ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Θέμα διπλωματικής εργασίας:

ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΥΠΟΛΟΓΙΣΤΙΚΑ ΑΡΓΩΝ ΚΟΜΜΑΤΙΩΝ ΑΛΓΟΡΙΘΜΩΝ ΑΝΑΓΝΩΡΙΣΗΣ ΠΡΟΣΩΠΟΥ ΚΑΙ ΓΕΝΕΤΙΚΗΣ ΜΕ ΧΡΗΣΗ ΑΝΑΔΙΑΤΑΣΣΟΜΕΝΗΣ ΛΟΓΙΚΗΣ

ΣΩΤΗΡΟΠΟΥΛΟΣ ΙΩΑΝΝΗΣ

Εξεταστική επιτροπή:

Παπαευσταθίου Ιωάννης Επίκουρος Καθηγητής (Επιβλέπων)	Δόλλας Απόστολος Καθηγητής	Πνευματικάτος Διονύσιος Αναπληρωτής Καθηγητής

Χανιά 2008

ΕΥΧΑΡΙΣΤΙΕΣ

Ολοκληρώνοντας τη διπλωματική αυτή εργασία, θα ήθελα να ευχαριστήσω όλους όσους μου συμπαραστάθηκαν κατά τη διάρκεια της φοίτησής μου στο Πολυτεχνείο Κρήτης.

Ιδιαίτερα θα ήθελα να ευχαριστήσω τον καθηγητή μου κ. Παπαευσταθίου Ιωάννη, για την επίβλεψη καθώς και την σημαντική καθοδήγησή του κατά την διάρκεια της διπλωματικής μου εργασίας.

Ακόμη θα ήθελα να ευχαριστήσω τους καθηγητές κ. Δόλλα Απόστολο και κ. Πνευματικάτο Διονύσιο για τη συνεισφορά τους σε αυτή την εργασία, καθώς και τον κ. Μάρκο Κιμιωνή, μέλος ΕΕΔΙΠ και υπεύθυνο του εργαστηρίου Μικροεπεξεργαστών και Υλικού, για την υποστήριξη του σε τεχνικά ζητήματα.

Επίσης, νιώθω την ανάγκη να ευχαριστήσω τους διδακτορικούς φοιτητές κ. Ευρυπίδη Σωτηριάδη, κ. Κυπριανό Παπαδημητρίου, κ. Δημήτριο Χρυσό και κ. Γρηγόρη Χρυσό για τη συμβουλές τους κατά τη διάρκεια εκπόνησης της εργασίας, όπως και όλους τους μεταπτυχιακούς (και ιδιαίτερα τον κ. Γεώργιο Μπλεμένο και τον κ. Κωνσταντίνο Παπαδόπουλο) αλλά και προπτυχιακούς φοιτητές του εργαστηρίου για την στήριξή τους και την καλή συνεργασία που είχαμε.

Τέλος, οφείλω ένα μεγάλο ευχαριστώ στην οικογένειά μου αλλά και στους φίλους μου, που βοήθησαν, αυτά τα πέντε χρόνια φοίτησης στο Πολυτεχνείο Κρήτης, να γίνουν εκτός από δημιουργικά, και αξέχαστα.

ΠΕΡΙΛΗΨΗ

Σκοπός της παρούσας διπλωματικής εργασίας είναι η βελτιστοποίηση χρόνων εκτέλεσης αλγορίθμων, οι οποίοι καταλαμβάνουν το μεγαλύτερο μερίδιο στο συνολικό χρόνο εκτέλεσης των συστημάτων που χρησιμοποιούνται. Πιο συγκεκριμένα, τα δύο συστήματα με τα οποία ασχολούμαστε αφορούν το ένα μία εφαρμογή αξιολόγησης αλγορίθμων αναγνώρισης προσώπου και το δεύτερο μία εφαρμογή επιλογής καρκινικών γονιδίων.

Η πρώτη εφαρμογή σχεδιάστηκε από το Πανεπιστήμιο του Κολοράντο και τους Ross Beveridge, David Bolme, Marcio Teixeira and Bruce Draper και ονομάζεται “CSU Face Identification Evaluation System ”. Το υπολογιστικά αργό κομμάτι της σχεδίασης αφορά τον αλγόριθμο πολλαπλασιασμού πινάκων μεγάλων διαστάσεων.

Η δεύτερη εφαρμογή προτάθηκε από τους I.Guyon, J.Weston, S.Barnhill και V.Vapnik και ονομάζεται “Gene selection for cancer classification using Support Vector Machines (SVMs)”. Το αργό υπολογιστικά κομμάτι της συγκεκριμένης σχεδίασης αφορά τον αλγόριθμο δυαδικής αναζήτησης στοιχείων και διαγραφής τους.

Για τη βελτιστοποίηση των χρόνων αυτών εκμεταλλευόμαστε τα πλεονεκτήματα που μας παρέχει ο, ολοένα και περισσότερο αναπτυσσόμενος τις τελευταίες δεκαετίες, τομέας των Field Programmable Gate Arrays (FPGAs). Σχεδιάζονται παράλληλες υλοποιήσεις των αλγορίθμων, με σκοπό την όσο το δυνατόν μεγαλύτερη επιτάχυνση του χρόνου εκτέλεσής τους.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΚΟΝΕΣ

ΚΕΦΑΛΑΙΟ 2 : ΠΑΡΟΥΣΙΑΣΗ ΤΩΝ ΓΕΝΙΚΩΝ ΑΛΓΟΡΙΘΜΩΝ

ΚΕΦΑΛΑΙΟ 3 : ΣΧΕΤΙΚΗ ΕΡΓΑΣΙΑ

ΚΕΦΑΛΑΙΟ 4 : ΥΛΟΠΟΙΗΣΗ ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΥ ΠΙΝΑΚΩΝ ΣΕ ΑΝΑΔΙΑΤΑΣΣΟΜΕΝΗ ΛΟΓΙΚΗ

ΚΕΦΑΛΑΙΟ 5 : ΥΛΟΠΟΙΗΣΗ ΔΥΑΔΙΚΗΣ ΑΝΑΖΗΤΗΣΗΣ ΣΕ ΑΝΑΔΙΑΤΑΣΣΟΜΕΝΗ ΛΟΓΙΚΗ

ΚΕΦΑΛΑΙΟ 6 : ΑΠΟΔΟΣΗ ΣΥΣΤΗΜΑΤΟΣ ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΥ ΠΙΝΑΚΩΝ – ΣΥΓΚΡΙΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

ΚΕΦΑΛΑΙΟ 7 : ΑΠΟΔΟΣΗ ΣΥΣΤΗΜΑΤΟΣ ΔΥΑΔΙΚΗΣ ΑΝΑΖΗΤΗΣΗΣ – ΣΥΓΚΡΙΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

1

ΕΙΣΑΓΩΓΗ

1.1 Αντικείμενο της διπλωματικής

Στην παρούσα διπλωματική παρουσιάζονται δύο υλοποιήσεις αλγορίθμων με χρήση αναδιατασσόμενης λογικής, ένας για παράλληλο πολλαπλασιασμό πινάκων και ένας για παράλληλη δυαδική αναζήτηση και διαγραφή στοιχείων. Και οι δύο σχεδιάσεις είναι ρυθμισμένες με βάση τις απαιτήσεις των συστημάτων που περιλαμβάνουν αυτούς τους αλγορίθμους.

Όσον αφορά το "CSU Face Identification Evaluation System", χρησιμοποιεί πολλαπλασιασμούς πινάκων μεγάλων διαστάσεων (εκατομμυρίων στοιχείων) κάτι το οποίο καθυστερεί κατά πολύ την εκτέλεση του συστήματος. Είναι χαρακτηριστικό ότι για κάθε αλγόριθμο αναγνώρισης προσώπου που χρησιμοποιεί και αξιολογεί το σύστημα, οι πολλαπλασιασμοί πινάκων κοστίζουν πάνω από το 80% του χρόνου εκτέλεσής τους.

Η υλοποίηση του συστήματος "Gene selection for cancer classification using SVMs" προσπαθώντας να επιλέξει γονίδια που διαθέτουν συγκεκριμένα καρκινικά χαρακτηριστικά επαναλαμβάνει μία διαδικασία δυαδικής αναζήτησης στοιχείων από μία μεγάλη βάση δεδομένων που αποτελείται από γονίδια. Η κατά δεκάδες χιλιάδες φορές επανάληψη αυτής της διαδικασίας μετατρέπει αυτόν τον αλγόριθμο στο αργότερο κομμάτι του συστήματος, επιβαρύνοντας τον τελικό χρόνο εκτέλεσης του συστήματος κατά αρκετά δευτερόλεπτα.

1.2 Δομή της εργασίας

Η περιγραφή της διπλωματικής εργασίας αποτελείται από 8 κεφάλαια. Στα κεφάλαια αυτά γίνεται αναλυτική περιγραφή της διαδικασίας η οποία ακολουθήθηκε κατά την εκπόνηση της εργασίας. Συγκεκριμένα, περιγράφονται έννοιες σχετικές με το θέμα και παρόμοιες εργασίες, παρουσιάζονται λεπτομέρειες των μονάδων που σχεδιάστηκαν καθώς και τα τελικά αποτελέσματα που προκύπτουν από αυτές.

Πιο αναλυτικά:

Στο $2^{ο}$ κεφάλαιο γίνεται μια αναφορά στα συστήματα τα οποία χρησιμοποιούν τους αλγορίθμους που επιχειρήθηκε να βελτιστοποιηθούν.

Στο $3^{ο}$ κεφάλαιο παρουσιάζονται κάποιες παρόμοιες δουλειές που έχουν υλοποιηθεί από άλλους ερευνητές.

Στα κεφάλαια 4 και 5, γίνεται η ανάλυση της αρχιτεκτονικής κάθε σχεδίασης.

Τα κεφάλαια 6 και 7 παρουσιάζουν τα αποτελέσματα που προκύπτουν από κάθε μονάδα που υλοποιήθηκε.

Στο 8^{ο} κεφάλαιο παρουσιάζονται κάποιες προτάσεις για βελτιστοποιήσεις και επεκτάσεις που μπορούν να γίνουν πάνω στις συγκεκριμένες υλοποιήσεις.

Στο 9^{ο} κεφάλαιο παρουσιάζεται η βιβλιογραφία που χρησιμοποιήθηκε για την διεκπεραίωση της διπλωματικής.

2

Παρουσίαση των γενικών συστημάτων

2.1 Εισαγωγή

Στο κεφάλαιο αυτό θα γίνει μια σύντομη παρουσίαση των δύο αλγορίθμων, των οποίων τα υπολογιστικά αργά κομμάτια υλοποιήσαμε σε αναδιατασσόμενη λογική. Ο πρώτος αλγόριθμος που παρουσιάζεται (CSU Face Identification Evaluation System, version 5.0) αφορά την υλοποίηση και αξιολόγηση διαφόρων μεθόδων αναγνώρισης προσώπου. Ο δεύτερος αλγόριθμος αναφέρεται στην επιλογή γονιδίων για καρκινική ταξινόμηση με τη χρήση μιας νέας τεχνικής ταξινόμησης (Gene Selection for Cancer Classification using Support Vector Machines).

2.2 Παρουσίαση του συστήματος "CSU Face Identification Evaluation System"

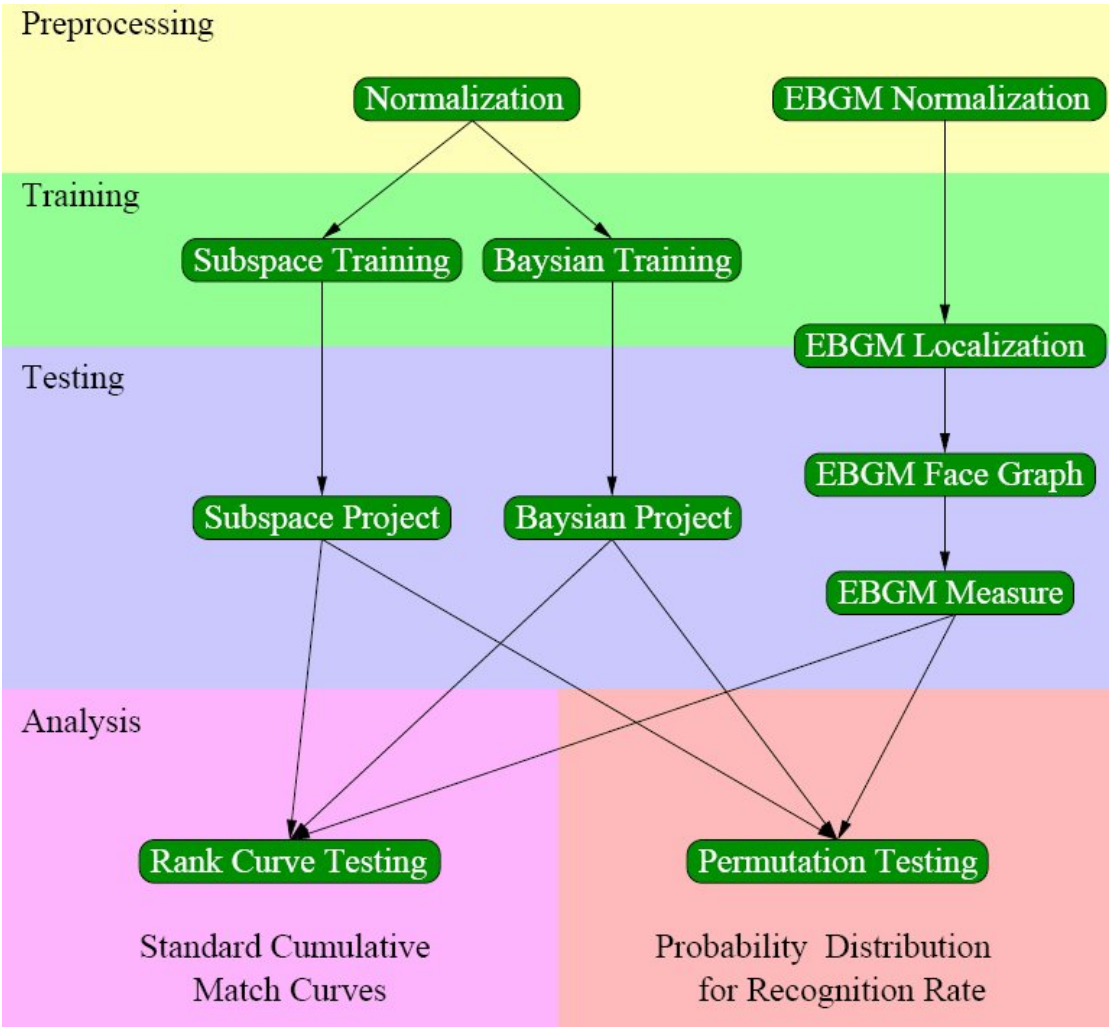
2.2.1 Εισαγωγή

Η υλοποίηση του "Colorado State University(CSU) Face Identification Evaluation System" περιέχει ένα σύνολο από 4 αλγορίθμους

αναγνώρισης προσώπου και 2 στατιστικές μεθόδους για τη σύγκριση αυτών των αλγορίθμων. Οι τέσσερεις αλγόριθμοι, που περιγράφονται παρακάτω, είναι οι: "PCA (Principle Components Analysis)", PCA+LDA (συνδυαστικός αλγόριθμος των "Principle Components Analysis" και "Discriminant Analysis Algorithm"), BIC (Bayesian Intrapersona/Extrapersonal Classifier) και EBGM (Elastic Bunch Graph Matching). Με τον τελευταίο αλγόριθμο δε θα ασχοληθούμε διότι η υλοποίησή του ακολουθεί διαφορετική δομή από τους υπόλοιπους. Για λόγους πληρότητας αναφέρουμε ότι οι δύο στατιστικές μέθοδοι που χρησιμοποιήθηκαν είναι οι "csuAnalyzeRankCurve" και "csuAnalyzePermute".

Παρουσιάζοντας συνοπτικά τη λειτουργία των αλγορίθμων αναγνώρισης προσώπου, θα λέγαμε ότι κάθε ένας από αυτούς υπολογίζει τις αποστάσεις μεταξύ κάθε ζεύγους φωτογραφιών της βάσης δεδομένων του συστήματος. Η συνολική υλοποίηση διαχωρίζεται, όπως παρουσιάζεται και στην εικόνα 2.1 σε τέσσερα βασικά στάδια: προεπεξεργασία των δεδομένων εικόνας, αλγόριθμος εκπαίδευσης, αλγόριθμος ελέγχου και ανάλυση των αποτελεσμάτων.

Πολλαπλασιασμοί πινάκων εκτελούνται στα στάδια Training και Testing, όπου εκτελούνται οι αλγόριθμοι PCA, PCA+LDA και BIC. Πολλαπλασιασμοί διανυσμάτων μικρών διαστάσεων εκτελούνται και στο στάδιο του Preprocessing αλλά εξαιτίας του συγκριτικά μικρού χρόνου εκτέλεσής τους, η παρούσα εργασία δεν ασχολείται με αυτό το στάδιο.



Εικόνα 2.1: Διάγραμμα Ροής εκτέλεσης του CSU Face Identification Evaluation System

2.2.2 Προεπεξεργασία (Preprocessing)

Όλες οι εικόνες της βάσης δεδομένων περνάνε από μία διαδικασία προεπεξεργασίας στην οποία ακολουθούνται τα παρακάτω βήματα:

- Μετατροπή της εικόνας, που είναι τύπου PGM, σε διάνυσμα στοιχείων.
- Γεωμετρική κανονικοποίηση, μέσω κλιμάκωσης, αντιστροφής και αναστροφής της εικόνας, έτσι ώστε να είναι κεντραρισμένη και συμμετρική σε σχέση με τα μάτια του προσώπου.

- Χρησιμοποίηση ελλειπτικής μάσκας, μέσω της οποίας η εικόνα κόβεται, ώστε να μείνει ορατό σε αυτήν μόνο η περιοχή του προσώπου από το πηγούνι μέχρι το κούτελο.
- Το ιστόγραμμα της εικόνας ισοσταθμίζεται μέσω της χρήσης ενός κβαντιστή.
- Οι τιμές του κάθε pixel της εικόνας κλιμακώνονται αναλόγως, ώστε να έχουν όλες μέση τιμή μηδέν.

Στο τέλος της προεπεξεργασίας όλες οι εικόνες έχουν την ίδια μορφή. Ένα παράδειγμα παρουσιάζεται στην εικόνα 2.1.2.

Εικόνα 2.2: Παραδείγματα εικόνων πριν και μετά το στάδιο της προεπεξεργασίας.

2.2.3 Αλγόριθμος Principle Components Analysis (PCA)

Ο αλγόριθμος βασίζεται σε μία γραμμική μετατροπή ενός χώρου χαρακτηριστικών (feature space), δημιουργώντας, μέσω μιας διαδικασίας εκπαίδευσης, διανύσματα χαρακτηριστικών (feature vectors) από τις τιμές των pixel της εικόνας. Κάθε τέτοιο διάνυσμα έχει περίπου 20.000 τιμές και είναι έντονα συσχετισμένο με τα υπόλοιπα διανύσματα της εικόνας. Ο αλγόριθμος μετατρέπει τον μεγάλο υποχώρο που περιέχει αυτά τα διανύσματα, σε έναν αρκετά μικρότερο, ενώ εξαλείφει από τα τροποποιημένα (εκπαιδευμένα) διανύσματα χαρακτηριστικών όλη τη στατιστική συμμεταβολή, κάτι που είναι πολύ χρήσιμο για τη μετέπειτα επεξεργασία των εικόνων. Ένα πλεονέκτημα του αλγορίθμου είναι ότι καταφέρνει να μειώσει αρκετά τις διαστάσεις των χαρακτηριστικών διανυσμάτων. Η διαδικασία της εκπαίδευσης παράγει ένα αρχείο που περιέχει τις παραμέτρους εκπαίδευσης, τη μέση τιμή κάθε εκπαιδευμένης εικόνας και τη βάση του υποχώρου. Με βάση τη λίστα των εικόνων και το αρχείο του εκπαιδευμένου υποχώρου παράγονται "αρχεία απόστασης", τα οποία περιέχουν την απόσταση κάθε εικόνας σε σχέση με όλες τις άλλες εικόνες.

2.2.4 Αλγόριθμος Linear Discriminant Analysis (PCA+LDA)

Ο αλγόριθμος LDA δημιουργεί έναν υποχώρο ο οποίος διαχωρίζεται γραμμικά σε κλάσεις, όπου κάθε κλάση αποτελείται από εικόνες του ίδιου προσώπου. Απαραίτητη προϋπόθεση για την ορθή εκτέλεση του αλγορίθμου είναι να υπάρχουν πάνω από μίας διαφορετικές εικόνας για κάθε πρόσωπο. Στο στάδιο της εκπαίδευσης του αλγορίθμου, αρχικά χρησιμοποιείται η PCA εκπαίδευση, όπου αποφασίζεται για να δεδομένα μία βέλτιστη βάση και οι εικόνες εκπαίδευσης προβάλλονται στον

υποχώρο του PCA, για να μειωθούν οι διαστάσεις του διανύσματος. Στη συνέχεια εφαρμόζεται μία επιπρόσθετη εκπαίδευση για τη βελτιστοποίηση των χαρακτηριστικών διανυσμάτων. Τελικά ο αλγόριθμος παράγει αντίστοιχα δεδομένα με αυτά του PCA, και αντίστοιχα "αρχεία απόστασης".

2.2.5 Αλγόριθμος Bayesian Intrapersona/Extrapersonal Classifier (BIC)

Ο Bayesian I/E Classifier είναι ο πιο πολύπλοκος από αυτή την ομάδα αλγορίθμων. Βασικό χαρακτηριστικό του είναι ότι ορίζει δύο υποχώρους: τον intrapersonal και τον extrapersonal υποχώρο. Σε αντίθεση με τη μέθοδο LDA (που επιλέγει μια προβολή που αντιπροσωπεύει καλύτερα τα δεδομένα του προσώπου, λαμβάνοντας υπόψη τις διαφορετικές κλάσεις), ο BIC ορίζει τις κλάσεις χρησιμοποιώντας τη διαφορά μεταξύ των αντίστοιχων pixels δύο διαφορετικών φωτογραφιών, και την χρησιμοποιεί για να αποφασίσει αν οι δύο φωτογραφίες ανήκουν στο ίδιο πρόσωπο. Αν οι νέες εικόνες που παράγονται από τη διαφορά δύο φωτογραφιών, προκύπτουν από φωτογραφίες από διαφορετικά πρόσωπα, τότε χαρακτηρίζονται ως extrapersonal, ενώ αν προκύπτουν από δύο φωτογραφίες του ίδιου προσώπου χαρακτηρίζονται ως intrapersonal.

Μέσω μίας διαδικασίας εκπαίδευσης, που είναι όμοια με αυτή που ακολουθείται στην εκπαίδευση του PCA, εκτιμούνται οι στατιστικές ιδιότητες των δύο υποχώρων. Όταν η εκπαίδευση ολοκληρώνεται παράγονται δύο αρχεία, ένα για κάθε υποχώρο, που περιλαμβάνουν μία περιγραφή των παραμέτρων εκπαίδευσης, τη μέση τιμή της εκπαιδευμένης εικόνας, και ένα σύνολο διανυσμάτων βάσης του υποχώρου. Η διαδικασία που ακολουθεί αφορά στον υπολογισμό

διαφορών είτε μέσω της μεθόδου Maximum Likelihood (ML) , η οποία χρησιμοποιεί πληροφορία που προέρχεται μόνο από εικόνες intrapersonal, είτε μέσω της Maximum a Posteriori (MAP), η οποία χρησιμοποιεί πληροφορία που προέρχεται και από τις δύο κατηγορίες. Έτσι, τα διανύσματα χαρακτηριστικών προβάλλονται σε κάθε ένα από τα δύο σύνολα των διανυσμάτων βάσης και υπολογίζεται η πιθανότητα κάθε χαρακτηριστικό διάνυσμα να ανήκει στον έναν ή στον άλλον υποχώρο. Η τελική έξοδος του αλγορίθμου είναι ένα σύνολο από αρχεία "αποστάσεων", που περιέχουν τις αποστάσεις από κάθε εικόνα προς κάθε άλλη εικόνα.

