Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Methodology of CNN Projection and Acceleration on FPGA with Tensorflow Framework

Διπλωματικη Εργασια

του

ΜΟΥΣΕΛΙΝΟΥ ΣΠΥΡΙΔΩΝΟΣ

Επιβλέπων: Κιαμάλ Πεκμεστζή
Καθηγητής Ε.Μ.Π.

Εργαστηριο Μικροϋπολογιστων και Ψηφιακων Συστηματων
Αθήνα, Φεβρουάριος 2019

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Μικρουπολογιστών και Ψηφιακών Συστημάτων

Methodology of CNN Projection and Acceleration on FPGA with Tensorflow Framework

Διπλωματικη Εργασια

του

ΜΟΥΣΕΛΙΝΟΥ ΣΠΥΡΙΔΩΝΟΣ

Επιβλέπων: Κιαμάλ Πεκμεστζή
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 21η Φεβρουαρίου 2019.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Κιαμάλ Πεκμεστζή	Δημήτριος Σούντηρης	Γεώργιος Γκούμας
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Επ.Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2019

(Υπογραφή)

.....

Μουσελινος Σπυριδων

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Μικρουπολογιστών και Ψηφιακών Συστημάτων

Περίληψη

Τα τελευταία χρόνια όλο και περισσότερο βάρος δίνεται στην έρευνα και βελτιστοποίηση των Αλγορίθμων Μηχανικής Μάθησης και "Βαθειάς Γνώσης" ιδιαίτερα με την μορφή των Νευρωνικών Δικτύων. Τα Νευρωνικά Δίκτυα αποτελούν υπολογιστικούς γράφους, εμπνευσμένους απο την λειτουργία των βιολογικών νευρώνων και έχουν ξεπεράσει σε απόδοση και ακρίβεια άλλες τεχνικες μηχανικής μάθησης καθώς και παραδοσιακές τεχνικές στατιστικής πρόβλεψης.

Μια τέτοια κατηγορία Νευρωνικών Δικτύων αποτελούν τα Συνελικτικά Νευρωνικά Δίκτυα, με ιδιαίτερα επιτυχημένη εφαρμογή σε προβλήματα όρασης υπολογιστών όπως η αναγνώριση και κατηγοριοποίηση αντικειμένων.

Τα προβλήματα όρασης ωστόσο συναντούνται κυρίως σε εφαρμογές και απαιτήσεις πραγματικου χρόνου, με την ταχύτητα και την κλιμάκωση των παραδοσιακών δομών υποστήριξης των υπολογισμών (π.χ ένας τυπικός επεξεργαστής) να είναι ανεπαρκώς μικρές.

Στόχος της εργασίας αυτής αποτελεί η παρουσίαση και εφαρμογή τεχνικών τόσο από θεωρητική σκοπία, όσο και σε επίπεδο υλικού, για την εξαγωγή μιας μεθοδολογίας απεικόνισης συνελικτικων νευρωνικών δικτύων σε πλατφόρμες FPGA. Συγκεκριμένα επιτυγχάνεται η αποδοτική σχεδίαση ενός επιταχυντή σε επίπεδο πυλών, καθώς και η παράθεση των αποτελεσμάτων με διάφορες παραλλαγές για το πρότυπο Συνελικτικό Νευρωνικό Δίκτυο αναγνώρισης χειρόγραφων ψηφίων MNIST.

Αυτό που μπορεί να εξαχθεί ως γενικό συμπέρασμα της εργασίας είναι ότι η εξαγόμενη μεθοδολογία σχεδίασης επιλύει αποδοτικά προβλήματα πραγματικού χρόνου, θυσιάζοντας ελάχιστα την ακρίβεια για χάρη της ταχύτητας και έγκαιρης απόκρισης.

Λέξεις Κλειδιά

Μηχανική Μάθηση, Νευρωνικά Δίκτυα, Συνελικτικά Νευρωνικά Δίκτυα,
Υλικό, Επιταχυντές, Συστολικότητα, Προβλήματα Πραγματικού Χρόνου FPGA.

Abstract

During the last decade a considerable amount of research effort has been made towards the development and optimization of Machine Learning Algorithms such as the Deep Learning Model, especially in the form of Artificial Neural Networks.
Artificial Neural Networks are computing graphs, inspired by the functions of biological brain neurons and have managed to surpass in terms of performance and accuracy other Machine Learning approaches as well as their traditional counterparts of statistic-mathematical analysis.

Between the many network types of Artificial Neural Networks this paper focuses on the Convolutional Neural Networks, that excel in computer vision applications, such as object recognition and classification.

However, such applications belong to the Real-Time constraint spectrum, meaning that speed and scalability are of utmost importance. Keeping that in mind, standard computational units (such as a typical CPU) tend to become unfit for these tasks.

The goal of this thesis is to suggest and apply techniques from a theoretical as well as a purely hardware-oriented standpoint,in order to present a solid methodology of projecting Convolutional Neural Networks in a modular and pipelined fashion on FPGA platforms. Finally, an efficient Gate-Level design of a convolutional Accelerator is achieved, followed by the presentation of metrics and results for different design strategies. The results as well as the development of the Accelerator are based on the standard Convolutional Neural Net Model of handwritten digit recognition MNIST.

As a general conclusion of this thesis we can establish that the proposed design methodology performs efficiently on the real-time constraints of the task, sacrificing slightly the accuracy for the sake of speed and immediate response.

Keywords

Machine Learning, Neural Networks, Convolutional Neural Networks,
Hardware, Accelerators, Pipelining, Real-Time, FPGA.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Κιαμάλ Πεκμεστζή για την εξαιρετική συνεργασία και καθοδήγηση κατα την εκπόνηση της εργασιας.

Επίσης ευχαριστώ ιδιαίτερα τον μεταπτυχιακό φοιτητή και φίλο κ. Βασίλη Λέων για βοήθειά του σε αυτο το εγχείρημα.

Τέλος θα ήθελα να ευχαριστήσω την οικογένειά μου, που με στήριξε στον αγώνα μου όλα αυτα τα χρόνια.

Περιεχόμενα

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή στα Νευρωνικά Δίκτυα

1.1 Αλγόριθμοι Μηχανικής Μάθησης

Η Μηχανική μάθηση είναι υποπεδίο της επιστήμης των υπολογιστών που αναπτύχθηκε από τη μελέτη της αναγνώρισης προτύπων και της υπολογιστικής θεωρίας μάθησης στην τεχνητή νοημοσύνη. Το 1959, ο Άρθουρ Σάμουελ ορίζει τη μηχανική μάθηση ως "Πεδίο μελέτης που δίνει στους υπολογιστές την ικανότητα να μάθαίνουν, χωρίς να έχουν ρητά προγραμματιστεί". Η μηχανική μάθηση διερευνά τη μελέτη και την κατασκευή αλγορίθμων που μπορούν να μάθαίνουν από τα δεδομένα και να κάνουν προβλέψεις σχετικά με αυτά. Τέτοιοι αλγόριθμοι λειτουργούν κατασκευάζοντας μοντέλα από πειραματικά δεδομένα, προκειμένου να κάνουν προβλέψεις βασιζόμενες στα δεδομένα ή να εξάγουν αποφάσεις που εκφράζονται ως το αποτέλεσμα

1.1.1 Κατηγορίες Μηχανικής Μάθησης

Οι αλγόριθμοι μηχανικής μάθησης συνήθως ταξινομούνται σε τρεις μεγάλες κατηγορίες, ανάλογα με τη φύση του εκπαιδευτικού «σήματος» ή την «ανατροφοδότηση» που είναι διαθέσιμες σε ένα σύστημα εκμάθησης. Οι τρεις αυτες κατηγορίες είναι:

1. ***Επιτηρούμενη μάθηση*** **(Supervised learning)**: Το υπολογιστικό πρόγραμμα δέχεται τις παραδειγματικές εισόδους καθώς και τα επιθυμητά αποτελέσματα από έναν «δάσκαλο», και ο στόχος είναι να μάθει έναν γενικό κανόνα - μοντέλο προκειμένου να αντιστοιχίσει τις εισόδους με τα αποτελέσματα.

2. ***Μη Επιτηρούμενη μάθηση*** **(Unsupervised learning)**: Χωρίς να παρέχεται κάποια εμπειρία στον αλγόριθμο μάθησης, αυτός πρέπει να βρεί την δομή των δεδομένων εισόδου. Η Μη Επιτηρούμενη μάθηση μπορεί να είναι αυτοσκοπός (ανακαλύπτοντας κρυμμένα μοτίβα σε δεδομένα) ή μέσο για ένα τέλος (χαρακτηριστικό της μάθησης).

3. ***Ενισχυτική μάθηση*** **(Reinforcement learning)**: Έχει στόχο την ανάπτυξη ενός συστήματος (πράκτορα) που βελτιώνει την απόδοσή του καθώς αλληλεπιδρά με το περιβάλλον.Στην ενισχυτική μάθηση, η πληροφορία σχετικά με την τρέχουσα κατάσταση

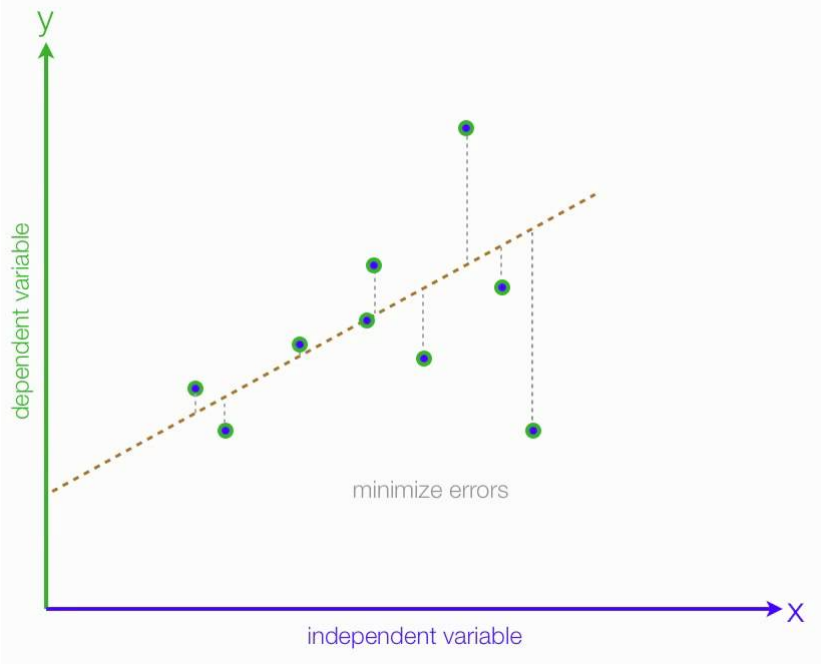
του περιβάλλοντος περιλαμβάνει επιπλέον και ενα σήμα επιβράβευσης, που μοντελοποιεί πόσο καλή ειναι η δράση του πράκτορα, ο οποίος θα επιτύχει το στόχο του μεγιστοποιώντας την επιβράβευσή του.

1.2 Στόχοι Επιτηρούμενης Μάθησης

Από τις προηγούμενες μεθόδους μηχανικής μάθησης ας επικεντρώσουμε το ενδιαφέρον μας στην επιτηρούμενη μηχανική μάθηση, όπου ανάλογα με τη φύση του προβλήματος, μπορεί να αναζητήσει διαφορετικούς στόχους-λύσεις. Έτσι ορίζουμε ως υποκατηγορίες της επιτηρούμενης μηχανικής μάθησης τις εξής μεθόδους:

1. ***Παλινδρόμιση* (Regression)**: Πρόκειται για την πρόβλεψη αποτελεσμάτων που αποτελούν συνεχείς μεταβλητές με βάση διάφορα αλλα, συνεχή-και-μη χαρακτηριστικά. Στην Παλινδρόμιση, υποθέτουμε ότι δίνεται ένα πλήθος από μεταβλητές πρόβλεψης και μία συνεχής μεταβλητή απόκρισης. Στόχος είναι η εύρεση μιας σχέσης μεταξύ των μεταβλητών πρόβλεψης που επιτρέπει το συμπερασμό του αποτελέσματος (π.χ η πρόβλεψη της αγοραστικής αξίας ενός ακινήτου με βάση τις διαστάσεις του, τους ορόφους του, το αν βρίσκεται εντός η εκτός πόλης κ.α).

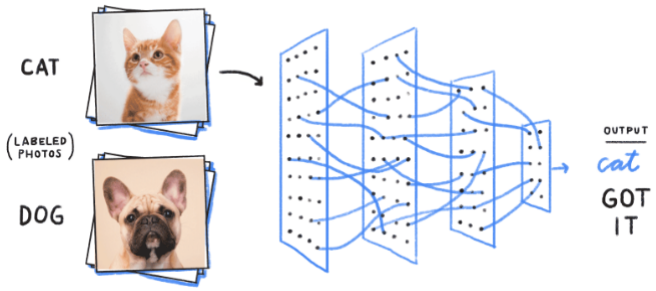
   Στο σχήμα 1.1 παρουσιάζεται η ιδέα της γραμμικής παλινδρόμισης. Δεδομένων μιας μεταβλητής πρόβλεψης X και μιας μεταβλητής απόκρισης Υ, ο αλγόριθμος προσπαθεί να "ταιριάξει" τα δεδομένα, ελαχιστοποιώντας την απόσταση μεταξύ των δειγμάτων-προβλέψεων και της γραμμής-πραγματικής τιμής. Η ίδια αυτή γραμμή θα χρησιμοποιηθεί για την πρόβλεψη μελλοντικών δειγμάτων.



Σχήμα 1.1

2. ***Ταξινόμιση* (Classification)**: Στην ταξινόμιση στόχος ειναι η πρόβλεψη της κλάσης ενός νέου αντικειμένου, με βάση παλαιότερες παρατηρήσεις. Η ταξινόμιση ενός αντικειμένου σε μια κατηγορία γίνεται με την ανάθεση μιας ετικέτας σε αυτό, με τις ετικέτες να είναι μία από τις διάφορες μεταβλητές εισόδου, λαμβάνοντας προφανώς διακριτές τιμές. Εδώ, το μοντέλο μάθησης προσπαθεί να αναθέσει οποιαδήποτε ετικέτα που εμφανίστηκε στο σύνολο των δεδομένων κατά την διάρκεια της μάθησης σε ένα νέο μη ταξηνομημένο σύνολο αντικειμένων (π.χ η ταξινόμηση των φωτογραφιών ενός φακέλου με γάτες και σκύλους σε δύο υποφάκελους με τίτλο "γάτα" και "σκύλος", καθένας απο τους οποίους θα περιέχει ιδανικά όλες τις φωτογραφίες που ανήκουν την κλάση του).

   Στο σχήμα 1.2 παρουσιάζεται η ιδέα της ταξινόμισης φωτογραφιών σε μια κλάση "σκύλος" και μια κλάση "γάτα". Ο αλγόριθμος προσπαθεί να αναθέσει την κατάλληλη ετικέτα που εμφανίστηκε στην διάρκεια της μάθησης στο νέο σύνολο αντικειμένων.



Σχήμα 1.2

1.3 Νευρωνικά Δίκτυα

1.3.1 Περιγραφή Δικτύων

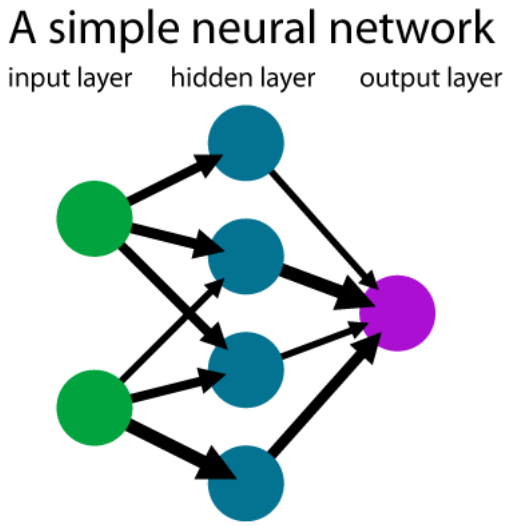
Ως ένα υπολογιστικό μοντέλο Μηχανικής Μάθησης, το νευρωνικό δίκτυο είναι ένα δίκτυο από απλούς υπολογιστικούς κόμβους (νευρώνες, νευρώνια), διασυνδεδεμένους μεταξύ τους. Είναι εμπνευσμένο από το Κεντρικό Νευρικό Σύστημα , το οποίο προσπαθεί να προσομοιώσει.

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), επιτελεί έναν υπολογισμό με βάση αυτές τις εισόδους και παράγει μία έξοδο. Η εν λόγω έξοδος είτε κατευθύνεται στο περιβάλλον, είτε τροφοδοτείται ως είσοδος σε άλλους νευρώνες του δικτύου. Υπάρχουν τρεις τύποι νευρώνων:

1. ***Νευρώνες Εισόδου* (Input Neurons)**: 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.

2. ***Νευρώνες Εξόδου* (Output Neurons)**: Οι νευρώνες εξόδου διοχετεύουν στο περιβάλλον τις τελικές αριθμητικές εξόδους του δικτύου.

3. ***Υπολογιστικοί-Κρυφοί Νευρώνες* (Hidden Neurons)**: 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.



Σχήμα 1.3

Στην εικόνα 1.3 έχουμε το παράδειγμα ενός απλού Νευρωνικού Δικτύου με 2 Νευρώνες Εισόδου, 4 Κρυφούς Νευρώνες και 1 Νευρώνα Εξόδου. Να σημειωθεί πως κάθε ομάδα νευρώνων ιδίου τυπου που βρίσκονται στην ίδια απόσταση από την αρχή του υπολογιστικού γράφου, και απεικονίζονται κατακόρυφα ονομάζονται επίπεδο(layer) του νευρωνικού δικτύου. Συνεπώς στο ανώτερο δίκτυο έχουμε 3 επίπεδα: 1 επίπεδο εισόδου, 1 κρυφό επίπεδο και 1 επίπεδο εξόδου.

1.3.2 Λειτουργία Νευρώνα

Ένας τεχνητός νευρώνας είναι μια υπολογιστική μονάδα που θα κάνει έναν υπολογισμό βασισμένο σε άλλες μονάδες στις οποίες είναι συνδεδεμένη. Ας εξετάσουμε την περίπτωση ενός μοναδικού κρυφού νευρώνα. Ο νευρώνας θα διαβάσει τις πληροφορίες από την είσοδο (η οποία έχει τη μορφή ενός διανύσματος), θα εκτελέσει έναν συγκεκριμένο υπολογισμό και θα υπολογίσει την αξία του. Με βάση αυτή την τιμή, ένας νευρώνας θα αποφασίσει εάν υπάρχουν ή όχι κάποια ενδιαφέροντα, προς την επίλυση του προβλήματος χαρακτηριστικά στην είσοδο.

Αυτό το βήμα υπολογισμού χωρίζεται σε δύο μέρη:

1. ***Προενεργοποίηση Νευρώνων* (Pre-Activation)**: Εδώ έχουμε τον εξής υπολογισμό, σε αλγεβρική μορφή:

$$a(x) = b + \sum_i w_i \cdot x_i$$

Αλλά και με τη μορφή πινάκων:

$$a(x) = b + \mathbf{w^T} \cdot \mathbf{x}$$

Όπου:

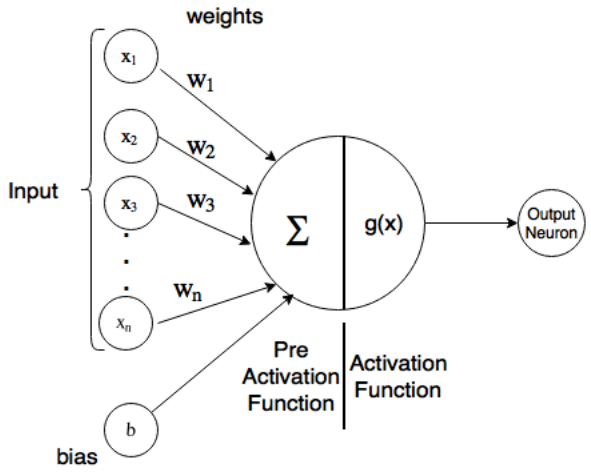
- ***a**: Μερικό αποτέλεσμα προ-ενεργοποίησης.*
- ***x**: Διάνυσμα/Τιμή Εισόδου.*
- ***w**: Διάνυσμα/Τιμή των βαρών σύνδεσης. Αντιπροσωπεύει τη δύναμη μεταξύ των συνδέσεων.*
- ***b**: Προκατάληψη. Εάν δεν έχουμε εισόδους, το β θα είναι η συνεισφορά μας για τον νευρώνα. Η προκατάληψη δεν είναι παρά μια τιμή προδιάθεσης του νευρώνα για την επιλογή συγκεκριμένου αποτελέσματος.*

Όλες αυτές οι διαδικασίες συνοψίζονται στο σχήμα 1.4 όπου η προενεργοποίηση νευρώνων εντοπίζεται στο πρώτο μισό, ακολουθούμενη απο την συνάρτηση ενεργοποίησης.

2. ***Ενεργοποίηση Νευρώνων* (Activation)**: Στο τελικό κομμάτι λειτουργίας του νευρώνα, χρησιμοποιούμε τιμές από τη λειτουργία προ-ενεργοποίησης για να υπολογίσουμε τη λειτουργία ενεργοποίησης, η αλλιώς το τελικό αποτέλεσμα που θα προωθηθεί στα επόμενα στάδια υπολογισμού.

Εδώ έχουμε πολύ απλά την εφαρμογή μιας συνάρτησης, της λεγόμενης συνάρτησης ενεργοποίησης στο προηγούμενο αποτέλεσμα a(x), έτσι ώστε: h(x) = g(a(x))
Με:

- ***h(x)**: Το αποτέλεσμα του νευρώνα.*
- ***g(x)**: Συνάρτηση ενεργοποίησης.*



Σχήμα 1.4

1.4 Κατηγορίες Υπολογιστικών Επιπέδων

Όπως προαναφέρθηκε, οι υπολογιστικοί-κρυφοί νευρώνες οργανώνονται σε συστάδες, τα επονομαζόμενα επίπεδα. Ένα νευρωνικό δίκτυο μπορεί να περιλαμβάνει διαφόρων ειδών επίπεδα, καθένα από τα υποία διαθέτει νευρώνες με διαφορετικό υπολογιστικό μοντέλο, αλλα και διαφορετική συνάρτηση ενεργοποίησης. Τα πιο γνωστά επίπεδα είναι:

- ***Πλήρως Συνδεδεμένο - Fully Connected Layer**: Το πιο απλό και βασικό επίπεδο που υλοποιεί την πράξη του πολλαπλασιασμού της εισόδου με μια σταθερά-βάρος και την πρόσθεση μιας τιμής προκατάληψης.*

- ***Συνελικτικό Μονοδιάστατο Επίπεδο - Convolutional 1D Layer**: Το επίπεδο αυτο χρησιμοποιείται κυρίως στην επεξεργασία ακολουθιακών δεδομένων, όπως για παράδειγμα στην επεξεργασία φυσικής γλώσσας για την εξαγωή τοπικών μοτίβων σε αλυσίδες εισόδων.*

- ***Συνελικτικό Δυσδιάστατο Επίπεδο - Convolutional 2D Layer**: Το επίπεδο αυτο χρησιμοποιείται κυρίως στην επεξεργασία εικόνων, πραγματοποιώντας την πράξη της ολίσθησης σταθερού πίνακα βαρών πάνω απο ένα πίνακα εικονοστοιχείων, με σκοπό την εξαγωγή τοπικών μοτίβων και χαρακτηριστικών για τον προσδιορισμό αποτελεσμάτων.*

- ***Δειγματοληπτικό Επίπεδο - Pooling Layer**: Το επίπεδο αυτο χρησιμοποιείται ύστερα από επίπεδα εξαγωγής χαρακτηριστικών, όπως τα συνελικτικά, προκειμένου να μειώσει τον όγκο των δεδομένων που θα μεταβούν στο επόμενο επίπεδο, πραγματοποιώντας μια εικασία για το ποιά είναι σημαντικά. Ανάλογα με την εικασία τα επίπεδα αυτά*

επιλέγουν ένα υποσύνολο δεδομένων, το οποίο προωθούν με η και χωρίς αλλαγές. Χαρακτηριστικό παράδειγμα είναι το επίπεδο Μέγιστης Εικασίας - Max Pooling, σύμφωνα με το οποίο από μια περιοχή θα προωθηθεί μόνο το μεγαλύτερο στοιχείο.

