ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ ΚΑΙ ΥΛΙΚΟΥ

# Υλοποίηση σε αναδιατασσόμενη λογική του αλγορίθμου βιοπληροφορικής **LISSOM**

*Συντάκτης:*
Νικόλαος Κοντός

*Επιβλέπων:*
Αν. Καθηγητής Ιωάννης Παπαευσταθίου

*Εξεταστική επιτροπή:*

Αν. Καθηγητής: Ιωάννης Παπαευσταθίου
Καθηγητής: Απόστολος Δόλλας
Επ. Καθηγητής: Δάφνη Μανουσάκη

Ιούνιος ,2017

# Περίληψη

Η ανάπτυξη βιολογικών νευρωνικών μοντέλων αποτελεί ενδιαφέρον των ερευνητών εδώ και πολλά χρόνια. Στόχος είναι η καλύτερη κατανόηση του εγκεφάλου και των λειτουργιών του ώστε να μπορέσει να προσομοιωθεί με αρκετή ακρίβεια. Για αυτό το λόγο αρκετά βιολογικά νευρωνικά μοντέλα έχουν αναπτυχθεί τα οποία καταφέρνουν να προσομοιώσουν σε μεγάλο βαθμό τον τρόπο επεξεργασίας και διάδοσης της πληροφορίας στα νευρωνικά δίκτυα.

Η συγκεκριμένη διπλωματική εργασία στοχεύει στην μελέτη του βιο-αλγορίθμου LISSOM. Το συγκεκριμένο μοντέλο βασίζεται στη +λειτουργία των αυτο-οργανωτικών χαρτών για να προσομοιώσει τις λειτουργίες των νευρώνων και συγκεκριμένα την περιοχή V1 του οπτικού φλοιού. Λόγο του μεγάλου όγκου δεδομένων που χαρακτηρίζει το συγκεκριμένο μοντέλο αποφασίσαμε να χρησιμοποιήσουμε τον υβριδικό υπερ-υπολογιστή της Convey ο οποίος βασίζεται σε αναδιατασσόμενη λογική. Με αυτό τον τρόπο ήταν δυνατό να αποθηκευτούν τα δεδομένα μας στην εξωτερική μνήμη που μας παρέχει το σύστημα με υψηλό εύρος ζώνης των ελεγκτών της. Ταυτόχρονα εποφελούμαστε και από την αναδιατασσόμενη λογική για την ταχύτερη επεξεργασία των δεδομένων μας.

Συγκεκριμένα η σχεδίαση μας μπορεί να προσομοιώσει νευρωνικά δίκτυα μεγέθους μέχρι $192 \times 192$ νευρώνων. Το δίκτυο αποτελείται από δυο επίπεδα, την επιφάνεια του αμφιβληστροειδή, που λειτουργεί σαν είσοδος, και την περιοχή V1 και στηρίζεται σε δυο βασικές υποθέσεις: πως οι πλευρικές συνδέσεις είναι κυρίως ανασταλτικές σε μεγάλες αποστάσεις και διεγερτικές στις κοντινές και πως οι πλευρικές και οι εισάγων συνδέσεις προσαρμόζονται από τον ίδιο μηχανισμό Hebbian.

Τέλος τα αποτελέσματα στις πειραματικές μετρήσεις που έγιναν σε μία Virtex-6 LX760 έδειξαν πως η σχεδίαση μας μπορεί να τρέξει 2 φορές γρηγορότερα από μια συμβατική GPU προσομοιώνοντας μάλιστα μεγαλύτερου μεγέθους νευρωνικά δίκτυα και 3 φορές γρηγορότερα από την υλοποίηση στο Software. Όμως δεν καταφέρνει να ξεπεράσει τις επιδόσεις μιας πολύ καλής GPU όπως η GTX980 που χρησιμοποιήθηκε για τις συγκεκριμένες μετρήσεις.

# Ευχαριστίες

Πρωτίστως θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου Ιωάννη Παπαευσταθίου για την πρακτική καθοδήγηση του καθ' όλη τη διάρκεια αυτής της διπλωματικής εργασίας και την εμπιστοσύνη που έδειξε στο πρόσωπό μου. Επίσης θα ήθελα ακόμη να τον ευχαριστήσω γιατί μέσω της διδασκαλίας του μου έδωσε το κίνητρο να ασχοληθώ με τον συγκεκριμένο τομέα.

Το επόμενο πρόσωπο που θα ήθελα να ευχαριστήσω είναι ο Δρ. Γρηγόρης Χρυσός αφού ήταν πάντοτε διαθέσιμος όταν τον χρειαζόμουν να με συμβουλέψει και να με καθοδηγήσει από την αρχή μέχρι και το τέλος αυτής της εργασίας. Ακόμη θέλω να ευχαριστήσω τους δυο μεταπτυχιακούς φοιτητές, τον Κουσανάκη Μανώλη και Χρήστο Ρουσόπουλο για τον χρόνο τους και την βοήθεια τους πάνω στη διασύνδεση της αρχιτεκτονικής μου με το σύστημα της Convey και τον έλεγχο λειτουργίας του συνολικού συστήματος.

Ένα μεγάλο ευχαριστώ αξίζουν και στους φίλους που έκανα τα χρόνια της παραμονής μου στα Χανιά, και όχι μόνο, για την στήριξη τους όταν τους είχα ανάγκη και τις ευχάριστες στιγμές που περάσαμε όλα αυτά τα χρόνια. Επίσης θα ήθελα να ευχαριστήσω την Εύα για την υπομονή,κατανόηση και την στήριξη της όλα αυτά τα χρόνια.

Το μεγαλύτερο ευχαριστώ όμως αξίζει στην αδερφή μου Ελένη και τους γονείς μου που με υπομονή με στηρίζουν στις επιλογές μου και βρίσκονται πάντα δίπλα μου .

# Περιεχόμενα

**1 Εισαγωγή** **1**
1.1 Αντικείμενο Διπλωματικής . . . . . 1
1.2 Συνεισφορά . . . . . 2
1.3 Δομή Διπλωματικής Εργασίας . . . . . 2

**2 Σχετική Έρευνα** **3**
2.1 Αυτο-οργανωτικοί Χάρτες . . . . . 3
2.2 Λειτουργία των αυτο-οργανωτικών χαρτών . . . . . 4
2.3 Παρόμοιες δουλειές . . . . . 7
2.4 Βασικά χαρακτηριστικά του υπερ-υπολογιστή της Convey . . . . . 13
2.4.1 Γενικά . . . . . 13
2.4.2 Coprocessor Architecture . . . . . 13
2.4.3 Ανάπτυξη ενός Custom Personality . . . . . 14
2.4.4 Διεπαφή ελεγκτών Μνήμης . . . . . 14
2.4.5 Μνήμη Συστήματος . . . . . 15

**3 Μελέτη του αλγορίθμου LISSOM** **17**
3.1 Αλγορίθμος LISSOM . . . . . 17
3.2 Profiling . . . . . 19
3.3 Συναρτήσεις ενδιαφέροντος . . . . . 20
3.3.1 Ανάλυση της συνάρτησης Step . . . . . 20
3.3.2 Ανάλυση της συνάρτησης AdjustWeights . . . . . 21
3.3.3 Ανάλυση της συνάρτησης FirstStep . . . . . 22

**4 Σχεδίαση και Υλοποίηση Αρχιτεκτονικής** **25**
4.1 Σχεδίαση και υλοποίηση της αρχικής Αρχιτεκτονικής . . . . . 25
4.2 Τελική σχεδίαση και υλοποίηση της Αρχιτεκτονικής . . . . . 31

**5 Αποτελέσματα** **41**
5.1 Εγκατάσταση Αρχιτεκτονικής και Κατανάλωση πόρων . . . . . 41
5.2 Πειραματικά Αποτελέσματα . . . . . 42
5.3 Σύγκριση Απόδοσης . . . . . 45

**6 Συμπεράσματα και Μελλοντική Εργασία** **49**
6.1 Ανασκόπηση και Συμπεράσματα . . . . . 49
6.2 Μελλοντική Εργασία . . . . . 50

# Κατάλογος Σχημάτων

2.1 Γενική αρχιτεκτονική του μοντέλου στον πρωτογενή οπτικό φλοιό με χρήση αυτο-οργανωτικών χαρτών . . . . . 5
2.2 Εκμάθηση ενός αυτο-οργανωτικού χάρτη με Gaussian μοτίβα δραστηριότητας. . . . . 6
2.3 Αυτο-οργάνωση του πίνακα των βαρών . . . . . 7
2.4 Διάγραμμα ροής του αλγορίθμου LISSOM για την υλοποίηση σε σύστημα με παραλληλία . . . . . 9
2.5 MicroBlaze Overview . . . . . 10
2.6 Αρχιτεκτονική μιας σύναψης και ενός δικτύου νευρώνων της δημοσίευσης [23 . . . . . 11
2.7 Διάγραμμα του πλήρους συστήματος πολλαπλών FPGA. . . . . 12
2.8 Διάγραμμα του Συνεπεξεργαστή του Convey. . . . . 13
2.9 Συνδεσιμότητα μεταξύ AE-to-MC στο συνεπεξεργαστή. . . . . 14
2.10 Συνδεσιμότητα της διεπαφής του MC με τους ελεγκτές μνήμης στο συνεπεξεργαστή. . . . . 15
2.11 Ιεραρχία Μνήμης. . . . . 16

3.1 Εκτέλεση αλγορίθμου . . . . . 18
3.2 Χρόνος εκτέλεσης συναρτήσεων . . . . . 19

4.1 Top module Αρχικής Αρχιτεκτονικής . . . . . 26
4.2 Αρχιτεκτονική συνάρτησης Step . . . . . 28
4.3 Αρχιτεκτονική προβολής. . . . . 29
4.4 Αρχιτεκτονική σιγμοειδής συνάρτησης. . . . . 30
4.5 Αρχιτεκτονική αθροίσματος. . . . . 31
4.6 Σχεδίαση του Control και για τα 8 cores. . . . . 33
4.7 Στάδια αντιγραφής. . . . . 35
4.8 Σχεδίαση του Top module των 8 Cores. . . . . 37
4.9 Σχεδίαση αρχιτεκτονικής της μονάδας core. . . . . 38
4.10 Αρχιτεκτονική της συνάρτησης AdjustWeights . . . . . 40

5.1 Δίκτυο νευρώνων 64x64 . . . . . 43
5.2 Δίκτυο νευρώνων 92x92 . . . . . 44
5.3 Δίκτυο νευρώνων 128x128 . . . . . 44
5.4 Κοινό διάγραμμα χρόνου εκτέλεσης του μοντέλου για διαφορετικό μέγεθος δίκτυου και για τα τέσσερα πειραματικά αποτελέσματα. . . . . 47

# Κατάλογος Πινάκων

4.1 Μορφή πινάκων . . . . . 27
4.2 Τιμές του σήματος offset για κάθε ένα από τα οκτώ Core . . . . . 36
4.3 Τιμές του σήματος offset_copy για κάθε ένα από τα οκτώ Core σε κάθε στάδιο της αντιγραφής . . . . . 36
4.4 Κατανομή bits του κάθε στοιχείου στον πίνακα των συνδέσεων. . . . . 38
4.5 Τιμές των σημάτων data mc 0 και data mc 1 για κάθε στάδιο της εκτέλεσης του αλγορίθμου . . . . . 39

5.1 Συνολικοί πόροι σχεδίασης και συχνότητα ρολογιού μετά την εγκατάσταση στο Convey . . . . . 42
5.2 Χαρακτηριστικά δικτύου για διάφορα μεγέθη του . . . . . 43
5.3 Χρόνοι εκτέλεσης της σχεδίασης για διαφορετικά μεγέθη δικτύου στο Convey . . . . . 45
5.4 Χρόνοι εκτέλεσης της σχεδίασης για διαφορετικά μεγέθη δικτύου στην κάρτα γραφικών GT720M . . . . . 45
5.5 Χρόνοι εκτέλεσης της σχεδίασης για διαφορετικά μεγέθη δικτύου στην κάρτα γραφικών GTX980 . . . . . 46
5.6 Χρόνοι εκτέλεσης της σχεδίασης για διαφορετικά μεγέθη δικτύου στο Software . . . . . 46
5.7 Μέγεθος του συνόλου δεδομένων για διαφορετικά μεγέθη του δικτύου . 46

# Κεφάλαιο 1

# Εισαγωγή

Ο οπτικός φλοιός του εγκεφάλου είναι ένα κομμάτι του εγκεφαλικού φλοιού που παίζει έναν πολύ σημαντικό ρόλο στην επεξεργασία της οπτικής πληροφορίας. Η οπτική πληροφορία που έρχεται από το μάτι περνάει από το έξω γονατώδες σώμα (LGN), το οποίο βρίσκεται στο θάλαμο, και αποτελεί την κύρια κεντρική σύνδεση του αμφιβληστροειδή με τον οπτικό φλοιό. Στην συνέχεια φτάνει στον οπτικό φλοιό και συγκεκριμένα η πρώτη περιοχή που δέχεται την πληροφορία από τον θάλαμο είναι η περιοχή του πρωτοταγή οπτικού φλοιού(V1). Οι ιδιότητες απόκρισης των νευρώνων σε αρκετές περιοχές του φλοιού διατάσσονται τοπογραφικά, δηλαδή, οι κοντινοί νευρώνες ανταποκρίνονται σε γειτονικές περιοχές της επιφάνειας του υποδοχέα. Τέτοιοι τοπογραφικοί χάρτες σχηματίζονται από την αυτό-οργάνωση των εισάγων συνδέσεων στον φλοιό, οδηγούμενοι από εξωτερικές εισόδους[1,2]. Αρκετά μοντέλα νευρικών δικτύων έχουν δείξει πως η τοπογραφική τάξη μπορεί να προκύψει από τοπικές συνεταιριστικές και ανταγωνιστικές πλευρικές αλληλεπιδράσεις εντός του φλοιού[3,4]. Αυτά τα μοντέλα βασίζονται σε προκαθορισμένες πλευρικές αλληλεπιδράσεις και επικεντρώνονται στη εξήγηση του τρόπου οργάνωσης των εισάγων συναπτικών βαρών.

Ένας αριθμός από νευροβιολογικά πειράματα δείχνουν ότι οι πλευρικές συνδέσεις αυτο-οργανώνονται όπως οι εισάγων συνδέσεις: (1) Η πλευρική συνδεσιμότητα δεν είναι ομοιόμορφη ή γενετικά προκαθορισμένη, αλλά σχηματίζεται κατά την πρώιμη ανάπτυξη με βάση την εξωτερική εισροή πληροφοριών [5]. (2) Οι πλευρικές συνδέσεις συνδέουν κυρίως περιοχές με παρόμοιες ιδιότητες απόκρισης, όπως στήλες με τον ίδιο προσανατολισμό ή προτίμησης ματιών [6]. Οι πλευρικές συνδέσεις είναι πολύ περισσότερες από τις εισάγων και πιστεύεται ότι έχουν σημαντική επίδραση στην φλοιώδη δραστηριότητα [7]. Για να ληφθεί πλήρως η αυτο-οργάνωση των νευρώνων στον φλοιό, ένα μοντέλο πρέπει να αποδείξει ότι τόσο οι εισάγων όσο και οι πλευρικές συνδέσεις μπορούν να οργανωθούν ταυτόχρονα, από την ίδια είσοδο, με έναν αμοιβαίο υποστηρικτικό τρόπο.

## 1.1 Αντικείμενο Διπλωματικής

Η διπλωματική εργασία που παρουσιάζεται, αφορά τη μελέτη και υλοποίηση του βιοαλγορίθμου LISSOM με χρήση αναδιατασσόμενης λογικής (FPGA). Ο συγκεκριμένος αλγόριθμος προσπαθεί να προσομοιώσει την λειτουργία του οπτικού φλοιού και πιο συγκεκριμένα του πρωτοταγή οπτικού φλοιού (περιοχή V1). Σκοπός του είναι, η κατασκευή ρεαλιστικών μοντέλων για την καλύτερη κατανόηση της συμπεριφοράς των συνδέσεων μεταξύ των νευρώνων στην περιοχή του πρωτοταγή οπτικού φλοιού. Λόγω του μεγάλου αριθμού συνδέσεων μεταξύ των νευρώνων το πρόβλημα είναι απαιτητικό όχι τόσο υπολογιστικά αλλά κυρίως στο κομμάτι της μνήμης που απαιτείται για την αποθήκευση όλων των συνδέσεων. Γι' αυτό τον λόγο χρησιμοποιήσαμε έναν υβριδικό υπερ-υπολογιστή

βασισμένο σε FPGA ώστε να εκμεταλλευτούμε και τα πλεονεκτήματα της αναδιατασσόμενης λογικής όπως επίσης και την υψηλού εύρους ζώνης εξωτερική μνήμη που μας δίνεται από το υβριδικό σύστημα.

## 1.2 Συνεισφορά

Το εργαστήριο Μικροεπεξεργαστών και Υλικού του Πολυτεχνείου Κρήτης έχει ασχοληθεί επιτυχώς με αρκετούς τομείς τις βιο-πληροφορικής. Μέσω της συγκεκριμένης διπλωματικής εργασίας παρουσιάζεται η μελέτη και υλοποίηση ενός ακόμη αλγορίθμου με την χρήση των FPGAs, ο αλγόριθμος LISSOM.

Η εργασία επικεντρώνεται στην δημιουργία ενός προσομοιωτή βιολογικού δικτύου νευρώνων βασισμένο στον αλγόριθμο LISSOM. Η βασική συνεισφορά της εργασίας είναι ότι η υλοποίηση του προσομοιωτή έγινε σε ένα υβριδικό σύστημα βασισμένο σε αναδιατασσόμενη λογική με επιπλέον μια μεγάλου μεγέθους εξωτερική μνήμη η οποία είναι προσβάσιμη και από τα FPGAs όπως επίσης και τον επεξεργαστή. Συγκεκριμένα η λειτουργία του συν-επεξεργαστή του συστήματος είναι βοηθητική στην συγκεκριμένη σχεδίαση και παρέχει τα βασικά δεδομένα για την σωστή εκτέλεση του αλγορίθμου όπως επίσης διευκολύνει την παραμετροποίηση του μοντέλου για κάθε προσομοίωση. Με αυτόν τον τρόπο μπορούμε να μελετήσουμε την απόδοση του συγκεκριμένου αλγορίθμου σε μια ακόμη πλατφόρμα και να συγκρίνουμε τις επιδόσεις της συγκριτικά με τους χρόνους σε αντίστοιχες υλοποιήσεις όπως αυτή σε κάρτες γραφικών. Επίσης λόγο της μεγάλης κοινής μνήμης που μας παρέχει το σύστημα του Convey μπορούμε να πετύχουμε προσομοιώσεις δικτύων για αρκετά μεγάλο αριθμό νευρώνων.

## 1.3 Δομή Διπλωματικής Εργασίας

Η δομή που ακολουθείται στη διπλωματική εργασία είναι η παρακάτω:

Στο κεφάλαιο 2 γίνεται μια εισαγωγή στους αυτο-οργανωτικούς χάρτες και στη λειτουργία τους. Αναφέρονται δημοσιεύσεις για παρόμοιες υλοποιήσεις σε FPGA και τέλος γίνεται μια αναφορά στα βασικά χαρακτηριστικά του υβριδικού υπερ-υπολογιστή της Convey.

Το κεφάλαιο 3 αναφέρεται στην μελέτη του αλγορίθμου LISSOM. Πρώτο στάδιο είναι η διαδικασία του Profiling για την εύρεση του πιο υπολογιστικά βαρύ τμήματος. Στην συνέχεια γίνεται η ανάλυση των συναρτήσεων που μας ενδιαφέρουν και θα ασχοληθούμε στην συνέχεια.

Το κεφάλαιο 4 ασχολείται με τη σχεδίαση και υλοποίηση της Αρχιτεκτονικής που υλοποιήθηκε στην συγκεκριμένη εργασία. Αρχικά παρουσιάζεται η πρώτη προσέγγιση του προβλήματος που υλοποιούσε μόνο την συνάρτηση Step και στη συνέχεια αναλύεται η λειτουργία της τελικής σχεδίασης όπως υλοποιήθηκε με όλες τις συναρτήσεις και της 8 διαφορετικές μονάδες επεξεργασίας.

Στο κεφάλαιο 5 παρουσιάζονται τα πειραματικά αποτελέσματα όπως προέκυψαν από την εκτέλεση της σχεδίασης στην υβριδική πλατφόρμα του Convey και συγκρίνεται η απόδοση με αντίστοιχες εκτελέσεις σε 2 κάρτες γραφικών.

Στο κεφάλαιο 6 πραγματοποιείται μια σύνοψη της διπλωματικής εργασίας. Εξάγονται συμπεράσματα και προτείνονται διάφορες μελλοντικές βελτιώσεις που μπορούν να γίνουν πάνω στην σχεδίαση.

# Κεφάλαιο 2

# Σχετική Έρευνα

Σε αυτό το κεφάλαιο αναλύονται οι αυτο-οργανωτικοί χάρτες και ο τρόπος με τον οποίο λειτουργούν , για την καλύτερη κατανόηση εννοιών που θα αναφερθούν στα επόμενα κεφάλαια. Επίσης αναφέρονται παρόμοιες δουλειές σε software και hardware υπολοιήσεις βιολογικών μοντέλων. Τέλος, γίνεται μια αναφορά σχετικά με τα βασικά χαρακτηριστικά του υβριδικού υπέρ-υπολογιστή Convey πάνω στον οποίο υλοποιήθηκε η σχεδίαση.

## 2.1 Αυτο-οργανωτικοί Χάρτες

Όπως αναφέρθηκε και στην εισαγωγή οι νευρώνες στον οπτικό φλοιό δε δρουν μεμονωμένα αλλά κάθε νευρώνας επηρεάζεται έντονα από πλευρικές συνδέσεις οι οποίες διεγείρουν θετικά ή αρνητικά τον νευρώνα. Αν και ο κάθε νευρώνας προσαρμόζει τις δικές του συνδέσεις, οι δραστηριότητες των άλλων νευρώνων επηρεάζουν την μάθηση του. Επομένως για να κατανοήσουμε την ανάπτυξη των νευρώνων πρέπει να λάβουμε υπόψιν μας τις αλληλεπιδράσεις σε όλο το δίκτυο του φλοιού.

Η ιδέα αυτή διαμορφώνεται υπολογιστικά σε αυτο-οργανωτικούς χάρτες (SOM). Ο ανταγωνισμός και η συνεργασία εισάγονται μεταξύ των νευρώνων, έτσι ώστε μόνο μια ή λίγες μονάδες στο δίκτυο να ανταποκρίνονται σε κάθε πρότυπο εισόδου. Εάν προσαρμοστούν μόνο αυτοί οι νευρώνες, κάθε νευρώνας θα μάθει να ανταποκρίνεται καλύτερα σε παρόμοια συμπλέγματα εισόδων. Διαφορετικοί νευρώνες θα ανταποκρίνονται σε διαφορετικά ερεθίσματα εισόδου και το δίκτυο θα μάθει μια απεικόνιση εισόδων που μοιάζει με χάρτη. Αυτοί οι αυτο-οργανωτικοί χάρτες αποτελούν τη συνηθέστερη και καταλληλότερη υπολογιστική δομή για την κατανόηση των υπολογισμών στους χάρτες του οπτικού φλοιού.

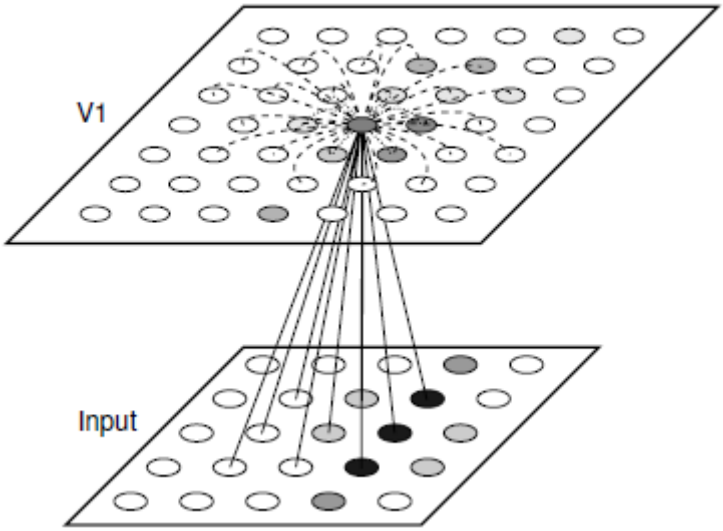
Οι αυτο-οργανωτικοί χάρτες είναι μια γενικότερη κατηγορία μαθησιακών μοντέλων, μερικά από τα οποία είναι αυστηρά κατευθυνόμενα προς την κατανόηση βιολογικών χαρτών ενώ άλλα είναι πιο γενικά και χρησιμοποιούνται κυρίως σε διάφορες εφαρμογές μηχανικής. Το πρώτο μοντέλο αυτο-οργανωτικού χάρτη δημιουργήθηκε από τον von der Malsburg το 1973 και προσομοιώθηκε σε ένα UNIVAC 1Mhz. Για την προσομοίωση χρησιμοποίησε ένα μικρό δίκτυο δύο διαστάσεων από νευρώνες για να μοντελοποιήσει τον φλοιό, βασισμένος στην υπόθεση οτι τα κύτταρα σε μια κάθετη στήλη έχουν τις ίδιες ιδιότητες απόκρισης και μπορούν να αντιμετωπισθούν σαν μια υπολογιστική μονάδα. Κάθε μονάδα έχει σταθερές διεγερτικές πλευρικές συνδέσεις με τους γείτονες της και σταθερές ανασταλτικές πλευρικές συνδέσεις με μονάδες σε μεγαλύτερη απόσταση. Κάθε φορά που παρουσιάζεται μια είσοδος, η πλευρική διεγερτική και ανασταλτική δραστηριότητα επικεντρώνεται στις περιοχές του δικτύου που ανταποκρίνονται καλύτερα. Τα εισάγων βάρη των ενεργών μονάδων τροποποιούνται στην συνέχεια σύμφωνα με τον κανόνα Hebbian, κανονικοποιημένα έτσι ώστε το συνολικό βάρος τις κάθε μονάδας να είναι σταθερό.

Όταν εκπαιδεύονται σε απλά δυαδικά σχέδια που αποτελούνται από προσανατολισμένες μπάρες οι μονάδες έμαθαν να ανταποκρίνονται σε συγκεκριμένους προσανατολισμούς. Επιπλέον οι γειτονικές μονάδες ανταποκρίνονται σε παρόμοιους προσανατολισμούς ενώ οι αμέσως επόμενες μονάδες από τις γειτονικές ανταποκρίνονται σε σχεδόν κάθετους προσανατολισμούς, έτσι ώστε το προφίλ απόκρισης στο δίκτυο να σχηματίσει έναν χάρτη προσανατολισμού παρόμοιο με αυτόν που παρατηρείται στον οπτικό φλοιό.

