**ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ**
**ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ**
**ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ**

**ΤΟΜΕΑΣ: Ηλεκτρονικής και Υπολογιστών**

**Εργαστήριο Σχεδιασμού Ολοκληρωμένων Κυκλωμάτων Μεγάλης Κλίμακας**

---

# Διπλωματική Εργασία

των φοιτητών του Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών της Πολυτεχνικής Σχολής του Πανεπιστημίου Πατρών

Γεωργίου Ευαγγέλου του Αθανασίου

Αριθμός Μητρώου: 1046900

Νικολάου Ρούσσου του Αλεξάνδρου

Αριθμός Μητρώου: 1046939

Θέμα

## «Αρχιτεκτονικές και Υλοποιήσεις του Νευρωνικού Δικτύου LeNet-5 σε FPGAs»

Επιβλέπων
Γεώργιος Θεοδωρίδης,
Επίκουρος Καθηγητής

**Αριθμός Διπλωματικής Εργασίας:**
**(Αριθμός μητρώου /2020)**

Πάτρα, Ιούλιος 2020
……./……./…….

# ΠΙΣΤΟΠΟΙΗΣΗ

Πιστοποιείται ότι η Διπλωματική Εργασία με θέμα

## «Αρχιτεκτονικές και Υλοποιήσεις του Νευρωνικού Δικτύου LeNet-5 σε FPGAs»

Των φοιτητών του Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών


Γεωργίου Ευαγγέλου του Αθανασίου

Αριθμός Μητρώου: 1046900

Νικολάου Ρούσσου του Αλεξάνδρου

Αριθμός Μητρώου: 1046939


Παρουσιάστηκε δημόσια και εξετάστηκε στο Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών στις ……../……../……

Ο Επιβλέπων

Γεώργιος Θεοδωρίδης

Επίκουρος Καθηγητής

Ο Διευθυντής του Τομέα

Βασίλειος Παλιουράς

Καθηγητής

**Αριθμός Διπλωματικής Εργασίας:**

**Θέμα: «Αρχιτεκτονικές και Υλοποιήσεις του Νευρωνικού Δικτύου LeNet-5 σε FPGAs»**

**Φοιτητές:**

Γεώργιος Ευαγγέλου του Αθανασίου
Νικόλαος Ρούσσος του Αλεξάνδρου

**Επιβλέπων:**

Γεώργιος Θεοδωρίδης

## Περίληψη

Με την αυξανόμενη διαθεσιμότητα σε υπολογιστικούς πόρους, η δημοσιότητα της μηχανικής μάθησης αυξάνεται ραγδαία όσον αφορά την ανάλυση και επεξεργασία δεδομένων, παρέχοντας λύσεις σε προβλήματα διαφόρων επιστημονικών πεδίων. Τα νευρωνικά δίκτυα αποτελούν έναν από τους πιο μελετημένους και χρησιμοποιούμενους υπο-τομείς της μηχανικής μάθησης, προσφέροντας αποτελεσματικές μεθόδους κατηγοριοποίησης δεδομένων. Διαδεδομένο παράδειγμα είναι η αναγνώριση εικόνων σε πραγματικό χρόνο, που παρουσιάζει ιδιαίτερο εμπορικό ενδιαφέρον, όπως η ταυτοποίηση προσώπων σε smartphones. Συνεπώς, υπάρχει αυξανόμενη ανάγκη για χωρικά και ενεργειακά αποδοτικές λύσεις, οι οποίες ικανοποιούν τις προδιαγραφές των εφαρμογών αυτών, διατηρώντας, ωστόσο, υψηλές επιδόσεις. Ως εκ τούτου, η μηχανική μάθηση απομακρύνεται από υλοποιήσεις λογισμικού, καθώς κατά κανόνα χρησιμοποιούν μη αποδοτικούς επεξεργαστές γενικής χρήσης. Ακολουθώντας αυτή την πορεία, η παρούσα διπλωματική εργασία έχει σκοπό τη σχεδίαση μιας αρχιτεκτονικής σε υλικό ως εναλλακτική λύση. Η προτεινόμενη δομή είναι βασισμένη στο νευρωνικό δίκτυο LeNet-5 και έχει τη δυνατότητα αναγνώρισης χειρόγραφων ψηφίων με υψηλή ακρίβεια. Το κύριο αποτέλεσμα της εργασίας αυτής είναι μια υλοποίηση σε FPGA, η οποία επιτυγχάνει τον παραπάνω στόχο, αφού πρώτα προηγήθηκε εξοικείωση και χρήση των εργαλείων της πλατφόρμας σχεδίασης Vivado Design Suite. Επεκτείνοντας το πεδίο μελέτης, ένας δεύτερος στόχος είναι η βελτίωση της συστημικής σχεδίασης, όσον

αφορά την αποδοτικότητα και επίδοση, εκμεταλλεύοντας μηχανισμούς συμπίεσης μήκους λέξεων και μεθόδους επιτάχυνσης της αρχιτεκτονικής. Μία σχολαστική ανάλυση της δομής του δικτύου, αποκαλύπτει λύσεις διαφορετικών λειτουργικών χαρακτηριστικών, με στόχο την βελτίωση των κύριων δεικτών επίδοσης του συνολικού συστήματος. Η παρούσα διπλωματική εργασία είναι βασισμένη, αλλά δεν περιορίζεται, στην συγκεκριμένη εφαρμογή. Συνεπώς, ακολουθώντας παρόμοια διαδικασία, διαφορετικά νευρωνικά δίκτυα μπορούν επίσης να υλοποιηθούν.

Λέξεις κλειδιά: Μηχανική Μάθηση; Νευρωνικά Δίκτυα; LeNet-5; Αναγνώριση Εικόνων; Σύνθεση Υψηλού Επιπέδου; Επιτάχυνση Υλικού; Σχεδίαση FPGA;

## Abstract

With the increasing availability of computational resources, machine learning has seen skyrocketing popularity in data analysis and processing, since it offers solutions to problems in various scientific areas. Neural networks are one of the most researched and applied subdomains of machine learning, providing effective data classification methods. For instance, real-time image recognition systems are of great commercial interest, such as face identification in smartphones. Thus, there is a growing need to develop area and power efficient solutions that meet the constraints of such applications, while maintaining high performance. As a consequence, machine learning is drifting away from software solutions, as they generally utilize inefficient general-purpose processors. Following this trend, this dissertation aims to design a hardware realization as an alternative. The proposed framework is based upon the LeNet-5 neural network and is able to recognize hand-written digits with great accuracy. The main achievement is an FPGA implementation, which achieves this goal, while the authors managed to familiarize with and utilize design tools provided with the Vivado Design Suite. Expanding the horizons of this diploma thesis, a second objective is to improve the system design, with regard to efficiency and performance, by exploiting word-length compression mechanisms and architectural acceleration methods. A close inspection of the network's structure reveals solutions of different functional characteristics, in order for the overall system's Key Performance Indicators (KPIs) to be enhanced. This framework is based upon, but not limited to, the current application. Therefore, following a similar workflow, other neural networks can be implemented, as well.



Keywords: Machine Learning; Neural Networks; LeNet-5; Image Recognition; High-Level Synthesis; Hardware Acceleration; FPGA Design;

# Ευχαριστίες

Αρχικά, θα θέλαμε να ευχαριστήσουμε τον καθηγητή μας κύριο Γεώργιο Θεοδωρίδη για τα κίνητρα που μας παρείχε και τις ιδέες που μας παρακινούσε να αναπτύξουμε, τόσο στα μαθήματα όσο και στην παρούσα διπλωματική εργασία.

Επίσης, σημαντικοί αρωγοί στην προσπάθειά μας όλα τα χρόνια της φοίτησής μας αποτελούν αναμφίβολα οι γονείς και τα αδέλφια μας, που μας συμπαραστάθηκαν με κάθε τρόπο σε όλα τα βήματά μας.

Τέλος, δε θα μπορούσαμε να είχαμε φτάσει ως εδώ χωρίς την παρουσία των φίλων μας, που κάθε στιγμή μας γέμιζαν με εμπειρίες, χαρές και μοναδικές στιγμές που θα χρωματίζουν για πάντα τις αναμνήσεις μας για αυτή τη σταδιοδρομία μας.

# Περιεχόμενα

# Κατάλογος σχημάτων

# Κατάλογος πινάκων

# 1. ΕΙΣΑΓΩΓΗ

## 1.1 Κίνητρο και στόχος της διπλωματικής εργασίας

Εφαλτήριο κάθε μελλοντικού μηχανικού είναι η εκμάθηση και εξοικείωση του με τα διαθέσιμα εργαλεία. Έχοντας ως βάση θεωρίες μαθηματικών, φυσικής, και μηχανικής, καταπιάνεται με την κατασκευή ή χρήση τεχνολογικών μέσων, έτσι ώστε να μπορεί να αντεπεξέλθει δημιουργικά και αποτελεσματικά στα προβλήματα που θα κληθεί να απαντήσει, αλλά και σε τυχόν υπάρχουσες λύσεις που θα χρειαστεί να βελτιώσει. Βάσει αυτών, η παρούσα διπλωματική εργασία ξεκίνησε με την εξερεύνηση των διαθέσιμων σχεδιαστικών και προγραμματιστικών εργαλείων. Περνώντας από όλα τα στάδια μιας συστημικής σχεδίασης, απώτερος στόχος ήταν η υλοποίηση ενός συστήματος τεχνητής νοημοσύνης.

Δεύτερο κίνητρο ενός μηχανικού αποτελεί ο συνδυασμός υπαρχόντων τεχνολογιών μεταξύ τους, με όραμα να κατασκευάσει κάτι καινοτόμο ή διαφορετικό από ήδη υπάρχοντα συστήματα. Η δόμηση εναλλακτικών λύσεων και η αξιολόγηση τους βάσει επίδοσης, κόστους, μεγέθους κ.τ.λ. απαιτεί μεθοδικότητα, πειραματισμό και εύστοχο ορισμό απαιτήσεων. Επόμενο ορόσημο της παρούσας διπλωματικής εργασίας αποτέλεσε, λοιπόν, η δημιουργία παραλλαγών της αρχικής υλοποίησης, με στόχο τη βελτιστοποίηση διαφορετικών μετρικών του συστήματος. Τα μέσα για την επίτευξη αυτού του στόχου ήταν η μεταβολή της αρχιτεκτονικής των δομικών στοιχείων του πρωταρχικού συστήματος και οι επαναλαμβανόμενες λειτουργικές και στατιστικές δοκιμές, για την διερεύνηση εναλλακτικών προσεγγίσεων.

Όσον αφορά την επιλογή σχεδιασμού ενός συστήματος τεχνητής νοημοσύνης, είναι γεγονός πως τα τελευταία χρόνια έχει κεντρίσει το ενδιαφέρον της βιομηχανίας, ενώ εφαρμόζεται ολοένα και περισσότερο στην καθημερινή ζωή. Ένας από τους κυρίαρχους κλάδους της τεχνητής νοημοσύνης είναι η αναγνώριση προτύπων. Ο κλάδος αυτός ασχολείται με την εξαγωγή χρήσιμων πληροφοριών από μεγάλο όγκο δεδομένων και τελευταία γνωρίζει αυξανόμενη διάδοση. Η διάδοση αυτή συνδέεται με την άνοδο της μαζικής συλλογής και εκμετάλλευσης πληροφοριών (Google, Facebook κ.τ.λ.), των αυτόνομων οχημάτων, νέων διαγνωστικών εργαλείων κ.ά..

Μία συνήθης εφαρμογή συστημάτων αναγνώρισης προτύπων είναι η κατηγοριοποίηση αντικειμένων σε εικόνες. Αυτό ήταν και το θέμα που διαπραγματευόμαστε στην παρούσα διπλωματική εργασία, εξετάζοντας το νευρωνικό δίκτυο LeNet-5, το οποίο χρησιμοποιείται για την αναγνώριση χειρόγραφων αριθμητικών ψηφίων. Μετά την εξοικείωση με υπάρχουσες μελέτες και υλοποιήσεις διαθέσιμες στη βιβλιογραφία, απώτερο στόχο αποτέλεσε η δημιουργία μίας εξ ολοκλήρου νέας σχεδίασης, τόσο σε λογισμικό, όσο και σε υλικό. Η σχεδίαση αυτή αποτέλεσε σημείο αναφοράς για τη διεξαγωγή στατιστικών μετρήσεων και βελτιστοποιήσεων, έτσι ώστε η διπλωματική εργασία αυτή να αποτελεί μία συγκριτική μελέτη ανάμεσα σε παραλλαγές της αρχικής υλοποίησης. Τέλος, εξετάστηκαν ορισμένες μέθοδοι συμπίεσης του νευρωνικού δικτύου, μέσω κβαντισμού των συντελεστών του, και έγινε εκτενής μελέτη της επίδρασης μετασχηματισμών κώδικα και αρχιτεκτονικής στα φυσικά και λειτουργικά μεγέθη του συστήματος.

## 1.2 Διάρθρωση της διπλωματικής εργασίας

**Κεφάλαιο 1:** Πραγματοποιείται μία σύντομη αναφορά στα εφαλτήρια σημεία και τους στόχους της παρούσας διπλωματικής εργασίας.

**Κεφάλαιο 2:** Αρχικά, γίνεται μία συνοπτική αναφορά στο επιστημονικό πεδίο της μηχανικής μάθησης. Δίνεται ιδιαίτερη έμφαση στα νευρωνικά δίκτυα, αναλύοντας τα είδη που χρησιμοποιήθηκαν στην παρούσα διπλωματική εργασία, ενώ εξετάζονται και άλλες χρήσιμες έννοιες. Τέλος, γίνεται επισκόπηση του νευρωνικού δικτύου LeNet-5, το οποίο χρησιμοποιήθηκε στην παρούσα διπλωματική εργασία.

**Κεφάλαιο 3:** Περιγράφονται τα βήματα της σχεδίασης του νευρωνικού δικτύου LeNet-5 σε γλώσσα λογισμικού (Python). Έπειτα, γίνεται αναφορά στην μοντελοποίηση και την εξαγωγή των βαρών του δικτύου για μετέπειτα χρήση.

**Κεφάλαιο 4:** Παρουσιάζεται το εργαλείο Vivado HLS, καθώς και η χρήση του για τον μετασχηματισμό του μοντέλου λογισμικού σε κώδικα χαμηλού επιπέδου και γλώσσα περιγραφής υλικού. Τέλος, περιγράφεται το σύστημα ελέγχου που αναπτύχθηκε για τον έλεγχο της ακρίβειας του νευρωνικού δικτύου.

**Κεφάλαιο 5:** Εξετάζονται διάφορες μελέτες που αφορούν την κβάντιση των συντελεστών σε νευρωνικά δίκτυα, ενώ παρουσιάζονται οι επιπτώσεις που είχαν αντίστοιχες μέθοδοι συμπίεσης στην υλοποίηση της παρούσας διπλωματικής εργασίας.

**Κεφάλαιο 6:** Παρουσιάζονται διάφορες τεχνικές βελτιστοποίησης κώδικα και αρχιτεκτονικής, καθώς και τα αποτελέσματα που είχαν στο σύστημα της παρούσας διπλωματικής εργασίας κατά την εφαρμογή τους σε αυτό.

**Κεφάλαιο 7:** Περιγράφεται η εφαρμογή μίας εκ των υλοποιήσεων του κεφαλαίου 6 σε πλήρως λειτουργικό σύστημα, εντός μίας πλακέτας FPGA.

**Κεφάλαιο 8:** Επίλογος και συμπεράσματα της παρούσας διπλωματικής εργασίας.

# 2. ΑΝΑΓΝΩΡΙΣΗ ΠΡΟΤΥΠΩΝ ΚΑΙ ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ

## 2.1 Εισαγωγή

Η αναγνώριση προτύπων είναι το επιστημονικό πεδίο το οποίο ασχολείται με την αυτοματοποιημένη απόδοση τιμών ή κάποιων διακριτικών στοιχείων, σε εισαγόμενα δεδομένα. Η διαδικασία αυτή αφορά την αυτόματη εύρεση μοτίβων στα *δεδομένα εισόδου* (*παραδείγματα*), χρησιμοποιώντας κατάλληλους υπολογιστικούς αλγορίθμους, οδηγώντας στην ταξινόμησή τους σε ξεχωριστές *κατηγορίες* (*κλάσεις*). Το επιστημονικό αυτό πεδίο ξεκίνησε να αναπτύσσεται τη δεκαετία του '60 και έχει πλέον ευρεία εφαρμογή σε πληθώρα επιστημονικών τομέων, όπως στην ιατρική και στην επιστήμη ηλεκτρονικού μηχανικού [1], [2], [3], [4].

Συστήματα αναγνώρισης προτύπων εμφανίζονται σε μεγάλο εύρος εφαρμογών. Παραδείγματα αποτελούν συστήματα πρόβλεψης καιρού, επεξεργασίας σημάτων, βιοπληροφορικής, μηχανικής όρασης, στατιστικής ανάλυσης κ.ά.. Η παρούσα διπλωματική εργασία επικεντρώνεται στην ανάλυση εικόνων και συγκεκριμένα την αναγνώριση γραπτών αριθμητικών ψηφίων.

Αναλόγως με την περίπτωση στην οποία εφαρμόζεται ένα σύστημα αναγνώρισης προτύπων, προτείνονται διαφορετικά μαθηματικά μοντέλα αλγορίθμων και υπολογιστικές υλοποιήσεις. Κύρια παραδείγματα αποτελούν οι *αλγόριθμοι κατηγοριοποίησης* (*classification algorithms*), *αλγόριθμοι ομαδοποίησης* (*clustering algorithms*), *αλγόριθμοι παλινδρόμησης* (*regression*

*algorithms*), καθώς και *κατηγορηματικοί αλγόριθμοι επισήμανσης ακολουθίας* (*sequence labeling algorithms*). Τα νευρωνικά δίκτυα αποτελούν ένα είδος αλγορίθμων κατηγοριοποίησης.

Στο κεφάλαιο αυτό γίνεται, αρχικά, μια εισαγωγή στα νευρωνικά δίκτυα και περιγράφεται ο τρόπος λειτουργίας τους αναλύοντας, παράλληλα, τα δομικά τους χαρακτηριστικά. Στη συνέχεια, πραγματοποιείται μια λεπτομερής περιγραφή του νευρωνικού δικτύου LeNet-5, το οποίο μελετήθηκε και υλοποιήθηκε στην παρούσα διπλωματική εργασία.

## 2.2 Νευρωνικά δίκτυα

Τα νευρωνικά δίκτυα είναι ο πλέον διαδεδομένος κλάδος μηχανικής μάθησης. Αποτελούνται από τρία βασικά στοιχεία: την **είσοδο**, τα **επίπεδα επεξεργασίας** και την **έξοδο**. Τα δεδομένα εισόδου είναι αριθμητικές τιμές που έχουν αποδοθεί σε κάποιο παράδειγμα, βάσει των γνωρισμάτων του που θα χρησιμοποιηθούν για την ταξινόμηση. Τα δεδομένα αυτά διαδίδονται διαμέσου των νευρώνων του εκάστοτε επιπέδου επεξεργασίας. Σε κάθε επίπεδο πραγματοποιούνται γραμμικοί και μη-γραμμικοί μετασχηματισμοί ή/και αναδιατάξεις στα δεδομένα. Μέσω των μετασχηματισμών αυτών πραγματοποιείται στο τελικό επίπεδο η ταξινόμηση των δεδομένων εισόδου σε κάποια κλάση. Η κατηγοριοποίηση αυτή γίνεται πιθανοτικά και επομένως στην έξοδο εμφανίζεται η πιθανότητα του παραδείγματος εισόδου να ανήκει στην κάθε κατηγορία.

Βάσει των παραπάνω, τα νευρωνικά δίκτυα απαιτούν μεγάλο αριθμό υπολογιστικών πράξεων και υψηλό ρυθμό διακίνησης δεδομένων. Λόγω της αύξησης των διαθέσιμων υπολογιστικών πόρων και βελτίωση της ταχύτητας προσπέλασης δεδομένων, τα νευρωνικά δίκτυα βρίσκουν ευρεία εφαρμογή τα τελευταία χρόνια.

Είναι εφικτό μέσα σε ένα σύστημα νευρωνικού δικτύου να συνυπάρχουν διαφορετικοί τύποι νευρώνων και επίπεδα επεξεργασίας. Κάθε τύπος βρίσκει διαφορετική εφαρμογή, ανάλογα με το είδος των δεδομένων και των απαιτήσεων επιδόσεων και ακρίβειας που ορίζονται. Παρακάτω αναλύονται οι τύποι των επιπέδων που χρησιμοποιήθηκαν στην παρούσα εργασία (Υποπαρ. 2.2.1 έως 2.2.4), ενώ στο τέλος γίνεται μία συνοπτική αναφορά στην εκπαίδευση και τον αλγόριθμο οπισθοδρομικής διάδοσης σφάλματος (Υποπαρ. 2.2.5).

### 2.2.1 Πλήρως διασυνδεδεμένα επίπεδα

Το περισσότερο διαδεδομένο είδος επιπέδου νευρώνων είναι το *πλήρως διασυνδεδεμένο επίπεδο* (*fully connected layer*). Πρόκειται για ένα είδος μη-γραμμικών επιπέδων επεξεργασίας, μικρής σχετικά πολυπλοκότητας. Η ονομασία του προκύπτει από το γεγονός ότι οι νευρώνες ενός τέτοιου επιπέδου συνδέονται με όλους ανεξαιρέτως τους νευρώνες του προηγούμενου και του επόμενου επιπέδου. Ένας σημαντικός αριθμός νευρωνικών δικτύων που απαντώνται στην βιβλιογραφία περιέχουν πλήρως διασυνδεδεμένα επίπεδα.

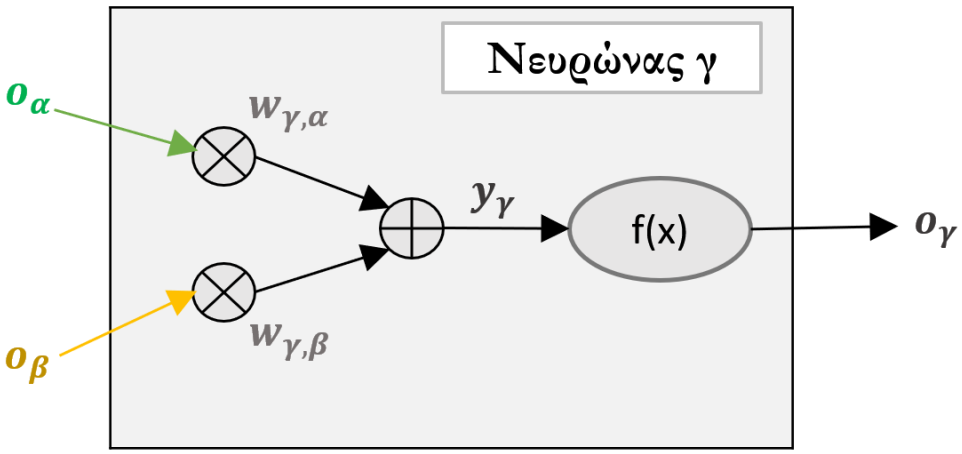
Η δομή των πλήρως διασυνδεδεμένων επιπέδων είναι παρόμοια με την αντίστοιχη των επιπέδων με νευρώνες Perceptron[1] [3], [5]. Για την παρακάτω ανάλυση, αρχικά θεωρείται ότι έχουμε δύο συνεχόμενα πλήρως διασυνδεδεμένα επίπεδα, με **j** (**1** έως **M**) τους νευρώνες του πρώτου και **i** (**1** έως **N**) τους νευρώνες του δεύτερου. Όλες οι είσοδοι του δεύτερου πλήρως διασυνδεδεμένου επιπέδου, $\mathbf{o_j}$ (οι έξοδοι του προηγούμενου επιπέδου), εισέρχονται σε κάθε νευρώνα του, αφού πρώτα πολλαπλασιαστούν με μία παράμετρο (*βάρος*), $\mathbf{w_{i,j}}$ (βάρος που συνδέει έναν νευρώνα **j** του πρώτου επιπέδου με έναν νευρώνα **i** του δεύτερου επιπέδου). Κάθε γινόμενο εισόδων-βαρών συνεισφέρει προσθετικά στην διαμόρφωση της εσωτερικής κατάστασης $\mathbf{y_i}$ του εκάστοτε νευρώνα, ενώ προστίθεται στο τελικό αποτέλεσμα μία *σταθερά αρχικοποίησης* (*bias*), $\mathbf{b_i}$ (Εξ. 2.1). Για λόγους όπως είναι η κανονικοποίηση και η αποφυγή του κορεσμού των αποτελεσμάτων, εφαρμόζεται στο τέλος μία *συνάρτηση ενεργοποίησης* (*activation function*) **f(x)** (βλ. Υποπαρ. 2.2.5), η οποία δέχεται ως είσοδο την εσωτερική κατάσταση και υπολογίζει την έξοδο του νευρώνα, $\mathbf{o_i}$ (Εξ. 2.2).

$$y_i = \sum_{j=1}^{M} w_{i,j} \cdot o_j + b_i \qquad \text{Εξ. 2.1}$$

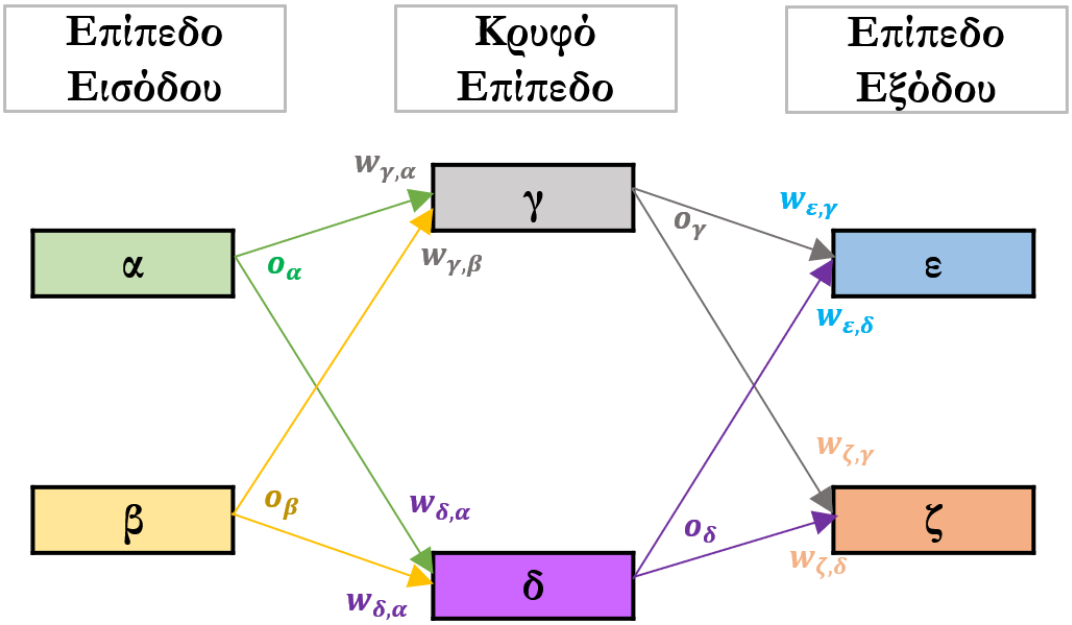
$$o_i = f(y_i) \qquad \text{Εξ. 2.2}$$

Γραφικές αναπαραστάσεις της ως άνω ανάλυσης για έναν πλήρως διασυνδεδεμένο νευρώνα και ένα αντίστοιχο επίπεδο νευρωνικού δικτύου φαίνονται στην Εικ. 2.1 και στην Εικ. 2.2 αντίστοιχα.

[1] Οι νευρώνες Perceptron είναι από τα πρώτα είδη νευρώνων που αναπτύχθηκαν. Χρησιμοποιούνται για κατηγοριοποίηση παραδειγμάτων, όπως και όλοι οι νευρώνες που εξετάζονται στην παρούσα διπλωματική εργασία. Η τιμή τους παράγεται από αθροίσματα πολλαπλασιασμών μεταξύ εισόδων και βαρών, ενώ η έξοδος τους είναι είτε χαμηλή (-1 ή 0) είτε υψηλή (1).




Εικ. 2.1: Αναπαράσταση νευρώνα γ του νευρωνικού δικτύου της Εικ. 2.2.




Εικ. 2.2: Παράδειγμα πλήρως διασυνδεδεμένου νευρωνικού δικτύου.

Όπως φαίνεται και στην ανάλυση που προηγήθηκε, τα νευρωνικά δίκτυα με πλήρως διασυνδεδεμένα επίπεδα αποτελούνται από επαναλαμβανόμενες δομές πολλαπλασιασμού και συσσώρευσης (*Multiply and ACcumulate – MAC* ). Το γεγονός αυτό καθιστά τις δομή τους αρκετά απλή. Ωστόσο, περεταίρω ανάλυση της υλοποίησής τους οδηγεί σε ορισμένα αρνητικά αποτελέσματα. Για κάθε έναν νευρώνα ο οποίος ακολουθεί επίπεδο **M** νευρώνων, απαιτούνται **M** βάρη και, συνεπώς, **M** πολλαπλασιασμοί και **M** προσθέσεις. Θεωρώντας ότι ο νευρώνας αυτός βρίσκεται σε ένα επίπεδο **N** αντίστοιχων νευρώνων, απαιτούνται συνολικά **M·N** βάρη, πολλαπλασιασμοί και προσθέσεις για το επίπεδο αυτο. Όπως είναι προφανές, το γινόμενο αυτό καθίσταται πολύ μεγάλο σε εκτενή νευρωνικά δίκτυα. Συνεπώς, εγείρονται προβλήματα υλοποίησης, καθώς απαιτείται συχνή μετάδοση σημαντικού όγκου δεδομένων.

Για παράδειγμα, στην παρούσα διπλωματική εργασία εμφανίζεται πλήρες επίπεδο ακόμα και με 70.000 βάρη.

Προκειμένου να αντιμετωπιστεί το πρόβλημα των υψηλών απαιτήσεων μνήμης και συμφόρησης δεδομένων στα κανάλια μεταφοράς, σε συνδυασμό με τα πλήρη επίπεδα, χρησιμοποιούνται τα *συνελικτικά επίπεδα* (*convolutional layers*). Τα τελευταία, αποδεικνύονται εξίσου αποτελεσματικά, απαιτώντας όμως πολύ μικρότερο όγκο ανταλλαγής δεδομένων.

### 2.2.2 Συνελικτικά επίπεδα

Τα συνελικτικά επίπεδα χρησιμοποιούνται ευρέως στον επιστημονικό κλάδο της αναγνώρισης προτύπων, καθώς αποτελούν μία αποτελεσματική εναλλακτική έναντι των πλήρως διασυνδεδεμένων επιπέδων. Λόγω της επαναληψιμότητας των συντελεστών τους (όπως αναλύεται παρακάτω), χαρακτηρίζονται από μικρότερες απαιτήσεις μεταφοράς δεδομένων, σε σύγκριση με τα πλήρως διασυνδεδεμένα επίπεδα.

Τα συνελικτικά επίπεδα βασίζονται στην τεχνική *συνέλιξης* σημάτων [6]. Στην προκειμένη περίπτωση, το ρόλο του σήματος αναλαμβάνει ο *χάρτης γνωρισμάτων*[1] (*feature map*) του επιπέδου, πάνω στον οποίο ολισθαίνονται διάφορα *φίλτρα - πυρήνες* (*kernels*). Σε κάθε *βήμα ολίσθησης* (*stride*), η τιμή του εκάστοτε στοιχείου εξόδου ορίζεται ως το άθροισμα των γινομένων των στοιχείων της εισόδου με τα αντίστοιχα στοιχεία του χρησιμοποιούμενου φίλτρου. Αναλυτικότερα, για έναν συνελικτικό νευρώνα και μία δισδιάστατη είσοδο ορίζονται τα εξής:

- **P** ο χάρτης γνωρισμάτων εισόδου, διαστάσεων **h×w** (Εξ. 2.3),
- **W** το φίλτρο, διαστάσεων **m×n** (Εξ. 2.4), που εφαρμόζεται στον **P** και
- **O** ο χάρτης γνωρισμάτων εξόδου, διαστάσεων **f×e** (Εξ. 2.5). Σημειώνεται ότι τα **f** και **e** δίνονται από τις εξισώσεις Εξ. 2.6 και Εξ. 2.7 αντίστοιχα, όπου θεωρείται βήμα ολίσθησης **s**[2].

[1] Ο χάρτης γνωρισμάτων είναι η ν-διάστατη είσοδος(-έξοδος) συνελικτικού επιπέδου. Ονομάζεται έτσι ώστε να δοθεί έμφαση στο ότι αποτελεί στοιχείο ενός συνόλου τέτοιων ν-διάστατων δομών (συνήθως εικόνων) που αποτελούν την είσοδο(-έξοδο).

[2] Στην παρούσα ανάλυση θεωρείται ότι το βήμα ολίσθησης είναι ίσο και για τις δύο διαστάσεις.

$$P = \begin{pmatrix} p_{0,0} & \cdots & p_{0,w-1} \\ \vdots & \ddots & \vdots \\ p_{h-1,0} & \cdots & p_{h-1,w-1} \end{pmatrix}_{h \times w} \quad \text{Εξ. 2.3}$$

$$W = \begin{pmatrix} w_{0,0} & \cdots & w_{0,n-1} \\ \vdots & \ddots & \vdots \\ w_{m-1,0} & \cdots & w_{m-1,n-1} \end{pmatrix}_{m \times n} \quad \text{Εξ. 2.4}$$

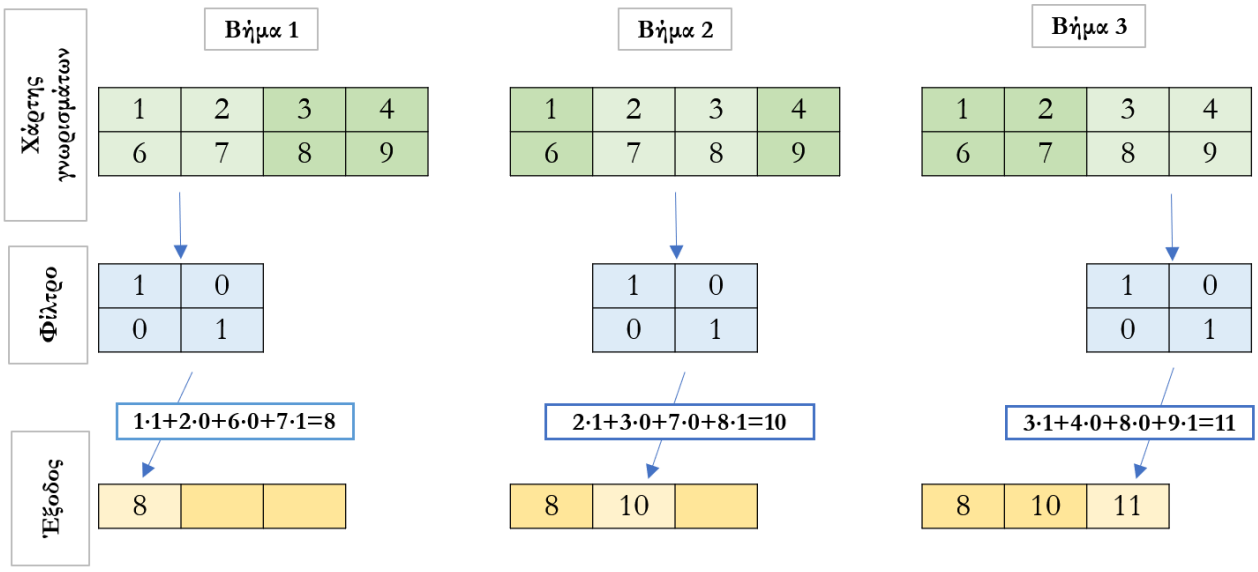
$$O = \begin{pmatrix} o_{0,0} & \cdots & o_{0,e-1} \\ \vdots & \ddots & \vdots \\ o_{f-1,0} & \cdots & o_{f-1,e-1} \end{pmatrix}_{f \times e} \quad \text{Εξ. 2.5}$$

$$f = \frac{h - m + s}{s} \quad \text{Εξ. 2.6}$$

$$e = \frac{w - n + s}{s} \quad \text{Εξ. 2.7}$$

Βάσει των παραπάνω ορισμών, τα στοιχεία του πίνακα γνωρισμάτων εξόδου **O**, υπολογίζονται σύμφωνα με την Εξ. 2.8, όπου η παράμετρος **b**, είναι η αρχική κατάσταση του εκάστοτε νευρώνα, ενώ οι δείκτες **r** και **c** κυμαίνονται στα διαστήματα **[1, f]** και **[1, c]** αντίστοιχα. Στην Εικ. 2.3 αναπαρίσταται με βήματα η λειτουργία ενός συνελικτικού νευρώνα για βήμα ολίσθησης **s=1**.

$$o_{r,c} = f\left(\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} w_{i,j} \cdot p_{(r \cdot s+i),(c \cdot s+j)}\right) + b \quad \text{Εξ. 2.8}$$




Εικ. 2.3: Παράδειγμα συνελικτικού επιπέδου

Σημειώνεται ότι η παραπάνω ανάλυση μπορεί να επεκταθεί και σε συστήματα με περισσότερες εισόδους και για δεδομένα περισσότερων διαστάσεων.

### 2.2.3 Επίπεδα υποδειγματοληψίας μέσης τιμής

Για να περιοριστεί το μέγεθος των ενδιάμεσων σημάτων και να μειωθούν οι απαιτούμενες αριθμητικές πράξεις, τα συνελικτικά επίπεδα συνήθως ακολουθούν ή/και ακολουθούνται από *επίπεδα υποδειγματοληψίας μέσης τιμής* (*average pooling layers*). Ένα επίπεδο αυτού του είδους πραγματοποιεί συμπίεση των ενδιαμέσων χαρτών γνωρισμάτων, συνδυάζοντας πολλά γειτονικά εικονοστοιχεία σε ένα, έχοντας ως τιμή το μέσο όρο τους.

Συγκεκριμένα, θεωρούνται των εξής:

- **P** ο χάρτης γνωρισμάτων εισόδου διαστάσεων **h×w** (Εξ. 2.3) και
- **O** ο χάρτης γνωρισμάτων εξόδου διαστάσεων **f×e** (Εξ. 2.5). Σημειώνεται ότι τα **f** και **e** δίνονται από τις εξισώσεις Εξ. 2.9 και Εξ. 2.10 αντίστοιχα. Θεωρείται ότι το βήμα ολίσθησης **s**[1] είναι ίσο με τις διαστάσεις της γειτονιάς των εικονοστοιχείων εκ των οποίων λαμβάνεται ο μέσος όρος[2].

$$f = h \; div \; s \qquad \text{Εξ. 2.9}$$

$$e = w \; div \; s \qquad \text{Εξ. 2.10}$$

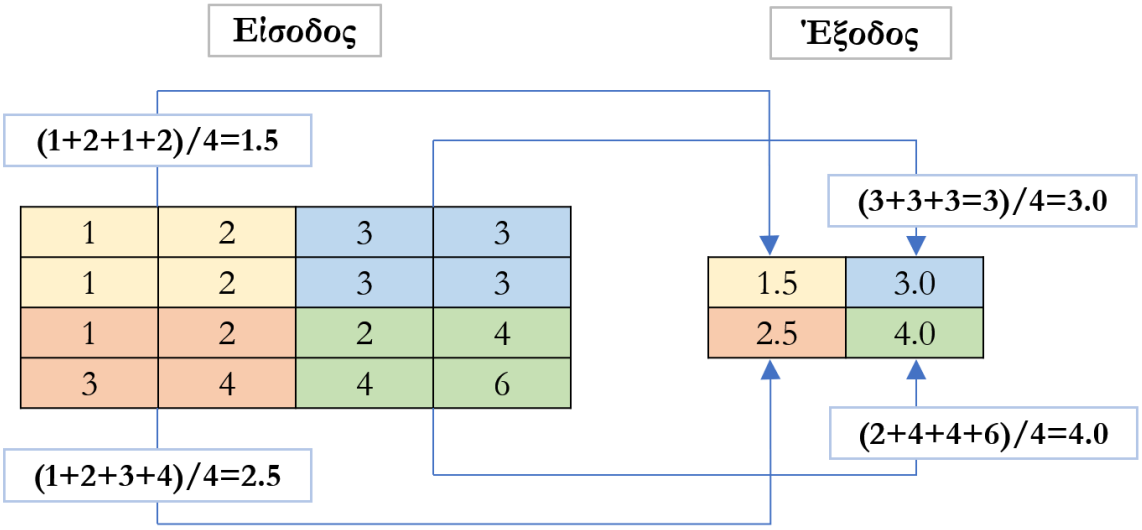
$$o_{r,c} = \left( \sum_{i=0}^{s-1} \sum_{j=0}^{s-1} p_{(r \cdot s+i),(c \cdot s+j)} \right) / \, s^2 \qquad \text{Εξ. 2.11}$$

Για τις εξισώσεις αυτές σημειώνεται ότι οι δείκτες **r** και **c** κυμαίνονται στα διαστήματα **[1, f]** και **[1, c]** αντίστοιχα, ενώ **div** είναι η πράξη της ακέραιας διαίρεσης μεταξύ δύο τελεστέων. Βάσει όλων των παραπάνω, στην Εξ. 2.11 φαίνεται ο υπολογισμός των τιμών των στοιχείων του πίνακα **O**. Ένα αντίστοιχο παράδειγμα παρουσιάζεται στην Εικ. 2.4, όπου εμφανίζεται ένα επίπεδο μέσης υποδειγματοληψίας, το οποίο συμπιέζει έναν δισδιάστατο πίνακα εισόδου, πραγματοποιώντας υποδειγματοληψία με **βήμα 2** και **παράθυρο 2x2**. Φαίνεται ότι ανά τέσσερα στοιχεία εισόδου παράγεται ένα στοιχείο εξόδου. Πράγματι, οι διαστάσεις του

[1] Στην παρούσα ανάλυση θεωρείται ότι το βήμα ολίσθησης είναι ίσο και για τις δύο διαστάσεις.