- ***Επίπεδο Αποκοπής - Dropout Layer**: Το επίπεδο αυτο χρησιμοποιείται για την βελτίωση της απόδοσης ενός νευρωνικού δικτύου σε βάρος της ταχύτητας εκμάθησης. Με βάση μια σταθερά πιθανότητας με τιμές από 0 έως 1, αποφασίζει τυχαία πόσες απο τις συνάψεις του προηγουμένου επιπέδου όντως θα φτάσουν στο επόμενο επίπεδο. Με την τεχνική αυτή μειώνεται ο κίνδυνος overfitting , δηλαδή ο κίνδυνος το δίκτυο να λειτουργεί ως μνήμη και όχι ως προβλέπτης με αποτέλεσμα να αποδίδει καλά σε δεδομένα που έχει ξαναδεί, αλλά άσχημα σε νέα δεδομένα. Αυτό συμβαίνει διότι θα δώσει μεγαλύτερη βαρύτητα στην εκμάθηση των συντελεστών προκατάληψης b, παρά στους συντελεστές βαρύτητας w.*

1.5 Κατηγορίες Συναρτήσεων Ενεργοποίησης

Εξίσου σημαντικές στα επίπεδα ενός νευρωνικού δικτύου είναι οι διαφορες συναρτήσεις ενεργοποίησης. Σημειώνεται ότι συχνά παρατηρούνται επίπεδα με το ίδιο πρώτο υπολογιστικό μέρος, αλλά διαφορετικές συναρτήσεις ενεργοποίησης. Αυτό συμβαίνει διότι η συνάρτηση ενεργοποίησης εκτός από το να κανονικοποιεί τα δεδομένα σε μια μικρότερη κλίμακα, συχνά αντιπροσωπεύει τη φυσική ερμηνεία του στόχου του νευρωνικού δικτύου.
Οι πιο συνήθεις συναρτήσεις είναι:

- ***Γραμμική ενεργοποίηση - Linear Activation***: Σε αυτή την περίπτωση, το g(x) παίρνει μια τιμή προενεργοποίησης και επιστρέφει την ίδια τιμή προενεργοποίησης χωρίς να την χειριστεί. Δεν εκτελεί καμία κατάτμηση εισόδου. Δεν έχει ανώ και κάτω όριο και δεν εισάγει καμία μη γραμμικότητα.

- ***Σιγμοειδής ενεργοποίηση - Sigmoid Activation***: Ίσως η πιο γνωστή συνάρτηση ενεργοποίησης, εδώ η g(x) υπολογίζει την τιμή ενεργοποίησης χρησιμοποιώντας τον παρακάτω τύπο:

$$g(x) = sigm(x) = \frac{1}{1 + exp(-x)}$$

  Η Σιγμοειδής ενεργοποίηση έχει πάντα θετικό αποτέλεσμα μεταξύ του 0 και του 1, αυξάνεται αυστηρά και ονομάζεται έτσι λόγω του χαρακτηριστικού s που δημιουργεί η καμπύλη της.

- ***Ενεργοποίηση Softmax***: Πρόκειται για μια συνάρτηση μετασχηματισμού των δεδομένων ενός διανύσματος έτσι ώστε να αντιπροσωπεύουν εκατοστιαίες πιθανοτήτες στο εύρος (0,1). Υλοποιεί την εξής συνάρτηση:

$$g(x_i) = \frac{exp(x_i)}{\sum_i exp(x_i)}$$

- ***Υπερβολική εφαπτομένη - TanH Activation***: Επίσης συχνά χρησιμοποιούμενη, η ενεργοποίηση υπερβολικής εφαπτομένης υλοποιεί την ομόνυμη συνάρτηση:

$$g(x) = tanh(x) = \frac{exp(2x) - 1}{exp(2x) + 1}$$

Η Υπερβολική εφαπτομένη έχει πάντα αποτέλεσμα μεταξύ του -1 και του 1 και αυξάνεται επίσης αυστηρά, ενώ περνά απο την αρχή τον αξόνων.

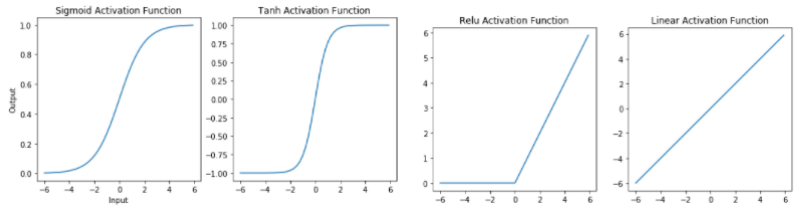
- ***Διορθωμένη Γραμμική Ενεργοποίηση - ReLu Activation***:

Η Διορθωμένη Γραμμική Ενεργοποίηση εμφανίστηκε προκειμένου να επιταχύνει την εκμάθηση των νευρωνικών δικτύων.

Υλοποιεί την τμηματικά γραμμική συνάρτηση:

$$g(x) = a \cdot max(x, 0)$$

Ο απαραίτητος για την εκμάθηση συντελεστής παραγώγου, εδώ υπολογίζεται πολύ γρήγορα (είναι είτε 0 είτε α ανάλογα με το πρόσημο του x). Να σημειωθεί πως ο συντελεστής a συνηθίζεται να λαμβάνει την τιμή 1, χωρίς αυτό να αποκλείει την χρήση άλλων τιμών.



Σχήμα 1.5

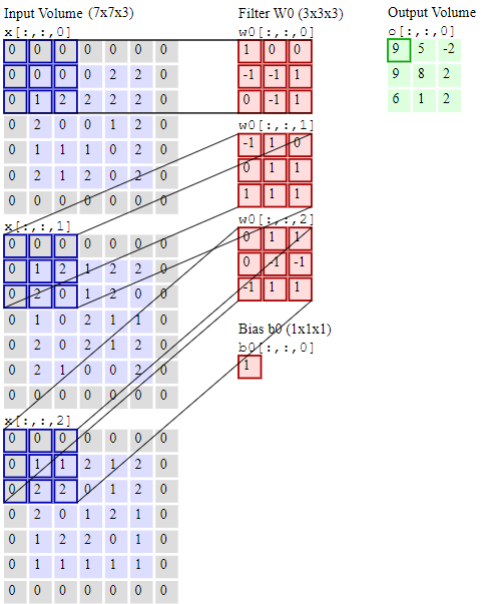
1.6 Το Συνελικτικό Νευρωνικό Δίκτυο (ΣΝΝ)

Τα Συνελικτικά Νευρωνικά Δίκτυα αποτελούν μια ειδική τοπολογία Νευρωνικών Δικτύων, εμπνευσμένη από τον οπτικό φλοιό των ζώων. Πατέρας των Συνελικτικών Νευρωνικών Δικτύων θεωρείται ο Yann LeCun που εφάρμοσε τα δίκτυα αυτά σε προβλήματα όρασης υπολογιστών το 1990. Το Συνελικτικό Νευρωνικό Μοντέλο αποσκοπεί στην αναγνώριση μοτίβων και χαρακτηριστικών, ειδικότερα σε δυσδιάστατες δομές (εικόνες, ήχοι), παρουσιάζοντας υψηλό βαθμό αναλλοίωτης συμπεριφοράς σε κάθε είδους παραμόρφωση και στρέβλωση των δεδομένων εισόδου.
Στη συνέχεια της εργασίας δεδομένη να θεωρηθεί η παραδοχή πως αναφερόμαστε σε δίκτυα επεξεργασίας εικόνων, σαν συνέπεια προηγείται ένα βήμα κατάτμησης της προς επεξεργασία εικόνας σε διανύσματα εικονοστοιχείων (pixels) τα οποία αποτελούν και την είσοδο του δικτύου. Οι προαναφερόμενες μέθοδοι και λειτουργίες των νευρώνων και των συναρτήσεων ενεργοποίησης συνεχίζουν να υφίστανται και εδώ, με το μοντέλο ενός δικτύου να αποτελείται από τα εξής στοιχεία:

- ***Επίπεδο Εισόδου***: Διαθέτει τα δεδομένα εισόδου, διανύσματα εικονοστοιχείων για ένα η πολλαπλά κανάλια χρωμάτων. Τα δεδομένα εισόδου μπορεί να προέρχονται απέυθείας απο την εικόνα ή να αποτελούν προϊόν προεπεξεργασίας της (π.χ σμύκρινση, μεγέθυνση, στρέβλωση, απεικόνιση σε κλίμακα γκρίζου). Να σημειωθεί πως το βάθος της εισόδου είναι ο αριθμός των χρωματικών καναλιών της εικόνας. Για παράδειγμα η ίδια εικόνα μεγέθους 32 pixel σε RGB και Greyscale θα αποθηκευτεί ως τρεις πίνακες με τιμές από 0-255 (διάυσμα [3,32,32]) και ως ένας πίνακας με τιμές από 0-255 (διάνυσμα [1,32,32]) αντίστοιχα.

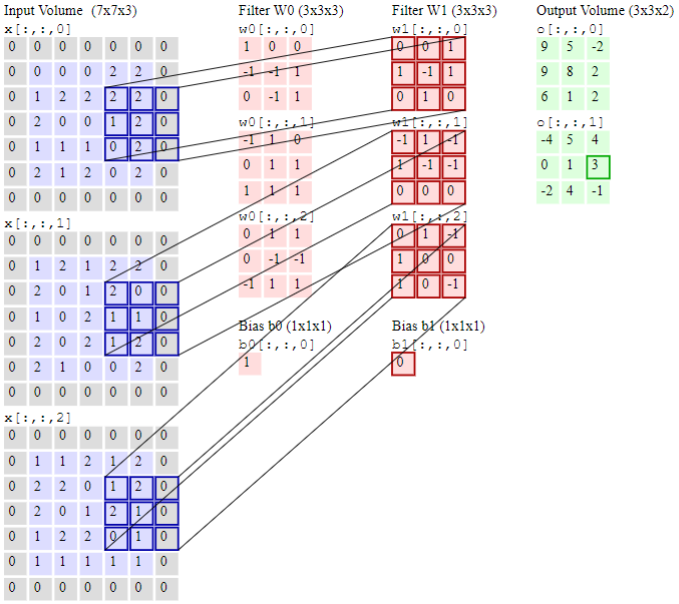
- ***Δισδυάστατο Συνελικτικό Επίπεδο***: Στο επίπεδο αυτό, που οφείλεται και το όνομα του δικτύου, υπολογίζεται σε ένα "παράθυρο" - φίλτρο η πράξη της συνέλιξης σε όλο το βάθος της εισόδου. Αυτό σημαίνει ότι ένα φίλτρο θα συνελιχθεί με το αντίστοιχο τμήμα της εικόνας σε κάθε χρώμα-επίπεδο και το αποτέλεσμα θα προκύψει σε ένα μοναδικό επίπεδο ως το άθροισμα αυτών συν μιας σταθεράς προκατάληψης.

Ένα παράδειγμα αποτελεί το σχήμα 1.6 όπου τα 3 κανάλια χρώματος μιας εικόνας 7x7 στα αριστερά συνελίσονται με το φίλτρο W0 με βάθος 3, ένα για κάθε κανάλι χρώματος, προκειμένου να παράξουν το πρώτο στοιχείο της εξόδου O.



Σχήμα 1.6

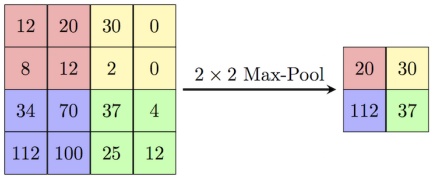
Στην πράξη χρησιμοποιούνται περισσότερα του ενός φίλτρα ανά επίπεδο, με τη μόνη διαφορά να είναι πως πλέον η έξοδος θα διαθέτει τόσες διαστάσεις όσες και τα φίλτρα που θα χρησιμοποιηθούν. Αυτό φαίνεται και στο σχήμα 1.7 όπου γίνεται χρήση δυο φίλτρων και έτσι η ίδια είσοδος μας δίνει δυσδιάστατη έξοδο.



Σχήμα 1.7

Συνοψίζοντας μετά απο κάθε συνελικτικό επίπεδο στην πραγματικότητα γίνεται ένας μετασχηματισμός μείωσης του μεγέθους τις εικόνας, αλλά στοίβαξης περισσότερων μικρότερων φιλτραρισμένων παραγώγων της ως διάνυσμα εισόδου στο επόμενο επίπεδο. Έτσι μια RGB είσοδος 32 επί 32 [3,32,32] ύστερα από φιλτράρισμα 2 φίλτρων με παράθυρο 3 επί 3 [2,3,3] θα μας δώσουν ένα διάνυσμα [2,3,3] που προκύπτει ως [αριθμός φίλτρων, μέγεθος παραγόμενης εικόνας ύστερα από ολίσθηση, μέγεθος παραγόμενης εικόνας ύστερα από ολίσθηση].

- ***Δειγματοληπτικό Επίπεδο Μεγίστου***: Όπως προαναφέρθηκε στο επίπεδο αυτό πραγματοποιείται μια σμύκρινση του μεγέθους της εισόδου με βάση κάποια τεχνική συμπίεσης. Στην περίπτωσή μας ορίζεται ως παράμετρος του επιπέδου ένα τετράγωνο παράθυρο ολίσθησης, από το οποίο μόνο το μέγιστο στοιχείο προωθείται σαν έξοδος. Το αποτέλεσμα είναι να μειώνεται το πλάτος και μήκος αλλά όχι το βάθος της εισόδου.



Σχήμα 1.7

Στο σχήμα 1.7 βλέπουμε το αποτέλεσμα της ολίσθησης ενός 2x2 παραθύρου πάνω σε δεδομένα εισόδου 4x4 και την εξαγωγή ενός πίνακα διαστάσεων 2x2 με τα μέγιστα στοιχεία.

- ***Πλήρως Συνδεδεμένο Επίπεδο***: Το κλασικό επίπεδο με κάθε νευρώνα αυτής της στρώσης να συνδέεται με όλα τα στοιχεία του όγκου της εισόδου της.

- ***Συνάρτηση Διορθωμένης Γραμμικής Ενεργοποίησης***: Στα Συνελικτικά Νευρωνικά Δίκτυα όλα τα προηγούμενα επίπεδα πλην του τελευταίου Πλήρως Συνδεδεμένου, υλοποιούν ως συνάρτηση ενεργοποίησης την Διορθωμένη Γραμμική. Έτσι απαλείφονται τυχόν αρνητικές τιμές απο το ένα επίπεδο στο άλλο με τον όγκο εισόδου να παραμένει αμετάβλητος από αυτή την αλλαγή.

- ***Συνάρτηση Ενεργοποίησης Softmax***: Πρόκειται για την συνάρτηση ενεργοποίησης που χρησιμοποιείται αποκλειστικά στο τελευταίο Πλήρως Συνδεδεμένο επίπεδο. Μετασχηματίζει το διάνυσμα εξόδου του νευρωνικού δικτύου σε ένα διάνυμα πιθανοτήτων, υποδεικνύοντας ποιά απο τις κατηγορίες-στόχους αναγνώρισε το δίκτυο ύστερα από την επεξεργασία της εκάστοτε εικόνας.

1.7 Παράδειγμα Πλήρους Λειτουργίας ενός ΣΝΝ

Κλείνοντας, ας δώσουμε ένα παράδειγμα λειτουργίας ενός ΣΝΝ, πολύ κοντά σε αυτό που χρησιμοποιήθηκε στα πλαίσια της εργασίας.
Κατά το σχεδιασμό ενός ΣΝΝ οφείλουμε να γνωρίζουμε κατάρχάς το είδος και τις διαστάσεις του διανύσματος εισόδου καθώς και τις ζητούμενες κλάσεις εξόδου. Στο παράδειγμά μας θα χρησιμοποιήσουμε το σετ εικόνων MNIST, που αποτελείται απο 60.000 ασπρόμαυρες εικόνες χειρόγραφων ψηφίων από το 0-9, μεγέθους 28x28 pixel η καθεμία. Συνεπώς εξάγουμε τα εξής χαρακτηριστικά:

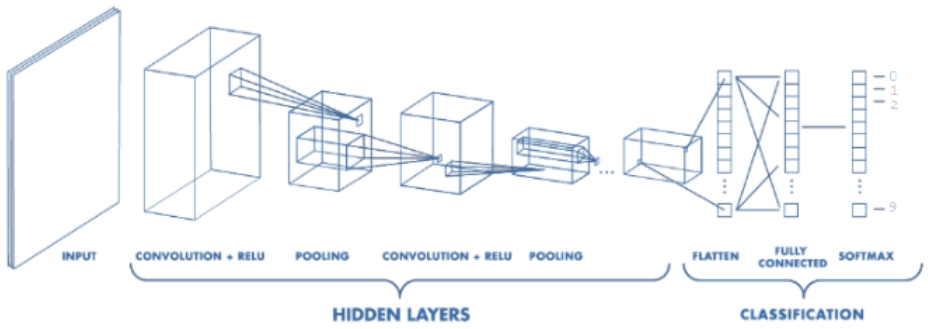
1. ***Διάνυσμα εισόδου***: Μια ασπρόμαυρη εικόνα, διαστάσεων 28x28 [1,28,28,1]

2. ***Διάνυσμα εξόδου***: Ένα διάνυσμα 10 στοιχείων με την πιθανότητα ταύτισης της εισόδου με την εκάστοτε κλάση-ψηφίο [1,10].

Η ροή επεξεργασίας είναι η ακόλουθη:

Επίπεδο	Είσοδος	Πράξεις	Έξοδος
Input	[1,28,28,1]	-	[1,28,28,1]
Conv2D(32,(3,3))	[1,28,28,1]	194688*/21632+	[32,26,26,1]
ReLu	[32,26,26,1]	-	[32,26,26,1]
Conv2D(32,(3,3))	[32,26,26,1]	5308416*/589824+	[32,24,24,1]
ReLu	[32,24,24,1]	-	[32,24,24,1]
MaxPooling2D(2,2)	[32,24,24,1]	-	[32,12,12,1]
Flatten	[32,12,12,1]	-	[1,4608]
Dense(128)	[1,4608]	589824*/128+	[1,128]
ReLu	[1,128]	-	[1,128]
Dense(10)	[1,128]	1280*/10+	[1,10]
Softmax	[1,10]	-	[1,10]

Αξιοσημείωτη είναι η μεταβολή της εισόδου η οποία σταδιακά "χάνει" σε μήκος και πλάτος και φτάνει στο 1x1, ενώ αυξάνει διαρκώς στον όγκο με την τελική μορφή να αποτελεί ένα διάνυσμα, όπως φαίνεται χαρακτηριστικά στο σχήμα 1.8:



Σχήμα 1.8

1.8 Σχετικές Εργασίες

Η μεγάλη απήχηση και ενδιαφέρον των τελευταίων χρόνων στα νευρωνικά δίκτυα έχει επικεντρωθεί σε ακαδημαικό επίπεδο στην εξερεύνηση μεθοδολογιών, αρχιτεκτονικών και αφαιρετικών μηχανισμών για την βελτιστοποίηση τους τοσο στα πλαίσια της ταχύτητας όσο και στα πλαίσια της ακριβειας.

Για το σκοπο αυτό,υλοποιήθηκε ένα πλήθος frameworks για την αναπτυξη τους σε ερευνητική και βιομηχανική κλίμακα, με χαρακτηριστικό παράδειγμα το framework Caffe του πανεπιστημίου Berkley που εγινε γνωστο για την εύκολη ανάπτυξη κώδικα όσο και για την ευελιξία του ως open source project, ενώ αποτελεί το περιβάλλον που οι περισσότερες υπάρχουσες εργασίες χρησιμοποιούν για την δημιουργία και προπόνηση του δικτύου τους [4][7].

Ένα επίσης πρόσφατο περιβαλλον αποτελεί η Tensorflow [9] με την ιδιαίτερη οπτική της να αντιμετωπίζει ένα νευρωνικό δικτύο σαν ένα υπολογιστικό γράφο μέσα στον οποίο ρέουν δεδομένα-διανύσματα(tensors). Αν και ειναι λιγότερο φιλική προς τον προγραμματιστή αποτελεί το state of the art στην ανάπτυξη εφαρμογών Deep learning.

Στο επίπεδο του Hardware εκτενής βιβλιογραφία υπάρχει σε διάφορες μεθόδους απεικόνισης και βελτιστης μεταφορας ενός νευρωνικού δικτύου σε μια πλακετα fpga. Οι περισοτερες προσεγγίσεις υλοποιουν τη συνέλιξη ως ενα κλασσικό φίλτρο dsp-type, εκμεταλευόμενες την υπάρχουσα μεθοδολογία του Dataflow Hardware Mapping ενώ ρίχνουν περισσότερο βάρος στην χρήση μικρότερου bitwidth για τους υπολογισμούς [4][5][6][7]. Επιπλέον αποτελεί συνήθης πρακτική η υλοποίηση μόνο του τμήματος επιτάχυνσης των συνελικτικών επιπέδων και η προώθηση των αποτελεσμάτων για τα πλήρως συνδεδεμένα επίπεδα σε κλασσικές επεξεργαστικές μονάδες όπως στη GPU [7].

Τέλος, το μεγαλυτερο μέρος τους αναζητά αυτοματοποιημένες λύσεις σε επίπεδο παραγωγής χρησιμοποιώντας εργαλεία υψηλής σύνθεσης HLS που επιταχύνουν δραματικά την ικανότητα ανάπτυξης των διαφορων design σε βάρος ωστόσο της δυνατότητας low level παραμετροποίησης [2][7].

1.9 Κίνητρα και Επιδιώξεις

Η εργασία αυτή έχει σκοπό την χρήση υπάρχοντων μεθοδολογιών του Dataflow Hardware Mapping, καθώς και βασικών αρχών Hardware Design όπως το Pipelining και Fir Filter Design προκειμένου να αντιμετωπίσει το πρόβλημα της επιτάχυνσης ενός συνελικτικού νευρώνικου δικτύου αναγνώρισης ψηφίων συμπληρώνοντας κάποιες ελλείψεις των προηγούμενων μεθοδολογιών και εμπλουτίζοντάς τες με την ιδέα τις χρήσης online 8-bit Integer Quantization και 1-bit Image Mapping σε low level επίπεδο.

Συγκεκριμένα επιδιώκουμε:

1. Την εκσυχρόνιση των μηχανισμών διασύνδεσης Hardware και Neural Nets με την εισαγωγή της Tensorflow σαν παραγωγό του μοντέλου ειδόδου στο περιβάλλον σύνθεσης Vivado.

2. Την απεικόνιση και επιτάχυνση σε FPGA κώδικα VHDL διατηρώντας την ευελιξία του HLS design σε μια καθαρά low level υλοποίηση.

3. Την ενσωμάτωση της ιδέας των Wide and Shallow CNNs στο σχεδιασμό μας, υλοποιώντας έτσι ελαφρύτερα απλούστερα design ικανά να μεταφερθούν σε FPGA με μεγαλύτερη ευκολία, μέσω της απλοποίησης του 1ου συνελικτικού επιπέδου χάρις στο 1-bit mapping.

4. Να αποδείξουμε πως εμβαθύνοντας ακόμη περισσότερο σε υπάρχουσες δομές και αρχιτεκτονικές μπορούμε να καταλήξουμε σε ένα πλήρως βέλτιστο design που σέβεται τις ιδαιτερότητες και την φύση του προβλήματος που μας δίνεται. Αυτό είναι και ένα απο τα σημαντικότερα αποτελέσματα-στόχους που πρέπει να τονιστούν, πως δηλαδή το Hardware Design και οι μέθοδοι απεικόνισης βελτιώνονται ακόμη περισσότερο όταν η ίδια η φύση ενός προβλήματος μας επιτρέπει βελτιώσεις (1-bit Image Mapping) και αφαιρέσεις (online 8-bit Integer Quantization) που μια γενικής φύσης αρχιτεκτονική θα παρέβλεπε εντελώς.

Παραθέτουμε λοιπόν τους προβληματισμούς μας καθώς και τη χρονολογική σειρά που αναπτύξαμε και αντιμετωπίσαμε τους εκάστοτε περιορισμούς μέσω της τροποποίησης και εμπλουτισμού νέων και υπάρχοντων δομών.

Κεφάλαιο 2

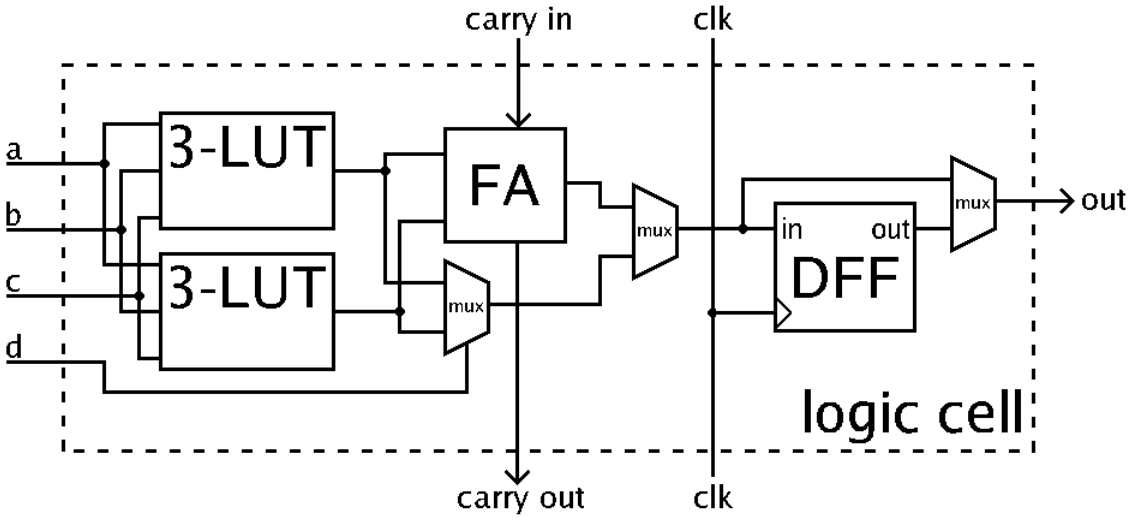
Τα FPGA ως επιταχυντές

2.1 Εισαγωγή στα FPGA

Το FPGA ή συστοιχία επιτόπια προγραμματιζόμενων πυλών είναι τύπος προγραμματιζόμενου ολοκληρωμένου κυκλώματος γενικής χρήσης το οποίο διαθέτει πολύ μεγάλο αριθμό τυποποιημένων πυλών και άλλων ψηφιακών λειτουργιών όπως απαριθμητές, καταχωρητές μνήμης, γεννήτριες PLL κα. Σε ορισμένα από αυτά ενσωματώνονται και αναλογικές λειτουργίες.
Κατά τον προγραμματισμό του FPGA, ο οποίος γίνεται πάντοτε ενώ αυτό είναι τοποθετημένο στο τυπωμένο κύκλωμα, ενεργοποιούνται οι επιθυμητές λειτουργίες και διασυνδέονται μεταξύ τους έτσι ώστε το FPGA να συμπεριφέρεται ως ολοκληρωμένο κύκλωμα με συγκεκριμένη λειτουργία.

