[10]http://www.cs.sunysb.edu/~skiena/549/lectures/geneprediction/node1.html

Βιβλιογραφία

[1] JOHN HENDERSON, STEVEN SALZBERG, KENNETH H. FASMAN. Journal of Computational Biology. 4(2): 127-141. doi:10.1089/cmb. 4.127, 1997.

[2] Isidore Rigoutsos, G. Stephanopoulos. Systems Biology: Genomics, 2000.

[3] Krogh, A., In Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology (eds. T.Gaasterland et al.), pp.179 –186, AAAI Press, Menlo Park, CA, 1997.

[4] Allen, J.E., et al. Computational gene prediction using multiple sources of evidence. Genome Research. 14: 142-148, 2004.

[5] Burge C., Karlin S. Prediction of complete gene structures in human genomic DNA. Journal of Molecular Biology. 268(1):78-94, 1997.

[6] Brown, T.:Genomes. BIOS Scientific Publishers Ltd, 1999.

[7] Durbin, R., Eddy, S., Krogh, A., Mitchison, G.: Biologcial Sequence Analysis, Probabilistic models of proteins and nucleic acids, Cambridge University Press, 1998.

[8] W. Zhu, Y. Niu, J. Lu and G.R. Gao, Implementing Parallel Hmm-Pfam on the Earth Mulithreaded Architecture, in The 2nd IEEE Computer Society Bioinformatics Conference, 2003.

[9] J. Landman, J. Ray and J.P. Walters, BAccelerating Hmmer Searches on Opteron Processors with Minimally Invasive Recoding, in AINA _06: Proc. of the 20th International Conference on Advanced Information Networking and Applications—Volume 2 (AINA_06), IEEE Computer Society, Washington, DC, USA, pp. 628–636, 2006.

[10]Marketa J. Zvelebil, Jeremy O. Baum. Understanding bioinformatics, 1999.

[11]Thanaraj and Alan J Robinson, Briefings in Bioinformatics 1 (4) 343-356, 2000.

[12]Michael W. Berry, Umeshwar Dayal,Chandrika Kamath,David Skillicorn. Proceedings of the Fourth SIAM International Conference on Data Mining, 1998.

[13]Kevin B. Korb,Ann E. Nicholson. Bayesian artificial intelligence, 2000.

[14]*Grigorios Chrysos, Euripides Sotiriades, Ioannis Papaefstathiou, Apostolos Dollas*. A FPGA BASED COPROCESSOR FOR GENE FINDING USING INTERPOLATED MARKOV MODEL (IMM), 2007.

[15]H. Schmit and D. E. Thomas, Hidden markov modeling and fuzzy controllers in FPGAs, in FPGAs for Custom Computing Machines, 1995. Proceedings. IEEE Symposium on, pp. 214–221, 1995.

[16]Maddimsetty, R.P., Buhler, J., Chamberlain, R., Franklin, M., Harris, B.: Accelerator design for protein sequence HMM search, Proc. 20th ACM International Conference on Supercomputing (ICS06) 288-296, 2006.

[17]A. Darte, R. Schreiber, B. Ramakrishna Rau, and F. Vivien. Constructing and exploiting linear schedules with prescribed parallelism. ACM Trans. Des. Autom. Electron. Syst., 7(1):159–172, 2002.

[18]Ben Wun, Jeremy Buhler, and Patrick Crowley. Ex-ploiting Coarse-Grained Parallelism to Accelerate Pro-tein Motif Finding with a Network Processor. In PACT'05: Proceedings of the 14th International Conference on Parallel Architectures and Compilation Techniques, 2005.

[19]Di Blas, A. et al: The UCSC Kestrel Parallel Processor, IEEE Transactions on Parallel and Distributed Systems 16 (1) 80-92, 2005.