2.2.6 Αλγόριθμος Elastic Bunch Graph Matching (EBGM)

Η κύρια ιδέα του αλγορίθμου είναι ο εντοπισμός συγκεκριμένων σημείων σε μία εικόνα, όπως είναι τα μάτια, η μύτη και το στόμα, από όπου εξάγονται αντίστοιχες τιμές, οι οποίες χρησιμοποιούνται για να δημιουργηθεί ένας γράφος προσώπου, που αντιστοιχεί σε κάθε φωτογραφία. Ο γράφος προσώπου εξυπηρετεί τον ίδιο σκοπό με τα διανύσματα χαρακτηριστικών που χρησιμοποιούνται στους προηγούμενους αλγόριθμους. Μετά τη δημιουργία των γράφων από κάθε εικόνα, ο αλγόριθμος για να καταλήξει σε αποτέλεσμα μετράει την ομοιότητα των γράφων. Όπως αναφέρθηκε και στην εισαγωγική παράγραφο, ο αλγόριθμος αυτός, εφόσον δεν ακολουθεί ίδια γενική δομή με τους άλλους τρεις (δε χρησιμοποιεί διανύσματα χαρακτηριστικών για μία εικόνα), δεν καλεί τις ρουτίνες για τους πολλαπλασιασμούς πινάκων και επομένως δεν αφορά την παρούσα διπλωματική.

2.3 Παρουσίαση του συστήματος “Gene Selection for Cancer Classification using Support Vector Machines”

2.3.1 Εισαγωγή

Η χρήση των DNA micro-arrays έδωσε τη δυνατότητα στους επιστήμονες να εξετάζουν ταυτοχρόνως χιλιάδες γονίδια και να αποφασίζουν εάν τα γονίδια αυτά είναι ενεργά, υπερενεργά ή αδρανή σε φυσιολογικούς ή καρκινικούς ιστούς. Εξαιτίας των πολύ μεγάλων ποσοτήτων δεδομένων που παράγουν αυτές οι μηχανές των DNA micro-arrays, νέες αναλυτικές μέθοδοι οφείλουν να εφαρμοστούν για να διακρίνουν εάν καρκινικοί ιστοί έχουν διακριτά σημάδια γενετικής έκφρασης σε σχέση με υγιείς ιστούς, ή άλλου τύπου καρκινικούς ιστούς.

Στην ενότητα αυτή παρουσιάζεται μία αποτελεσματική λύση για το πρόβλημα της επιλογής ενός μικρού υποσυνόλου γονιδίων από ευρεία σύνολα δεδομένων γενετικής έκφρασης (αποθηκευμένα σε DNA micro arrays).

Χρησιμοποιώντας διαθέσιμα παραδείγματα εκπαίδευσης από καρκινικά και υγιή δείγματα, δημιουργήθηκε ένας classifier με χρήση των SVMs και της μεθόδου Recursive Feature Elimination (RFE). Συγκεντρωτικά, η μέθοδος που προτείνεται παράγει καλύτερη απόδοση ταξινόμησης, και αποδεικνύεται ότι τα γονίδια που επιλέγονται συνδέονται βιολογικά με την ανάπτυξη καρκίνου.

2.3.2 Περιγραφή του προβλήματος ταξινόμησης

Ως είσοδος του συστήματος θεωρείται ένα διάνυσμα, που αποτελείται από δείγματα τα οποία σχετίζονται με n συντελεστές, που αποτελούν τα χαρακτηριστικά του δείγματος. Εδώ, τα χαρακτηριστικά μοντελοποιούν τους συντελεστές των γονιδιακών εκφράσεων και τα δείγματα μοντελοποιούν τους ασθενείς, ενώ καλείται ως F ο δειγματοχώρος n-διαστάσεων.

Περιορίζοντας το πρόβλημα της σχεδίασης σε ταξινόμηση δύο κλάσεων, οι κλάσεις αναγνωρίζονται από τα σύμβολα (+) και (-) αντίστοιχα. Επίσης θεωρείται ότι το σύνολο είναι γραμμικά διαχωριζόμενο, δηλαδή μία συνάρτηση ευθείας μπορεί να διαχωρίσει το σύνολο χωρίς λάθη. Δίνεται ένα σύνολο εκπαίδευσης δειγμάτων $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots \mathbf{x}_k, \ldots \mathbf{x}_l\}$ με γνωστές τιμές ταξινόμησης $\{y_1, y_2, y_3, \ldots y_k, \ldots y_l\}$, όπου y_k ανήκει στο σύνολο {-1,+1}. Τα δείγματα εκπαίδευσης χρησιμοποιούνται για να παράγουν μία συνάρτηση απόφασης $D(\mathbf{x})$, έτσι ώστε τα νέα δείγματα να ταξινομούνται με βάση το πρόσημο της συνάρτησης απόφασης :

$D(\mathbf{x}) > 0 \Rightarrow \mathbf{x} \in$ class $(+)$

$D(\mathbf{x}) < 0 \Rightarrow \mathbf{x} \in$ class $(-)$

$D(\mathbf{x}) = 0$, όριο απόφασης

Οι γραμμικές συναρτήσεις απόφασης (ή συναρτήσεις διαχωρισμού) είναι αθροίσματα των βαρών των δειγμάτων εκπαίδευσης μαζί με την κλίση, και έχουν τη μορφή:

$$D(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + \mathbf{b},$$

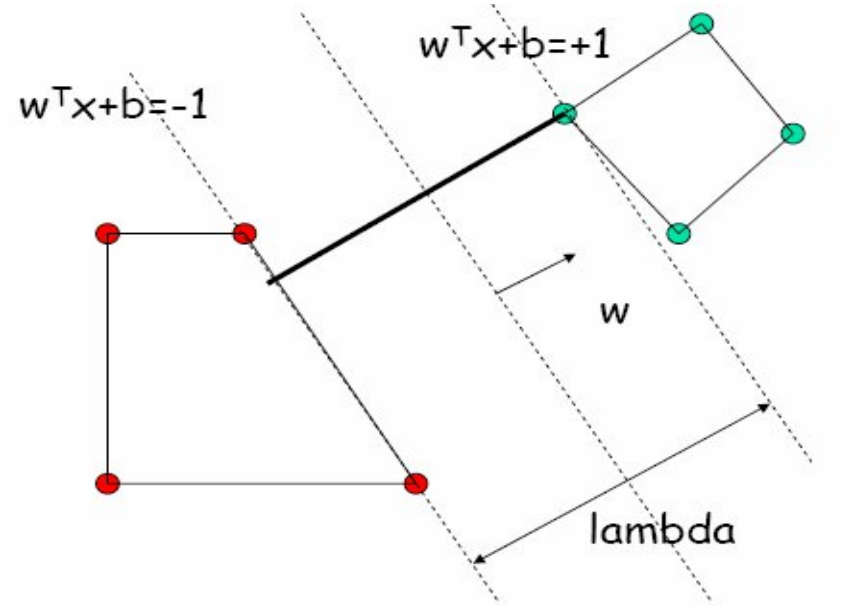
όπου w είναι το διάνυσμα βάρους και b είναι η τιμή της κλίσης.

Ένα χαρακτηριστικό πρόβλημα ταξινόμησης είναι η μείωση της διάστασης n του δειγματοχώρου F, χωρίς να εμφανιστεί το πρόβλημα του "overfitting" (όπου παρουσιάζονται στις κλάσεις της ταξινόμησης σφάλματα διαχωρισμού). Το πρόβλημα αυτό γίνεται φανερό όταν ο αριθμός n των χαρακτηριστικών είναι μεγάλος (στην συγκεκριμένη περίπτωση μερικές χιλιάδες γονίδια) και ο αριθμός των l δειγμάτων εκπαίδευσης είναι συγκριτικά μικρός (στην περίπτωσή μας μερικές δεκάδες ασθενείς). Στη συγκεκριμένη εφαρμογή ερευνήθηκαν μέθοδοι που εξοντώνουν ορισμένα χαρακτηριστικά από την αρχική λίστα και διατηρούν ένα ελάχιστο υποσύνολο χαρακτηριστικών που αποφέρει βέλτιστη απόδοση ταξινόμησης.

2.3.3 Η μέθοδος SVM-RFE

Κλασικές μέθοδοι επιλογής γονιδίων επιλέγουν χαρακτηριστικά που ατομικά ταξινομούν βέλτιστα τα δεδομένα εκπαίδευσης, και συχνά ως κριτήριο αξιολόγησης χρησιμοποιούνται συντελεστές αυτοσυσχέτισης. Ωστόσο, αν και εξοντώνουν αρκετά άχρηστα γονίδια, λόγω των συγκεκριμένων κριτηρίων ταξινόμησης, μπορεί να συμπεριληφθεί αρκετές φορές το ίδιο αντίγραφο γονιδίου, ενώ συμπαγή σύνολα γονιδίων με ισχυρή διαχωριστική ισχύ ενδεχομένως εξοντώνονται, καθώς ατομικά έχουν ασθενή διαχωριστική ισχύ.

Μια πιθανή χρήση της αξιολόγησης των χαρακτηριστικών είναι η σχεδίαση ενός class predictor βασισμένο σε προεπιλεγμένο σύνολο



Εικόνα 2.3: Ο SVMδημιουργεί μία γραμμή διαχωρισμού που απέχει το μέγιστο και από τις δύο κλάσεις.

γονιδίων. Ωστόσο, αυτή η μέθοδος, επειδή εξετάζει κάθε γονίδιο ξεχωριστά, απέχει πολύ από τη βέλτιστη προσέγγιση όταν απαλείφονται πολλά χαρακτηριστικά μαζί, κάτι που είναι απαραίτητο για να ληφθεί σε εύλογο χρονικό διάστημα ένα μικρό υποσύνολο χαρακτηριστικών. Αυτό το πρόβλημα αντιμετωπίζεται με την εφαρμογή του επαναληπτικού αλγόριθμου Recursive Feature Elimination (RFE):

1. Train the classifier (optimize the weights).
2. Compute the ranking criterion for all features ((wi) ^{2}).
3. Remove the feature with smallest ranking criterion.

Η πολλαπλή απαλοιφή χαρακτηριστικών την ίδια στιγμή βελτιώνει το χρόνο εκτέλεσης της εφαρμογής με κόστος μία πιθανή υποβάθμιση της απόδοσης ταξινόμησης. Πρέπει να σημειωθεί ότι ο RFE δεν έχει επίδραση στις μεθόδους αυτοσυσχέτισης, αφού το κριτήριο αξιολόγησης υπολογίζεται με πληροφορία ενός χαρακτηριστικού.

Για τον έλεγχο της χρήσης των βαρών για την παραγωγή ενός κριτηρίου χαρακτηριστικών, χρησιμοποιήθηκε μία τεχνική ταξινόμησης που ονομάζεται γραμμικά Support Vector Machines (SVMs). Σε περίπτωση συνόλων δεδομένων εκπαίδευσης που μπορούν να διαχωριστούν γραμμικά, όπως είναι αυτή που εξετάζεται, ο γραμμικός SVM προσφέρει μέγιστο περιθώριο ταξινόμησης. Αυτό σημαίνει ότι το όριο απόφασης (μία ευθεία γραμμή για διαχωρισμό δύο κλάσεων), είναι τοποθετημένο έτσι ώστε να απέχει τη μέγιστη δυνατή απόσταση και από τις δύο πλευρές. Τα παραδείγματα που βρίσκονται πιο κοντά στο όριο απόφασης και εφάπτονται του περιθωρίου, ονομάζονται Support Vectors. Τα χαρακτηριστικά και η απόδοση του αλγορίθμου βασίζονται στην ύπαρξη τέτοιων διανυσμάτων.

Ο συνδυασμένος αλγόριθμος SVM-RFE είναι μία εφαρμογή του RFE που χρησιμοποιεί το πλάτος των βαρών ως κριτήριο αξιολόγησης. Παρακάτω παρατίθεται μία γενική εφαρμογή του γραμμικού αλγόριθμου, με χρήση της εκπαίδευσης SVM.

1. Let m be the initial number of features.
2. While $(m \geq 0)$
3. Estimate the direction vector $\boldsymbol{w}$ of the separating hyperplane using linear SVM.
4. Rank features according to the components of $|\boldsymbol{w}|$.
5. Remove the feature with the smallest weight in absolute value $(m \leftarrow m-1)$. More than one features can be removed in each iteration.
6. Estimate classification accuracy of the m surviving features using a linear SVM classifier.
7. End While
8. Output as marker genes the set of surviving features achieving maximum accuracy performance.

Εικόνα 2.4: Αλγόριθμος SVM-RFE

3

ΣΧΕΤΙΚΗ ΕΡΓΑΣΙΑ

3.1 Εισαγωγή

Στο κεφάλαιο αυτό θα γίνει μια αναφορά σε σχετικές εργασίες ή εφαρμογές που έχουν παρόμοιο θέμα με την παρούσα διπλωματική και έχουν αναπτυχθεί και παρουσιαστεί από άλλους ερευνητές.

3.2 Σχετικές εφαρμογές πολλαπλασιασμού πινάκων

Όσον αφορά το θέμα του πολλαπλασιασμού πινάκων σε επίπεδο αναδιατασσόμενης λογικής, είναι προφανές ότι εξαιτίας της ευρείας χρήσης αυτής της πράξης της Γραμμικής Άλγεβρας, η σχετική βιβλιογραφία είναι αρκετά ευρεία. Υπάρχουν υλοποιήσεις που επικεντρώνουν στον πολλαπλασιασμό πινάκων αριθμών συγκεκριμένου τύπου και συγκεκριμένου μεγέθους (π.χ. “64-bit Floating-Point FPGA Matrix Multiplication, by Yong Dou, S.Vassiliadis, G.K. Kuzmanov, G.N. Gaydadjiev”), και υλοποιήσεις που προσπαθούν να βελτιστοποιήσουν τον αλγόριθμο του πολλαπλασιασμού πινάκων, με τη χρήση μεθόδων, όπως είναι η αυτή των Bagh-Wooley, που

εκμεταλλεύονται την παραλληλία που προσφέρει η αναδιατασσόμενη λογική (π.χ. “An FPGA Based Parameterisable System for Matrix Product Implementation, by A.Amira and F.Bensaall” και “Design and Implementation of a High Performance Matrix Multiplier Core for Xilinx Virtex FPGAs by S.Belacemi, K.Benkrid, D.Crookes, A.Benkrid”). Στην παρούσα ενότητα θα παρουσιάσουμε δύο υλοποιήσεις οι οποίες σχετίζονται αρκετά με τη δική μας εργασία, καθώς η τελική μας υλοποίηση διαθέτει στοιχεία από αυτές τις δύο. Στην πρώτη αναλύεται ο αλγόριθμος τμηματισμού των πινάκων με αριθμούς σταθερής υποδιαστολής, ενώ στη δεύτερη παρουσιάζεται μία μικρή σχεδίαση σε πλακέτα της εταιρείας Xilinx, που χρησιμοποιεί τις ip cores που προσφέρει η πλακέτα όσον αφορά τις Block Rams και τα DSPs.

3.2.1 Επιτάχυνση πολλαπλασιασμού πινάκων σε Xilinx FPGA

Ο πρώτος διαγωνισμός MEMOCODE, που έγινε το έτος 2007, έθεσε το πρόβλημα της βελτιστοποίησης του πολλαπλασιασμού τετραγωνικών πινάκων αριθμών σταθερής υποδιαστολής, με χρήση του αλγορίθμου τμηματοποίησης των πινάκων και υλοποίηση ενός ενσωματωμένου συστήματος στην Xilinx Virtex IIPro 30. Η ομάδα των N. Dave, K. Fleming, M.King, M.Pellauer, M.Vijayaraghaven, από το Massachusetts Institute of Technology πρότεινε μία ενδιαφέρουσα λύση για το παραπάνω πρόβλημα.

Καταρχήν, σε αντίθεση με τον κλασικό αλγόριθμο πολλαπλασιασμού πινάκων που παρουσιάζεται παρακάτω:

```
for (j = 0; j < N; j++)
  for (i = 0; i < N; i++)
    for (k = 0; k < N; k++)
```

C[i][j] + = A[i][k]*B[k][j] ;

ο οποίος εμφανίζει φτωχή χωρική τοπικότητα, η δημιουργία τμημάτων (blocks) στον πολλαπλασιαστέο και τον πολλαπλασιαστή, αν και δε μειώνει τον αριθμό τον προσθέσεων και των πολλαπλασιασμών (παραμένουν N^3), αυξάνει τη χωρική τοπικότητα, μειώνοντας τις συνεχείς μετακινήσεις δεδομένων από τον αργό μεγάλο αποθηκευτικό χώρο των δεδομένων και αυξάνοντας τις μετακινήσεις από την πιο γρήγορη μικρή μνήμη. Έτσι, υποθέτοντας n παράλληλους πολλαπλασιασμούς, ο αλγόριθμος πολλαπλασιασμού γίνεται:

```
for(i=0; i < N; i++)
  for(j=0; j < N; j++)
    for(k = 0; k = N; k+=n)//cycle
      for(z = 0; z < n; z++)
        c[i][j]+=a[i][k+z]*b[k+z][j];
```

Σε αυτή τη μορφή, όμως, σε κάθε υπολογιστικό βήμα απαιτείται να προστεθούν τα n παραγόμενα γινόμενα, που δημιουργεί «δέντρο» πρόσθεσης βάθους $\log_2(n)$, που μοιραία περιορίζει το μέγεθος του n. Έτσι, αν ακολουθηθεί ο επόμενος αλγόριθμος:

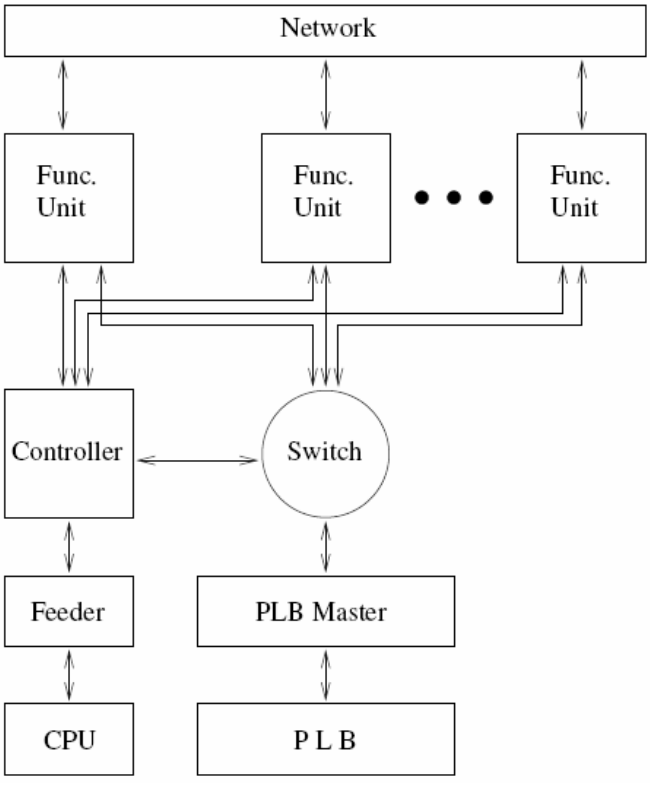
```
for(i=0; i < N; i++)
  for(j=0; j < N; j++)
    for(k = 0; k = N; k+=n) // cycle
      for(z = 0; z < n; z++)
         c[k+z][j]+=a[i][j]*b[j][k+z];
```

αντί να δημιουργείται ένα δέντρο πρόσθεσης, εκτελούνται n παράλληλες προσθέσεις, οδηγώντας σε μικρότερο critical path.

Πέρα από το καθαρά αλγοριθμικό κομμάτι, η υλοποίηση που προτάθηκε παρουσιάζεται στην εικόνα 3.1. Τα κύρια κομμάτια της σχεδίασης είναι:

- Power PC: Ο επεξεργαστής διευθύνει τον υπολογισμό. Επικοινωνεί με το Fedder μέσω δύο στοιβών.

- Feeder: Αναλαμβάνει τη μεταφορά των εντολών από τον Power PC στον controller, και ενημερώνει τον επεξεργαστή όταν τερματίζει ο υπολογισμός.
- Controller: Ο controller αποκωδικοποιεί τις εντολές του επεξεργαστή και ενημερώνει ασύγχρονα την κατάλληλη μονάδα.
- PLB Master: Επικοινωνεί απευθείας με το Processor-Local Bus (PLB) και εκτελεί το χειρισμό της μνήμης.
- Memory Switch: Δρομολογεί τη μεταφορά της μνήμης μεταξύ των Functional Units και του PLB Master.
- Functional Unit: Εκτελεί τον πολλαπλασιασμό των πινάκων και διατηρεί τα στοιχεία σε τρεις καταχωρητές(για τους A,B και C)
- Functional Unit Network: Επιτρέπει να μεταφερθούν δεδομένα από FU σε FU χωρίς να περάσουν από τη μνήμη.



Εικόνα 3.1 : Block diagram της σχεδίασης

Ο πίνακας 3.1 παρουσιάζει το χρόνο που χρειάζεται για να εκτελέσει πολλαπλασιασμούς πινάκων διαφόρων μεγεθών.

HW Pipelined System - 8 muls, 64×64 block	
64^2	799
128^2	5,122
256^2	45,318
512^2	332,011
1024^2	2,711,073
Pentium 4, Reference Algorithm	
64^2	11,000
128^2	75,000
256^2	608,000
512^2	12,805,000
1024^2	139,000,000

Πίνακας 3.1: Αποτελέσματα απόδοσης του υλοποιημένου συστήματος σε σύγκριση με έναν Pentium4

3.2.2 Πολλαπλασιασμός πινάκων σε FPGAs

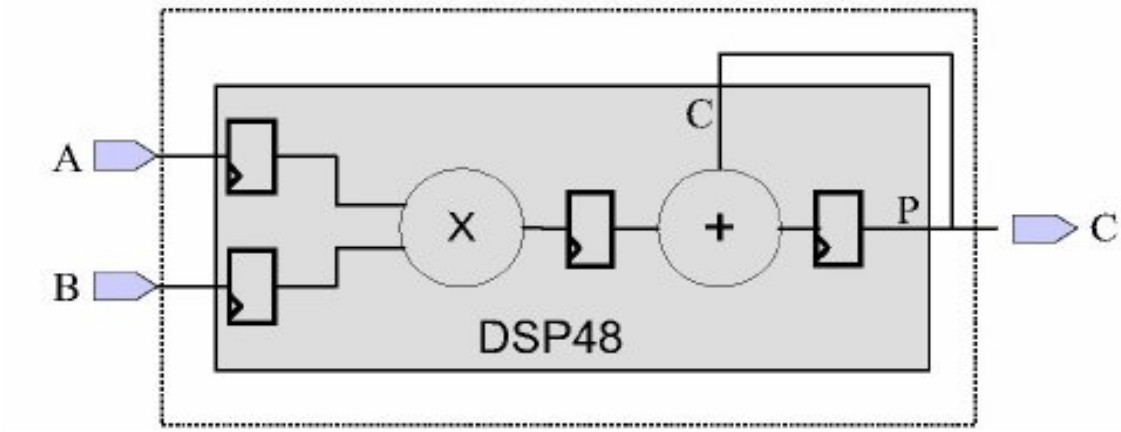
Το paper που δημοσιεύθηκε τον Οκτώβριο του 2007 από τους R.El-Atfy (Mentor Graphics), M.A.Dessouky (Mentor Graphics), και H.El-Ghitani (Misr International University) ασχολείται με τον πολλαπλασιασμό πινάκων σταθερής υποδιαστολής μέσω της Xilinx Virtex 4 device και εστιάζει στην αποτελεσματική χρήση των DSP (Digital Signal Processing) blocks που αυτή διαθέτει.