Ο κώδικας με τον οποίο προγραμματίζεται το FPGA γράφεται σε γλώσσες περιγραφής υλικού (VHDL, AHDL, Verilog). Βασική δομική μονάδα του FPGA είναι το λογικό μπλοκ, με τη χρήση του οποίου υλοποιούνται οι λογικές συναρτήσεις που εκφράζουν τη λειτουργία ενός ψηφιακού κυκλώματος. Ανάλογα με το μέγεθος του κυκλώματος πολλά λογικά μπλοκ συνδεόνται για να υλοποιήσουν το πλήθος των απαραίτητων λογικών συναρτήσεων.

Γενικά, ένα λογικό μπλοκ αποτελείται από μερικά λογικά κελιά (που ονομάζονται ALM, LE, slice κλπ). Ένα τυπικό κελί αποτελείται από ένα LUT, ένα πλήρες αθροιστή FA και ένα flip-flop τύπου D.



Σχήμα 2.1

Στο παραπάνω σχήμα βλέπουμε ένα απλό λογικό κελί. Οι LUTs χωρίζονται σε δύο LUTs 3 εισόδων. Στην κανονική λειτουργία αυτές συνδυάζονται σε ένα LUT 4 εισόδων μέσω του αριστερού Mux. Σε αριθμητική λειτουργία, οι εξόδοί τους τροφοδοτούνται στην FA. Η επιλογή της λειτουργίας προγραμματίζεται στο μεσαίο πολυπλέκτη. Η έξοδος μπορεί να είναι συγχρονισμένη ή ασύγχρονη, ανάλογα με τον προγραμματισμό του Mux προς τα δεξιά, στο παράδειγμα της εικόνας. Στην πράξη ωστόσο, ολόκληρα ή τμήματα του FA τοποθετούνται ως λειτουργίες στα LUT προκειμένου να εξοικονομηθεί χώρος.

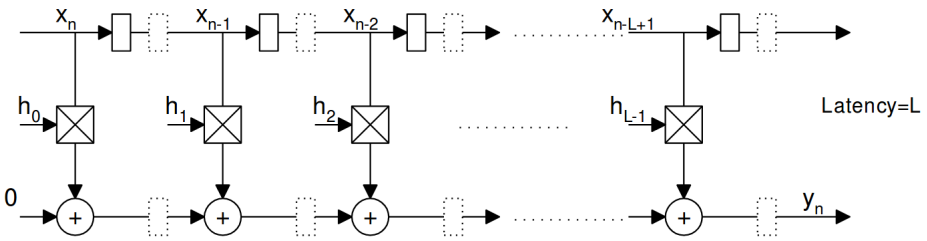
2.2 FPGA και Φίλτρα

Εξαιτίας της μεγάλης τους ευελιξίας τα FPGA βρήκαν ιδιαίτερη αξία στην υλοποίηση ψηφιακών φίλτρων και ιδιαίτερα φίλτρων πεπερασμένης απόκρισης (FIR Filters) για σήματα πραγματικού χρόνου. Σημαντικές απαιτήσεις της συγκεκριμένης εφαρμογής αποτελούν μεγέθη όπως η απόκριση latency, η καθυστέρηση delay, και το fanout ενώ σημαντική κρίνεται και η ύπαρξη η μη συστολικότητας (pipelining), που καθιστά ενα κύκλωμα ικανό να παράγει διαρκώς αποτελέσματα αναπαράγοντας ίδια βασικά σχεδιαστικά τμήματα στο design του.
Η σχέση μεταξύ εισόδου και εξόδου σε ένα ψηφιακό φίλτρο δίνεται απο την σχέση:

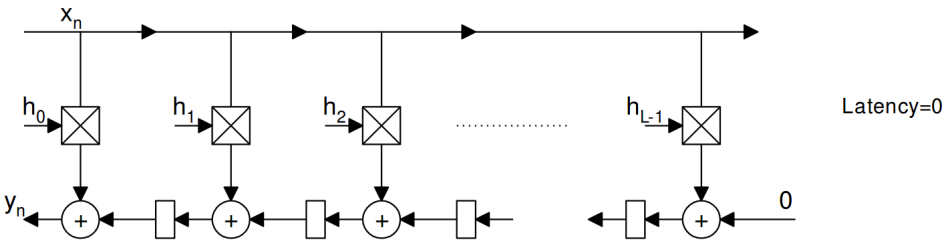
$$y_n = \sum_{i=0}^{L-1} x_{n-i} \cdot h_i$$

Στο πεδίο του χρόνου η έξοδος y δίνεται απο τη συνέλιξη της εισόδου με τους συντελεστές (σταθερές) h, όπου οι συντελεστές αποτελούν την κρουστική απόκριση του φίλτρου.
Υπάρχουν κυρίως 2 μέθοδοι υλοποίησης ενός ψηφιακού φίλτρου η απευθείας (Direct) και η ανάστροφη (Transpose).

Στο σχήμα 2.2 φαίνεται μια Direct ενώ στο σχήμα 2.3 μια Transpose υλοποίηση όπου τα παραλληλόγραμα συνιστούν μονάδες καθυστέρησης(D Flip Flops), τα παραλληλόγραμμα με X πολλαπλασιαστές ενώ οι κύκλοι με + αθροιστές.



Σχήμα 2.2 Direct Form



Σχήμα 2.3 Transpose Form

Παρατηρούμε ότι η βασική διαφορά είναι η τοποθέτηση των μονάδων καθυστέρησης που δημιουργεί διαφορετικά μερικά αποτελέσματα αποθηκευμένα στις διάφορες μοναδες καθυστέρησης, παράγοντας ωστόσο το ίδο τελικό αποτέλεσμα.

Αξίζει να σημειωθεί πως και οι 2 μορφές μπορούν να αποτελέσουν συστολικά συστήματα, ωστόσο η Transpose μορφή έχει 0 latency και μπορεί να δίνει αποτελέσματα από τον πρώτο κύκλο ρολογιού, ενώ έχει μικρότερο "κρίσιμο μονοπάτι" (critical path) και εισάγει μικροτερη καθυστερηση στον υπολογισμό του αποτελέσματος.

2.3 Η αρχή του Pipelined Design

Όπως έχουμε αναφέρει, ο προγραμματισμός ενός FPGA είναι μια διαδικασία προσαρμογής των πόρων του για την υλοποίηση μιας ορισμένης λογικής λειτουργίας ή ψηφιακής επεξεργασίας. Κατά τη διάρκεια της διαδικασίας σχεδιασμού, ένα σημαντικό κριτήριο που πρέπει να ληφθεί υπόψη είναι το πρόβλημα χρονισμού που είναι εγγενές στο σύστημα, καθώς και οι τυχόν περιορισμοί που θέτει ο χρήστης. Ένας μηχανισμός σχεδίασης που μπορεί να βοηθήσει έναν σχεδιαστή να επιτύχει αυτόν τον στόχο είναι σωληνώσεις(Pipelining).

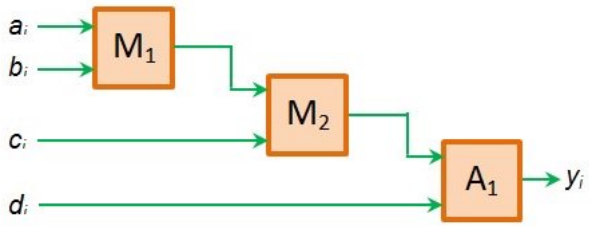
Ένα pipelined design επιτρέπει την παράλληλη εκτέλεση των υπολογισμών μιας αρχιτεκτονικής ώστε να παράγει όσο το δυνατόν αποτελέσματα συνεχόμενα κατανέμοντας πάντα πόρους σε ετερόχρονες διαδικασίες. Στα FPGAs, αυτό επιτυγχάνεται με την οργάνωση πολλαπλών μπλοκ επεξεργασίας δεδομένων με ένα συγκεκριμένο τρόπο. Γι άυτό, διαιρούμε πρώτα το γενικό μας κύκλωμα λογικής σε αρκετά -πιθανώς επαναλαμβανόμενα- μικρά κομμάτια και στη συνέχεια τα ξεχωρίζουμε χρησιμοποιώντας κόμβους καθυστέρησης (flip-flops).

Για παράδειγμα ας ρίξουμε μια ματιά σε ένα σύστημα τριών πολλαπλασιασμών που ακολουθείται από μια πρόσθεση σε τέσσερις συστοιχίες εισόδου. Επομένως, το αποτέλεσμά μας θα είναι ίσο με

$$y_i = (a_i \cdot b_i \cdot c_i) + d_i.$$

Σχεδιασμός χωρίς σωλήνωση:

Ο πρώτος σχεδιασμός που έρχεται στο μυαλό για να δημιουργηθεί ένα τέτοιο σύστημα θα ήταν πολλαπλασιαστές ακολουθούμενοι από έναν αθροιστή, όπως φαίνεται στο Σχήμα 2.4



Σχήμα 2.4 Χωρίς Σωλήνωση

Εδώ αναμένουμε ότι η ακολουθία των λειτουργιών θα είναι ο πολλαπλασιασμός των στοιχείων a και b με τον πολλαπλασιαστή M1, ακολουθούμενη από τον πολλαπλασιασμό του προϊόντος του με τον πολλαπλασιαστή M2 και τελικά την προσθήκη του προκύπτοντος προϊόντος με το d με τον αθροιστή A1.

Παρόλα αυτά, όταν το σύστημα είναι σχεδιασμένο να είναι συγχρονισμένο, στο πρώτο κύκλο ρολογιού, μόνο ο πολλαπλασιαστής M1 μπορεί να παράγει έγκυρα δεδομένα στην έξοδο του . Αυτό οφείλεται στο γεγονός ότι, σε αυτή τη στιγμή, μόνο οι M1 έχουν έγκυρα δεδομένα (a και b) στους ακροδέκτες εισόδου τους, σε αντίθεση με τα M2 και A1.

Στο δεύτερο κύκλο ρολογιού, θα υπήρχαν έγκυρα δεδομένα στους ακροδέκτες εισόδου και των δύο M1 και M2. Ωστόσο, τώρα πρέπει να διασφαλίσουμε ότι λειτουργεί μόνο το M2 ενώ ο M1 διατηρεί την παραγωγή του όπως είναι. Αυτό οφείλεται στο γεγονός ότι, σε αυτή τη στιγμή, εάν η M1 λειτουργεί, τότε η γραμμή εξόδου της μετατρέπεται σε (a2 × b2) αντί της αναμενόμενης τιμής της (a1 × b1) οδηγώντας σε λανθασμένη έξοδο M2

$$(a2 \cdot b2 \cdot c1)$$

και όχι

$$(a1 \cdot b1 \cdot c1).$$

Στον τρίτο κύκλο ρολογιού, θα υπάρχουν έγκυρες εισόδοι και στα τρία στοιχεία: M1, M2 και A1. Παρόλα αυτά, θέλουμε μόνο τον αθροιστή να είναι λειτουργικός, όπως με έξοδο

$$y1 = (a1 \cdot b1 \cdot c1 + d1)$$

Αυτό σημαίνει ότι η πρώτη έξοδος του συστήματος θα είναι διαθέσιμη μετά το τρίτο ρολόι. Στη συνέχεια, καθώς φτάνει το τέταρτο ρολόι, το M1 μπορεί να λειτουργήσει πάνω από το επόμενο σύνολο δεδομένων: a2 και b2. Αλλά αυτή τη στιγμή, οι M2 και A1 αναμένεται να είναι αδρανείς. Αυτό θα πρέπει να ακολουθείται από την ενεργοποίηση του M2 μόνο στο πέμπτο ρολόι και την ενεργοποίηση του A1 μόνο στο έκτο. Αυτό εξασφαλίζει την επόμενη έξοδο μας,

$$y2 = (a2 \cdot b2 \cdot c2) + d2.$$

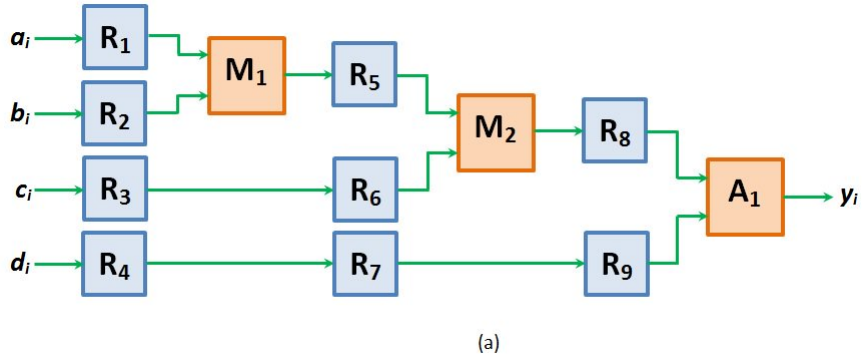
Όταν ακολουθείται ένα παρόμοιο μοτίβο διέγερσης μπορούμε να αναμένουμε ότι οι επόμενες εξόδοι θα εμφανιστούν στους κύκλους 9, 12, 15 και ούτω καθεξής (Σχήμα 2.5).

Clk	1	2	3	4	5	6	7	8	9	10	…
M_1	$a_1 \times b_1$	-	-	$a_2 \times b_2$	-	-	$a_3 \times b_3$	-	-	$a_4 \times b_4$	
M_2	-	$a_1 \times b_1 \times c_1$	-	-	$a_2 \times b_2 \times c_2$	-	-	$a_3 \times b_3 \times c_3$	-	-	
A_1	-	-	$a_1 \times b_1 \times c_1 + d_1$	-	-	$a_2 \times b_2 \times c_2 + d_2$	-	-	$a_3 \times b_3 \times c_3 + d_3$	-	
			y_1			y_2			y_3	…	

Σχήμα 2.5

Σχεδιασμός με Σωλήνωση:

Τώρα, ας υποθέσουμε ότι προσθέτουμε καταχωρητές σε αυτό το σχέδιο στις εισόδους (R1 έως R4), μεταξύ M1 και M2 (R5 και R8, αντίστοιχα) και κατά μήκος των μονοπατιών άμεσης εισόδου (R6, R7 και R9), όπως φαίνεται στο σχήμα 2.6



(a)

Σχήμα 2.6

Εδώ, στο πρώτο ρολόι, οι έγκυρες εισόδοι εμφανίζονται μόνο για τους καταχωρητές R1 έως R4 (a1, b1, c1 και d1 αντίστοιχα) και για τον πολλαπλασιαστή M1 (a1 και b1). Ως αποτέλεσμα, μόνο αυτά μπορούν να παράγουν έγκυρες εξόδους. Επιπλέον, όταν ο M1 παράγει το αποτέλεσμα του, μεταβιβάζεται στον R5 και να αποθηκεύεται σε αυτόν.

Στο δεύτερο ρολόι, οι τιμές που αποθηκεύονται στους καταχωρητές R5 και R6 γίνονται είσοδοι στο M2 που του επιτρέπουν να καταστήσει την έξοδο του ως

$$a1 \cdot b1 \cdot c1$$

ενώ η έξοδος του R4 (d1) μετατοπίζεται στο R7. Εν τω μεταξύ, το δεύτερο σύνολο δεδομένων (a2, b2, c2 και d2) εισέρχεται στο σύστημα και εμφανίζεται στις εξόδους του R1 έως R4.

Στην περίπτωση αυτή, επιτρέπεται στο M1 να λειτουργεί στις εισόδους του έτσι ώστε η γραμμή εξόδου του να μεταβάλλεται από a1 x b1 σε a2 x b2, σε αντίθεση με την περίπτωση του προηγούμενου σχεδιασμού.

Αυτό συμβαίνει επειδή σε αυτό το σχέδιο, οποιαδήποτε μεταβολή στην έξοδο του M1 δεν επηρεάζει την έξοδο του M2. Αυτό οφείλεται στο γεγονός ότι τα δεδομένα που απαιτούνται για την εξασφάλιση της σωστής λειτουργικότητας του M2 ήταν ήδη μανδαλωμένα στον R5 κατά τη διάρκεια του πρώτου ρολογιού.

Η εισαγωγή του καταχωρητή R5 έχει καταστήσει τα M1 και M2 λειτουργικά ανεξάρτητα και μπορούν να λειτουργούν ταυτόχρονα σε διαφορετικά σύνολα δεδομένων.

Στον τρίτο κύκλο ρολογιού, οι έξοδοι των καταχωρητών R8 και R9 a1 x b1 x c1 και d1 διαβιβάζονται ως είσοδοι στον αθροιστή A1. Ως αποτέλεσμα, παίρνουμε την πρώτη μας έξοδο

$$y1 = ((a1 \cdot b1 \cdot c1) + d1)$$

Παρόλα αυτά, στο ίδιο ρολόι, οι M1 και M2 θα είναι ελεύθεροι να λειτουργούν με a3, b3 και a2, b2, c2, αντίστοιχα. Αυτό είναι εφικτό λόγω της παρουσίας καταχωρητών R5 που απομονώνουν το μπλοκ M1 από τα M2 και R8 απομονώνοντας τον πολλαπλασιαστή M2 από τον αθροιστή A1.

Επομένως, θα έχουμε ακόμη (a3 × b3) και (a2 × b2 × c2) από M1 και M2, αντίστοιχα, εκτός από το y1.

Τώρα όταν φτάσει ο τέταρτος χτύπος, ο αθροιστής A1 λειτουργεί στις εισόδους του για να δώσει τη δεύτερη έξοδο,

$$y2 = ((a2 \cdot b2 \cdot c2) + d2).$$

Επιπλέον, η έξοδος του M1 αλλάζει από (a3 × b3) σε (a4 × b4) ενώ η τιμή του M2 αλλάζει από (a2 × b2 × c2) σε (a3 × b3 × c3).

Ακολουθώντας τον ίδιο τρόπο λειτουργίας, μπορούμε να αναμένουμε να εμφανιστούν τα δεδομένα εξόδου για κάθε χτύπο από τότε (Σχήμα 2.7), αντίθετα με την περίπτωση μη σχεδιασμένου σχεδιασμού όπου έπρεπε να περιμένουμε τρεις κύκλους ρολογιού για να βγάλουμε κάθε ένα δεδομένων εξόδου.

Clk	1	2	3	4	5	...
R_1	a_1	a_2	a_3	a_4	…	
R_2	b_1	b_2	b_3	b_4	…	
R_3	c_1	c_2	c_3	c_4	…	
R_4	d_1	d_2	d_3	d_4	…	
M_1	$a_1 \times b_1$	$a_2 \times b_2$	$a_3 \times b_3$	$a_4 \times b_4$	…	
R_5	-	$a_1 \times b_1$	$a_2 \times b_2$	$a_3 \times b_3$	$a_4 \times b_4$	…
R_6	-	c_1	c_2	c_3	c_4	…
R_7	-	d_1	d_2	d_3	d_4	…
M_2	-	$a_1 \times b_1 \times c_1$	$a_2 \times b_2 \times c_2$	$a_3 \times b_3 \times c_3$	$a_4 \times b_4 \times c_4$	…
R_8	-	-	$a_1 \times b_1 \times c_1$	$a_2 \times b_2 \times c_2$	$a_3 \times b_3 \times c_3$	...
R_9	-	-	d_1	d_2	d_3	...
A_1	-	-	$a_1 \times b_1 \times c_1 + d_1$	$a_2 \times b_2 \times c_2 + d_2$	$a_3 \times b_3 \times c_3 + d_3$	…
			y_1	y_2	y_3	...

(b)

Σχήμα 2.7

Συνέπειες του Pipelining

1. **Latency**:

   Στο παραδειγμένο παράδειγμα, ο σχεδιασμός μέσω σωληνώσεων δείχνει ότι παράγει μία έξοδο για κάθε ρολόι από τον τρίτο κύκλο του ρολογιού. Αυτό οφείλεται στο γεγονός ότι κάθε είσοδος πρέπει να περάσει από τρεις καταχωρητές (που συνθέτουν το βάθος του αγωγού) κατά την επεξεργασία πριν φτάσει στην έξοδο. Παρομοίως, αν έχουμε αγωγό βάθους n, τότε οι έγκυρες έξοδοι εμφανίζονται μία ανά κύκλο μόνο από τον n-οστό κύκλο.

   Αυτή η καθυστέρηση που συνδέεται με τον αριθμό των κύκλων ρολογιού που χάθηκαν πριν από την πρώτη έγκυρη έξοδο εμφανίζεται ως latency. Όσο μεγαλύτερος είναι ο αριθμός των σταδίων, τόσο μεγαλύτερη είναι η καθυστέρηση που θα συσχετιστεί με αυτό.

2. **Αύξηση της λειτουργικής συχνότητας ρολογιού**:

   Ο μη-σωληνομένος προηγούμενος σχεδιασμός παράγει μία έξοδο για κάθε τρεις κύκλους ρολογιού. Δηλαδή, αν έχουμε ένα ρολόι περιόδου 1 ns, τότε η είσοδος παίρνει 3 ns (3 $\times$ 1 ns) για να επεξεργαστεί και να εμφανιστεί ως έξοδος.

   Αυτή η μεγαλύτερη διαδρομή δεδομένων θα ήταν τότε η κρίσιμη διαδρομή, η οποία αποφασίζει την ελάχιστη συχνότητα του ρολογιού του σχεδιασμού μας. Με άλλα λόγια, η συχνότητα του σχεδιασμένου συστήματος δεν πρέπει να είναι μεγαλύτερη από (1/3 ns) = 333,33 MHz για να εξασφαλιστεί ικανοποιητική λειτουργία.

   Στον σχεδιασμό με σωληνώσεις, μετά από την αρχική καθυστέρηση, παράγεται μία έξοδος για κάθε χτύπο ρολογιού. Έτσι, η συχνότητα του ρολογιού λειτουργίας μας είναι ίδια με αυτή του καθορισμένου ρολογιού (εδώ, είναι 1 / 1ns = 1000 MHz).

   Αυτά τα αριθμητικά στοιχεία δείχνουν σαφώς ότι ο σχεδιασμός μέσω σωληνώσεων αυξάνει σημαντικά τη λειτουργική συχνότητα σε σύγκριση με τη μη αγωγού.

3. **Μεγαλύτερη αξιοποίηση λογικών πόρων**:

   Στο σχεδιασμό με σωληνώσεις, χρησιμοποιούμε καταχωρητές για την αποθήκευση των αποτελεσμάτων των μεμονωμένων σταδίων του σχεδιασμού.

   Αυτά τα στοιχεία προσθέτονται στους πόρους και κάνουν το design αρκετά μεγάλο από πλευράς υλικού. Εξαναγκάζουν ωστόσο όσο περισσότερους υπολογιστικούς πόρους γίνεται σε διαρκή λειτουργία και κατα συνέπεια εξασφαλίζουν μεγαλύτερη παραγωγή αποτελεσμάτων(throughput).

Κεφάλαιο 3

Υλοποίηση του **FPGA Accelerator**

3.1 Βιβλιοθήκη Απεικόνισης

Οι παρακάτω ενότητες αποτελούν την generic βιβλιοθήκη που ο τελικός γεννήτορας χρησιμοποιεί για το μετασχηματισμό ενός μοντέλου από Python σε VHDL.

3.2 Βιβλιογραφικές Μεθοδολογίες Σχεδιασμού

Στο κεφάλαιο αυτό θα παρουσιάσω τις βιβλιογραφικές μεθοδολογίες που χρησιμοποιήθηκαν για την αποτελεσματική απεικόνιση και μεταφορά των διαφόρων στοιχείων του Νευρωνικού δικτύου σε επίπεδο Hardware, μέσω της γλώσσας περιγραφής υλικού VHDL.

3.2.1 Έλεγχος Εισόδου

Τα δεδομένα του μοντέλου μας αποτελούν ασπρόμαυρες εικόνες με μοναδιαίο βάθος χρώματος διαστάσεων 28 επί 28 pixel, συνεπώς ένας 3D πίνακας 1x28x28. Η είσοδος ωστόσο γίνεται με το λεγόμενο C-Ordering, δηλαδή ανά pixel ανα βάθος χρώματος, στην περίπτωσή μας δηλαδή σειριακά απο την πάνω αριστερή προς την κάτω δεξιά γωνία. Χρησιμοποιούμε για το λόγο αυτό τα εξής σήματα:

1. **in_dv (Input Data Valid)**: *Έγκυρο Πίξελ Εισόδου*
   Παίρνει την τιμή 1 όσο το επόμενο Pixel είναι έγκυρο και τίθεται προς επεξεργασία, 0 σε άλλη περίπτωση.