[2] Η παραδοχή αυτή γίνεται ούτως ώστε οι γειτονιές των εικονοστοιχείων, που λαμβάνονται για τον υπολογισμό του εκάστοτε στοιχείου εξόδου, να είναι μη-επικαλυπτόμενες (βλ. Εικ. 2.4).

πίνακα εξόδου (**2x2**) είναι ίσες με το πηλίκο της ακέραιας διαίρεσης των διαστάσεων εισόδου (**4x4**) με τις διαστάσεις του παραθύρου (**2x2**), βάσει των Εξ. 2.9 και Εξ. 2.10.

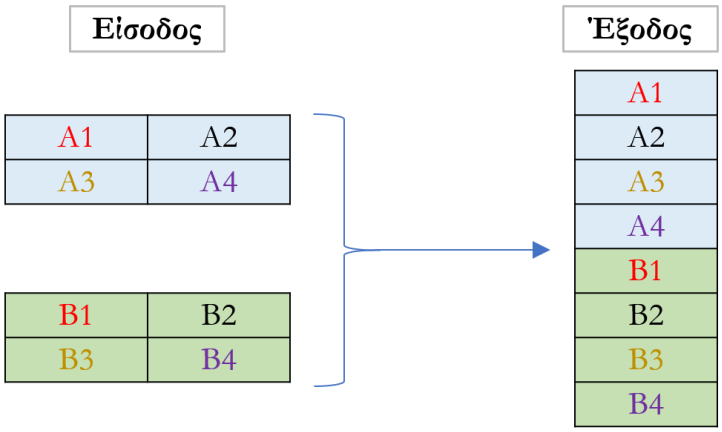



Εικ. 2.4: Παράδειγμα επιπέδου υποδειγματοληψίας

Γενικότερα, υπάρχουν και άλλα είδη επιπέδων υποδειγματοληψίας (όπως αυτά της υποδειγματοληψίας μέγιστης/ελάχιστης τιμής). Ωστόσο, στην παρούσα μελέτη χρησιμοποιείται το αντίστοιχο της μέσης τιμής.

### 2.2.4 Επίπεδα ισοπέδωσης

Τα συνελικτικά επίπεδα αντιμετωπίζουν τα ενδιάμεσα σήματα ως σύνολο πολυδιάστατων διανυσμάτων, ενώ τα πλήρως διασυνδεδεμένα επίπεδα ως μονοδιάστατα. Συνεπώς, απαιτείται μία δομή η οποία να πραγματοποιεί μετασχηματισμό των δεδομένων ανάμεσα στα δύο, σε όλες τις περιπτώσεις που ένα συνελικτικό ακολουθείται από ένα πλήρες επίπεδο. Ένα *επίπεδο ισοπέδωσης* (*flattening layer*) αναδομεί τα δεδομένα, ξεδιπλώνοντας τους χάρτες γνωρισμάτων εξόδου ενός συνελικτικού δικτύου, και μετατρέποντάς τα σε έναν πίνακα μίας διάστασης Εικ. 2.5.




Εικ. 2.5: Μετασχηματισμός δεδομένων που πραγματοποιείται σε ένα επίπεδο ισοπέδωσης.

### 2.2.5 Συναρτήσεις ενεργοποίησης

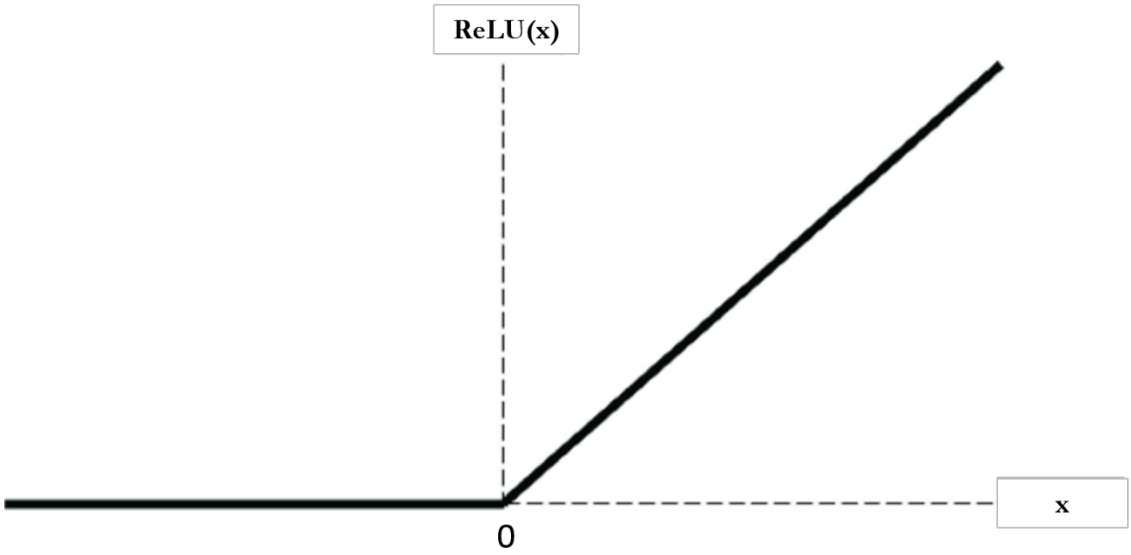
Όπως φάνηκε στην παραπάνω ανάλυση, ορισμένα είδη νευρώνων (π.χ. πλήρως διασυνδεδεμένοι, συνελικτικοί) εφαρμόζουν στην εσωτερική τους κατάσταση κάποια συνάρτηση ενεργοποίησης. Οι συναρτήσεις ενεργοποίησης είναι ειδικές μη-γραμμικές συναρτήσεις και χρησιμοποιούνται έτσι ώστε οι παράμετροι των νευρώνων των νευρωνικών δικτύων να έχουν περιορισμένο εύρος τιμών. Σε αντίθετη περίπτωση, θα ήταν πιθανό το ενδεχόμενο ορισμένοι νευρώνες να εμφανίζουν στην έξοδό τους πολύ μεγαλύτερες τιμές, συγκριτικά με τους υπόλοιπους. Έτσι, οι πρώτοι θα μονοπωλούσαν την επιρροή προς τα επόμενα επίπεδα και το νευρωνικό δίκτυο δεν θα λειτουργούσε όπως αναμένεται. Οι συναρτήσεις ενεργοποίησης συμβάλλουν, δηλαδή, στην ορθή λειτουργία των νευρωνικών επιπέδων.

Παρακάτω παρουσιάζονται δύο μαθηματικές συναρτήσεις που χρησιμοποιούνται συχνά ως συναρτήσεις ενεργοποίησης σε νευρωνικά δίκτυα βαθιάς μάθησης. Οι συγκεκριμένες εφαρμόστηκαν και στην παρούσα διπλωματική εργασία στα πλήρως διασυνδεδεμένα και συνελικτικά επίπεδα.

#### Συνάρτηση ενεργοποίησης ReLU

Η συνάρτηση *ReLU* (*Rectified Linear Unit*) [7] έχει ως έξοδο την τιμή μηδέν, αν η είσοδος είναι μηδέν ή αρνητική, ενώ έχει τιμή ίση με αυτή της εισόδου, όταν η είσοδος είναι θετική. Πιο συγκεκριμένα, εκφράζεται από την εξίσωση Εξ. 2.12, ενώ στην Εικ. 2.6 απεικονίζεται η γραφική της παράσταση.

$$ReLU(x) = \max(0, x)$$ Εξ. 2.12




Εικ. 2.6: Γραφική παράσταση της συνάρτησης ReLU.

Συνάρτηση ενεργοποίησης Softmax

Η συνάρτηση *Softmax* [8], [9] λαμβάνει ένα σύνολο εισόδων και τις κανονικοποιεί εντός του διαστήματος (0,1). Συνεπώς, μπορεί να χρησιμοποιηθεί για την αντιστοίχιση δεδομένων σε πιθανότητες. Για ένα διάνυσμα εισόδων $\mathbf{X}=\{x_1,x_2,\ldots,x_n\}$, η μαθηματική έκφραση που αποδίδει την τιμή της εξόδου για κάθε μία είσοδο $x_i$ ($1 \le i \le n$) δίνεται από την Εξ. 2.13.

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_{i=1}^{n} e^{x_i}}$$ Εξ. 2.13

Η συνάρτηση Softmax έχει την ιδιότητα να διατηρεί τις τιμές εξόδου πάντα σε άθροισμα ίσο με 1. Θεωρώντας τις εξόδους αυτές ως πιθανότητες, η συγκεκριμένη ιδιότητα της Softmax καθιστά τις πιθανότητες αυτές αμοιβαία αποκλειόμενες. Για τον λόγο αυτό, η συγκεκριμένη συνάρτηση χρησιμοποιείται σε προβλήματα όπου κάθε είσοδος πρέπει να κατηγοριοποιηθεί σε μία και μόνο μία κλάση.

### 2.2.6 Εκπαίδευση νευρωνικών δικτύων

Τα νευρωνικά δίκτυα πριν την χρήση τους απαιτούν προσαρμογή των συντελεστών (βαρών) τους, έτσι ώστε να επιτυγχάνουν επιθυμητή ακρίβεια κατηγοριοποίησης. Αυτό πραγματοποιείται μέσω της διαδικασίας της *εκπαίδευσης*, κατά την οποία το νευρωνικό δίκτυο

ενεργοποιείται με δεδομένα εισόδου, για τα οποία η κλάση είναι γνωστή. Έπειτα, οι έξοδοι που παράγονται από το δίκτυο συγκρίνονται με τις κατηγορίες των δεδομένων, έτσι ώστε να ρυθμιστούν κατάλληλα οι συντελεστές του δικτύου, προκειμένου η ακρίβειά του να βελτιωθεί.

Για τον σκοπό αυτό, ένας από τους βασικότερους αλγορίθμους εκπαίδευσης νευρωνικών δικτύων που έχουν αναπτυχθεί, είναι ο *αλγόριθμος της οπισθοδρομικής διάδοσης σφάλματος* (*error backpropagation algorithm*) [10]. Περιγράφει μία επαναληπτική διαδικασία στην οποία γίνεται επαναπροσδιορισμός των βαρών μεταξύ των νευρώνων του δικτύου, με σκοπό την καλύτερη προσαρμογή των εξόδων στις επιθυμητές τιμές. Οδηγεί, δηλαδή, στη βελτίωση της ακρίβειας και μείωση του ποσοστού των σφαλμάτων κατηγοριοποίησης.

Όπως αναφέρθηκε, ο αλγόριθμος της οπισθοδρομικής διάδοσης σφάλματος μεταβάλλει τα βάρη μεταξύ των νευρώνων. Συγκεκριμένα, για τρία διαδοχικά συνδεδεμένα επίπεδα νευρώνων **J**, **I** και **K**, οι εξισώσεις του αλγορίθμου περιγράφονται ως εξής:

- Η μεταβολή του βάρους μεταξύ ενός νευρώνα **i** (του επιπέδου **I**) και ενός νευρώνα **j** (του επιπέδου **J**), $\mathbf{\Delta w_{i,j}}$, περιγράφεται από την Εξ. 2.14. Παρατηρείται, ότι η μεταβολή αυτή είναι ανάλογη της εξόδου, $\mathbf{o_j}$, και ενός συντελεστή, $\boldsymbol{\delta_i}$, που αντιστοιχούν στον νευρώνα **j**, καθώς και του *συντελεστή/ρυθμού εκπαίδευσης* **α**,. Σημειώνεται ότι ο συντελεστής **α** ρυθμίζει την ταχύτητα εκπαίδευσης του νευρωνικού δικτύου.
- Ο συντελεστής **δ** για τους **εσωτερικούς νευρώνες** ορίζεται στην Εξ. 2.15, στην οποία ο δείκτης **k** αφορά τους νευρώνες του επόμενου επιπέδου του δικτύου. Αντίθετα, για τους **νευρώνες εξόδου** ο συντελεστής **δ** ορίζεται από διαφορετική εξίσωση (Εξ. 2.16) και εξαρτάται από το σφάλμα μεταξύ επιθυμητών και πραγματικών εξόδων. Συγκεκριμένα, η παράμετρος $\mathbf{o_i}$ εκφράζει την έξοδο ενός νευρώνα **i**, με $\mathbf{q_i}$ την επιθυμητή τιμή αυτής. Τέλος, $\mathbf{y_i}$ είναι η εσωτερική κατάσταση του αντίστοιχου νευρώνα, ενώ **f(x)** η συνάρτηση ενεργοποίησης.

$$\Delta w_{i,j} = \alpha \cdot \delta_i \cdot o_j \qquad \text{Εξ. 2.14}$$

$$\delta_i = f'(y_i) \cdot \sum_{k=1}^{K} \delta_k \cdot w_{k,i} \qquad \text{Εξ. 2.15}$$

$$\delta_i = -f'(y_i) \cdot (o_i - q_i) \qquad \text{Εξ. 2.16}$$

Η διαδικασία, δηλαδή, ξεκινάει από τους εξωτερικούς νευρώνες και προχωράει σταδιακά, μέσω των εσωτερικότερων, προς τους νευρώνες εισόδου του δικτύου. Σημαντική προσοχή πρέπει να δίνεται τόσο στις επιθυμητές τιμές, $\mathbf{q_i}$, όσο και στον συντελεστή εκπαίδευσης, **α**. Αυτοί καθορίζουν σε μεγάλο βαθμό την επιτυχία της διαδικασίας της εκπαίδευσης και, άρα, τη διακριτική ικανότητα που θα έχει το δίκτυο μετά την ολοκλήρωσή της. Συγκεκριμένα, για τον συντελεστή εκπαίδευσης **α**, αν επιλεχθεί πολύ μεγάλη τιμή, το δίκτυο μπορεί να επέλθει σε κατάσταση αστάθειας. Αντίστοιχα, για πολύ μικρές τιμές του **α**, η διαδικασία της μάθησης καθυστερεί αρκετά και είναι δυνατό το δίκτυο να παγιδευτεί σε ένα τοπικό (και όχι ολικό) ελάχιστο της συνάρτησης σφάλματος. Ως αποτέλεσμα, το δίκτυο θα έχει μη-βέλτιστη ακρίβεια αναγνώρισης κατά την ενεργοποίησή του.

Ένα δεύτερο κομβικό σημείο όσον αφορά την εκπαίδευση ενός νευρωνικού δικτύου είναι το φαινόμενο της *υπερεκπαίδευσης* (*overfitting*) [11]. Το φαινόμενο αυτό παρατηρείται όταν το δίκτυο, κατά τον καθορισμό των συντελεστών του, εκπαιδεύεται πολλές φορές με τα ίδια ζεύγη εισόδου - εξόδου. Έτσι, προσαρμόζεται στην κατηγοριοποίηση αυτών μόνο των δεδομένων και δεν λειτουργεί βάσει κάποιου γενικού κανόνα. Με άλλα λόγια, το δίκτυο δεν μαθαίνει, αλλά αποστηθίζει. Ένας βασικός τρόπος ανίχνευσής του φαινομένου αυτού είναι η ύπαρξη μεγάλης διαφοράς μεταξύ ακρίβειας αναγνώρισης στα σετ εκπαίδευσης και δοκιμών (βλ. Υποπαρ. 2.3.1).

## 2.3 Το νευρωνικό δίκτυο LeNet-5

Το νευρωνικό δίκτυο το οποίο θα υλοποιηθεί σε υλικό στην εργασία αυτή είναι το LeNet-5, το οποίο παρουσιάστηκε για πρώτη φορά στην [12]. Ωστόσο, η υλοποίηση της παρούσας διπλωματικής εργασίας διαφέρει ελαφρώς και αποτελεί παραλλαγή του νευρωνικού δικτύου της [12]. Γενικότερα, το LeNet-5[1] αποτελεί ένα από τα πρώτα και βασικότερα συνελικτικά νευρωνικά δίκτυα και χρησιμοποιείται ακόμη και σήμερα ως σημείο αναφοράς για τη σύγκριση των επιδόσεων διαφορετικών νευρωνικών δικτύων και υλοποιήσεων σε υλικό. Στόχος του είναι η ορθή αναγνώριση εικόνων που περιέχουν χειρόγραφα ψηφία.

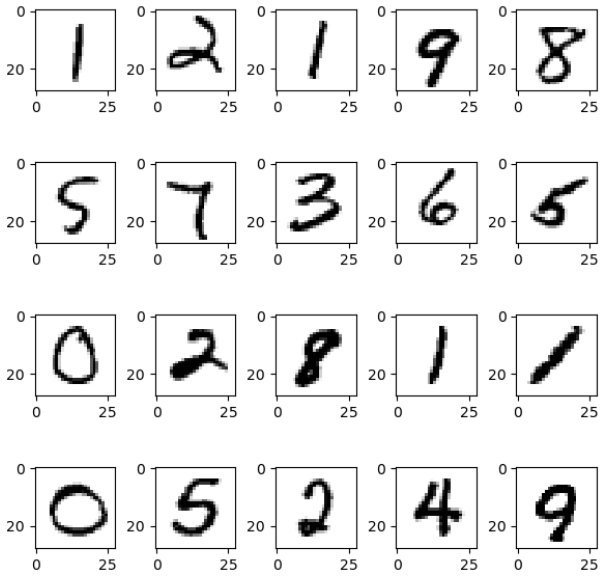
[1] Η κατάληξη -5 προκύπτει από το γεγονός ότι το δίκτυο αποτελείται από 5 επίπεδα με παραμέτρους προς εκπαίδευση (βάρη). Υπάρχουν και αρχιτεκτονικές του LeNet με διαφορετικό αριθμό επιπέδων. Για παράδειγμα, το LeNet-4 αποτελείται από 2 πλήρως διασυνδεδεμένα επίπεδα αντί για 3 που διαθέτει το LeNet-5.

### 2.3.1 Δεδομένα

Τα δεδομένα που χρησιμοποιήθηκαν τόσο ως είσοδοι, όσο και ως μέτρο αξιολόγησης του νευρωνικού δικτύου αντλήθηκαν από τη βάση δεδομένων MNIST (Modified NIST) [13]. Η τελευταία αποτελεί τροποποίηση της βάσης δεδομένων NIST και περιέχει ανακατεμένες εικόνες χειρόγραφων ψηφίων (Εικ. 2.7) περισσότερων από 250 ανθρώπων. Οι εικόνες αυτές έχουν κανονικοποιηθεί προκειμένου να είναι στο κέντρο ενός πλαισίου 28x28 εικονοστοιχείων διατηρώντας, ωστόσο, τον αρχικό λόγο ύψους και πλάτους.

Η βάση δεδομένων MNIST αποτελείται από 60.000 εικόνες εκπαίδευσης (*σετ εκπαίδευσης*) και 10.000 εικόνες δοκιμής (*σετ δοκιμών*). Το πρώτο σετ εικόνων, όπως υποδεικνύει και το όνομά του, χρησιμοποιείται για την εκπαίδευση του νευρωνικού δικτύου. Η ποιότητα της εκπαίδευσής του, δηλαδή η επίδοσή του, ελέγχεται από τις 10.000 εικόνες του σετ δοκιμών. Επομένως, ορίζονται δύο ακρίβειες αναγνώρισης: *ακρίβεια εκπαίδευσης* και *ακρίβεια δοκιμής*.

Τέλος, αξίζει να σημειωθεί πως οι εικόνες της βάσης MNIST είναι ασπρόμαυρες, με αποτέλεσμα να απαιτείται μόνο ένα κανάλι για την επεξεργασία τους από τα συνελικτικά επίπεδα του LeNet-5. Αν οι εικόνες ήταν έγχρωμες (RGB), τότε θα απαιτούνταν 3 κανάλια (1 για κάθε χρώμα) από τα συνελικτικά επίπεδα. Το τελευταίο θα είχε ως συνέπεια τη χρήση τριπλάσιων παραμέτρων εκπαίδευσης στα συνελικτικά επίπεδα (βλ. Παρ. 2.3.2). Είναι εμφανές, λοιπόν, ότι ακόμα και η φύση των δεδομένων έχει αντίκτυπο στις απαιτήσεις μνήμης και χρόνου του νευρωνικού δικτύου.



Εικ. 2.7: Εικόνες χειρόγραφων ψηφίων του σετ εικόνων MNIST.

### 2.3.2 Αρχιτεκτονική

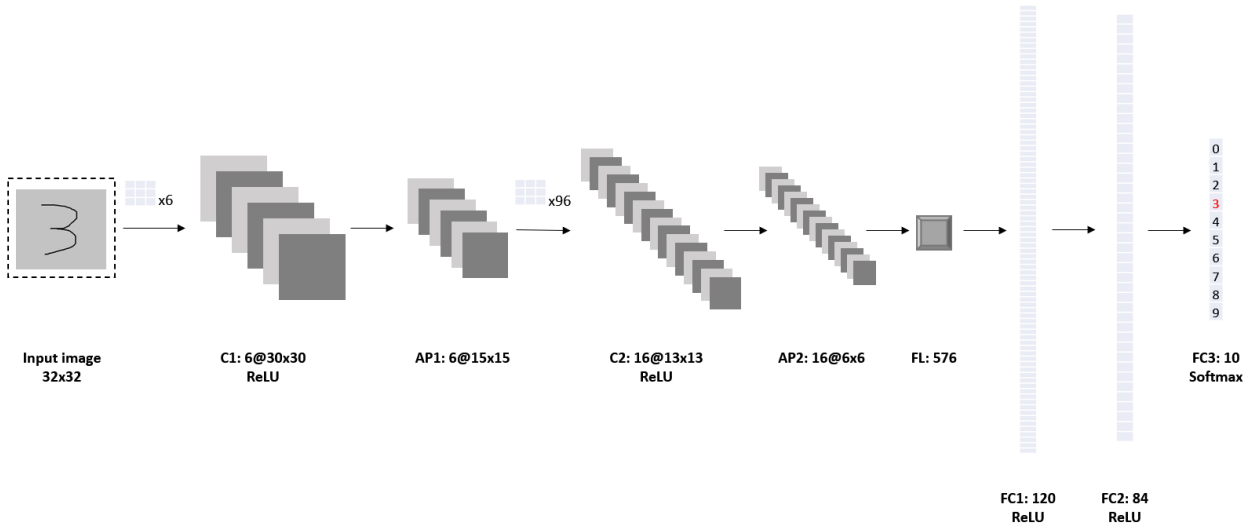
Όπως φαίνεται στην Εικ. 2.8, οι **εικόνες εισόδου** του LeNet-5 είναι διαστάσεων **32x32**, αντί για **28x28** που αναφέρθηκε στην Παρ. 2.3.1. Συγκεκριμένα, οι εικόνες υπόκεινται σε *γέμισμα με μηδενικά* (*zero padding*) μέχρι να αποκτήσουν μέγεθος 32x32 εικονοστοιχείων. Αυτό γίνεται διότι η εφαρμογή της δισδιάστατης συνέλιξης στο πρώτο επίπεδο του LeNet-5 (το οποίο είναι συνελικτικό) μικραίνει τη διάσταση της εικόνας εισόδου (βλ. Εξ. 2.6 και Εξ. 2.7), καθώς δε μπορεί να εφαρμοστεί στα όρια της.

Το πρώτο επίπεδο του LeNet-5 (**C1**), όπως αναφέρθηκε στην προηγούμενη παράγραφο, είναι συνελικτικό και αποτελείται από **6 πυρήνες** διαστάσεων **3x3**. Αυτό έχει σαν αποτέλεσμα τη χρήση **3x3+1=10 βαρών**[1] προς εκπαίδευση ανά πυρήνα. Σε κάθε εικονοστοιχείο της εικόνας εφαρμόζεται το **παράθυρο 3x3**. Ως εκ τούτου, είναι εμφανές πως η εφαρμογή του παραθύρου στα εικονοστοιχεία περιμετρικά της εικόνας θα ανάγκαζε το παράθυρο να βγει εκτός αυτής. Για τον λόγο αυτό, το παράθυρο εφαρμόζεται μόνο στα εσωτερικά εικονοστοιχεία, με αποτέλεσμα οι διαστάσεις του κάθε χάρτη γνωρισμάτων να ελαττώνονται σε **30x30**. Τέλος, ως συνάρτηση ενεργοποίησης χρησιμοποιείται η ReLU.

Κατόπιν, ακολουθεί ένα επίπεδο μέσης υποδειγματοληψίας **2x2** (**AP1**), το οποίο υπολογίζει τον μέσο όρο μιας **γειτονιάς 2x2**. Οι γειτονιές αυτές είναι μη επικαλυπτόμενες, με συνέπεια τη μείωση του μεγέθους κάθε χάρτη γνωρισμάτων στο μισό (από **30x30** σε **15x15**). Η λειτουργία του επιπέδου αυτού έγκειται μόνο στην υποδειγματοληψία του χάρτη γνωρισμάτων και, ως εκ τούτου, δε διαθέτει παραμέτρους προς εκπαίδευση.

Στο σημείο αυτό, αξίζει να σημειωθεί ότι η χρήση ζευγών συνελικτικών επιπέδων ακολουθούμενων από επίπεδα υποδειγματοληψίας είναι συχνή στα συνελικτικά νευρωνικά δίκτυα. Τα συνελικτικά επίπεδα έχουν στόχο την εξαγωγή των χαρακτηριστικών μιας εικόνας (π.χ. ακμές, γωνίες κλπ.). Ωστόσο, υπάρχει ο κίνδυνος το νευρωνικό δίκτυο να προσαρμοστεί στις θέσεις που βρίσκονται τα χαρακτηριστικά αυτά και ως εκ τούτου η κατηγοριοποίηση να μη βασίζεται σε κάποιο γενικό κανόνα. Με άλλα λόγια, το δίκτυο υφίσταται υπερεκπαίδευση (βλ. Υποπαρ. 2.2.6). Σκοπός, λοιπόν, των επιπέδων μέσης υποδειγματοληψίας είναι η αποφυγή της υπερεκπαίδευσης.

[1] Απαιτούνται 3x3 βάρη για τον πυρήνα (διαστάσεων 3x3), συν μία σταθερά αρχικοποίησης.




Εικ. 2.8: Η αρχιτεκτονική του νευρωνικού δικτύου LeNet-5.

Στη συνέχεια, ακολουθεί άλλο ένα ζεύγος επιπέδων συνέλιξης (**C2**) και μέσης υποδειγματοληψίας (**AP2**). Το **C2** πραγματοποιεί εξαγωγή χαρακτηριστικών μεγαλύτερης λεπτομέρειας (σε σχέση με το **C1**), ενώ χρησιμοποιεί επίσης τη συνάρτηση ReLU. Αυτή τη φορά χρησιμοποιούνται **96 πυρήνες** διαστάσεων **3x3** (16 νευρώνες με τον καθένα να διαθέτει έναν πυρήνα για καθεμία από τις 6 εισόδους του) με αποτέλεσμα τη δημιουργία **16 χαρτών γνωρισμάτων** μεγέθους **13x13**. Έπειτα, το **AP2** μειώνει το μέγεθος των ενδιάμεσων εικόνων στο μισό (μέγεθος 6x6), ή ακριβέστερα στο πηλίκο της ακέραιας διαίρεσης με το 2 (επειδή οι διαστάσεις του προηγούμενου είναι περιττοί αριθμοί). Αυτό συμβαίνει προκειμένου να αποφευχθεί η διαφυγή του παραθύρου υποδειγματοληψίας από τα όρια του ενδιάμεσου χάρτη γνωρισμάτων. Επί της ουσίας, λοιπόν, αγνοούνται η τελευταία γραμμή και στήλη.

Τέλος, το LeNet-5 διαθέτει 3 πλήρως διασυνδεδεμένα επίπεδα (**FC1**, **FC2**, **FC3**). Οι ενδιάμεσοι χάρτες γνωρισμάτων οργανώνονται με τέτοιο τρόπο, ούτως ώστε οι 3 διαστάσεις τους να μετατραπούν σε 1. Συγκεκριμένα, οι **16 χάρτες γνωρισμάτων** μεγέθους **6x6** μετατρέπονται σε έναν διάνυσμα **576** στοιχείων (16x6x6=576) μέσω ενός επιπέδου ισοπέδωσης (**FL**). Σύμφωνα με την Εικ. 2.8, τα **FC1** και **FC2** αποτελούνται από **120** και **84 νευρώνες** αντίστοιχα, ενώ κάνουν χρήση της συνάρτησης ενεργοποίησης ReLU. Το **FC3**, ωστόσο, παρουσιάζει μικρή υπολογιστική επιβάρυνση διαθέτοντας μόλις **10 νευρώνες**. Η επιλογή αυτή γίνεται, διότι το επίπεδο αυτό δίνει την τελική πληροφορία κατηγοριοποίησης (εφόσον είναι το επίπεδο εξόδου του δικτύου). Συνεπώς, το πλήθος των νευρώνων του θα

πρέπει να ισούται με τον αριθμό των διαφορετικών κατηγοριών που πρέπει να αναγνωριστούν (10 χειρόγραφα ψηφία).

Επιπλέον, παρατηρείται η χρήση της συνάρτησης ενεργοποίησης Softmax στο επίπεδο εξόδου (FC3). Όπως αναφέρεται και στην Υποπαρ. 2.2.5, η Softmax κανονικοποιεί την έξοδο του κάθε νευρώνα στο διάστημα (0,1), με τις επιμέρους εξόδους να έχουν πάντα άθροισμα ίσο με 1, μετατρέποντάς την καθεμία σε μια πιθανότητα. Η εκάστοτε πιθανότητα εκφράζει, επί της ουσίας, το ποσοστό βεβαιότητας του δικτύου για τη συγκεκριμένη κατηγορία και είσοδο. Για παράδειγμα, μια τιμή 0.901 στην έξοδο 0 συνεπάγεται 90.1% βεβαιότητα αναγνώρισης της εικόνας εισόδου ως 0.

Συγκεντρωτικά, τα επίπεδα που δομούν το νευρωνικό δίκτυο LeNet-5 της παρούσας διπλωματικής εργασίας φαίνονται στον Πίν. 2.1.

Πίν. 2.1: Επιδόσεις ανά σετ του εκπαιδευμένου LeNet-5.

| **Επίπεδο** | **Διαστάσεις εισόδου** | **Αριθμός βαρών** | **Διαστάσεις εξόδου** |
|---|---|---|---|
| Συνελικτικό 1 (**C1**) | 1x32x32 | 60 | 6x30x30 |
| Μέσης υποδειγματοληψίας 1 (**AP1**) | 6x30x30 | 0 | 6x15x15 |
| Συνελικτικό 2 (**C2**) | 6x15x15 | 880 | 16x13x13 |
| Μέσης υποδειγματοληψίας 2 (**AP2**) | 16x13x13 | 0 | 16x6x6 |
| Επίπεδο ισοπέδωσης (**FL**) | 16x6x6 | 0 | 1x1x576 |
| Πλήρως διασυνδεδεμένο 1 (**FC1**) | 1x1x576 | 69240 | 1x1x120 |
| Πλήρως διασυνδεδεμένο 2 (**FC2**) | 1x1x120 | 10164 | 1x1x84 |
| Πλήρως διασυνδεδεμένο 3 (**FC3**) | 1x1x84 | 850 | 1x1x10 |

# 3. ΥΛΟΠΟΙΗΣΗ ΤΟΥ LENET-5 ΣΕ ΛΟΓΙΣΜΙΚΟ

## 3.1 Εισαγωγή

Το νευρωνικό δίκτυο LeNet-5 έχει ως στόχο την ορθή κατηγοριοποίηση των εικόνων του σετ δεδομένων MNIST. Ως εκ τούτου, είναι απαραίτητη η εκπαίδευση του LeNet-5 και η εξαγωγή κατάλληλων τιμών για τα βάρη του, πριν την χρήση του. Επειδή η διαδικασία αυτή δεν είναι εύκολα υλοποιήσιμη απευθείας σε σχεδιασμό υλικού, κρίνεται αναγκαία η ανάπτυξη και η εκπαίδευση του LeNet-5 πρώτα σε επίπεδο λογισμικού. Ως αποτέλεσμα των παραπάνω, ένα σημαντικό μέρος της παρούσας εργασίας αποτέλεσε η μετέπειτα μεταφορά της αρχιτεκτονικής του LeNet-5 σε κώδικα C/C++ (έτσι ώστε να μεταφραστεί τελικά σε μία υλοποίηση σε υλικό, όπως αναλύεται στο Κεφ. 4). Τα δυσκολότερα σημεία της διαδικασίας αυτής αποτέλεσαν η εξαγωγή των βαρών από το μοντέλο λογισμικού, προκειμένου να επιτυγχάνεται η ίδια ακρίβεια αναγνώρισης, και η εξασφάλιση της σωστής λειτουργικότητας.

Για την υλοποίηση του νευρωνικού δικτύου LeNet-5 σε λογισμικό έγινε χρήση της βιβλιοθήκης ανάπτυξης και εκπαίδευσης νευρωνικών δικτύων Keras της γλώσσας Python. Το Keras βασίζεται στο Tensorflow το οποίο αποτελεί επίσης μια βιβλιοθήκη σχεδιασμού νευρωνικών δικτύων. Το Keras, ωστόσο, είναι πιο απλό στη χρήση του, αν και έχει

περιορισμένες δυνατότητες σε σχέση με το Tensorflow. Παρόλα αυτά, είναι αρκετό για τον σκοπό της παρούσας διπλωματικής εργασίας.

Το κεφάλαιο ξεκινά με την ανάλυση του μοντέλου λογισμικού του LeNet-5 που αναπτύχθηκε σε Keras. Στη συνέχεια, περιγράφονται οι λεπτομέρειες και οι παράμετροι εκπαίδευσης του νευρωνικού δικτύου. Τέλος, παρουσιάζεται η διαδικασία που ακολουθήθηκε για την εξαγωγή των βαρών του εκπαιδευμένου πλέον δικτύου και την χρήση τους στο τελικό μοντέλο λογισμικού.

## 3.2 Υλοποίηση του LeNet-5 στο Keras

Η υλοποίηση του LeNet-5 σε Keras βασίστηκε σε αυτή της [14]. Αρχικά, πραγματοποιείται το κατέβασμα και η αποσυμπίεση του σετ εικόνων MNIST. Ωστόσο, οι εικόνες αυτές γεμίζουν περιμετρικά με μηδενικά (βλ. Παρ. 2.3), προκειμένου να αυξηθεί το μέγεθός τους από 28x28 σε 32x32 (Εικ. 3.1). Κατόπιν, ορίζεται το μοντέλο του νευρωνικού δικτύου, όπως παρουσιάζεται στην Εικ. 3.2.

```
#2. Lenet used 32x32 images, so we need to apply zero padding to our 28x28 images
train['features']      = np.pad(train['features'], ((0,0),(2,2),(2,2),(0,0)), 'constant')
validation['features'] = np.pad(validation['features'], ((0,0),(2,2),(2,2),(0,0)), 'constant')
test['features']       = np.pad(test['features'], ((0,0),(2,2),(2,2),(0,0)), 'constant')
```

Εικ. 3.1: Γέμισμα με μηδενικά των εικόνων εισόδου.

```
# Definition of LeNet-5 Neural Network
def build_lenet():
    model = keras.Sequential()

    model.add(layers.Conv2D(filters=6, kernel_size=(3, 3),
                            activation='relu', input_shape=(32,32,1)))  # Convolutional Layer 1
    model.add(layers.AveragePooling2D())                                # Average Pooling Layer 1
    model.add(layers.Conv2D(filters=16, kernel_size=(3, 3),
                            activation='relu'))                         # Convolutional Layer 2
    model.add(layers.AveragePooling2D())                                # Average Pooling Layer 2
    model.add(layers.Flatten())                                         # Flatenning Layer
    model.add(layers.Dense(units=120, activation='relu'))               # Fully Connected Layer 1
    model.add(layers.Dense(units=84, activation='relu'))                # Fully Connected Layer 2
    model.add(layers.Dense(units=10, activation = 'softmax'))           # Fully Connected Layer 3
    model.summary()
    model.compile(loss=keras.losses.categorical_crossentropy,
                  optimizer=keras.optimizers.Adam(), metrics=['accuracy'])
    return model
```

Εικ. 3.2: Το μοντέλο του LeNet-5 σε Keras που αναπτύχθηκε.

Κατά τον ορισμό του μοντέλου (Εικ. 3.2), χρησιμοποιούνται κατάλληλα αντικείμενα του Keras για τον ορισμό των επιπέδων του δικτύου. Τα αντικείμενα αυτά, τα οποία χρησιμοποιήθηκαν ώστε να οριστεί επακριβώς η αρχιτεκτονική που παρουσιάστηκε στην Υποπαρ. 2.3.2, ορίζονται ως εξής:

- Conv2D: Δομή που περιγράφει ένα συνελικτικό επίπεδο 2 διαστάσεων (βλ. Υποπαρ. 2.2.2).
- AveragePooling2D: Δομή που ορίζει ένα επίπεδο μέσης υποδειγματοληψίας 2 διαστάσεων (βλ. Υποπαρ. 2.2.3).
- Flatten: Δομή που αναπαριστά ένα επίπεδο ισοπέδωσης (βλ. Υποπαρ. 2.2.4).
- Dense: Δομή που αντιστοιχεί σε πλήρως διασυνδεδεμένο επίπεδο (βλ. Υποπαρ. 2.2.1).

Μέσω κατάλληλης συνάρτησης, το πρόγραμμα τυπώνει ως έξοδο την δομή του μοντέλου που υλοποιήθηκε (Εικ. 3.3). Παρατηρείται ότι η δομή αυτή συμφωνεί πλήρως με την αρχιτεκτονική που περιεγράφηκε με τον κώδικα σε Keras.

