ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΙΤΛΟΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Αναδιατασσόμενη υλοποίηση και αξιολόγηση της Bit Pragmatic Inference μηχανής βαθιάς μάθησης

Reconfigurable implementation and evaluation of the Bit Pragmatic Deep Learning Inference engine.

Κωνσταντίνος Κασφίκης

Επιβλέπων Καθηγητής:
Απόστολος Δόλλας
Επιτροπή:
Διονύσιος Πνευματικάτος
Μιχάλης Ζερβάκης

Διπλωματική εργασία που υποβλήθηκε στα πλαίσια της ολοκλήρωσης του διπλώματος Ηλεκτρολόγου Μηχανικού και Μηχανικού Υπολογιστών

Πολυτεχνείο Κρήτης
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

Περίληψη

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

ΠΕΡΙΓΡΑΦΗ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Αναδιατασσόμενη υλοποίηση και αξιολόγηση της Bit Pragmatic Inference μηχανής βαθιάς μάθησης

Reconfigurable implementation and evaluation of the Bit Pragmatic Deep Learning Inference engine.

Κωνσταντίνος Κασφίκης

Τα τελευταία χρόνια τα Συνελικτικά Νευρωνικά Δίκτυα (ΣΝΔ) έχουν δείξει εξαιρετικά αποτελέσματα σε σύνθετα προβλήματα αναγνώρισης εικόνων. Έχουν υιοθετηθεί επί του παρόντος για την επίλυση ενός συνεχώς αυξανόμενου αριθμού προβλημάτων, που κυμαίνονται από την αναγνώριση της φωνής έως την κατάτμηση και ταξινόμηση της εικόνας. Η συνεχής αύξηση του όγκου επεξεργασίας που απαιτείται από ΣΝΔ δημιουργεί το πεδίο για τις μεθόδους υποστήριξης υλικού, προκειμένου να μειωθούν περαιτέρω οι χρόνοι εκτέλεσης. Το πλήθος των ερευνών για την Μηχανική Μάθηση και ειδικά για τα ΣΝΔ, που υλοποιούνται σε πλατφόρμες FPGA, καταδεικνύει το τεράστιο βιομηχανικό και ακαδημαϊκό ενδιαφέρον.

Αυτή η εργασία παρουσιάζει την σχεδίαση και την υλοποίηση ενός Inference Accelerator για ΣΝΔ σε FPGA χρησιμοποιώντας την τεχνική Bit Pragmatic. Με την επιτάχυνση της λειτουργίας βασικών δομικών επιπέδων που συναντώνται σε κάθε ΣΝΔ, επιτυγχάνεται η επιτάχυνση της συνολικής λειτουργίας του υπό εξέταση ΣΝΔ. Παρουσιάζεται η βασική δομή του εν λόγω Accelerator, ο οποίος αναπτύχθηκε με γνώμονα να είναι λειτουργικός ακόμα και σε FPGA μέτριων δυνατοτήτων. Γίνεται επαλήθευση των αποτελεσμάτων του και έπειτα αναλύεται η βελτιστοποίηση των επιδόσεων του, χρησιμοποιώντας παραλληλισμό σε εσωτερικά δομικά modules, με την μεταφορά της σχεδίασης σε υψηλότερων δυνατοτήτων FPGA. Οι επιδόσεις του συστήματος, ειδικά με την χρήση παραλληλισμού, υπερβαίνουν τις αντίστοιχες επιδόσεις της CPU i7 -8700K ενώ είναι συγκρίσιμες με υπάρχουσες υλοποιήσεις σε FPGA.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή κ. Διονύσιο Πνευματικάτο για την πολύτιμη καθοδήγησή του κατά τη διάρκεια της διαδικασίας αυτής της εργασίας, καθώς και για την ευκαιρία που μου έδωσε να διευρύνω τους ορίζοντές μου στον τομέα της Τεχνητής Νοημοσύνης και της Αρχιτεκτονικής Υπολογιστών. Επίσης, θα ήθελα να ευχαριστήσω τον κ. Απόστολο Δόλλα και τον κ. Μιχάλη Ζερβάκη που δέχτηκαν να είναι μέλη της εξεταστικής επιτροπής.

Τέλος, ευχαριστώ την οικογένειά μου και τους φίλους μου για την υποστήριξη τους κατά την διάρκεια της ανάπτυξης της παρούσας διπλωματικής εργασίας.

Κωνσταντίνος Κασφίκης
Χανιά, 2019

Πίνακας περιεχομένων

Εικόνες

Πίνακες

Διαγράμματα

Συντομογραφίες

ANN:	Artificial Neural Network
ΤΝΝ:	Τεχνητά Νευρωνικά Δίκτυα
CNN:	Convolutional Neural Network
ΣΝΔ:	Συνελικτικό Νευρωνικό Δίκτυο
ΠΥΑ:	Πραγματικό Ύψος Ενεργοποιήσεων
ΠΜΑ:	Πραγματικό Μήκος Ενεργοποιήσεων
ΠΥΦ:	Πραγματικό ύψος φίλτρου
ΠΜΦ:	Πραγματικό μήκος φίλτρου
ΜΥΑ:	Μέγιστο Ύψος Ενεργοποιήσεων
ΜΜΑ:	Μέγιστο Μήκος Ενεργοποιήσεων
ΜΥΦ:	Μέγιστο ύψος φίλτρου
ΜΜΦ:	Μέγιστο μήκος φίλτρου
CCU:	Convolution Calculation Unit
SFC:	Single Filter Calculator
4-CFC:	4 - Channel Frame Calculator
MSB:	Most Significant Bit(s)
LSB:	Least Significant Bit(s)
CPU:	Central Processor Unit
GPU:	Graphic Processor Unit
FPGA:	Field Programmable Gate Array
AI:	Artificial Intelligence
ML:	Machine Learning
ReLU:	Rectified Linear Unit
FF:	Flip Flop(s)
LUT:	Look Up Table
RAM:	Random Access Memory
B-RAM:	Block Random Access Memory
DSP:	Digital Signal Processor
FC:	Fully Connected

Αφιερωμένο στην Οικογένειά μου και στους Φίλους μου

1. Εισαγωγή

1.1. Κίνητρο

Τα τελευταία χρόνια η εκθετική αύξηση παραγωγής ψηφιακού περιεχομένου είναι συνεχής, οδηγώντας στην ανάγκη αυτοματοποιημένης κατηγοριοποίησης. Η εξαγωγή και ανάλυση αυτών των όγκων δεδομένων πληροφορίας είναι δύσκολη έως ακατόρθωτη χρησιμοποιώντας συμβατικά εργαλεία software και τεχνολογίες λογισμικού. Τα μεγέθη και οι διαστάσεις των ψηφιακών πληροφοριών αυξάνουν με την πάροδο του χρόνου με εκπληκτικούς ρυθμούς. Χαρακτηριστικό παράδειγμα αποτελεί, ότι ήδη από το 2013 στο internet διακινούνται 1,8 petabytes πληροφορίας σε καθημερινή βάση, σύμφωνα με την National Security Agency – NSA [1]. Σύμφωνα με το Forbes [2] η ανθρωπότητα παράγει καθημερινά 2,5 exabytes ψηφιακών δεδομένων. Η ταχεία αυτή αύξηση των ψηφιακών δεδομένων δημιουργεί την ανάγκη αναζήτησης μεθόδων που θα μας επιτρέπουν την εξαγωγή συμπερασμάτων από πληροφορίες χαμηλού επιπέδου που δίνονται από τα ψηφιακά δεδομένα (εικόνες, video, ήχος κλπ). Μεταξύ των προτεινόμενων μεθόδων για την κάλυψη της ανάγκης αυτής περιέχεται η Βαθιά Μαθηση και συνεπώς τα Συνελικτικά Νευρωνικά Δίκτυα (ΣΝΔ), τα οποία υπερέχουν σε πολλές περιπτώσεις ακόμη και έναντι του ανθρώπου σε εφαρμογές σχετικές με την μηχανική όραση [11][12][13] και την αναγνώριση ομιλίας [14]. Τα ΣΝΔ προσφέρουν υψηλή ακρίβεια στην αναγνώριση εικόνων, ομιλίας, γλωσσών και γενικά σε πληθώρα προβλημάτων τα οποία τους ανατίθενται [3][4][5][6][7]. Η υψηλή απόδοση την οποία παρουσιάζουν αντισταθμίζεται από το γεγονός ότι για την λειτουργία τους απαιτούνται τεράστιοι υπολογιστικοί πόροι. Χαρακτηριστικό παράδειγμα αποτελεί το γεγονός ότι για την κατηγοριοποίηση μίας και μόνο εικόνας απαιτούνται 40 GOP το δευτερόλεπτο [8]. Ως αποτέλεσμα απαιτείται εξειδικευμένο υλικό για την επιτάχυνση της λειτουργίας τους. Τα GPUs ενδείκνυνται για την εκπαίδευση (training) των ΣΝΔ, διαδικασία υπολογιστικά απαιτητική, που μπορεί να διαρκέσει ώρες, ημέρες ακόμα ίσως και εβδομάδες για να ολοκληρωθεί, σε μία ή περισσότερες GPUs, καθώς στις περισσότερες περιπτώσεις η διαδικασία αυτή πραγματοποιείται μια φορά. Η inference λειτουργία των ΣΝΔ, σε αντίθεση με την διαδικασία της εκπαίδευσης, είναι συνεχής. Προκύπτει λοιπόν η ανάγκη επιτάχυνσης της διαδικασίας inference, η οποία καλύπτεται με την

χρήση εξειδικευμένου υλικού. Υποκατηγορία του εξειδικευμένου υλικού αποτελούν τα Field Programmable Gate Arrays (FPGAs). Τα τελευταία χρόνια ειδικά, έχουν προταθεί πολλές σχεδιάσεις για FPGA οι οποίες αφορούν κυρίως την inference λειτουργία. Οι υλοποιήσεις αυτές στοχεύουν στην κάλυψη των αναγκών τόσο σε υψηλής απόδοσης υπολογιστικών μονάδων (π.χ. Data Centers) [9], όσο και σε ενσωματωμένα χαμηλής ενέργειας συστήματα [15].

1.2. Επιστημονική Συνεισφορά

Η συγκεκριμένη εργασία αποσκοπεί στην επιτάχυνση της inference λειτουργίας οποιουδήποτε μοντέλου ΣΝΔ με την χρήση και αξιολόγηση της τεχνικής Bit Pragmatic.

Αρχικά αναλύεται η δομή ενός τυπικού ΣΝΔ και απομονώνται τα επίπεδα αυτού που τείνουν να συναντώνται σε κάθε μοντέλο. Επιχειρείται η υλοποίηση αυτών με αναδιατασσόμενη λογική, με στόχο την επιτάχυνση λειτουργίας του καθε επιπέδου ξεχωριστά και επομένως την βελτίωση των επιδόσεων του συνολικού ΣΝΔ. Η χρήση εξειδικευμένου υλικού μας επιτρέπει την παραλληλοποίηση της λειτουργίας διακριτών επιπέδων, όταν αυτά συναντώνται διαδοχικά στο μοντέλο του υπό επεξεργασία ΣΝΔ. Η σχεδίαση που προτείνεται αναπτύχθηκε με γνώμονα την χρήση όσο το δυνατόν λιγότερων πόρων υλικού προκειμένου να είναι λειτουργική σε FPGA χαμηλού κόστους. Ταυτόχρονα εξετάζονται οι τρόποι χρήσης των modules του συστήματος έτσι ώστε να επιτυγχάνονται καλύτεροι χρόνοι εκτέλεσης, όταν πολλαπλά όμοια δομικά modules χρησιμοποιούνται παράλληλα, με την μεταφορά της σχεδίασης σε FPGA μεγαλύτερων δυνατοτήτων.

Το σύνολο των απαιτούμενων πολλαπλασιασμών, πραγματοποιούνται με την χρήση της τεχνικής Bit Pragmatic. Η τεχνική αυτή εκμεταλεύεται την χαμηλή περιεκτικότητα σε bit 1 στα δεδομένα που καλείται να επεργαστεί κάθε ΣΝΔ (μόνο το 8% των bit των δεδομένων είναι 1), ισοροπώντας μεταξύ ταχύτητας και χρήσης πόρων υλικού. Συγκεκριμένα η τεχνική αυτή, αγνοεί τα bit 0 στον πολλαπλασιαστή, παράγοντας μερικά γινόμενα ολισθαίνοντας τον πολλαπλασιαστέο κατά θέσεις ίσες με την θέση που εντοπίστηκε ο εκάστοτε άσσος στον πολλαπλασιαστή. Η εργασία παρουσιάζει και επαληθεύει τα απολέσματα του συστήματος με την ενσωμάτωση της τεχνικής Bit Pragmatic, συγκρίνοντας τις επιδόσεις τόσο της βασικής σχεδίασης όσο και πιθανών σχεδιάσεων με χρήση παραλληλισμού, με άλλα συστήματα, παρουσιάζοντας τελικά τα πλεονεκτήματα και τα μειονεκτήματα που προκύπτουν από την χρήση αυτής.

1.3. Περιγραφή Εργασίας

Στο Κεφάλαιο 2, περιγράφεται με λεπτομέρεια το θεωρητικό υπόβαθρο της Μηχανικής μάθησης, Βαθιάς Μάθησης, καταλήγοντας στα Νευρωνικά Δικτύα και συγκεκριμένα στα Συνελικτικά Νευρωνικά Δίκτυα.

Στο Κεφάλαιο 3, περιγράφεται αναλυτικά η δομή της υλοποίησης που αναπτύχθηκε για την δημιουργία ενός inference Accelerator για ΣΝΔ με χρήση FPGA.

Στο Κεφάλαιο 4, παρουσιάζονται αποτελέσματα του συστήματος και ελέγχεται η εγκυρότητά τους με την χρήση εξωτερικών εργαλείων λογισμικού που αναπτύχθηκαν από εμάς.

Στο Κεφάλαιο 5, παρουσιάζονται οι χρόνοι εκτέλεσης του συστήματος και αναλύονται τεχνικές επιτάχυνσης με χρήση παραλληλισμού. Επίσης γίνεται σύγκριση της υλοποίησής μας ως προς τους χρόνους εκτέλεσης με άλλες πλατφόρμες.

Στο Κεφάλαιο 6, παρουσιάζονται συμπεράσματα και σκέψεις για την μελλοντική επέκταση της υλοποίησης.

2. Θεωρητικό Υπόβαθρο

2.1. Τεχνητή Νοημοσύνη (ΤΝ - AI)

Ο όρος τεχνητή νοημοσύνη αναφέρεται στον κλάδο της Επιστήμης των Υπολογιστών ο οποίος ασχολείται με τη σχεδίαση και την υλοποίηση υπολογιστικών συστημάτων που μιμούνται στοιχεία της ανθρώπινης συμπεριφοράς τα οποία υπονοούν έστω κάποια στοιχειώδη ευφυΐα όπως μάθηση, προσαρμοστικότητα, εξαγωγή συμπερασμάτων, κατανόηση από συμφραζόμενα, επίλυση προβλημάτων κλπ. Η τεχνητή νοημοσύνη αποτελεί σημείο τομής μεταξύ πολλαπλών επιστημών όπως της πληροφορικής, της ψυχολογίας, της φιλοσοφίας, της νευρολογίας, της γλωσσολογίας και της επιστήμης μηχανικών, με στόχο τη σύνθεση ευφυούς συμπεριφοράς, με στοιχεία συλλογιστικής, μάθησης και προσαρμογής στο περιβάλλον.

Με βάση τον επιθυμητό επιστημονικό στόχο η ΤΝ κατηγοριοποιείται σε άλλου τύπου ευρείς τομείς, όπως επίλυση προβλημάτων, μηχανική μάθηση, ανακάλυψη γνώσης, συστήματα γνώσης κλπ. Επίσης υπάρχει επικάλυψη με συναφή επιστημονικά πεδία όπως η μηχανική όραση, η επεξεργασία φυσικής γλώσσας ή η ρομποτική, τα οποία μπορούν να τοποθετηθούν μες στο ευρύτερο πλαίσιο της σύγχρονης τεχνητής νοημοσύνης ως ανεξάρτητα πεδία της.

Η σύγχρονη τεχνητή νοημοσύνη αποτελεί ένα από τα πλέον «μαθηματικοποιημένα» και ταχέως εξελισσόμενα πεδία της πληροφορικής [36].

2.2. Μηχανική Μάθηση (ΜΜ - ML)

Η Μηχανική Μάθηση αποτελεί υποσύνολο της επιστήμης των υπολογιστών που αναπτύχθηκε από τη μελέτη της αναγνώρισης προτύπων και της υπολογιστικής θεωρίας μάθησης στην τεχνητή νοημοσύνη. Η μηχανική μάθηση ορίζεται από τον Άρθουρ Σάμουελ ως "Πεδίο μελέτης που δίνει στους υπολογιστές την ικανότητα να μαθαίνουν, χωρίς να έχουν ρητά προγραμματιστεί" [16]. Διερευνά τη μελέτη και την κατασκευή αλγορίθμων που μπορούν να μαθαίνουν από τα δεδομένα και να

κάνουν προβλέψεις σχετικά με αυτά[17]. Τέτοιοι αλγόριθμοι λειτουργούν κατασκευάζοντας μοντέλα από πειραματικά δεδομένα, προκειμένου να κάνουν προβλέψεις βασιζόμενες στα δεδομένα που τους δίνονται ή να εξάγουν αποφάσεις που εκφράζονται ως το αποτέλεσμα.

Η μηχανική μάθηση είναι στενά συνδεδεμένη και συχνά συγχέεται με υπολογιστική στατιστική, ένας κλάδος, που επίσης επικεντρώνεται στην πρόβλεψη μέσω της χρήσης των υπολογιστών. Έχει ισχυρούς δεσμούς με την μαθηματική βελτιστοποίηση, η οποία παρέχει μεθόδους, τη θεωρία και τομείς εφαρμογής. Εφαρμόζεται σε μια σειρά από υπολογιστικές εργασίες, όπου τόσο ο σχεδιασμός όσο και ο ρητός προγραμματισμός των αλγορίθμων είναι ανέφικτος. Παραδείγματα εφαρμογών αποτελούν η αναγνώριση κειμένου (OCR), οι μηχανές αναζήτησης και η μηχανική όραση. Μερικές φορές συγχέεται με την εξόρυξη δεδομένων[19], όπου η τελευταία επικεντρώνεται περισσότερο στην εξερευνητική ανάλυση των δεδομένων, γνωστή και ως μη επιτηρούμενη μάθηση[18].

Στο πεδίο της ανάλυσης δεδομένων, η μηχανική μάθηση είναι μια μέθοδος που χρησιμοποιείται για την επινόηση πολύπλοκων μοντέλων και αλγορίθμων που οδηγούν στην πρόβλεψη. Τα αναλυτικά μοντέλα επιτρέπουν στους ερευνητές, τους επιστήμονες δεδομένων, τους μηχανικούς και τους αναλυτές να παράγουν αξιόπιστες αποφάσεις και αποτελέσματα ώστε να αναδείξουν αλληλοσυσχετίσεις μέσω της μάθησης από ιστορικές σχέσεις και τάσεις στα δεδομένα [20][39].

2.3. Βαθιά Μαθηση (ΒΜ - DL)

Η βαθιά μάθηση μπορεί να ταξινομηθεί ως υποκατηγορία της Μηχανικής Μάθησης. Στην πραγματικότητα, η βαθιά μάθηση τεχνικά ταυτίζεται με την μηχανική μάθηση και λειτουργεί με παρόμοιο τρόπο (για αυτό τον λόγο και οι όροι μερικές φορές εναλλάσσονται). Ωστόσο, οι δυνατότητές τους είναι διαφορετικές.

Ενώ τα βασικά μοντέλα μηχανικής μάθησης γίνονται ολοένα και καλύτερα στην λειτουργία που τους έχει ανατεθεί, χρειάζοναι κάποια καθοδήγηση για την παραγωγή ορθών αποτελεσμάτων. Στην περίπτωση που κάποιος αλγόριθμος μηχανικής μάθησης επιστρέψει ανακριβή αποτελέσματα, τότε ο μηχανικός πρέπει να επέμβει και να προβεί σε προσαρμογές. Με την χρήση ενός μοντέλου βαθιάς

μάθησης, ο αλγόριθμος είναι σε θέση να καθορίσει από μόνος του αν μια εκτίμηση - αποτέλεσμα είναι ακριβές ή όχι μέσω του νευρωνικού του δικτύου.

Ένα μοντέλο βαθιάς μάθησης είναι σχεδιασμένο ώστε να αναλύει δεδομένα με μια λογική δομή παρόμοια με το πως ένας άνθρωπος θα συνάγει συμπεράσματα. Για την επίτευξη των παραπάνω, οι εφαμορμογές βαθιά μάθησης χρησιμοποιούν μια πολυεπίπεδη δομή αλγορίθμων που ονομάζεται τεχνητό νευρωνικό δίκτυο. Ο σχεδιασμός ενός τεχνητού νευρωνικού δικτύου εμπνέεται από το βιολογικό νευρωνικό δίκτυο του ανθρώπινου εγκεφάλου, οδηγώντας σε μια διαδικασία μάθησης που είναι πολύ πιο ικανή από εκείνη των πρότυπων μοντέλων μηχανικής μάθησης.

2.4. Νευρωνικά Δίκτυα

Νευρωνικό δίκτυο ονομάζεται ένα κύκλωμα διασυνδεδεμένων νευρώνων. Στην περίπτωση βιολογικών νευρώνων, πρόκειται για ένα τμήμα νευρικού ιστού. Στην περίπτωση τεχνητών νευρώνων, πρόκειται για ένα αφηρημένο αλγοριθμικό κατασκεύασμα το οποίο εμπίπτει στον τομέα της υπολογιστικής νοημοσύνης, όταν στόχος του νευρωνικού δικτύου είναι η επίλυση κάποιου υπολογιστικού προβλήματος.

Το νευρωνικό δίκτυο είναι ένα δίκτυο από απλούς υπολογιστικούς κόμβους τους νευρώνες, διασυνδεδεμένους μεταξύ τους. Είναι εμπνευσμένο από το Κεντρικό Νευρικό Σύστημα (ΚΝΣ), το οποίο και προσπαθεί να προσομοιώσει.

Οι νευρώνες είναι τα δομικά στοιχεία του δικτύου. Κάθε τέτοιος κόμβος δέχεται ένα σύνολο αριθμητικών εισόδων από διαφορετικές πηγές (είτε από άλλους νευρώνες, είτε από το περιβάλλον), πραγματοποιεί έναν υπολογισμό με βάση αυτές τις εισόδους και παράγει μία έξοδο [27]. Η εν λόγω έξοδος είτε κατευθύνεται στο περιβάλλον, είτε τροφοδοτείται ως είσοδος σε άλλους νευρώνες του δικτύου. Υπάρχουν τρεις τύποι νευρώνων: οι νευρώνες εισόδου, οι νευρώνες εξόδου και οι υπολογιστικοί νευρώνες ή κρυμμένοι νευρώνες. Οι νευρώνες εισόδου δεν επιτελούν κανέναν υπολογισμό, μεσολαβούν απλώς ανάμεσα στις περιβαλλοντικές εισόδους του δικτύου και στους υπολογιστικούς νευρώνες. Οι νευρώνες εξόδου διοχετεύουν στο περιβάλλον τις τελικές αριθμητικές εξόδους του δικτύου. Οι υπολογιστικοί νευρώνες πολλαπλασιάζουν κάθε είσοδό τους με το αντίστοιχο

συναπτικό βάρος και υπολογίζουν το ολικό άθροισμα των γινομένων. Το άθροισμα αυτό τροφοδοτείται ως όρισμα στη συνάρτηση ενεργοποίησης, την οποία υλοποιεί εσωτερικά κάθε κόμβος. Η τιμή που λαμβάνει η συνάρτηση για το εν λόγω όρισμα είναι και η έξοδος του νευρώνα για τις τρέχουσες εισόδους και βάρη. Κάθε κρυφό επίπεδο αποτελείται από ένα σύνολο νευρώνων, όπου κάθε νευρώνας είναι πλήρως συνδεδεμένος με όλους τους νευρώνες του προηγούμενου επιπέδου, και λειτουργεί ανεξάρτητα από τους νευρώνες του ίδιου επιπέδου (στις συνήθεις προσεγγίσεις δεν αλληλεπιδρούν μεταξύ τους).

Το κύριο χαρακτηριστικό των νευρωνικών δικτύων είναι η εγγενής ικανότητα μάθησης. Ως μάθηση μπορεί να οριστεί η σταδιακή βελτίωση της ικανότητας του δικτύου να επιλύει κάποιο πρόβλημα. Η μάθηση επιτυγχάνεται μέσω της εκπαίδευσης, μίας επαναληπτικής διαδικασίας σταδιακής προσαρμογής των παραμέτρων του δικτύου (συνήθως των βαρών και της πόλωσής του) σε τιμές κατάλληλες ώστε να επιλύεται με επαρκή επιτυχία το προς εξέταση πρόβλημα. Αφού ένα δίκτυο εκπαιδευτεί, οι παράμετροί του συνήθως σταθεροποιούνται στις κατάλληλες τιμές και από εκεί κι έπειτα είναι σε λειτουργική κατάσταση. Το ζητούμενο είναι το λειτουργικό δίκτυο να χαρακτηρίζεται από μία ικανότητα γενίκευσης: αυτό σημαίνει πως δίνει ορθές εξόδους για εισόδους διαφορετικές από αυτές με τις οποίες εκπαιδεύτηκε [26].

Υποκατηγορία των τεχνητών νευρωνικών δικτύων αποτελούν τα Συνελικτικά Νευρωνικά δίκτυα, τα οποία θα αναλυθούν στην επόμενη ενότητα [37].

2.5. Ανάλυση και Δομή ΣΝΔ

2.5.1. Ανάλυση ΣΝΔ

Τα συνελικτικά νευρωνικά δίκτυα (ΣΝΔ) έχουν αποδειχθεί ως μια κατηγορία τεχνητών νευρωνικών δικτύων (ΤΝΝ) που μπορούν να ανταπεξέλθουν με μεγάλη ακρίβεια σε προβλήματα ταξινόμησης εικόνας [38]. Η αρχιτεκτονική τους και οι μέθοδοι τους είναι αρκετά συναφείς με τις ήδη υπάρχουσες μεθόδους μοντελοποίησης των τεχνητών νευρωνικών δικτύων.

Τα ΣΝΔ αποτελούνται από τεχνητούς νευρώνες με υπό-μάθηση (learnable) παραμέτρους. Οι πιο γνωστές παράμετροι είναι τα βάρη (weights) και οι πολώσεις (bias). Κάθε νευρώνας δέχεται ως είσοδο ένα διάνυσμα (X), και εξάγει το εσωτερικό γινόμενο των βαρών (W) του με την είσοδο, προστιθέμενης μιας προαιρετικής πόλωσης (b).

$$Y = f(x) = (w * x) + b$$

Ολόκληρο το δίκτυο αποτελεί μια διαφορίσιμη συνάρτηση, προκειμένου να μπορεί να εφαρμοσθεί η συνάρτηση οπισθοδιάδοσης των σφαλμάτων (Backward propagation of errors) [21], που χρησιμοποιείται κατά την εκπαίδευση του δικτύου.

Ο κύριος λόγος που τα τυπικά TNN δεν μπορούν να χρησιμοποιηθούν για προβλήματα ταξινόμησης εικόνας και προκύπτει η ανάγκη χρήσης των Συνελικτικών Νευρωνικών Δικτύων είναι επειδή, υποθέτοντας μια εικόνα μεγέθους 227x227x3 ως είσοδο, ένας πλήρως συνδεδεμένος με την είσοδο νευρώνας, θα αποτελούνταν από (227x227x3=) 154.587 βάρη. Επιπλέον, σε ένα κρυφό επίπεδο, προκειμένου να επιτευχθεί η απαιτούμενη λειτουργικότα, θα υπάρχουν σίγουρα περισσότεροι από έναν νευρώνα, και σίγουρα απαιτούνται περισσότερα από ένα κρυφά επίπεδα. Επομένως, οι υπό-μάθηση παράμετροι θα αυξάνονταν δραματικά, δημιουργώντας αχανή δίκτυα, μη πρακτικά και επιρρεπή στην υπερεκπαίδευση. Επιπλέον, τα συγκεκριμένα δίκτυα θα αγνοούσαν της αυξημένη συσχέτιση που υπάρχει μεταξύ των γειτονικών εικονοστοιχείων σε σχέση με πιο απόμακρα, κάτι το οποίο αποτελεί κύριο χαρακτηριστικό γνώρισμα των φυσικών εικόνων. Τα συνελικτικά νευρωνικά δίκτυα αποτελούν παρακλάδι των βαθέων νευρωνικών δικτύων (Deep Neural Networks – DNNs).

Στα ΣΝΔ, ένα φίλτρο αντιπροσωπεύεται από έναν συνελικτικό πυρήνα ο οποίος είναι κατά πολύ μικρότερος χωρικά από την είσοδο. Κάθε συνελικτικό επίπεδο αποτελείται από πολλά φίλτρα, τα οποία είναι στοιχισμένα με τέτοιο τρόπο ώστε η απόκριση τους στην έξοδο να ανταποκρίνεται στην ίδια περιοχή της εισόδου (η οποία ονομάζεται και δεκτικό πεδίο). Τα φίλτρα σαρώνουν την είσοδο με τέτοιο τρόπο ώστε τα δεκτικά πεδία να αποτελούνται από αλληλεπικαλυπτόμενες περιοχές της εισόδου, προκειμένου να εξαχθεί μια πιο ομαλή αναπαράσταση της εισαγόμενης εικόνας/δείγματος. Αυτή η μέθοδος ονομάζεται διαμοιρασμός βαρών

(weight sharing) και χρησιμοποιείται λόγω του ότι στις φυσικές εικόνες υπάρχουν πανομοιότυπα χαρακτηριστικά διάσπαρτα στον χώρο. Με αυτή τη μέθοδο το πλήθος των ελεύθερων παραμέτρων προς μάθηση, και κατ'επέκταση η επιρρέπεια στην υπέρ-εκπαίδευση μειώνεται δραστικά, καθώς τα επίπεδα του δικτύου δεν είναι πλήρως συνδεδεμένα μεταξύ τους, αλλά τα φίλτρα συνελίσονται με τα δεκτικά πεδία που σαρώνουν όλες τις περιοχές της εισόδου. Με αυτή τη μέθοδο επίσης, σε επίπεδο υλικού, μειώνονται οι απαιτήσεις μνήμης και βελτιώνεται η συνολική απόδοση.

2.5.2. Προβλήματα που παρουσιάζουν τα ΣΝΔ

Τα ΣΝΔ πολλές φορές αποτυγχάνουν να κατηγοριοποιήσουν ορθά μετασχηματισμούς της εισόδου. Εάν ένα ΣΝΔ εκπαιδευτεί να ταξινομεί ένα αντικείμενο κάτω από μια συγκεκριμένη γωνία, κατά πάσα πιθανότητα θα αποτύχει να το ταξινομήσει όταν του δοθεί, έστω και ελαφρώς, περιστραμμένο, παρόλο που πρόκειται για το ίδιο αντικείμενο. Η ίδια υπόθεση ισχύει και για οριζόντια αναστροφή του αντικειμένου, σκεβρωμένο/λοξό αντικείμενο ή ακόμα και ελαφρώς μετατοπισμένο σε μια διαφορετική θέση εντός της περιοχής εισόδου.

Ένα ακόμα πρόβλημα που προκύπτει από την χρήση των ΣΝΔ είναι πως κατά την διάρκεια της εκπαίδευσης, δεν μπορούμε να οδηγήσουμε το ΣΝΔ στο ποια χαρακτηριστικά να εξάγει προκειμένου να διαχωρίζει τα αντικείμενα. Κατά την αναγνώριση ενός αντικειμένου, τα ΣΝΔ μπορεί να επικεντρωθούν σε ένα χαρακτηριστικό που φέρει το αντικείμενο, το οποίο μπορεί να προσομοιάζει σε άλλα αντικείμενα, μη δίνοντας την δέουσα βαρύτητα σε άλλα χαρακτηριστικά, οδηγώντας στην λανθασμένη αναγνώριση του [10].

Δεδομένου ότι τα ΣΝΔ μπορούν να “ξεγελαστούν” εξαιρετικά εύκολα με χρήση τεχνητών εικόνων η χρήση τους αντενδείκνυται σε κρίσιμες εφαρμογές.

Ο κύκλος ζωής ενός ΣΝΔ διακρίνεται σε δύο μέρη:

- Εκπαίδευση (Training)

  Η διαδικασία της εκπαίδευσης είναι διαδικασία που λαμβάνει χώρα μία φορά (στις περισσότερες περιπτώσεις) απαιτώντας μεγάλο όγκο υπολογιστικών πόρων. Η εκπαίδευση γίνεται με την διαδικασία της οπισθοδιάδοσης (backpropagation) χρησιμοποιώντας ένα εκπαιδευτικό σύνολο δεδομένων. Μόλις ολοκληρωθεί η διαδικασία της εκπαίδευσης όλα τα βάρη και παράμετροι του δικτύου έχουν βελτιστοποιηθεί κατά τέτοιο τρόπο ώστε να κατηγοριοποιήσουν σωστά εικόνες που ανήκουν στο εκπαιδευτικό σύνολο δεδομένων τους. Όταν μια νέα εικόνα εισαχθεί σαν είσοδος στο δίκτυο αυτό θα εκτελέσει μια εμπρόσθια εξάπλωση από όλα τα επιμέρους επίπεδα και θα παράξει μια πιθανότητα για κάθε κλάση που υπάρχει. Αν το εκπαιδευτικό σύνολο δεδομένων που τροφοδοτείται κατά την διάρκεια της εκπαίδευσης είναι αρκετά εκτενές, το δίκτυο θα είναι σε θέση να δεχτεί νέες εικόνες, παρεμφερείς με κάποιες από τις εικόνες του εκπαιδευτικού συνόλου, και να τις κατηγοριοποιήσει στην σωστή κατηγορία.

- Inference

  Μετά την διαδικασία της εκπαίδευσης, το δίκτυο είναι έτοιμο να δεχτεί στην είσοδό του δεδομένα και να τα κατηγοριοποιήσει. Προφανώς η inference λειτουργία δεν θα αποδώσει εάν δεν έχει προηγηθεί εκπαίδευση καθώς το δίκτυο δεν έχει λάβει την κατάλληλη γνώση για την είσοδο που καλείται να κατηγοριοποιήσει. Κατά την διαδικασία αυτή το σύνολο των φίλτρων και των παραμέτρων έχουν ήδη καθορισθεί και βάσει αυτών εξάγονται τα τελικά αποτελέσματα.