Το στοιχείο κλειδί για τον αλγόριθμο πολλαπλασιασμού πινάκων είναι η μονάδα MAC (Multiplier-Accumulator). Η ταχύτητα και η αποτελεσματική εκτέλεση του συστήματος εξαρτάται από τον αριθμό και την ταχύτητα των MAC που βρίσκονται στο chip. Η Virtex4 οικογένεια διαθέτει μονάδες DSP48 , όπου κάθε μία αποτελείται από έναν 18x18 bit two's complement πολλαπλασιαστή ακολουθούμενο από έναν 48 bit sign-extended adder/subtracter/accumulator. Σε περίπτωση που τα

στοιχεία που πολλαπλασιάζονται έχουν μεγαλύτερο μέγεθος, για κάθε PE χρησιμοποιούνται περισσότερα DSPs. Ο αριθμός των DSPs που χρησιμοποιούνται είναι το τετράγωνο του εύρους των δεδομένων εισόδου(W_{data}) διαιρεμένο με το εύρος του δεδομένου εισόδου του DSP(W_{DSP}), στρογγυλοποιημένο προς τα πάνω:

$$PE\_DSPs = (\lceil W_{data} / W_{dsp} \rceil)^2$$

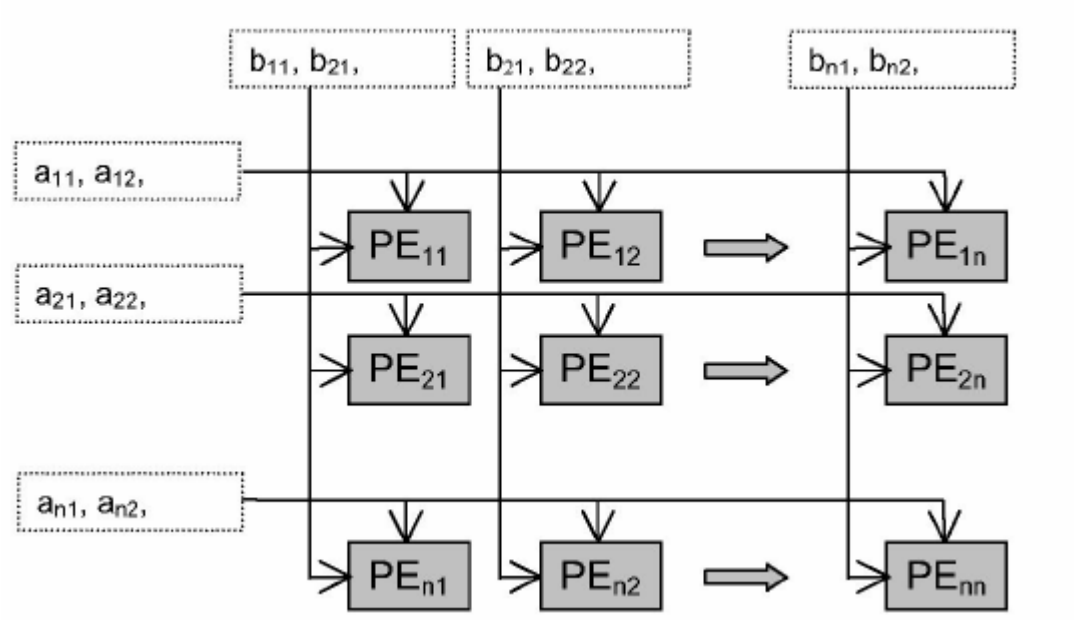
Η αρχιτεκτονική της προτεινόμενης σχεδίασης βασίζεται στην παράλληλη τοποθέτηση των στοιχείων επεξεργασίας (Processing Elements). Η δομή ενός PE παρουσιάζεται στην εικόνα 3.2. Σε κάθε κύκλο ρολογιού διαβάζεται μία είσοδος από κάθε πίνακα. Στον επόμενο κύκλο, οι δύο είσοδοι πολλαπλασιάζονται, και τελικά το γινόμενο προστίθεται στην έξοδο του προηγούμενου αποτελέσματος. Το τελικό αποτέλεσμα του πίνακα C αποθηκεύεται σε μία Block Ram.



Εικόνα 3.2 Δομή ενός στοιχείου επεξεργασίας (PE).

Η διάταξη των PEs οργανώνεται με τον τρόπο που παρουσιάζεται στην εικόνα 3.3. Κάθε Memory Block αποθηκεύει μία γραμμή του πίνακα A ή μία στήλη του πίνακα B. Κάθε PE δέχεται στον ίδιο κύκλο ένα στοιχείο από τους A και B, που είναι αποθηκευμένοι σε Block Rams, και

εκτελούνται παράλληλα όλα τα MAC στοιχεία. Έτσι, για πολλαπλασιασμό πινάκων [n,n]x[n,n] , εκτελούνται n^2 υπολογισμοί κάθε κύκλο ρολογιού. Εξαιτίας της ανεξαρτησίας των PEs η αρχιτεκτονική μπορεί να παραμετροποιηθεί και να υλοποιηθεί για οποιοδήποτε μέγεθος, χωρίς να αυξηθεί η συχνότητα του ρολογιού. Η σχεδίαση περιλαμβάνει και πολλαπλασιασμούς μη τετραγωνικών πινάκων καθώς κάθε PE υπολογίζει μόνο ένα στοιχείο του πίνακα C.



Εικόνα 3.3 : Διάταξη των στοιχείων επεξεργασίας

Κάθε memory block του A αποθηκεύει k=(n/m) λέξεις, από κάθε στήλη του A, για κάθε γραμμή του A. Αντίστοιχα, κάθε memory block του B αποθηκεύει k=(n/m) λέξεις, από κάθε γραμμή του B, για κάθε στήλη του B. Αυτό επιτρέπει στον πολλαπλασιασμό πινάκων οποιωνδήποτε μεγεθών. Ο αριθμός των Block Rams που απαιτούνται, αν υποθέσουμε ότι κάθε bram έχει βάθος D και εύρος λέξης W_{BRAM}, ενώ το εύρος των δεδομένων εισόδου είναι W_{data}, ισούται με:

$$\text{Input Block Rams} = 2 * (\lceil W_{data} / W_{BRAM} \rceil) * (n^2/d)$$

Για την αποθήκευση των στοιχείων εξόδου, χρειαζόμαστε μεγαλύτερο εύρος λέξης, καθώς τα γινόμενα έχουν διπλάσιο πλάτος, κι έτσι

απαιτούνται μεγαλύτερες Block Rams. Ο αριθμός των Block Rams που απαιτούνται υπολογίζεται αν από το εύρος των δεδομένων εισόδου (W_{data}) διαιρέσουμε το εύρος λέξης της Block RAM (W_{BRAM}), στρογγυλοποιώντας το προς τα πάνω, το πολλαπλασιάσουμε με το αριθμό των λέξεων του πίνακα C, και το διαιρέσουμε με το βάθος d της Block RAM:

$$\text{MatrixCBlockRams} = (|W_{data}/W_{BRAM}|) * n^2$$

Ως παράδειγμα της σχεδίασης χρησιμοποιείται ένας πολλαπλασιαστής πινάκων 8x8 για στοιχεία των 16 bit σταθερής υποδιαστολής, υλοποιημένος σε μία πλακέτα Xilinx XCV4LX160. Για την εφαρμογή αυτή απαιτούνται 64 PEs όπου κάθε ένα καταλαμβάνει 1 DSP block. Κάθε γραμμή του A και κάθε στήλη του B καταλαμβάνουν 1 BRAM (επομένως απαιτούνται 16 BRAMS για τις εισόδους και άλλες 8 για τον πίνακα C. Η σχεδίαση έχει συχνότητα 400MHz και απαιτεί 668 slices (1% της συνολικής ποσότητας της πλακέτας), 24 BRAMS(8% της συνολικής ποσότητας) και 64 DSPs(66% της συνολικής ποσότητας). Η συγκεκριμένη εφαρμογή συγκρίνεται ως προς την ταχύτητα και το μέγεθος με άλλες παλαιότερες σχετικές εφαρμογές και αποδεικνύεται ότι εμφανίζει καλύτερα αποτελέσματα τόσο ως προς την χρονική απόδοση αλλά και ως προς τη χωρική απόδοση.

3.3 Σχετικές εφαρμογές δυαδικής αναζήτησης

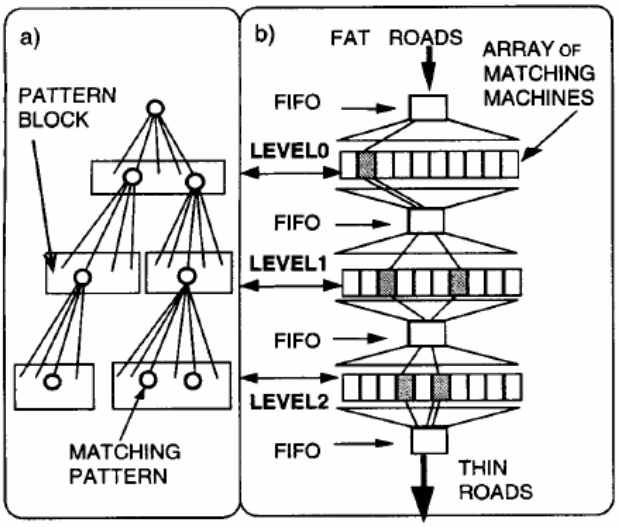
Η αναζήτηση ταξινομημένων και μη στοιχείων σε μεγάλες βάσεις δεδομένων είναι μία συνηθισμένη διαδικασία και αποτελεί βασική λειτουργία πολλών εφαρμογών. Ένα μεγάλο εύρος αλγορίθμων και τεχνικών προσπαθούν να καλύψουν την ανάγκη γρήγορης αναζήτησης. Η λογαριθμική πολυπλοκότητα του αλγορίθμου δυαδικής αναζήτησης σε συνδυασμό με την απλή υλοποίησή του, τον έχει μετατρέψει σε μία από

τις πιο δημοφιλείς μεθόδους αναζήτησης ταξινομημένων στοιχείων. Πολλές εφαρμογές με σκοπό την εκμετάλλευση της παραλληλίας της αναδιατασσόμενης λογικής, χρησιμοποίησαν τον αλγόριθμο αυτό για να μειώσουν ακόμα περισσότερο το κόστος μίας αναζήτησης. Μία εφαρμογή δυαδικής αναζήτησης που προσεγγίζει τη δική μας υλοποίηση παρουσιάζεται συνοπτικά παρακάτω.

3.3.1 Επεξεργαστής δενδρικής αναζήτησης εντοπισμού διαδρομής

Η εφαρμογή αυτή (“The tree search processor for real-time track finding”) παρουσιάζει μία παράλληλη και ομόχειρη υλοποίηση δυαδικής αναζήτησης για το πρόβλημα της αναζήτησης συντομότερου μονοπατιού, βασισμένο σε μεγάλες τράπεζες προϋπολογιζόμενων συνδυασμών σημείων τροχιάς. Για την όσο το δυνατόν πιο γρήγορη εύρεση μονοπατιού η μονάδα χρησιμοποιεί πολλές βαθμίδες παράλληλης δυαδικής αναζήτησης (κάθε μία από τις οποίες δοκιμάζει διαφορετικό μονοπάτι) , ενώ κάθε βαθμίδα αποτελείται από pipelined βήματα.

Το πρόβλημα της εύρεσης συντομότερου μονοπατιού περιορίζεται σε αναζητήσεις στοιχείων σε κάθε τράπεζα. Στην εικόνα 3.4.a παρουσιάζεται η μορφή της δενδρικής δομής μιας τράπεζας. Μεγαλύτερο βάθος αντιστοιχεί σε μεγαλύτερη χωρική ανάλυση της διαδρομής. Ο αλγόριθμος αναζήτησης σε δυαδικό δέντρο υλοποιείται εύκολα σε παράλληλη αρχιτεκτονική, γιατί διαφορετικές διαδρομές μπορούν να εξετάζονται ξεχωριστά. Η δυαδική αναζήτηση γίνεται ομόχειρα, εφόσον όλα τα τεστ ενός επιπέδου εκτελούνται μετά από τα τεστ των προηγούμενων επιπέδων. Στο σχήμα 3.4.b, παρουσιάζονται οι ομόχειρες μηχανές δυαδικής αναζήτησης, χωρισμένες από μνήμες FIFO.



Εικόνα 3.4: (a) Ιεραρχική οργάνωση μιας δενδρικής δομής . (b) Παράλληλη αρχιτεκτονική ομόχειρων μηχανών που αντιστοιχούν στα επίπεδα δέντρου. Fat roads είναι οι μεγάλες μη βελτιστοποιημένες διαδρομές, και Thin roads είναι τα αποτελέσματα του αλγόριθμου.

4

ΥΛΟΠΟΙΗΣΗ ΠΟΛΛΑΠΛΑΣΑΙΣΜΟΥ ΠΙΝΑΚΩΝ ΣΕ ΑΝΑΔΙΑΤΑΣΣΟΜΕΝΗ ΛΟΓΙΚΗ

4.1 Εισαγωγή

Όπως έχει αναφερθεί παραπάνω, ο πολλαπλασιασμός πινάκων αποτελεί το πιο αργό υπολογιστικά κομμάτι του συστήματος "CSU Face Identification Evaluation System, version 5.0" και σκοπός μας είναι η βελτιστοποίηση αυτού. Υπενθυμίζουμε ότι κάθε εικόνα μετατρέπεται σε ένα διάνυσμα με διαστάσεις 19500x1. Για το λόγο αυτό απαιτείται συχνά ο πολλαπλασιασμός πινάκων με μεγέθη μεγαλύτερα του ενός εκατομμυρίου στοιχείων (π.χ. συχνά συναντάμε πίνακες της μορφής 19500x100, που δημιουργούνται από το στάδιο της εκπαίδευσης του αλγορίθμου PCA).

Η χρονική υπολογιστική πολυπλοκότητα του κλασικού αλγορίθμου, όπως αυτός παρουσιάζεται στην εικόνα 4.1, για τρεις πίνακες C=AxB, με διαστάσεις [M,R], [M,N], [N,R] αντίστοιχα είναι 2xMxRxN (π.χ. για τετραγωνικούς πίνακες η πολυπλοκότητα είναι $O(n^3)$). Γενικά, αν και η παράλληλη επεξεργασία μειώνει την πολυπλοκότητα υπολογισμού,

αυξάνει τις απαιτήσεις σε bandwidth της μνήμης δεδομένων, και ιδιαίτερα για μεγάλους πίνακες δεν εκμεταλλεύεται σχεδόν καθόλου τη χωρική τοπικότητα.

```
for (i = 0; i < M; i = i + 1)
    for (j = 0; j < R; j = j + 1){
        C[i, j] = 0;
        for (k = 0; k < N; k = k + 1)
            C[i, j] = C[i, j] + A[i, k] * B[k, j];}
```

Εικόνα 4.1: Τυπικός αλγόριθμος πολλαπλασιασμού πινάκων

Για το λόγο αυτό και εξαιτίας των συγκεκριμένων χαρακτηριστικών των πινάκων που επεξεργαζόμαστε στην εφαρμογή, κρίνεται αναγκαία η εφαρμογή του πολλαπλασιασμού τμηματοποιημένων πινάκων, Παρακάτω θα περιγράψουμε τον τρόπο που εφαρμόσαμε αυτή τη μέθοδο στη σχεδίασή μας και θα αναλύσουμε τη δομή με την οποία επιλέξαμε να την υλοποιήσουμε.

4.2 Γενική περιγραφή του αλγορίθμου του παράλληλου πολλαπλασιασμού πινάκων

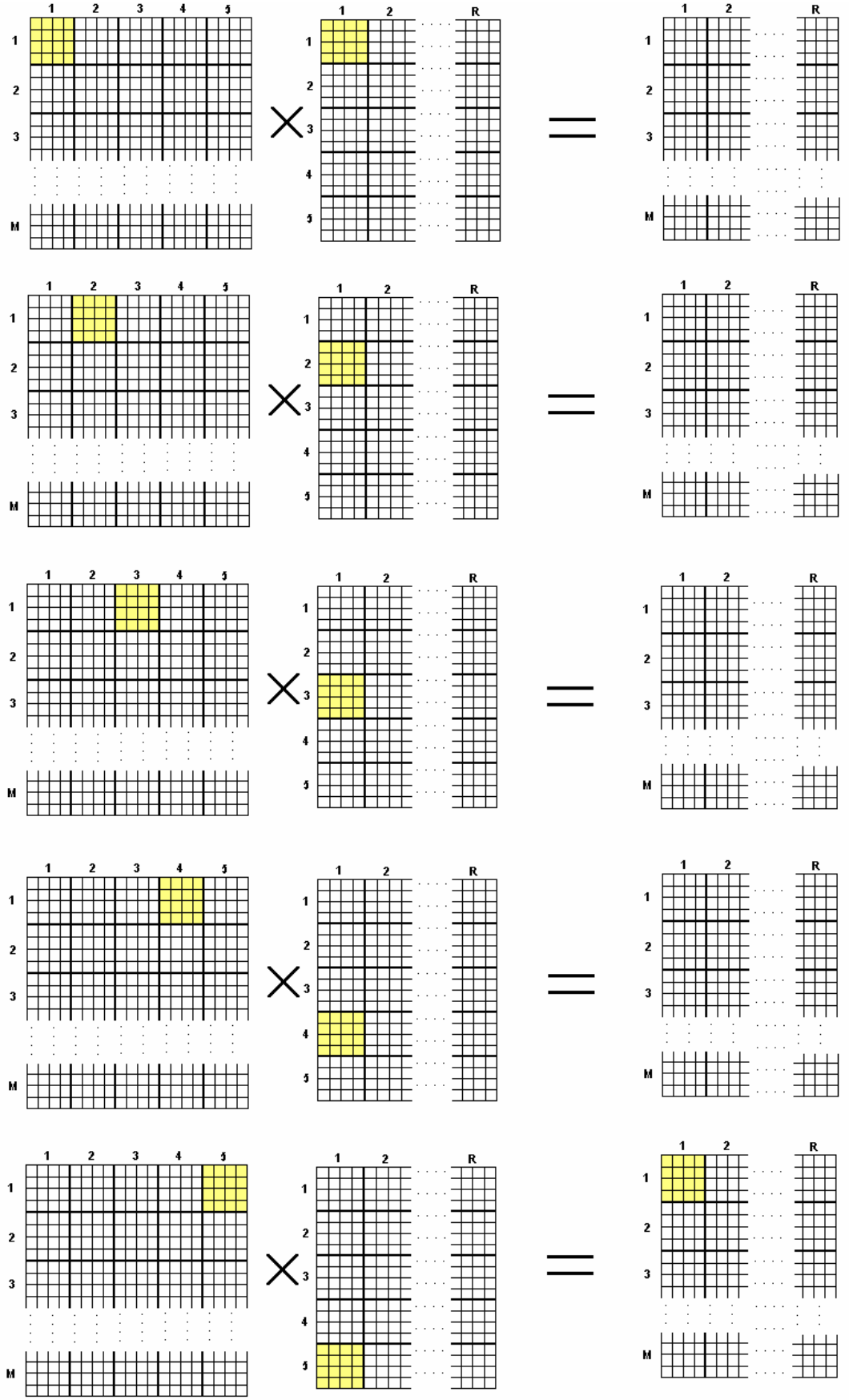
Ο γενικός αλγόριθμος που εφαρμόσαμε για τον πολλαπλασιασμό πινάκων, (αν θεωρήσουμε ότι ο πίνακας A έχει διαστάσεις [M,N], και ο B [N,R], παράγοντας έναν πίνακα C μεγέθους [M,R], αποτελούμενο από υποπίνακες διαστάσεων [S,S]) είναι:

```
for (i=0; i<M/S; i=i+1)
    for (j=0; j<R/S; j=j+1) {
        for (Li=0; Li<S; Li=Li+1)
            for (Lj=0; Lj<S; Lj=Lj+1)
                C [ ixS+Li, jxS+Lj ] =0;
        for (p=0; p<N/S; p=p+1)
            for (k=0; k<S; k=k+1) {
                for (Li=0; Li<S; Li=Li+1)
                    for (Lj=0; Lj<S; Lj=Lj+1)
                        C [ ixS+Li, jxS+Lj ] =
                        = C [ ixS+Li, jxS+Lj ] +
                        + A [ ixS+Li, pxS+k ] x B [ pxS+k, jxS+Lj ]; }}
```

Εικόνα 4.2: Αλγόριθμος πολλαπλασιασμού τμηματοποιημένων πινάκων, όπως εφαρμόστηκε στη σχεδίασή μας

Οι δύο πρώτοι βρόχοι μας εξασφαλίζουν ότι κάθε ν-οστός υποπίνακας, διαστάσεων SxS, όλων των υπο-γραμμών του πίνακα Α, θα πολλαπλασιαστεί με κάθε ν-οστό υποπίνακα, διαστάσεων SxS, όλων των υπο-στηλών του πίνακα Β. Στη συνέχεια με ένα διπλό loop αρχικοποιούμε τον αντίστοιχο υποπίνακα του C, και ακολούθως με άλλα δύο loop καθορίζουμε τη στήλη του στοιχείου του Α και τη γραμμή του στοιχείου του Β που θα πολλαπλασιαστούν. Τελικά, δηλώνοντας με ένα ακόμη διπλό βρόχο τη γραμμή του στοιχείου του Α και τη στήλη του στοιχείου του Β, πολλαπλασιάζουμε τα στοιχεία.

Στην υλοποίησή μας εκμεταλλευόμαστε την λογική των FPGA, εκτελώντας παράλληλα τα δύο τελευταία loop. Έτσι, αν η διάσταση κάθε υποπίνακα είναι S=4, τότε θα εκτελούνται παράλληλα 16 πολλαπλασιασμοί, και μετά από k_max=4 συνεχόμενους πολλαπλασιασμούς οι δύο υποπίνακες θα έχουν πολλαπλασιαστεί. Όπως φαίνεται και στην εικόνα 4.3, για να παραχθεί ένας υποπίνακας του C, πρέπει όλοι οι υποπίνακες μίας γραμμής του Α να πολλαπλασιαστούν με τους αντίστοιχους μίας στήλης του Β. Στη δική μας υλοποίηση έχουμε θέσει S=16, έτσι ώστε να εκτελούνται παράλληλα 256 πολλαπλασιασμοί.



Εικόνα 4.3: Σταδιακή παρουσίαση παραγωγής των πρώτων στοιχείων ενός πίνακα [SxM,SxR], κατά των πολλαπλασιασμό τμηματοποιημένων πινάκων [SxM,Sx5] και [Sx5,SxN],όπου εδώ S=4.

4.3 Γενική περιγραφή της σχεδίασης σε αναδιατασσόμενη λογική

Στην παρούσα ενότητα παρουσιάζεται η γενική λογική με την οποία επιλέξαμε να υλοποιηθεί την εφαρμογή. Οι κύριες βαθμίδες της σχεδίασης (εικόνα 4.4), οι οποίες είναι pipelined, είναι:

- Η βαθμίδα ανάγνωσης από τις SRAMS A και B, όπου είναι αποθηκευμένα όλα στοιχεία των πινάκων. Σε κάθε περίοδο ανάγνωσης στέλνει σε block RAMs 4096 στοιχεία από κάθε πίνακα, ανά υποπίνακες [64,64].
- Η βαθμίδα εγγραφής στις block RAMs A και B (που κάθε μία μονάδα περιέχει δύο αντίγραφα μνημών μεγέθους [64,64] στοιχείων) στοιχείων που διαβάζονται από τις εξωτερικές μνήμες, και ανάγνωσης από τις BRAMs υποπινάκων [16,16] που μεταφέρονται σε buffers ανάλογου μεγέθους. Ενώ το ένα αντίγραφο μιας BRAM[64,64] διαβάζει 4096 καινούργια δεδομένα από την εξωτερική μνήμη, το άλλο στέλνει σταδιακά στους buffers υποπίνακες [16,16], μέχρι να ολοκληρωθεί ο πολλαπλασιασμός των υποπινάκων A[64,64]xB[64,64].
- Η βαθμίδα πολλαπλασιασμού των buffers [16,16]. Γίνονται 256 παράλληλοι πολλαπλασιασμοί.
- Η βαθμίδα εγγραφής των αποτελεσμάτων σε Block RAM C. Διαθέτει δύο επίπεδα (κάθε ένα αποτελούμενο από μνήμη μεγέθους 4096 στοιχείων) για να μην υπάρχει επιπλέον καθυστέρηση για την εγγραφή των τελικών δεδομένων στην SRAM C. Ενώ η πρώτη βαθμίδα αποθηκεύει τα προσωρινά αποτελέσματα που παράγονται από τη μονάδα πολλαπλασιασμού,

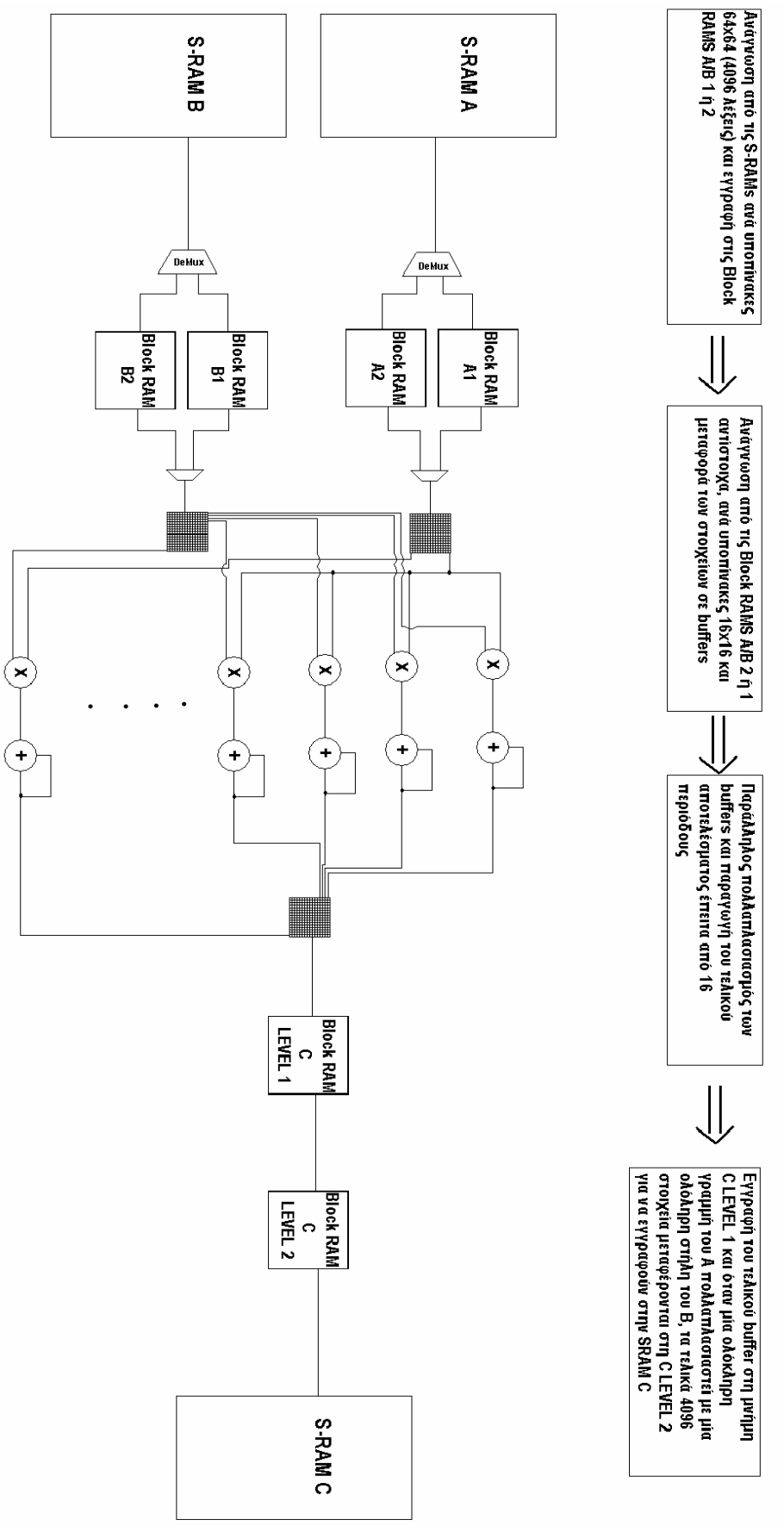
η δεύτερη μεταφέρει προηγούμενα αποτελέσματα στην αργή SRAM.

