Εθνικό & Καποδιστριακό Πανεπιστήμιο Αθηνών
Τμήμα Φυσικής
Τομέας Ε′

Παράλληλες Αρχιτεκτονικές και Αλγόριθμοι για Ψηφιακή Επεξεργασία Σήματος και Εικόνας

Διδακτορική Διατριβή
του

Γεωργίου Λεντάρη

η συμβουλευτική επιτροπή:
επίκ. καθ. Δ. Ρείσης (επιβλ.)
καθ. Α. Κατσάγγελος
καθ. Γ. Τόμπρας

Αθήνα, Ιούλιος 2011

Abstract

The current dissertation contributes in designing efficient parallel architectures for image, video and graphics applications. The main objective is to organize parallel memories for supporting the most common algorithmic requirements encountered in the certain scientific field. Moreover, as a case study, the dissertation focuses on the implementation of motion estimation algorithms for video compression.

An elaborate study of bibliography related to both of the above objectives precedes the design of a programmable architecture executing a variety of block-matching algorithms with real-time performance. The design involves a parallel memory for the local storage of pixels, parallel arithmetic units for the examination of candidate blocks, and an application-specific instruction-set processor to meet the computational requirements of each algorithm. Overall, the architecture bases on pipelining techniques and data level parallelism to speed-up the block matching procedure. It introduces a novel instruction set and a speculative execution technique, which leads in almost 100% utilization of the data-path by allowing the algorithm to operate concurrently with the examination of candidate blocks. The cost and the performance of the reconfigurable motion estimation module are evaluated on FPGA platforms. Comparison to similar works of the literature highlights the advantages and verifies the efficiency of the proposed design.

Furthermore, the dissertation considers the problem of storing and retrieving pixels in parallel for a wider range of graphics applications. It introduces a technique tackling this problem and leading to an efficient memory organization, which facilitates parallel processing by allowing any algorithm to access numerous pixels in a single cycle. The solution involves a non-linear skew function to map each image pixel in one out of B memory banks. The proposed mapping supports the most common graphics requirements by achieving parallel access to *rows*, *columns*, *rectangles*, and *sparse* sets of pixels originating at any location on the image. Overall, the organization utilizes less memory banks compared to the corresponding solutions of the literature, which use $B>E$ when E pixels are requested in parallel. In contrast to the common bibliographic approach using prime –or other peculiar– numbers for B, the proposed solution organizes the memory with only $B=E$ banks allowing B to be any power of 2. Moreover, based on the properties of its

mapping function, the solution exploits successive request correlations during image/video processing to handle possible memory conflicts by spending only one cycle every B memory accesses. The resulting organization improves the resource utilization, reduces the hardware cost and leads to efficient parallel memory designs. The dissertation presents theorems and proofs regarding the properties of the proposed mapping function and the conflict handling technique. Subsequently, it analyzes the performance of the novel memory in example applications to show the merit of the proposed organization and to compare with hitherto published solutions at the implementation level. In this direction, the analysis also includes the integration of the novel memory to the developed motion estimation architecture; the results show improved performance/cost for the entire estimator and quantify the benefits of the proposed memory in terms of hardware resources.

The dissertation was submitted to NKUA written in Greek. For further information regarding the content of this dissertation, the reader can refer to papers published in English (see Appendix at final page). Specifically, most of the theoretical results regarding the proposed memory organization are reported in {1}, while a practical application can be found in {2}. Publications {5}{4}{3}{2} describe successively the development and improvement of the proposed motion estimation architecture.

keywords: parallel architectures, parallel memory organization, image/video processing, motion estimation

λέξεις κλειδιά: παράλληλες αρχιτεκτονικές, οργάνωση παράλληλων μνημών, επεξεργασία εικόνων/εικονοροών, εκτίμηση κίνησης

Αριθμός Μητρώου: 2006507, Δ.Δ.Φ.Ε. Φυσικής Περιβάλλοντος, Τμ. Φυσικής, ΕΚΠΑ

Περίληψη

Η παρούσα διδακτορική διατριβή συνεισφέρει ερευνητικά στην ανάπτυξη παράλληλων αρχιτεκτονικών για αλγορίθμους επεξεργασίας εικόνων και γραφικών. Κύριος στόχος της είναι η οργάνωση παράλληλων μνημών για αποδοτική υποστήριξη των απαιτήσεων που εμφανίζουν οι αλγόριθμοι του συγκεκριμένου επιστημονικού πεδίου. Επιπροσθέτως, ως πρακτική εφαρμογή, η διατριβή εστιάζει στην υλοποίηση αλγορίθμων εκτίμησης κίνησης σε εικονοροές.

Η εργασία ξεκινάει με μια εκτεταμένη μελέτη της σχετικής βιβλιογραφίας. Έπειτα, περιγράφει τη σχεδίαση και ανάπτυξη σε υλισμικό μιας προγραμματιζόμενης μονάδας εκτίμησης κίνησης με δυνατότητα εκτέλεσης πολλών διαφορετικών αλγορίθμων ταιριάσματος περιοχών σε πραγματικό χρόνο. Η μονάδα απαρτίζεται από παράλληλη μνήμη για τοπική αποθήκευση εικονοστοιχείων, από αριθμητικά κυκλώματα για σάρωση και σύγκριση περιοχών, καθώς κι από έναν εφαρμογοείδιο επεξεργαστή για την εκτέλεση των αλγοριθμικών βημάτων. Η προτεινόμενη αρχιτεκτονική βασίζεται σε τεχνικές σωλήνωσης και παραλληλισμό σε επίπεδο δεδομένων για την επιτάχυνση των υπολογισμών. Εισάγει ένα πρωτοποριακό σύνολο εντολών, σχεδιασμένο ειδικά για τη συγκεκριμένη κατηγορία αλγορίθμων. Επίσης, εισάγει μια εξειδικευμένη τεχνική για εκτέλεση εντολών παράλληλα με την διαδικασία εξέτασης των περιοχών, η οποία έχει ως αποτέλεσμα την εξάλειψη των κενών κύκλων στη σωλήνωση της διόδου δεδομένων και την αύξηση της εκμετάλλευσης του υλικού. Η επαναδιαρθρώσιμη αρχιτεκτονική που προκύπτει τελικά αναπτύσσεται σε προγραμματιζόμενους πίνακες λογικών πυλών (FPGA), όπου αξιολογείται το κόστος και η λειτουργία της. Η σύγκριση των αποτελεσμάτων με αντίστοιχα της βιβλιογραφίας καταδεικνύει την αποδοτικότητα και τα πλεονεκτήματα της αρχιτεκτονικής.

Γενικεύοντας σε ένα ευρύτερο πεδίο εφαρμογών, η διατριβή συνεχίζει προτείνοντας μια λύση στο πρόβλημα της οργάνωσης παράλληλων μνημών για αποθήκευση εικόνων. Η λύση επιτρέπει την ταυτόχρονη ανάκτηση πολλαπλών εικονοστοιχείων από τη μνήμη υποστηρίζοντας με αυτόν τον τρόπο την παράλληλη επεξεργασία των δεδομένων από τον εκάστοτε αλγόριθμο. Τα εικονοστοιχεία τοποθετούνται σε πολλαπλές τράπεζες μνήμης μέσω μιας προτεινόμενης συνάρτησης αντιστοίχισης, η οποία δημιουργεί δυνατότητα ταυτόχρονης πρόσβασης σε σύνολα εικονοστοιχείων που εμφανίζονται επάνω στην εικόνα ως γραμμές, στήλες, ορθογώνια, ή αραιά ορθογώνια. Τα εν λόγω σχήματα μπορούν να ανακτηθούν από οποιαδήποτε θέση της εικόνας με αποτέλεσμα την ικανοποίηση των συνήθων απαιτήσεων στις εφαρμογές γραφικών. Η καινοτομία της προτεινόμενης λύσης έγκειται στην μείωση του αριθμού των τραπεζών που χρησιμοποιούν οι προγενέστερες εργασίες της βιβλιογραφίας προκειμένου να πετύχουν τις ίδιες ή παρόμοιες δυνατότητες πρόσβασης. Συγκεκριμένα, αποφεύγει την έως σήμερα χρήση πρώτων ή άλλων δύσκολων αριθμών B με $B>E$,

όπου B το πλήθος των τραπεζών κι E το πλήθος των εικονοστοιχείων στα οποία απαιτείται ταυτόχρονη πρόσβαση ενός κύκλου. Αντί αυτών, οργανώνει την παράλληλη μνήμη με τον ιδανικό αριθμό τραπεζών $B{=}E$ επιτρέποντας στο B να πάρει τιμή ίση με οποιαδήποτε δύναμη του 2. Επί πλέον, εκμεταλλεύεται τις ιδιότητες της νέας συνάρτησης αντιστοίχισης και τις συσχετίσεις που εμφανίζουν οι διαδοχικές αιτήσεις μνήμης κατά την επεξεργασία εικόνων προκειμένου να διορθώσει τυχόν συγκρούσεις που παρουσιάζονται στη μνήμη ξοδεύοντας μόνο έναν κύκλο ανά B αιτήσεις. Συνεπώς, η προτεινόμενη λύση οδηγεί στη μείωση της υποεκμετάλλευσης του υλικού, στη μείωση του κόστους κατασκευής των κυκλωμάτων λειτουργίας και στη σχεδίαση βελτιωμένων και αποδοτικότερων παράλληλων μνημών. Η διατριβή παραθέτει θεωρήματα κι αποδείξεις των ιδιοτήτων της αντιστοίχισης και της τεχνικής διόρθωσης των αναπόφευκτων συγκρούσεων. Ακολούθως, αναλύει τη χρήση της νέας οργάνωσης μνήμης σε ευρέως διαδεδομένες εφαρμογές προκειμένου να αξιολογήσει τις επιδόσεις της και να τη συγκρίνει με τις προγενέστερες λύσεις σε πρακτικό επίπεδο. Μεταξύ αυτών, αναλύει τη βελτίωση που επιφέρει η ενσωμάτωση της νέας μνήμης στην προτεινόμενη μονάδα εκτίμησης κίνησης και ποσοτικοποιεί τα οφέλη της σε πόρους υλικού.

Η εξεταστική επιτροπή:

Διονύσιος Ρεΐσης, επίκ. καθ. ΕΚΠΑ	Άγγελος Κατσάγγελος, καθ. ΕΚΠΑ
Γεώργιος Τόμπρας, καθ. ΕΚΠΑ	Ανδρέας Πολύδωρος, καθ. ΕΚΠΑ
Ηλίας Μανωλάκος, αν. καθ. ΕΚΠΑ	Γεώργιος Στασινόπουλος, καθ. ΕΜΠ
Δημήτριος Σούντρης, επίκ. καθ. ΕΜΠ	

Ευχαριστίες

προς την οικογένειά μου για την αμέριστη συμπαράστασή τους, τον επιβλέποντα καθηγητή μου για την πολύτιμη καθοδήγηση/βοήθεια/γνώση, τους συναδέλφους μου στο εργ. 'Μελέτης κι Ανάπτυξης Ψηφιακών Συστημάτων' του τμ. Φυσικής για την επιτυχημένη συνεργασία μας, τους καθηγητές που με εκπαίδευσαν και με ενέπνευσαν στη μακρόχρονη αυτή προσπάθεια, καθώς και τους φίλους που την έκαναν πιο εύκολη.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Η επεξεργασία εικόνων/εικονοροών, η συμπίεσή τους, η μηχανική όραση και, γενικότερα, ένα πλήθος από ψηφιακές εφαρμογές που εμπίπτουν στο ευρύτερο πεδίο των γραφικών υπολογιστή αποτελούν πλέον αναπόσπαστο κομμάτι της ηλεκτρονικής πραγματικότητας υποστηρίζοντας βασικές λειτουργίες της σύγχρονης κοινωνίας. Η αύξηση της διαθέσιμης υπολογιστικής ισχύος που επιφέρει η πρόοδος της τεχνολογίας συνδυάζεται με νέους, πρωτοποριακούς, αλγορίθμους για αποτελεσματικότερη επεξεργασία των εν λόγω σημάτων κι επέκταση της χρήσης τους σε περισσότερους τομείς της επιστήμης και της καθημερινής ζωής. Προς την κατεύθυνση αυτή, σημαντικό ρόλο παίζει η παραλληλοποίηση των υπολογισμών και η ανάπτυξη κατάλληλων αρχιτεκτονικών για την επιτάχυνση των διαδικασιών με επιδόσεις πραγματικού χρόνου.

Οι περισσότεροι αλγόριθμοι για επεξεργασία εικόνων και γραφικών εμφανίζουν υψηλές υπολογιστικές απαιτήσεις εξαιτίας των πολυάριθμων πράξεων που εκτελούν επαναληπτικά σε διαφορετικά, αλλά αλληλοεπικαλυπτόμενα, σύνολα εικονοστοιχείων. Παρά ταύτα, ένα σημαντικό μέρος των υπολογισμών τους εμφανίζει χαμηλά επίπεδα αλληλοεξάρτησης επιτρέποντας τον παραλληλισμό τους σε επίπεδο διεργασιών και σε επίπεδο δεδομένων. Συνεπώς, σε περιπτώσεις όπου η εφαρμογή επιβάλλει μικρά χρονικά περιθώρια λειτουργίας, μπορούμε να τους επιταχύνουμε με παράλληλα επεξεργαστικά κυκλώματα και τροφοδότηση πολλαπλών δεδομένων από παράλληλες μνήμες. Χαρακτηριστικά παραδείγματα τέτοιων αλγορίθμων, ευρείας χρήσης και ιδιαίτερου ενδιαφέροντος, συναντάμε στα *Ψηφιακά Φίλτρα* εικόνων και στην *Εκτίμηση Κίνησης* εικονοροών.

1.1 Εκτίμηση Κίνησης

Με τον όρο 'Εκτίμηση Κίνησης' (ΕΚ) αναφερόμαστε σε ένα σύνολο διαφορετικών τεχνικών που ανιχνεύουν τη χρονική συσχέτιση των εικονοστοιχείων μιας εικονοροής. Γενικότερα [1], δεχόμαστε ότι οι εικονοροές περιλαμβάνουν περιοχές εικονοστοιχείων που μετακινούνται από καρέ σε καρέ δίνοντας στον θεατή την αίσθηση της κίνησης των απεικονιζόμενων αντικειμένων ή/και της κάμερας. Η διαδικασία ΕΚ επεξεργάζεται την εκάστοτε εικονοροή με σκοπό να ανακαλύψει και να καταγράψει όλες τις φαινόμενες μετακινήσεις των εικονοστοιχείων της, παράγοντας τελικά έναν πίνακα από ανύσματα κίνησης που μπορεί να φανεί χρήσιμος σε μια πληθώρα εφαρμογών.

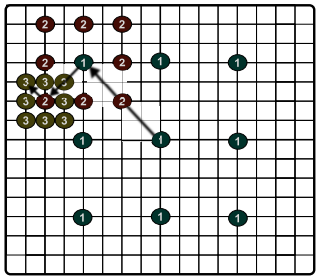
Οι τεχνικές ΕΚ που έχουν προταθεί στη βιβλιογραφία ποικίλουν ως προς την ανάλυση/πυκνότητα με την οποία καταγράφουν την κίνηση (ανά περιοχές εικονοστοιχείων αυθαιρέτου σχήματος, ανά μικρά μπλοκ εικονοστοιχείων, ή ανά μοναδικό εικονοστοιχείο –"optical flow"), ως προς το πεδίο στο οποίο βασίζουν τους υπολογισμούς τους (χώρου/χρόνου, ή συχνοτήτων), καθώς και ως προς τις αρχές λειτουργίας των επί μέρους μεθόδων που χρησιμοποιούν. Ως συνέπεια, οι τεχνικές ΕΚ διαφοροποιούνται σημαντικά ως προς τις υπολογιστικές τους απαιτήσεις και ως προς την ακρίβεια των αποτελεσμάτων τους. Οι βασικότερες από τις κατηγορίες στις οποίες μπορούμε να τις κατατάξουμε είναι οι εξής [2]: τεχνικές βαθμίδας (gradient), αναδρομικές ανά εικονοστοιχείο (pel-recursive), πεδίου συχνοτήτων (frequency-domain), και ταιριάσματα περιοχών (block-matching). Η επιλογή της κατάλληλης τεχνικής ΕΚ εξαρτάται άμεσα από την υποκείμενη εφαρμογή/πρόβλημα στην οποία αυτή θα χρησιμοποιηθεί.

Η παρούσα εργασία επικεντρώνεται στις τεχνικές ταιριάσματος περιοχών οι οποίες, μεταξύ άλλων, χρησιμοποιούνται ευρέως στη συμπίεση εικονοροών. Στην συνήθη μορφή, η διαδικασία ταιριάσματος περιοχών λαμβάνει ως είσοδο μια ορθογώνια περιοχή εικονοστοιχείων (π.χ. 16×16) από το τυχαίο καρέ K και την ταιριάζει με μια περιοχή αντιστοίχων διαστάσεων που βρίσκεται σε κάποιο προηγούμενο ή επόμενο καρέ $K \pm l$, $l \in \{1, 2, 3, \ldots\}$. Το ταίριασμα των περιοχών βασίζεται στην οπτική τους ομοιότητα, η οποία εκφράζεται μέσω της ελαχιστοποίησης ή μεγιστοποίησης κάποιου αντικειμενικού κριτηρίου. Τα προταθέντα κριτήρια ποικίλουν, με τα περισσότερα να εδράζονται στην απευθείας αφαίρεση των δυο περιοχών –εικονοστοιχείο προς εικονοστοιχείο. Τα επικρατέστερα κριτήρια ελαχιστοποίησης είναι τα: α) Άθροισμα Απολύτων Διαφορών (ΑΑΔ), όπου αφαιρούμε τις τιμές φωτεινότητας των αντιπαρατιθέμενων εικονοστοιχείων και συσσωρεύουμε τα μέτρα των διαφορών τους σε ένα τελικό νούμερο, β) Άθροισμα Τετραγώνων των Διαφορών (ΑΤΔ), όπου αντί για απόλυτες

τιμές υψώνουμε τις διαφορές στο τετράγωνο πριν τις αθροίσουμε, γ) Μέσο Τετραγωνικό Σφάλμα (ΜΤΣ), όπου τα τετράγωνα των διαφορών αθροίζονται και τελικώς διαιρούνται με το εμβαδόν της περιοχής. Κάθε ένα από αυτά τα κριτήρια συνιστούν ένα βαθμωτό μέγεθος, το οποίο αντιπροσωπεύει το βαθμό στον οποίο δυο περιοχές ομοιάζουν (γίνεται μηδέν όταν αυτές ταυτίζονται). Σημειώνεται ότι, ποιοτικά, το ΑΑΔ θεωρείται ελαφρώς υποδεέστερο των άλλων, αλλά εμφανίζει σημαντικά μικρότερη πολυπλοκότητα.

Η διαδικασία που ακολουθείται προκειμένου να επιτευχθεί το σωστό ταίριασμα περιλαμβάνει τη σύγκριση της τρέχουσας περιοχής (του υπό αναζήτηση μπλοκ εικονοστοιχείων) με έναν αριθμό από διαφορετικές υποψήφιες περιοχές (μπλοκ σε προηγούμενα/επόμενα καρέ). Κάθε σύγκριση δίνει ως αποτέλεσμα την τιμή ενός κριτηρίου ομοιότητας της υποψήφιας με την τρέχουσα περιοχή, ώστε στο τέλος της διαδικασίας να προκριθεί η υποψήφια με την καλύτερη τιμή (π.χ., ΑΑΔ). Οι θέσεις των υποψηφίων και η σειρά με την οποία αυτές θα εξεταστούν καθορίζεται από τον 'αλγόριθμο αναζήτησης'. Η βιβλιογραφία περιλαμβάνει πληθώρα τέτοιων αλγορίθμων, καθένας με διαφορετικές υπολογιστικές απαιτήσεις και ποιότητα ταιριασμάτων [3] [4] [5] [6] [7] [8] [9] [10]. Ο πιο άμεσος είναι αυτός της 'Πλήρους Αναζήτησης', ο οποίος ορίζει έκταση αναζήτησης (πολύ μεγαλύτερη από τις διαστάσεις του τρέχοντος μπλοκ) στο προηγούμενο καρέ κι εξετάζει όλους τους υποψηφίους που περιέχονται μέσα σε αυτήν. Από πλευράς ποιότητας ταιριασμάτων είναι ο καλύτερος, αλλά εμφανίζει υψηλές υπολογιστικές απαιτήσεις. Για τη συντόμευση της διαδικασίας έχουν προταθεί 'γρήγοροι' αλγόριθμοι, οι οποίοι είναι υποβέλτιστοι ευρετικοί κι εξετάζουν μόνο ένα μικρό αριθμό από υποψηφίους. Στην Εικόνα 1.1 φαίνεται ένα παράδειγμα εκτέλεσης του χαρακτηριστικού αλγορίθμου *Three-Step-Search*, ο οποίος υποδειγματοληπτεί την μεγάλη περιοχή αναζήτησης (χώρος υποψηφίων ανυσμάτων κίνησης) σε τρία διαδοχικά βήματα: αρχικά τοποθετεί ένα αραιό τετράγωνο μοτίβο ψαξίματος στο κέντρο της περιοχής αναζήτησης κι εξετάζει τα 8 σημεία του (8 υποψήφια μπλοκ), ενώ σε κάθε επόμενο βήμα επανατοποθετεί το μοτίβο στο σημείο που προηγουμένως βρήκε το καλύτερο ταίριασμα κι εξετάζει 8 νέους υποψηφίους (σε κάθε βήμα μειώνει το πλάτος του μοτίβου στο ήμισυ). Αυτή η ευρετική τεχνική είναι μια από τις πολλές που έχουν προταθεί κατά καιρούς, οι οποίες διαφοροποιούνται ως προς το μοτίβο ψαξίματος, τους ενδιάμεσους ελέγχους, ή ακόμα και την επεξεργασία επιπρόσθετων πληροφοριών σχετικά με την κίνηση μιας ευρύτερης περιοχής γύρω από την τρέχουσα.

Η Εκτίμηση Κίνησης κρίνεται εξαιρετικής σημασίας σε εφαρμογές συμπίεσης κι επεξεργασίας εικονοροών, καθώς και σε εφαρμογές μηχανικής όρασης. Στη συμπίεση εικονοροών χρησιμοποιείται για την αφαίρεση της περίσσειας πληρο-



Εικόνα 1.1: Παράδειγμα ευρετικού αλγορίθμου ταιριάσματος περιοχών. Απεικονίζεται ο χώρος ανυσμάτων κίνησης της περιοχής αναζήτησης και 25 υποψήφιοι (εξέταση σε 3 αναδρομικά βήματα, τετράγωνο μοτίβο ψαξίματος).

φοριών από τα διαδοχικά καρέ. Στην επεξεργασία εικονοροών, γενικότερα, χρησιμοποιείται για ιχνηλάτηση αντικειμένων (object tracking), υπερδειγματοληψεία εικόνων στο χώρο και στο χρόνο (super-resolution & frame-rate up-conversion), περιορισμό θορύβου (noise suppression), απαλοιφή θόλωσης (deblurring) [1]. Περιγράφουμε συνοπτικά μερικές από τις πρακτικές εφαρμογές που κάνουν χρήση της ΕΚ και ειδικότερα του ταιριάσματος περιοχών.

Εφαρμογή: *Συμπίεση Εικονοροών*

Τα σύγχρονα πρότυπα συμπίεσης, π.χ., H.264/AVC και MPEG-4 [3] [11], όπως τα παλαιότερα (H.263, MPEG-2) αλλά και αυτά που προετοιμάζονται για το μέλλον (H.265), βασίζονται σε μεγάλο βαθμό στις τεχνικές ΕΚ για την επίτευξη υψηλών λόγων συμπίεσης των εικονοροών. Μια από τις κεντρικές ιδέες πίσω από τη συμπίεση τέτοιου είδους σημάτων είναι η απαλοιφή της περίσσειας πληροφορίας που εμφανίζεται ανάμεσα στα διαδοχικά καρέ εξαιτίας της χρονικής τους συσχέτισης. Πιο συγκεκριμένα, ο κωδικοποιητής χρησιμοποιεί πολλαπλές περιοχές από προηγούμενα καρέ για να συνθέσει μια 'εικόνα-πρόβλεψη' που ομοιάζει με το τρέχον καρέ. Στη συνέχεια, κωδικοποιεί την –ελάχιστη– πληροφορία που περιγράφει τον τρόπο σύνθεσης της εικόνας-πρόβλεψης (π.χ., τα ανύσματα κίνησης) μαζί με μια 'εικόνα-διαφορά' που περιέχει τη διαφοροποίηση της εικόνας-πρόβλεψης από την πραγματική εικόνα. Απ' την άλλη, ο αποκωδικοποιητής χρησιμοποιεί τα δυο σταλθέντα δεδομένα (τρόπος πρόβλεψης και εικόνα-διαφορά) για να ανακτήσει την αρχική –τρέχουσα– εικόνα. Στην παραπάνω διαδικασία, όταν τα διαδοχικά καρέ του βίντεο εμφανίζουν ικανή συσχέτιση μεταξύ τους κι όταν εφαρμόζουμε κάποια αποτελεσματική τεχνική ΕΚ, τότε συνθέτουμε πολύ ποιοτικές προβλέψεις, άρα εικόνες-διαφορές με

πολύ μικρή πληροφορία που οδηγούν σε αποδοτική κωδικοποίηση και μικρό διφυόρρευμα (bitstream) ανάμεσα στον κωδικοποιητή και τον αποκωδικοποιητή. Η κοινότητα έχει καταλήξει στη χρήση τεχνικών ταιριάσματος περιοχών για τις εφαρμογές συμπίεσης εικονοροών, καθώς οι τεχνικές αυτές προσφέρουν ικανοποιητική ακρίβεια εκτίμησης σε συνδυασμό με μειωμένο όγκο σηματοδοσίας (π.χ., ανύσματα κίνησης) και πρακτικό χρόνο επεξεργασίας. Η συνήθης διαδικασία περιλαμβάνει την a priori τμηματοποίηση του τρέχοντος καρέ σε πλακίδια 16×16 εικονοστοιχείων (Macroblocks) και το ταίριασμα καθενός με κάποια ομοιάζουσα περιοχή από γειτονικό καρέ [3]. Με τον τρόπο αυτό ξοδεύουμε μόνο ένα άνυσμα κίνησης ανα 256 εικονοστοιχεία για τη σύνθεση μιας εικόνας-διαφοράς, η εντροπία της οποίας πολλές φορές τείνει στο μηδέν. Για ακόμα καλύτερες προβλέψεις, τα πρότυπα επιτρέπουν περαιτέρω κατακερματισμό του κάθε πλακιδίου ανάλογα με την κινητικότητα της περιοχής που αυτό απεικονίζει (έως 4×4). Όμως, η απόφαση αυτή επαφίεται στον κωδικοποιητή κατά την επεξεργασία και προϋποθέτει κάποιον αλγόριθμο βελτιστοποίησης διφυορυθμού-παραμόρφωσης (rate-distortion optimization), καθώς ο κατακερματισμός ενέχει κίνδυνο σημαντικής αύξησης των διφύων σηματοδοσίας (εισάγει περισσότερα ανύσματα κίνησης ανά πλακίδιο).