Δεκάδες παρόμοια αυτο-οργανωτικά μοντέλα έχουν προταθεί για διαφορετικές πτυχές της φλοιικής αυτο-οργάνωσης όπως η τοπογραφία στον φλοιό, η προτίμηση προσανατολισμού και η οπτική κυριαρχία μεταξύ των δύο ματιών. Ανάμεσά τους δυο είναι αυτά που ξεχωρίζουν κυρίως λόγω της κομψότητας και απλότητας τους[8,9]. Στα περισσότερα μοντέλα αυτο-οργάνωσης που ασχολούνται με τον φλοιό οι πλευρικές αλληλεπιδράσεις μεταξύ των νεύρων έχουν αντικατασταθεί από έναν απλούστερο και λιγότερα υπολογιστικά δαπανηρό μηχανισμό. Αρχικά κάθε είσοδος θεωρείται οτι παράγει μόνο μια ενεργή περιοχή στον φλοιό. Στην συνέχεια αντί να χρησιμοποιήσουμε τις πλευρικές αλληλεπιδράσεις για να βρούμε τις περιοχές με την μεγαλύτερη δραστηριότητα, είναι δυνατόν απλά να ψάξουμε για το μεγαλύτερα ενεργά νεύρο και να προσαρμόσουμε τα εισάγων βάρη σε μία κυκλική περιοχή γύρω του. Με αυτόν τον τρόπο, τα μοντέλα υποθέτουν οτι ο φλοιός είναι στατικός και οι πλευρικές συνδέσεις είναι σταθερές ή μεταβάλλονται με έναν απλό και προκαθορισμένο τρόπο.

## 2.2 Λειτουργία των αυτο-οργανωτικών χαρτών

Σε αυτό το σημείο θα αναφερθεί η λειτουργία των αυτο-οργανωτικών χαρτών που χρησιμοποιούνται σε μοντέλα, που σκοπό έχουν την αναπαράσταση βιολογικών χαρτών του εγκεφάλου. Η αρχιτεκτονική τους αποτελείται από ένα δισδιάστατο πίνακα νευρώνων που αντιπροσωπεύουν την επιφάνεια του φλοιού, η οποία συνδέεται με έναν πίνακα εισόδου που αντιπροσωπεύει μια επιφάνεια υποδοχής, όπως ο αμφιβληστροειδής [Σχήμα 2.1]. Κάθε σύνδεση έχει ένα θετικό συναπτικό βάρος. Αρχικά αυτά τα βάρη είναι τυχαία και συνεπώς κάθε νευρώνας αποκρίνεται τυχαία σε κάθε δραστηριότητα του υποδοχέα. Σε κάθε φάση προσαρμογής τα βάρη αυτά ρυθμίζονται και οι νευρώνες βαθμιαία γίνονται όλο και πιο συγκεκριμένοι, προσαρμόζοντας τον συντονισμό τους με τέτοιο τρόπο ώστε κάθε νευρώνας να μπορεί να διεγερθεί μόνο από ένα μικρό και χωρικά περιορισμένο σύνολο νευρώνων από την επιφάνεια του υποδοχέα. Στην τελική οργανωμένη κατάσταση του δικτύου τα βάρη των γειτονικών νευρώνων διευθετούνται έτσι ώστε η θέση της μέγιστης νευρικής διέγερσης να μεταβάλλεται ομαλά με τη θέση διέγερσης στην επιφάνεια του υποδοχέα. Ο νευρικός φλοιός δρα ως τοπογραφικός χάρτης αντιπροσωπεύοντας τη θέση διέγερσης στην επιφάνεια του υποδοχέα.




Σχήμα 2.1: **Γενική αρχιτεκτονική του μοντέλου στον πρωτογενή οπτικό φλοιό με χρήση αυτο-οργανωτικών χαρτών**.Το μοντέλο συνήθως αποτελείται από δύο επιφάνειες νευρωνικών στοιχείων: την είσοδο(αμφιβληστροειδής στην συγκεκριμένη περίπτωση) και την περιοχή V1. Στο συγκεκριμένο παράδειγμα η είσοδος αποτελείται από ένα δίκτυο 5x5 στο οποίο η ενεργοποίηση παρουσιάζεται σαν μια Gaussian κατανομή γύρω από έναν νευρώνα με κωδικοποίηση από άσπρο σε μαύρο (low - high). Οι νευρώνες στην επιφάνεια της περιοχής V1, η οποία είναι επίσης δύο διαστάσεων 7x7, έχουν εισάγων συνδέσεις από τους νευρώνες στην επιφάνεια της εισόδου (συνεχόμενες γραμμές). Επίσης όπως φαίνεται και στο σχήμα έχουν κοντινές πλευρικές διεγερτικές συνδέσεις με τους γειτονικούς νευρώνες και πλευρικές ανασταλτικές συνδέσεις με πιο μακρινούς νευρώνες (διακεκομμένες γραμμές).

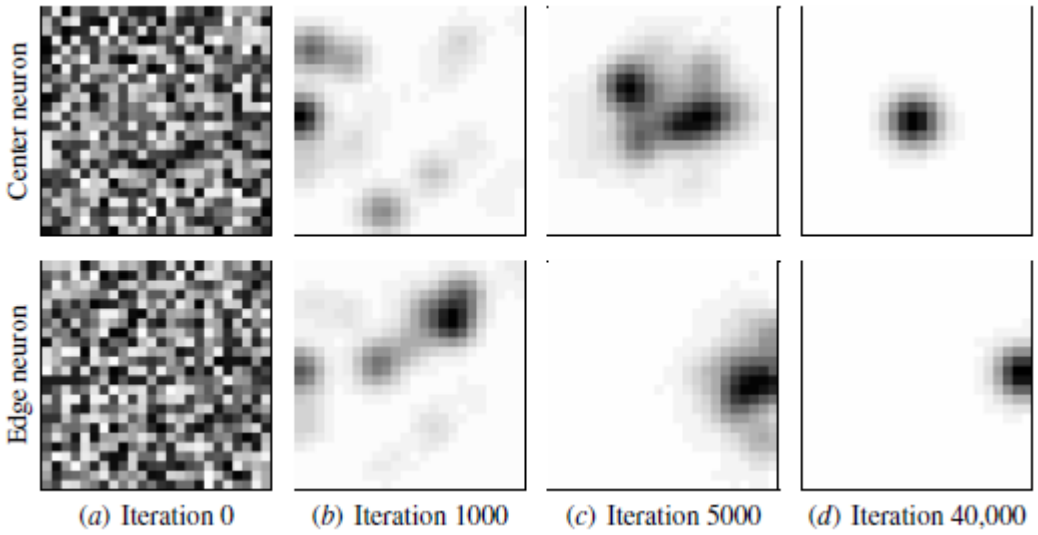
Στην φάση προσαρμογής μια τυχαία είσοδος παρουσιάζεται κάθε φορά στο δίκτυο των νευρώνων του φλοιού. Το δίκτυο ανταποκρίνεται στην κάθε είσοδο κάθε φορά αναπτύσσοντας μια τοπική δραστηριότητα στον φλοιό. Ακόμη τα βάρη του νευρώνα με την μεγαλύτερη δραστηριότητα, όπως επίσης και των γειτονικών του, αλλάζουν σε κάθε είσοδο έτσι ώστε αυτοί οι νευρώνες να παράγουν ακόμη μεγαλύτερες αποκρίσεις σε μία ίδια είσοδο στο μέλλον. Δηλαδή ο χάρτης προσαρμόζεται με δύο τρόπους σε κάθε δεδομένη είσοδο : (1) Τα βάρη αλλάζουν ώστε το δίκτυο να προσεγγίζει καλύτερα την είσοδο , (2) τα γειτονικά βάρη προσαρμόζονται ώστε να γίνουν πιο όμοια. Αυτές οι δύο διαδικασίες προσαρμογής επιβάλουν στο δίκτυο να γίνει ένας οργανωμένος χάρτης της εισόδου. Η διαδικασία αυτή ξεκινά σε αρκετά μεγάλες περιοχές νευρώνων, δηλαδή τα βάρη αλλάζουν μαζί σε μεγάλες περιοχές, και σταδιακά οι περιοχές αυτές στενεύουν όπως επίσης μειώνεται και ο ρυθμός εκμάθησης καθώς το μοντέλο συνεχίζει να εκτελείται. Τελικά η κατανομή των βαρών γίνεται μια προσέγγιση της κατανομής της εισόδου και αναπτύσσεται ένας ομαλός τοπογραφικός χάρτης.

Σχήμα 2.2: **Εκμάθηση ενός αυτο-οργανωτικού χάρτη με Gaussian μοτίβα δραστηριότητας.** Κάθε είσοδος είναι ένα τέτοιο μοτίβο που διεγείρει τους νευρώνες γύρω από μια περιοχή του δικτύου. Τέσσερις διαφορετικές είσοδοι φαίνονται στο σχήμα με την μόνη διαφορά να είναι η τοποθεσία τις διεγερμένης περιοχής κάθε φορά.

Η διαδικασία της αυτο-οργάνωσης σε έναν τοπογραφικό χάρτη μπορεί να περιγραφεί καλύτερα οπτικά όταν η είσοδος είναι δυο διαστάσεων, όπως και ο χάρτης. Μια τέτοια είσοδος αποτελείται από ένα εστιασμένο σημείο με μεγάλη δραστηριότητα σε μια τυχαία περιοχή [Σχήμα 2.2]. Η δραστηριότητα του κάθε νευρώνα στην επιφάνεια της εισόδου περιγράφεται απο την παρακάτω εξίσωση :

$$\chi_k = exp\bigg( - \frac{(x - x_c)^2 + (y - y_c)^2}{\sigma_u^2} \bigg) \tag{2.1}$$

όπου (x,y) περιγράφουν την θέση του νεύρωνα $k, (x_c, y_c)$ το κέντρο της περιοχής με την μεγάλη δραστηριότητα στην είσοδο και $(\sigma_{\mathrm{u}})$ το πλάτος της περιοχής. Καθώς η εκμάθηση του δικτύου γίνεται από αυτές τις εισόδους, το μοντέλο μαθαίνει να αναπαριστά τις δισδιάστατες περιοχές του υποδοχέα. Με άλλα λόγια, αν θεωρήσουμε ότι ο φλοιός είναι η περιοχή V1 του οπτικού φλοιού και σαν είσοδο θεωρήσουμε τον αμφιβληστροειδή, το μοντέλο μαθαίνει μια χαρτογράφηση του αμφιβληστροειδή.




Σχήμα 2.3: **Αυτο-οργάνωση του πίνακα των βαρών.**Οι πίνακες των βαρών για δύο διαφορετικές εισόδους παρουσιάζονται στο παραπάνω σχήμα. Όπως και προηγουμένως οι τιμές των βαρών κωδικοποιούνται σε αποχρώσεις του γκρί από το άσπρο ως το μαύρο(low-high). Αρχικά οι τιμές των βαρών είναι τυχαίες (επανάληψη 0) και μετά από μερικές παρουσιάσεις εισόδων, όπως αυτές στο [σχήμα 2.2], στο δίκτυο τα βάρη αρχίζουν σιγά σιγά να μοιάζουν με τα μοτίβα που δέχονται σαν είσοδο σε διαφορετικές τοποθεσίες του φλοιού(επαναλήψεις 1000,5000,40000).

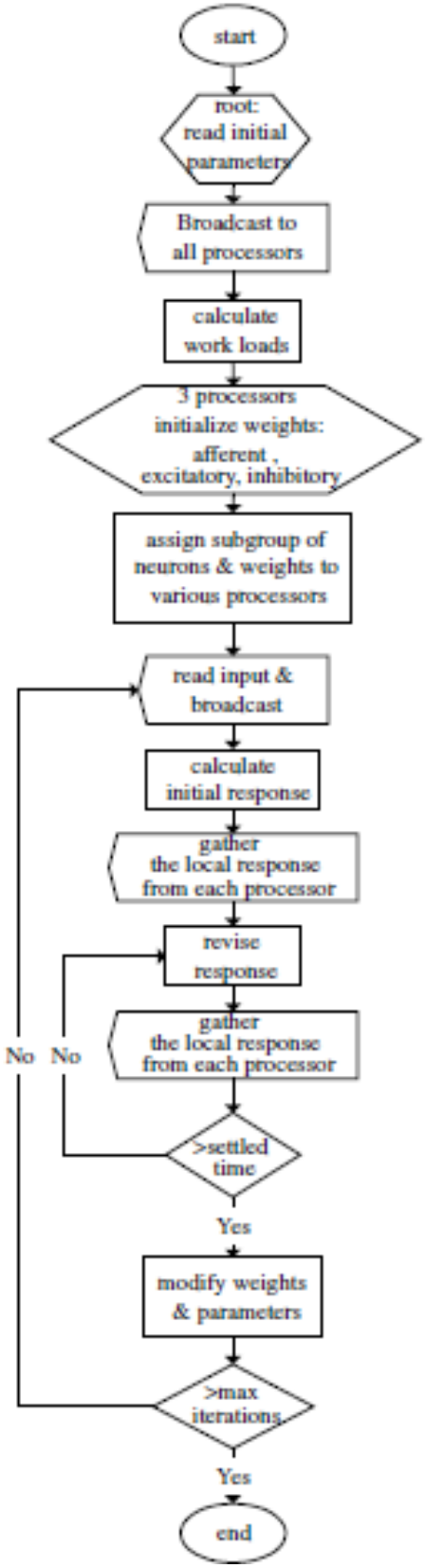
Τα βάρη μεταξύ των νευρώνων έχουν αρχικά τυχαίες τιμές μεταξύ 0-1[Σχήμα 2.3]. Μετά από αρκετές παρουσιάσεις εισόδων τα βάρη των νευρώνων έχουν αλλάξει σε τέτοιο βαθμό ώστε το δίκτυο να αναπαριστά την κάθε είσοδο στην αντίστοιχη τοποθεσία του με τον σωστό τρόπο. Για παράδειγμα η είσοδος με την δραστηριότητα των νευρώνων στο κέντρο του δικτύου δημιουργεί ένα αντίστοιχο Gaussian πρότυπο στο κέντρο του φλοιού, αντίστοιχα μια είσοδος με μεγαλύτερη δραστηριότητα στην άκρη του δικτύου αναπαρίσταται στον φλοιό με την αντίστοιχη δραστηριότητα στην άκρη του δικτύου. Σε αυτά τα παραδείγματα το βασικό χαρακτηριστικό της κάθε εισόδου είναι η τοποθεσία της διέγερσης των νευρώνων και τελικά το δίκτυο μαθαίνει να αναπαριστά τις διάφορες διεγέρσεις του αμφιβληστροειδή. Τέτοιοι χάρτες εισόδων του αμφιβληστροειδή αποτελούνται από δύο διαστάσεις και επειδή αντίστοιχα ο φλοιός είναι και αυτός δύο διαστάσεων τέτοιες αναπαραστάσεις είναι αρκετά εύκολες. Αν και δεν θα ασχοληθούμε στην συγκεκριμένη εργασία απλά αναφέρεται πως σαν είσοδος μπορεί να δοθεί και χάρτης με περισσότερες από δύο διαστάσεις, όπως για παράδειγμα όταν μας αφορούν οι είσοδοι και των δύο ματιών.

## 2.3 Παρόμοιες δουλειές

Εδώ και αρκετά χρόνια ερευνητές προσπαθούν να προσομοιώσουν την λειτουργία των νευρωνικών δικτύων με διάφορα μοντέλα. Η ανάγκη τους για μοντέλα που θα μπορούσαν να λειτουργούν αντίστοιχα με διάφορες λειτουργίες που εκτελεί ο ανθρώπινος εγκέφαλος τους έκανε να εξερευνήσουν περισσότερο τα βιολογικά νευρωνικά δίκτυα σε βαθμό που θα μπορούσαν να χρησιμοποιήσουν την γνώση αυτή για να εξελίξουν ή να αναπτύξουν νέα πιο ρεαλιστικά μοντέλα. Έτσι μοντέλα όπως το Integrate and Fire, Integrate and Fire with Adaptation, Spiking Model από τον Izhikevich, Hodgkin-Huxley, Morris-Lecar, Kohonen's SOM model αναπτύχθηκαν έχοντας διαφορετικές προσεγγίσεις σε

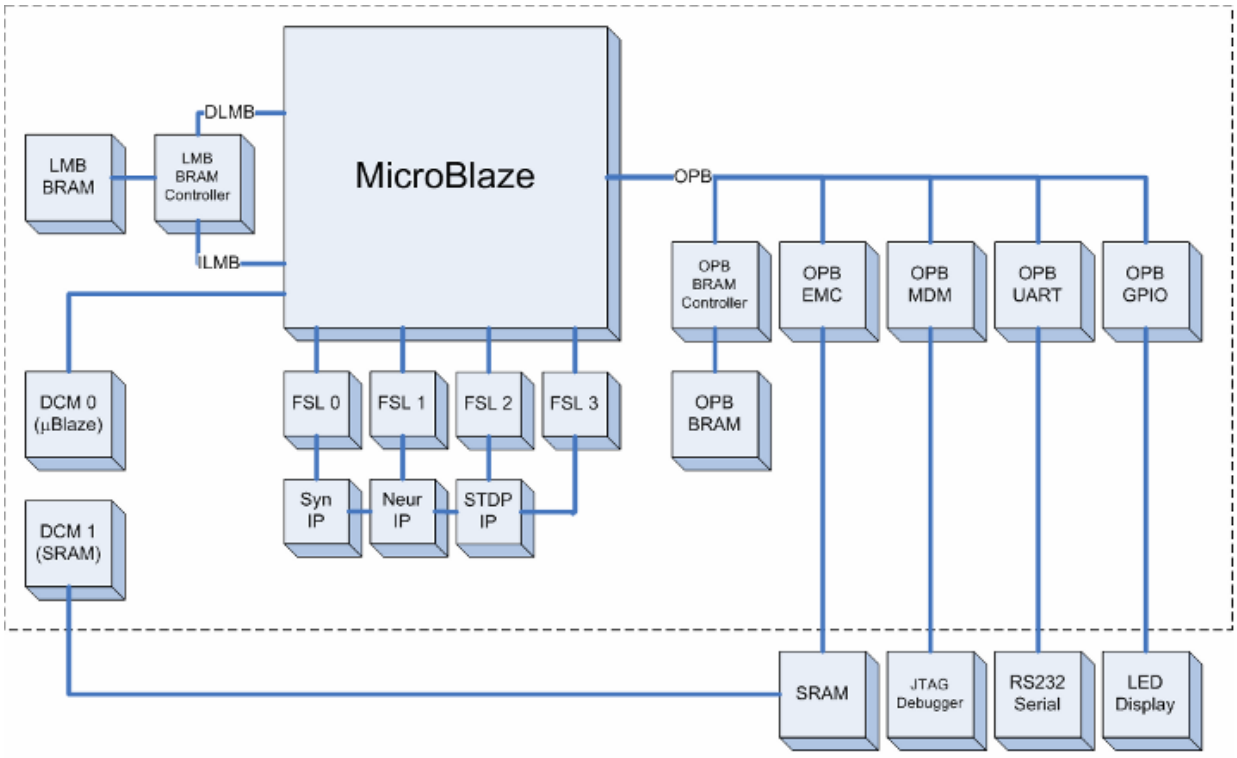
επίπεδο λεπτομέρειας του δικτύου αλλά και σε επίπεδο μεγέθους του νευρωνικού δικτύου που μπορούν να επεξεργαστούν. Παραδείγματος χάριν το μοντέλο του Hodgkin-Huxley [16] είναι ένα από τα πιο λεπτομερή μοντέλα όσον αναφορά τις ιδιότητες του νευρώνα και του δικτύου. Η λεπτομέρεια που παρέχει το μοντέλο όμως το κάνει αρκετά απαιτητικό υπολογιστικά στην υλοποίηση του το οποίο έχει σαν αποτέλεσμα να μπορεί να αντεπεξέλθει σε μόνο ένα μικρό αριθμό νευρώνων ώστε να μπορεί να τρέχει μέσα σε φυσιολογικά πλαίσια χρόνου. Αντίθετα μοντέλα όπως το Integrate and Fire[17] μας δίνουν την δυνατότητα να προσομοιώσουμε πολύ μεγάλο αριθμό νευρώνων στο δίκτυο αφού δεν λαμβάνουν υπόψιν τους τόσο μεγάλη λεπτομέρεια των ιδιοτήτων των νευρώνων. Αρκετές παραλλαγές των μοντέλων που αναφέρθηκαν έχουν δημιουργηθεί για να εξυπηρετήσουν τις ανάγκες για επιπλέον ιδιότητες του νευρώνα ή για μοντελοποιήσεις μεγάλης κλίμακας. Ένα ακόμη σημαντικό μοντέλο είναι το spiking model του Izhikevich [18] το οποίο είναι ικανό να αναπαραστήσει όλα τα μοτίβα fire του νευρώνα και αρκετές λειτουργίες του αλλά ταυτόχρονα προσφέρει την δυνατότητα για αρκετά μεγάλης κλίμακας spiking neural networks. Τέλος ένα διαφορετικό είδος μοντέλων από αυτά που αναφέρθηκαν και πάνω στο οποίο έχει βασιστεί και ο αλγόριθμος LISSOM είναι αυτά που βασίζονται πάνω στους αυτό οργανωτικούς χάρτες όπως το μοντέλο του Kohonen's[8].

Βασισμένοι πάνω σε αυτά τα μοντέλα που αναφέραμε αρκετοί προσομοιωτές έχουν δημιουργηθεί σε Software όπως ο Neuron[19] ή ο SpikerNNS[20] που μας επιτρέπουν να μελετήσουμε ρεαλιστικά βιολογικά μοντέλα. Λόγο του αυξημένου υπολογιστικού φόρτου και χρόνου που καταναλώνουν αυτές οι προσομοιώσεις πολλοί ερευνητές επέλεξαν να χρησιμοποιήσουν το Hardware για γρηγορότερους χρόνους εκτέλεσης συγκριτικά με το Software. Στις περισσότερες υλοποιήσεις σε Hardware το μοντέλο που χρησιμοποιήθηκε είναι του Izhikevich αφού προσφέρει ταυτόχρονα αρκετή βιολογική ακρίβεια στους νευρώνες όπως επίσης και υπολογιστική απόδοση σε μεγαλύτερα μεγέθη νευρωνικών δικτύων.




Σχήμα 2.4: **Διάγραμμα ροής του αλγορίθμου LISSOM για την υλοποίηση σε σύστημα με παραλληλία**

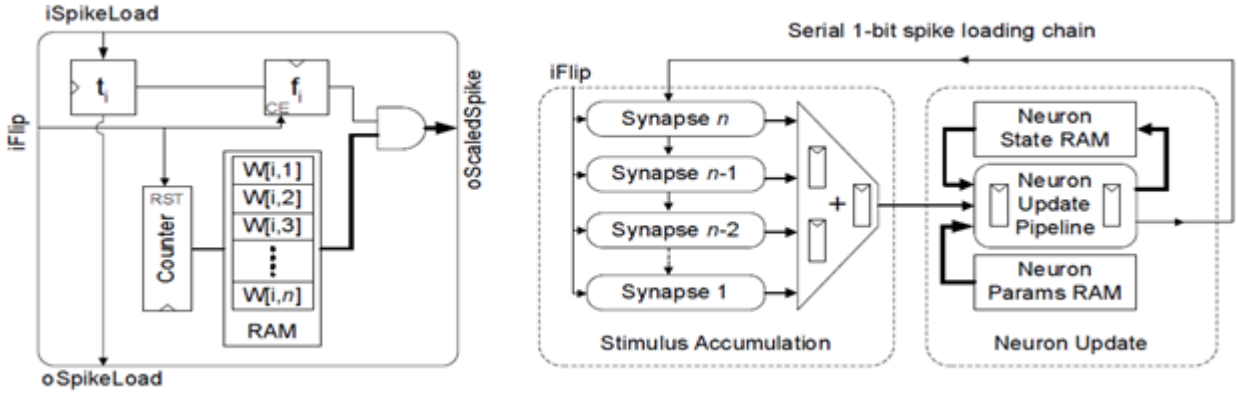
Οι συγγραφείς της [22] δημοσίευσης χρησιμοποιώντας το μοντέλο LISSOM δημιούργησαν μια παράλληλη εκδοχή του συγκριτικά με τις μέχρι τότε σειριακές υλοποιήσεις σε Software. Ο προσομοιώτης χρησιμοποιούσε 8 cores στους οποίους δεν χώριζε απλά το δίκτυο σε 8 κομμάτια αλλά προσπαθούσε να διαμοιράσει το φόρτο εργασίας στον ίδιο βαθμό για τον καθένα. Αυτό συμβαίνει γιατί οι νευρώνες που βρίσκονται στην άκρη του δικτύου έχουν λιγότερες συνδέσεις από αυτούς στο κέντρο τους λόγω της τοπολογίας του δικτύου. Έτσι τα πρώτα όπως και τα τελευταία cores θα κληθούν να επεξεργαστούν τα δεδομένα για περισσότερους νευρώνες σε αριθμό το οποίο όμως θα εκτελεστεί στον ίδιο περίπου χρόνο με τους υπόλοιπους πυρήνες. Στο σχήμα 2.4 φαίνεται το διάγραμμα ροής του μοντέλου που χρησιμοποίησαν στο οποίο περιλαμβάνονται στάδια όπως ο υπολογισμός του φόρτου εργασίας για τον κάθε πυρήνα ώστε να χωριστεί σωστά το δίκτυο όπως αναφέρθηκε. Ακόμη διεργασίες όπως η αρχικοποίηση των βαρών για τις συνδέσεις των νευρώνων, από συγκεκριμένους μόνο πυρήνες, και αντίστοιχα ο συγχρονισμός του όλου συστήματος έχουν ληφθεί υπόψιν και αποτελούν επιπλέον στοιχεία στην εκτέλεση του αλγορίθμου στην σειριακή του μορφή. Στα αποτελέσματα φάνηκε πως για μικρά μεγέθη δικτύου η σχεδίαση αυτή δεν αποδίδει σε αντίθεση με μεγαλύτερα δίκτυα όπου κατάφεραν να μειώσουν τον χρόνο εκτέλεσης μέχρι και 6 φορές συγκριτικά με το σειριακό σύστημα. Γενικά μέσω της συγκεκριμένης δημοσίευσης φάνηκε πως η παραλληλοποίηση του αλγορίθμου LISSOM είναι δυνατή και έχει αποτελέσματα από άποψη απόδοσης.