- Η βαθμίδα εγγραφής στην SRAM C. Σε κάθε περίοδο εγγραφής στέλνονται από το δεύτερο επίπεδο της βαθμίδας της Block RAM C 4096 στοιχεία, σε μορφή πίνακα [64,64].

Οι μονάδες α) ανάγνωσης από τις SRAMs, β) πολλαπλασιασμού πινάκων μεγέθους [64x64] (που περιλαμβάνει τις υπομονάδες i) διαχειρισμού των Block RAMs ανάγνωσης, ii) πολλαπλασιασμού των buffers, και iii) διαχειρισμού Block RAMs εγγραφής), και γ) εγγραφής σε SRAMs είναι πλήρως pipelined. Επίσης, υπάρχει και ένα δεύτερο επίπεδο pipeline, μεταξύ των τριών υπομονάδων του πολλαπλασιασμού υποπινάκων μεγέθους [64x64].

4.4 Περιγραφή των βαθμίδων ανάγνωσης και εγγραφής εξωτερικών μνημών

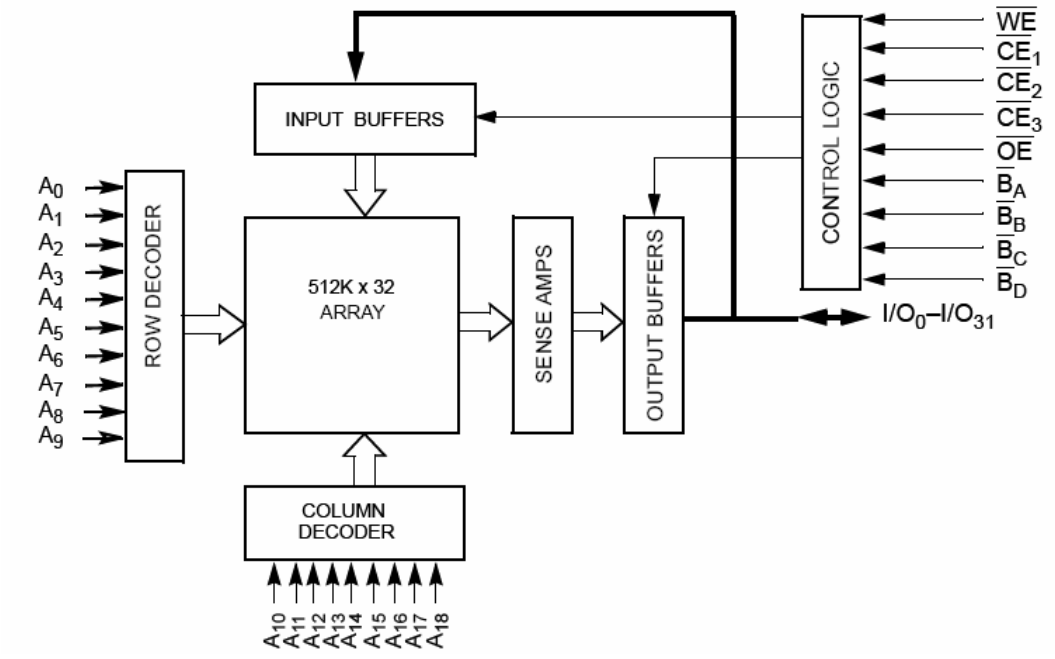
Όπως έχει ειπωθεί στην προηγούμενη ενότητα οι πίνακες A και B, που πρέπει να πολλαπλασιαστούν, και ο πίνακας C, που διατηρεί το τελικό αποτέλεσμα, αποθηκεύονται κάθε ένας σε ξεχωριστή ομάδα από SRAMs. Ενώ σε μία άμεση υλοποίηση θα αναμενόταν κάθε πίνακας να αποθηκεύεται μόνο σε μία εξωτερική μνήμη, εξαιτίας της μεγάλης καθυστέρησης της περιόδου ανάγνωσης και εγγραφής των μνημών (χρόνος ανάγνωσης/εγγραφής:10ns), χρησιμοποιήσαμε αρκετές εξωτερικές μνήμες για να διατηρήσουμε τα δεδομένα κάθε πίνακα. Έτσι, δίνεται η δυνατότητα πρόσβασης πολλαπλών δεδομένων σε έναν κύκλο ανάγνωσης/εγγραφής. Όπως θα αναλυθεί και παρακάτω, δοκιμάσαμε τη μονάδα μας με χρήση 1,2,4,8 και 16 διαφορετικών εξωτερικών μνημών για κάθε πίνακα.



Εικόνα 4.4: Εποπτική παρουσίαση του datapath της σχεδίασης

4.4.1 Δομή εξωτερικής μνήμης SRAM

Για λόγους που θα αναφερθούν στη συνέχεια, επιλέξαμε ως κύρια σχεδίαση αυτή που αποτελείται από 8 εξωτερικές μνήμες για κάθε πίνακα. Το μέγεθος κάθε μνήμης καθορίστηκε από τη διάσταση του μεγαλύτερου πίνακα που χρησιμοποιείται στην εφαρμογή, που αυτή είναι [173,19500]. Επομένως μία ομάδα εξωτερικών μνημών ενός πίνακα πρέπει συνολικά να έχει διαστάσεις 4M λέξεων (όπου κάθε λέξη έχει μέγεθος 32 bits). Για μοντελοποίηση με 8 εξωτερικές μνήμες κάθε μνήμη πρέπει να έχει διαστάσεις 512Kx32. Το μοντέλο της εξωτερικής μνήμης που χρησιμοποιήθηκε είναι το CY7C1062AV33-10BGC που είναι οργανωμένο σε 524.288 λέξεις των 32 bits. Η δομή της μνήμης παρουσιάζεται στην εικόνα 4.5



Εικόνα 4.5: Block Diagram της SRAM που χρησιμοποιήθηκε

Η εγγραφή στη μονάδα επιτυγχάνεται κάνοντας τις εισόδους Chip Enable (CE_1,CE_2,CE_3) και Write Enable (WE) LOW. Εάν το Byte Enable A (B_A)

είναι LOW, τότε δεδομένα από τα pins I/O_0 μέχρι I/O_7 γράφονται στην καθοριζόμενη από την είσοδο Address(A_0 μέχρι A_{18}) διεύθυνση. Εάν το Byte Enable B (B_B) είναι LOW, τότε δεδομένα από τα pins I/O_8 μέχρι I/O_{15} γράφονται στην καθοριζόμενη από την είσοδο Address(A_0 μέχρι A_{18}) διεύθυνση. Παρομοίως, τα B_C και B_D ελέγχουν τα pins I/O_{16} μέχρι I/O_{23} και I/O_{24} μέχρι I/O_{31}, αντιστοίχως.

Η ανάγνωση στη μονάδα επιτυγχάνεται κάνοντας τις εισόδους Chip Enable (CE_1,CE_2,CE_3) και Output Enable (OE) LOW και to Write Enable (WE) HIGH. Εάν το Byte Enable A (B_A) είναι LOW, τότε δεδομένα από την καθοριζόμενη από την είσοδο Address (A_0 μέχρι A_{18}) διεύθυνση εμφανίζονται στα pins I/O_0 μέχρι I/O_7. Εάν το Byte Enable B (B_B) είναι LOW, τότε δεδομένα από την καθοριζόμενη από την είσοδο Address (A_0 μέχρι A_{18}) διεύθυνση εμφανίζονται στα pins I/O_8 μέχρι I/O_{15}. Παρομοίως, τα B_C και B_D ελέγχουν τα pins I/O_{16} μέχρι I/O_{23} και I/O_{24} μέχρι I/O_{31}, αντιστοίχως.

Τα pins I/O_0 έως I/O_{31} βρίσκονται σε κατάσταση υψηλής σύνθετης αντίστασης όταν η μονάδα είναι απενεργοποιημένη (CE'_1 ή CE'_2 ή CE'_3 HIGH), εάν οι έξοδοι είναι απενεργοποιημένοι (OE' HIGH),τα Byte Selects είναι απενεργοποιημένα (B'_{A-D} HIGH), ή κατά τη διάρκεια μιας εγγραφής (CE'_1, CE'_2 και CE'_3 LOW, και WE' LOW). Τα παραπάνω παρουσιάζονται στην εικόνα 4.6.

Τέλος αναφέρουμε, ότι εξαιτίας της αργής περιόδου πρόσβασης της SRAM σε σχέση με το ρολόι του συστήματος, μία ανάγνωση ή εγγραφή της SRAM αντιστοιχεί σε τέσσερεις κύκλους ρολογιού του συστήματος.

$\overline{CE}_1$	$\overline{CE}_2$	$\overline{CE}_3$	$\overline{OE}$	$\overline{WE}$	$\overline{B}_A$	$\overline{B}_B$	$\overline{B}_c$	$\overline{B}_D$	I/O_0–I/O_7	I/O_8–I/O_{15}	I/O_{16}–I/O_{23}	I/O_{24}–I/O_{31}	Mode	Power
H	X	X	X	X	X	X	X	X	High-Z	High-Z	High-Z	High-Z	Power Down	(I_{SB})
X	H	X	X	X	X	X	X	X	High-Z	High-Z	High-Z	High-Z	Power Down	(I_{SB})
X	X	H	X	X	X	X	X	X	High-Z	High-Z	High-Z	High-Z	Power Down	(I_{SB})
L	L	L	L	H	L	L	L	L	Data Out	Data Out	Data Out	Data Out	Read All Bits	(I_{CC})
L	L	L	L	H	L	H	H	H	Data Out	High-Z	High-Z	High-Z	Read Byte A Bits Only	(I_{CC})
L	L	L	L	H	H	L	H	H	High-Z	Data Out	High-Z	High-Z	Read Byte B Bits Only	(I_{CC})
L	L	L	L	H	H	H	L	H	High-Z	High-Z	Data Out	High-Z	Read Byte C Bits Only	(I_{CC})
L	L	L	L	H	H	H	H	L	High-Z	High-Z	High-Z	Data Out	Read Byte D Bits Only	(I_{CC})
L	L	L	X	L	L	L	L	L	Data In	Data In	Data In	Data In	Write All Bits	(I_{CC})
L	L	L	X	L	L	H	H	H	Data In	High-Z	High-Z	High-Z	Write Byte A Bits Only	(I_{CC})
L	L	L	X	L	H	L	H	H	High-Z	Data In	High-Z	High-Z	Write Byte B Bits Only	(I_{CC})
L	L	L	X	L	H	H	L	H	High-Z	High-Z	Data In	High-Z	Write Byte C Bits Only	(I_{CC})
L	L	L	X	L	H	H	H	L	High-Z	High-Z	High-Z	Data In	Write Byte D Bits Only	(I_{CC})
L	L	L	H	H	X	X	X	X	High-Z	High-Z	High-Z	High-Z	Selected, Outputs Disabled	(I_{CC})

Εικόνα 4.6: Πίνακας αληθείας της SRAM

4.4.2 Μονάδα ανάγνωσης δεδομένων από SRAM

Η μονάδα ανάγνωσης δεδομένων από SRAM αποτελεί το πρώτο από τα τρία στάδια της ομοχειρίας πρώτου επιπέδου του συστήματος. Τα άλλα δύο είναι η μονάδα πολλαπλασιασμού πινάκων [64,64] και η μονάδα εγγραφής δεδομένων σε SRAM. Μεταφέρει στις block RAM της FPGA υποπίνακες από κάθε πίνακα, διαστάσεων [64,64]. Ο τρόπος ανάγνωσης αυτών των 4096 (64x64) στοιχείων, καθορίζεται από μία FSM. Επειδή ο κύκλος ρολογιού ανάγνωσης/εγγραφής των SRAMS που χρησιμοποιούμε είναι αρκετά μεγαλύτερος από το ρολόι του συστήματος, αν η μονάδα διάβαζε μόνο ένα στοιχείο σε κάθε κύκλο ανάγνωσης, για να ολοκληρωθούν οι 4096 αναγνώσεις, η περίοδος κάθε βαθμίδας ομοχειρίας θα αυξανόταν πολύ, με αποτέλεσμα τη γενικότερη καθυστέρηση του συστήματος. Για να αποφευχθεί αυτή η σημαντική καθυστέρηση, που θα έκανε το σύστημά μας πολλές φορές πιο αργό, επιλέχθηκε η λύση της χρήσης πολλαπλών SRAMs για κάθε πίνακα, έτσι

ώστε να είναι δυνατή η παράλληλη ανάγνωση πολλαπλών δεδομένων σε έναν κύκλο ανάγνωσης.

Αν χρησιμοποιηθεί η υλοποίηση στην οποία κάθε πίνακας φορτώνεται σε μόνο μία εξωτερική μνήμη, τα στοιχεία του πίνακα αποθηκεύονται με προτεραιότητα γραμμής (row majored). Έτσι, για έναν πίνακα [5,3] τα στοιχεία της πρώτης γραμμής είναι αποθηκευμένα στις πρώτες 3 διευθύνσεις της μνήμης, τα στοιχεία της δεύτερης γραμμής στις επόμενες 3 διευθύνσεις κ.ο.κ.. Παρόμοια μέθοδος χρησιμοποιείται και όταν εφαρμόζονται πολλαπλοί πίνακες για έναν πίνακα. Σε αυτή την περίπτωση, όμως, αν διατίθενται y μνήμες για έναν πίνακα [m,n] τότε: τα n στοιχεία της πρώτης γραμμής θα βρίσκονται στις πρώτες n διευθύνσεις της πρώτης μνήμης, τα n στοιχεία της δεύτερης γραμμής θα βρίσκονται στις πρώτες n διευθύνσεις της δεύτερης μνήμης, τα n στοιχεία της y-οστής γραμμής στις πρώτες n διευθύνσεις της y-οστής μνήμης, κτλ.

Η επιλογή του αριθμού των μνημών που τελικά χρησιμοποιήθηκε έγινε με τέτοιον τρόπο, ώστε ο συνολικός χρόνος ανάγνωσης 4096 στοιχείων να είναι παραπλήσιος με το χρόνο εκτέλεσης των υπόλοιπων σταδίων της ομοχειρίας. Όπως αναλύεται και εκτενέστερα στην ενότητα παρουσίασης των αποτελεσμάτων, σχεδόν όσος χρόνος απαιτείται για να διαβαστούν 4096 στοιχεία, εκτελώντας 8 παράλληλες αναγνώσεις από εξωτερικές μνήμες, τόσος χρόνος απαιτείται για να ολοκληρωθεί ο πολλαπλασιασμός ενός πίνακα [64,64]. Η παρατήρηση αυτή αποτελεί το σημαντικότερο λόγο για τον οποίο επιλέχθηκε η χρήση 8 εξωτερικών μνημών για την αποθήκευση των στοιχείων κάθε πίνακα.

Η ανάγνωση δεδομένων από τις μνήμες του πίνακα A έχει πολλές ομοιότητες με την ανάγνωση από τις μνήμες του πίνακα B, δεν είναι

όμως πανομοιότυπη. Οι υποπίνακες του Α διαβάζονται με προτεραιότητα γραμμής ενώ οι υποπίνακες του Β διαβάζονται με προτεραιότητα στήλης. Σε αυτό το σημείο αξίζει να αναφερθεί ότι στην εφαρμογή του CSU εκτός από τον κλασικό πολλαπλασιασμό πινάκων (AxB), λαμβάνει χώρα αρκετά συχνά και ο πολλαπλασιασμός αριστερής αναστροφής (A^TxB), κατά τον οποίο ο πίνακας Α, προκειμένου να πολλαπλασιαστεί ο ανάστροφός του, διαβάζεται με προτεραιότητα στήλης. Για το λόγο αυτό, στη σχεδίασή μας υπάρχει αντίστοιχο σήμα εισόδου στη μονάδα ανάγνωσης του πίνακα Α, που δηλώνει αν υπάρχει περίπτωση ανάγνωσης των υποπινάκων [64,64] του Α με προτεραιότητα στήλης ή γραμμής. Ο ψευδοκώδικας που περιγράφει με γενικό τρόπο τη διαδικασίας ανάγνωσης των εξωτερικών μνημών του πίνακα Α με προτεραιότητα γραμμής παρουσιάζεται στην εικόνα 4.7

```
while ! read all subrowsA {
        while ! read all subcolsB {
                while ! read all subcolsA{
                        while  count blockrows < (64/number_of_SRAMs){
                                while count blockcols < 64{
                                        if  addr != out of border
                                        {       read data  from addr of SRAMs }
                                        else
                                        {       do not read }
                                count blockcols+=1
                                }
                        count blockrows+=1
                        }
                subcolsA+=1
                }
         subcolsB+=1
        }
subrowsA+=1
}
```

Εικόνα 4.7: Ψευδοκώδικας ανάγνωσης δεδομένων του πίνακα Α με προτεραιότητα γραμμής

Όπου:

- subrowsA είναι ο αριθμός των γραμμών του A διαιρεμένος με τον αριθμό των γραμμών του υποπίνακα, στρογγυλοποιημένος προς τα πάνω. Έτσι, για πίνακα A[160,110] και υποπίνακα [64,64], subrowsA=3.
- subcolsA είναι ο αριθμός των στηλών του A διαιρεμένος με τον αριθμό των στηλών του υποπίνακα, στρογγυλοποιημένος προς τα πάνω. Έτσι, για πίνακα A[160,110] και υποπίνακα [64,64], subcolsA=2. Αντίστοιχα υπολογίζεται το subcolsB.
- count blockcolsA είναι ο αριθμός των στηλών του υποπίνακα [64,64], που έχουν διαβαστεί από κάθε SRAM
- count blockrowsA είναι ο αριθμός των γραμμών του υποπίνακα [64,64] που έχουν ήδη διαβαστεί
- out of border δηλώνει ότι η διεύθυνση που υπολογίστηκε είναι λανθασμένη. Επειδή η παραπάνω διαδικασία διαβάζει δεδομένα υποπινάκων [64,64], σε περίπτωση που κάποιος πίνακας δεν έχει διαστάσεις πολλαπλάσιες του 64 (το οποίο προφανώς είναι και το πιθανότερο), η μονάδα διαθέτει ελεγκτικούς μηχανισμούς για αποφυγή ανάγνωσης δεδομένων από εσφαλμένες διευθύνσεις. Με τη χρήση των σημάτων control_rows και control_cols το σύστημα διαπιστώνει αν η διεύθυνση που διαβάζεται βρίσκεται σε επιτρεπτά όρια γραμμών και στηλών του πίνακα. Σε περίπτωση μη έγκυρης διεύθυνσης αγνοούνται τα δεδομένα που διαβάστηκαν, και αντικαθίστανται από “0”.
- στον ψευδοκώδικα, για λόγους απλότητας, δεν παρουσιάζεται ο μηδενισμός των μετρητών. Από τη δομή του κώδικα είναι φανερό ότι όταν ένας μετρητής αυξάνεται, όσοι έχουν αυξηθεί πριν από αυτόν μηδενίζονται. Έτσι, για παράδειγμα στην εντολή

subcolsA+=1 υπονοούνται και οι εντολές count blockrows = 0 και count blockcols = 0.

Σε περίπτωση που απαιτείται να διαβαστούν τα στοιχεία του Α ανεστραμμένα, η μόνη αλλαγή είναι η αντιμετάθεση του 1ου και του 3ου while-loop, ώστε να έχουμε προσπέλαση με προτεραιότητα στήλης. Αντίστοιχα, ο ψευδοκώδικας για ανάγνωση δεδομένων από τις SRAMs του πίνακα Β, όπου έχουμε πάντα ανάγνωση με προτεραιότητα γραμμής, φαίνεται στην εικόνα 4.8.

```
while ! read all subrowsA {
        while ! read all subcolsB {
                while ! read all subrowsB {
                        while  count blockrows < (64/number_of_SRAMs){
                                while count blockcols < 64{
                                        if  addr != out of border
                                        {       read data  from addr of SRAMs }
                                        else
                                        {       do not read }
                                count blockcols+=1
                                }
                        count blockrows+=1
                        }
                subcolsA+=1
                }
         subcolsB+=1
        }
subrowsA+=1
}
```

Εικόνα 4.8: Ψευδοκώδικας ανάγνωσης δεδομένων του πίνακα Β

4.4.3 Μονάδα εγγραφής δεδομένων σε SRAM

Η μονάδα αυτή αποτελεί την τρίτη βαθμίδα ομοχειρίας του συστήματος. Λειτουργεί με παρεμφερή τρόπο με τη μονάδα ανάγνωσης από SRAM, με τις πιο ουσιαστικές διαφορές να είναι ότι η μονάδα, προφανώς, αποθηκεύει δεδομένα αντί να διαβάζει ενώ διαχειρίζεται μόνο μία ομάδα από SRAMs, που είναι αυτή η οποία διατηρεί το τελικό αποτέλεσμα του πολλαπλασιασμού και ονομάζεται SRAM C. Τα δεδομένα που λαμβάνει η μονάδα στέλνονται από μία ομάδα από block RAMS, η οποία αποτελεί το δεύτερο επίπεδο εγγραφής δεδομένων στις C-BRAMs, όπως φάνηκε και στην εικόνα 4.4. Όπως και στην μονάδα ανάγνωσης η τελική μας υλοποίηση περιλαμβάνει 8 SRAMs για την αποθήκευση του πίνακα C.