Σε ορισμένες εφαρμογές μετάδοσης και προβολής τρισδιάστατων εικονοροών συναντάμε την τεχνική *'απεικόνισης βάθους εικόνας' (depth image based rendering, DIBR)*, όπου εκτός από τον κλασσικό διδιάστατο μονοσκοπικό χώρο των χρωμάτων έχουμε και το χάρτη βάθους (depth-map) των εικονοστοιχείων. Ο χάρτης αυτός βοηθά στην προβολή των αντικειμένων από διαφορετικές, εικονικές, γωνίες θέασης. Η εκτίμηση κίνησης με ταιριάσματα περιοχών μπορεί να χρησιμοποιηθεί με επιτυχία σε τέτοιες εφαρμογές (π.χ., για αποδοτικότερη συμπίεση) [12], αν επεκτείνουμε την αναζήτηση στην τρίτη διάσταση: αρχικά ψάχνουμε το καλύτερο ταίριασμα στον συνήθη δισδιάστατο χώρο κι έπειτα, βελτιώνουμε το ταίριασμα με αναζήτηση στην τρίτη διάσταση –βάθος– δοκιμάζοντας διαφορετικές αυξομειώσεις στις τιμές φωτεινότητας των εξεταζόμενων εικονοστοιχείων (θεωρούμε ότι η 'θόλωση' μιας περιοχής είναι ανάλογη του οπτικού της βάθους). Η διαδικασία αυτή οδηγεί σε πολύ καλύτερες εικόνες-πρόβλεψη, αφού λαμβάνει υπόψη την κίνηση των αντικειμένων και στις τρεις διαστάσεις του πραγματικού κόσμου.

Εφαρμογή: *Υπερδειγματοληψία στο Χρόνο*

Μια επικουρική μέθοδος συμπίεσης των εικονοροών που βασίζεται ακόμα περισσότερο στη χρονική συσχέτιση των διαδοχικών εικόνων είναι η άμεση απόρριψη

κάποιων καρέ από τον κωδικοποιητή και η ανασύστασή τους στον αποκωδικοποιητή χρησιμοποιώντας τα γειτονικά τους. Αναφέρουμε [13] τρεις απλούς τρόπους ανασύστασης: α) απευθείας αντιγραφή του προηγούμενου καρέ στο ελλείπον, β) μέσος όρος ανάμεσα στο προηγούμενο και το επόμενο καρέ, και γ) εκτίμηση κίνησης του επόμενου καρέ από το προηγούμενο και ανασύσταση του ενδιάμεσου καρέ θεωρώντας ότι οι κινήσεις είναι ομαλές (άρα οι ενδιάμεσες μετατοπίσεις εμφανίζονται στο μέσον της υπολογισμένης τροχιάς). Πιο συγκεκριμένα, στο (γ) η φωτεινότητα κάθε εικονοστοιχείου υπολογίζεται ως η μέση τιμή του αντίστοιχου εικονοστοιχείου από όπου ξεκίνησε η υπολογισμένη τροχιά (προηγούμενο καρέ) κι εκείνου όπου τελείωσε (επόμενο καρέ). Όποιες ασυνέχειες εξομαλύνονται με ειδικούς ελέγχους και ψηφιακά φίλτρα. Ανάμεσα στις τρεις παραπάνω μεθόδους, σαφώς καλύτερη ποιότητα παρέχει η τρίτη, η οποία μάλιστα μπορεί να κάνει κι απευθείας χρήση των ανυσμάτων κίνησης των πλακιδίων που υπολόγισε ο κωδικοποιητής (περιέχονται στο διφυόρρευμα). Βεβαίως, παρόμοιες ιδέες υποστηριζόμενες από περίπλοκους αλγορίθμους ταιριάσματος περιοχών μπορούν να εφαρμοστούν οπουδήποτε απαιτείται αύξηση του ρυθμού προβολής των καρέ (frame-rate up-conversion) [14].

Στη *'δυναμική απεικόνιση μαγνητικής τομογραφίας' (dynamic MRI)*, η απόκτηση διαδοχικών εικόνων (data acquisition) γίνεται με χρονική καθυστέρηση που οφείλεται σε φυσικούς και φυσιολογικούς περιορισμούς (π.χ., διέγερση νεύρων). Ως αποτέλεσμα, είναι δύσκολο να επιτευχθεί ο κατάλληλος ρυθμός προβολής για τη σωστή απεικόνιση των αντικειμένων. Βέβαια, ο μεγάλος βαθμός χωροχρονικής συσχέτισης που εμφανίζουν τα σήματα αυτού του είδους επιτρέπει τη φαινομενική αύξηση της ταχύτητας απόκτησης δεδομένων με χρήση κατάλληλων τεχνικών όπως, π.χ., την εκτίμηση κίνησης με ταιριάσματα περιοχών [15]. Εδώ, η ΕΚ συνδυάζεται με εναλλακτικές τεχνικές παραγωγής προβλέψεων χαμηλότερης ποιότητας και οδηγεί στη χρονική παρεμβολή νέων εικόνων, ανάμεσα σε αυτές που καταγράφουν οι αισθητήρες, η οποία αυξάνει το ρυθμό και την ποιότητα της τελικής εικονοροής MRI.

Εφαρμογή: *Υπερδειγματοληψία στο Χώρο*

Η *υπερ-ανάλυση (super-resolution) εικόνας* στοχεύει στην παραγωγή εικόνων και βίντεο υψηλής ανάλυσης λαμβάνοντας ως είσοδο καταγεγραμμένες εικόνες (της ίδιας πραγματικής σκηνής) που φέρουν χαμηλή ανάλυση. Η υπερ-ανάλυση βελτιώνει την απεικόνιση των οπτικών λεπτομερειών και το πεδίο εφαρμογής της ποικίλει από δορυφορικές φωτογραφήσεις και συμπίεση δεδομένων έως ιατρικές διαγνώσεις. Η σχετική έρευνα έχει οδηγήσει σε πλήθος διαφορετικών

τεχνικών υπερ-ανάλυσης, μεταξύ των οποίων συναντάμε και διαδικασίες που κάνουν χρήση μεθόδων εκτίμησης κίνησης προκειμένου να παράξουν πληροφορία από τη χωροχρονική συσχέτιση των διαδοχικών εικόνων [16]. Οι μέθοδοι εκτίμησης κίνησης που υλοποιούνται περιλαμβάνουν παραμετρική μοντελοποίηση (parametric modeling), υπολογισμό οπτικής ροής (optical flow), καθώς και ταιριάσματα περιοχών (block matching).

Εφαρμογή: *Ιχνηλάτηση Αντικειμένων*

Μια κλασσική χρήση της εκτίμησης κίνησης προκύπτει στην 'ιχνηλάτηση αντικειμένων' (object tracking) σε γενικότερες εφαρμογές μηχανικής όρασης, όπως π.χ., αλληλεπίδραση ανθρώπου-μηχανής, εικονική πραγματικότητα, έλεγχο κυκλοφορίας, ασφάλεια/επιτήρηση χώρου, κ.α. Ο τρόπος με τον οποίο υλοποιείται η ΕΚ σε όλες αυτές τις περιπτώσεις ποικίλει. Ένα παράδειγμα στο οποίο τη συναντάμε με τη μορφή ταιριασμάτων μπλοκ είναι στην επιτάχυνση περιοχοπαγών (region-based) αλγορίθμων ιχνηλάτησης όπου, μεταξύ άλλων, συμβάλλει στην αποφυγή χρονοβόρων τεχνικών χωρικής τμηματοποίησης [17]. Εδώ, η μεταφορά κι οι μεταβολές στο περίγραμμα του αντικειμένου ανιχνεύονται μέσα από ταιριάσματα μπλοκ και ομαδοποιήσεις ανυσμάτων κίνησης. Για μεγαλύτερη ακρίβεια, τα μπλοκ αναζήτησης φέρουν μεταβλητές διαστάσεις όταν βρίσκονται στα όρια του αντικειμένου (από 16×16 έως 4×4), ενώ η αρχικοποίηση του περιγράμματος μπορεί να γίνει είτε με ανθρώπινη παρέμβαση, είτε αυτόματα (μέσω πολυζωνικής πολύτιμης τμηματοποίησης, που λειτουργεί αποδοτικά όταν έχουμε στατικές κάμερες). Πιο αναλυτικά, η διαδικασία περιλαμβάνει την αρχικοποίηση του αντικειμένου, τη διαδοχική εκτέλεση του αλγορίθμου ταιριάσματος περιοχών για μπλοκ εντός κι εκτός του αντικειμένου, έλεγχο κινητικότητας της εικόνας (από τα μέτρα των ανυσμάτων κίνησης) για την αυξομείωση του βήματος της ΕΚ, καθώς κι εξέταση και ομαδοποίηση ανυσμάτων που βρίσκονται κοντά στα όρια του αντικειμένου ώστε να επαναρυθμιστεί το περίγραμμά του (αντιμετώπιση φαινομένου επικάλυψης περιοχών κατά τη μετακίνηση αντικειμένων, occlusion/disocclusion). Σημειώνεται ότι, ενα από τα πλεονεκτήματα της χρήσης ταιριάσματος περιοχών σε εφαρμογές οπτικής ιχνηλάτησης είναι η δυνατότητα παραλληλοποίησης της αναζήτησης των διαφορετικών μπλοκ, δυνατότητα που επιτρέπει την κατάλληλη επιτάχυνση για επεξεργασία σε πραγματικό χρόνο.

Εφαρμογή: *Παραγωγή Μετα-πληροφορίας*

Το πρότυπο MPEG-7 [18] καθορίζει τον τρόπο αναπαράστασης/καταγραφής μετα-πληροφοριών σχετικά με το χαρακτηρισμό του πραγματικού περιεχομένου ενός αρχείου πολυμέσων (π.χ., ονόματα πρωταγωνιστών, αλλαγή σκηνών, και πλήθος άλλων στοιχείων για τη δομή, τη σημασιολογία, κλπ). Χρησιμοποιείται ανεξάρτητα από το πρότυπο συμπίεσης του αρχείου και σκοπός του είναι η διευκόλυνση της ανακάλυψης πληροφοριών σε μεγάλες πολυμεσικές συλλογές (π.χ, βλ. μηχανές αναζήτησης σε ψηφιακές βιβλιοθήκες). Κάνει δυνατή τη γρήγορη σάρωση βίντεο, την ανάκτηση και διαχείρηση περιεχομένου, την ασφάλεια, κ.α. Έτσι, το πεδίο εφαρμογής του επεκτείνεται σε υπηρεσίες ηλεκτρονικού καταλόγου, σε εκπομπές μέσων μαζικής ενημέρωσης (επιτρέπει αναγνώριση και φιλτράρισμα για επιλογή καναλιών), σε σύνταξη πολυμέσων, σε ηλεκτρονικό εμπόριο, καθώς και σε υπηρεσίες εκπαιδευτικές, τουριστικές, κ.α. Πρακτικά, το MPEG-7 ορίζει διαφορετικούς 'περιγραφητές' (descriptors) της πληροφορίας. Το τρίτο μέρος του προτύπου (part3–visual) ασχολείται με την οπτική πληροφορία του αρχείου και περιλαμβάνει περιγραφητές για το χρώμα, την υφή, τα σχήματα, την αναγνώριση αντικειμένων, και φυσικά, για την κίνηση: μεταφορά κάμερας, συνολική κινητικότητα, τροχιές κινουμένων περιοχών, κ.α.

Μια σημαντική πληροφορία στην περιγραφή της κίνησης αφορά στη 'χρονική τμηματοποίηση' της εικονοροής (temporal segmentation), στην οποία βρίσκουν εφαρμογή και τα ταιριάσματα περιοχών που μελετάμε στην παρούσα εργασία. Η χρονική τμηματοποίηση χωρίζει ένα βίντεο στα βασικά χρονικά συστατικά του –τις διαδοχικές σκηνές– επιτρέποντας γρήγορη προσπέλαση (browsing), ομαδοποίση/χαρακτηρισμό σύμφωνα με τις μεταξύ τους ομοιότητες, κ.α. Για την αυτόματη οριοθέτηση σκηνών (shot boundaries detection) έχουν προταθεί πολλές τεχνικές, οι οποίες βασίζονται άλλοτε στην επεξεργασία ιστογραμμάτων χρώματος, άλλοτε στο ρυθμό αλλαγής των ακμών, άλλοτε στις βαθμιαίες εναλλαγές (fade in/out), κι άλλοτε στην επεξεργασία των ανυσμάτων κίνησης που προκύπτουν από τα ταιριάσματα περιοχών [19]. Η τελευταία τεχνική συνίσταται, αρχικά, στην εκτίμηση των ανυσμάτων κίνησης μέσα σε κάθε καρέ, κι έπειτα, στον υπολογισμό μιας συνάρτησης που λαμβάνει υπόψη τα μέτρα των ανυσμάτων κι ορισμένες στατιστικές ροπές αυτών (π.χ., διακύμανση) προκειμένου να χαρακτηρίσει την γενικότερη 'ένταση της κίνησης' του καρέ. Παρακολουθώντας την εξέλιξη της έντασης από καρέ σε καρέ (π.χ., διαδοχικές διαφορές έντασης) είναι δυνατόν να εξάγουμε συμπεράσματα σχετικά με τις αλλαγές των σκηνών.

1.2 Παράλληλη μνήμη

Η επιτάχυνση της ψηφιακής επεξεργασίας εικονορροών σε όλες τις παραπάνω εφαρμογές, κι όχι μόνο, καθίσταται εξαιρετικής σημασίας, ειδικά όταν απαιτούνται επιδόσεις πραγματικού χρόνου. Για την επιτάχυνση των διαδικασιών χρησιμοποιείται ένα πλήθος από διαφορετικές τεχνικές, ειδικής ή γενικής χρήσης. Εξαιτίας της φύσης του εν λόγω σήματος και των αλγορίθμων που χρησιμοποιούνται στο πεδίο, μια αρκετά διαδεδομένη τεχνική επιτάχυνσης είναι ο παραλληλισμός σε επίπεδο δεδομένων (*data parallelism*). Η τεχνική αυτή συνίσταται στην ταυτόχρονη επεξεργασία πολλών διαφορετικών δεδομένων της εικόνας, δηλαδή στην προώθηση πολλών διαφορετικών εικονοστοιχείων σε πολλαπλές επεξεργαστικές μονάδες του συστήματος.

Ο παραλληλισμός σε αυτό το επίπεδο προϋποθέτει τη δυνατότητα ανάγνωσης κι εγγραφής πολλαπλών δεδομένων από και προς τη μνήμη του συστήματος σε κάθε κύκλο ρολογιού. Συνήθως, πρέπει να είναι δυνατή η συνεχής, άμεση, πρόσβαση σε διαφορετικά σύνολα δεδομένων της εικόνας. Σε αντίθετη περίπτωση, η συλλογή των στοιχείων που συνθέτουν την είσοδο των υπολογιστικών μονάδων της αρχιτεκτονικής απαιτεί επιπρόσθετους κύκλους ρολογιού και καθυστερεί συνολικά την επεξεργασία του σήματος. Δηλαδή, μια μνήμη ανεπαρκών δυνατοτήτων συνιστά σημείο συμφόρησης (*bottleneck*) της αρχιτεκτονικής κι, επί της ουσίας, ακυρώνει τη χρήση οποιονδήποτε τεχνικών επιτάχυνσης.

Οργάνωση Μνήμης για Υποστήριξη Παράλληλων Υπολογισμών

Η βέλτιστη οργάνωση της μνήμης του συστήματος εξαρτάται άμεσα από τον τρόπο λειτουργίας του αλγορίθμου επεξεργασίας της εικόνας. Συγκεκριμένα, εξαρτάται από τον ρυθμό με τον οποίο αυτός θα ζητήσει τα εικονοστοιχεία (όγκος προς χρόνο), καθώς κι από τη χωρική θέση ή τη συσχέτιση που εμφανίζουν οι διαδοχικές κλήσεις του αλγορίθμου. Γενικά, ο ρυθμός καθορίζει το πλάτος διεπαφής της μνήμης, ενώ η συσχέτιση καθορίζει το πλήθος των τραπεζών της. Για παράδειγμα, αν πρόκειται να επεξεργαστούμε τις τιμές φωτεινότητας 4 εικονοστοιχείων ανά κύκλο ρολογιού τότε η μνήμη μας θα πρέπει να σχεδιαστεί με διεπαφή 32 διφύων ($= 4 \times 8$ bits). Επιπλέον, αν στο απλό αυτό παράδειγμα κάθε δυνητικά κληθείσα τετράδα εικονοστοιχείων είναι προκαθορισμένη κι εντελώς ξένη με τις υπόλοιπες, τότε μπορούμε να την αντιστοιχίσουμε σε μια ξεχωριστή διεύθυνση της μνήμης και να χρησιμοποιήσουμε μια μόνο τράπεζα για την αποθήκευση ολόκληρης της εικόνας. Βέβαια, στη γενική περίπτωση η λειτουργία ενός αλγορίθμου δεν είναι τόσο

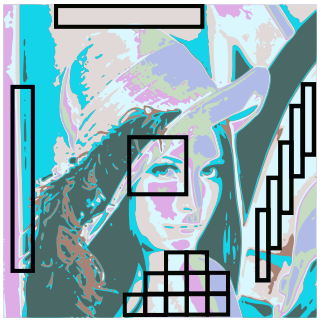
απλή. Η σειρά με την οποία αυτός θα ζητήσει τα ξεχωριστά σύνολα εικονοστοιχείων (π.χ. τις τετράδες) μπορεί να μην είναι γνωστή, οι θέσεις των ζητούμενων εικονοστοιχείων πάνω στην εικόνα μπορεί να μην είναι προκαθορισμένες, κι επιπρόσθετα, τα ζητούμενα σύνολα να μην είναι ξένα μεταξύ τους. Ως εκ τούτου, οι σχεδιαστές καλούνται να επιλύσουν ένα πρόβλημα ελαχιστοποίησης (των απαιτούμενων κύκλων πρόσβασης, του αριθμού των τραπεζών, και του όγκου της μνήμης) λαμβάνοντας υπόψη τους συγκεκριμένες προδιαγραφές.

Ένας από τους βασικούς στόχους της επίλυσης τέτοιων προβλημάτων είναι η ανάπτυξη γενικευμένων –κι όχι εξειδικευμένων– λύσεων. Δηλαδή, στοχεύουμε στην ανάπτυξη μιας τεχνικής που να οδηγεί στη βέλτιστη οργάνωση μνήμης για ένα πλήθος εφαρμογών κι όχι για την υποστήριξη ενός συγκεκριμένου αλγορίθμου ή ενός μόνο συστήματος. Προς αυτή την κατεύθυνση, η ερευνητική διαδικασία συμπεριλαμβάνει τη μελέτη των προδιαγραφών πολλών διαφορετικών αλγορίθμων με σκοπό την αναγνώριση των κοινών τους σημείων. Κατόπιν, σχεδιάζεται μια παραμετροποιημένη λύση η οποία εξυπηρετεί τις κοινές προδιαγραφές κι επομένως είναι κατάλληλη για ένα πλήθος περιπτώσεων. Οι παράμετροι της λύσης χρησιμοποιούνται για την προσαρμογή στα ειδικά χαρακτηριστικά του εκάστοτε προβλήματος, π.χ., στη ρυθμαπόδοση της μνήμης ή στον τρόπο ομαδοποίησης των δεδομένων που αυτή περιέχει.

Απαιτήσεις μνήμης σε εφαρμογές γραφικών

Στις εφαρμογές γραφικών το μεγαλύτερο μέρος των υπολογισμών σχετίζεται με την επεξεργασία της εικόνας ή του καρέ που είναι αποθηκευμένο σε κάποιο μέσο, προσωρινά ή μόνιμα. Οι συνήθεις διεργασίες που εκτελούνται είναι δυνατόν να περιλαμβάνουν την παραγωγή εικονοστοιχείων (π.χ., σχεδίαση εικόνας), την τροποποίηση εικονοστοιχείων (π.χ., φιλτράρισμα εικόνας), ή τη σάρωση εικονοστοιχείων (π.χ., αναγνώριση εικόνας/προτύπων). Στην πρώτη περίπτωση η υλοποίηση απαιτεί υποστήριξη για εγγραφή πολλαπλών δεδομένων σε μνήμη, στην τρίτη για ανάγνωση πολλαπλών δεδομένων από μνήμη, ενώ στη δεύτερη περίπτωση απαιτείται ανάγνωση κι εγγραφή μαζί.

Η φύση του σήματος και των εκτελούμενων διεργασιών επιβάλει πολλές φορές το χωρικό κατακερματισμό της εικόνας, δηλαδή την επεξεργασία της κατά συνεκτικές –χωρικά– περιοχές. Δηλαδή, κάθε μερική είσοδος του παράλληλου αλγορίθμου αποτελείται από εικονοστοιχεία που γειτνιάζουν μεταξύ τους (βλ. Εικόνα 1.2). Στην πράξη, η ανάκτηση των εικονοστοιχείων γίνεται με βάση κάποιο σχήμα πρόσβασης που καθορίζει περιοχή πάνω στην εικόνα (*access pattern*): τετράγωνο (π.χ., 4×4 pixels), γραμμή ή στήλη (π.χ., 4×1), τρίγωνο,

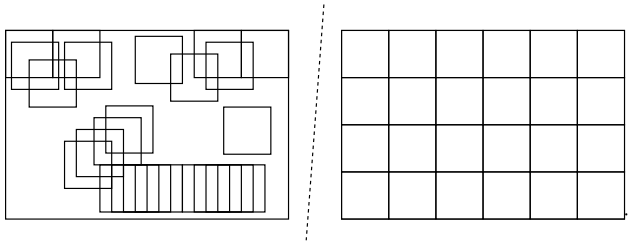


Εικόνα 1.2: Περίπτωση ορθογώνιων περιοχών εικονοστοιχείων, κάθε μια εκ των οποίων συνιστά μερική είσοδο του αλγόριθμου επεξεργασίας.

κτλ. Κατά τη λειτουργία του αλγορίθμου εκτελούνται διαδοχικές κλήσεις τέτοιων σχημάτων από την εικόνα έως ότου ανακτηθεί ολόκληρη η είσοδός του. Σημειώνουμε εδώ ότι αυτός ο τρόπος πρόσβασης στην εικόνα είναι ο πλέον διαδεδομένος και χρησιμοποιείται ακόμα κι όταν η χωρική συνοχή των μερικών εισόδων δεν αποτελεί αυστηρή προϋπόθεση.

Παρότι η ανάκτηση γειτονικών εικονοστοιχείων αποτελεί κοινό γνώρισμα των περισσότερων εφαρμογών γραφικών, το ακριβές σχήμα ανάκτησης εξαρτάται από τις εκάστοτε προδιαγραφές του αλγορίθμου ή της αρχιτεκτονικής. Έτσι, για παράδειγμα, ένα ψηφιακό φίλτρο εικόνας θα χρησιμοποιούσε διαδοχικά τετράγωνες περιοχές εικονοστοιχείων, ενώ ένας μετασχηματισμός στο πεδίο των συχνοτήτων θα χρησιμοποιούσε διαδοχικά γραμμές και στήλες. Μάλιστα, δεν είναι λίγοι οι αλγόριθμοι εκείνοι που συνδυάζουν πολλά διαφορετικά σχήματα προκειμένου να αυξήσουν την απόδοσή τους, π.χ, όπως συμβαίνει στην αναζήτηση κλασματικών κινήσεων με χρήση τεχνικών ταιριάσματος περιοχών. Συνεπώς, η σχεδίαση μιας γενικής λύσης για τη μνήμη σε εφαρμογές γραφικών θα πρέπει να βασίζεται σε μια αρκετά ευρεία υπόθεση σχετικά με τα απαιτούμενα σχήματα πρόσβασης. Με άλλα λόγια, η μνήμη θα πρέπει να υποστηρίζει πρόσβαση τουλάχιστον κατά τετράγωνα, γραμμές και στήλες εικονοστοιχείων, κι όχι να περιορίζεται σε ένα μόνο είδος σχήματος.

Εκτός από την ποικιλία των σχημάτων πρόσβασης, είναι σημαντικό να καθορίσουμε και τις δυνατές θέσεις τους πάνω στην εικόνα. Είναι λογικό να υποθέσουμε ότι κάθε σχήμα από μόνο του πρέπει να έχει τη δυνατότητα να καλύψει ολόκληρη την εικόνα (ώστε οποιοσδήποτε αλγόριθμος να έχει άμεση πρόσβαση σε οποιοδήποτε εικονοστοιχείο της εικόνας). Μια τέτοια υπόθεση όμως ικανοποιείται με δυο διαφορετικούς τρόπους: α) το σχήμα να επιτρέπει



Εικόνα 1.3: Δυνατότητα πρόσβασης στην εικόνα κατά τετράγωνα: απεριόριστη (αριστερά) και περιορισμένη (δεξιά).