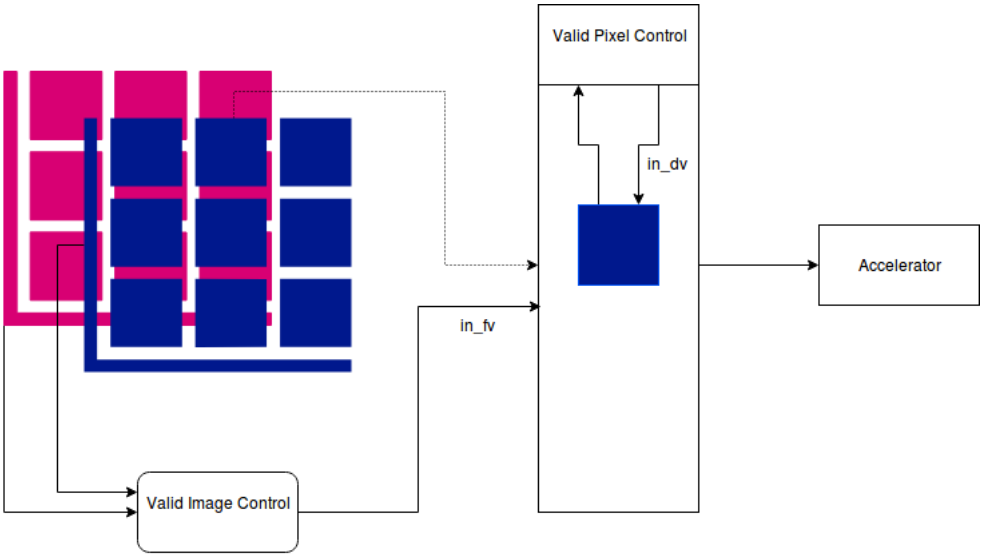
2. **in_fv (Input Frame Valid)**: *Έγκυρη Εικόνα Εισόδου*
   Παίρνει την τιμή 1 όσο η τρέχουσα εικόνα είναι έγκυρη και τίθεται προς επεξεργασία, 0 σε άλλη περίπτωση.

   Σαν συνέπεια το σήμα in_fv διατηρείται 1 για 28*28*1 = 784 κύκλους σε περίπτωση που το σήμα in_dv είναι 1, δηλαδή δεν έχει προκύψει κάποιο πρόβλημα μεταφοράς η αλλοίωσης της εισόδου.

3. **Enable**: *Διακόπτης λειτουργίας*
   Παίρνει την τιμή 1 όσο ο επιταχυντής τίθεται προς λειτουργία, 0 σε άλλη περίπτωση.

4. **Reset_n**: *Επαναφορά*
   Παίρνει την τιμή 0 για τον καθαρισμό των ενδιάμεσων καταχωτητών του επιταχυντή και επαναφορά της λειτουργίας του, 1 σε άλλη περίπτωση.

Παρακάτω παρουσιάζεται η αναπαράσταση του τμήματος εισόδου:



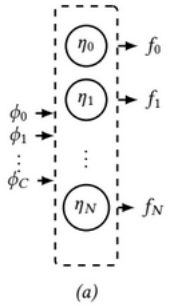
Σχήμα 2.8

3.2.2 Συνελικτικό Επίπεδο -Naive Design

Στη συνέχεια του design μας, υλοποιούμε τα συνελικτικά επίπεδα. Η πρώτη σχεδιαστική μας επιλογή χρησιμοποιεί ως δομικά στοιχεία συνελικτικά και αθροιστικά κύτταρα, τα οποία χρησιμοποιούν την τεχνική πολλαπλασιασμού - συσσώρευσης (multiply-accumulate).

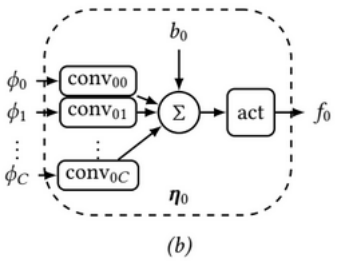
Συγκεκριμένα τα συνελικτικά κύτταρα χρησιμοποιούν τους αποθηκευμένους σε ROM συντελεστές του εκάστοτε συνελικτικού επιπέδου προκειμένου να δώσουν το άθροισμα των γινομένων τους με τους συνεχείς ερχόμενους χάρτες pixel. Στη συνέχεια τα αθροιστικά κύτταρα προσθέτουν τα ανά κανάλι σταθμισμένα βάρη στο προηγούμενο αποτέλεσμα δίνοντας μας του συνελιγμένους χάρτες pixel -βάθους όσα και τα φίλτρα του συνελικτικού επιπέδου.

Συνεπώς κάθε συνελικτικό επίπεδο αποτελείται από τόσα συνελικτικά κύτταρα όσα τα φίλτρα του επιπέδου επί τα κανάλια εισόδου καθώς και απο ένα αθροιστικό κύτταρο ανα φίλτρο καθένα από τα οποία λαμβάνει τους συντελεστές του απο την δική του μνήμη.



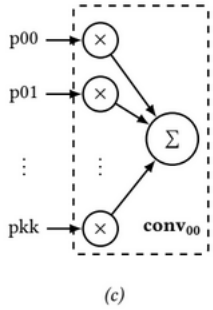
Σχήμα 2.9 Συνελικτικό Επίπεδο

Στο σχήμα 2.9 έχουμε ένα συνελικτικό επίπεδο με τα $\varphi_0, \varphi_1, ..., \varphi_n$ να αποτελούν κανάλια εισόδου μιάς εικόνας, τα $\eta_0, \eta_1, ..., \eta_n$ οι συνελικτικοί νευρώνες και τα $f_0, f_1, ..., f_n$ τα κανάλια εξόδου.



Σχήμα 2.10 Συνελικτικός Νευρώνας

Στο σχήμα 2.10 έχουμε ένα συνελικτικό νευρώνα η_0, με όλα τα $\varphi_0, \varphi_1, ..., \varphi_n$ να αποτελούν εισόδους του. Εσωτερικά χρησιμοποιεί $conv00, conv01, ..., conv0C$, C σε αριθμό, συνελικτικά κύτταρα καθώς και 1 αθροιστικό κύτταρο και αφού εφαρμόσει στο αποτέλεσμα τη συνάρτηση ενεργοποίησης act(activation function) δίνει το f_0 ,το κανάλι-εξόδό του.

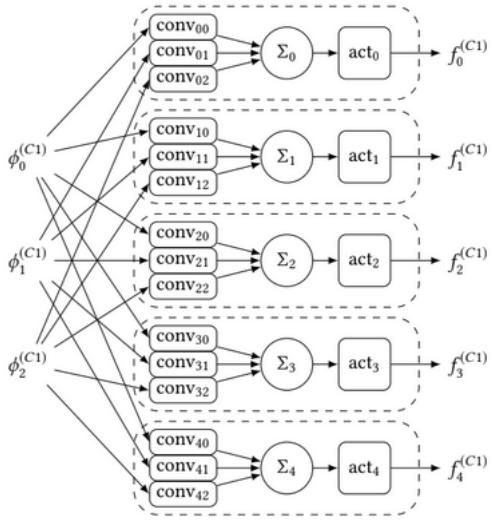


Σχήμα 2.11 Συνελικτικό Κύτταρο

Στο σχήμα 2.11 έχουμε το συνελικτικό κύτταρο *conv*00, με τον χάρτη pixel p00,p01,...,pkk για μια KxK εικόνα, ως είσοδο.

Εσωτερικά χρησιμοποιεί kxk σε αριθμό, πολλαπλασιαστές καθώς και 1 αθροιστή ώστε να πραγματοποιήσει την λειτουργία mac (multiply-accumulate) και να δώσει το συσσωρευμένο αποτέλεσμα ως έξοδο.

Στο design αυτό η άμεση χαρτογράφηση του CNN στο υλικό καταργεί πλήρως την ανάγκη για μια εξωτερική μνήμη για την αποθήκευση ενδιάμεσων αποτελέσματων ή παραμέτρων. Επιπλέον, χάρη στη χρήση μοντέλου σωληνώσεων στην επεξεργασία και είσοδο δεδομένων, το throughput περιορίζεται μόνο από τη μέγιστη συχνότητα ρολογιού.



Σχήμα 2.12 Αρχικός Σχεδιασμός Συνελικτικού επιπέδου

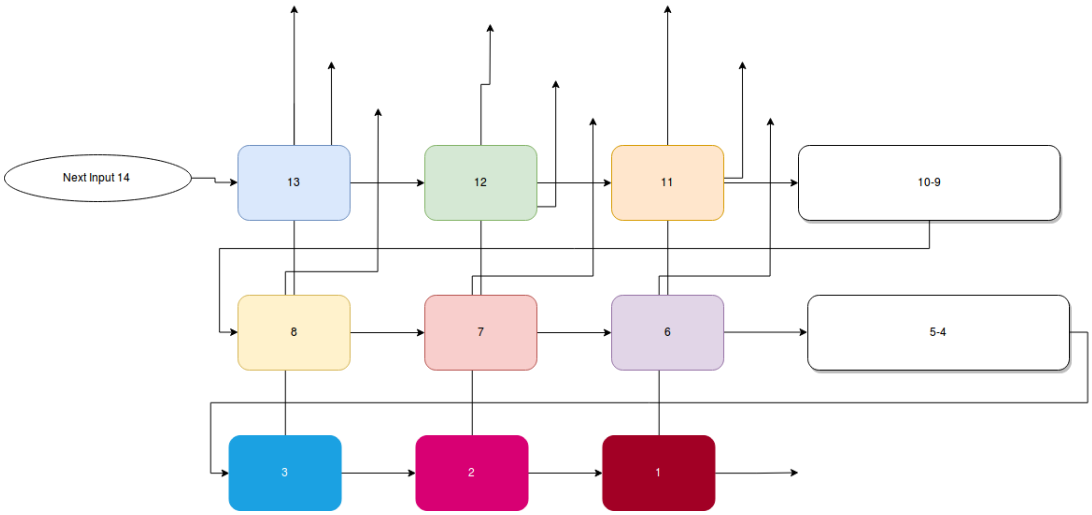
Ωστόσο, αυτά τα πλεονεκτήματα έρχονται με το κόστος υψηλής κατανάλωσης πόρων, αφού ολόκληρο το μοντέλο πρέπει να χαρτογραφηθεί στους φυσικούς πόρους του FPGA. Επομένως, είναι ζωτικής σημασίας να διασφαλίσουμε ότι οι βασικές λειτουργίες που εμπλέκονται στους φορείς του CNN μπορούν να μεταφραστούν με ευελιξία στο υλικό. Τα σημαντικότερα ζητήματα είναι εκείνα που σχετίζονται με τις απαιτήσεις μνήμης on-chip και την αποτελεσματική εφαρμογή αριθμητικών πράξεων.

Οι λόγοι αυτοί μας οδήγησαν στην δημιουργία ενός εμβόλιμου τμήματος μεταξύ εισόδου και συνελικτικού επιπέδου, μετασχηματίζοντας το αρχικό μας design, του Συστολικού Παραγωγού Παραθύρων Γειτνίασης.

3.2.3 Συστολικός Παραγωγός Παραθύρων Γειτνίασης

Στο βελτιωμένο design μας επιλέγουμε να αντιμετωπίζουμε τα συνελικτικά επίπεδα όχι απλά ώς συστοιχία πολλαπλασιασμών και προσθέσεων αλλά σαν συστολικά ψηφιακά φίλτρα ψευδο-ανάστροφης μορφής (Pseudo-Transposed FIR Filters). Για το σκοπό αυτό οι είσοδοι θα έπρεπε να έρχονται με τέτοιο και η συνέλιξη να μετασχηματισθεί ώστε να διατηρώνται οι απαιτήσεις του σχεδιασμού με σωληνώσεις. Όσο αναφορά την έλευση των δεδομένων αυτή αντιμετωπίζεται με ένα σύστημα σωληνώσεων και καταχωρητών (buffered taps system) που οργανώνει τα δεδομένα ώστε μετά την αρχική συσσώρευση ενός αρχικού παραθύρου συνέλιξης δίνει αποτελέσματα ανά κύκλο ρολογιού, όπως ακριβώς και στα ψηφιακά φίλτρα.

Μια σχηματική αναπαράσταση καθώς και η λειτουργία της για μία εικόνα δίνεται παρακάτω:



Σχήμα 2.13

Ας σκευτούμε μια εικόνα διαστάσεων 5x5 pixel με παράθυρο συνέλιξης 3x3 pixel. Το πρώτο αποτέλεσμα συνέλιξης θα δωθεί όταν έχουν εισέλθει στο σύστημα τα pixel 1-13, μιάς και αυτά συμμετέχουν στο πρώτο παράθυρο, όπως φαίνεται στο σχήμα 2.14.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Σχήμα 2.14

Δημιουργώντας ένα σύστημα αγωγών-καταχωρητών 3x3, δηλαδή με το μέγεθος του παραθύρου συνέλιξης του εκάστοτε επιπέδου και επιπλέον σε κάθε πλήν της τελευταίας γραμμής μιας αποθήκης μήκους Μέγεθος Εικόνας - Μέγεθος Παραθύρου, (5-3 = 2) επιτυγχάνουμε το σωστό αποτέλεσμα.

Αξίζει να παρατηρήσουμε τη συμπεριφορά του συστήματος τον επόμενο κύκλο όπου κάθε τιμή θα προχωρήσει μια θέση προς την έξοδο της σωλήνωσης ενώ την πρώτη θέση θα κατα-

λάβει η τιμή 14. Τότε η έξοδος θα είναι ακριβώς ίδια με το επόμενο παράθυρο συνέλιξης που φαίνεται στο σχήμα 2.15.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Σχήμα 2.15

Σαν έξοδο ο συστολικός παραγωγός έχει ένα χάρτη pixel μεγέθους ίσου με ένα συνελικτίκό παράθυρο.
Ενημερώνει επίσης τα προαναφερθέντα σήματα ελέγχου in_dv,in_fv ανάλογα με την παραγωγή σωστού αποτελέσματος καθώς και την εναλλαγή εικόνας επεξεργασίας μιας και το σύστημα μπορεί να διαθέτει pixel απο 2 εικόνες λόγω του pipelined design κάποια δεδομένη στιγμή. Παραθέτω τα κομμάτια κώδικα του συστολικου παραγωγού (neighExctractor):

και του συστήματος αγωγών-καταχωρητών (taps):

Η σχεδιαστική αυτή επιλογή δεν είναι τυχαία, αντίθετα στηρίζεται στο μετασχηματισμό του νευρωνικού δικτύου σε ενα υπολογιστικό γράφο ροής δεδομένων Dataflow Model -Graph- of Computation (MoC). Έτσι έχουμε μια αρχιτεκτονική όπου πολλά κομμάτια οδηγιών μπορούν να επεξεργαστούν ταυτόχρονα μια ροή δεδομένων. Αρχιτεκτονικές που σέβονται τη σημασιολογία ροής δεδομένων θεωρούνται ως ένα δίκτυο (γράφος) των βασικών μονάδων επεξεργασίας, που συνήθως ονομάζονται δράστες και μεταδίδουν αφηρημένα μηνύματα δεδομένων σε μονοκατευθυντικά κανάλια πληροφορίας.

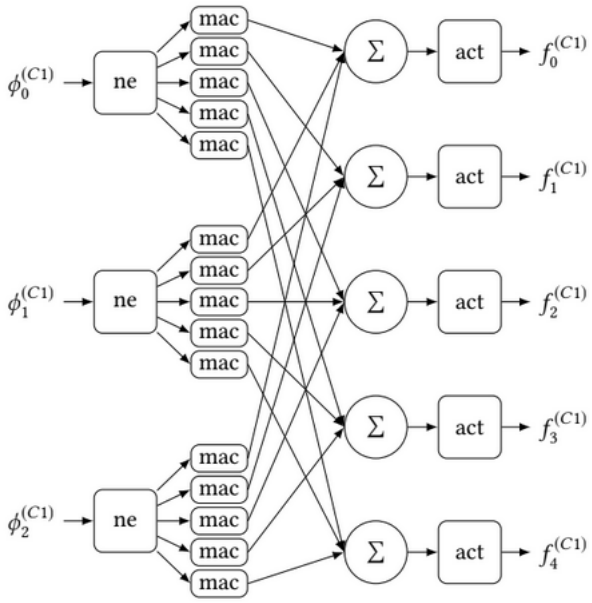
Όσον αφορά την αντιστοίχιση αρχιτεκτονικής-εφαρμογής, η διάταξη του CNN είναι φυσικά ένα μοντέλο υπολογισμών που βασίζεται σε ροή. Με άλλα λόγια, οι αλγόριθμοι επεξεργασίας του CNN μπορούν να μοντελοποιηθούν ως δίκτυα επεξεργασίας ροής (DPNs) όπου οι κόμβοι αντιστοιχούν στους επεξεργαστές-δράστες και οι ακμές αντιστοιχούν στα κανάλια επικοινωνίας. Κάθε δράστης ενεργοποιείται μόνο από τη διαθεσιμότητα των τελεστών εισόδου του και μόνο τότε προωθεί πληροφορία.

Η προσέγγιση αυτή συνίσταται στη φυσική χαρτογράφηση εξ ολοκλήρου γραφήματος των δραστών στο υλικό,τους πόρους του FPGA, ακριβώς όπως αναφέρθηκε παραπάνω.

3.2.4 Συνελικτικό Επίπεδο -Optimized Design

Όταν υιοθετούμε μια προσέγγιση χαρτογράφησης ροής δεδομένων DHM, είναι δυνατόν να παραγοντοποιήσουμε τη διαδικασία παραγωγής παραθύρων γειτνίασης για τη βελτιστοποίηση του memory footprint των συνελικτικών επιπέδων.

Σε αυτή την περίπτωση, είναι δυνατό να βασιστούμε μόνο σε on-chip memory για να επεξεργαστούμε ένα ολόκληρο συνελικτικό επίπεδο - ανεξαρτήτως του αριθμού των φίλτρων, μιας και πλεον το overhead δεν υπάρχει στην ροή των δεδομένων. Αντίθετα, επειδή οι πολλαπλοί νευρώνες σε ένα δεδομένο στρώμα έχουν τα ίδια χαρακτηριστικά εισόδου για επεξεργασία (μόνο οι συντελεστές των φίλτρων αλλάζουν),ο παραγωγός παραθύρων μπορεί να παραγοντοποιηθεί για κάθε επίπεδο χαρακτηριστικών εισόδου. Αυτό θα διαιρέσει τις απαιτήσεις μνήμης για κάθε στρώμα με παράγοντα N, όσο δηλαδή και ο βαθμός παραλληλιας του παραγωγού παραθύρων.



Σχήμα 2.16 Βελτιστοποιημένος Σχεδιασμός Συνελικτικού Επιπέδου

Στο σχήμα 2.16 έχουμε τη συνέλιξη μιας εικόνας 3 καναλιών εισόδου με 5 συνελικτικά φίλτρα για την παραγωγή 5 καναλιών εξόδου. Η παραγοντοποίηση του παραγωγού παραθύρων έχει βαθμό 3 και οι συνελικτικοί νευρώνες έχουν αντικατασταθέι απο συνελικτικά κύτταρα που αναλαμβάνουν για τα ίδια δεδομένα εισόδου τον υπολογισμό του μερικού αποτελέσματος των 5 φίλτρων. Τα μερικά αποτελέσματα συν τα σταθμισμένα βάρη προστίθενται στα αθροιστικά κύτταρα στις εξόδους των οποίων εφαρμόζεται η συνάρτηση ενεργοποίησης και τελικά η έξοδος.

Τα ωφέλη του παραπάνω σχεδιασμού είναι δραματικά σχετικά με τη μείωση των απαιτήσεων μνήμης. Για παράδειγμα, το πρώτο στρώμα του AlexNet CNN με 96 RGB φίλτρα 11 επί 11 πιξελ θα απαιτούσε την επεξεργασία $96 \times 3 \times 11 \times 11 = 34$KB μνήμης ενώ η παραγοντοποίηση οδηγεί σε 96 φορές λιγότερες απαιτήσεις μνήμης 0.3KB.

3.3 Προτεινόμενες Μεθοδολογίες Σχεδιασμού

Στο κεφάλαιο αυτό θα παρουσιάσω τις προτεινόμενες από εμάς μεθοδολογίες που χρησιμοποιήθηκαν για την υποστήριξη του γενικότερου πλαισίου τροποποιήσεων και βελτιώσεων του σχεδιασμού.

3.3.1 Αριθμητικοί υπολογισμοί

Διάφορες μελέτες κατέδειξαν ότι τα CNN, και γενικότερα οι εφαρμογές βαθιάς μάθησης, συνήθως ανέχονται προσεγγιστικούς υπολογισμούς με πράξεις περιορισμένου εύρους bit. Συγκεκριμένα, μια ακρίβεια 16-8 bit είναι επαρκής για να συναχθεί CNN με ελάχιστη ή μηδενική υποβάθμιση. Η DHM προσέγγιση που υποστηρίζουμε μπορεί πραγματικά να επωφεληθεί από αυτό για να μειώσει σημαντικά τους απαιτούμενους πόρους υλικού με αντίτιμο μικρή ή και ακόμη μηδενική απώλεια πληροφορίας.

Οι συνελίξεις απαιτούν πολλούς πολλαπλασιασμούς. Εάν αυτοί οι πολλαπλασιασμοί υλοποιούνται με τη χρήση των προεγκατεστημένων DSP μπλοκ μέσα στο FPGA, αυτό περιορίζει δραματικά την πολυπλοκότητα του CNN που μπορεί να υλοποιηθεί. Για παράδειγμα, ένα τυπικό επίπεδο μιας ¨ελαφριάς' αρχιτεκτονικής απαιτεί περίπου 2400 πολλαπλασιαστές. Αυτός ο αριθμός υπερβαίνει σε μεγάλο βαθμό τον αριθμό των μπλοκ DSP που παρέχονται από πολλά FPGA και ιδιαίτερα από τις ενσωματωμένες συσκευές.

Αντιμετωπίζουμε αυτό το πρόβλημα συστηματικά με 3 τρόπους σε επίπεδο design:

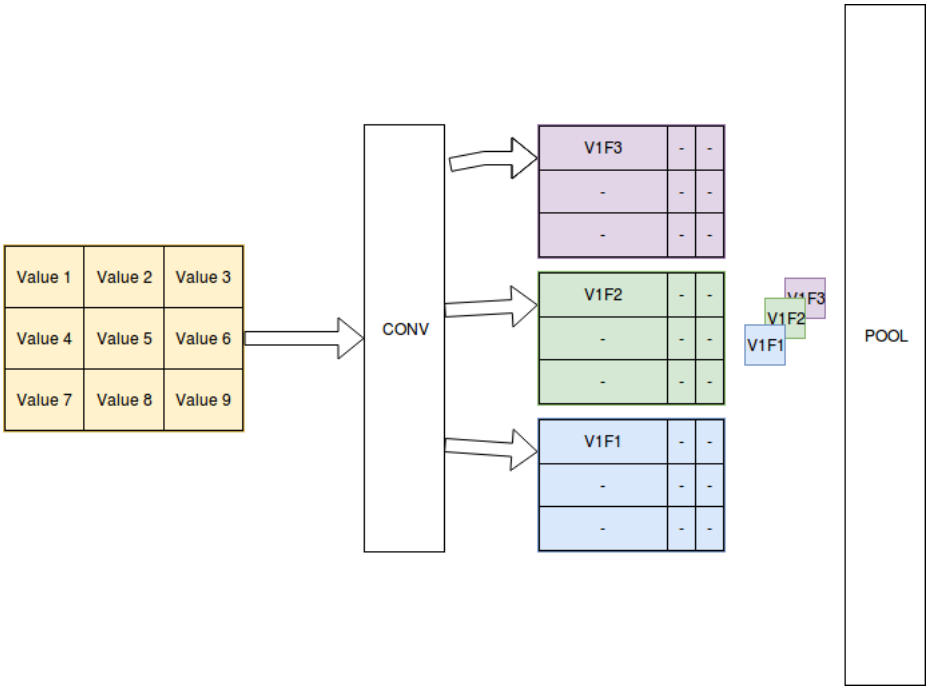
1. Αναγκάζοντας τη σύνθεση (Vivado Synthesis Strategies) στην υλοποίηση πολλαπλασιασμών με λογικά στοιχεία αντί για μπλοκ DSP, βασιζόμενοι έτσι σε πύλες και δέντρα ημι-αθροιστών.

2. Επωφελούμαστε από το γεγονός ότι στην περίπτωση των CNN οι πυρήνες συνέλιξης και επομένως, ο δεύτερος τελεστής των περισσότερων πολλαπλασιασμών είναι στην πραγματικότητα μια σταθερά η οποία λαμβάνεται απο την εκάστοτε ROM. Έτσι πολλές πράξεις μπορούν να γίνουν hardwired μέσα στο design ως αλληλουχία πυλών και LUT.

3. Όπως θα αναφέρθεί και στο επόμενο κεφάλαιο, λόγω συγκεκριμένης προεπεξεργασίας η είσοδος του CNN θα αποτελεί εν τέλει ένα χάρτη pixel του 1 bit με τιμή λευκό (1) ή μαύρο (0). Έτσι το πρώτο συνελικτικό επίπεδο θα μπορεί να απλοποιηθεί σε πύλες AND μεταξύ της εισόδου και της ROM χωρίς την ανάγκη ύπαρξης πολλαπλασιασμών.