Σχήμα 2.5: **MicroBlaze Overview**

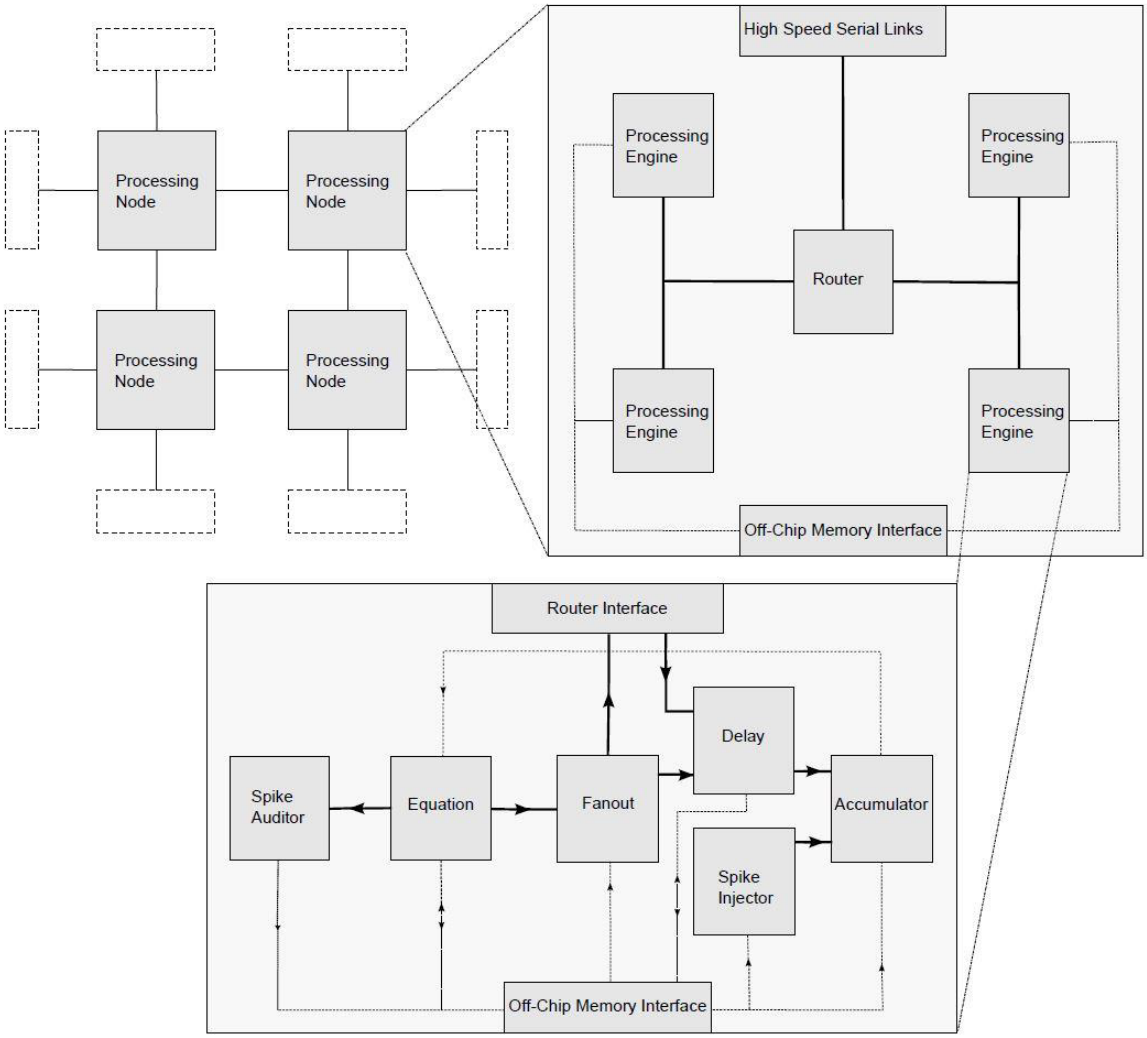
Μια άλλη ενδιαφέρουσα δημοσίευση [14] ασχολείται με το Integrate and Fire μοντέλο το οποίο υλοποιήθηκε σε FPGA. Στόχος τους ήταν να μπορέσουν να προσομοιώσουν μεγάλου μεγέθους νευρωνικά δίκτυα. Για το σύστημα χρησιμοποίησαν τον MicroBlaze επεξεργαστή που διαθέτουν οι συγκεκριμένες FPGA. Όπως φαίνεται και στο σχήμα 2.5 η λογική που εκτελεί τις διεργασίες για τους νευρώνες, συνάψεις και τις STDP συνάψεις έχει υλοποιηθεί στο FPGA και χρησιμοποιώντας την διεπαφή FSL, η οποία είναι

αμφίδρομη, ο επεξεργαστής μπορεί να τις χρησιμοποιήσει. Τα δεδομένα του δικτύου είναι αποθηκευμένα σε εξωτερική μνήμη από το FPGA και συγκεκριμένα στην SRAM η οποία μπορεί να προσπελαστεί από τον δίαυλο OPB και μέσω του ελεγκτή εξωτερικής μνήμης (EMC). Το σύστημα που μόλις περιγράψαμε αποτελεί μόνο απεικόνιση του συνολικού συστήματος. Πιο συγκεκριμένα οι προσομοιώσεις έγιναν πάνω στο σύστημα BenNuey το οποίο παρέχει μια πλατφόρμα αρκετών FPGA. Στα παραδείγματα τους 2 από τις 7 FPGA χρησιμοποιήθηκαν για τα 4 MicroBlaze συστήματα και συγκεκριμένα η XC2V8000 και η XC2V4000. Κατάφεραν να προσομοιώσουν δίκτυο μεγέθους 1400 x 1400 x 1400 και να βελτιώσουν τον χρόνο εκτέλεσης πάνω από 100 φορές συγκριτικά με την εκτέλεση του αλγορίθμου στην Matlab.




Σχήμα 2.6: .
]**Αρχιτεκτονική μιας σύναψης και ενός δικτύου νευρώνων της δημοσίευσης [23]**

Η συγκεκριμένη δημοσίευση [23] αναλύει την υλοποίηση που παρουσιάζεται στο σχήμα 2.6 και βασίζεται στο spiking μοντέλο του Izhikevich. Ένα μοντέλο 1024 νευρώνων με 1024 συνάψεις για τον καθένα δημιουργήθηκε με τους νευρώνες να είναι πλήρως συνδεδεμένοι μεταξύ τους. Διαφορετικά είδη spikes (phasic spiking, tonic spiking και tonic bursting) δημιουργούνται τα οποία παράγονται ανάλογα με το άθροισμα των συνάψεων και την κατάσταση του νευρώνα σε κάθε στιγμή. Τα συναπτικά βάρη του κάθε νευρώνα είναι αποθηκευμένα σε μια Block RAM και σε συνδιασμό με τον πίνακα των νευρώνων που έχουν πυροδοτήσει αιχμή, υπολογίζεται το συνολικό δυναμικό στο σώμα για τον κάθε νευρώνα και ανανεώνεται ο πίνακας με τις αιχμές. Το μοντέλο υλοποιήθηκε σε μια Virtex-5 xc5vlx330 και χρησιμοποιούσε το 20% και 80% σε επίπεδο λογικής και μνήμης αντίστοιχα. Το μοντέλο πέτυχε 16 φορές καλύτερο χρόνο από Software που έτρεχε σε έναν επεξεργαστή στα 3Ghz 2 πυρήνων. Τέλος κατάφεραν να πάρουν μια βελτίωση στον χρόνο εκτέλεσης συγκριτικά με μια κάρτα γραφικών 30 πυρήνων στα 1.2GHz τις τάξεως του 10%.




Σχήμα 2.7: **Διάγραμμα του πλήρους συστήματος πολλαπλών FPGA.**

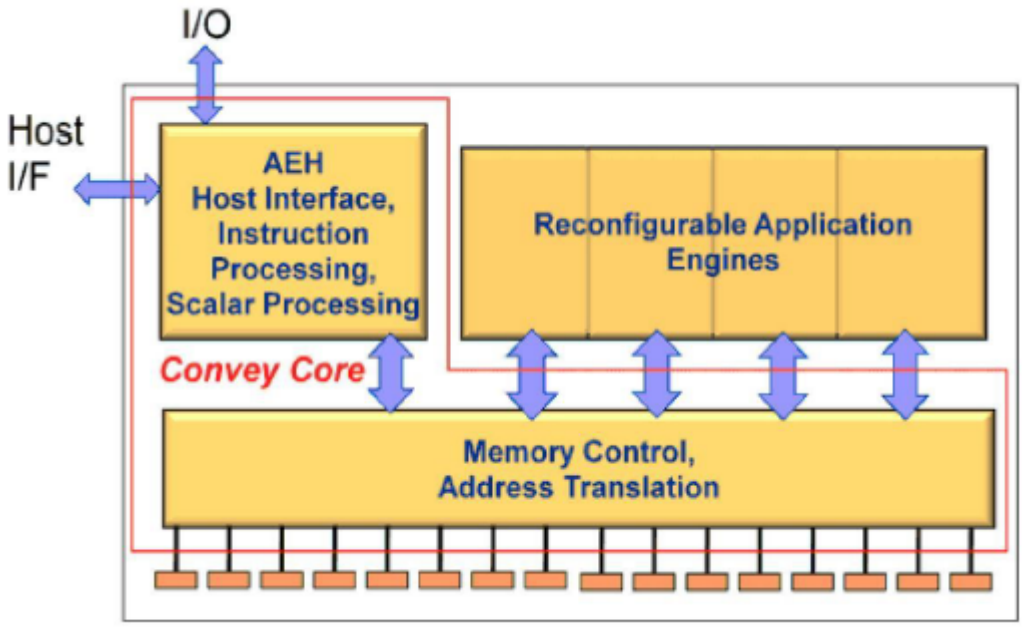
Μια ακόμη αξιόλογη δουλεία έγινε στην δημοσίευση [24] όπου δημιούργησαν ένα Real-time Spiking Neural Network σε ενα Cluster από FPGA. Το σύστημα τους ονομάζεται BlueHive και αποτελείται από 64 FPGA οι οποίες είναι συνδεδεμένες μεταξύ τους μέσω SATA connectors. Ο στόχος του συστήματος είναι η προσομοίωση μοντέλων μεγάλης κλίμακας. Κάθε FPGA έχει 64000 νευρώνες και 1000 συνάψεις για τον κάθε νευρώνα οι οποίοι είναι βασισμένοι στο μοντέλο του Izhikevich. Για τον υπολογισμό των εξισώσεων του μοντέλου διαβάζονται από την εξωτερική μνήμη οι παράμετροι και τα συναπτικά βάρη. Στην συνέχεια για κάθε νευρώνα που πυροδοτεί μια αιχμή ομαδοποιούνται και προστίθενται τα συναπτικά βάρη των νευρώνων που θα ενεργοποιηθούν. Για την σύνδεση των νευρώνων μεταξύ των FPGA υπάρχει ο κατάλληλος μηχανισμός συγχρονισμού. Για την υλοποίηση της σχεδίασης χρησιμοποίησαν Bluespec SystemVerilog και η αρχιτεκτονική αποτυπώθηκε σε FPGA της Altera Stratix IV 230. Η σχεδίαση τους πέτυχε 162 φορές μικρότερο χρόνο εκτέλεσης από λογισμικό σε Software που έτρεξε σε έναν Xeon X5560 2.8GHz 4ων πυρήνων με 48GB RAM.

## 2.4 Βασικά χαρακτηριστικά του υπερ-υπολογιστή της Convey

Σε αυτή την ενότητα θα περιγραφούν κάποια από τα βασικά χαρακτηριστικά του υπερ-υπολογιστή πάνω στον οποίο υλοποιήθηκε η αρχιτεκτονική.

### 2.4.1 Γενικά

Ο συνεπεξεργαστής του Convey είναι βασισμένος σε αναδιατασσόμενη λογική με σκοπό να αυξήσει την απόδοση του χρόνου εκτέλεσης των εφαρμογών, παραπάνω από ότι είναι συνήθως δυνατό από τα συστήματα που είναι βασισμένα μόνο στο πρότυπο ×86. Λόγο της προγραμματιζόμενης φύσης του συστήματος , το hardware επιτρέπει την αναπροσαρμογή της αρχιτεκτονικής ανάλογα με τις απαιτήσεις της κάθε εφαρμογής. Αυτά τα επαναπροσδιοριζόμενα σύνολα εντολών ονομάζονται personalities. Το σύστημα προσφέρει κάποια personalities, όπως single-precision, double precision vector personalities, financial analytics personality και Smith-Waterman personality, τα οποία μπορούν να χρησιμοποιηθούν για να επιταχύνουν ορισμένες εφαρμογές. Παρά τα personalities που προσφέρονται από το σύστημα κάποιες εφαρμογές απαιτούν εξειδικευμένη λειτουργικότητα και για αυτό τον λόγο η Convey σχεδίασε ενα framework για να μπορούν να δημιουργηθούν Custom Application Engine Personalities, συμπεριλαμβανομένων και επεκτάσεων συνόλων εντολών που επιτρέπουν εκτέλεση custom εντολών.




Σχήμα 2.8: **Διάγραμμα του Συνεπεξεργαστή του Convey.**

### 2.4.2 Coprocessor Architecture

Ο συνεπεξεργαστής του Convey αποτελείται από τρία βασικά μέρη: The Application Engine Hub (AEH), the Memory Controllers (MCs) και the Application Engines(AEs). Οι custom εντολές που αναπτύχθηκαν για τον συνεπεξεργαστή εφαρμόζονται στις Application Engines FPGAs και οι AEs είναι οι μόνες FPGAs οι οποίες επαναπροσδιορίζονται για διαφορετικά personalities. Τα AEs περιέχουν τέσσερις βασικές διεπαφές

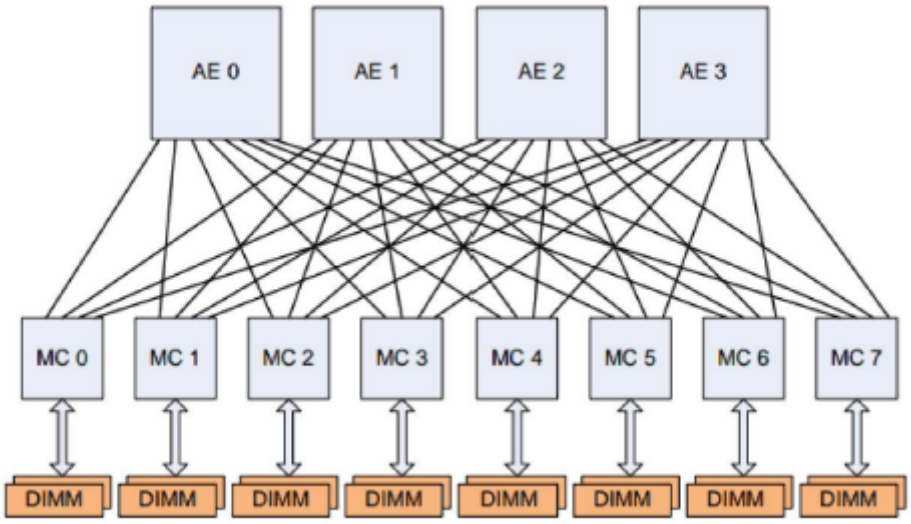
στο υπόλοιπο σύστημα: την διεπαφή αποστολής, την διεπαφή του ελεγκτή μνήμης, την διεπαφή CSR/debug και την διεπαφή AE-to-AE.

### 2.4.3 Ανάπτυξη ενός Custom Personality

Το πρώτο βήμα για την ανάπτυξη ενός νέου personality είναι η ανάλυση της εφαρμογής σε τέτοιο βαθμό ώστε να γνωρίζουμε πως η συγκεκριμένη εφαρμογή εκτελείται στο hardware, την δομή των δεδομένων που θα περιέχονται, πόσο παράλληλη μπορεί να γίνει η εκτέλεση της εφαρμογής και τι περιορίζει την απόδοση της. Στην συνέχεια πρέπει να οριστούν οι Custom εντολές που θέλουμε να τρέχουν στο σύστημα ώστε να βελτιωθεί η απόδοση για την συγκεκριμένη εφαρμογή που ενδιαφερόμαστε. Το επόμενο βήμα είναι η ανάπτυξη του Software γραμμένο σε γλώσσα C, την οποία υποστηρίζει το σύστημα, και βοηθά να προσομοιωθεί το κομμάτι της εφαρμογής που δεν θα υλοποιηθεί στο hardware. Τέλος η ανάπτυξη της σχεδίασης που θα τρέξει τελικά στις FPGAs του Convey. Επίσης ο υπερ-υπολογιστής διαθέτει εργαλεία για την προσομοίωση του hardware στο περιβάλλον του συστήματος.

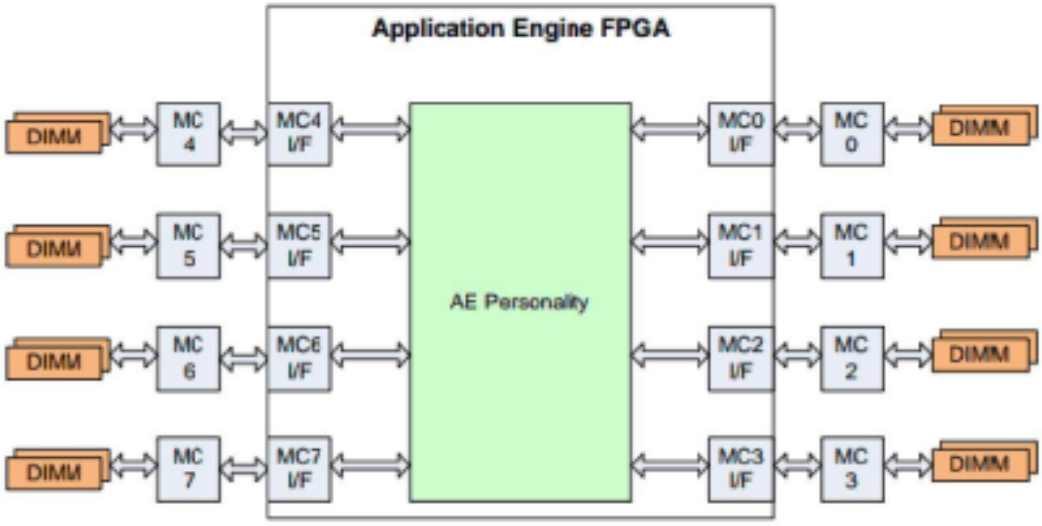
### 2.4.4 Διεπαφή ελεγκτών Μνήμης

Η διεπαφή των Memory Controller (MC) δίνει στις AEs άμεση πρόσβαση στη μνήμη του συνεπεξεργαστή. Κάθε μία από τις 4 AEs είναι συνδεδεμένη με κάθε ένα από τους 8 MCs μέσω μίας DDR διεπαφής με συχνότητα ρολογιού 333MHz. Η διεπαφή του MC μέσα στην AE FPGAs παρέχεται από το Convey. Κάθε μία διεπαφή από τους 8 MC στην AE FPGA είναι άμεσα συνδεδεμένη με ένα ενιαίο Memory Controller και κάθε MC συνδέεται με το 1/8 της μνήμης του συνεπεξεργαστή. Το διάγραμμα παρακάτω δείχνει την συνδεσιμότητα μεταξύ AE-to-MC στο συνεπεξεργαστή.




Σχήμα 2.9: **Συνδεσιμότητα μεταξύ AE-to-MC στο συνεπεξεργαστή.**

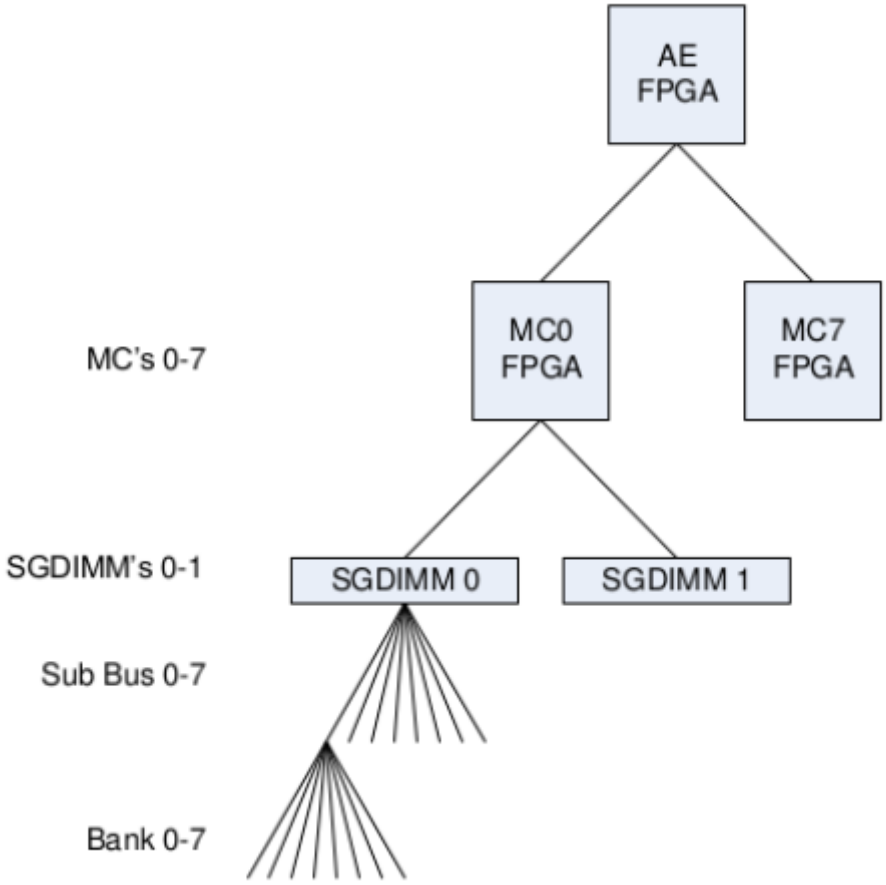
Οι 8 διεπαφές του MC βρίσκονται στην αριστερή και δεξιά πλευρά του AE FPGA όπως φαίνεται και στο σχήμα 2.4.3 παρακάτω. Κάθε ελεγκτής μνήμης συνδέεται με 2 μονάδες DIMM. Η AE personality είναι υπεύθυνη για την αποκωδικοποίηση της διεύθυνσης εικονικής μνήμης, έτσι ώστε μόνο οι αιτήσεις που προορίζονται για ένα συγκεκριμένο MC να στέλνονται σε αυτό το MC.




Σχήμα 2.10: **Συνδεσιμότητα της διεπαφής του MC με τους ελεγκτές μνήμης στο συνεπεξεργαστή.**

### 2.4.5 Μνήμη Συστήματος

Το σύστημα μνήμης του Convey χρησιμοποιεί Scatter/Gather DIMMs τα οποία έχουν 1024 bank μνήμης. Τα banks αυτά είναι μοιρασμένα σε οκτώ ελεγκτές μνήμης. Κάθε ελεγκτής μνήμης έχει 2 64-bit διαύλους και κάθε δίαυλος έχει πρόσβαση σε οκτώ υπό διαύλους (8bit ανά υπό δίαυλο). Τέλος κάθε υπό δίαυλος έχει οκτώ banks. Τα 1024 banks προκύπτουν ως εξής: 8 MCs * 2 DIMMs/MC *8 sub bus/DIMM * 8 bank/sub bus. Στο παρακάτω διάγραμμα φαίνεται η ιεραρχία της μνήμης του συνεπεξεργαστή.




Σχήμα 2.11: **Ιεραρχία Μνήμης.**

# Κεφάλαιο 3

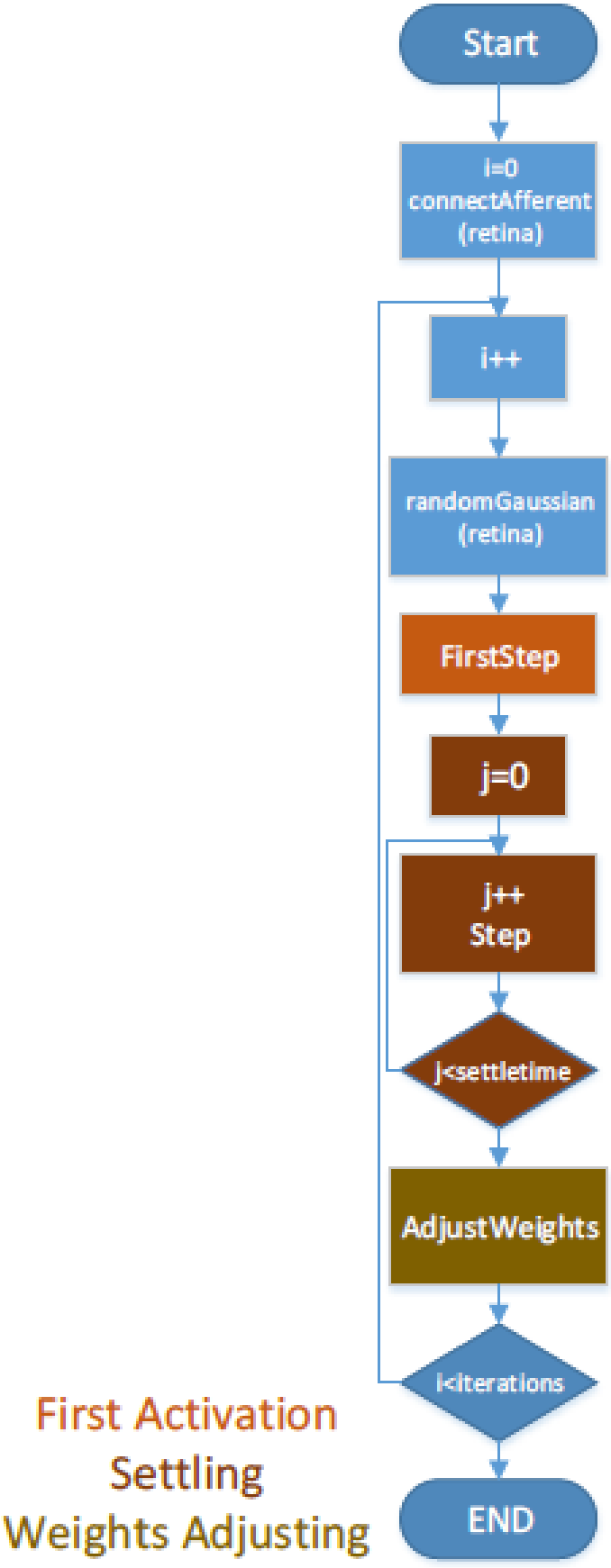
# Μελέτη του αλγορίθμου LISSOM

Αυτή η ενότητα αναφέρεται στη πρώτη προσέγγιση του προβλήματος. Περιγράφεται πιο αναλυτικά ο αλγόριθμος LISSOM και οι βασικές συναρτήσεις οι οποίες χρησιμοποιούνται για την προσομοίωση του οπτικού φλοιού. Επίσης περιγράφεται η διαδικασία του profiling μέσω της οποίας εκτιμήθηκαν τα βαριά υπολογιστικά κομμάτια του αλγορίθμου. Τέλος γίνεται μια πιο εκτενής ανάλυση στις συναρτήσεις που θα μας απασχολήσουν στη συνέχεια.