απεριόριστη πρόσβαση, δηλαδή, η θέση απο όπου αυτό θα ζητηθεί επάνω στην εικόνα να μην υπόκειται σε περιορισμούς, β) το σχήμα να επιτρέπει πρόσβαση σε περιοχές με συγκεκριμένη θέση επάνω στην εικόνα, δηλαδή, να περιορίζεται σε ένα προκαθορισμένο πλέγμα. Η Εικόνα 1.3 αντιπαραβάλει τους δυο τρόπους πρόσβασης χρησιμοποιώντας τετράγωνα. Στην πρώτη περίπτωση, τα σύνολα εικονοστοιχείων που μπορούμε να ανακτήσουμε εμφανίζουν ισχυρή επικάλυψη, ενώ στη δεύτερη περίπτωση είναι εντελώς ξένα μεταξύ τους. Ένα παράδειγμα εφαρμογής που επωφελείται από την απεριόριστη πρόσβαση είναι η Εκτίμηση Κίνησης, αφού η αναζήτηση μπορεί να κινηθεί αργά προς οποιαδήποτε κατεύθυνση. Παράδειγμα εφαρμογής που αρκείται στην περιορισμένη πρόσβαση είναι η σάρωση εικόνας κατά στήλες, αφού τα σύνολα που θα ανακτηθούν (μικρές κολώνες) δεν χρειάζεται να έχουν καμιά επικάλυψη μεταξύ τους. Ξεκάθαρα, η περιορισμένη πρόσβαση είναι υποπερίπτωση της απεριόριστης, αφού μια μνήμη που παρέχει δυνατότητα για τη δεύτερη, ουσιαστικά, παρέχει και για την πρώτη. Φυσικά, μια γενική λύση οφείλει να εξυπηρετεί τη γενική περίπτωση.

Το τελευταίο χαρακτηριστικό των σχημάτων πρόσβασης που πρέπει να ληφθεί υπόψη κατά τη σχεδίαση της μνήμης είναι το εμβαδόν τους. Το εμβαδόν του σχήματος καθορίζει τον αριθμό των εικονοστοιχείων στα οποία αποκτούμε πρόσβαση σε κάθε κύκλο ρολογιού. Ο αριθμός αυτός σχετίζεται άμεσα με τις τεχνικές προδιαγραφές της εκάστοτε αρχιτεκτονικής (π.χ., ρυθμαπόδοση) και παρουσιάζει μεγάλες διαφοροποιήσεις απο εφαρμογή σε εφαρμογή. Για το λόγο αυτό, ενσωματώνεται ως παράμετρος στη μέθοδο σχεδίασης επιτρέποντας στους κατασκευαστές να προσαρμόσουν τη μνήμη στις απαιτήσεις τους.

1.3 Στόχος και οργάνωση της διατριβής

Καθώς η τεχνολογία εξελίσσεται παρουσιάζει αυξανόμενη πρόοδο στην επεξεργασία εικόνων/εικονοροών/γραφικών βρίσκοντας όλο και περισσότερες εφαρμογές σε τομείς της καθημερινότητας που ποικίλουν από την ψυχαγωγία έως την ιατρική. Η χρήση της τεχνολογίας σε τέτοιου είδους εφαρμογές εξαρτάται άμεσα από τη δυνατότητα επιτάχυνσης της ψηφιακής επεξεργασίας με επιδόσεις πραγματικού χρόνου. Η εν λόγω επιτάχυνση προκύπτει από παραλληλισμό των υπολογισμών με κατάλληλη αρχιτεκτονική σχεδίαση, η αποδοτικότητα της οποίας στηρίζεται σε κάποιο υποκείμενο σύστημα μνήμης που παρέχει δυνατότητα ανάγνωσης κι εγγραφής δεδομένων σε παράλληλη μορφή.

Στόχος της παρούσας διατριβής είναι η σχεδίαση παράλληλων μνημών που να ικανοποιούν τις απαιτήσεις της πλειοψηφίας των εφαρμογών γραφικών προσφέροντας μια γενική λύση, οικονομικότερη κι αποδοτικότερη αυτών που έχουν προταθεί στη βιβλιογραφία. Επιπρόσθετα σε αυτή την κατεύθυνση, ως πρακτικό παράδειγμα εφαρμογής με χρήσιμες προεκτάσεις, η εργασία μελετά την ανάπτυξη αλγορίθμων εκτίμησης κίνησης σε υλισμικό και προτείνει μια προγραμματιζόμενη αρχιτεκτονική για την εκτέλεσή τους σε πραγματικό χρόνο.

Η παρουσίαση της διατριβής οργανώνεται σε επτά κεφάλαια. Πέραν του παρόντος εισαγωγικού, το *Κεφάλαιο 2* προσδιορίζει επακριβώς τα δυο προβλήματα που καλείται να επιλύσει η εργασία: αρχιτεκτονική σχεδίαση μονάδας εκτίμησης κίνησης και οργάνωση παράλληλης μνήμης για εφαρμογές γραφικών. Επιπλέον, καταγράφει τις λύσεις που έχουν προταθεί στη βιβλιογραφία σχετικά με τα δυο θέματα και καταλήγει σε κάποια γενικά συμπεράσματα. Το *Κεφάλαιο 3* περιγράφει τη σχεδίαση και την υλοποίηση της προτεινόμενης μονάδας εκτίμησης κίνησης. Τα *Κεφάλαια 4* και *5* αναλύουν την προτεινόμενη οργάνωση μνήμης κι αποδεικνύουν τις ιδιότητές της: το *4* υπό τις κλασσικές υποθέσεις που περιέχει η βιβλιογραφία, ενώ το *5* λαμβάνοντας υπόψη και μια νέα παρατήρηση σχετικά με τη συμπεριφορά των αλγορίθμων επεξεργασίας γραφικών. Ακολούθως, το *Κεφάλαιο 6* περιγράφει ορισμένες μελέτες περιπτώσεων στις οποίες βρίσκει εφαρμογή η προτεινόμενη οργάνωση μνήμης και ποσοτικοποιεί τις επιδόσεις της στον πραγματικό κόσμο. Μεταξύ των παραδειγμάτων του κεφαλαίου 6 συμπεριλαμβάνεται και η μονάδα ΕΚ του κεφαλαίου 3, η οποία ενσωματώνει πλέον τη νέα μνήμη και υπόκειται σε επανα-υλοποίηση προκειμένου να καταγραφούν τα οφέλη που αποκομίζουμε από την προτεινόμενη οργάνωση μνήμης. Τέλος, το *Κεφάλαιο 7* συνοψίζει τη διατριβή κι αναφέρει ορισμένες κατευθύνσεις για μελλοντική έρευνα.

Κεφάλαιο 2

Προσδιορισμός Προβλημάτων και Σχετική Βιβλιογραφία

Η παρούσα διατριβή εστιάζει σε δυο –όχι εντελώς ανεξάρτητα– προβλήματα που εντάσσονται στο ευρύτερο ερευνητικό πεδίο της επεξεργασίας εικόνων και εικονοροών: α) στην υλοποίηση αλγορίθμων εκτίμησης κίνησης, και β) στην οργάνωση παράλληλων μνημών για εφαρμογές γραφικών. Οι ενότητες που ακολουθούν παρουσιάζουν τις επί μέρους λεπτομέρειες των προβλημάτων που αντιμετωπίζει η διατριβή και περιγράφουν τις κοινά αποδεκτές απαιτήσεις που πρέπει να ικανοποιούν οι λύσεις τους. Επιπλέον, παραθέτουν τη σχετική βιβλιογραφία με τις έως σήμερα προταθείσες λύσεις/αρχιτεκτονικές, καθώς και τα συμπεράσματα που προκύπτουν από τη σφαιρική μελέτη τους.

2.1 Εκτίμηση Κίνησης με ταιριάσματα περιοχών

Στη γενική περίπτωση, η εκτίμηση κίνησης μέσω ταιριασμάτων αφορά σε περιοχές εικονοστοιχείων με αυθαίρετο σχήμα και μεταβλητές διαστάσεις. Εδώ, επικεντρωνόμαστε στην πλέον διαδεδομένη κι απλή εκδοχή των αλγορίθμων εκτίμησης κίνησης, όπου η αναζητούμενη περιοχή είναι ένα τετράγωνο 16×16 εικονοστοιχείων (Macroblock). Δηλαδή, λαμβάνουμε ως είσοδο το τρέχον πλακίδιο μεγέθους 16×16 και μια περιοχή αναζήτησης $N \times M$ εικονοστοιχείων με N,M>16 (ενδεχομένως από πολλαπλά καρέ αναφοράς), πάνω στην οποία προσπαθούμε να ανακαλύψουμε το βέλτιστο ταίριασμα του πλακιδίου. Η αναζήτηση περιλαμβάνει διαδοχικές συγκρίσεις του πλακιδίου με διάφορα υποψήφια

μπλοκ από την περιοχή $N \times M$ και το αποτέλεσμά της είναι ένα 'διάνυσμα κίνησης' που δείχνει τη θέση στην οποία εντοπίζεται τελικά το πλακίδιο.

Υπό την απλή αυτή μορφή, η εκτίμηση κίνησης συναντάται σε πλήθος εφαρμογών με πιο διαδεδομένη τη συμπίεση εικονοροών κατά τα πλέον σύγχρονα πρότυπα (π.χ., H.264/AVC [11]). Οι αλγόριθμοι και τα κριτήρια ομοιότητας που χρησιμοποιούνται κατά την αναζήτηση του ταιριάσματος ποικίλουν ανάλογα με τις εκάστοτε προδιαγραφές κόστους και ποιότητας αποτελέσματος. Ως εκ τούτου, έχουν προταθεί πολυάριθμοι σχετικοί αλγόριθμοι –ευρετικοί ή όχι– που κάνουν χρήση διαφορετικών τεχνικών κι εμφανίζουν ιδιότυπα χαρακτηριστικά λειτουργίας [3] [4] [5] [6] [7] [8] [9] [10] [20]. Οι απαιτήσεις τους σε υπολογιστική ισχύ ή επιπρόσθετη πληροφόρηση (είσοδο/μνήμη) κυμαίνονται από πολύ μικρές έως πολύ μεγάλες, ανάλογα με το είδος της αναζήτησης που εκτελούν.

Η παραπάνω εκδοχή του προβλήματος εκτίμησης κίνησης αποτελεί το παράδειγμα στο οποίο βασίζεται η παρούσα εργασία προκειμένου να μελετήσει το πρώτο της, πρακτικό, θέμα:

> ανάπτυξη αποδοτικών αρχιτεκτονικών για την εκτέλεση αλγορίθμων ταιριάσματος περιοχών, με κατάλληλο παραλληλισμό κι επιτάχυνση που επιτρέπει επεξεργασία πραγματικού χρόνου.

Στόχος της αρχιτεκτονικής λύσης είναι η βελτιστοποίηση ως προς τους καταναλισκόμενους πόρους υλικού, το ρυθμό διεκπεραιωτικότητας, και την ποιότητα των αποτελεσμάτων. Διευκρινίζουμε εδώ ότι το πρόβλημα που εξετάζουμε είναι η ανάπτυξη σε υλισμικό, όχι η σχεδίαση νέων αλγορίθμων αναζήτησης. Παρακάτω συνοψίζουμε τις πλέον χαρακτηριστικές προσεγγίσεις επί του θέματος και συγκρίνουμε τις αντίστοιχες κατασκευές ως προς το κόστος υλοποίησης και την ταχύτητά τους.

2.1.1 Αρχιτεκτονικές για επεξεργασία πραγματικού χρόνου

Η βιβλιογραφία περιέχει μια πληθώρα από αρχιτεκτονικές προτάσεις και υλοποιήσεις αλγορίθμων ταιριάσματος περιοχών σε υλισμικό. Πρωταρχικός στόχος των προτεινόμενων λύσεων είναι η δυνατότητα επεξεργασίας της εικονοροής σε πραγματικό χρόνο, δυνατότητα που επιτρέπει την ενσωμάτωση της κατασκευής σε τεχνολογικές εφαρμογές τελευταίας λέξης (state-of-the-art).

Οι προτάσεις καλύπτουν ένα ευρύ φάσμα απαιτήσεων ανταλλάσσοντας ποιότητα αποτελεσμάτων με κόστος υλοποίησης (cost-quality tradeoff). Πιο συγκεκριμένα, σε εφαρμογές που η ακρίβεια του ταιριάσματος παίζει καθοριστικό

ρόλο για την ορθή λειτουργία του συστήματος, οι μηχανικοί υλοποιούν τον αλγόριθμο 'Πλήρους Αναζήτησης' ξοδεύοντας πολλούς πόρους υλικού για την υποστήριξη των αυξημένων υπολογιστικών του απαιτήσεων. Σε εφαρμογές με ανεκτικότητα σε ελαφρώς υποβέλτιστα ταιριάσματα, οι μηχανικοί υλοποιούν κάποιον ευρετικό αλγόριθμο προκειμένου να εξοικονομήσουν σημαντικό υλικό κόστος. Η επιλογή του ευρετικού γίνεται κατά τη σχεδίαση και λαμβάνει υπόψη της όλες τις επί μέρους απαιτήσεις της εφαρμογής, καθώς και τον ισολογισμό κόστους-ποιότητας που εμφανίζουν όλες οι δημοσιευμένες τεχνικές αναζήτησης. Εκτός από την ανάπτυξη σε υλισμικό ενός συγκεκριμένου αλγορίθμου, ευρετικού ή όχι, η βιβλιογραφία εμφανίζει κι αρχιτεκτονικές λύσεις που μεταβιβάζουν την επιλογή του αλγορίθμου αναζήτησης στο χρήστη. Η επιλογή διατίθεται μετά την κατασκευή, είτε μέσα από έναν μικρό αριθμό προεγκατεστημένων αλγορίθμων, είτε μέσω μιας προγραμματιζόμενης μονάδας με δυνατότητα φόρτωσης κι εκτέλεσης των σχετικών εντολών. Σε όλες τις ανωτέρω περιπτώσεις, η πρωτοτυποποίηση κι η αξιολόγηση των προτεινόμενων αρχιτεκτονικών περιλαμβάνει υλοποιήσεις με 'ολοκλήρωση μεγάλης κλίμακας' (VLSI) ή/και υλοποιήσεις σε 'προγραμματιζόμενους πίνακες πυλών' (FPGA).

Αρχιτεκτονικές για πλήρη αναζήτηση

Η σχεδίαση αρχιτεκτονικών για την εκτέλεση της 'Πλήρους Αναζήτησης' σε πραγματικό χρόνο οδηγεί, αναγκαστικά, σε μεγάλο βαθμό παραλληλισμού με πολλαπλά κυκλώματα που καταναλώνουν σημαντικό αριθμό πόρων κι ενέργεια [21] [22] [23] [24] [25] [26] [27]. Το κόστος αυξάνει ακόμα περισσότερο όταν η εφαρμογή απαιτεί επεξεργασία μπλοκ μεταβλητών διαστάσεων (κατάτμηση του τρέχοντος πλακιδίου κι εξέταση έως και 41 διαφορετικών υποψηφίων) ή υπολογισμό περίπλοκων κριτηρίων σύγκρισης (π.χ., Lagrangian R/D cost criterion). Στην πράξη, η εργασία [22] βασίζεται σε έναν πίνακα από 16×16 επεξεργαστικές μονάδες που λειτουργούν παράλληλα πάνω στην περιοχή αναζήτησης κι εξάγουν 4×4 διαφορετικές τιμές ΑΑΔ[1]. Οι τιμές ΑΑΔ συνδυάζονται έτσι, ώστε να παραχθούν οι μετρικές ομοιότητας των διαφόρων κατατμήσεων του υποψηφίου πλακιδίου (μπλοκ μεταβλητών διαστάσεων με ανύσματα ±32). Η υλοποίηση της [22] σε VLSI (TSMC 0.18um standard cell library) απαιτεί 154K λογικές πύλες και έχει ρυθμό διεκπεραιωτικότητας 15 καρέ το δευτερόλεπτο για βίντεο ανάλυσης 4CIF (704×576). Παρόμοια, στην εργασία [23] χρησιμοποιούνται 16 επεξεργαστικές μονάδες για τη δημιουργία σωληνωτής αρχιτεκτονικής που

[1] *Άθροισμα Απολύτων Διαφορών* μεταξύ του τρέχοντος πλακιδίου και του υποψηφίου.

καταλήγει σε ένα συγκριτή ΑΑΔ. Οι τιμές ΑΑΔ όλων των διαφορετικών υποπλακιδίων υπολογίζονται ταυτόχρονα εξετάζοντας ανύσματα μήκους έως και ±3.75 (από 5 καρέ αναφοράς). Η υλοποίηση σε FPGA (Xilinx Virtex2) απαιτεί 14.5K slices[2] (ή 225K ισδύναμες λογικές πύλες) κι οδηγεί σε επεξεργασία πραγματικού χρόνου για βίντεο ανάλυσης CIF (352 × 288). Η εργασία [25] καταφέρνει επεξεργασία πραγματικού χρόνου ακόμα και για βίντεο πολύ υψηλής ανάλυσης (1920 × 1088 @ 91 fps) με 8 ή 16 επεξεργαστικές μονάδες που μοιράζονται εικονοστοιχεία κατά τη σωληνωτή επεξεργασία 16 υποπλακιδίων. Βέβαια, το κόστος σε πόρους FPGA (Xilinx Virtex5) ανέρχεται, αντίστοιχα, σε 77K ή σε 154K LUTs (περίπου 20K ή 40K slices). Η εργασία [26] χρησιμοποιεί ένα συστολικό πίνακα 256 επεξεργαστικών μονάδων, οι οποίες 'ολισθαίνουν' επάνω στην περιοχή αναζήτησης για τον υπολογισμό των ΑΑΔ των 41 δυνατών κατατμήσεων του πλακιδίου. Επιπροσθέτως, υπολογίζει και τη Λαγκραντζιανή συνάρτηση κόστους για να επιτύχει βελτιστοποίηση ρυθμού/παραμόρφωσης (rate/distortion otpimization) [28] κάνοντας χρήση του πολλαπλασιαστή Lagrange λ_{SAD} προκειμένου να επιλέξει τον τρόπο κατάτμησης-ταιριάσματος. Το κόστος της [26] σε πόρους FPGA (Xilinx Virtex4) ανέρχεται σε 8K slices (περίπου 182K ισοδύναμες πύλες) για επεξεργασία βίντεο 4CIF στα 38 fps.

Αρχιτεκτονικές για συγκεκριμένους ευρετικούς αλγορίθμους

Από όλους τους ευρετικούς αλγορίθμους, πιθανόν, πιο κοντά στην ιδέα της πλήρους αναζήτησης να βρίσκονται οι ιεραρχικοί. Εδώ, εξοικονομούμε πράξεις διότι η πλήρης αναζήτηση εκτελείται σε υποδειγματοληπτημένες περιοχές της εικόνας (μικρότεροι υποψήφιοι). Η τεχνική είναι αρκετά διαδεδομένη και έχει οδηγήσει σε διαφορετικές υλοποιήσεις [29] [30]. Ως παράδειγμα αναφέρουμε την [29] που παρουσιάζει και υλοποιεί έναν ιεραρχικό αλγόριθμο αναζήτησης τριών επιπέδων. Η εικόνα υποδειγματοληπτείται με διγραμμικά φίλτρα δυο φορές (επαναληπτικά): στο χαμηλότερο επίπεδο εκτελείται μια τοπική πλήρης αναζήτηση που οδηγεί σε δυο υποψήφια ανύσματα, στο μεσαίο επίπεδο εξετάζονται οι γειτονικές περιοχές των δυο προηγούμενων ανυσμάτων, ενώ στην κανονική διάσταση εξετάζονται πλήρως οι γείτονες του ανύσματος που προκρίθηκε από το μεσαίο επίπεδο. Τελικά, το εύρος των ανυσμάτων είναι ±16. Η αρχιτεκτονική περιλαμβάνει μνήμη πολλαπλών τραπεζών και βασίζεται σε δισδιάστατες συστοιχίες αφαίρεσης/συσσώρευσης εικονοστοιχείων με σωληνωτή δομή, κάθε μια εκ των οποίων εκτελεί μια πλήρη αναζήτηση ±2 γύρω

[2]προγραμματίσιμο δομικό κελί του Xilinx FPGA. Περιέχει LUTs, πολυπλέκτες, flip-flops.

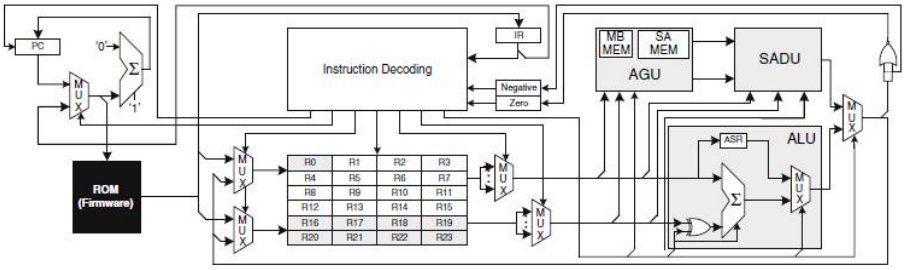
από το υπό αναζήτηση 4×4 υποπλακίδιο (με κριτήριο ομοιότητας το ΑΑΔ). Οι συστοιχίες τροφοδοτούνται από φίλτρα υποδειγματοληψίας, αλλά και από φίλτρα παρεμβολής για αναζήτηση ταιριάσματος σε θέσεις $\pm 1/2$. Η υλοποίηση της [29] σε VLSI (UMC 0.18um CMOS) απαιτεί 59K λογικές πύλες και 1393 bytes μνήμης για επεξεργασία πραγματικού χρόνου σε βίντεο ανάλυσης CIF (352×288).

Περαιτέρω, η βιβλιογραφία παρουσιάζει κι αρχιτεκτονικές για λιγότερο ποιοτικούς ευρετικούς αλγορίθμους [31] [32] [33]. Για παράδειγμα, στην εργασία [31] προτείνεται και υλοποιείται ο αλγόριθμος Two-Step-Search, ο οποίος εξετάζει 25 ομοιόμορφα διάσπαρτες –σταθερές– θέσεις της περιοχής αναζήτησης κι έπειτα εξετάζει τη γειτονιά της επικρατέστερης (όμοια με 3SS). Η αρχιτεκτονική περιέχει τοπικές μνήμες για το τρέχον πλακίδιο και για την περιοχή αναζήτησης (εύρος ± 12), καθώς και δυο επεξεργαστικά στοιχεία για σειριακό υπολογισμό των ΑΑΔ των υποψηφίων. Η υλοποίηση σε FPGA (Altera Cyclone) απαιτεί 1,062 LE[3] κι επιτυγχάνει επεξεργασία βίντεο με ανάλυση 352×240 στα 30 καρέ το δευτερόλεπτο. Στην εργασία [32] προτείνεται και υλοποιείται μια βελτιωμένη έκδοση του αλγορίθμου One-Step-Search, η Fast One-Step-Search (FOSS). Χρησιμοποιεί δυο τοπικές μνήμες για την αποθήκευση του προηγούμενου καρέ και του τρέχοντος πλακιδίου, μια μονάδα για ταυτόχρονη αφαίρεση πολλαπλών εκονοστοιχείων, κι ένα στοιχείο ελέγχου που εκτελεί σειριακά τα βήματα του FOSS. Η υλοποίηση σε FPGA (Altera FLEX 10K100A EPLD) απαιτεί 80K λογικές πύλες κι επεξεργάζεται βίντεο ανάλυσης 1024×768 στα 25 fps.

Προγραμματιζόμενες αρχιτεκτονικές

Ένας ευρετικός αλγόριθμος μπορεί να αποδειχθεί λιγότερο ή περισσότερο αποτελεσματικός σε σχέση με τους υπόλοιπους, ανάλογα με τα χαρακτηριστικά της υπό επεξεργασία εικονοροής (ανάλυση, κινητικότητα, κλπ). Επομένως, σε πολλές εφαρμογές, η επιλογή του κατάλληλου αλγορίθμου αναζήτησης είναι επιθυμητό να βασίζεται στην εμπειρία των χρηστών, παρέχοντας ακόμα και δυνατότητα δυναμικών αλλαγών. Για το σκοπό αυτο η βιβλιογραφία παρουσιάζει έναν αριθμό από προγραμματιζόμενες αρχιτεκτονικές [34] [35] [36] [37] [38] [39] [40] [41] [42] [43] [44]. Στις εργασίες [34] [35] περιγράφονται δυο ξεχωριστές προσεγγίσεις στο ζητούμενο της υποστήριξης διαφορετικών αλγορίθμων. Η πρώτη λύση χρησιμοποιεί μια μικρή μνήμη για την αποθήκευση μοτίβων ψαξίματος, δηλαδή, αποθήκευση σχετικών θέσεων πάνω στην εικόνα με βάση τις

[3] "logic element", δομικό στοιχείο της Altera, αντίστοιχο του Xilinx "slice".



Εικόνα 2.1: Προγραμματιζόμενη αρχιτεκτονική [35] με αποκωδικοποιητή εντολών, αρχείο καταχωρητών, μονάδα ΑΑΔ (SADU), μονάδα αριθμητικών πράξεων (ALU), και τοπικές μνήμες με κυκλώματα διευθυνσιοδότησης (AGU).