3.3.2 Δειγματοληπτικό Επίπεδο (Pooling Layer)

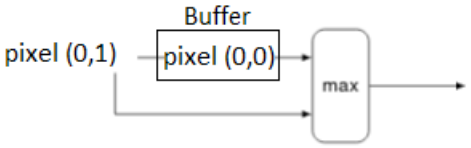
Στη συνέχεια, υλοποιούμε τα δειγματοληπτικά επίπεδα τύπου επιλογής μεγίστου (max pooling layers).Στόχος του επιπέδου είναι η εξαγωγή της μέγιστης τιμής απο ένα παράθυρο εικονοστοιχείων και στη δική μας περίπτωση από ένα παράθυρο 2x2. Ένα δειγματοληπτικό επίπεδο θεωρητικά δέχεται ως είσοδο έναν τρισδιάστατο προεπεξεργασμένο από συνελικτικό επίπεδο πίνακα με τις 2 διαστάσεις του να αποτελούν ένα πίνακα εικονοστοιχείων και την 3η ένα πλήθος - ίσο με τον αριθμό των φίλτρων του προηγούμενου επιπέδου - τέτοιων πινάκων. Στην πραγματικότητα προκειμένου το σύστημα να υπακούει στην αρχή των σωληνώσεων και της διαρκούς λειτουργίας η είσοδος θα αποτελείται από τα παραγόμενα επεξεργασμένα pixel N τη φορά όσα και το πλήθος των φίλτρων που αντιστοιχούν στην ίδια θέση του τελικού πίνακα. Η διαδικασία γίνεται πίο κατανοητή στο ακόλουθο σχήμα:



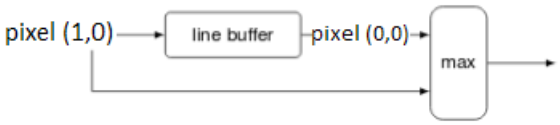
Σχήμα 2.17 Είσοδος Δειγματοληπτικού επιπέδου με pipeline

Λόγω της μορφολογίας εισόδου, το design του δειγματοληπτικού επιπέδου χρησιμοποιεί 2 δομές με όνομα poolH,poolV που αντιστοιχούν στην οριζόντια και κατακόρυφη δειγματοληψία αντιστοίχως. Η δομή poolH χρησιμοποιεί ένα μικρό buffer μεγέθους ίσο με το παράθυρο δειγματοληψίας (2), ώστε να πραγματοποιεί την σύγκριση ανα 2 εισερχομένων σειριακά στοιχείων. Σαν αποτέλεσμα ανα 2 κύκλους ξέρουμε επιτόπου τους μέγιστους αριθμούς της εισερχόμενης γραμμής. Η δομή poolV χρησιμοποιεί ένα buffer μεγέθους ίσο με το μέγεθος της εικόνας (ν), ώστε να πραγματοποιεί την σύγκριση ανα ν εισερχομένων σειριακά στοιχείων. Σαν αποτέλεσμα ανα ν κύκλους ξέρουμε επιτόπου τους μέγιστους αριθμούς της εισερχόμενης στήλης. Ο συνδυασμός των παραπάνω στοιχείων μας δίνει ένα συγκριτή που ανά πάσα στιγμή γνωρίζει μέγιστα στοιχεία ανα 2 οριζοντίως και κατακορύφως. Το μόνο που μένει είναι ένα επιπλέον στοιχείο max_pool_cell που θα απομονώνει τον πραγματικό μέγιστο ανα γειτονιές των 4ων

στοιχείων,ένα παράθυρο 2x2 δηλαδή. Το επίπεδο δειγματοληψίας δεν έχει πέρα απο το να υλοποιήσει τόσα στοιχεία όσα και το βάθος της εισόδου του. Αν λάβουμε για παράδειγμα την εικόνα 2.17 το δειγματοληπτικό επίπεδο θα υλοποιούσε εσωτερικά 3 max_pool_cell όπου το καθένα θα ήταν υπεύθυνο για την υπολοίηση poolV και poolH για τις δικιές του εισόδους.



Σχήμα 2.18 Στοιχείο poolH (Horizontal)

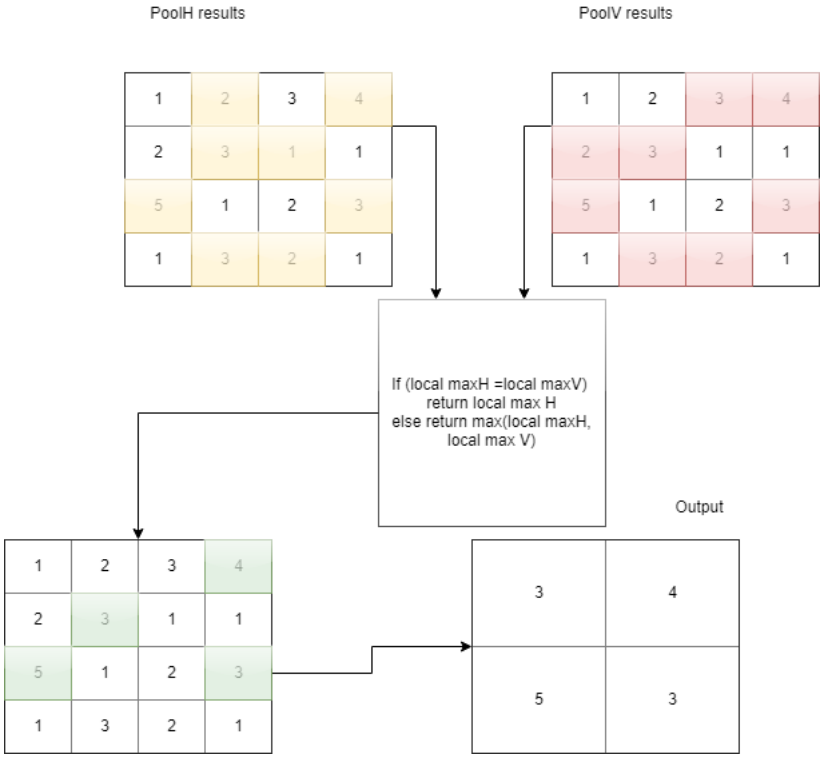


Σχήμα 2.19 Στοιχείο poolV (Vertical)

Στο σχήμα 2.18 και 2.19 φαίνεται χαρακτηριστικά η πορεία εξαγωγής μεγίστου απο μια εικόνα nxn με αρίθμιση απο 0 έως n-1 για γραμμλες και στήλες.

Στην 2.18 το πανω αριστερά στοιχείο (0,0) θα αποθηκευθεί σε ενα καταχωρητή μίας θέσης ώστε να συγκριθεί με το επόμενο (0,1) στοιχείο προωθώντας το τελικό αποτέλεσμα.Στη συνέχεια το (0,1) θα μπεί στον καταχωρητή ώστε να συγκριθεί με τη σειρά του με το επόμενο streaming pixel το (0,2) και ούτω καθεξής.

Αντίστοιχα το πρώτο στοιχείο της 2ης γραμμής (1,0) θα συγκριθεί με το ακριβώς από πάνω του στοιχείο το (0,0). Στη συνέχεια θα μετακινηθεί και αυτό στον καταχωρητή μήκους ίσο με το πλάτος της εικόνας ώστε να συχρονιστεί η σύγκρισή του με το ακριβώς απο κάτω του στοιχείο όταν αυτό πρωτοεμφανιστεί.



Σχήμα 2.20 Κύτταρο εύρεσης τοπικού μεγίστου (max_pool_cell)

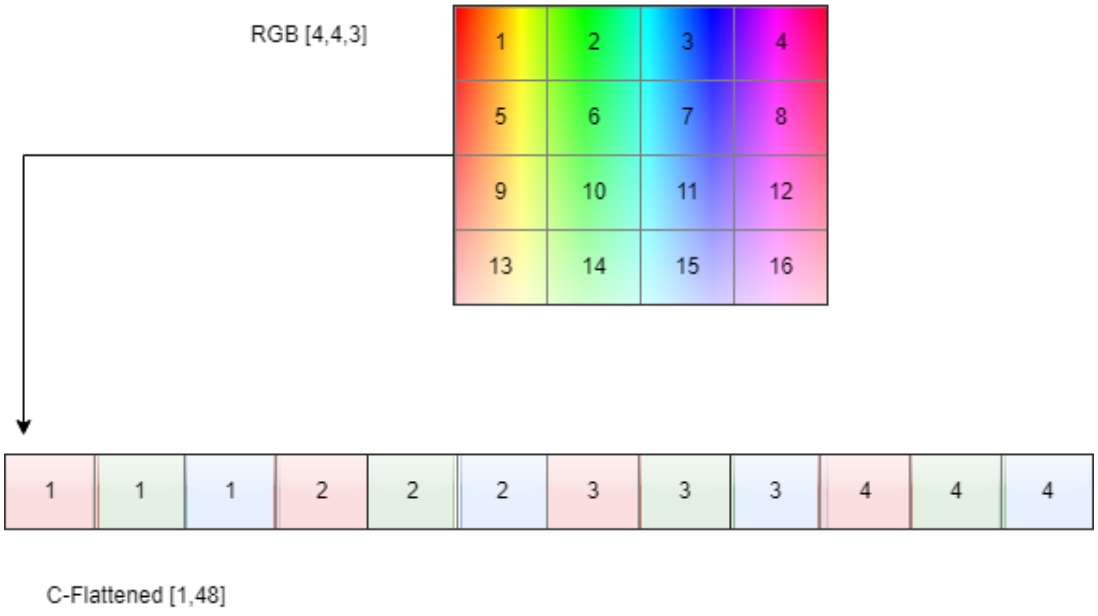
Στην 2.20 παρουσιάζεται η ολοκληρωμένη λειτουργία του κυττάρου εύρεσης τοπικού μεγίστου. Τα αποτελέσματα της poolH αριστερά και της poolV δεξιά για μια εικόνα 4x4 συγκρίνονται ώστε να διαμορφωθούν τα επιμέρους αποτελέσματα μεγίστων γραμμής και στήλης. Αν το μέγιστο γραμμής και στήλης σε μια περιοχή 2x2 είναι μοναδικό τότε επιστρέφεται ως το τοπικό μέγιστο στοιχείο της εκάστοτε περιοχής, αλλιώς οι 2 ενδεχόμενοι ανταγωνιστές συγκρίνονται και επιστρέφει ο μεγαλύτερός τους.

3.3.3 Επιπεδοποίηση και Πλήρως Συνδεδεμένα Επίπεδα

Το τελευταίο στάδιο ενός CNN είναι μιά στοίβα από πλήρως συνδεδεμένα επίπεδα όπου πραγματοποιούν την απλή λειτουργία του πολλαπλασιασμού πινάκων.
Ο πίνακας των αποθηκευμένων συντελεστών είναι μεγέθους (κόμβοι επιπέδου ν x κόμβοι επιπέδου ν+1) ενώ η είσοδος σε κάθε επίπεδο αποτελεί ένα διάνυσμα της μορφής (1 x κόμβοι επιπέδου ν). Από αυτό το σημείο και μετά τα δεδομένα αποτελούν ένα μονοδιάστατο διάνυσμα οποίο θα συνεχίσει να διατηρεί τη μοναδαία διάστασή του καθόλη την πορεία του μέσα απο τους εκάστοτε πολλαπλασιασμούς πινάκων, έως ότου φτάσει να αποτελεί ένα διάνυσμα μήκους τόσων στοιχείων όσων και κατηγοριών των αντικειμένων που θέλουμε να αναγνωρίσουμε.

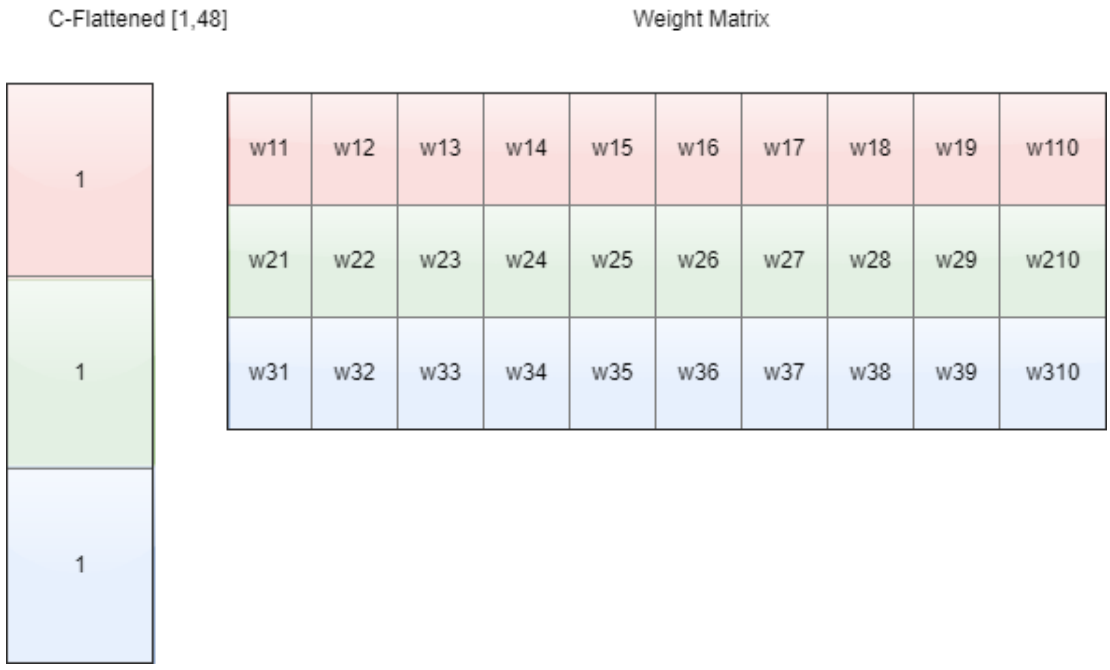
Για το λόγο αυτό τα δεδομένα ύστερα απο το τελευταίο συνελικτικό επίπεδο υφίστανται μια διαδικασία επιπεδοποίησης flattening που μετατρέπει την τρισδιάστατη δομή τους ([h,w,f]) σε μονοδιάστατη ([1,h x w x f], όπου h είναι η διάσταση μήκους ύστερα απο επεξεργασία, w η διάσταση πλάτους ύστερα απο επεξεργασία και f ο αριθμός φίλτρων τελευταίου επιπέδου. Η διαδικασία αυτή ακολουθεί το λεγόμενο C - Ordering δηλαδή ξεκινά να τοποθετεί από την τελευταία διάσταση προς την πρώτη.

Για μιά εικόνα RGB [4,4,3] η επιπεδοποιημένη έκδοσή της θα αποτελεί ένα διάνυσμα με στοιχεία από το πάνω αριστερά pixel από το κάθε κανάλι χρώματος μέχρι το τελευταίο δεξιά pixel για κάθε κανάλι χρώματος.Το παράδειγμα γίνεται κατανοητό στην εικόνα 2.21



Σχήμα 2.21 Επιπεδοποίηση (C-ordering flattening)

Ας μείνουμε λίγο στο παράδειγμα της 2.21 και ας παρατηρήσουμε την είσοδο αυτού του διανύσματος σε ένα τυχαίο πλήρως συνδεδεμένο επίπεδο με 10 κόμβους εξόδου. Ο πίνακας συντελεστών θα έχει διαστάσεις [48,10] ώστε [1,48] x [48,10] = [1,10].



Σχήμα 2.22 Διάταξη Πολλαπλασιασμών

Στο παράδειγμα της εικόνας 2.22 φαίνονατι τα 3 πρώτα στοιχεία του διανύσματος εισόδου και οι αντίστοιχοι συντελεστές του πίνακα. Σημαντική παρατήρηση είναι ότι το διάνυσμα εισόδου πολλαπλασιάζεται στοιχείο προς στοιχείο με κάθε στήλη του πίνακα. Αυτό μας δίνει 2 κανόνες - εργαλεία για τυχόν παραμετροποιήσεις της παραπάνω διαδικασίας:

1. Κάθε αντιμετάθεση 2 στοιχείων στον πίνακα εισόδου αντιστοιχεί στην αντιμετάθεση των 2 αντίστοιχων γραμμών στον πίνακα των συντελεστών.

2. Στον πολλαπλασιασμό πινάκων το αποτέλεσμα του καθενός εκ των στοιχείων του πίνακα εξόδου αποτελείται απο το άθροισμα των μερικών αποτελεσμάτων των πράξεων γραμμής εισόδου επί όλων των στηλών. Άρα η αντιμετάθεση στηλών στον πίνακα των συντελεστών δεν θα επιφέρει καμία αλλαγή στο τελικό αποτέλεσμα.

Βασιζόμενοι πάνω στην πρώτη παρατήρηση προκειμένου να αποφύγουμε ένα επιπλέον στοιχείο στο design μας που να πραγματοποιεί C - Ordering επιλέξαμε να λαμβάνουμε την είσοδο στα πλήρως συνδεδεμένα επίπεδα ως μια αλληλουχία χρωματικών καναλιών. Δηλαδή ο πίνακας εισόδου μας αποτελείται από όλα τα pixel που αντιστοιχούν στο πρώτο κανάλι -εδώ κόκκινο ακολουθούμενα με τον ίδιο τρόπο απο το πράσινο και το μπλέ κανάλι αντίστοιχα, πράξη μηδενικού κόστους μιάς και έτσι αποθηκεύονται εξαρχής στα προηγούμενα επίπεδα.

Αξίζει να παρατηρήσει κανείς ωστόσο ότι σαν πράξη ο πολλαπλασιασμός πινάκων δεν υπακούει καθόλου στην αρχή της συνεχούς λειτουργίας και στο σχεδιασμό με σωληνώσεις. Αυτο συμβαίνει διότι σαν μαθηματική πράξη απαιτεί παρόντες και τους 2 τελεσταίους, χρειάζεται δηλαδή να συγκεντρώσει όλα τα στοιχεία του διανύσματος εισόδου -τα οποία έρχονται σαν

αλλήλουχία προκειμένου να υλοποιηθεί. Παρόλα αυτά τα αποτελέσματα των προηγούμενων επιπέδων έρχονται ανα χτύπο ρολογιού. Αυτό σημαίνει ότι επειδή η πράξη των πολλαπλασιασμών και προσθέσεων για την παραγωγή του τελικού αποτελέσματος διαρκεί τουλάχιστον 1 κύκλο ρολογιού θα έπρεπε να υπάρχει και ένας in-line buffer που να αποθηκεύει μερικώς την είσοδο έως ότου το σύστημα πολλαπλασιαστών και αθροιστών απελευθερωθεί.

Αυτό θα παρεμπόδιζε δραματικά τόσο την ταχύτητα όσο καί την κλιμάκωση του design ενώ θα απαιτούσε την εισαγωγή έξτρα καταχωρητών ως buffers ανά επίπεδο, καθιστώντας το design απαγορευτικό για απαιτητικά πολυεπίπεδα δίκτυα .

Εδώ θα εκμεταλευτούμε την 2η ιδιότητα μας, πως δηλαδή μπορούμε να πραγματοποιήσουμε αντιμετάθεση οποιονδήποτε στηλών του πίνακα χωρίς να μεταβάλουμε το τελικό αποτέλεσμα μας.

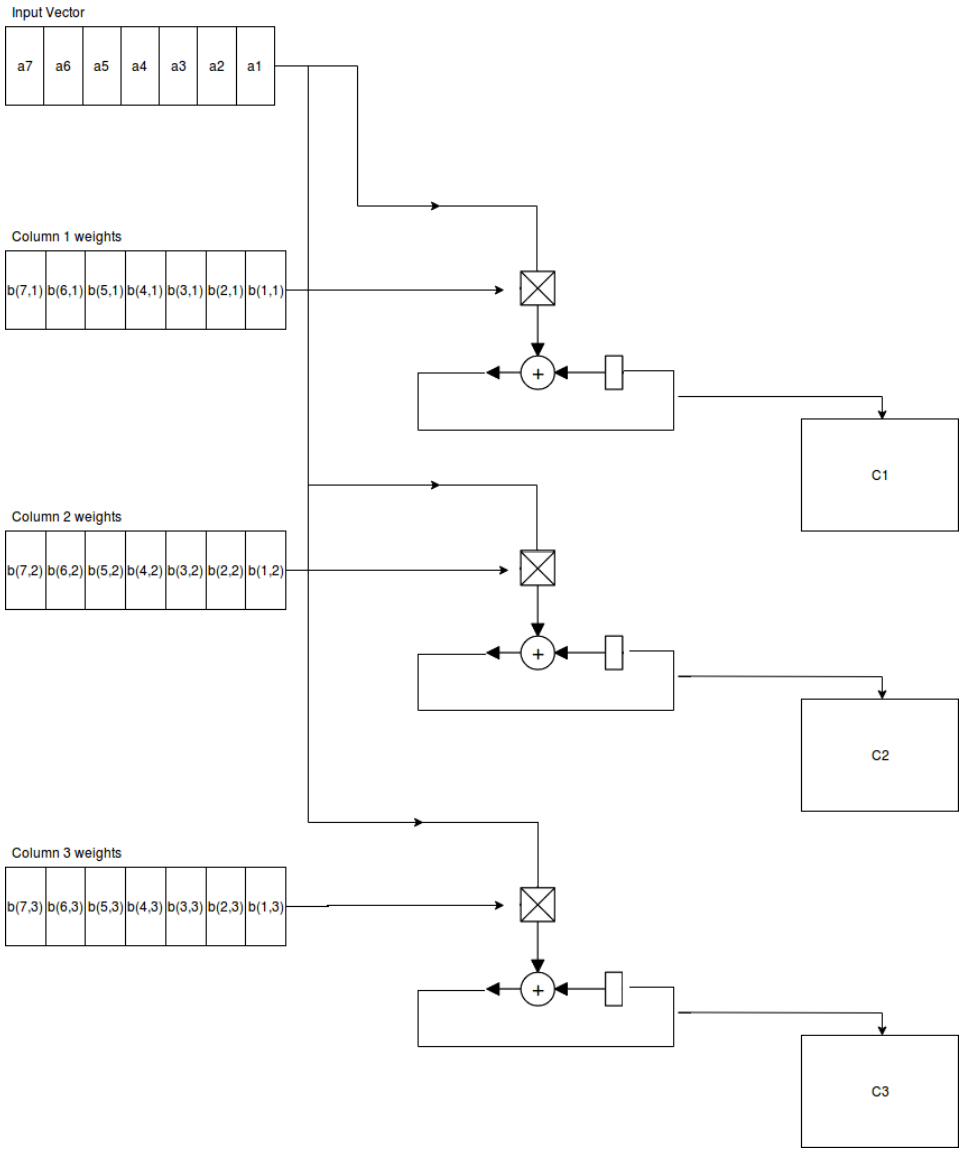
Έτσι μπορούμε να ξαναδούμε την πράξη του πολλαπλασιασμού διανύσματος επι πίνακα ως μια παράλληλη διαδικασία αθροίσματος πολλαπλασιασμών στοιχείου με στοιχείο. Εν ολίγοις ο πολλαπλασιασμός του διανύσματος με την 1η στήλη του πίνακα έχει στόχο-αποτέλεσμα το 1ο στοιχείο του διανύσματος εξόδου και μπορεί να υπολογιστεί ανεξάρτητα με την ύπαρξη 2ης,3ης,ν-οστής στήλης. Επιπλέον στον πολλαπλασιασμό διανύσματος επι στήλη υπάρχει μια 1 προς 1 σχέση των στοιχείων που θα πολλαπλασιαστούν συνεπώς μπορεί να θεωρηθέι σαν ένα επιπλέον τμήμα πολλαπλασιασμού - συσσώρευσης, δηλαδή ένα φίλτρο σαν αυτα που χρησιμοποιήσαμε και προηγουμένως.

Συγκεκριμένα η πράξη του πολλαπλασιασμού 2 πινάκων A(1 x m),B(m x p) δίνεται απο τη σχέση:

$$c_j = \sum_{k=1}^{m} a_k b_{kj}$$

Όπου C(1 x p) ο πίνακας εξόδου.

Σύμφωνα με τη δική μας οπτική, η πράξη μπορεί να γραφτεί σαν p παράλληλα φίλτρα με κοινή είσοδο και συντελεστές τους συντελεστές του πίνακα B.



Σχήμα 2.23 Πολλαπλασιασμός πίνακα με τη μορφή φίλτρου

Στο σχήμα 2.23 φαίνεται χαρακτηριστικά ο πολλαπλασιασμός ενός πίνακα διανύσματος 7 στοιχείων με ένα πίνακα 7x3 ως 3 στήλες διανύσματα 7 στοιχείων για την παραγωγή του διανύσματος εξόδου 1x3.

Συνεπώς με βάση αυτές τις παραδοχές-μετασχηματισμούς, εξασφαλίζουμε τη διαρκή λειτουργία με σωληνώσεις. Ο πολλαπλασιασμός πινάκων πραγματοποιείται μερικώς κατά τη διάρκεια έλευσης των pixel στο τελευταίο επίπεδο χωρίς αναμονή, διατηρώντας παράλληλα την ορθότητα των πράξεων και προωθώντας τα πλήρη αποτελέσματα μαζί με την έλευση και του τελευταίου απαιτούμενου pixel εισόδου. Να σημειωθεί πως η πρακτική αυτή μπορεί να χρησιμοποιηθεί και για περισσότερα του ενός πλήρως συνδεδεμένα επίπεδα με μικρές παραλλαγές για την επιτάχυνση και μη συνελικτικών νευρωνικών δικτύων.