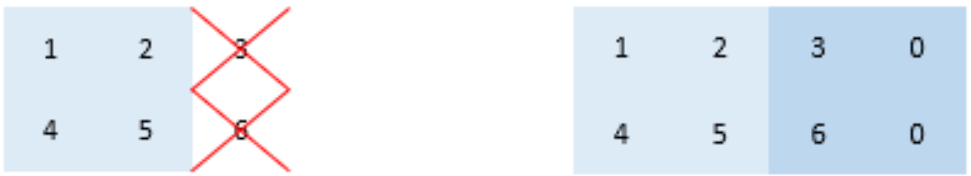
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 30, 30, 6)         60
_________________________________________________________________
average_pooling2d_1 (Average (None, 15, 15, 6)         0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 13, 13, 16)        880
_________________________________________________________________
average_pooling2d_2 (Average (None, 6, 6, 16)          0
_________________________________________________________________
flatten_5 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_13 (Dense)             (None, 120)               69240
_________________________________________________________________
dense_14 (Dense)             (None, 84)                10164
_________________________________________________________________
dense_15 (Dense)             (None, 10)                850
=================================================================
Total params: 81,194
Trainable params: 81,194
Non-trainable params: 0
_________________________________________________________________
```

Εικ. 3.3: Αρχιτεκτονική και αριθμός βαρών του LeNet-5, όπως προέκυψαν από το Keras.

Σε αυτό το σημείο είναι σημαντικό να αναφερθούν ορισμένες παρατηρήσεις όσον αφορά τον τρόπο λειτουργίας του Keras. Παρότι δε φαίνεται άμεσα στον κώδικα της Εικ. 3.2, τόσο

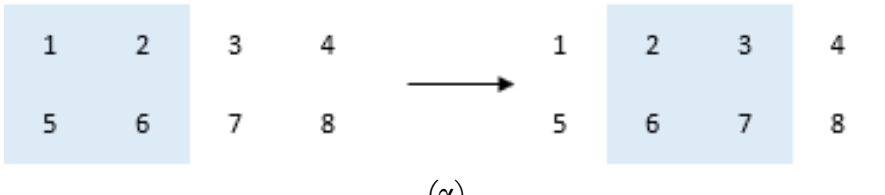
στα συνελικτικά επίπεδα, όσο και στα επίπεδα μέσης υποδειγματοληψίας χρησιμοποιείται *μη γέμισμα* (*valid padding*) των χαρτών γνωρισμάτων. Αυτό σημαίνει πως όταν ο πυρήνας ή το παράθυρο υποδειγματοληψίας βρίσκεται στα άκρα της εικόνας, οι τιμές οι οποίες περισσεύουν αγνοούνται. Αντίθετη περίπτωση αποτελεί το γέμισμα με μηδενικά (βλ. Υποπαρ. 2.3.2), κατά την οποία πραγματοποιείται κατάλληλη συμπλήρωση μηδενικών, προκειμένου να μην αγνοηθεί καμία τιμή. Η διαφορά των δύο αυτών μεθόδων γίνεται εμφανέστερη στην Εικ. 3.4.

Μια εξίσου σημαντική λεπτομέρεια που δεν είναι εμφανής, είναι η χρήση βήματος ολίσθησης ίσο με 1 από τα συνελικτικά επίπεδα και βήματος 2 από τα επίπεδα μέσης υποδειγματοληψίας. Το πρώτο συνεπάγεται μετακίνηση κάθε φορά κατά 1 θέση, τόσο στον άξονα x, όσο και στον y, ενώ το τελευταίο μετακίνηση κατά 2 θέσεις. Η διαφορά γίνεται περισσότερο κατανοητή στην Εικ. 3.5.




(α) (β)

Εικ. 3.4: Σάρωση εικόνας με παράθυρο 2x2 και βήμα 2, (α) μη γέμισμα και (β) γέμισμα με μηδενικά.




(α)

(β)

Εικ. 3.5: Σάρωση εικόνας με παράθυρο 2x2 με (α) βήμα 1 και (β) βήμα 2.

## 3.3 Εκπαίδευση του μοντέλου

Όπως αναφέρθηκε και στην Υποπαρ. 2.3.1, το σετ εικόνων MNIST αποτελείται από δύο μέρη: το σετ εκπαίδευσης και το σετ δοκιμών. Όπως υποδεικνύουν και τα ονόματά τους, το πρώτο χρησιμοποιείται για την εκπαίδευση του νευρωνικού δικτύου, ενώ το δεύτερο για τον έλεγχο των επιδόσεών του, εκπαιδευμένου πλέον, δικτύου σε άγνωστα δεδομένα. Ωστόσο, ένα μέρος του σετ εκπαίδευσης και συγκεκριμένα το 20% χρησιμοποιείται ως μέσο ανατροφοδότησης της διαδικασίας εκπαίδευσης, καθοδηγώντας τη *συνάρτηση σφάλματος* (*loss function*). Το σετ αυτό καλείται *σετ ελέγχου* (*validation set*) και η βασική διαφορά του με το σετ δοκιμών είναι ότι εφαρμόζεται κατά τη διάρκεια της εκπαίδευσης, αντί για μετά από αυτή.

Ως συνάρτηση σφάλματος χρησιμοποιήθηκε η *συνάρτηση κατηγορικής έτερο-εντροπίας* (*categorical cross-entropy function*). Απαραίτητη προϋπόθεση για τη χρήση της είναι τα δεδομένα να ανήκουν σε μία και μόνο μία κατηγορία. Με άλλα λόγια πρέπει οι κατηγορίες του σετ δεδομένων να είναι αμοιβαία αποκλειόμενες. Είναι προφανές πως στο MNIST η συνθήκη αυτή πληρείται, αφού κάθε εικόνα απεικονίζει μόνο ένα χειρόγραφο ψηφίο, καθιστώντας την συνάρτηση κατηγορικής έτερο-εντροπίας πολύ καλή επιλογή.

Για κάθε εικόνα, οι πραγματικές πιθανότητές της ορίζονται ως ένα διάνυσμα διάστασης 10, το οποίο περιέχει 1 στην ορθή κλάση της εικόνας και 9 μηδενικά σε όλες τις άλλες[1]. Η συνάρτηση κατηγορικής έτερο-εντροπίας συγκρίνει τις πραγματικές πιθανότητες με τις εξόδους του νευρωνικού δικτύου. Οι τελευταίες εκφράζουν την πιθανότητα η εικόνα εισόδου να ανήκει στην αντίστοιχη κλάση (0, 1, 2 κλπ.). Στόχος, λοιπόν, της συνάρτησης αυτής είναι η ποσοτικοποίηση της απόστασης των εξόδων του δικτύου από τις αναμενόμενες. Συνεπώς, το πρόβλημα της εκπαίδευσης μεταφράζεται σε πρόβλημα ελαχιστοποίησης (πρόβλημα βελτιστοποίησης) της συνάρτησης σφάλματος. Ένα παράδειγμα των παραπάνω απεικονίζεται στην Εικ. 3.6.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 0.001 | 0 | 0.993 | 0 | 0 | 0 | 0 | 0 | 0.004 | 0.002 |

(α)

[1] Η κωδικοποίηση αυτή ονομάζεται one-hot encoding στη βιβλιογραφία.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

(β)

Εικ. 3.6: (α) Οι πιθανότητες πρόβλεψης του LeNet-5 και (β) οι πραγματικές (ιδανικές) πιθανότητες όταν ως εικόνα εισόδου δίνεται το ψηφίο 2.

Όπως φαίνεται και στην Εικ. 3.2 (προτελευταία γραμμή του κώδικα), για την εκπαίδευση του μοντέλου στο Keras ως *αλγόριθμος εκπαίδευσης* επιλέχθηκε ο αλγόριθμος Adam [15]. Πρόκειται για έναν αλγόριθμο με *προσαρμοστικό ρυθμό εκπαίδευσης* (*adaptive learning rate*), πράγμα που σημαίνει ότι ο αλγόριθμος εκπαίδευσης δεν είναι σταθερός, αλλά μεταβάλλεται κατά τη διάρκεια της εκπαίδευσης του δικτύου. Ο αλγόριθμος Adam είναι ο κατεξοχήν αλγόριθμος που χρησιμοποιείται για την εκπαίδευση μοντέλων βαθιάς μάθησης, εξαιτίας της υψηλής ταχύτητας σύγκλισής του. Στην παρούσα διπλωματική εργασία, ως παράμετροι του Adam χρησιμοποιήθηκαν αυτές που προτείνονται από τους συγγραφείς του [15], δηλαδή:

- **$\alpha$=0.001**: ρυθμός εκπαίδευσης,
- **$\beta_1$=0.9** και **$\beta_2$=0.999**: εκθετικοί ρυθμοί μείωσης της εκτίμησης των ροπών (παράμετροι του αλγορίθμου).

Η εκπαίδευση του νευρωνικού δικτύου πραγματοποιείται με τον κώδικα της Εικ. 3.7. Συγκεκριμένα, χρησιμοποιώντας *πακέτα εικόνων* μεγέθους 128 (batch size=128) και διασχίζοντας το πλέον χωρισμένο σετ εκπαίδευσης 10 φορές (epochs=10), λαμβάνονται τα αποτελέσματα που παρουσιάζονται στον Πίν. 3.1. Τα δεδομένα επιβεβαιώνουν το γεγονός ότι το δίκτυο συμπεριφέρεται καλύτερα στο σετ εκπαίδευσης σε σχέση με το σετ δοκιμών. Τέλος, παρατηρείται ότι το σετ ελέγχου προσομοιάζει αρκετά καλά τις επιδόσεις του LeNet-5 σε άγνωστα δεδομένα (σετ δοκιμών).

Πίν. 3.1: Επιδόσεις ανά σετ του εκπαιδευμένου LeNet-5. Το μέσο σφάλμα καθενός σετ αντιστοιχεί στη τιμή που προκύπτει από τη συνάρτηση σφάλματος.

| **Σετ** | **Ακρίβεια αναγνώρισης (%)** | **Μέσο σφάλμα** |
|---|---|---|
| Εκπαίδευσης | 99.47 | 0.0156 |
| Ελέγχου | 98.49 | 0.0811 |
| Δοκιμών | 98.72 | 0.0444 |

```
#4. Trainning the network
#to_categorical = one_hot encoding
X_train, y_train = train['features'],
        to_categorical(train['labels'])
X_validation, y_validation = validation['features'],
        to_categorical(validation['labels'])

train_generator = ImageDataGenerator().flow(X_train,
                        y_train, batch_size=BATCH_SIZE)
validation_generator = ImageDataGenerator().flow(X_validation,
                        y_validation, batch_size=BATCH_SIZE)

steps_per_epoch = X_train.shape[0]//BATCH_SIZE
validation_steps = X_validation.shape[0]//BATCH_SIZE
model.fit_generator(train_generator, steps_per_epoch=steps_per_epoch,
                    epochs=EPOCHS, validation_data=validation_generator,
                    validation_steps=validation_steps, shuffle=True)
```

Εικ. 3.7: Κώδικας εκπαίδευσης του νευρωνικού δικτύου στο Keras.

## 3.4 Εξαγωγή βαρών

Όπως αναφέρθηκε και στην Παρ 3.1, για να πραγματοποιηθεί η μετάβαση από την παραπάνω υλοποίηση λογισμικού σε αντίστοιχη σε υλικό[1], χρειάζεται να αναπαραχθεί τόσο η αρχιτεκτονική του δικτύου όσο και οι τιμές των βαρών του. Συγκεκριμένα, το νευρωνικό δίκτυο LeNet-5 διαθέτει περίπου 80 χιλιάδες βάρη (βλ. Εικ. 3.3), τα οποία πρέπει να μεταφερθούν στο C/C++ μοντέλο.

Αρχικά, χρησιμοποιώντας τις ρουτίνες που διαθέτει το Keras, πραγματοποιήθηκε εξαγωγή της αρχιτεκτονικής του νευρωνικού δικτύου σε μορφή JSON και των βαρών του σε αρχείο «.h5» (Εικ. 3.8). Έπειτα, αναπτύχθηκε κατάλληλο πρόγραμμα σε Python με σκοπό το διάβασμα των δύο παραπάνω αρχείων και την αποθήκευση των δεδομένων που περιέχουν σε τέτοια μορφή, ώστε να είναι ευκολότερη η αυτοματοποίηση της προσπέλασής τους (Εικ. 3.9). Τέλος, με αντίστοιχο κώδικα, πραγματοποιήθηκε προσπέλαση του νέου αρχείου για τη δημιουργία ενός αρχείου «.h», προκειμένου να αναγνωρίζεται από γλώσσα C/C++. Στο αρχείο αυτό αποθηκεύτηκαν σε μορφή πινάκων όλα τα βάρη και οι διαστάσεις των επιπέδων του LeNet-5 (Εικ. 3.10). Το αρχείο «.h» χρησιμοποιήθηκε στο επόμενο στάδιο της παρούσας

[1] Υπενθυμίζεται ότι ο λόγος που αναπτύχθηκε πρώτα σχεδίαση σε λογισμικό είναι επειδή έτσι διευκολύνθηκε η διαδικασία της εκπαίδευσης.

διπλωματικής εργασίας, δηλαδή αυτό της υλοποίησης του νευρωνικού δικτύου LeNet-5 σε γλώσσα C/C++ και, τελικά, σε γλώσσα περιγραφής υλικού (Κεφ. 4).

```
#6. Saving all data of the model
model.save(DESTINATION_FOLDER + 'LeNet.h5')
f = open(DESTINATION_FOLDER + 'LeNet_architecture.json','w')
f.write(model.to_json())
f.close()
model.save_weights(DESTINATION_FOLDER + 'LeNet_weights.h5')
```

Εικ. 3.8: Αποθήκευση της αρχιτεκτονικής και των βαρών του νευρωνικού δικτύου σε αρχεία JSON και «.h5».

```
arch = open('LeNet_architecture.json').read()
model = model_from_json(arch)
model.load_weights('LeNet_weights.h5')
arch = json.loads(arch)
save_LeNet5_to_generic_formatting()
```

Εικ. 3.9: Μετασχηματισμός των δεδομένων του νευρωνικού δικτύου σε πιο ευανάγνωστη μορφή.

```
with open("weights.h", 'w') as f:
    f.write('#include "dimensions.h"\n\n')
    f.write(declare_array_4D("float", "conv_1_weights", conv_weights1))
    f.write(declare_array_1D("float", "conv_1_biases", conv_b1))
    f.write(declare_array_4D("float", "conv_2_weights", conv_weights2))
    f.write(declare_array_1D("float", "conv_2_biases", conv_b2))
    f.write(declare_array_2D("float", "dense_1_weights", dense_weights1))
    f.write(declare_array_1D("float", "dense_1_biases", dense_b1))
    f.write(declare_array_2D("float", "dense_2_weights", dense_weights2))
    f.write(declare_array_1D("float", "dense_2_biases", dense_b2))
    f.write(declare_array_2D("float", "dense_3_weights", dense_weights3))
    f.write(declare_array_1D("float", "dense_3_biases", dense_b3))
```

Εικ. 3.10: Αποθήκευση των βαρών του δικτύου σε αρχείο «.h».

Οι κώδικες, οι οποίοι συγγράφηκαν για την σταδιακή εξαγωγή των δεδομένων του νευρωνικού δικτύου σε μορφή αναγνώσιμη από τη C/C++, αποδείχθηκαν ιδιαίτερα χρήσιμοι. Συγκεκριμένα, κατά την υλοποίηση του συστήματος, απαιτήθηκε αρκετές φορές η επανεκπαίδευση του δικτύου και η αλλαγή των βαρών στην σχεδίαση σε υλικό. Η διαδικασία αυτή θα ήταν σημαντικά πιο χρονοβόρα σε περίπτωση που οι παράμετροι του δικτύου μεταφέρονταν χειροκίνητα. Τέλος, τονίζεται ότι η αυτοματοποίηση της εξαγωγής των βαρών καθιστά δυνατή και την υλοποίηση άλλων νευρωνικών δικτύων με παρόμοιο τρόπο.

# 4. ΥΛΟΠΟΙΗΣΗ ΤΟΥ LENET-5 ΣΕ ΥΛΙΚΟ

## 4.1 Εισαγωγή

Όπως αναφέρθηκε στην εισαγωγή της παρούσας μελέτης, σκοπός της είναι η κατασκευή μιας υλοποίησης σε υλικό ενός νευρωνικού δικτύου, συγκεκριμένα του LeNet-5, και η εφαρμογή της υλοποίησης αυτής σε μία πλακέτα *FPGA* (*Field Programmable Gate Array*). Όμως, τα αναφερθέντα πλεονεκτήματα, έναντι μίας υλοποίησης λογισμικού, επισκιάζονται, αρχικά, από την μεγαλύτερη πολυπλοκότητα όσον αφορά την ανάπτυξη και αποσφαλμάτωση του κώδικα.

Οι γλώσσες περιγραφής υλικού που διατίθενται προς χρήση (VHDL και Verilog) απαιτούν εκτενή ενασχόληση και προσοχή κατά τη σχεδίαση των υποσυστημάτων. Επίσης, η διαδικασία της αποσφαλμάτωσης πραγματοποιείται με εξαντλητικό έλεγχο των κυματομορφών των διαφόρων σημάτων, ενώ η αυτοματοποίηση της διαδικασίας δυσκολεύει όταν χρησιμοποιείται μεγάλος όγκος δεδομένων. Στην παρούσα περίπτωση, για παράδειγμα, για τον έλεγχο ενός μόνο ζεύγους εισόδου - εξόδου (εικόνα - αντιστοιχούσα κλάση) απαιτείται εισαγωγή 784 εισόδων (τιμές εικονοστοιχείων) και έλεγχος 10 εξόδων (κατηγορίες ταξινόμησης). Μάλιστα, αν ληφθεί υπόψιν ότι στην αποσφαλμάτωση απαιτείται και μελέτη των ενδιάμεσων σημάτων, ο εξαντλητικός έλεγχος καθίσταται σχεδόν αδύνατος. Συνεπώς, προτιμήθηκε η χρήση ενός εργαλείου *σύνθεσης υψηλού επιπέδου* (*High Level Synthesis - HLS*).

Μία από τις ευρύτερα διαδεδομένες πλατφόρμες για αυτό τον σκοπό είναι το Vivado High-Level Synthesis.

Στο κεφάλαιο αυτό περιγράφονται, αρχικά, οι δυνατότητες του εργαλείου Vivado HLS, το οποίο χρησιμοποιήθηκε στην παρούσα διπλωματική εργασία, καθώς και των δυσκολιών που αντιμετωπίστηκαν κατά τη χρήση του. Έπειτα, παρουσιάζεται η υλοποίηση του νευρωνικού δικτύου LeNet-5 που πραγματοποιήθηκε στο εργαλείο Vivado HLS. Τέλος, το κεφάλαιο κλείνει με την ανάλυση της μεθόδου που χρησιμοποιήθηκε, προκειμένου να εξασφαλιστεί η αναμενόμενη συμπεριφορά από άποψη λειτουργικότητας της υλοποίησης του δικτύου σε υλικό.

## 4.2 Το εργαλείο Vivado High-Level Synthesis

Το εργαλείο αυτό προσφέρει τη δυνατότητα ανάπτυξης πλήρως λειτουργικού συστήματος σε FPGA, με τη κωδικοποίηση της λειτουργικότητάς του μέσω κώδικα γλώσσας C/C++ (είναι δυνατή και η χρήση δομών της SystemC, στην οποία όμως δεν θα γίνει περεταίρω αναφορά). Με αυτόν τον τρόπο καθίσταται γρηγορότερη η ανάπτυξη του αλγοριθμικού στοιχείου, ενώ ο χρήστης έχει τη δυνατότητα να παρέμβει στο παραγόμενο κύκλωμα μέσα από ένα εκτενές σύνολο από *οδηγίες σύνθεσης* (*directives*). Έπειτα από επεξεργασία του κώδικα και των *ψευδοεντολών* (*pragmas*) του προεπεξεργαστή, το εργαλείο παράγει αυτόματα τον αντίστοιχο κώδικα περιγραφής υλικού στο *Επίπεδο Μεταφοράς-Καταχωρητών* (*Register-Transfer Level - RTL*). Παράλληλα, δίνεται η δυνατότητα για αναπαράσταση του συστήματος τόσο σε VHDL, όσο και σε Verilog. Ο χρήστης μπορεί να επεξεργαστεί το παραχθέν κύκλωμα περεταίρω, αλλά αυτή η τεχνική δεν συνιστάται, διότι ο παραχθείς κώδικας δεν είναι εύκολα αναγνώσιμος.

Κατά την ανάπτυξη του συστήματος, οι τρεις βασικές επιλογές που έχει ο σχεδιαστής είναι οι εξής:

- Εξομοίωση του κώδικα C/C++ (**C Simulation**): Η σουίτα αναλαμβάνει να μεταγλωττίσει τον C/C++ κώδικα της υλοποίησης και να πραγματοποιήσει έλεγχο της αλγοριθμικής λογικής μέσω ενός ελεγκτικού προγράμματος. Ένα τέτοιο πρόγραμμα αναπτύσσεται από τον σχεδιαστή και σκοπός του είναι να ελέγξει την ορθή

λειτουργία και την ακρίβεια του συστήματος, βάσει γνωστών ζευγών εισόδων-εξόδων (περεταίρω ανάλυση στις Παρ. 4.5.1 και Παρ. 4.5.2).

- Σύνθεση (**Synthesis**) του κυκλώματος σε RTL σύστημα (κώδικας VHDL ή Verilog): Το εργαλείο αναλαμβάνει να συμβουλευτεί τον αλγοριθμικό κώδικα και τις εντολές του προεπεξεργαστή, παράγοντας μια κυκλωματική σχεδίαση εφαρμόσιμη σε ένα ορισμένο FPGA. Συνεπώς, γίνεται ενδελεχής εκτίμηση των απαραίτητων πόρων, των διασυνδέσεων μεταξύ των δομικών αριθμητικών, λογικών κα αποθηκευτικών στοιχείων και γενικότερα πραγματοποιείται έλεγχος της *συνθεσιμότητας* (*synthesizability*) του κυκλώματος. Επίσης, γίνεται εκτίμηση για τα ενεργειακά, χωρικά και χρονικά χαρακτηριστικά του συστήματος, ελέγχοντας, παράλληλα, την ικανοποίηση των περιορισμών (π.χ. περιορισμοί συχνότητας ρολογιού, χρήσης πόρων κλπ.) που τίθενται από τον σχεδιαστή.
- Ταυτόχρονη εξομοίωση (**C/RTL Co-simulation**) της αλγοριθμικής και της φυσικής υλοποίησης: Πρόκειται για το τελικό στάδιο ελέγχου μιας συστηματικής υλοποίησης, μέσω της διαδικασίας HLS. Με αυτή τη λειτουργία πραγματοποιείται σύγκριση μεταξύ των αποτελεσμάτων που παράγονται από το λογισμικό (Keras; C/C++) και το υλικού μοντέλου (RTL; VHDL, Verilog). Εξετάζεται, δηλαδή, αν το κύκλωμα που σχεδιάστηκε παράγει αποτελέσματα που είναι συνεπή με τον αλγόριθμο τον οποίο πρέπει να υλοποιεί.

## 4.3 Υλοποίηση του LeNet-5 σε γλώσσα C/C++

Λαμβάνοντας υπόψιν τους παραπάνω περιορισμούς, η ανάπτυξη του κώδικά συνοψίζεται σε ξεχωριστές συναρτήσεις για κάθε διαφορετικό επίπεδο (συνελικτικά, πλήρη, κ.τ.λ.). Όπως φαίνεται και στην Εικ. 4.1, ακολουθείται η αρχιτεκτονική που παρουσιάστηκε στην Υποπαρ. 2.3.2, με 2 συνελικτικά και 3 πλήρη επίπεδα, τα οποία διαχωρίζονται από 2 επίπεδα μέσης υποδειγματοληψίας, μεταξύ των συνελικτικών, και 1 επίπεδο ισοπέδωσης πριν το πρώτο πλήρες επίπεδο.

Όλα τα επίπεδα έχουν συνάρτηση ενεργοποίησης την ReLU, εκτός από το τελευταίο που χρησιμοποιεί την Softmax ώστε να πραγματοποιήσει κανονικοποίηση των εξόδων σε τιμές

πιθανοτήτων (βλ. Υποπαρ. 2.2.4). Εφόσον, όμως, σκοπός της υλοποίησής είναι η κατηγοριοποίηση των δεδομένων και όχι η ανάλυση σε πιθανότητες, παραλείφθηκε η Softmax στο τελευταίο επίπεδο, θεωρώντας ως επικρατούσα κλάση εκείνη με την μεγαλύτερη τιμή εξόδου. Επιπλέον, η Softmax απαιτεί πολλαπλές πράξεις με εκθετικά, κάτι το οποίο επιβαρύνει την ταχύτητα και τις απαιτήσεις σε χώρο του κυκλώματος. Τελικά, για την ερμηνεία τον αποτελεσμάτων εξόδου, απαιτείται μόνο μία απλή σύγκριση μεταξύ των τιμών.

```
/**
 * The top-level function of our LeNet-5 implementation in C/C++
 */
void ActivateNetwork(
        CL1_inp_t input_image[INPUT_DIMENSION][INPUT_DIMENSION], // Input image (feature map)
        DL3_out_t output8[OUTPUT8_LENGTH]) // Output probabilities
{
    // Intermediate parameters
    CL1_out_t output1[CONV1_NEURONS][OUTPUT1_DIM][OUTPUT1_DIM];
    PL1_out_t output2[CONV1_NEURONS][OUTPUT2_DIM][OUTPUT2_DIM];
    CL2_out_t output3[CONV2_NEURONS][OUTPUT3_DIM][OUTPUT3_DIM];
    PL2_out_t output4[CONV2_NEURONS][OUTPUT4_DIM][OUTPUT4_DIM];
    FL1_out_t output5[OUTPUT5_LENGTH];
    DL1_out_t output6[OUTPUT6_LENGTH];
    DL2_out_t output7[OUTPUT7_LENGTH];

    // Activation of each consecutive layer of the network
    conv_layer1(input_image, output1); // Activation of Convolutional Layer 1
    avg_pooling_layer1(output1, output2); // Activation of Average Pooling Layer 1
    conv_layer2(output2, output3); // Activation of Convolutional Layer 2
    avg_pooling_layer2(output3, output4); // Activation of Average Pooling Layer 1
    flattening_layer(output4, output5); // Activation of Flattening Layer
    dense_layer1(output5, output6); // Activation of Fully Connected Layer 1
    dense_layer2(output6, output7); // Activation of Fully Connected Layer 2
    dense_layer3(output7, output8); // Activation of Fully Connected Layer 3
}
```

Εικ. 4.1: Συνάρτηση ανώτατου επιπέδου στην υλοποίηση γλώσσας C/C++.

Όσον αφορά την μελέτη της επίδρασης διαφορετικών αριθμητικών συστημάτων στην επίδοση και ακρίβεια του συστήματος, σχεδιάστηκε μια απλή δομή ώστε κατά την επεξεργασία του κώδικα από το εργαλείο, να επιλέγεται με απλό τρόπο η επιθυμητή παραμετροποίηση. Στο αρχείο «.h» της βασικής συνάρτησης του συστήματος, ορίστηκε το αρχείο ορισμού αριθμητικών τύπων το οποίο θα χρησιμοποιηθεί (Εικ. 4.2). Η δυνατότητα αυτή χρησιμοποιήθηκε εκτενώς κατά τη μελέτη που πραγματοποιήθηκε στο Κεφ. 5. Στο κεφάλαιο αυτό δοκιμάστηκαν εναλλακτικά συστήματα αριθμητικής αναπαράστασης, με σκοπό την βελτιστοποίηση του κυκλώματος, τόσο για τα χρονικά, όσο και για τα χωρικά μεγέθη του.

```
#ifndef LENET_H
#define LENET_H

// Definitions of the networks and its dimensions according to different arithmetics.
// By uncommenting a line, its configuration is used.
//#include "lenet_float(16).h" // Uniform 16-bit floating-point arithmetic
//#include "lenet_float(32).h" // Uniform 32-bit floating-point arithmetic
//#include "lenet_fixed(8).h" // Uniform 8-bit fixed-point arithmetic
//#include "lenet_fixed(16,8).h" // Uniform 16-bit fixed-point arithmetic
//#include "lenet_fixed(32,16).h" // Uniform 32-bit fixed-point arithmetic
//#include "lenet_fixed(judd_uniform).h" // Uniform fixed-point arithmetic (custom 1)
//#include "lenet_fixed(judd_mixed).h" // Non-Uniform fixed-point arithmetic (custom 2)
//#include "lenet_fixed(ristretto_like).h" // Non-Uniform fixed-point arithmetic (custom 3)

void ActivateNetwork(CL1_inp_t input_image[INPUT_DIMENSION][INPUT_DIMENSION],
                     DL3_out_t output8[OUTPUT8_LENGTH]);
#endif
```

Εικ. 4.2: Επιλογή αριθμητικού συστήματος κατά τον ορισμό της παραμετροποίησης του LeNet-5.

## 4.4 Ιδιαιτερότητες του εργαλείου Vivado HLS

Η συγκεκριμένη πλατφόρμα, αν και προσφέρει πλειάδα χρήσιμων δυνατοτήτων, απαιτεί, επίσης, και κάποια ειδική μεταχείριση, για την όσο το δυνατόν ικανότερη εκμετάλλευσή τους. Ορισμένοι από αυτούς τους περιορισμούς προκύπτουν από τις δομικές διαφορές μεταξύ υλοποιήσεων λογισμικού και υλικού και κάποιοι από την λειτουργικότητα του εργαλείου. Παρακάτω, τίθενται τα κύρια θέματα τα οποία αντιμετωπίστηκαν. Μεγάλη συνεισφορά στην αναγνώριση και αντιμετώπισή τους λήφθηκε από τα φόρουμ της Xilinx [16] και της σημειώσεις για το Vivado HLS από την [17].

### 4.4.1 Υποστήριξη C++

Κατά το πρώτο ολοκληρωμένο στάδιο της δημιουργίας του κώδικα, πραγματοποιήθηκε υλοποίηση σε γλώσσα C++. Η αντικειμενοστρεφής προσέγγιση επέτρεψε την ενθυλάκωση των λειτουργιών σε ειδικές κλάσεις (π.χ. **επίπεδο**, **νευρώνας**, **συνάρτηση ενεργοποίησης**, κ.τ.λ.), καθώς και την επαναχρησιμοποίηση και την πολυμορφικότητα στην ανάπτυξη του κώδικα, αποδίδοντας αναγνωσιμότητα στον σχεδιαστή.

Στην πορεία, όμως, αποδείχθηκε ότι η συγκεκριμένη γλώσσα δεν υποστηρίζεται πλήρως από το εργαλείο. Η εξομοίωση αποτύγχανε, με κύρια παρατήρηση ότι πολλές από τις συνδέσεις μεταξύ των μονάδων δεν ήταν ορθά πραγματοποιημένες. Φαίνεται πως το εργαλείο δεν εκμεταλλεύεται σωστά το παραχθέν εκτελέσιμο από τον μεταγλωττιστή και δεν

μετασχηματίζει σωστά τον κώδικα σε κύκλωμα. Η υπόθεση αυτή εδραιώθηκε, αφού πρώτα εξασφαλίστηκε, μέσω της πλατφόρμας Visual Studio, ότι ο κώδικας λειτουργούσε σωστά σε περιβάλλον προσαρμοσμένο για ανάπτυξη λογισμικού (χρησιμοποιώντας μάλιστα τον ίδιο μεταγλωττιστή).

Για την αντιμετώπιση αυτού του προβλήματος, έγινε πλήρης αντικατάσταση όλων των αντικειμενοστρεφών δομών και δημιουργήθηκε μια υλοποίηση με πλήρη διαδικαστικό κώδικα (σε γλώσσα C). Κάθε λειτουργική μονάδα του κυκλώματος περιγράφεται πλέον σε δική της, πλήρως εξατομικευμένη συνάρτηση, με καμία επαναχρησιμοποίηση κώδικα μεταξύ τους. Ο κώδικας πλέον δεν είναι το ίδιο φιλικός ως προς τον σχεδιαστή, αλλά διευκολύνει το εργαλείο ως προς την εφαρμογή βελτιστοποιήσεων (βλ. Κεφ. 6), ενώ παραμένει σχετικά ευανάγνωστος.

### 4.4.2 Συνδέτης κώδικα και οργάνωση του προγράμματος

Όπως φαίνεται και στην Παρ. 4.3 κατά την ανάπτυξη του συστήματος, ο κώδικας δομήθηκε σε διαφορετικά επίπεδα, έχοντας ως πρότυπο την δομή ενός βαθέως νευρωνικού δικτύου. Για την επιτάχυνση της διαδικασίας χρησιμοποιήθηκαν αρχικά άλλα εργαλεία ανάπτυξης κώδικα, τα οποία ήταν προσαρμοσμένα για τον προγραμματισμό σε γλώσσα C/C++ (βλ. Υποπαρ. 4.4.1) Κατά την εισαγωγή του εν λόγω κώδικα στο εργαλείο HLS, διαπιστώθηκε ότι ο *συνδέτης κώδικα* (*linker*) του έχει κάποιες αδυναμίες όσον αφορά τη σύνδεση των επιμέρους αρχείων και την πραγματοποίηση των οδηγιών «include». Κατά την επεξεργασία του κώδικα, λήφθηκαν ενδείξεις οι οποίες προειδοποιούσαν για πολλαπλή συμπερίληψη αρχείων, καθιστώντας τη μεταγλώττιση ανεπιτυχή.

Απαιτείτο, λοιπόν, επαναδόμηση του συστήματος, ώστε για κάθε αρχείο «.c» να υπάρχει κατάλληλο αρχείο «.h», το οποίο να συμπεριλαμβάνει τις κατάλληλες εντολές προεπεξεργαστή «ifndef...define». Οι εντολές αυτές είναι απαραίτητες προκειμένου ο μεταγλωττιστής να μη συμπεριλαμβάνει κώδικα πάνω από μία φορά. Σε περίπτωση που συμβαίνει κάτι τέτοιο εμφανίζει σφάλματα τύπου «multiple definition of function/variable». Με αυτόν τον τρόπο, λοιπόν, τα συστήματα σύνδεσης και μεταγλώττισης κώδικα που παρέχει το εργαλείο μπορούν να επεξεργαστούν ορθά τον κώδικα, ώστε η λειτουργία C simulation να μπορέσει να πραγματοποιηθεί.

### 4.4.3 Δομές δεδομένων

Ένας άλλος περιορισμός, όσον αφορά το χρησιμοποιηθέν εργαλείο, προκύπτει από μια θεμελιώδη διαφορά μεταξύ υλικού και λογισμικού, την προσπέλαση στη μνήμη. Ενώ η γλώσσα C (ως γλώσσα λογισμικού) επιτρέπει την χρήση δεικτών για την προσπέλαση και επεξεργασία δεδομένων, είναι προφανές ότι το ίδιο δεν μπορεί να συμβεί σε μία υλοποίηση σε υλικό. Σε υλοποιήσεις υλικού, τα μεγέθη πρέπει να είναι αυστηρώς καθορισμένα εξαρχής, καθώς οι πόροι και οι συνδέσεις δεν δύνανται να δεσμεύονται δυναμικά. Δεν είναι δυνατή, δηλαδή, η μεταβλητότητα που παρέχεται μέσω ενός υπολογιστικού συστήματος γενικού σκοπού. Όπως ήταν αναμενόμενο, το εργαλείο εμφάνιζε μηνύματα αποτυχίας κατά την επεξεργασία του κώδικα.

Συνεπώς, πραγματοποιήθηκε αντικατάσταση όλων των δομών δεδομένων σε πίνακες, καλώς ορισμένου μεγέθους και γνωστού κατά το χρόνο μεταγλώττισης, όπως απαιτούσε το εργαλείο. Έτσι, το τελευταίο αποκτά πλήρη γνώση για τις μνήμες και τις συνδέσεις που χρησιμοποιούνται, έτσι ώστε να πραγματοποιείται επιτυχώς η σύνθεση του συστήματος.

### 4.4.4 Αριθμητική αυθαίρετης ακρίβειας

Κατά τη μελέτη της επίδοσης του συστήματος σε διαφορετικά συστήματα αριθμητικής, χρησιμοποιήθηκε η βιβλιοθήκη «ap_fixed», για τη δημιουργία αυθαίρετης ακριβείας αριθμούς σταθερής υποδιαστολής. Διαπιστώθηκε όμως, ότι η χρήση της συγκεκριμένης βιβλιοθήκης και των δημιουργημένων τύπων δεδομένων σε περιβάλλον Windows 10 είναι ελαττωματική. Συνεπώς, προτιμήθηκε εγκατάσταση και χρήση της πλατφόρμας σε λειτουργικό σύστημα Ubuntu 18.04, η οποία συνιστάται για τη χρήση του εργαλείου. Όντως, επετεύχθη ορθή υποστήριξη της προαναφερθείσας βιβλιοθήκης, ενώ παράλληλα η γενικότερη εμπειρία χρήσης κατέστη ομαλότερη.

## 4.5 Έλεγχος ακρίβειας

Η ανάπτυξη ενός συστήματος είναι άρρηκτα συνδεδεμένη με τον έλεγχο της επίτευξης ορθότητας των αποτελεσμάτων που παράγει, λαμβάνοντας υπόψιν τον αλγόριθμο και τις διαδικασίες που πρέπει να εκτελεί. Επίσης, κατά την διαδικασία βελτιστοποίησης της

υλοποίησης, μέσω αναδιαμόρφωσης του κώδικα, προσθήκης ψευδοεντολών παραμετροποίησης του κυκλώματος και αλλαγής αριθμητικών συστημάτων, συχνά εγείρονται θέματα μεταβολής της ακριβείας μεταξύ των αναμενομένων αποτελεσμάτων και των εξόδων που εμφανίζει το σύστημα για κάποια συγκεκριμένη είσοδο.

Συνεπώς, είναι απαραίτητη η υιοθέτηση ενός μηχανισμού, ο οποίος θα προσπελαύνει ταχύτατα τα διαθέσιμα δεδομένα του σετ δεδομένων, θα τροφοδοτεί το σύστημα με πλήθος εικόνων και θα συγκρίνει τα αποτελέσματα μεταξύ τους. Τελικά, πρέπει να παράγει ορισμένα στατιστικά στοιχεία, τα οποία αφορούν την ακρίβεια των τιμών εξόδου. Η υλοποίηση αυτού του μηχανισμού βασίστηκε στον τρόπο που παρουσιάστηκε στους παρεχόμενους οδηγούς του εργαλείου [18], [19] και αποτελείται από την κατασκευή καταλλήλων αρχείων δεδομένων και κώδικα.

### 4.5.1 Ζεύγη εισόδων και αναμενόμενων εξόδων

Το πρώτο βήμα είναι η πρόσβαση στην ορθή κατηγοριοποίηση των εικόνων που είναι διαθέσιμες. Με ένα απλό πρόγραμμα σε Python, εξήχθη σε ένα αρχείο «golden_labels.dat» το σύνολο των κλάσεων στις οποίες ανήκει κάθε εικόνα του MNIST dataset που χρησιμοποιήθηκε. Σε αντίστοιχο αρχείο, τοποθετήθηκε ένα σύνολο από 10.000 εικόνες «inputs.dat», το οποίο περιέχει τις τιμές όλων των εικονοστοιχείων των εικόνων του σετ δοκιμών (βλ. Υποπαρ. 2.3.1). Επίσης παρήχθη και το αρχείο «golden_outputs.dat» το οποίο χρησιμοποιήθηκε στο πρώτο σκέλος ανάπτυξης, για την επιμέρους σύγκριση των νευρώνων εξόδου, μεταξύ της υλοποίησης σε Keras και αυτής σε C. Με αυτό τον τρόπο είναι διαθέσιμες οι εικόνες εισόδου και τα αναμενόμενα αποτελέσματα του κυκλώματος, τόσο σαν κατηγοριοποίηση, όσο και σαν επιμέρους τιμές εξόδου.

### 4.5.2 Αποτελέσματα προσομοίωσης

Διαθέτοντας τα αρχεία με τα παραπάνω δεδομένα, πραγματοποιήθηκε ανάπτυξη ενός ελεγκτικού προγράμματος. Όπως αναφέρθηκε και στην Παρ. 4.2, η λειτουργία **C Simulation** του εργαλείου πραγματοποιεί εκτίμηση της ακριβείας του παραχθέντος αλγορίθμου, μέσω ενός ελεγκτικού προγράμματος. Ο συγκεκριμένος κώδικας ενεργοποιεί επαναληπτικά το σύστημα, θέτοντάς του μία διαφορετική εικόνα κάθε φορά ως είσοδο, και

αποκτά την εκάστοτε απόκρισή του, δηλαδή, τις τιμές των νευρώνων του τελευταίου επιπέδου (Εικ. 4.3). Σε κάθε βήμα, πραγματοποιεί έλεγχο συνάφειας των επιμέρους αποτελεσμάτων για την συνάφεια ως προς την κατηγοριοποίηση των δεδομένων εισόδου. Μετά την προσπέλαση του σετ εισόδων και την εξαγωγή των αποτελεσμάτων του αλγορίθμου, υπολογίζει μία συνολική εκτίμηση για την **TOP-1**, **TOP-3** και **TOP-5** ακρίβεια του μοντέλου (Εικ. 4.4).

Ως **TOP-n** ακρίβεια ορίζεται το ποσοστό των εικόνων των οποίων η κατηγορία ανήκει στις **n** υψηλότερες πιθανότητες εξόδου του δικτύου[1]. Για παράδειγμα, όταν η ορθή κατηγορία μιας εικόνας ανήκει στις πρώτες 3 που εκτίμησε το δίκτυο, τότε αυτή προσμετράται στην **TOP-3** ακρίβεια. Αντίστοιχα, η περίπτωση **n=1** αφορά, ουσιαστικά, την σωστή αναγνώριση της εικόνας. Συνεπώς, είναι προφανές ότι ισχύει:

$$\mathrm{TOP}-1 \leq \mathrm{TOP}-2 \leq \cdots \leq \mathrm{TOP}-\mathrm{n} \qquad \text{Εξ. 4.1}$$

```
printf("Running the design and gathering its output...\n");
for(int i=0;i<TEST_INPUTS_NUMBER;i++){
    ActivateNetwork(test_input[i], test_output_synthesized[i]);
```

Εικ. 4.3: Επαναληπτική εκτέλεση του συστήματος.