οποίες θα κινείται ο αλγόριθμος σε κάθε του βήμα. Η μνήμη τροφοδοτεί ανά διαστήματα τη μονάδα ελέγχου προκειμένου να προκύψουν οι θέσεις των υποψηφίων μπλοκ που προτίθεται να εξετάσει ο εκάστοτε αλγόριθμος. Η προσέγγιση αυτή επιτρέπει την εκτέλεση απλών αλγορίθμων με τακτικά μοτίβα ψαξίματος (π.χ., τετράγωνο, διαμάντι, εξάγωνο, κλπ) τα οποία επαναλαμβάνονται με σταθερό τρόπο καθόλη τη διάρκεια της αναζήτησης. Η δεύτερη λύση εισάγει ένα ειδικό σύνολο εντολών που επιτρέπει την κωδικοποίηση πιο εξελιγμένων και περίπλοκων αλγορίθμων. Οι εντολές υλοποιούν πολύ βασικές λειτουργίες, π.χ. όπως γίνεται σε έναν επεξεργαστή RISC (mov, add, jmp, κ.α.), συν μια ειδικευμένη σύνθετη λειτουργία: η εντολή SAD16 εκτελεί την αφαίρεση μιας ομάδας 16 εικονοστοιχείων του τρέχοντος πλακιδίου από μια αντίστοιχη ομάδα της περιοχής αναζήτησης προκειμένου να προκύψει ένα μερικό ΑΑΔ (το οποίο προορίζεται για μετέπειτα συσσώρευση και σύγκριση). Η προσέγγιση αυτή οδηγεί σε μια αρχιτεκτονική με χαρακτηριστικά επεξεργαστή γενικής χρήσης (Εικόνα 2.1), δηλαδή με αρχείο καταχωρητών και αριθμητική/λογική μονάδα, συν μια μονάδα με επαρκή παραλληλισμό για την υποστήριξη της εντολής SAD16. Σημειώνεται ότι οι λύσεις [34] [35] κάνουν χρήση τοπικής μνήμης για την αποθήκευση της περιοχής αναζήτησης και του τρέχοντος πλακιδίου. Η υλοποίηση των λύσεων σε FPGA (Xilinx Virtex2-Pro) απαιτεί 2,213 και 2,046 slices (λύση μνήμης και λύση εντολών, αντίστοιχα, αφού η πρώτη διαθέτει μονάδα υπολογισμού ΑΑΔ με μεγαλύτερο παραλληλισμό). Ο ρυθμός διεκπεραιωτικότητάς τους επιτρέπει επεξεργασία πραγματικού χρόνου για εικονοροές μεγέθους CIF (352×288 στα 31 fps, για τη λύση με τις εντολές που υποστηρίζουν πιο περίπλοκους/γρήγορους αλγορίθμους).

Στις εργασίες [41] [42] παρουσιάζεται μια προγραμματιζόμενη αρχιτεκτονική που, επιπροσθέτως, είναι κι επαναδιαρθρώσιμη. Χρησιμοποιεί έναν μεταβλητό αριθμό από μονάδες εξέτασης εικονοστοιχείων ή υποεικονοστοιχείων (μονάδες IPEU ή FPEU, αντίστοιχα). Κάθε IPEU είναι κατασκευασμένη με πολλαπλά επίπεδα σωλήνωσης κι ανακτά δεδομένα από τη μνήμη (πλακίδιο και περιοχή αναζήτησης) ώστε να επεξεργάζεται παράλληλα δυο οκτάδες εικονοστοιχείων (με κριτήριο το ΑΑΔ). Παρόμοια δομή εμφανίζει και η FPEU, με τη διαφορά ότι η λειτουργία της προϋποθέτει τον τερματισμό μιας ξεχωριστής υπομονάδας ψηφιακής παρεμβολής. Η υπομονάδα αυτή απαιτεί ένα σεβαστό αριθμό από κύκλους ρολογιού προκειμένου να παράξει τα νέα υποεικονοστοιχεία, διάστημα στο οποίο το υπόλοιπο σύστημα παραμένει ανενεργό (δεν υποστηρίζει παρεμβολή κατά τη μετάβαση, on-the-fly). Προτείνεται εξειδικευμένο σύνολο εντολών για προγραμματισμό σε υψηλό επίπεδο με επιρροές από τη γλώσσα "C". Η υλοποίηση σε FPGA (Xilinx Virtex4) απαιτεί σχεδόν 2K slices ανά μονάδα IPEU και 7K slices ανά μονάδα FPEU. Ο ρυθμός διεκπεραιωτικότητας ρυθμίζεται κατά την τοποθέτηση στο υλισμικό (με αντίστοιχη αύξηση κόστους) διαλέγοντας την κατάλληλη διάρθρωση της αρχιτεκτονικής: π.χ., για επεξεργασία πραγματικού χρόνου πολύ υψηλής ανάλυσης ($1080p@30fps$) με ποιοτικά αποτελέσματα (αλγόριθμος UMH) απαιτούνται 4 μονάδες IPEU.

Ακόμα πιο έντονα χαρακτηριστικά επεξεργαστών γενικής χρήσης εμφανίζουν εργασίες όπως οι [36] [37] [43]. Η λύση [36] υλοποιεί σε FPGA μια αρχιτεκτονική υποστήριξης εντολών μεγάλου μήκους (VLIW processor) με στόχο τον παραλληλισμό σε επίπεδο εντολών. Η αρχιτεκτονική περιλαμβάνει μνήμη πολλαπλών τραπεζών –με χαμηλή όμως εκμετάλλευση– και απαιτεί περίπου 9K slices του FPGA. Στην εργασία [37] ο έλεγχος της λειτουργίας του αλγορίθμου αναζήτησης βασίζεται εξ΄ ολοκλήρου σε έναν αυτόνομο επεξεργαστή RISC που, αναγκαστικά, καταναλώνει πολλούς κύκλους μετάκλησης-αποκωδικοποίησης εντολών (fetch-decode cycles). Η εργασία [43] χρησιμοποιεί έναν επεξεργαστή Media/DSP για την εκτέλεση των αλγορίθμων εκτίμησης κίνησης, η υλοποίηση του οποίου σε VLSI (0.09um process) απαιτεί 8 mm^2. Προς άλλη κατεύθυνση, η βιβλιογραφία περιλαμβάνει λύσεις όπου η αρχιτεκτονική ενσωματώνει συγκεκριμένους αλγορίθμους που μπορούν να επιλεχθούν κατά τη λειτουργία μέσω σημάτων ελέγχου. Για παράδειγμα, στην [44] υιοθετούνται δυο αλγόριθμοι (3SS, Gradient Decent) και προτείνεται ένα αντικειμενικό κριτήριο για την επιλογή τους (με χρήση δυναμικού κατωφλίου). Η αρχιτεκτονική περιλαμβάνει κατάλληλη μονάδα ελέγχου (hardwired control) και πίνακα 9 επεξεργαστικών μονάδων για παράλληλη επεξεργασία των υποψηφίων μπλοκ. Η υλοποίησή της σε VLSI (0.35um CMOS) απαιτεί 0.95 mm^2.

2.2 Παραλληλοποίηση μνήμης για εφαρμογές γραφικών

Το δεύτερο θέμα με το οποίο ασχολείται η παρούσα διατριβή αφορά στην οργάνωση μνήμης με πολλαπλές τράπεζες έτσι, ώστε να καθίσταται δυνατή η παράλληλη και ποικιλότροπη ανάγνωση πολλαπλών δεδομένων. Λαμβάνοντας υπόψη τις γενικές παρατηρήσεις του εισαγωγικού κεφαλαίου για τον τρόπο που χρησιμοποιείται η μνήμη στις εφαρμογές γραφικών προχωράμε σε μια λεπτομερέστερη περιγραφή του προβλήματος που έχουμε να επιλύσουμε.

> Αναζητούμε μέθοδο οργάνωσης μνήμης για αποθήκευση εικόνων, η οποία θα επιτρέπει πρόσβαση σε σύνολα εικονοστοιχείων μέσω τουλάχιστον τριών διαφορετικών σχημάτων: γραμμές, στήλες, και ορθογώνια προκαθορισμένων διαστάσεων. Η πρόσβαση θα πρέπει να είναι απεριόριστη ως προς τη θέση –για οποιοδήποτε σχήμα. Οι διαστάσεις των σχημάτων θα αποτελούν παράμετρο της σχεδίασης.
>
> Περαιτέρω, η οργάνωση θα πρέπει να έχει ως αποτέλεσμα μια μνήμη που θα χρησιμοποιεί όσο το δυνατόν λιγότερο αποθηκευτικό χώρο (bits), που θα εξυπηρετεί τις αιτήσεις σε όσο το δυνατόν λιγότερους κύκλους ρολογιού, και που θα απαρτίζεται από όσο το δυνατόν λιγότερες τράπεζες (memory banks).

Με άλλα λόγια, στοχεύουμε στην εξυπηρέτηση των γενικότερων αναγκών των εφαρμογών γραφικών, ελαχιστοποιώντας ταυτόχρονα τα σημαντικότερα κόστη της εκάστοτε υλοποίησης (χρόνος λειτουργίας και υλικό). Είναι σαφές ότι η ελαχιστοποίηση των κύκλων πρόσβασης μειώνει, δυνητικά, το χρόνο λειτουργίας του αλγόριθμου. Επίσης, η ελαχιστοποίηση του όγκου μνήμης μειώνει το κόστος κατασκευής της. Αυτό που μένει να διευκρινιστεί εδώ είναι ότι σημαντική μείωση κόστους κατασκευής επιτυγχάνεται και μέσω της ελαχιστοποίησης του αριθμού των τραπεζών της μνήμης. Αυτό συμβαίνει, κυρίως, εξαιτίας της μείωσης των μονάδων που απαιτούνται για την υποστήριξη της λειτουργίας μιας τράπεζας μνήμης: πολυπλέκτες, αποπλέκτες, μονάδες διευθυνσιοδότησης κι ελέγχου, κανάλια δρομολόγησης δεδομένων, καθώς κι επίβαρο κόστος (*overhead*) τοποθέτησης.

Η χρήση πολλαπλών τραπεζών παίζει εξαιρετικά σημαντικό ρόλο στο πρόβλημα που μελετάμε. Για το λόγου το αληθές, καταδεικνύουμε το άτοπο μιας προσπάθειας επίλυσης με χρήση μιας μοναδικής τράπεζας δεδομένων. Μια

τέτοια προσπάθεια θα οδηγούσε τα υπόλοιπα δυο ζητούμενα (όγκο και κύκλους) σε μεγιστοποίηση, αντί για ελαχιστοποίηση. Αναφέρουμε ενδεικτικά τις δυο επόμενες –ακραίες– εναλλακτικές. Στην πρώτη, θα έπρεπε να αποθηκεύσουμε κάθε πιθανό σύνολο-αίτηση του αλγόριθμου σε ξεχωριστή θέση μνήμης έτσι, ώστε να είναι δυνατή η ανάκτησή του σε έναν κύκλο ρολογιού. Λόγω όμως της ισχυρής επικάλυψης των εν λόγω συνόλων (όπως αυτή περιγράφηκε στο κεφάλαιο 1) θα χρειαζόμασταν πολλαπλάσιο όγκο μνήμης σε σχέση με τον ελάχιστο, αφού η μνήμη θα έπρεπε να αποθηκεύσει κάθε εικονοστοιχείο πολλαπλές φορές. Στην δεύτερη εναλλακτική, κάθε εικονοστοιχείο θα αποθηκευόταν μια φορά, πιθανώς στη δική του μοναδική διεύθυνση μνήμης. Όμως έτσι, η πλήρης ανάκτηση ενός συνόλου-αίτηση του αλγόριθμου θα απαιτούσε πολλαπλούς κύκλους ρολογιού, δηλαδή πολλούς περισσότερους από τον ελάχιστο ($= 1$). Φυσικά, μπορούμε να κάνουμε συνδυασμούς ανάμεσα στις δυο αυτές εναλλακτικές μειώνοντας τις ακραίες τιμές κύκλων και όγκου. Εύκολα όμως διαπιστώνει κανείς ότι οι συνδυασμοί αυτοί ποτέ δεν οδηγούν τις δυο ζητούμενες μεταβλητές στην ταυτόχρονη ελαχιστοποίησή τους. Συνεπώς, είναι λογικό να περιοριστούμε σε χρήση πολλαπλών τραπεζών και να μιλάμε πλέον για οργάνωση *παράλληλης* μνήμης.

Από τα προηγούμενα παραδείγματα γίνεται φανερό ότι η λύση που αναζητάμε στοχεύει στην αποφυγή εισαγωγής περίσσειας πληροφορίας στη μνήμη, δηλαδή στοχεύει στην αποθήκευση κάθε εικονοστοιχείου σε μια μοναδική θέση. Ο ελάχιστος όγκος μνήμης ισούται με το μέγεθος της αποθηκευμένης εικόνας και, πράγματι, η πλειοψηφία των λύσεων που έχουν προταθεί στη βιβλιογραφία δεν χρησιμοποιούν περισσότερο. Επίσης, γίνεται φανερό ότι η λύση στοχεύει στην δυνατότητα ανάκτησης οποιουδήποτε συνόλου-αίτησης στον ελάχιστο χρόνο, που είναι ένας μοναδικός κύκλος ρολογιού. Ο μοναδικός κύκλος αποτελεί επίσης κοινό χαρακτηριστικό των περισσότερων λύσεων της βιβλιογραφίας, με εξαίρεση βέβαια τις οργανώσεις που βασίζονται σε κρυφές μνήμες (*cache based memories*). Ο ελάχιστος αριθμός τραπεζών δεν είναι εξίσου εύκολο να προσδιοριστεί, όπως έγινε με τους ελάχιστους κύκλους πρόσβασης και τον ελάχιστο όγκο μνήμης. Αρχικά, ο πιο απλοϊκός στόχος είναι να κρατήσουμε τον αριθμό των τραπεζών ίσο με τον αριθμό των εικονοστοιχείων στα οποία θέλουμε να πετύχουμε πρόσβαση ενός κύκλου (ίσο με το εμβαδόν των σχημάτων). Όμως, το ελάχιστο πλήθος τραπεζών σχετίζεται άμεσα με την ελαχιστοποίηση των δυο εναπομείναντων ζητουμένων του προβλήματος (κύκλοι και όγκος), καθώς και με τα επιθυμητά χαρακτηριστικά των σχημάτων πρόσβασης. Η άρρηκτη σχέση μεταξύ όλων αυτών των μεταβλητών γίνεται κατανοητή μέσα από την μελέτη διαφορετικών οργανώσεων μνήμης, όπως περιγράφεται στην επόμενη ενότητα.

bank(x,y) = (4·x + y) mod 8

addr(x,y) = y

y \ x	0	1	2	3	4	5	6	7	8	9	A	B
0	0	4	0	4	0	4	0	4	0	4	0	4
1	1	5	1	5	1	5	1	5	1	5	1	5
2	2	6	2	6	2	6	2	6	2	6	2	6
3	3	7	3	7	3	7	3	7	3	7	3	7
4	4	0	4	0	4	0	4	0	4	0	4	0
5	5	1	5	1	5	1	5	1	5	1	5	1
6	6	2	6	2	6	2	6	2	6	2	6	2
7	7	3	7	3	7	3	7	3	7	3	7	3

Εικόνα 2.2: Αντιστοίχιση εικονοστοιχείων σε τράπεζες μνήμης με βάση τις θέσεις τους στην εικόνα: περιγραφή με συνάρτηση (αριστ.) και πίνακα (δεξιά).

Έχοντας σκιαγραφήσει την κατεύθυνση προς την οποία θα κινηθεί η προτεινόμενη λύση σε ό,τι αφορά στη χρήση πόρων, μπορούμε πλέον να τονίσουμε την κεντρική δυσκολία του προβλήματος που έχουμε να αντιμετωπίσουμε. Η δυσκολία έγκειται στην κατάλληλη αντιστοίχιση των δεδομένων της εικόνας στις διευθύνσεις και στις τράπεζες της παράλληλης μνήμης. Η αντιστοίχιση θα πρέπει να σέβεται τις απαιτήσεις του προβλήματος ώστε να αποφεύγονται οι λεγόμενες 'συγκρούσεις' (*conflicts*) κατά την πρόσβαση στη μνήμη[4]. Συνήθως, μια λύση περιγράφεται από μια συνάρτηση που δέχεται ως είσοδο τη θέση ενός εικονοστοιχείου πάνω στην εικόνα και δίνει ως έξοδο το διακριτικό μιας τράπεζας της μνήμης και μια διεύθυνση μέσα σε αυτήν.

Στην Εικόνα 2.2 απεικονίζεται ενδεικτικά μια αντιστοίχιση των εικονοστοιχείων μιας περιοχής διαστάσεων 12×8 σε μια παράλληλη μνήμη οκτώ τραπεζών. Η αντιστοίχιση σε τράπεζες περιγράφεται εδώ με δυο τρόπους: α) τη συνάρτηση, β) έναν πίνακα (χάρτη) σε κάθε θέση του οποίου αναγράφεται το διακριτικό της τράπεζας στην οποία αποθηκεύεται το εικονοστοιχείο που έχει την αντίστοιχη θέση στην εικόνα. Στο συγκεκριμένο παράδειγμα παρατηρούμε ότι είναι δυνατή η ανάκτηση οποιασδήποτε κολόνας και οποιασδήποτε 2×4 ορθογώνιας περιοχής σε έναν κύκλο ρολογιού, αφού για κάθε πιθανό σύνολο-αίτηση τα εικονοστοιχεία που αυτό περιέχει βρίσκονται αποθηκευμένα σε ξεχωριστές τράπεζες. Αντίθετα, η ανάκτηση οποιασδήποτε γραμμής είναι αδύνατο να επιτευχθεί σε έναν κύκλο, αφού παντού εμφανίζονται συγκρούσεις.

[4]σύγκρουση λέμε ότι έχουμε όταν δυο εικονοστοιχεία που περιέχονται στο ίδιο σύνολο-αίτημα του αλγορίθμου (π.χ., στην ίδια τετράγωνη περιοχή) δεν μπορούν να ανακτηθούν ταυτόχρονα σε έναν κύκλο. Αυτό μπορεί να συμβεί όταν τα εικονοστοιχεία βρίσκονται αποθηκευμένα στην ίδια τράπεζα μνήμης, αλλά σε διαφορετικές διευθύνσεις. Η σύγκρουση έχει ως αποτέλεσμα την καθυστέρηση της εξυπηρέτησης, ή την απώλεια δεδομένων.

2.3 Προγενέστερες λύσεις παράλληλων μνημών

Στην παρούσα ενότητα κάνουμε μια επισκόπηση των λύσεων που έχουν προταθεί κατά καιρούς για την οργάνωση παράλληλης μνήμης. Επικεντρωνόμαστε στις λύσεις που εξυπηρετούν τους ίδιους ή παρεμφερείς σκοπούς με αυτούς που θέσαμε στην προηγούμενη ενότητα. Στόχος της επισκόπησης είναι η αναζήτηση των κατάλληλων σχεδιαστικών κατευθύνσεων και των τεχνικών εκείνων που θα μας οδηγήσουν στην ανάπτυξη μιας αποδεκτής λύσης για το πρόβλημα που έχουμε θέσει. Επιπρόσθετα, η συγκέντρωση και η μελέτη πολλών διαφορετικών λύσεων θα αποκαλύψει τη σχέση μεταξύ των τριών μεταβλητών που προσπαθούμε να ελαχιστοποιήσουμε (όγκος μνήμης, πλήθος τραπεζών, και κύκλοι πρόσβασης), καθώς και τον τρόπο με τον οποίο αυτές επηρεάζουν την υποστηριζόμενη ποικιλία και τις θέσεις των σχημάτων πρόσβασης της μνήμης.

2.3.1 Επισκόπηση βιβλιογραφίας

Η παρουσίαση των δημοσιευμένων λύσεων βασίζεται, αρχικά, στο είδος της συνάρτησης (αντιστοίχισης) που αυτές χρησιμοποιούν. Έπειτα, δίνεται βάρος στις δημιουργούμενες δυνατότητες πρόσβασης, οι οποίες αποτελούν και το ουσιαστικότερο σημείο διαφοροποίησης των λύσεων. Σημειώνουμε εδώ ότι σε κάθε μία από τις παρακάτω οργανώσεις μνήμης θεωρούμε ένα εικονοστοιχείο ανά διεύθυνση μνήμης και μηδενική περίσσεια πληροφορίας (κάθε εικονοστοιχείο αποθηκεύεται σε μια μοναδική θέση στη μνήμη).

Αντιστοίχιση με γραμμική λόξωση

Οι πρώτες δημοσιεύσεις σχετικά με την σχεδίαση παράλληλων μνημών χρονολογούνται στα τέλη της δεκαετίας του ′60, αρχές ′70, κι είχαν ως αφορμή τη σχεδίαση του παράλληλου υπολογιστή ILLIAC IV[5]. Οι Budnick και Kuck [45] μελέτησαν πρώτοι μια τεχνική αντιστοίχισης δεδομένων σε τράπεζες μνήμης, την οποία ονόμασαν *λόξωση* (*skew*). Η ονομασία προκύπτει από τον τρόπο με τον οποίο αποθηκεύεται 'λοξά' κάθε γραμμή εικονοστοιχείων σε σχέση με την προηγούμενη γραμμή και γίνεται κατανοητή μέσα από το επόμενο απλό παράδειγμα.

[5]αναπτύχθηκε στο πανεπιστήμιο του Illinois, USA. Αποτελούνταν από 256 υπολογιστικές μονάδες οι οποίες επεξεργάζονταν παράλληλα μεγάλα σύνολα δεδομένων (*vector processing*).

0	1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8	9
2	3	4	5	6	7	8	9	A
3	4	5	6	7	8	9	A	B
4	5	6	7	8	9	A	B	C
5	6	7	8	9	A	B	C	D
6	7	8	9	A	B	C	D	E
7	8	9	A	B	C	D	E	F

0	1	2	3	0	1	2	3	0
1	2	3	0	1	2	3	0	1
2	3	0	1	2	3	0	1	2
3	0	1	2	3	0	1	2	3
0	1	2	3	0	1	2	3	0
1	2	3	0	1	2	3	0	1
2	3	0	1	2	3	0	1	2
3	0	1	2	3	0	1	2	3

Εικόνα 2.3: Δυο παραδείγματα αντιστοίχισης με γραμμική λόξωση κατα 1: χρήση απεριόριστου αριθμού τραπεζών (αριστερά) και περιορισμένου (δεξιά).

Έστω $\sigma_{x,y}$ το στοιχείο που βρίσκεται στη θέση x, y του πίνακα (ή της εικόνας). Επιλέγουμε να αποθηκεύσουμε διαδοχικά τα στοιχεία της πρώτης γραμμής του πίνακα στις τράπεζες μνήμης αντιστοιχίζοντας τη στήλη του κάθε στοιχείου με το διακριτικό της κάθε τράπεζας, δηλαδή επιλέγουμε να αποθηκεύσουμε το στοιχείο $\sigma_{x,0}$ στη x-οστή τράπεζα. Η δεύτερη γραμμή του πίνακα αποθηκεύεται με τον ίδιο τρόπο, όμως τώρα ξεκινάμε την αντιστοίχιση από τη δεύτερη τράπεζα κι όχι από την πρώτη. Δηλαδή, η δεύτερη γραμμή εμφανίζεται αποθηκευμένη λοξά σε σχέση με την πρώτη γραμμή (σα να ολισθαίνει) κατά ένα στοιχείο. Με τον ίδιο τρόπο αποθηκεύουμε και τις υπόλοιπες γραμμές του πίνακα ακολουθώντας *ομοιόμορφη* λόξωση κατά 1: η τρίτη γραμμή ξεκινάει από την τρίτη τράπεζα, η τέταρτη από την τέταρτη, κ.ο.κ. Στην Εικόνα 2.3 παρουσιάζεται ο παραπάνω τρόπος αντιστοίχισης. Στην περίπτωση που δεν επιβάλουμε περιορισμό στον αριθμό των τραπεζών που έχουμε στη διάθεσή μας οδηγούμαστε σε μεγέθη αντίστοιχα με τις διαστάσεις του πίνακα. Συνήθως περιορίζουμε τις τράπεζες κι αναγκαζόμαστε να συνεχίσουμε τη αντιστοίχιση από το 0 κάθε φορά που εξαντλείται ο διαθέσιμος αριθμός. Η διαδικασία αυτή οδηγεί στο επαναλαμβανόμενο μοτίβο που φαίνεται στη δεύτερη περίπτωση της Εικόνας 2.3 (δεξιά), όπου έχουμε στη διάθεσή μας μόνο 4 τράπεζες.

Είναι σημαντικό να παρατηρήσουμε εδώ ότι η συγκεκριμένη τεχνική δεν καθορίζει από μόνη της τον αριθμό των τραπεζών, αλλά τον χρησιμοποιεί ως παράμετρο αφήνοντας τον σχεδιαστή να προσαρμόσει τη λύση στην εκάστοτε εφαρμογή. Επίσης, η τεχνική δεν καθορίζει το ποσό λόξωσης ούτε το αν αυτή θα εφαρμοστεί κατά γραμμές ή κατά στήλες. Για παράδειγμα, στην Εικόνα 2.2 έχουμε μια ομοιόμορφη λόξωση κατά 4, η οποία εφαρμόστηκε στις στήλες.

Με πιο σύγχρονους όρους, θα περιγράφαμε την τεχνική αυτή μέσω της

συνάρτησης αντιστοίχισης, η οποία είναι της μορφής

$$bank(x, y) = (x + s \cdot y) \bmod B \tag{2.1}$$

όπου B είναι ο αριθμός των τραπεζών (Banks) που έχουμε στη διάθεσή μας και s είναι το ποσό λόξωσης (skew, το πόσο ολισθαίνει η μια γραμμή σε σχέση με την άλλη κατά την αποθήκευσή τους στη μνήμη). Στο αριστερό παράδειγμα της Εικόνας 2.3 έχουμε $s = 1, B = 15$, ενώ στο δεξί έχουμε $s = 1, B = 4$. Σημειώνουμε εδώ ότι η συνάρτηση 2.1 προκύπτει από την πιο γενική μορφή $bank(x, y) = (a \cdot x + b \cdot y + c) \bmod B$ για $a = 1$, $b = s$, $c = 0$. Με βάση τη μορφή αυτή μπορούν να περιγραφούν κι άλλοι τρόποι αντιστοίχισης, που όμως φέρουν παρόμοιες ιδιότητες με αυτούς που ήδη περιγράφονται από την 2.1. Ένα τέτοιο παράδειγμα φαίνεται στην Εικόνα 2.2, όπου έχουμε $a = 4$, $b = 1$, $c = 0$, και το οποίο θα μπορούσε επίσης να περιγραφεί μέσω της 2.1 θέτοντας $s = 4$ και ανταλλάσσοντας το x με το y. Σε κάθε περίπτωση, τονίζουμε τη γραμμικότητα των παραπάνω συναρτήσεων, δηλαδή τη *γραμμική λόξωση* που εμφανίζεται κατά την αποθήκευση.