2.5.3. Δομή Συνελικτικού Νευρωνικού Δικτύου

Ένα τυπικό συνελικτικό νευρωνικό δίκτυο αποτελείται από εκατομμύρια νευρώνες διατεταγμένους σε αρκετά layers. Προκειμένου να δομηθεί ένα ΣΝΔ μπορούν να χρησιμοποιηθούν οι παρακάτω θεμελιώδεις τύποι επιπέδων:

1. Συνελικτικά επίπεδα / Convolutional Layers.

2. Συναρτήσεις ενεργοποίησης / Activation Functions.

3. Επίπεδα χωρικής υποδειγματοληψίας / Pooling Layers.

4. Πλήρως συνδεδεμένα επίπεδα / Fully Connected layers.

Κάθε επίπεδο είναι παραμετροποιήσιμο μέσω των υπέρ-παραμέτρων του. Δεν υφίσταται συγκεκριμένος τρόπος δόμησης των ΣΝΔ ή επιλογής των υπέρ-παραμέτρων των επιπέδων προκειμένου να δημιουργηθεί ένα ΣΝΔ που να μπορεί να χρησιμοποιηθεί με απόλυτη επιτυχία. Ωστόσο, ισχύουν κάποιες γενικές παραδοχές:

• Τα συνελικτικά επίπεδα τοποθετούνται στην αρχή του ΣΝΔ προκειμένου να επενεργήσουν αφαιρετικά στην πολυπλοκότητα της εισόδου μειώνοντας τον διανυσματικό χώρο της, εκμεταλλευόμενα την χωρική επαναληπτικότητα συγκεκριμένων μοτίβων που εμφανίζονται στις φυσικές εικόνες.

• Η διαδικασία της χωρικής υποδειγματοληψίας εφαρμόζεται μεταξύ δυο διαδοχικών συνελικτικών επιπέδων με σκοπό την περαιτέρω μείωση του διανυσματικού χώρου των χαρακτηριστικών. Ωστόσο, πειραματικά αποδεικνύεται πως ο τρόπος που επενεργούν στην μείωση των διαστάσεων δημιουργεί προβλήματα γενίκευσης και η τάση που επικρατεί είναι η μείωση της χρήσης τους ή/και η μείωση του θορύβου που εισάγουν με την χρήση μικρότερων υποδειγματοληπτικών περιοχών.

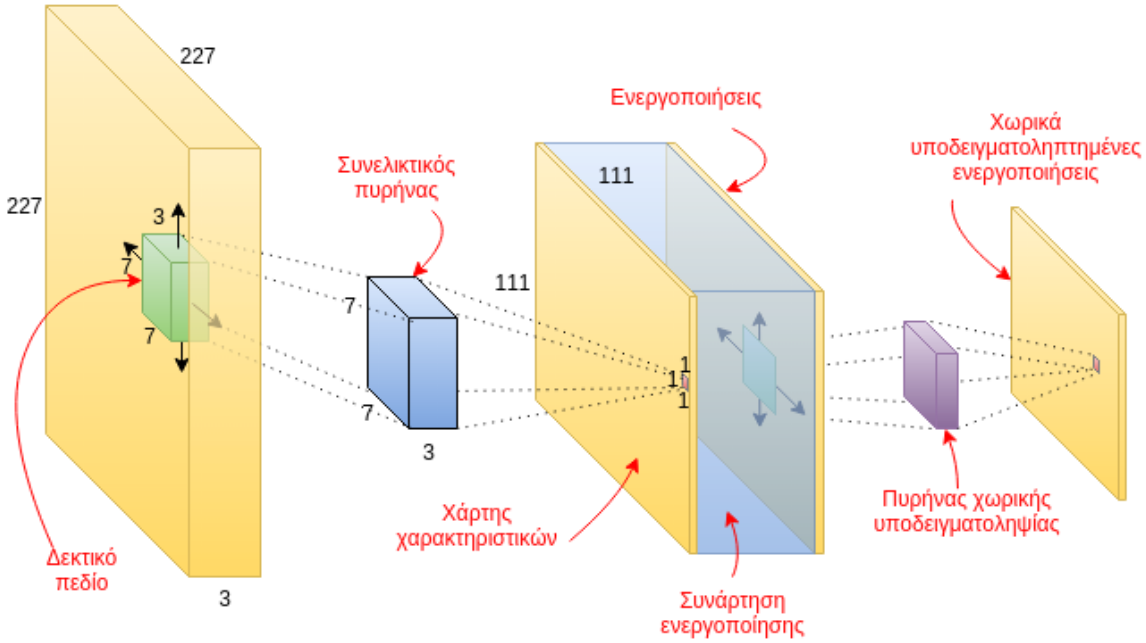
• Τα πλήρως συνδεδεμένα επίπεδα τοποθετούνται στο τέλος του ΣΝΔ προκειμένου να δημιουργήσουν το μοντέλο απόφασης (classification).

2.5.3.1. Συνελικτικά επίπεδα / Convolutional layers

Το συνελικτικό επίπεδο είναι το δομικό στοιχείο των ΣΝΔ. Οι υπό-μάθηση παράμετροι αποτελούνται από ένα πλήθος τριδιάστατων φίλτρων και προαιρετικά, κάθε φίλτρο μπορεί να περιέχει μια πόλωση (bias).

Οι διαστάσεις του κάθε φίλτρου διαχωρίζονται σε: ύψος, πλάτος και βάθος. Το φίλτρο έχει το ίδιο βάθος με το δεκτικό πεδίο στο οποίο εφαρμόζεται. Ωστόσο κατά ύψος και πλάτος το μέγεθος του φίλτρου είναι πολύ μικρότερο από το ύψος και το πλάτος της εισόδου. Κατά την διάρκεια της σάρωσης το φίλτρο σαρώνει την είσοδο κατά ύψος και πλάτος, πραγματοποιώντας 3D εσωτερικά γινόμενα με τα δεδομένα που βρίσκονται κάθε φορά στο δεκτικό του πεδίο και παράγει έναν 2D πίνακα, ο οποίος αποτελεί τον χάρτη χαρακτηριστικών της εισόδου.

Στην συνέχεια αυτός ο πίνακας τροφοδοτείται στην συνάρτηση ενεργοποίησης και εξάγονται οι ενεργοποιήσεις που αποτελούν έναν μη-γραμμικό μετασχηματισμό του χάρτη χαρακτηριστικών.



Εικόνα1:Συνελικτικό επίπεδο ακολουθούμενο από συνάρτηση ενεργοποίησης και επίπεδο υποδειγματοληψίας

Για παράδειγμα, υποθέτουμε πως έχουμε ένα συνελικτικό επίπεδο που περιέχει 100 φίλτρα με ένα στατικό μέγεθος πυρήνα μεγέθους 5x5. Το συνελικτικό επίπεδο δέχεται στην είσοδο του έναν όγκο μεγέθους [Hact X Wact X 3]. Το πλήθος των βαρών ανά συνελικτικό πυρήνα εύκολα βρίσκεται πως είναι (5x5x3=)75. Το σύνολο των βαρών στο συνελικτικό επίπεδο είναι (5x5x3x100=) 7500 και οι πολώσεις 100 (μια για κάθε φίλτρο). Η έξοδος του συγκεκριμένου συνελικτικού επιπέδου θα έχει διαστάσεις

$$Hr * Wr * 100$$

(σε κάθε d-οστή βαθμίδα βρίσκεται ο χάρτης χαρακτηριστικών του d-οστού φίλτρου).

Για την λειτουργία του πρέπει να ορισθούν οι εξής υπερπαράμετροι:

- Kernel Size: Μέγεθος του πυρήνα – φίλτρου.
- Stride: Το stride είναι η υπέρ-παράμετρος που καθορίζει πόσο πυκνή θα είναι η δειγματοληψία της εισόδου. Με άλλα λόγια, το stride καθορίζει πόσες χωρικές μονάδες (π.χ. pixels) θα μετακινείται το δεκτικό πεδίο του συνελικτικού επιπέδου οριζοντίως και καθέτως. Θέτοντας το stride ίσο με 1, το δεκτικό πεδίο θα μετακινείται κατά μια μονάδα οδηγώντας σε πυκνή σάρωση της εισόδου, το οποίο οδηγεί σε πιο ομαλές αναπαραστάσεις της εισόδου στην έξοδο, αλλά επίσης και μεγαλύτερες σε όγκο. Μεγαλύτερα stride θα προκαλούν πιο αραιή σάρωση της εισόδου, δίνοντας μικρότερες εξόδους αλλά ταυτόχρονα μπορεί να χαθεί χρήσιμη πληροφορία.
- Zero Padding: Το Padding είναι η υπέρ-παράμετρος που χρησιμοποιείται για να γεμίσει με μηδενικά το περίγραμμα της εισόδου. Αυτό το χαρακτηριστικό του γεμίσματος με μηδενικά μπορεί να βοηθήσει σημαντικά εάν θέλουμε να διατηρήσουμε τις χωρικές διαστάσεις της εισόδου και να εφαρμόσουμε ένα συγκεκριμένο μέγεθος πυρήνα για την σάρωση των χαρακτηριστικών.
- Αριθμός φίλτρων: Ο αριθμός των Μοτίβων (feature maps) που θα παραχθούν από το συνελικτικό επίπεδο. Κάθε feature map αντιστοιχεί και σε ένα διαφορετικό φίλτρο πυρήνα.

Οι διαστάσεις της εξόδου του συνελικτικού επιπέδου μπορούν να υπολογισθούν με την χρήση των υπερπαραμέτρων Μέγεθος Πυρήνα, Zero Padding και Stride βάσει των τύπων:

Υψος πίνακα αποτελεσμάτων:

$$Hr = \left(\frac{(Hact - Fdim + 2 * Pw)}{S}\right) + 1$$

Μήκος πίνακα αποτελεσμάτων:

$$Wr = \left(\frac{(Wact - Fdim + 2 * Ph)}{S}\right) + 1$$

Όπου Hact, Wact ύψος και μήκος ενεργοποιήσεων, Ph, Pw πλήθος περιμετρικών μηδενικών που εφαρμόζονται κατά μήκος και ύψος των ενεργοποιήσεων και Fdim η διάσταση του φίλτρου κοινή για ύψος και μήκος.

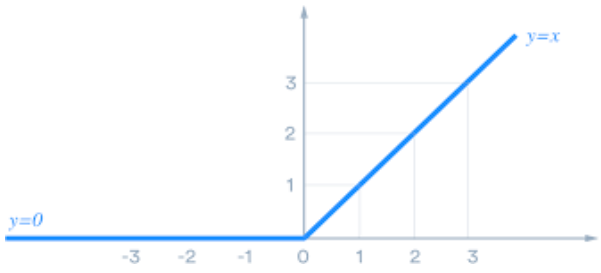
2.5.3.2. Συναρτήσεις ενεργοποίησης / Activation functions

Κάθε νευρώνας εμπεριέχει μια συνάρτηση ενεργοποίησης η οποία πραγματοποιεί έναν μη γραμμικό μετασχηματισμό στην έξοδο του. Προκειμένου να εφαρμοστεί η οπισθοδιάδοση των σφαλμάτων, μια μη-γραμμική συνάρτηση πρέπει να είναι διαφορίσιμη και γενικώς είναι μια καλή πρακτική η συνάρτηση ενεργοποίησης να είναι φραγμένη.

Μερικές ήδη γνωστές συναρτήσεις ενεργοποίησης που χρησιμοποιούνται στα TNN είναι οι σιγμοειδείς, οι συναρτήσεις υπερβολικής εφαπτομένης και οι συναρτήσεις τόξου εφαπτομένης. Ωστόσο, οι συγκεκριμένες συναρτήσεις έχουν μειωμένη απόδοση όταν εφαρμόζονται σε ένα ΣΝΔ συγκρινόμενες με την μονάδα γραμμικής ανόρθωσης (ReLU).

Η ReLU είναι η πιο δημοφιλής συνάρτηση ενεργοποίησης για τεχνητά νευρωνικά δίκτυα και συνεπώς για συνελικτικά νευρωνικά δίκτυα. Η συνάρτηση της ReLU ορίζεται ως:

$$f(x) = \max(0, x)$$



Εικόνα 2:Συνάρτηση ReLU

Μερικά από τα πλεονεκτήματα της χρήσης των ReLU είναι οι αραιές χωρικές ενεργοποιήσεις, η βιολογική συσχέτιση, η αποδοτικότητα στην οπισθοδιάδοση των σφαλμάτων και η ευκολία στους υπολογισμούς.

Άλλες συναρτήσεις ενεργοποίησης είναι:

Η σιγμοειδής, γνωστή και ως λογιστική συνάρτηση, η οποία δίνεται από την εξίσωση:

$$sig(x) = 1/(1 + e^{-x})$$

Η συνάρτηση διάδοσης υπερβολικής εφαπτομένης *(tanh)* δίνεται από την εξίσωση:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

2.5.3.3. Επίπεδα χωρικής υποδειγματοληψίας / Pooling Layers

Αυτό το είδος των επιπέδων χρησιμοποιείται για να μειώσει τους όγκους εισόδου εφαρμόζοντας μια συνάρτηση υποδειγματοληψίας κατά ύψος και πλάτος της εισόδου. Με αυτόν τον τρόπο, επιτυγχάνεται:

1. Η μείωση των διαστάσεων των παραγόμενων διανυσματικών χώρων.

2. Η μείωση της πιθανότητας υπερεκπαίδευσης.

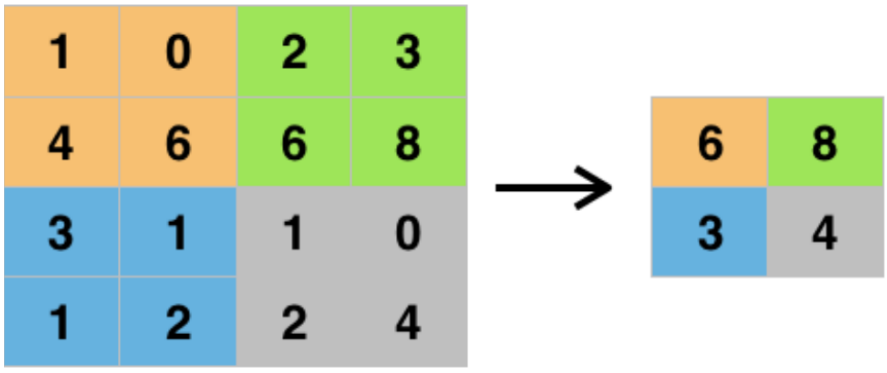
3. Μείωση του όγκου της υπό-επεξεργασίας πληροφορίας από επόμενα επίπεδα.

Η λειτουργία των επίπεδων χωρικής υποδειγματοληψίας βασίζεται στις ίδιες αρχές που διέπουν και τα συνελικτικά επίπεδα. Αυτό σημαίνει πως έχουν υπέρ-παραμέτρους για να δηλώσουν το μέγεθος του πυρήνα τους, του βήματος σάρωσης (S), και της προσθήκης μηδενικών (P).

Τα επίπεδα αυτά λειτουργούν ανεξάρτητα σε κάθε βαθμίδα του βάθους της εισόδου και την μειώνουν χωρικά, κάνοντας χρήση μιας συγκεκριμένης συνάρτησης χωρικής υποδειγματοληψίας. Γενικά, χρησιμοποιούνται διαστάσεις πυρήνα 2x2 εως 5x5, ανάλογα με τον όγκο τον δεδομένων, με την χρήση stride 2 έως 5 αντίστοιχα.

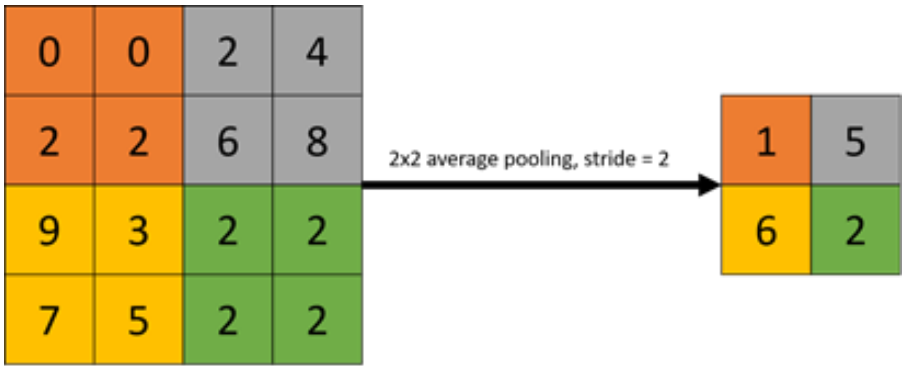
Για την επίτευξη της χωρικής υπόδειγματοληψίας χρησιμοποιούνται κυρίως οι παρακάτω συναρτήσεις :

- Συνάρτηση εξαγωγής τοπικού μεγίστου / Max Pooling:
  Η λειτουργία των επιπέδων τύπου max pooling βασίζεται στην εξαγωγή της μέγιστης τιμής (της πιο δυνατής ενεργοποίησης) που εντοπίζεται σε μια υπό-περιοχή.



Εικόνα 3: Max Pooling

- Συνάρτηση εξαγωγής τοπικού μέσου όρου / Average Pooling:
  Η λειτουργία των επιπέδων τύπου avg pooling βασίζεται στην εξαγωγή του μέσου όρου τιμών που εντοπίζονται σε μια υπό-περιοχή.



Εικόνα 4: Average Pooling

2.5.3.4. Πλήρως Συνδεδεμένα Επίπεδα / Fully Connected Layers

Τα πλήρως συνδεδεμένα επίπεδα τοποθετούνται στα τελικά επίπεδα των ΣΝΔ. Το μοντέλο απόφασης του ΣΝΔ καθορίζεται στα επίπεδα αυτά. Κάθε νευρώνας των πλήρως συνδεμένων επιπέδων διατηρεί συνδέσεις με όλους τους νευρώνες του προηγούμενου επιπέδου, όπως ισχύει και στα τυπικά τεχνητά νευρωνικά δίκτυα.

Πρόκειται για μία παραδοσιακή αρχιτεκτονική πολλών επιπέδων με νευρώνες, η οποία χρησιμοποιεί μια συνάρτηση ενεργοποίησης (συνήθως την softmax) στην έξοδό της. Οι έξοδοι των συνελικτικών και συγκεντρωτικών επιπέδων αναπαριστούν χαρακτηριστικά υψηλών στρωμάτων. Σκοπός του πλήρως συνδεδεμένου επιπέδου είναι να χρησιμοποιήσει αυτά τα χαρακτηριστικά προκειμένου να κατηγοριοποιήσει την εικόνα εισόδου σε διάφορες κλάσεις, βασιζόμενο στο σύνολο δεδομένων που χρησιμοποιήθηκαν στην εκπαίδευση. Το άθροισμα των πιθανοτήτων των κλάσεων που παράγει το πλήρως συνδεδεμένο επίπεδο είναι ένα, πράγμα που μας το επιβεβαιώνει η συνάρτηση ενεργοποίησης softmax που χρησιμοποιείται ευρέως στην έξοδο του επιπέδου αυτού. Η συνάρτηση αυτή, δέχεται σαν είσοδο ένα διάνυσμα από τυχαίες πραγματικές τιμές και τις αντιστοιχίζει σε ένα διάνυσμα με τιμές από μηδέν έως ένα και με συνολικό άθροισμα τιμών ίσο με ένα.

2.6. Γνωστά Συνελικτικά Δίκτυα

Μερικά, ήδη υπάρχοντα, γνωστά συνελικτικά δίκτυα που χρησιμοποιούνται ευρέως, και τα οποία επιλέγονται βάσει του τύπου του εκάστοτε προβλήματος που θέλουμε να επιλύσουμε, είναι:

- LeNet : Αποτελεί την πρώτη επιτυχημένη εφαρμογή Συνελικτικών Δικτύων που αναπτύχθηκε από τον Yann LeCun την δεκαετία του 1990. Η συγκεκριμένη αρχιτεκτονική χρησιμοποιήθηκε κυρίως για αναγνώριση κωδικών, ψηφίων, επιταγών, σε τραπεζικά συστήματα κυρίως, τα οποία ψηφιοποιούνταν σε 32 X 32 grayscale εικόνες – είσοδους. Το δίκτυο αυτό αποτελείται από 7 επίπεδα από όπου προκύπτει και η ικανότητά του να επεξεργάζεται εικόνες περίπου των διαστάσεων αυτών. [22]

- AlexNet : Το συγκεκριμένο δίκτυο ήταν το πρώτο το οποίο έκανε τα Συνελικτικά Δίκτυα διάσημα στο χώρο της Μηχανικής Όρασης. Είχε μία πολύ παρόμοια αρχιτεκτονική με αυτή του LeNet, ωστόσο, ήταν βαθύτερο, μεγαλύτερο και είχε πολλά συνελικτικά επίπεδα, στοιβαγμένα το ένα πάνω στο άλλο, χρησιμοποιώντας πυρήνες 11 X 11, 5 X 5 και 3 X 3, που είχε αποτελέσει μια πρωτοποριακή τεχνική. [23]
- ZFNet : Το δίκτυο αυτό αποτελεί μια βελτίωση του AlexNet, διορθώνοντας κάποιες υπερπαραμέτρους της αρχιτεκτονικής, διατηρώντας την ίδια δομή. Πιο συγκεκριμένα, επεκτάθηκε το μέγεθος του μεσαίου συνελικτικού επιπέδου, ενώ παράλληλα το βήμα και τα μέγεθη φίλτρων του πρώτου επιπέδου έγιναν μικρότερα. Το συγκεκριμένο δίκτυο απέσπασε την πρώτη θέση στον διαγωνισμό ILSVRC 2013.
- GoogleNet : Νικητής του διαγωνισμού ILSVRC 2014 αναδείχθηκε το συγκεκριμένο δίκτυο το οποίο αναπτύχθηκε από την Google. Το βασικό πλεονέκτημά του έναντι στα προηγούμενα μοντέλα, έγκειται στην δραματική μείωση των παραμέτρων του δικτύου. Πιο συγκεκριμένα, ο αριθμός των παραμέτρων μειώθηκε στα 4 εκατομμύρια σε σύγκριση με το AlexNet που είχε 60 εκατομμύρια παραμέτρους[24].
- VGGNet : Το συγκεκριμένο δίκτυο, αν και απέσπασε την δεύτερη θέση στον διαγωνισμό ILSVRC 2014, ήταν αντίστοιχα αποτελεσματικό με το GoogleNet. Η βασική συνεισφορά του ήταν στο γεγονός ότι το βάθος ενός δικτύου είναι ένα σημαντικό συστατικό για την καλή απόδοση, ωστόσο μειονέκτημά του έναντι του παραπάνω, αποτελεί το μεγάλος πλήθος παραμέτρων (138 εκατομμύρια).
- ResNet : Το δίκτυο αυτό ήταν το νικητήριο δίκτυο για τον διαγωνισμό ILSVRC 2015. Εισήγαγε για πρώτη φορά παραλειπούμενες συνδέσεις (skip connections) και εκτεταμένη χρήση κανονικοποίησης. Η αρχιτεκτονική του δικτύου δεν περιέχει πλήρως συνδεδεμένο επίπεδο στο τέλος [25].

3. Υλοποίηση Επιπέδων ΣΝΔ σε FPGA

Στην παρούσα υλοποίηση αναπτύχθηκε σε γλώσσα περιγραφής υλικού (VHDL) σύστημα accelerator για τον υπολογισμό της εξόδου της inference λειτουργίας συγκεκριμένων επιπέδων κάποιου Συνελικτικού Νευρωνικού Δικτύου (ΣΝΔ).

Ο Accelerator αποτελείται από τις παρακάτω βασικές μονάδες:

- Μονάδα Convolution, πραγματοποιεί την σάρωση των ενεργοποιήσεων με ένα σύνολο από φίλτρα παράγοντας feature maps. Η μονάδα αυτή μπορεί να χρησιμοποιηθεί σε ένα ΣΝΔ προκειμένου να εκτελεστεί το κάθε συνελικτικό επίπεδο.
- Μονάδα ReLU, της οποίας η χρήση είναι προαιρετική. Για κάθε αποτέλεσμα της συνέλιξης που θα προκύψει από την εφαρμογή φίλτρου σε μια θέση των ενεργοποιήσεων, εφόσον έχει αρνητική τιμή, αποδίδει τη τιμή μηδέν, αλλιώς διατηρεί τη τιμή εισόδου.
- Μονάδα Pooling, της οποίας η χρήση είναι επίσης προαιρετική. Λαμβάνει ως είσοδο το αποτέλεσμα της σάρωσης των ενεργοποιήσεων από ένα συγκεκριμένο φίλτρο και αποσκοπεί στην μείωση των διαστάσεων του feature map και στην πύκνωση της χρήσιμης πληροφορίας με την χρήση avg ή max pooling.
- RAM Interconnect. Διεπαφή για την επικοινωνία του core element με την εξωτερική μνήμη για την υλοποίηση write και read request σε I/O ή Burst mode. Στην παρούσα υλοποίηση η εξωτερική RAM αντιπροσωπεύεται από text αρχεία με την χρήση της βιβλιοθήκης stdio, με αποτέλεσμα το συγκεκριμένο module να είναι σχεδιασμένο κατά τέτοιο τρόπο ώστε να μπορεί να ανασύρει ή να αποθηκεύσει δεδομένα στα εν λόγω αρχεία.

Οι παράμετροι που πρέπει να δοθούν για την αρχικοποίηση του συστήματος είναι οι παρακάτω:

- Μήκος Ενεργοποιήσεων (W_{act})

- Ύψος Ενεργοποιήσεων (H_{act})
- Διάσταση Φίλτρου Πυρήνα (F_{dim}). Επιλέγονται φίλτρα με ίδιο ύψος και μήκος ενώ κάθε φίλτρο ενός συνελικτικού επιπέδου έχει κοινές διαστάσεις με όλα τα υπόλοιπα.
- Stride (S)
- Πλήθος Καναλιών (N_{Ch}) Υποχρεωτικά ο αριθμός των καναλιών ενεργοποιήσεων και φίλτρων πρέπει να ταυτίζονται.
- Πλήθος Φίλτρων (N_F). Το πλήθος των φίλτρων που θα σαρώσουν τις ενεργοποιήσεις.
- Διάσταση Pooling (dp)
- Λειτουργία Pooling (MAX ή AVG)
- ReLU enabled
- Pooling enabled

Περιορισμοί Υλοποίησης:

- Μέγεθος φίλτρου: Από 2 X 2 έως 16 X 16. Επιλέγεται ως μέγιστο μήκος-ύψος φίλτρου 16X16 καθώς στα περισσότερα ΣΝΔ χρησιμοποιούνται φίλτρα 3X3, 5X5, 11X11.
- Ύψος και μήκος ενεργοποιήσεων μέχρι 1024 X 1024
- Τα φίλτρα λαμβάνουν ακέραιες ή δεκαδικές τιμές εύρους σε δυαδική μορφή ίσου με την ακρίβεια σε bit που χρησιμοποιείται. Η τιμές αυτές μπορεί να είναι θετικές ή αρνητικές.
- Δεν υποστηρίζεται bias

Οι ενεργοποιήσεις αποτελούν εισόδους του συνελικτικου επιπέδου, που στην περίπτωση που βρισκόμαστε στο πρώτο συνελικτικό επίπεδο ταυτίζονται με τα δεδομένα του συνόλου των καναλιών της εικόνας-εισόδου του ΣΝΔ, ενώ σε μετέπειτα συνελικτικά επίπεδα ταυτίζονται με το σύνολο των feature maps (με κάθε feature map να αποτελεί πλέον ένα κανάλι), που παράχθηκαν από προηγούμενα επίπεδα. Οι ενεργοποιήσεις αποτελούν δηλαδή ένα 3d όγκο δεδομένων με συγκεκριμένο ύψος, μήκος και με βάθος ίσο με τον αριθμό των καναλιών.

3.1 Εργαλεία που χρησιμοποιήθηκαν

Η εργασία υλοποιήθηκε χρησιμοποιώντας την πλατφόρμα Xilinx ISE Design Suite – ISE Webpack 13.4. Το ISE Design Suite αποτελεί software που αναπτύχθηκε από την Xilinx για την ανάλυση και σύνθεση VHDL κωδίκων.

VHDL

Η VHDL αποτελεί γλώσσα περιγραφής υλικού που χρησιμοποιείται στον αυτοματισμό ηλεκτρονικών σχεδιάσεων για την περιγραφή ψηφιακών συστημάτων, όπως FPGA. Βασικό πλεονέκτημα της VHDL, όταν αυτή χρησιμοποιείται για σχεδίαση συστημάτων, είναι ότι επιτρέπει την περιγραφή – μοντελοποίηση και την επαλήθευση – προσομοίωση του επιθυμητού συστήματος πριν τα εργαλεία σύνθεσης μετατρέψουν την σχεδίαση σε πραγματικό υλικό. Η VHDL αποτελεί γλώσσα ροής δεδομένων. Σε αντίθεση με διαδικαστικές γλώσσες προγραμματισμού (π.χ. C), οι οποίες εκτελούνται ακολουθιακά με κάθε εντολή να ακολουθεί την προηγούμενη. Με τον όρο γλώσσα ροής δεδομένων εννοείται η αρχιτεκτονική λογισμικού που βασίζεται στην ιδέα ότι η τροποποίηση της τιμής μιας μεταβλητής πρέπει αυτόματα να επιβάλλει το υπολογισμό από τη αρχή όλων των μεταβλητών που εξαρτώνται από αυτή (concurrency).

ISIM

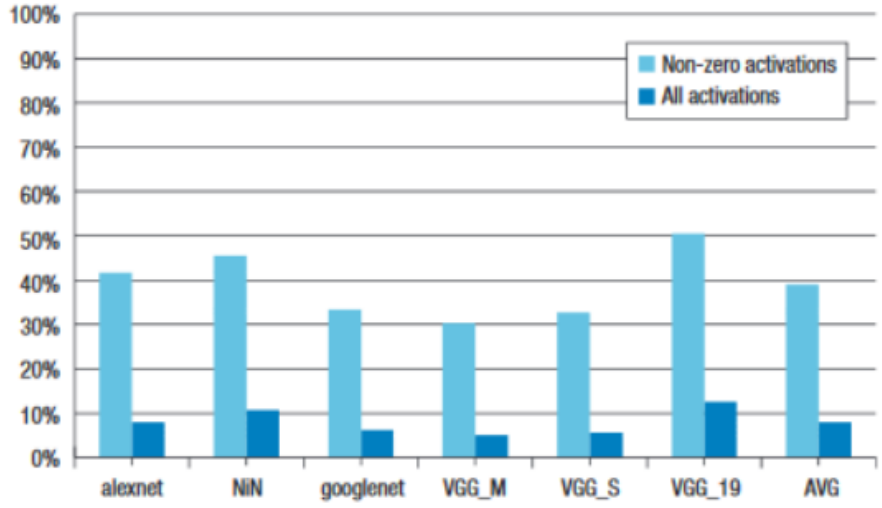
Για τις δοκιμές του υπό ανάπτυξη συστήματος, χρησιμοποιείται ο simulator ISIM. Στο ISIM μέσω testbench δίνονται ως είσοδοι κυματομορφικά σήματα και βάσει αυτών μπορεί να παρατηρηθεί η συμπεριφορά που παρουσιάζει το σύστημα καθώς και οι έξοδοι που προκύπτουν. Γενικά οι μοντελοποιήσεις που προσφέρει το ISIM χρησιμοποιούνται για:

- Επαλήθευσης της ορθότητας λογικής: Το εκάστοτε module που αναπτύσσεται παράγει τα αναμενόμενα αποτελέσματα.
- Επαλήθευση της συμπεριφορικής λειτουργίας (behavioural): Δεν προκύπτουν προβλήματα λογικής και χρονισμού στο υπό ανάπτυξη module.
- Προσομοίωση post place & route: Προσομοιώνει την συμπεριφορά που θα έχει το υπό ανάπτυξη module μετά την τοποθέτησή του στην αναδιατασσόμενη λογική της FPGA.

Το ISE Design Suite προσφέρει την λειτουργία Synthesis. Όταν γίνεται σύνθεση του VHDL κώδικα παράγεται αναφορά που περιέχει μετρικές απόδοσης του παραγόμενου design. Οι πληροφορίες της αναφοράς περιέχουν κυρίως μετρικές για το area που χρησιμοποιείται, δηλαδή το πλήθος των πόρων υλικού που απαιτούνται για τη τοποθέτηση του design στην προοριζόμενη FPGA, βασιζόμενο στους διαθέσιμους πόρους που περιέχει η τελευταία καθώς και μια εκτίμηση συχνότητας λειτουργίας. Οι τύποι των πόρων είναι οι εξής: Look-Up Tables (LUT), Flip Flops (FF), Block RAMS (BRAMs) και DSP48s.

3.2. Bit Pragmatic

Οι ενεργοποιήσεις σε επίπεδο bit που περιέχονται στα διακριτά επίπεδα ενός ΣΝΔ παρουσιάζουν μία κλίση προς το μηδέν. Σύμφωνα με έρευνες, στα περισσότερα ΣΝΔ περίπου το 8% των bits που περιέχονται στο σύνολο των ενεργοποιήσεων είναι 1, όπως φαίνεται στο παρακάτω διάγραμμα [29].