## 3.1 Αλγορίθμος LISSOM

Όπως έχει αναφερθεί σκοπός του αλγορίθμου LISSOM είναι να προσομοιώσει την λειτουργία του οπτικού φλοιού του εγκεφάλου. Μια συγκεκριμένη ακολουθία από διεργασίες πρέπει να εκτελεστούν ώστε να έχουμε τα αναμενόμενα αποτελέσματα. Αρχικά πρέπει να δημιουργηθεί το επίπεδο της εισόδου. Στην συνέχεια δημιουργείται το επίπεδο της περιοχής V1 το οποίο αποτελείται από τρείς προβολές, εισάγουσα, διεγερτική, ανασταλτική όπως επίσης και τους νευρώνες. Τα δύο αυτά επίπεδα συνδέονται μέσω της συνάρτησης ConnectAfferent ώστε οι τιμές της εισόδου να συνδεθούν με την εισάγουσα προβολή του επιπέδου της περιοχής V1. Σε αυτό το σημείο ξεκινά να εκτελείται ο αλγόριθμος επαναλαμβανόμενα για 20.000 επαναλήψεις. Σε κάθε επανάληψη η είσοδος αλλάζει χρησιμοποιώντας τη συνάρτηση randomGaussian για να παραχθούν τυχαία Gaussian μοντέλα δραστηριότητας. Στο πρώτο στάδιο υπολογίζεται η πρώτη ενεργοποίηση των νευρώνων, από τα βάρη της εισάγουσας προβολής και την αντίστοιχη είσοδο, η οποία κανονικοποιείται με βάση τη σιγμοειδή συνάρτηση. Στην συνέχεια με βάση την πρώτη ενεργοποίηση των νευρώνων και τα βάρη των ανασταλτικών και διεγερτικών συνδέσεων υπολογίζεται μια νέα τιμή για τους νευρώνες η οποία κανονικοποιείται και αυτή, όπως και παραπάνω, με τη σιγμοειδή συνάρτηση. Αυτή η διαδικασία επεξεργασίας των νευρώνων επαναλαμβάνεται για 9-13 φορές με βάση το σημείο που βρισκόμαστε κάθε φορά στις 20.000 επαναλήψεις(αυξάνεται σταδιακά) με την είσοδο σταθερή. Τέλος αφού η ενεργοποίηση των νευρώνων έχει κατασταλάξει (μετά από τις 9-13 επαναλήψεις), τα βάρη στις συνδέσεις κάθε νευρώνα και στις τρεις προβολές πρέπει να προσαρμοστούν ξανά με βάση τον κανόνα Hebbian και να κανονικοποιηθούν ώστε το άθροισμά τους να είναι σταθερό και ίσο με 1.

Στο [σχήμα 3.1] φαίνεται και γραφικά η σειρά της εκτέλεσης των βασικών συναρτήσεων που ακολουθείται από τον αλγόριθμο LISSOM.

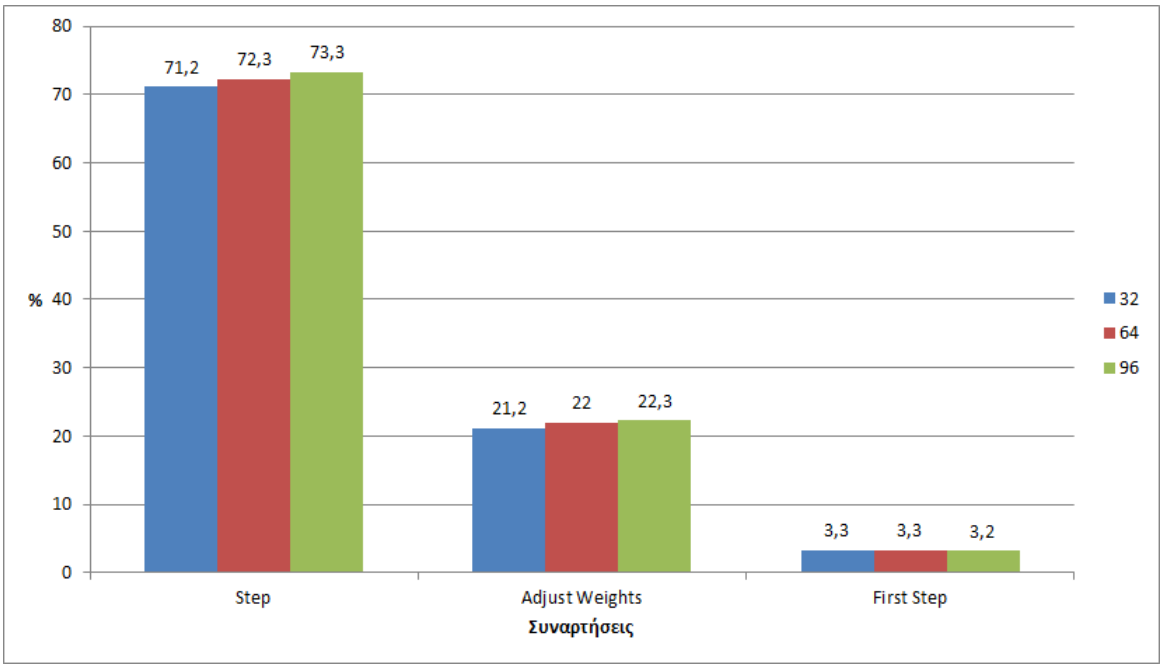



Σχήμα 3.1: **Εκτέλεση αλγορίθμου**

## 3.2 Profiling

Η πρώτη φάση της μελέτης του αλγορίθμου LISSOM ήταν το profiling. Με αυτή την διαδικασία εκτιμήθηκε πόσο βαρύς είναι υπολογιστικά ο συγκεκριμένος αλγόριθμος. Πιο συγκεκριμένα μέσω του profiling μπορούμε να βρούμε τις συναρτήσεις αυτές οι οποίες καταναλώνουν τον μεγαλύτερο χρόνο από τον συνολικό χρόνο εκτέλεσης του αλγορίθμου. Ο αλγόριθμος που βασιστήκαμε είναι γραμμένος στις γλώσσες C και CUDA. Η C έχει κυρίως βοηθητική λειτουργία στην εκτέλεση του αλγορίθμου αφού όλα τα υπολογιστικά κομμάτια του, καταλήγουν να εκτελούνται στην κάρτα γραφικών του υπολογιστή από την CUDA. Για την διαδικασία του profiling χρησιμοποιήθηκε το εργαλείο Visual Profiler της NVIDIA που εμπεριέχεται στο Toolkit τις CUDA.

Με βάση το profiling του αλγορίθμου πήραμε τα εξής συμπεράσματα. Όπως φαίνεται και στο [σχήμα 3.2] η συνάρτηση που καταναλώνει την περισσότερη ώρα, με διαφορά, σε μια εκτέλεση του αλγορίθμου, ανεξάρτητα του μεγέθους της περιοχής V1 ή της εισόδου, είναι η συνάρτηση Step (περίπου το 70% του χρόνου εκτέλεσης). Η συνάρτηση Step υπολογίζει τις νέες τιμές των νευρώνων, οι οποίες στην συνέχεια χρησιμοποιούνται από την συνάρτηση AdjustWeights η οποία όπως φαίνεται και στο [σχήμα 3.2] είναι η δεύτερη σε σειρά υπολογιστικά βαριά συνάρτηση του αλγορίθμου (περίπου το 21% του χρόνου εκτέλεσης). Τέλος η τρίτη σε σειρά συνάρτηση που καταναλώνει τον μεγαλύτερο χρόνο είναι η FirstStep. Αν και το ποσοστό της συγκεκριμένης συνάρτησης δεν είναι ιδιαίτερα μεγάλο, μόλις 3%, αποφασίσαμε να ασχοληθούμε και με αυτή, κυρίως λόγο της σημασίας της αφού υπολογίζει την πρώτη ενεργοποίηση των νευρώνων στον φλοιό όπου στην συνέχεια χρησιμοποιείται σαν είσοδος στην συνάρτηση Step.




Σχήμα 3.2: **Χρόνος εκτέλεσης συναρτήσεων**

Οπότε με βάση τα αποτελέσματα του profiling αποφασίσαμε να ασχοληθούμε με τη μελέτη των τριών αυτών συναρτήσεων, Step, AdjustWeights, FirstStep. Έχοντας σαν χρόνο εκτέλεσης και για τις τρεις αυτές συναρτήσεις περίπου το 95% του συνολικού χρόνου εκτέλεσης του αλγορίθμου περιμένουμε να έχουμε μια αρκετά καλή επιτάχυνση στην εκτέλεση του. Σύμφωνα με τον νόμο του Amdahl [10], έχουμε οτι το τμήμα που

κατέχει το 5% του συνολικού χρόνου εκτέλεσης δεν θα αλλάξει, ενώ το 95% είναι αυτό που θέλουμε να βελτιώσουμε. Οπότε η μέγιστη επιτάχυνση που μπορεί να επιτευχθεί θεωρητικά είναι 1/(1-0.95)=20 φορές πιο γρήγορα σε σχέση με το αρχικό μοντέλο.

## 3.3 Συναρτήσεις ενδιαφέροντος

Όπως αναφέρθηκε και προηγουμένως επιλέξαμε να ασχοληθούμε με τρεις συναρτήσεις που όπως φάνηκε από το profiling του αλγορίθμου καταναλώνουν τον μεγαλύτερο χρόνο της εκτέλεσης του. Η σειρά εκτέλεσης των συναρτήσεων αυτών είναι σταθερή σε όλη την διάρκεια που εκτελείται ο αλγόριθμος συνολικά. Επίσης δεν μπορούν να γίνουν παράλληλα γιατί τα αποτελέσματα για κάθε μια από τις τρεις συναρτήσεις είναι είσοδος της επόμενης. Στην συνέχεια του κειμένου θα γίνει η ανάλυση των τριών αυτών συναρτήσεων από την μεριά των υπολογισμών που υλοποιεί η συνάρτηση όπως επίσης τα ορίσματα που είναι αναγκαία και τα αποτελέσματα που παράγει με αυτά.

### 3.3.1 Ανάλυση της συνάρτησης Step

Όπως έχουμε προαναφέρει η συνάρτηση Step καταναλώνει τον μεγαλύτερο χρόνο από τον συνολικό χρόνο εκτέλεσης του αλγορίθμου με διαφορά. Ο κύριος σκοπός της συγκεκριμένης συνάρτησης είναι ο υπολογισμός των νέων τιμών των νευρώνων στην περιοχή V1 του οπτικού φλοιού με βάση τα βάρη των διεγερτικών, ανασταλτικών συνδέσεων και της πρώτης ενεργοποίησης των νευρώνων από την δεδομένη είσοδο κάθε φορά. Είναι μια διαδικασία που επαναλαμβάνεται για 9-13 φορές καθώς ο αλγόριθμος προχωρά.

Η συνάρτηση Step δέχεται σαν ορίσματα:

- τον πίνακα με τις τιμές των νευρώνων
- το μέγεθος των δυο διαστάσεων w,h του οπτικού φλοιού (περιοχή V1)
- τον πίνακα με τις τιμές της πρώτης ενεργοποίησης των νευρώνων της περιοχής V1 από την διαφορετική είσοδο σε κάθε επανάληψη
- Δεδομένα για τη διεγερτική προβολή:
  - Numreceptors: μονοδιάστατος πίνακας που περιέχει τον αριθμό των διεγερτικών συνδέσεων για κάθε νευρώνα.
  - Startindex: μονοδιάστατος πίνακας που περιέχει δείκτες από τους οποίους ξεκινούν τα βάρη που μας ενδιαφέρουν για το κάθε νευρώνα στον πίνακα των διεγερτικών βαρών.
  - Gamma: συντελεστής κλίμακας για τις διεγερτικές συνδέσεις.
  - Weights: πίνακας με τα βάρη των διεγερτικών συνδέσεων.
- Δεδομένα για τη ανασταλτική προβολή:
  - Numreceptors_I: μονοδιάστατος πίνακας που περιέχει τον αριθμό των ανασταλτικών συνδέσεων για κάθε νευρώνα.
  - Startindex_I: μονοδιάστατος πίνακας που περιέχει δείκτες από τους οποίους ξεκινούν τα βάρη που μας ενδιαφέρουν για το κάθε νευρώνα στον πίνακα των ανασταλτικών βαρών.
  - Gamma_I: συντελεστής κλίμακας για τις ανασταλτικές συνδέσεις.

– Weights_I: πίνακας με τα βάρη των ανασταλτικών συνδέσεων.

Για κάθε επανάληψη η έξοδος της συνάρτησης είναι οι νέες τιμές των νευρώνων της περιοχής V1 στον πίνακα.

Η διαδικασία υπολογισμού που ακολουθείται είναι η εξής:

Για κάθε νευρώνα που υπάρχει στον οπτικό φλοιό έχει υπολογιστεί ο αριθμός των συνδέσεων και ο δείκτης της αρχής των βαρών στον αντίστοιχο πίνακα (διεγερτικό, ανασταλτικό) και δίνεται ως είσοδος στη συνάρτηση. Με βάση αυτές τις δύο μεταβλητές ξεκινά μια επανάληψη μεγέθους w*h όπου w,h οι διαστάσεις του οπτικού φλοιού. Για κάθε νευρώνα βρίσκουμε το δείκτη που το αφορά στον πίνακα Startindex και ξεκινάμε μια επανάληψη ίση με την τιμή στον αντίστοιχο πίνακα Numreceptors που το αφορά. Με τις τιμές από τους δύο αυτούς πίνακες βρίσκονται όλα οι αλλοί νευρώνες και τα βάρη τους που το επηρεάζουν στον πίνακα των νευρώνων και Weights αντίστοιχα. Η τιμή του νευρώνα (με το οποίο επιδρά) πολλαπλασιάζεται με το βάρος του και προστίθεται σε μια προσωρινή μεταβλητή. Όταν η διαδικασία ολοκληρωθεί το άθροισμα αυτό πολλαπλασιάζεται με έναν συντελεστή κλίμακας. Η παραπάνω διαδικασία συμβαίνει με τον ίδιο ακριβώς τρόπο και για τις διεγερτικές και για τις ανασταλτικές συνδέσεις και μας δίνει το τελικό αποτέλεσμα της τιμής του νεύρου απλά προσθέτοντας στην πρώτη ενεργοποίηση του νεύρου την τιμή που υπολογίστηκε από τις διεγερτικές συνδέσεις και αφαιρώντας την τιμή από τις ανασταλτικές συνδέσεις.

$$\eta_{ij}(t) = \sigma(s_{ij} + \gamma E \sum_{kl} \eta_{kl}(t-1) E_{kl,ij} - \gamma I \sum_{kl} \eta_{kl}(t-1) I_{kl,ij}) \tag{3.1}$$

Όπου $\eta(t)$ είναι η τιμή του νευρώνα και τα τρία ορίσματα που αθροίζονται μέσα στην σιγμοειδή συνάρτηση είναι: $s_{ij}$ πρώτη ενεργοποίηση του νευρώνα,$E_{kl,ij}$είναι η διεγερτική ενεργοποίηση από τον νευρώνα (k,l) στο (i,j), $I_{kl,ij}$ είναι η ανασταλτική ενεργοποίηση από το νευρώνα (k,l) στο (i,j)και $\eta(t-1)$ είναι η τιμή του νευρώνα στην προηγούμενη επανάληψη. Τέλος τα $\gamma E$ και $\gamma I$ αποτελούν έναν συντελεστή δύναμης των δύο συνδέσεων.

### 3.3.2 Ανάλυση της συνάρτησης AdjustWeights

Πρόκειται για την δεύτερη συνάρτηση από πλευράς υπολογιστικού χρόνου με την οποία πρόκειται να ασχοληθούμε. Ο κύριος σκοπός αυτής της συνάρτησης είναι ο υπολογισμός των νέων τιμών των βαρών και στις τρείς προβολές εισάγων, ανασταλτική, διεγερτική. Αυτό υπολογίζετε λαμβάνοντας υπόψιν μας την προηγούμενη τιμή τους και τις νέες τιμές των νευρώνων που υπολογίστηκαν από την συνάρτηση Step αφού έφτασε σε σημείο ισορροπίας (μετά από 9-13 επαναλήψεις).

Η συνάρτηση AdjustWeights δέχεται σαν ορίσματα:

- τον πίνακα με τις τιμές των νευρώνων
- το μέγεθος των δυο διαστάσεων w,h του οπτικού φλοιού (περιοχή V1)
- Δεδομένα για κάθε μια από τις τρεις προβολές(εισάγων, ανασταλτική, διεγερτική):

- Numreceptors: μονοδιάστατος πίνακας που περιέχει τον αριθμό των συνδέσεων για κάθε νευρώνα.
- Startindex: μονοδιάστατος πίνακας που περιέχει δείκτες απο τους οποίους ξεκινούν τα βάρη που μας ενδιαφέρουν για το κάθε νευρώνα στον πίνακα των βαρών.
- Alpha: μεταβλητή με τον ρυθμό μάθησης για κάθε τύπο σύνδεσης.
- Weights: πίνακας με τα βάρη των συνδέσεων.

Η έξοδος της συνάρτησης είναι ο πίνακας με τις νέες τιμές των βαρών.

Η διαδικασία υπολογισμού που ακολουθείται είναι η εξής:

Για κάθε νευρώνα που υπάρχει στον οπτικό φλοιό έχει υπολογιστεί ο αριθμός των συνδέσεων και ο δείκτης της αρχής των βαρών στον αντίστοιχο πίνακα (εισάγων, διεγερτικό, ανασταλτικό) και δίνεται ως είσοδος στη συνάρτηση. Αρχικά για κάθε νευρώνα που εξετάζουμε δημιουργούμε ένα άθροισμα από όλες τις συνδέσεις που το αφορούν το οποίο αποτελείται από : η τιμή του νευρώνα (που μας απασχολεί) πολλαπλασιάζεται με την τιμή του νευρώνα της σύνδεσης και στην συνέχεια με την τιμή του ρυθμού μάθησης για τον συγκεκριμένο τύπο σύνδεσης , τέλος το γινόμενο αυτό προστίθεται στην τιμή που είχε το βάρος για την συγκεκριμένη σύνδεση. Η διαδικασία συνεχίζει κάνοντας τον παραπάνω υπολογισμό ξανά για κάθε σύνδεση αλλά αυτή την φορά το αποτέλεσμα διαιρείται με το άθροισμα που είχαμε υπολογίσει, ώστε το άθροισμα των βαρών των συνδέσεων για τον συγκεκριμένο νευρώνα να έχει την τιμή 1. Η διαδικασία αυτή συνεχίζεται για όλους τους νευρώνες που υπάρχουν στον πίνακα. Επίσης σειριακά υπολογίζονται και οι τρεις συνδέσεις.

$$w_{ij,mn}(t+\delta t) = \frac{w_{ij,mn}(t) + \alpha\eta_{ij}X_{mn}}{\sum_{mn}[w_{ij,mn}(t) + \alpha\eta_{ij}X_{mn}]} \tag{3.2}$$

Όπου το $\eta_{ij}$ είναι η τιμή του νευρώνα (i,j) μετά το τέλος της συνάρτησης Step. Το $w_{ij,mn}$ είναι το βάρος της σύνδεσης (εισάγων, ανασταλτικής, διεγερτικής), το $\alpha$ είναι ο ρυθμός μάθησης για κάθε τύπο σύνδεσης και το $X_{mn}$ είναι η τιμή του νευρώνα στην θέση της σύνδεσης.

### 3.3.3 Ανάλυση της συνάρτησης FirstStep

Η τελευταία συνάρτηση που θα ασχοληθούμε είναι η FirstStep Ο κύριος σκοπός αυτής της συνάρτησης είναι ο υπολογισμός της πρώτης ενεργοποίησης των νευρώνων, από τα βάρη της εισάγουσας προβολής και την είσοδο, η οποία κανονικοποιείται με βάση τη σιγμοειδή συνάρτηση.

Η συνάρτηση FirstStep δέχεται σαν ορίσματα:

- τον πίνακα με τις τιμές των εισάγων νευρώνων του αμφιβληστροειδή
- το μέγεθος των δυο διαστάσεων w,h του οπτικού φλοιού (περιοχή V1)
- Δεδομένα για τη εισάγων προβολή:

- Numreceptors: μονοδιάστατος πίνακας που περιέχει τον αριθμό των εισάγων συνδέσεων για κάθε νευρώνα .
- Startindex: μονοδιάστατος πίνακας που περιέχει δείκτες από τους οποίους ξεκινούν τα βάρη που μας ενδιαφέρουν για το κάθε νευρώνα στον πίνακα των εισάγων βαρών.
- Weights: πίνακας με τα βάρη των εισάγων συνδέσεων.

Η έξοδος της συνάρτησης είναι ο πίνακας με την πρώτη ενεργοποίηση των νευρώνων.

Η διαδικασία υπολογισμού που ακολουθείται είναι η εξής:

Για κάθε νευρώνα που υπάρχει στον οπτικό φλοιό έχει υπολογιστεί ο αριθμός των συνδέσεων και ο δείκτης της αρχής των βαρών στον εισάγων πίνακα και δίνεται ως είσοδος στη συνάρτηση. Με βάση αυτές τις δύο μεταβλητές ξεκινά μια επανάληψη μεγέθους w*h όπου w,h οι διαστάσεις του οπτικού φλοιού. Για κάθε νευρώνα βρίσκουμε το δείκτη που το αφορά στον πίνακα Startindex και ξεκινάμε μια επανάληψη ίση με την τιμή στον αντίστοιχο πίνακα Numreceptors που το αφορά. Με τις τιμές από τους δύο αυτούς πίνακες βρίσκονται όλοι οι άλλοι νευρώνες και τα βάρη τους που το επηρεάζουν στον πίνακα των εισάγων νευρώνων και Weights αντίστοιχα. Η τιμή του εισάγων νευρώνα (με το οποίο επιδρά) πολλαπλασιάζεται με το βάρος του και προστίθεται σε μια προσωρινή μεταβλητή. Το άθροισμα αυτό είναι η τιμή της πρώτης ενεργοποίησης του νευρώνα .

$$\eta_{ij} = \sigma\Big(\sum_{a,b} \xi_{ab}\mu_{ij,ab}\Big) \tag{3.3}$$

Όπου το $\eta_{ij}$ είναι η τιμή του νευρώνα (i,j) για την πρώτη του ενεργοποίηση, το $\xi_{ab}$ είναι η τιμή του εισάγων νευρώνα (a,b) και το $\mu_{ij,ab}$ είναι το εισάγων βάρος για την αντίστοιχη σύνδεση. Τέλος το $\sigma$ είναι μια τμηματική γραμμική προσέγγιση της σιγμοειδής συνάρτησης.

# Κεφάλαιο 4

# Σχεδίαση και Υλοποίηση Αρχιτεκτονικής

Σε αυτή την ενότητα θα αναφερθούμε στην διαδικασία σχεδίασης της αρχιτεκτονικής του συστήματος που υλοποιήθηκε σε αναδιατασσόμενη λογική. Αρχικά περιγράφεται η πρώτη αρχιτεκτονική που υλοποιήθηκε με μια μόνο υπολογιστική μονάδα πάνω στην οποία βασίστηκε και η τελική αρχιτεκτονική. Παρατίθενται οι δυσκολίες και τα προβλήματα που προέκυψαν κατά την διάρκεια της υλοποίησης και ο λόγος που επιλέξαμε να χρησιμοποιήσουμε την υβριδική πλατφόρμα του Convey αντί για μια απλή FPGA. Τέλος αναφέρονται κάποια στοιχεία σχετικά με την κατανάλωση των πόρων του συστήματος από την σχεδίαση μας.

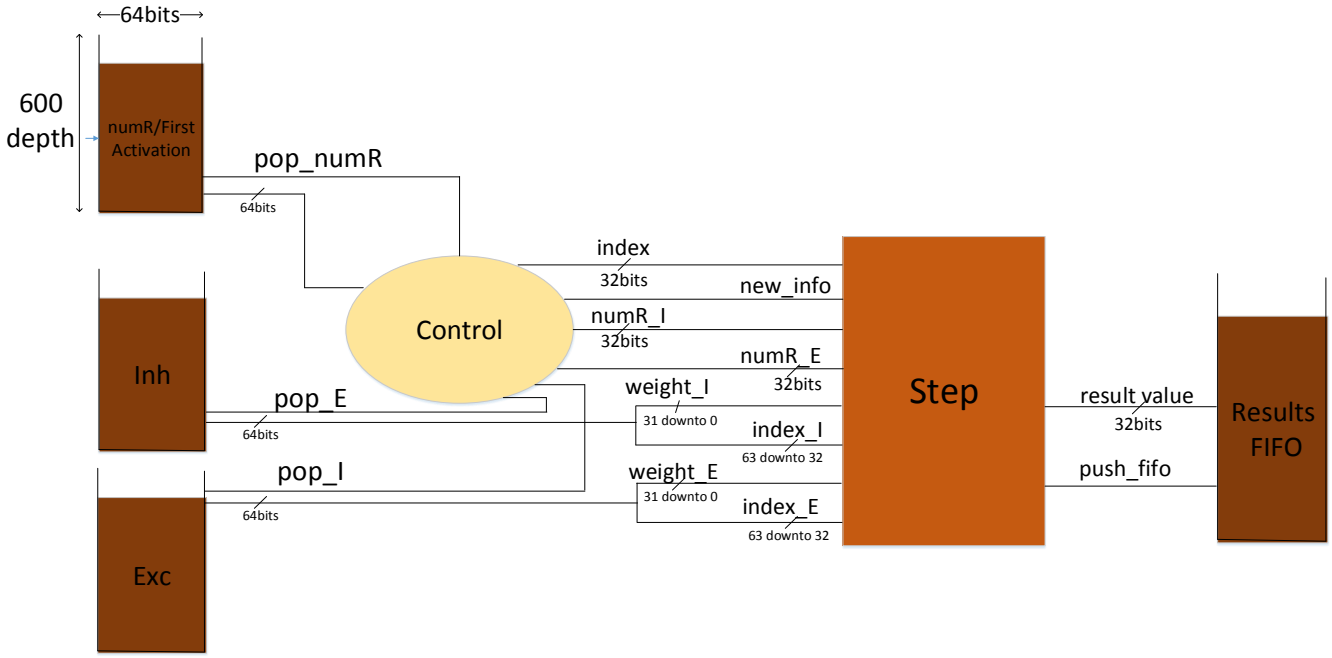
## 4.1 Σχεδίαση και υλοποίηση της αρχικής Αρχιτεκτονικής

Κατά την αρχική προσέγγιση του προβλήματος σχεδιάστηκε μια αρχιτεκτονική η οποία υλοποιούσε μόνο την συνάρτηση Step από τον αλγόριθμο LISSOM. Λόγο του πολύ μεγάλου χρόνου εκτέλεσης που έχει η συνάρτηση Step θα άξιζε να μελετηθεί, ακόμη και χωρίς τις συναρτήσεις AdjustWeights και FirstStep, αν μπορεί μέσω της αναδιατασσόμενης λογικής να βελτιωθεί η απόδοση του χρόνου εκτέλεσης του αλγορίθμου. Επίσης αποτελεί την καρδία του αλγορίθμου αφού υπολογίζει τις νέες τιμές των νευρώνων, από την προηγούμενη κατάσταση του δικτύου, μέσω των οποίων σταδιακά γίνεται η μάθηση του μοντέλου. Μέσω της αρχικής αυτής σχεδίασης ελέγχθηκε η σωστή λειτουργία της συγκεκριμένης συνάρτησης αυτόνομα όπως επίσης υπήρξε και η πρώτη επαφή με τον υβριδικό υπερ-υπολογιστή της Convey και της ιδιαιτερότητες του αφού οι πίνακες των βαρών ήταν αποθηκευμένοι στην κοινή μνήμη του συστήματος λόγο του μεγάλου μέγεθος τους.