Μια σημαντική ιδιότητα της γραμμικής λόξωσης είναι η 'ισοτροπικότητα' [46]. Σύμφωνα με αυτήν, αν δυο στοιχεία του πίνακα αποθηκευτούν στην ίδια τράπεζα μνήμης, b_k, τότε και τα αντίστοιχα γειτονικά τους (π.χ., αυτά που βρίσκονται στα αριστερά τους) θα αποθηκευτούν στην ίδια τράπεζα μνήμης, b_l. Όπως θα δούμε σε επόμενη ενότητα, η ιδιότητα αυτή καθίσταται εξαιρετικά σημαντική κατά την υλοποίηση της οργάνωσης της μνήμης. Η τάξη που προσφέρει η ισοτροπικότητα επιτρέπει τη χρήση σχεδιαστικά απλών μονάδων δρομολόγησης των δεδομένων, όπως τους κυκλικούς ολισθητές (barrel shifters).

Πέρα από την ισοτροπικότητα, πιο ουσιώδεις είναι οι ιδιότητες εκείνες που καθορίζουν τα υποστηριζόμενα σχήματα πρόσβασης. Καταρχάς, είναι εύκολο να δει κανείς ότι για μηδενική λόξωση, $s = 0$, μπορούμε να ανακτήσουμε οποιαδήποτε γραμμή ή διαγώνιο μήκους B, από όπου κι αν αυτή ξεκινά. Αυτό γιατί, εξ' ορισμού, έχουμε διαλέξει τα $B - 1$ στοιχεία που βρίσκονται δεξιά του τυχαίου $\sigma_{x,y}$ να βρίσκονται αποθηκευμένα σε ξεχωριστές τράπεζες. Παρόμοια, με λόξωση κατά ένα, $s = 1$, μπορούμε να ανακτήσουμε οποιαδήποτε γραμμή ή στήλη μήκους B, από όπου κι αν αυτή ξεκινά (π.χ., Εικόνα 2.3). Μάλιστα, η ιδιότητα αυτή ισχύει γενικότερα όταν το s και το B είναι αριθμοί πρώτοι μεταξύ τους[6]. Αντίθετα, όταν το s και το B έχουν μέγιστο κοινό διαιρέτη το

[6]τετριμμένο για τις γραμμές. Για τις στήλες αρκεί να σκεφτούμε σε ποια ολίσθηση (γραμμής) επανερχόμαστε για πρώτη φορά στο 0. Έστω στην j-οστή, όπου έχουμε $j \cdot s = i \cdot B$, j, i ακέραιοι. Αφού όμως gcd(s,B)=1, οι s και B διαφέρουν ως προς όλους τους παράγοντές τους και, άρα, ο μόνος τρόπος να εξισωθούν τα δυο μέλη είναι όταν $j = B$ (και $i = s$).

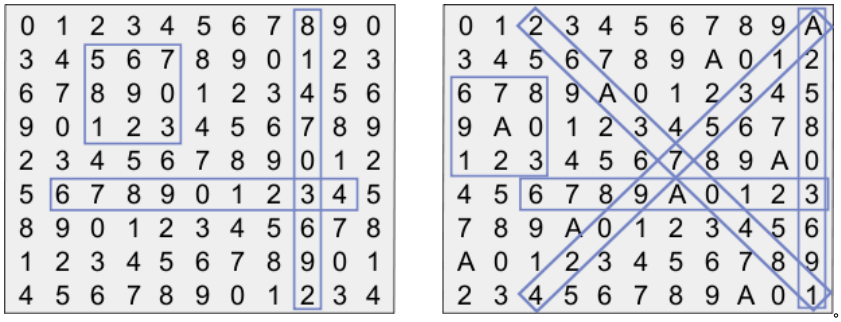
$q \neq 1$, τότε αντί για στήλες μπορούμε να ανακτήσουμε ορθογώνιες περιοχές διαστάσεων $q \times \frac{B}{q}$ από οπουδήποτε πάνω στον πίνακα (π.χ., Εικόνα 2.2)[7]. Έτσι, για πρόσβαση σε B στοιχεία είτε κατά τετράγωνα, είτε κατά γραμμές, αρκεί να θέσουμε $s = \sqrt{B}$.

Συνοψίζοντας τα παραπάνω, στη γραμμική λόξωση μπορούμε να αλλάξουμε την τιμή του s ώστε να αποκτήσουμε πρόσβαση σε B στοιχεία της εικόνας κατά {γραμμές,διαγώνιους}, {γραμμές,στήλες}, ή {γραμμές,ορθογώνια}. Η πρόσβαση σε οποιοδήποτε από τα προαναφερθέντα σχήματα καθίσταται απεριόριστη ως προς τη θέση τους πάνω στην εικόνα. Παρατηρούμε ότι για να υποστηρίξουμε αυτές τις δυνατότητες χρειαζόμαστε μόνο B τράπεζες. Όμως, παραμένει ανοιχτό ένα από τα αρχικά ζητούμενα, η δυνατότητα πρόσβασης μέσω τριών διαφορετικών σχημάτων στην ίδια μνήμη: {γραμμές,στήλες,ορθογώνια}.

Δυστυχώς, η δυνατότητα χρήσης και των τριών σχημάτων με απεριόριστη πρόσβαση αποδεικνύεται αδύνατη να επιτευχθεί με χρήση B τραπεζών μόνο [47]. Συνεπώς, αναγκαζόμαστε να χρησιμοποιήσουμε αριθμό τραπεζών μεγαλύτερο από το εμβαδόν των σχημάτων, δηλαδή $B > E$, όπου E είναι ο αριθμός των στοιχείων στα οποία θέλουμε να έχουμε άμεση πρόσβαση σε έναν κύκλο. Τονίζουμε ότι το αποτέλεσμα αυτό ισχύει για οποιοδήποτε τρόπο αντιστοίχισης (με μηδενική περίσσεια κι ένα στοιχείο ανά διεύθυνση) κι όχι μόνο για τη λόξωση.

Φυσικά, η πρώτη επιλογή είναι να χρησιμοποιήσουμε μια μόνο επιπλέον τράπεζα, δηλαδή $B = E + 1 = m \cdot n + 1$, όπου m και n είναι οι διαστάσεις του ορθογωνίου που θέλουμε να χρησιμοποιήσουμε ως σχήμα πρόσβασης. Με βάση τις ιδιότητες που παρουσιάσαμε παραπάνω, αρκεί να επιλέξουμε εδώ γραμμική λόξωση $s = m$ προκειμένου να αποκτήσουμε τις επιθυμητές δυνατότητες πρόσβασης: γραμμές μήκους B έχουμε εξ' ορισμού, στήλες έχουμε γιατί το s και το B (το m και το $m \cdot n + 1$) είναι αριθμοί πρώτοι μεταξύ τους, και ορθογώνια έχουμε γιατί κάθε γραμμή m στοιχείων μέσα στο ορθογώνιο διαφέρει από την επόμενή της, εξαιτίας της σχετικής ολίσθησης κατά m που έχει υποστεί ολόκληρη η γραμμή της εικόνας (βέβαια, το τελευταίο απαιτεί μια τυπική απόδειξη με πιο διακριτά επιχειρήματα [47]). Αριστερά στην Εικόνα 2.4 φαίνεται ένα παράδειγμα με $E = 9$ και $B = 10$ όπου έχουμε εφαρμόσει λόξωση $s = 3$ και μπορούμε να ανακτήσουμε γραμμές και στήλες μήκους 9, καθώς και τετράγωνα 3×3, από οπουδήποτε πάνω στο επίπεδο.

[7] ακολουθώντας το σκεπτικό της προηγούμενης υποσημείωσης παρατηρούμε ότι μπορούμε να ανακτήσουμε στήλες με μέγιστο μήκος B/q. Συγκεκριμένα, οι κοινοί παράγοντες (το q) απαλείφονται από τα δυο μέλη της εξίσωσης και, άρα, ισότητα έχουμε μόνο όταν $j = B/q$.



Εικόνα 2.4: Δυνατότητες πρόσβασης με χρήση γραμμικής λόξωσης και περίσσειας τραπεζών ($B > E$). Αριστερά: E=9, B=10. Δεξιά: E=9, B=11.

Ως γνωστόν, πολλές φορές κατά τη σχεδίαση των ψηφιακών κυκλωμάτων επιλέγουμε να χρησιμοποιήσουμε αριθμούς ίσους με δυνάμεις του 2. Η συνετή αυτή επιλογή για τις διαστάσεις των σχημάτων και το εμβαδόν E μπορεί να οδηγήσει τον αριθμό των τραπεζών $B = E + 1 = 2^k + 1$ σε κάποιον πρώτο αριθμό $(3, 5, 17, 257, \ldots)$. Μάλιστα, η βιβλιογραφία περιέχει λύσεις που προτείνουν κατευθείαν τη χρήση ενός πρώτου αριθμού τραπεζών για την οργάνωση της μνήμης [48]. Ο αριθμός αυτός τυχαίνει μερικές φορές να είναι κατά λίγο μεγαλύτερος του επιθυμητού εμβαδού (π.χ., $E = 4$, $B = 5$), κι άλλοτε κατά πολύ (π.χ., $E = 512$, $B = 521$). Αξίζει λοιπόν να τονίσουμε εδώ ότι η οργάνωση με χρήση πρώτων αριθμών προσφέρει επιπλέον δυνατότητες πρόσβασης, αφού επιτρέπει χρήση περισσότερων σχημάτων, όπως διαγώνιες γραμμές (δυο κατευθύνσεων). Δεξιά στην Εικόνα 2.4 φαίνεται ένα παράδειγμα όπου χρησιμοποιούμε δυο τράπεζες παραπάνω από το E προκειμένου να φτάσουμε τον πρώτο αριθμό 11. Γίνεται σαφές μέσω της αντιπαραβολής με το διπλανό παράδειγμα ($B = 10$) πως οι ιδιότητες της γραμμικής λόξωσης εξαρτώνται άμεσα από τον διαθέσιμη 'περίσσεια' σε αριθμό τραπεζών.

Η χρήση πρώτων αριθμών μπορεί να φαίνεται ελκυστική λύση λόγω των παραπάνω ιδιοτήτων, όμως στην πραγματικότητα κρύβει πολλές δυσκολίες κατά την υλοποίηση. Οι δυσκολίες μεταφράζονται σε χρήση πολλών υπολογιστικών πόρων, κυρίως γιατί οι μονάδες ελέγχου και διευθυνσιοδότησης πρέπει να εκτελούν πράξεις ακέραιας διαίρεσης (modulo, div) με πρώτους αριθμούς. Για την αποφυγή ακριβώς αυτών των διαιρέσεων έχουν προταθεί οργανώσεις μνήμης που χρησιμοποιούν ακόμα και το διπλάσιο αριθμό από τράπεζες (που προφανώς δεν είναι πρώτος), δηλαδή $B{=}2{\cdot}E$ [49]. Φυσικά, η μεγάλη περίσσεια τραπεζών υποκρύπτει το δικό της μεγάλο επίβαρο κόστος υλοποίησης και δημιουργεί

φαινόμενα υπο-εκμετάλλευσης (underutilization) των πόρων μνήμης από την υπόλοιπη αρχιτεκτονική.

Αντιστοίχιση με μη-γραμμική λόξωση

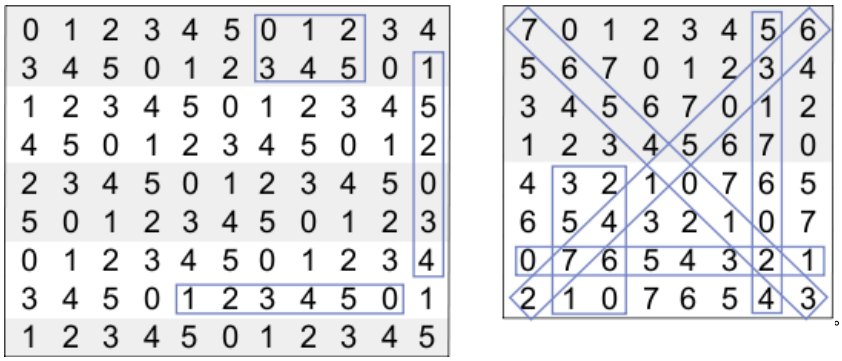
Η απλή, γραμμική, μορφή των συναρτήσεων αντιστοίχισης αποτελεί μόνο ένα μικρό υποσύνολο των λύσεων που έχουμε στη διάθεσή μας για την οργάνωση της μνήμης. Πολλές από τις λύσεις της βιβλιογραφίας εκφράζονται μέσα από μη-γραμμικές συναρτήσεις. Στην παράγραφο αυτή αναφέρουμε μερικές αντιπροσωπευτικές, που όμως διατηρούν ακόμα σχετικά εμφανή τα χαρακτηριστικά της λόξωσης (skew). Με άλλα λόγια, εξετάζουμε συναρτήσεις που η μορφή τους δε διαφέρει δραματικά από αυτή της 2.1.

Ξεκινάμε ιστορικά, με μια λύση που χρησιμοποιεί τον ελάχιστο αριθμό τραπεζών, δηλαδή $B = E = m \cdot n$ [47], όπου m και n είναι η οριζόντια και η κατακόρυφη διάσταση του επιθυμητού ορθογωνίου πρόσβασης. Η αντιστοίχιση γίνεται μέσω της συνάρτησης

$$bank(x, y) = \left(x + m \cdot y + \left\lfloor \frac{y}{n} \right\rfloor \bmod m\right) \bmod B \tag{2.2}$$

Αυτό που συμβαίνει με την εφαρμογή της 2.2 είναι, ουσιαστικά, και πάλι μια ολίσθηση των γραμμών κατά m, μόνο που αυτή τη φορά η ολίσθηση αυξάνει επιπλέον κατά ένα στοιχείο κάθε φορά που αποθηκεύουμε n συνεχόμενες γραμμές. Δηλαδή, εκτός από ολίσθηση γραμμών έχουμε και ολίσθηση λωρίδων πάχους n γραμμών (προσθετικός παράγοντας $\lfloor \frac{y}{n} \rfloor$). Παρατηρούμε επίσης ότι η ολίσθηση λωρίδων περιορίζεται σε m φορές (πράξη $\bmod m$ στον προσθετικό παράγοντα). Ένα παράδειγμα φαίνεται αριστερά στην Εικόνα 2.5, όπου έχουμε $m = 3, n = 2$. Παρατηρούμε ότι εντός κάθε λωρίδας πάχους n έχουμε μια γραμμική ολίσθηση κατά m, όπου m είναι διαιρέτης του B. Συνεπώς, σύμφωνα με τις ιδιότητες που παρουσιάσαμε στην προηγούμενη παράγραφο, εντός κάθε λωρίδας έχουμε απεριόριστη πρόσβαση σε γραμμές και ορθογώνια $m \times n$. Επιπρόσθετα, η σχετική ολίσθηση των λωρίδων επιτρέπει απεριόριστη πρόσβαση σε στήλες μήκους B. Συνοψίζουμε λοιπόν ότι μέσω της 2.2 έχουμε αποκτήσει απεριόριστη πρόσβαση σε γραμμές και στήλες, καθώς και περιορισμένη πρόσβαση σε ορθογώνια, χρησιμοποιώντας τον ιδανικό αριθμό τραπεζών, $B = E$. Σαφώς, αυτή είναι μια βελτίωση σε σχέση με την απλή γραμμική λόξωση. Από την άλλη πλευρά, η ελαφρώς πιο περίπλοκη μορφή της 2.2 από την 2.1 οδηγεί σε μικρή αύξηση του κόστους υλοποίησης των μονάδων ελέγχου του κυκλώματος.

Με μια πιο εποπτική ματιά θα μπορούσαμε να πούμε ότι η προηγούμενη τεχνική χωρίζει την εικόνα σε ξεχωριστά κομμάτια (λωρίδες πάχους n γραμμών)



Εικόνα 2.5: Μη-γραμμική λόξωση με ελάχιστο αριθμό τραπεζών. Αριστερά, σύμφωνα με τη συνάρτηση 2.2 (m=3, n=2). Δεξιά, σύμφωνα με τη 2.3 (B=8).

και εφαρμόζει σε κάθε ένα από αυτά μια διαφορετική λόξωση, αλλά της ίδιας γενικότερης μορφής. Η ιδέα αυτή γενικεύεται, δηλαδή η εικόνα μπορεί να χωριστεί αυθαίρετα και σε κάθε κομμάτι της να χρησιμοποιηθεί εντελώς ξεχωριστή αντιστοίχιση. Εδώ αναφέρουμε μια σχετικά απλή εφαρμογή της ιδέας αυτής, η οποία προορίζεται περισσότερο για επεξεργασία τετράγωνων πινάκων κι όχι για εικόνες. Σύμφωνα με τη λεγόμενη 'πολυλόξωση' (*multiskewing*) [50] χωρίζουμε τον $B \times B$ πίνακα στο πάνω μισό και στο κάτω μισό μέρος του. Στο πάνω κομμάτι εφαρμόζουμε μια απλή γραμμική λόξωση, ενώ στο κάτω εφαρμόζουμε μια ανάποδη γραμμική λόξωση, δηλαδή, αντιστοιχούμε στις τράπεζες τα στοιχεία κάθε γραμμής χρησιμοποιώντας φθίνουσα σειρά, κι όχι αύξουσα. Συγκεκριμένα, έχουμε την εξής συνάρτηση αντιστοίχισης

$$bank(x,y) = \begin{cases} (x - 2{\cdot}y - 1) \bmod B, & \text{αν } y < {}^{B}/_{2} \\ (2{\cdot}y - x - {}^{B}/_{2}) \bmod B, & \text{αν } y \geq {}^{B}/_{2} \end{cases} \tag{2.3}$$

Δεξιά στην Εικόνα 2.5 φαίνεται ένα παράδειγμα για μέγεθος πίνακα 8×8. Παρατηρούμε ότι εκτός της απεριόριστης πρόσβασης σε γραμμές και στήλες μήκους B, έχουμε πρόσβαση σε ορθογώνια $2 \times {}^{B}/_{2}$ που βρίσκονται εντός μιας λωρίδας (της πάνω ή της κάτω). Όμως, το πλεονέκτημα της 2.3 σε σχέση με την 2.2 είναι η δυνατότητα περιορισμένης πρόσβασης σε κύριες διαγωνίους (δυο κατευθύνσεων), καθώς επίσης και σε μια ποικιλία από παράξενα σχήματα που περιέχουν $B{\cdot}k$ στοιχεία και που μπορούν να συλλεχθούν σε ακριβώς k κύκλους (βέλτιστος χρόνος) [50]. Το μειονέκτημα της 2.3 είναι η ελαφρώς μεγαλύτερη πολυπλοκότητα των κυκλωματικών μονάδων που την υλοποιούν.

Αντιστοιχίσεις σαν αυτή που φαίνεται δεξιά στην Εικόνα 2.5, στην πράξη,

επαναλαμβάνονται κατά την οριζόντια και κατακόρυφη διεύθυνση προκειμένου να αποθηκευτεί μια εικόνα με μεγαλύτερες διαστάσεις στη μνήμη. Έχουμε δηλαδή ένα 'βασικό πεδίο' διαστάσεων $p \times q$ και το χρησιμοποιούμε ανά p γραμμές και q στήλες για να κάνουμε αντιστοίχιση όλων των εικονοστοιχείων. Το βασικό πεδίο μπορεί να είναι πολύ απλό, όπως για παράδειγμα μια γραμμική λόξωση κατά p: $bank(x, y) = x \bmod p + y \bmod q$, με απεριόριστη πρόσβαση μόνο σε ορθογώνια $p \times q$ [51]. Εναλλακτικά, το βασικό πεδίο μπορεί να είναι πιο περίπλοκο, όπως αυτό της συνάρτησης 2.3. Σε κάθε περίπτωση, η αντιστοίχιση που προκύπτει με αυτόν τον τρόπο για ολόκληρη την εικόνα είναι μια *περιοδική* συνάρτηση. Μπορούμε να γενικεύσουμε την ιδέα αυτή λίγο περισσότερο και να χρησιμοποιήσουμε *πολυ-περιοδικές* συναρτήσεις [46]. Εδώ, το βασικό πεδίο δεν επαναλαμβάνεται απαράλλαχτο, αλλά έχοντας υποστεί μια μετάθεση στοιχείων (permutation), π.χ., $\pi_1^k(Z) = (Z + k \cdot 5) \bmod B$, όπου $k = \lfloor \frac{x}{p} \rfloor$ είναι ο αριθμός της οριζόντιας επανάληψης στην οποία βρισκόμαστε. Βέβαια, μεταθέσεις μπορεί να έχουμε και κατά τις επαναλήψεις μας στη κατακόρυφη διεύθυνση ως π_2^l με $l = \lfloor \frac{y}{q} \rfloor$ (γενικά, οι π_1 και π_2 είναι διαφορετικές). Οι δυο μεταθέσεις συνδυάζονται κι έτσι προκύπτει η γενική μορφή των 'πολυπεριοδικών' αντιστοιχίσεων: $bank(x, y) = \pi_1^k \pi_2^l S(x', y')$, όπου $S(x', y')$ με $x' = x \bmod p$ και $y' = y \bmod q$ είναι η αντιστοίχιση που έχουμε εντός του βασικού πεδίου. Παραδείγματα 'πολυπεριοδικών' αντιστοιχίσεων έχουν εμφανιστεί στη βιβλιογραφία για χρήση σε εφαρμογές γραφικών, αλλά είναι στοχευμένες σε συγκεκριμένες ανάγκες [52].

Η μη-γραμμική λόξωση είναι δυνατόν να εφαρμοστεί και με χρήση περίσσειας τραπεζών. Ένα χαρακτηριστικό παράδειγμα είναι η λύση που προτείνεται στην εργασία [49] προκειμένου να αποφευχθεί η χρήση του πρώτου αριθμού τραπεζών της εργασίας [48]. Εδώ, επιλέγεται διπλάσιος αριθμός $B = 2 \cdot E = 2 \cdot m \cdot n$ και αντιστοίχιση $bank(x, y) = \left(\lfloor \frac{y}{2m} \rfloor \cdot (n + 1) + (y \bmod (2m)) \cdot n + x \right) \bmod B$, όπου m και n η οριζόντια και η κατακόρυφη διάσταση του ορθογωνίου πρόσβασης. Η λύση αυτή πετυχαίνει τις ίδιες δυνατότητες πρόσβασης με εκείνες της εργασίας [48], αλλά απαιτεί πολλές τράπεζες και δεν καταφέρνει να περιορίσει τις αριθμητικές πράξεις μόνο σε αριθμούς της μορφής $2^k, k \in \mathbb{N}$.

Συναρτήσεις που χρησιμοποιούν αποκλειστική διάζευξη

Μια ειδική κατηγορία μη-γραμμικών, ανισοτροπικών, συναρτήσεων αντιστοίχισης είναι εκείνες που βασίζονται σε υπολογισμούς αποκλειστικής διάζευξης (XOR). Εδώ, οι συνιστώσες x και y του διανύσματος θέσης του εικονοστοιχείου αναπαρίστανται σε δυαδική μορφή και εισέρχονται στην είσοδο απλών λογικών

συναρτήσεων, η αποτίμηση των οποίων καθορίζει τα διφύα του διακριτικού της τράπεζας που αντιστοιχείται το (x, y).

Οι περισσότερες συναρτήσεις αντιστοίχισης με XOR στη σχεδίαση παράλληλων μνημών δίνονται μέσω πινάκων και ακολουθούν τη γενική μορφή [46]

$$BANK(x, y) = C \cdot X_u \oplus D \cdot Y_v \tag{2.4}$$

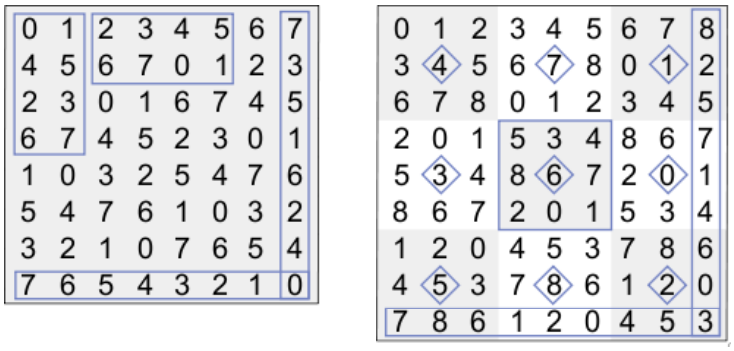
όπου οι πίνακες $X_u = \langle x \bmod u \rangle_2$ και $Y_v = \langle y \bmod v \rangle_2$ έχουν διαστάσεις $u \times 1$ και $v \times 1$ και περιέχουν τα u και v πρώτα διφύα των μεταβλητών x και y, αντίστοιχα. Οι σταθεροί δυαδικοί πίνακες C και D έχουν διαστάσεις $k \times u$ και $k \times v$, αντίστοιχα, όπου k είναι ο αριθμός των διφύων του διακριτικού της τράπεζας, δηλαδή η διάσταση του δυαδικού πίνακα-στήλη $BANK(x, y)$. Το σύμβολο $\oplus$ στην 2.4 υποδηλώνει πράξη XOR, ενώ ο δυαδικός πολλαπλασιασμός πινάκων υλοποιείται με πράξεις AND και XOR (δηλαδή, στο GF(2)).