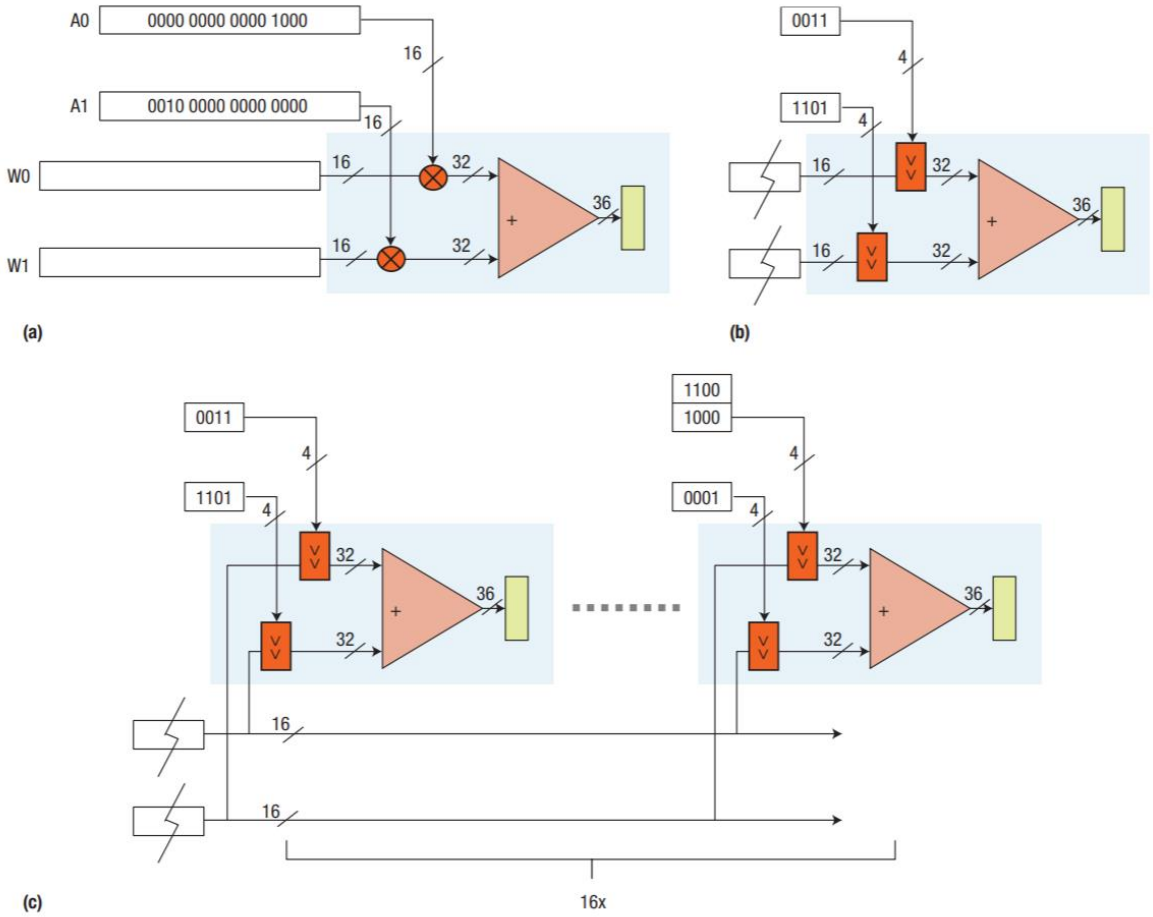


Εικόνα 5: Πλήθος άσσων στα bits του συνόλου των τιμών των ενεργοποιήσεων

Στο παραπάνω διάγραμμα, ως non-zero activations, εμφανίζεται το ποσοστό, εκ του συνόλου των ενεργοποιήσεων σε όλα τα επίπεδα του εκάστοτε ΣΝΔ, που περιέχουν τουλάχιστον έναν άσσο σε επίπεδο bit. Ως All Activations παρουσιάζονται τα ποσοστά των άσσων σε επίπεδο bit του συναντόνται στό σύνολο των non-zero activations. Ο μέσος όρος που προκύπτει, για το ποσοστό των bit άσσων των ενεργοποιήσεων στα ΣΝΔ που εξετάζονται, είναι 8%.

Καθώς οι αριθμοί οι οποίοι επεξεργαζόμαστε αναπαρίστανται σε δυαδικό σύστημα κάθε bit του πολλαπλασιαστή θα είναι είτε μηδέν ή ένα, όπου κάθε Bit μηδέν δεν συμβάλει στην παραγωγή του τελικού γινομένου. Στην περίπτωση που οι πολλαπλασιασμοί γίνονται bit προς bit με την παραδοχή ότι το 8% των bit του συνόλου των πολλαπλασιαστών είναι άσσοι, τότε το 92% των πράξεων που εκτελούνται στους επιμέρους πολλαπλασιασμούς δεν συμβάλλουν στα επιμέρους γινόμενα. Η τεχνική Bit Pragmatic προτείνει τον εντοπισμό του συνόλου των θέσεων των άσσων που βρίσκονται στον πολλαπλασιαστή. Γνωρίζοντας την θέση κάθε άσσου, για την παραγωγή του μερικού γινομένου αρκεί να επιτελέσουμε ένα πολλαπλασιασμό με πολλαπλασιαστή κάποια δύναμη του 2, αρκεί δηλαδή να ολισθήσουμε τον πολλαπλασιαστέο θέσεις ίσες με την θέση στην οποία εντοπίσθηκε ο συγκεκριμένος άσσος στον πολλαπλασιαστή. Με αυτό τον τρόπο η μονάδα πολλαπλασιασμού αντικαθίσταται με ολισθητές. Στην περίπτωση που ο πολλαπλασιαστής περιέχει έναν ακριβώς άσσο, το γινόμενο παράγεται σε έναν κύκλο, ωστόσο στην περίπτωση που ο πολλαπλασιαστής περιέχει περισσότερους από έναν άσσους, το γινόμενο παράγεται σε κύκλους ίσους με το πλήθος των άσσων. Προκειμένου να αποφευχθούν καθυστερήσεις που οφείλονται στην ύπαρξη περισσότερων του ενός άσσου στον πολλαπλασιαστή χρησιμοποιείται παραλληλισμός.

Η τεχνική Bit Pragmatic προτάθηκε από την ομάδα του Καθηγητή Ανδρέα Μοσχοβού (Πανεπιστήμιο Τορόντο) για την βελτίωση της λειτουργίας του Accelerator για ΣΝΔ DaDianNao, ο οποίος επεξεργάζεται με τη χρήση Bit Parallel ζεύγη τιμών ενεργοποιήσεων και φίλτρου [29], [31]. Στην παρακάτω εικόνα, στο (α) φαίνεται το block διάγραμμα μιας μονάδας MAC με χρήση πολλαπλασιαστή Bit Parallel ενώ στο (b) με χρήση Bit Pragmatic πολλαπλασιαστή. Το (c) αποτελείται από 16 αντίγραφα του (b) προκειμένου στην χειρότερη περίπτωση – όπου όλα τα bit είναι 1 – να επιτυγχάνουμε ίδιες αποδόσεις με την τεχνική Bit Parallel.



Εικόνα 6: Block διάγραμμα Bit Pragmatic

Στην υλοποίηση που ακολουθεί, το σύνολο των πολλαπλασιασμών πραγματοποιείται με την χρήση της τεχνικής αυτής, με την διαφορά ότι σε κάθε πολλαπλασιασμό που επιτελείται, πολλαπλασιαστή αποτελεί η τιμή, εκ των δύο που πρέπει να πολλαπλασιαστούν, με το μικρότερο πλήθος bits άσσους.

3.3. Σχεδίαση Μονάδας Convolution

Στόχος της παρούσας ενότητας είναι η ανάλυση της λειτουργικότητας της Convolution μονάδας που έχει αναπτυχθεί. Καθώς η εργασία περιορίζεται στο inference κομμάτι, απαραίτητα για την λειτουργία είναι τόσο οι ενεργοποιήσεις όσο και το σύνολο των φίλτρων να είναι αποθηκευμένα σε εξωτερική RAM, σε

κατάλληλο format. Στην υλοποίηση που θα περιγραφεί, τίθεται ακρίβεια τιμών 8 bit (απόλυτη τιμή) με ένα επιπλέον bit που προορίζεται για το πρόσημο.

Το συνελικτικό επίπεδο χρησιμοποιείται επαναλαμβανόμενα εντός ενός ΣΝΔ, λαμβάνοντας διαφορετικές παραμέτρους, ενεργοποιήσεις και φίλτρα κάθε φορά.

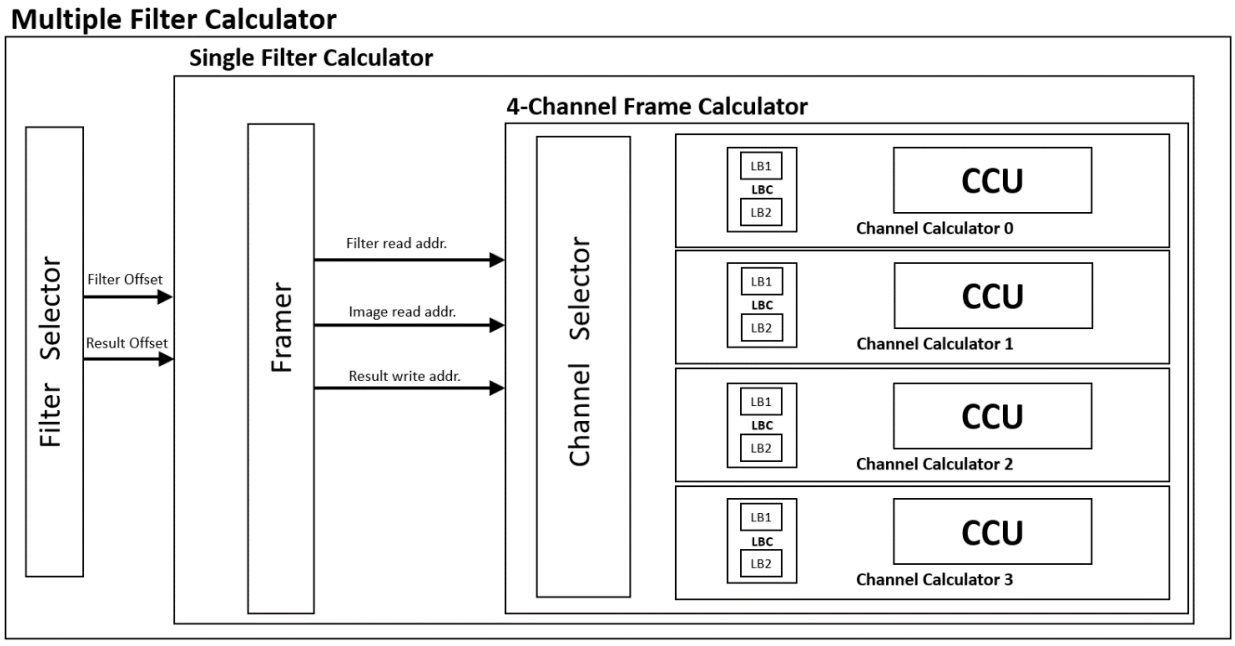
Το συνελικτικό επίπεδο πρέπει να σαρώσει τις ενεργοποιήσεις με το σύνολο των φίλτρων διαδοχικά. Για κάθε εφαρμογή του φίλτρου πάνω στην περιοχή των ενεργοποιήσεων που καλύπτει, όλες οι τιμές, όλων των καναλιών του φίλτρου πρέπει να πολλαπλασιαστούν με τις αντίστοιχες τιμές της περιοχής των ενεργοποιήσεων. Το σύνολο των γινομένων – του συνόλου των καναλιών – αθροίζονται και προκύπτει μια τιμή η οποία καταλαμβάνει μια θέση του διδιάστατου πίνακα αποτελεσμάτων (feature map) για το συγκεκριμένο φίλτρο. Το πλήθος των φίλτρων που περιέχονται σε ένα συνελικτικό επίπεδο καθορίζουν τον αριθμό των feature maps που θα παραχθούν στο επίπεδο αυτό. Το σύνολο των feature maps θα αποτελέσουν τις ενεργοποιήσεις του επόμενου συνελικτικού επιπέδου, με κάθε feature map πλέον να αποτελεί ένα κανάλι (τα feature maps προαιρετικά θα επεξεργαστούν από την επιλεγμένη συνάρτηση ενεργοποίησης και το επίπεδο χωρικής υποδειγματοληψίας).

Η ολίσθηση του φίλτρου πάνω στις ενεργοποιήσεις γίνεται βάσει της παραμέτρου stride, η οποία και καθορίζει τις θέσεις όπου μπορούμε να εφαρμόσουμε το εκάστοτε φίλτρο. Προφανώς, το stride παραμένει αμετάβλητο για το σύνολο των φίλτρων κατά την λειτουργία του συνελικτικού επιπέδου, καθώς το σύνολο των feature maps που θα παραχθούν πρέπει να έχουν τις ίδιες διαστάσεις. Θέτοντας ως stride την τιμή 1, το εκάστοτε φίλτρο τοποθετείται στην πάνω αριστερά θέση των ενεργοποιήσεων (0,0). Το φίλτρο μετακινείται στον οριζόντιο άξονα κατά μια θέση κάθε φορά (η κάθε θέση περιέχει πολλαπλές τιμές ίσες με τον αριθμό των καναλιών), και η οριζόντια μετακίνηση ολοκληρώνεται μόλις η τελευταία οριζόντια θέση του φίλτρου φθάσει στην τελευταία θέση των ενεργοποιήσεων. Στην συνέχεια το φίλτρο τοποθετείται ξανά στην αρχή του οριζόντιου άξονα και μετακινείται στον κάθετο άξονα κατά μια θέση. Η διαδικασία επαναλαμβάνεται έως ότου η τελευταία κάθετη και οριζόντια θέση του φίλτρου να συμπίπτει με την τελευταία κάθετη και οριζόντια θέση των ενεργοποιήσεων. Για stride μεγαλύτερο του 1 η διαδικασία που ακολουθείται είναι παρόμοια με την παραπάνω, με την διαφορά ότι αντί να

προχωράμε μεταξύ διαδοχικών εφαρμογών του φίλτρου κατά μια θέση, προχωράμε κατά stride θέσεις τόσο στον οριζόντιο όσο και στον κάθετο άξονα.

Η αρχιτεκτονική που προτείνεται σε top-down επιγραμματική ανάλυση είναι η εξής:

- Multiple Filter Calculator/Convolution: Σάρωση των ενεργοποιήσεων με το σύνολο των φίλτρων που δίνονται και παραγωγή διδιάστατων feature maps αριθμού ίσου με το πλήθος των φίλτρων που εφαρμόστηκαν.
- Single Filter Calculator: Σάρωση των ενεργοποιήσεων με ένα φίλτρο και παραγωγή ενός μοναδικού feature map.
- 4-Channel Frame Calculator: Υπολογισμός συνέλιξης για την εφαρμογή ενός φίλτρου πάνω σε μια περιοχή των ενεργοποιήσεων. Από το σύνολο των καναλιών υπολογίζεται η τιμή της συνέλιξης για μια τετράδα καναλιών.
- Channel Calculator: Υπολογισμός τιμής συνέλιξης, για την εφαρμογή ενός φίλτρου πάνω σε μια περιοχή των ενεργοποιήσεων, ενός καναλιού ενεργοποιήσεων-φίλτρου.



Εικόνα 7: Block διάγραμμα της μονάδας Convolution

3.3.1. Μορφή αποθηκευμένων δεδομένων σε εξωτερική RAM

Προκειμένου το σύστημα να μπορεί να λειτουργήσει, πρέπει να είναι αποθηκευμένα σε εξωτερική RAM τόσο οι ενεργοποιήσεις όσο και το σύνολο των φίλτρων που αφορούν την Convolution μονάδα. Προκειμένου να λάβουμε δεδομένα από την εξωτερική RAM πρέπει να πραγματοποιήσουμε burst request παρέχοντας ταυτόχρονα μια διεύθυνση και ένα αριθμό στοιχείων που πρέπει να επιστραφούν. Σε επόμενους κύκλους η RAM θα επιστρέψει σε Burst τιμές διαδοχικών θέσεων μνήμης (μεγέθους bus width), συμπεριλαμβανομένου των στοιχείων που περιέχονται στην διεύθυνση που δόθηκε, αριθμού ίσου με το burst size.

Στο πρώτο συνελικτικό επίπεδο, εφόσον οι ενεργοποιήσεις ταυτίζονται με την εικόνα είσοδο, κάθε τιμή του κάθε καναλιού μπορεί να λάβει τιμές από 0 έως 255, δηλαδή απαιτούνται $\log_2 255 = 8$ bits. Τα 8 bits αυτά μπορούν να αναπαραστούν τόσο ακέραιη τιμή όσο και δεκαδική τιμή εύρους 0 έως 1 εφόσον έχει προηγηθεί κατάλληλη αναγωγή. Σε επόμενα συνελικτικά επίπεδα το εύρος των τιμών των ενεργοποιήσεων καθορίζεται από την ακρίβεια του ΣΝΔ. Επιλέγονται φίλτρα με εύρος από -255 έως 255 (ή -1 έως 1 με χρήση αναγωγής), καθώς η ακρίβεια του δικτύου που θα επεξεργαστεί θα είναι 9 bit, τα οποία απαιτούν 8 bits για την αποθήκευση του απόλυτου αριθμού και 1 bit για την αποθήκευση του πρόσημου. Θεωρούμε ότι χρησιμοποιείται εξωτερική RAM με bus width 64 Bits, σε κάθε θέση της οποίας χωράνε τέσσερεις τιμές των ενεργοποιήσεων και των φίλτρων. Τόσο οι ενεργοποιήσεις όσο και τα φίλτρα, τα οποία αποτελούνται από ένα ή περισσότερα κανάλια, στην αποθήκευσή τους στην εξωτερική RAM, χωρίζονται σε περιοχές πλήθους ίσου με τον αριθμό των καναλιών, όπου κάθε περιοχή περιέχει τις τιμές ενός καναλιού. Ανάλογα με τις διαστάσεις (μήκος - ύψος) των ενεργοποιήσεων και του φίλτρου μπορούν να ορισθούν offsets βάσει των οποίων μπορούμε να έχουμε πρόσβαση στα διαφορετικά κανάλια τους στην RAM. Ξεκινώντας την αρίθμηση των καναλιών από 0, για τον υπολογισμό των offsets του ν-οστού καναλιού ενεργοποιήσεων και φίλτρου χρησιμοποιούνται οι εξής τύποι:

$$Filter\ Channel\ Offset = NumOfCurrentChannel * FilterWidth * FilterHeight$$

$$Activations\ Channel\ Offset = ActivationsWidth * ActivationsHeight$$

Στην παρούσα σχεδίαση το σύστημα δέχεται ενεργοποιήσεις, που αντιστοιχούν στα δεδομένα μίας εικόνας ή στα feature maps προηγούμενου επιπέδου, και πολλαπλά φίλτρα. Προκειμένου να έχουμε πρόσβαση σε διαφορετικά φίλτρα στη RAM πρέπει να ορισθεί άλλο ένα offset το οποίο θα δείχνει στο τρέχον φίλτρο. Ξεκινάμε την αρίθμηση των φίλτρων από 0. Ο τύπος του συγκεκριμένου offset είναι:

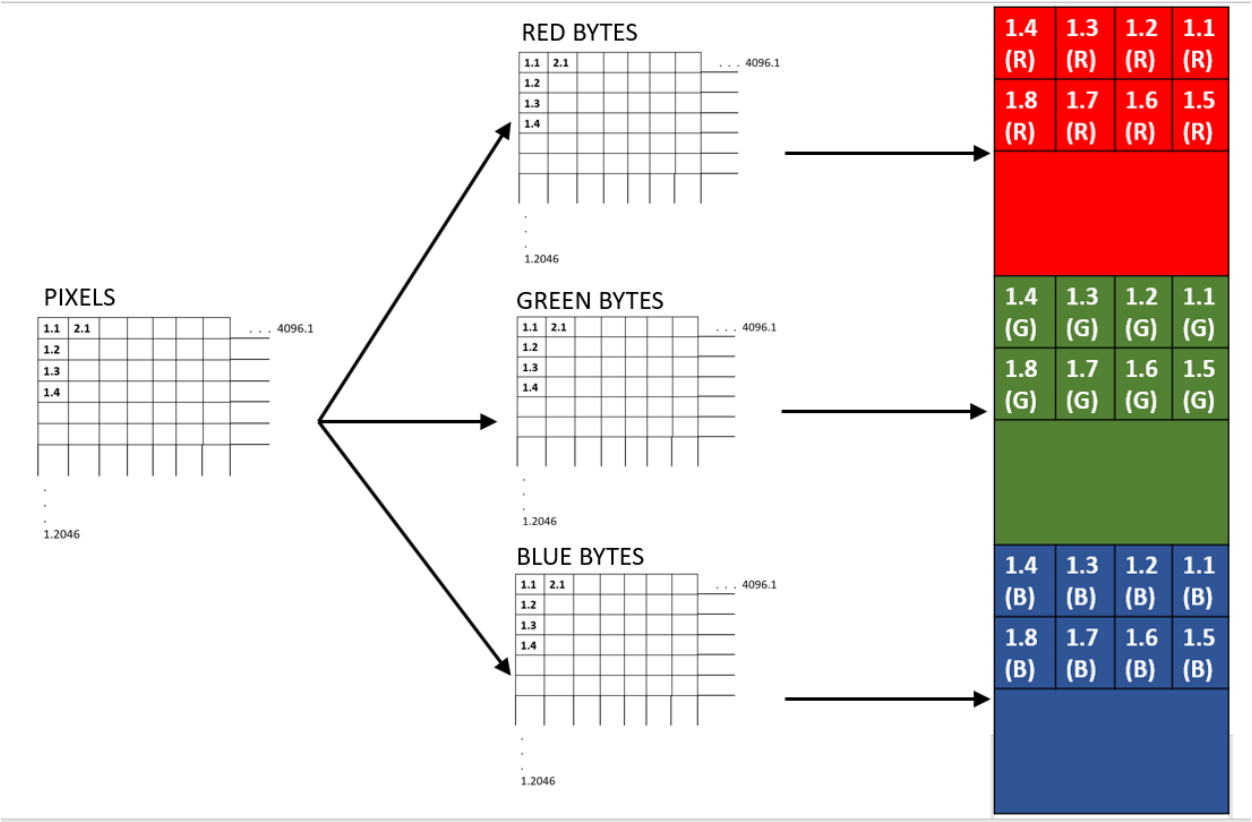
$$FilterOffset = NumOfChannels * FilterWidth * FilterHight$$

Τα offsets προστίθενται στην αρχική διεύθυνση αποθήκευσης ενεργοποιήσεων και φίλτρου με σκοπό την παραγωγή της αρχικής διεύθυνσης των δεδομένων που θέλουμε να ανασύρουμε.

Καθώς η σάρωση των ενεργοποιήσεων γίνεται γραμμή προς γραμμή και επειδή συνήθως το μήκος της εικόνας / feature maps είναι αρκετά μεγαλύτερο από το ύψος τους, είναι προτιμότερο η αποθήκευση κάθε καναλιού των ενεργοποιήσεων και των φίλτρων να γίνονται βάσει των στηλών τους.

Στην περιοχή της εξωτερικής RAM όπου βρίσκονται αποθηκευμένες οι ενεργοποιήσεις πρέπει πρώτα να είναι σειριακά αποθηκευμένη η πρώτη στήλη του πρώτου καναλιού των ενεργοποιήσεων, στη συνέχεια η επόμενη στήλη του πρώτου καναλιού κ.ο.κ μέχρι να εξαντληθούν οι στήλες του. Επόμενα κανάλια - εφόσον υπάρχουν – αποθηκεύονται κατά τον ίδιο τρόπο διαδοχικά.

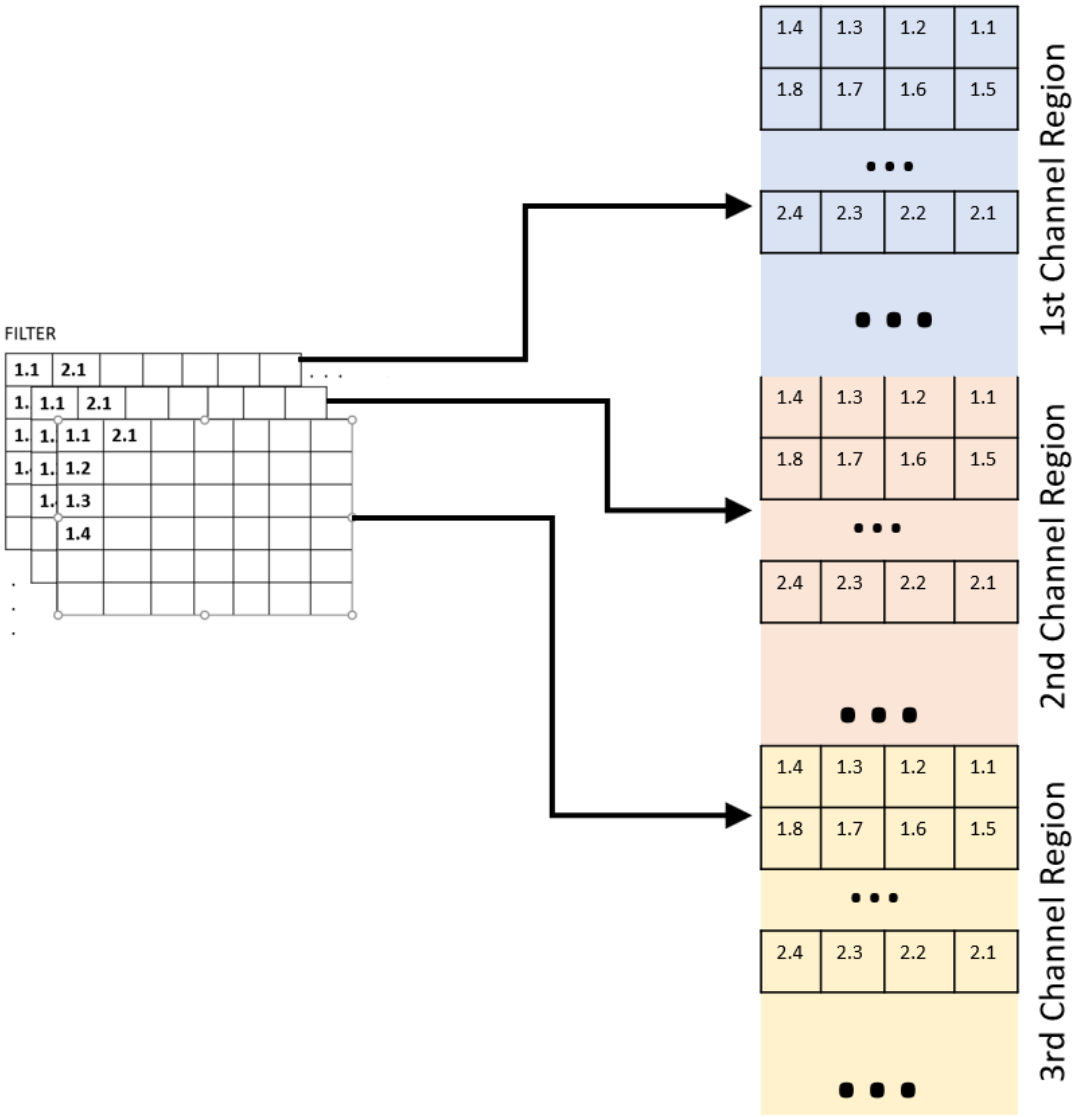
Ο τρόπος με τον οποίο αποθηκεύεται μια εικόνα τριών καναλιών (RBG) στην εξωτερική RAM φαίνεται στην παρακάτω εικόνα.



Εικόνα 8: Διάταξη των δεδομένων εικόνας RGB στη RAM

Με αντίστοιχο τρόπο γίνεται η αποθήκευση του πρώτου φίλτρου (σειριακά αποθηκευμένη η πρώτη στήλη του πρώτου καναλιού του φίλτρου, στη συνέχεια η επόμενη στήλη του πρώτου καναλιού κ.ο.κ. μέχρι να εξαντληθούν οι στήλες του). Ακολουθούν διαδοχικά τα υπόλοιπα κανάλια του φίλτρου και έπειτα κατά ίδιο τρόπο βρίσκονται αποθηκευμένα τα υπόλοιπα φίλτρα.

Παρακάτω φαίνεται ο τρόπος με τον οποίον αποθηκεύεται το σύνολο των δεδομένων ενός φίλτρου τριών καναλιών στην RAM.

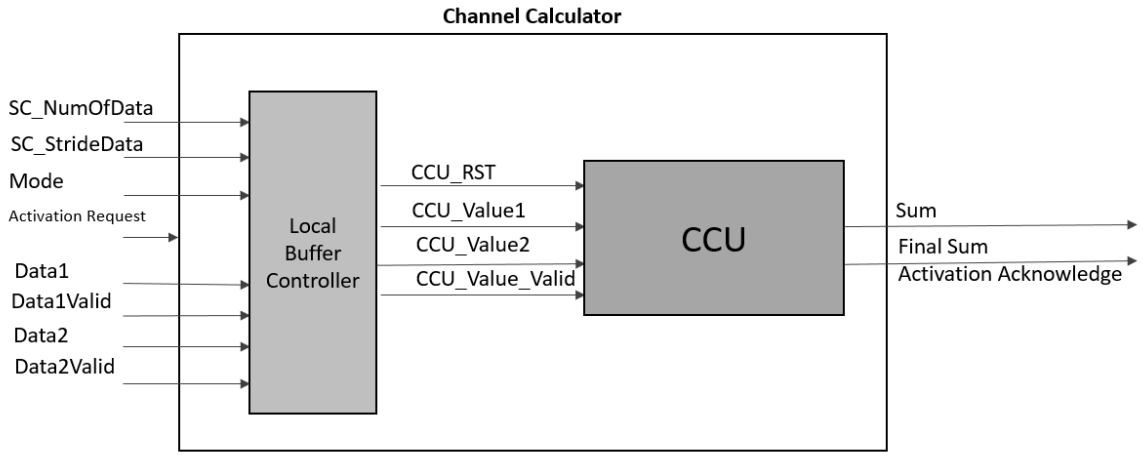


Εικόνα 9: Διάταξη των δεδομένων φίλτρου 3 καναλιών στη RAM

3.3.2. Channel Calculator

Σκοπός του Channel Calculator, κατά την εφαρμογή του φίλτρου επάνω σε μία περιοχή των ενεργοποιήσεων, είναι να υπολογίζει την τιμή της συνέλιξης μεταξύ φίλτρου και ενεργοποιήσεων για ένα channel αυτών, πραγματοποιώντας όσο το δυνατόν λιγότερα αιτήματα προς την εξωτερική RAM για ανάσυρση δεδομένων.

Ο Channel Calculator αποτελείται από δύο modules: Local Buffer Controller (LBC) και Convolution Calculation Unit (CCU). Με την ενεργοποίηση του Channel Calculator ενεργοποιείται πρώτος ο LBC, αποθηκεύοντας τοπικά μη υπάρχουσες τιμές ενός καναλιού ενεργοποιήσεων και φίλτρου. Μόλις σχηματισθούν ζεύγη τιμών ενεργοποιήσεων – φίλτρου στους Local Buffers του, προωθούνται στην CCU, χωρίς ο LBC να έχει λάβει απαραίτητα το σύνολο των τιμών. Η CCU περιμένει να λάβει πλήθος ζευγών ίσο με τον αριθμό των τιμών που περιέχονται σε ένα κανάλι του φίλτρου. Μόλις λάβει το σύνολο αυτών των τιμών και υπολογίσει την συνέλιξη τους, ο Channel Calculator απενεργοποιείται, διατηρώντας στην έξοδό του την τιμή της συνέλιξης, και μπορεί να επαναχρησιμοποιηθεί.



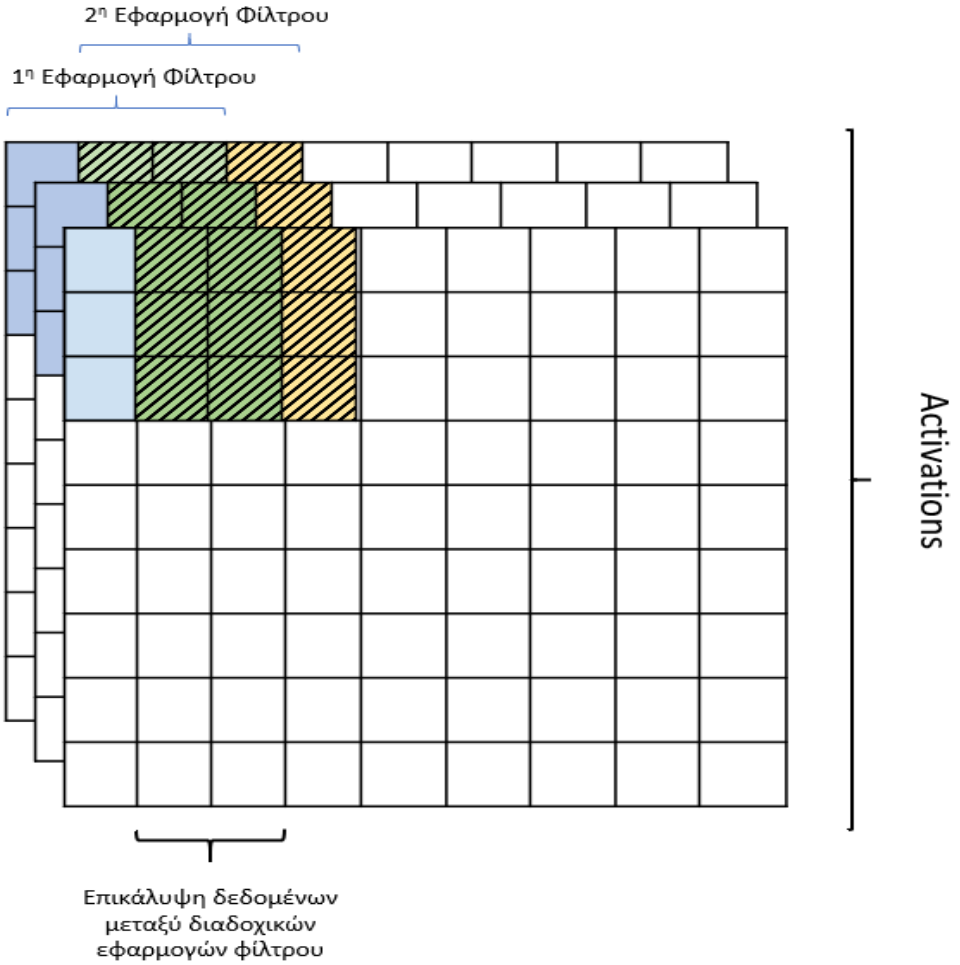
Εικόνα 10: Block διάγραμμα του Channel Calculator

3.3.2.1. Local Buffer Controller (LBC):

Στο συγκεκριμένο module χρησιμοποιούνται δύο Local Buffers (DPRAM που λειτουργούν ως circular buffers), διαστάσεων 256 θέσεων με 32 bit κάθε θέση, ώστε να μπορούν να αποθηκευτούν το σύνολο των τιμών ενός καναλιού σε περίπτωση που το φίλτρο έχει μέγιστο μήκος και ύψος (16 X 16). Ο πρώτος περιέχει τις τιμές ενός καναλιού των ενεργοποιήσεων και ο δεύτερος τις τιμές ενός καναλιού του φίλτρου που πρέπει να προωθηθούν σε αντιστοιχία στην CCU. Η χρησιμότητα των Local Buffers έγκειται στην μείωση όσο το δυνατόν περισσότερο των αιτημάτων προς την RAM. Κατά την σάρωση των ενεργοποιήσεων από ένα φίλτρο τα δεδομένα του φίλτρου παραμένουν αμετάβλητα, επομένως υπάρχει ανάγκη φόρτωσης των

τιμών αυτού στον αντίστοιχο Local Buffer μόνο μία φορά, κατά την πρώτη εφαρμογή του. Όσον αφορά τα δεδομένα των ενεργοποιήσεων, εφόσον υπάρχει επικάλυψη κατά την μετακίνηση του φίλτρου, πολλά από αυτά επαναχρησιμοποιούνται κατά την μετακίνηση του φίλτρου στον οριζόντιο άξονα. Οπότε αντί να ανασύρουμε δεδομένα εκ νέου από τις εξωτερικές RAM αποθηκεύονται τοπικά και ανασύρουμε μόνο τα δεδομένα που δεν έχουμε λάβει μέχρι στιγμής.