```
TESTS IMAGES USED: 10000
> TOP-1 ACCURACY: 98.72
> TOP-3 ACCURACY: 99.81
> TOP-5 ACCURACY: 99.81
TEST PASSED !!!
```

Εικ. 4.4: Παράδειγμα αποτελεσμάτων του ελεγκτικού προγράμματος.

Το σύστημα ελέγχου κατασκευάστηκε ώστε να προσφέρει υψηλή ελευθερία παραμετροποίησης, με τον σχεδιαστή να επιλέγει την προέλευση των δεδομένων εισόδου (σετ εκπαίδευσης ή σετ δοκιμών), καθώς και το πλήθος των εικόνων. Παράλληλα, δίνεται επιλογή επιμέρους ελέγχου και σύγκρισης των εξόδων με τις τιμές που εξάγει το λειτουργικό μοντέλο σε Keras. Το τελευταίο αποδείχθηκε ιδιαίτερα χρήσιμο κατά την αρχική ανάπτυξη του συστήματος, όταν υπήρχαν συνεχείς απαιτήσεις για αποσφαλμάτωση του παραχθέντος κώδικα και χρειάστηκε να κάνουμε ενδελεχή έρευνα στα προβληματικά υποσυστήματα.

[1] Υπενθυμίζεται ότι το LeNet-5 έχει ως έξοδο την πιθανότητα της εικόνας εισόδου να ανήκει στην εκάστοτε κατηγορία (ψηφίο 0, 1, 2 κλπ., βλ. Υποπαρ. 2.3.2).

# 5. ΣΥΜΠΙΕΣΗ ΤΟΥ LENET-5

## 5.1 Εισαγωγή

Το μοντέλο του νευρωνικού δικτύου LeNet-5 που παράχθηκε στο Keras χρησιμοποιεί αριθμητική κινητής υποδιαστολής 32-bit, τόσο για τα δεδομένα, όσο και για τα βάρη του δικτύου. Εφαρμόζοντας αυτό το είδος αριθμητικής, μια σχεδίαση σε υλικό θα είχε σημαντική καθυστέρηση (άρα μικρή διεκπεραιωτική ικανότητα), καθώς και θα χαρακτηριζόταν από μεγάλη σπατάλη πόρων και ενέργειας. Ως εκ τούτου, κρίνεται σκόπιμη η υιοθέτηση αριθμητικής σταθερής υποδιαστολής, η οποία παρουσιάζει σημαντικά λιγότερο υπολογιστικό φόρτο σε σύγκριση με την αριθμητική κινητής υποδιαστολής. Παράλληλα, περεταίρω βελτίωση στα φυσικά και λειτουργικά μεγέθη του συστήματος μπορεί να επιτευχθεί και με τη χρήση μικρότερου μήκους λέξης στα δεδομένα και τις παραμέτρους του νευρωνικού δικτύου [20].

Ωστόσο, οι παραπάνω ενέργειες έχουν (δυνητικά) σαν αποτέλεσμα την υποβάθμιση της ακρίβειας του νευρωνικού δικτύου. Για την περίπτωση του LeNet-5, αυτό μεταφράζεται σε μεγαλύτερα ποσοστά σφάλματος κατά την αναγνώριση των χειρόγραφων ψηφίων. Απαιτείται, λοιπόν, μια μέθοδος κβαντισμού των παραμέτρων του νευρωνικού δικτύου, προκειμένου να μειωθεί η υπολογιστική πολυπλοκότητα διατηρώντας, όμως, την απώλεια ακρίβειας σε αποδεκτά πλαίσια.

Στο κεφάλαιο αυτό παρουσιάζονται ορισμένες τεχνικές συμπίεσης νευρωνικών δικτύων, οι οποίες έχουν καθιερωθεί στη βιβλιογραφία [14], [21], [22], [23], [24], τόσο για εμπρόσθιας διάδοσης νευρωνικά δίκτυα, όσο και για συνελικτικά νευρωνικά δίκτυα. Έπειτα, ακολουθεί η

μεθοδολογία με την οποία εφαρμόστηκαν οι τεχνικές αυτές στο νευρωνικό δίκτυο LeNet-5 της παρούσας διπλωματικής εργασίας. Τέλος, το κεφάλαιο κλείνει με τη σύγκριση των αποτελεσμάτων που προέκυψαν, όσον αφορά τα φυσικά και λειτουργικά χαρακτηριστικά, των διαφόρων υλοποιήσεων που δοκιμάστηκαν και βασίστηκαν στις εκάστοτε μεθόδους συμπίεσης.

## 5.2 Ανασκόπηση βιβλιογραφίας

### 5.2.1 Μέθοδος ομοιόμορφου κβαντισμού παραμέτρων

Αποτελεί την απλούστερη μέθοδο κβαντισμού των παραμέτρων σε νευρωνικά δίκτυα, διότι, όπως υποδεικνύει και το όνομά της, εφαρμόζει τον ίδιο κβαντισμό σε όλα τα επίπεδα του δικτύου, χωρίς επιμέρους διαφοροποιήσεις. Αυτό συμπεριλαμβάνει: 1) τις **εισόδους**, 2) τα **βάρη** και 3) τις **εξόδους** του εκάστοτε επιπέδου. Για να πραγματοποιηθεί περεταίρω ανάλυση αυτής της μεθόδου συμπίεσης, καθώς και των υπόλοιπων μεθόδων που θα παρουσιαστούν παρακάτω, χρειάζεται να περιγραφούν ορισμένες έννοιες. Για έναν αριθμό σταθερής υποδιαστολής ορίζονται, λοιπόν, οι παρακάτω ποσότητες:

- **IL (Integer Length – μήκος ακεραίου μέρους)**: Το πλήθος των bits που αποτελούν το ακέραιο μέρος του αριθμού.
- **FL (Fractional Length – μήκος κλασματικού μέρους):** Το πλήθος των bits που συνιστούν το κλασματικό μέρος του αριθμού.
- **BL (Bit Length – μήκος λέξης):** Το συνολικό πλήθος των bits του αριθμού.

Βάσει των ορισμών, είναι προφανές ότι ισχύει η σχέση της Εξ. 5.1, σύμφωνα με την οποία το μήκος σε bits ενός αριθμού είναι ίσο με το άθροισμα του ακέραιου και του κλασματικού μήκους του σε bits αντίστοιχα. Συνεπώς, η μέθοδος του ομοιόμορφου κβαντισμού απαιτεί την επιλογή μόνο ενός ζεύγους παραμέτρων, αφού η τελευταία προκύπτει από τις πρώτες δύο.

$$BL = IL + FL \qquad \text{Εξ. 5.1}$$

Στην μελέτη [14] χρησιμοποιείται αριθμητική σταθερής υποδιαστολής με **BL=16**. Βασικό συμπέρασμα της έρευνας αυτής είναι ότι η εκπαίδευση βαθέων νευρωνικών δικτύων μπορεί να πραγματοποιηθεί με αριθμητική σταθερής υποδιαστολής μικρότερου **BL** (σε σχέση με τη

συνήθη υλοποίηση του αντίστοιχου δικτύου) μέσω της χρήσης *στοχαστικής στρογγυλοποίησης* (*stochastic rounding*). Για την επαλήθευση των ευρημάτων τους χρησιμοποιήθηκαν τα σετ δεδομένων MNIST, το οποίο χρησιμοποιείται και στην παρούσα διπλωματική εργασία (βλ. Υποπαρ. 2.3.1), και CIFAR-10 [25]. Σημειώνεται ότι η στοχαστική στρογγυλοποίηση ορίζεται ως εξής:

$$Round(x) = \begin{cases} floor(x), με\ πιθανότητα\ 1 - \dfrac{x - floor(x)}{q} \\ floor(x) + q, με\ πιθανότητα\ \dfrac{x - floor(x)}{q} \end{cases} \qquad \text{Εξ. 5.2}$$

όπου **floor(x)** η προς τα κάτω στρογγυλοποίηση της τιμής **x** και **q** το βήμα κβαντισμού (ή ανάλυση). Συγκεκριμένα, για το βήμα κβαντισμού ισχύει:

$$q = 2^{-FL} \qquad \text{Εξ. 5.3}$$

Για παράδειγμα, στην περίπτωση που **q=1**, και άρα **FL=0**, ο αριθμός 4.2 έχει 80% πιθανότητα να στρογγυλοποιηθεί στο 4 και 20% πιθανότητα να στρογγυλοποιηθεί στο 5.

Όπως αναφέρεται και στην [14], κατά τη χρήση αυτής της μεθόδου, δύναται να δημιουργηθούν προβλήματα κατά την εκπαίδευση του δικτύου, όπως καθυστέρηση σύγκλισης ή, ακόμα χειρότερα, αστάθεια. Τα προβλήματα αυτά προκύπτουν από το γεγονός ότι η μέθοδος αυτή είναι αρκετά περιοριστική και δεύτερον επειδή είναι πιθανή η εμφάνιση υπερχείλισης στο ακέραιο μέρος του αριθμού. Μια λύση είναι η μεταφορά ορισμένου αριθμού bits από το κλασματικό μέρος του αριθμού προς το ακέραιο[1] (το άθροισμα **FL+IL** παραμένει σταθερό και στην προκειμένη περίπτωση ίσο με 16). Μειονέκτημα της τεχνικής αυτής είναι η υποβάθμιση της ακρίβειας εξαιτίας της περικοπής των **FL** bits. Επομένως, αν έστω και ένα επίπεδο παρουσιάζει υψηλή ευαισθησία (είτε στην εκπαίδευση, είτε στην αναγνώριση), τότε απαιτείται αλλαγή του ζεύγους παραμέτρων **(IL, FL)**. Αυτοί ακριβώς οι περιορισμοί αποτελούν και την αδυναμία της μεθόδου ομοιόμορφου κβαντισμού. Παρόλα αυτά, η χρήση του ίδιου **BL** σε όλα τα επίπεδα, σε συνδυασμό με την χρήση αρχιτεκτονικής σταθερής (έναντι κινητής) υποδιαστολής, οδηγεί σε απλούστερες υλοποιήσεις σε υλικό.

[1]Επί της ουσίας, μεταφορά της υποδιαστολής ορισμένες θέσεις δεξιά.

### 5.2.2 Μέθοδος αριθμητικής δυναμικής σταθερής υποδιαστολής

Πρόκειται για μια περισσότερο πολύπλοκη τεχνική, σε σχέση με τον ομοιόμορφο κβαντισμό παραμέτρων. Η πολυπλοκότητά της οφείλεται στο γεγονός ότι στη γενική περίπτωση εφαρμόζονται διαφορετικά ζεύγη παραμέτρων **(IL, FL)** σε κάθε επίπεδο του δικτύου. Συγκεκριμένα, υπάρχουν δύο παραλλαγές της:

**Παραλλαγή 1:** Σε κάθε επίπεδο ισχύει πάντα η σχέση:

$$(IL, FL)_{in} = (IL, FL)_{wei} = (IL, FL)_{out} \qquad \text{Εξ. 5.4}$$

όπου $\mathbf{(IL, FL)_{in}}$, $\mathbf{(IL, FL)_{wei}}$ και $\mathbf{(IL, FL)_{out}}$ το ζεύγος παραμέτρων **(IL, FL)** της **εισόδου**, των **βαρών** και της **εξόδου** ενός συγκεκριμένου επιπέδου αντίστοιχα. Βασικό της πλεονέκτημα είναι η απλότητα όσον αφορά την υλοποίηση. Ωστόσο είναι λιγότερο ευέλικτη από τη δεύτερη παραλλαγή.

**Παραλλαγή 2:** Αποτελεί μία γενίκευση της πρώτης παραλλαγής. Στην περίπτωση αυτή, για κάθε τύπο δεδομένων του εκάστοτε επιπέδου ορίζεται και ένα διαφορετικό ζεύγος **(IL, FL)** και, ως αποτέλεσμα, η Εξ. 5.4 δε βρίσκει εφαρμογή. Εφόσον κάθε τύπος δεδομένων προσαρμόζεται στις ανάγκες του κάθε στοιχείου ξεχωριστά, η πολυπλοκότητα της συμπίεσης του δικτύου αυξάνεται. Όμως, δυνητικά εμφανίζεται περεταίρω μείωση των υπολογιστικών απαιτήσεων, επειδή προσφέρει μεγαλύτερη ελευθερία κβαντισμού των παραμέτρων του νευρωνικού δικτύου. Ωστόσο, οδηγεί σε υλοποιήσεις υψηλότερης πολυπλοκότητας σε σχέση με την πρώτη παραλλαγή.

Είναι προφανές, ωστόσο, ότι και στις δύο περιπτώσεις πρέπει να υπάρχει συνέπεια στα ζεύγη $\mathbf{(IL, FL)_{in}}$ και $\mathbf{(IL, FL)_{out}}$ σε διαδοχικά επίπεδα. Απαιτείται, συνεπώς, κάποια τεχνική εύρεσης των μικρότερων **(IL, FL)** σε κάθε επίπεδο επιτρέποντας, όμως, μικρή απώλεια ακρίβειας.

Στην [22], συμπεραίνεται ότι οι απαιτήσεις ακρίβειας των αριθμητικών μεγεθών ποικίλουν ανάμεσα στα επίπεδα των νευρωνικών δικτύων και ως εκ τούτου η κβάντιση ανά επίπεδο μπορεί να οδηγήσει σε αποδεκτή ακρίβεια με επιπλέον, ωστόσο, εξοικονόμηση πόρων. Ως μέθοδος εύρεσης των συνδυασμών ζευγών **(IL, FL)** ακολουθήθηκε ο παρακάτω αλγόριθμος:

**Βήμα 1:** Κβάντιση των δεδομένων όλων των επιπέδων με τη μέθοδο ομοιόμορφου κβαντισμού, περιορίζοντας την πτώση ακρίβειας σε λιγότερο από το 0.1% της αρχικής υλοποίησης κινητής υποδιαστολής 32-bit..

**Βήμα 2:** Δημιουργία καινούργιων συνδυασμών **IL** και **FL**, μειώνοντας τα σε κάθε επίπεδο κατά ένα.

**Βήμα 3:** Επιλογή του συνδυασμού που έχει τη μεγαλύτερη ακρίβεια και επιστροφή στο Βήμα 2. Ο αλγόριθμος τερματίζει όταν προκύψει ένας συνδυασμός που πληροί τις επιθυμητές προδιαγραφές, όσον αφορά την ακρίβεια και το βαθμό συμπίεσης των παραμέτρων του νευρωνικού δικτύου.

Στις [21] και [26] εξάγεται επίσης το συμπέρασμα ότι η μέθοδος δυναμικής σταθερής υποδιαστολής οδηγεί σε διατάξεις αποδεκτής ακρίβειας με περεταίρω σμίκρυνση του χρησιμοποιούμενου υλικού. Στην [21] επιπλέον υποστηρίζεται πως η χρήση μικρότερου εύρους λέξης μειώνει την υπερεκπαίδευση στα συνελικτικά δίκτυα. Όπως αναφέρθηκε στην Υποπαρ. 2.2.6, υπερεκπαίδευση είναι η κατάσταση κατά την οποία το νευρωνικό δίκτυο προσαρμόζεται στο σετ δεδομένων πάνω στο οποίο εκπαιδεύτηκε και αστοχεί σε δείγματα εκτός αυτού. Για την εύρεση του μήκους λέξης ανά επίπεδο πραγματοποιείται κβάντιση του προς εξέταση επιπέδου κρατώντας τα υπόλοιπα σε υψηλή ανάλυση και επαναλαμβάνεται η διαδικασία αυτή για όλα τα επίπεδα.

Είναι προφανές πως η αύξηση του βάθους ενός νευρωνικού δικτύου έχει σαν αποτέλεσμα ραγδαία διεύρυνση του *χώρου λύσεων* (*solution space*). Το γεγονός αυτό δυσχεραίνει τη διαδικασία εύρεσης των ελαχίστων **(IL, FL)** ξεχωριστά για κάθε επίπεδο (σε αντίθεση με την ομοιόμορφη μέθοδο) διατηρώντας, ταυτόχρονα, την ακρίβεια του δικτύου σε αποδεκτά όρια. Παρόλα αυτά, η τεχνική αυτή εκμεταλλεύεται πολύ πιο αποδοτικά την ανοχή στην ακρίβεια κάθε επιπέδου. Παράλληλα, δίνεται η δυνατότητα για πιο «επιθετική» κβάντιση σε σχέση με την ομοιόμορφη μέθοδο. Πλέον, το μήκος λέξης σε κάθε επίπεδο μπορεί να ελαττωθεί ανεξάρτητα από τα υπόλοιπα, δίνοντας τη δυνατότητα χρήσης ακόμα μικρότερων (και γρηγορότερων) μνημών.

### 5.2.3 Μέθοδος του εργαλείου Ristretto

Το εργαλείο Ristretto [24] χρησιμοποιείται για την εύρεση των παραμέτρων **(IL, FL)** κάθε επιπέδου ενός νευρωνικού δικτύου. Παράλληλα, δίνει τη δυνατότητα στους σχεδιαστές να εξετάσουν την επίδραση που έχουν διαφορετικές αριθμητικές αναπαραστάσεις και μήκη λέξεων στη συμπεριφορά του δικτύου τους [26]. Εκτός αυτών, το Ristretto μπορεί να προσαρμόσει με ακρίβεια τα βάρη του (πλέον συμπιεσμένου) νευρωνικού δικτύου για τη βελτίωση των επιδόσεών του.

Το Ristretto πρόκειται για ένα εργαλείο το οποίο στήνεται πάνω στο Caffe [27], λογισμικό που, όπως το Keras, χρησιμοποιείται για την περιγραφή και εκπαίδευση νευρωνικών δικτύων. Η δημιουργία ενός μοντέλου γίνεται σε text αρχείο (συγκεκριμένα .prototxt), σε αντίθεση με το Keras όπου γίνεται μέσω Python (βλ. Κεφ. 3).

Αξίζει να σημειωθεί πως το εργαλείο Ristretto υποστηρίζει τρεις μεθόδους κβαντισμού:

1. *Αριθμητική δυναμικής σταθερής υποδιαστολής* (*dynamic fixed point*) (βλ. Παρ. 5.2.2).
2. *Αριθμητική κινητής υποδιαστολής περιορισμένου μήκους λέξης* (*minifloat*), δηλαδή αριθμητική κινητής υποδιαστολής μήκους ≤16-bit.
3. *Αριθμητική ελεύθερη πολλαπλασιασμών* (*multiplier-free arithmetic*), στην οποία τα βάρη στρογγυλοποιούνται στην κοντινότερη δύναμη του 2, προκειμένου οι πολλαπλασιασμοί να μετατραπούν σε ολισθήσεις.

Το λογισμικό Ristretto ακολουθεί μια μεθοδολογία παρόμοια με αυτές που αναφέρθηκαν στις Υποπαρ. 5.2.1 και Υποπαρ. 5.2.2. Συγκεκριμένα, ακολουθεί μια διαδικασία των εξής 5 βημάτων [23]:

**Βήμα 1:** Ανάλυση του δυναμικού εύρους των βαρών του δικτύου. Το **IL** επιλέγεται τέτοιο ώστε να αποφεύγεται ο κορεσμός.

**Βήμα 2:** Πολλαπλές ενεργοποιήσεις του δικτύου και υπολογισμός στατιστικών δεδομένων.

**Βήμα 3:** Εύρεση των βέλτιστων $\mathbf{(IL, FL)_{in}}$, $\mathbf{(IL, FL)_{wei}}$ και $\mathbf{(IL, FL)_{out}}$ για κάθε επίπεδο κρατώντας τα υπόλοιπα σε υψηλή ακρίβεια.

**Βήμα 4:** Έλεγχος της ακρίβειας αναγνώρισης με τη χρήση του σετ εκπαίδευσης.

**Βήμα 5:** Προσαρμογή ακριβείας: Επανεκπαίδευση του συμπιεσμένου, πλέον, δικτύου.

Το τελευταίο βήμα εκτελείται ώστε να αντισταθμιστεί η απώλεια ακρίβειας, η οποία προκαλείται εξαιτίας του κβαντισμού του δικτύου. Λεπτομέρεια-κλειδί του σταδίου αυτού είναι πως η επανεκπαίδευση πραγματοποιείται σε πλήρες εύρος bit (32-bit) παρότι τα βάρη και τα ενδιάμεσα δεδομένα έχουν κβαντιστεί. Αυτό γίνεται διότι μικρές αλλαγές στα βάρη δε γίνονται αντιληπτές στο συμπιεσμένο δίκτυο. Επειδή, ωστόσο, η εκπαίδευση είναι επαναληπτική διαδικασία, η συσσώρευσή τους είναι δυνατό να έχει σημαντική διαφορά.

## 5.3 Εφαρμογή μεθόδων συμπίεσης στο LeNet-5

Στην παρούσα διπλωματική εργασία δόθηκε ιδιαίτερη έμφαση στη μέθοδο της δυναμικής σταθερής υποδιαστολής, δεδομένου ότι πετυχαίνει καλές επιδόσεις όσον αφορά την ακρίβεια, ελαττώνοντας σημαντικά το μέγεθος του δικτύου. Αντίθετα, η μέθοδος minifloat καθίσταται ακατάλληλη για υλοποίηση σε υλικό, εξαιτίας της κινητής υποδιαστολής, ενώ η αριθμητική ελεύθερη πολλαπλασιασμών παρουσιάζει τα μεγαλύτερα σφάλματα κβαντισμού [26].

Αρχικά, εφαρμόστηκαν οι μέθοδοι ομοιόμορφου κβαντισμού παραμέτρων και αριθμητικής δυναμικής σταθερής υποδιαστολής, οι οποίες αναλύθηκαν στις Υποπαρ. 5.2.1 και Υποπαρ. 5.2.2, αντίστοιχα. Ιδιαιτέρως χρήσιμα αποτελέσματα λήφθηκαν από τους συνδυασμούς που προτάθηκαν στην [22]. Εκτός από τις μεθόδους αυτές, κύρια έμφαση δόθηκε στην παραμετροποίηση που προέκυψε από τη χρήση του εργαλείου Ristretto (βλ. Υποπαρ. 5.2.3).

### 5.3.1 Εφαρμογή ομοιόμορφου κβαντισμού παραμέτρων

Μία σημαντική παρατήρηση που εγείρεται στην [22] κατά την ανάλυση ενός τυπικού νευρωνικού δικτύου, είναι ότι τα βάρη σε κάθε νευρώνα όλων των επιπέδων τυπικά κυμαίνονται εντός του αριθμητικού διαστήματος [-1,1]. Το γεγονός αυτό καθιστά δυνατή την αναπαράσταση των βαρών με τη χρήση ενός μόνο ακεραίου ψηφίου ($\mathbf{IL_{wei}=1}$). Όσον αφορά το κλασματικό μέρος, συγκεκριμένα για το LeNet 7 ψηφία επαρκούν ώστε να μην υπάρχει παρατηρήσιμη απώλεια ακρίβειας του συστήματος ($\mathbf{BL_{wei}=7}$). Παράλληλα, για τα δεδομένα φαίνεται να επαρκούν 8 ψηφία για το ακέραιο ($\mathbf{IL_{data}=8}$)και 2 για το κλασματικό μέρος

(**$BL_{data}$=2**). Έπειτα από τη στατιστική μελέτη που πραγματοποιήθηκε, το δίκτυο έδειξε να διατηρεί την ακρίβειά του σταθερή, παρά τη δραματική μείωση σε υπολογιστικούς πόρους [22].

Ακολουθώντας πιστά την παραπάνω παραμετροποίηση και αφού προσαρμόστηκαν οι τιμές των παραμέτρων για το ένα επιπλέον επίπεδο που διαθέτει η παραλλαγή του LeNet στην παρούσα διπλωματική εργασία, ορίστηκαν τα μήκη λέξεων για είσοδο και έξοδο ως **$(IL, FL)_{data}$=(8, 2)** και αντίστοιχα **$(IL, FL)_{wei}$=(1, 7)** για τα βάρη (βλ. Πίν. 5.1).

Πίν. 5.1: Αρχικά και προσαρμοσμένα ζεύγη (IL, FL) για εφαρμογή ομοιόμορφου κβαντισμού παραμέτρων.

| **Επίπεδο** | **Από βιβλιογραφία** | | | **Χρησιμοποιούμενα** | | |
|---|---|---|---|---|---|---|
| | είσοδος | βάρη | έξοδος | είσοδος | βάρη | έξοδος |
| Συνελικτικό 1 | (8,2) | (1,7) | (8,2) | (8,2) | (1,7) | (8,2) |
| Μέσης υποδειγματοληψίας 1 | (8,2) | - | (8,2) | (8,2) | - | (8,2) |
| Συνελικτικό 2 | (8,2) | (1,7) | (8,2) | (8,2) | (1,7) | (8,2) |
| Μέσης υποδειγματοληψίας 2 | (8,2) | - | (8,2) | (8,2) | - | (8,2) |
| Πλήρως διασυνδεδεμένο 1 | (8,2) | (1,7) | (8,2) | (8,2) | (1,7) | (8,2) |
| Πλήρως διασυνδεδεμένο 2 | (8,2) | (1,7) | (8,2) | (8,2) | (1,7) | (8,2) |
| Πλήρως διασυνδεδεμένο 3 | - | - | - | (8,2) | (1,7) | (8,2) |

### 5.3.2 Εφαρμογή αριθμητικής δυναμικής σταθερής υποδιαστολής

Σύμφωνα με την [22], επιπλέον βελτιστοποίηση στις απαιτήσεις του κυκλώματος επιτυγχάνεται με τον ανά επίπεδο προσαρμοσμένο κβαντισμό. Ανάλογα με την ευαισθησία της ακρίβειας της διάταξης των ψηφίων ανά επίπεδο, τα μήκη λέξεων μπορούν συγκεντρωτικά να μειωθούν ακόμη περισσότερο. Πλέον, το κάθε επίπεδο προσαρμόζεται σε δικά του μήκη λέξεων και η πολυπλοκότητα μειώνεται ακόμα πιο αισθητά, σε σχέση με την αναπαράσταση κινητής υποδιαστολής 32 ψηφίων. Συγκεκριμένα, προτείνεται η διάταξη του Πίν. 5.2. Έπειτα από στατιστική μελέτη που πραγματοποιήθηκε στην [21], το δίκτυο έδειξε να διατηρεί ανοχή σφάλματος εντός του 1%, επιτυγχάνοντας παράλληλα μείωση των απαιτήσεων σε μετάδοση/προσπέλαση δεδομένων κατά 92%.

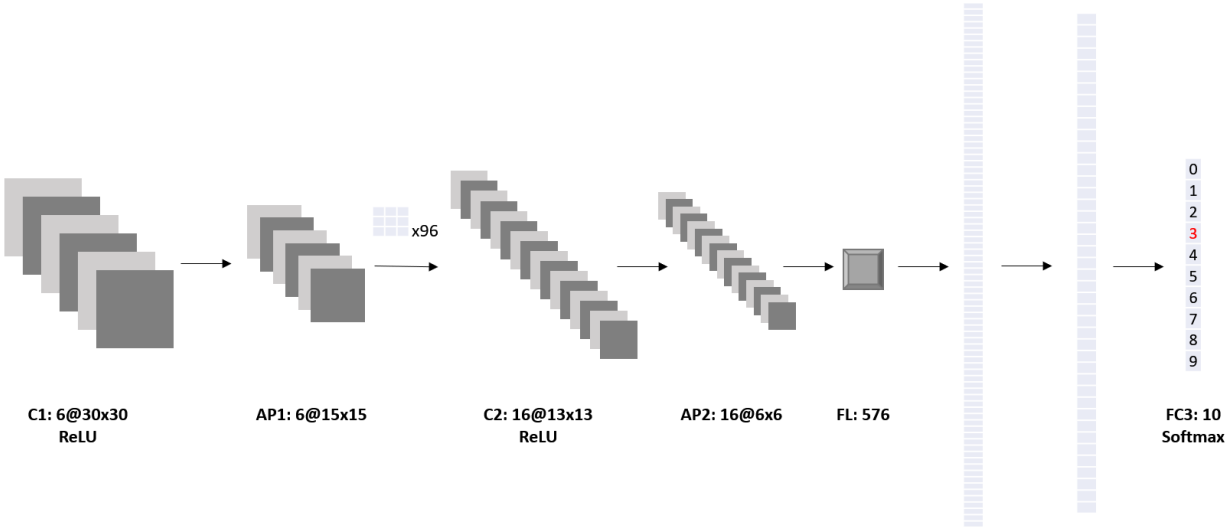
Στην περίπτωση αυτή επίσης ακολουθήθηκαν στενά οι υποδείξεις της ανάλυσης. Ήταν απαραίτητες, όμως, κάποιες προσαρμογές, οι οποίες αφορούν κυρίως την είσοδο του πρώτου επιπέδου (βλ. δεξιά στήλη του Πίν. 5.2). Κατά τα άλλα, η μέγιστη απόκλιση από το προτεινόμενο μοντέλο δεν ξεπερνά σε καμία περίπτωση το ένα ψηφίο.

Πίν. 5.2: Αρχικά και προσαρμοσμένα ζεύγη (IL, FL) αριθμητικής δυναμικής σταθερής υποδιαστολής.

| **Επίπεδο** | **Από βιβλιογραφία** | | | **Χρησιμοποιούμενα** | | |
|---|---|---|---|---|---|---|
| | είσοδος | βάρη | έξοδος | είσοδος | βάρη | έξοδος |
| Συνελικτικό 1 | (1,1) | (1,3) | (3,0) | (8,0) | (1,2) | (3,1) |
| Μέσης υποδειγματοληψίας 1 | (3,0) | - | (3,0) | (3,1) | - | (3,1) |
| Συνελικτικό 2 | (3,0) | (1,5) | (3,0) | (3,1) | (1,4) | (3,1) |
| Μέσης υποδειγματοληψίας 2 | (3,0) | - | (3,0) | (3,1) | - | (3,1) |
| Πλήρως διασυνδεδεμένο 1 | (3,0) | (1,7) | (3,0) | (3,1) | (1,6) | (3,1) |
| Πλήρως διασυνδεδεμένο 2 | (3,0) | (1,5) | (3,0) | (3,1) | (1,6) | (3,1) |
| Πλήρως διασυνδεδεμένο 3 | - | - | - | (3,1) | (1,4) | (3,1) |

### 5.3.3 Εφαρμογή της μεθόδου του εργαλείου Ristretto

Αρχικά, αναπτύχθηκε η αρχιτεκτονική του νευρωνικού δικτύου LeNet-5 σε μορφή «.prototxt». Η αρχιτεκτονική αυτή (βλ. Παρ. 2.3) επαναλαμβάνεται στην Εικ. 5.1 για διευκόλυνση του αναγνώστη. Η εισαγωγή της αρχιτεκτονικής στο Caffe έγινε με τη βοήθεια σχετικών παραδειγμάτων που προσφέρει το λογισμικό στην ιστοσελίδα του [28]. Προφανώς, χρειάστηκαν τροποποιήσεις σχετικά με το LeNet-5 των παραδειγμάτων, έτσι ώστε η αρχιτεκτονική του να συμφωνεί με την παραλλαγή που χρησιμοποιήθηκε στην παρούσα διπλωματική εργασία. Συγκεκριμένα, χρειάστηκαν αλλαγές όσον αφορά τις διαστάσεις και τον αριθμό νευρώνων των επιπέδων. Ένα παράδειγμα τροποποίησης για το 1ο συνελικτικό επίπεδο του LeNet-5 παρουσιάζεται στην Εικ. 5.2. Στην περίπτωση αυτή απαιτήθηκε τροποποίηση της παραμέτρου **num_output** σε **6** (το πρώτο συνελικτικό επίπεδο έχει 6 χάρτες γνωρισμάτων εξόδου) και της παραμέτρου **kernel_size** σε **3**, υποδεικνύοντας μέγεθος πυρήνα **3x3**.




Εικ. 5.1: Η αρχιτεκτονική του νευρωνικού δικτύου LeNet-5.