Μία σημαντική παρατήρηση που αφορά τη λειτουργία της μονάδας είναι ότι δεδομένα στις εξωτερικές μνήμες γράφονται μόνο όταν ολοκληρωθεί ο υπολογισμός ενός υποπίνακα [64,64] του τελικού πίνακα C. Αυτό σημαίνει ότι αν οι πίνακες A και B έχουν αριθμό γραμμών και στηλών, αντίστοιχα, μεγαλύτερο του 64, δε θα γράφονται δεδομένα στις C-SRAMs σε κάθε περίοδο ομοχειρίας, παρά μόνο όταν ολοκληρωθεί ο υπολογισμός του υποπίνακα.

```
while ! read all subrowsC {
    while ! read all subcolsC {
        while ! read all subcolsC{
            while count blockrows < (64/number_of_SRAMs){
                while count blockcols < 64{
                    if writeEn from block = 1 {
                        if addr != out of border
                        {   write data to addr of RAMs}
                        else
                        {   do not write }
                        count blockcols+=1 }
                    else { continue } }
                count blockrows+=1 }
            subcolsA+=1 }
        subrowsB+=1 }
subrowsA+=1 }
```

Εικόνα 4.9: Ψευδοκώδικας εγγραφής δεδομένων στις C-SRAMs

4. 5 Μονάδα πολλαπλασιασμού υποπινάκων

Η μονάδα αυτή αποτελεί τη δεύτερη βαθμίδα της κύριας ομοχειρίας της σχεδίασης. Αποτελείται από 3 υπομονάδες, οι οποίες είναι και αυτές πλήρως pipelined. Σκοπός της μονάδας είναι να διαβάζει δεδομένα από δύο ομάδες BRAMs, να τα μεταφέρει σε buffers διαστάσεων 256 λέξεων (που αποτελούν υποπίνακες μεγέθους [16,16]), να εκτελείται παράλληλος πολλαπλασιασμός των buffers και το αποτέλεσμα που μεταφέρεται σε ένα τρίτο buffer, και τέλος το τελικό περιεχόμενο κάθε buffer αποτελέσματος, να αποθηκεύεται σε BRAMs.

Η πρώτη υπομονάδα της βαθμίδας πολλαπλασιασμού εγγράφει τα δεδομένα που διαβάστηκαν από τις SRAMs των πινάκων Α και Β, σε

Block RAMs συνολικού αποθηκευτικού χώρου, για κάθε υποπίνακα, 4096 λέξεων των 32 bits. Παράλληλα, διαβάζει από block RAMs στις οποίες έχουν εγγραφεί δεδομένα την προηγούμενη περίοδο της βασικής ομοχειρίας, και μεταφέρει τα δεδομένα υποπινάκων [16,16] σε δύο buffers (BufferA, BufferB) αντίστοιχων διαστάσεων.

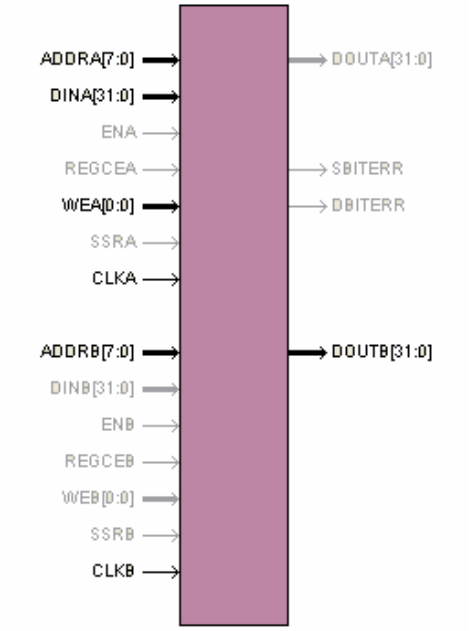
Η δεύτερη υπομονάδα δέχεται ως είσοδο τα δύο buffers και με χρήση 256 παράλληλων βαθμίδων MAC (Multiplier - Accumulator), υπολογίζει το αποτέλεσμα του BufferC.

Η τρίτη υπομονάδα αποτελείται από δύο επίπεδα από BRAMs. Στο πρώτο αποθηκεύονται τα δεδομένα των BufferC, τα οποία αθροίζονται με τέτοιο τρόπο ώστε μετά το τέλος μιας περιόδου της βασικής ομοχειρίας να έχει υπολογιστεί ένας πίνακας διαστάσεων [64,64]. Το δεύτερο επίπεδο χρησιμοποιείται μόνο για να μεταφέρει δεδομένα από την FPGA στις εξωτερικές μνήμες του πίνακα C.

4.5.1 Δομή μονάδας Block RAM

Οι BRAMs αποτελούν βασικό στοιχείο της πρώτης και της τρίτης βαθμίδας του εσωτερικού pipeline της μονάδας πολλαπλασιασμού πινάκων [64,64]. Κάθε ομάδα από BRAMs αποθηκεύει συνολικά 4096 στοιχεία. Το μέγεθος κάθε μονάδας, οπότε και ο αριθμός των μνημών κάθε ομάδας, καθορίζεται από τις πόσες παράλληλες προσβάσεις απαιτούνται ώστε οι βαθμίδες του pipeline να έχουν παραπλήσια χρονική διάρκεια. Όπως θα αναλυθεί και παρακάτω, αυτό επιτυγχάνεται με χρήση 16 BRAMs για την αποθήκευση κάθε υποπίνακα [64,64]. Επομένως, αντί να χρησιμοποιείται, για τις συγκεκριμένες διαστάσεις ενός υποπίνακα, μία μονάδα BRAM διαστάσεων 4096x32 bits,

χρησιμοποιούνται 16 BRAMs διαστάσεων 256x32 bits. Στην εικόνα 4.10 παρουσιάζεται το block diagram μιας τέτοιας μνήμης, όπως παράγεται από το Core Generator της Xilinx. Η μονάδα μπορεί να εκτελέσει ταυτόχρονα από μία εγγραφή και μία ανάγνωση, με απόδοση μέχρι και 550 MHz.



Εικόνα 4.10: Block Diagram μιας Simple Dual Port RAM 256x32

4.5.2 Μονάδα ανάγνωσης υποπινάκων από Block RAMs

Για κάθε μία από τις δύο βασικές λειτουργίες της μονάδας (ανάγνωση από SRAMs και εγγραφή σε Buffers) χρησιμοποιείται και μία ξεχωριστή Μηχανή Πεπερασμένων Καταστάσεων. Προκειμένου να μην προστίθεται επιπλέον καθυστέρηση στη μονάδα, οι δύο αυτές διεργασίες εκτελούνται παράλληλα. (Έτσι, σχεδόν όσος χρόνος απαιτείται για την ανάγνωση από τις SRAMs και εγγραφή στις BRAMs 4096 στοιχείων για κάθε πίνακα, απαιτείται και για την ανάγνωση 64 φορές υποπινάκων [16,16].)

Για κάθε πίνακα [64,64] χρησιμοποιούνται 16 μνήμες 256x32. Επομένως, για τους Α υποπίνακες [64,64] χρησιμοποιούνται δύο

αντίγραφα 16 μνημών 256x32. Στο ένα αντίγραφο αποθηκεύονται τα δεδομένα που έρχονται από τις A SRAMs και από το άλλο διαβάζονται διαδοχικά 256 λέξεις και αποθηκεύονται στο BufferA. Την επόμενη περίοδο της ομοχειρίας, το αντίγραφο που αποθήκευσε τα δεδομένα, θα διαβάζει τα στοιχεία και θα τα στέλνει στα buffers και το αντίγραφο που διάβαζε δεδομένα στη νέα περίοδο θα αποθηκεύει καινούρια δεδομένα, κ.ο.κ. Αντίστοιχη διαδικασία ακολουθείται και για τις μνήμες του B. Ο ψευδοκώδικας της διαδικασίας παρουσιάζεται στην εικόνα 4.11.

```
if reset =1
 {
        writeA1_readA2=1
        writeB1_readB2=1
}
else
{
        writeA1_readA2= not (writeA1_readA2)
        writeB1_readB2= not (writeB1_readB2)
}
```

Εικόνα 4.11: Ψευδικώδικας επιλογής ομάδας από BRAMs

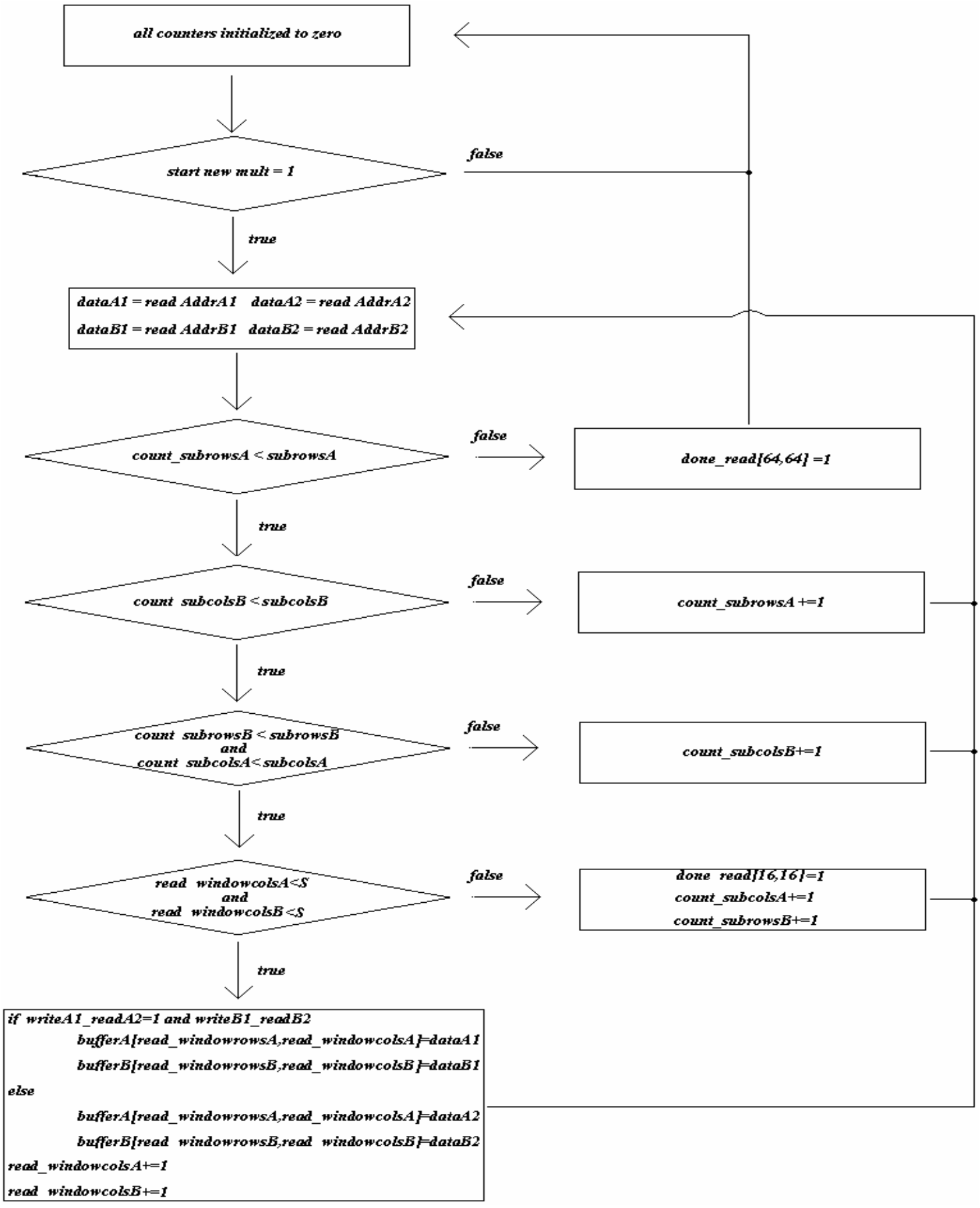
Στην εικόνα 4.12 παρουσιάζονται οι βασικές λειτουργίες της μονάδας ανάγνωσης στοιχείων των υποπινάκων A[64,64] και B[64,64] από τις BRAMs και καταχώρησης των στοιχείων αυτών στα bufferA και bufferB. Στην παρούσα FSM θεωρείται ότι το μέγεθος των πινάκων είναι [64,64], ενώ οι διαστάσεις στους υποπίνακες θεωρούνται [16,16]. Ως S δηλώνεται το μέγεθος του παραθύρου του υποπίνακα, που στη δική μας εφαρμογή είναι 16, ενώ ως read windowrowsA, windowcolsA, windowrowsB και windowcolsB, δηλώνονται οι δείκτες διεύθυνσης στα

buffersA και B αντίστοιχα. Τα υπόλοιπα σήματα της FSM γίνονται κατανοητά από τις επεξηγήσεις στις προηγούμενες ενότητες.

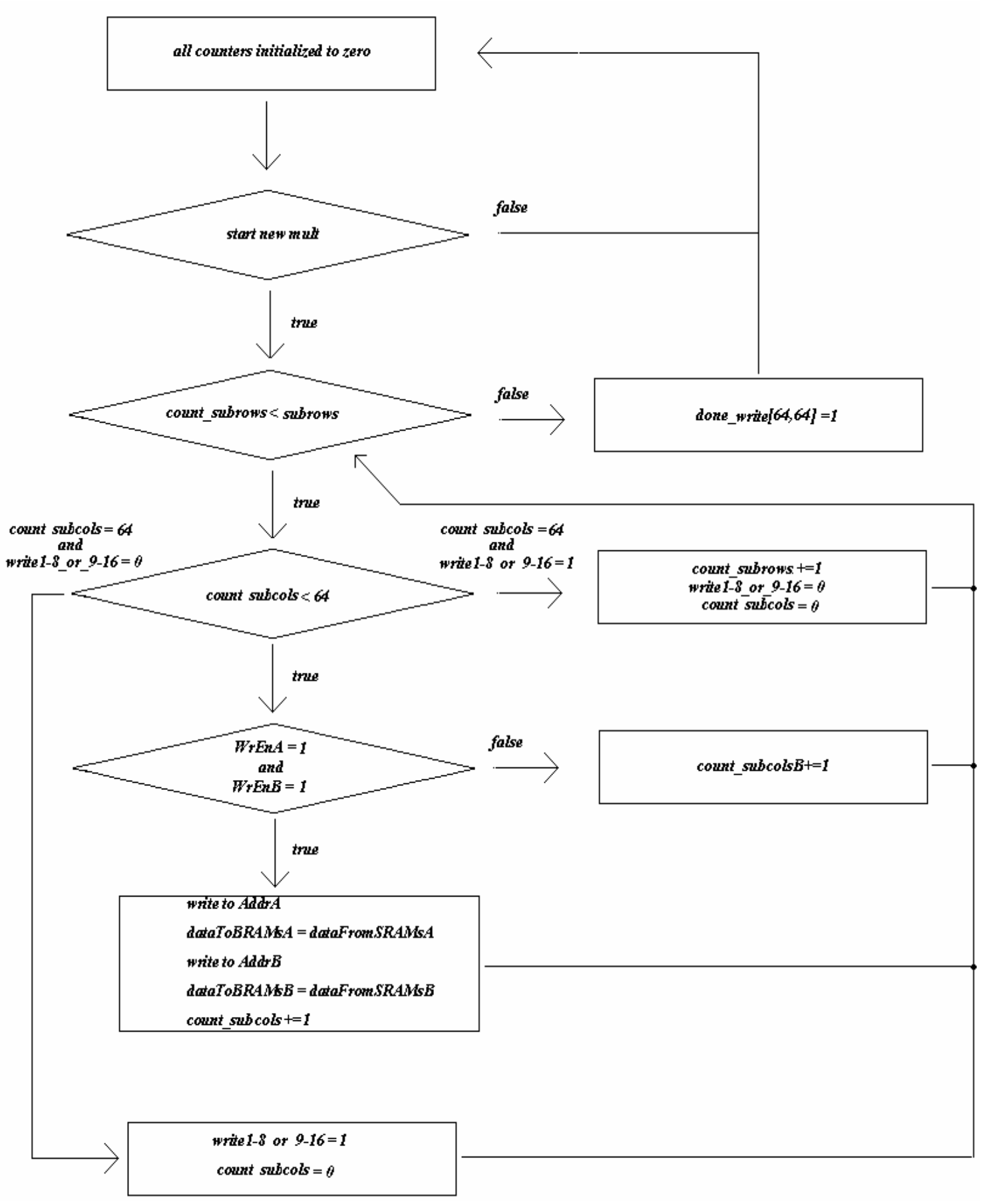
Μπορεί να παρατηρηθεί ότι μετά από 16 προσβάσεις από τις BRAMs τα buffers έχουν δεχθεί 256 στοιχεία, οπότε μπορεί να ξεκινήσει η επόμενη βαθμίδα του εσωτερικού pipeline, που είναι ο πολλαπλασιασμός των buffers, ενώ η μονάδα ανάγνωσης ξεκινάει μία νέα ανάγνωση 256 δεδομένων. Σημειώνεται ότι η διαδικασία αυτή επαναλαμβάνεται 64 φορές, οπότε και ολοκληρώνονται οι αναγνώσεις για τον πολλαπλασιασμό A[64,64] x B[64,64].

Τέλος, για την περίπτωση του ανάστροφου πολλαπλασιασμού πινάκων, η μονάδα αντιμεταθέτει τις τιμές μεταξύ subcolsA και subrowsA, ενώ αναστρέφει τη διευθυνσιοδότηση του bufferA.

Η δεύτερη Μηχανή Πεπερασμένων Καταστάσεων της μονάδας αναφέρεται στην εγγραφή των δεδομένων που έρχονται από τις SRAMs. Στο σχήμα 4.13 παρουσιάζεται η γενική δομή της συγκεκριμένης FSM. Αξίζει να σημειωθεί ότι επειδή η εξωτερικές μνήμες παράγουν 8 στοιχεία ανά κύκλο ανάγνωσης ενώ διατίθενται 16 BRAMs για κάθε πίνακα [64,64], με ειδικά σήματα ελέγχου (στο σχήμα φαίνεται το write1-8_or_9-16) ρυθμίζεται η FSM ώστε τα αποτελέσματα που παράγονται από τις SRAMs να οδηγούνται στις σωστές διευθύνσεις των BRAMs. Με παρόμοιο τρόπο υλοποιήθηκε και η διασύνδεση των δεδομένων, όταν έχουμε 1, 2,4 ή 16 εξωτερικές μνήμες για κάθε πίνακα.



Εικόνα 4.12: Γενική FSM για τη μονάδα ανάγνωσης από BRAMs



Εικόνα 4.13: Γενική FSM για τη μονάδα εγγραφής σε BRAMs

4.5.3 Μονάδα εκτέλεσης παράλληλων πολλαπλασιασμών

Η μονάδα εκτέλεσης παράλληλων πολλαπλασιασμών είναι η πιο βασική της σχεδίασης. Η κύρια λειτουργία που εκτελεί είναι ο παράλληλος πολλαπλασιασμός στοιχείων των Α και Β. Περιγραφικά, ο βασικός αλγόριθμος που εκτελεί η μονάδα παρουσιάζεται στον ψευδοκώδικα της εικόνας 4.14:

```
for k=0; i<16; k++ {
    start parallel stage :
    for i=0; i<16; i++ {
        for j=0; j<16; j++ {
                BufferC (i)(j)= BufferC (i)(j) + BufferA(i)(k) x BufferB(k)(j)   }}}
```

Εικόνα 4.14: Ψευδοκώδικας που υλοποιεί η μονάδα εκτέλεσης παράλληλων πολλαπλασιασμών

Στην παραπάνω εικόνα φανερώνεται ότι σε κάθε κύκλο πολλαπλασιασμού εκτελούνται 256 πολλαπλασιασμοί. Έτσι, κάθε γραμμή της k-οστής στήλης του πίνακα BufferA[16,16] πολλαπλασιάζεται με κάθε στήλη της k-οστής γραμμής του πίνακα BufferB[16,16]. Μόλις ολοκληρωθεί ο πολλαπλασιασμός στον επόμενο κύκλο εκτελείται η πράξη της πρόσθεσης με το μερικό γινόμενο του BufferC. Κάθε φορά που ολοκληρώνεται ο πολλαπλασιασμός μίας γραμμής του A[64,64] (και αντίστοιχα μίας στήλης του B[64,64]) το μερικό γινόμενο του BufferC μηδενίζεται.

Για λόγους που θα εξηγηθούν στο κεφάλαιο παρουσίασης αποτελεσμάτων, για τους πολλαπλασιασμούς χρησιμοποιήθηκαν DSPs.

4.5.4 Μονάδα εγγραφής υποπινάκων σε Block RAMs

Η μονάδα εγγραφής αποτελεί το τρίτο επίπεδο pipeline της βαθμίδας πολλαπλασιασμού πινάκων [64,64]. Αποτελείται από δύο επίπεδα από BRAMs, όπου κάθε επίπεδο διαθέτει 16 BRAMs των 256x32 bits η κάθε μία.

Στο πρώτο επίπεδο αποθηκεύονται δεδομένα που φτάνουν από την υπομονάδα παράλληλου πολλαπλασιασμού. Έτσι, κάθε φορά που τερματίζει ο υπολογισμός των πινάκων [16,16]x[16,16] και παράγεται ένα νέο buffer [16,16], τα 256 αυτά δεδομένα αποθηκεύονται σε κατάλληλη διεύθυνση το καθένα. Θυμίζεται ότι η μονάδα παράλληλου πολλαπλασιασμού στέλνει δεδομένα στην μονάδα εγγραφής μόνο όταν ολοκληρωθεί ο πολλαπλασιασμός μίας γραμμής του πίνακα A[64,64] με μία στήλη του πίνακα B[64,64] (όταν, δηλαδή, υπολογίζεται ένα γινόμενο [16,64]x[64,16]).

Είναι προφανές, ότι επειδή οι περισσότεροι πίνακες που διαχειρίζεται το σύστημα έχουν διαστάσεις μεγαλύτερες του 64, το τελικό αποτέλεσμα πιθανόν να έχει και αυτό διαστάσεις μεγαλύτερες από αυτές που μπορεί να αποθηκεύσει η μονάδα C-BRAMs. Για το λόγο αυτό, το σύστημα πρέπει να αντιλαμβάνεται πότε ολοκληρώνεται ένας πολλαπλασιασμός μιας ολόκληρης γραμμής του πίνακα A (με πάχος γραμμής 64) με μία στήλη του πίνακα B (με πάχος στήλης 64). Μέχρι να ολοκληρωθεί, όλα τα αποτελέσματα που φτάνουν από το BufferC πρέπει να λαμβάνονται ως μερικά γινόμενα και να προστίθενται με τα προηγούμενα των αντίστοιχων διευθύνσεων.

Έστω λοιπόν, ότι έχουμε τον πολλαπλασιασμό A[64,128]xB[128,192]. Το τελικό αποτέλεσμα θα έχει διαστάσεις C[64,192], ενώ ο πρώτος υποπίνακας C[64,64] θα προκύψει έπειτα από τον πολλαπλασιασμό της πρώτης γραμμής του A(με πάχος 64 στήλες) και της πρώτης στήλης του B(με πάχος 64 γραμμές). Για το αποτέλεσμα του συγκεκριμένου υποπίνακα, προφανώς, απαιτούνται δύο περίοδοι πολλαπλασιασμού πινάκων [64,64] (αφού η κοινή διάσταση του πολλαπλασιασμού έχει μέγεθος 128). Επομένως, όταν αρχίσει η εκτέλεση στη δεύτερη περίοδο, όλα τα στοιχεία του BufferC θα προστίθενται στα αντίστοιχα στοιχεία που είναι ήδη αποθηκευμένα στο πρώτο επίπεδο της C-BRAM. Ωστόσο, το αποτέλεσμα της άθροισης (που αποτελεί μέρος του τελικού αποτελέσματος) αποθηκεύεται στο δεύτερο επίπεδο των BRAMs, ενώ το πρώτο επίπεδο αντικαθίσταται με 0.

Τα παραπάνω παρουσιάζονται στο σχήμα 4.14, όπου φαίνεται ο αντίστοιχος ψευδοκώδικας. Υποθέτουμε ότι διαθέτουμε δύο σήματα: start_line και end_line, τα οποία, όπως είναι φανερό, δηλώνουν την αρχή και το τέλος μίας σειράς πολλαπλασιασμού.