Διατηρεί, λοιπόν, δεδομένα τα οποία θα επαναχρησιμοποιηθούν, δέχεται νέα δεδομένα και τελικά τροφοδοτεί την CCU με το σύνολο των δεδομένων που πρέπει να λάβει, είτε αυτά προϋπήρχαν στους Local Buffers ή είναι νέα.

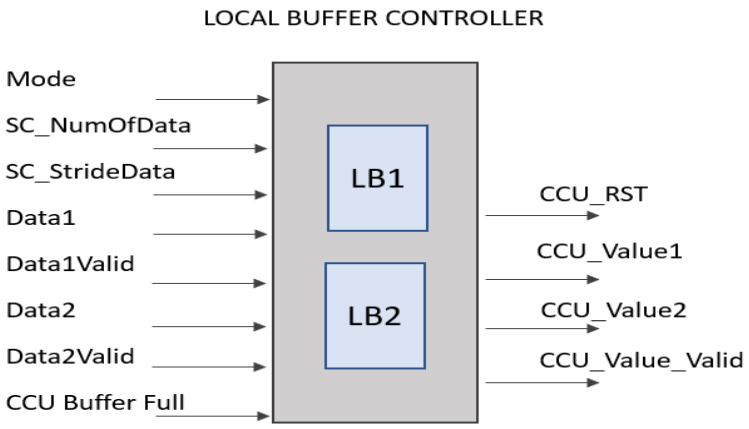


Εικόνα 11: Επικάλυψη Δεδομένων κατά την οριζόντια μετακίνηση του φίλτρου.

Ο LBC ανάλογα με το που εφαρμόζεται το φίλτρο επάνω στις ενεργοποιήσεις πρέπει να λειτουργήσει με διαφορετικούς τρόπους, θέτοντας διαφορετικό αριθμό στοιχείων που θα πρέπει να δεχτούν και έπειτα να τροφοδοτήσουν οι Local Buffers. Ο ρυθμός αποθήκευσης των δεδομένων στους δύο Local Buffers μπορεί να είναι διαφορετικός, ωστόσο ο ρυθμός αποστολής των δεδομένων στην CCU πρέπει να είναι ίδιος. Προκειμένου να επιτευχθεί κάτι τέτοιο χρησιμοποιούνται δύο διαφορετικοί Buffer Counters που δείχνουν τον αριθμό των χρήσιμων δεδομένων που περιέχονται στους Local Buffers. Εφόσον και οι δύο Buffer Counters είναι θετικοί τότε υπάρχουν δεδομένα που πρέπει να πολλαπλασιαστούν μεταξύ τους, δηλαδή πρέπει να προωθηθούν στην CCU. Προφανώς, μόλις προωθηθούν οι τιμές στην CCU, η τιμή των buffer Counters μειώνεται.

Συγκεκριμένα για ένα κανάλι φίλτρου - ενεργοποιήσεων:

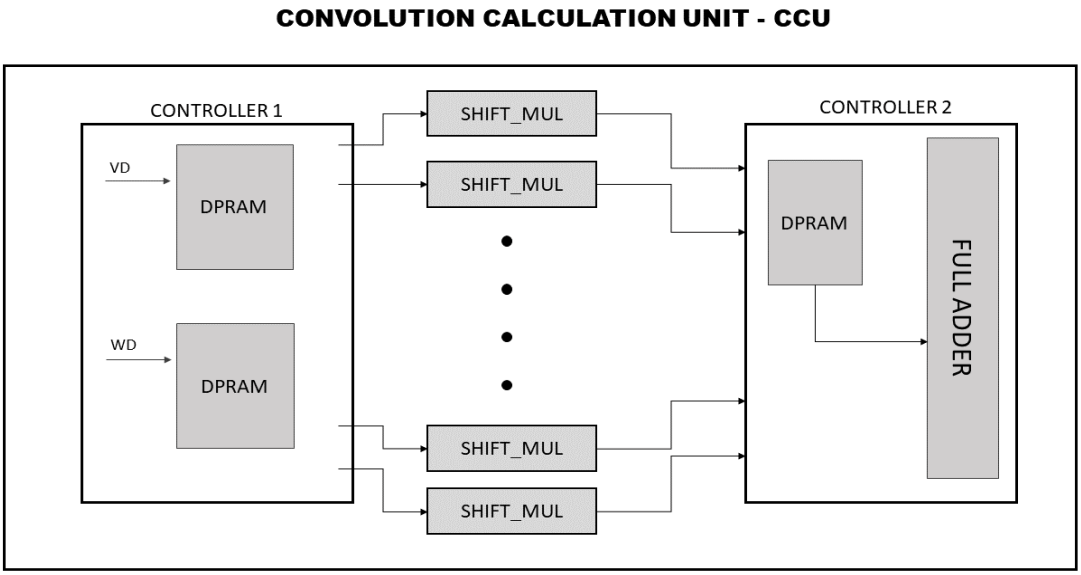
- Όταν το εκάστοτε φίλτρο τοποθετείται για πρώτη φορά επάνω στις ενεργοποιήσεις οι δύο Local Buffers δεν περιέχουν χρήσιμα δεδομένα. Στην περίπτωση αυτή λοιπόν, σε επόμενους κύκλους οι Local Buffers πρέπει να δεχθούν αριθμό τιμών ίσο με τον πλήθος τιμών που περιέχει ένα κανάλι του φίλτρου, από μια περιοχή των ενεργοποιήσεων και του φίλτρου αντίστοιχα.
- Όταν το εκάστοτε φίλτρο τοποθετείται για δεύτερη ή μεταγενέστερη φορά πάνω σε μία γραμμή των ενεργοποιήσεων, ο Local Buffer του φίλτρου περιέχει όλες τις απαιτούμενες τιμές, οπότε δεν περιμένει να δεχτεί νέα δεδομένα, ενώ ο Local Buffer των ενεργοποιήσεων πρέπει να απορρίψει τις stride πρώτες στήλες και να δεχθεί τις stride επόμενες (δηλαδή stride Χ ΠΥΦ δεδομένα)
- Όταν το φίλτρο εφαρμόζεται στην πρώτη θέση μιας γραμμής των ενεργοποιήσεων (εκτός της πρώτης), δηλαδή πραγματοποιείται κατακόρυφη κίνηση του φίλτρου,ο Local Buffer του φίλτρου δεν πρέπει να δεχτεί νέα δεδομένα ενώ ο Local Buffer των ενεργοποιήσεων πρέπει να δεχτεί το σύνολο των τιμών της περιοχής που καλύπτει το φίλτρο.



Εικόνα 12: Block διάγραμμα του Local Buffer Controller

3.3.2.2. Convolution Calculation Unit (CCU)

Για κάθε τοποθέτηση φίλτρου η CCU πρέπει να δεχτεί ζεύγη τιμών (value1, value2), αριθμού ίσου με το πλήθος των τιμών που περιέχονται σε ένα κανάλι του φίλτρου, τα οποία πολλαπλασιάζονται μεταξύ τους και το σύνολο των γινομένων αθροίζεται. Προκύπτει έτσι, από μεταβλητό πλήθος ζευγών τιμών, ένα αποτέλεσμα συνέλιξης.



Εικόνα 13: Block διάγραμμα του CCU

Για την ορθή λειτουργία της, η CCU αποτελείται από τα εξής modules:

3.3.2.2.1. CCU Input Controller

Καθώς ο ρυθμός αποστολής δεδομένων προς την CCU είναι διαφορετικός από τον ρυθμό επεξεργασίας των αντίστοιχων δεδομένων, υπάρχει ανάγκη τοπικής αποθήκευσης των δεδομένων αυτών. Οι ενεργοποιήσεις όσο και οι τιμές του φίλτρου που δέχεται η CCU πρέπει να αποθηκευτούν σε δύο μονοδιάστατους πίνακες. Δεν μας απασχολεί η σειρά καταχώρησης των τιμών, μας ενδιαφέρει όμως να βρίσκονται σε αντιστοιχία οι τιμές που πρέπει να πολλαπλασιαστούν μεταξύ τους στους δύο πίνακες. Στην σχεδίασή μας κάθε μονοδιάστατος πίνακας αντιπροσωπεύεται από μία DPRAM (Dual Port RAM). Το port-A της DPRAM θα χρησιμοποιηθεί για την εγγραφή δεδομένων, ενώ το port-B για την ανάγνωση. Κάθε DPRAM θα λειτουργεί ως circular buffer. Τα δεδομένα εισέρχονται ταυτόχρονα και στους δύο circular buffers ώστε να τηρηθεί η αντιστοιχία. Το μέγεθος το οποίο επιλέγεται για τις δύο DPRAM είναι 256 θέσεις των 32 bit,ώστε να είναι σε θέση να μπορούν να διατηρήσουν το σύνολο των τιμών ενός καναλιού της περιοχής των ενεργοποιήσεων και του φίλτρου (με δεδομένο ότι το μέγιστο φίλτρο έχει μήκος και ύψος 16 X 16).

Το module CCU Input Controller είναι υπεύθυνο για την ανάθεση τιμών σε ένα Shift_Mul module από το σύνολο των διαθέσιμων (το Shift_Mul module επιλέγεται βάσει round – robin). Για την ορθή ανάθεση ο συγκεκριμένος controller έχει επίγνωση του ποια Shift_Mul βρίσκονται σε κατάσταση λειτουργίας ή αδράνειας.

3.3.2.2.2. CCU Shift Mul

Ζητείται ο πολλαπλασιασμός δύο ακέραιων τιμών 8 bit η κάθε μία ή δύο πραγματικών τιμών επίσης 8 bit κάθε μία (MSB ακέραιο μέρος 0 ή 1, τα 7 LSB δεκαδικό μέρος με το σύνολο των τιμών που μπορούν να απαρασταθούν σε 7 bit), value1 και value2. Ο πολλαπλασιασμός δύο 8 bit αριθμών παράγει μέχρι έναν 16 bit αριθμό και θα πραγματοποιηθεί με την χρήση ολισθήσεων, καθώς ακολουθείται η τεχνική Bit Pragmatic. Συγκεκριμένα, αναζητούμε στον πολλαπλασιαστή τις θέσεις που περιέχουν άσσο και για κάθε άσσο που συναντούμε δημιουργούμε ένα partial

product, το οποίο είναι ο πολλαπλασιαστέος ολισθημένος αριστερά κατά την θέση του άσσου στην τιμή. Στην συνέχεια το σύνολο των partial products που προέκυψαν, αθροίζονται σειριακά. Οι κύκλοι που απαιτούνται για τον πολλαπλασιασμό των δύο 8 bits τιμών ταυτίζονται με τον αριθμό των άσσων που περιέχει η μικρότερη σε πλήθος άσσων τιμή. Η χειρότερη περίπτωση είναι η λειτουργία του συγκεκριμένου module να διαρκέσει 8 κύκλους (στην περίπτωση που και οι δύο τιμές περιέχουν μόνο άσσους «11111111»), στους οποίους το module δεν μπορεί να δεχθεί και να επεξεργασθεί νέες τιμές. Επειδή η ροή των δεδομένων στην CCU είναι συνεχής και προκειμένου να αποφευχθούν καθυστερήσεις χρησιμοποιούνται 8 modules τα οποία λειτουργούν παράλληλα. Η ανάθεση τιμών στο κάθε module γίνεται με την τεχνική round-robin. Το Shift Mul δέχεται 32 bit αριθμούς, από τους οποίους πολλαπλασιάζει τα 8 LSB τους. Ο πολλαπλασιασμός αυτός παράγει γινόμενο των 16 bit. Στην περίπτωση που πολλαπλασιάζουμε ακέραιους, από το γινόμενο απομονώνονται τα 8 LSB τα οποία και τοποθετούνται στα 8 LBS του σήματος αποτελέσματος. Στην περίπτωση που πολλαπλασιάζουμε δεκαδικούς αριθμούς, από τα 16 bit του γινομένου, απομονώνουμε το δεύτερο MSB ως ακέραιο μέρος και το δεκαδικό μέρος συγκροτείται από το $3^{ο}$ έως το $9^{ο}$ MSB του γινομένου. Στα 8 LSB του σήματος αποτελέσματος θα τοποθετηθούν τα Bit που απομονώθηκαν από το γινόμενο. Οι τιμές των ενεργοποιήσεων είναι σε όλα τα επίπεδα του ΣΝΔ θετικές (είτε γιατί προέρχονται από το πρώτο επίπεδο του ΣΝΔ όπου γίνεται ανάλυση της εικόνας σε RGB (τιμές 0 – 255 ή 0 - 1), ή καθώς προέρχονται από προηγούμενο επίπεδο στο οποίο έχει μεσολαβήσει ReLU που έχει αντικαταστήσει τις αρνητικές τιμές με 0. Ωστόσο, προκειμένου να καλυφθεί η περίπτωση που τα ReLU επίπεδα είναι απενεργοποιημένα, θεωρούμε ότι και οι ενεργοποιήσεις μπορούν να διατηρούν αρνητικές τιμές. Οι τιμές του φίλτρου δεν είναι υποχρεωτικά θετικές καθώς στην συγκεκριμένη υλοποίηση λαμβάνουν ακέραιες τιμές από -255 έως 255 ή -1 έως 1. Στους 32 bit αριθμούς που τροφοδοτούνται στο Shift Mul, το MSB χρησιμοποιείται ως bit πρόσημου (0 για θετικά, 1 για αρνητικά). Κατά το πολλαπλασιασμό των τιμών το γινόμενο που θα παραχθεί θα έχει πρόσημο που θα προκύπτει βάσει των προσήμων της τιμής της ενεργοποίησης και του φίλτρου.

3.3.2.2.3. CCU Full Adder Controller

Όλες οι έξοδοι των Shift Mul καταλήγουν στο συγκεκριμένο module. Ο ρόλος του περιορίζεται στο να δεχθεί κάθε έτοιμο αποτέλεσμα από τα Shift Mul, να τα αποθηκεύσει σε μία DPRAM (η οποία λειτουργεί επίσης ως circular buffer) και στην συνέχεια να τροφοδοτήσει τον Full Adder με τις τιμές αυτές. Η DPRAM η οποία επιλέγεται περιέχει 256 θέσεις των 32 bit, ώστε να μπορεί να διατηρήσει το σύνολο των γινομένων που προκύπτουν για μέγιστο φίλτρο. Ο Full Adder δέχεται σε κάθε κύκλο μία τιμή και την προσθέτει στο σύνολο των προηγουμένων. Για κάθε ενεργοποίηση της CCU, περιμένει να αθροίσει συγκεκριμένο πλήθος στοιχείων. Μόλις ολοκληρωθεί η άθροιση αυτών, στην έξοδο της CCU υπάρχει το τελικό αποτέλεσμα της συνέλιξης.

3.3.3. 4-Channel Frame Calculator

Σκοπός του 4-Channel Frame Calculator είναι ο υπολογισμός της συνέλιξης, κατά την τοποθέτηση του φίλτρου επάνω σε μια περιοχή των ενεργοποιήσεων, τεσσάρων ή λιγότερων καναλιών εκ του συνόλου παράλληλα. Για τον παράλληλο υπολογισμό των καναλιών απαιτούνται 4 modules Channel Calculator καθώς και ένα ακόμη, ο Channel Selector, το οποίο θα τροφοδοτεί κάθε Channel calculator με δεδομένα του καναλιού που του αντιστοιχούν. Ο 4-Channel Frame Calculator δέχεται από υψηλότερο επίπεδο μία διεύθυνση, η οποία ανεξάρτητα από το κανάλι στο οποίο βρισκόμαστε, δείχνει σε ποια περιοχή των ενεργοποιήσεων (μήκος-ύψος) όπου εφαρμόζεται το φίλτρο, την οποία και μεταβιβάζει στον Channel Selector, καθώς και μία διεύθυνση στην οποία πρέπει να αποθηκευτούν τα αποτελέσματα στην εξωτερική μνήμη. Με την ενεργοποίηση του 4-Channel Frame Calculator, πρώτα ενεργοποιείται ο Channel Selector ο οποίος δεχόμενος τις αρχικές διευθύνσεις φίλτρου – ενεργοποιήσεων κάνει διαδοχικές αιτήσεις στην εξωτερική RAM, για κάθε κανάλι προς επεξεργασία ξεχωριστά, και διοχετεύει τα δεδομένα που λαμβάνει στο κατάλληλο Channel Calculator. Περιμένει να συλλέξει αποτελέσματα από τα 4 Channel Calculators (τα αποτελέσματα παράγονται από την CCU μόλις ο Full Adder αθροίσει συγκεκριμένο αριθμό τιμών) αθροίζοντάς τα, και στη συνέχεια προωθεί την διεύθυνση αποθήκευσης καθώς και την τιμή της συνέλιξης που υπολόγισε στο top level της σχεδίασης. Το άθροισμα που προκύπτει

αναπαριστά την συνέλιξη τεσσάρων ή λιγότερων καναλιών μιας περιοχής των ενεργοποιήσεων και του φίλτρου.

Το συγκεκριμένο module πρέπει να εμπεριέχει Channel Calculators πλήθους κάποιας δύναμης του δύο. Κατά αυτόν τον τρόπο αποφεύγεται η χρήση διαιρετών και πολλαπλασιαστών σε υψηλότερο επίπεδο που συντονίζει την λειτουργία του εν λόγω module, που θα οδηγούσαν σε ανεπιθύμητες καθυστερήσεις (Framer – πχ για την παραγωγή offset). Συγκεκριμένα επιλέγεται ο αριθμός των Channel Calculators να είναι τέσσερα (2^2), καθώς λόγω περιορισμών σε πόρους υλικού, η υλοποίηση μας δεν θα μπορούσε να τοποθετηθεί στην επιλεγμένη FPGA με χρήση οκτώ (2^3) Channel Calculators.

3.3.3.1. Channel Selector

Η λειτουργία του Channel Selector, εφόσον τροφοδοτεί τον κάθε Channel calculator με δεδομένα, σχετίζεται άμεσα με την περιοχή επάνω στις ενεργοποιήσεις όπου τοποθετείται το φίλτρο. Και εδώ, όπως και στον LBC του Channel Calculator, διακρίνονται τρείς διαφορετικές περιπτώσεις βάσει των οποίων η λειτουργία του Channel Selector διαφέρει.

- ‘Οταν το εκάστοτε φίλτρο τοποθετείται για πρώτη φορά λαμβάνει από υψηλότερο επίπεδο διευθύνσεις που δείχνουν στα δεδομένα της πρώτης θέσης των ενεργοποιήσεων και του φίλτρου όπως αυτά είναι αποθηκευμένα στην εξωτερική μνήμη. Βάσει των διευθύνσεων αυτών, πραγματοποιεί αίτημα ανάγνωσης προς την RAM και περιμένει στους επόμενους κύκλους να λάβει Fdim τιμές του φίλτρου και των ενεργοποιήσεων. Οι τιμές αυτές αποθηκεύονται στους Local Buffers του LBC του πρώτου Channel Calculator και αποτελούν την πρώτη στήλη του πρώτου καναλιού ενεργοποιήσεων και φίλτρου που θα επεξεργαστεί ο 4-Channel Frame Calculator. Συνολικά πρέπει να λάβουμε Fdim στήλες. Τα δεδομένα ενεργοποιήσεων – φίλτρου της δεύτερης στήλης βρίσκονται μετέπειτα κατά ΠΥΕ και ΠΥΦ αντίστοιχα τιμές, όπως είναι αποθηκευμένες στην RAM, από τις αρχικές διευθύνσεις. Για να λάβουμε τα δεδομένα της επόμενης στήλης ενεργοποιήσεων και φίλτρου προσθέτουμε στις τρέχουσες διευθύνσεις ενεργοποιήσεων - φίλτρου ΠΥΕ και ΠΥΦ αντίστοιχα, μέχρι να λάβουμε τις τελευταίες στήλες.

Για τα επόμενα κανάλια ακολουθείται η ίδια διαδικασία, με την διαφορά ότι στις αρχικές διευθύνσεις προστίθενται Activations Channel Offset και Filter Channel Offset αντίστοιχα επί τον αριθμό του καναλιού για το οποίο προορίζονται τα δεδομένα (ξεκινώντας αρίθμηση από το 0, κανάλι 1, 2,3)

- Όταν το φίλτρο εφαρμόζεται στην πρώτη θέση μιας γραμμής των ενεργοποιήσεων (εκτός της πρώτης), δηλαδή πραγματοποιείται κατακόρυφη κίνηση του φίλτρου, η διαδικασία ταυτίζεται με την παραπάνω, με την διαφορά ότι αιτούμαστε και λαμβάνουμε στήλες διαφορετικών καναλιών μόνο των ενεργοποιήσεων.
- Όταν το εκάστοτε φίλτρο τοποθετείται για δεύτερη ή μεταγενέστερη φορά πάνω σε μία γραμμή των ενεργοποιήσεων, εφαρμόζεται η ίδια διαδικασία με την δεύτερη περίπτωση, με την διαφορά ότι ο Channel Selector αιτείται και λαμβάνει stride στήλες των ενεργοποιήσεων για κάθε κανάλι.

3.3.4. Single Filter Calculator

Το συγκεκριμένο module αποσκοπεί στην σάρωση των ενεργοποιήσεων με ένα φίλτρο, δηλαδή στην εφαρμογή του φίλτρου σε όλες τις κατάλληλες θέσεις επάνω στις ενεργοποιήσεις βάσει stride, και την εξαγωγή ενός διδιάστατου πίνακα αποτελεσμάτων (feature map). Για την επίτευξη των παραπάνω απαραίτητη είναι η συνεργασία των 4-Channel Frame Calculator καθώς και ενός επιπλέον module, του Framer. Με την ενεργοποίηση του Single Filter Calculator πρώτα ενεργοποιείται ο Framer ο οποίος υπολογίζει κατάλληλα την διεύθυνση των στοιχείων (φίλτρου – ενεργοποιήσεων) που πρέπει να ανασυρθούν από την εξωτερική μνήμη, καθώς και την διεύθυνση στην οποία πρέπει να αποθηκευτούν τα αποτελέσματα που θα παράξει ο 4-Channel Frame Calculator.

Διακρίνονται δύο περιπτώσεις:

- Στην περίπτωση που ενεργοποιήσεις – φίλτρο έχουν αριθμό καναλιών μικρότερο ή ίσο του 4, τότε ο Framer θα υποδείξει την σάρωση των ενεργοποιήσεων από τον 4-Channel Frame Calculator μία φορά.

- Στην περίπτωση που ενεργοποιήσεις – φίλτρο έχουν αριθμό καναλιών μεγαλύτερο του 4, ο Framer θα υποδείξει την σάρωση των ενεργοποιήσεων από τον 4-Channel Frame Calculator φορές ίσες ⌈NumOfChannels/4⌉

Για ενεργοποιήσεις - φίλτρο 5 καναλιών, θα πραγματοποιηθούν δύο σαρώσεις από το συγκεκριμένο φίλτρο. Στην πρώτη σάρωση θα υπολογιστεί η συνέλιξη για κάθε εφαρμογή του φίλτρου επάνω στις ενεργοποιήσεις των 4 πρώτων καναλιών ενώ στην δεύτερη σάρωση θα υπολογισθεί η συνέλιξη του $5^{ου}$ καναλιού. Στην περίπτωση που τα κανάλια είναι περισσότερα από 4, και εφόσον βρισκόμαστε στην $2^{η}$ ή σε επόμενη σάρωση των ενεργοποιήσεων από το φίλτρο, για κάθε εφαρμογή του φίλτρου που θα υποδειχθεί στον 4-Channel Frame Calculator, το αποτέλεσμα συνέλιξης πρέπει να αθροιστεί στο αποτέλεσμα συνέλιξης των προηγούμενων καναλιών, που είναι αποθηκευμένο στη RAM. Μόλις ο Framer υποδείξει την τελευταία θέση που μπορεί να εφαρμοσθεί το φίλτρο για τα τελευταία 4 ή λιγότερα κανάλια και εφόσον προκύψει το αποτέλεσμα της συνέλιξης από τον 4-Channel Frame Calculator, το module Single Filter Calculator απενεργοποιείται και μπορεί να επαναχρησιμοποιηθεί για επόμενο φίλτρο.

3.3.4.1. Framer

Ο Framer είναι υπεύθυνος για την παραγωγή των διευθύνσεων ανάγνωσης δεδομένων από τον 4-Channel Frame Calculator και την παραγωγή των διευθύνσεων εγγραφής του αποτελέσματος συνέλιξης που θα προκύψει, στην RAM.

Οι ενεργοποιήσεις είναι μοναδικές και για κάθε σάρωση η θέση μνήμης του πρώτου στοιχείου του πρώτου καναλιού είναι γνωστή. Τα φίλτρα είναι πολλαπλά, με αποτέλεσμα για διαφορετικά φίλτρα η πρώτη τιμή του πρώτου καναλιού να είναι αποθηκευμένη σε διαφορετική θέση μνήμης. Οι διδιάστατοι πίνακες αποτελεσμάτων που προκύπτουν από διαφορετικά φίλτρα πρέπει να αποθηκεύονται σε διαφορετικές περιοχές στη μνήμη. Ο Framer λαμβάνει από υψηλότερο επίπεδο ένα offset βάσει του οποίου υποδεικνύεται το φίλτρο με το οποίο θα σαρωθούν οι ενεργοποιήσεις και ένα ακόμη offset το οποίο υποδεικνύει την αρχική διεύθυνση αποθήκευσης αποτελεσμάτων σε εξωτερική RAM. Είναι

σχεδιασμένος έτσι ώστε να συνεργάζεται με τον 4-Channel Frame Calculator, υποδεικνύοντας του την διεύθυνση που αντιστοιχεί στο πρώτο από τα 4 κανάλια φίλτρου και ενεργοποιήσεων που προορίζονται για επεξεργασία, καθώς και ένα αριθμό καναλιών (<=4) που ο 4-Channel Frame Calculator πρέπει να επεξεργασθεί στην παρούσα σάρωση. Μεταξύ διαδοχικών σαρώσεων, για διαφορετικά κανάλια φίλτρου – ενεργοποιήσεων, τα offset αυτών, που αφορούν τα κανάλια προς επεξεργασία, αυξάνονται κατά

$$4 * ActivationsChannelOffset$$

και

$$4 * FilterChannelOffset$$

αντίστοιχα. Κάθε σάρωση περιέχει πολλές ενεργοποιήσεις του 4-Channel Frame Calculator, αριθμού ίσου με το πλήθος των κατάλληλων θέσεων που μπορεί να εφαρμοσθεί το φίλτρο επάνω στις ενεργοποιήσεις βάσει stride. Στο σύνολο των εφαρμογών εντός σάρωσης η διεύθυνση του φίλτρου παραμένει αμετάβλητη και ταυτίζεται με το offset του φίλτρου, που δίνεται από υψηλότερο επίπεδο, αθροισμένο με το offset του πρώτου καναλιού εκ των τεσσάρων που είναι προς επεξεργασία στην συγκεκριμένη σάρωση. Ανά εφαρμογή εντός σάρωσης αυτό που αλλάζει είναι η διεύθυνση των ενεργοποιήσεων καθώς θέλουμε να υποδείξουμε για κάθε εφαρμογή του φίλτρου, διαφορετικές περιοχές που καλύπτει.

Μέχρι στιγμής για κάθε σάρωση, όσον αφορά τις ενεργοποιήσεις, γνωστό είναι μόνο το offset του πρώτου από τα τέσσερα κανάλια, που είναι για επεξεργασία. Στο offset αυτό πρέπει να αθροιστεί ακόμη μια διεύθυνση η οποία θα υποδεικνύει το πρώτο στοιχείο της κατάλληλης περιοχής των ενεργοποιήσεων για κάθε εφαρμογή του φίλτρου. Για την εύρεση αυτής της διεύθυνσης, σε κάθε σάρωση, χρησιμοποιούνται μεταβλητές συντεταγμένων (x,y) ενώ θεωρούνται γνωστές οι μέγιστες τιμές που μπορούν να λάβουν οι συντεταγμένες αυτές (x_{max},y_{max}).

Συγκεκριμένα:

$$x_{Max} = ΠΜΕ - Fdim + 1$$

και

$$y_{Max} = ΠΥΕ - Fdim + 1$$

Η ολίσθηση του φίλτρου γίνεται κατά γραμμή και έπειτα κατά στήλη. Για κάθε γραμμή, όπου το y παραμένει αμετάβλητο, διαδοχικές εφαρμογές διαφέρουν στο x κατά stride (>=1). Παραμένουμε στην ίδια γραμμή έως ότου το x + stride < x_{max}. Σε αντίθετη περίπτωση το x μηδενίζεται και το y αυξάνεται κατά stride. Μόλις το x και το y λάβουν την τιμή της τελευταίας δυνατής θέσης επάνω στις ενεργοποιήσεις ο Framer ολοκληρώνει την παρούσα σάρωση και αναμένονται τα αποτελέσματα από το 4-Channel Frame Calculator προκειμένου ο Framer να συνεχίσει σε επόμενη σάρωση για διαφορετικά κανάλια ενεργοποιήσεων – φίλτρου είτε το module Single Filter Calculator να ολοκληρώσει την λειτουργία του.

Ο Framer διατηρεί επίσης δύο ακόμα μεταβλητές συντεταγμένων x_{result}, y_{result}, οι οποίες - εφόσον το stride είναι 1 – καθόλη την σάρωση έχουν τις ίδιες τιμές με τα x,y αντίστοιχα. Εάν το stride είναι μεγαλύτερο από 1, τότε για κάθε προσαύξηση του x και του y κατά stride τα x_{result} και y_{result} αυξάνονται κατά 1. Προφανώς σε περίπτωση που το x μηδενισθεί, το x_{result} μηδενίζεται επίσης.

Τα x,y αναπαριστούν την επάνω αριστερά θέση της περιοχής των ενεργοποιήσεων στην οποία εφαρμόζεται το φίλτρο. Τα x_{result}, y_{result}αναπαριστούν τις συντεταγμένες του αποτελέσματος της συνέλιξης (η συνέλιξη πραγματοποιείται βάσει των x, y) στον διδιάστατο πίνακα που έχει διαστάσεις x_{max},y_{max} (εφόσον το stride είναι 1). Τόσο οι συντεταγμένες των ενεργοποιήσεων όσο και των αποτελεσμάτων δεν αντιστοιχούν στα επιθυμητά δεδομένα όπως αυτά είναι αποθηκευμένα σε εξωτερική RAM, ωστόσο βάσει αυτών υπολογίζονται οι αντίστοιχες διευθύνσεις της RAM. Οι διευθύνσεις ανάγνωσης και εγγραφής (αποτελεσμάτων) που θα προκύψουν θα αθροιστούν με τα κατάλληλα offsets (στην διεύθυνση ανάγνωσης προστίθεται το offset που σχετίζεται με το πρώτο κανάλι από αυτά που είναι προς επεξεργασία και στην διεύθυνση αποτελεσμάτων προστίθεται το offset που σχετίζεται με την περιοχή αποθήκευσης για το συγκεκριμένο φίλτρο).

Οι δύο διδιάστατοι πίνακες, τα στοιχεία των οποίων αναπαρίστανται από x, x_{result} και y, y_{result} πρέπει να μετατραπούν σε επίσης δύο διδιάστατους πίνακες οι οποίοι ανά γραμμή φιλοξενούν τέσσερα στοιχεία, καθώς έτσι είναι διαταγμένη η RAM. Οι τύποι για την μετατροπή δίνονται παρακάτω:

Για την εύρεση της γραμμής στην οποία είναι αποθηκευμένα τα δεδομένα που μας ενδιαφέρουν:

$$line = \left(\left(\frac{h}{4}\right) * j\right) + (\frac{\left((h\%4) * j\right)}{4}) + (\frac{i}{4})$$

Ενώ για την εύρεση του επιθυμητού στοιχείου ανά γραμμή εκ των τεσσάρων που περιέχονται σε κάθε γραμμή:

$$element_{position} = (h * j)\%4 + i\%4$$

Στους παραπάνω τύπους το j αντικαθίσταται με x, x_{result} και το i με y, y_{result} ανάλογα με το εάν θέλουμε να βρούμε διεύθυνση ανάγνωσης ή αποτελεσμάτων. Το h ταυτίζεται με το ύψος των ενεργοποιήσεων.

Το σύνολο των διαιρέσεων που γίνονται στους παραπάνω τύπους είναι με διαιρέτη το 4, δηλαδή αρκεί να απομονώσουμε από τον σύνολο των bits του διαιρετέου όλα τα bits εκτός από τα δύο LSB. Η πράξη του mod γίνεται επίσης με βάση το 4, δηλαδή αρκεί να απομονώσουμε τα δύο LSB του αριθμού. Το σύνολο των πολλαπλασιασμών πραγματοποιείται με την βοήθεια παραλλαγής του module Shift Mul (τα οποία έχουν την ικανότητα να πολλαπλασιάσουν αριθμούς 32 bit). Για την εύρεση της συνολικής διεύθυνσης ο αριθμός που προκύπτει από το line καταλαμβάνει τα 30 MSB ενός 32bit σήματος, ενώ το $element_{position}$ καταλαμβάνει τα 2 LSB. Σε περίπτωση που το $element_{position}$ είναι μεγαλύτερο του 3, δηλαδή καταλαμβάνει παραπάνω από 2 bit, το overflow bit προστίθεται στο line.

Για να προκύψουν οι διευθύνσεις αποτελεσμάτων και ανάγνωσης απαιτούνται πολλαπλοί κύκλοι, μέρος των οποίων καταλαμβάνουν οι κύκλοι που απαιτούνται για την παραγωγή των γινομένων από τα Shift Mul. Προκειμένου να αποφευχθούν οι συγκεκριμένες καθυστερήσεις, ενώ ο 4-Channel Frame Calculator υπολογίζει την τιμή της συνέλιξης για τα δεδομένα που του έχουν ανατεθεί ο Framer παράλληλα υπολογίζει τις διευθύνσεις ανάγνωσης και αποτελεσμάτων για την επόμενη εφαρμογή του φίλτρου επάνω στις ενεργοποιήσεις.

Μαζί με τις υπολογισμένες διευθύνσεις ο Framer αναθέτει στον 4-Channel Frame Calculator και ένα mode, ανάλογα με το που τοποθετεί το φίλτρο επάνω στις ενεργοποιήσεις, με βάση το οποίο ο 4-Channel Frame Calculator και συνεπώς οι Channel Calculators που εμπεριέχονται, λειτουργούν με συγκεκριμένο τρόπο:

Mode=11:	Πρώτη εφαρμογή του φίλτρου στις ενεργοποιήσεις.
Mode=10:	Δεύτερη ή μετέπειτα εφαρμογή του φίλτρου σε μια γραμμή των ενεργοποιήσεων.
Mode=01:	Εφαρμογή του φίλτρου στο πρώτο στοιχείο κάποιας γραμμής των ενεργοποιήσεων (εκτός της πρώτης).