Για να μπορέσει να λειτουργήσει αυτόνομα η συνάρτηση Step πρέπει να τροφοδοτηθεί με κάποια δεδομένα τα οποία αναφέρθηκαν και προηγουμένως όταν αναλύθηκε η συγκεκριμένη συνάρτηση. Πιο συγκεκριμένα τα βάρη των ανασταλτικών και διεγερτικών συνδέσεων,για κάθε νευρώνα στο δίκτυο, όπως επίσης και οι δείκτες του νευρώνα με τον οποίο αλληλεπιδρούν είναι αποθηκευμένα στην κοινή μνήμη του Convey και πρέπει να διαβαστούν από την σχεδίαση μας με συγκεκριμένο τρόπο. Εκτός των δυο αυτών πινάκων με τα βάρη υπάρχουν αντίστοιχα και δυο βοηθητικοί πίνακες μεγέθους όσο και ο αριθμός των νευρώνων στο δίκτυο στους οποίους υπάρχει η εξής πληροφορία: για κάθε νευρώνα του δικτύου έχει κρατηθεί ο αριθμός των συνδέσεων του με τους γειτονικούς νευρώνες που τον επηρεάζουν. Με βάση την πληροφορία από τους πίνακες με τον αριθμό των συνδέσεων η σχεδίαση μας γνωρίζει πόσα από τα στοιχεία στον πίνακα των βαρών

πρέπει να διαβαστούν για να υπολογιστεί το άθροισμα μέσω του οποίου στην συνέχεια θα υπολογιστεί η νέα τιμή του νευρώνα.

Στην σχεδίαση μας από την C δίνουμε τις διευθύνσεις μνήμης των πινάκων από τα διεγερτικά βάρη με τους δείκτες των συνδέσεων, τα ανασταλτικά βάρη με τους αντίστοιχους δείκτες των συνδέσεων τους, την διεύθυνση μνήμης του ενιαίου πίνακα με τον αριθμό των συνδέσεων για κάθε προβολή, την διεύθυνση μνήμης του πίνακα της πρώτης αρχικοποίησης των νευρώνων από την είσοδο, την διεύθυνση μνήμης του πίνακα για την εγγραφή των αποτελεσμάτων στο τέλος, όπως επίσης τα μεγέθη όλων αυτών των πινάκων που προαναφέρθηκαν και ο αριθμός των επαναλήψεων που πρέπει να εκτελέσει η συνάρτηση Step. Με αυτά τα δεδομένα το Top module της σχεδίασης μας [σχήμα 4.1] μπορεί να εισάγει όλες τις τιμές που είναι αναγκαίες για την εκτέλεση της συνάρτησης Step η οποία θα τρέξει επαναλαμβανόμενα για 9-13 φορές.




Σχήμα 4.1: **Top module Αρχικής Αρχιτεκτονικής**

Για τον σκοπό αυτό χρησιμοποιήθηκαν 3 από τους 16 ελεγκτές μνήμης για την εισαγωγή των πινάκων και αυτό γιατί η αρχικοποίηση και το γράψιμο των αποτελεσμάτων γίνεται σε διαφορετικό χρόνο από την εκτέλεση τις συνάρτησης με αποτέλεσμα ο μέγιστος αριθμός ελεγκτών που χρειαζόμαστε για να εισάγουμε τα δεδομένα παράλληλα να είναι τρεις. Τα δεδομένα που εισάγει στην σχεδίαση ο κάθε ελεγκτής μνήμης αποθηκεύονται σε τρεις διαφορετικές FIFO μεγέθους 600 x 64 bits η κάθε μια, που λειτουργούν σαν Buffers, και μέσω ενός Control τα δεδομένα αυτά περνάνε μέσα στην σχεδίαση μας με την σωστή σειρά. Τρεις διαφορετικές FSM λειτουργούν ανεξάρτητα από το Control στο Top module της σχεδίασης μας με σκοπό να ζητούν από τους ελεγκτές μνήμης τις σωστές διευθύνσεις στη κοινή μνήμη. Οι τρεις αυτές FSM γνωρίζουν την αρχική διεύθυνση μνήμης του κάθε πίνακα όπως επίσης και τον αριθμό των στοιχείων από τον οποίο αποτελείται ο πίνακας. Αν και οι δυο πίνακες των βαρών με του δείκτες αποτελούνται από πολύ μεγάλο αριθμό στοιχείων, για αυτό άλλωστε και δεν αποθηκεύονται ολόκληροι μέσα σε Block Rams στην σχεδίαση, οι FIFO που λειτουργούν σαν Buffers έχουν βάθος μόνο 600 θέσεις αφού λίγα δεδομένα την φορά είναι αρκετά για να λειτουργεί η συνάρτηση αρκεί οι Buffers αυτοί να μην μένουν χωρίς στοιχεία.

Οι τρεις αυτοί πίνακες που εισάγονται στην σχεδίαση μας από την κοινή μνήμη του Convey στην πραγματικότητα είναι έξι διαφορετικοί πίνακες. Οι πληροφορίες που είναι αποθηκευμένες στους πίνακες αυτούς είναι συγκεκριμένα μεγέθους 32bit για κάθε δεδομένο του πίνακα. Οι πίνακες των βαρών αποτελούνται από 32bit floating τιμές ενώ οι Index, numR πίνακες από 32bit ακεραίους. Το Convey μας δίνει την δυνατότητα να εισάγουμε 64bits δεδομένα σε κάθε κύκλο από κάθε ελεγκτή μνήμης που έχουμε στην διάθεση μας. Γι αυτό τον λόγο οι πίνακες αυτοί περνάνε από μια επεξεργασία στην C ώστε να ενωθούν από έξι σε τρεις. Άρα η μορφή με την οποία οι τιμές των πινάκων καταλήγουν να φτάνουν στην σχεδίαση μας φαίνεται στον πίνακα 4.1.

| Πίνακες | 63 downto 32 | 31 downto 0 |
|---|---|---|
| Ανασταλτικά | index_I | weight_I |
| Διεργετικά | index_E | weight_E |
| Αρχικοποιήσεις | zeros | value |
| Αριθμός συνδέσεων | numReceptors_E | numReceptors_I |

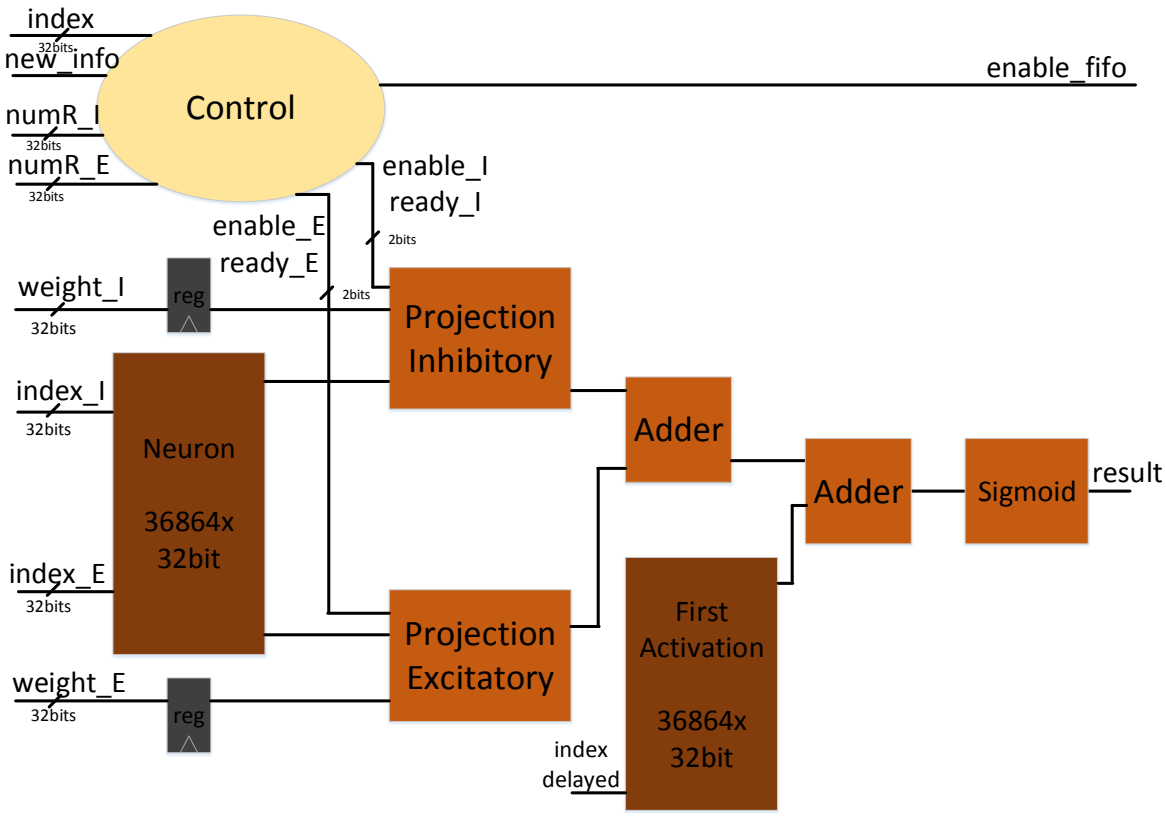
Πίνακας 4.1: Μορφή πινάκων

Το πιο σημαντικό κομμάτι στο συγκεκριμένο σημείο της σχεδίασης είναι το Control. Μέσω αυτού αρχικοποιείται η Block Ram FirstActivation που βρίσκετε μέσα στην σχεδίαση Step [σχήμα 4.2], και περιέχει τις τιμές της πρώτης ενεργοποίησης των νευρώνων από την είσοδο. Για την διαδικασία αυτή μόνο ένας από τους τρεις ελεγκτές μνήμης χρησιμοποιείται και συγκεκριμένα ο πρώτος που στην συνέχεια εισάγει τον πίνακα με τον αριθμό των συνδέσεων για κάθε νευρώνα. Στην συνέχεια το Control ελέγχει με ποιόν νευρώνα ασχολούμαστε κάθε φορά και διαβάζοντας τις αντίστοιχες τιμές από τον βοηθητικό πίνακα γνωρίζει πόσα από τα στοιχεία πρέπει να διαβαστούν από τους δυο πίνακες των βαρών και τροφοδοτεί την σχεδίαση μας με τα εξής δεδομένα:

- index: τον αριθμό του νευρώνα που ασχολούμαστε κάθε φορά,
- numR_I,numR_E: τον αριθμό των διεγερτικών και ανασταλτικών συνδέσεων για τον συγκεκριμένο νευρώνα,
- Weight_I,index_I: το βάρος για την κάθε ανασταλτική σύνδεση όπως επίσης και ο δείκτης του νεύρωνα με τον οποίο σχετίζεται,
- Weight_E,index_E: το βάρος για την κάθε διεγερτική σύνδεση όπως επίσης και ο δείκτης του νεύρωνα με τον οποίο σχετίζεται,
- newinfo: λειτουργεί σαν valid bit στην σχεδίαση μας ώστε να γνωρίζει πως σε αυτό τον κύκλο λαμβάνει ένα σωστό πακέτο δεδομένων.

Όταν και το τελευταία δεδομένα δοθούν στο module Step από το Top module της σχεδίασης το Control ελέγχει σε πια επανάληψη βρίσκεται κάθε φορά και αν έχει φτάσει στο τέλος γράφει τα αποτελέσματα που βρίσκονται στην FIFO Results πίσω στις διευθύνσεις μνήμης της κοινής μνήμης του Convey ώστε να μπορούν να διαβαστούν από το Software. Σε αντίθετη περίπτωση, όταν οι επαναλήψεις δεν έχουν φτάσει στο τέλος τους, τα αποτελέσματα που είναι αποθηκευμένα στην FIFO Results πρέπει να γραφτούν με την σειρά στην Block Ram που κρατάει τις τιμές των νευρώνων(Neurons σχήμα 4.2). Για την συγκεκριμένη διαδικασία το Control του Top module τραβάει τα δεδομένα από

την FIFO και τα στέλνει στο module Step για να γραφτούν στην μνήμη Neurons. Το μέγεθος των μνήμων Neurons, FirstActivation όπως επίσης και της FIFO με τα αποτελέσματα έχουν βάθος 36864 γιατί αυτό είναι και το μέγεθος του μεγαλύτερου δικτύου από νευρώνες που προσομοιώνουμε.

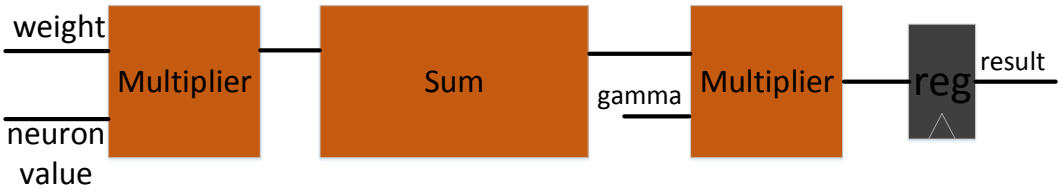



Σχήμα 4.2: **Αρχιτεκτονική συνάρτησης Step**

Όλη η σχεδίαση του Top module λειτουργεί κυρίως βοηθητικά ώστε να παρέχονται τα σωστά δεδομένα στην σχεδίαση Step που υλοποιεί και την αντίστοιχη συνάρτηση. Σκοπός του module Step είναι να υπολογίσει τις νέες τιμές των νευρώνων από την ήδη υπάρχον κατάσταση του δικτύου. Στην πρώτη επανάληψη η κατάσταση δίνεται από την αρχικοποίηση των μνημών Neurons και FirstActivation, ενώ στις επόμενες οι νέες τιμές των νευρώνων κάθε φορά γράφονται στην μνήμη Neurons στο τέλος της κάθε επανάληψης. Η όλη σχεδίαση μπορεί να δεχθεί ένα νέο σετ δεδομένων κάθε κύκλο και παράγει αποτελέσματα ανάλογα με τον αριθμό των συνδέσεων για το συγκεκριμένο νευρώνα που υπολογίζει κάθε φορά. Το βάθος του pipeline είναι 70 κύκλοι αλλά πρέπει πρώτα να υπολογιστεί το άθροισμα των γινομένων των βαρών με τους αντίστοιχους νευρώνες για να συνεχίσει η διαδικασία υπολογισμού.

Πιο συγκεκριμένα η επεξεργασία των δεδομένων ξεκίνα με ένα νέο πακέτο δεδομένων που δέχεται η σχεδίαση Step με το new_info να λειτουργεί σαν valid_bit πως τα δεδομένα του συγκεκριμένου κύκλου είναι σωστά. Με βάση τον δείκτη του νευρώνα με τον οποίο συνδέεται ο νευρώνας που μελετάμε την συγκεκριμένη στιγμή (index) βρίσκουμε την τιμή του μέσω του πίνακα Neurons. Ο συγκεκριμένος πίνακας είναι διπλής πόρτας ώστε να μπορούμε σε κάθε κύκλο να διαβάσουμε 2 στοιχεία του ταυτόχρονα λόγο των 2 προβολών που έχουμε, ανασταλτική και διεγερτική. Η τιμή του νευρώνα που παίρνουμε από τον πίνακα Neurons δίνεται μαζί με την τιμή του βάρους για την συγκεκριμένη σύνδεση

περνάνε σε ένα από τα δυο module Projection ανάλογα αν πρόκειται για ανασταλτική ή διεγερτική σύνδεση αντίστοιχα. Μέσα στο module Projection οι δυο αυτές τιμές πολλαπλασιάζονται και στην συνέχεια δημιουργείται ένα άθροισμα που αποτελείται από όλες τις συνδέσεις που απασχολούν τον νευρώνα που ασχολούμαστε κάθε φορά. Όταν το άθροισμα υπολογιστεί πρέπει να πολλαπλασιαστεί με την έναν συντελεστή κλίμακας, ο οποίος είναι διαφορετικός για κάθε προβολή και το αποτέλεσμα του γινομένου αποθηκεύεται σε έναν καταχωρητή [σχήμα 4.3]. Ο λόγος που το αποτέλεσμα αποθηκεύεται στον καταχωρητή είναι γιατί ο αριθμός των συνδέσεων που έχει ο κάθε νευρώνας είναι διαφορετικός για την ανασταλτική και διεγερτική προβολή. Συγκεκριμένα πάντα οι ανασταλτικές συνδέσεις είναι περισσότερες από τις διεγερτικές. Οπότε το αποτέλεσμα περιμένει μέχρις ότου και οι δυο καταχωρητές να έχουν τα αποτελέσματα που χρειαζόμαστε για να συνεχιστεί ο υπολογισμός.

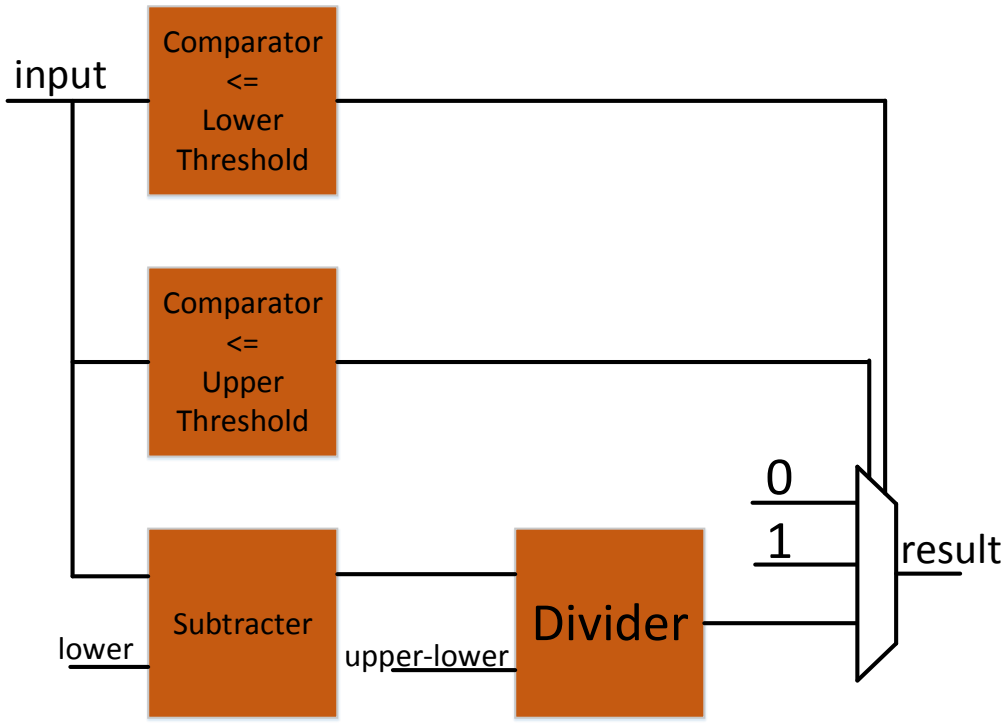



Σχήμα 4.3: **Αρχιτεκτονική προβολής.**

Στην συνέχεια της διαδικασίας τα δυο αποτελέσματα των Projections πρέπει να προστεθούν μεταξύ τους(το αποτέλεσμα της ανασταλτικής προβολής έχει αρνητικό πρόσημο το οποίο αποκτά από το γινόμενο του αθροίσματος με τον αρνητικό συντελεστή κλίμακας) και με την τιμή του νευρώνα από την πρώτη ενεργοποίηση του. Την τιμή της πρώτης ενεργοποίησης την διαβάζουμε από τον πίνακα FirstActivation μέσω του δείκτη (index) του νευρώνα που ασχολούμαστε κάθε φορά. Το αποτέλεσμα του αθροίσματος αυτού πρέπει να κανονικοποιηθεί με την χρήση της σιγμοειδής συνάρτησης. Η υλοποίηση της συγκεκριμένης συνάρτησης φαίνεται στο [σχήμα 4.4]. Το άθροισμα του αποτελέσματος περνάει σαν είσοδο στην συνάρτηση και παράλληλα συγκρίνεται αν είναι μικρότερο από το κατώτερο όριο τιμής του νευρώνα και το ανώτερο όριο τιμής. Τα δυο αυτά bits που βγαίνουν σαν αποτέλεσμα από τις δυο αυτές συγκρίσεις χρησιμοποιούνται σαν select στον πολυπλέκτη που φαίνεται στην συνέχεια. Επίσης από την είσοδο της συνάρτησης παράλληλα με της συγκρίσεις αφαιρείται από την είσοδο το κατώτερο όριο τιμής και στην συνέχεια το αποτέλεσμα αυτό διαιρείται με την τιμή του αποτελέσματος της αφαίρεσης του ανώτερου ορίου με το κατώτερο όριο. Η τελική τιμή του νευρώνα θα δοθεί από τα αποτελέσματα των συγκρίσεων.

Αν η τιμή που έρχεται σαν είσοδος είναι:

- μικρότερη από το κατώτερο όριο τότε η νέα τιμή του νευρώνα είναι 0,
- μεγαλύτερη από το ανώτερο όριο τότε η νέα τιμή του νευρώνα είναι 1,
- μικρότερη από το ανώτερο όριο και μεγαλύτερη από το κατώτερο όριο τότε η νέα τιμή του νευρώνα δίνεται από το αποτέλεσμα της διαίρεσης.

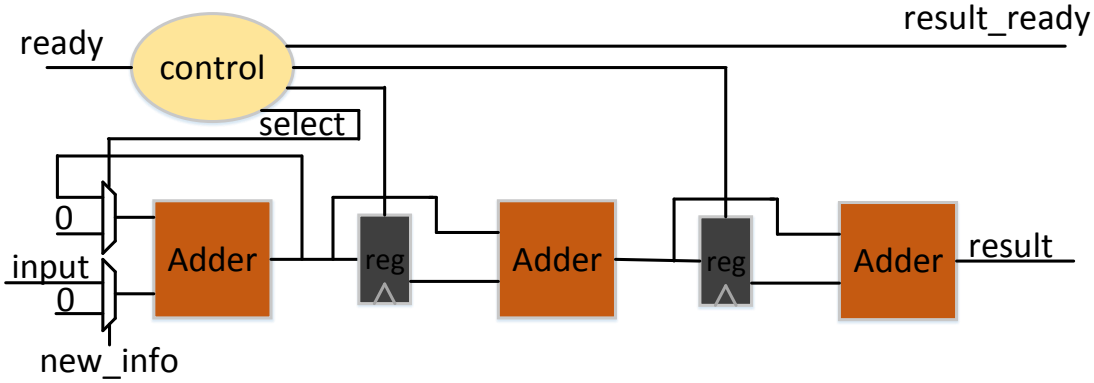



Σχήμα 4.4: **Αρχιτεκτονική σιγμοειδής συνάρτησης.**

Το Control της σχεδίασης που υλοποιεί την συνάρτηση Step έχει σαν σκοπό του να ελέγχει κυρίως την σωστή λειτουργία των αθροισμάτων που υλοποιούνται μέσα στις δυο ξεχωριστές σχεδιάσεις των προβολών. Μέσω των τιμών από τις διεγερτικές και αναστολτικές συνδέσεις γνωρίζει πότε έρχονται τα τελευταία δεδομένα για την κάθε προβολή και στέλνει αντίστοιχα σήματα προς τις σχεδιάσεις Projection ώστε ο καταχώρητης που βρίσκετε στο τέλος της συγκεκριμένης μονάδας να κρατήσει την τιμή του αποτελέσματος για να μην χαθεί στους επόμενους κύκλους. Όταν τα δεδομένα και των δυο προβολών εισαχθούν στο σύστημα ενεργοποιεί το bit enable_fifo για να αποθηκευτεί το νέο αποτέλεσμα του νευρώνα στην FIFO Results. Το σήμα της ενεργοποίησης που θα στείλει το Control θα καθυστερηθεί για 70 κύκλους ώστε να φτάσει στην έξοδο του module Step μαζί με το αποτέλεσμα της συνάρτησης.