Η μορφή της 2.4 υποδηλώνει ότι, πρώτον, χρησιμοποιούμε ένα υποσύνολο των διφύων των x και y για να κάνουμε την αντιστοίχιση. Για παράδειγμα, αν έχουμε στη διάθεσή μας 8 τράπεζες τότε χρησιμοποιούμε τα τρία πρώτα διφύα ($u = v = k = 3$). Δεύτερον, παρατηρούμε ότι αυτό που πραγματικά διαφοροποιεί τις XOR οργανώσεις μνήμης μεταξύ τους είναι οι τιμές των σταθερών πινάκων C και D. Αλλάζοντας τους C και D αποκτούμε δυνατότητες πρόσβασης κατά γραμμές, στήλες, τετράγωνα, κ.α. Η πιο απλή λύση αντικαθιστά τους C και D με τον μοναδιαίο πίνακα I δημιουργώντας την αντιστοίχιση $bank(x, y) = x \oplus y$. Για την ιστορία, η αντιστοίχιση αυτή είναι μια από τις πρώτες που εφαρμόστηκαν κι επέτρεπε στον παράλληλο υπολογιστή STARAN[8] απεριόριστη πρόσβαση στη μνήμη κατά γραμμές και στήλες. Ως παράδειγμα αναφέρουμε μια παρεμφερή οργάνωση που χρησιμοποιεί τον μοναδιαίο I και τον αντι-ανάστροφό του [46]. Για την περίπτωση 8 τραπεζών παίρνουμε την ακόλουθη συνάρτηση και την οργάνωση που φαίνεται αριστερά στην Εικόνα 2.6 (με δείκτη 0 συμβολίζουμε τα διφύα μικρότερης αξίας)

$$BANK(x, y) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ x_1 \\ x_0 \end{bmatrix} \oplus \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} y_2 \\ y_1 \\ y_0 \end{bmatrix} = \begin{bmatrix} x_2 \\ x_1 \\ x_0 \end{bmatrix} \oplus \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}$$

Η παραπάνω διάταξη επιτρέπει απεριόριστη πρόσβαση κατά γραμμές και στήλες, καθώς και περιορισμένη πρόσβαση κατά ορθογώνια διαφορετικών προσανατολισμών. Στη βιβλιογραφία περιέχεται πλήθος παρόμοιων λύσεων με πιο χαρακτηριστική αυτή του Frailong [53], η οποία πετυχαίνει πρόσβαση κατά γραμμές,

[8] αποτελούνταν από $n \times 256$, $n \leq 32$, επεξεργαστικές μονάδες ενός διφύου (με αρχιτεκτονική SIMD) και χρησιμοποιούσε συσχετιστική μνήμη (CAM). Goodyear Aerospace Corp., 1972.



Εικόνα 2.6: XOR-αντιστοίχιση (αριστ.), Τέλειο Λατινικό Τετράγωνο (δεξιά).

στήλες, τετράγωνα, ορθογώνια, ακόμα και κατά αραιά τετράγωνα (τη λεγόμενη 'σκακιέρα') όταν ο αριθμός των τραπεζών είναι άρτια δύναμη του 2. Συνοπτικά, η τεχνική του συνίσταται στην αρχική αντιστοίχιση των δεδομένων ώστε να αποφεύγονται οι συγκρούσεις για κάποια βασικά σχήματα πρόσβασης, κι έπειτα στην τροποποίηση της συνάρτησης προκειμένου να συμπεριλάβει δευτερεύοντα σχήματα.

Αξίζει να σημειωθεί ότι πολλές από τις δημοσιευμένες XOR λύσεις χρησιμοποιούν συναρτήσεις με μικρή πολυπλοκότητα, και ειδικά σε περιπτώσεις όπως αυτή της Εικόνας 2.6 μπορούν να υλοποιηθούν ακόμα και με κυκλώματα σταθερού βάθους. Στα μειονεκτήματά τους συγκαταλέγεται η ανισοτροπικότητα και η δυσκολία κατασκευής/επιλογής της συνάρτησης που οδηγεί στη ζητούμενη ποικιλία σχημάτων. Η κατασκευή αυτή είναι αρκετά διαισθητική και πολλές φορές επιτυγχάνεται μόνο μέσα από κάποια διαδικασία δοκιμής και λάθους. Για το λόγο αυτό έχουν προταθεί συγκεκριμένες μεθοδολογίες, οι οποίες έχουν ως στόχο την απλοποίηση της κατασκευής μιας XOR αντιστοίχισης. Ένα χαρακτηριστικό παράδειγμα αυτής της προσπάθειας είναι η αλλαγή του τρόπου αναπαράστασης της ίδιας της XOR αντιστοίχισης: από την κλασσική αναπαράσταση μέσω πίνακα περνάμε στο 'χώρο-μηδέν' (*null space*) και στο 'χώρο-στήλη' (*column space*) [54]. Ο πρώτος βοηθά σε μια πιο διαισθητική αντιστοίχιση σε σχήματα χωρίς συγκρούσεις, ενώ ο δεύτερος βοηθά –επιπροσθέτως– στην κατασκευή συναρτήσεων ειδικών προδιαγραφών, π.χ., στη χρήση πυλών XOR με λίγες εισόδους (fan-in).

Λατινικά Τετράγωνα

Ένα 'Λατινικό Τετράγωνο' (ΛΤ) τάξης k ορίζεται ως ένας τεράγωνος πίνακας $k \times k$ που περιέχει στοιχεία από το σύνολο $\{0, 1, \ldots, k-1\}$ έτσι, ώστε κανένα στοιχείο δεν εμφανίζεται δυο φορές σε οποιαδήποτε γραμμή ή οποιαδήποτε στήλη του πίνακα. Από τον ορισμό γίνεται αμέσως σαφής η ομοιότητα του ΛΤ με τους πίνακες/χάρτες που παρουσιάστηκαν στις παραπάνω Εικόνες για την αντιστοίχιση των δεδομένων στη μνήμη. Είναι λοιπόν λογικό τα συνδυαστικά αυτά αντικείμενα να έχουν χρησιμοποιηθεί στην πράξη για τη σχεδίαση παράλληλων μνημών και ως τέτοια αναφέρονται στην παρούσα παράγραφο.

Τα ΛΤ ήταν γνωστά ήδη από το μεσαίωνα, ενώ το όνομά τους είναι μεταγενέστερο και οφείλεται στον Leonhard Euler, ο οποίος για τη μελέτη τους τοποθετούσε λατινικούς χαρακτήρες σε δισδιάστατο πλέγμα. Πέραν των μνημών, βέβαια, τα ΛΤ βρίσκουν εφαρμογή σε πολλά διαφορετικά πεδία, όπως, π.χ., στους κώδικες διόρθωσης σφαλμάτων, στη στατιστική, ακόμα και στα ψυχαγωγικά μαθηματικά [55]. Ανάλογα με το σκοπό της χρήσης τους διακρίνουμε διάφορες κατηγορίες ΛΤ, οι οποίες φέρουν –επιπλέον– τις δικές τους ιδιότητες. Για παράδειγμα, έχουμε τα 'Διαγώνια ΛΤ', στα οποία επιβάλλεται ο επιπρόσθετος περιορισμός ότι οι δυο κύριες διαγώνιοι είναι ελεύθερες συγκρούσεων (όπως οι γραμμές και οι στήλες). Έχουμε επίσης τα ΛΤ που χρησιμοποιούνται στο πασίγνωστο παιχνίδι SUDOKU, όπου επιβάλλεται η αποφυγή συγκρούσεων σε οποιοδήποτε από τα κύρια υπο-τετράγωνα του ΛΤ. Ακόμα περισσότερο ενδιαφέρον παρουσιάζει η τομή των δυο προαναφερθέντων κατηγοριών, η οποία περιέχει τα λεγόμενα 'Τέλεια Λατινικά Τετράγωνα' (ΤΛΤ) που επιτρέπουν πρόσβαση κατά γραμμές, στήλες, κύριες διαγώνιους και κύρια υπο-τετράγωνα.

Η μελέτη της ύπαρξης και της κατασκευή ΛΤ με συγκεκριμένες ιδιότητες έχει οδηγήσει σε δημοσίευση πλήθους εργασιών [55]. Εδώ στεκόμαστε στους Kim και Prasanna που ονόμασαν τα ΤΛΤ και μελέτησαν την κατασκευή τους με σκοπό τη χρήση σε εφαρμογές επεξεργασίας πινάκων (όχι γραφικών) [56]. Στην εργασία τους δείχνουν ότι μοναδική συνθήκη ύπαρξης ενός ΤΛΤ τάξης n^2 είναι η ύπαρξη δυο ΛΤ τάξης n, έστω A και B, τέτοια ώστε το A να είναι ορθογώνιο[9] με το ανάστροφο και με το αντι-ανάστροφο του B. Επίσης, παρέχουν μια μέθοδο για την κατασκευή ενός ΤΛΤ από τα A και B. Η μέθοδος αυτή οδηγεί σε επιπλέον δυνατότητες πρόσβασης (εκτός από αυτές που προκύπτουν από τον ορισμό των ΤΛΤ). Συγκεκριμένα, αποκτούμε άμεση πρόσβαση σε:

[9]δυο Λατινικά Τετράγωνα A και B τάξης k είναι ορθογώνια μεταξύ τους όταν ο πίνακας $\Gamma_{k\times k}$ με στοιχεία $\gamma_{i,j} = \langle a_{i,j}, b_{i,j}\rangle$ περιέχει k^2 διαφορετικά στοιχεία (κάθε διατεταγμένο ζεύγος $\langle a_{i,j}, b_{i,j}\rangle$ είναι μοναδικό). Ο πίνακας $\Gamma_{k\times k}$ ονομάζεται 'Ελληνο-Λατινικό Τετράγωνο'.

α) τετράγωνα $n \times n$ τα οποία βρίσκονται σε θέσεις (x, y) πάνω στον πίνακα με $x \bmod n = 0$, ή $y \bmod n = 0$, (δηλαδή το τετράγωνο πρέπει να βρίσκεται ολόκληρο εντός μιας βασικής λωρίδας πάχους n) και β) 'αραιά τετράγωνα' χωρίς περιορισμό θέσης (αυτό το σχήμα πρόσβασης αποτελείται από εικονοστοιχεία τοποθετημένα σε τετράγωνο πλέγμα με απόσταση n μεταξύ τους). Δεξιά στην Εικόνα 2.6 φαίνεται ένα ΤΛΤ τάξης 9 $(= 3^2)$ με τα 9 κύρια υπο-τετράγωνά του κι ένα αραιό τετράγωνο (συμβολίζεται με ρόμβους).

Ένα σημαντικό πλεονέκτημα της χρήσης ΤΛΤ για την οργάνωση της παράλληλης μνήμης είναι η μικρή πολυπλοκότητα των υπολογισμών διευθυνσιοδότησης. Συγκεκριμένα, όταν ο αριθμός των τραπεζών είναι άρτια δύναμη του 2, η διευθυνσιοδότηση μπορεί να υλοποιηθεί με κυκλώματα σταθερού βάθους [56]. Στα μειονεκτήματα συγκαταλέγεται η ανισοτροπικότητα της αντιστοίχισης και ο περιορισμός που έχουμε για τον αριθμό των τραπεζών: πρέπει να είναι της μορφής n^2, όπου n περιττός, ή δύναμη του 2, ή της μορφής $2^l m^2$ με $l \geq 2$ και m περιττό.

Αντιστοίχιση με πλέγμα Fibonacci

Σε μια προσπάθεια προσδιορισμού του ελαχίστου αριθμού τραπεζών μνήμης για εφαρμογές γραφικών, οι Chor, Leiserson, Rivest, και Shearer, δημοσίευσαν το 1986 μια θεωρητική εργασία με αξιοσημείωτα αποτελέσματα [57]. Η υπόθεση που κάνουν είναι ότι ο αλγόριθμος μπορεί να απαιτήσει απεριόριστη πρόσβαση ενός κύκλου κατά γραμμές, στήλες, ή ορθογώνια με αυθαίρετες διαστάσεις $p \times q$, για $p,q>1$ (χρήση διαφορετικών ορθογώνιων σχημάτων στην ίδια μνήμη). Ο στόχος που θέτουν αφορά στον προσδιορισμό ενός ορίου E_{max} τέτοιο, που να διασφαλίζει την άνευ συγκρούσεων πρόσβαση σε οποιοδήποτε ορθογώνιο με εμβαδό $E < E_{max}$. Η εργασία τους παρουσιάζει μια οργάνωση που μεγιστοποιεί το E_{max} όταν δοθεί ο αριθμός των τραπεζών B. Επιπρόσθετα, υπολογίζει ότι με την προτεινόμενη οργάνωση, για οποιονδήποτε αριθμό B, το εμβαδό των προσβάσιμων ορθογωνίων δεν γίνεται ποτέ μικρότερο του $B/\sqrt{5}$ (βέβαια, κατά περίπτωση, το E_{max} μπορεί να είναι πολύ μεγαλύτερο, πλησιάζοντας ακόμα και το ίδιο το B). Τέλος, αποδεικνύει με ασυμπτωτική ανάλυση ότι η προτεινόμενη λύση είναι βέλτιστη, αφού καμιά οργάνωση δεν μπορεί να οδηγήσει –ασυμπτωτικά– σε κατώτατο προσβάσιμο εμβαδό μεγαλύτερο από $B/\sqrt{5}$.

Περιγράφουμε εδώ συνοπτικά την εν λόγω τεχνική. Έστω (α, β) και $(-\beta, \alpha)$ κάθετα διανύσματα που χρησιμοποιούνται ως βάση δισδιάστατου πλέγματος. Θεωρούμε ότι οι α και β είναι ακέραιοι, πρώτοι μεταξύ τους. Τοποθετούμε νοητά το πλέγμα αυτό επάνω στην εικόνα και προσδιορίζουμε τα εικονοστοιχεία που συμπίπτουν με τους κόμβους του. Τα συγκεκριμένα εικονοστοιχεία αποθη-

κεύονται στην ίδια τράπεζα μνήμης, έστω b_0. Έπειτα, μετακινούμε το πλέγμα κατά μια θέση και αποθηκεύουμε τα νέα εικονοστοιχεία σε διαφορετική τράπεζα, έστω b_1. Επαναλαμβάνουμε τη διαδικασία μέχρι να αποθηκευτεί όλη η εικόνα. Οι περιορισμοί που έχουμε επιβάλει στα α και β διασφαλίζουν ότι η διαδικασία αυτή δεν οδηγεί σε κενά ή διπλές εγγραφές (τέλεια κάλυψη της επιφάνειας). Άρα ο αριθμός των τραπεζών που θα χρειαστούμε είναι ίσος με το εμβαδόν του βασικού παραθύρου του πλέγματος, $B = \alpha^2 + \beta^2$. Από όλο το δυνατό εύρος τιμών των α και β περιοριζόμαστε σε επιλογή οποιονδήποτε δυο διαδοχικών αριθμών της ακολουθίας Fibonacci[10], δηλαδή $\alpha = F_r$, $\beta = F_{r+1}$, και άρα $B = F_{2r+1}$. Ο περιορισμός αυτός οδηγεί τελικά στη βέλτιστη λύση, καθώς και στο όνομα της τεχνικής ως 'αντιστοίχιση με πλέγμα Fibonacci'.

Η παραπάνω περιγραφή χρησιμεύει περισσότερο στην απόδειξη των ιδιοτήτων της οργάνωσης και λιγότερο στην υλοποίησή της. Για το δεύτερο χρησιμοποιούμε τη συνάρτηση αντιστοίχισης

$$bank(x, y) = (x - c \cdot y) \bmod B \tag{2.5}$$

όπου c ένας σταθερός ακέραιος. Φυσικά, ο c δεν επιλέγεται αυθαίρετα, αλλά έχουμε $c = \beta \cdot k + \alpha \cdot l$, όπου k και l ακέραιοι για τους οποίους ισχύει ότι $\alpha \cdot k - \beta \cdot l = 1$ (πάντα υπάρχει τέτοιο ζευγάρι γιατί οι α και β είναι πρώτοι μεταξύ τους). Παρατηρούμε ότι η τεχνική αυτή είναι, ουσιαστικά, μια γραμμική λόξωση με χρήση μεγάλης περίσσειας τραπεζών (περίπου $E \cdot \sqrt{5}$). Έτσι, πέραν της δυνατότητας πρόσβασης σε μεγάλη ποικιλία ορθογώνιων σχημάτων, επιτυγχάνει και προσβάσιμο μήκος γραμμών/στηλών ίσο με B.

Τα παραπάνω αποτελέσματα δίνουν απάντηση σε ένα πολύ ενδιαφέρον θεωρητικό ερώτημα, δηλαδή, δείχνουν ποιες είναι οι δυνατότητες και το κόστος της βέλτιστης οργάνωσης μνήμης για μια πολύ γενικευμένη εφαρμογή γραφικών. Στην πράξη, όμως, η προτεινόμενη λύση επιβάλει περιορισμούς που ενδέχεται να αποτρέψουν τους μηχανικούς από τη χρήση της. Είναι ενδεικτικό το πόσο λίγες επιλογές έχουν για το πλήθος των τραπεζών, $B = \alpha^2 + \beta^2$, με α και β διαδοχικούς αριθμούς Fibonacci. Σε εφαρμογές που επιτρέπεται να ξοδέψουν έως τριψήφιο αριθμό τραπεζών, έχουν να επιλέξουν για το B μόνο κάτι από τα εξής: $\{5, 13, 34, 89, 233, 610\}$. Για κάθε τέτοιο B θα πετύχουν το αντίστοιχο E_{max}: $\{5, 11, 23, 53, 125, 307\}$. Άρα, αφενός, οι μηχανικοί είναι πιθανό να αναγκαστούν να χρησιμοποιήσουν πολύ μεγάλη περίσσεια τραπεζών συγκριτικά με το πραγματικό E που θα μπορεί να επεξεργαστεί η υπόλοιπη αρχιτεκτονική τους. Αφετέρου, θα πρέπει να υλοποιήσουν πράξεις modulo με κάποιο 'δύσκολο' αριθμό.

[10] αναδρομική ακολουθία: $F_0 = 0, \quad F_1 = 1, \quad F_r = F_{r-1} + F_{r-2}$ για $r \geq 2$.

Ο περιορισμός Fibonacci και οι πρακτικές επιπτώσεις του μπορούν να αποφευχθούν όταν αρθούν κάποιες από τις απαιτήσεις του προβλήματος. Για παράδειγμα, αν μας ενδιαφέρει απεριόριστη πρόσβαση μόνο κατά γραμμές, στήλες, και τετράγωνα, φτιάχνουμε πλέγμα με διανύσματα βάσης $(1, s)$ και $(-s, 1)$, όπου s οποιοσδήποτε αριθμός (όχι κατ' ανάγκη Fibonacci). Η απλοποιημένη αυτή οργάνωση χρησιμοποιεί $B = s^2 + 1$ τράπεζες κι επιτρέπει ανάκτηση γραμμών/στηλών μήκους s^2+1 και τετραγώνων επιφάνειας s^2. Έτσι, και με τη συγκεκριμένη μέθοδο επιβεβαιώνεται το γεγονός ότι απαιτείται μια επιπρόσθετη τράπεζα για απεριόριστη χρήση της ποικιλίας {γραμμή,στήλη,ορθογώνιο}. Επίσης, παρατηρούμε ότι η απλοποιημένη λύση είναι μια γραμμική λόξωση κατά $-(s^2 + s + 1)$, ή αλλιώς, κατά $B - s$. Τέλος, σημειώνουμε το πλεονέκτημα της ισοτροπικότητας της οργάνωσης που προκύπτει από τη χρήση όλων των παραπάνω διπλά περιοδικών αντιστοιχίσεων με χρήση πλεγμάτων.

Ψευδοτυχαία αντιστοίχιση και μεταβαλλόμενη αντιστοίχιση

Μια πιο γενική προσέγγιση στο πρόβλημα της οργάνωσης μνήμης είναι η χρήση ψευδοτυχαίων συναρτήσεων για την αντιστοίχιση των δεδομένων στις τράπεζες (ή/και στις διευθύνσεις εντός τους). Εδώ ο σχεδιασμός δεν λαμβάνει υπόψη του τόσο συγκεκριμένες εφαρμογές, π.χ., γραφικά ή πίνακες, όσο μια πιο αφηρημένη απαίτηση για πρόσβαση στη μνήμη κατά 'τακτικά' σχήματα, όπως π.χ., κατά 'διασκελισμούς' (*stride*)[11]. Οι ψευδοτυχαίες αντιστοιχίσεις βρίσκουν εφαρμογή περισσότερο σε αρχιτεκτονικές πολυ-επεξεργαστών γενικής χρήσης.

Παρόμοια με τις προαναφερθείσες συναρτήσεις αντιστοίχισης, οι ψευδοτυχαίες είναι συναρτήσεις κατακερματισμού (*hash functions*) που έχουν ως στόχο την ομοιόμορφη διασπορά των δεδομένων στη μνήμη προκειμένου να αποφευχθούν οι συγκρούσεις κατά την παράλληλη ανάκτησή τους. Επίσης, πρέπει να είναι εύκολα υπολογίσιμες ώστε να μην αυξάνεται το κόστος της εφαρμογής (είτε σε υλικό, είτε σε χρόνο). Συνεπώς, όπως είδαμε στις προηγούμενες παραγράφους, έτσι κι εδώ, οι σχεδιαστές καταφεύγουν συχνά σε χρήση αποκλειστικών διαζεύξεων ή/και υπολοίπων για την εισαγωγή ψευδοτυχαιότητας [58]. Η επιτυχία της διασποράς των δεδομένων ελέγχεται στην πράξη με προσομοιώσεις και μετρήσεις του χρόνου εκτέλεσης διαφορετικών αλγορίθμων (ή του μέσου αριθμού εντολών ανά κύκλο –IPC) [58] [59]. Σημειώνουμε εδώ ότι στους πολυ-επεξεργαστές η σύγκρουση στη μνήμη μεταφράζεται σε καθυστερημένη ανάκτηση δεδομένων. Έτσι, ο σχεδιασμός στοχεύει όχι ακριβώς

[11] δηλαδή πρόσβαση σε μια ακολουθία k στοιχείων που είναι αποθηκευμένα στις διευθύνσεις $\alpha,\ \alpha+\delta,\ \alpha+2\delta, \ldots,\ \alpha+(k-1)\delta,$ όπου α μια αρχική διεύθυνση και δ ο διασκελισμός.

στη διασφάλιση πρόσβασης με μηδενικές συγκρούσεις για όλα τα σχήματα, αλλά στην στατιστική ελάττωση των συγκρούσεων στη μνήμη για τη μείωση του χρόνου αναμονής (ειδικά στις ιεραρχίες μνήμης με κρυφά επίπεδα –*caches*– [59]).

Η προσφυγή στη τυχαιότητα είναι ίσως ενδεικτική της δυσκολίας κατασκευής μιας αντιστοίχισης που θα ικανοποιεί πολλές διαφορετικές απαιτήσεις, αλλά ταυτόχρονα θα έχει και μικρό κόστος υλοποίησης. Προς μια εναλλακτική κατεύθυνση, μπορούμε να παρακάμψουμε εντελώς το πρόβλημα χρησιμοποιώντας πολλές διαφορετικές, απλές, αντιστοιχίσεις στην ίδια μνήμη. Δηλαδή, κατασκευάζουμε επαναδιαρθρώσιμα κυκλώματα, ή επαναπρογραμματιζόμενες μονάδες, που αλλάζουν την συνάρτηση αντιστοίχισης ανάλογα με τις απαιτήσεις του αλγορίθμου [60]. Στην περίπτωση των εφαρμογών γραφικών, βέβαια, κάτι τέτοιο προϋποθέτει αργές μεταβολές των απαιτήσεων του αλγόριθμου (π.χ., μεγάλο χρονικό διάστημα στο οποίο εργαζόμαστε με τετράγωνα, ακολουθούμενο από μεγάλο διάστημα χρήσης γραμμών, κ.ο.κ.). Οι γρήγορες μεταβολές θα οδηγούσαν σε μεγάλη σπατάλη χρόνου, λόγω της ανάγκης για επανοργάνωση της μνήμης (πιθανώς, επανεγγραφής των στοιχείων). Αν μπορούμε να κάνουμε αυτή την υπόθεση κατά τη σχεδίαση κι αν η τεχνολογία που έχουμε στη διάθεσή μας επιτρέπει επαναπρογραμματισμό, τότε η συγκεκριμένη μέθοδος επιτρέπει πρόσβαση με εξαιρετικά μεγάλη ποικιλία σχημάτων και με ελεγχόμενο κόστος κατασκευής (αφού ενσωματώνουμε απλές συναρτήσεις, κατά βούληση). Φυσικά, η μέθοδος εισάγει επίβαρο κόστος λόγω της χρήσης επιπλέον μονάδων ελέγχου, της επαναδιάρθρωσης, και του χρόνου επανοργάνωσης.

2.3.2 Παρατηρήσεις και συμπεράσματα

Η βιβλιογραφική επισκόπηση που προηγήθηκε καταδεικνύει το ενδιαφέρον της επιστημονικής κοινότητας για την οργάνωση παράλληλων μνημών. Επί τέσσερις δεκαετίες προτείνονται διαφορετικές λύσεις για ένα πρόβλημα που παραλλάσσεται ανάλογα με την εφαρμογή. Οι –φαινομενικά– μικρές διαφοροποιήσεις στις απαιτήσεις των εφαρμογών οδηγούν τελικά σε πλήθος εξειδικευμένων τεχνικών σχεδίασης, αφήνοντας έως και σήμερα ανοικτά θέματα γενίκευσης ή/και πρακτικών βελτιστοποιήσεων.

Κοιτάζοντας εποπτικά τις οργανώσεις που προκύπτουν και τα επί μέρους θεωρητικά αποτελέσματα μπορούμε να κάνουμε γενικές, χρήσιμες, παρατηρήσεις. Αρχικά, είναι σαφές ότι όσο μεγαλύτερη περίσσεια τραπεζών χρησιμοποιούμε, τόσο μεγαλύτερη ποικιλία σχημάτων πρόσβασης δημιουργούμε. Όμως, η κατεύθυνση αυτή μας απομακρύνει από ένα βασικό ζητούμενο, αυτό της αποδοτικής σχεδίασης. Για παράδειγμα, όταν επιλέγουμε διπλάσιο αριθμό τραπεζών σε

σχέση τον αριθμό των εικονοστοιχείων που μπορούν να εισάγουν οι επεξεργαστικές μονάδες, οδηγούμε ένα σημαντικό κομμάτι της αρχιτεκτονικής (τις τράπεζες και όλες τις μονάδες υποστήριξής τους) σε υπο-εκμετάλλευση κατά 50%. Η ρυθμαπόδοση της μνήμης αξιοποιείται μόνο κατά 50% και, γενικότερα, παρουσιάζεται μια αρχιτεκτονική με μεγάλο φαινόμενο επίβαρο κόστος σε υλικό, σε κατανάλωση ενέργειας, κ.α.