Στην περίπτωση που ο Framer εφαρμόζει το φίλτρο στην ίδια γραμμή των ενεργοποιήσεων (Mode = 10) για δεύτερη η μετέπειτα φορά η διεύθυνση ανάγνωσης δεν υπολογίζεται βάσει του x αλλά βάσει:

$$X_{mode=10} = x + ΠΥΦ - Stride$$

Η παραπάνω μετατροπή λαμβάνει χώρα καθώς κάθε Channel Calculator του 4-Channel Frame Calculator περιέχει ήδη στους Local Buffers του στήλες της περιοχής των ενεργοποιήσεων που πρέπει να προωθηθούν στην CCU από προηγούμενες εφαρμογές του φίλτρου και πλέον βάσει του $X_{mode=10}$ θα πρέπει να αιτηθεί Stride νέες στήλες από την εξωτερική RAM.

3.3.5. Multiple Filter Calculator

Το συγκεκριμένο Module αποσκοπεί στον υπολογισμό των συνελίξεων των ενεργοποιήσεων με το σύνολο των φίλτρων, παράγοντας για κάθε φίλτρο ένα feature map. Για τον υπολογισμό του κάθε φίλτρου ξεχωριστά ενεργοποιείται το Module Single Filter Calculator από το Filter Selector.

3.3.5.1. Filter Selector

Είναι υπεύθυνος εκτός από την ενεργοποίηση του Single Filter Calculator για κάθε ξεχωριστό φίλτρο, να υπολογίσει και στην συνέχεια να τροφοδοτήσει το Single Filter Calculator με δύο Offsets, ένα που δείχνει την περιοχή στην εξωτερική RAM στην οποία είναι αποθηκευμένο το εκάστοτε φίλτρο και ένα που δείχνει στην αρχική διεύθυνση μνήμης όπου θα αποθηκευτεί ο διδιάστατος πίνακας αποτελεσμάτων που θα παράξει ο Single Filter Calculator για το εκάστοτε φίλτρο. Με την

ενεργοποίηση του Multiple Filter Calculator πρώτα ενεργοποιείται ο Filter Selector, ενεργοποιώντας και τροφοδοτώντας το Single Filter calculator με μηδενικά Offsets. Μόλις ο Single Filter Calculator ολοκληρώσει την λειτουργία του, ο Filter Selector αυξάνει κατάλληλα τα δύο offsets και τον ενεργοποιεί πάλι. Η διαδικασία επαναλαμβάνεται έως ότου εξαντληθούν τα διαθέσιμα φίλτρα. Για κάθε φίλτρο τα offsets προσαυξάνονται ως εξής:

Στο offset του φίλτρου προστίθεται

$$WholeFilterOffset = FilterWidth * FilterHeight * NumOfChannels$$

Στο offset αποτελεσμάτων προστίθεται:

$$ResultOffset = ResultWidth * ResultHeight$$

Το ResultWidth και το ResultHeight ταυτίζεται με τον αριθμό των εφαρμογών του φίλτρου επάνω στις ενεργοποιήσεις στον οριζόντιο και στον κάθετο άξονα αντίστοιχα.

3.4. Μονάδα ReLU

Η ενεργοποίηση της μονάδας ReLU, στην συγκεκριμένη υλοποίηση, δεν είναι υποχρεωτική.

Αποτελέσματα συνέλιξης που αφορούν τα τελευταία κανάλια ενεργοποιήσεων – φίλτρου που παράγονται από το 4-Channel Frame Calculator μεταβιβάζονται ως είσοδοι στην μονάδα ReLU. Εφόσον το αποτέλεσμα αυτό είναι αρνητικό (δηλαδή το 31ο bit είναι άσσος), επιστρέφεται σε υψηλότερο επίπεδο η τιμή μηδέν, αλλιώς επιστρέφεται η αρχική τιμή.

3.5. Μονάδα Pooling

Η λειτουργία της μονάδας Pooling, η οποία αποσκοπεί στην μείωση των διαστάσεων του αποτελέσματος συνέλιξης κάθε φίλτρου και τη συμπύκνωση της πληροφορίας αυτού, δεν ακολουθεί απαραίτητα κάθε συνελικτικό επίπεδο.

Το Pooling μπορεί να πραγματοποιηθεί με την χρήση δύο διαφορετικών τεχνικών Average Pooling (Μ.Ο. των ελεγχόμενων τιμών) και Max Pooling (Μέγιστη τιμή των ελεγχόμενων τιμών). Παράμετρο της μονάδας αποτελεί η διάσταση (Dimension) του Pooling (dp) που θα εφαρμοσθεί, η οποία εκφράζει το ύψος και το μήκος της τετραγωνικής περιοχής ελέγχου, που θα διασχίσει τον πίνακα τιμών των αποτελεσμάτων της συνέλιξης με stride ίσο με την διάσταση αυτού.

Για την ορθή λειτουργία της μονάδας χρησιμοποιείται DPRAM 512 θέσεων των 32 bit η καθεμία. Η μονάδα Pooling δέχεται τον πίνακα αποτελεσμάτων της συνέλιξης φίλτρου με ενεργοποιήσεις, στοιχείο προς στοιχείο, ανά γραμμή.

Θεωρούμε ότι ο πίνακας αποτελεσμάτων της συνέλιξης έχει μήκος Wr και ύψος Hr. Ο πίνακας αποτελεσμάτων μετά την διαδικασία του Pooling θα έχει:

$$Wp = (\frac{Wr - dp}{dp}) + 1$$

$$Hp = \left(\frac{Hr - dp}{dp}\right) + 1$$

Για κάθε dp γραμμές του πίνακα αποτελεσμάτων (χωρίς Pooling) παράγονται Wp στοιχεία που αντιστοιχούν σε μια γραμμή του πίνακα αποτελεσμάτων Pooling.

Στο Max Pooling ακολουθείται η παρακάτω διαδικασία:

Ο πίνακας αποτελεσμάτων της συνέλιξης ελέγχεται ανά ομάδες των dp γραμμών χωρίς επικάλυψη. Το πρώτο στοιχείο της πρώτης γραμμής θα καταχωρηθεί χωρίς σύγκριση στην πρώτη θέση της DPRAM. Τα επόμενα dp-1 στοιχεία της πρώτης γραμμής – εφόσον προωθούνται σειριακά ένα προς ένα στην μονάδα – θα συγκριθούν με την τιμή της πρώτης θέσης της DPRAM και εάν είναι μεγαλύτερα θα αντικαταστήσουν την τιμή της. Επόμενες θέσεις της DPRAM θα λάβουν τιμή, κατά αντίστοιχο τρόπο, με βάση τα υπόλοιπα στοιχεία της γραμμής, που εξετάζονται ανά ομάδες μεγέθους dp. Για το σύνολο των επόμενων γραμμών, πλην της τελευταίας, τα στοιχεία αυτών εξετάζονται επίσης ανά ομάδες τιμών dp, και εφόσον για κάθε ομάδα κάποια τιμή είναι μεγαλύτερη της υπάρχουσας στην DPRAM, στην αντίστοιχη θέση, καταχωρείται. Στην τελευταία γραμμή για κάθε ομάδα dp στοιχείων, αφού γίνει σύγκριση αυτών με την τιμή της αντίστοιχης θέσης στην DPRAM, η μέγιστη τιμή δίνεται σε υψηλότερο επίπεδο μαζί με μία διεύθυνση για

την αποθήκευση της τιμής αυτής, σε κατάλληλη θέση σε εξωτερική RAM. Η παραπάνω διαδικασία επαναλαμβάνεται για το σύνολο γραμμών του πίνακα αποτελεσμάτων.

Στο AVG Pooling ακολουθείται η παρακάτω διαδικασία:

Ο πίνακας αποτελεσμάτων της συνέλιξης ελέγχεται ανά ομάδες των dp γραμμών χωρίς επικάλυψη. Το πρώτο στοιχείο της πρώτης γραμμής θα καταχωρηθεί στην πρώτη θέση της DPRAM. Τα επόμενα dp-1 στοιχεία της πρώτης γραμμής θα αθροιστούν ένα προς ένα με την τιμή της πρώτης θέσης της DPRAM και το αποτέλεσμα θα καταχωρηθεί στην πρώτη θέση της DPRAM. Επόμενες θέσεις της DPRAM θα λάβουν τιμή, κατά αντίστοιχο τρόπο, με βάση τα υπόλοιπα στοιχεία της γραμμής, που εξετάζονται ανά ομάδες μεγέθους dp. Για το σύνολο των επόμενων γραμμών, πλην της τελευταίας, τα στοιχεία αυτών εξετάζονται επίσης ανά ομάδες τιμών dp, αθροίζονται με την τιμή της κατάλληλης θέσης της DPRAM και καταχωρούνται στην ίδια θέση. Στην τελευταία γραμμή, για κάθε ομάδα dp στοιχείων, αφού γίνει άθροιση αυτών με την τιμή της αντίστοιχης θέσης στην DPRAM. Το άθροισμα που προκύπτει διαιρείται με dp^2 και δίνεται σε υψηλότερο επίπεδο μαζί με μία διεύθυνση για την αποθήκευση της τιμής αυτής, σε κατάλληλη θέση σε εξωτερική RAM. Η παραπάνω διαδικασία επαναλαμβάνεται για το σύνολο γραμμών του πίνακα αποτελεσμάτων.

Εκτός από την παραπάνω βασική λειτουργία είτε για Max ή για AVG Pooling, απαιτείται και μία διαδικασία απόδοσης διεύθυνσης των αποτελεσμάτων του Pooling για την αποθήκευσή τους στην RAM. Η παραγωγή διεύθυνσης είναι αναγκαία όταν βρισκόμαστε στο τελευταίο στοιχείο κάθε dp ομάδας στοιχείων και dp γραμμής από την ομάδα γραμμών που ελέγχονται. Χρησιμοποιούνται δύο μεταβλητές x_p και y_p. Το y_p αυξάνεται κατά ένα κάθε dp γραμμές που εξετάζουμε, ενώ το x_p αυξάνεται κατά 1 για κάθε dp στοιχεία της τελευταίας γραμμής της ομάδας. Οι συντεταγμένες x_p και y_p αντιστοιχούν σε τιμές του ιδεατού πίνακα αποτελεσμάτων Pooling μήκους Wp και ύψους Hp. Για την μετατροπή των συντεταγμένων αυτών σε θέσεις μνήμης χρησιμοποιούνται οι τύποι που έχουν αναλυθεί στον Framer της Convolution μονάδας.

3.6. RAM interconnect

Θεωρείται ότι για την ανάγνωση δεδομένων ενεργοποιήσεων και φίλτρων καθώς και για την αποθήκευση αποτελεσμάτων συνέλιξης – με ή χωρίς Pooling – χρησιμοποιείται μία ενιαία εξωτερική RAM. Στην παρούσα υλοποίηση την RAM αντιπροσωπεύουν τέσσερα διακριτά text αρχεία, η επικοινωνία με τα οποία γίνεται με την χρήση τεσσάρων διαφορετικών interconnects:

- Image File, το οποίο περιέχει αποθηκευμένα τα δεδομένα των ενεργοποιήσεων. Στο αρχείο αυτό περιέχεται το σύνολο των καναλιών των ενεργοποιήσεων, με κάθε κανάλι να είναι αποθηκευμένο κατά στήλες, από το πρώτο έως το τελευταίο διαδοχικά. Η επικοινωνία με το συγκεκριμένο αρχείο γίνεται με την χρήση του Image_RAM_IC, το οποίο όταν ενεργοποιηθεί επιστρέφει σε διαδοχικούς κύκλους το σύνολο των δεδομένων των ενεργοποιήσεων που ζητήθηκαν, πλήθους Fdim, από την διεύθυνση που δίνεται.
- Filter File, το οποίο περιέχει αποθηκευμένα τα δεδομένα του συνόλου των φίλτρων. Στο αρχείο αυτό περιέχεται το σύνολο των καναλιών κάθε φίλτρου, με κάθε κανάλι να είναι αποθηκευμένο κατά στήλες, από το πρώτο έως το τελευταίο διαδοχικά. Πολλαπλά φίλτρα αποθηκεύονται διαδοχικά. Η επικοινωνία με το συγκεκριμένο αρχείο γίνεται με την χρήση του Filter_RAM_IC, το οποίο όταν ενεργοποιηθεί επιστρέφει δεδομένα ενός φίλτρου, πλήθους Fdim, από την διεύθυνση που δίνεται.
- Result File, στο αρχείο αυτό αποθηκεύονται τα αποτελέσματα της συνέλιξης ενεργοποιήσεων με φίλτρα όπως προκύπτουν χωρίς να προηγηθεί Pooling. Τα αποτελέσματα αποθηκεύονται κατά στήλες ενώ αποτελέσματα διαδοχικών φίλτρων αποθηκεύονται διαδοχικά. Η επικοινωνία με το συγκεκριμένο αρχείο γίνεται με την χρήση του Result_RAM_IC, το οποίο όταν ενεργοποιηθεί αποθηκεύει μία τιμή στα κατάλληλα bits βάσει της διεύθυνσης που δίνεται. Ωστόσο, σε περίπτωση που ο αριθμός των καναλιών ενεργοποιήσεων - φίλτρων είναι μεγαλύτερος από τέσσερα και ειδικότερα όταν υπολογίζουμε την συνέλιξη καναλιών που δεν ανήκουν στην πρώτη τετράδα, υπάρχει η ανάγκη ανάγνωσης από το αρχείο της τιμής της συνέλιξης των προηγούμενων καναλιών στην οποία θα αθροιστεί το

αποτέλεσμα της τρέχουσας συνέλιξης και το άθροισμα θα αποθηκευτεί εκ νέου στο αρχείο στην ίδια θέση.

- Pool File, στο αρχείο αυτό αποθηκεύονται τα αποτελέσματα της συνέλιξης ενεργοποιήσεων με φίλτρα όπως αυτά προκύπτουν μετά την διαδικασία του Pooling. Η επικοινωνία με το συγκεκριμένο αρχείο γίνεται με την χρήση του Pool_RAM_IC, το οποίο όταν ενεργοποιηθεί αποθηκεύει δεδομένα στην διεύθυνση που δίνεται.

Οι διευθύνσεις και το Burst size που δίνονται προς το RAM interconnect αφορούν τιμές ενεργοποιήσεων - φίλτρου και όχι θέσεις μνήμης. Κάθε στοιχείο ανεξάρτητα από το αρχείο στο οποίο ανήκει, αποτελείται από δύο bytes το δεύτερο από τα οποία χρησιμοποιείται για την αποθήκευση του πρόσημου και το πρώτο για την αποθήκευση της απόλυτης τιμής. Το core element πραγματοποιώντας αίτηση ανάγνωσης σε κάποιο από τα αρχεία περιμένει να λάβει μια τιμή των 32 Bits από την οποία το 31^{ο} bit εκφράζει το πρόσημο. Τα interconnect για κάθε τιμή που πρέπει να προωθήσουν στο core element τοποθετούν το byte της απόλυτης τιμής στα 8 τελευταία bits ενός σήματος, και τοποθετούν ένα bit από το byte πρόσημου στο 31^{ο}bit του σήματος, το οποίο και θα επιστραφεί. Η αποθήκευση μιας 32 bit τιμής, που θα δοθεί από το core element προς κάποιο interconnect, θα μετατραπεί σε δύο bytes αντίστοιχα.

Στην υλοποίηση χρησιμοποιούνται τέσσερα διακριτά αρχεία που αντιπροσωπεύουν μια RAM. Για την μεταφορά των δεδομένων των αρχείων σε μια ενιαία RAM θα χρησιμοποιηθούν τέσσερεις διαφορετικές περιοχές, με την χρήση κατάλληλων offsets. Στην περίπτωση που τα δεδομένα αποθηκεύονται σε RAM και όχι text αρχεία, ο ρόλος των Interconnect θα περιορίζεται στο να μετατρέπουν τις διευθύνσεις που τους δίνονται και το burst size (τα οποία αφορούν πλήθος τιμών προς ανάγνωση/εγγραφή) σε κατάλληλες διευθύνσεις προκειμένου να πραγματοποιήσουν αιτήσεις στο DMA.

3.7. Accelerator Top Level

Για την ορθή λειτουργία των επιμέρους μονάδων, απαιτείται αρχικοποίηση πλήθους μεταβλητών του συστήματος, η τιμή των οποίων προκύπτει βάσει των παραμέτρων που δίνονται:

$$\max_{x} = Wact - Fdim + 1$$

Το $\max_X$ αντιπροσωπεύει την τελευταία θέση κατά μήκος των ενεργοποιήσεων στην οποία μπορεί να εφαρμοσθεί κάποιο φίλτρο.

$$\max_{y} = Hact - Fdim + 1$$

Το $\max_Y$ αντιπροσωπεύει την τελευταία θέση κατά ύψος των ενεργοποιήσεων στην οποία μπορεί να εφαρμοσθεί κάποιο φίλτρο.

$$ActivationsChannelOffset = Wact * Hact$$

Το offset που πρέπει να προστεθεί στην διεύθυνση που δείχνει σε κάποιο στοιχείο προηγούμενου καναλιού των ενεργοποιήσεων ώστε να μεταφερθούμε στο αντίστοιχο στοιχείο επόμενου καναλιού.

$$FilterChannelOffset = Fdim * Fdim$$

Το offset που πρέπει να προστεθεί στην διεύθυνση που δείχνει σε κάποιο στοιχείο προηγούμενου καναλιού φίλτρου ώστε να μεταφερθούμε στο αντίστοιχο στοιχείο επόμενου καναλιού του ίδιου φίλτρου.

$$WholeFilterOffset = Fdim * Fdim * NCh$$

Offset το οποίο προστιθέμενο στην διεύθυνση στοιχείου ενός φίλτρου δείχνει στο αντίστοιχο στοιχείο του επόμενου φίλτρου

$$Wr = \left(\frac{Wact - Fdim}{S}\right) + 1$$

Το μήκος του ιδεατού πίνακα αποτελεσμάτων της συνέλιξης, χωρίς Pooling, ενός φίλτρου με τις ενεργοποιήσεις

$$Hr = \left(\frac{Hact - Fdim}{S}\right) + 1$$

Το ύψος του ιδεατού πίνακα αποτελεσμάτων της συνέλιξης, χωρίς Pooling, ενός φίλτρου με τις ενεργοποιήσεις

$$ResultOffset = Wr * Hr$$

Κάθε σάρωση των ενεργοποιήσεων από ένα φίλτρο παράγει ένα διδιάστατο πίνακα αποτελεσμάτων. Για την ορθή αποθήκευση του συνόλου των πινάκων χρησιμοποιείται το ResultOffset πολλαπλασιασμένο με τον αριθμό του φίλτρου (ξεκινώντας την αρίθμηση από μηδέν) και το offset που προκύπτει προστίθεται στην διεύθυνση των αποτελεσμάτων.

$$ResultDataSize = Wr * Hr * NF$$

Το μέγεθος του αρχείου αποτελεσμάτων, προκειμένου να είναι σε θέση να περιέχει όλες τις πιθανές τιμές.

$$Wp = \left(\frac{Wr - pd}{pd}\right) + 1$$

Το μήκος του ιδεατού πίνακα αποτελεσμάτων της συνέλιξης, με Pooling, ενός φίλτρου με τις ενεργοποιήσεις

$$Hp = \left(\frac{Hr - pd}{pd}\right) + 1$$

Το ύψος του ιδεατού πίνακα αποτελεσμάτων της συνέλιξης, με Pooling, ενός φίλτρου με Οι ενεργοποιήσεις

$$PoolOffset = Wr * Hr$$

Αντίστοιχα με το ResultOffset, με την διαφορά ότι σε κάθε διδιάστατο πίνακα περιέχονται τα μισά ή λιγότερα στοιχεία.

$$PoolDataSize = Wr * Hr * NF$$

Αντίστοιχα με το ResultSize.

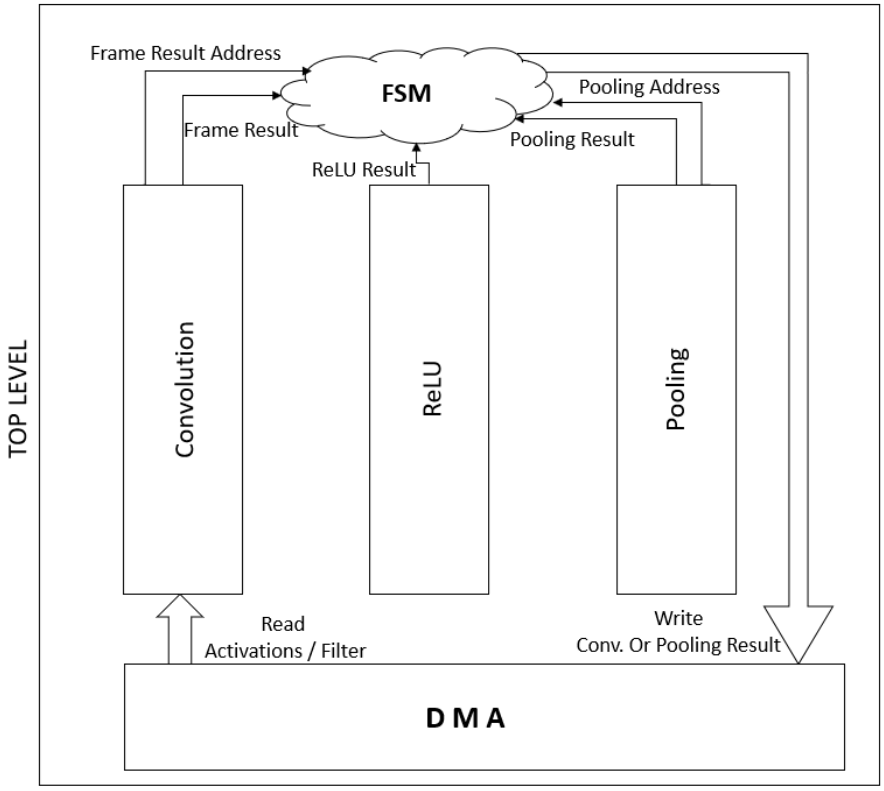
Για τον υπολογισμό των παραπάνω μεταβλητών εκτελούνται πολλαπλασιασμοί και διαιρέσεις. Οι πολλαπλασιασμοί πραγματοποιούνται με την χρήση παραλλαγής του Shift_Mul ενώ για τις διαιρέσεις χρησιμοποιείται ένα επιπλέον module, ο divider.

Για την ολοκλήρωση της αρχικοποίησης απαιτείται μεταβλητό πλήθος κύκλων, εξαρτώμενο από τις τιμές των παραμέτρων που τροφοδοτήθηκαν στο σύστημα.

Μετά την αρχικοποίηση του συστήματος, από το σύνολο των μονάδων, ενεργοποιείται πρώτη η μονάδα Convolution η οποία θα υπολογίσει την τιμή της συνέλιξης για τα τέσσερα πρώτα κανάλια του φίλτρου και των ενεργοποιήσεων εφαρμόζοντας το φίλτρο στην 0,0 θέση των ενεργοποιήσεων. Εφόσον ο συνολικός αριθμός των καναλιών είναι μικρότερος ή ίσος του τέσσερα, τα αποτελέσματα της συνέλιξης θα επιστραφούν στο Top Level. Εάν όμως ο αριθμός των καναλιών είναι μεγαλύτερος του τέσσερα Οι ενεργοποιήσεις θα σαρωθούν από το φίλτρο ⌈NumOfChannels/4⌉ φορές. Στην πρώτη και στις ενδιάμεσες σαρώσεις για κάθε εφαρμογή του φίλτρου, παράγονται αποτελέσματα συνέλιξης τετράδων καναλιών ενεργοποιήσεων – φίλτρου που επιστρέφονται στο top level και καταχωρούνται σε κατάλληλη θέση στην RAM – αφού πρώτα προστεθούν με τα δεδομένα που περιέχει η RAM στην ίδια θέση. Στην τελευταία σάρωση, για κάθε εφαρμογή του φίλτρου, επιστρέφονται επίσης τα δεδομένα στο Top Level. Για τα δεδομένα αυτά, αφού αθροιστούν με τα υπάρχοντα στην RAM, διακρίνονται οι εξής περιπτώσεις:

- ReLU και Pooling ενεργοποιημένα: Τα δεδομένα προωθούνται στην μονάδα ReLU. Τα αποτελέσματα τα οποία θα προκύψουν από την μονάδα ReLU θα αποθηκευτούν στην περιοχή της εξωτερικής RAM που αφορά τα αποτελέσματα χωρίς Pooling και θα προωθηθούν στην μονάδα Pooling. Τα αποτελέσματα που θα προκύψουν σε επόμενους κύκλους από την μονάδα Pooling αποθηκεύονται στην περιοχή της εξωτερικής RAM που αφορά τα αποτελέσματα Pooling.
- ReLU απενεργοποιημένο και Pooling ενεργοποιημένο: Τα δεδομένα αποθηκεύονται στην περιοχή αποτελεσμάτων της RAM και προωθούνται στην μονάδα Pooling. Τα αποτελέσματα που θα προκύψουν μετά την διαδικασία του Pooling θα αποθηκευτούν σε κατάλληλη περιοχή στην εξωτερική RAM που αφορά τα αποτελέσματα Pooling.
- ReLU ενεργοποιημένο και Pooling απενεργοποιημένο: Τα δεδομένα προωθούνται στη μονάδα ReLU και τα αποτελέσματα που θα προκύψουν αποθηκεύονται στην κατάλληλη θέση της περιοχής της RAM που αφορά τα αποτελέσματα.

- ReLU και Pooling απενεργοποιημένα: Εφόσον δεν απαιτείται κάποια επεξεργασία από τις δύο αυτές μονάδες, τα αποτελέσματα που προκύπτουν αποθηκεύονται στην περιοχή της RAM που αφορά τα αποτελέσματα.



Εικόνα 14: Block διάγραμμα Συστήματος

Μόλις η μονάδα convolution επιστρέψει μία τιμή στο top level, η τιμή αυτή επεξεργάζεται απευθείας από τις μονάδες ReLU και Pooling εάν είναι απαραίτητο και αποθηκεύεται στη RAM. Παράλληλα με την επεξεργασία της προηγούμενης τιμής που επιστράφηκε από το Convolution και την αποθήκευσης αυτής, η μονάδα Convolution προχωράει στην εφαρμογή φίλτρου σε επόμενη κατάλληλη θέση των ενεργοποιήσεων. Γενικά η λειτουργία που πρέπει να διεκπεραιώσει η μονάδα Convolution είναι πολύ πιο απαιτητική σε χρόνο από το σύνολο των λειτουργιών που πρέπει να διεκπεραιώσουν οι υπόλοιπες μονάδες, ωστόσο σε περίπτωση που η μονάδα Convolution παράξει αποτέλεσμα για κάποια εφαρμογή του φίλτρου και δεν έχει ολοκληρωθεί η επεξεργασία της προηγούμενης τιμής που επέστρεψε από τις υπόλοιπες

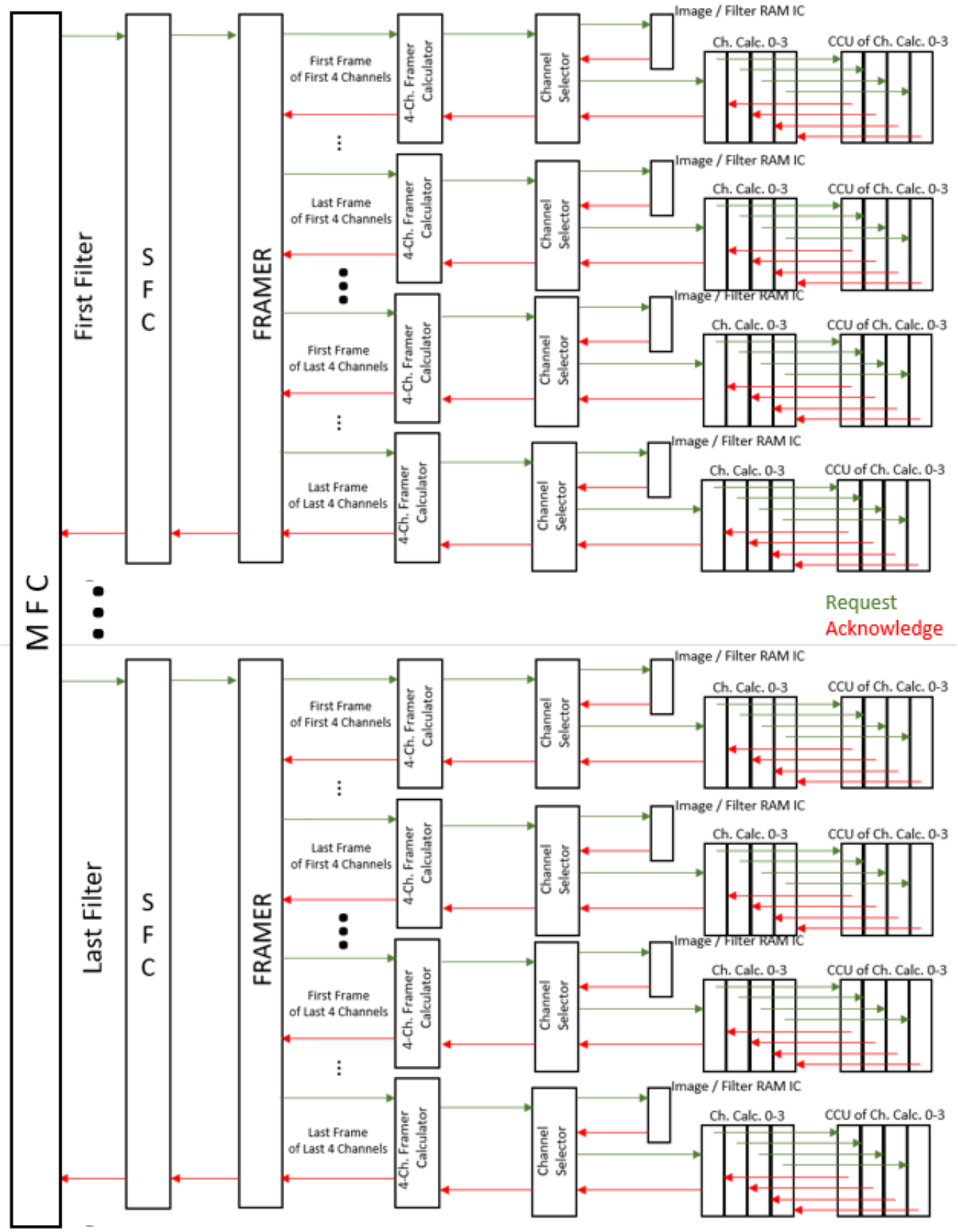
μονάδες, περιμένει να ολοκληρωθεί η επεξεργασία της προηγούμενης τιμής και έπειτα επιστρέφει την τρέχουσα τιμή στο top level.

3.8. Χρονισμός

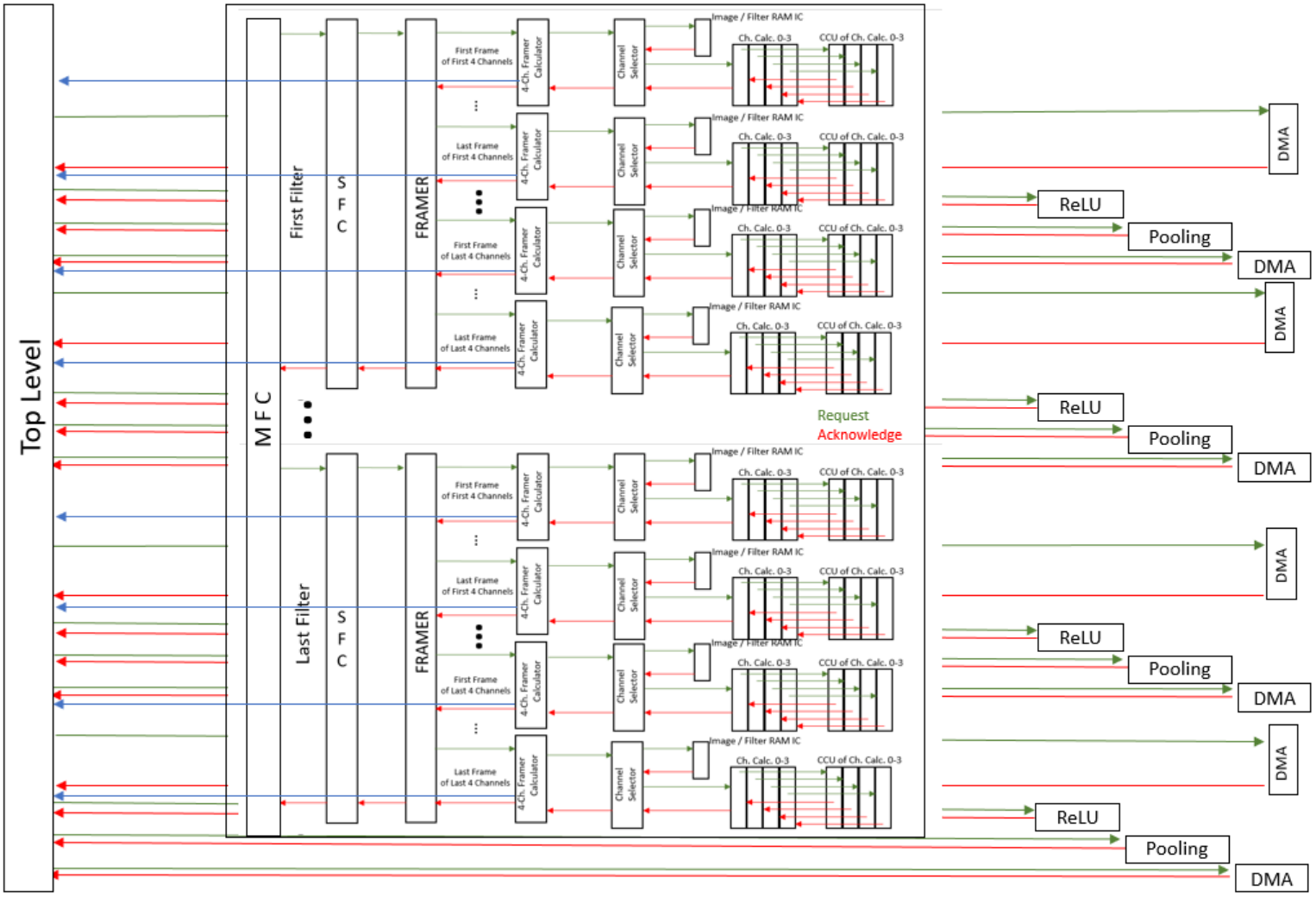
Για τον χρονισμό των μονάδων μεταξύ τους, καθώς και των επί μέρους modules εντός των μονάδων, χρησιμοποιείται ευρέως σύστημα χρονισμού Request – Acknowledge.