```
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 6
    kernel_size: 3
    stride: 1
    pad: 2
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
```

Εικ. 5.2: Παράδειγμα ορισμού του 1ου συνελικτικού επιπέδου του LeNet-5 σε αρχείο «.prototxt».

Η εκπαίδευση του δικτύου πραγματοποιήθηκε σε περιβάλλον Ubuntu 18.04, ενώ χρησιμοποιήθηκαν οι ίδιες ακριβώς παράμετροι που τέθηκαν στο Keras (βλ. Κεφ. 3). Συγκεκριμένα, χρησιμοποιήθηκε ο ίδιος αλγόριθμος εκπαίδευσης, συνάρτηση σφάλματος, μέγεθος πακέτων εικόνων, ενώ πραγματοποιήθηκε και ίδιο πλήθος περασμάτων από ολόκληρο το σετ εκπαίδευσης. Επομένως, παρότι αυτή τη φορά η εκπαίδευση έγινε με διαφορετικό λογισμικό, εξασφαλίστηκε ότι η εκπαίδευση μέσω του εργαλείου Ristretto

πραγματοποιήθηκε υπό τις ίδιες συνθήκες σε σχέση με την εκπαίδευση μέσω Keras. Δίνοντας εντολή στο εργαλείο Ristretto να συμπιέσει το νευρωνικό δίκτυο με τη μέθοδο δυναμικής σταθερής υποδιαστολής και με περιθώριο πτώσης ακρίβειας ίσο με 1%, λαμβάνουμε την παραμετροποίηση που παρουσιάζεται στο αριστερό μέρος του Πίν. 5.3. Αξίζει να σημειωθεί πως σε κάθε περίπτωση το περισσότερο σημαντικό ψηφίο έχει αρνητικό βάρος, ακολουθώντας τους κανόνες αναπαράστασης συμπληρώματος ως προς 2. Για την περίπτωση που το μήκος του ακεραίου μέρους του αριθμού ισούται με 0, αρνητικό βάρος έχει το πρώτο bit του κλασματικού μέρους. Αυτό συνεπάγεται δυναμικό εύρος ίσο με [-0.5, 0.5).

Ωστόσο, με τη χρήση των αρχικών ζευγών **(IL, FL)**του Πίν. 5.3το LeNet-5, παρουσίαζε λανθασμένη λειτουργικότητα, εξαιτίας του κορεσμού ορισμένων τύπων δεδομένων. Συγκεκριμένα, μερικά από τα ενδιάμεσα δεδομένα του δικτύου ξεπερνούσαν το εύρος τιμών ορισμένων τύπων δεδομένων (π.χ. η είσοδος του 2$^{ου}$συνελικτικού επιπέδου μπορεί να απαιτούσε 9-bit αναπαράσταση αντί για 8-bit). Η σοβαρότερη ένδειξη που υπήρχε για αυτό το πρόβλημα ήταν το γεγονός πως το νευρωνικό δίκτυο αναγνώριζε όλες τις εικόνες σαν το ψηφίο 0. Επομένως, αυξάνοντας κατά 1 το μήκος λέξης κάθε επιπέδου επαναληπτικά και δοκιμάζοντας διαφορετικά ζεύγη **(IL, FL)** για κάθε επίπεδο (κυρίως αυξάνοντας επίσης το **IL** για την αποφυγή του κορεσμού), προέκυψαν οι παράμετροι που φαίνονται στον δεξί μέρος του Πίν. 5.3.

Σύμφωνα με τον Πίν. 5.3, φαίνεται ότι το δίκτυο χρησιμοποιεί 8-bit λέξεις για τα δεδομένα και 4-bit για τα βάρη. Μάλιστα, παρότι το εύρος bit είναι κοινό, υπάρχουν διαφορές όσον αφορά τη θέση της υποδιαστολής (π.χ. (8,0) και (7,1)). Σε σύγκριση, επίσης, με τα μη προσαρμοσμένα ζεύγη (IL, FL) παρατηρείται αύξηση ενός bit στην λέξη εισόδου του 2$^{ου}$ συνελικτικού επιπέδου και αύξηση κατά 2 bits στην τελική έξοδο του δικτύου. Η τελευταία, ωστόσο, δεν εισάγει σημαντική επιβάρυνση, διότι το τελικό επίπεδο αποτελείται από μόνο 10 νευρώνες. Παράλληλα, γίνεται φανερή η σημασία του διαχωρισμού του κάθε επιπέδου σε είσοδο, βάρη και έξοδο, αφού τα βάρη συνήθως είναι αρκετά μικρότερα από τα δεδομένα [23]. Τέλος, επισημαίνεται πως τα επίπεδα μέσης υποδειγματοληψίας και το επίπεδο ισοπέδωσης δεν έχουν βάρη και καταγράφονται μόνο τα **(IL, FL)$_{in}$**, και **(IL, FL)$_{out}$** προκειμένου να είναι συνεπή με τα γειτονικά τους επίπεδα.

Πίν. 5.3: Αρχικά και προσαρμοσμένα ζεύγη (IL, FL) του εργαλείου Ristretto.

| **Επίπεδο** | **Ristretto** | | | **Χρησιμοποιούμενα** | | |
|---|---|---|---|---|---|---|
| | είσοδος | βάρη | έξοδος | είσοδος | βάρη | έξοδος |
| Συνελικτικό 1 | (8,0) | (1,3) | (8,0) | (8,0)[1] | (0,4) | (8,0) |
| Μέσης υποδειγματοληψίας 1 | (8,0) | - | (8,0) | (8,0) | - | (9,0) |
| Συνελικτικό 2 | (8,0) | (0,4) | (8,0) | (9,0) | (0,4) | (8,0) |
| Μέσης υποδειγματοληψίας 2 | (8,0) | - | (7,1) | (8,0) | - | (7,1) |
| Πλήρως διασυνδεδεμένο 1 | (7,1) | (0,4) | (8,0) | (7,1) | (0,4) | (8,0) |
| Πλήρως διασυνδεδεμένο 2 | (8,0) | (0,4) | (8,0) | (8,0) | (0,4) | (8,0) |
| Πλήρως διασυνδεδεμένο 3 | (8,0) | (0,4) | (7,1) | (8,0) | (0,4) | (6,4) |

## 5.4 Σύγκριση μεθόδων συμπίεσης

Τα αποτελέσματα που αναφέρονται στην παρούσα παράγραφο λήφθηκαν με τη χρήση του εργαλείου Vivado HLS. Όπως αναφέρθηκε και στην Παρ. 4.2, πρόκειται περί ενός λογισμικού το οποίο μετατρέπει κώδικα λογισμικού σε γλώσσα περιγραφής υλικού και μπορεί να χρησιμοποιηθεί για τον προγραμματισμό FPGAs.

### 5.4.1 Αποτελέσματα C προσομοίωσης

Το Vivado HLS διαθέτει ειδικά σχεδιασμένη λειτουργία προσομοίωσης του συστήματος, μέσω ρουτίνας λογισμικού (C/C++) για την διασφάλιση της ορθής λειτουργικότητας του εκάστοτε σχεδιασμού. Για το σκοπό αυτό αναπτύχθηκε κατάλληλο ελεγκτικό πρόγραμμα για τον έλεγχο της εγκυρότητας του νευρωνικού δικτύου (βλ. Παρ. 4.5).

Εφαρμόζοντας το πρόγραμμα αυτό για διάφορες παραμετροποιήσεις προέκυψαν οι TOP-1, TOP-3 και TOP-5 ακρίβειες κατηγοριοποίησης που παρουσιάζονται στον Πίν. 5.4 Οι ακρίβειες αυτές μετρήθηκαν σε ένα σύνολο 1.000 εικόνων εισόδου. Αξίζει να σημειωθεί ότι οι εικόνες αυτές ανήκουν στο σετ δοκιμών του MNIST και επομένως αναγράφονται οι ακρίβειες δοκιμών του δικτύου.

[1] Επειδή είναι γνωστό ότι τα εικονοστοιχεία των εικόνων έχουν αποκλειστικά θετικές τιμές, ο τύπος δεδομένων της εισόδου του νευρωνικού δικτύου ορίζεται ως μη-προσημασμένος.

Ως σημεία αναφοράς ορίζονται οι ακρίβειες της παραμετροποίησης **κινητής υποδιαστολής 32-bit**. Αρχικά, γίνεται φανερό ότι οι ίδιες επιδόσεις επιτυγχάνονται με κβαντισμούς παραμέτρων **κινητής υποδιαστολής 16-bit**, αλλά και με **ομοιόμορφης σταθερής υποδιαστολής 32-bit**. Παράλληλα, με αριθμητική **ομοιόμορφης σταθερής υποδιαστολής 16-bit** επιτυγχάνεται περίπου 1% πτώση στην TOP-1 ακρίβεια, μειώνοντας, ωστόσο, το μήκος λέξης στο μισό. Καλές επιδόσεις παρουσιάζουν επίσης η παραμετροποίηση **δυναμικής σταθερής υποδιαστολής (από Ristretto)**, καθώς και η μέθοδος **ομοιόμορφης σταθερής υποδιαστολής (από** [22]**)**, έχοντας πτώση περίπου 2% (παρά το αρχικό περιθώριο 1% που είχε δοθεί σαν είσοδος στο εργαλείο Ristretto στην Υποπαρ. 5.3.3) και 3% TOP-1 ακρίβειας, αντίστοιχα.

Πίν. 5.4: Σύγκριση της ακρίβειας του LeNet-5 για διαφορετικές παραμετροποιήσεις. Οι αναγραφόμενες ακρίβειες έχουν μετρηθεί για 1000 εικόνες από το σετ δοκιμών.

| **Παραμετροποίηση** | **Ακρίβεια (%)** | | |
|---|---|---|---|
| | TOP-1 | TOP-3 | TOP-5 |
| Κινητής υποδιαστολής 32-bit | 98.3 | 100.0 | 100.0 |
| Κινητής υποδιαστολής 16-bit | 98.3 | 100.0 | 100.0 |
| Ομοιόμορφης σταθερής υποδιαστολής 32-bit | 98.3 | 100.0 | 100.0 |
| Ομοιόμορφης σταθερής υποδιαστολής 16-bit | 97.5 | 99.6 | 99.6 |
| Ομοιόμορφης σταθερής υποδιαστολής 8-bit | 51.7 | 74.8 | 77.0 |
| Ομοιόμορφης σταθερής υποδιαστολής (από [22]) | 95.2 | 97.6 | 97.6 |
| Δυναμικής σταθερής υποδιαστολής (από [22]) | 52.4 | 60.6 | 60.6 |
| Δυναμικής σταθερής υποδιαστολής (από Ristretto) | 96.1 | 99.3 | 99.4 |

Οι παραμετροποιήσεις **ομοιόμορφης σταθερής υποδιαστολής 8-bit** και **δυναμικής σταθερής υποδιαστολής (από** [22]**)** εμφανίζουν κακές επιδόσεις και δεν ολοκληρώνουν επιτυχώς τους ελέγχους ακριβείας. Συγκεκριμένα, θεωρήθηκε ότι το νευρωνικό δίκτυο περνά με επιτυχία το ελεγκτικό πρόγραμμα αν έχει τουλάχιστον 90% TOP-1 ακρίβεια. Αξιοσημείωτο αποτελεί το γεγονός πως, παρότι η παραμετροποίηση **δυναμικής σταθερής υποδιαστολής (από Ristretto)** χρησιμοποιεί μικρότερα μήκη λέξεων (4-bit για τα βάρη), παρουσιάζει υψηλότερες ακρίβειες από την έκδοση **ομοιόμορφης σταθερής υποδιαστολής 8-bit**, λόγω της μεθοδικότερης κατανομής των ψηφίων. Σημειώνεται, επίσης, ότι οι

παραμετροποιήσεις ομοιόμορφου κβαντισμού σταθερής υποδιαστολής αποτελούνται από τον ίδιο αριθμό **IL** και **FL** bits. Έχουν, δηλαδή, ίσο αριθμό ακέραιων και δεκαδικών ψηφίων.

Τέλος, είναι εμφανές πως οι ακρίβειες TOP-3 και TOP-5 έχουν πολύ μικρές διαφορές. Επομένως, επειδή ο αριθμός των κλάσεων είναι 10, θεωρούνται ως πιο αξιόπιστα μεγέθη οι ακρίβειες TOP-1 και TOP-3. Ωστόσο, όπως φάνηκε και παραπάνω για τη σύγκριση των επιδόσεων, χρησιμοποιείται κατά βάση η TOP-1 ακρίβεια.

### 5.4.2 Αποτελέσματα σύνθεσης

Το Vivado HLS διαθέτει, επίσης, την λειτουργία σύνθεσης όπου μεταφράζει το σχεδιασμό σε γλώσσα περιγραφής υλικού (βλ. Κεφ. 4). Παράλληλα, υπολογίζει την αξιοποίηση των μπλοκ του FPGA, καθώς και την *καθυστέρηση* (*latency*) και *διεκπεραιωτική ικανότητα* (*throughput*) του σχεδιασμού. Για τα αποτελέσματα που ακολουθούν χρησιμοποιήθηκε το **Kintex-7 FPGA xc7k160tfbg484-1** της Xilinx και ρολόι περιόδου **10 ns**.

Προκειμένου να σχηματιστεί μια καλύτερη εικόνα όσον αφορά τη χρήση πόρων που πραγματοποιούν οι διάφορες παραμετροποιήσεις, κρίνεται απαραίτητη μια σύντομη αναφορά στις βασικές δομικές μονάδες ενός FPGA. Συγκεκριμένα, τα βασικά μπλοκ ενός FPGA της Xilinx είναι τα ακόλουθα [18]:

1. **LookUp Table (LUT)**: Πρόκειται για βασικό μπλοκ το οποίο μπορεί να υλοποιήσει λογικές συναρτήσεις. Μπορεί παράλληλα να χρησιμοποιηθεί και ως μικρή μνήμη, αφού ουσιαστικά συγκρατεί έναν πίνακα αληθείας.
2. **Flip Flop (FF)**: Αποτελεί το βασικότερο στοιχείο μνήμης το οποίο συγκρατεί ενδιάμεσες τιμές.
3. **DSP48**: Πρόκειται για μια *Αριθμητική Λογική Μονάδα* (*Arithmetic Logic Unit – ALU*) η οποία περιέχει έναν πολλαπλασιαστή 25x18 bit, καθώς και συσσωρευτή 48-bit.
4. **Block RAM (BRAM)**: Οι μονάδες BRAM είναι δίθυρες μνήμες που αποθηκεύουν μεγαλύτερο όγκο δεδομένων (18 kbits ή 36 kbits). Ως δίθυρες χαρακτηρίζονται οι μνήμες οι οποίες επιτρέπουν έως δύο προσπελάσεις δεδομένων σε έναν κύκλο ρολογιού.

Στον Πίν. 5.5 παρουσιάζεται η ποσοστιαία χρήση πόρων του FPGA για τις διάφορες παραμετροποιήσεις. Παρατηρείται πως οι παραμετροποιήσεις που χαρακτηρίζονταν από μη αποδεκτή ακρίβεια (συγκεκριμένα η **ομοιόμορφης σταθερής υποδιαστολής 8-bit** και η **δυναμικής σταθερής υποδιαστολής (από** [22]**)**) απουσιάζουν, διότι δεν έχει νόημα η υλοποίησή τους. Με μια πρώτη ματιά γίνεται εμφανές πως η χρήση μνημών (BRAM) φθίνει στις πιο συμπιεσμένες υλοποιήσεις, γεγονός το οποίο είναι αναμενόμενο. Αξιοσημείωτο αποτελεί το γεγονός ότι ο αρχικός σχεδιασμός (**κινητής υποδιαστολής 32-bit**) δεσμεύει τις μισές BRAM του FPGA, αποδεικνύοντας την αναγκαιότητα σύμπτυξης του νευρωνικού δικτύου. Παράλληλα, οι αρχιτεκτονικές σταθερής υποδιαστολής απαιτούν ελάχιστες έως μηδαμινές μονάδες DSP48 σε αντίθεση με τις αντίστοιχες κινητής υποδιαστολής. Όσον αφορά τα FFs και τα LUTs παρατηρούνται μικρότερες διαφορές ανάμεσα στις διαφορετικές παραμετροποιήσεις.

Πίν. 5.5: Σύγκριση της χρήσης πόρων του LeNet-5 για διαφορετικές παραμετροποιήσεις.

| **Παραμετροποίηση** | **Χρήση Πόρων (%)** | | | |
|---|---|---|---|---|
| | BRAM-18K | DSP48E | FF | LUT |
| Κινητής υποδιαστολής 32-bit | 50 | 5 | 2 | 8 |
| Κινητής υποδιαστολής 16-bit | 25 | 4 | 1 | 5 |
| Ομοιόμορφης σταθερής υποδιαστολής 32-bit | 28 | 1 | 1 | 8 |
| Ομοιόμορφης σταθερής υποδιαστολής 16-bit | 14 | ≈0 | ≈0 | 6 |
| Ομοιόμορφης σταθερής υποδιαστολής (από [22]) | 12 | ≈0 | ≈0 | 6 |
| Δυναμικής σταθερής υποδιαστολής (από Ristretto) | 7 | 0 | ≈0 | 5 |

Εκτός από τη σύγκριση της χρήσης πόρων μεταξύ των διαφόρων παραμετροποιήσεων, η λειτουργία σύνθεσης του εργαλείου Vivado HLS, καθιστά δυνατή και τη σύγκριση της ταχύτητάς τους. Ειδικότερα, ο Πίν. 5.6 παρουσιάζει τις καθυστερήσεις σε ms των παραμετροποιήσεων αυτών. Είναι εμφανές πως οι **αρχιτεκτονικές αριθμητικής κινητής υποδιαστολής** χαρακτηρίζονται από μεγαλύτερες καθυστερήσεις, με την παραμετροποίηση **κινητής υποδιαστολής 16-bit** να έχει τη μέγιστη. Αντιθέτως οι **αρχιτεκτονικές σταθερής υποδιαστολής**, εκτός από το ότι επιτυγχάνουν υψηλότερες ταχύτητες, χαρακτηρίζονται και από αποδοτικότερη αξιοποίηση πόρων.

Πίν. 5.6: Σύγκριση καθυστέρησης του LeNet-5 για διαφορετικές παραμετροποιήσεις.

| **Παραμετροποίηση** | **Καθυστέρηση (ms)** |
|---|---|
| Κινητής υποδιαστολής 32-bit | 28.3 |
| Κινητής υποδιαστολής 16-bit | 31.0 |
| Ομοιόμορφης σταθερής υποδιαστολής 32-bit | 13.3 |
| Ομοιόμορφης σταθερής υποδιαστολής 16-bit | 10.6 |
| Ομοιόμορφης σταθερής υποδιαστολής (από [22]) | 10.6 |
| Δυναμικής σταθερής υποδιαστολής (από Ristretto) | 10.6 |

Η παραμετροποίηση που δόθηκε από το Ristretto χρησιμοποιεί τους λιγότερους πόρους σύμφωνα με τον Πίν. 5.5. Αναλυτικότερα, απαιτεί μόλις το 14% των μνημών που χρειάζεται το αρχικό δίκτυο (διάταξη **κινητής υποδιαστολής 32-bit**), ενώ δε χρησιμοποιεί καθόλου μονάδες DSP48. Το τελευταίο υποδεικνύει ότι οι αριθμητικές πράξεις εκτελούνται από τα μικρότερα και ταχύτερα LUTs. Επιπλέον, η δέσμευση μικρότερης επιφάνειας έχει σαν αποτέλεσμα την ελάττωση της *καθυστέρησης των καλωδιώσεων* (*routing delay*), το οποίο σε συνδυασμό με την επιτάχυνση των πράξεων, προσδίδει στο LeNet-5 σημαντική ταχύτητα. Με τον τρόπο αυτό η παραμετροποίηση που λήφθηκε από το εργαλείο Ristretto σχεδόν τριπλασιάζει την ταχύτητα σε σχέση με την αρχική (παραμετροποίηση **κινητής υποδιαστολής 32-bit**), όπως φαίνεται στον Πίν. 5.6.

Στην παρούσα διπλωματική εργασία υιοθετήθηκε η παραμετροποίηση που προέκυψε από το Ristretto για περεταίρω μελέτη και βελτιστοποιήσεις. Η συρρίκνωση της επιφάνειας του LeNet-5 σε συνδυασμό με την επιτάχυνση που επιτυγχάνεται συνιστούν τους βασικούς παράγοντες της συγκεκριμένης επιλογής. Παρά τη πτώση της ακρίβειας (2.2%), η παραμετροποίηση αυτή αποτελεί έναν καλό συμβιβασμό μεταξύ συμπίεσης και επιδόσεων.

# 6. ΒΕΛΤΙΣΤΟΠΟΙΗΣΕΙΣ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ

## 6.1 Εισαγωγή

Η εξασφάλιση της ορθής λειτουργίας του νευρωνικού δικτύου LeNet-5 (βλ. Παρ. 4.5), καθώς και η συμπίεσή του (βλ. Κεφ. 5), αποτελούν τα πρώτα βήματα της υλοποίησής του σε υλικό. Ωστόσο, υπάρχει περιθώριο βελτιστοποίησης τόσο ως προς την αποτελεσματικότερη χρήση πόρων, όσο και ως προς την παράλληλη επεξεργασία των δεδομένων. Εξάλλου, είναι σημαντικό πως, εξαιτίας της εξοικονόμησης πόρων που επιτεύχθηκε από την παραμετροποίηση που προέκυψε από το εργαλείο Ristretto (Πίν. 5.5), υπάρχει αρκετός χώρος για πειραματισμούς. Όπως γίνεται φανερό και στην πορεία, πολλές αρχιτεκτονικές δεσμεύουν αρκετά μεγάλο ποσοστό του διαθέσιμου υλικού. Άλλωστε, το τελευταίο αποτελεί βασικό αντίτιμο της επιτάχυνσης ενός σχεδιασμού.

Το κεφάλαιο αυτό ξεκινά με την παρουσίαση ορισμένων τεχνικών βελτιστοποίησης. Αυτές αφορούν αναδόμηση σε επίπεδο πηγαίου κώδικα, αλλά κυρίως μετασχηματισμούς σε αυτόν και στο παραχθέν κύκλωμα. Παράλληλα, γίνεται αναφορά στις εξαρτήσεις δεδομένων, τις οποίες πρέπει να λαμβάνει υπόψιν του ο σχεδιαστής κατά την βελτιστοποίηση του κώδικα. Έπειτα, περιγράφονται λεπτομέρειες που σχετίζονται με την εφαρμογή των εν λόγω μετασχηματισμών στο εργαλείο Vivado HLS, καθώς και οι λόγοι για τους οποίους χρησιμοποιήθηκαν οι μετασχηματισμοί αυτοί. Τέλος, επιλέγονται και συγκρίνονται ορισμένες από τις αρχιτεκτονικές που προέκυψαν κατά τη διαδικασία βελτιστοποίησης.

## 6.2 Αναδόμηση του κώδικα

Το πρώτο βήμα για την βελτίωση της επίδοσης του παραγομένου εκτελέσιμου, και δη, του αντίστοιχου κυκλώματος, αποτελεί η επεξεργασία του κώδικα από τον σχεδιαστή. Σε πολλές περιπτώσεις, η πιο ευανάγνωστη και καλύτερα δομημένη (οπτικά) μορφή ενός αλγορίθμου, δεν αποτελεί και την ιδανικότερη υλοποίησή του.

Ειδικά κατά τη υλοποίηση συστήματος μέσω εργαλείου σχεδίασης υψηλού επιπέδου, πρέπει να δοθεί ιδιαίτερη έμφαση στην αποτελεσματικότερη χρήση των στοιχείων αποθήκευσης (ανάγνωση / εγγραφή σε πίνακες), περιορίζοντας τα σημεία προσπέλασης σε αυτά. Καθώς οι διαδικασίες αυτές είναι χρονοβόρες, στα στοιχεία εκείνα πρέπει να εγγράφονται και να διαβάζονται δεδομένα όσο το δυνατόν πιο σπάνια, καθιστώντας την λειτουργία του συστήματος λιγότερο κοστοβόρα χρονικά. Οι αλλαγές που πραγματοποιήθηκαν παρουσιάζονται στην Υποπαρ. 6.2.1.

Παράλληλα, είναι απαραίτητη η αποδοτική εκμετάλλευση των επεξεργαστικών πόρων, έτσι ώστε να περιοριστούν τα βήματα των διεργασιών. Εκτελώντας μονομερώς όλες τις απαραίτητες πράξεις επί των δεδομένων, τα στάδια επεξεργασίας συγχωνεύονται και, συνεπώς, μειώνονται. Αυτό έχει ως αποτέλεσμα να ελαττώνεται το *κρίσιμο μονοπάτι* (*critical path*), το οποίο επηρεάζει και τη συνολική καθυστέρηση. Η μέθοδος αυτή εφαρμόστηκε με επιτυχία, αφού ενσωματώθηκαν οι συναρτήσεις ενεργοποίησης εντός άλλων υπαρχόντων επαναληπτικών δομών (βλ. Υποπαρ. 6.2.2).

### 6.2.1 Μη-Διπλότυπη προσπέλαση στοιχείων – Παράκαμψη αρχικοποιήσεων

Εξετάζοντας την βελτιστοποίηση του κατασκευασμένου κυκλώματος, παρατηρήθηκε ελαφρώς αυξημένη καθυστέρηση και χρήση πόρων από κάποια ζεύγη επαναληπτικών δομών (for). Για την εξήγηση αυτού του φαινομένου, απαιτείται μία περεταίρω ανάλυση του αλγορίθμου.

Αρχικά, εξετάζεται η περίπτωση των νευρώνων ενός πλήρους επιπέδου, με τα αντίστοιχα να ισχύουν και για τα υπόλοιπα επίπεδα, χωρίς βλάβη της γενικότητας. Όπως προαναφέρθηκε στην Υποπαρ. 2.2.1, οι νευρώνες σε πλήρη επίπεδα νευρωνικών δικτύων υπολογίζουν την εσωτερική και εξωτερική τους κατάσταση, βάσει των Εξ. 6.1 και Εξ. 6.2 που

επαναλαμβάνονται εδώ για διευκόλυνση του αναγνώστη (αναφέρονται ως Εξ. 2.1 και Εξ. 2.2 στην Υποπαρ. 2.2.1).

$$y_i = \sum_{j=1}^{M} w_{i,j} \cdot o_j + b_i \qquad \text{Εξ. 6.1}$$

$$o_i = f(y_i) \qquad \text{Εξ. 6.2}$$

Σημειώνεται ότι για τις Εξ. 6.1 και Εξ. 6.2 ισχύουν τα εξής:

- $\mathbf{y_i}$, $\mathbf{o_i}$ η εσωτερική και εξωτερική κατάσταση του νευρώνα **i** αντίστοιχα,
- $\mathbf{w_{i,j}}$ το βάρος του νευρώνα **j** προς τον **i**,
- $\mathbf{o_j}$ η εξωτερική κατάσταση του νευρώνα **j**,
- $\mathbf{b_i}$ η σταθερά αρχικοποίησης του νευρώνα **i** και
- **f(y)** η συνάρτηση ενεργοποίησης.

Μοντελοποιώντας τα παραπάνω σε μορφή ψευδογλώσσας, η λειτουργία ενός πλήρους επιπέδου του νευρωνικού δικτύου συνοψίζεται στον αλγόριθμο που φαίνεται στην Εικ. 6.1. Παρατηρείται η ύπαρξη τριών δομών επανάληψης:

**Στάδιο 1:** Αρχικοποίηση των εσωτερικών καταστάσεων (παράγοντας $\mathbf{b_i}$).

**Στάδιο 2:** Προσθετικοί πολλαπλασιασμοί για την διαμόρφωση της τελικής τιμής των εσωτερικών καταστάσεων ($\mathbf{x_i}$).

**Στάδιο 3:** Εφαρμογή της συνάρτησης ενεργοποίησης ($\mathbf{f(x_i)}$) για την παραγωγή των εξωτερικών καταστάσεων ($\mathbf{o_i}$).

Συγκεκριμένα, τα στάδια 1 και 2 που περιγράφουν ουσιαστικά την Εξ. 6.1, δηλώνουν εγγραφή σε ίδια στοιχεία πίνακα σε διαφορετικές επαναληπτικές δομές. Σύμφωνα με τις υποδείξεις [16], διαπιστώθηκε ότι επιτυγχάνεται αξιοσημείωτη βελτίωση στις υπολογιστικές απαιτήσεις του συστήματος ενσωματώνοντας την αρχικοποίηση των πινάκων εντός της κύριας επαναληπτικής δομής (Εικ. 6.2). Έτσι, μειώνεται η πολυπλοκότητα του κώδικα, ενώ το εργαλείο αντιλαμβάνεται καλύτερα τον τρόπο με τον οποίο θα πραγματοποιηθούν οι προσπελάσεις στα δεδομένα και παράγει καλύτερες υλοποιήσεις.

```
// Ορισμός δεδομένων
Έστω Χ επίπεδο νευρωνικού δικτύου
Έστω Υ επίπεδο νευρωνικού δικτύου που βρίσκεται ακριβώς πριν το επίπεδο Χ
Έστω ν νευρώνας του επιπέδου
Έστω βάρος(ν,μ) το βάρος που συνδέει νευρώνα μ του επιπέδου Υ με νευρώνα ν του επιπέδου Χ

// Αρχικοποίηση εσωτερικών καταστάσεων
Για κάθε νευρώνα ν του επιπέδου_Χ:
    εσωτερική_κατάσταση(ν) = bias(ν)

// Υπολογισμός εσωτερικών καταστάσεων
Για κάθε νευρώνα ν του επιπέδου_Χ:
    Για κάθε νευρώνα μ του επιπέδου_Υ:
        εσωτερική_κατάσταση(ν) += εξωτερική_κατάσταση(μ) * βάρος(ν,μ)

// Εφαρμογή συνάρτησης ενεργοποίησης και υπολογισμός εξωτερικών καταστάσεων
Για κάθε νευρώνα ν του επιπέδου_Χ:
    εξωτερική_κατάσταση(ν) = ReLU(εσωτερική_κατάσταση(ν))
```

Εικ. 6.1: Αλγόριθμος πλήρους επιπέδου πριν την ενσωμάτωση της αρχικοποίησης των νευρώνων εντός της κύριας δομής.

```
// Ορισμός δεδομένων
Έστω Χ επίπεδο νευρωνικού δικτύου
Έστω Υ επίπεδο νευρωνικού δικτύου που βρίσκεται ακριβώς πριν το επίπεδο Χ
Έστω βάρος(ν,μ) το βάρος που συνδέει νευρώνα μ του επιπέδου Υ με νευρώνα ν του επιπέδου Χ

// Αρχικοποίηση και υπολογισμός εσωτερικών καταστάσεων
Για κάθε νευρώνα ν του επιπέδου_Χ:
    εσωτερική_κατάσταση(ν) = bias(ν)
    Για κάθε νευρώνα μ του επιπέδου_Υ:
        εσωτερική_κατάσταση(ν) += εξωτερική_κατάσταση(μ) * βάρος(ν,μ)

// Εφαρμογή συνάρτησης ενεργοποίησης και υπολογισμός εξωτερικών καταστάσεων
Για κάθε νευρώνα ν του επιπέδου_Χ:
    εξωτερική_κατάσταση(ν) = ReLU(εσωτερική_κατάσταση(ν))
```

Εικ. 6.2: Αλγόριθμος πλήρους επιπέδου μετά την ενσωμάτωση της αρχικοποίησης των νευρώνων εντός της κύριας δομής.

Αντίστοιχα με την περίπτωση των πλήρων επιπέδων, υπήρχαν αντίστοιχες δομές αρχικοποίησης και στα συνελικτικά επίπεδα. Για τη προσομοίωση της λειτουργίας ενός συνελικτικού επιπέδου, συγγράφηκε ο κώδικας της Εικ. 6.3. Παρατηρήθηκε, ότι η επαναληπτική δομή που αρχικοποιεί τον πίνακα εξόδου μπορεί να παραληφθεί πλήρως, αν προφορτωθεί στον προσωρινό συσσωρευτή η εκάστοτε σταθερά αρχικοποίησης (bias), κατά την αποθήκευση της τιμής του στον πίνακα εξόδου (Εικ. 6.4). Δόθηκε ιδιαίτερη προσοχή έτσι ώστε ο συνυπολογισμός της αρχικοποίησης να πραγματοποιηθεί μόνο μία φορά ανά νευρώνα. Συνεπώς, για τις δύο λειτουργίες (αρχικοποίηση και ενεργοποίηση των νευρώνων) δομήθηκε μία υλοποίηση με μόνο μία εγγραφή στην μήτρα εξόδου, μειώνοντας τους απαιτούμενους κύκλους ρολογιού και προσπελάσεις μνήμης.

```
// Ορισμός δεδομένων
Έστω Χ επίπεδο νευρωνικού δικτύου, με εισ και εξ η είσοδος και έξοδος του αντίστοιχα
Έστω βάρος(ν, φ, α, β) ένα στοιχείο του δισδιάστατου φίλτρου φ του νευρώνα ν
Έστω μία προσωρινή μεταβλητή συσσωρετής

// Ενεργοποίηση επιπέδου
Για κάθε νευρώνα ν του επιπέδου_Χ:
    // Αρχικοποίηση εσωτερικών καταστάσεων
    Για κάθε γραμμή γ της εξόδου:
        Για κάθε στήλη σ της εξόδου:
            εξ(ν)(γ)(σ) = bias(ν)
    // Υπολογισμός εσωτερικών καταστάσεων νευρώνων
    Για κάθε φίλτρο φ:
        Για κάθε γραμμή γ της εξόδου:
            Για κάθε στήλη σ της εξόδου:
                συσσωρευτής = 0
                Για α από -1 μέχρι 1:
                    Για β από -1 μέχρι 1:
                        συσσωρευτής += εισ(φ)(γ+α)(σ+β) * βάρος(ν, φ, α+1, β+1)
                εξ(ν)(γ)(σ) += συσσωρευτής

// Εφαρμογή συνάρτησης ενεργοποίησης (ReLU) και υπολογισμός εξωτερικών καταστάσεων
Για κάθε νευρώνα ν του επιπέδου_Χ:
    // Εφαρμογή συναρτήσεων ενεργοποίησης
    Για κάθε γραμμή γ της εξόδου:
        Για κάθε στήλη σ της εξόδου:
            εξ(ν)(γ)(σ) = ReLU(εξ(ν)(γ)(σ))
```

Εικ. 6.3: Αλγόριθμος συνελικτικού επιπέδου πριν την ενσωμάτωση της αρχικοποίησης των νευρώνων εντός της κύριας δομής.

```
// Ορισμός δεδομένων
Έστω Χ επίπεδο νευρωνικού δικτύου, με εισ και εξ η είσοδος και έξοδος του αντίστοιχα
Έστω βάρος(ν, φ, α, β) ένα στοιχείο του δισδιάστατου φίλτρου φ του νευρώνα ν
Έστω μία προσωρινή μεταβλητή συσσωρετής

// Ενεργοποίηση επιπέδου
Για κάθε νευρώνα ν του επιπέδου_Χ:
    // Αρχικοποίηση και υπολογισμός εσωτερικών καταστάσεων νευρώνων
    Για κάθε φίλτρο φ:
        Για κάθε γραμμή γ της εξόδου:
            Για κάθε στήλη σ της εξόδου:
                συσσωρευτής = 0
                Για α από -1 μέχρι 1:
                    Για β από -1 μέχρι 1:
                        συσσωρευτής += εισ(φ)(γ+α)(σ+β) * βάρος(ν, φ, α+1, β+1)
                Αν φ είναι το πρώτο φίλτρο:
                    εξ(ν)(γ)(σ) += συσσωρευτής + bias(ν)
                Αλλιώς:
                    εξ(ν)(γ)(σ) += συσσωρευτής

// Εφαρμογή συνάρτησης ενεργοποίησης (ReLU) και υπολογισμός εξωτερικών καταστάσεων
Για κάθε νευρώνα ν του επιπέδου_Χ:
    // Εφαρμογή συναρτήσεων ενεργοποίησης
    Για κάθε γραμμή γ της εξόδου:
        Για κάθε στήλη σ της εξόδου:
            εξ(ν)(γ)(σ) = ReLU(εξ(ν)(γ)(σ))
```

Εικ. 6.4: Αλγόριθμος συνελικτικού επιπέδου μετά την ενσωμάτωση της αρχικοποίησης των νευρώνων εντός της κύριας δομής.

### 6.2.2 Ενσωμάτωση συναρτήσεων ενεργοποίησης

Εξετάζοντας περεταίρω την αλγοριθμική αναπαράσταση της ενεργοποίησης του δικτύου, έγινε μία επιπλέον διαπίστωση. Επιπρόσθετη κατάργηση επαναληπτικών βρόχων επιτυγχάνεται με την ενσωμάτωση των συναρτήσεων ενεργοποίησης, ή μέρους αυτών, εντός των ήδη υπαρχόντων βρόχων που περιγράφουν την ενεργοποίηση των νευρώνων.

Συγκεκριμένα, συνεχίζοντας τα παραπάνω παραδείγματα με την ReLU ως συνάρτηση ενεργοποίησης, καταργήθηκε η τελική επαναληπτική δομή και προστέθηκε η ReLU στο τέλος της διεργασίας που αφορά τον υπολογισμό των καταστάσεων των νευρώνων (Εικ. 6.5 για πλήρη και Εικ. 6.6 για συνελικτικά επίπεδα). Αντίστοιχα, πράχθηκε και για την Softmax, όπου αυτό απαιτήθηκε. Σε αυτή την περίπτωση, ωστόσο, η πλήρης ενσωμάτωση δεν είναι δυνατή, καθώς αυτή η συνάρτηση ενεργοποίησης απαιτεί τον υπολογισμό των καταστάσεων όλων των νευρώνων του εκάστοτε επιπέδου.

```
// Ορισμός δεδομένων
Έστω X επίπεδο νευρωνικού δικτύου
Έστω Y επίπεδο νευρωνικού δικτύου που βρίσκεται ακριβώς πριν το επίπεδο X
Έστω βάρος(ν,μ) το βάρος που συνδέει νευρώνα μ του επιπέδου Y με νευρώνα ν του επιπέδου X

// Αρχικοποίηση και υπολογισμός εσωτερικών καταστάσεων
Για κάθε νευρώνα ν του επιπέδου_X:
    εσωτερική_κατάσταση(ν) = bias(ν)
    Για κάθε νευρώνα μ του επιπέδου_Y:
        εσωτερική_κατάσταση(ν) += εξωτερική_κατάσταση(μ) * βάρος(ν,μ)
    // Εφαρμογή συνάρτησης ενεργοποίησης και υπολογισμός εξωτερικών καταστάσεων
    εξωτερική_κατάσταση(ν) = ReLU(εσωτερική_κατάσταση(ν))
```

Εικ. 6.5: Αλγόριθμος πλήρους επιπέδου με ενσωμάτωση της αρχικοποίησης των νευρώνων και της συνάρτησης ενεργοποίησης εντός της κύριας δομής.

```
// Ορισμός δεδομένων
Έστω Χ επίπεδο νευρωνικού δικτύου, με εισ και εξ η είσοδος και έξοδος του αντίστοιχα
Έστω βάρος(ν, φ, α, β) το δισδιάστατο φίλτρο φ του νευρώνα ν
Έστω μία προσωρινή μεταβλητή συσσωρετής

// Ενεργοποίηση επιπέδου
Για κάθε νευρώνα ν του επιπέδου_Χ:
    // Αρχικοποίηση και υπολογισμός εσωτερικών καταστάσεων νευρώνων
    Για κάθε φίλτρο φ:
        Για κάθε γραμμή γ της εξόδου:
            Για κάθε στήλη σ της εξόδου:
                συσσωρευτής = 0
                Για α από -1 μέχρι 1:
                    Για β από -1 μέχρι 1:
                        συσσωρευτής += εισ(φ)(γ+α)(σ+β) * βάρος(ν, φ, α+1, β+1)
                Αν φ είναι το πρώτο φίλτρο:
                    εξ(ν)(γ)(σ) += συσσωρευτής + bias(ν)
                Αλλιώς:
                    εξ(ν)(γ)(σ) += συσσωρευτής

    // Εφαρμογή συνάρτησης ενεργοποίησης (ReLU) και υπολογισμός εξωτερικών καταστάσεων
    Για κάθε γραμμή γ της εξόδου:
        Για κάθε στήλη σ της εξόδου:
            εξ(ν)(γ)(σ) = ReLU(εξ(ν)(γ)(σ))
```

Εικ. 6.6: Αλγόριθμος συνελικτικών επιπέδου με ενσωμάτωση της αρχικοποίησης των νευρώνων και της συνάρτησης ενεργοποίησης εντός της κύριας δομής.

## 6.3 Μετασχηματισμοί κώδικα

Πριν την επίδραση του μεταγλωττιστή στον κώδικα του κυκλώματος ώστε να παραχθεί το εκτελέσιμο, ο κώδικας αυτός δύναται, βάσει οδηγιών του σχεδιαστή, να υποστεί κάποιους μετασχηματισμούς από τον προεπεξεργαστή. Ο προεπεξεργαστής αναλύει τον κώδικα και τις οδηγίες (*pragmas* και *directives*) που παρατίθενται, και μεταποιεί αντιστοίχως τις εντολές, πριν αυτές χρησιμοποιηθούν για την παραγωγή του προγράμματος. Στόχος είναι η μεταβολή της δομής, των χρησιμοποιούμενων πόρων και των επιδόσεων του συστήματος, δεδομένου όμως ότι η συμπεριφορά του, ως προς την γενική λειτουργία και τα ζεύγη εισόδων – εξόδων, παραμένει η ίδια [29].

Σε ορισμένες περιπτώσεις οι μετασχηματισμοί αυτοί δεν αφορούν κάποια αλλαγή στον κώδικα, αλλά το πώς ο κώδικας θα μεταφραστεί σε συνθέσιμο κύκλωμα. Ο σχεδιαστής έχει την επιλογή να επηρεάσει σε βάθος τους πόρους που θα ανατεθούν, όπως οι μνήμες και οι καταχωρητές. Για παράδειγμα, στο επίπεδο ελέγχου και διάτμησης των λειτουργικών συστημάτων του κυκλώματος, δίνεται η δυνατότητα για δομές *διοχέτευσης βρόχων* (*pipelining*) και *βελτιστοποίησης ροής δεδομένων* (*dataflow*), αυξάνοντας τη διεκπεραιωτική ικανότητα του κυκλώματος.

Οι μετασχηματισμοί αυτοί διακρίνονται σε κατηγορίες ανάλογα με τα δομικά προγραμματιστικά στοιχεία που επηρεάζουν. Αυτά πρόκεινται είτε για δομές δεδομένων, όπως οι πίνακες ή οι μεταβλητές, είτε για ακολουθίες εντολών, όπως οι συναρτήσεις ή οι επαναληπτικές δομές «for».

### 6.3.1 Εξαρτήσεις δεδομένων

Για να μπορούν όμως οι μετασχηματισμοί να εφαρμοστούν σε κάποιο κομμάτι κώδικα (και ιδιαιτέρως σε κάποια επαναληπτική δομή) απαιτείται η εξάλειψη πιθανών εξαρτήσεων δεδομένων, μεταξύ των διαφορετικών στιγμιοτύπων[1] του βρόχου. Οι εξαρτήσεις δεδομένων διακρίνονται στις εξής κατηγορίες [30]:

- **Ανάγνωση μετά από Εγγραφή (Read after Write – RAW):** Είναι το φαινόμενο κατά το οποίο δύο διαδοχικές διεργασίες προσπελαύνουν την ίδια θέση μνήμης, με την πρώτη διεργασία να πραγματοποιεί εγγραφή και με τη δεύτερη πραγματοποιεί ανάγνωση στα εγγεγραμμένα δεδομένα. Αν η εγγραφή δεν έχει ολοκληρωθεί επιτυχώς πριν την ανάγνωση, ή δεν υπάρχει κάποιος ειδικός μηχανισμός διόρθωσης, η δεύτερη διεργασία δεν θα λάβει την ενημερωμένη τιμή των δεδομένων. Στην χείριστη περίπτωση, η δεύτερη διεργασία μπορεί να διαβάσει ακόμα και κατακερματισμένα δεδομένα.
- **Εγγραφή μετά από Ανάγνωση (Write after Read – WAR):** Εδώ, συνεχόμενες διεργασίες πραγματοποιούν ανάγνωση και έπειτα εγγραφή στο ίδιο στοιχείο. Η εξάρτηση αυτή συνήθως δεν προξενεί κάποια σημαντική επίπτωση στον χειρισμό του κώδικα, ενώ λύνεται με μετονομασία στις μεταβλητές.
- **Εγγραφή μετά από εγγραφή (Write after Write – WAW):** Σε αυτή την περίπτωση δύο διεργασίες πραγματοποιούν εγγραφή στο ίδιο στοιχείο μνήμης, ενώ ενδιάμεσά τους παρεμβάλλεται μία (ή πολλές) διεργασία (-ες) που πραγματοποιούν ανάγνωση της πρώτης εγγραφής. Αυτό έχει ως αποτέλεσμα να υπάρχει αδυναμία στην αλλαγή της σειράς της εκτέλεσης αυτών των διεργασιών, απαγορεύοντας την παραλληλοποίησή τους. Το τελευταίο απαγορεύει την ταυτόχρονη εκτέλεση ξεχωριστών βημάτων μιας επαναληπτικής διαδικασίας, όταν υπάρχει τέτοια εξάρτηση μεταξύ τους.

[1] Με τον όρο στιγμιότυπο εννοείται η κατάσταση του βρόχου για δεδομένη τιμή του μετρητή του βρόχου.

- **Εξάρτηση ελέγχου (Control Dependency):** Στην κατάσταση αυτή η εκτέλεση κάποιας διεργασίας εξαρτάται από κάποια συνθήκη (π.χ. εντολές «if»). Συνεπώς, η μεταβολή δεδομένων εξαρτάται από το αποτέλεσμα κάποιας άλλης επεξεργασίας, η οποία δεν έχει εκ των προτέρων γνωστό αποτέλεσμα. Συνεπώς, δεν μπορεί να επιτευχθεί παραλληλοποίηση και οι εντολές πρέπει να εκτελεστούν ακολουθιακά.

Συνεπώς, ο σχεδιαστής μίας υλοποίησης πρέπει να μελετήσει τη δομή του παραχθέντος κώδικα για τυχόν εξαρτήσεις μεταξύ εντολών, κυρίως όταν αυτές αφορούν διαφορετικά βήματα επαναληπτικών δομών. Ως επακόλουθο, είναι απαραίτητη η αναμόρφωση του κώδικα, ώστε να εξαλειφθούν πιθανές τέτοιες καταστάσεις και να μπορεί το κύκλωμα να δεχθεί τους παρακάτω μετασχηματισμούς.

### 6.3.2 Ξετύλιγμα βρόχου

Κατά την κατασκευή ενός κυκλώματος μέσω διαδικαστικού κώδικα, η πιο χρησιμοποιούμενη δομή εντός των συναρτήσεων είναι αυτή των επαναληπτικών δομών for. Σε κάθε βήμα, το σύστημα ελέγχει μία συνθήκη εξόδου και αποφασίζει αν η εκτέλεση του αλγορίθμου θα παραμείνει εντός της δομής ή αν θα εξέλθει από αυτήν. Η συνθήκη ελέγχου δρα πάνω σε κάποια μεταβλητή, συνήθως ένα μετρητή, και επιτρέπει την εκτέλεση των επαναλήψεων, μόνο όταν η τιμή αυτής της μεταβλητής είναι εντός κάποιον ορίων.

Στην περίπτωση που η μεταβλητή αυτή έχει εκ των προτέρων γνωστή μεταβλητότητα (π.χ. μεταβάλλεται αυξητικά κατά ένα σε κάθε βήμα) και τα όρια της συνθήκης είναι στατικά, τότε η επαναληπτική δομή θα εκτελεστεί με a-priori πλήθος βημάτων. Υπό αυτό το πρίσμα, το σύνολο των **ν** βημάτων μπορούν να εκφραστούν αντίστοιχα από **ν** επαναλήψεις του σώματος του βρόχου. Ο μετασχηματισμός αυτός καλείται *ξετύλιγμα βρόχου* (*loop unrolling*).

Μετά την εφαρμογή του μετασχηματισμού αυτού, όλος ο βρόχος, μπορεί να εκφραστεί ισοδύναμα από μία διαδοχή εντολών, με επίτευξη της ίδιας λειτουργικότητας. Σε αντίθεση, όμως, με την επαναληπτική δομή, οι εντολές αυτές δεν απαιτούν τον έλεγχο της οριακής συνθήκης, ούτε και την ύπαρξη κάποιου μετρητή. Αυτό έχει ως αποτέλεσμα να απελευθερωθούν οι πόροι και οι ωρολογιακοί κύκλοι που αφορούσαν τον έλεγχο της συνθήκης, την αύξηση/μείωση του μετρητή καθώς και την είσοδο και έξοδο στην επαναληπτική δομή. Εκτός αυτού, τα βήματα της επαναληπτικής διαδικασίας μπορούν πλέον

να εκτελούνται ταυτόχρονα από ξεχωριστά υποσυστήματα επεξεργασίας. Έτσι, επιτυγχάνεται, σημαντική επιτάχυνση της εκτέλεσης του αλγορίθμου. Όμως, καθώς τα στιγμιότυπα εκτελούνται σε πολλαπλά «αντίγραφα» του σώματος της επαναληπτικής δομής, απαιτούνται πολλαπλάσιοι πόροι συστήματος.

Ακόμα και στην γενική περίπτωση, δηλαδή όταν ο αριθμός των επαναληπτικών βημάτων που πρέπει να εκτελεστούν δεν είναι σίγουρα καθορισμένος εξαρχής, μπορεί να πραγματοποιηθεί ξετύλιγμα του βρόχου. Ο σχεδιαστής πρέπει χειροκίνητα να επιλέξει τον αριθμό των διπλότυπων σωμάτων της επανάληψης που θα κατασκευαστούν στη θέση της αρχικής. Εδώ, δεν πραγματοποιείται κατάργηση της επαναληπτικής δομής, αλλά επιτάχυνσή της κατά έναν σημαντικό παράγοντα. Υλοποιείται **μερικό** και όχι **πλήρες ξετύλιγμα** της δομής.

Για παράδειγμα, στην Εικ. 6.7 φαίνεται ένας αλγόριθμος πολλαπλασιασμού ενός πίνακα **A** των **20 στοιχείων**, με μία σταθερά **κ**. Εφαρμόζοντας **μερικό ξετύλιγμα** τις επαναληπτικής δομής, χρησιμοποιώντας έναν συντελεστή **4**, ο προεπεξεργαστής παράγει τον βελτιστοποιημένο κώδικα της Εικ. 6.8. Παρατηρείται, ότι η συνθήκη του βρόχου θα εκτελεστεί **5 φορές** αντί για **20**, ενώ σε κάθε επαναληπτικό βήμα πραγματοποιούνται **4 εντολές πολλαπλασιασμού**. Αντίστοιχα, εφαρμόζοντας **πλήρες ξετύλιγμα** της επαναληπτικής δομής, η δομή καταργείται τελείως. Στη θέση της τοποθετείται ισοδύναμος κώδικας (βλ. Εικ. 6.9), ο οποίος όμως δεν περιέχει κάποια συνθήκη και είναι σαφώς γρηγορότερος κατά την εκτέλεση, συγκριτικά με τους κώδικες χωρίς ή με μερικό ξετύλιγμα βρόχου.