Η καλύτερη επιλογή από άποψη εκμετάλλευσης του υλικού είναι να χρησιμοποιήσουμε $B = E$, δηλαδή, μνήμη που να προωθεί ακριβώς τόσα δεδομένα όσα μπορούν να επεξεργαστούν οι υπόλοιπες μονάδες της αρχιτεκτονικής. Όμως, είναι αποδεδειγμένο ότι στην περίπτωση αυτή δεν μπορούμε να πετύχουμε απεριόριστη πρόσβαση κατά {γραμμές,στήλες,ορθογώνια}[12], κάτι που αποτελεί βασική απαίτηση των εφαρμογών γραφικών. Έτσι, πρέπει να στραφούμε σε κάποιον πρώτο αριθμό τραπεζών, $B = \text{πρώτος} > E$, ή έστω στην ελάχιστη περίσσεια κατά ένα, $B = E + 1$, η οποία επίσης μπορεί να οδηγήσει σε πρώτο ή άλλο 'δύσκολο' αριθμό (αφού συνήθως χρησιμοποιούμε E που είναι δύναμη του 2). Η χρήση πρώτων αριθμών είναι ίσως η πιο γνωστή πρακτική ανά τις δεκαετίες και λύνει πολλά από τα προβλήματα πρόσβασης. Όμως, εισάγει δυσκολίες στην υλοποίηση των μονάδων που εκτελούν τις αριθμητικές πράξεις, καθώς και στα κυκλώματα που δρομολογούν τα δεδομένα από και προς τη μνήμη[13] [46] [47] [50]. Φυσικά, οι δυσκολίες αυτές μεταφράζονται σε αύξηση του κόστος του υλικού.

Συνυπολογίζοντας τις παραπάνω παρατηρήσεις, ίσως έλεγε κανείς ότι είναι αναπόφευκτο στη παρούσα εργασία να παρουσιάσουμε λύση με μεγάλη υποεκμετάλλευση και ακριβά κυκλώματα. Στο σημείο αυτό θέτουμε για πρώτη φορά ένα ερώτημα με το οποίο θα ασχοληθούμε, μεταξύ άλλων, στα επόμενα κεφάλαια: ποιο είναι το καλύτερο που μπορούμε να πετύχουμε αν περιοριστούμε αυστηρά στην περίπτωση $B = E$; Σίγουρα, θα αναγκαστούμε να ξοδέψουμε κάποιο χρόνο για τη διόρθωση των συγκρούσεων που –αποδεδειγμένα– θα δημιουργηθούν. Όμως, αν οργανώσουμε κατάλληλα τις τράπεζες ίσως το χρονικό αυτό κόστος να ελαχιστοποιείται τόσο ώστε να καθίσταται πιο ανεκτό από τα παραπάνω επίβαρα κόστη.

[12] σημειώνουμε πως η λογική της απόδειξης που δίνεται στην εργασία [47] μπορεί να χρησιμοποιηθεί και για διαφορετικά σύνολα σχημάτων, π.χ., για τον αποκλεισμό της απεριόριστης πρόσβασης κατά {γραμμές,στήλες,αραιά-ορθογώνια}, όπου το αραιό-ορθογώνιο αντιστοιχεί στη λεγόμενη 'σκακιέρα' (chessboard, ή αραιό-2 σύμφωνα με την ορολογία του κεφαλαίου 4)

[13] τα δεδομένα θεωρούμε ότι μπαίνουν και βγαίνουν σε μια προκαθορισμένη σειρά από τη μνήμη, ανεξάρτητα από τις όποιες αναδιατάξεις υφίστανται μέσα σε αυτήν προκειμένου να αντιστοιχηθούν στις τράπεζές τους. Η σειρά αυτή διατηρείται μέσω κυκλωμάτων δρομολόγησης.

Σχετικά με τη χρήση ενός ακόμα πιο μικρού αριθμού τραπεζών, $B < E$, παρατηρούμε στη βιβλιογραφία ότι είναι μια δυνατότητα που αποφεύγεται να προταθεί ως λύση στο γενικό πρόβλημα που μελετάμε. Αυτό συμβαίνει γιατί οδηγεί σε οργανώσεις με σημαντικά μειονεκτήματα. Συγκεκριμένα, όταν συνδυαστεί με τον περιορισμό πρόσβασης ενός κύκλου, μας αναγκάζει να καταφύγουμε σε περίσσεια πληροφορίας ή σε υπο-εκμετάλλευση της μνήμης. Παραθέτουμε εδώ συνοπτικά ένα επιχείρημα που τεκμηριώνει την παρατήρηση αυτή. Έστω $D = E - B$ το 'έλλειμμα' τραπεζών. Προκειμένου να μπορέσει η μνήμη να εξυπηρετήσει τον απαιτούμενο ρυθμό εικονοστοιχείων ανά κύκλο ($= E$), τουλάχιστον D από τις τράπεζες πρέπει να έχουν πλάτος τουλάχιστον διπλάσιο από τις υπόλοιπες. Έστω ζευγάρι εικονοστοιχείων $\{a_1, a_2\}$ αποθηκευμένο στην τυχαία διεύθυνση A_x κάποιας από τις τράπεζες μεγάλου πλάτους. Ακόμα και στην καλύτερη δυνατή οργάνωση, δεν είναι δύσκολο (χάρη στη δυνατότητα απεριόριστης πρόσβασης, ή στην αλλαγή σχήματος) να εντοπίσουμε πιθανό σύνολο-αίτηση στο οποίο να περιέχεται το a_1 κι όχι το a_2. Αυτό σημαίνει πως, κατά τη σχεδίαση, αναγκαζόμαστε να επιλέξουμε ανάμεσα στις δυο εναλλακτικές: i) το a_1 να ανακτάται από την A_x, ή ii) το a_1 να ανακτάται από μια διαφορετική A_y. Στην πρώτη εναλλακτική θα εμφανιστεί υπο-εκμετάλλευση της συγκεκριμένης τράπεζας (το a_2 οδηγείται στο κενό) και άρα γενικότερη υποεκμετάλλευση της μνήμης. Η υπο-εκμετάλλευση αυτή μεγαλώνει με το πλάτος της τράπεζας (μεγαλύτερο μέρος μπορεί να οδηγηθεί στο κενό), καθώς και με τις δυνατότητες πρόσβασης, οι οποίες αυξάνουν τα πιθανά σύνολα-αιτήσεις και άρα τους τρόπους 'αδρανοποίησης' μέρους του πλάτους λέξης σε έναν συνδυασμό από τράπεζες. Η δυνητική αυτή αδρανοποίηση οδηγεί σε αύξηση του πλάτους των υπολοίπων τραπεζών προκειμένου να μπορέσει να εξυπηρετηθεί ο ρυθμός πρόσβασης στη μνήμη σε οποιαδήποτε περίπτωση αιτήσεων. Τονίζουμε επίσης ότι η μερική χρήση του πλάτους των λέξεων εισάγει σημαντικό επίβαρο κόστος υλοποίησης, αφού βασίζεται σε επιπρόσθετες μονάδες δρομολόγησης δεδομένων όπως, π.χ., πολυπλέκτες. Στη δεύτερη εναλλακτική, ξεκάθαρα, έχουμε διπλοεγγραφή του a_1 στη μνήμη κι επομένως περίσσεια πληροφορίας ανάλογη του όγκου της μνήμης (αφού το a_1 επιλέχθηκε τυχαία και άρα αντιπροσωπεύει οτιδήποτε είναι αποθηκευμένο στις D τράπεζες).

Πέρα από τον αριθμό των τραπεζών, μια δεύτερη σειρά παρατηρήσεων που αξίζει να σημειωθεί αφορά στη συνάρτηση αντιστοίχισης. Η συνάρτηση αυτή αποτελεί το πιο σημαντικό γνώρισμα της εκάστοτε λύσης, αφού προσδίδει τις επιθυμητές ιδιότητες στην οργάνωση. Στη βιβλιογραφία οι συναρτήσεις χωρίζονται σε κλάσεις που σχετίζονται μεταξύ τους με συγκεκριμένο τρόπο [46]. Παρουσιάσαμε αναλυτικά τη κλάση των *γραμμικών* λύσεων, η οποία είναι υπο-

κλάση των *ισοτροπικών*. Μια κλάση ξένη με την ισοτροπική είναι η κλάση των *δυαδικών* συναρτήσεων, την οποία περιγράψαμε μέσω των πράξεων XOR. Η ισοτροπική και η δυαδική είναι υπο-κλάσεις των *περιοδικών* συναρτήσεων, οι οποίες με τη σειρά τους είναι υπο-κλάση των *πολυπεριοδικών* συναρτήσεων που αναφέραμε συνοπτικά. Η κλάση των *γενικών* αντιστοιχίσεων περιέχει όλες τις προαναφερθείσες, συν ένα ιδιαίτερο κομμάτι που θεωρείται χαμηλού ενδιαφέροντος για τις πρακτικές εφαρμογές, αφού συνήθως εμφανίζει δυσκολίες κατά την υλοποίηση. Η αναζήτηση/κατασκευή της κατάλληλης συνάρτησης είναι μια διαδικασία που πολλές φορές βασίζεται στη διαίσθηση και τον πειραματισμό (με βοήθεια υπολογιστή, ή χωρίς). Σημειώνουμε χαρακτηριστικά ότι η αναζήτηση μη-γραμμικών λύσεων, π.χ., στη κλάση των δυαδικών συναρτήσεων, είναι από μόνο του ένα NP-complete πρόβλημα για $B > 2$ [46]. Από την άλλη, βέβαια, η αναζήτηση γραμμικής λύσης είναι πιο εύκολη, αφού ο αριθμός τους (κι επομένως οι δοκιμές) περιορίζεται στις $O(B)$.

Συγκρίνοντας τις δημοσιευμένες λύσεις διαπιστώνουμε ότι μπορούμε να πετύχουμε καλύτερες δυνατότητες πρόσβασης εισάγοντας μεγαλύτερη πολυπλοκότητα στην αντιστοίχιση. Είναι χαρακτηριστικά τα πλεονεκτήματα της μη-γραμμικής λόξωσης έναντι της γραμμικής, καθώς και το πως αυτά αυξάνονται όταν υιοθετούμε πιο περίπλοκους κανόνες για τον προσδιορισμό της $bank(x, y)$. Όμως, η πολυπλοκότητα της αντιστοίχισης μεταφράζεται σε μεγάλο κόστος υλοποίησης (υλικού ή/και χρόνου, ανάλογα με την αρχιτεκτονική). Συνεπώς, η μη-γραμμικότητα δεν αποτελεί μια οριστική απάντηση στο γενικότερο πρόβλημα που μελετάμε, αλλά ένα σύνολο λύσεων με διαβάθμιση κόστους στο οποίο πρέπει να στρέφεται κανείς με φειδώ, λαμβάνοντας προσεκτικά υπόψη τις απαιτήσεις της εκάστοτε εφαρμογής.

Το προηγούμενο συμπέρασμα ισχύει όχι μόνο για τις μη-γραμμικές λύσεις λόξωσης, αλλά ακόμα και για εκείνες που χρησιμοποιούν XOR συναρτήσεις και Λατινικά Τετράγωνα. Ο λόγος δεν είναι η πολυπλοκότητα της αντιστοίχισης, αφού οι συγκεκριμένες λύσεις οδηγούν συνήθως σε πολύ απλά κυκλώματα. Το μειονέκτημα εντοπίζεται σε ένα γενικότερο φαινόμενο που χαρακτηρίζει όλες τις μη-γραμμικές οργανώσεις: την ανισοτροπικότητα. Όπως έχουμε ήδη αναφέρει, η ανισοτροπικότητα επηρεάζει αρνητικά το κόστος υλοποίησης της μνήμης στις εφαρμογές γραφικών [46]. Μια συνοπτική εξήγηση είναι η ακόλουθη. Τα εικονοστοιχεία είναι διατεταγμένα πάνω στην εικόνα και, ως εκ τούτου, πρέπει να προωθούνται στις επεξεργαστικές μονάδες ακολουθώντας κάποια σειρά (κάποια προκαθορισμένη φόρμα, όπως π.χ., αριστερά προς δεξιά στο ψηφιδόπλεγμα –raster scan). Η σειρά αυτή δεν διατηρείται στη μνήμη εξαιτίας της διασποράς που υφίστανται τα δεδομένα κατά την αντιστοίχισή τους στις τράπεζες.

Στην πράξη, αυτό σημαίνει ότι κάθε συστοιχία δεδομένων που εμφανίζεται στην είσοδο ή στην έξοδο των τραπεζών πρέπει να αναδιαταχτεί. Εδώ μας ενδιαφέρει ο συνολικός αριθμός όλων των δυνατών αναδιατάξεων, ο οποίος προσδιορίζεται από τρεις παράγοντες: την ποικιλία των σχημάτων πρόσβασης, τις δυνατές θέσεις τους πάνω στην εικόνα, και τη μορφή της διασποράς που υφίστανται τα δεδομένα στις τράπεζες. Οι δυο πρώτοι παράγοντες εξαρτώνται από τις προδιαγραφές της εφαρμογής και είναι διαισθητικά εύκολο να δούμε πως επηρεάζουν τους δυνατούς συνδυασμούς κοιτάζοντας οποιονδήποτε πίνακα/χάρτη αντιστοίχησης: περισσότερα σχήματα και μάλιστα περισσότερες θέσεις (π.χ., απεριόριστη πρόσβαση) σημαίνουν περισσότερες αναδιατάξεις. Ο τρίτος παράγοντας μπορεί να διατηρήσει τον αριθμό των διατάξεων σε χαμηλά επίπεδα, ή αντίθετα, να τον οδηγήσει σε εκθετική αύξηση. Πιο συγκεκριμένα, όταν έχουμε ισοτροπική οργάνωση διαπιστώνουμε πως έχουμε μια κυκλική διαδοχή των τραπεζών πάνω στην εικόνα, είτε διαβάζουμε κατά γραμμές, είτε κατά στήλες, είτε κατά ορθογώνια (κάθε σχήμα με τη δική του φόρμα). Με άλλα λόγια, αρκούμαστε σε κυκλικές μεταθέσεις των εισόδων και των εξόδων των τραπεζών προκειμένου να πετύχουμε την επιθυμητή σειρά των εικονοστοιχείων. Αντίθετα, όσο πιο ανισοτροπική είναι η οργάνωση, τόσο περισσότερες αλλαγές βλέπουμε στη σειρά με την οποία εμφανίζονται οι διαδοχικές τράπεζες πάνω στον πίνακα/χάρτη της αντιστοίχισης. Οι ασυνέχειες αυτές αυξάνονται ακόμα περισσότερο όταν επιτρέψουμε στο σχήμα μας να 'κινηθεί' σε όλες τις θέσεις της εικόνας (π.χ. εφαρμογές γραφικών). Κάθε ασυνέχεια πολλαπλασιάζει τον αριθμό των δυνατών διατάξεων των δεδομένων, ο οποίος καθορίζει τελικά το κόστος του κυκλώματος δρομολόγησης μπροστά και πίσω από τις τράπεζες. Αυτό συμβαίνει γιατί τα συγκεκριμένα κυκλώματα πρέπει να είναι σε θέση να εκτελέσουν οποιαδήποτε μετάθεση απαιτείται για την τοποθέτηση των δεδομένων στη σωστή φόρμα. Συνοψίζοντας, είναι λογικό να αποφεύγουμε την εκτεταμένη χρήση της μη-γραμμικότητας (ακόμα κι όταν αυτή υλοποιείται με απλές αντιστοιχίσεις) λόγω της ανισοτροπικότητας που αυτή εισάγει, η οποία οδηγεί έμμεσα στην αύξηση του κόστους των μονάδων υποστήριξης της μνήμης.

Κλείνοντας την ενότητα, παρουσιάζουμε εποπτικά στον Πίνακα 2.1 τις δημοσιευμένες λύσεις που αναφέρθηκαν και τα κύρια χαρακτηριστικά τους. Διαχωρίζουμε τις λύσεις ανάλογα με το αν προσφέρουν απεριόριστη πρόσβαση σε όλη την έκταση της εικόνας, για όλα τα σχήματα. Ουσιαστικά, η ιδιότητα αυτή διαχωρίζει τις λύσεις σε καταλληλότερες για εφαρμογές γραφικών και σε καταλληλότερες για, π.χ., επεξεργασία πινάκων. Επιπλέον, τις διαχωρίζουμε ανάλογα με το αν είναι γραμμικές, δηλαδή αν είναι ισοτροπικές. Παρατηρούμε πως οι γραμμικές και οι μη-γραμμικές λύσεις αναπτύχθηκαν παράλληλα μέσα

Πίνακας 2.1: Κατηγοριοποίηση των δημοσιευμένων λύσεων

		Αριθμός Τραπεζών	*Προσβάσιμα Σχήματα*	*Εργασία*	
Πρόσβαση άνευ συγκρούσεων	Απεριόριστη	E+1, πρώτος $> E$	r, c, b, d, d'	[45], 1971	Γραμμική
		E+1, $2E$, $E^{3/2}$	r, c, b	[47], 1978	
		E+1, $\sqrt{5}E$	r, c, b^*	[57], 1986	
		πρώτος $> E$	r, c, b, s, d, d'	[48], 2004	
		$2E$	r, c, b, s, d, d'	[49], 2007	Μη-γραμμική
	Περιορισμένη	E	r, c, b, s	[53], 1985	
		E	r, c, b, s	[56], 1993	
		E	r, c, b, d	[50], 1996	
		E	r, c, b, b'	[46], 2005	

E: εμβαδόν (εικονοστοιχεία ανά σχήμα), r: γραμμές, c: στήλες, d: διαγώνιοι, d': ανεστραμμένες διαγώνιοι, b: ορθογώνια, b': ανεστραμμένα ορθογώνια, b^*: ορθογώνια μεταβλητού μεγέθους, s: αραιά ορθογώνια

στις δεκαετίες και πως η πρόσβαση κατά {γραμμές,στήλες,ορθογώνια} είναι κοινός τόπος όλων των εργασιών. Εκτός από αυτή τη βασική ποικιλία σχημάτων (με την οποία ασχολείται –όχι τυχαία– και η παρούσα εργασία) μερικές λύσεις επιτρέπουν επιπρόσθετη πρόσβαση κατά αραιά ορθογώνια, ένα σχήμα που χρησιμεύει στην υποδειγματοληψία της εικόνας και που θα το εξετάσουμε ενδελεχώς στα κεφάλαια 4 και 5. Ο Πίνακας 2.1 τονίζει την αύξηση στον αριθμό των τραπεζών που συνοδεύει την απαίτηση για απεριόριστη πρόσβαση, καθώς και το πως αυτή η αύξηση προσεγγίζεται ποικιλοτρόπως, στοιχείο ενδεικτικό της δυσκολίας να βελτιστοποιηθούν ταυτόχρονα όλες οι παράμετροι του προβλήματος που αντιμετωπίζουμε: *αποθήκευση πλήθους διαφορετικών συνόλων δεδομένων που δεν είναι ξένα μεταξύ τους, χρησιμοποιώντας λίγες τράπεζες και απλά κυκλώματα, χωρίς να εισάγουμε περίσσεια πληροφορίας.*

Κεφάλαιο 3

Προγραμματιζόμενη Μονάδα Εκτίμησης Κίνησης

Με στόχο την αποδέσμευση ενός υλισμικού συμπιεστή εικονοροών από τις επιδόσεις ενός μοναδικού –υποβέλτιστου– αλγορίθμου ταιριάσματος περιοχών, το παρόν κεφάλαιο περιγράφει τη σχεδίαση και την ανάπτυξη μιας προγραμματιζόμενης μονάδας εκτίμησης κίνησης (ΕΚ). Η προτεινόμενη αρχιτεκτονική επιτρέπει τη εκτέλεση πολλών διαφορετικών αλγορίθμων, η επιλογή των οποίων μπορεί να γίνει ανάλογα με τις υπολογιστικές απαιτήσεις της εκάστοτε εφαρμογής ή/και τα χαρακτηριστικά της υπό επεξεργασία εικονοροής. Επιπροσθέτως, με στόχο την επιτάχυνση των διαδικασιών, χρησιμοποιούμε παραλληλισμό σε επίπεδο δεδομένων για την επεξεργασία των εικονοστοιχείων και προτείνουμε μια τεχνική υποθετικής εκτέλεσης εντολών (speculative execution), σχεδιασμένη ειδικά για αλγορίθμους ταιριάσματος περιοχών.

Υπό το γενικότερο πρίσμα της αρχιτεκτονικής υπολογιστών, σκοπός του κεφαλαίου είναι η δημιουργία ενός εφαρμογοείδιου επεξεργαστή (*application specific processor*) με επιδόσεις πραγματικού χρόνου. Για το σκοπό αυτό σχεδιάζουμε ένα εξειδικευμένο σύνολο σύνθετων εντολών, το οποίο υποστηρίζει τους πλέον διαδεδομένους αλγορίθμους ταιριάσματος περιοχών ξοδεύοντας μόνο έναν μικρό αριθμό κύκλων μετάκλησης-αποκωδικοποίησης (*fetch-decode*) με εκτελέσιμα αρχεία πολύ περιορισμένου μεγέθους. Το προτεινόμενο σύνολο απαρτίζεται από έναν πυρήνα εντολών που χρησιμοποιούνται σε όλους τους αλγορίθμους ταιριάσματος περιοχών, καθώς και από ένα σύνολο προαιρετικών, σύνθετων, εντολών για την υλοποίηση εξεζητημένων τεχνικών. Ο πυρήνας των κοινών εντολών αρκεί για την εκτέλεση απλών αλγορίθμων όπως, π.χ., τους *Three Step Search*, *One Step Search*, *Diamond Search*, *2D Logarith-*

mic Search, *Hexagonal Search*, κ.α. [3] [4] [5] [6]. Οι σύνθετες εντολές υποστηρίζουν προηγμένους αλγορίθμους όπως, π.χ., τους *MVFAST*, *FAME*, *PMVFAST*, κ.α. [7] [8] [9] [10].

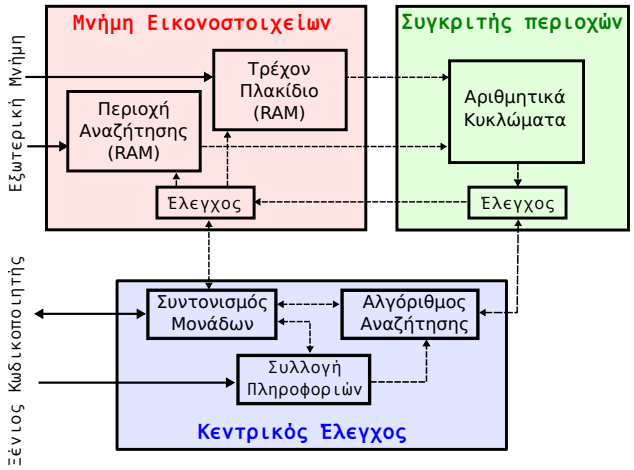
Για την τήρηση των περιορισμών που θέτουν οι εφαρμογές πραγματικού χρόνου βασιζόμαστε, πρωτίστως, στην επιτάχυνση των πράξεων που απαιτούνται για την εξέταση μιας υποψήφιας περιοχής εικονοστοιχείων. Η λειτουργία αυτή λαμβάνει χώρα πολλές φορές κατά την εκτέλεση των αλγορίθμων κι αντιστοιχεί στο μεγαλύτερο μέρος του συνολικού υπολογισμού τους. Ακολουθούμε στρατηγική διαχωρισμού της εκάστοτε υποψήφιας περιοχής σε ομάδες πολλαπλών εικονοστοιχειών, κάθε μια εκ των οποίων εξετάζεται σε έναν μοναδικό κύκλο ρολογιού. Η διαδικασία της εξέτασης ολοκληρώνεται σε διαδοχικά βήματα αξιοποιώντας την υποκείμενη σωληνωτή αρχιτεκτονική [61] που συνδέει την τοπική μνήμη εικονοστοιχείων με το δομοστοιχείο της επεξεργασίας τους. Ο εν λόγω παραλληλισμός ανάγεται στη σχεδίαση μνήμης πολλαπλών τραπεζών με ικανό ρυθμό διεκπεραιωτικότητας, καθώς και στη σωλήνωση του επεξεργαστικού δομοστοιχείου με πολλαπλά αριθμητικά κυκλώματα.

Πέρα από τον παραλληλισμό σε επίπεδο δεδομένων κι από την ελάττωση των κύκλων μετάκλησης-αποκωδικοποίησης, η προτεινόμενη αρχιτεκτονική εισάγει και μια τρίτη τεχνική για την επιτάχυνση της διαδικασίας ταιριάσματος περιοχών: μια στοχευμένη υποθετική εκτέλεση εντολών. Σκοπός της συγκεκριμένης τεχνικής είναι η μεγιστοποίηση της εκμετάλλευσης (100% utilization) της σωλήνωσης απαλείφοντας τα κενά μεταξύ των διαδοχικών διαδικασιών εξέτασης υποψηφίων περιοχών. Σχεδιάζουμε την αρχιτεκτονική έτσι, ώστε να είναι δυνατός ο παραλληλισμός της διαδικασίας εξέτασης των περιοχών με τη διαδικασία μετάκλησης-αποκωδικοποίησης των εντολών. Ο παραλληλισμός αυτός επιτρέπει στην υποθετική εκτέλεση να υπολογίζει τη θέση του επόμενου υποψηφίου μπλοκ πριν ακόμα ολοκληρωθεί η εξέταση του τρέχοντος, καθιστώντας άμεση την εκκίνηση της ακόλουθης διαδικασίας εξέτασης.

Στην ενότητα 3.1 περιγράφεται αναλυτικότερα η αρχιτεκτονική σχεδίαση, ενώ στην ενότητα 3.2 παρατίθενται τα αποτελέσματα της πρωτοτυποποίησης της μονάδας ΕΚ σε συσκευές προγραμματιζόμενης λογικής FPGA.