Μια μονάδα, η οποία για να συνεχίσει την λειτουργία της απαιτεί την ολοκλήρωση της λειτουργίας μιας άλλης μονάδας, αρχικά την ενεργοποιεί στέλνοντας ενεργό σήμα request προς αυτήν, τροφοδοτώντας την με κατάλληλα δεδομένα και συνεχίζει την λειτουργία της όταν λάβει ενεργό σήμα Acknowledge από την μονάδα που ενεργοποίησε, μπορώντας πλέον να αξιοποιήσει δεδομένα που επέστρεψε η ενεργοποιημένη μονάδα. Μόλις μια μονάδα παράξει σήμα Acknowledge για εξωτερική μονάδα η λειτουργία της ολοκληρώνεται και ενεργοποιείται πάλι μόνον εφόσον λάβει εκ νέου ενεργό σήμα Request.

Ο χρονισμός με την χρήση Request - Acknowledge στα εσωτερικά modules του Convolution/Multiple Filter Calculator φαίνεται στο παρακάτω sequence διάγραμμα.


MFC
First Filter
SFC
FRAMER
First Frame of First 4 Channels
Last Frame of First 4 Channels
First Frame of Last 4 Channels
Last Frame of Last 4 Channels
4-Ch. Framer Calculator
Channel Selector
Image / Filter RAM IC
Ch. Calc. 0-3
CCU of Ch. Calc. 0-3
Request
Acknowledge
Last Filter
SFC
FRAMER
First Frame of First 4 Channels
Last Frame of First 4 Channels
First Frame of Last 4 Channels
Last Frame of Last 4 Channels
4-Ch. Framer Calculator
Channel Selector
Image / Filter RAM IC
Ch. Calc. 0-3
CCU of Ch. Calc. 0-3

Το sequence διάγραμμα για το χρονισμό της μονάδας Convolution με ReLU και Pooling (εφόσον και τα δύο είναι ενεργοποιημένα) καθώς και για την εγγραφή των δεδομένων στην RAM μέσω DMA φαίνεται παρακάτω.



Εικόνα 15a – 15b:Sequence diagrams

3.9. Software – Hardware CoDesign

Στην υλοποίηση έχει αναπτυχθεί ένα συνελικτικό επίπεδο, ένα επίπεδο ReLU (activation function) και ένα επίπεδο υποδειγματοληψίας.

Σε ένα ΣΝΔ υπάρχουν πολλά διακριτά συνελικτικά, ReLU και υποδειγματοληπτικά επίπεδα, σε διαφορετικό πλήθος και με διαφορετική σειρά εκτέλεσης για κάθε ΣΝΔ. Μόνος περιορισμός αποτελεί ότι για να υπάρχει επίπεδο ReLU ή/και υποδειγματοληψίας, θα πρέπει πριν από αυτά να υπάρχει υποχρεωτικά συνελικτικό επίπεδο.

Στόχος μας είναι η εκτέλεση όλων των διακριτών συνελικτικών, ReLU και υποδειγματοληπτικών επιπέδων που περιέχονται στο ΣΝΔ με την σειρά που επιβάλλεται από την σχεδίαση του ΣΝΔ, χρησιμοποιώντας τις τρεις διακριτές μονάδες της υλοποίησης.

Με βάση τον παραπάνω περιορισμό, μπορούμε να διακρίνουμε τις παρακάτω τέσσερεις ενδιάμεσες καταστάσεις λειτουργίας:

- Convolution >ReLU> Pooling
- Convolution >ReLU
- Convolution > Pooling
- Convolution

Μόλις ολοκληρωθεί μία από τις παραπάνω ενδιάμεσες καταστάσεις, το ΣΝΔ θα πρέπει υποχρεωτικά να περάσει στην ίδια ή σε μία από τις υπόλοιπες τρείς, με την διαδικασία να επαναλαμβάνεται μέχρι η εκτέλεση του ΣΝΔ να φθάσει στο classification τμήμα.

Όταν η υλοποίησή μας εκτελεί μία από τις ενδιάμεσες καταστάσεις που περιέχουν δύο ή περισσότερα διακριτά επίπεδα, λειτουργεί με μεγάλο βαθμό παραλληλίας. Για παράδειγμα το επίπεδο Pooling αρχίζει να επεξεργάζεται τις τιμές που έχουν παραχθεί από το συνελικτικό επίπεδο πριν αυτό ολοκληρώσει την λειτουργία του.

Τον έλεγχο της ροής των ενδιάμεσων καταστάσεων αναλαμβάνει επεξεργαστής. Ο επεξεργαστής ενεργοποιεί κάποια ενδιάμεση κατάσταση με την παροχή κατάλληλων τιμών στις παραμέτρους της υλοποίησης μέσω των οποίων ορίζεται και εάν θα εκτελεσθούν τα επίπεδα του ReLU ή/και του Pooling.

Όταν μία ενδιάμεση κατάσταση ολοκληρωθεί, αποστέλλεται κατάλληλο exception στον επεξεργαστή έτσι ώστε να μεταβεί στην επόμενη ενδιάμεση κατάσταση θέτοντας τις εισόδους αυτής. Συγκεκριμένα ως ενεργοποιήσεις θέτει τις εξόδους της προηγούμενης ενδιάμεσης κατάστασης, μεταβιβάζοντας τις τιμές των φίλτρων του επιπέδου και τις κατάλληλες τιμές παραμέτρων.

Ο επεξεργαστής εκτός από τον συντονισμό των επιπέδων που περιεγράφηκαν παραπάνω είναι υπεύθυνος και για την εκτέλεση επιπέδων που δεν περιέχονται στην υλοποίησή μας, π.χ. fully connected, προκειμένου να παραχθούν τελικά αποτελέσματα από το ΣΝΔ.

Παρά το γεγονός ότι υλοποιούνται συγκεκριμένα επίπεδα του ΣΝΔ, με την χρήση παραλληλισμού – είτε σε επίπεδο εσωτερικών modules του συστήματος είτε με την χρήση πολλαπλών FPGAs– μπορεί να επιτευχθεί σημαντική επιτάχυνση κυρίως στο συνελικτικό επίπεδο το οποίο εκτελεί πλήθος υπολογισμών, η οποία θα οδηγήσει τελικά στην επιτάχυνση της λειτουργίας του όλου ΣΝΔ.

3.10. Διακριτές Υλοποιήσεις

Η υλοποίηση που περιεγράφηκε μέχρι στιγμής αφορά ΣΝΔ οι τιμές των οποίων έχουν ακρίβεια 9 Bit (8 bit απόλυτης τιμής και ένα bit πρόσημου). Για την αποθήκευση κάθε τιμής (είτε αυτή είναι τιμή ενεργοποίησης ή φίλτρου) απαιτούνται 2 bytes από τα οποία το ένα χρησιμοποιείται μόνο για την αποθήκευση του πρόσημου. Από το byte του πρόσημου τα 7 bits μπορούν αν χρησιμοποιηθούν για την αποθήκευση bit των απόλυτων τιμών και θα αποτελούν τα 7 MSB της απόλυτης τιμής που πλέον θα αποτελείται από 15 bits. Η προσθήκη αυτή συνολικά έχει αρνητικό αντίκτυπο τόσο όσον αφορά τους πόρους υλικού που πρέπει να χρησιμοποιηθούν, όσο και στην απόδοση σε χρόνο, καθώς πλέον για κάθε γινόμενο που παράγεται στην χειρότερη περίπτωση απαιτούνται 15 κύκλοι, έναντι 8 της προηγούμενης υλοποίησης. Ωστόσο είναι απαραίτητη η ύπαρξη της υλοποίησης αυτής καθώς υπάρχουν ΣΝΔ που ανά επίπεδο λειτουργούν με ακρίβεια τιμών 13 – 16 bits (ακέραιες ή δεκαδικές τιμές). Συνολικά, όπως φαίνεται και στον παρακάτω πίνακα [31], υπάρχουν ΣΝΔ τα layers των οποίων έχουν ακρίβεια σε bit μέχρι 9 bit και άλλα τα οποία έχουν ακρίβεια μέχρι 13 bit. Ενώ η υλοποίηση με ακρίβεια 16 bit μπορεί να καλύψει τις ανάγκες των ΣΝΔ με ακρίβεια μέχρι 9 bit επιλέγεται να διατηρηθούν και οι δύο διακριτές υλοποιήσεις καθώς η πρώτη έχει σημαντικά πλεονεκτήματα σε σχέση με την δεύτερη.

Network	Per-Layer Activation Precision in Bits
AlexNet	9-8-5-5-7
NiN	8-8-8-9-7-8-8-9-9-8-8-8
GoogLeNet	10-8-10-9-8-10-9-8-9-10-7
VGG M	7-7-7-8-7
VGG S	7-8-9-7-9
VGG _19	12-12-12-11-12-10-11-11-13-12-1313-13-13-13-13

Εικόνα 16: Ακρίβεια σε bit επιπέδων γνωστών ΣΝΔ

Όσον αφορά την δεύτερη υλοποίηση, η διαφορά της σε σχέση με την πρώτη βρίσκεται στην λειτουργία των CCU. Καθώς οι πολλαπλασιαστέοι αποτελούνται από 15 bit, για λόγους που εξηγήθηκαν στην ενότητα της CCU, απαιτείται η παράλληλη λειτουργία 15 διαφορετικών Shift Muls.

Πλέον θα αναφερόμαστε στην σχεδίαση με ακρίβεια 9 bits ως A9 και στην σχεδίαση με ακρίβεια 16 bits ως A16.

4. Έλεγχος υλοποίησης

Στο προηγούμενο κεφάλαιο αναφερθήκαμε στην μορφή που πρέπει να έχουν τα δεδομένα τα οποία εισέρχονται στο σύστημα (μέσω της εξωτερικής RAM) και συνεπώς στην μορφή των δεδομένων όπως αυτά εξέρχονται από το σύστημα. Για τον έλεγχο ορθότητας των δεδομένων που παράγει το σύστημά μας, αναπτύχθηκαν επιπλέον τα εξής:

Α. Πρόγραμμα σε γλώσσα προγραμματισμού C# για την μετατροπή της εικόνας σε κατάλληλη μορφή για επεξεργασία από το σύστημα και αντίστροφα.

Για την μετατροπή εικόνας σε κατάλληλο format για την επεξεργασία από το σύστημα, λαμβάνεται ως είσοδος αρχείο εικόνας οποιασδήποτε μορφής (.jpg, .png, .tif, κλπ.), μετατρέπεται σε bitmap και στη συνέχεια από κάθε pixel του bitmap απομονώνονται οι τιμές των bytes που αντιστοιχούν σε RGB. Δημιουργούνται έτσι τρείς πίνακες που περιέχουν το σύνολο των bytes των R, G, B, περιοχών της εικόνας.

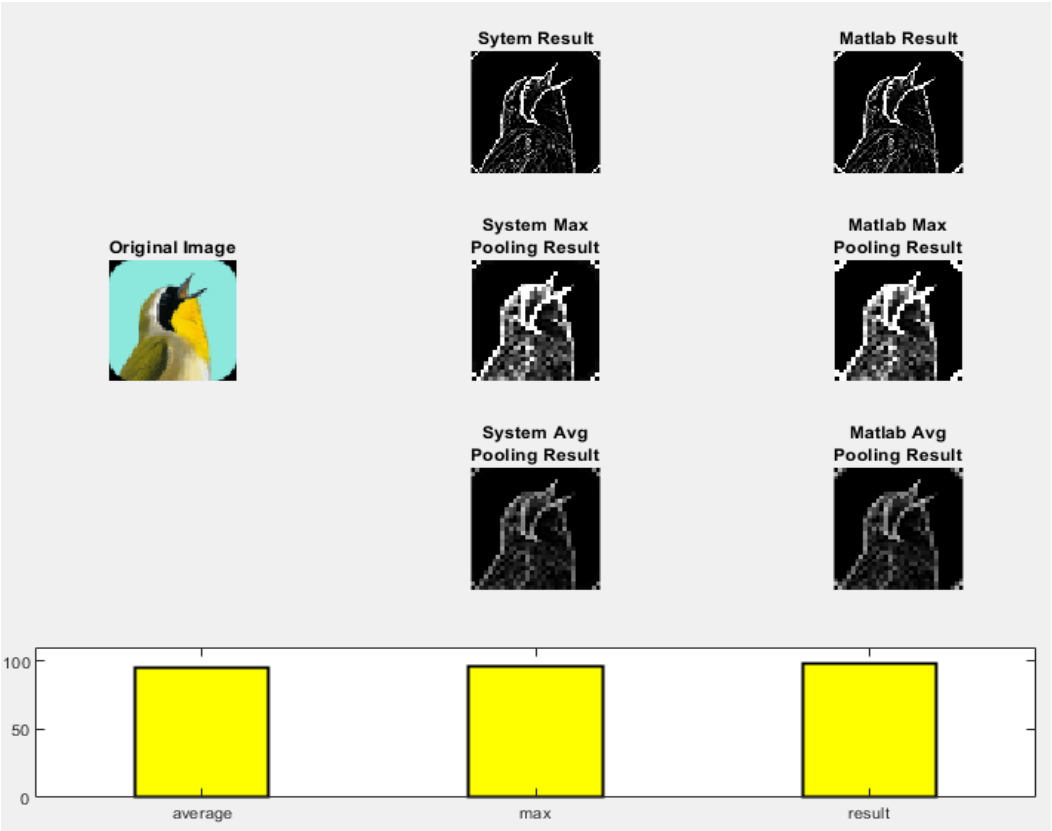
Για την μετατροπή των πινάκων αυτών σε κατάλληλη μορφή για το σύστημα, πρώτα πρέπει να επεξεργασθεί η R περιοχή και να αποθηκευτεί σε text αρχείο (που αντιπροσωπεύει την RAM), έπειτα η G περιοχή και έπειτα η B περιοχή. Για κάθε περιοχή διασχίζεται ο αντίστοιχος διδιάστατος πίνακας κατά στήλες. Τα bytes που περιέχει κάθε στήλη αποθηκεύονται ανά τετράδες σε δυαδική μορφή, σε διαδοχικές γραμμές του αρχείου που δημιουργείται. Καθώς το σύστημα προορίζεται για την χρήση RAM 64 bits δίπλα από κάθε byte που αποθηκεύεται προστίθεται και άλλο ένα byte πρόσημου (που μπορεί να περιέχει και bits της απόλυτης τιμής εφόσον προορίζεται για την A16) που εφόσον μετατρέπουμε εικόνα είναι μηδενικό (θετικό). Αντίστοιχη διαδικασία ακολουθείται και για την δημιουργία αρχείου που θα περιέχει τα δεδομένα των φίλτρων.

Για την μετατροπή του αρχείου που παράχθηκε από το σύστημα σε εικόνα/νες απαιτούνται οι προβλεπόμενες διαστάσεις του κάθε feature map καθώς και ο αριθμός φίλτρων / feature maps. Το σύνολο των δεδομένων των feature maps είναι αποθηκευμένο σε ένα text αρχείο με αντίστοιχη μορφή με το αρχείο που παράχθηκε για την είσοδο του συστήματος, με την διαφορά ότι διαδοχικά feature maps είναι αποθηκευμένα σε διαδοχικές θέσεις στο αρχείο. Για την παραγωγή εικόνας από τα δεδομένα του κάθε feature map χρησιμοποιείται η αντίστροφη διαδικασία σε σχέση με παραπάνω. Το αρχείο διαβάζεται σειρά προς σειρά και τα δεδομένα αντιστοιχίζονται σε κάποια στήλη ενός bitmap. Κάθε feature map που παράγεται από το σύστημα περιέχει ένα κανάλι. Κάθε byte δεδομένων που διαβάζει το πρόγραμμα από το αρχείο, τοποθετείται αυτούσιο στις RGB περιοχές κάθε pixel του bitmap που παράγεται. Με την ολοκλήρωση της μετατροπής σε bitmap το αρχείο αποθηκεύεται σε format .jpg. Μπορούν να τροφοδοτηθούν στο πρόγραμμα τόσο τα δεδομένα των feature maps χωρίς Pooling όσο και με Pooling με χρήση των κατάλληλων διαστάσεων.

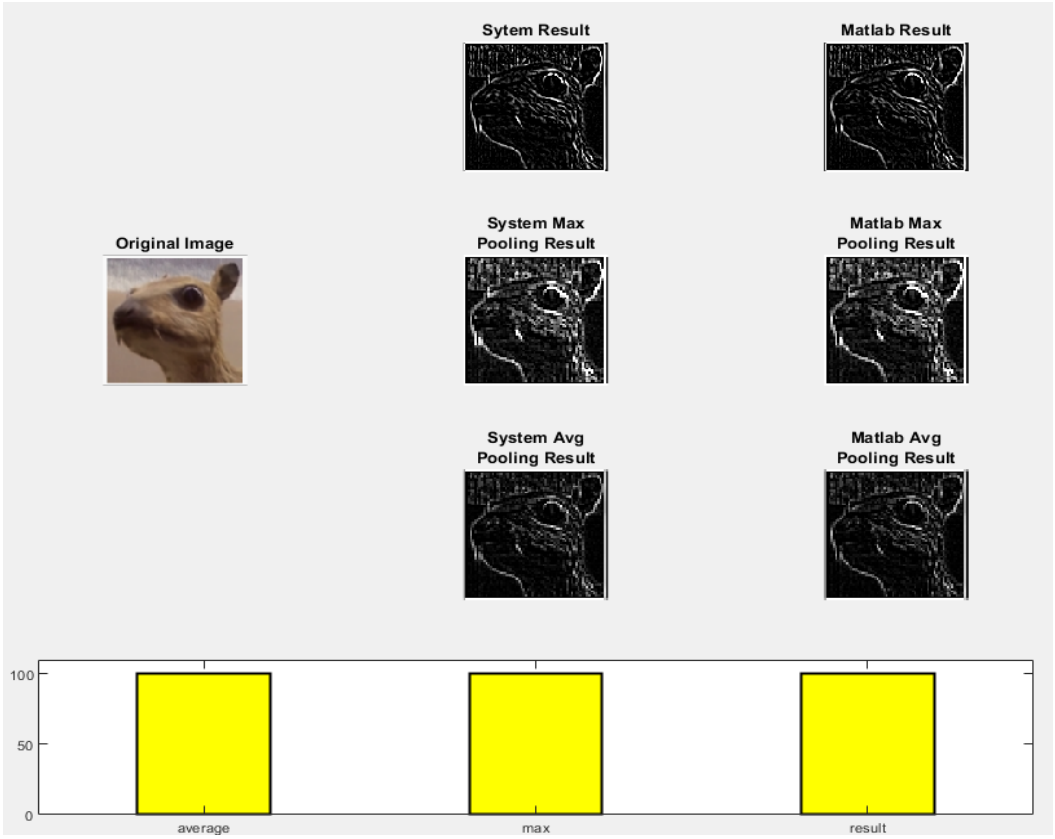
Β. Κώδικας σε MatLab για τον έλεγχο ορθότητας των feature maps που παράχθηκαν. Ο συγκεκριμένος κώδικας με την χρήση της συνάρτησης imread λαμβάνει τέσσερεις εικόνες, την πρωτότυπη, την παραχθείσα από το σύστημα χωρίς Pooling, με Avg Pooling και MaxPooling. Οι εικόνες αυτές μετατρέπονται σε ανεξάρτητους τριδιάστατους πίνακες κατάλληλων διαστάσεων. Στον κώδικα ορίζεται επίσης ένας τριδιάστατος πίνακας που αντιστοιχεί στο φίλτρο, με ίδιες τιμές με το φίλτρο που χρησιμοποιήθηκε από το σύστημα για την παραγωγή του

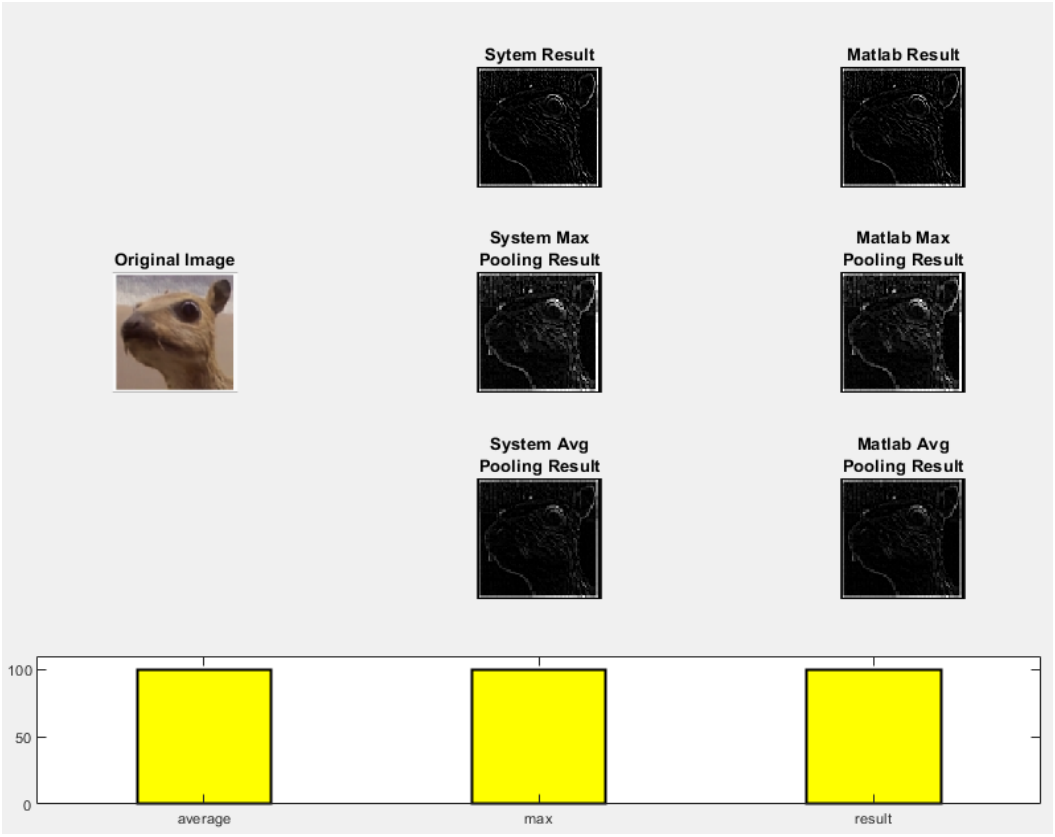
feature map. Ο πίνακας που αντιστοιχεί στην πρωτότυπη εικόνα, σαρώνεται με το πίνακα του φίλτρου πραγματοποιώντας κατάλληλους υπολογισμούς και τελικά παράγεται ένας πίνακας αποτελεσμάτων. Από τον πίνακα αποτελεσμάτων με κατάλληλη χρήση τεχνικής average και max pooling παράγονται δύο πίνακες αποτελεσμάτων pooling. Οι τρεις πίνακες αποτελεσμάτων συγκρίνονται με τους αντίστοιχους πίνακες που παράχθηκαν από τα feature maps με ή χωρίς pooling και προκύπτουν ποσοστά ομοιότητας.

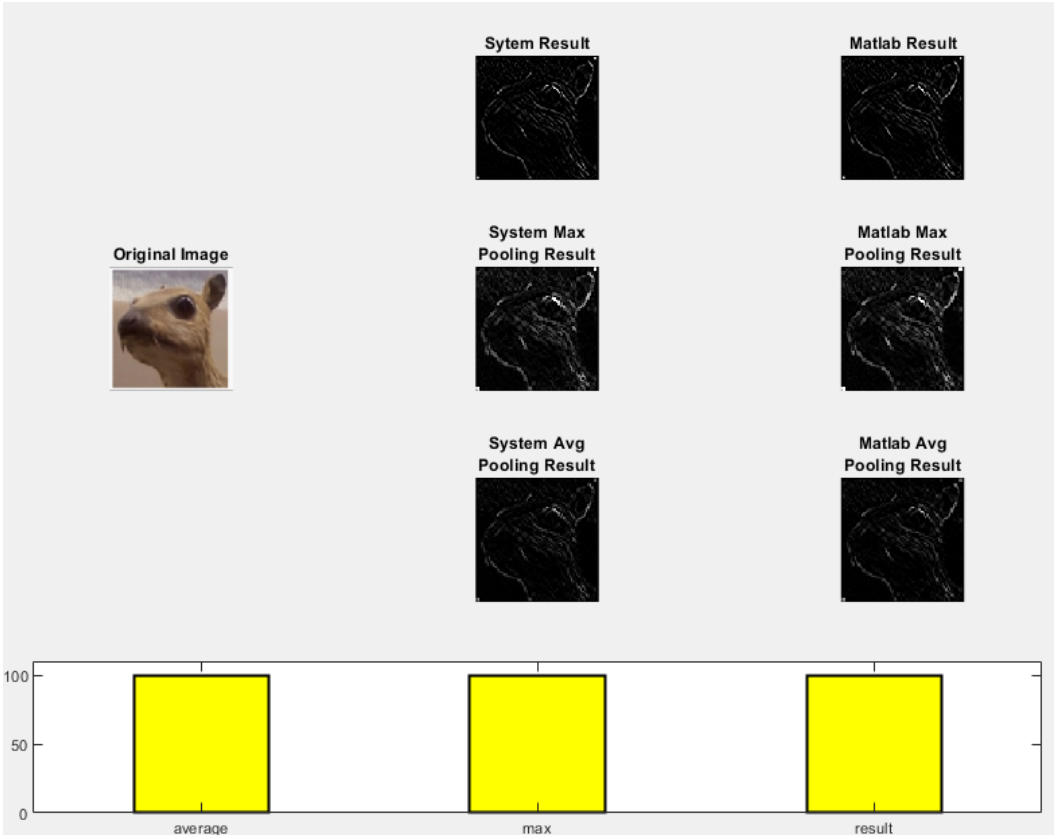
Παρακάτω φαίνονται τα αποτελέσματα της σύγκρισης όπως αυτά προέκυψαν από τον κώδικα Matlab για τρεις εικόνες διαφορετικών διαστάσεων οι οποίες σαρώθηκαν με τρία διαφορετικά φίλτρα κατάλληλα για edge detection. Στις παρακάτω εικόνες φαίνονται τα feature maps όπως αυτά προέκυψαν με την βοήθεια του προγράμματος Α όπως και τα feature maps που υπολογίστηκαν βάσει κώδικα Matlab. Σε κάθε εικόνα δίνεται ραβδοδιάγραμμα σύγκρισης των feature maps συστήματος - Matlab, με ανώτατο όριο 100% ομοιότητα (για τα αποτελέσματα χωρίς Pooling, με Max Pooling και με Average Pooling).

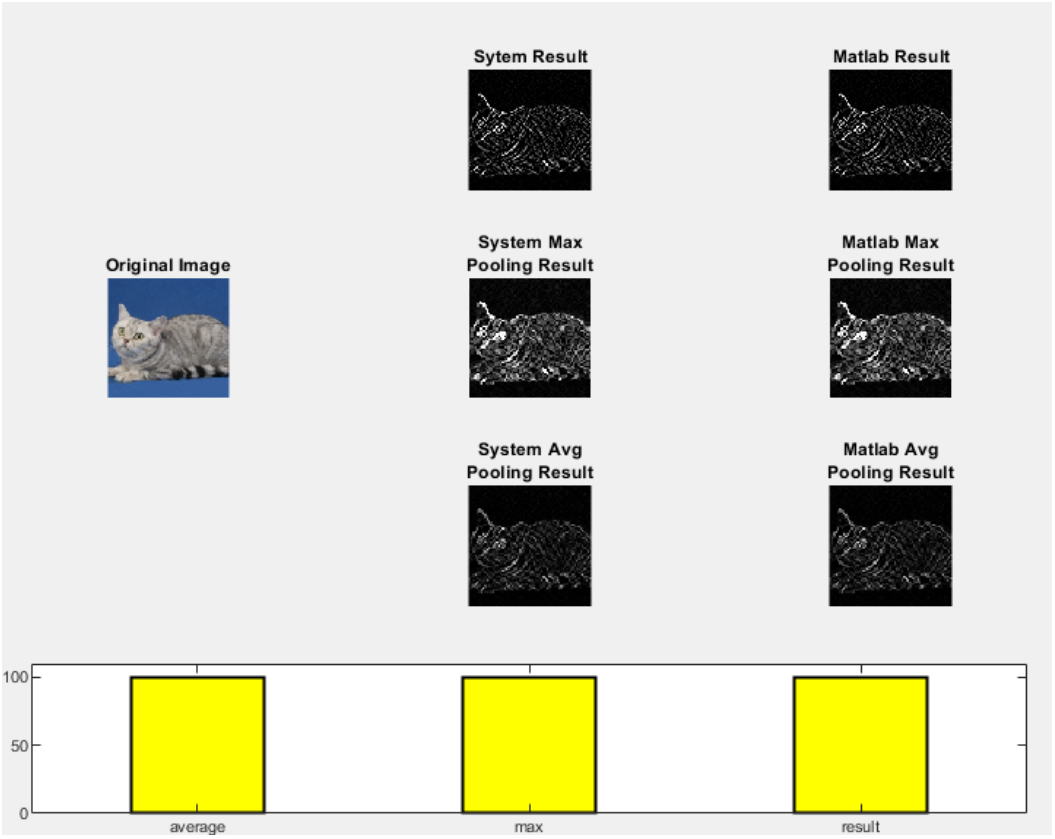

Sytem Result
Matlab Result
Original Image
System Max Pooling Result
Matlab Max Pooling Result
System Avg Pooling Result
Matlab Avg Pooling Result
100
50
0
average
max
result

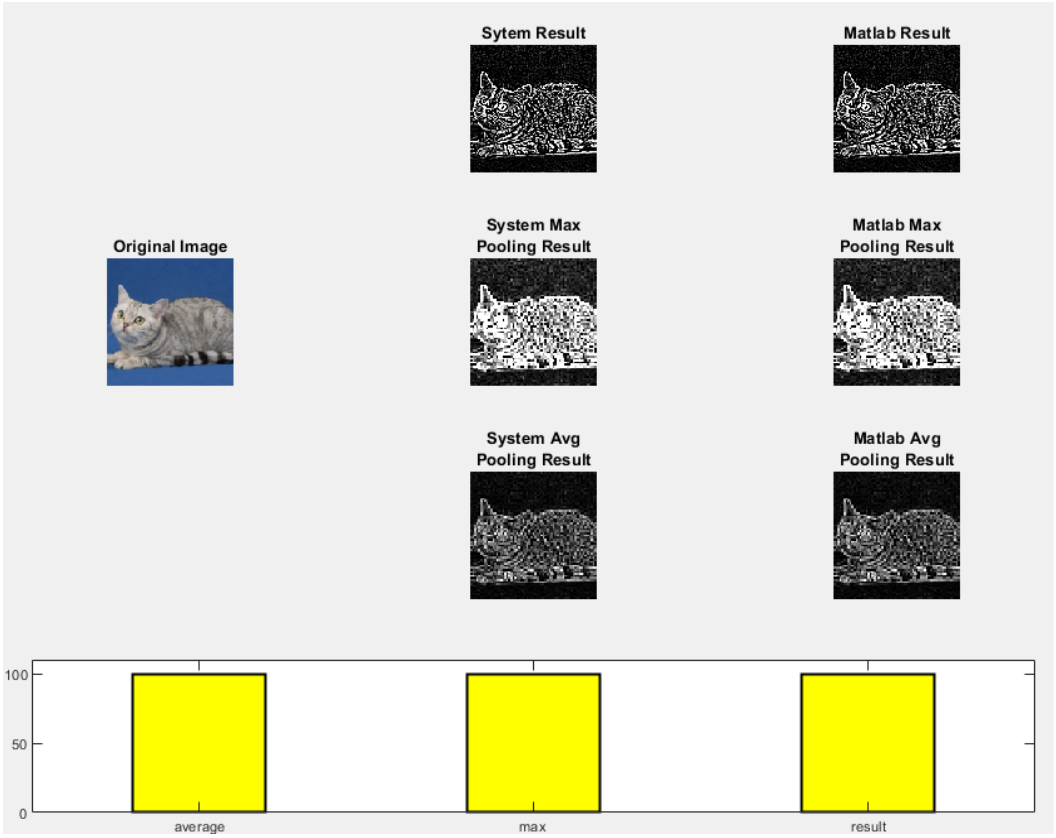

Sytem Result
Matlab Result
Original Image
System Max Pooling Result
Matlab Max Pooling Result
System Avg Pooling Result
Matlab Avg Pooling Result
100
50
0
average
max
result

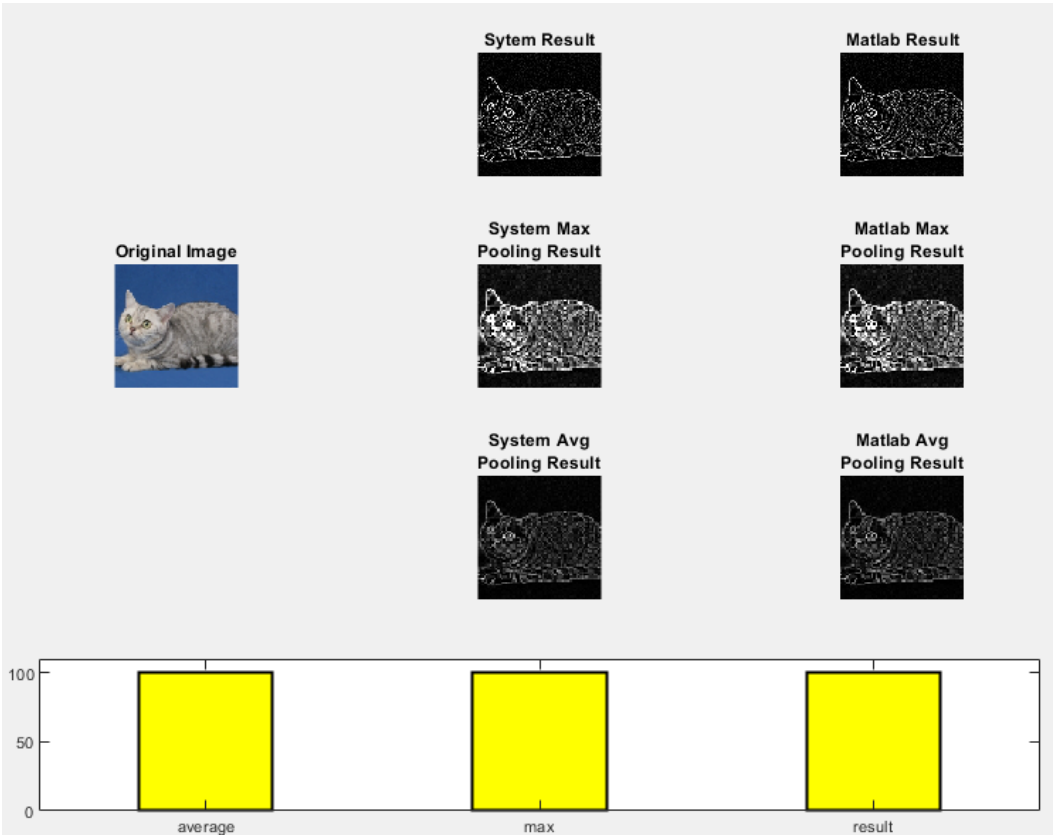

Sytem Result
Matlab Result
Original Image
System Max Pooling Result
Matlab Max Pooling Result
System Avg Pooling Result
Matlab Avg Pooling Result
100
50
0
average
max
result


Sytem Result
Matlab Result
Original Image
System Max Pooling Result
Matlab Max Pooling Result
System Avg Pooling Result
Matlab Avg Pooling Result
100
50
0
average
max
result


Sytem Result
Matlab Result
Original Image
System Max Pooling Result
Matlab Max Pooling Result
System Avg Pooling Result
Matlab Avg Pooling Result
100
50
0
average
max
result


Sytem Result
Matlab Result
Original Image
System Max Pooling Result
Matlab Max Pooling Result
System Avg Pooling Result
Matlab Avg Pooling Result
100
50
0
average
max
result





Εικόνα 17-26: Αποτελέσματα Συστήματος και έλεγχος ορθότητας με χρήση Matlab

Στα ραβδογράμματα των παραπάνω εικόνων, όλες οι συγκρίσεις που έγιναν μεταξύ αποτελεσμάτων συστήματος και Matlab έχουν ομοιότητα 100%, δηλαδή το σύνολο των αποτελεσμάτων που προκύπτουν από το σύστημα είναι τα αναμενόμενα.