```
// Ορισμός δεδομένων
Έστω τυχαίος πίνακας A[20]
Έστω τυχαία σταθερά κ

// Πολλαπλασιασμός του πίνακα A με τη σταθερά κ
Για ν από 1 μέχρι 20 με βήμα 1:
    A[ν] = κ*A[ν]
```

Εικ. 6.7: Αλγόριθμος πριν από εφαρμογή ξετυλίγματος βρόχου.

```
// Ορισμός δεδομένων
Έστω τυχαίος πίνακας A[20]
Έστω τυχαία σταθερά κ

// Πολλαπλασιασμός του πίνακα A με τη σταθερά κ
Για ν από 1 μέχρι 20 με βήμα 4:
    A[ν]   = κ*A[ν]
    A[ν+1] = κ*A[ν+1]
    A[ν+2] = κ*A[ν+2]
    A[ν+3] = κ*A[ν+3]
```

Εικ. 6.8: Αλγόριθμος μετά από εφαρμογή μερικού ξετυλίγματος βρόχου συντελεστή 4.

```
// Ορισμός δεδομένων
Έστω τυχαίος πίνακας A[20]
Έστω τυχαία σταθερά κ

// Πολλαπλασιασμός του πίνακα A με τη σταθερά κ
A[1] = κ*A[1]
A[2] = κ*A[2]
     ...
A[20] = κ*A[20]
```

Εικ. 6.9: Αλγόριθμος μετά από εφαρμογή πλήρους ξετυλίγματος βρόχου.

### 6.3.3 Διοχέτευση βρόχου

Η τεχνική της διοχέτευσης είναι συνήθως η αποτελεσματικότερη και ευρύτερα διαδεδομένη μέθοδος αύξησης της διεκπεραιωτικής ικανότητας ενός συστήματος. Η αποτελεσματικότητα της μεθόδου αυτής είναι εμφανής τόσο σε αρχιτεκτονικές γενικού σκοπού, όπως οι κεντρικές μονάδες επεξεργασίας των κοινών υπολογιστικών συστημάτων, όσο και σε αρχιτεκτονικές ειδικού σκοπού, όπως αυτή της παρούσας διπλωματικής εργασίας. Ο μηχανισμός της διοχέτευσης, γενικά, διαιρεί μία εκτενή επεξεργαστική διεργασία σε πολλές μικρότερες, οι οποίες εκτελούνται ταυτόχρονα, για διαφορετικά δεδομένα [18]. Δοσμένων ορισμένων δεδομένων εισόδου, το σύστημα σε κάθε ωρολογιακό κύκλο επεξεργάζεται τα δεδομένα αυτά σε κάποιο διαφορετικό στάδιο.

Για να μελετηθεί καλύτερα αυτός ο μηχανισμός, είναι πιο εύχρηστο να παρουσιαστεί με το εξής παράδειγμα: Έστω μία αλγοριθμική διαδικασία που πραγματοποιεί ανάγνωση μίας **εισόδου**, εκτέλεση μιας αριθμητικής/λογικής **επεξεργασίας** και εγγραφή μίας **εξόδου**, όπου κάθε βήμα απαιτεί έναν κύκλο ρολογιού για να ολοκληρωθεί. Πριν από την εφαρμογή της διοχέτευσης, τα **3** στάδια εκτελούνται διαδοχικά και συνεχόμενα για κάποια δοσμένα

δεδομένα εισόδου. Παρατηρείται, ότι η έξοδος θα εμφανίζει το σωστό αποτέλεσμα για κάποια είσοδο μετά από καθυστέρηση ίση με **3** κύκλους ρολογιού, ενώ αυτό είναι και το *βήμα* (*interval*) με το οποίο η είσοδος μπορεί να δέχεται νέα δεδομένα. Για τα **3** δεδομένα εισόδου, απαιτούνται συνολικά **9** κύκλοι ρολογιού, όπως φαίνεται και στην Εικ. 6.10.

| | | Κύκλος ρολογιού | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Δεδομένα | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| | A | είσοδος | επεξεργασία | έξοδος | | | | | | |
| | B | | | | είσοδος | επεξεργασία | έξοδος | | | |
| | Γ | | | | | | | είσοδος | επεξεργασία | έξοδος |

Εικ. 6.10: Παράδειγμα διαδικασίας χωρίς βελτιστοποίηση διοχέτευσης.

Έπειτα, εφαρμόστηκε ο μηχανισμός της διοχέτευσης στην ανωτέρω διαδικασία. Θεωρείται ότι τα **3** στάδια μπορούν να ανεξαρτητοποιηθούν μεταξύ τους, ώστε το καθένα να επεξεργάζεται διαφορετικά δεδομένα στον ίδιο κύκλο ρολογιού. Τώρα, για **3** διαφορετικές συστάδες δεδομένων **A**, **B**, **Γ** προκύπτει ότι το σύστημα πραγματοποιεί μέρος της συνολικής διαδικασίας ταυτόχρονα (Εικ. 6.11). Ως αποτέλεσμα, παρότι η καθυστέρηση για τα δεδομένα παραμένει στους **3** κύκλους ρολογιού, το σύστημα παράγει **1** αποτέλεσμα σε κάθε κύκλο. Έχει, δηλαδή, **τριπλάσια** διεκπεραιωτική ικανότητα από το αρχικό.

| | | Κύκλος ρολογιού | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Δεδομένα | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| | A | είσοδος | επεξεργασία | έξοδος | | | | | | |
| | B | | είσοδος | επεξεργασία | έξοδος | | | | | |
| | Γ | | | είσοδος | επεξεργασία | έξοδος | | | | |

Εικ. 6.11: Παράδειγμα διαδικασίας με βελτιστοποίηση διοχέτευσης.

### 6.3.4 Βελτιστοποίηση ροής δεδομένων

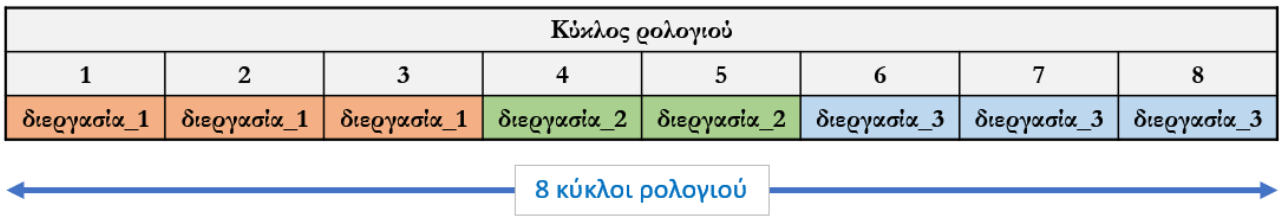
Ο μετασχηματισμός ροής δεδομένων είναι ευθέως σχετιζόμενος με την μέθοδο της διοχέτευσης που αναφέρθηκε στην Υποπαρ. 6.3.3. Ο όρος εδώ αναφέρεται στην διοχέτευση σε υψηλό επίπεδο, με τις διεργασίες να μην αφορούν εντολές εντός κάποιας επαναληπτικής δομής. Αφορά το σύνολο των λειτουργιών του σώματος της κύριας συνάρτησης ενός (υπό-) συστήματος. Πραγματοποιείται, δηλαδή, επικάλυψη στην εκτέλεση συναρτήσεων και επαναληπτικών βρόχων, αυξάνοντας τη συνολική διεκπεραιωτική ικανότητα της σχεδίασης.

Όταν χρησιμοποιείται η τεχνική αυτή, ο προεπεξεργαστής αναλύει τη ροή δεδομένων μεταξύ ακολουθιακών συναρτήσεων ή βρόχων και δημιουργεί διαύλους (βασισμένους σε ping-pong RAMs ή FIFOs), οι οποίοι επιτρέπουν τις συναρτήσεις ή βρόχους «καταναλωτές» να ξεκινήσουν την εκτέλεσή τους πριν οι αντίστοιχοι «παραγωγοί» έχουν τερματίσει. Έτσι, οι δομές που επιδρά η τεχνική αυτή δύναται να δουλεύουν παράλληλα, μειώνοντας την καθυστέρηση και αυξάνοντας την διεκπεραιωτική ικανότητα του παραγομένου κυκλώματος.

Για παράδειγμα, θεωρείται η μέθοδος **κύρια_μέθοδος()**, η οποία εμπεριέχει την εκτέλεση τριών διεργασιών (Εικ. 6.12). Πριν την εφαρμογή κάποιας μεθόδου βελτιστοποίησης, η εκτέλεση των εσωτερικών διεργασιών πραγματοποιείται ακολουθιακά και χωρίς καμία επικάλυψη (Εικ. 6.13). Αυτό συμβαίνει, διότι οι διεργασίες αυτές χρησιμοποιούν (καταναλώνουν) δεδομένα που εξάγουν (παράγουν) οι προηγούμενες. Με την εφαρμογή της βελτιστοποίησης ροής δεδομένων, το παραγόμενο κύκλωμα προσαρμόζεται έτσι ώστε τα απαραίτητα δεδομένα να παράγονται όσο το δυνατόν νωρίτερα, προκειμένου να μπορούν να καταναλωθούν σε πιο άμεσο χρόνο από τις διεργασίες-καταναλωτές. Συνεπώς, προκύπτει σημαντική μείωση στο χρόνο για την πλήρη επεξεργασία των δεδομένων, φτάνοντας τους **5** κύκλους ανά επεξεργασία από τους **8** αρχικούς (Εικ. 6.14).

```
κύρια_μέθοδος(α, β, γ, δ) {
    διεργασία_1(α, β, ε1)
    διεργασία_2(γ, ε1, ε2)
    διεργασία_3(ε2, δ)
    επίστρεψε δ
}
```

Εικ. 6.12: Παράδειγμα εκτέλεσης τριών διεργασιών σε ψευδογλώσσα.




Εικ. 6.13: Εκτέλεση των τριών διεργασιών πριν την εφαρμογή της βελτιστοποίησης ροής δεδομένων.

| Κύκλος ρολογιού | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| διεργασία_1 | διεργασία_1 | διεργασία_1 | διεργασία_1 | διεργασία_1 | διεργασία_1 | | |
| | διεργασία_2 | διεργασία_2 | | διεργασία_2 | διεργασία_2 | | |
| | | διεργασία_3 | διεργασία_3 | διεργασία_3 | διεργασία_3 | διεργασία_3 | διεργασία_3 |


Εικ. 6.14: Εκτέλεση των τριών διεργασιών μετά την εφαρμογή της βελτιστοποίησης ροής δεδομένων.

### 6.3.5 Ισοπέδωση βρόχου

Ο μετασχηματισμός *ισοπέδωσης βρόχου* (*loop flattening*) επιδρά πάνω σε κώδικα με εμφωλευμένες επαναληπτικές δομές και προκαλεί την κατάρρευσή τους σε μία μόνο επαναληπτική δομή, εφόσον αυτό είναι δυνατό. Δοθέντος ενός αλγορίθμου τριών εμφωλευμένων βρόχων (Εικ. 6.15), μπορεί να πραγματοποιηθεί συγχώνευσή τους, με σύμπτυξη των λογικών συνθηκών και διατήρηση της λειτουργικότητας (Εικ. 6.16).

```
Για α από 1 μέχρι Α:
    Για β από 1 μέχρι Β:
        Για γ από 1 μέχρι Γ:
            διαδικασία_1
            ...
            διαδικασία_ν
```

Εικ. 6.15: Κώδικας πριν την ισοπέδωση των βρόχων.

```
Για κ από 1 μέχρι (Α*Β*Γ):
    διαδικασία_1
    ...
    διαδικασία_ν
```

Εικ. 6.16: Κώδικας μετά την ισοπέδωση των βρόχων.

Η συνολική διάρκεια εκτέλεσης ($\mathbf{d_t}$) ενός επαναληπτικού βρόχου υπολογίζεται από την Εξ. 6.3, με $\mathbf{d_{in}}$ και $\mathbf{d_{out}}$ να συμβολίζουν την καθυστέρηση εισόδου και εξόδου αντίστοιχα στον βρόχο[1], $\mathbf{n_s}$ τα συνολικά βήματα του και $\mathbf{d_s}$[2] την καθυστέρηση του καθενός από αυτά.

$$d_t = d_{in} + n_s \cdot d_s + n_s \cdot d_{out} \qquad \text{Εξ. 6.3}$$

Όταν ένας αλγόριθμος περιέχει ένα σύνολο εμφωλευμένων βρόχων, εμφανίζεται αυξημένη καθυστέρηση, καθώς οι καθυστερήσεις $\mathbf{d_{in}}$ και $\mathbf{d_{out}}$ από τους διαδοχικούς βρόχους συνδράμουν γεωμετρικά κατά την πρόσθεση επιπέδων. Αυτή η αύξηση προκύπτει από τον

[1] Αυτές οι καθυστερήσεις αφορούν μεταβάσεις της εκτέλεσης σε διαφορετικά σημεία κώδικα, πράγμα που ισοδυναμεί με άλματα.

[2] Στην καθυστέρηση κάθε επαναληπτικού βήματος $d_s$ συνυπολογίζεται και η καθυστέρηση για τον έλεγχο της συνθήκης.

πολλαπλασιασμό τον βημάτων κάθε επαναληπτικής δομής ($\mathbf{n_t}$) με την καθυστέρηση του εκάστοτε βήματος ($\mathbf{d_s}$).

Η γεωμετρική αύξηση φαίνεται ξεκάθαρα αν θεωρηθεί μία δεύτερη επαναληπτική δομή (Εξ. 6.4), εμφωλευμένη της αρχικής (Εξ. 6.3). Υποθέτοντας ότι ο νέος βρόχος αποτελεί το σώμα του αρχικού, προκύπτει η Εξ. 6.5 και, συνδυάζοντας τις τρεις αυτές εξισώσεις, λαμβάνεται η Εξ. 6.6 για την συνολική καθυστέρηση των δύο βρόχων. Εδώ, διακρίνεται η αυξητική τάση της καθυστέρησης, συναρτήσει του αριθμού των σταδίων, με έναν παράγοντα $\mathbf{n_s}(\mathbf{d_{in2+}d_{out2}})$, ο οποίος εμφανίζεται μόνο στην περίπτωση που υπάρχει εμφωλευμένη επαναληπτική δομή και, άρα, μεταβάσεις εισόδου και εξόδου. Με την τεχνική της ισοπέδωσης βρόχου, προκαλείται ο μηδενισμός του παρεισακτου όρου (εδώ: $\mathbf{n_s}(\mathbf{d_{in2+}d_{out2}})$). Συνεπώς, επιτυγχάνεται μείωση της συνολικής καθυστέρησης (εδώ: $\mathbf{d_t}$).

$$d_{t2} = d_{in2} + n_{s2} \cdot d_{s2} + d_{out2} \qquad \text{Εξ. 6.4}$$

$$d_{t2} = d_s \qquad \text{Εξ. 6.5}$$

$$d_t = d_{in} + n_s \cdot n_{s2} \cdot d_{s2} + n_s \cdot d_{out} + n_s \cdot (d_{in2} + d_{out2}) \qquad \text{Εξ. 6.6}$$

Για να μπορέσει να εφαρμοστεί η τεχνική της ισοπέδωσης σε ένα σύνολο επαναληπτικών δομών, πρέπει το σύνολο αυτό να είναι τέλειο ή ημι-τέλειο. Συνεπώς, είναι απαραίτητο να ικανοποιούνται οι εξής κανόνες:

**Κανόνας 1:** Μόνο η εσωτερικότερη επαναληπτική δομή περιέχει προς εκτέλεση εντολές.

**Κανόνας 2:** Δεν παρεμβάλλεται κάποια λογική δομή ενδιάμεσα στα διαφορετικά επίπεδα επαναληπτικών δομών.

**Κανόνας 3:** Όλες οι επαναληπτικές δομές πρέπει να έχουν σταθερά όρια, εκτός ίσως από την εξωτερικότερη δομή, η οποία είναι η μόνη που επιτρέπεται να έχει μεταβλητά όρια.

Η παράβαση έστω και ενός από τους παραπάνω κανόνες, επιφέρει αδυναμία ισοπέδωσης του επαναληπτικού συνόλου.

Ως αποτέλεσμα της τεχνικής της ισοπέδωσης, εξοικονομείται σημαντικός αριθμός κύκλων ρολογιού, καθώς δεν εξαλείφονται οι μεταβάσεις μεταξύ εμφωλευμένων βρόχων διαφορετικών επιπέδων, ενώ υπάρχει μία, καθολική, συνθήκη τερματισμού των επαναλήψεων.

### 6.3.6 Κατάτμηση πίνακα

Κατά την παραπάνω ανάλυση (και ιδιαιτέρως με τη μέθοδο της διοχέτευσης), δίνεται η δυνατότητα στον σχεδιαστή να επιταχύνει σημαντικά την εκτέλεση του αλγοριθμικού στοιχείου εντός μίας υλοποίησης. Συνεπώς, διακρίνεται η ανάγκη για ανάλογη αύξηση της ταχύτητας με την οποία προσπελαύνονται τα χρησιμοποιηθέντα δεδομένα, τα οποία αποτελούν τις εισόδους και εξόδους των αλγοριθμικών στοιχείων.

Ο συντριπτικός όγκος των δεδομένων αυτών βρίσκεται εντός πινάκων. Ως γνωστόν, για την εξοικονόμηση πόρων και χώρου πάνω στο FPGA, το εργαλείο HLS υλοποιεί τα στοιχεία αποθήκευσης με τον πλέον διαθέσιμο οικονομικό τρόπο – τις μνήμες RAM [31]. Οι μνήμες αυτές είναι το ιδανικότερο στοιχείο αποθήκευσης όσον αφορά την αποτελεσματικότερη αξιοποίηση του χώρου. Όμως, όπως έγινε εμφανές και στην Υποπαρ. 5.4.2, έχουν περιορισμένη ταχύτητα και πρόσβαση μόνο σε 2 το πολύ δεδομένα ταυτόχρονα.

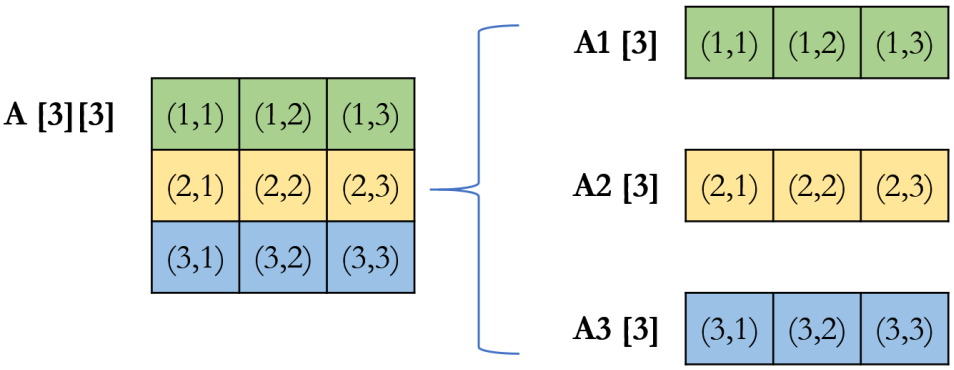
Η *κατάτμηση πίνακα* (*array partition*) είναι μία μέθοδος που επιτρέπει την επιτάχυνση της πρόσβασης στη μνήμη, δεσμεύοντας επιπλέον μονάδες μνήμης. Συγκεκριμένα, αντί οι πίνακες να αποθηκεύονται σε μία ενιαία μονάδα μνήμης, το εργαλείο αναλαμβάνει να αποθηκεύσει διαφορετικές ομάδες στοιχείων του πίνακα σε διαφορετικές δομές μνήμης (Εικ. 6.17). Έτσι, αυξάνονται πρακτικά οι διαθέσιμοι δίαυλοι ανάγνωσης/εγγραφής σε πολλαπλάσιο αριθμό του 2, που ήταν αρχικά, ανάλογα με τον συντελεστή κατάτμησης[1] που επιλέγεται από τον σχεδιαστή.

Αξίζει να αναφερθεί, ότι η μέθοδος της κατάτμησης χωρίζεται σε τρεις κατηγορίες, ανάλογα με διαδικασία που πραγματοποιείται ο καταμερισμός των θέσεων του πίνακα. Οι κατηγορίες αυτές είναι οι εξής:

1. **Συνεχής (block)**: Η μήτρα διαμερίζεται σε ισομερή μπλοκ από συνεχόμενα στοιχεία της αρχικής μήτρας.
2. **Κυκλική (cyclic)**: Η μήτρα διαμερίζεται σε ισομερή μπλοκ από κυκλικώς εναλλάσσοντα στοιχεία της αρχικής μήτρας.
3. **Ολική (complete)**: Η μήτρα διαμερίζεται εξ ολοκλήρου στα στοιχεία που την αποτελούν. Ως αποτέλεσμα, κάθε στοιχείο αποθηκεύεται σε έναν ξεχωριστό

[1] Δηλαδή σε πόσους πίνακες χωρίζεται ο αρχικός.

καταχωρητή. Έτσι, επιτρέπεται ταυτόχρονη προσπέλαση όλων των στοιχείων, με δέσμευση, όμως, αντιστοίχου πλήθους καταχωρητών.




Εικ. 6.17: Κατάτμηση της δεύτερης διάστασης ενός πίνακα A σε τρεις πίνακες A1, A2 και A3.

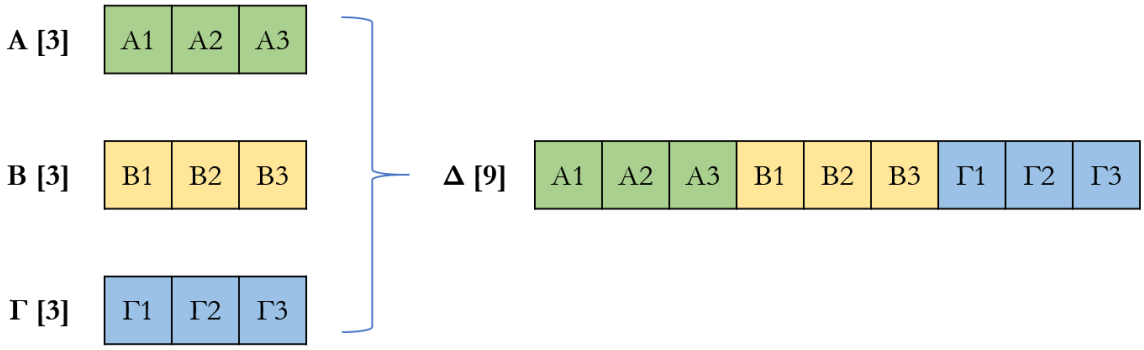
### 6.3.7 Χαρτογράφηση πίνακα

Σε πολλές υλοποιήσεις κώδικα εμφανίζονται πίνακες σχετικά μικρού μεγέθους, αρκετά κάτω από το τυποποιημένο της παρεχόμενης μνήμης RAM των FPGAs (18 kbits ή 36 kbits). Αυτό, στην προεπιλεγμένη σύνθεση, οδηγεί σε σπατάλη ολόκληρων μπλοκ RAM για πίνακες που περιέχουν υποπολλαπλάσια ποσά δεδομένων. Αυτό το γεγονός έρχεται να καταπολεμήσει ο μετασχηματισμός *χαρτογράφησης πίνακα* (*array map*).

Με τη βελτιστοποίηση αυτή το συνθέσιμο κύκλωμα προσαρμόζεται έτσι ώστε πολλαπλές μικρές μήτρες να συνθέσουν ένα ενιαίο σύνολο. Αυτό το σύνολο μπορεί να συμπεριληφθεί εντός μίας μνήμης RAM (εφόσον δεν ξεπερνά την χωρητικότητά της), οδηγώντας σε καλύτερη εκμετάλλευση των χρησιμοποιούμενων πόρων.

Οι τρόποι με τους οποίους πραγματοποιείται ο μηχανισμός αυτός είναι οι εξής:

- **Οριζόντια χαρτογράφηση (horizontal mapping):** Στη φυσική σχεδίαση δημιουργείται ένας ενιαίος πίνακας με περισσότερα στοιχεία. Ένα παράδειγμα παρουσιάζεται στην Εικ. 6.18.
- **Κατακόρυφη χαρτογράφηση (vertical mapping):** Στη φυσική σχεδίαση δημιουργείται ένας ενιαίος πίνακας με μεγαλύτερο μήκος λέξεων.




Εικ. 6.18: Συγχώνευση τριών πινάκων (Α, Β, Γ) σε έναν ενιαίο (Δ).

### 6.3.8 Ανασχηματισμός πίνακα

Η τεχνική του *ανασχηματισμού πίνακα* (*array reshape*) συνδυάζει την κατάτμηση πίνακα (βλ. Υποπαρ. 6.3.6) και την κατακόρυφη χαρτογράφηση της τεχνικής χαρτογράφησης πίνακα (βλ. Υποπαρ. 6.3.7), συγχωνεύοντας στοιχεία πινάκων αυξάνοντας τα μήκη λέξεων. Έτσι, μειώνονται τα μπλοκ RAM που χρησιμοποιούνται, διατηρώντας όμως την παραλληλία στην προσπέλαση των δεδομένων. Ο μετασχηματισμός αυτός δημιουργεί μία νέα μήτρα με λιγότερα στοιχεία, αλλά με μεγαλύτερα μήκη λέξεων, αυξάνοντας το ποσό των δεδομένων που προσπελαύνονται ανά κύκλο ρολογιού.

## 6.4 Εφαρμογή μετασχηματισμών στο Vivado HLS

Το εργαλείο Vivado HLS προσφέρει τη δυνατότητα δημιουργίας διαφόρων αρχιτεκτονικών ενός σχεδιασμού, τις οποίες αποκαλεί *λύσεις* (*solutions*). Η λειτουργία αυτή επιτρέπει τον πειραματισμό με διαφορετικούς μετασχηματισμούς κώδικα, καθώς και τη σύγκριση των παραγόμενων λύσεων. Οι μετασχηματισμοί που αναφέρθηκαν στην Παρ. 6.3 μπορούν να εφαρμοστούν στο Vivado HLS με τη χρήση οδηγιών προς τον μεταγλωττιστή. Η εισαγωγή τους στο εργαλείο γίνεται με τους δύο ακόλουθους τρόπους:

1. Εισαγωγή **pragmas** στον πηγαίο κώδικα: Με τη χρήση pragmas οι οδηγίες αποτελούν πλέον μέρος του κώδικα. Για τον λόγο αυτό, πρόκειται για ιδανική επιλογή στην περίπτωση που ο χρήστης επιθυμεί οι οδηγίες που δίνει να έχουν καθολική ισχύ στις επιμέρους αρχιτεκτονικές.

2. Εισαγωγή **directives** σε ξεχωριστό αρχείο: Στην περίπτωση αυτή, οι οδηγίες του χρήστη αποθηκεύονται σε αρχείο διαφορετικό από αυτό του πηγαίου κώδικα. Η επιλογή αυτή προτείνεται στην περίπτωση που απαιτείται οι οδηγίες μεταγλωττιστή να έχουν ισχύ μόνο για μια συγκεκριμένη αρχιτεκτονική, καθώς το αρχείο αποθήκευσης είναι μοναδικό για κάθε λύση.

```
#pragma HLS PIPELINE
```

(α)