Ένα από τα προβλήματα που παρουσιάστηκαν στην υλοποίηση της συνάρτησης Step ήταν πως για να μπορεί η σχεδίαση μας να τρέχει με 150Mhz ρολόι έπρεπε όλες οι υπολογιστικές μονάδες (αθροιστές, πολλαπλασιαστές και τα άλλα) να παίρνουν παραπάνω από ένα κύκλο, επειδή οι τιμές των μεταβλητών που υπολογίζουν δεν είναι ακέραιοι αλλά floating point. Το πρόβλημα παρουσιάστηκε μόνο στο άθροισμα των γινομένων μέσα στο module Projection αφού για τις υπόλοιπες υπολογιστικές μονάδες από την στιγμή που είναι πλήρως pipelined δεν μας ενοχλεί η καθηστέρηση που προστίθεται. Η όλη σχεδίαση θέλουμε να μπορεί να δεχθεί ένα νέο πακέτο δεδομένων κάθε κύκλο. Για να πετύχουμε τα 150Mhz στον αθροιστή πρέπει να υλοποιηθεί σαν ένας pipelined αθροιστής 4ων κύκλων τουλάχιστον, ο οποίος αν και μπορεί να δεχθεί νέες τιμές κάθε κύκλο εμείς θέλουμε κάθε κύκλο να προστίθεται η νέα τιμή του γινομένου με το άθροισμα μέχρι εκείνη την στιγμή. Γι αυτό τον λόγο δημιουργήθηκε μια σχεδίαση η οποία φαίνεται στο [σχήμα 4.5]. Η σχεδίαση αυτή αποτελείτε από τρεις αθροιστές floating point 32bits δυο καταχωρητών και μίας FSM που ελέγχει τον ένα από τους δυο πολυπλέκτες και τα enable των καταχωρητών. Ο τρόπος με τον οποίο λειτουργεί η συγκεκριμένη συνάρτηση είναι

αρκετά απλός. Αρχικά δέχεται σαν είσοδο την νέα τιμή που θέλουμε να προσθέσουμε στο άθροισμα μαζί με το valid_bit (new_info) όπως επίσης και την πληροφορία αν η τιμή εισόδου στον κάθε κύκλο είναι η τελευταία του αθροίσματος. Μέσα στον πρώτο αθροιστή μέχρι να έρθουν όλες οι τιμές που μας ενδιαφέρουν δημιουργούνται 4 ημιαθροίσματα. Για κάθε κύκλο που δεν έχουμε νέα τιμή εισόδου ο δεύτερος πολυπλέκτης προσθέτει το αποτέλεσμα του πρώτου αθροιστή με το μηδέν για να μην χαλάσουν τα δεδομένα μας από σκουπίδια που θα υπάρχουν σε κάποιους κύκλους. Όταν το σήμα ready γίνει ένα, δηλαδή έχουμε το τελευταίο στοιχείο του αθροίσματος, η είσοδος θα αθροιστεί με το ημιάθροισμα σε αυτόν τον κύκλο αλλά για τους επόμενους 4 κύκλους ο πρώτος πολυπλέκτης θα επιλέγει να δώσει στον αθροιστή μηδέν για είσοδο ώστε το επόμενο άθροισμα που μπορεί να αρχίσει να υπολογίζεται από τον επόμενο κιόλας κύκλο να μην επηρεαστεί από τα αποτελέσματα που αφορούν το προηγούμενο άθροισμα. Από την στιγμή που θα έρθει το σήμα πως το τελευταίο στοιχείο για τον υπολογισμό ήρθε σαν είσοδος για τους επόμενους 4 κύκλους στην έξοδο του πρώτου αθροιστή θα έχουμε τα 4 ημιαθροίσματα που θέλουμε να προσθέσουμε. Αυτό υλοποιείται μέσω των επόμενων δυο αθροιστών και των δυο καταχωρητών με τον εξής τρόπο. Τον πρώτο κύκλο το αποτέλεσμα του πρώτου αθροιστή θα αποθηκευθεί στον πρώτο καταχωρητή και τον επόμενο το δεύτερο αποτέλεσμα μαζί με την έξοδο του καταχωρητή θα δοθεί σαν είσοδος στον δεύτερο αθροιστή. Τους επόμενους δυο κύκλους θα επαναληφθεί η ίδια διαδικασία για το τρίτο και τέταρτο ημιάθροισμα. Τέλος τα δυο αυτά αποτελέσματα θα δοθούν σαν είσοδος στον τρίτο αθροιστή, πάλι με την βοήθεια του δεύτερου καταχωρητή, ώστε να υπολογιστεί η τελική τιμή του αθροίσματος. Η συνολική καθυστέρηση που έχει η σχεδίαση για τον υπολογισμό του αθροίσματος από την στιγμή που θα δοθεί το τελευταίο στοιχείο είναι 12 κύκλοι.



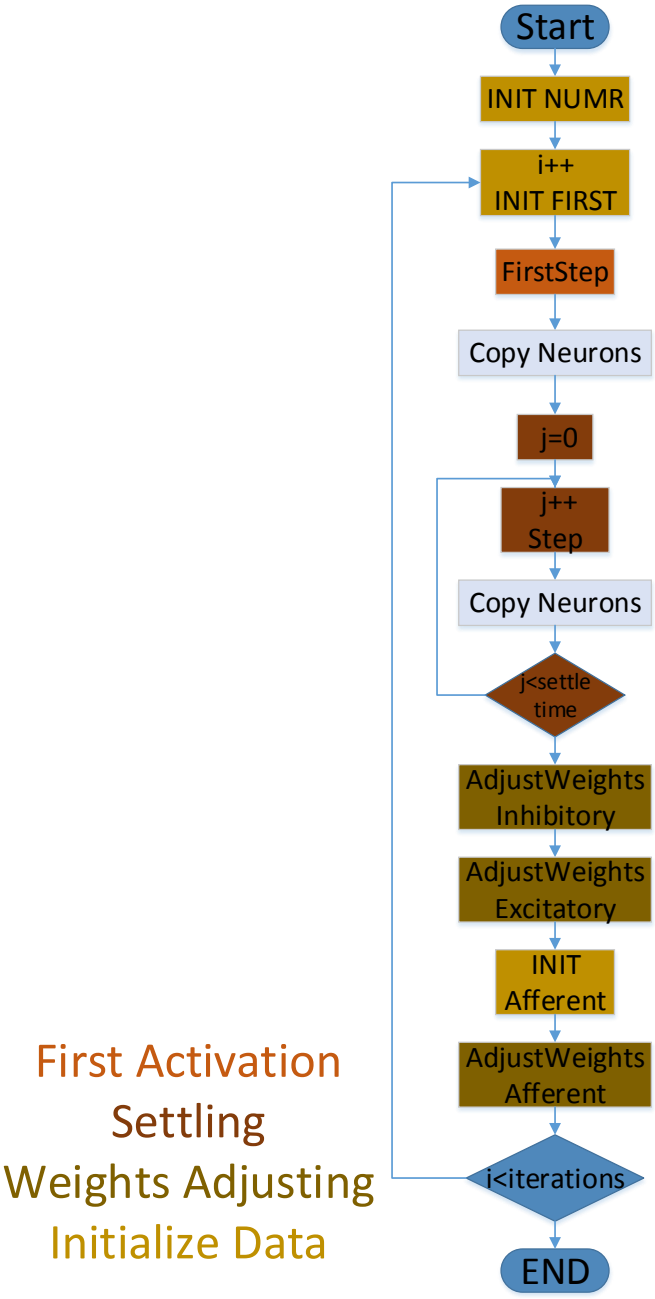

Σχήμα 4.5: **Αρχιτεκτονική αθροίσματος.**

## 4.2 Τελική σχεδίαση και υλοποίηση της Αρχιτεκτονικής

Όπως προαναφέρθηκε η συνάρτηση Step καταλαμβάνει το μεγαλύτερο χρόνο από την εκτέλεση του αλγορίθμου LISSOM και θα μπορούσε από μόνη της να μας προσφέρει κάποια επιτάχυνση στον χρόνο εκτέλεσης του αλγορίθμου. Όμως το βασικό πρόβλημα που δημιουργείται με το να υλοποιήσουμε μόνο την συνάρτηση Step στο FPGA είναι η αλληλεπικοινωνία μεταξύ των συναρτήσεων FirstStep και AdjustWeights οι οποίες εκτελούνται για κάθε επανάληψη (από τις 20000) πριν και μετά την συνάρτηση Step. Με

τον τρόπο που λειτουργεί το μηχάνημα της Convey θα έπρεπε για κάθε επανάληψη η συναρτήση FirstStep να εκτελεστεί στο Software στην συνέχεια να καλέσει την διεπαφή για να εκτελεστεί η συνάρτηση Step μέσα στο FPGA και αφού η συνάρτηση τελειώσει και γράψει τις νέες τιμές των νευρώνων πίσω στην κοινή μνήμη τότε μπορεί να εκτελεστεί η συνάρτηση AdjustWeights. Φαίνεται λοιπόν το πρόβλημα τις συνεχής αλλαγής της εκτέλεσης του αλγορίθμου από το Software στο Hardware η οποία εναλλαγή έχει και ένα μεγάλο χρονικό κόστος για να αρχικοποιηθεί το μηχάνημα της Convey κάθε φορά. Για αυτό τον λόγο οι συναρτήσεις FirstStep και AdjustWeights υλοποιούνται στην τελική σχεδίαση και αυτές μέσα στο FPGA. Με αυτόν τον τρόπο το Software καλεί μια φορά την διεπαφή της FPGA και για τις 20000 επαναλήψεις δίνοντας τους πίνακες με τα βάρη για κάθε προβολή, τον αριθμό των συνδέσεων για κάθε προβολή, τον πίνακα με τις εισόδους για να ξεκινήσει να εκτελείται η συνάρτηση FirstStep σε κάθε επανάληψη, τα μεγέθη όλων αυτών τον πινάκων και τον αριθμό των επαναλήψεων που θέλουμε να εκτελέσει ο αλγόριθμος για να γίνει η μάθηση του δικτύου.

Εκτός της προσθήκης των δυο συναρτήσεων στο FPGA υπήρξαν και κάποιες ακόμη αλλαγές που θα εξυπηρετήσουν κυρίως στην δημιουργία 8 παράλληλων υπολογιστικών μονάδων ώστε να μειωθεί ο χρόνος εκτέλεσης του αλγορίθμου. Η κάθε υπολογιστική μονάδα θα επεξεργάζεται το 1 / 8 των συνολικών νευρώνων και αντίστοιχα τις συνδέσεις για αυτούς τους νευρώνες. Ο αριθμός των υπολογιστικών μονάδων επιλέχθηκε λόγο του περιορισμού που έχει το σύστημα μας στους ελεγκτές μνήμης οι οποίοι όπως έχει αναφερθεί ξανά είναι μόνο 16. Στην αρχική σχεδίαση χρειαζόμασταν 3 ελεγκτές ταυτόχρονα για να λειτουργεί η συνάρτηση Step κάτι το οποίο έχει αλλάξει στην τελική σχεδίαση και χρειαζόμαστε μόνο 2 όπως θα φανεί και παρακάτω. Επίσης κάτι που απασχόλησε αρκετά την συγκεκριμένη σχεδίαση είναι η αντιγραφή των νέων τιμών των νευρώνων στο τέλος κάθε εκτέλεσης της συνάρτησης FirstStep και Step. Για την υλοποίηση της αντιγραφής κυρίως και για τον συνχρονισμό των 8 διαφορετικών Cores που επεξεργάζονται τα δεδομένα υλοποιήθηκε ένα νέο Control σε μία νέα κεντρική σχεδίαση για να μπορεί να ελέγχει την σωστή λειτουργία των μονάδων.

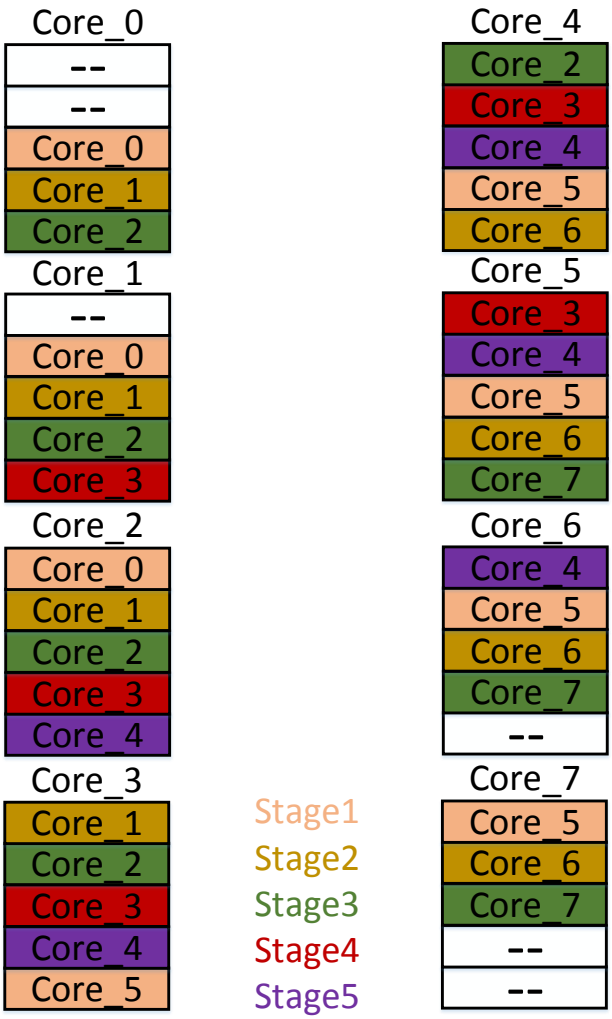



Σχήμα 4.6: **Σχεδίαση του Control και για τα 8 cores.**

Όπως φαίνεται και στο σχέδιο 4.6 η FSM που έχει δημιουργηθεί για τον έλεγχο της σχεδίασης έχει αρκετές καταστάσεις. Αρχικά πρέπει να αρχικοποιηθούν οι μνήμες που κρατάνε τον αριθμό των συνδέσεων του κάθε νευρώνα για κάθε μια από τις τρεις προβολές. Στην συνέχεια η σχεδίαση διαβάζει από την κοινή μνήμη μέσω ενός ελεγκτή μνήμης την είσοδο και την γράφει στον πίνακα Νευρονς που υπάρχει μέσα σε κάθε δρε για να ξεκινήσει η εκτέλεση της συνάρτησης FirstStep. Επόμενο βήμα είναι η εκτέλεση της συνάρτησης FirstStep για να υπολογιστούν οι πρώτες ενεργοποιήσεις των νευρώνων με βάση την κάθε είσοδο σε κάθε επανάληψη. Αφού όλες οι πρώτες ενεργοποιήσεις για

κάθε νευρώνα υπολογιστούν πρέπει να γραφτούν στον πίνακα Νευρονς για να μπορέσει να ξεκινήσει η συνάρτηση Step. Για την αντιγραφή των νευρώνων από τις FIFO που είναι αποθηκευμένες οι νέες τιμές στης μνήμες υπάρχει μια διαδικασία που θα αναλυθεί παρακάτω. Ο αλγόριθμος συνεχίζει εκτελώντας την συνάρτηση Step για 9-13 φορές. Σε κάθε ολοκλήρωση της επανάληψης οι νέες τιμές των νευρώνων που υπολογίστηκαν πρέπει να γραφτούν ξανά πίσω στις μνήμες των νευρώνων όπως και στο τέλος της εκτέλεσης της συνάρτησης FirstStep. Στην συνέχεια αφού τα νεύρα έχουν σταθεροποιηθεί για την συγκεκριμένη είσοδο πρέπει να υπολογιστούν οι νέες τιμές των βαρών. Αυτό γίνεται μέσα σε 4 στάδια. Αρχικά υπολογίζονται οι τιμές των νέων βαρών για την ανασταλτική προβολή και μέσω ενός ελεγκτή μνήμης γράφονται στις ίδιες θέσεις μνήμης που ήταν αποθηκευμένες στην κοινή μνήμη του συστήματος του Convey. Αφού τελειώσει ο υπολογισμός για την ανασταλτική προβολή με τον ίδιο ακριβώς τρόπο ξεκινάνε να υπολογίζονται οι νέες τιμές των βαρών και για την διεγερτική προβολή. Για τον υπολογισμό των νέων βαρών για την εισάγων προβολή πρέπει πρώτα να φέρουμε μέσα στην σχεδίαση μας ξανά τον πίνακα τις εισόδου. Αυτή την φορά δεν θα γραφτεί στην μνήμη Neurons αλλά στην μνήμη FirstActivation. Αφού η εισαγωγή του πίνακα ολοκληρωθεί υπολογίζονται οι νέες τιμές των βαρών για την εισάγων προβολή και γράφονται και αυτές πίσω στην κοινή μνήμη του συστήματος. Τέλος γίνεται ένας έλεγχος για τον αριθμό των επαναλήψεων που έχουν ολοκληρωθεί μέχρι αυτό το σημείο και αν έχει φτάσει στον επιθυμητό αριθμό η εκτέλεση της σχεδίασης σταματά, σε αντίθετη περίπτωση ξεκινάμε πάλι από την αρχή κάνοντας ξανά την ίδια διαδικασία.

Όπως φάνηκε και από την περιγραφή της εκτέλεσης του αλγορίθμου στην σχεδίαση μας, η αντιγραφή των νέων νευρώνων από την κάθε FIFO που αποθηκεύονται προσωρινά, στην μνήμη Neurons μέσα σε κάθε Core γίνεται αρκετές φορές μέσα σε μια επανάληψη. Η αντιγραφή αυτή δεν είναι τόσο απλή διαδικασία γιατί το κάθε Core όπως έχουμε αναφέρει υπολογίζει τις τιμές για ένα ξεχωριστό κομμάτι του δικτύου. Δυστυχώς όμως για τον υπολογισμό των νέων τιμών των νευρώνων το κάθε Core χρειάζεται παραπάνω σημεία του δικτύου από τα οποία ασχολείται λόγω των πλευρικών συνδέσεων που υπάρχουν μεταξύ των νευρώνων. Για αυτό τον λόγο η μνήμη με τους νευρώνες (Neurons) έχει μέγεθος 5 φορές μεγαλύτερο από το 1 / 8 του μέγιστου συνολικού αριθμού νευρώνων στο δίκτυο (36864 συνολικοί νευρώνες). Στην ουσία σε κάθε μνήμη κρατάμε τις τιμές των νευρώνων που ασχολείται το συγκεκριμένο Core όπως επίσης και τις τιμές που παράγουν τα δυο προηγούμενα και επόμενα από αυτό Cores.




Σχήμα 4.7: **Στάδια αντιγραφής.**

Όπως φαίνεται και στο σχήμα 4.7 η διαδικασία αντιγραφής των νευρώνων από την FIFO του κάθε Core, που αποθηκεύονται προσωρινά, στην μνήμη Neurons γίνεται μέσα σε 5 στάδια. Σε κάθε ένα από τα στάδια ένας ή δυο Cores στέλνουν τα αποτελέσματα τους για εγγραφή στα γειτονικά τους. Παραδείγματος χάρη στο πρώτο στάδιο το Core_0 γράφει τα αποτελέσματα του στης μνήμες των Core_0,Core_1,Core_2 και ταυτόχρονα το Core 5 στις μνήμες των Core_3,Core_4,Core_5,Core_6,Core_7. Με αυτόν τον τρόπο μέσα σε αυτά τα 5 στάδια όλα τα Cores θα αντιγράψουν τα αποτελέσματα τους στα γειτονικά τους Cores που τα έχουν ανάγκη για τις επόμενες διαδικασίες.

Το πρόβλημα που δημιουργείται με την μορφή που έχει πάρει η μνήμη Neurons και τον τρόπο που ο πίνακας των συνδέσεων είναι υλοποιημένος στο Software είναι πως ο δείκτης που έχει ο κάθε νευρώνας στον πίνακα των συνδέσεων δεν αντιστοιχεί στον αντίστοιχο δείκτη στην κάθε μνήμη Neurons. Για αυτό τον λόγο το κάθε Core δέχεται σαν είσοδο 2 σήματα τα οποία διορθώνουν το πρόβλημα που αναφέρθηκε. Τα σήματα αυτά δίνονται από το Top module της σχεδίασης μας και συγκεκριμένα από την FSM που έχει σχεδιαστεί για τον συγχρονισμό των 8 Cores. Τα σήματα αυτά είναι το offset το οποίο διορθώνει το πρόβλημα που αναφέραμε στην εκτέλεση όλων των συναρτήσεων (FirstStep,Step,AdjustWeights), το σήμα αυτό στην ουσία μεταφέρει την πληροφορία

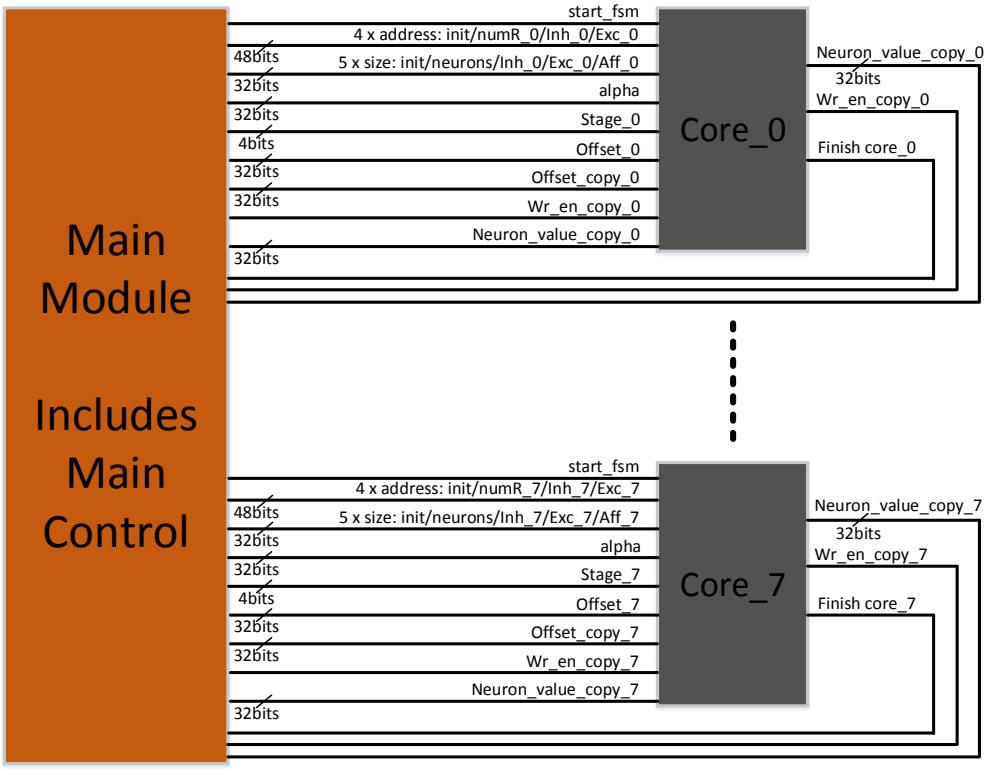
τις θέσης του κάθε Core. Παραδείγματος χάρη το Core_0 για κάθε δείκτη που δέχεται πρέπει να προσθέτει μέσω του offset 2 φορές το 1 / 8 του αριθμού των νευρώνων για να βρεθεί στο σωστό νευρώνα μέσα στην μνήμη Neurons. Αντίστοιχα στο τελευταίο Core το offset έχει αρνητικό πρόσημο και μέγεθος 5 φορες το 1 / 8 του αριθμού των νευρώνων για να λειτουργεί σωστά. Το δεύτερο σήμα που παίρνει σαν είσοδο το κάθε Core για την σωστή λειτουργία των μνήμων είναι το offset_copy. Το συγκεκριμένο σήμα αφορά την σωστή εγγραφή των νευρώνων στο στάδιο αντιγραφής από τις FIFO στις μνήμες Neurons μέσα σε κάθε Core. Στους παρακάτω πίνακες 4.2,4.3 φαίνονται οι τιμές που έχουν αυτά τα δυο σήματα για κάθε Core. Αν και το offset παραμένει σταθερό σε όλη την διάρκεια εκτέλεσης το offset_copy αλλάζει με βάση το κάθε στάδιο που είμαστε κάθε φορά στην αντιγραφή των νέων τιμών των νευρώνων.

| cores | offset |
|---|---|
| Core_0 | $2 \times$ size |
| Core_1 | size |
| Core_2 | 0 |
| Core_3 | -size |
| Core_4 | $-2 \times$ size |
| Core_5 | $-3 \times$ size |
| Core_6 | $-4 \times$ size |
| Core_7 | $-5 \times$ size |

Πίνακας 4.2: Τιμές του σήματος offset για κάθε ένα από τα οκτώ Core

| offset_copy | | | | | |
|---|---|---|---|---|---|
| | stage 1 | stage 2 | stage 3 | stage 4 | stage 5 |
| Core_0 | $2 \times$ size | $3 \times$ size | $4 \times$ size | – | – |
| Core_1 | size | $2 \times$ size | $3 \times$ size | $4 \times$ size | – |
| Core_2 | 0 | size | $2 \times$ size | $3 \times$ size | $4 \times$ size |
| Core_3 | $4 \times$ size | 0 | size | $2 \times$ size | $3 \times$ size |
| Core_4 | $3 \times$ size | $4 \times$ size | 0 | size | $2 \times$ size |
| Core_5 | $2 \times$ size | $3 \times$ size | $4 \times$ size | 0 | size |
| Core_6 | size | $2 \times$ size | $3 \times$ size | – | 0 |
| Core_7 | 0 | size | $2 \times$ size | – | – |

Πίνακας 4.3: Τιμές του σήματος offset_copy για κάθε ένα από τα οκτώ Core σε κάθε στάδιο της αντιγραφής




Σχήμα 4.8: **Σχεδίαση του Top module των 8 Cores.**

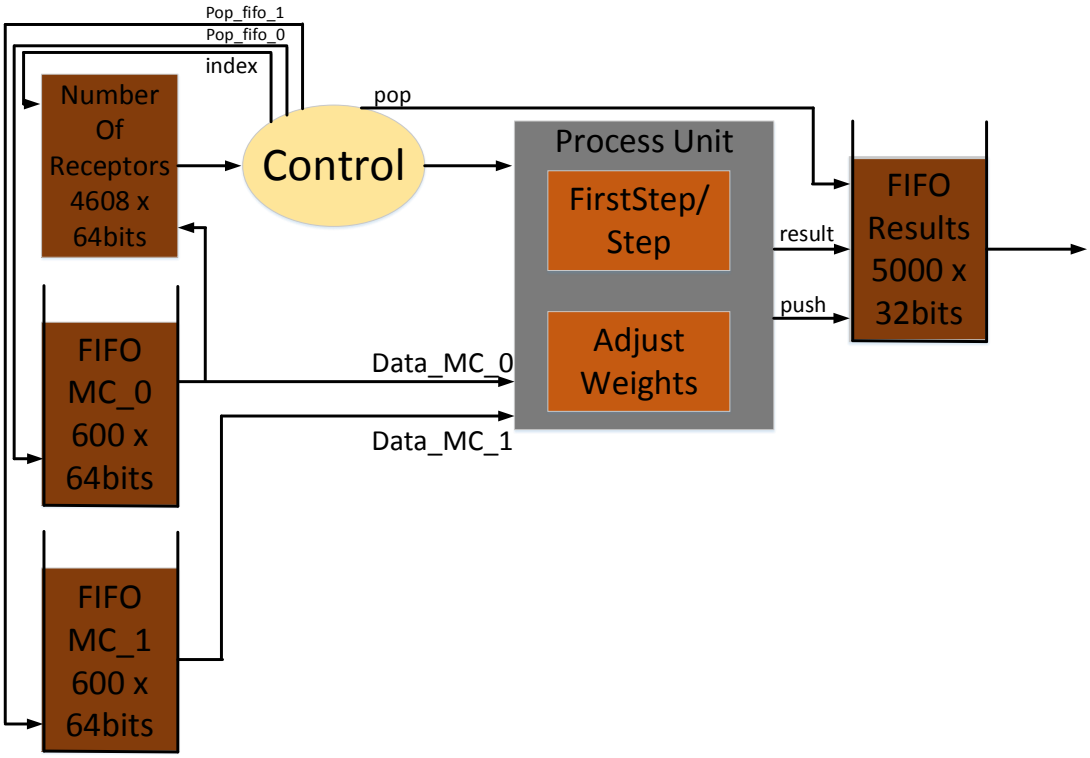
Στο σχήμα 4.8 φαίνεται η επικοινωνία του Top module στο οποίο λειτουργεί και το Main Control με τα 8 Cores. Το Top module δέχεται τις διευθύνσεις μνήμης όλων των πινάκων που πρόκειται να χρησιμοποιηθούν από την σχεδίαση μας και τα αντίστοιχα μεγέθη τους. Δουλειά του είναι να διαχωρίσει αυτούς τους πίνακες όπως επίσης και τα μεγέθη τους σε 8 κομμάτια και να στείλει σε κάθε Core ξεχωριστά τις σωστές αρχικές διευθύνσεις μνήμης και τον αριθμό των στοιχείων που πρόκειται να διαβάσουν στην συνέχεια της εκτέλεσης του αλγορίθμου από την κοινή μνήμη του συστήματος. Επίσης μέσω του Main Control ελέγχει την σωστή και συγχρονισμένη λειτουργία και των 8 Cores. Συγκεκριμένα μέσω των σημάτων Start_fms, Stage που στέλνονται στα Cores όπως επίσης και το σήμα Finish_Core που δέχεται από κάθε ξεχωριστό Core συγχρονίζεται η εκτέλεση και των 8 στοιχείων ώστε να περιμένουν αν κάποια τελειώσουν πιο γρήγορα από τα υπόλοιπα και να βρίσκονται στο ίδιο στάδιο του αλγορίθμου κάθε φορά.