5. Αξιολόγηση Υλοποίησης

5.1. FPGA Υλοποίησης

Οι πόροι υλικού που περιέχει μία FPGA καθορίζονται από τον αριθμό των slices που περιέχει, όπου κάθε slice αποτελείται από LUTs (Lookup tables) και FlipFlops. Ο αριθμός των LUTs και FlipFlops ανά slice διαφέρει ανάλογα με την οικογένεια που ανήκει το chip της FPGA. Ένα Lookup table σε γενικές γραμμές αποτελεί έναν πίνακα ο οποίος καθορίζει ποια θα είναι η έξοδος για δεδομένες εισόδους. Ο τρόπος με τον οποίο η FPGA υλοποιούν αναδιατασσόμενη λογική είναι χρησιμοποιώντας πλήθος LUTs που όταν η FPGA παραμετροποιηθεί ο πίνακας του κάθε LUT γεμίζει με τις επιθυμητές τιμές εξόδου (LUT mask). Κάθε πίνακας σε ένα LUT αποτελείται από SRAMs του ενός bit, πλήθους 2^K όπου K ο αριθμός των εισόδων του LUT. Τα FlipFlops που περιέχονται μέσα σε ένα slice έχουν την ικανότητα να διατηρήσουν μια τιμή ενός bit.

Οι RAM interconnect που αναπτύχθηκαν δεν περιλαμβάνονται στην διαδικασία της σύνθεσης, οπότε τα παρακάτω αφορούν το top level των μονάδων Convolution, ReLU και Pooling της σχεδίασης.

Για την μεταφορά της υλοποίησης σε FPGA θα γίνει χρήση FPGA της οικογένειας KINTEX 7 series, η οποία παρέχει τον καλύτερο λόγο τιμής / απόδοση σε σχέση με τις αντίστοιχες οικογένειες ARTIX και VIRTEX. Οι FPGA που ανήκουν σε αυτήν την οικογένεια σε κάθε slice περιέχουν τέσσερα LUTs και οκτώ FlipFlops. Κάθε LUT έχει έξι εισόδους επομένως για τον πίνακα χρησιμοποιούνται 2^6 SRAM του ενός bit [28].

Σχεδιάσθηκαν δύο διαφορετικές υλοποιήσεις του ίδιου συστήματος, μία που υποστηρίζει τιμές φίλτρων και ενεργοποιήσεων με ακρίβεια 9 bit (8 bit τιμής και 1 πρόσημου) – A9 και μία που υποστηρίζει τιμές φίλτρων και ενεργοποιήσεων με

ακρίβεια 16 bit (15 bit τιμής και 1 πρόσημου) – A16. Η υλοποίηση με την μικρότερη ακρίβεια είναι λιγότερο απαιτητική σε πόρους υλικού από ότι η υλοποίηση με τη μεγαλύτερη ακρίβεια, κυρίως καθώς στην CCU χρησιμοποιείται μικρότερος αριθμός παράλληλων Shift-Muls.

Όσον αφορά την A9, κατά την διαδικασία του Synthesis χρησιμοποιήθηκε ως target device από την οικογένεια KINTEX 7 η FPGA XC7K70T, η οποία αποτελείται από 10.250 slices με συνολικό αριθμό LUTs 41.000 και FlipFlops 82.000. Η σχεδίαση A9 αξιοποιεί 39.109 LUTs (95%) του συνόλου και 27.499 D-FlipFlops (Registers) (33%). Η συγκεκριμένη FPGA υποστηρίζει επίσης μέχρι 135 Block RAM – FIFO των 36 KB από τις οποίες αξιοποιούνται οι 24 (17%).

Συνολικά εφόσον η σχεδίαση A9 χρησιμοποιεί το 95% των διαθέσιμων LUTs καταλαβαίνουμε ότι η σχεδίαση A16 δεν είναι δυνατόν να τοποθετηθεί στην συγκεκριμένη FPGA.

Για την σχεδίαση A16 χρησιμοποιείται, λοιπόν, FPGA της ίδιας οικογένειας η XC7K160T, η οποία αποτελείται από 25.350 slices με συνολικό αριθμό LUTs 101.400 και FlipFlops 202.800. Η σχεδίαση A16 αξιοποιεί 75.555 LUTs (74%) εκ του συνόλου και 53.229 D-FlipFlops (Registers) (26%). Η συγκεκριμένη FPGA υποστηρίζει επίσης μέχρι 325 Block RAM – FIFO των 36 KB από τις οποίες αξιοποιούνται οι 24 (7%).

Στον παρακάτω πίνακα παρουσιάζονται συγκεντρωτικά οι πόροι υλικου που απαιτούνται για τις δύο διακριτές υλοποιήσεις στις FPGA για τις οποίες προορίζονται. Ενώ η A9 μπορεί στην FPGA XC7K70T, επιλέγεται να τοποθετηθεί στην XC7K160T για λόγους σύγκρισης.

	A9	**A9**	**A16**
FPGA	XC7K70T	XC7K160T	XC7K160T
Slices	10.250	25.350	25.350
LUTs (Δεσμευμένα / Σύνολο)	39.109 / 41.000 **(94%)**	39.109 / 101.400 **(38%)**	75.555 / 101.400 **(74%)**
Registers (Δεσμευμένα / Σύνολο)	27.499 / 82.000 **(33%)**	27.499 / 202.800 **(13%)**	53.229 / 202.800 **(26%)**
36 KB Block RAM – FIFO (Δεσμευμένα / Σύνολο)	24 /135 **(17%)**	24 / 325 **(7%)**	24 / 325 **(7%)**

Πίνακας 1: Απαιτούμενοι πόροι υλικού ανά υλοποίηση και FPGA

5.2. Χρόνος εκτέλεσης

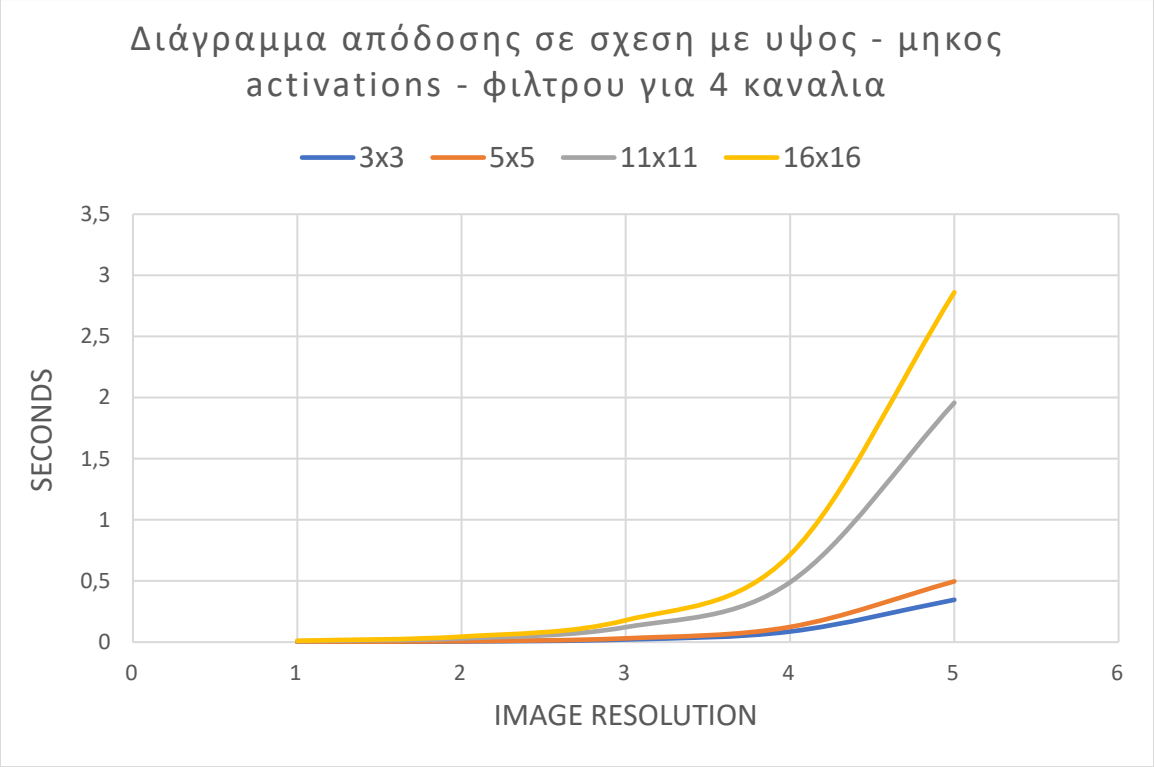
Ο χρόνος που απαιτείται για την ολοκλήρωση της λειτουργία του συστήματος μεταβάλλεται βάσει των παρακάτω παραγόντων:

- Διαστάσεις ενεργοποιήσεων (ύψος και μήκος)
- Διαστάσεις φίλτρου (ύψος και μήκος)
- Αριθμός καναλιών των ενεργοποιήσεων και του φίλτρου
- Πλήθος άσσων στις τιμές του φίλτρου και των ενεργοποιήσεων (Bit Pragmatic)

Ο χρόνος εκτέλεσης του συστήματος αυξάνεται για ενεργοποιήσεις μεγαλύτερων διαστάσεων καθώς το εκάστοτε φίλτρο πρέπει να τοποθετηθεί σε περισσότερες περιοχές των ενεργοποιήσεων. Συγκεκριμένα για διπλασιασμό ύψους και μήκους ενεργοποιήσεων τετραπλασιάζονται οι αναγκαίοι υπολογισμοί.

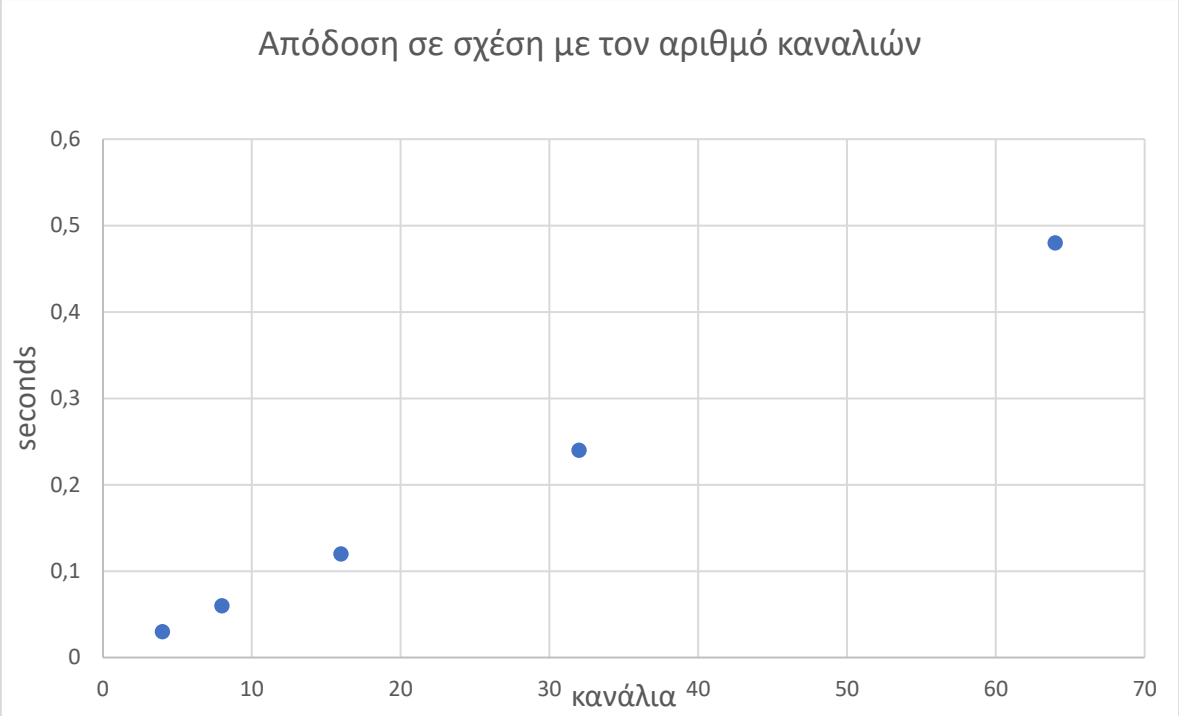
Όσον αφορά την αύξηση του μεγέθους των διαστάσεων του φίλτρου η λειτουργία του συστήματος γίνεται πιο χρονοβόρα καθώς για κάθε εφαρμογή του επάνω στις ενεργοποιήσεις απαιτούνται περισσότερα στοιχεία των ενεργοποιήσεων, ίσα σε πλήθος με τα στοιχεία του φίλτρου. Κατά αυτόν τον τρόπο αυξάνονται τόσο οι αιτήσεις προς την RAM για δεδομένα, όσο και οι χρόνοι επεξεργασίας των δεδομένων από τις CCU.

Στο διάγραμμα που ακολουθεί φαίνονται οι χρόνοι λειτουργίας του συστήματος για φίλτρα και ενεργοποιήσεις διαφόρων διαστάσεων, θεωρώντας ότι ενεργοποιήσεις και φίλτρα αποτελούνται από τέσσερα κανάλια και κάθε τιμή του φίλτρου περιέχει σε δυαδική μορφή έναν άσσο. Το διάγραμμα αφορά την σάρωση των ενεργοποιήσεων από ένα φίλτρο.



Διάγραμμα 1: Διάγραμμα απόδοσης σε σχέση με ύψος – μήκος ενεργοποιήσεων – φίλτρου για 4 κανάλια

Στις υλοποιήσεις που παρουσιάσθηκαν, όπου χρησιμοποιείται μόνο ένας 4 Channel Frame Calculator, ο υπολογισμός τετράδων καναλιών, ενεργοποιήσεων και φίλτρων, εκτελείται σειριακά οπότε ο χρόνος λειτουργίας σε σχέση με το παραπάνω διάγραμμα πολλαπλασιάζεται. Με την ύπαρξη ν τετράδων καναλιών ο χρόνος εκτέλεσης σε σχέση με το παραπάνω διάγραμμα πολλαπλασιάζεται με το ν. Η συμπεριφορά αυτή φαίνεται στο παρακάτω διάγραμμα για ενεργοποιήσεις διαστάσεων 256 X 256 με φίλτρο 5 X 5 τεσσάρων καναλιών.

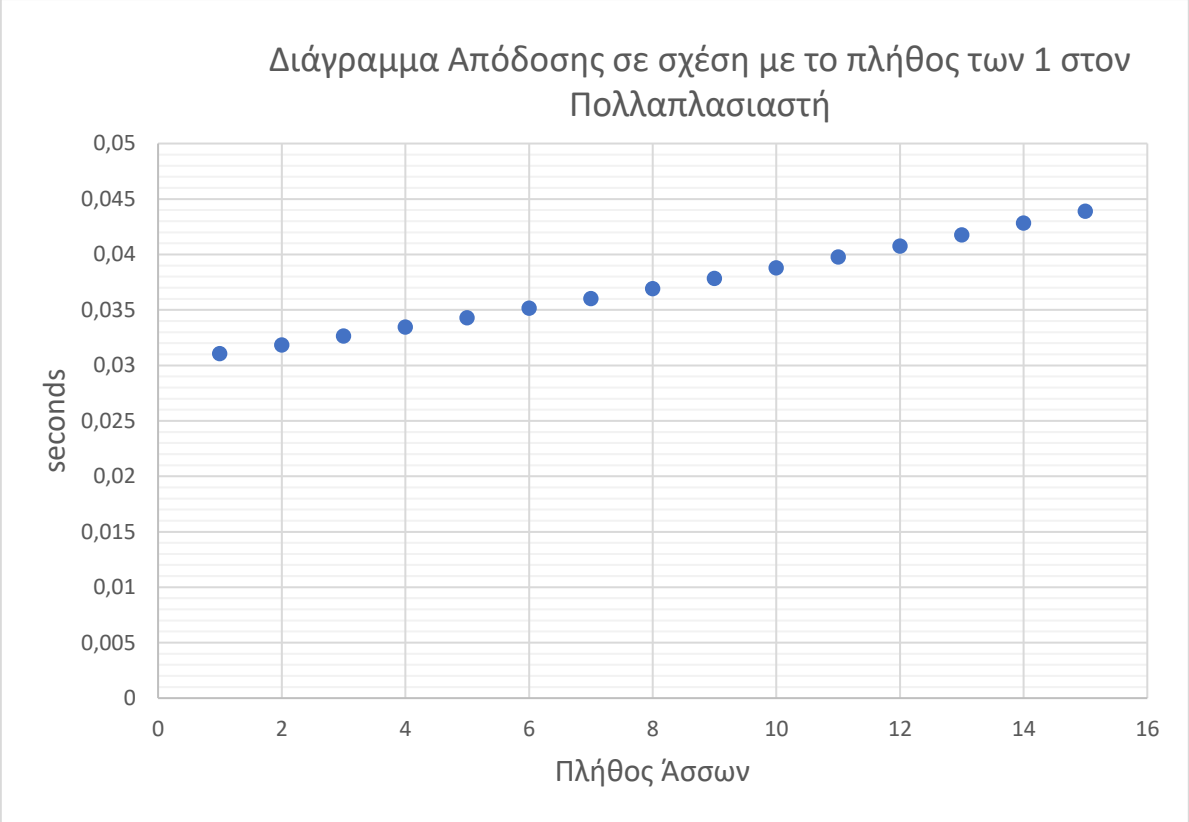


Διάγραμμα 2: Απόδοση σε σχέση με τον αριθμό καναλιών

Χρησιμοποιώντας την μέθοδο Bit Pragmatic, το πλήθος των bits που αποτελούν άσσο, σε τιμές των ενεργοποιήσεων και φίλτρων, σχετίζονται άμεσα με τον χρόνο που απαιτείται για την παραγωγή του γινομένου και σε μεγαλύτερη κλίμακα με τον χρόνο εκτέλεσης του συστήματος.

Στο παρακάτω διάγραμμα φαίνονται οι αποκλίσεις σε χρόνους εκτέλεσης του συστήματος για την παραγωγή feature map από φίλτρο του οποίου οι τιμές έχουν μεταβλητό πλήθος άσσων.

Χρησιμοποιήθηκαν ενεργοποιήσεις διαστάσεων 256 X 256 και φίλτρο 5 X 5.



Διάγραμμα 3: Απόδοση σε σχέση με το πλήθος των bits που είναι 1 στον Πολ/στή

Το παραπάνω διάγραμμα δίνει ποιοτικά χαρακτηριστικά καθώς χρησιμοποιήθηκαν ενεργοποιήσεις κάθε τιμή των οποίων έχει μεταβλητό πλήθος άσσων, με αποτέλεσμα στην πράξη του πολλαπλασιασμού, σε περίπτωση που η τιμή των ενεργοποιήσεων περιέχει μικρότερο πλήθος άσσων από την τιμή του φίλτρου να τίθεται ως πολλαπλασιαστής η τιμή της ενεργοποίησης.

Προκειμένου να επιτευχθούν καλύτεροι χρόνοι εκτέλεσης, μπορούν να υλοποιηθούν οι παρακάτω τρείς προσεγγίσεις:

1. Χρήση πολλαπλών μονάδων Single Filter Calculator εντός των υλοποιήσεων. Η προσθήκη Single Filter Calculator μπορεί να οδηγήσει στην μείωση του χρόνου εκτέλεσης με τους εξής τρόπους:

- Στα πλαίσια της επεξεργασίας των ενεργοποιήσεων από ένα φίλτρο, μπορεί να διαιρεθεί ο όγκος δεδομένων που περιέχονται στις ενεργοποιήσεις σε ισομεγέθη τμήματα, όπου το κάθε τμήμα θα ανατεθεί σε έναν Single Filter Calculator. Κατά αυτόν τον τρόπο η διαδικασία της σάρωσης των

ενεργοποιήσεων από το φίλτρο παραλληλοποιείται με αποτέλεσμα να μειώνεται ο χρόνος εκτέλεσης του συστήματος ανάλογα με το πλήθος των Single Filter Calculators.

- Κάθε Single Filter Calculator αναλαμβάνει την σάρωση των ενεργοποιήσεων με ένα διαφορετικό φίλτρο, με αποτέλεσμα να μειώνεται ο χρόνος εκτέλεσης του συστήματος συνολικά, όταν οι ενεργοποιήσεις πρέπει να σαρωθούν από πολλαπλά φίλτρα, ανάλογα με τον πλήθος των Single Filters Calculators που λειτουργούν παράλληλα.

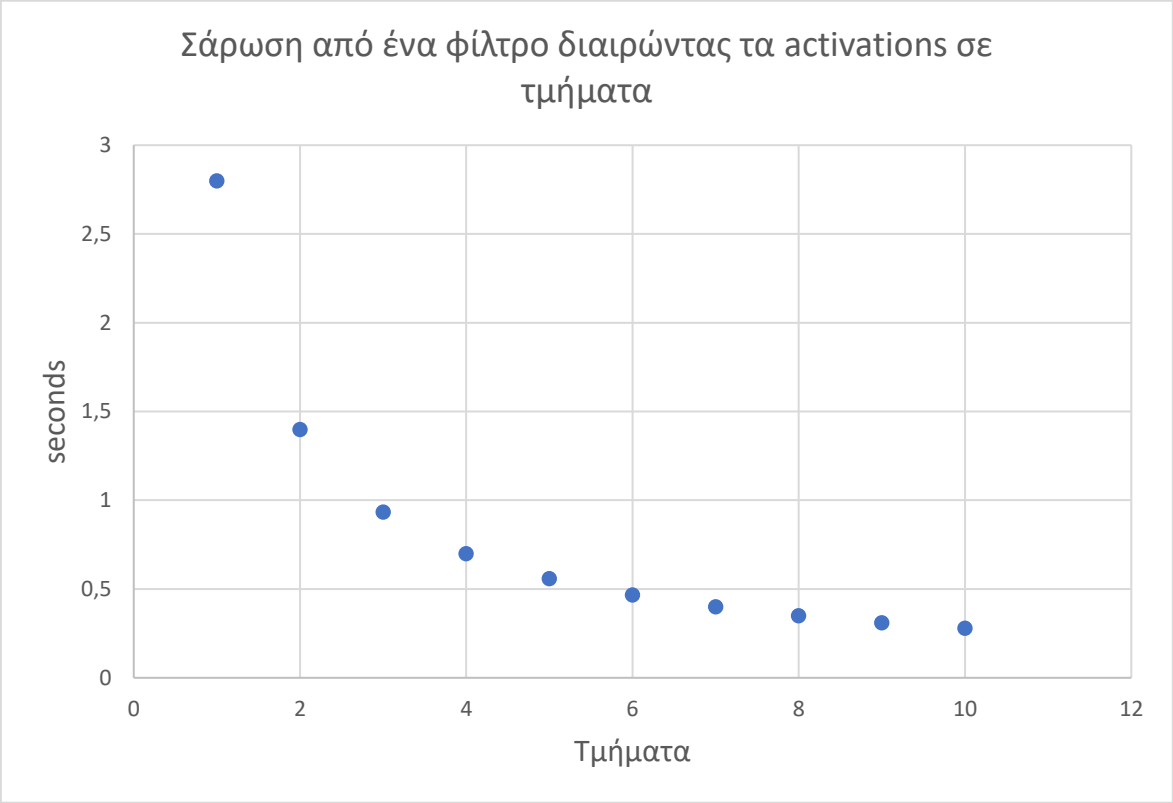
2. Χρήση πολλαπλών 4 – Channel Frame Calculators για την ταυτόχρονη επεξεργασία πολλαπλών τετράδων καναλιών ενεργοποιήσεων και φίλτρων. Κατά αυτόν τον τρόπο παράγονται τα αποτελέσματα συνέλιξης πολλαπλών τετράδων καναλιών κατά την εφαρμογή του εκάστοτε φίλτρου επάνω σε μια περιοχή των ενεργοποιήσεων, σε ίδιο χρόνο με την επεξεργασία μιας τετράδας καναλιών από τις τρέχουσες υλοποιήσεις.

3. Συνδυασμός των δύο παραπάνω προσεγγίσεων, δηλαδή χρήση πολλαπλών Single Filter Calculators, όπου κάθε Single Filter Calculator θα περιέχει πολλαπλά 4 – Channel Frame Calculators. Με αυτόν το τρόπο μειώνεται ο χρόνος εκτέλεσης δραστικά συνδυάζοντας τα οφέλη που δίνονται από την παράλληλη επεξεργασία πολλαπλών τετράδων καναλιών και την παράλληλη σάρωση των ενεργοποιήσεων από πολλά φίλτρα ή/και την ταυτόχρονη σάρωση διαφορετικών τμημάτων των ενεργοποιήσεων από το ίδιο φίλτρο.

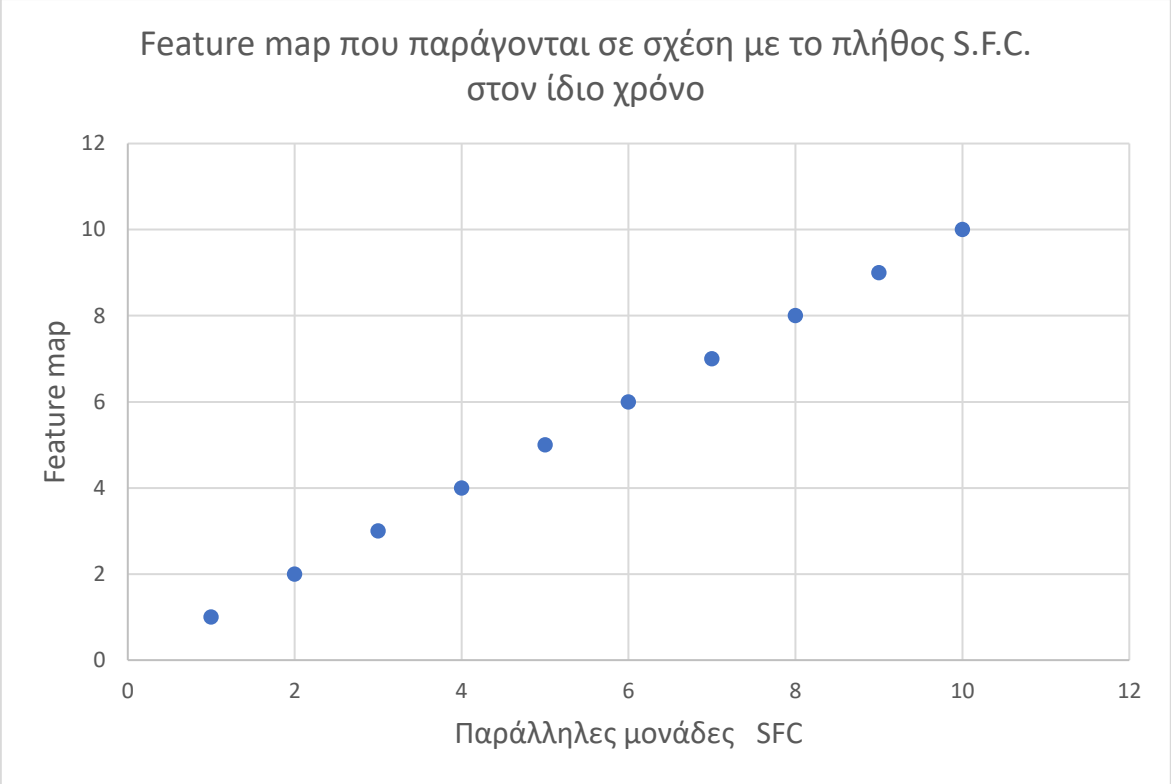
Προφανώς για την προσθήκη παράλληλων modules απαιτούνται επιπλέον πόροι υλικού και συνεπώς μεγαλύτερες FPGA. Για κάθε προσθήκη παραλληλισμού πρέπει το bandwidth του συστήματος να ικανοποιείται από τον bandwidth της RAM. Εναλλακτική αποτελεί η χρήση πολλαπλών chip– με κάθε chip να διαθέτει δική του RAM-, τα οποία λειτουργούν παράλληλα, έχοντας την δυνατότητα να επεξεργαστούν πολλαπλά φίλτρα για τις ίδιες ενεργοποιήσεις, ή διαφορετικά τμήματα των ενεργοποιήσεων για το ίδιο φίλτρο.

Η επίδραση των τριών παραπάνω προσεγγίσεων στον χρόνο εκτέλεσης του συστήματος, φαίνεται στα παρακάτω διαγράμματα. Θεωρείται ότι ο εσωτερικός παραλληλισμός που χρησιμοποιείται μειώνει γραμμικά τον συνολικό χρόνο εκτέλεσης του συστήματος. Σε πραγματικές υλοποιήσεις μπορεί να παρατηρηθούν

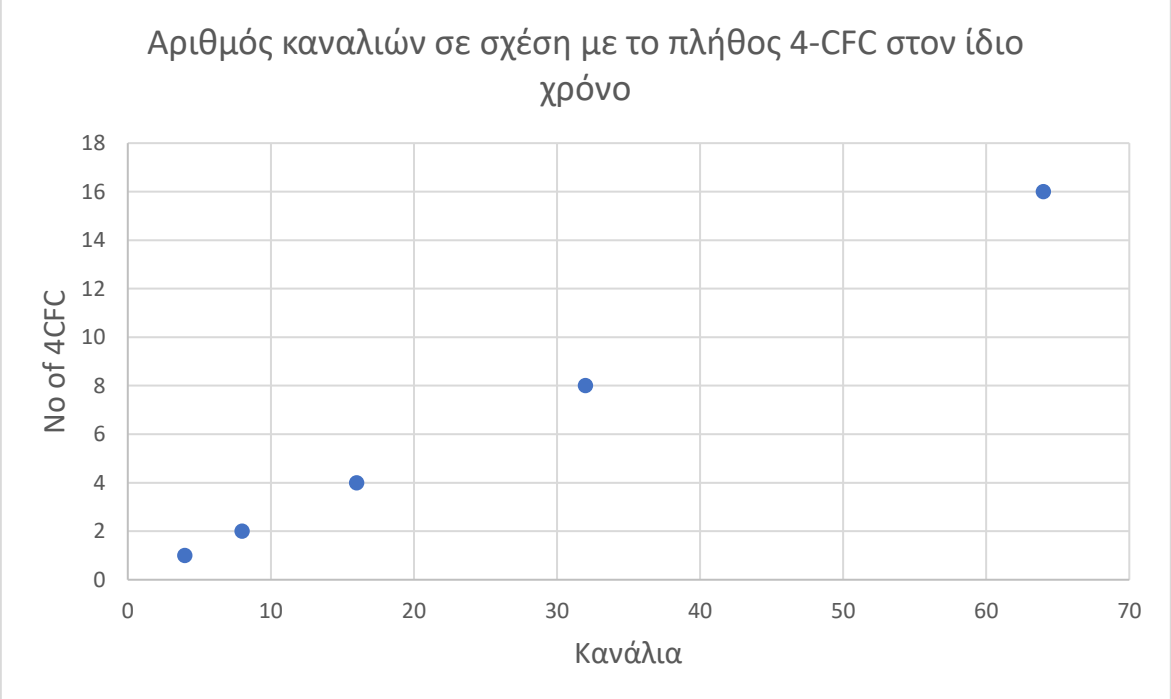
μικρές αποκλίσεις. Τα διαγράμματα αφορούν ενεργοποιήσεις διαστάσεων 1024 X 1024 που σαρώνονται από φίλτρο 16 X 16 τεσσάρων καναλιών.



Διάγραμμα 4: Χρόνος σάρωσης από ένα φίλτρο με χρήση παραλληλισμού διαιρώντας τις ενεργοποιήσεις σε τμήματα



Διάγραμμα 5: Feature map που παράγονται σε σχέση με το πλήθος S.F.C. στον ίδιο χρόνο



Διάγραμμα 6: Αριθμός καναλιών σε σχέση με το πλήθος 4-CFC στον ίδιο χρόνο

5.3. RAM Συστήματος

Για τον υπολογισμό του Bandwidth στις δύο υλοποιήσεις, είναι απαραίτητος ο υπολογισμός των συνολικών κύκλων, μετά την αρχικοποίηση, για την παραγωγή ενός feature map από ένα συγκεκριμένο φίλτρο. Η πιο απαιτητική περίπτωση για την RAM, είναι όταν το σύνολο των τιμών του φίλτρου περιέχει σε δυαδική τιμή έναν και μόνο άσσο (λόγω της λειτουργίας της CCU), οι ενεργοποιήσεις προς επεξεργασία περιέχουν τουλάχιστον τέσσερα κανάλια, το φίλτρο έχει τις μέγιστες διαστάσεις δηλαδή 16 X 16 και χρησιμοποιείται stride = 1.

Χρησιμοποιήθηκαν, λοιπόν, ενεργοποιήσεις τεσσάρων καναλιών με διαστάσεις ίσες με τις μέγιστες που δέχεται το σύστημα (1024 X 1024) και μετρήθηκε με την βοήθεια του εργαλείου ISIM ο χρόνος σε nanoseconds που απαιτείται για την παραγωγή του feature map.