```
set_directive_pipeline "dense_layer1/dense_layer1_inner_loop"
```

(β)

Εικ. 6.19: (α) εισαγωγή pragma στον πηγαίο κώδικα και (β) εισαγωγή directive σε ξεχωριστό αρχείο.

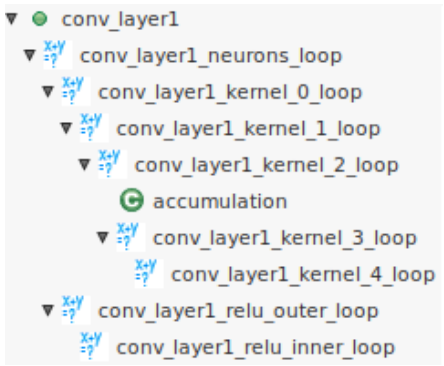
Τέλος, επισημαίνεται ότι το σημαντικότερο πλεονέκτημα των οδηγιών προς το μεταγλωττιστή είναι πως εφαρμόζονται μετασχηματισμοί κώδικα με ελάχιστες (**pragmas**) ή και καθόλου (**directives**) αλλαγές στον πηγαίο κώδικα.

### 6.4.1 Η δομή του σχεδιασμού πριν τις βελτιστοποιήσεις

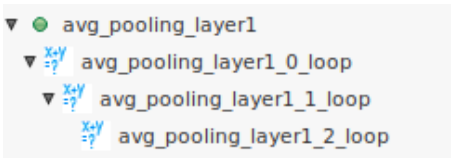
Προτού γίνει αναφορά στους μετασχηματισμούς κώδικα όπως εφαρμόστηκαν στο LeNet-5, απαιτείται η εξοικείωση με την οργάνωση του αρχικού σχεδιασμού. Η τελευταία παρουσιάζεται στην Εικ. 6.20 ανά επίπεδο, βάσει όσων αναφέρονται στην Παρ. 4.3. Αναλυτικότερα:

1. **Συνελικτικά επίπεδα:** Στα συνελικτικά επίπεδα πραγματοποιείται, αρχικά, η προσπέλαση των νευρώνων τους (**neurons_loop**) και έπειτα των ενδιάμεσων χαρτών γνωρισμάτων του δικτύου. Η προσπέλαση των χαρτών γνωρισμάτων απαιτεί 3 βρόχους: έναν που αφορά τον αριθμό τους και άλλους δύο που αντιπροσωπεύουν το ύψος και το πλάτος της εκάστοτε εικόνας (**kernel_0_loop** έως **kernel_2_loop**). Στη συνέχεια, εφαρμόζεται ο πυρήνας για την αντίστοιχη είσοδο, επομένως χρειάζονται 2 επιπλέον εμφωλευμένοι βρόχοι (**kernel_3_loop** και **kernel_4_loop**). Τέλος, στους δύο τελευταίους βρόχους (ένας για κάθε διάσταση εικόνας) πραγματοποιείται η εφαρμογή της συνάρτησης ενεργοποίησης ReLU στην έξοδο του εκάστοτε νευρώνα.

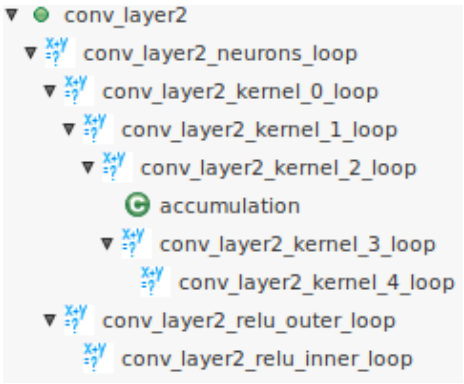
2. **Επίπεδα μέσης υποδειγματοληψίας:** Επειδή τα επίπεδα αυτά διαδέχονται τα συνελικτικά, είναι επίσης απαραίτητη η προσπέλαση των ενδιάμεσων εικόνων. Έτσι, λοιπόν, και σε αυτή την περίπτωση χρησιμοποιούνται 3 εμφωλευμένοι βρόχοι.
3. **Επίπεδο ισοπέδωσης:** Το επίπεδο αυτό μετατρέπει τον τρισδιάστατο πίνακα που συγκρατεί τους ενδιάμεσους χάρτες γνωρισμάτων σε μονοδιάστατο. Έτσι, όπως και τα επίπεδα μέσης υποδειγματοληψίας απαιτεί τρεις εμφωλευμένους βρόχους.
4. **Πλήρως διασυνδεδεμένα επίπεδα:** Κάθε νευρώνας χαρακτηρίζεται από ένα βάρος για κάθε είσοδό του. Συνεπώς, στα επίπεδα αυτά χρειάζονται δύο εμφωλευμένοι βρόχοι: ένας που να προσπελαύνει τους νευρώνες και ένας επιπλέον ο οποίος να προσπελαύνει τα βάρη και τις εισόδους του εκάστοτε νευρώνα. Για την περίπτωση του τελευταίου επιπέδου, ωστόσο υπάρχει ένας ακόμα βρόχος για τον υπολογισμό της συνάρτησης ενεργοποίησης Softmax, διότι δεν είναι δυνατή η πλήρης ενσωμάτωσή της στους προηγούμενους βρόχους (βλ. Υποπαρ. 6.2.2).



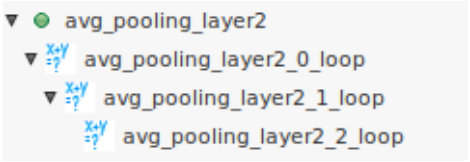

(α)



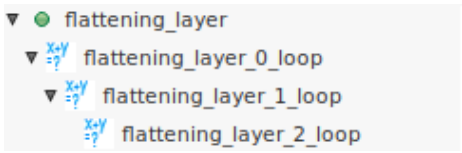

(β)



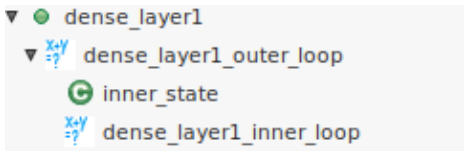

(γ)



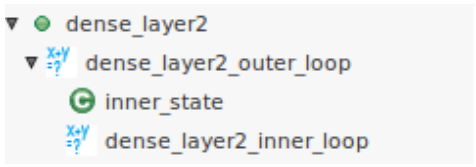

(δ)



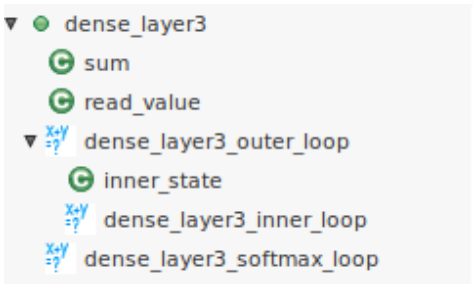

(ε)




(στ)




(ζ)




(η)

Εικ. 6.20: (α-η) Η δομή του LeNet-5 σε επίπεδο κώδικα.

Μετά την ολοκλήρωση των βελτιστοποιήσεων σε επίπεδο κώδικα, (βλ. Παρ. 6.2), ακολούθησαν αυτές που μπορούν να πραγματοποιηθούν μέσω οδηγιών μεταγλωττιστή, προκειμένου ο πηγαίος κώδικας να παραμείνει αμετάβλητος. Οι οδηγίες που επιλέχθηκαν, αλλά και η σειρά εφαρμογής τους βασίστηκαν στον επίσημο οδηγό του εργαλείου Vivado HLS [19]. Τέλος, σημειώνεται πως η πορεία των μετασχηματισμών που αναλύεται παρακάτω είναι προσθετική, δηλαδή κάθε αρχιτεκτονική κληρονομεί τους μετασχηματισμούς της προηγούμενης.

### 6.4.2 Εφαρμογή μετασχηματισμού διοχέτευσης

Βασικό πλεονέκτημα της τεχνικής της διοχέτευσης είναι η επίτευξη υψηλής διεκπεραιωτικής ικανότητας με μικρή, ωστόσο, επιβάρυνση σε υλικό [30]. Το γεγονός αυτό οφείλεται στην απαίτηση κυρίως ενδιάμεσων καταχωρητών ανάμεσα στα στάδια της διοχέτευσης. Αποτελεί, λοιπόν, έναν καλό συμβιβασμό μεταξύ ταχύτητας και χρήσης πόρων.

Ο επίσημος οδηγός του εργαλείου [19] συνιστά την εισαγωγή οδηγιών διοχέτευσης στον εσωτερικότερο βρόχο. Η πρόταση αυτή πηγάζει από το γεγονός ότι η διοχέτευση ενός βρόχου απαιτεί το ξετύλιγμα όλων των εσωτερικών βρόχων, με αποτέλεσμα τη δέσμευση

επιπλέον υλικού. Ωστόσο, η προκαθορισμένη συμπεριφορά του Vivado HLS είναι η εφαρμογή και του μετασχηματισμού ισοπέδωσης βρόχου, εφόσον η διαδοχή των βρόχων είναι τέλεια (βλ. Παρ. 6.3.5). Συγκεκριμένα, το εργαλείο εξετάζει αν είναι δυνατό να ενώσει το βρόχο στον οποίο εφαρμόζεται η οδηγία διοχέτευσης με τους ιεραρχικά ανώτερους.

Αρχικά, η οδηγία διοχέτευσης βρόχου εφαρμόστηκε στους εσωτερικότερους βρόχους των επιπέδων μέσης υποδειγματοληψίας, ισοπέδωσης και πλήρως διασυνδεδεμένων επιπέδων. Στα τελευταία, όμως, επειδή η διαδοχή των βρόχων **outer** και **inner** (Εικ. 6.20 (στ), (η) και (ζ)) δεν είναι τέλεια, το εργαλείο αδυνατεί να εφαρμόσει και τον μετασχηματισμό ισοπέδωσης βρόχου. Για τον λόγο αυτό δοκιμάστηκε η εισαγωγή οδηγίας διοχέτευσης στον εξωτερικό βρόχο των πλήρως διασυνδεδεμένων επιπέδων. Η σύνθεση, ωστόσο, έδειξε ότι στην περίπτωση αυτή χρησιμοποιείται το 147% των LUTs βγαίνοντας εκτός ορίων του FPGA.

Παρόλα αυτά, εξαιτίας των εξαρτήσεων μεταξύ των δεδομένων, δεν είναι η δυνατή η διοχέτευση βήματος 1 στα επίπεδα μέσης υποδειγματοληψίας. Το πρόβλημα αυτό λύνεται με τη χρήση οδηγιών ανασχηματισμού και κατάτμησης των πινάκων εισόδου, διότι δίνουν τη δυνατότητα παράλληλης πρόσβασης στα στοιχεία τους.

Με βάση τα παραπάνω, εκτελώντας σύνθεση λαμβάνονται τα αποτελέσματα που παρουσιάζονται στον Πίν. 6.1. Συγκρίνοντας τα αποτελέσματα με τα αρχικά (βλ. Πίν. 5.5 και Πίν. 5.6) παρατηρείται 14% μείωση της καθυστέρησης με μόλις 2% αύξηση στα δεσμευόμενα LUTs.

Πίν. 6.1: Στατιστικά αρχιτεκτονικής διοχέτευσης όλων των επιπέδων πλην των συνελικτικών (αρχιτεκτονική **Διοχέτευσης 1**).

| **Καθυστέρηση (ms)** | **Χρήση πόρων (%)** | | | |
|---|---|---|---|---|
| | BRAM-18K | DSP48E | FF | LUT |
| 9.1 | 7 | 0 | ≈0 | 7 |

Στη συνέχεια, προστέθηκαν οδηγίες διοχέτευσης και στα συνελικτικά επίπεδα. Αναλυτικότερα, επιλέχθηκε η επαναληπτική δομή **kernel_2** (Εικ. 6.20 (α) και (γ)), αφενός επειδή αποτελεί τον εσωτερικότερο βρόχο που σχηματίζει τέλεια διαδοχή μέχρι και τον **kernel_0** και αφετέρου εξαιτίας του μικρού αριθμού προσπελάσεων (3 το καθένα, σύνολο 9) των ιεραρχικά κατώτερων βρόχων **kernel_3** και **kernel_4**. Οι δύο αυτοί λόγοι έχουν σαν

επακόλουθο τη μειωμένη καθυστέρηση εισόδου/εξόδου στις επαναληπτικές δομές και τη χαμηλή επιβάρυνση σε υλικό, αντίστοιχα. Ωστόσο, και σε αυτή την περίπτωση λόγω των εξαρτήσεων δεδομένων δεν είναι δυνατή η διοχέτευση βήματος 1, αλλά το πρόβλημα αυτό αντιμετωπίστηκε όπως και προηγουμένως με τον μετασχηματισμό των πινάκων εισόδου. Τέλος, προστέθηκε οδηγία διοχέτευσης και στο βρόχο **relu_inner**, στον οποίο επιτεύχθηκε αμέσως βήμα διοχέτευσης ίσο με 1.

Συνθέτοντας το νευρωνικό δίκτυο και λαμβάνοντας υπόψη τα παραπάνω προκύπτουν οι μετρήσεις που αναγράφονται στον Πίν. 6.2. Είναι φανερό πως δεσμεύοντας 5% περισσότερα LUTs και 1% επιπλέον FFs επιτυγχάνεται 85% μείωση της καθυστέρησης του κυκλώματος, πράγμα που ισοδυναμεί με περίπου 7 φορές επιτάχυνση σε σχέση με τον αρχικό σχεδιασμό.

Πίν. 6.2: Στατιστικά πλήρους αρχιτεκτονικής διοχέτευσης (αρχιτεκτονική **Διοχέτευσης 2**).

| **Καθυστέρηση (ms)** | **Χρήση πόρων (%)** | | | |
|---|---|---|---|---|
| | BRAM-18K | DSP48E | FF | LUT |
| 1.57 | 7 | 0 | 1 | 10 |

### 6.4.3 Εφαρμογή μετασχηματισμού ξετυλίγματος

Περεταίρω επιτάχυνση μπορεί να επιτευχθεί με την εφαρμογή του μετασχηματισμού ξετυλίγματος. Ο τελευταίος, όπως προαναφέρθηκε (βλ. Υποπαρ. 6.3.2), παραλληλοποιεί τους υπολογισμούς δεσμεύοντας επιπλέον υλικό. Επειδή η διαδικασία αυτή μπορεί να έχει ως αποτέλεσμα τη δέσμευση αρκετά περισσότερων πόρων απαιτείται ιδιαίτερη προσοχή στην επιλογή του σημείου εφαρμογής της. Στόχος, λοιπόν, είναι ένας καλός συμβιβασμός μεταξύ επιτάχυνσης και χρήσης πόρων.

Με βάση τα παραπάνω είναι εμφανές πως είναι απαραίτητη η εύρεση των *περιοριστικών σημείων* (*bottlenecks*) του σχεδιασμού. Για τον λόγο αυτό, χρησιμοποιήθηκε η λειτουργία ανάλυσης που διαθέτει το Vivado HLS. Η τελευταία επιτρέπει τη διεξοδική επιθεώρηση όλων των διαδικασιών και υπολογισμών που λαμβάνουν χώρα κατά τη διάρκεια της λειτουργίας του νευρωνικού δικτύου. Συγκεκριμένα, το εργαλείο παρουσιάζει λεπτομερώς τον χρονισμό των επιμέρους διαδικασιών, καθώς και τη χρήση πόρων σε κάθε κύκλο ρολογιού.

Παράλληλα, το Vivado HLS αναφέρει τα στατιστικά σύνθεσης και ανά συνάρτηση, προκειμένου ο χρήστης να έχει μια αναλυτικότερη εικόνα των λειτουργιών που εκτελούνται στα κατώτερα επίπεδα. Επειδή στην προκειμένη περίπτωση κάθε συνάρτηση αφορά τη λειτουργία ενός επιπέδου του LeNet-5, προέκυψαν τα αποτελέσματα που παρουσιάζονται στον Πίν. 6.3. Το γεγονός ότι το 2ο συνελικτικό επίπεδο σε συνδυασμό με το 1ο πλήρες διασυνδεδεμένο επίπεδο συνιστούν το 79% της συνολικής καθυστέρησης, καθιστά τα επίπεδα αυτά τους καλύτερους υποψηφίους για την εφαρμογή του μετασχηματισμού ξετυλίγματος.

Πίν. 6.3: Συνεισφορά των καθυστερήσεων των επιμέρους επιπέδων.

| **Επίπεδο** | **Καθυστέρηση (ms)** | **Καθυστέρηση (%)** |
|---|---|---|
| Συνελικτικό 1 | 0.19 | 12.1 |
| Μέσης υποδειγματοληψίας 1 | 0.01 | 0.6 |
| Συνελικτικό 2 | 0.54 | 34.4 |
| Μέσης υποδειγματοληψίας 2 | 0.01 | 0.6 |
| Πλήρως διασυνδεδεμένο 1 | 0.70 | 44.6 |
| Πλήρως διασυνδεδεμένο 2 | 0.10 | 6.4 |
| Πλήρως διασυνδεδεμένο 3 | 0.01 | 0.6 |
| **Σύνολο** | 1.57 | 100.0 |

Αρχικά, εξετάστηκε το 1ο πλήρως διασυνδεδεμένο επίπεδο, καθώς χαρακτηρίζεται από τη μεγαλύτερη καθυστέρηση. Για τον λόγο αυτό, δοκιμάστηκε η εφαρμογή της οδηγίας ξετυλίγματος του βρόχου **inner_loop** (Εικ. 6.20 (στ)) αντί για διοχέτευση. Έπειτα από δοκιμές διαπιστώθηκε πως η χρήση **μερικού** ξετυλίγματος με έναν συντελεστή ίσο με **36**, αποτελεί έναν καλό συμβιβασμό ταχύτητας και δέσμευσης πόρων. Δεδομένου ότι ο βρόχος αποτελείται από **576** επαναλήψεις, πλέον θα εκτελούνται μόνο **576 / 36 = 16**. Εκτελώντας σύνθεση, το Vivado HLS αναφέρει τα στατιστικά που φαίνονται στον Πίν. 6.4. Συγκρίνοντας την αρχιτεκτονική αυτή με την προηγούμενη (βλ. Πίν. 6.2), παρατηρείται πως με 7% αύξηση στα LUTs, επιτυγχάνεται περίπου 20% επιτάχυνση.

Πίν. 6.4: Στατιστικά αρχιτεκτονικής ξετυλίγματος βρόχου του 1ου πλήρως διασυνδεδεμένου επιπέδου (αρχιτεκτονική **Ξετυλίγματος 1**).

| **Καθυστέρηση (ms)** | **Χρήση πόρων (%)** | | | |
|---|---|---|---|---|
| | BRAM-18K | DSP48E | FF | LUT |
| 1.26 | 7 | 0 | 1 | 17 |

Εκτός αυτού δοκιμάστηκε και η εφαρμογή της οδηγίας διοχέτευσης στον ανώτερο ιεραρχικά βρόχο **outer_loop** (Εικ. 6.20 (στ)), πράγμα το οποίο έμμεσα ξετυλίγει εντελώς τον εσωτερικό βρόχο, όπως προαναφέρθηκε. Ωστόσο, μετά τη σύνθεση προέκυψε ότι η αρχιτεκτονική αυτή χρησιμοποιεί 110% των LUTs, οπότε βγαίνει εκτός επιλογής.

Το επόμενο επίπεδο στο οποίο εφαρμόστηκε μετασχηματισμός ξετυλίγματος είναι το 2ο συνελικτικό, διότι αποτελεί το αμέσως πιο αργό επίπεδο (Πίν. 6.3). Αναλυτικότερα, η οδηγία διοχέτευσης μεταφέρθηκε από το βρόχο **kernel_2** (Εικ. 6.20 (γ)) ένα επίπεδο παραπάνω, δηλαδή στο **kernel_1**. Όπως προαναφέρθηκε, οι εσωτερικοί βρόχοι αυτόματα ξετυλίγονται, οπότε με αυτή τη τεχνική ξετυλίγεται έμμεσα και ο **kernel_1**. Επιπλέον, και σε αυτή την περίπτωση δεν επιτυγχάνεται απευθείας βήμα διοχέτευσης ίσο με 1, εξαιτίας των εξαρτήσεων μεταξύ των δεδομένων. Δρώντας όπως και προηγουμένως, εφαρμόστηκε επιπλέον κατάτμηση στον πίνακα εισόδου (**πλήρης** αυτή τη φορά αντί για **μερική**). Συνθέτοντας εκ νέου βάσει των παραπάνω, λήφθηκαν τα αποτελέσματα του Πίν. 6.5. Πλέον, με 30% αύξηση στα LUTs επιτυγχάνεται 37% μείωση της καθυστέρησης σε σχέση με την αρχιτεκτονική του Πίν. 6.4.

Πίν. 6.5: Στατιστικά αρχιτεκτονικής ξετυλίγματος ενός επιπλέον βρόχου του 2ου συνελικτικού επιπέδου (αρχιτεκτονική **Ξετυλίγματος 2**).

| **Καθυστέρηση (ms)** | **Χρήση πόρων (%)** | | | |
|---|---|---|---|---|
| | BRAM-18K | DSP48E | FF | LUT |
| 0.80 | 8 | 0 | 3 | 47 |

Στο σημείο αυτό, οι ενέργειες που πραγματοποιήθηκαν στο 2ο συνελικτικό επίπεδο (μεταφορά οδηγίας διοχέτευσης και πλήρης κατάτμηση εισόδου), εφαρμόζονται και στο 1ο συνελικτικό, διότι αποτελεί το τρίτο πιο αργό επίπεδο. Ο Πίν. 6.6 καταγράφει τις μετρήσεις καθυστέρησης και χρήσης πόρων της αρχιτεκτονικής αυτής. Σε σύγκριση με τον Πίν. 6.5 διαπιστώνεται κυρίως αυξημένη χρήση κατά 41% σε LUTs, αλλά επιτάχυνση ίση με 15%.

Πίν. 6.6: Στατιστικά αρχιτεκτονικής ξετυλίγματος ενός επιπλέον βρόχου του 1ου συνελικτικού επιπέδου (αρχιτεκτονική **Ξετυλίγματος 3**).

| **Καθυστέρηση (ms)** | **Χρήση πόρων (%)** | | | |
|---|---|---|---|---|
| | BRAM-18K | DSP48E | FF | LUT |
| 0.68 | 8 | 0 | 4 | 88 |

### 6.4.4 Εφαρμογή μετασχηματισμού ροής δεδομένων

Προκειμένου να επιτευχθεί περεταίρω παραλληλοποίηση, εφαρμόζεται η οδηγία ροής δεδομένων στη συνολική αρχιτεκτονική. Η οδηγία αυτή πραγματοποιεί, επί της ουσίας, διοχέτευση σε επίπεδο συναρτήσεων (βλ. Υποπαρ. 6.3.4), δηλαδή στα επίπεδα του νευρωνικού δικτύου στην προκειμένη περίπτωση. Εφαρμόζοντας, λοιπόν, αυτή την οδηγία στις τρεις αρχιτεκτονικές που προέκυψαν μετά την εφαρμογή του μετασχηματισμού ξετυλίγματος (βλ. Υποπαρ. 6.4.3), λαμβάνονται αντιστοίχως τα στατιστικά που παρουσιάζονται στους Πίν. 6.7 έως Πίν. 6.9.

Πλέον, η διεκπεραιωτική ικανότητα δεν ταυτίζεται με την καθυστέρηση, οπότε εμφανίζονται και τα δύο μεγέθη στους πίνακες. Αν και η τελευταία αρχιτεκτονική χαρακτηρίζεται από μικρότερη καθυστέρηση, παρουσιάζει την ίδια διεκπεραιωτική ικανότητά σε σχέση με αυτή του Πίν. 6.8 με χρήση σχεδόν διπλάσιων, ωστόσο, LUTs. Αυτό σημαίνει πως και οι δύο δέχονται με την ίδια ταχύτητα νέα είσοδο.

Πίν. 6.7: Στατιστικά αρχιτεκτονικής με μετασχηματισμό ροής δεδομένων και ξετυλίγματος βρόχου του 1ου πλήρως διασυνδεδεμένου επιπέδου (αρχιτεκτονική **Ροής Δεδομένων 1**).

| **Καθυστέρηση (ms)** | **Διεκπεραιωτική ικανότητα (ms)** | **Χρήση πόρων (%)** | | | |
|---|---|---|---|---|---|
| | | BRAM-18K | DSP48E | FF | LUT |
| 1.26 | 0.54 | 7 | 0 | 1 | 17 |

Πίν. 6.8: Στατιστικά αρχιτεκτονικής με μετασχηματισμό ροής δεδομένων και ξετυλίγματος ενός επιπλέον βρόχου του 2ου συνελικτικού επιπέδου (αρχιτεκτονική **Ροής Δεδομένων 2**).

| **Καθυστέρηση (ms)** | **Διεκπεραιωτική ικανότητα (ms)** | **Χρήση πόρων (%)** | | | |
|---|---|---|---|---|---|
| | | BRAM-18K | DSP48E | FF | LUT |
| 0.80 | 0.39 | 8 | 0 | 3 | 46 |

Πίν. 6.9: Στατιστικά αρχιτεκτονικής με μετασχηματισμό ροής δεδομένων και ξετυλίγματος ενός επιπλέον βρόχου του 1ου συνελικτικού επιπέδου (αρχιτεκτονική **Ροής Δεδομένων 3**).

| **Καθυστέρηση (ms)** | **Διεκπεραιωτική ικανότητα (ms)** | **Χρήση πόρων (%)** | | | |
|---|---|---|---|---|---|
| | | BRAM-18K | DSP48E | FF | LUT |
| 0.68 | 0.39 | 8 | 0 | 4 | 87 |

## 6.5 Υλοποιήσεις και συγκρίσεις αρχιτεκτονικών

Ο Πίν. 6.10 συνοψίζει τα στατιστικά των αρχιτεκτονικών που αναλύθηκαν στην Παρ. 6.4 μαζί με την αρχική. Η **Αρχική** αρχιτεκτονική κάνει τη μικρότερη χρήση πόρων, αλλά ταυτόχρονα είναι και η πιο αργή. Ωστόσο, με την εφαρμογή διοχέτευσης σε όλα τα επίπεδα του LeNet-5 (αρχιτεκτονική **Διοχέτευσης 2**) επιτυγχάνεται επιτάχυνση κατά περίπου 7 φορές με τη χρήση, όμως, 5% παραπάνω LUTs. Συνολικά, λοιπόν, η επιτάχυνση είναι ραγδαία, αλλά και η δέσμευση πόρων του FPGA παραμένει ακόμα σε χαμηλά επίπεδα.

Πίν. 6.10: Σύγκριση των αρχιτεκτονικών (της αρχικής και των βελτιστοποιημένων).

| **Αρχιτεκτονική** | **Καθυστέρηση (ms)** | **Διεκπεραιωτική ικανότητα (ms)** | **Χρήση πόρων (%)** | | | |
|---|---|---|---|---|---|---|
| | | | BRAM-18K | DSP48E | FF | LUT |
| Αρχική | 10.6 | 10.6 | 7 | 0 | ≈0 | 5 |
| Διοχέτευσης 1 | 9.1 | 9.1 | 7 | 0 | ≈0 | 7 |
| Διοχέτευσης 2 | 1.57 | 1.57 | 7 | 0 | 1 | 10 |
| Ξετυλίγματος 1 | 1.26 | 1.26 | 7 | 0 | 1 | 17 |
| Ξετυλίγματος 2 | 0.8 | 0.8 | 8 | 0 | 3 | 47 |
| Ξετυλίγματος 3 | 0.68 | 0.68 | 8 | 0 | 4 | 88 |
| Ροής Δεδομένων 1 | 1.26 | 0.54 | 7 | 0 | 1 | 17 |
| Ροής Δεδομένων 2 | 0.8 | 0.39 | 8 | 0 | 3 | 46 |
| Ροής Δεδομένων 3 | 0.68 | 0.39 | 8 | 0 | 4 | 87 |

Οι αρχιτεκτονικές ξετυλίγματος καταλήγουν σε ακόμα μικρότερη καθυστέρηση, ωστόσο καμία από αυτές δεν κατορθώνει τόσο μεγάλη συγκριτική πτώση, όσο η αρχιτεκτονική **Ροής Δεδομένων 2**. Η αρχιτεκτονική **Ξετυλίγματος 2** είναι η πρώτη που ρίχνει την καθυστέρηση

κάτω από 1 ms με τη χρήση, όμως, σχεδόν του 50% των LUTs του FPGA. Περεταίρω επιτάχυνση κατορθώνει η αρχιτεκτονική **Ξετυλίγματος 3**, η οποία, παράλληλα, αποτελεί και την ακριβότερη αρχιτεκτονική από άποψη υλικού με χρήση σχεδόν του 90% των LUTs.

Ωστόσο, με τη χρήση πρακτικά του ίδιου υλικού με την οδηγία ροής δεδομένων αυξήθηκε η διεκπεραιωτική ικανότητα. Το τελευταίο διαπιστώνεται συγκρίνοντας τις αρχιτεκτονικές **Ξετιλίγματος** με τις παράγωγες **Ροής Δεδομένων** (**Ξετιλίγματος 1 → Ροής Δεδομένων 1** κλπ.). Όπως προαναφέρθηκε κατά την εφαρμογή του μετασχηματισμού ροής δεδομένων (βλ. Υποπαρ. 6.4.4), οι αρχιτεκτονικές **Ροής Δεδομένων 2** και **Ροής Δεδομένων 3** εμφανίζουν την ίδια διεκπεραιωτική ικανότητα με την τελευταία όμως να χρησιμοποιεί σχεδόν διπλάσιο υλικού. Παρά τη μικρότερη καθυστέρηση και οι δύο αρχιτεκτονικές μπορούν να δεχτούν νέα είσοδο με τον ίδιο ρυθμό.

Συνοψίζοντας, στην περίπτωση που τα περιθώρια υλικού είναι στενά κατάλληλες επιλογές αποτελούν οι **Αρχική** έως και την **Διοχέτευσης 2** με την τελευταία να είναι σημαντικά ταχύτερη, χωρίς σημαντική επιβάρυνση σε υλικό. Στην περίπτωση, όμως, που βασική απαίτηση είναι η ταχύτητα, καταλληλότερες είναι οι αρχιτεκτονικές **Ξετυλίγματος 1** έως **Ροής Δεδομένων 3**, οι οποίες επίσης παρουσιάζουν διαβαθμίσεις όσον αφορά την ταχύτητα αλλά και τη δέσμευση πόρων. Τελικά, για την υλοποίηση σε πλακέτα FPGA (βλ. Κεφ. 7) επιλέχτηκε η αρχιτεκτονική **Ξετυλίγματος2**.

# 7. ΥΛΟΠΟΙΗΣΗ ΤΟΥ LENET-5 ΣΕ ΑΝΑΠΤΥΞΙΑΚΟ ΣΥΣΤΗΜΑ

## 7.1 Εισαγωγή

Το σύνολο της ανάλυσης που πραγματοποιήθηκε στα προηγούμενα κεφάλαια αφορούσε μελέτες, διαδικασίες και πειράματα σε περιβάλλοντα εξομοίωσης. Ως τελικός, όμως, στόχος της παρούσας διπλωματικής εργασίας ήταν η εκτέλεση δοκιμών και σε πλακέτα FPGA. Γι' αυτό πραγματοποιήθηκε η επιβεβαίωση της λειτουργικότητας της σχεδίασης, μέσα από εφαρμογή υλοποίησης σε πραγματικό σύστημα.

Για να πραγματοποιηθεί η μεταγωγή αυτή, χρησιμοποιήθηκαν δύο *πλακέτες εκπαίδευσης και αξιολόγησης* (*Education and Evaluation Boards*), τα **ZedBoard Zynq-7000 ARM/FPGA SoC Development Board**[1] και **ZYNQ-7000 AP SoC ZC702 Evaluation Kit**[2]. Τα συστήματα αυτά, όπως αναλύεται παρακάτω, είναι πλήρως συμβατά με τις πλατφόρμες Vivado HLS, Vivado και Xilinx SDK. Παράλληλα, και τα δύο διαθέτουν *Κεντρική Μονάδα Επεξεργασίας* (*ΚΜΕ*) αρχιτεκτονικής ARM, Field Programmable Gate Array, καθώς και απαραίτητα περιφερειακά, όπως μνήμες DRAM και σειριακές θύρες. Τα παραπάνω επιτρέπουν την υλοποίηση πολύπλοκων υπολογιστικών δομών στις προαναφερθείσες

[1] ZedBoard Zynq-7000 ARM/FPGA SoC Development Board, https://www.xilinx.com/products/boards-and-kits/1-elhabt.html

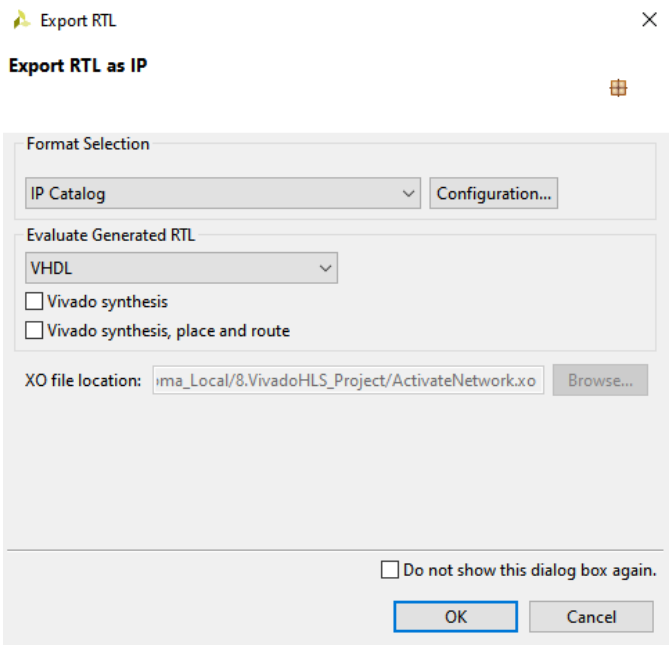
[2] Xilinx Zynq-7000 SoC ZC702 Evaluation Kit, https://www.xilinx.com/products/boards-and-kits/ek-z7-zc702-g.html

πλακέτες και τις καθιστούν κατάλληλες για την ενσωμάτωση του νευρωνικού δικτύου της παρούσας εργασίας εντός τους.

Το κεφάλαιο αυτό ξεκινά με την περιγραφή της διαδικασίας που ακολουθήθηκε, προκειμένου το μπλοκ του νευρωνικού δικτύου LeNet-5 που υλοποιήθηκε να φορτωθεί στο FPGA. Κατόπιν, παρουσιάζεται ο τρόπος με τον οποίο εξασφαλίστηκε η επικοινωνία μεταξύ του επεξεργαστή και του μπλοκ LeNet-5. Τέλος, γίνεται αναφορά σε περεταίρω βελτιώσεις που πραγματοποιήθηκαν στο φυσικό πλέον σύστημα (επεξεργαστής – LeNet-5), έτσι ώστε να αυτοματοποιηθεί η διαδικασία μέτρησης των επιδόσεων της υλοποίησης.

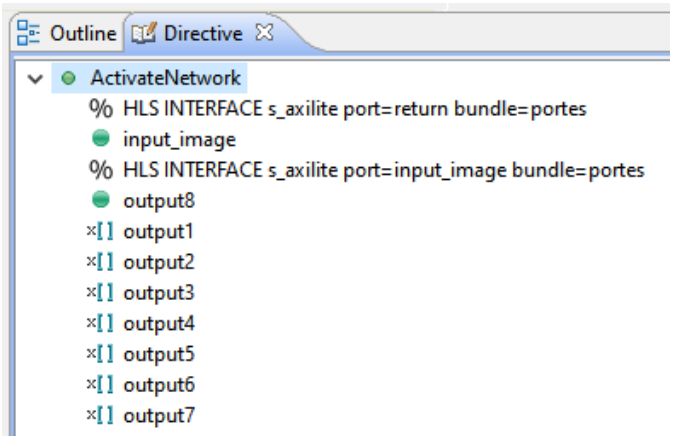
## 7.2 Σύνθεση κύριας μονάδας συν-επεξεργασίας

Το πρώτο βήμα για την εφαρμογή της υλοποίησης στα προαναφερθέντα αναπτυξιακά συστήματα ήταν η εξαγωγή του σε κατάλληλο *λειτουργικό πακέτο* (*Intellectual Property - IP*). Το εργαλείο Vivado HLS, το οποίο χρησιμοποιήθηκε κατά βάσιν για την διάρθρωση των υλοποιήσεών, παρέχει την δυνατότητα αυτή (Εικ. 7.1). Βέβαια, απαιτείται πρώτα ο κώδικας να έχει περάσει επιτυχώς από σύνθεση και προσομοίωση και να είναι μετατρέψιμος και λειτουργικός σε γλώσσα περιγραφής υλικού.




Εικ. 7.1: Λειτουργία δημιουργίας πακέτου IP στο Vivado HLS.

Σημαντικό βήμα πριν την εξαγωγή του κώδικα, είναι ο σωστός ορισμός των εξωτερικών διεπαφών που θα διαθέτει, καθώς και των πρωτοκόλλων επικοινωνίας που θα εξυπηρετεί. Συγκεκριμένα, για να είναι συμβατός με τα κοινά στάνταρ που χρησιμοποιούνται, ορίστηκαν διεπαφές τύπου *AXI* (*Advanced eXtensible Interface*). Προσθέτοντας κατάλληλες εντολές προεπεξεργαστή, ορίστηκε ότι το κύκλωμα θα εξυπηρετείται από μία δομή AXI-lite διεπαφής (μία ειδική έκδοση του AXI), η οποία αφιερώνεται τόσο στον έλεγχο του μπλοκ, όσο και την διεπαφή του με τις συναλλαγές δεδομένων εισόδου και εξόδου (Εικ. 7.2).




Εικ. 7.2: Ορισμός διεπαφής AXI-lite για το νευρωνικό δίκτυο στο εργαλείο Vivado HLS.

Για να είναι δυνατή η εκμετάλλευση των δυνατοτήτων που παρέχει το συγκεκριμένο πρωτόκολλο επικοινωνίας, δόθηκε βάση στο αντίστοιχο εγχειρίδιο της Xilinx [32], καθώς και σε παρουσιάσεις υλοποιήσεων διαθέσιμες στο διαδίκτυο [33]. Βασική σημείωση αποτελεί το ότι έπρεπε να μεταβληθεί ελαφρώς η σχεδίαση, έτσι ώστε το δίκτυο να δίνει μία έξοδο αντί για δέκα που είχε αρχικά. Η απλοποίηση αυτή οδηγεί στην εμφάνιση στην έξοδο ενός μόνο αριθμού, ο οποίος δηλώνει απευθείας την επικρατούσα κατηγορία την οποία το νευρωνικό δίκτυο αποδίδει στο εκάστοτε παράδειγμα εισόδου (Εικ. 7.3). Το βήμα αυτό δεν επηρεάζει σε ουσιαστικό βαθμό τα λειτουργικά και δομικά μεγέθη της υλοποίησης, καθώς η επιπρόσθετη λογική είναι πολύ συνεκτική (μερικοί συγκριτές). Επίσης, βοήθησε σε παράκαμψη κάποιων δυσκολιών που εμφανίστηκαν κατά την προσπέλαση των τιμών εξόδου, όταν σε πρώτη φάση το σύστημά φορτώθηκε στο FPGA βάσει της αρχικής υλοποίησης.

Επιβεβαιώνοντας ότι το σύστημα παράγει τα ίδια αποτελέσματα με πριν, πραγματοποιήθηκε εξαγωγή του σε πακέτο IP. Το πακέτο αυτό περιγράφει το νευρωνικό δίκτυο που αναπτύχθηκε, καθώς και τις συν αυτώ απαραίτητες διεπαφές.

```
int ActivateNetworkScalar(CL1_inp_t input_image[INPUT_DIMENSION][INPUT_DIMENSION]) {
    CL1_out_t output1[CONV1_NEURONS][OUTPUT1_DIM][OUTPUT1_DIM];
    PL1_out_t output2[CONV1_NEURONS][OUTPUT2_DIM][OUTPUT2_DIM];
    CL2_out_t output3[CONV2_NEURONS][OUTPUT3_DIM][OUTPUT3_DIM];
    PL2_out_t output4[CONV2_NEURONS][OUTPUT4_DIM][OUTPUT4_DIM];
    FL1_out_t output5[OUTPUT5_LENGTH];
    DL1_out_t output6[OUTPUT6_LENGTH];
    DL2_out_t output7[OUTPUT7_LENGTH];
    DL3_out_t output8[OUTPUT8_LENGTH];


    conv_layer1(input_image, output1);
    avg_pooling_layer1(output1, output2);
    conv_layer2(output2, output3);
    avg_pooling_layer2(output3, output4);
    flattening_layer(output4, output5);
    dense_layer1(output5, output6);
    dense_layer2(output6, output7);
    dense_layer3(output7, output8);