Το κάθε Core εσωτερικά μοιάζει σε μεγάλο βαθμό με το Top module της αρχικής σχεδίασης. Μια από τις σημαντικές διαφορές του είναι η προσθήκη μίας μνήμης η οποία γεμίζει στην αρχή της εκτέλεσης του αλγορίθμου και συγκεκριμένα στο στάδιο INIT_NUMR με τον αριθμό των συνδέσεων για κάθε νευρώνα σε κάθε προβολή του. Στην αρχική σχεδίαση ο πίνακας αυτός έπρεπε να διαβαστεί σε κάθε επανάληψη από την κοινή μνήμη του συστήματος. Εδώ αυτό συμβαίνει μόνο μια φορά στην αρχή της εκτέλεσης για εξικονόμηση χρόνου και κυρίως για να αποδεσμευτεί ένας από τους ελεγκτές μνήμης και να χρησιμοποιούμε μόνο 2 σε όλη την διάρκεια εκτέλεσης του αλγορίθμου. Αυτές οι μνήμες μεγέθους $4608 \times 64$ bits βρίσκονται εσωτερικά σε κάθε ένα από τα Cores και απασχολούν μόνο τους νευρώνες που ασχολείται το συγκεκριμένο Core. Για την αρχικοποίηση τους χρησιμοποιούμε μόνο ένα ελεγκτή μνήμης αφού ο συγκεκριμένος πίνακας έχει μορφοποιηθεί έτσι στο Software ώστε από 3 διαφορετικούς πίνακες να γίνει

μόνο ένας. Αφού ελέγχθηκε πως 16 bits είναι αρκετά για να αποθηκευτεί ο αριθμός των συνδέσεων για κάθε νευρώνα σε κάθε προβολή δημιουργήθηκε ένας νέος πίνακας που περιέχει και τις 3 πληροφορίες σε κάθε στοιχείο του για την εξοικονόμηση χρόνου κατά την διάρκεια της αρχικοποίησης. Στον παρακάτω πίνακα 4.4 φαίνεται πως έχουν μοιραστεί τα 64 bits του κάθε στοιχείου από τον πίνακα των συνδέσεων.

| 63-48 | 47-32 | 31-16 | 15-0 |
|---|---|---|---|
| zeros | aff | exc | inh |

Πίνακας 4.4: Κατανομή bits του κάθε στοιχείου στον πίνακα των συνδέσεων.




Σχήμα 4.9: **Σχεδίαση αρχιτεκτονικής της μονάδας core.**

Όπως φαίνεται και στο σχήμα 4.9 τα σήματα δεν έχουν αλλάξει σε μεγάλο βαθμό στην συγκεκριμένη σχεδίαση. Πάλι η λειτουργία του συγκεκριμένου module είναι να τροφοδοτεί το Process Unit με τα κατάλληλα δεδομένα για τον υπολογισμό των νέων τιμών των νευρώνων και των βαρών αντίστοιχα, διαβάζοντας από τις σωστές θέσεις τις κοινής μνήμης. Επίσης στα στάδια του αλγορίθμου που υπολογίζουμε τις νέες τιμές των βαρών για την κάθε σύνδεση η σχεδίαση αυτή αναλαμβάνει να γράψει τις τιμές αυτές πίσω στην κοινή μνήμη. Λόγο των αρκετών σταδίων που περνά ο αλγόριθμος τα δεδομένα που διαβάζονται από την κοινή μνήμη και δίνονται στην υπολογιστική σχεδίαση περιέχουν διαφορετικές πληροφορίες σε κάθε στάδιο εκτέλεσης του αλγορίθμου. Στον πίνακα 4.5 φαίνονται τα διαφορετικά στάδια και τα αντίστοιχα δεδομένα που έρχονται στην σχεδίαση

μας.

| | data mc 0 | data mc 1 |
|---|---|---|
| init numR | αριθμός συνδέσεων | – |
| init first | είσοδος | – |
| FirstStep | βάρη εισόδου | – |
| Step | ανασταλτικά βάρη | διεγερτικά βάρη |
| Adjust | βάρη της κάθε προβολής | – |
| init aff | είσοδος | – |

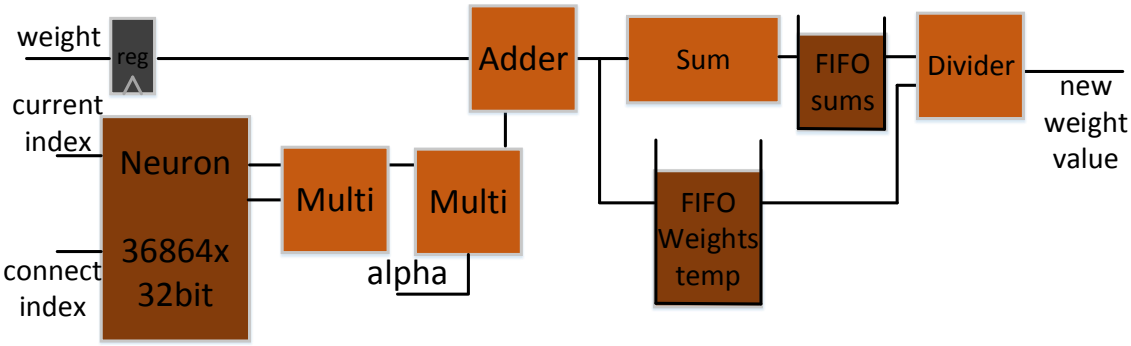
Πίνακας 4.5: Τιμές των σημάτων data mc 0 και data mc 1 για κάθε στάδιο της εκτέλεσης του αλγορίθμου

Στην τελική σχεδίαση όπως έχουμε αναφέρει ξανά δυο ακόμη συναρτήσεις η FirstStep και η AdjustWeights συμπεριλήφθηκαν μέσα στην σχεδιασή μας. Για την συνάρτηση FirstStep δεν χρειάστηκε να γίνει κάποια αλλαγή στην ήδη υπάρχον σχεδίαση γιατί ο τρόπος με τον οποίο υπολογίζονται οι πρώτες ενεργοποιήσεις των νευρώνων από την είσοδο είναι ίδιος με τον τρόπο που υπολογίζουμε και τις νέες τιμές των νευρώνων στην συνάρτηση Step. Η μόνη διαφορά που υπάρχει είναι οι διαφορετικές τιμές που έχει ο πίνακας Neurons αφού περιέχει την είσοδο κάθε φορά και επίσης οτι διαβάζουμε διαφορετικό πίνακα από την κοινή μνήμη αφού χρειαζόμαστε τον πίνακα με τις εισάγων συνδέσεις και τα αντίστοιχα βάρη τους για κάθε νευρώνα.

Η υλοποίηση της συνάρτησης AdjustWeights έρχεται μέσα από μια νέα σχεδίαση που παρουσιάζεται στο σχήμα 4.10. Η σχεδίαση αυτή μπορεί να δεχτεί ένα νέο σετ δεδομένων κάθε κύκλο και αντίστοιχα να παράγει μια νέα τιμή βάρους αντίστοιχα. Δέχεται σαν είσοδο το βάρος της σύνδεσης που μελετάμε κάθε φορά όπως επίσης και 2 δείκτες. Ο πρώτος από τους δύο δείκτες υποδηλώνει τον νευρώνα του οποίου οι τιμές των βαρών για τις συνδέσεις του πρόκειται να αλλάξουν. Ο δεύτερος δείκτης δείχνει τον νευρώνα ο οποίος επηρεάζει τον νευρώνα του οποίου μελετάμε τις συνδέσεις (1ος δείκτης). Για τον υπολογισμού των νέων βαρών για τις συνδέσεις του νευρώνα που μελετάμε αρχικά πρέπει να πολλαπλασιάσουμε τις τιμές των 2 νευρώνων που αλληλεπιδρούν και στην συνέχεια το γινόμενο αυτό να πολλαπλασιαστεί με τον ρυθμό μάθησης του δικτύου η οποία είναι διαφορετική για κάθε μια από τις 3 προβολές. Στην συνέχεια το νέο αυτό γινόμενο προστίθεται με την τιμή του βάρους για την συγκεκριμένη σύνδεση η οποία έχει καθυστερήσει όσο χρειάζεται για να φτάσει μαζί με το αποτέλεσμα του 2ου γινομένου. Το αποτέλεσμα του αθροίσματος αυτού αποθηκεύεται στην FIFO Weights_temp και επίσης δίνεται στο module Sum το οποίο θα αθροίσει τις τιμές για όλες τις συνδέσεις που αφορούν το νευρώνα που ασχολούμαστε κάθε φορά. Όταν το αποτέλεσμα του αθροίσματος από το module Sum είναι έτοιμο η FIFO Weights_temp θα μας δώσει μια μια τις τιμές που αφορούν τις συνδέσεις του νευρώνα που ασχολούμαστε οι οποίες θα διαιρεθουν στην συνέχεια με το άθροισμα που υπολογίστηκε. Η διαδικασία αυτή συμβαίνει γιατί θέλουμε το άθροισμα των βαρών για όλες τις συνδέσεις του κάθε νευρώνα να είναι ίσο με 1 σε κάθε στάδιο της εκτέλεσης του αλγορίθμου. Το αποτέλεσμα της διαίρεσης αυτής είναι η νέα τιμή του βάρους για την κάθε συνδέση του νευρώνα και θα γραφτεί στην συνέχεια από τον 2ο ελεγκτή μνήμης στην σωστή θέση του πίνακα στην κοινή μνήμη από όπου και διαβάστηκε. Άρα για το στάδιο της εκτέλεσης της συνάρτησης AdjustWeights ο πρώτος ελεγκτής μνήμης διαβάζει της μια μια τις συνδέσεις των νευρώνων και αντίστοιχα ο 2ος ελεγκτης όταν μια νέα τιμή του βάρους για κάποια σύνδεση είναι έτοιμη την γράφει πίσω

στην αντίστοιχη θέση μνήμης.

Η FIFO Sums που έχει υλοποιηθεί στην σχεδίαση έχει μέγεθος $32 \times 64$ bits και σκοπός της είναι να αποθηκεύει το αποτέλεσμα του αθροίσματος ώστε να μην χαθεί στην περίπτωση που ο υπολογισμός των συνδέσεων του προηγούμενου νευρώνα δεν έχει ολοκληρωθεί ακόμη. Αυτό συμβαίνει γιατί ο αριθμός των συνδέσεων για κάθε νευρώνα δεν είναι ίδιος και στην περίπτωση που ο προηγούμενος νευρώνας, του οποίου τα νέα βάρη των συνδέσεων υπολογίζεται, έχει μεγαλύτερο αριθμό συνδέσεων τότε το άθροισμα των βαρών για τον επόμενο νευρώνα μπορεί να είναι έτοιμο πριν τελειώσει ο υπολογισμός των νέων βαρών του προηγούμενου. Η FIFO AdjustWeights έχει μέγεθος $8192 \times 32$ bits και κρατά τις τιμές των βαρών για την κάθε σύνδεση που στην συνέχεια θα κανονικοποιήθουν ώστε να μην χρειάζεται να υπολογιστούν ξανά ή να σταματά η εισαγωγή νέων δεδομένων μέχρι να υπολογιστεί το τελικό άθροισμα όλων των συνδέσεων. Με αυτόν τον τρόπο και στέλνοντας απλά τον αριθμό των συνδέσεων για κάθε νευρώνα μπορούμε αφού υπολογίσουμε το άθροισμα να διαβάσουμε τον αντίστοιχο αριθμό στοιχείων από την FIFO AdjustWeights για να υπολογιστούν οι νέες τιμές. Τέλος ο λόγος που η FIFO Sums έχει πλάτος 64 bits είναι γιατί εκτός από το τελικό άθροισμα κρατάει και τον αριθμό των συνδέσεων ώστε να γνωρίζουμε πόσα στοιχεία πρέπει να διαβαστούν στην συνέχεια.




Σχήμα 4.10: **Αρχιτεκτονική της συνάρτησης AdjustWeights**

# Κεφάλαιο 5

# Αποτελέσματα

Στο παρόν κεφάλαιο θα παρουσιαστούν τα αποτελέσματα της απόδοσης της αρχιτεκτονικής που σχεδιάστηκε στο Convey συγκριτικά με παρόμοιες προσομοιώσεις του αλγορίθμου LISSOM σε κάρτες γραφικών. Επίσης γίνεται μια αναφορά σχετικά με τα χαρακτηριστικά των διαφορετικών συνόλων δεδομένων που έχουν δημιουργηθεί για την εκτέλεση του αλγορίθμου.

## 5.1 Εγκατάσταση Αρχιτεκτονικής και Κατανάλωση πόρων

Η αρχιτεκτονική που υλοποιήθηκε και προσομοιώνει τον αλγόριθμο LISSOM λόγο του μεγάλου μεγέθους των δεδομένων για τις συνδέσεις των τριών προβολών αποτυπώθηκε σε έναν υπερ-υπολογιστή βασισμένο σε αναδιατασσόμενη λογική. Συγκεκριμένα χρησιμοποιήσαμε την πλατφόρμα της Convey HC-2, η οποία περιέχει 4 Virtex-6 LX760 FPGA. Όλες οι FPGA μπορούν να χρησιμοποιήσουν την ίδια μνήμη και έναν συνεπεξεργαστή. Επίσης μέσω 16 ελεγκτών μνήμης η σχεδίαση μας μπορεί να διαβάσει και να γράψει στην κοινή μνήμη. Στην συγκεκριμένη υλοποίηση χρησιμοποιούμε μόνο μια FPGA και όλους τους ελεγκτές μνήμης για διάβασμα δεδομένων αλλά και γράψιμο όταν εκτελείται η συνάρτηση AdjustWeights.

Για την ανάπτυξη της αρχιτεκτονικής χρησιμοποιήσαμε το εργαλείο της Xilinx ISE Design 12.4. Η σχεδίαση των μονάδων έγινε σε γλώσσα περιγραφής υλικού VHDL αλλά και Verilog. Η κάθε μονάδα σχεδιάστηκε και ελέγχθηκε ξεχωριστά μέσω του εργαλείου ISE Modelsim 12.4 για την σωστή λειτουργία της. Στην συνέχεια αφού ενώθηκαν όλες οι μονάδες από τις οποίες αποτελείται η τελική αρχιτεκτονική προσομοιώθηκε όλο το σύστημα και έγιναν οι αντίστοιχοι έλεγχοι ώστε να επιβεβαιωθεί η σωστή λειτουργία της αρχιτεκτονικής. Για την σύνδεση της πλατφόρμας HC-2 του Convey με τον συνεπεξεργαστή υλοποιήθηκε κώδικας σε γλώσσα C και Assembly ώστε να περάσουν στην σχεδίαση μας οι θέσεις μνήμης των πινάκων που μας αφορούσε να έχει πρόσβαση και διάφορες βοηθητικές τιμές. Στον παρακάτω πίνακα παρουσιάζεται η κατανάλωση πόρων ολόκληρης της σχεδίασης ύστερα από την διαδικασία Synthesis και Place and Route στην Virtex-6 LX760 FPGA που περιέχεται στην πλατφόρμα του Convey. Επίσης αναφέρεται και η συχνότητα του ρολογιού για την συγκεκριμένη σχεδίαση.

Στα συνολικά ποσοστά πρέπει να διευκρινιστεί ότι ένα ποσοστό είναι εξαιτίας της πλατφόρμας της Convey. Ο λόγος είναι ότι η πλατφόρμα αξιοποιεί ένα ποσοστό λογικής και BRAMs για την υλοποίηση της διεπαφής των ελεγκτών μνήμης. Από τα ποσοστά που φαίνονται και στον πίνακα 5.1 το περιοριστικό στοιχείο της σχεδίασης είναι ο αριθμός των BRAMs της FPGA το οποίο περιορίζει και το μέγιστο μέγεθος του δικτύου που

| | Virtex-6 LX760 FPGA |
|---|---|
| Number of occupied Slices | 42% |
| Number of BRAM/FIFO | 88% |
| Συχνότητα ρολογιού | 150Mhz |

Πίνακας 5.1: Συνολικοί πόροι σχεδίασης και συχνότητα ρολογιού μετά την εγκατάσταση στο Convey

μπορεί να προσομοιώσει η συγκεκριμένη σχεδίαση (192 x 192 νευρώνες).

## 5.2 Πειραματικά Αποτελέσματα

Μέσω του συστήματος που υλοποιήθηκε η σχεδίαση μας στο Hardware διαβάζει τα δεδομένα εισόδου και τις αρχικές καταστάσεις των βαρών για κάθε σύνδεση από την κοινή μνήμη του συστήματος. Σε κάθε εκτέλεση μίας επανάληψης από τις συναρτήσεις που εκτελούνται στο FPGA οι πίνακες με τα βάρη των συνδέσεων γράφονται ξανά πίσω στην κοινή μνήμη του συστήματος. Αφού φτάσουμε στις επιθυμητές επαναλήψεις (συνήθως 20000) τα στοιχεία των πινάκων με τις τιμές των βαρών για κάθε σύνδεση και για τις τρεις προβολές εξάγονται προς τον χρήστη μέσω κάποιον αρχείων txt ώστε να ενημερωθεί για τα πειραματικά αποτελέσματα της προσομοίωσης. Για καλύτερη οπτικοποίηση των αποτελεσμάτων ο κώδικας στην CUDA περιέχει κάποιες συναρτήσεις που εκτυπώνουν την δεδομένη είσοδο και αντίστοιχα την ενεργοποίηση των νευρώνων του δικτύου για την συγκεκριμένη είσοδο κάθε φορά. Αυτό που αλλάζει είναι η συμπεριφορά των νευρώνων στην κάθε είσοδο με βάση την μάθηση που έχει γίνει πάνω στις συνδέσεις των βαρών μέχρι εκείνη την στιγμή. Δηλαδή το δίκτυο θα ανταποκριθεί διαφορετικά στην αρχή των επαναλήψεων σε σχέση με την ολοκλήρωση της μάθησης. Οι συναρτήσεις αυτές δεν υλοποιήθηκαν στην σχεδίαση μας αλλά από την στιγμή που μπορούμε να πάρουμε την κατάσταση του δικτύου σε οποιοδήποτε σημείο της μάθησης που επιθυμούμε απλά μεταφέρουμε την απαραίτητη πληροφορία και εκτελούμε τις συναρτήσεις. Εικόνες μέσω αυτής της διαδικασίας παρατίθενται στις επόμενες σελίδες.

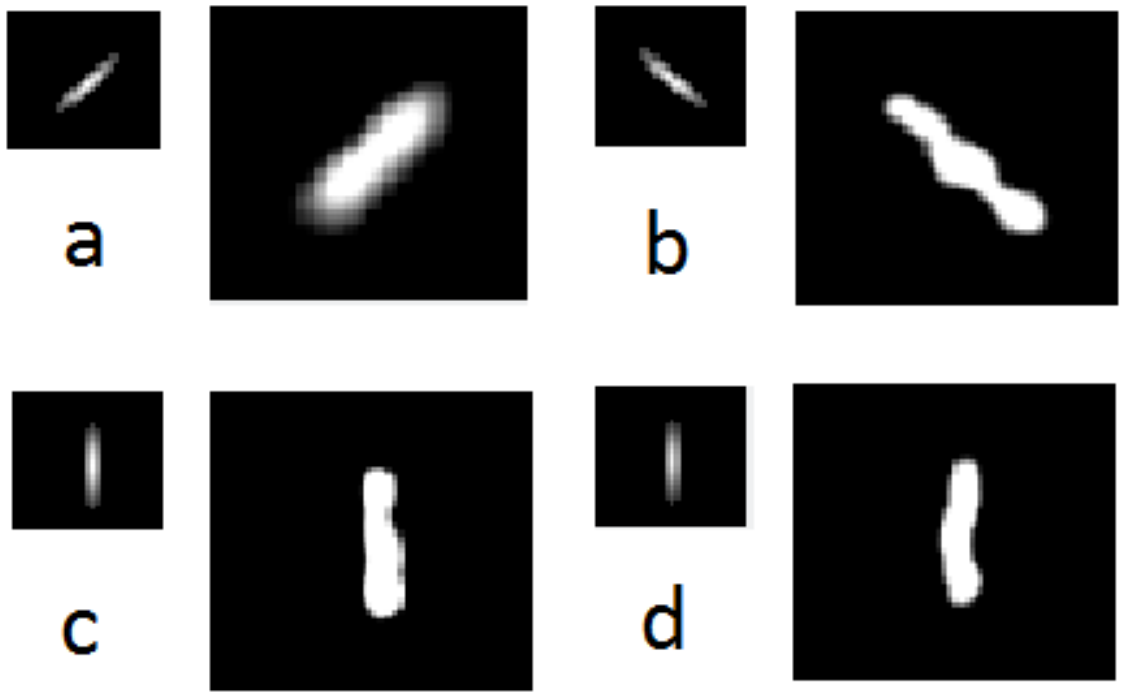
Για τις πειραματικές μετρήσεις και προσομοιώσεις ακολουθήσαμε κάποιους κανόνες που ακολουθούνται επίσης και στην σχεδίαση του αλγορίθμου σε CUDA και έχουν την βάση τους σε διάφορες δημοσιεύσεις για τον αλγόριθμο LISSOM [13]. Αρχικά για την κλιμάκωση του δικτύου των νευρώνων σε σχέση με την είσοδο ακολουθήσαμε την λογική που χρησιμοποιούν σε αντίστοιχες πειραματικές μετρήσεις δηλαδή το μέγεθος της εισόδου που αναπαριστά το δίκτυο των νευρώνων του αμφιβληστροειδή (R) και το μέγεθος του δικτύου νευρώνων της περιοχής V1 (N) πρέπει να αυξηθούν κατά το ίδιο ποσοστό. Επίσης από την στιγμή που αλλάζουμε το μέγεθος της περιοχής της εισόδου για να μην διαταραχθεί η λειτουργική συμπεριφορά του δικτύου πρέπει να αλλάξει ανάλογα και το παράθυρο (r Aff) μέσα στο οποίο ο κάθε νευρώνας του δικτύου συνδέεται με στοιχεία τις εισόδου. Τέλος αλλαγή πρέπει να γίνει και στα αντίστοιχα παράθυρα των πλευρικών συνδέσεων, ανασταλτικών (r Inh) και διεγερτικών (r Exc), τα οποία πρέπει να μεταβληθούν για τον ίδιο παράγοντα όπως και το μέγεθος του δικτύου των νευρώνων. Στον πίνακα 5.2 φαίνονται τα χαρακτηριστικά των διαφορετικών συνόλων δεδομένων που δημιουργήθηκαν, βάση αυτών που αναφέρθηκαν προηγουμένως, και πάνω στα οποία έγιναν οι προσομοιώσεις.