3.3.4 Τελικό Επίπεδο

Καθόλη την διάρκεια ροής πληροφορίας μέσα από τα προαναφερθέντα επίπεδα διατηρούνται τα σήματα ελέγχου έγκυρου πίξελ εισόδου και έγκυρης εικόνας έισόδου τα οποία μεταφέρονται απο επίπεδο σε επίπεδο και υφίστανται χειρισμούς ανάλογα με την ορθότητα των παραγόμενων αποτελεσμάτων καθώς και τη συνέπεια της διαδικασίας.

Το τελικό επίπεδο που περιλαμβάνει απλά ένα καταχωρητή 10 θέσεων (όσες και οι απαιτούμενες κατηγορίες στις οποίες θα μαντέψει το νευρωνικό) καθώς και ένα ελεγκτή σήματος. Στους καταχωρητές συσσωρεύονται από τον πολλαπλασιασμό πινάκων τα μερικά αποτελέσματα, ενώ η εξαγωγή του πιθανότερου υποψηφίου για αναγνώρηση αποτελεί ο καταχωρητής με το μεγαλύτερο περιεχόμενο.

Σαν συνέπεια όταν ο ελεγκτής μετρήσει το σωστό αριθμό βημάτων και επιπλέον τα 2 σήματα ελέγχου είναι στο λογικό 1, θα δώσει ως έξοδο τον αριθμό του καταχωρητή με το μεγαλύτερο μέχρι στιγμής αποτέλεσμα, ενώ θα μηδενίσει τον καταχωρητή στον επόμενο κύκλο για την εκ νέου συσσώρευση αποτελεσμάτων για την επόμενη εικόνα - είσοδο.

Κεφάλαιο 4

Υπερπαραμετροποίηση-Βελτιστοποιήσεις

Παρά τα σημαντικά αποτελέσματα και βελτιώσεις που πετύχαμε με βάση τις προαναφερθήσες μεθοδολογίες απεικόνισης, σημαντικό τμήμα στην αύξηση της απόδοσης ενός design και γενικότερα ενός σχεδιασμού επίλυσης κάποιου προβλήματος αποτελεί η σε βάθος κατανόησή του.

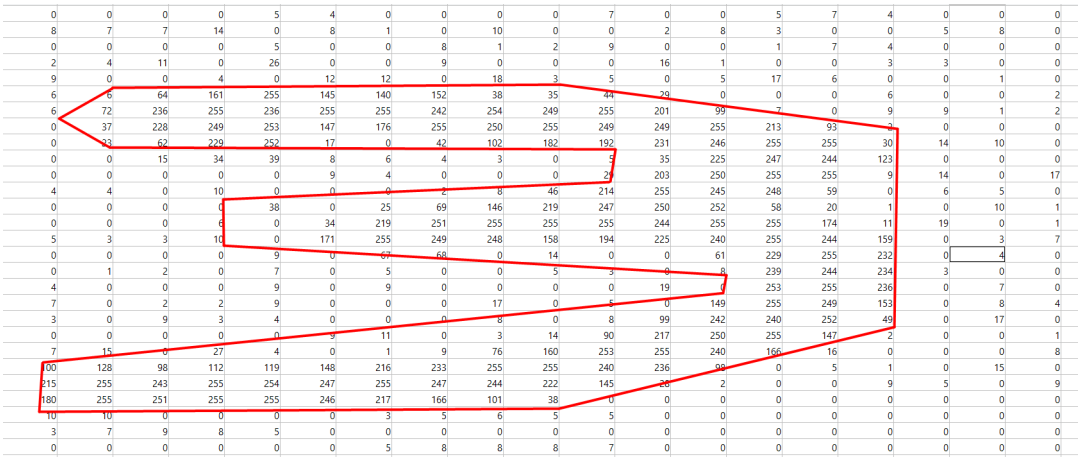
Το πρόβλημα που καλείται το design μας να αντιμετωπίσει αποτελεί το πρόβλημα της αναγνώρισης ψηφίων MNIST με είσοδο ασπρόμαυρες εικόνες 28x28 pixel και έξοδο 10 κλάσεις από 0 έως 9. Η επιλογή του συγκεκριμένου προβλήματος έγινε λόγω των πολλών μετρήσεων και αποτελεσμάτων που προυπάρχουν απο άλλες αρχιτεκτονικές, λόγω της γρήγορης εκπαίδευσής του καθώς και λόγω της εύκολης επιβεβαίωσης αποτελεσμάτων με την τροφοδοσία χειρόγραφων εικόνων από το χρήστη.

Χαρακτηριστικά αναφέρουμε ότι η μεγαλύτερη καταγεγραμμένη ακρίβεια ανέρχεται στα 99,6% χρησιμοποιώντας το επίσημο dataset εικόνων στα 64bit με τη χρήση floating point αριθμών.

4.1 Απλοποίηση εισόδου

Οι εικόνες - είσοδοι έχουν μαύρο φόντο ενώ το ψηφίο προς αναγνώρηση είναι γραμμένο με λευκό χρώμα. Για καλύτερη επίβλεψη των διακυμάνσεων των RGB τιμών υλοποιήθηκε ένα πρόγραμμα σε Python ικανό να καταγράψει για κάθε pixel την τιμή του και να παρουσιάσει τα αποτελέσματα ως πίνακα 28x28 σε ένα αρχείο .txt.

Σχήμα 4.1 Εικόνα εισόδου με το ψηφίο 4



Σχήμα 4.2 RGB τιμές εικόνας

Όπως φαίνεται και στην εικόνα 4.2 οι περιοχές ενδιαφέροντος λαμβάνουν τιμές μεγαλυτερες του 125 στην περίπτωση σκιάς και μεγαλύτερης του 200 στην περίπτωση σίγουρης γραμμής. Σημεία εξωτερικά της περιοχής ενδιαφέροντος λαμβάνουν τιμές κοντά στο 0, ενώ η μεγάλη αυτή διαφοροποίηση συντελεί στην έντονη διαγραφή του περιγράμματος του ψηφίου της πάνω εικόνας.

Σαν μια πρώτη ιδέα βελτίωσης του σχεδιασμού πραγματοποιήθηκε μια διαδικασία χαρτογράφησης των εικόνων εισόδου με τον εξής αλγόριθμο:

Για κάθε εικονοστοιχείο στην εικόνα, εάν η τιμή του είναι μεγαλύτερη απο 160 τότε λαμβάνει την τιμή 1 αλλιώς την τιμή 0. Καταλήγουμε έτσι σε μια αναπαράσταση της εικόνας σε κλίμακα 0-1 μετατρέποντάς τη σε ένα χάρτη 28x28 bit. Για λόγους συνέπειας παραθέτουμε 2 εικόνες πρίν και μετά την επεξεργασία προκειμένου να επιβεβαιωθεί η σχεδόν μηδενική απώλεια πληροφορίας:

Σχήμα 4.3 Παράθεση αρχικής εικόνας(αριστερά) και επεξεργασμένης(δεξιά)

Η προεπεξεργασία αυτή, εκτός από ότι μειώνει τις απαιτήσεις μνήμης εισόδου κατά 99,65% μας επιτρέπει να υλοποιήσουμε το πρώτο συνελικτικό επίπεδο με πύλες AND αντί για πολλαπλασιαστές, όπως αναφέρθηκε στο προηγούμενο κεφάλαιο, μειώνοντας ακόμα περισσότερο την κατανάλωση,την πολυπλοκότητα και το εύρος bit του design.

4.2 Κβάντιση και Νευρωνικά Δίκτυα

Τα περισσότερα σύγχρονα συνελικτικά νευρωνικά δίκτυα δεν είναι κατάλληλα για χρήση σε συσκευές περιορισμένου υλικού όπως FPGA,ASIC και ακόμη κινητές συσκευές.Αυτό σημβαίνει διότι κατα την ανάπτυξή τους, πρωτίστως αξιολογήθηκαν σύμφωνα με την ακρίβεια της ανίχνευσης του επιθυμιτού αποτελέσματος. Έτσι, οι αρχιτεκτονικές δικτύων έχουν εξελιχθεί αδιαφορώντας για τη σχέση πολυπλοκότητας μοντέλων και υπολογιστικής αποδοτικότητας. Αυτό έχει πρόσφατα οδηγήσει σε ένα πεδίο ανάπτυξης που επικεντρώνεται στη μείωση του μεγέθους του μοντέλου και του συμπερασμού χρόνου των CNN με ελάχιστες απώλειες ακριβείας.

Παρά την αφθονία των μεθόδων κβάντισης, μια αρκετά αποδοτική και υλοποιήσιμη στο προγραμματιστικό περιβάλλον μας (Tensorflow 1.10) είναι η τεχνική της κβάντισης σε 8-bit. Υπάρχουν διάφοροι λόγοι που δίνουν αξία σε αυτήν την τεχνική τροποποίηση:

1. Η αριθμητική με μικρότερο βάθος bit είναι ταχύτερη ενώ απαιτεί μικρότερους και ευκολότερα υλοποιήσιμους αθροιστές και πολλαπλασιαστές. Οι σύχρονες τάσεις των λειτουργιών με κινητή υποδιαστολή στα 32 bit θα είναι πάντα πιο αργές, πιο ογκώδεις και υψηλότερης κατανάλωσης απο αυτές των ακεραίων 8 bit.

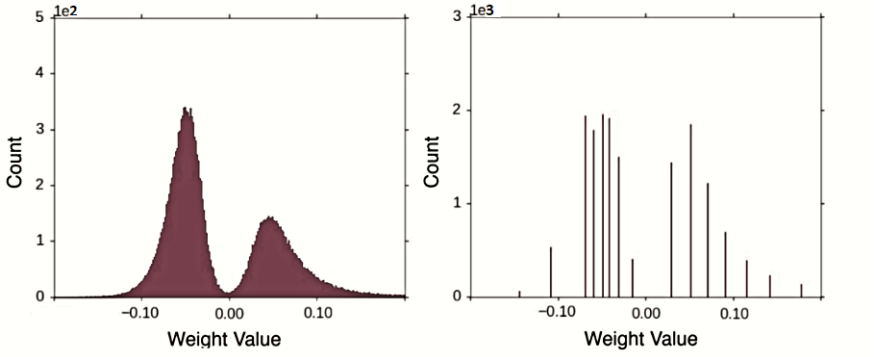
2. Κατά τη μετάβαση από 32-bit σε 8-bit, παίρνουμε 4x μείωση στην on-chip μνήμη. Τα μοντέλα ελαφρύτερου τύπου έχουν μικρότερο αποθηκευτικό χώρο και διευκολύνουν την κοινή χρήση τους σε μικρότερα εύρη ζώνης.

3. Η αριθμητική με κινητή υποδιαστολή μπορεί να μην υποστηρίζεται πάντα από μικροελεγκτές σε ορισμένες ενσωματωμένες συσκευές εξαιρετικά χαμηλής ισχύος, όπως τα image recognition drones, τα ρολόγια ή οι συσκευές IoT,καθιστώντας ένα επιταχυντή FPGA δυσλειτουργικό σε περίπτωση που πρόκειται να τυποποιηθεί σε κάποιο μόνιμο design.

Γιατί ωστόσο μια τόσο δραματική τροποποίηση λειτουργεί· Υπάρχουν σε γενικές γραμμές, δύο λόγοι για αυτό. Πρώτον, τα νευρωνικά δίκτυα είναι γνωστό ότι είναι αρκετά ανθεκτικά στον θόρυβο και σε άλλες μικρές διαταραχές μόλις εκπαιδεύονται. Αξίζει να σημειωθεί εδώ ότι μια τέτόια περίπτωση noising-denoising αποτελεί και η πρώτη τεχνική χαρτογράφισης της εικόνας εισόδου σε χάρτη 1-bit. Αυτό σημαίνει ότι ακόμη και αν υποδιαιρούμε αριθμούς, μπορούμε ακόμα να περιμένουμε μια λογικά ακριβή απάντηση.

Επιπλέον, τα βάρη και οι ενεργοποιήσεις από ένα συγκεκριμένο στρώμα συχνά τείνουν να βρίσκονται σε μικρή κλίμακα, η οποία μπορεί να εκτιμηθεί εκ των προτέρων. Αυτό σημαίνει ότι δεν χρειαζόμαστε την ικανότητα να αποθηκεύουμε αριθμούς της τάξης του 10^6 και 10^{-6}στον ίδιο τύπο δεδομένων.

Αυτό μας επιτρέπει να περιορίσουμε τα βάρη σε μικρότερο εύρος, π.χ. -3 έως +3. Έτσι, αν γίνει σωστά, η κβάντιση προκαλεί μόνο μια μικρή απώλεια ακρίβειας, η οποία συνήθως δεν αλλάζει σημαντικά την απόδοση. Τέλος, μπορούν να ανακτηθούν μικρές απώλειες ακρίβειας με την επανεκπαίδευση των μοντέλων μας ώστε να προσαρμοστούν στην κβάντιση.



Σχήμα 4.4 Κατανομή Βαρών με και χωρίς κβάντιση

Στο σχήμα 4.4 παρατηρούμε την κατανομή των βαρών στο δίκτυο μας, με τις περισσότερες τιμές να συγκεντρώνονται σε ένα μικρό εύρος. Μπορούμε να κβαντοποιήσουμε το εύρος και να καταγράψουμε με ακρίβεια τις αλλαγμένες τιμές. Το δεξιά υπο-γράφημα δείχνει μία τέτοια κβαντοποίηση χρησιμοποιώντας 8-δυαδικά ψηφία (255 διακριτές τιμές).

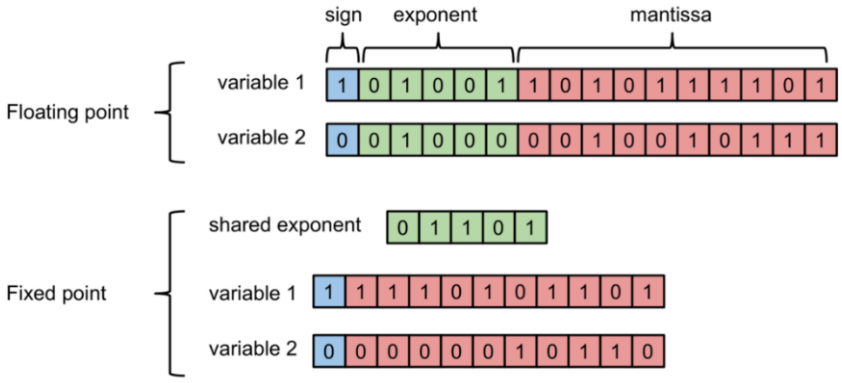
Γιατί ωστόσο να μην προπονούνται τα νευρωνικά δίκτυα άμεσα με χαμηλότερη ακρίβεια, αντί να καταφεύγουμε σε επεξεργασία των βαρών μετά την προπόνιση· Αν και δεν είναι αδύνατο, δεν έχουμε ακόμα επαρκώς καλές μεθόδους υλοποίησης κβάντισης on-line. Αυτό οφείλεται στην ίδια τη φύση του τρόπου εκμάθησης των μοντέλων που εκπαιδεύονται με πολύ μικρές ενημερώσεις κλίσης, για τις οποίες χρειαζόμαστε μεγάλη ακρίβεια.

4.3 Η Κβάντιση 8bit uint

Προκειμένου να κατανοήσουμε τη μεθοδολογία της κβάντισης 8bit απρόσημου δεκαδικού πρέπει να μιλήσουμε αρχικά για την κβάντιση από αριθμούς κινητής υποδιαστολής σε αριθμούς σταθερής ακρίβειας.

Οι αριθμοί κινητής υποδιαστολής χρησιμοποιούν μια μαντίσσα και έναν εκθέτη για να αντιπροσωπεύσουν πραγματικές τιμές. Ο εκθέτης επιτρέπει τον υπολογισμό ενός ευρέος φάσματος αριθμών, και η μαντίσσα δίνει την ακρίβεια. Το δεκαδικό σημείο μπορεί να επιπλέει (float), δηλαδή να εμφανίζεται οπουδήποτε σε σχέση με τα ψηφία.

Εάν αντικαταστήσουμε τον εκθέτη με σταθερό παράγοντα κλιμάκωσης, μπορούμε να χρησιμοποιήσουμε ακέραιους αριθμούς για να αναπαριστούμε την τιμή ενός αριθμού σε σχέση με (ήτοι ένα ακέραιο πολλαπλάσιο) αυτής της σταθεράς. Η θέση του δεκαδικού σημείου είναι τώρα ¨σταθερή' από τον συντελεστή κλιμάκωσης.Ένα παράδειγμα αποτελεί αυτό της γραμμής αριθμών,όπου η τιμή του συντελεστή κλιμάκωσης προσδιορίζει τη μικρότερη απόσταση μεταξύ των 2 σημείων στη γραμμή και ο αριθμός των σημείων αυτών καθορίζεται από τον αριθμό των βιτς που χρησιμοποιούμε για να αντιπροσωπεύσουμε τον ακέραιο αριθμό. Μπορούμε να χρησιμοποιήσουμε διαφορετικές κατανομές bit για να ανταλλάξουμε μεταξύ εμβέλειας και ακρίβειας. Οποιαδήποτε τιμή δεν είναι ακριβές πολλαπλάσιο της σταθεράς στρογγυλοποιείται στο πλησιέστερο σημείο.



Σχήμα 4.5 Αριθμοί κινητής υποδιαστολής(πάνω) και σταθερής ακρίβειας(κάτω)

Σε αντίθεση με την κινητή υποδιαστολή, δεν υπάρχει καθολικό πρότυπο για τους αριθμούς σταθερής ακρίβειας και εξαρτάται από την εφαρμογή. Το σχέδιό μας (χαρτογράφηση μεταξύ πραγματικών και κβαντοποιημένων αριθμών) απαιτεί τα εξής:

1. Θα πρέπει να είναι γραμμική η αφινική σχέση. Αν δεν είναι, τότε το αποτέλεσμα των υπολογισμών σταθερού σημείου δεν θα συνιστά απευθείας απεικόνιση πραγματικού αριθμού.

2. Μας επιτρέπει να εκφράζουμε πάντα το 0.f με ακρίβεια. Αν κβαντοποιήσουμε και επαναφέρουμε οποιαδήποτε πραγματική τιμή, μόνο 2^B (B ψηφία κβάντισης), από αυτές θα επιστρέψουν τον ακριβή ίδιο αριθμό, ενώ οι άλλες θα υποστούν κάποια απώλεια ακρίβειας. Εάν διασφαλίσουμε ότι το 0.f είναι μία από αυτές τις τιμές, αποδεικνύεται ότι τα CNN μπορούν να κβαντιστούν με μεγαλύτερη ακρίβεια. Αυτό βελτιώνει την ακρίβεια επειδή το 0 έχει ιδιαίτερη σημασία στα CNN (όπως padding). Εκτός αυτού, έχοντας το 0.f να δείχνει σε άλλη τιμή που είναι υψηλότερη / χαμηλότερη από μηδέν θα εισαγάγει μια προκατάληψη-bias στο σχήμα κβαντισμού.

Επομένως, το σχήμα κβαντισμού μας θα είναι απλώς μια μετατόπιση και κλιμάκωση της γραμμής πραγματικού αριθμού σε μια κβαντισμένη γραμμή αριθμών. Για μια δεδομένη ομάδα πραγματικών τιμών, θέλουμε οι ελάχιστες / μέγιστες πραγματικές τιμές σε αυτό το εύρος [rmin, rmax] να αντιστοιχούν στις ελάχιστες / μέγιστες ακέραιες τιμές [0,2^{B-1}] αντίστοιχα. Αυτό μας δίνει μια αρκετά απλή γραμμική εξίσωση:

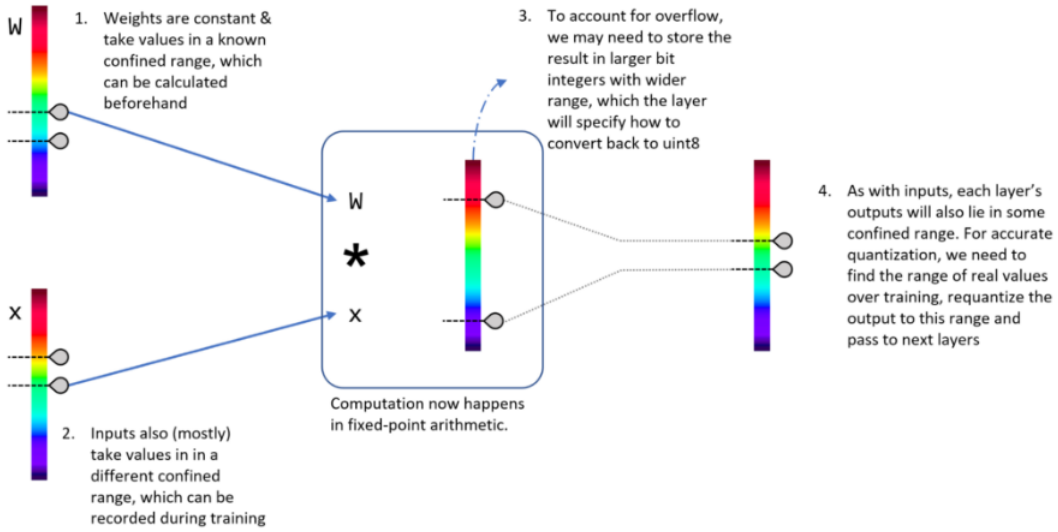
$$r = \frac{r_{max} - r_{min}}{2^{B-1}} \times (q - z) = S \times (q - z)$$

Όπου:

1. r είναι η πραγματική τιμή (συνήθως float32)

2. q είναι η κβαντισμένη αναπαράστασή του ως ακέραιος B-bit (uint8, uint32, κλπ)

3. S (float32) και z (uint) είναι οι παράγοντες με τους οποίους έχουμε την κλίμακα και τη μετατόπιση της γραμμής αριθμών. Το z είναι το κβαντισμένο «μηδενικό σημείο» το οποίο θα αντιστοιχεί πάντα ακριβώς προς το 0.f.



Σχήμα 4.6 Διαδικασία κβάντισης με ψευδουπολογιστικούς κόμβους για παραγωγή 8 bit integer

Στην εικόνα 4.6 φαίνεται η διαδικασία κβάντισης με ψευδουπολογιστικούς κόμβους,όπου τα βάρη και οι σταθερές λαμβάνουν τιμές σε ένα γνωστό εκ των προτέρων εύρος τιμών(πάνω αριστερά), ενώ το ίδιο ακριβώς συμβαίνει και για τις είσοδους. Οι υπολογισμοί πραγματοποιούνται ως πράξεις μεταξύ αριθμών σταθερής ακρίβειας(κέντρο) ενώ αυξάνουμε το πιθανό εύρος των bit αποτελέσματος από το επιθυμητό προσωρινά ώστε να προλάβουμε ενδεχόμενα υπερχείλησης. Τέλος, με γνωστό το εύρος των αποτελεσμάτων, κλιμακώνουμε και διορθώνουμε τις τιμές σε ένα προκαθορισμένο διάστημα κβάντισης το οποίο και προωθούμε ως αποτέλεσμα.

Το design μας συνεπώς υλοποιήθηκε με κβαντισμένα βάρη και σταθερές τύπου 8-bit u-int, δηλαδή θετικούς ακεραίους με τιμές από το 0-255. Πέρα από τους προαναφερθέντες λόγους περι εξοικονόμισης χώρου,κατανάλωσης και αύξησης της ταχύτητας, η επιλογή αυτή μας επέτρεψε να παραλείωουμε σε όλα τα συνελικτικά επίπεδα την συνάρτηση ενεργοποίησης ReLu. Αυτό συμβαίνει διότι η ReLu αφήνει θετικές τιμές να περάσουν ενώ κόβει τις αρνητικές τιμές. Στην δική μας περίπτωση, τόσο η είσοδος όσο και οι συντελεστές των επιμέρους πράξεων είναι θετικοί, συνεπώς δεν υπάρχει λόγος να συμπεριληφθεί η συνάρτηση ενεργοποίησης μειώνοντας έτσι ακόμη περισσότερο την πολυπλοκότητα του design.

4.4 Ευελιξία σχεδιασμού

Ως τελικό στάδιο της παραμετροποίησης του design μας, προσπαθήσαμε να απαντήσουμε σε 3 σημαντικά ερωτήματα:

1. Έιναι εύκολα τροποποιήσιμο ; *(Reconfigurability)*

2. Μπορεί να μεταφερθεί σε διαφορετικές εφαρμογές ; *(Transferability)*

3. Ποιά είναι η ικανότητα σχεδιαστικής κλιμάκωσής του ; *(Scalability)*

Για το σκοπό αυτό η δομή του design μας χωρίστηκε στον κύριο κώδικα VHDL και στις επιμέρους βιβλιοθήκες και αρχεία παραμέτρων (configuration files). Οι υλοποιήσεις των διαφόρων επιπέδων (συνελικτικά, πλήρως συνδεδεμενα, δειγματοληπτικά) έγιναν σε επίπεδο βιβλιοθήκης όπως επίσης και οι δομές που χρησιμοποιήθηκαν στα διάφορα τμήματα του design. Τα βάρη-συντελεστές των διαφόρων επιπέδων γράφονταν σε αρχεία txt τα οποία εισάγονταν στο τελικό project.