Ο λόγος για τον οποίο χρειαζόμαστε και Level 2 για τη μονάδα εγγραφής σε BRAMs, το οποίο αποτελείται από ένα δεύτερο αντίγραφο 16 μνημών μεγέθους 256x32, είναι γιατί, όπως και στην περίπτωση της ανάγνωσης από SRAMs, η εγγραφή υποπινάκων [64, 64] είναι αρκετά χρονοβόρα. Με χρήση 8 εξωτερικών μνημών για τον πίνακα C απαιτούνται 256 αργοί κύκλοι εγγραφής, ενώ κάθε 16 υπολογισμούς MAC στέλνονται δεδομένα στη μονάδα εγγραφής των BRAMs. Με τη μέθοδο των δύο επιπέδων αποφεύγεται ο κίνδυνος να πανωγραφούν χρήσιμα δεδομένα που ακόμα δεν έχουν μεταφερθεί στις C-SRAMs. Έτσι, ενώ στο πρώτο επίπεδο μεταφέρονται συνεχώς δεδομένα από τη μονάδα παράλληλου

πολλαπλασιασμού, το δεύτερο επίπεδο μεταφέρει τελικά δεδομένα στις C-SRAMs, με τρόπο ανάλογο με αυτόν που η μονάδα ανάγνωσης διαβάζει δεδομένα από A-SRAMs και B-SRAMs.

```
if (start_line='0') and (end_line='0') {
        dataCinToLevel_1 = dataCoutFromLevel_1 + BufferC
        dataCinToLevel_2 = "0"
        WrEnLevel_1 = "1"
        WrEnLevel_2 = "0"}
elsif (start_line='1') and (end_line='0') {
        dataCinToLevel_1 = BufferC
        dataCinToLevel_2 = "0"
        WrEnLevel_1 = "1"
        WrEnLevel_2 = "0" }
elsif (start_line='0') and (end_line='1') {
        dataCinToLevel_1 = "0"
        dataCinToLevel_2 = dataCoutFromLevel_1 + BufferC
        WrEnLevel_1 = "1"
        WrEnLevel_2 = "1" }
elsif (start_line='1') and (end_line='1') {
        dataCinToLevel_1 = "0"
        dataCinToLevel_2 = BufferC
        WrEnLevel_1 = "1"
        WrEnLevel_2 = "1" }
```

Εικόνα 4.14: Ψευδοκώδικας αποθήκευσης των δεδομένων του BufferC στη μονάδα εγγραφής σε BRAMs

5

ΥΛΟΠΟΙΗΣΗ ΔΥΑΔΙΚΗΣ ΑΝΑΖΗΤΗΣΗΣ ΣΕ ΑΝΑΔΙΑΤΑΣΣΟΜΕΝΗ ΛΟΓΙΚΗ

5.1 Εισαγωγή

Ο αλγόριθμος της εφαρμογής που περιγράφει την κατηγοριοποίηση καρκινογόνων γονιδίων με τη χρήση των SVMs υλοποιήθηκε μέσω Matlab, στο εργαστήριο "Ψηφιακής Εικόνας και Επεξεργασία Σήματος (DISPLAY)", του Πολυτεχνείου Κρήτης, από τον καθηγητή κ. Μ. Ζερβάκη και τον διδακτορικό κ. Ε. Μπλαζαντωνάκη.

Χρησιμοποιώντας τη λειτουργία "Profiling" που διαθέτει το εργαλείο της Matlab, διαπιστώθηκε ότι το αργότερο υπολογιστικά κομμάτι ήταν η εκτέλεση της συνάρτησης "Remove Genes Binary". Ο αλγόριθμος που υλοποιήθηκε, έχοντας ως σκοπό τη σταδιακή μείωση του αριθμού των γονιδίων (καθώς και των δειγμάτων που αντιστοιχούν σε κάθε γονίδιο) καλεί μία ρουτίνα 120 φορές, κατά τη διάρκεια της οποίας τα γονίδια εντοπίζονται και διαγράφονται από τη λίστα.

Έτσι, αφού τα γονίδια χωριστούν σε κλάσεις με βάση τα κριτήρια που καθορίζει η χρήση των Support Vector Machines, τα "υγιή" γονίδια, μαζί

με τα δείγματά τους, πρέπει να διαγραφούν από τη λίστα. Καθώς η λίστα των γονιδίων που πρέπει να διαγραφούν δεν είναι ταξινομημένη, τα γονίδια εντοπίζονται με τη χρήση του αλγορίθμου Binary Search, στην ταξινομημένη λίστα εισόδου και στη συνέχεια κάθε ένα από αυτά διαγράφονται από τη λίστα (μαζί με τα δείγματά τους). Η "RemoveGenesBinary" είναι η συνάρτηση που εκτελεί την παραπάνω διεργασία.

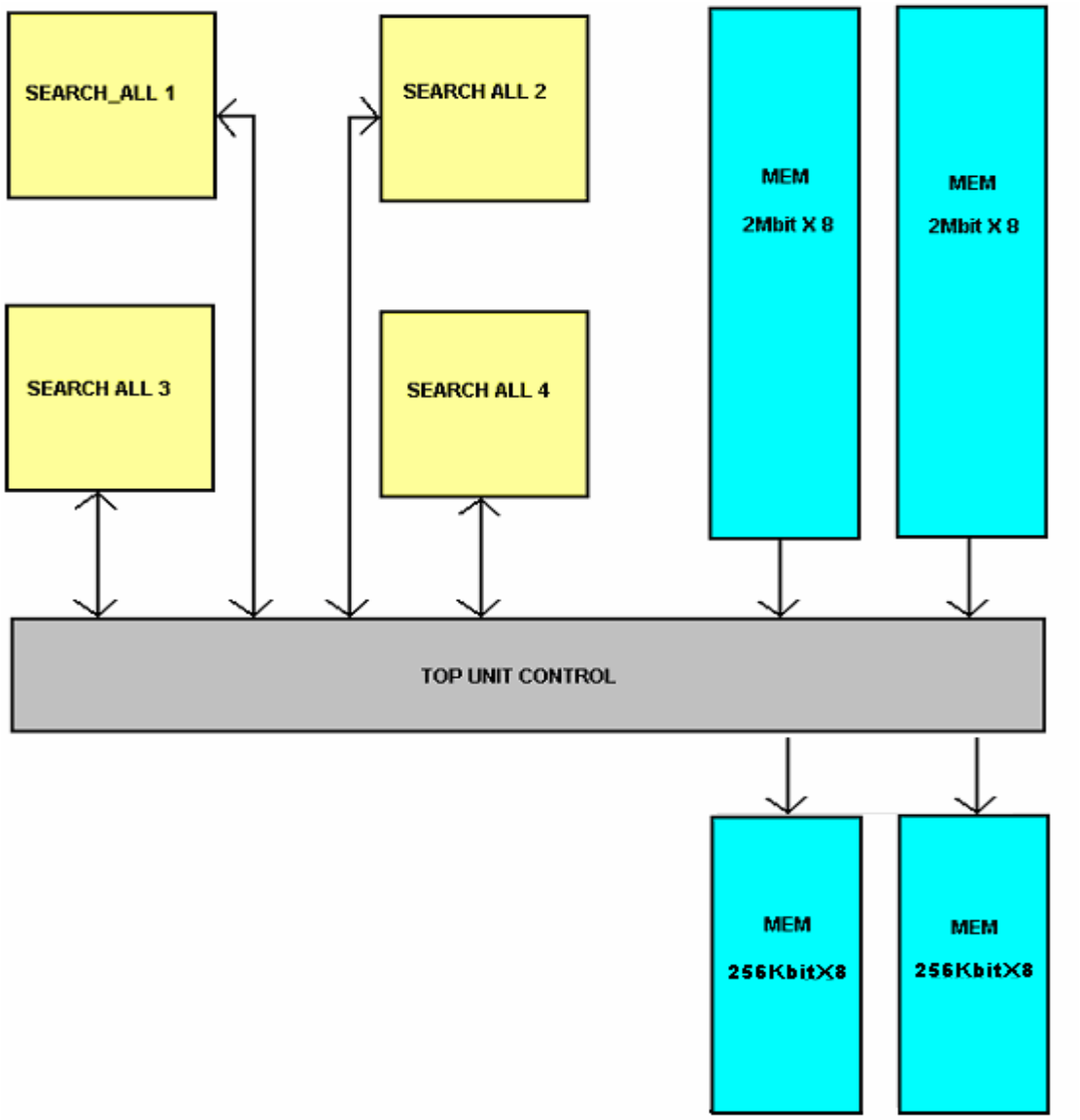
Πιο συγκεκριμένα, μετά το πέρας της πρώτης εκτέλεσης της συνάρτησης, τα γονίδια της λίστας μειώνονται από 24188 σε 2048. Αυτό σημαίνει ότι στη συγκεκριμένη εφαρμογή, μόλις στην πρώτη επανάληψη της ρουτίνας (από τις συνολικά 120 επαναλήψεις), ο αλγόριθμος δυαδικής αναζήτησης καλείται 22140 φορές. Η επανάληψη της δυαδικής αναζήτησης προκαλεί σημαντική καθυστέρηση στο χρόνο εκτέλεσης της εφαρμογής, η οποία αυξάνεται όσο μεγαλώνει το μέγεθος των δεδομένων εισόδου.

5.2 Περιγραφή σχεδίασης της μονάδας δυαδικής αναζήτησης

Η εικόνα 5.1 παρουσιάζει τη γενική σχεδίαση της υλοποίησης. Αποτελείται από τέσσερεις μονάδες "Search_All", οι οποίες εκτελούν τον αλγόριθμο της παράλληλης δυαδικής αναζήτησης (για διαφορετικά στοιχεία η κάθε μονάδα), δύο εξωτερικές SRAMs για ανάγνωση των δεδομένων εισόδου και δύο εξωτερικές SRAMs για εγγραφή των δεδομένων εξόδου.

Συνολικά, αρχικά φορτώνονται στις μονάδες Search_All όλα τα γονίδια που υπάρχουν στην αρχική λίστα. Στη συνέχεια, από τη λίστα διαγραφής

γονιδίων διαβάζονται τα στοιχεία που πρέπει να διαγραφούν. Η “top unit control” ελέγχει τον τρόπο με τον οποίο διαμοιράζονται τα στοιχεία αυτά στις Search_All μονάδες. Κάθε μία από τις μονάδες αυτές αναλαμβάνει να εντοπίσει διαφορετικές ομάδες στοιχείων. Ακολούθως, με τον αλγόριθμο δυαδικής αναζήτησης, σημειώνεται η διεύθυνση των στοιχείων που βρέθηκαν. Τελικά, όταν ολοκληρωθεί η διαδικασία αυτή, τα στοιχεία που δεν έχουν εντοπιστεί, μαζί με τα δείγματά τους, μεταφέρονται στην εξωτερική μνήμη εξόδου, η οποία αντικαθιστά πλέον την αρχική λίστα γονιδίων.



Εικόνα 5.1: Γενικό σχηματικό διάγραμμα της συνολικής σχεδίασης

5.2.1 Περιγραφή της εξωτερικής μονάδας SRAM

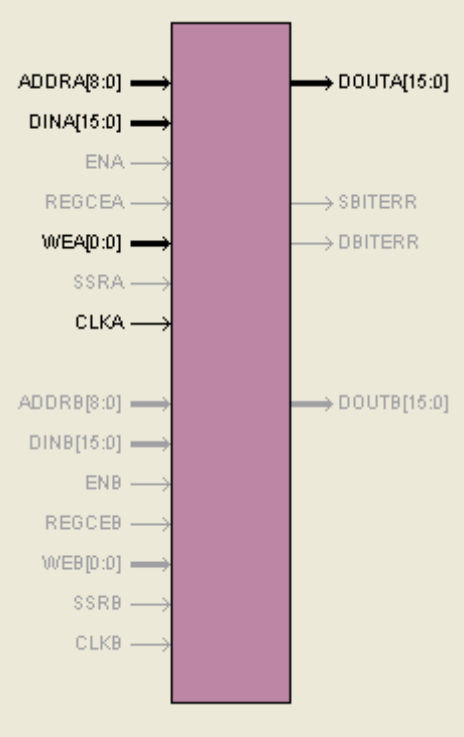
Όπως αναφέρθηκε παραπάνω, η προτεινόμενη σχεδίαση δέχεται ως είσοδο την λίστα εισόδου των 24188 γονιδίων, μαζί με τα δείγματά τους (κάθε γονίδιο συνοδεύεται με 78 δείγματα). Οι μεταβλητές αυτές περιγράφονται με αριθμούς δυαδικής μορφής των 16 bits. Προκειμένου λοιπόν να διατηρηθεί η πληροφορία αυτή, απαιτείται μία μνήμη που να διαθέτει 24188x(78+1) = 1.910.852 διευθύνσεις, για λέξεις των 16 bits. Για το λόγο αυτό επιλέχθηκε το μοντέλο CY7C1069AV33 (2Mx 8 Static RAM) που παράγεται από την εταιρία Cypress και έχει λειτουργία αντίστοιχη με αυτή που περιγράφηκε στην ενότητα 4.4.1. Τα δεδομένα που είναι αποθηκευμένα στη μνήμη έχουν την ίδια μορφή με αυτά που παράγονται από τον κώδικα της Matlab, που σημαίνει ότι κάθε γονίδιο ακολουθείται από 78 τιμές δειγμάτων. Ξεκινώντας, λοιπόν, από τη διεύθυνση 0, κάθε 79 διαδοχικές διευθύνσεις, είναι τοποθετημένη η χαρακτηριστική τιμή ενός γονιδίου. Με αυτόν τον τρόπο λαμβάνονται όλα τα γονίδια και τοποθετούνται σε Block RAMs, όπως θα περιγραφεί στη συνέχεια.

Αντίστοιχη λογική ακολουθείται για την εξωτερική μονάδα SRAM που αποθηκεύονται τα τελικά αποτελέσματα. Η μόνη διαφορά έγκειται στο ότι χρειάζονται να αποθηκευτούν μόνο 2048 ομάδες γονιδίων και δειγμάτων, οπότε μία μνήμη με 256K διευθύνσεις καλύπτει τις απαιτήσεις για τη συγκεκριμένη εφαρμογή. Για το λόγο αυτό χρησιμοποιήθηκαν δύο αντίγραφα της μνήμης CY7C1010DV33(256Kx8 Static RAM) που παράγεται από τη Cypress.

5.2.2 Περιγραφή της Top Binary Search (TBS) Unit

Βασική μονάδα της σχεδίασης είναι η "Top Binary Search Unit" ή αλλιώς μονάδα TBS. Κάθε μονάδα TBS διαθέτει μία Binary Search Unit, η οποία υλοποιεί τον αλγόριθμο δυαδικής αναζήτησης, και με επιπρόσθετη λογική ελέγχει τη λειτουργία της..

Η Binary Search Unit αποτελείται από μία Block RAM 512 λέξεων των 16 bits (εικόνα 5.2), και μία Μηχανή Πεπερασμένων Καταστάσεων, που διαβάζοντας δεδομένα από τη μνήμη εκτελεί το Binary Search. Ο τυπικός αναδρομικός αλγόριθμος του Binary Search παρουσιάζεται στην εικόνα 5.3, ενώ μία γενική μορφή της FSM που χρησιμοποιήθηκε φαίνεται στην εικόνα 5.4.

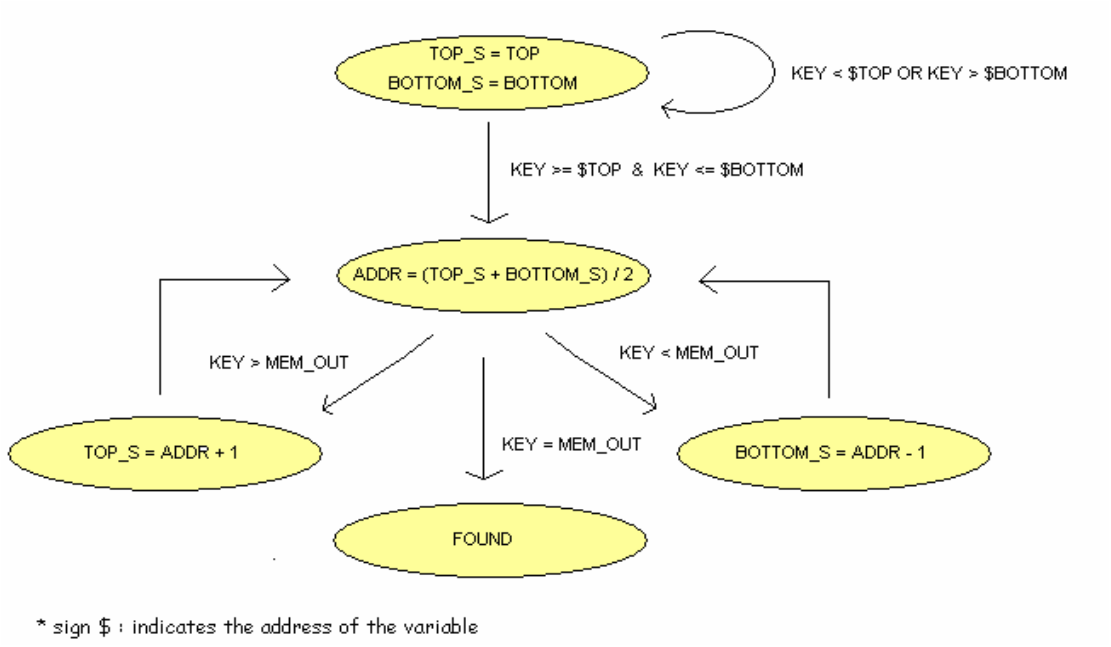


Εικόνα 5.2: Block Diagram μιας Single Port RAM 500x16

Όπως είναι φανερό, η μονάδα δέχεται ως είσοδο την χαρακτηριστική τιμή ενός γονιδίου που πρέπει να διαγραφεί (κλειδί αναζήτησης), και μετά το πέρας της αναζήτησης, επιστρέφει στην TBS ένα σήμα που δηλώνει ότι το στοιχείο βρέθηκε, ενώ παράλληλα μετατρέπει το αντίστοιχο bit εντοπισμού που διαθέτει κάθε θέση μνήμης από '0' σε '1'. Κάθε εκτέλεση του αλγορίθμου διαρκεί έως και 25 κύκλους ρολογιού.

```
BinarySearch (A [0..N-1], value, low,high) {
        if (high < low)
            Return -1          //not found
        else {
            Mid = (low + high) /2
            if (A[mid] > value)
                Return BinarySearch (A, value, low, mid-1)
             elsif (A[mid] < value)
                Return BinarySearch (A, value, mid+1, high)
             else
                return mid              //found
        }
}
```

Εικόνα 5.3: Αναδρομικός αλγόριθμος Binary Search



Εικόνα 5.4: FSM αλγορίθμου Binary Search

Η μονάδα TBS διαχειρίζεται τα δεδομένα εισόδου της Binary Search Unit και καθορίζει πότε η συγκεκριμένη μονάδα είναι απασχολημένη και πότε είναι διαθέσιμη να εξυπηρετήσει νέα αιτήματα για αναζήτηση στοιχείων. Πιο συγκεκριμένα, η μονάδα TBS γνωρίζει πότε η Binary Search Unit εκτελεί μία αναζήτηση. Σε περίπτωση που ισχύει κάτι τέτοιο (δηλαδή, η Binary Search Unit έχει δεχθεί ως είσοδο ένα κλειδί και εκτελείται ο αλγόριθμος προκειμένου να εντοπιστεί) και ταυτόχρονα εμφανιστεί στην είσοδο της TBS ένα άλλο κλειδί για αναζήτηση, για να αποφευχθούν δομικοί κίνδυνοι, η TBS διαθέτει μία θέση buffer (που ονομάζεται θέση αναμονής κλειδιού ή wait_key). Αυτή η θέση χρησιμοποιείται για να διατηρηθεί εκεί προσωρινά η τιμή του δεύτερου κλειδιού και να εξυπηρετηθεί το αίτημα για νέα αναζήτηση, όταν αυτή που εκτελείται ήδη στην Binary Search Unit τερματίσει. Σε μία τέτοια περίπτωση η μονάδα TBS στέλνει ως έξοδο ένα σήμα που δηλώνει ότι η συγκεκριμένη μονάδα είναι απασχολημένη. Μόνο εφόσον η Binary Search Unit της TBS ολοκληρώσει την αναζήτηση του αρχικού γονιδίου και η TBS στείλει το “wait_key” στη μονάδα Binary Search ως κύριο κλειδί, η μονάδα TBS γίνεται και πάλι μη-απασχολημένη. Υποθέτοντας ότι το κλειδί το οποίο στέλνεται στην Binary Search Unit ονομάζεται “main_key”, το κλειδί που αποθηκεύεται στο buffer ονομάζεται “wait_key” και το κλειδί που φτάνει στη μονάδα TBS ονομάζεται “new_arrival”, ο ψευδοκώδικας που παρουσιάζει συνοπτικά τους ελέγχους της TBS παρουσιάζεται στην εικόνα 5.5.

5.2.3 Περιγραφή της μονάδας Search_All

Η μονάδα Search_All αποτελείται από μία Block RAM (που ονομάζεται μνήμη διαγραφής), μεγέθους 22140x16, 49 μονάδες Top Binary Search (TBS) και μία μονάδα ελέγχου. Το σχηματικό διάγραμμα της μονάδας παρουσιάζεται στην εικόνα 5.6. Ο λόγος για τον οποίο χρησιμοποιούνται 49 μονάδες TBS εξηγείται στο κεφάλαιο παρουσίασης των αποτελεσμάτων. Σκοπός της μονάδας είναι να εντοπίσει και να διαγράψει μία ομάδα από γονίδια που βρίσκονται στη μνήμη διαγραφής όσο το δυνατόν πιο γρήγορα.

```
if reset {
      main_key = EMPTY
      wait_key = EMPTY
      occupied = 0 }
elsif !binary_search_works {
     main_key = wait key
     wait_key = EMPTY
     occupied = 0 }
elsif binary_search_works AND wait_key = EMPTY {
      wait_key = new_arrival
      occupied = 1 }
elsif binary_search_works AND wait_key != EMPTY {
      occupied = 1 }
```

Εικόνα 5.5: Ψευδοκώδικας λειτουργίας της μονάδας TBS

Η μνήμη διαγραφής της μονάδας περιλαμβάνει τα 22140 γονίδια που πρέπει να διαγραφούν. Για την υλοποίηση της μνήμης χρησιμοποιείται το εργαλείο Core Generator της Xilinx και παράγει μία Single Port RAM 22140x16.

Όπως έχει αναφερθεί, κάθε μονάδα TBS περιλαμβάνει και μία Block RAM που αποθηκεύει μέχρι 512 γονίδια. Κάθε ομάδα των στοιχείων αυτών είναι μέρος των 24188 γονιδίων που είναι αποθηκευμένα στην αρχική λίστα.

Επομένως:

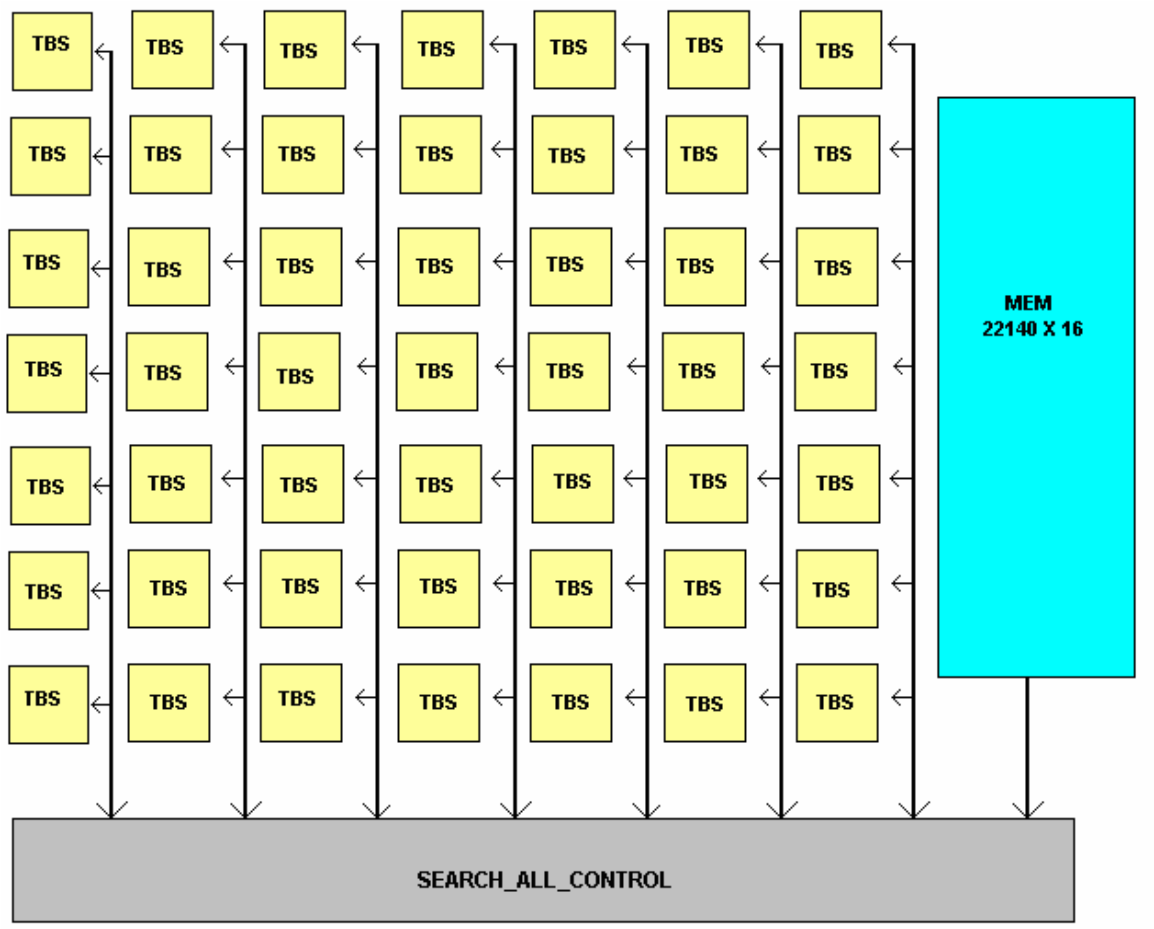
- τα πρώτα (24188/49) γονίδια αποθηκεύονται στην BRAM της TBS_1
- τα επόμενα (24188/49) γονίδια στην BRAM της TBS_2

.....