    int bestClass = 0;
    DL3_out_t bestValue = output8[0];
    for (int i = 1; i < 10; i++) {
        if (bestValue < output8[i]) {
            bestValue = output8[i];
            bestClass = i;
        }
    }
    return bestClass;
}
```

Εικ. 7.3: Κώδικας κύριας συνάρτησης μετά τις αλλαγές.

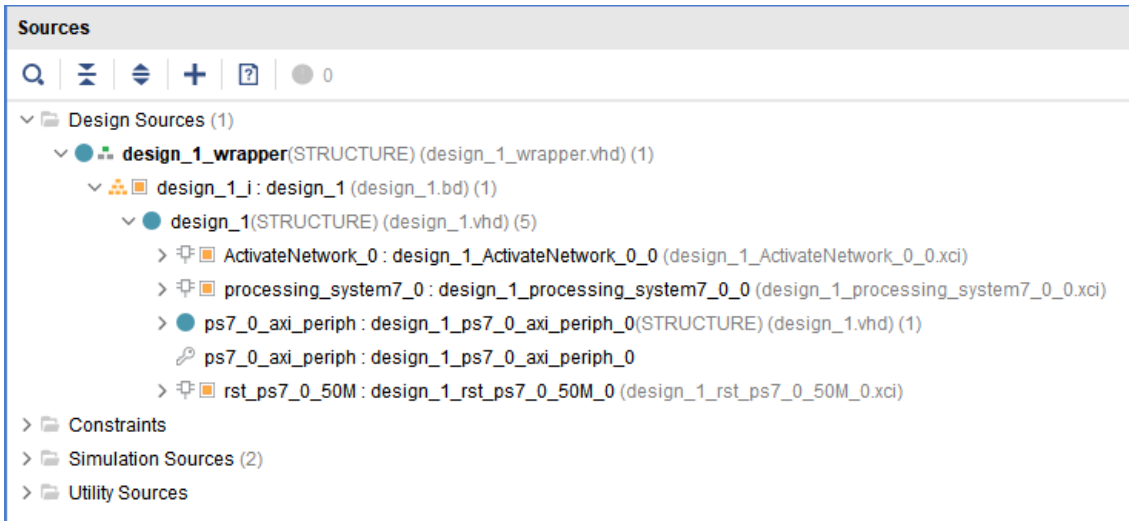
## 7.3 Δόμηση ολικού συστήματος

Ακολουθώντας τα απαραίτητα βήματα, όπως περιγράφονται στα κατάλληλα εγχειρίδια [34], το πακέτο IP εισήχθη στο περιβάλλον Vivado. Εδώ, η οπτική της σχεδίασης αφορά το πλήρες σύστημα αποτελούμενο από δομικά στοιχεία – μαύρα κουτιά, όπου οι εκάστοτε λειτουργίες τους είναι γνωστές. Πλέον, η διαδικασία αφορά την επιμέρους διασύνδεση των απαραίτητων δομικών στοιχείων μεταξύ τους, καθώς και με τυχόν απαραίτητα εξωτερικά σήματα. Για επιτάχυνση της διαδικασίας αυτής, χρησιμοποιήθηκαν οι βοηθητικές λειτουργίες της εισαγωγής απαραιτήτων μπλοκ διαμεσολάβησης, όπως το μπλοκ διασύνδεσης AXI (AXI Interconnect blocks), και η αυτόματη πραγματοποίηση των κύριων διαδρομών. Τελικά, τα μπλοκ που στελεχώνουν την υλοποίηση είναι τα εξής:

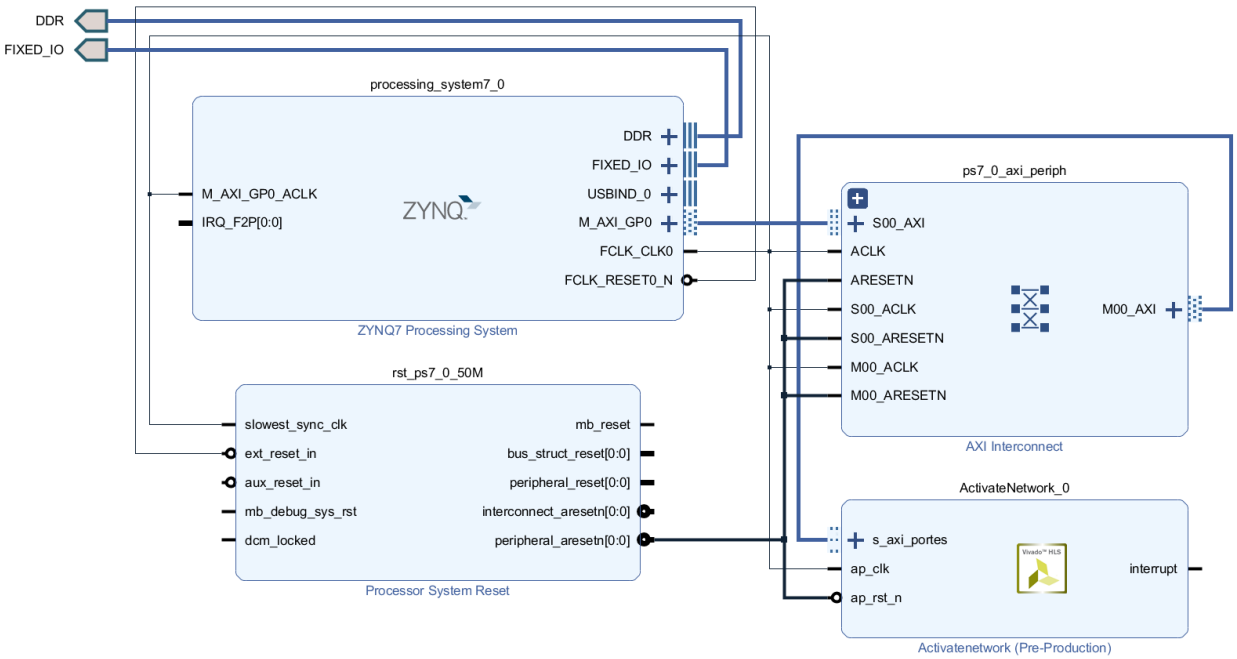
- **ZYNQ7 Processing System**: Η ΚΜΕ του συστήματος. Αποτελείται από έναν επεξεργαστή ARM Cortex A9 δύο πυρήνων. Αξίζει να σημειωθεί ότι επιλέχθηκαν συγκεκριμένες παραμετροποιήσεις του μπλοκ, χρησιμοποιώντας τις προεπιλεγμένες ρυθμίσεις της παραλλαγής ZC702. Η μόνες διαφορές αφορούν την παράλειψη του Timer 0 και την προσθήκη του σήματος διακοπής IRQ_F2P, όπως προτείνονται και από αντίστοιχους οδηγούς στο διαδίκτυο [19].

- **Activatenetwork**: Το μπλοκ που αναπτύχθηκε στην παρούσα μελέτη και ενσωματώνει το νευρωνικό δίκτυο LeNet-5. Διαθέτει κατάλληλα σήματα αρχικοποίησης, ελέγχου και μεταφοράς δεδομένων, ενώ περιέχει την AXI-lite διεπαφή που ορίστηκε στο εργαλείο Vivado HLS.
- **AXI Interconnect**: Επιτρέπει την επικοινωνία μεταξύ της ΚΜΕ (**ZYNQ7 Processing System**) και του νευρωνικού δικτύου (**Activatenetwork**). Χρησιμοποιήθηκε διάταξη με ένα **master** και ένα **slave port**.
- **Processor System Reset**: Προσφέρει έναν μηχανισμό για τον έλεγχο των σημάτων reset στο σύστημα, βάσει των συνθηκών λειτουργίας.

Έπειτα από εκτενή έλεγχο των καλωδιώσεων μεταξύ των υποσυστημάτων και προσεκτική υλοποίηση πιθανών διαδρομών που δεν τοποθετήθηκαν από το εργαλείο, πραγματοποιήθηκε η δημιουργία του απαραίτητου κώδικα διασύνδεσης (Εικ. 7.4) και, έπειτα, το πακετάρισμα του συστήματος σε *ροή bits* (*bitstream*) προγραμματισμού του FPGA. Αξιοσημείωτη μέριμνα πρέπει να δίνεται και στο αν οι διευθύνσεις που ανατίθεται στα επιμέρους υποσυστήματα επιτρέπουν ορθή επικοινωνία μεταξύ τους και δεν υπάρχουν επικαλυπτόμενες ή ελλιπείς θέσεις. Η τελική διάταξη του συστήματος με όλα τα δομικά μπλοκ και τις διασυνδέσεις μεταξύ του φαίνεται στην Εικ. 7.5.




Εικ. 7.4: Οπτική ανώτατου επιπέδου της τελικής δομής.




Εικ. 7.5: Τελική διάταξη του συστήματος.

## 7.4 Προγραμματισμός του FPGA και του μικροεπεξεργαστή

Ακολουθώντας τις παραπάνω διαδικασίες, είναι πλέον διαθέσιμο ένα ολοκληρωμένο σύστημα ακολουθιακής λογικής, έτοιμο για εγγραφή στην πλακέτα. Το τελευταίο βήμα αφορά την ρύθμιση της πλακέτας (βάσει των οδηγιών [34]) και τη δημιουργία κατάλληλου λογισμικού για την υποστήριξη της συγκεκριμένης δομής.

Για το σκοπό αυτό, χρησιμοποιήθηκε το εργαλείο Xilinx Software Development Kit (SDK). Κατά την εξαγωγή του bitstream από το Vivado, εισήχθησαν τα παραχθέντα του πρότζεκτ στο SDK. Οι απαραίτητοι οδηγοί που επιτρέπουν την διασύνδεση μεταξύ της ακολουθιακής λογικής και του διαθέσιμου διπύρηνου επεξεργαστή ARM Cortex™-A9 παράγονται αυτόματα από το εργαλείο. Όπως είναι προφανές, ήταν απαραίτητη η σχολαστική μελέτη των παραπάνω οδηγών, προκειμένου να καταγραφούν οι διαθέσιμες ρουτίνες και οι αναγκαίες σταθερές. Συγκεκριμένα, οι ρουτίνες αφορούν την πρόσβαση στις λειτουργίες του συστήματος (προκαλώντας την έναρξη των υπολογισμών, τον έλεγχο της κατάστασης του συστήματος κ.τ.λ.) ενώ οι σταθερές αφορούν τις διευθύνσεις μνήμης των μεταβλητών.

Η ιδέα, σύμφωνα με την οποία λειτουργεί η διάταξη, βασίζεται στο ότι η ΚΜΕ δίνει κατάλληλες εντολές στην μονάδα ακολουθιακής λογικής που υλοποιεί το νευρωνικό δίκτυο, μέσω των υφιστάμενων διεπαφών AXI. Οι εντολές αυτές αφορούν τον έλεγχο του FPGA,

ώστε αυτό να διαβάσει τις απαραίτητες θέσεις μνήμης που αποθηκεύονται οι εικόνες, να προκληθεί η ενεργοποίηση του δικτύου και τέλος να επιστραφούν τα αποτελέσματα στην ΚΜΕ. Μετά την αποσφαλμάτωση τόσο των διασυνδέσεων μεταξύ των μπλοκ, όσο και του κώδικα προγραμματισμού της ΚΜΕ, το σύστημα αποδείχθηκε πλήρως λειτουργικό και δοκιμάστηκε επιτυχώς για μικρό αριθμό εικόνων. Για περεταίρω έλεγχο της διακριτικής του ικανότητας, χρειάστηκε να αναπτυχθεί ένας μηχανισμός που επιτρέπει την αυτόματη ενεργοποίηση του δικτύου για μεγάλο αριθμό εικόνων. Προς αυτό το σκοπό, αναπτύχθηκαν κατάλληλες ρουτίνες και αντίστοιχο πρωτόκολλο, όπως περιγράφεται στην Παρ. 7.5.

## 7.5 Επιπλέον ανάπτυξη του συστήματος

### 7.5.1 Αποστολή εικόνων μέσω της σειριακής θύρας

Πηγαίνοντας την υλοποίηση ένα βήμα παραπέρα, δοκιμάστηκε και εφαρμόστηκε η αποστολή των εικόνων εισόδου μέσω σειριακής θύρας. Η ιδέα αυτή προέκυψε από την ανάγκη να αντιμετωπιστεί η έλλειψη διαθέσιμου αποθηκευτικού χώρου. Ο ARM Cortex-A9 εντός της πλακέτας διαθέτει εσωτερική μνήμη (on-chip RAM) συνολικής χωρητικότητας 256 KB [35]. Αντίθετα, οι διαθέσιμες εικόνες του σετ δεδομένων ξεπερνούν κατά πολύ αυτό το ποσό, με το συνολικό τους μέγεθος να αγγίζει τα 10 MB.

Άλλωστε, ακόμα και να χρησιμοποιείτο η εξωτερική μνήμη DRAM της πλακέτας (η οποία φτάνει τα 512 MB DDR3 για το Zedboard [36]), πάλι τα δεδομένα θα έπρεπε να διέλθουν μέσω της σειριακής θύρας, ώστε να αποθηκευτούν προσωρινά στην μνήμη (μεταφορά μέσω Ethernet δεν εξετάστηκε ως λύση, καθώς θεωρήθηκε ότι υπερβαίνει τους στόχους της παρούσας διπλωματικής εργασίας). Συνεπώς, η εξοικείωση με την μεταφορά των εικόνων μέσω της σειριακής θύρας αποτέλεσε άμεσο στόχο και εργαλείο για περεταίρω ανάπτυξη του συστήματος. Οδήγησε, επίσης, σε μία υλοποίηση που είναι εύκολα επαναχρησιμοποιήσιμη.

Για την κατασκευή μίας τέτοιας διάταξης, ήταν απαραίτητη η δημιουργία κατάλληλων διεργασιών λήψης των εικόνων από την ΚΜΕ (ώστε να μπορεί να τις αποστείλει έπειτα στο νευρωνικό δίκτυο για περεταίρω επεξεργασία). Αντίστοιχα, αναπτύχθηκε ένα πρόγραμμα σε Python (σε εξωτερικό υπολογιστή – αποστολέα), έτσι ώστε να προσπελαύνει αρχεία από το

σετ δεδομένων MNIST και να τα αποστέλλει μέσω σειριακής θύρας. Για την επίτευξη ορθής επικοινωνίας μεταξύ των δύο τερματικών, αναπτύχθηκε και εφαρμόστηκε ένα *προσαρμοσμένο στην υλοποίηση* (*custom*) πρωτόκολλο επικοινωνίας βασισμένο σε *χειραψία* (*handshaking*). Το πρωτόκολλο αυτό περιγράφεται από τα εξής βήματα:

**Βήμα 1**: Ο κύριος υπολογιστής (**ΚΥ**) αποστέλλει έναν χαρακτήρα **SOH** (Start Of Heading) στην πλακέτα ανάπτυξης (**ΠΑ**) για την έναρξη της συνεδρίας. Έπειτα, για κάθε εικόνα ακολουθούνται τα **βήματα 2 έως 6**.

**Βήμα 2:** Αποστολή (από **ΚΥ**) χαρακτήρα **STX** (Start of TeXt).

**Βήμα 3:** Αποστολή (από **ΚΥ**) τιμής κλάσης (**label**) και έπειτα χαρακτήρας **CR** (Carriage Return).

**Βήμα 4:** Αποστολή (από **ΚΥ**) με τη σειρά όλων των 1024 (32 x 32) pixels της εικόνας, το καθένα ακολουθούμενο από χαρακτήρα **CR**.

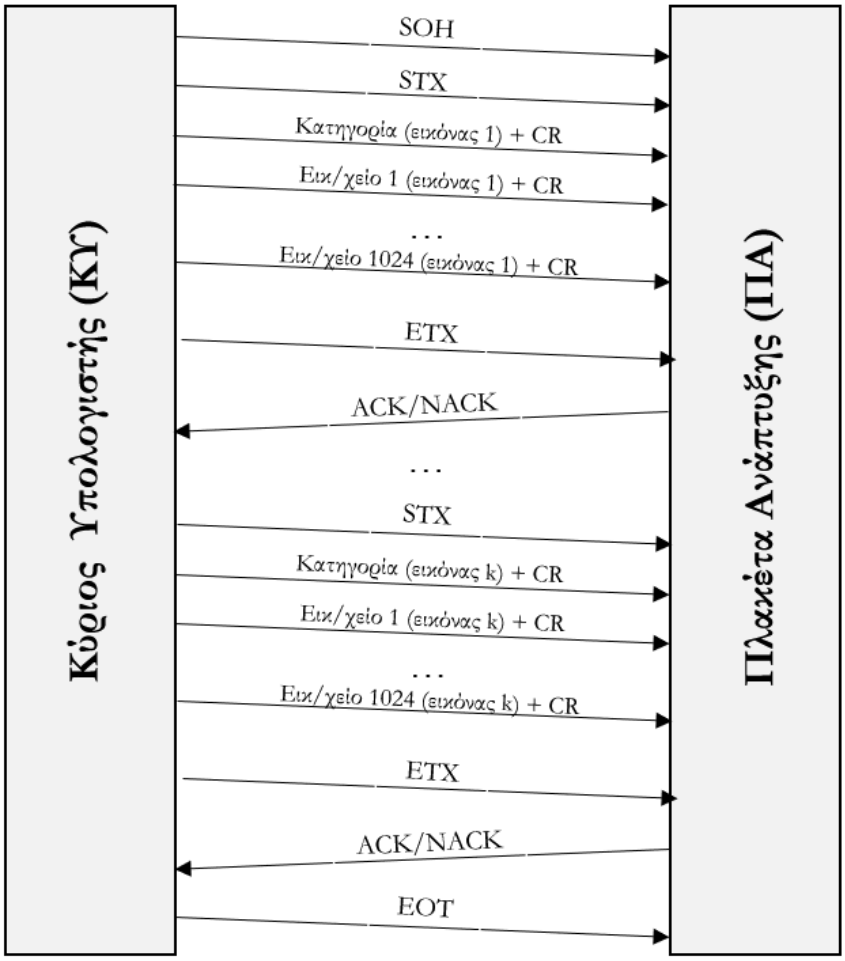
**Βήμα 5:** Αποστολή (από **ΚΥ**) χαρακτήρα **ETX** (End of TeXt).

**Βήμα 6:** Αποστολή (από **ΠΑ**) **ACK** (ACKnowledgement) ή **NACK** (Negative ACKnowledgement), ανάλογα με το αν το δίκτυο εκτίμησε ορθά την κατηγορία για την παρούσα εικόνα (σε σχέση με την κατηγορία που δόθηκε ως ορθή από τον αποστολέα).

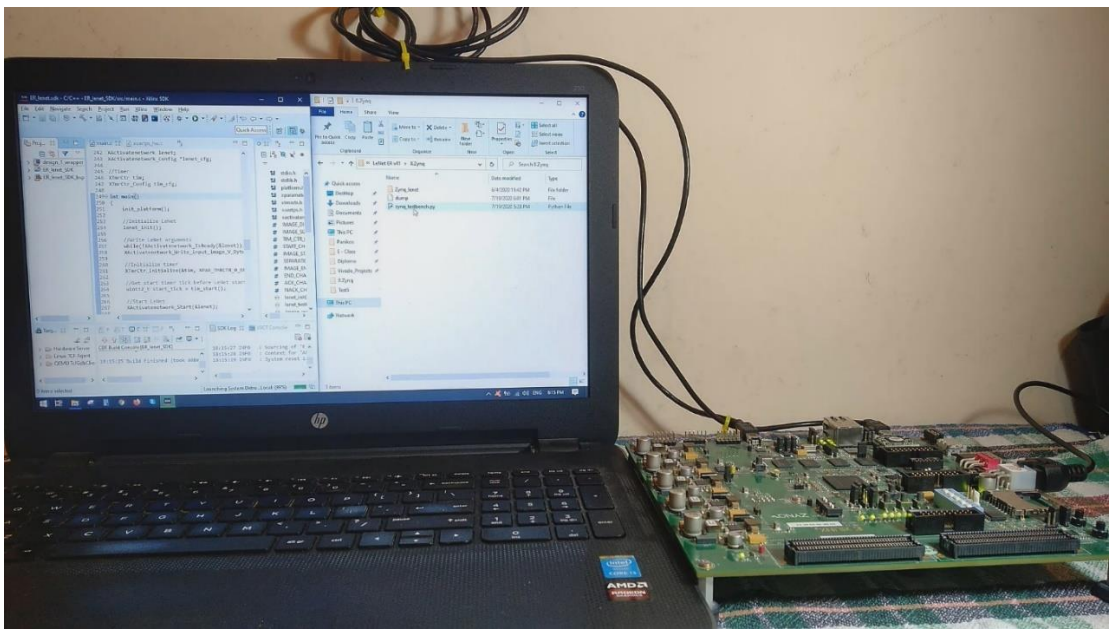
**Βήμα 7**: Μετά την επιτυχή αποστολή και επεξεργασία και της τελευταίας εικόνας, η ρουτίνα αποστολής σηματοδοτεί τη λήξη της επικοινωνίας μέσω ενός χαρακτήρα **EOT** (End Of Transaction).

Σχηματικά, η παραπάνω διαδικασία παρουσιάζεται και στην Εικ. 7.6. Ο χειρισμός αυτός πραγματοποιήθηκε για αυθαίρετο αριθμό εικόνων εισόδου, όπου επιβεβαιώθηκε η ορθή λειτουργία του συστήματος. Η διάταξη και η συνδεσμολογία της πλακέτας ανάπτυξης (**ΠΑ**) και του κύριου υπολογιστή (**ΚΥ**) που υλοποιήθηκε φαίνεται στην Εικ. 7.7. Εφαρμόζοντας, λοιπόν, τις πρώτες 1000 εικόνες του σετ δοκιμών ως είσοδο, παρατηρήθηκε πλήρης αντιστοίχιση **θεωρητικών** (αποτελέσματα των προσομοιώσεων στο Vivado HLS) και **πειραματικών** αποτελεσμάτων. Το σύστημα, δηλαδή, πέτυχε **TOP-1** ακρίβεια ίση με **96.1%** (βλ. Εικ. 7.8), η οποία συμβαδίζει με την αντίστοιχη τιμή που παρουσιάζεται στον Πίν. 5.4. Συνεπώς, διαπιστώθηκε ότι η λειτουργία co-simulation του εργαλείου Vivado HLS είναι μία

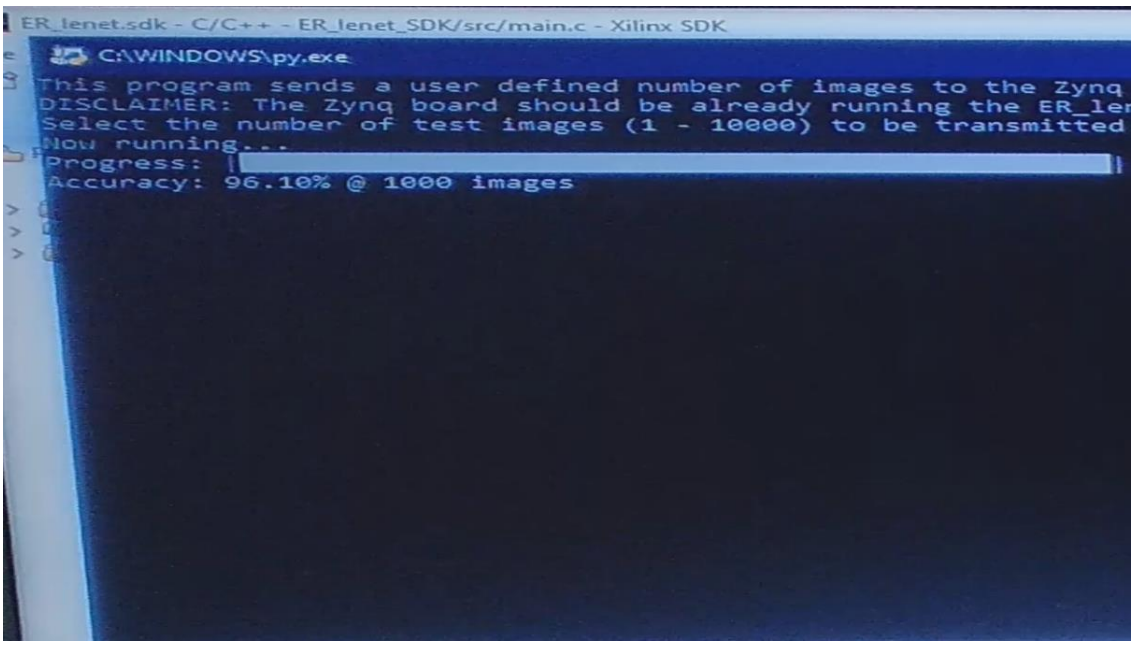
εξαιρετική αναπαράσταση της λειτουργίας του πραγματικού κυκλώματος, όσον αφορά την διακριτική του ικανότητα.




Εικ. 7.6: Πρωτόκολλο επικοινωνίας που χρησιμοποιήθηκε για την αποστολή των εικόνων.



Εικ. 7.7: Διάταξη και συνδεσμολογία πλακέτας ανάπτυξης και κύριου υπολογιστή.




Εικ. 7.8: Ακρίβεια του LeNet-5 για 1.000 εικόνες του σετ δοκιμών κατά τη λειτουργία του στην πλακέτα ανάπτυξης.

Τέλος, αξίζει να αναφερθεί, ότι στην παρούσα υλοποίηση, για την εξυπηρέτηση των σημάτων εισόδου-εξόδου, χρησιμοποιήθηκε διαδικασία *περιοδικού ελέγχου*[1] (*polling*). Η αναμονή των απαντήσεων μεταξύ αποστολέα και παραλήπτη γίνεται με συνεχείς ελέγχους και όχι με έλευση κάποιας *διακοπής*[2] (*interrupt*) [37]. Η επιλογή αυτή έχει να κάνει με την απλούστευση του μηχανισμού επικοινωνίας και με την ευκολότερη ανάπτυξη και αποσφαλμάτωσή του.

### 7.5.2 Μέτρηση της συχνότητας λειτουργίας

Για τον έλεγχο της ταχύτητας του συστήματος, προστέθηκε ένας μηχανισμός μέτρησης χρόνου. Η βιβλιοθήκη που χρησιμοποιήθηκε είναι η «xtmrctr.h», βάσει της οποίας χτίστηκαν ρουτίνες έναρξης, δειγματοληψίας και λήξης του χρονομετρητή (Εικ. 7.9).

[1] Αυτή είναι η διαδικασία κατά την οποία γίνεται συνεχόμενα έλεγχος μίας συνθήκης από την επεξεργαστική μονάδα και δεν συνεχίζεται η εκτέλεση του προγράμματος, μέχρι αυτή η συνθήκη να ικανοποιηθεί (συνήθως με την έλευση κάποιου εξωτερικού σήματος).

[2] Έτσι ονομάζεται η διαδικασία κατά την οποία η επεξεργαστική μονάδα εκτελεί προκαθορισμένο τμήμα κώδικα, όταν διεγερθεί με κατάλληλο εξωτερικό σήμα (σήμα διακοπής). Με τον τρόπο αυτό, η επεξεργαστική μονάδα δεν υποχρεούται να σπαταλά κύκλους μηχανής αναμένοντας την έλευση ενός τέτοιου εξωτερικού σήματος.

```c
uint32_t tim_start(void) {
    XTmrCtr_Reset(&tim, TIM_CTR_NUM);
    uint32_t tick = XTmrCtr_GetValue(&tim, TIM_CTR_NUM);
    XTmrCtr_Start(&tim, TIM_CTR_NUM);
    return tick;
}

uint32_t tim_stop(void) {
    XTmrCtr_Stop(&tim, TIM_CTR_NUM);
    return XTmrCtr_GetValue(&tim, TIM_CTR_NUM);
}

double tim_elapsed_time_seconds(uint32_t start_tick, uint32_t end_tick) {
    double period = 1.0 / tim.Config.SysClockFreqHz;
    return (double) ((end_tick - start_tick) * period);
}
```

Εικ. 7.9: Ρουτίνες ελέγχου του χρονομετρητή.

Ο χρονομετρητής χρησιμοποιήθηκε με σκοπό να πραγματοποιείται παρακολούθηση του της χρονικής διάρκειας που απαιτείται ώστε να πραγματοποιηθεί η επεξεργασία των δεδομένων από το νευρωνικό δίκτυο. Εκκινώντας τον κατά την αποστολή του σήματος έναρξης της τροφοδοσίας του δικτύου και λήγοντας τον κατά την ολοκλήρωση της κατηγοριοποίησης, λήφθηκε το χρονικό διάστημα που απαιτείται για την λειτουργία του. Στην Εικ. 7.10 παρουσιάζεται ένα κομμάτι κώδικα που αφορά την αρχικοποίηση και ενεργοποίηση του νευρωνικού δικτύου για μία εικόνα εισόδου, όπου συμπεριλαμβάνονται οι εντολές για τη χρήση του χρονομετρητή (**έναρξη**, **λήξη** και **υπολογισμός χρονικού διαστήματος**).

```c
init_platform(); // Initializing the system
lenet_init(); // Initializing the Neural Network

while (!XActivatenetwork_IsReady(&lenet)); // Waiting until the NN is ready
XActivatenetwork_Write_input_image_V_Bytes(&lenet, 0, (char *)images[2], IMAGE_SIZE); // Writing NN arguments

XTmrCtr_Initialize(&tim, XPAR_TMRCTR_0_DEVICE_ID); // Initializing the timer
uint32_t start_tick = tim_start(); // Getting a start timer tick before the network starts

XActivatenetwork_Start(&lenet); // Starting the NN
while (!XActivatenetwork_IsDone(&lenet)); // Polling - Waiting for the NN to finish calculations

uint32_t end_tick = tim_stop(); // Getting an end timer tick after the NNs ends

uint8_t label = XActivatenetwork_Get_return(&lenet); // Gathering the NN's output when ready

double seconds = tim_elapsed_time_seconds(start_tick, end_tick); // Calculating total time elapsed
printf("Label: %u\n", label);
printf("LeNet run time in milliseconds: %.3f\n", 1000 * seconds);
```

Εικ. 7.10: Παράδειγμα αρχικοποίησης και ενεργοποίησης νευρωνικού δικτύου και χρονομετρητή.

Χρησιμοποιώντας τον χρονομετρητή αυτόν κατά την επαναληπτική ενεργοποίηση του νευρωνικού δικτύου για αρκετές εικόνες, παρατηρήθηκε ένας μέσος όρος καθυστέρησης ίσος με **2.5 ms** ανά εικόνα. Η τιμή αυτή είναι περίπου τριπλάσια από την αναμενόμενη τιμή των

**0.8 ms** (βλ. Πίν. 6.10). Αυτό μάλλον οφείλεται σε δύο παράγοντες. Ο πρώτος λόγος πιθανότατα είναι η αδυναμία του εργαλείου να λάβει υπόψιν του με ακρίβεια όλες τις πηγές καθυστέρησης, όπως τη μεταφορά δεδομένων και τον επιπλέον χρόνο που προστίθεται λόγω της λειτουργίας του πρωτοκόλλου επικοινωνίας AXI-lite. Όμως, ίσως ο πιο σημαντικός παράγοντας που προκάλεσε αυτή τη διαφορά μεταξύ θεωρητικής και πειραματικής τιμής στην καθυστέρηση, αφορά την συχνότητα λειτουργίας του συστήματος. Το εργαλείο σχεδίασης, κατά την εξαγωγή του συστήματος προς εφαρμογή στην αναπτυξιακή πλακέτα, αύξανε αυτόματα την περίοδο λειτουργίας στο **διπλάσιο**, δηλαδή από τα **10ns** στα **20ns**, παρόλο που η υλοποίησή ικανοποιούσε τους χρονικούς περιορισμούς για λειτουργία στα **10ns** (όπως είχε μοντελοποιηθεί στις προσομοιώσεις). Οι δύο παραπάνω λόγοι δικαιολογούν πλήρως τον υποτριπλασιασμό της ταχύτητας του συστήματος και οφείλονται αμφότεροι σε αδυναμία του εργαλείου.

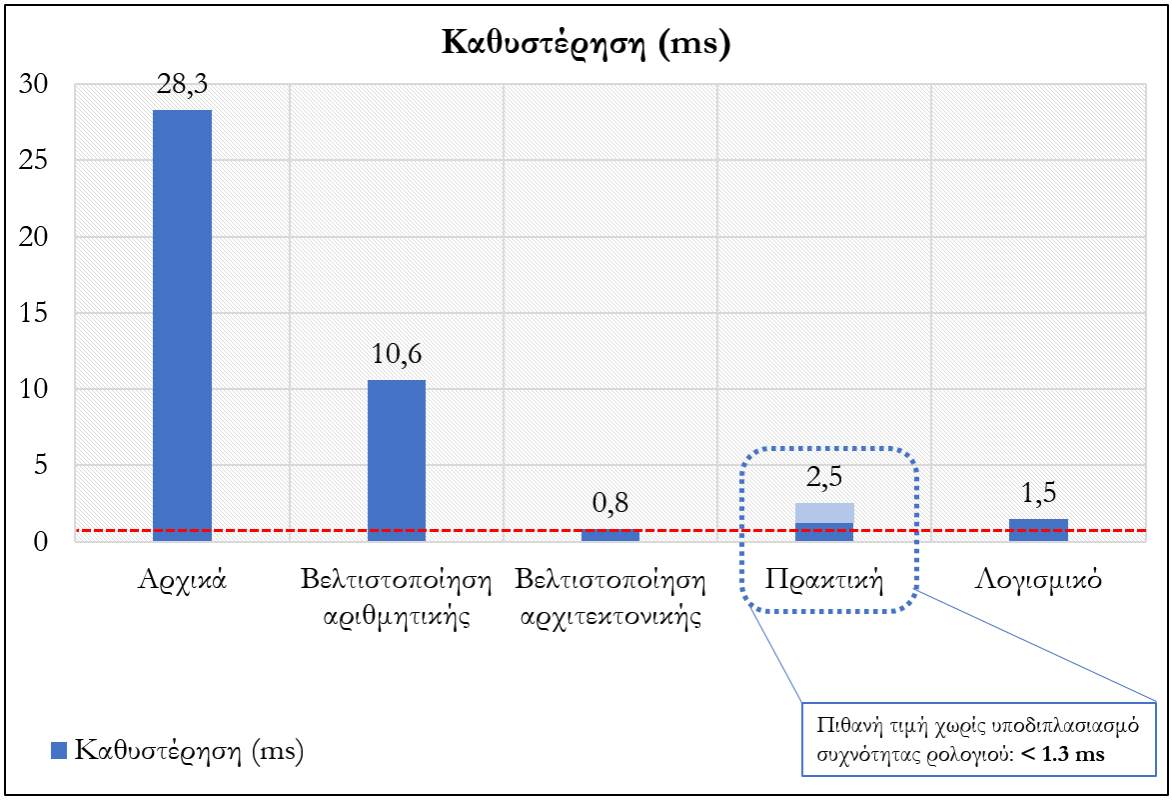
Κλείνοντας, αξίζει να πραγματοποιηθεί μία σύγκριση της επιτάχυνσης που επιτεύχθηκε συνολικά. Συγκεκριμένα, μετρήθηκε η ταχύτητα αναγνώρισης του νευρωνικού δικτύου LeNet-5 σε επίπεδο λογισμικού (χρησιμοποιώντας τη C/C++ υλοποίηση της Παρ. 4.3) και παρατηρήθηκε καθυστέρηση ίση με **1.5 ms** ανά εικόνα. Συγκρίνοντάς την τιμή αυτή με την καθυστέρηση των **0.8 ms** που επιτεύχθηκε μέσω της υλοποίησης σε υλικό[1], παρατηρείται ότι η υλοποίηση σε υλικό για το LeNet-5 ξεπερνά σε ταχύτητα την αντίστοιχη υλοποίηση σε λογισμικό.

Συγκεντρωτικά, οι καθυστερήσεις των υλοποιήσεων που πραγματοποιήθηκαν στην παρούσα διπλωματική εργασία συνοψίζονται στο διάγραμμα της Εικ. 7.11. Στο διάγραμμα αυτό συγκρίνονται ως προς την καθυστέρηση οι εξής υλοποιήσεις:

- **Αρχική**: η πρώτη υλοποίηση σε υλικό (βλ. Κεφ. 4).
- **Βελτιστοποίηση αριθμητικής**: η υλοποίηση που προέκυψε μετά από τη συμπίεση του νευρωνικού δικτύου και επιλογή κατάλληλης αριθμητικής (βλ. Κεφ. 5).
- **Βελτιστοποίηση αρχιτεκτονικής**: η υλοποίηση σε υλικό μετά από τους μετασχηματισμούς κώδικα (βλ. Κεφ. 6).
- **Πρακτική**: η φυσική υλοποίηση σε αναπτυξιακό σύστημα (παρόν κεφάλαιο).

[1] Εδώ γίνεται χρήση της θεωρητικής (0.8 ms) και όχι της πρακτικής (2.5 ms) τιμής, καθώς η δεύτερη, όπως προαναφέρθηκε, δεν αντικατοπτρίζει σωστά την ιδανική ταχύτητα της σχεδίασης, λόγω των αστοχιών του εργαλείου που περιεγράφηκαν.

- **Λογισμικό**: η υλοποίηση λογισμικού σε C και εκτέλεσή της σε περιβάλλον Windows 10.




Εικ. 7.11: Σύγκριση των καθυστερήσεων των υλοποιήσεων που πραγματοποιήθηκαν στην παρούσα διπλωματική εργασία.

# 8. ΕΠΙΛΟΓΟΣ

Η παρούσα διπλωματική εργασία είχε ως στόχο την δημιουργία και βελτιστοποίηση ενός συστήματος αναγνώρισης εικόνων, ακολουθώντας όλα τα στάδια ανάπτυξης μιας πλήρους υλοποίησης. Τα στάδια αυτά συνοψίζονται ως εξής:

1. Ορισμός της επιθυμητής λειτουργικότητας του συστήματος και μελέτη της αντίστοιχης βιβλιογραφίας.
2. Επιλογή ενός μοντέλου νευρωνικού δικτύου (LeNet-5) και προσαρμογή του στις απαιτήσεις που ορίστηκαν.
3. Υλοποίηση και εκπαίδευση του δικτύου αυτού σε λογισμικό, χρησιμοποιώντας πλατφόρμες όπως το Keras και το Caffe.
4. Μετατροπή του μοντέλου και των δεδομένων του γλώσσα χαμηλού επιπέδου και, έπειτα, σε υλικό, μέσω της χρήσης εργαλείου σύνθεσης υψηλού επιπέδου.
5. Συμπίεση της αρχικής υλοποίησης, εξετάζοντας μεθόδους κβαντισμού των παραμέτρων και των δεδομένων του νευρωνικού δικτύου.
6. Πραγματοποίηση βελτιστοποιήσεων σε επίπεδο αλγορίθμου και αρχιτεκτονικής, εξετάζοντας και συγκρίνοντας εναλλακτικές σχεδιάσεις.
7. Επιλογή μίας αρχιτεκτονικής και φυσική υλοποίησή της σε πλακέτα FPGA, συνθέτοντας παράλληλα ένα πλήρως λειτουργικό ενσωματωμένο σύστημα.

Σημειώνεται ότι για την ολοκλήρωση των παραπάνω σταδίων χρησιμοποιήθηκε μια πληθώρα από προγραμματιστικά και σχεδιαστικά εργαλεία, όπως η πλατφόρμα λογισμικού Vivado Design Suite, το προγραμματιστικό περιβάλλον Visual Studio, το εργαλείο Ristretto κ.ά..

Βασικό αποτέλεσμα της εργασίας είναι ότι δομήθηκε μία ροή σχεδιασμού που είναι επαναχρησιμοποιήσιμη. Ακολουθώντας, δηλαδή, τα ίδια βήματα και χρησιμοποιώντας τα βοηθητικά προγράμματα που δημιουργήθηκαν, δύναται να μοντελοποιηθούν και άλλα νευρωνικά δίκτυα. Άλλωστε, η επεκτασιμότητα της των προϊόντων της εργασίας αυτής ήταν εξαρχής επιθυμητή.

Όσον αφορά τα εργαλεία που χρησιμοποιήθηκαν, βασικό συμπέρασμα που εξήχθη κατά τη σχεδίαση με τη βοήθεια του Vivado HLS, είναι η ευκολία ανάπτυξης και μετατροπής του κώδικα σε γλώσσα περιγραφής υλικού. Παράλληλα, προσφέρονται ποικίλες δυνατότητες σχετικά με την παραμετροποίηση και τον έλεγχο της σωστής λειτουργικότητας του συστήματος, επιτρέποντας στον σχεδιαστή να πειραματιστεί με εναλλακτικές αρχιτεκτονικές. Επίσης, οι εκτιμήσεις του εργαλείου, όσον αφορά τη διακριτική ικανότητα του συστήματος, αποδεικνύονται πλήρως συμβατές με τα πραγματικά δεδομένα που προκύπτουν από την εφαρμογή του κώδικα σε πραγματική πλακέτα. Όμως, το αντίστοιχο δεν ισχύει πάντα όσον αφορά τη συχνότητα λειτουργίας και την καθυστέρηση.

Τέλος, η διπλωματική εργασία αυτή δεν ανέδειξε απλώς μία υλοποίηση ενός νευρωνικού δικτύου σε αναπτυξιακό σύστημα, αλλά και τη βελτιστοποίηση αυτής, κυρίως όσον αφορά την καθυστέρηση. Ξεκινώντας από μία σχεδίαση κινητής υποδιαστολής 32-bit (με καθυστέρηση ανά εικόνα ~**30 ms**), πραγματοποιήθηκαν παραμετροποιήσεις αριθμητικής, αναδόμηση κώδικα και βελτιστοποιήσεις κυκλώματος, καταλήγοντας σε μία αρκετά ταχύτερη σχεδίαση (με καθυστέρηση ανά εικόνα ~**1 ms**). Επιτεύχθηκε, δηλαδή, μία επιτάχυνση περίπου **30 φορές**.

# ΒΙΒΛΙΟΓΡΑΦΙΑ


[1] C. M. Bishop, Pattern Recognition and Machine Learning, 2006.

[2] R. Dass, «Pattern Recognition Techniques: A Review,» 2018.

[3] S. T. Konstantinos Koutroumbas, Pattern Recognition, 2008.

[4] S. Elie, «An overview of Pattern Recognition,» 2013.

[5] F. Rosenblatt, «The Perceptron - A perceiving and recognizing algorithm,» 1957.

[6] A. S. W. S. H. N. Alan V. Oppenheim, Signals and Systems, 1997.

[7] A. F. M. Agarap, «Deep Learning using Rectified Linear Units (ReLU),» 2019.

[8] Google, «Multi-Class Neural Networks: Softmax,» [Ηλεκτρονικό]. Available: https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax.

[9] L. P. Bolin Gao, «On the Properties of the Softmax Function with Application in Game Theory and Reinforcement Learning,» 2017.

[10] R. Rojas, Neural Networks: A Systematic Introduction, 1996.

[11] S. L. L. G. Rich Caruana, «Overfitting in Neural Nets: Backpropagation,Conjugate Gradient, and Early Stopping,» 2001.

[12] L. B. Y. B. P. H. Yann LeCun, «Gradient-Based Learning Applied to Document Recognition,» 1998.

[13] «The MNIST database of handwritten digits,» [Ηλεκτρονικό]. Available: http://yann.lecun.com/exdb/mnist/.

[14] M. Gazar, 2018. [Ηλεκτρονικό]. Available: https://colab.research.google.com/drive/1CVm50PGE4vhtB5I_a_yc4h5F-itKOVL9.

[15] J. L. B. Diederik P. Kingma, «Adam: a Method for Stochastic Optimization,» σε *ICLR*, 2015.

[16] «Xilinx community forums,» [Ηλεκτρονικό]. Available: https://forums.xilinx.com/.

[17] H. Giesen, «Vivado HLS,» [Ηλεκτρονικό]. Available: https://fling.seas.upenn.edu/~giesen/dynamic/wordpress/vivado-hls-learnings/.

[18] Xilinx, «Introduction to FPGA Design with Vivado High-Level Synthesis,» 2019. [Ηλεκτρονικό].

[19] Xilinx, «Vivado Design Suite TutorialHigh-Level Synthesis,» 2019. [Ηλεκτρονικό].

[20] D. W. P. Z. T. Z. Yu Cheng, «A Survey of Model Compression and Acceleration,» *IEEE SIGNAL PROCESSING MAGAZINE,* 2020.

[21] K. H. a. W. S. Sajid Anwar, «Fixed Point Optimization of Deep Convolutional Neural Networks for Object Recognition,» 2015.

[22] J. A. T. H. T. A. N. E. J. R. U. A. M. Patrick Judd, «Reduced-Precision Strategies for Bounded Memory in Deep Neural Nets,» 2016.

[23] P. Gysel, «Ristretto: Hardware-Oriented Approximation of Convolutional Neural Networks,» σε *M.Sc. Thesis*, 2016.

[24] «Ristretto, CNN Approximation,» [Ηλεκτρονικό]. Available: http://lepsucd.com/ristretto-cnn-approximation.

[25] «The CIFAR-10 dataset,» [Ηλεκτρονικό]. Available: https://www.cs.toronto.edu/~kriz/cifar.html.

[26] J. P. M. M. S. G. Philipp Gysel, «Ristretto: A Framework for Empirical Study of Resource-Efficient Inference in Convolutional Neural Networks,» 2018.

[27] «Caffe, Deep Learning Framework,» [Ηλεκτρονικό]. Available: https://caffe.berkeleyvision.org.

[28] «Caffe, LeNet example,» [Ηλεκτρονικό]. Available: https://caffe.berkeleyvision.org/gathered/examples/mnist.html.

[29] S. L. G. O. J. S. David F. Bacon, «Compiler Transformations for High-Performance Computing,» 1994.

[30] J. H. David Paterson, Computer Organization and Design, 2010.

[31] Xilinx, «Vivado Design Suite User GuideHigh-Level Synthesis,» 2019. [Ηλεκτρονικό].

[32] Xilinx, «AXI Reference Guide,» 2011. [Ηλεκτρονικό].

[33] M. Sadri, «ZYNQ Training,» [Ηλεκτρονικό]. Available: http://www.googoolia.com/wp/category/zynq-training/page/2/.

[34] Xilinx, «Zynq-7000 SoC: Embedded Design Tutorial,» 2011. [Ηλεκτρονικό].

[35] Xilinx, «Zynq-7000 SoC Data Sheet: Overview,» 2018. [Ηλεκτρονικό].

[36] AVNET, «ZedBoard Getting Started Guide,» 2017. [Ηλεκτρονικό].

[37] S. Heath, Embedded Systems Design, 2002.

[38] Wikipedia, «Αναγνώριση Προτύπων,» [Ηλεκτρονικό]. Available: https://el.wikipedia.org/wiki/Αναγνώριση_προτύπων.

[39] A. A. K. G. P. N. Suyog Gupta, «Deep Learning with Limited Numerical Precision,» 2015.