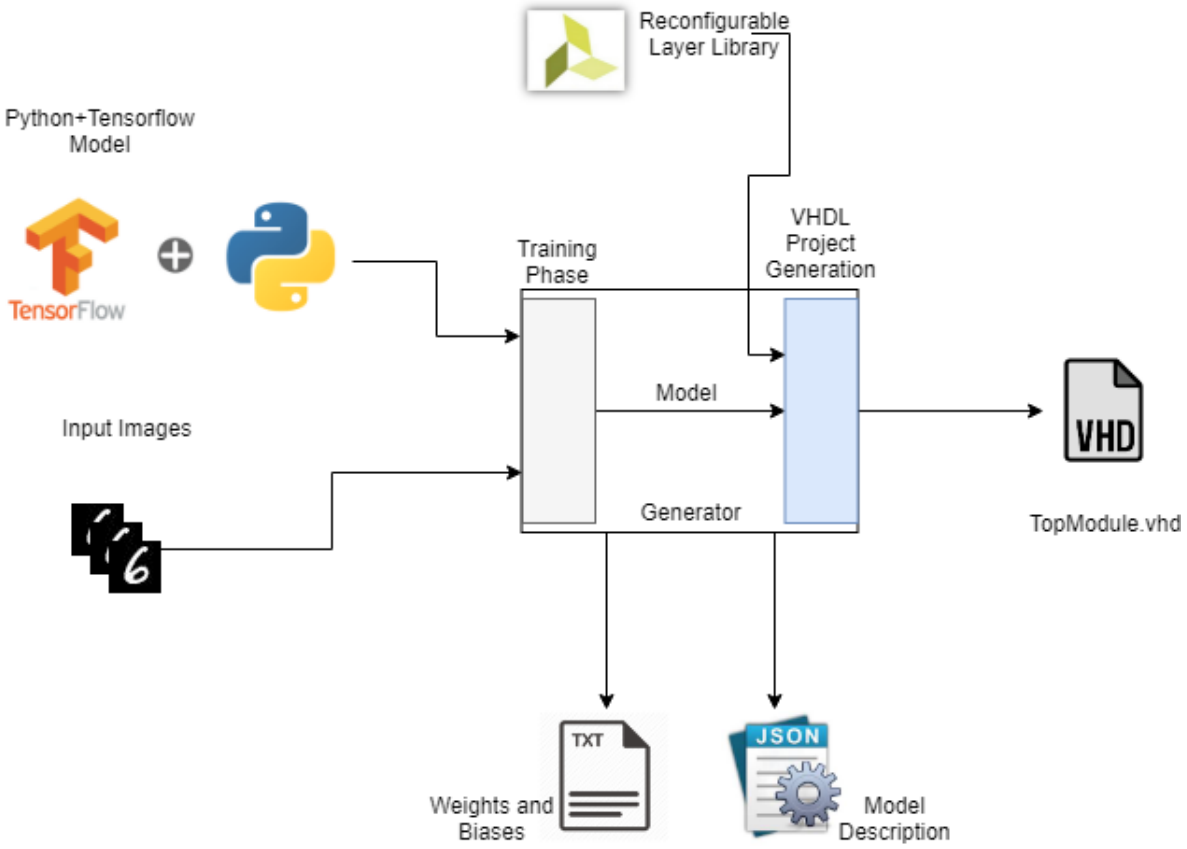
Προκειμένου ωστόσο να αυτοματοποιηθεί η παραπάνω διαδικασία και να απαντήσουμε έτσι στο πρώτο μας ερώτημα περι Reconfigurability δημιουργήθηκε σε Python μια μηχανή αυτόματης κατασκευής και σύνθεσης (Automatic Project Generator) project σε Vivado VHDL design. Ο ¨γεννήτορας' αυτός λαμβάνει ως εισόδους:

1. Τον κώδικα σε Python-Tensorflow που περιγράφει το μοντέλο και τις οδηγίες προς την προπόνησή του.

2. Ένα όρισμα προκειμένου να εκτελέσει προεπεξεργασία χαρτογράφισης 1-bit στην εικόνα εισόδου.

3. Ένα όρισμα προκειμένου να εκτελέσει κβάντιση η όχι στα υπολογισμένα βάρη.

Στη συνέχεια αφού προπονήσει το νευρωνικό στην GPU αποθηκεύει την δομή του καθώς και τα βάρη του σε ένα αρχείο json και ένα txt αντίστοιχα, τα οποία αφήνει προς παρατήριση στον χρήστη. Τέλος με βάση ένα απλό template σε VHDL δημιουργεί το αρχείο top-module του Vivado Project. Ανάλογα με το τι διάβασε από το json θα χρησιμοποιήσει από τις βιβλιοθήκες μας τα διάφορα επίπεδα που απαιτούνται ως entities και θα τα στοιβάξει το ένα μετά το άλλο, ενώ θα κάνει hardcode τα βάρη από το txt σαν μία μνήμη ROM ενσωματωμένη στο design.



Σχήμα 4.7 Λειτουργία του γεννήτορα Project Generator

Προκειμένου να απαντήσουμε τώρα στο δεύτερο ερώτημα περί μεταφερσιμότητας, ο γεννήτορας είναι ικανός να προπονήσει και να παρουσιάσει οποιοδήποτε συνελικτικό (και μη) νευρωνικό δίκτυο. Η δημιουργία των αρχείων περιγραφής (json, txt) είναι επίσης ανεξάρτητη του τύπου του νευρωνικού δικτύου. Ωστόσο, εξαιτίας της κληρονομικότητας των δομών από τις προκατασκευασμένες βιβλιοθήκες μας, μπορεί να υποστηρίξει νευρωνικά δίκτυα που περιέχουν επίπεδα που έχουμε υλοποιήσει. Τα περισσότερα συνελικτικά νευρωνικά δίκτυα χρησιμοποιούν όμως τα κατασκευασμένα επίπεδα με ελάχιστες εξαιρέσεις. Η διαφορές απο μοντέλο σε μοντέλο έγκεινται κυρίως στο πλήθος των επιπέδων, στον αριθμό των φίλτρων καθώς και τον τύπο της δειγματοληψίας.

Κλείνοντας, όσο αφορά την ικανότητα κλιμάκωσης ένας από τους βασικότερους στόχους της εργασίας ήταν η χρήση χαμηλού bitwidth και η εξάλειψη οποιασδήποτε μορφής εξωτερικής μνήμης και ανάγκης σε επικοινωνιακό εύρος ζώνης (transfer bandwidth), καθιστώντας του παράγοντες αυτούς ανεξάρτητους από το μέγεθος της εφαρμογής. Επιπλέον,ο επιταχυντής χρησιμοποιεί μικρές σε μέγεθος και κατανάλωση υπολογιστικές μονάδες 8-bit uint που μπορούν να αυξηθούν σε βαθμό που θα ικανοποιούσε το μεγαλύτερο εύρος των συνηθισμένων αρχιτεκτονικών νευρωνικών δικτύων.

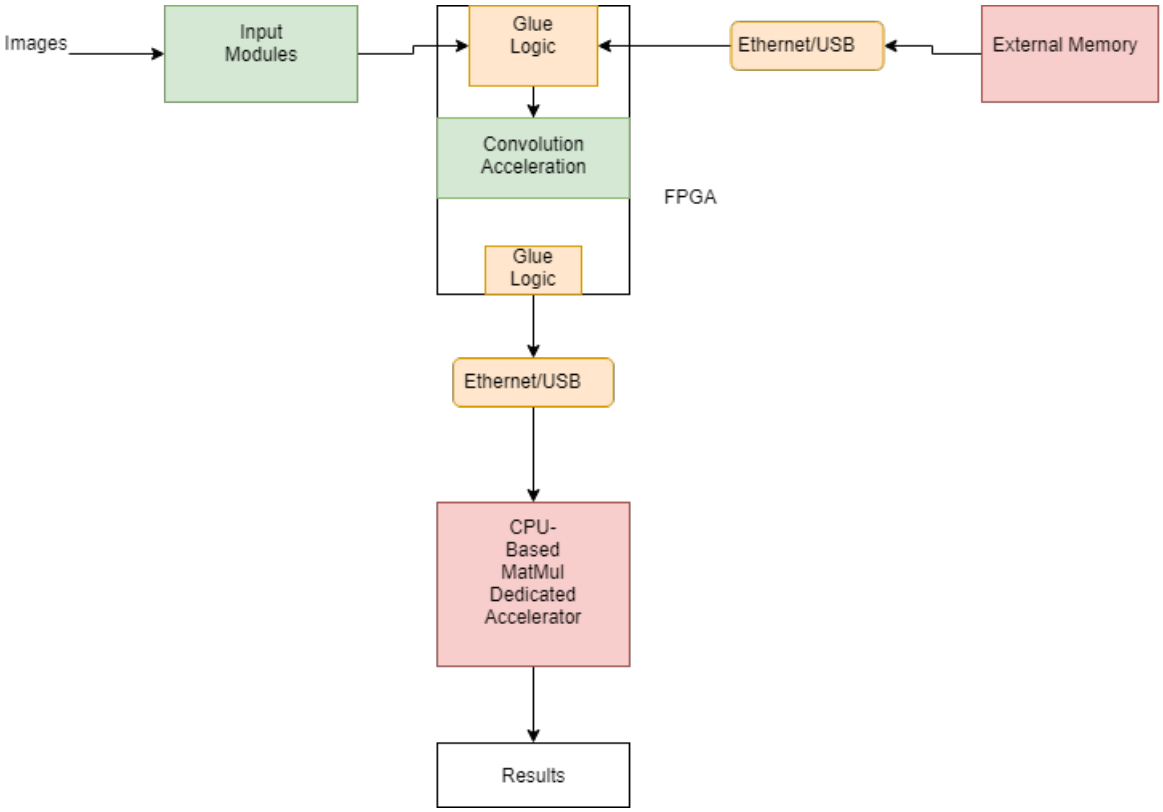
Η μόνη ¨επικίνδυνη' μετρική θα ήταν η κατανάλωση σε LUT και Static Power του FPGA. Για την μοντελοποίηση της κύριας εφαρμογής μας χρησιμοποιήθηκαν διάφορα FPGA μικρής

- μεσαίας κλίμακας στα οποία δεν αντιμετωπίστηκε κανένα πρόβλημα πόρων, ωστόσο ένα υπερβολικά περίπλοκο μοντέλο θα χρειαστεί FPGA μεγάλης κλίμακας.

4.5 Τελικός Σχεδιασμός

4.5.1 Εξωτερικές βελτιώσεις

Συνοψίζοντας, παραθέτουμε αναλυτικά τις διαφορές του προτεινόμενου design μας σε σχέση με τους υπάρχοντες DHM σχεδιασμούς:



Σχήμα 4.8 Τυπικό DHM design

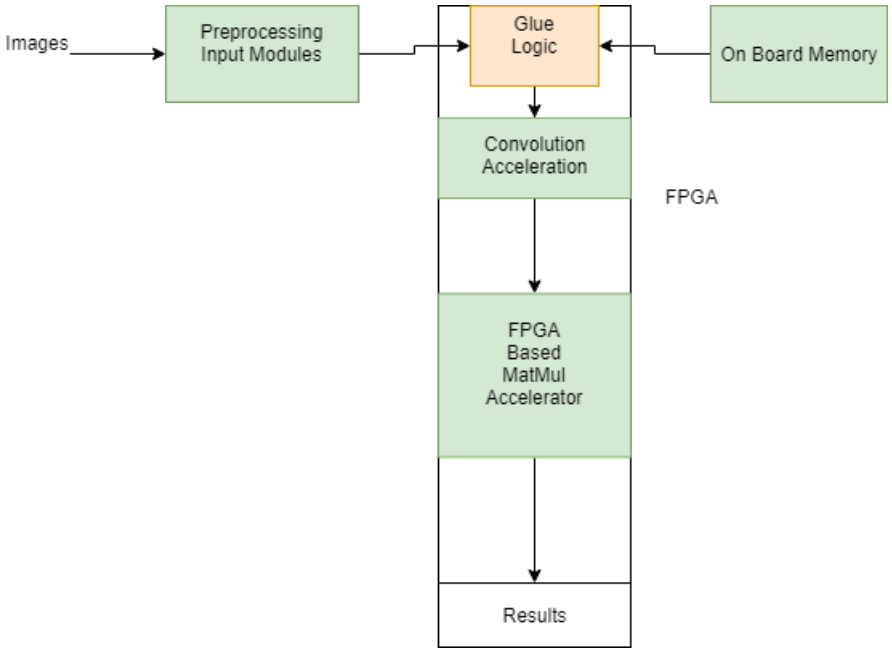
Στο σχήμα 4.8 παρατηρούμε τη δομή των DHM design στα οποία βασιστήκαμε. Αποτελούνται από τις δομές χειρισμού εισόδου, εξωτερική μνήμη, διεπαφή USB/Ethernet για επικοινωνία με τον έξω κόσμο καθώς και στοιχεία συχρονισμού και μορφοποίησης(glue logic) τα οποία διασφαλίζουν πως τα δεδομένα θα μετατρέπονται στην εκάστοτε μορφή που αναμένει ο δίαυλος μέσω του οποίου μεταφέρονται/τροποποιούνται.

Ιδιαίτερη προσοχή πρέπει να δωθεί στις περιοχές υψηλού φόρτου/καθυστέρησης τις οποίες θα αποκαλούμε εφεξής κρίσιμες περιοχές. Οι περιοχές αυτές (που σημαδεύονται με κίτρινο χρώμα) αποτελούν σημεία συσσώρευσης δεδομένων και εισάγουν μεγάλες απαιτήσεις σε bandwidth καθώς και on-board λογικής για την εξυπηρέτησή τους.

Αξίζει να σημειωθεί επίσης πως σημαντικά κομμάτια του επιταχυντή είναι σύνηθες να βρίσκονται εκτός του FPGA. Τα τμήματα αυτά σημαδεύονται με κόκκινο χρώμα και αποτελούν

στοιχεία που η ενσωμάτωσή τους στο design θα οδηγούσε σε καλύτερη απόδοση, μεγαλύτερη επίβλεψη των διαδικασιών επεξεργασίας καθώς και στην εξάλειψη ανάγκης επικοινωνίας με εξωτερικούς υπολογιστικούς και αποθηκευτικούς πόρους.

Το FPGA έτσι θα αποτελούσε μια αυτόνομη επεξεργαστική μονάδα υπεύθυνη για όλες τις επί μέρους λειτουργίες.

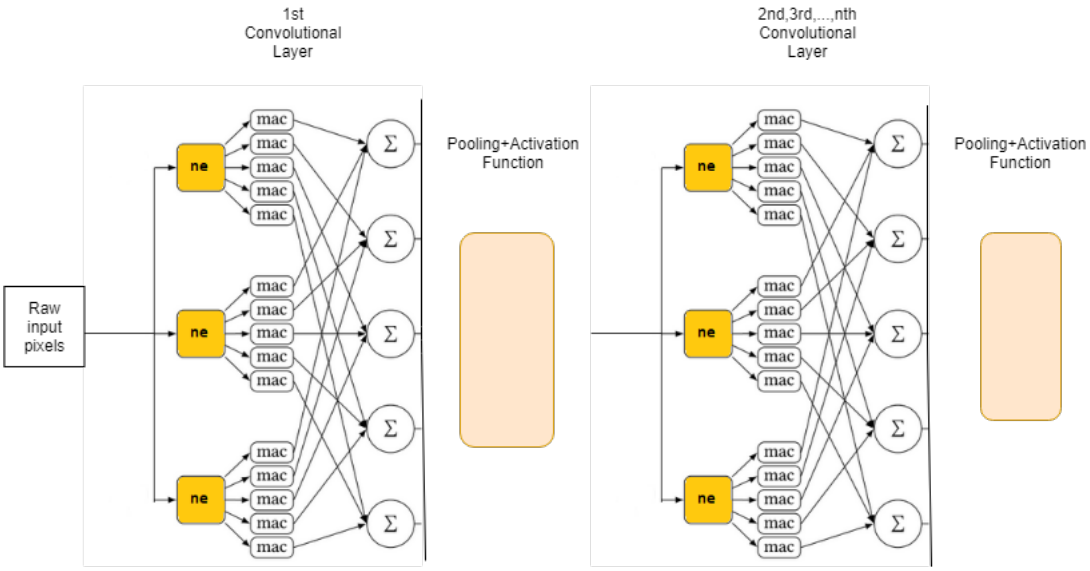


Σχήμα 4.9 Προτεινόμενη δομή DHM Accelerator

Στο σχήμα 4.9 παρατηρούμε την προτεινόμενη δομή DHM design. Αποτελείται και αυτή από τη δομή χειρισμού εισόδου, με προσθήκη ενός προεπεξεργαστή δεδομένων για την υλοποίηση του 1-bit mapping. Δεν διαθέτει εξωτερική αλλά on-board (ROM) μνήμη, εξαλείφοντας έτσι την ανάγκη για glue logic τοπικών και εξωτερικών δομών καθώς μηδενίζει την ανάγκη για bandwidth. Επιπλέον διατηρεί on-board δομές για την αποτελεσματική υλοποίηση του πολλαπλασιασμού πινάκων συνιστώντας έτσι μια ενιαία μονάδα επιτάχυνσης (standalone accelerator device).

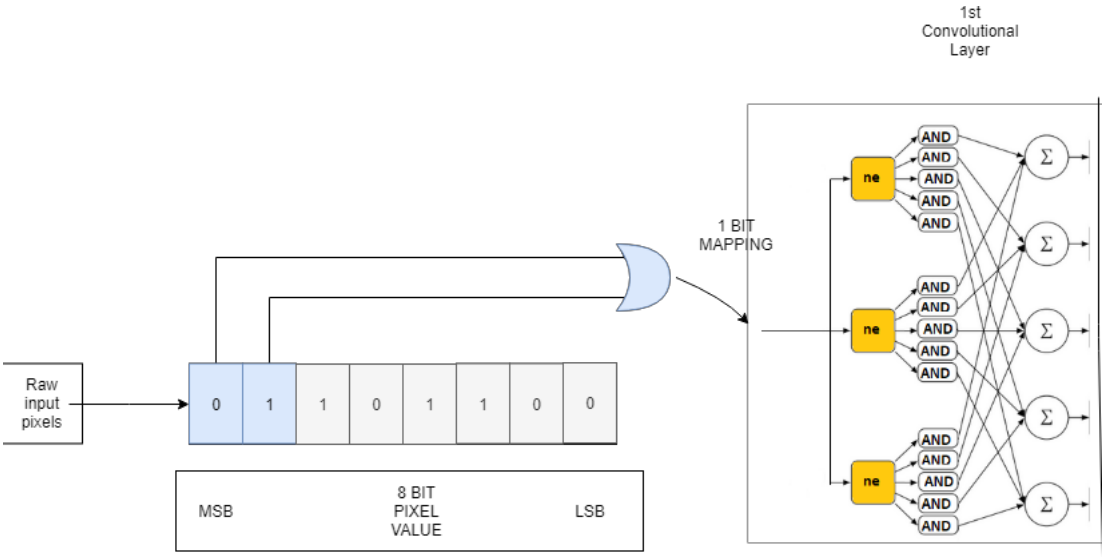
4.5.2 Εσωτερικές βελτιώσεις

Στην εικόνα 4.10 παρατηρούμε την εσωτερική δομή ενός DHM Accelerator σύμφωνα με τις προτεινόμενες τεχνικές και δικές μας τροποποιήσεις κατά τον βασικό σχεδιασμό. Παρατηρούμε την εκτεταμένη - αλλά βέλτιστη - χρήση μονάδων mac ύστερα από κάθε Neighbour Extractor, μονάδες που αποτελούν το βασικό καταναλωτή υλικού σε ένα συνελικτικό επίπεδο. Οι υπόλοιπες δομές αποτελούνται απλά από καταχωρητές,μικρούς αθροιστές και χρονικούς-λογικούς ελεγκτές.



Σχήμα 4.10 Εσωτερική δομή DHM Accelerator με προσθήκη Neighbour Extractor.

Στην εικόνα 4.11 παρατηρούμε την εσωτερική δομή του βελτιωμένου σχεδιασμού μας μέσω του 1-bit mapping. Παρατηρούμε πως η χρήση μονάδων mac αντικαθίσταται με πύλες AND μειώνοντας δραματικά την κατανάλωση υλικού. Επιπλέον παραθέτουμε μια εναλακτική μέθοδο πραγματιποίησης του 1-bit mapping όχι σε επίπεδο εξωτερικής προεργασίας αλλά online με την χρήση μιας πύλης OR στα 2 σημαντικότερα ψηφία του pixel value. Αξίζει να σημειωθεί πως τα υπόλοιπα συνελικτικά επίπεδα υλοποιούνται κατά τον σχεδιασμό που παρουσιάστηκε στο σχήμα 4.10.



Σχήμα 4.11 Βελτιωμένη Εσωτερική δομή DHM Accelerator.

4.5.3 Shallow CNNs και η σημασία του 1ου συνελικτικού επιπέδου

Γιατί όμως η δραστική μείωση υπολογιστικών πόρων μόνο στό πρώτο συνελικτικό επίπεδο έχει τόση σημασία; Η απάντηση δίνεται από μια κατηγορία συνελικτικών νευρωνικών δικτύων, τα επονομαζόμενα πλατιά-ρηχά συνελιτκικά δίκτυα (Wide and Shallow CNNs τα οποία επιτυχάνουν με αρκετές τροποποιήσεις ίδια και πολλές φορές καλύτερα ακόμα αποτελέσματα σε σχέση με τα πολυεπίπεδα δίκτυα.

Υπάρχει πολύ γνωστό πρώιμο θεωρητικό έργο σχετικά με την ικανότητα γενίκευσης και μοντελοποίησης αποφάσεων από ένα νευρωνικό δίκτυο. Για παράδειγμα, αποδείχθηκε ότι ένα δίκτυο με αρκετά μεγάλο ενιαίο κρυμμένο στρώμα σιγμοειδών μονάδων μπορεί να προσεγγίζει κάθε όριο απόφασης προτασιακής λογικής. Με βάση αυτά, συγκεκριμένες κατηγορίες προβλημάτων όπως η αναγνώριση ακμών μπορεί να υλοποιηθεί με αρχιτεκτονικές που δίνουν περισσότερο βάρος στα πρώτα και λιγότερο υπολογιστικό φόρτο στα μεθεπόμενα επίπεδα. Έτσι καταδεικνύεται η σημαντικότητα της ελάφρυνσης και βελτίωσης της απόδοσης του 1ου συνελικτικού επιπέδου καθώς και η τροποποίηση του μοντέλου για υλοποίηση των περισσότερων πράξεών του εκεί, ενώ δικαιολογείται η κατά pareto αναζήτησή μας για ένα δίκτυο τέτοιων προδιαγραφών.

Κεφάλαιο 5

Πειραματική Αξιολόγηση

Στο κεφάλαιο αυτό θα παρουσιαστεί αναλυτικά η πειραματική αξιολόγηση και ο συγκριτικός έλεγχος της απόδοσης των διαφόρων μεθολογιών στην υλοποίηση του επιταχυντή μας.

5.1 Παράμετροι εισόδου

Τα πειράματά μας θα διαθέτουν 5 βαθμούς ελευθερίας - παράμετρους εισόδου:

- Χρήση η μη της χαρτογράφισης εισόδου 1-bit.
- Χρήση διαφορετιού τύπου κβάντισης συντελεστών.
- Χρήση adder trees ή των on-board DSP.
- Εύρος bit του design.

5.2 Παράμετροι αξιολόγησης

Η αξιολόγιση των design με τους παραπάνω συνδυασμούς σχεδιαστικούς παραμέτρων θα γίνει με βάση τις εξής μετρικές:

- Απώλεια ακρίβειας σε σχέση με το αρχικό μοντέλο (Accuracy Loss).
- Ταχύτητα υπολογισμών - ρυθμός παραγωγής αποτελεσμάτων (Speed and Throughput).
- Αποτύπωμα Μνήμης (Memory Footprint).
- Κατανάλωση Πόρων (Resource Utilization).
- Απαιτούμενη Ισχύς (Power).

5.3 Οργάνωση πειραμάτων

Όλα τα υπό εξέταση design προέκυψαν από τον αυτόματο ¨γεννήτορα¨, χρησιμοποίησαν την ίδια βιβλιοθήκη που δημιουργήσαμε και προπονήθηκαν σε μοντέλα ίδιας έκδοσης Tensorflow 1.10. Η σύνθεσή τους πραγματοποιήθηκε στο περιβάλλον Vivado 2017.2. Τα αποτελέσματα εξετάστηκαν με τη μέθοδο Pareto και τα επικρατέστερα design στη συνέχεια συγκρίθηκαν με άλλες σχετικές εργασίες και μεθοδολογίες.

Πίνακας 5.1: Accuracy Results

Αποτελέσματα Απώλειας Ακρίβειας	
Design	Accuracy Loss
Κβάντιση 8-bit uint	1.0%
Χαρτογράφιση+Κβάντιση 8-bit uint	1.0%
Κβάντιση rescaling + rounding approximation	3.8%
Χαρτογράφιση+Κβάντιση rescaling+rounding approximation	2.9%

Από τον πρώτο πίνακα συμπεραίνουμε πως η χρήση κβάντισης τύπου 8-bit uint αντί για κλασσικής κλιμάκωσης στρογγυλοποίησης έχει δραματικά μικρότερες απώλειες ακρίβειας κατά τη μεταφορά σε FPGA και πως η μέθοδος χαρτογράφησης σε πίνακα 1-bit δεν επιβαρύνει την ακρίβεια του σχεδιασμού.