Στην συνέχεια παρουσιάζονται οι εικόνες από τις καταστάσεις των δικτύων για δια-

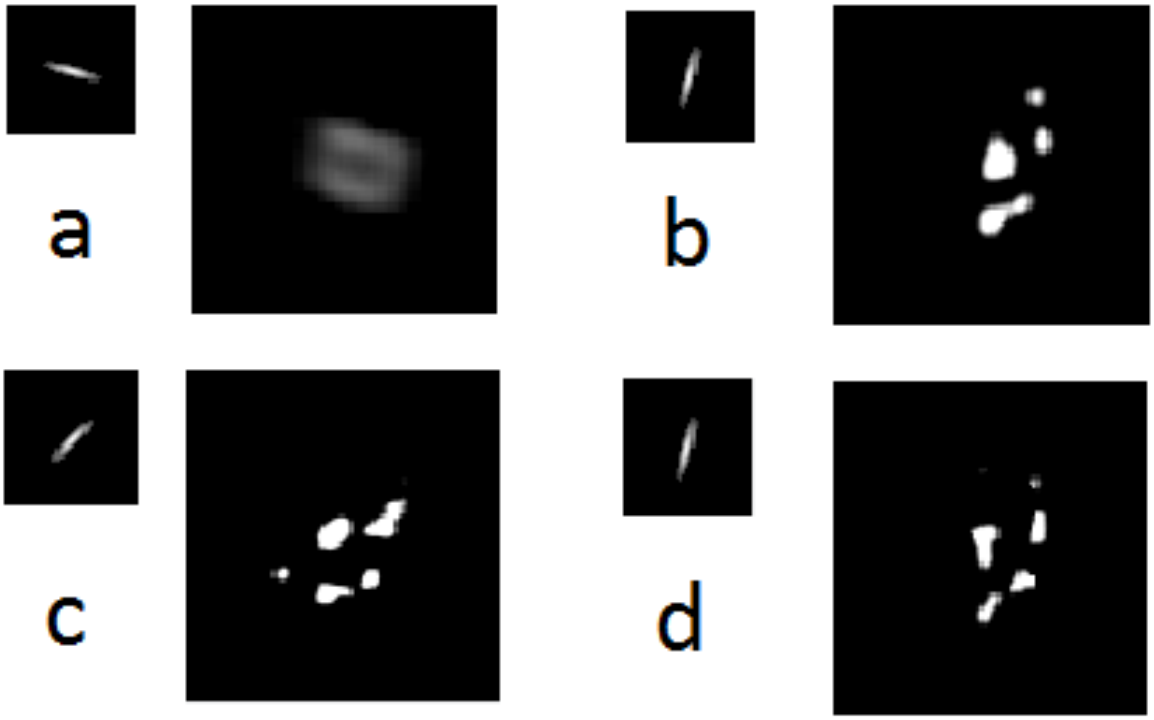
| Neuron | Retina | r Aff | r Inh | r Exc |
|---|---|---|---|---|
| 192 | 24 | 11 | 37 | 19 |
| 160 | 20 | 9,16 | 30,84 | 15,84 |
| 128 | 16 | 7,33 | 24,67 | 12,67 |
| 96 | 12 | 5,5 | 18,5 | 9,5 |
| 64 | 8 | 3,66 | 12,34 | 6,34 |
| 32 | 4 | 1,83 | 6,17 | 3,17 |

Πίνακας 5.2: Χαρακτηριστικά δικτύου για διάφορα μεγέθη του

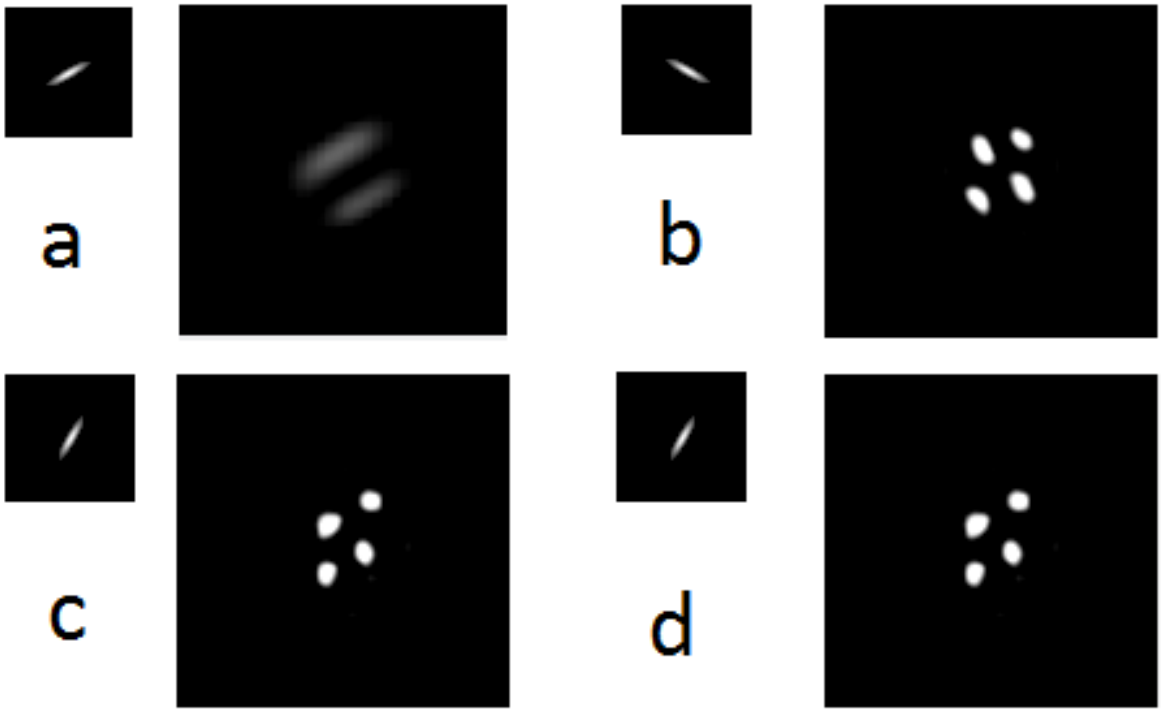
φορετικά σύνολα δεδομένων σε διαφορετικά σημεία της μάθησης του δικτύου.




Σχήμα 5.1: **Δίκτυο νευρώνων 64x64**




Σχήμα 5.2: **Δίκτυο νευρώνων 92x92**




Σχήμα 5.3: **Δίκτυο νευρώνων 128x128**

Στις εικόνες 5.1,5.2,5.3 παρουσιάζεται η κατάσταση του δικτύου των νευρώνων για 4 διαφορετικούς χρόνους μάθησης και μια τυχαία είσοδο κάθε φορά. Συγκεκριμένα το (a) αναπαριστά την απόκριση του δικτύου που μόλις έχει αρχικοποιηθεί στην είσοδο. Αντίστοιχα το (b) μετά από 5000 επαναλήψεις το (c) μετά από 10000 επαναλήψεις και το (d) την τελική απόκριση του δικτύου σε μια τυχαία είσοδο μετά από 20000 επαναλήψεις του αλγορίθμου. Επίσης οι τρεις διαφορετικές εικόνες 5.1,5.2,5.3 διαφοροποιούνται στο μέγεθος του δικτύου των νευρώνων. Ακόμη παρατηρούμε πως όσο περισσότερες επαναλήψεις έχουν γίνει το δίκτυο μπορεί να ανταποκριθεί σε ένα πιο "καθαρό" αποτέλεσμα για κάθε είσοδο που λαμβάνει. Στις εικόνες 5.2 και 5.3 αντίστοιχα η δημιουργία των κενών ανάμεσα στις φυσαλίδες που έχουν δημιουργηθεί γίνονται γιατί μετά από κάποιες επαναλήψεις οι νευρώνες αποκτούν σιγά σιγά προσανατολισμό και σκοτώνουν συνδέσεις που δεν ανταποκρίνονται σε αυτόν.

## 5.3 Σύγκριση Απόδοσης

Τελικό στάδιο της εργασίας ήταν η σύγκριση των επιδόσεων της σχεδίασης μας που υλοποιήθηκε σε Hardware σε σχέση με παρόμοιες προσομοιώσεις υλοποιημένες σε GPU. Η προσομοίωση της σχεδίασης μας έγινε στον υπέρ-υπολογιστή της Convey με χρήση μιας FPGA για διαφορετικά μεγέθη δικτύου από νευρώνες. Τα αποτελέσματα του χρόνου εκτέλεσης στο Hardware συγκρίθηκαν με τους χρόνους εκτέλεσης σε δυο διαφορετικές κάρτες γραφικών της NVIDIA.

Στον παρακάτω πίνακα φαίνονται τα αποτελέσματα για την εκτέλεση του αλγορίθμου στο σύστημα της Convey για διαφορετικά μεγέθη δικτύου κάθε φορά.

| $N \times N$ | 32 | 64 | 96 | 128 | 160 | 192 |
|---|---|---|---|---|---|---|
| Convey (seconds) | 21,4 | 304 | 1550 | 4766 | 11706 | 25568 |

Πίνακας 5.3: Χρόνοι εκτέλεσης της σχεδίασης για διαφορετικά μεγέθη δικτύου στο Convey

Αντίστοιχα μια προσομοίωση του ίδιου αλγορίθμου αυτή την φορά σε μια συμβατική κάρτα γραφικών την GT720M η οποία έχει μνήμη 1024MB DDR3 και 192 πυρήνες CUDA στα 797Mhz. Στην συγκεκριμένη κάρτα γραφικών δεν μπορέσαμε να πάρουμε την μέτρηση για $192 \times 192$ δίκτυο νεύρωνων γιατί δεν μπορούσαν τα δεδομένα να αποθηκευτούν στην κάρτα γραφικών. Τα αποτελέσματα της οποίας παρατίθενται στον πίνακα 5.4.

| $N \times N$ | 32 | 64 | 96 | 128 | 160 | 192 |
|---|---|---|---|---|---|---|
| GT 720M (seconds) | 100 | 610 | 2760 | 8200 | 21174 | – |

Πίνακας 5.4: Χρόνοι εκτέλεσης της σχεδίασης για διαφορετικά μεγέθη δικτύου στην κάρτα γραφικών GT720M

Επίσης ο ίδιος ακριβώς αλγόριθμος εκτελέστηκε και σε μια δεύτερη κάρτα γραφικών η οποία βρίσκεται στον Indefix Server του Πολυτεχνείου Κρήτης. Πρόκειται για μια GTX 980 High End κάρτα γραφικών με μνήμη 4GB GDDR5 και 2048 πυρήνες CUDA

στα 1216Mhz. Αντίστοιχα τα αποτελέσματα του χρόνου εκτέλεσης για τα ίδια μεγέθη δικτύου φαίνονται στον πίνακα 5.5

| $N \times N$ | 32 | 64 | 96 | 128 | 160 | 192 |
|---|---|---|---|---|---|---|
| GTX 980 (seconds) | 11,6 | 33,8 | 68,4 | 400 | 968 | 2170 |

Πίνακας 5.5: Χρόνοι εκτέλεσης της σχεδίασης για διαφορετικά μεγέθη δικτύου στην κάρτα γραφικών GTX980

Τέλος στον πίνακα 5.6 φαίνονται οι χρόνοι εκτέλεσης του αλγορίθμου LISSOM στο Software. Για τον υπολογισμό των συγκεκριμένων χρόνων βασιστήκαμε στην αναφορά πως το 2008 ο συγκεκριμένος αλγόριθμος μπορούσε να εκτελεστεί μέχρι και 9 φορές πιο γρήγορα στην GPU συγκριτικά με τους χρόνους εκτέλεσης στο Software [27]. Με βάση αυτό και την υπόθεση πως η ταχύτητα του Software από το 2008 διπλασιάστηκε, όπως επίσης την σύγκριση των GPUs που χρησιμοποιήθηκαν τότε (GTX 280) και τώρα (GTX 980) υπολογίζουμε τις τιμές.

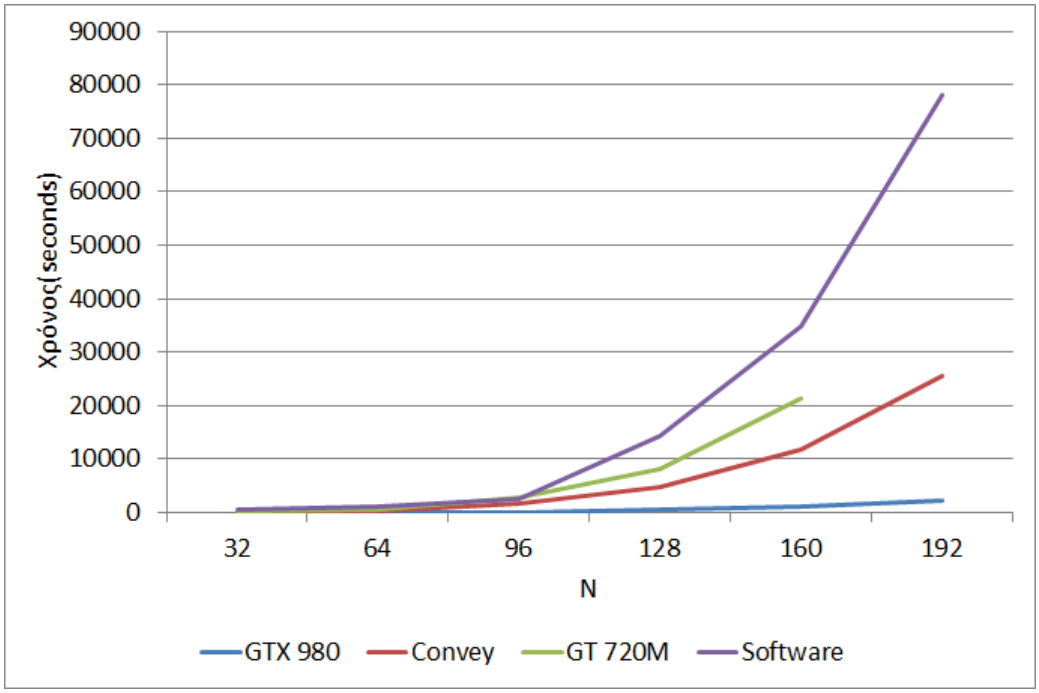
| $N \times N$ | 32 | 64 | 96 | 128 | 160 | 192 |
|---|---|---|---|---|---|---|
| Software (seconds) | 417,6 | 1216,8 | 2462,4 | 14400 | 34848 | 78120 |

Πίνακας 5.6: Χρόνοι εκτέλεσης της σχεδίασης για διαφορετικά μεγέθη δικτύου στο Software

Στον πίνακα 5.7 φαίνονται τα μεγέθη των δεδομένων που χρησιμοποιούνται από το σύστημα για μία εκτέλεση του αλγορίθμου για τις διάφορες προσομοιώσεις που έγιναν. Τα δεδομένα αυτά αφορούν όλους τους πίνακες που χρησιμοποιεί ο αλγόριθμος για να περιγράψει το δίκτυο νευρώνων, τις συνδέσεις και τα βάρη μεταξύ τους όπως επίσης και την είσοδο για κάθε επανάληψη.

| $N \times N$ | 32 | 64 | 96 | 128 | 160 | 192 |
|---|---|---|---|---|---|---|
| Data size (MB) | 2 | 28 | 154 | 459 | 1225 | 2751 |

Πίνακας 5.7: Μέγεθος του συνόλου δεδομένων για διαφορετικά μεγέθη του δικτύου




Σχήμα 5.4: **Κοινό διάγραμμα χρόνου εκτέλεσης του μοντέλου για διαφορετικό μέγεθος δίκτυου και για τα τέσσερα πειραματικά αποτελέσματα.**

Στο παραπάνω σχήμα παρουσιάζονται γραφικά τα πειραματικά αποτελέσματα του χρόνου εκτέλεσης που παρουσιάστηκαν παραπάνω. Παρατηρείται μια εκθετική αύξηση του χρόνου εκτέλεσης και για τα τέσσερα αποτελέσματα. Κάτι τέτοιο είναι φυσικό αφού όπως φαίνεται και στον πίνακα 5.1 εκτός από την αύξηση του δικτύου ταυτόχρονα υπάρχει και αύξηση στον αριθμό των συνδέσεων σε όλες τις προβολές για κάθε νευρώνα το οποίο προσθέτει αρκετό χρόνο εκτέλεσης. Επίσης φαίνεται πως η σχεδίαση μας καταφέρνει να τρέξει περίπου 2 φορές πιο γρήγορα από την συμβατική κάρτα γραφικών (GT720M) και μάλιστα μπορεί να εκτελέσει προσομοιώσεις για μεγαλύτερο μέγεθος δικτύου. Αντίστοιχα η σχεδίαση μας συγκριτικά με αυτή στο Software, έχει 3 φορές γρηγορότερο χρόνο εκτέλεσης για τα ίδια μεγέθη δικτύου. Τέλος συγκριτικά με την HIGH END κάρτα γραφικών (GTX 980) η σχεδίαση μας τρέχει αρκετά πιο αργά, περίπου 10-12 φορές, ειδικά όσο το μέγεθος του δικτύου μεγαλώνει.

# Κεφάλαιο 6

# Συμπεράσματα και Μελλοντική Εργασία

## 6.1 Ανασκόπηση και Συμπεράσματα

Στην συγκεκριμένη διπλωματική εργασία μελετήσαμε τον αλγόριθμο LISSOM ο οποίος ασχολείται με την προσομοίωση βιολογικών νευρώνων στην περιοχή V1 του εγκεφάλου. Είναι ένας τομέας ξένος προς τον τομέα της αρχιτεκτονικής υπολογιστών με τη οποία ασχολούμαστε. Σκοπός μας ήταν να μελετήσουμε την απόδοση του αλγορίθμου ο οποίος έχει δημιουργηθεί στην γλώσσα προγραμματισμού CUDA και να προσπαθήσουμε να την βελτιώσουμε με την χρήση της αναδιατασσόμενης λογικής. Αφού βρήκαμε τις συναρτήσεις που καταλάμβαναν τον μεγαλύτερο χρόνο εκτέλεσης επικεντρωθήκαμε σε αυτές που μας ενδιέφεραν και συγκεκριμένα στις FirstStep, Step ,AdjustWeights. Επόμενο βήμα ήταν η σχεδίαση των συναρτήσεων αυτών στο Hardware με την χρήση των γλωσσών περιγραφής υλικού VHDL και Verilog. Αφού επιβεβαιώσαμε την σωστή λειτουργία της αρχιτεκτονικής για κάθε επιμέρους συνάρτηση μέσω εργαλείων προσομοίωσης που μας παρέχει η XILINX συνδέσαμε τις σχεδιάσεις αυτές και συγχρονίστηκαν ώστε να εκτελούν την αντίστοιχη λειτουργία του αλγορίθμου που μας ενδιέφερε. Η σωστή λειτουργία της σχεδίασης δεν θα ήταν εφικτή αν δεν μπορούσαμε να χρησιμοποιήσουμε το σύστημα της Convey το οποίο μας έδωσε την δυνατότητα να αποθηκεύουμε πολύ μεγάλου όγκου δεδομένα τα οποία μπορεί να διαβάσει η σχεδίαση μας στο Hardware αλλά δεν μπορούν να αποθηκευτούν στην περιορισμένη μνήμη που έχει ένα FPGA.

Από τα αποτελέσματα που παρουσιάστηκαν στο κεφάλαιο 5 βλέπουμε πως η σχεδίαση που υλοποιήθηκε μπορεί να τρέξει μέχρι και 2 φορές γρηγορότερα από μια κοινή κάρτα γραφικών και 3 φορές γρηγορότερα από την αντίστοιχη υλοποίηση στο Software αλλά δεν καταφέρνει να φτάσει κοντά στην απόδοση μια πολύ καλής κάρτας γραφικών. Αυτό εξηγείται κυρίως λόγο της μεγάλης διαφοράς των υπολογιστικών μονάδων που έχουν μεταξύ τους τα διαφορετικά συστήματα. Η GTX980 έχει 2048 μονάδες επεξεργασίας το οποίο σημαίνει οτι μπορεί να σπάσει τον υπολογισμό των δεδομένων της κάθε συνάρτησης σε πάρα πολλά κομμάτια τα οποία θα εκτελεστούν πολύ πιο γρήγορα. Πάνω στην ίδια λογική διαχωρισμού των δεδομένων σε διαφορετικές υπολογιστικές μονάδες έχει βασιστεί και η σχεδίαση μας στο Convey με την διαφορά ότι περιέχει μόνο 8 μονάδες επεξεργασίας λόγο του περιορισμού των 16 ελεγκτών μνήμης του συστήματος του Convey. Βέβαια συγκριτικά με την δεύτερη κάρτα που πειραματιστήκαμε έχουμε καλύτερη απόδοση από μεριάς εκτέλεσης χρόνου, αν και πάλι η σχεδίαση μας έχει πολύ μικρότερο αριθμό υπολογιστικών μονάδων, και ταυτόχρονα μπορούμε να προσομοιώσουμε μεγαλύτερα δίκτυα νευρώνων τα οποία δεν είναι σε θέση να διαχειριστεί η δεύτερη κάρτα.

## 6.2 Μελλοντική Εργασία

Σε αυτή την ενότητα αναφερόμαστε στο πλάνο της μελλοντικής εργασίας που μπορεί να γίνει για να βελτιωθεί η απόδοση της σχεδίασης που δημιουργήθηκε για την προσομοίωση νευρωνικών δικτύων. Μια από τις πιο σημαντικές βελτιώσεις που μπορεί να γίνει στο σύστημα που υλοποιήθηκε είναι η αξιοποίηση και των τεσσάρων FPGA που μας παρέχονται από τον Convey. Με αυτόν τον τρόπο στην ουσία μπορούμε να αυξήσουμε τις υπολογιστικές μας μονάδες από τις 8 που έχουμε αυτή την στιγμή στις 32. Προφανώς θα πρέπει να υπάρξει κάποια επιπλέον λογική στην σχεδίαση ώστε οι 4 FPGA να επικοινωνούν μεταξύ τους ώστε να λειτουργούν συγχρονισμένα και για να μπορούν να μεταφέρουν δεδομένα από την μια στην άλλη όπως περίπου γίνεται και στο στάδιο αντιγραφής των νευρώνων από την μια μονάδα επεξεργασίας στην άλλη. Αυτό θα αυξήσει την πολυπλοκότητα της σχεδίασης αλλά ταυτόχρονα θα αυξηθεί η απόδοση της από μεριάς εκτέλεσης χρόνου. Επίσης θα μπορέσουμε να προσομοιώσουμε μεγαλύτερου μεγέθους νευρωνικά δίκτυα με την χρήση και των τεσσάρων FPGA αφού ένας από τους περιορισμούς που έχει η συγκεκριμένη σχεδίαση είναι η μεγάλη ανάγκη σε μνήμη για την αποθήκευση του δικτύου.

Μια ακόμη βελτιστοποίηση που θα μπορούσε να γίνει στην συγκεκριμένη σχεδίαση είναι η επεξεργασία των δεδομένων πριν διαβαστούν από το Hardware. Αυτή την στιγμή περιοριζόμαστε στις 8 επεξεργαστικές μονάδες λόγο των 16 ελεγκτών μνήμης που μας παρέχει το σύστημα της Convey. Αυτό συμβαίνει γιατί στο στάδιο Step χρειάζεται να διαβάζουμε ταυτόχρονα από 2 πίνακες της κοινής μνήμης και αντίστοιχα στο στάδιο AdjustWeights θέλουμε να διαβάζουμε και να γράφουμε την ίδια στιγμή. Αν με κάποιο έξυπνο τρόπο συμπιέσουμε τα δεδομένα που διαβάζουμε από την κοινή μνήμη, όπως έχει γίνει και στον πίνακα που κρατάει των αριθμό των συνδέσεων για κάθε νευρώνα και για τις τρεις προβολές, θα μπορούσαμε να αυξήσουμε την μεταφορά δεδομένων προς το Hardware σε μεγάλο βαθμό. Ειδικά στην περίπτωση εκτέλεσης της συνάρτησης Step αν καταφέρναμε να συμπιέσουμε τα δεδομένα των δυο πινάκων τόσο ώστε να χρησιμοποιούμε μόνο έναν ελεγκτή μνήμης θα μπορούσαμε να αυξήσουμε τις υπολογιστικές μονάδες τουλάχιστον για αυτό το στάδιο της εκτέλεσης.

# References


[1] **Hubel, D. H., and Wiesel, T. N. (1965).** Receptive fields and functional architecture in two nonstriate visual areas (18 and 19) of the cat, *Journal of Neurophysiology*, 28:229-89.

[2] **Miller, K. D., Keller, J. B., and Stryker, M. P. (1989).** Ocular dominance column development: Analysis and simulation, *Science*, 245, 605-615.

[3] **von der Malsburg, C. (1973).** Self-organization of orientation-sensitive cells in the striate cortex, *Kybernetik*, 14,85-100.

[4] **D. J. Willshaw; C. Von Der Malsburg (1976).** How Patterned Neural Connections Can Be Set Up by Self-Organization, *Proceedings of the Royal Society of London. Series B, Biological Sciences*, Vol. 194, No. 1117, pp. 431-445.

[5] **Katz, L. C., and Callaway, E. M. (1992).** Development of local circuits in mammalian visual cortex, *Annual Review of Neuroscience*, 15:31-56.

[6] **L¨owel, S., and Singer, W. (1992).** Selection of intrinsic horizontal connections in the visual cortex by correlated neuronal activity, *Science*, 255(5041):209-12.

[7] **Gilbert CD(1992).** Horizontal integration and cortical dynamics, *Neuron*, (1):1-13.

[8] **Kohonen T(2001).** Self-Organizing Maps, *Springer Berlin*, vol 30.3nd edn.

[9] **Obermayer, K., Ritter, H., and Schulten, K. J. (1990d).** A principle for the formation of the spatial structure of cortical feature maps, *Proceedings of the National Academy of Sciences, USA*, 87:8345–8349.

[10] **amdahl's law.** `http://en.wikipedia.org/wiki/Amdahl27s_law`

[11] **Miikkulainen, R., Bednar, J.A., Choe, Y., Sirosh, J. (2005).** *Computational Maps in the Visual Cortex*. Springer

[12] **James A. Bednar (2002).** *Learning to See: Genetic and Environmental Influences on Visual Development*. PhD Thesis Department of Computer Sciences, The University of Texas at Austin

[13] **James A. Bednar and Amol Kelkar and Risto Miikkulainen (2001).** Scaling Self-Organizing Maps To Model Large Cortical Networks, *Neuroinformatics*, 275-302.

[14] **B. Glackin, T.M. McGinnity, L.P. Maguire, Q.X. Wu, and A. Belatreche (2005).** A Novel Approach for the Implementation of Large Scale Spiking Neural Networks on FPGA Hardware., *Computational Intelligence and Bioinspired Systems, 8th International Work-Conference on Artificial Neural Networks, IWANN*, pp. 552 – 563.

[15] **Joseph Sirosh and Risto Miikkulainen (1994).** Cooperative Self-Organization Of Afferent And Lateral Connections In Cortical Maps, *Biological Cybernetics*, 66-78.

[16] **L. A. Hodgkin and F. A. Huxley (1952).** A quantitative description of membrane current and its application to conduction and excitation in nerve, *The Journal of physiology 117*, p. 500–544.

[17] **F. L. Abbotta (1999).** Lapicque's introduction of the integrate-and-fire model neuron, *in Brain Research Bulletin, vol. 50*, pp. 303-304.

[18] **Izhikevich (2003).** Simple Model of Spiking Neurons, *IEEE TRANSACTIONS ON NEURAL NETWORKS, vol. 14*, 1569 - 1572.

[19] . `http://neuron.duke.edu`

[20] **SNNS.** `http://www-ra.informatik.uni-tuebingen.de/SNNS`

[21] **D. B. Thomas and W. Luk (2009).** FPGA accelerated simulation of biologically plausible spiking neural networks, *In Proc. IEEE Symp. Field-Programmable Custom Computing Machines (FCCM)*, pp 279-288.

[22] **Li-Chiu Chang, Fi-John Chang (2002).** An efficient parallel algorithm for LISSOM neural network, *Parallel Computing 28*, 1611–1633.

[23] **D. Thomas and W. Luk(2009).** FPGA Accelerated Simulation of Biologically Plausible Spiking Neural Networks, *2009 17th IEEE Symposium on Field Programmable Custom Computing Machines*, 45-52.

[24] **S. Moore, P. Fox, S. Marsh, A. Markettos and A. Mujumdar (2012).** Bluehive–a field-programmable custom computing machine for extreme-scale real-time neural network simulation, *Proceedings of the 10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'12)*, p. 133–140.

[25] **Miikkulainen, R., Bednar, J.A., Choe, Y., Sirosh, J (2005).** *Computational Maps in the Visual Cortex*. Springer

[26] **Convey Manual.** `http://wikis.ece.iastate.edu/cpre584/images/6/64/ConveyPDKReferenceManual pdf`

[27] **LISSOM GPU.** `https://code.google.com/archive/p/lissom/`