Η σάρωση των ενεργοποιήσεων των τεσσάρων καναλιών από ένα φίλτρο ολοκληρώθηκε σε 491.147.968 κύκλους. Με τις ακραίες αυτές παραμέτρους συστήματος, σε λειτουργία των FPGA όπως ορίζεται από το Synthesis στα 171 MHz, ενεργοποιήσεις με τις συγκεκριμένες διαστάσεις με τέσσερα κανάλια θα σαρωθούν από το φίλτρο (16 X16) σε 2,87 sec. Για την εύρεση του bandwidth απαιτείται επιπλέον ο όγκος δεδομένων που αιτούνται από την RAM. Συγκεκριμένα:

- 16 X 16 = 4096 τιμές του φίλτρου για ένα κανάλι, δηλαδή 16.384 τιμές του φίλτρου για το σύνολο των καναλιών.
- Από τις ενεργοποιήσεις διαστάσεων 1024 X 1024 για κάθε στοιχείο κάθε γραμμής αιτούμαστε στοιχεία μεγέθους ίσου με το ύψος του φίλτρου -16- που αντιστοιχούν σε μια στήλη της περιοχής που καλύπτει το φίλτρο κάθε φορά. Για κάθε οριζόντια μετακίνηση του φίλτρου αιτούμαστε την επόμενη στήλη δεδομένων, λαμβάνοντας συνολικά για κάθε γραμμή 1024 στήλες δεδομένων. Κατά την κάθετη μετακίνηση το φίλτρο θα διασχίσει συνολικά 1024-16+1 γραμμές. Συνολικά δηλαδή για την σάρωση του συνόλου των ενεργοποιήσεων από ένα φίλτρο 16 X 16 με stride 1 θα λάβουμε (1024-16+1) X 1024 X 16 = 16.531.556 τιμές για κάθε κανάλι και συνολικά και για τα τέσσερα κανάλια 66.125.824 τιμές.

Γενικά το πλήθος δεδομένων που αιτούμεθα από την RAM για τέσσερα κανάλια και stride 1, ισούται με:

$$(Hact - Fdim + 1) * Wact * Fdim * 4$$

Κάθε τιμή και για τις δύο υλοποιήσεις, είτε ανήκει σε φίλτρο, ή σε ενεργοποιήσεις αποτελείται από δύο bytes. Συνολικά ζητήθηκαν από την RAM 66.142.208 τιμές των δύο bytes δηλαδή 1.058.275.328 bits.

Το bandwidth είναι το πηλίκο των δεδομένων που ζητήθηκαν από την RAM προς το χρόνο που απαιτείται για την διεκπεραίωση της λειτουργίας.

$$Bandwidth = data/time$$

Προκύπτει bandwidth 358.737.048 bits/sec δηλαδή:

$$Bandwidth_{system} = 359\ Mbits/s$$

Το bandwidth δεν επηρεάζεται από την επεξεργασία πολλαπλών τετράδων καναλιών ή/και πολλαπλών φίλτρων καθώς η επεξεργασία τους γίνεται σειριακά.

Οι RAM οι οποίες χρησιμοποιούνται και στις δύο υλοποιήσεις έχουν bus width = 64 bits. Το bandwidth μιας ιδανικής RAM (χωρίς να συνυπολογίζονται οι κύκλοι που απαιτούνται για τα commands) είναι:

$$Bandwidth_{ideal\ RAM} = bus\ width * Clk_{freq}$$

Στην περίπτωση ιδανικής RAM η συχνότητα λειτουργίας της προκύπτει ότι πρέπει να είναι μεγαλύτερη από 5.60 MHz. Για μη ιδανικές RAM, όπου απαιτούνται κύκλοι για τις εντολές - διαφορετικού πλήθους για κάθε κατασκευαστή -, το bandwidth είναι :

$$Bandwidth_{RAM} = bus\ width * Clk_{freq} * (\frac{burst_{cycles}}{commands_{cycles} + burst_{cycles}})$$

Σε περίπτωση που χρησιμοποιηθούν πολλαπλά 4-Channel Frame Calculators ή πολλαπλές μονάδες Single Filter Calculator (οι οποίες σαρώνουν διαφορετικές περιοχές των ενεργοποιήσεων) τότε το Bandwidth αυξάνει ανάλογα, καθώς ο χρόνος που απαιτείται για την διεκπεραίωση της λειτουργίας παραμένει σχετικά σταθερός, ενώ τα δεδομένα τα οποία ζητούνται δίνονται από τον τύπο:

$$S * ((Hact - Fdim + 1) * Wact * Fdim * 4) * C$$

Δηλαδή το bandwidth που προκύπτει είναι :

$$Bandwidth_{parallel} = S * Bandwidth_{system} * C$$

Όπου S ο αριθμός των παράλληλων μονάδων Single Filter Calculator και C ο αριθμός των παράλληλων μονάδων 4-Channel Frame Calculator.

5.4. Συγκρίσεις

Αρχικά, επιχειρείται η σύγκριση των υλοποιήσεων σε σχέση με το χρόνο εκτέλεσης προγράμματος σε C, το οποίο προσομοιώνει την απαραίτητη λειτουργικότητα. Η εκτέλεση του προγράμματος είναι σειριακή χωρίς την χρήση παραλληλισμού. Ο επεξεργαστής ο οποίος χρησιμοποιήθηκε είναι ο Intel i7 – 8700K. Ο επεξεργαστής αυτός συγκρίνεται σε κάθε πίνακα με FPGA με συχνότητα λειτουργίας 171MHz (όπως έχει προκύψει από το synthesis των δύο διαφορετικών υλοποιήσεων στις διαφορετικές FPGA - XC7K70T και XC7K160T). Στις περιπτώσεις που χρησιμοποιείται παραλληλισμός, θεωρείται ότι η FPGA στην οποία θα τοποθετηθεί η εκάστοτε υλοποίηση, θα έχει επίσης συχνότητα λειτουργίας 171 MHz.

Ο παρακάτω πίνακας συγκρίσεων αφορά το συνελικτικό επίπεδο με είσοδο ενεργοποιήσεις διαστάσεων 512 X 512 και φίλτρο 5 X 5 τεσσάρων καναλιών. Οι ενεργοποιήσεις σαρώθηκαν από ένα φίλτρο.

	FPGA	CPU
Clock Frequency (MHz)	171	3.800
Throughput (Activations/s)	8,33	13,18
Latency (s)	0,12	0,075
Αριθμός Πυρήνων	-	6
Αριθμός Λογικών Πυρήνων	-	12

Πίνακας 2: Σύγκριση Throughput, Latency σε σχέση με CPU

Παρά την μεγάλη διαφορά σε συχνότητα λειτουργίας μεταξύ CPU και FPGA, το Throughput και το Latency έχουν συγκρίσιμες τιμές. Καθώς η FPGA λειτουργεί σε

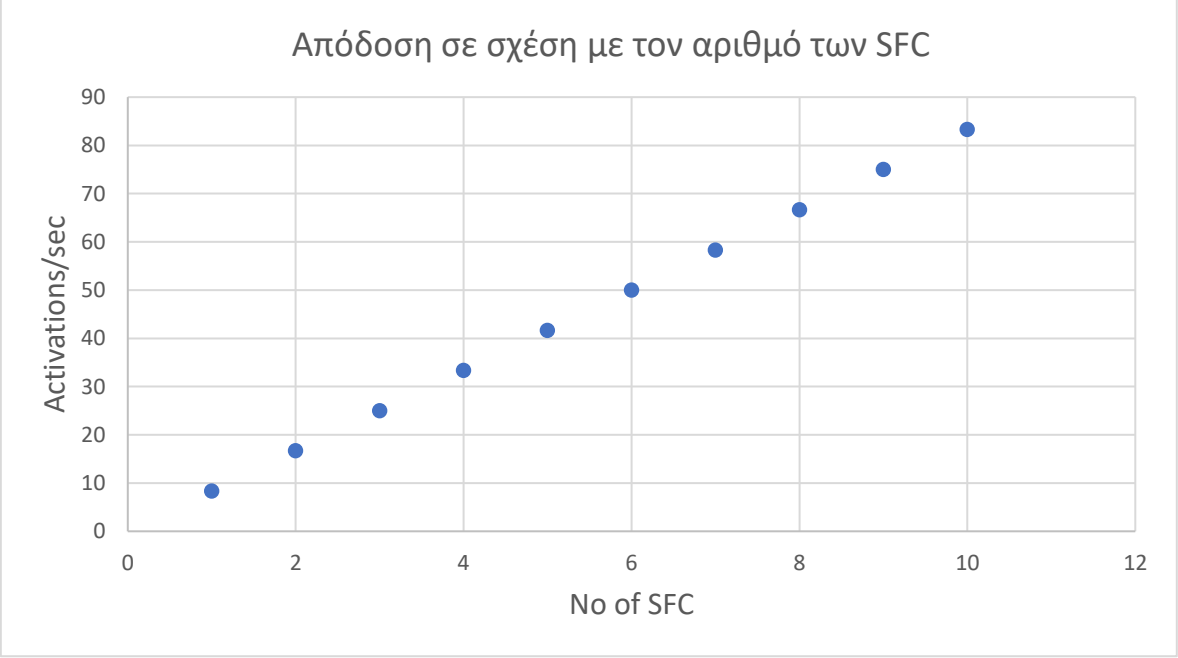
πολύ χαμηλότερη συχνότητα σε σχέση με την CPU, αναμένεται και χαμηλότερη κατανάλωση ενέργειας. Σύμφωνα με το διάγραμμα 4, με την χρήση παραλληλισμού και συγκεκριμένα με την χρήση ενός επιπλέον Single Filter Calculator, ο χρόνος εκτέλεσης του FPGA υποδιπλασιάζεται καθώς πλέον το σύνολο του όγκου των δεδομένων των ενεργοποιήσεων διαιρείται σε δύο τμήματα, των οποίων η επεξεργασία γίνεται παράλληλα.

Παρακάτω δίνεται πίνακας στον οποίον φαίνονται τα Latency, Throughput και που παρουσιάζει το σύστημα με την χρήση πολλαπλών παράλληλων Single Filter Calculators, σε καθέναν από τους οποίους θα ανατεθεί ένα τμήμα των ενεργοποιήσεων προς επεξεργασία από ένα φίλτρο.

Παράλληλα SFC	Throughput (activations/s)	Latency (s)
1 SFC	8,33	0,12
2 SFC	16,66	0,06
3 SFC	25	0,04
4 SFC	33,33	0,03

Πίνακας 3: Σύγκριση Throughput, Latency για παράλληλα SFC

Το αναμενόμενο Throughput από την συμμετοχή περισσότερων Single Filter Calculators παρουσιάζεται στο παρακάτω διάγραμμα.



Διάγραμμα 7: Απόδοση σε σχέση με τον αριθμό των SFC

Ο παραλληλισμός με την χρήση πολλαπλών single filter calculators μπορεί να αφορά την ταυτόχρονη επεξεργασία των ενεργοποιήσεων από πολλά φίλτρα. Παρακάτω φαίνεται το Latency και το Throughput που προκύπτουν από την λειτουργία του ίδιου επεξεργαστή για την σάρωση του συνόλου των ενεργοποιήσεων από πολλαπλά φίλτρα. Στον ίδιο πίνακα δίνονται τα αντίστοιχα μεγέθη για την λειτουργία του συστήματος σε FPGA. Χρησιμοποιούνται ενεργοποιήσεις διαστάσεων 512 X 512 με φίλτρα 5 X 5, τεσσάρων καναλιών.

No of Filters	FPGA Latency (sec)	FPGA Throughput(activations/sec)	CPU Latency (sec)	CPU Throughput (activations/sec)
1	0.12	8.33	0.075	13.3
2	0.24	4.16	0.15	6.66
4	0.48	2.08	0.30	3.33
8	0.96	1.04	0.59	1.69

Πίνακας 4: Σύγκριση Latency, Throughput σε FPGA και CPU με βάση τον αριθμό των φίλτρων

Και εδώ παρατηρείται ότι οι τιμές Latency και Throughput που προκύπτουν μεταξύ FPGA και CPU είναι συγκρίσιμες. Με την χρήση πολλαπλών single filter calculator, όπου κάθε ένας σαρώνει Οι ενεργοποιήσεις με διαφορετικό φίλτρο, επιτυγχάνονται καλύτερες τιμές.

SFC	No of Filters	Latency (sec)	Throughput(activations/sec)
1	1	0.12	8.33
2	2	0.12	8.33
4	4	0.12	8.33
8	8	0.12	8.33

Πίνακας 5: Σύγκριση Latency, Throughput FPGA για παράλληλα SFC

Στην περίπτωση που ενεργοποιήσεις και φίλτρα έχουν περισσότερα από τέσσερα κανάλια, η λειτουργία της CPU και του συστήματος είναι πιο χρονοβόρα. Παρακάτω δίνεται πίνακας ο οποίος περιέχει Latency και Throughput της CPU για την επεξεργασία ενεργοποιήσεων από ένα φίλτρο με διαφορετικό αριθμό καναλιών κάθε φορά. Οι τιμές του πίνακα αφορούν ενεργοποιήσεις διαστάσεων 512 X 512 με φίλτρο 5 X 5.

No of Channels	FPGA Latency (sec)	FPGA Throughput(activations/sec)	CPU Latency (sec)	CPU Throughput (activations/sec)
4	0.12	8.33	0.075	13.3
8	0.24	4.16	0.205	4.87
16	0.48	2.08	0.46	2.17
32	0.96	1.04	0.99	1.01

Πίνακας 6: Σύγκριση Latency, Throughput με βάση τον αριθμό των καναλιών

Με την χρήση πολλών παράλληλων 4-channel frame calculators μπορούμε στον ίδιο χρόνο να υπολογίσουμε πολλές τετράδες καναλιών. Στον παρακάτω πίνακα φαίνονται τα αντίστοιχα μεγέθη βάσει της παραλληλοποίησης που θα εφαρμόσουμε στην επεξεργασία καναλιών, ενεργοποιήσεων και φίλτρου.

4-CFC	No of Channels	Latency (sec)	Throughput(activations/sec)
1	4	0.12	8.33
2	8	0.12	8.33
4	16	0.12	8.33
8	32	0.12	8.33

Πίνακας 7: Σύγκριση Latency, Throughput σε FPGA με παράλληλα 4-CFC

Οι παραπάνω τιμές των πινάκων είναι προσεγγιστικές, καθώς θεωρούμε ότι ο εσωτερικός παραλληλισμός μεταβάλλει γραμμικά τις τιμές του Latency. Στην πραγματικότητα μπορεί να υπάρχουν μικρές αποκλίσεις. Με την χρήση εξωτερικού παραλληλισμού, δηλαδή με τη χρήση πολλών διαφορετικών FPGA, η μείωση του Latency για τις παραπάνω περιπτώσεις είναι όντως γραμμική.

Στη συνέχεια συγκρίνονται οι υλοποιήσεις μας με άλλες σχεδιάσεις ΣΝΔ accelerators σε αναδιατασσόμενη λογική. Πλέον αντί να συγκρίνουμε τον όγκο των ενεργοποιήσεων που μπορεί να επεξεργαστεί το εκάστοε σ΄συτημα ανά δευτερόλεπτο, συγκρίνουμε το πλήθος των πολλαπλασιασμών που λαμβάνουν χώρα ανα δευτερόλεπτο. Στον παρακάτω πίνακα παρουσιάζονται οι επιδόσεις accelerators που βρίσκονται στην βιβλιογραφία.

	[32]	[33]	[34]	[35]	A9/A16
Frequency (MHz)	115	125	200	150	171
FPGA chip	Virtex5 LX330T	Virtex5 SX240T	Virtex5 SX240T	Virtex6 SX240T	XC7K70T / XC7K160T
Performance (GMul/s)	3.37	3.50	8.00	8.50	0.37

Πίνακας 8: Σύγκριση απόδοσης σς σχέση με άλλες υλοποιήσεις FPGA

Το πλήθος των πολλαπλασιασμών που πραγματοποιεί το σύστημά μας υπολογίζεται βάσει των δεδομένων που παρουσιάστηκαν στην ενότητας 5.3. Συγκεκριμένα στην περίπτωση που το σύστημά μας δέχεται ενεργοποιήσεις διαστάσεων 1024 X 104 X 4 με φίλτρο διαστάσεων 16 X 16 X 4 και το stride τίθεται ίσο με 1 οι πολλαπλασιασμοί που απαιτούνται για την παραγωγή του feature map προκύπτουν ως εξής:

- Για κάθε εφαρμογή του φίλτρου επάνω σε κάποια θέση των ενεργοποιήσεων απαιτούνται 16 X 16 X 4 πολλαπλασιασμοί.
- Σε κάθε γραμμή, με stride 1, το φίλτρο θα εφαρμοσθεί Wact – Fdim + 1 φορές, δηλαδή 1009 φορές.
- Η διαδικασία θα επαναληφθεί για Hact – Fdim + 1 φορές, δηλαδή 1009 φορές.

Συνολικά οι πολλαπλασιασμοί που απαιτούνται για την παραγωγή του συγκεκριμένου feature map είναι 1.042.514.944 ενώ το feature map παράγεται σε 2,83 δευτερόλεπτα. Συνεπώς:

$$Performance_{A9-A16}\left(\frac{GMul}{s}\right) = \left(\frac{1.042.514.944}{2,83}\right) = 368.379.838\frac{Mul}{s} = 0,37\frac{GMul}{s}$$

Τα συστήματα σύγκρισης που παρουσιάζονται στον πίνακα λειτουργούν με FPGA υψηλότερων επιδόσεων και κόστους, χρησιμοποιώντας πολλά στάδια παραλληλισμού. Με χρήση παραλληλισμού στο module SFC ή/και στο 4-CFC μπορούμε να πλησιάσουμε ή και να υπερβούμε την απόδοση των accelerators που παρουσιάζονται στον παραπάνω πίνακα. Συγκεκριμένα επιλέγεται μία διάταξη 4 SFC όπου κάθε SFC περιέχει τέσσερεις 4-CFC. Με την χρήση των 4 SFC μπορούμε στον ίδιο χρόνο να επεξεργαστούμε ενεργοποιήσεις διαστάσεων 1024 X 1024 με 4 διαφορετικά φίλτρα ταυτόχρονα ή 4 διαφορετικών ενεργοποιήσεων των διαστάσεων αυτών με ένα φίλτρο (καθώς οι ενεργοποιήσεις διαχωρίζονται σε 4 ισομεγέθη τμήματα, με αποτέλεσμα η παραγωγή του feature map να

ολοκληρώνεται 4 φορές γρηγορότερα). Με την χρήση τεσσάρων 4-CFC σε κάθε SFC δίνεται η δυνατότητα ταυτόχρονης επεξεργασίας 16 καναλιών των ενεργοποιήσεων. Με την διάταξη αυτή οι πολλαπλασιαμοί που εκτελούνται είναι:

- Για κάθε εφαρμογή του φίλτρου επάνω σε κάποια θέση των ενεργοποιήσεων απαιτούνται 16 X 16 X 16 πολλαπλασιασμοί.
- Σε κάθε γραμμή, με stride 1, το φίλτρο θα εφαρμοσθεί Wact – Fdim + 1 φορές, δηλαδή 1009 φορές. Η διαδικασία θα επαναληφθεί για Hact – Fdim + 1 φορές, δηλαδή 1009 φορές.
- Πλέον στον ίδιο χρόνο μπορούμε να επεξεργαστούμε 4 ενεργοποιήσεις από τις παραπάνω.

$$Performance *_{A9-A16} \left(\frac{GMul}{s}\right) = \left(\frac{16.680.239.104}{2{,}83}\right) = 5.894.077.422 \frac{Mul}{s} = 5{,}9 \frac{GMul}{s}$$

Η παραπάνω διάταξη μπορεί να τοποθετηθεί στην FPGA της οικογενείας Virtex XC7V2000T. Στον παρακάτω πίνακα φαίνονται οι πόροι υλικού που χρησιμοποιούνται για την τοποθέτηση της διάταξης αυτής στην συγκεκριμένη FPGA με ακρίβεια τιμών ενεργοποιήσεων φίλτρου 16 bit. Με ακρίβεια 9 bit μπορούν να χρησιμοποιηθούν περισσότερα παράλληλα SFC στην ίδια FPGA οδηγώντας σε ακόμη καλύτερες αποδόσεις.

	A16*
FPGA	XC7V2000T
Slices	305.400
LUTs (Δεσμευμένα / Σύνολο)	1.187.800 / 1.221.600 **(97%)**
Registers (Δεσμευμένα / Σύνολο)	791.765 / 2.443.200 **(33%)**
36 KB Block RAM – FIFO (Δεσμευμένα / Σύνολο)	430 / 1.292 **(33%)**

Πίνακας 9: Απαιτούμενοι πόροι υλικού για διάταξη 4 SFC όπου κάθε SFC περιέχει 4 4-CFC

5.5. Πλεονέκτηματα Bit Pragmatic

Στις μη bit pragmatic προσεγγίσεις για την πράξη του πολλαπλασιασμού σε hardware μπορεί να παραχθεί αποτέλεσμα μετά το πέρας σταθερού αριθμού κύκλων ή ακόμη και εντός κύκλου (π.χ. με την χρήση της βιβλιοθήκης numeric_std), ανεξαρτήτως πλήθους και τιμών bit πολλαπλασιαστή / πολλαπλασιαστέου.

Στη σύγκριση του bit pragmatic με πολλαπλασιαστές που παράγουν γινόμενο μετά από σταθερό αριθμό κύκλων με δεδομένο ότι το 8% των bits των ενεργοποιήσεων ενός ΣΝΔ είναι άσσοι, η πιθανότητα να υπάρχει κέρδος χρόνου, στη συνολική λειτουργία σε μεγάλο αριθμό πολλαπλασιασμών, είναι μεγάλη.

Συγκρίνοντας το bit pragmatic με αρχιτεκτονικές που παράγουν γινόμενα εντός κύκλου, ωφελούμαστε σε σχέση με τους πόρους υλικού που χρησιμοποιούνται καθώς οι αρχιτεκτονικές αυτές χρησιμοποιούν πολλαπλά επίπεδα λογικής που παράγουν αποτελέσματα εντός κύκλου. Επιπλέον καθυστερήσεις που εισάγονται από την λογική πολλών επιπέδων του πολλαπλασιαστή μπορεί να οδηγήσουν στην μείωση της συχνότητας λειτουργίας του ρολογιού οδηγώντας τελικά στην μείωση της απόδοσης του συστήματος σε πραγματικό χρόνο, καθιστώντας την χρήση τους απαγορευτική.

Το Bit Pragmatic αποτελεί ικανοποιητική λύση μεταξύ χρόνου και πόρων που απαιτούνται για την παραγωγή γινομένου. Ωστόσο ως μειονέκτημα μπορεί να θεωρηθεί η εξαρτώμενη από τα δεδομένα συμπεριφορά της τεχνικής αυτής οδηγώντας σε μεταβλητούς χρόνους εκτέλεσης του συστήματος, ανάλογα με την είσοδο που δέχεται, το οποίο σε πολλές περιπτώσεις μπορεί να μην είναι επιθυμητό.

6. Συμπεράσματα Μελλοντικές Επεκτάσεις

Τα τελευταία χρόνια τα συνελικτικά νευρωνικά δίκτυα έχουν παρουσιάσει μεγάλη ανάπτυξη κυρίως λόγω της ικανότητάς τους στην αναγνώριση και κατηγοριοποίηση περίπλοκων εικόνων. Σκοπός της συγκεκριμένης εργασίας ήταν η επιτάχυνση της λειτουργίας ενός ΣΝΔ που επιτυγχάνεται με την επιτάχυνση των επιπέδων συνέλιξης και υποδειγματοληψίας χρησιμοποιώντας αναδιατασσόμενη λογική (FPGA). Τα επίπεδα του ΣΝΔ τα οποία αναπτύχθηκαν απευθύνονται σε οποιοδήποτε ΣΝΔ που πληρεί τους περιορισμούς, με την παροχή στο σύστημα των κατάλληλων παραμέτρων καθώς και των δεδομένων που περιέχουν ενεργοποιήσεις και φίλτρα. Για την πραγματοποίηση πολλαπλασιασμών χρησιμοποιείται η τεχνική Bit Pragmatic η οποία αποτελεί μια ισορροπημένη λύση μεταξύ πόρων που απαιτούνται και ταχύτητας για την παραγωγή γινομένων. Αναλύθηκε η δομή της υλοποίησης, μετρήθηκε η απόδοση και προτάθηκαν τρόποι για την περαιτέρω αύξηση του παραλληλισμού του συστήματος προκειμένου να επιτύχουμε καλύτερες αποδόσεις.

Μελλοντικά για την παρούσα υλοποίηση πρέπει να επιτευχθεί αύξηση της απόδοσης καθώς και μείωση των αιτημάτων προς την εξωτερική RAM. Η βελτίωση της απόδοσης μπορεί να επιτευχθεί με την μεταφορά της υλοποίησης σε FPGA που περιέχουν περισσότερους πόρους υλικού ώστε να μπορούν να χρησιμοποιηθούν πολλαπλές μονάδες Single Filter Calculators ή/και 4 –Channel Frame Calculators. Η επιπλέον μείωση των αιτημάτων ανάγνωσης από την εξωτερική RAM μπορεί να επιτευχθεί εκμεταλλευόμενοι την επικάλυψη δεδομένων που υπάρχει μεταξύ δύο διαδοχικών εφαρμογών του φίλτρου επάνω στις ενεργοποιήσεις. Ενώ στην παρούσα υλοποίηση εκμεταλλευόμαστε την επικάλυψη που παρουσιάζεται μεταξύ διαδοχικών εφαρμογών του φίλτρου επάνω στις ενεργοποιήσεις όταν μετακινούμαστε οριζόντια, δεν εκμεταλλευόμαστε την επικάλυψη που υπάρχει όταν το φίλτρο μετακινείται κάθετα.

Η λειτουργία μερικών λειτουργιών του ΣΝΔ είναι πιο αποδοτική όταν χρησιμοποιείται επεξεργαστής ενώ άλλα επίπεδα (τα οποία έχουν μεγάλες απαιτήσεις σε υπολογισμούς) αποδίδουν καλύτερα σε εξειδικευμένο hardware.

Χρειάζεται κατάλληλη μελέτη ώστε να δημιουργηθεί ένα σύστημα που θα κάνει χρήση Software–Hardware Codesing, ώστε η απόδοση σε σχέση με τους πόρους που χρησιμοποιούνται να είναι η καλύτερη δυνατή.

Τέλος, όσον αφορά την μονάδα pooling θα πρέπει να δίνεται και η επιλογή του stochastic pooling [30], τεχνικής που έχει προταθεί τα τελευταία χρόνια και είναι αρκετά υποσχόμενη, ενώ μπορεί να δίνεται η επιλογή και για επιπλέον συναρτήσεις ενεργοποίησης εκτός της ReLU.

Αναφορές

[1] National Security Agency.

The National Security Agency: Missions, Authorities, Oversight and Partnerships. Tech. rep. 2013.

[2] Bernard Marr.

How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read. 2016.

[3] Toneva, Mariya & Wehbe, Leila. (2019). Interpreting and improving natural-language processing (in machines) with natural language-processing (in the brain).

[4] Lawrence, Steve & Giles, C & Tsoi, Ah & Back, Andrew. (1997). Face Recognition: A Convolutional Neural Network Approach. Neural Networks, IEEE Transactions on. 8. 98 - 113. 10.1109/72.554195.

[5] Liu, Keng-Cheng & Hsu, Chen-Chien & Chiang, Hsin-Han. (2019). Real-Time Facial Expression Recognition Based on CNN. 120-123. 10.1109/ICSSE.2019.8823409.

[6] Shen, Yelong & He, Xiaodong & Gao, Jianfeng & Deng, li & Mesnil, Grégoire. (2014). Learning semantic representations using convolutional neural networks for web search. 373-374. 10.1145/2567948.2577348.

[7] Collobert, Ronan & Weston, Jason. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. Proceedings of the 25th International Conference on Machine Learning. 160-167. 10.1145/1390156.1390177.

[8] Simonyan, Karen & Zisserman, Andrew. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv 1409.1556.

[9] Potluri, Sasanka & Fasih, Alireza & Vutukur, Laxminand & Al Machot, Fadi & Kyamakya, Kyandoghere. (2014). CNN Based High Performance Computing for Real Time Image Processing on GPU. 10.1007/978-3-642-24806-1_20.

[10] Artem Khurshudov.
Suddenly, a leopard print sofa appears. 2015.

[11] Girshick, Ross. (2015). Fast r-CNN. 10.1109/ICCV.2015.169.

[12] Shelhamer, Evan & Long, Jonathon & Darrell, Trevor. (2016). Fully Convolutional Networks for Semantic Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence. 39. 1-1. 10.1109/TPAMI.2016.2572683.

[13] Matsugu, Masakazu & Mori, Katsuhiko & Mitari, Yusuke & Kaneda, Yuji. (2003). Subject Independent Facial Expression Recognition with Robust Face Detection Using a Convolutional Neural Network. Neural networks : the official journal of the International Neural Network Society. 16. 555-9. 10.1016/S0893-6080(03)00115-1.

[14] Zhang, Ying & Pezeshki, Mohammad & Brakel, Philemon & Zhang, Saizheng & Bengio, Y. & Courville, Aaron. (2017). Towards End-to-End Speech Recognition with Deep Convolutional Neural Networks.

[15] Howard, Andrew & Zhu, Menglong & Chen, Bo & Kalenichenko, Dmitry & Wang, Weijun & Weyand, Tobias & Andreetto, Marco & Adam, Hartwig. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.

[16] John R.; Bennett Koza.
How can computers learn to solve problems without being explicitly programmed. 1996.

[17] Kohavi, Ron & Provost, Foster. (1998). Glossary of Terms. Machine Learning. 2. 271-274. 10.1023/A:1017181826899.

[18] Lee, Honglak & Grosse, Roger & Ranganath, Rajesh & Ng, Andrew. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. Proceedings of the 26th International Conference On Machine Learning, ICML 2009. 77. 10.1145/1553374.1553453.

[19] Mannila, Heikki. (1996). Data mining: machine learning, statistics, and databases. 2 - 9. 10.1109/SSDM.1996.505910.

[20] Pinto, Agnese & Scioscia, Floriano & Loseto, Giuseppe & Ruta, Michele & Bove, Eliana & Di Sciascio, Eugenio. (2015). A semantic-based approach for Machine Learning data analysis. Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing, IEEE ICSC 2015. 324-327. 10.1109/ICOSC.2015.7050828.

[21] Soudry, Daniel & Hubara, Itay & Meir, Ron. (2014). Expectation Backpropagation: Parameter-Free Training of Multilayer Neural Networks with Continuous or Discrete Weights. Advances in Neural Information Processing Systems. 2.

[22] Lecun, Yann & Bottou, Leon & Bengio, Y. & Haffner, Patrick. (1998). Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE. 86. 2278 - 2324. 10.1109/5.726791.

[23] Krizhevsky, Alex & Sutskever, Ilya & Hinton, Geoffrey. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Neural Information Processing Systems. 25. 10.1145/3065386.

[24] Szegedy, Christian & Liu, Wei & Jia, Yangqing & Sermanet, Pierre & Reed, Scott & Anguelov, Dragomir & Erhan, Dumitru & Vanhoucke, Vincent & Rabinovich, Andrew. (2015). Going deeper with convolutions. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 1-9. 10.1109/CVPR.2015.7298594.

[25] He, Kaiming & Zhang, Xiangyu & Ren, Shaoqing & Sun, Jian. (2016). Deep Residual Learning for Image Recognition. 770-778. 10.1109/CVPR.2016.90.

[26] Haykin, Simon. (1998). Neural Networks: A Comprehensive Foundation (2nd Edition) Neural Networks: A Comprehensive Foundation.

[27] White, B. & Rosenblatt, Frank. (1963). Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. The American Journal of Psychology. 76. 705. 10.2307/1419730.

[28] Xilinx. 7 Series FPGAs Data Sheet: Overview

[29] Moshovos, Andreas & Albericio, Jorge & Judd, Patrick & Lascorz, Alberto & Sharifymoghaddam, Sayeh & Poulos, Zissis & Hetherington, Tayler & Aamodt, Tor & Jerger, Natalie. (2018). Exploiting Typical Values to Accelerate Deep Learning. Computer. 51. 18-30. 10.1109/MC.2018.2381114.

[30] Glorot, Xavier & Bordes, Antoine & Bengio, Y. (2011). Deep Sparse Rectifier Neural Networks. Proceedings of the 14th International Conference on Artificial Intelligence and Statisitics (AISTATS) 2011. 15. 315-323.

[31] Moshovos, Andreas. (2018). Value-Based Deep Learning Hardware Acceleration. 1-1. 10.1109/NOCARC.2018.8541207.

[32] Sankaradass, Murugan & Jakkula, Venkata & Cadambi, Hari & Chakradhar, Srimat & Durdanovic, Igor & Cosatto, Eric & Graf, H.P.. (2009). A Massively Parallel Coprocessor for Convolutional Neural Networks. 53-60. 10.1109/ASAP.2009.25.

[33] Cadambi, Hari & Majumdar, Abhinandan & Becchi, Michela & Chakradhar, Srimat & Graf, H.P. (2010). A Programmable Parallel Accelerator for Learning and Classification. 273-284. 10.1145/1854273.1854309.

[34] Chakradhar, Srimat & Sankaradass, Murugan & Jakkula, Venkata & Cadambi, Hari. (2010). A dynamically configurable coprocessor for convolutional neural networks. ACM Sigarch Computer Architecture News. 247-257. 10.1145/1816038.1815993.

[35] Peemen, Maurice & Setio, Arnaud & Mesman, B. & Corporaal, Henk. (2013). Memory-centric accelerator design for Convolutional Neural Networks. 2013 IEEE 31st International Conference on Computer Design, ICCD 2013. 13-19. 10.1109/ICCD.2013.6657019.

[36] Συντάκτες της Βικιπαίδειας, "Τεχνητή νοημοσύνη," Βικιπαίδεια, Η Ελεύθερη Εγκυκλοπαίδεια

[37] Συντάκτες της Βικιπαίδειας, "Νευρωνικό δίκτυο," Βικιπαίδεια, Η Ελεύθερη Εγκυκλοπαίδεια

[38] Song, Jia & Gao, Shaohua & Zhu, Yunqiang & Ma, Chenyan. (2019). A survey of remote sensing image classification based on CNNs. Big Earth Data. 1-23. 10.1080/20964471.2019.1657720.

[39] Συντάκτες της Βικιπαίδειας, "Μηχανική μάθηση," Βικιπαίδεια, Η Ελεύθερη Εγκυκλοπαίδεια