Πίνακας 5.2: Latency-Results Timings

Αποτελέσματα Ταχύτητας / Ρυθμού παραγωγής αποτελεσμάτων		
Design	Latency	Cycles to Next Result
Χρήση αποκλειστικά DSP	827cc	224cc
Χαρτογράφιση + μερική χρήση DSP	825cc	222cc
Χαρτογράφιση + χρήση adder treess και απλών πολλαπλασιαστών	823cc	219cc

Στον δεύτερο πίνακα αποτελεσμάτων παρατηρούμε πως η αποκλιστική χρήση των DSP αλλά και ενός συνδυασμού DSP με πύλες AND λόγω χαρτογράφισης υστερούν σε σχέση με το συνδυασμό χαρτογράφισης και των απλών δένδρων αθροιστών και default πολλαπλασιαστών που δίνουμε ως οδηγία στην σύνθεση. Παρόλα αυτά στους επόμενους πίνακες φαίνεται γιατί εν τέλει η 2η λύση μερικής χρήσης DSP αποτελεί την βέλτιστη επιλογή.

Πίνακας 5.3: Memory Footprint

Αποτύπωμα Μνήμης	
Design	ROM Memory Footprint
Αρχικό μέγεθος	13.28 KB
Κβάντιση rescaling + rounding approximation	4.56 KB
Κβάντιση 8-bit uint	3.32 KB

Στον πίνακα 3 παρατηρούμε την δραματική μείωση της on-board ROM memory ανάλογα με την επιλογή κάποιου τρόπου κβάντισης σε σχέση με την επιλογή των αρχικών συντελεστών.

Πίνακας 5.4: Resource Utilization

Κατανάλωση Πόρων					
Design	LUT	FF	DSP	IO	BUFG
Κβάντιση 8-bit uint + πλήρη χρήση DSP	11540	2530	120	19	1
Χαρτογράφιση + Κβάντιση 8-bit uint + χρήση adder trees	10360	2545	0	19	1
Χαρτογράφιση + Κβάντιση 8-bit uint + μερική χρήση DSP	5353	2536	60	19	1

Σύμφωνα με τον πίνακα 4 παρατηρείται δραματική μείωση των απαιτούμενων LUT με την μέθοδο χαρτογράφισης καθώς και μείωση των απαιτούμενων DSP με αποκλειστική χρήση τους στο τελευταίο πλήρως συνδεδεμένο επίπεδο. Καταλήγουμε λοιπόν πως ο βέλτιστος σχεδιαστικός συνδύασμός συνίσταται από όλες τις προαναφερθέντες υπαρπαραμετροποιήσεις και επιτυγχάνει χωρίς εξωτερική μνήμη, με πλάτος bit 8 ψηφίων και μόλις 1% απώλεια την σωστή λειτουργία και υλοποίηση του νευρωνικού δικτύου.

5.3.1 Αξιολόγιση Επιτάχυνσης

Στην ενότητα αυτή παρουσιάζουμε τα αποτελέσματα σύγκρισης του επιταχυντή μας με διάφορες αρχιτεκτονικές ενός τυπικού υπολογιστικού συστήματος και παραθέτουμε τα όρια βέλτιστης χρήσης τις κάθε επιλογής.

Το σύστημα υλοποίησης και ανταγωνιστής του επιταχυντή μας αποτελείται από τα εξής χαρακτηριστικά:

1. AMD RYZEN 1600 6 Core 3,3GHZ (default MIMD settings)
2. AMD RYZEN 1600 6 Core 3,3GHZ (Accelerated AVX2 SIMD settings)
3. NVIDIA EVGA 1060 3GB / CUDA Compute Capability 6.1
4. CUDA version 9 with cuDNN 7.0

Αντίθετα η πλατφόρμα FPGA στην οποία έγινε target το design είναι η Zybo Zynq-7000, με το ρολόι του accelerator να τοποθετείται ενδεικτικά στα 60Mhz.

Τα πειράματα αξιολογήθηκαν με βάση τη χρονομέτρηση εκτέλεσης ενός forward-pass μεγέθους 28.000 εικόνων, περιλαμβάνοντας κάθε φορά διαφορετικές περιοχές ενδιαφέροντος. Αναλυτικότερα η χρονομέτρηση περιλάμβανε χρόνους εκτέλεσης:

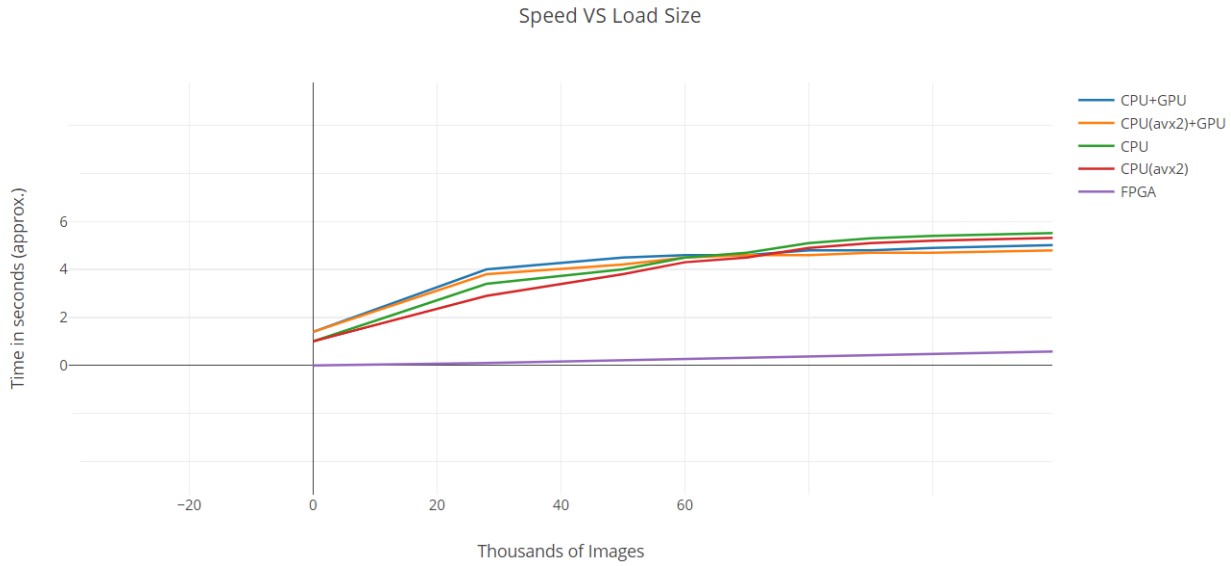
1. Αυτοτελούς όλου του προγράμματος σε Python για την λήψη προβλέψεων **(Full)**.
2. Των περιοχών μεταφοράς στη μνήμη,προεπεξεργασίας και τελικής εξαγωγής αποτελεσμάτων **(Main process)**.
3. Των περιοχών προεπεξεργασίας και εξαγωγής αποτελεσμάτων **(Execution with memory preprocessing operations)**.
4. Μόνο της εκτέλεσης εξαγωγής αποτελεσμάτων **(Execution only)**.

Πίνακας 5.5: Speedup Benchmarks

Χρόνος Εκτέλεσης				
Settings	Full	Main Process	ExM	Ex
AMD default + GPU	4.239s	2.851s	2.633s	1.223s
AMD AVX2 + GPU	3.839s	2.151s	1.933s	1.203s
AMD default	3.469s	2.497 s	2.611s	1.201s
AMD AVX2	2.920s	2.104 s	1.942s	1.007s
FPGA Accelerator	—	—	—	0.109s

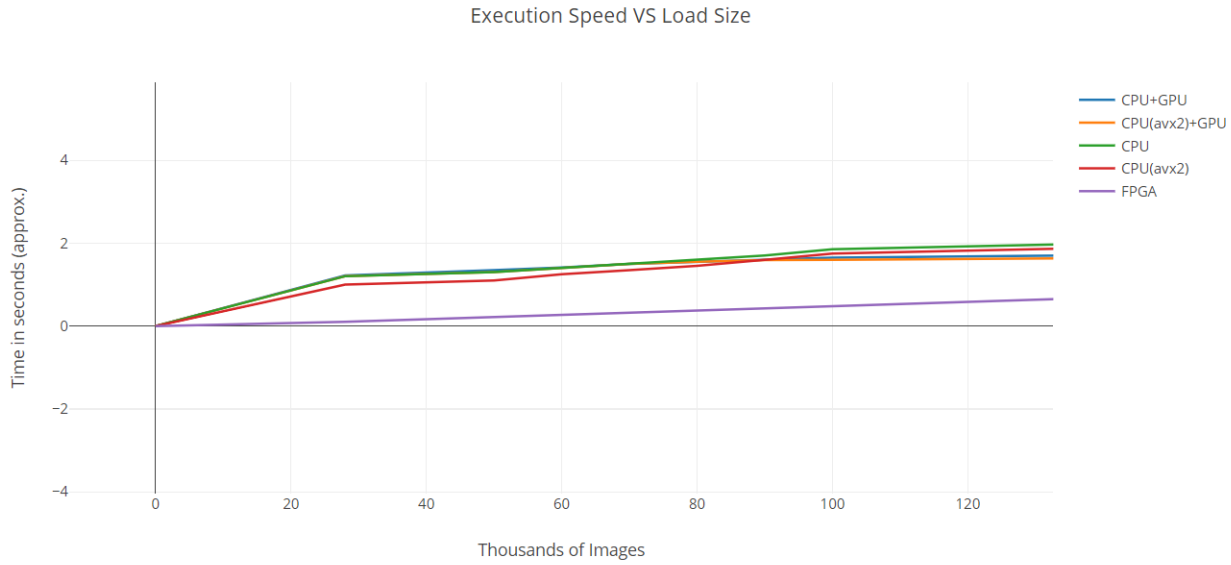
Με βάση τις παραπάνω μετρήσεις βλέπουμε πως σημαντικό χρονικό τμήμα της υλοποίησης ενός συνελικτικού νευρωνικού δικτύου αποτελούν διαδικασίες ανάγνωσης και μεταφοράς στη μνήμη εικόνων(περίπου 1 second). Επιπλέον τίθεται ένα σημαντικό overhead για την διαδικασία δεύσμευσης και ενεργοποίησης της GPU για την επιτάχυνση και παραλληλοποίηση των υπολογισμών που ανέρχεται επίσης κοντά στα 1.5 seconds, το οποίο σταδιακά ξεπερνάται από το όφελος του γρηγορότερου υπολογισμού μέσω της GPU των προσδωκώμενων αποτελεσμάτων. Τέλος αξίζει να σημειωθεί πως η εναλαγή της CPU σε SIMD εντολές για την επιτάχυνση πολλαπλασιασμών πινάκων οδηγεί σε πολύ καλύτερα αποτελέσματα στο τμήμα των πλήρως συνδεδεμένων επιπέδων.

Ένα χαρακτηριστικό διάγραμμα της προσδοκόμενης πλήρους ταχύτητας σε σχέση με το μέγεθος του υπολογιστικού φόρτου δίνεται παρακάτω:



Σχήμα 5.1 Διάγραμμα ταχύτητας/φόρτου

Στο σχήμα 5.1 παρουσιάζονται οι χρόνοι πλήρους εκτέλεσης διαφόρων συνδυασμών σε σχέση με την καθαρή εκτέλεση του FPGA. Παρατηρούμε πώς το design μας εξασφαλίζει επιτάχυνση x10 σε σχέση με τον γρηγορότερο συνδυασμό (CPU σε AVX2) για τις πρώτες 28.000 εικόνες.



Σχήμα 5.2 Διάγραμμα ταχύτητας εκτέλεσης/φόρτου

Στο σχήμα 5.2 παρουσιάζονται οι χρόνοι καθαρής εκτέλεσης διαφόρων συνδυασμών σε σχέση με την καθαρή εκτέλεση του FPGA, χρόνοι δηλαδή χωρίς να λαμβάνουμε υπόψιν την προεπεξεργασία και πολυνηματοποίηση.

Εδώ διαφαίνεται καθαρά πως σε ένα αρκετά μεγάλο φόρτο εργασίας η υλοποίηση GPU θα αρχίσει να ξεπερνά και τον επιταχυντή μας που έχει καθαρά γραμμική αύξηση του χρόνου υλοποίησης λόγω pipelined design σε σχέση με την σχεδόν λογαριθμική απόδοση της GPU.

Τέλος να σημειωθεί πως τα διαγράμματα προυποθέτουν πως καμία συσκευή δεν θα λειτουργεί σε ακραίες τιμές φόρτου ή μνήμης (no memory leaks) και πως οι υπολογισμοί πραγματοποιούνται ορθά (no overclocking).

Έτσι αποδεικνύεται πως προβλήματα απλούστερης δομής και ικανά να δεχτούν προσεγγιστικές μεθόδους επαναδιατύπωσης, ειδικότερα προβλήματα που απαιτούν λίγα φίλτρα και ελαφρότερα δίκτυα όπως το edge detection και η αναγνώριση χαρακτήρων σε grayscale είναι κατάλληλα για αντιμετώπιση με περισσότερο low-level δομές λόγου του περιττού φόρτου που εισάγουν στις πιο παραδοσιακές μεθόδους επίλυσής τους.

Κεφάλαιο 6

Επίλογος

6.1 Σύνοψη - Συμπεράσματα

Σύμφωνα με τα αποτελέσματα των παραπάνω μετρήσεων, οι σχεδιαστικές επιλογές και ιδέες μας προσφέρουν όντως ορθά και ακριβή αποτελέσματα και επιβεβαιώνονται αριθμητικά ως λύσεις επιτάχυνσης και ενεργειακά κομψού σχεδιασμού. Επιβεβαιώνεται έτσι η πρόταση της προσαρμογής των σχεδιαστικών επιλογών όχι μονάχα σε αυστηρά μαθηματικά μοντέλα, αλλά επίσης στις ανάγκες και ιδιαιτερότητες του εκάστοτε προβλήματος. Αυτές είναι πολλές φορές ικανές να οδηγήσουν σε ακόμη μεγαλύτερες βελτιώσεις και απλοποιήσεις όπως στην χαρτογράφιση 1-bit και την κβαντοποίηση 8-bit uint στην περίπτωσή μας.

Εν ολίγοις στην εργασία αυτή βρεθήκαμε αρχικά αντιμέτωποι με το πρόβλημα της δημιουργίας ενός FPGA επιταχυντή για την βελτίωση της απόδοσης ενός συνελικτικού νευρωνικού δικτύου. Επιλέξαμε ως μετρική, το MNIST, ένα χαρακτηριστικό πρόβλημα ταξινόμισης χειρόγραφων ψηφίων σε ασπρόμαυρη κλίμακα. Η επιλογή έγινε προς την ευκολία εύρεσης δεδομένων, καθώς και τις διάφορες υπάρχουσες μεθοδολογίες για την αντιμετώπισή του που αποτέλεσαν βάση για την υλοποίηση του δικού μας μοντέλου. Στη συνέχεια με τη χρήση Python και Tensorflow προπονήσαμε το μοντέλο μας και στη συνέχεια πραγματοποιήσαμε τις μεθόδους προεπεξεργασίας, κβάντισης και εξαγωγής των συντελεστών του δικτύου. Συγκρίναμε τα αρχικά με τα τροποποιημένα μοντέλα με μετρική την απώλεια ακρίβειας και καταλήξαμε στο βέλτιστο ισοζύγιο προσαρμογών έναντι πιστότητας. Ύστερα δημιουργήσαμε σε VHDL τα επιμέρους στοιχεία που αποτελούν ένα τυπικό νευρωνικό δίκτυο και τα οργανώσαμε σε μια βιβλιοθήκη. Μετά από αρχικές μετρήσεις και υλοποιήσεις καταλήξαμε στο μοντέλο απεικόνισης DHM με το δίκτυο να αποτελεί ουσιαστικά ένα υπολογιστικό γράφο και οργανώσαμε τα επιμέρους στοιχεία ώστε να υπακούν στις αρχές του σχεδιασμού με σωληνώσεις (pipeline). Μετατρέψαμε την συνέλιξη σε μια πράξη ψηφιακού φίλτρου transpose FIR και αναθεωρήσαμε τον τρόπο με τον οποίο πραγματοποιείται ο πολλαπλασιασμός πινάκων ώστε να συνάδει με τις κατασκευαστικές αρχές που επιδιώξαμε να διατηρήσουμε. Στη συνέχεια προσαρμόσαμε τα στοιχεία ώστε να εκμεταλεύονται τόσο την κβάντιση όσο και την χαρτογράφιση 1-bit και καταγράψαμε τις βελτιώσεις που σημειώθηκαν μέσω τις απλοποίησης του πρώτου συνελικτικού επιπέδου σε πύλες AND, της εξάλειψης ανάγκης ύπαρξης συνάρτησης

ενεργοποίησης και τις επικείμενης μείωσης εύρους σε bit του τελικού σχεδιασμού. Όλα αυτά οδήγησαν σε ένα μικρότερο ταχύτερο και ενεργειακά βιώσιμο design ικανό να αντιμετώπισει με την ίδια απόδοση απαιτιτικές αρχιτεκτονικές.

Τελικώς, αναπτύξαμε σε Python ένα γεννήτορα Tensorflow to VHDL ώστε να αυτοματοποιήσουμε τη διαδικασία σύνθεσης. Ο γεννήτορας χρησιμοποιεί την βελτιωμένη βιβλιοθήκη μας και αναλαμβάνει την προπόνηση, αποθήκευση, καταγραφή αποτελεσμάτων καθώς και την αυτόματη συγγραφή του top module του τελικού design.

Από την άποψη βελτιώσεων η εργασία πέτυχε:

1. Low Level Implementation flexibility έναντι στις προυπάρχουσες HDL επιλογές με την αυτοματοποίηση των υπερπαραμετροποιήσιμων τμημάτων σε βιβλιοθήκες.
2. 52% Μείωση Ισχύος σε σχέση με παραδοσιακό design.
3. 75% Μείωση Αποτυπώματος μνήμης σε σχέση με Μη κβαντισμένα design.
4. 27% Μείωση Αποτυπώματος μνήμης σε σχέση με off line quantization/rescaling.
5. Μόλις 1% απώλεια ακρίβειας μοντέλου με χρήση αποκλειστικά ακέραιων συντελεστών.
6. 53% Μείωση απαιτήσεων LUT σε σύγκριση με αρχιτεκτονικές χωρίς την τεχνική του 1-bit Image Mapping.
7. x10 επιτάχυνση στο επίσημο dataset σε σύκριση με GPU/SIMD AVX2 CPU.
8. Ενσωμάτωση με το περιβάλλον της Tensorflow.

6.2 Μελλοντικές επεκτάσεις

Κατα τη διάρκεια υλοποίησης της διπλωματικής εργασίας αντιμετωπίστηκαν αρκετές δυσκολίες και προβληματισμοί οι οποίοι υπήρξαν και πηγές έμπνευσης για την υλοποίηση συγκεκριμένων τμημάτων του σχεδιασμού.

Παρόλα αυτά δημιουργήθηκε ένα ευρύ φάσμα επιλογών οι οποίες αξίζει να ερευνηθούν και να ενσωματωθούν μελλοντικά:

1. *Ενσωμάτωση στο γεννήτορα επιπρόσθετων διαφορετικών Deep Learning Frameworks.*

   Προς το παρών ο γεννήτορας είναι ικανός να δημιουργεί αρχεία VHDL λαμβάνοντας υπόψιν ένα δεδομένο template όπως περιγράφει το περιβάλλον της Tensorflow ένα υπολογιστικό γράφο. Με τη συνεχή ανάπτυξη διαφόρων νέων frameworks αξίζει η δημιουργία ενός αρχείου παραμέτρων όπου θα διατηρεί διάφορα templates περιγραφής από εναλλακτικές όπως: Caffe2 ,O2, Chainer, Keras, Theano, PyTorch.

2. *Κατασκευή περισσότερων συναρτήσεων ενεργοποίησης.*

   Προς το παρών η εργασία υλοποιούσε την ευρέως χρησιμοποιούμενη συνάρτηση ενεργοποίησης ReLu. Ωστόσο διάφορες άλλες αρχιτεκτονικές χρησιμοποιούν τόσο κάποιες άλλες κλασσικές συναρτήσεις ενεργοποίησης όπως οι sigmoid, tanh αλλά και παραλλαγές της ReLu, όπως LeakyReLu, ELu. Πολλές από αυτές θα άξιζε να ενσωματωθούν μετά από την κατάλληλη βελτιστοποίηση στο σχεδιασμό.

3. *Επικοινωνία και καταμερισμός εργασιών με έναν on-board ARM processor.*

   Πολλές διαδικασίες όπως η προεπεξεργασία και η τροφοδότηση των εικόνων ως εισόδους πραγματοποιήθηκαν σε εξωτερικό υπολογιστή και δώθηκαν έτοιμα στον επιταχυντή μας. Κρίνεται δόκιμο λοιπόν οι εκάστοτε εικόνες να υφίστανται κατάλληλη προεπεξεργασία σε έναν on-board ARM επεξεργαστή που οι σύγχρονες πλακέτες SoC διαθέτουν δίπλα από το FPGA.

4. *Ενσωμάτωση προχωρημένων μεθόδων επιτάχυνσης συνελίξεων και πολλαπλασιασμών πινάκων*

   Εκτενής έρευνα έχει πραγματοποιηθεί στον μετασχηματισμό των συνελίξεων σε πράξεις λιγότερων πολλαπλασιασμών και περισσότερων προσθέσεων όπως ο αλγόριθμος Winograd και οι προσεγγίσεις FFT. Η ενσωμάτωσή τους αποτελεί ιδιαίτερα ενδιαφέρον εγχείρημα στην αναζήτηση ακόμη πιο γρήγορων λύσεων.

Βιβλιογραφία

[1] D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. Aurora: a new model and architecture for data stream management. *The VLDB Journal – The International Journal on Very Large Data Bases*, 12(2):120–139, 2003.

[2] Jocelyn S erot, Franois Berry, and Cdric Bourrasset. High-level dataflow programming for real-time image processing on smart cameras. *Journal of Real-Time Image Processing*, 12(4):635–647, 2016.

[3] Y LeCun, L Bottou, Y Bengio, and P Haffner. Gradient Based Learning Applied to Document Recogni- tion. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[4] Stylianos I. Venieris and Christos Savvas Bouganis. Fpg aConvNet: A Framework for Mapping Con- volutional Neural Networks on FPGAs. *Proceedings of the 24th IEEE International Symposium on Field-Programmable Custom Computing Machines*, pages 40–47, 2016.

[5] Edward A. Lee and Thomas M. Parks. Dataflow Process Netwo rks. *Proceedings of the IEEE*, 83(5):773– 801, 1995.

[6] Yaman Umuroglu, Nicholas J Fraser, Giulio Gambardella , Michaela Blott, Philip Leong, Magnus Jahre, and Kees Vissers. FINN: A Framework for Fast, Scalable Binar ized Neural Network Inference. *In Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays - FPGA '17*, pages 65–74, 2017.

[7] Kamel Abdelouahab, Maxime Pelcat, Jocelyn. Serot, Cedr ic Bourrasset, and Franois Berry. Tactics to Directly Map CNN graphs on Embedded FPGAs. *IEEE Embedded Systems Letters*, 2017.

[8] Hinton, Geoffrey E., Oriol Vinyals and Jeffrey Dean. "Distilling the Knowledge in a Neural Network." *CoRR*, 2015.

[9] Mart´ın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat

Monga, Sherry Moore, Derek Murray, Chris ´ Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol ´ Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-scale machine learning on heterogeneous systems, 2015*.

[10] Dong Yu, Adam Eversole, Mike Seltzer, Kaisheng Yao, Zhiheng Huang, Brian Guenter, Oleksii Kuchaiev, Yu Zhang, Frank Seide, Huaming Wang, et al. An introduction to computational networks and the computational network toolkit. *Technical report, Tech. Rep. MSR, Microsoft Research*, 2014.

[11] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *n CVPR'2015 arxiv.org/abs/1409.4842*, 2015.

[12] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. Multi-digit number recognition from Street View imagery using deep convolutional neural networks. *In International Conference on Learning Representations arxiv.org/pdf/1312.6082.*, 2014.

[13] Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. *In ACM SIGOPS Operating Systems Review, volume 41, pages 59–72. ACM*,2017.

[14] Deep Neural Network Compression with Single and Multiple Level Quantization Yuhui Xu, Yongzhuang Wang, Aojun Zhou, Weiyao Lin, Hongkai Xiong :1803.03289 [cs.LG].

[15] Do Deep Nets Really Need to be Deep? Lei Jimmy Ba, Rich Caruana, *arXiv:1312.6184*.

Γλωσσάριο

Ελληνικός όρος	Αγγλικός όρος
συνέλιξη	convolution
νευρωνικό δίκτυο	neural network
χβαντισμός	quantization
δειγματοληψία	sampling
βελτιστοποίηση	optimization
περιβάλλον ανάπτυξης	framework
σχεδιασμός σωληνώσεων	pipelining
παράθυρο	window
ρεύμα δεδομένων	data stream
φιλτράρισμα	filtering
ρυθμός παραγωγής αποτελεσμάτων	thoughput