- τα τελευταία (24188/49) γονίδια στην BRAM της TBS_49

Όταν όλα τα γονίδια αποθηκευτούν σε BRAMs, δημιουργείται ένας πίνακας που δηλώνει τα όρια κάθε μνήμης, δηλαδή την τιμή του πρώτου και του τελευταίου γονιδίου σε κάθε μνήμη. Με τον τρόπο αυτό, η αναζήτηση ενός γονιδίου δεν είναι αναγκαίο να γίνει σε όλες τις μνήμες, αλλά μόνο στη συγκεκριμένη, στης οποίας τα όρια περιλαμβάνουν την τιμή του γονιδίου που αναζητείται.

Για μία συγκεκριμένη αναζήτηση, λοιπόν, μόνο μία από τις 49 μνήμες απασχολείται, γεγονός που ωθεί τη σχεδίαση παράλληλής μηχανής δυαδικής αναζήτησης. Έτσι, με στόχο η Search_All unit να εκμεταλλευτεί την παραλληλία που παρέχει η FPGA, η μονάδα ελέγχου της μονάδας σχεδιάζεται ώστε να εκτελεί παράλληλες αναζητήσεις.



Εικόνα 5.6: Σχηματικό διάγραμμα μονάδας Search_All

Σε μία σειριακή υλοποίηση, η μνήμη διαγραφής στέλνει ένα στοιχείο στην μονάδα ελέγχου, το οποίο ανάλογα με την τιμή του οδηγείται στην κατάλληλη TBS και εκεί, αν η Binary Search Unit είναι διαθέσιμη, ξεκινάει η αναζήτηση της θέσης του. Όταν αυτή βρεθεί, σημειώνεται (κάνουμε το αντίστοιχο bit εύρεσης του στοιχείου '1', ώστε στη συνέχεια να διαγράψουμε αυτό μαζί με τα δείγματά του), και πλέον η μονάδα ελέγχου είναι έτοιμη να ζητήσει από τη μνήμη διαγραφής ένα νέο στοιχείο.

Αντίθετα, σκοπός της παράλληλης υλοποίησης της μονάδας Search_All είναι, στην ιδανική περίπτωση, τα στοιχεία της μνήμης να διαβάζονται σε διαδοχικούς κύκλους ρολογιού και να γίνονται 49 παράλληλες

αναζητήσεις. Πρακτικά, κάθε γονίδιο που πρέπει να διαγραφεί οδηγείται στην αντίστοιχη μονάδα Binary Search, αμέσως αφού το προηγούμενο γονίδιο της λίστας οδηγηθεί στη δική του μονάδα Binary Search.

Η μόνη περίπτωση να διακοπεί η παράλληλη αναζήτηση είναι ένα στοιχείο διαγραφής να σταλεί σε μία TBS που είναι απασχολημένη (δηλαδή, η μονάδα Binary Search εκτελεί αναζήτηση για προηγούμενο κλειδί, και η θέση αναμονής (“wait_key”) είναι ήδη κατειλημμένη). Σε μία τέτοια περίπτωση, για να αποφευχθεί η παράληψη αναζήτησης του στοιχείου, η μονάδα Search_All “κλειδώνει” (“locks”), δηλαδή διακόπτεται η ανάγνωση νέων στοιχείων από τη μνήμη διαγραφής. Όταν η συγκεκριμένη TBS πάψει να είναι απασχολημένη (δηλαδή, ολοκληρωθεί η αναζήτηση, αρχίσει νέα, με παράλληλο άδειασμα της θέσης “wait_key”), τότε αίρεται το κλείδωμα της μονάδας (“unlocks”), το γονίδιο που προκάλεσε το lock καταλαμβάνει τη θέση αναμονής της αντίστοιχης μονάδας TBS που μόλις ελευθερώθηκε, και η ανάγνωση στοιχείων διαγραφής συνεχίζεται κανονικά από τη θέση που είχε σταματήσει.

Τελικά, προκειμένου η σχεδίαση να εκμεταλλευτεί όσο το δυνατόν περισσότερο τους πόρους μιας FPGA, στην τελική υλοποίηση χρησιμοποιήθηκαν τέσσερα αντίγραφα μίας μονάδας Search_All, όπως παρουσιάστηκε και στο σχήμα 4.15. Κάθε μονάδα αναλαμβάνει το ¼ των συνολικών αναζητήσεων. Μόλις και οι τέσσερεις μονάδες ολοκληρώσουν τις αναζητήσεις που τους αντιστοιχούν, η κεντρική μονάδα ελέγχου (top unit control) εκτελεί το τελικό βήμα της υλοποίησης. Διασχίζοντας μία φορά όλα τα γονίδια, αντιγράφει όσα από αυτά δεν έχουν εντοπιστεί (δηλαδή, δεν βρίσκονται στην λίστα διαγραφής) μαζί με τα δείγματά τους, στην εξωτερική μνήμη εξόδου.

Μόλις ολοκληρωθεί και αυτή η διαδικασία, η εξωτερική μνήμη εξόδου μπορεί να θεωρείται νέα αρχική λίστα γονιδίων, και ο αλγόριθμος κατηγοριοποίησης καρκινογόνων γονιδίων με χρήση των SVMs να αρχίσει την εκτέλεση μίας νέας ρουτίνας.

6

ΑΠΟΔΟΣΗ ΣΥΣΤΗΜΑΤΟΣ ΠΟΛ/ΣΜΟΥ ΠΙΝΑΚΩΝ – ΣΥΓΚΡΙΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

6.1 Εισαγωγή

Στην παρούσα ενότητα, αρχικά, παρουσιάζονται αποτελέσματα σύνθεσης και Pace&Route της σχεδίασης παράλληλου πολ/σμου πινάκων. Στη συνέχεια, γίνεται παρουσίαση των μεθόδων που χρησιμοποιήθηκαν για την επαλήθευση των ξεχωριστών τμημάτων της σχεδίασης. Επίσης, παρουσιάζονται τελικά αποτελέσματα και συγκρίνονται με τα αντίστοιχα αποτελέσματα των συστημάτων που εκτελούνται σε CPU 2,66 GHz.

6.2 Αποτελέσματα Σύνθεσης και Place&Route

Καταρχήν, για την ανάπτυξη, τη σύνθεση και την υλοποίηση της σχεδίασης χρησιμοποιήθηκε το εργαλείο Xilinx ISE 10.1i και για την επαλήθευση της σωστής λειτουργίας το εργαλείο ModelSim 6.2c. Η σχεδίαση προσομοιώθηκε στην πλακέτα “XC5VSX240T”, με ταχύτητα

συσκευής -2, της οικογένειας Virtex 5. Ολοκληρώνοντας την υλοποίηση, μετρήθηκαν το χωρικό κόστος και η απόδοσή της με χρήση των εφαρμογών Synthesis και Place&Route του ISE. Είναι χρήσιμο να αναφερθεί ότι η μονάδα μέτρησης του χωρικού κόστους είναι τα Slice LUT's και Logic Cells (LC's). Ο αριθμός των Slice LUT's είναι ο αριθμός των Reported Slices πολλαπλασιασμένος με έναν παράγοντα ίσο με 4. Επίσης, ένα τμήμα BRAM (που αποτελείται από ένα block 36-Kbit) είναι η μονάδα μέτρησης της μνήμης που χρησιμοποιήθηκε. Τέλος για το χρονικό κόστος της σχεδίασης, από το βήμα Place&Route παράγεται η περίοδος κύκλου ρολογιού και η αντίστοιχη συχνότητα.

Πρέπει να σημειωθεί ότι δεν είναι δυνατόν να περάσει από τη διαδικασία της σύνθεσης ολόκληρη η σχεδίαση, καθώς το μοντέλο των εξωτερικών μνημών είναι behavioral. Έτσι, όλη η υπόλοιπη σχεδίαση εκτός των SRAMs περνάει από διαδικασίες Synthesis και Place&Route, προβλέποντας την περίοδο ρολογιού της σχεδίασης. Αυτό είναι εφικτό, καθώς οι SRAMs βρίσκονται εκτός του chip της FPGA.

Τελικά, για τη μονάδα πολλαπλασιασμού πινάκων έχουμε:

Number of Slice Registers (out of 149760)	69457 (46%)
Number of Slice LUTs (out of 149760)	58951 (39%)
Number used as Logic (out of 149760)	58951 (39%)

Εικόνα 6.1: Slice Logic Utilization

Number of Bit Slices used	79258
Number with an unused Flip Flop	9801 (12%)
Number with an unused LUT	20307 (25%)
Number of fully used Bit Slices	49150 (62%)

Εικόνα 6.2: Slice Logic Distribution

Number of Block RAM/FIFO (out of 516)	40 (7%)
Number of BUFG/BUFGCTRLs (out of 32)	16 (50%)
Number of DSP48Es (out of 1056)	771 (73%)

Εικόνα 6.3: Specific Feature Utilization

Minimum period (ns)	2.75
Maximum Frequency (MHz)	364

Εικόνα 6.4: Timing Summary

Εδώ πρέπει να σημειωθεί, ότι οι διαδικασίες Synthesis και Place&Route είναι αρκετά χρονοβόρες για μεγάλες σχεδιάσεις, και καταναλώνουν πολλούς υπολογιστικούς πόρους. Πρέπει, λοιπόν, να αναφερθεί η αδυναμία μας περάσουμε από σύνθεση τη σχεδίαση που για την υλοποίηση των πολλαπλασιαστών θα χρησιμοποιούσε, αντί για DSPs, LUTs (η σύνθεση κάθε πολλαπλασιαστή ξεχωριστά στα LUTs, απαιτούσε πολύ περισσότερο χρόνο και υπολογιστική ισχύ από τη σύνθεση μίας έτοιμης μονάδας πολλαπλασιασμού από τα DSPs).

Για να έχουμε, σε θεωρητικό τουλάχιστον επίπεδο, ένα μέτρο σύγκρισης μεταξύ υλοποίησης με χρήση πολλαπλασιαστών σε LUTs σε DSPs, σχεδιάζουμε ένα μικρό τμήμα των πολλαπλασιαστών με LUTs (με χρήση της μονάδας multiplier v10, που παράγει το Core Generator του ISE).

Παρατηρούμε ότι η περίοδος ρολογιού του πολλαπλασιασμού αυξάνεται κατά 130ps, ενώ για κάθε μονάδα πολλαπλασιαστή χρησιμοποιούνται 1088 slice LUTs. Επομένως, εάν θέλαμε να χρησιμοποιήσουμε λιγότερα DSPs, με παραπλήσια περίοδο ρολογιού, στη σχεδίαση θα μπορούσαν να χρησιμοποιηθούν μέχρι και 90 πολλαπλασιαστές υλοποιημένους σε LUTs στη συγκεκριμένη πλακέτα.

6.3 Επαλήθευση σωστής λειτουργίας

Με σκοπό την επαλήθευση της σωστής λειτουργικότητας της σχεδίασης, αρχικά ελέγχθηκε και προσομοιώθηκε η λειτουργία κάθε μονάδας μεμονωμένα και στη συνέχεια, με πολλαπλασιασμούς πινάκων ποικίλων διαστάσεων και δεδομένων οριακών συνθηκών, επαληθεύτηκε η ορθή λειτουργία όλης της σχεδίασης.

Το μοντέλο της SRAM που χρησιμοποιήθηκε παρέχεται από την Cypress και επομένως η λειτουργία του θεωρείται ότι έχει ήδη επαληθευτεί. Παρόλα αυτά, για λόγους διασφάλισης της εγκυρότητας της σχεδίασης, αρκετά testbenches δημιουργήθηκαν και εξέτασαν στο ModelSim 6.2c τις περισσότερες περιπτώσεις ανάγνωσης και εγγραφής σε εξωτερική μνήμη. Έτσι, διάφορα testbenches έλεγξαν τη δυνατότητα των μνημών να γράφουν και να διαβάζουν τυχαίες διευθύνσεις, να διαβάζουν άδεια αρχεία, καθώς και να προσπελάζουν δεδομένα από όλες τις θέσεις μνήμης του μοντέλου.

Επίσης, για τις μονάδες παράλληλης ανάγνωσης και εγγραφής σε SRAMs ελέγχθηκε ο σωστός τρόπος ανάγνωσης δεδομένων μεγέθους [64,64], από τις SRAMs και εγγραφής σε BRAMs και αντίστοιχα, ανάγνωσης από BRAMs και εγγραφής σε SRAMs. Αφού εντοπίστηκαν

τα σημεία στα οποία καθορίζεται ο αριθμός των παράλληλων προσπελάσεων, και δημιουργήθηκαν υλοποιήσεις για χρήση 1, 2, 4, 8 και 16 μνημών για κάθε πίνακα, ελέγχθηκε η σωστή προσπέλαση μνημών για κάθε μία από τις παραπάνω κατηγορίες.

Επίσης, αντίστοιχοι έλεγχοι πραγματοποιήθηκαν για τις μονάδες ανάγνωσης των BRAMs και εγγραφής των Buffers καθώς και ανάγνωσης των Buffers και εγγραφής των BRAMs, όπου οι πίνακες [64,64] προσπελάζονται με buffers [16,16].

Για τη μονάδα πολλαπλασιασμού υλοποιήθηκαν testbenches που έλεγχαν πολλές περιπτώσεις για πράξεις θετικών και αρνητικών 32-bit αριθμών σε μορφή συμπληρώματος ως προς δύο, σταθερής υποδιαστολής. Το γεγονός ότι τα αποτελέσματα των πολλαπλασιασμών της υλοποίησης δεν παράγουν ποτέ στοιχείο που να ξεπερνά κατά απόλυτη τιμή το 65535, μας διασφαλίζει ότι για προσημασμένους αριθμούς με 17-bit ακέραιο μέρος, δε θα παρουσιαστεί στην σχεδίασή μας υπερχείλιση κατά την εκτέλεση των πολλαπλασιασμών πινάκων.

Συνολικά, η σχεδίαση ελέγχθηκε με υλοποίηση πολλών testbenches που δοκίμαζαν πολλαπλασιασμούς πινάκων πολλών ειδών (πολλαπλασιασμοί με ποικίλες διαστάσεις πινάκων, με χρήση αναστροφής του A και μη, κλπ.). Για κάθε ξεχωριστό πολλαπλασιασμό, παράγονταν τυχαίοι πίνακες από τη Matlab. Τα στοιχεία των πινάκων μεταφέρονταν στις SRAMs μέσω αρχείων και εκτελώντας τον πολλαπλασιασμό στη Matlab, επαληθευόταν ότι προκύπτουν τα ίδια δεδομένα τόσο από την προσομοίωση στο Modelsim όσο και από τον πολλαπλασιασμό στη Matlab.

Ένα σημείο στο οποίο έπρεπε να δοθεί προσοχή, ήταν η επαλήθευση ότι αρκεί η αποτύπωση ενός στοιχείου του πίνακα με έναν προσημασμένο αριθμό 32 bit, σταθερής υποδιαστολής, με 17 bits ακέραιο μέρος και 15 bits δεκαδικό μέρος. Έτσι, συγκρίνοντας κάποια αποτελέσματα της δικιάς μας μονάδας με αυτά στην υλοποίηση σε C, διαπιστώνουμε ότι διατηρούμε ακρίβεια αποτελεσμάτων μέχρι και τέταρτου δεκαδικού ψηφίου. Η προσέγγιση αυτή καλύπτει τις απαιτήσεις του συστήματος, καθώς οι συγκρίσεις για την εξαγωγή των τελικών αποτελεσμάτων στο σύστημα "Face Identification Evaluation System" γίνονται με προσέγγιση εκατοστού. Βέβαια, πιο ακριβή αποτελέσματα στο συγκεκριμένο θέμα, μπορούν να εξαχθούν εάν η μονάδα μας ενσωματωθεί πλήρως με το σύστημα, κάτι που αναφέρεται στις μελλοντικές εργασίες.

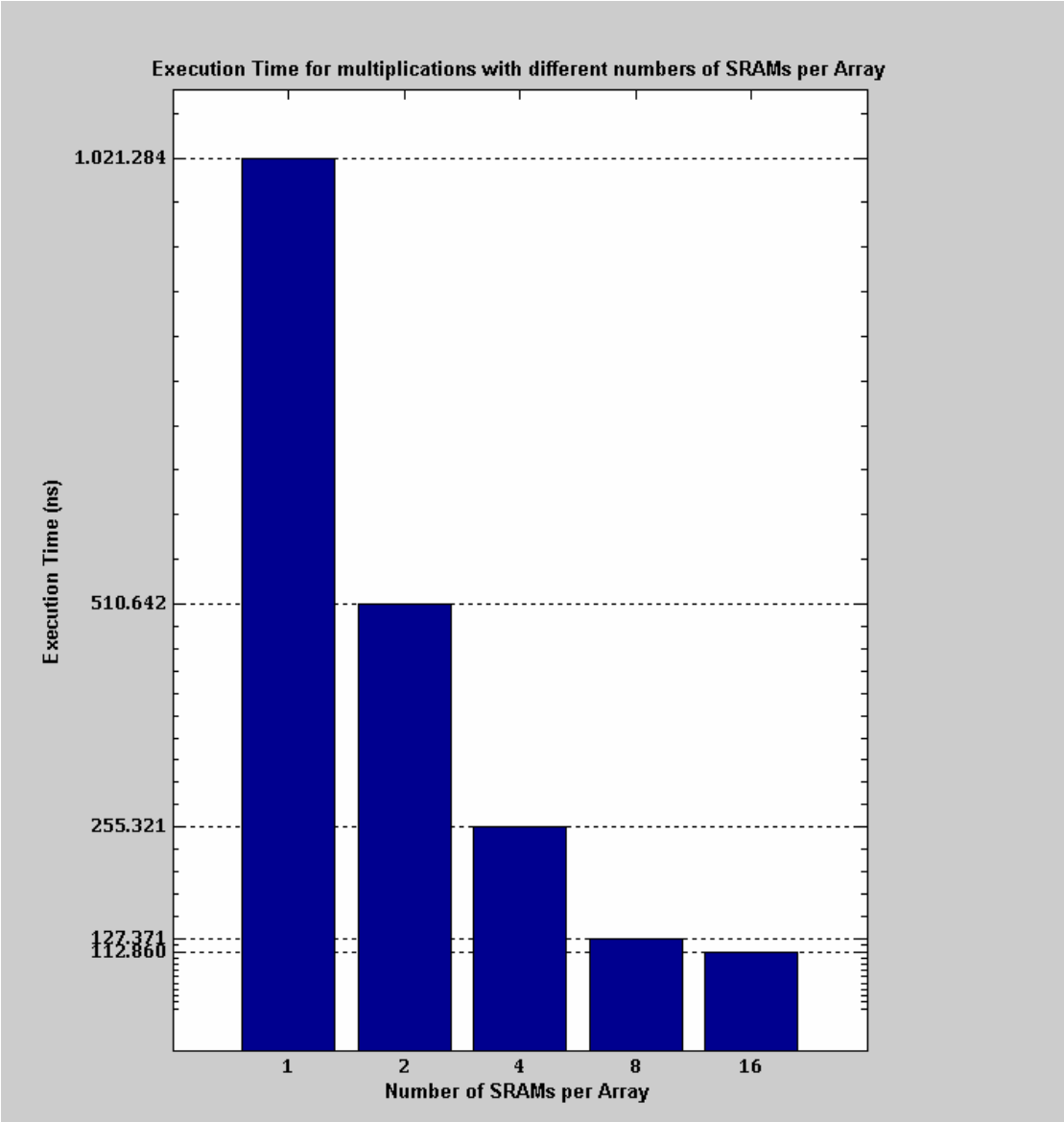
6.4 Παρουσίαση συγκριτικών αποτελεσμάτων

Στην ενότητα αυτή θα παρουσιαστούν αποτελέσματα απόδοσης της βασικής μας σχεδίασης σε σχέση τόσο με άλλες δικές μας υλοποιήσεις, όσο και με αυτά που παράγονται από πολλαπλασιασμούς πινάκων σε μία μονάδα Pentium 4 CPU 2.6 GHz.

Πριν παρουσιαστούν τα αποτελέσματα είναι σημαντικό να αναφερθεί ότι η προσομοίωση στο Modelsim ενός συστήματος που χρησιμοποιεί μεγάλες εξωτερικές μνήμες (πίνακες με εκατομμύρια στοιχεία) είναι μία πολύ χρονοβόρα διαδικασία. Είναι ενδεικτικό ότι για έναν μόνο πολλαπλασιασμό $[19500,100]^T$ x [1950,100] χρειάστηκαν 3 εικοσιτετράωρα συνεχούς προσομοίωσης για να ολοκληρωθεί μόλις το 5% της πράξης του πολλαπλασιασμού πινάκων. Είναι προφανές ότι η προσπάθεια πλήρους προσομοίωσης κάποιων από τους πίνακες που

χρησιμοποιούνται στο σύστημα “CSU Face Identification Evaluation System” θα διαρκούσε αρκετούς μήνες. Για τον παραπάνω λόγο εκτός από κάποιους πολλαπλασιασμούς πινάκων τυχαίων διαστάσεων (π.χ. [94,200] x [200,112]), που χρησιμοποιήθηκαν ως δείγματα για κάποιες μετρήσεις, οι πολλαπλασιασμοί που γίνονται στο σύστημα “CSU Face Identification Evaluation System” (και των οποίων η βελτιστοποίηση του χρόνου εκτέλεσης ήταν σκοπός της μονάδας μας) απλά εκτελούνται μέχρι να παράγουν κάποια πρώτα αποτελέσματα και στη συνέχεια αναγκαστικά τους διακόπτουμε. Επειδή, όμως, ο πολλαπλασιασμός δύο πινάκων βασίζεται στην επανάληψη της ίδιας διαδικασίας, (πολλαπλασιασμός υποπινάκων [64,64]), ο συνολικός χρόνος εκτέλεσης ενός πολλαπλασιασμού μπορεί να εκτιμηθεί, με βάση τις διαστάσεις των δύο πινάκων.

Παρακάτω παρουσιάζεται ο χρόνος εκτέλεσης ενός πολλαπλασιασμού [94,200] x [200,112] με χρήση διαφορετικού αριθμού από SRAMs κάθε φορά. Παρατηρούμε ότι ενώ για χρήση μνημών από 1 μέχρι 8, ο χρόνος εκτέλεσης του πολλαπλασιασμού μειώνεται στο μισό, ο χρόνος εκτέλεσης για επιλογή 8 και 16 μνημών είναι παραπλήσιος. Για να εξηγήσουμε αυτό το γεγονός, αρκεί να δούμε τους χρόνους περιόδου της βασικής ομοχειρίας για κάθε περίπτωση (εικόνα 5.6). Από τον πίνακα της εικόνας είναι φανερό ότι από 1 μέχρι 8 εξωτερικές μνήμες, κριτήριο για το ποια θα είναι η περίοδος της βασικής ομοχειρίας της σχεδίασης είναι ο χρόνος εγγραφής και ανάγνωσης 4096 λέξεων προς και από τις εξωτερικές μνήμες ($1^{η}$ και $3^{η}$ βαθμίδα του pipeline). Αντίθετα, στην περίπτωση των 16 μνημών κριτήριο αποτελεί η $2^{η}$ βαθμίδα, δηλαδή ο πολλαπλασιασμός υποπινάκων [64,64].



Εικόνα 6.5: Παρουσίαση χρόνων εκτέλεσης πολλαπλασιασμού [94,200] x [200,112] σε σχέση με τον αριθμό των SRAMs που χρησιμοποιείται για κάθε πίνακα.

Αριθμός μνημών	**Ελάχιστος Χρόνος στην 1^{ο} Βαθμίδα (ns)**	**Ελάχιστος Χρόνος στην 2^{ο} Βαθμίδα (ns)**	**Ελάχιστος Χρόνος στην 3^{ο} Βαθμίδα (ns)**	**Χρόνος Περιόδου κάθε Βαθμίδας (ns)**
1 SRAM	56.738	6.270	56.738	56.738
2 SRAMs	28.369	6.270	28.369	28.369
4 SRAMs	14.184	6.270	14.184	14.184
8 SRAMs	7.092	6.270	7.092	7092
16 SRAMs	3.546	6.270	3.546	6.270

Εικόνα 6.6: Παρουσίαση χρόνου για κάθε βαθμίδα της βασικής ομοχειρίας της σχεδίασης ανάλογα με τον αριθμό εξωτερικών μνημών για κάθε πίνακα

Για το λόγο ότι με τη χρήση 8 μνημών οι βαθμίδες του pipeline έχουν ισοσταθμισμένους χρόνους εκτέλεσης, αλλά και γιατί μειώνει το κόστος της σχεδίασης (παρόλο που η χρήση των 16 SRAMs κάνει λίγο πιο γρήγορη την εφαρμογή) επιλέξαμε αυτή ως βασική σχεδίαση.

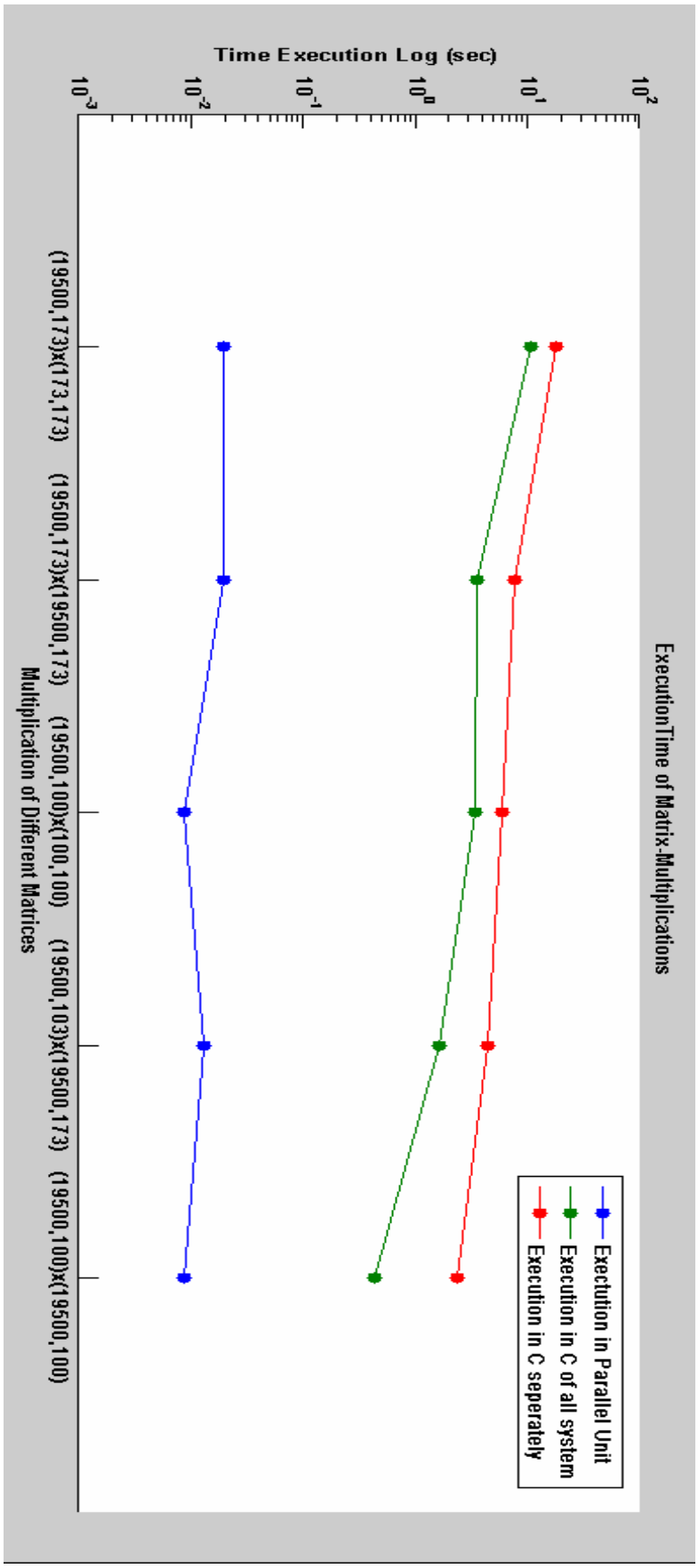
Στη συνέχεια, παρουσιάζονται αποτελέσματα για τους μεγαλύτερους και πιο αργούς (σε χρόνο εκτέλεσης) πολλαπλασιασμούς που χρησιμοποιούνται στο "CSU Face Identification Evaluation System". Οι μετρήσεις των χρόνων εκτέλεσης των πολλαπλασιασμών που υλοποιούνται σε C έγιναν με τη βοήθεια του προγράμματος VTUNE της Intel. Πρέπει να σημειωθεί, ότι όταν προσπαθήσαμε να μετρήσουμε το χρόνο εκτέλεσης για έναν πολλαπλασιασμό υλοποιημένο σε ξεχωριστό, δικό του εκτελέσιμο αρχείο, διαπιστώσαμε ότι ο χρόνος που μετρήσαμε ήταν αρκετά μεγαλύτερος από τον αντίστοιχο χρόνο που βρίσκαμε όταν εκτελούσαμε όλο το σύστημα μαζί (που συνεπάγεται εκτέλεση πολλών πολλαπλασιασμών). Αυτό συμβαίνει γιατί, πέρα του ότι ο επεξεργαστής Pentium 4 έχει δύο μονάδες ALU, κατά την εκτέλεσή του χρησιμοποιείται βελτιστοποιημένος compiler. Επίσης επειδή η υλοποίηση των πινάκων στο σύστημα "CSU Face Identification Evaluation System" έχουν υλοποιηθεί με βάση την προτεραιότητα γραμμής, παρατηρούμε ότι πολλαπλασιασμοί με ανάστροφη χρήση του A (π.χ. $[19500,100]^T$x[19500,100]), εκτελούνται πιο γρήγορα από πολλαπλασιασμούς πινάκων που απαιτούν τον ίδιο αριθμό πολλαπλασιασμών αλλά δεν χρησιμοποιούν τον ανάστροφο του A (π.χ. [19500,100] x[100,100]).

Στον πίνακα της εικόνας 6.7 βλέπουμε τα συγκριτικά αποτελέσματα των χρόνων εκτέλεσης των πολλαπλασιασμών όταν προσομοιώνονται στην

παράλληλη μονάδα μας (μπλε γραμμή), όταν εκτελούνται μαζί με το υπόλοιπο σύστημα σε έναν επεξεργαστή Pentium 4 (2.6 GHz) και όταν εκτελείται κάθε πολλαπλασιασμός με διαφορετικό εκτελέσιμο σε Pentium 4 (2.6 GHz). Τα αποτελέσματα αυτά παρουσιάζονται με τη μορφή γραφήματος στην εικόνα 6.8.

Μορφή Πολλαπλασιασμού	**Χρόνος Εκτέλεσης (εκτιμώμενος) στην Παράλληλη Μονάδα (sec)**	**Χρόνος Εκτέλεσης με πολλούς σειριακούς πολλαπλασιασμούς (sec)**	**Χρόνος Εκτέλεσης με έναν πολλαπλασιασμό (sec)**
(19500,173)x(173,173)	0.019482410	10.591	18.128
(19500,173)x(19500,173)	0.019482410	3.517	7.640
(19500,100)x(100,100)	0.008659637	3.413	6.035
(19500,103)x(19500,173)	0.012985909	1.662	4.359
(19500,100)x(19500,100)	0.008659637	0.427	2.375

Εικόνα 6.7: Πίνακας που παρουσιάζει το χρόνο εκτέλεσης (σε δευτερόλεπτα) των πολλαπλασιασμών με τρεις διαφορετικές μετρήσεις. Η πρώτη αντιστοιχεί στον εκτιμώμενο χρόνο εκτέλεσης της μονάδας μας, η δεύτερη όταν εκτελούνται μαζί σε γλώσσα C πάνω από ένας σειριακός πολλαπλασιασμός, και η τρίτη όταν εκτελείται ξεχωριστά κάθε πολλαπλασιασμός σε γλώσσα C.



Εικόνα 6.8: Σχηματικό διάγραμμα που παρουσιάζει το χρόνο εκτέλεσης σε λογαριθμική κλίμακα μεγάλων διανυσμάτων

7

ΑΠΟΔΟΣΗ ΣΥΣΤΗΜΑΤΟΣ ΔΥΑΔΙΚΗΣ ΑΝΑΖΗΤΗΣΗΣ – ΣΥΓΚΡΙΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

7.1 Εισαγωγή

Αντίστοιχα με στο κεφάλαιο 6, παρουσιάζονται αποτελέσματα σύνθεσης και Pace&Route της σχεδίασης παράλληλου πολ/σμου πινάκων. Στη συνέχεια, γίνεται παρουσίαση των μεθόδων που χρησιμοποιήθηκαν για την επαλήθευση των ξεχωριστών τμημάτων της σχεδίασης. Τέλος, παρουσιάζονται τελικά αποτελέσματα και συγκρίνονται με τα αντίστοιχα αποτελέσματα των συστημάτων που εκτελούνται σε CPU 2,66 GHz.

7.2 Αποτελέσματα Σύνθεσης και Place&Route

Καταρχήν, για την ανάπτυξη, τη σύνθεση και την υλοποίηση της σχεδίασης χρησιμοποιήθηκε το εργαλείο Xilinx ISE 9.1i και για την επαλήθευση της σωστής λειτουργικότητας το εργαλείο ModelSim 6.2c. Η σχεδίαση προσομοιώθηκε στην πλακέτα “XC5VLX110T”, με

ταχύτητα συσκευής -2, της οικογένειας Virtex 5. Ολοκληρώνοντας την υλοποίηση, μετρήθηκαν το χωρικό κόστος και η απόδοσή της με χρήση των εφαρμογών Synthesis και Place&Route του ISE. Ο λόγος για τον οποίο χρησιμοποιήθηκαν 4 μονάδες Search_all στην τελική σχεδίαση είναι ότι με αυτόν τον αριθμό, χρησιμοποιούνται σχεδόν όλες οι BRAMs της πλακέτας. Οι μονάδες μέτρησης του χωρικού και χρονικού κόστους είναι ίδιες με αυτές που αναφέρθηκαν στην αντίστοιχη μονάδα πολλαπλασιασμού πινάκων.

Πρέπει να σημειωθεί ότι δεν είναι δυνατόν να περάσει από τη διαδικασία της σύνθεσης ολόκληρη η σχεδίαση, καθώς το μοντέλο των εξωτερικών μνημών είναι behavioral. Έτσι, όλη η υπόλοιπη σχεδίαση εκτός των SRAMs περνάει από διαδικασίες Synthesis και Place&Route, προβλέποντας την περίοδο ρολογιού της σχεδίασης. Αυτό είναι εφικτό, καθώς οι SRAMs βρίσκονται εκτός του chip της FPGA.

Τελικά, για τη μονάδα δυαδικής αναζήτησης πινάκων έχουμε:

Number of Slice Registers (out of 69120)	20741 (30%)
Number of Slice LUTs (out of 69120)	53351 (77%)
Number used as Logic (out of 69120)	50411 (39%)
Number used as Memory (out of 17920)	2940 (16%)

Εικόνα 7.1: Slice Logic Utilization

Number of Bit Slices used	57118
Number with an unused Flip Flop	36377 (63%)
Number with an unused LUT	3767 (6%)
Number of fully used Bit Slices	16974 (29%)

Εικόνα 7.2: Slice Logic Distribution

Number of Block RAM/FIFO (out of 148)	138 (93%)
Number of BUFG/BUFGCTRLs (out of 32)	16 (50%)

Εικόνα 7.3: Specific Feature Utilization

Minimum period (ns)	4,20
Maximum Frequency (MHz)	238

Εικόνα 7.4: Timing Summary

7.3 Επαλήθευση σωστής λειτουργίας

Με σκοπό την επαλήθευση της σωστής λειτουργικότητας της σχεδίασης, αρχικά ελέγχθηκε και προσομοιώθηκε η λειτουργία κάθε μονάδας μεμονωμένα και στη συνέχεια επαληθεύτηκε η ορθή λειτουργία όλης της σχεδίασης.

Το μοντέλο της SRAM που χρησιμοποιήθηκε παρέχεται από την Cypress και επομένως η λειτουργία του θεωρείται ότι έχει ήδη επαληθευτεί. Παρόλα αυτά, όπως και στην περίπτωση της μονάδας πολλαπλασιασμού πινάκων, αρκετά testbenches δημιουργήθηκαν και εξέτασαν στο ModelSim 6.2c τις περισσότερες περιπτώσεις ανάγνωσης και εγγραφής σε εξωτερική μνήμη.

Για τις μονάδες Binary Search ελέγχθηκε ο σωστός τρόπος εκτέλεσης της αναζήτησης. Με ειδικά testbenches εξετάστηκαν ακραίες περιπτώσεις (αναζήτηση πρώτου ή τελευταίου στοιχείου, αναζήτηση στοιχείου που δεν είναι στη μνήμη κλπ). Στη συνέχεια, ελέγχθηκε η διασύνδεση της μονάδας με την TBS, ώστε να επιβεβαιωθεί η σωστή

απόκριση της μονάδας για περιπτώσεις διαχειρισμού πολλών αιτημάτων αναζήτησης.

Για τη σχεδίαση της Search_all ελέγχθηκε η δυνατότητα εύρεσης όλων των στοιχείων που βρίσκονται στη λίστα της μνήμης διαγραφής. Δόθηκε ιδιαίτερο βάρος στην επιβεβαίωση σωστής λειτουργίας των locks καθώς είναι η πιο σύνθετη διεργασία της μονάδας. Ειδικά testbenches στοχευμένα στην δημιουργία αλλεπάλληλων κλειδωμάτων της μονάδας, έδειξαν την ορθή εξυπηρέτηση όλων των αιτημάτων lock και unlock.

Η κεντρική μονάδα ελέγχθηκε για τη μεταφορά των γονιδίων στις BRAMs, τον ορθό διαμοιρασμό των αναζητήσεων στις τέσσερεις μονάδες Search_all καθώς και για τη διαδικασία ανάγνωσης δεδομένων από την αρχική SRAM και εγγραφής των σωστών αποτελεσμάτων στην SRAM τελικών αποτελεσμάτων. Testbenches που περιείχαν υποσύνολα των αρχικών δεδομένων, επιβεβαίωσαν την εξαγωγή σωστών αποτελεσμάτων.

Για την αποτύπωση κάθε χαρακτηριστικής τιμής των γονιδίων ήταν αρκετά 15 bits. Ωστόσο για την περιγραφή τους χρησιμοποιήθηκε ακόμα ένα bit το οποίο το ερμήνευε εάν το στοιχείο έχει εντοπιστεί ή όχι. Έτσι, για την περιγραφή κάθε στοιχείου χρησιμοποιήθηκαν 16 bits.

7.4 Παρουσίαση συγκριτικών αποτελεσμάτων

Όπως αναφέρθηκε παραπάνω, σε περίπτωση μη χρησιμοποίησης της παράλληλης μονάδας δυαδικής αναζήτησης, όλες οι αναζητήσεις θα εκτελεστούν σειριακά, αυξάνοντας το μέσο χρόνο μίας δυαδικής αναζήτησης κατά 2/3. Υποθέτοντας την ιδανική περίπτωση, ότι η ροή

δεδομένων εισόδου από τη μνήμη διαγραφής είναι οργανωμένη με τέτοιο τρόπο, ώστε να μην εμφανίζονται δομικοί κίνδυνοι (εννοώντας ότι δε χρησιμοποιούνται οι λειτουργίες των lock-unlock που καθυστερούν τον χρόνο εκτέλεσης) μία νέα δυαδική αναζήτηση θα ξεκινούσε κάθε 2 κύκλους ρολογιού σε κάθε μονάδα Search_all. Κατά τη διάρκεια αυτών των κύκλων ένα νέο γονίδιο διαβάζεται από τη μνήμη διαγραφής και στέλνεται στην αντίστοιχη TBS (αφού γίνει ο απαραίτητος έλεγχος ότι η μονάδα δεν είναι απασχολημένη). Επομένως, είναι φανερό ότι για να προσεγγιστεί αυτή η ιδανική περίπτωση έπρεπε να αποφευχθούν τα locks.

Η πρώτη τεχνική που εφαρμόστηκε για το σκοπό αυτό, είχε να κάνει με τον περιορισμό συνεχόμενων αιτημάτων για αναζήτηση στην ίδια TBS (που σημαίνει ότι γονίδια πρέπει να αναζητηθούν στην μονάδα Binary Search που χρησιμοποιεί την ίδια Block RAM). Χρησιμοποιώντας πολλές ξεχωριστές TBS μονάδες, η πιθανότητα να υπάρχουν συνεχόμενες αιτήσεις αναζήτησης για την ίδια μονάδα μειώθηκε σημαντικά.

Η δεύτερη τεχνική που χρησιμοποιήθηκε ήταν η χρήση ενός "wait_key" σε κάθε TBS. Όπως είναι προφανές, και αποδεικνύεται και πειραματικά παρακάτω, η πιθανότητα εμφάνισης την ίδια περίοδο πάνω από δύο γονιδίων για αναζήτηση στην ίδια μονάδα είναι αρκετά μικρή.

Συνολικά, προσομοιώνοντας τη σχεδίασή μας, διαπιστώσαμε ότι χρησιμοποιώντας 49 μνήμες και μία θέση buffer σε κάθε TBS, ο μέσος χρόνος για την εκκίνηση μίας νέας αναζήτησης σε κάθε μονάδα Search_all φτάνει τους 2.16 κύκλους (αυτή η μέτρηση υπολογίστηκε διαιρώντας το συνολικό χρόνο εκτέλεσης των αναζητήσεων με τον

αριθμό των στοιχείων που αναζητήθηκαν), που μπορεί να θεωρηθεί αρκετά καλή προσέγγιση της ιδανικής περίπτωσης. Αντίθετα, δοκιμάζοντας μία παρόμοια σχεδίαση με 25 TBS μονάδες και μία θέση "wait_key" για κάθε μία από αυτές, ο μέσος χρόνος για την εκκίνηση μίας νέας αναζήτησης σε κάθε μονάδα Search_all φτάνει τους 5.13 κύκλους, ενώ για μία σχεδίαση με 25 μονάδες TBS και χωρίς τη χρήση των θέσεων αναμονής, ο χρόνος αυτός αυξάνεται στους 9.58 κύκλους.

Συγκεντρωτικά, ενώ ο χρόνος της παράλληλης υλοποίησης του αλγόριθμου είναι 4.5 msec, ενώ ο χρόνος εκτέλεσης της συνάρτησης "RemoveGenesBinary" υλοποιημένης σε Matlab είναι 3.2 sec, όπως αναμέναμε η παράλληλη σχεδίαση επιτυγχάνει μία πολλή σημαντική βελτιστοποίηση στο χρόνο εκτέλεσης του αλγορίθμου, που εκφράζεται με ένα speedup που αγγίζει την τιμή 711.

8

ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

8.1 Εισαγωγή

Στο κεφάλαιο αυτό αναφέρονται κάποιες επεκτάσεις ή βελτιστοποιήσεις που μπορούν να γίνουν πάνω στις υλοποιήσεις ώστε να επεκταθεί η χρήση τους.

8.2 Μελλοντική δουλειά

Για τη μονάδα παράλληλου πολλαπλασιασμού επεκτάσεις που προτείνονται είναι :

- Σύνδεση της μονάδας με επεξεργαστή ώστε να υλοποιηθεί ένα πλήρως ενσωματωμένο σύστημα αξιολόγησης αλγορίθμων αναγνώρισης προσώπου. Για τη πλήρη απαλοιφή των απωλειών προτείνεται η χρήση λέξεων 64-bit κινητής υποδιαστολής, αντί για 32-bit σταθερής υποδιαστολής.

- Χρήση αποτελεσματικότερων μονάδων ελέγχου μεταφοράς δεδομένων από εξωτερικές μνήμες (π.χ. DMA), ώστε να είναι ταχύτερη η μεταφορά των δεδομένων στο chip της FPGA.

- Επειδή η υλοποίηση εστίασε στη βελτιστοποίηση χρόνων πολλαπλασιασμού πινάκων ιδιαίτερα μεγάλων διαστάσεων, σε περίπτωση που εισαχθούν ως είσοδοι μικροί πίνακες, οι απώλειες για τους αντίστοιχους πολλαπλασιασμούς θα είναι σημαντικές. Εάν είναι επιθυμητό κάτι τέτοιο, προτείνεται η υλοποίηση παραμετροποίησης των υποπινάκων καθώς και forwarding για περιπτώσεις όπου οι διαστάσεις των πινάκων είναι μικρότερες από [64,64].

Για τη μονάδα παράλληλης δυαδικής αναζήτησης μελλοντικές εργασίες θα μπορούσαν να είναι:

- Χρήση αποτελεσματικότερων μονάδων ελέγχου μεταφοράς δεδομένων από εξωτερικές μνήμες (π.χ. DMA), ώστε να είναι ταχύτερη η μεταφορά των δεδομένων στο chip της FPGA.

- Χρήση μεγαλύτερων εξωτερικών μνημών για χρησιμοποίηση της μονάδας και σε μεγαλύτερες βάσεις δεδομένων γονιδίων, όπου η ανάγκη βελτιστοποίησης θα είναι εντονότερη.

- Σύνδεση της μονάδας με επεξεργαστή ώστε η μονάδα της δυαδικής αναζήτησης να αποτελέσει μέρος ενός συστήματος γονιδιακής επιλογής καρκινικών κυττάρων με χρήση SVMs.

9

ΒΙΒΛΙΟΓΡΑΦΙΑ

(1) Official website of Xilinx (www.xilinx.com)

(2) Official website of Cypress (www.cypress.com)

Για τη μονάδα δυαδικής αναζήτησης:

(3) Guyon, J. Weston, S. Barnhill and V. Vapnik. "Gene selection for cancer classification using SVM". Machine Learning, vol 46, issue 1-3, pp 389–422, 2002.(Golub, 1999) Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. Golub et al, Science Vol 286

(4) Ying Lu, Jiawei Han. "Cancer classification using gene expression data"

(5) M. E. Blazadonakis, M. Zervakis, M. Kounelakis et al. "Support Vector Machines and Neural Networks as Marker Selectors for Cancer Gene Analysis", Proceeding of 3rd International IEEE Conference on Intelligent Systems, 2006, pp. 626-630.

(6) Pai-Hsuen Chen, Chih-Jen Lin, and Bernhard Scholkopf "A Tutorial on v-Support Vector Machines"

(7) “A Digital Architecture for Support Vector Machines: Theory, Algorithm, and FPGA Implementation.” Davide Anguita, Member, IEEE, Andrea Boni, and Sandro Ridella, Member, IEEE

(8) “The Tree Search Processor for Real-Time Track Finding” A. Bardi, M. Dell'Orso, S. Galeotti, P. Giannetti, G. Iannaccone, E. Meschi, F. Spinella

Για τη μονάδα πολλαπλασιασμού πινάκων:

(9) “The CSU Face Identification Evaluation System User’s Guide:Version 5.0”, Ross Beveridge, David Bolme, Marcio Teixeira and Bruce Draper

(10) “Accelerating Matrix Multiplication on FPGAs”, Rasha El-Atfy Mohamed, A. Dessouky, Hassan El-Ghitani

(11) “A High Throughput FPGA Implementation of A Bit-Level Matrix-Matrix Product” Amira, A. Bouridane, P. Milligan and P. Sage

(12) “Hardware Acceleration of Matrix Multiplication on a Xilinx FPGA”,Nirav Dave, Kermin Fleming, Myron King, Michael Pellauer, Muralidaran Vijayaraghavan

(13) “An FPGA based Parameterisable system for matrix product implementation”, A.Amira, F.Bensali

(14) “64-bit Floating-Point FPGA Matrix Multiplication”, Yong Dou S. Vassiliadis G. K. Kuzmanov G. N. Gaydadjiev

(15) “Design and implementation of a high performance matrix multiplier core for Xilinx Virtex FPGAs”, S Belkacemi, K Benkrid, D Crookes and A Benkrid

(16) “Area and Time Efficient Implementations of Matrix Multiplication on FPGAs”, Ju-wook Jang, Seoni Choi, and Viktor K. Prasanna

(17) “Energy-Efficient Matrix Multiplication on FPGAs”, Ju-wook Jang, Seonil Choi, and Viktor K. Prasanna