3.1 Προτεινόμενη αρχιτεκτονική

Η σχεδίαση της μονάδας ΕΚ, τόσο σε υψηλό όσο και σε χαμηλό επίπεδο, βασίζεται στην εκτεταμένη μελέτη των αλγορίθμων ταιριάσματος περιοχών και στον προσδιορισμό των κοινών τους γνωρισμάτων/απαιτήσεων. Σε χαμηλό



Εικόνα 3.1: Γενικό σχέδιο της προτεινόμενης μονάδας εκτίμησης κίνησης.

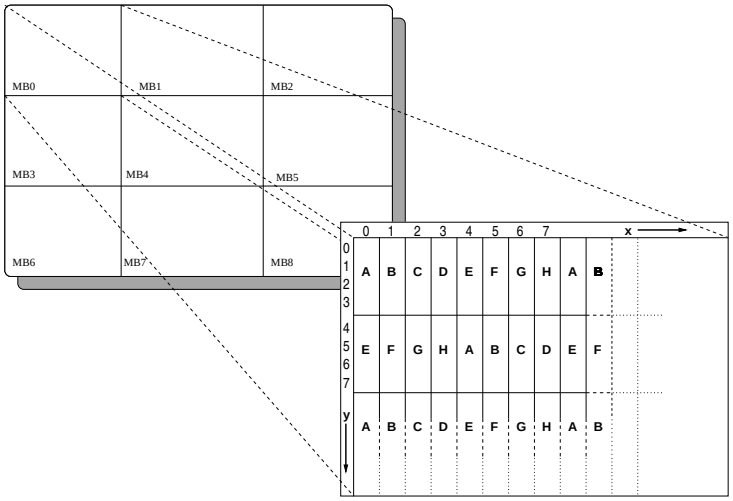
επίπεδο, τα κοινά χαρακτηριστικά καθορίζουν το σύνολο εντολών και τον τρόπο υλοποίησής τους (instruction set architecture). Σε υψηλό επίπεδο, η μελέτη καταδεικνύει τις κεντρικές λειτουργίες των αλγορίθμων, σύμφωνα με τις οποίες σχεδιάζουμε τις κεντρικές οντότητες της αρχιτεκτονικής (μια οντότητα ανά λειτουργία). Πιο συγκεκριμένα, αναγνωρίζουμε τρεις βασικές αλγοριθμικές λειτουργίες: (α) μαζική ανάκτηση εικονοστοιχείων από τη μνήμη, (β) υπολογισμό ενός μέτρου ομοιότητας για κάθε υποψήφια περιοχή εικονοστοιχείων, και (γ) έλεγχο των υπολογισμένων μέτρων και θέσεων για τον προσδιορισμό νέων υποψηφίων περιοχών. Αντιστοιχούμε τις τρεις ανωτέρω λειτουργίες σε τρία ξεχωριστά δομοστοιχεία της μονάδος ΕΚ: στην τοπική μνήμη, στον συγκριτή περιοχών, και στο στοιχείο κεντρικού ελέγχου (Εικόνα 3.1). Η μονάδα ΕΚ επεξεργάζεται την εικόνα σε επίπεδο πλακιδίων (ακολουθώντας την πάγια τακτική των συμπιεστών εικονοροής) λαμβάνοντας ως είσοδο εικονοστοιχεία και προωθώντας στην έξοδο το διάνυσμα κίνησης του εκάστοτε πλακιδίου μαζί με το μέτρο ομοιότητας του ταιριάσματός του.

Η τοπική μνήμη λειτουργεί ως ένας προσωρινός αποθηκευτικός χώρος άμεσης προσπέλασης για την τρέχουσα περιοχή αναζήτησης και το τρέχον πλακίδιο (data cache). Σκοπός της είναι η αποφόρτιση του ξένιου κωδικοποιητή εικονορροών από τις πολλαπλές επαναλαμβανόμενες αιτήσεις της μονάδας ΕΚ για ταυτόσημα σύνολα εικονοστοιχείων, έτσι όπως αυτά προκύπτουν από τις διαδοχικές εξετάσεις των υποψηφίων μπλοκ. Επιπροσθέτως, χρησιμοποιείται ως ένα μέσο ομαδοποίησης των δεδομένων, το οποίο επιτρέπει την παράλληλη

ανάγνωσή τους με συνέπεια να οδηγεί σε υψηλό ρυθμό τροφοδότησης των επεξεργαστικών κυκλωμάτων. Παρέχει δυνατότητα ανάγνωσης 16 εικονοστοιχείων ανά κύκλο ρολογιού, τόσο από την περιοχή αναζήτησης όσο κι από το τρέχον πλακίδιο (συνολικά 32 δεδομένα). Επίσης, πέρα από την αποθήκευση εικονοστοιχείων ακεραίων θέσεων (integer pixels), το δομοστοιχείο ενσωματώνει και φίλτρα ψηφιακής παρεμβολής για την παραγωγή υποεικονοστοιχείων με θέσεις 1/2 κι 1/4 (subpixels). Η παραγωγή γίνεται κατά τη μετάδοση (on-the-fly) παρέχοντας δυνατότητα ανάγνωσης 4 υποεικονοστοιχείων ανά κύκλο ρολογιού. Τα περιεχόμενα της τοπικής μνήμης αφορούν μόνο στις τιμές φωτεινότητας των εικονοστοιχείων (η εκτίμηση κίνησης δεν εξετάζει χρώματα) κι η ανανέωσή τους λαμβάνει χώρα με την εκκίνηση της επεξεργασίας κάθε νέου πλακιδίου μέσω διεπαφών συμβατών με το ξένιο κωδικοποιητή.

Ο συγκριτής περιοχών υλοποιεί την εξέταση ενός υποψηφίου μπλοκ λειτουργώντας με κάποια σχετική αυτονομία: ξεκινάει λαμβάνοντας την υποψήφια θέση από τη μονάδα κεντρικού ελέγχου και συνεχίζει εκτελώντας διαδοχικές κλήσεις στην τοπική μνήμη για την ανάκτηση του τρέχοντος πλακιδίου και του υποψηφίου μπλοκ. Καθώς ανακτά σταδιακά τα απαραίτητα εικονοστοιχεία υπολογίζει το 'άθροισμα των απολύτων διαφορών' τους (ΑΑΔ), το οποίο και προωθεί τελικώς στην κεντρική μονάδα ελέγχου ως μέτρο ομοιότητας της υποψήφιας περιοχής. Επιπροσθέτως, κατά τη συσσώρευση της μετρικής ΑΑΔ, το δομοστοιχείο χρησιμοποιεί τα μερικά αθροίσματα για να προβλέψει το τελικό αποτέλεσμα της σύγκρισης και να βελτιώσει την πορεία της υποθετικής εκτέλεσης εντολών που πραγματοποιείται στη κεντρική μονάδα ελέγχου ταυτόχρονα με την εξέταση του τρέχοντος υποψηφίου.

Ο κεντρικός έλεγχος είναι υπεύθυνος για το συγχρονισμό των επί μέρους λειτουργιών (αποθήκευση εικονοστοιχείων, συλλογή πληροφοριών, προώθηση αποτελεσμάτων) και, κυρίως, για την εκτέλεση του αλγορίθμου ταιριάσματος περιοχών. Υλοποιεί το σημαντικότερο κομμάτι της μικροαρχιτεκτονικής του προτεινόμενου εφαρμογοείδιου επεξεργαστή περιέχοντας μνήμη εντολών, αποκωδικοποιητή εντολών, καταχωρητές, αριθμητικά/λογικά στοιχεία, και εξειδικευμένα κυκλώματα για εκτέλεση σύνθετων εντολών. Κατά τη λειτουργία του οδηγεί το συγκριτή περιοχών υπολογίζοντας τις θέσεις των υποψηφίων μπλοκ (εκμεταλλευόμενος και την υποθετική εκτέλεση εντολών) και αποφασίζει το τελικό διάνυσμα κίνησης.

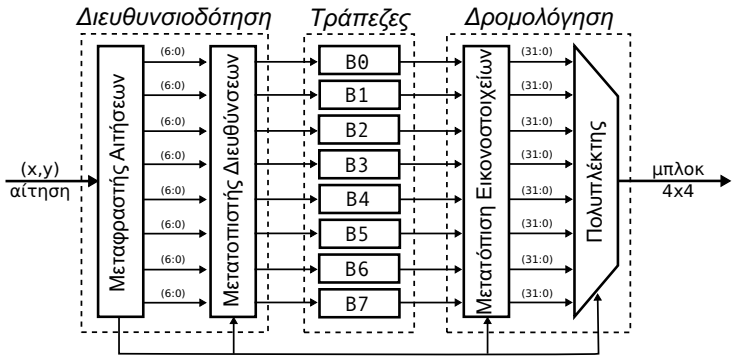


Εικόνα 3.2: Αντιστοίχιση της περιοχής αναζήτησης στις τράπεζες της μνήμης.

3.1.1 Τοπική μνήμη εικονοστοιχείων

Για την αποθήκευση της περιοχής αναζήτησης χρησιμοποιούμε παράλληλη μνήμη 8 τραπεζών, έκαστη πλάτους 32 διφύων και βάθους 72 διευθύνσεων. Κάθε διεύθυνση της τράπεζας περιέχει 4 τιμές φωτεινότητας (τιμή ενός byte) και ολόκληρη η μνήμη περιέχει 2304 τιμές, καθώς ορίζουμε περιοχή αναζήτησης 48×48 εικονοστοιχείων στο προηγούμενο από το υπό επεξεργασία καρέ. Η περιοχή περιλαμβάνει 9 πλακίδια, δηλαδή το συνεντοπισμένο του τρέχοντος (collocated) συν τα 8 γειτνιάζοντά του, επιτρέποντας εκτίμηση κίνησης με διανυσμάτα μήκους ± 16 στις δυο διευθύνσεις του επιπέδου. Η ανάγνωση εικονοστοιχείων από την παράλληλη μνήμη γίνεται με σχήμα πρόσβασης το τετράγωνο 4×4 (16 τιμές φωτεινότητας ανά κύκλο ρολογιού).

Η αίτηση ενός τετραγώνου εικονοστοιχείων πραγματοποιείται χρησιμοποιώντας τη θέση (x, y) του μπλοκ επάνω στην περιοχή αναζήτησης (θέση του πάνω-αριστερά στοιχείου του μπλοκ). Πιο συγκεκριμένα, το δομοστοιχείο σύγκρισης περιοχών αποστέλει ένα ζεύγος ακεραίων (x, y), το οποίο μεταφράζεται εσωτερικά σε 16 διευθύνσεις μνήμης κι αριθμούς τραπεζών προκειμένου να εντοπιστούν τα ζητούμενα εικονοστοιχεία μέσα στη μνήμη. Ένα δεδομένο που βρίσκεται στη θέση (x_δ, y_δ) της περιοχής αναζήτησης αποθηκεύεται μέσα στην τράπεζα με διακριτικό αριθμό $B(x_\delta, y_\delta) = (\lfloor \frac{y_\delta}{4} \rfloor \times 4 + x_\delta) \bmod 8$, στη διεύθυνση $A(x_\delta, y_\delta) = \lfloor \frac{y_\delta}{4} \rfloor \times 6 + \lfloor \frac{x_\delta}{8} \rfloor + 26$, και στην υπολέξη $\Lambda(x_\delta, y_\delta) = y_\delta \bmod 4$. Σημειώνουμε ότι οι ακέραιοι x και y λαμβάνουν τιμές στο διάστημα $[-16, 31]$



Εικόνα 3.3: Αρχιτεκτονική της μνήμης που περιέχει την περιοχή αναζήτησης.

υποθέτωντας ότι το $(0,0)$ συμβολίζει τη θέση του επάνω-αριστερά στοιχείου του συνεντοπισμένου μπλοκ, το οποίο τοποθετούμε στο κέντρο της περιοχής αναζήτησης (άρα κι ο προσθετικός παράγοντας 26 της $A(x,y)$). Η παραπάνω οργάνωση χωρίζει την περιοχή σε οριζόντιες λωρίδες πάχους 4 με κάθε λωρίδα να υπόκειται σε γραμμική λόξωση κατά 4 τράπεζες (Εικόνα 3.2), ακολουθώντας ευθεία διευθυνσιοδότηση από αριστερά προς τα δεξιά κι από πάνω προς τα κάτω. Το πάχος και η λόξωση των λωρίδων διασφαλίζουν την άνευ συγκρούσεων απεριόριστη πρόσβαση ενός κύκλου στο 4×4 τετράγωνο.

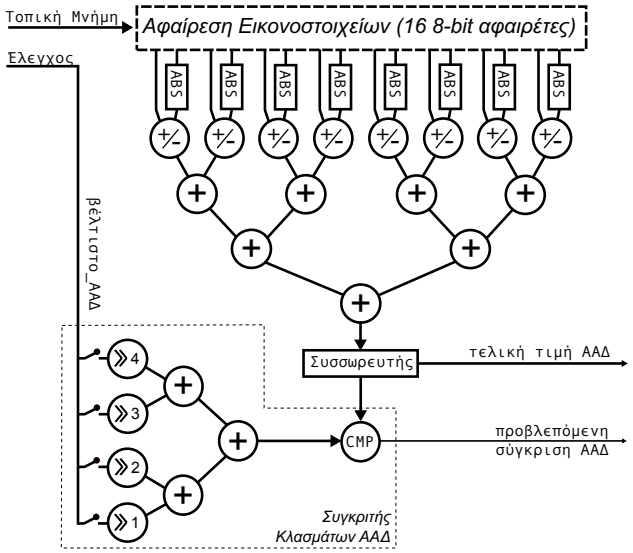
Η σχεδίαση της μνήμης επιτρέπει διαφορετική (x,y) αίτηση ανά κύκλο προσφέροντας δυνατότητα συνεχούς ροής 16 δεδομένων μέσα από την περιοχή αναζήτησης. Για το σκοπό αυτό χρησιμοποιούμε σωληνωτή αρχιτεκτονική βάθους 5 σταδίων, την οποία παραθέτουμε στην Εικόνα 3.3. Πιο αναλυτικά, με την είσοδο της αίτησης (x,y) στη σωλήνωση, εκτελείται μια μετάφραση κατά $B(x,y)$ και 16 μεταφράσεις κατά $A(x_\delta, y_\delta)$. Το $B(x,y)$ καθορίζει τη στροφή του κυλιδρικού μετατοπιστή που βρίσκεται μπροστά από το μεταφραστή, ώστε να στοιχηθούν σωστά οι 16 διευθύνσεις $A(x_\delta, y_\delta)$ και να δρομολογηθούν στις αντίστοιχες τράπεζες. Το γεγονός ότι για την εν λόγω δρομολόγηση αρκεί μόνο ένας κυλινδρικός μετατοπιστής και μια μετάφραση $B(x,y)$ οφείλεται στην ισοτροπικότητα που προκύπτει από την υποκείμενη γραμμική λόξωση (εν. 2.3.2). Στο τρίτο στάδιο της σωλήνωσης έχουμε τις 8 τράπεζες. Στο τέταρτο στάδιο έχουμε έναν κυλινδρικό μετατοπιστή για τη στοίχιση των εικονοστοιχείων που βγαίνουν μετατιθέμενα από τις τράπεζες (οδηγούνται στον συγκριτή περιοχών σε σειρά raser scan), ο οποίος ελέγχεται επίσης από το $B(x,y)$. Στο τελευταίο στάδιο τοποθετούμε έναν πολυπλέκτη για την επιλογή 16 τιμών (από τις 32 που προωθούν οι 8 τράπεζες), ο οποίος ελέγχεται από την τιμή $\Lambda(x,y)$.

Για την αποθήκευση του τρέχοντος πλακιδίου χρησιμοποιούμε μια τράπεζα μνήμης βάθους 16 διευθύνσεων και πλάτους 16×8 διφύων. Εφόσον οι αιτήσεις για δεδομένα του πλακιδίου αφορούν μόνο σε συγκεκριμένα σύνολα εικονοστοιχείων, ξένα μεταξύ τους, είναι δυνατόν να ομαδοποιήσουμε τα δεδομένα σε ξεχωριστές 16-αδες (δηλαδή, στα 16 βασικά 4×4 υπο-πλακίδια) και να τις αποθηκεύουσμε στις διευθύνσεις μιας μόνο τράπεζας[1]. Έτσι, σε κάθε κύκλο, το δομοστοιχείο σύγκρισης περιοχών ανακτά, μαζί με το 4×4 μπλοκ από την περιοχή αναζήτησης, κι ένα 4×4 μπλοκ από το τρέχον πλακίδιο. Σημειώνουμε ότι με την εκκίνηση της επεξεργασίας ενός νέου πλακιδίου ανανεώνονται μόνο τα στοιχεία της μνήμης που λείπουν: με δεδομένη τη συνήθη σειρά επεξεργασίας των πλακιδίων στον ξένιο κωδικοποιητή (raster scan), τα στοιχεία που λείπουν είναι το τρέχον πλακίδιο, καθώς και τα MB2, MB5, MB8 (Εικόνα 3.2), αφού η περιοχή αναζήτησης μετακινείται κάθε φορά μόνο κατά μια στήλη πλακιδίων δεξιά. Σε κάθε επανεκκίνηση, τα νέα {MB2,MB5,MB8} αποθηκεύονται στις θέσεις των παλαιών {MB0,MB3,MB6} και οι εξωτερικές αιτήσεις (x, y) διορθώνονται εσωτερικά σε $(x+16) \bmod 48$, ή $(x-16) \bmod 48$, ή $(x) \bmod 48$, ανάλογα με το που βρίσκεται η στήλη που περιέχει το συνεντοπισμένο πλακίδιο.

Η αρχιτεκτονική του δομοστοιχείου τοπικής μνήμης ενσωματώνει και ψηφιακά φίλτρα, ώστε να επιτυγχάνεται η παρεμβολή των εικονοστοιχείων κατά τη μετάδοσή τους και να καθίσταται η διαδικασία σχεδόν 'διάφανη' στο δομοστοιχείο σύγκρισης περιοχών. Πιο συγκεκριμένα, με αιτήσεις της μορφής $(x.z, y)$ ή $(x, y.z)$, όπου $z \in \{\frac{1}{4}, \frac{2}{4}, \frac{3}{4}\}$, προωθούνται 4 υποεικονοστοιχεία με ακρίβεια μετατόπισης $1/4$ κατά την οριζόντια ή την κατακόρυφη διεύθυνση. Για το σκοπό αυτό η σωλήνωση της Εικόνας 3.3 ενισχύεται με ένα 6^o στάδιο, από το οποίο περνάει το 4×4 μπλοκ όταν $z \neq 0$ (μέσω πολυπλέκτη). Το 6^o στάδιο περιέχει τέσσερα δικυβικά φίλτρα με συντελεστές $[-1, 5, 5, -1]$, τα οποία υλοποιούνται πλήρως παράλληλα με τέσσερα δέντρα ολισθητών και αθροιστών[2]. Κάθε δικυβικό φίλτρο λαμβάνει ως είσοδο μια στήλη ή μια γραμμή του 4×4 μπλοκ των ακέραιων εικονοστοιχείων, ανάλογα με το αν η ζητούμενη μετατόπιση είναι κατακόρυφη ή οριζόντια, αντίστοιχα. Τα δικυβικά φίλτρα χρησιμοποιούνται για παραγωγή υποεικονοστοιχείων $^1/_2$. Για την παραγωγή υποεικονοστοιχείων

[1] η μνήμη του τρέχοντος πλακιδίου διαφέρει από αυτή της περιοχή αναζήτησης στην απαίτηση της δεύτερης για απεριόριστη πρόσβαση (περιέχει επικαλυπτόμενα σύνολα-αιτήσεις).

[2] το φίλτρο $[-1, 5, 5, -1]$ αποτελεί άμεση απλοποίηση του φίλτρου $[1, -5, 20, 20, -5, 1]$ που προτείνει το πρότυπο H.264/AVC: θεωρούμε ότι το πρώτο εικονοστοιχείο είναι ίδιο με το δεύτερο, και το πέμπτο ίδιο με το έκτο. Η χρήση του απλοποιημένου φίλτρου μειώνει τους απομαστευτές (taps) από 6 σε 4, δηλαδή μειώνει την πολυπλοκότητα της κατασκευής του καθώς και τους απαιτούμενους κύκλους για τη συλλογή των εικονοστοιχείων-εισόδων του.



Εικόνα 3.4: Αρχιτεκτονική του συγκριτή περιοχών.

1/4 τοποθετούμε ένα φίλτρο διγραμμικής παρεμβολής μετά από κάθε δικυβικό φίλτρο, το οποίο λαμβάνει ως είσοδο το 1/2 και το αντίστοιχο ακέραιο εικονοστοιχείο (με πολυπλέκτες). Η παραγωγή ενός υποεικονοστοιχείου απαιτεί, εξ΄ ορισμού, ανάκτηση τεσσάρων ακεραίων στοιχείων και συνεπώς, όταν η σωλήνωση βρίσκεται σε λειτουργία παρεμβολής η διεκπεραιωτικότητά της μειώνεται κατά 1/4, δηλαδή ο συγκριτής περιοχών μπορεί να ανακτά 4 υποεικονοστοιχεία ανά κύκλο ρολογιού.

3.1.2 Συγκριτής περιοχών

Ο κεντρικός έλεγχος της μονάδος ΕΚ αιτείται την εξέταση μιας υποψήφιας περιοχής 16×16 εικονοστοιχείων με την αποστολή της θέσης της, (x, y), στο συγκριτή περιοχών. Ο συγκριτής ενσωματώνει μια μηχανή πεπερασμένων καταστάσεων, η οποία σχηματίζει διαδοχικά 16 αιτήσεις προς την τοπική μνήμη σαρώνοντας την υποψήφια περιοχή και το τρέχον πλακίδιο. Οι αιτήσεις εξυπηρετούνται από τη σωληνωτή αρχιτεκτονική της μνήμης με αποτέλεσμα, 5 κύκλους μετά την εκκίνηση της διαδικασίας, να φτάνουν διαδοχικά στο συγκριτή δυο ξεχωριστές 16-αδες εικονοστοιχείων ανά κύκλο. Οι δυο 16-αδες αφαιρούνται εσωτερικά, στοιχείο προς στοιχείο, δημιουργώντας 16 διαφορές (Εικόνα 3.4). Οι διαφορές ομαδοποιούνται σε 8 ζευγάρια. Το πρώτο μέλος

του ζευγαριού περνά από ένα κύκλωμα υπολογισμού απόλυτης τιμής (ABS: αντιστροφέας, αθροιστής και 2-προς-1 πολυπλέκτης 9 διφύων). Το δεύτερο μέλος προστίθεται ή αφαιρείται από την απόλυτη τιμή του πρώτου, ανάλογα με το προσήμό του. Από τη διαδικασία αυτή προκύπτει το άθροισμα απολύτων διαφορών δυο 2-αδων εικονοστοιχείων, χρησιμοποιώντας μόνο ένα κύκλωμα ABS. Το ΑΑΔ των 16 2-αδων της εισόδου του δομοστοιχείου σχηματίζεται στην απόληξη ενός δέντρου αθροιστών που συγκεντρώνει τα 8 επί μέρους αποτελέσματα (Εικόνα 3.4). Η απόληξη του δέντρου οδηγείται σε έναν συσσωρευτή προκειμένου να υπολογιστεί σταδιακά το ολικό ΑΑΔ της υποψήφιας περιοχής. Η συσσώρευση διαρκεί 16 κύκλους, ή λιγότερο, καθώς εφαρμόζουμε τεχνική πρόωρου τερματισμού: σε κάθε κύκλο συγκρίνουμε το μερικώς συσσωρευμένο ΑΑΔ της υποψήφιας περιοχής με την τιμή ΑΑΔ του επικρατέστερου ταιριάσματος που έχει προηγουμένως ανευρεθεί για το τρέχον πλακίδιο (η τιμή αυτή κρατείται στον κεντρικό έλεγχο κι αποστέλλεται μαζί με τη θέση της υποψήφιας περιοχής). Αν το μερικό ΑΑΔ ξεπεράσει το βέλτιστο ΑΑΔ τότε η εξέταση σταματά (καθίσταται μάταιη).

Όσο διαρκεί ο παραπάνω υπολογισμός του ΑΑΔ, ο κεντρικός έλεγχος αναμένει το τελικό αποτέλεσμα για να συνεχίσει με τον αλγόριθμο ταιριάσματος. Η μελέτη των αλγορίθμων μας έδειξε ότι η συνέχιση της πορεία τους εξαρτάται, όχι τόσο από την ακριβή τιμή του ζητούμενου ΑΑΔ, όσο από τη σύγκρισή του με το βέλτιστο ΑΑΔ που έχουν ανακαλύψει (παίρνουν αποφάσεις συγκρίνοντας τα ΑΑΔ των διαφόρων υποψηφίων μεταξύ τους). Προκειμένου να αποφευχθεί η αεργία του κεντρικού ελέγχου κατά τη συσσώρευση του ΑΑΔ, το δομοστοιχείο μας κάνει πρόβλεψη του αποτελέσματος της επερχόμενης σύγκρισης (0 ή 1) και ειδοποιεί, ώστε ο αλγόριθμος να συνεχίσει με μια υποθετική εκτέλεση των επόμενων εντολών. Η πρόβλεψη γίνεται σε κάθε κύκλο της συσσώρευσης χρησιμοποιώντας το μερικό ΑΑΔ και το αντίστοιχο μέρος του βέλτιστου ΑΑΔ (δηλαδή, $\frac{1}{16}$, $\frac{2}{16}$, $\frac{3}{16}$, κοκ), όπως φαίνεται στην Εικόνα 3.4. Αν προκύψει κάποια πρόβλεψη που διαφέρει απο τις προηγούμενες, τότε το δομοστοιχείο ενημερώνει τον κεντρικό έλεγχο ώστε να ακυρώσει τις υποθετικώς εκτελεσμένες εντολές. Σημειώνεται ότι μια αποτυχημένη πρόβλεψη δεν καθυστερεί την εξέταση της υποψήφιας περιοχής κι ότι το μοναδικό της κόστος αφορά στην επιτάχυνση που θα προσέφερε η υποθετική εκτέλεση.

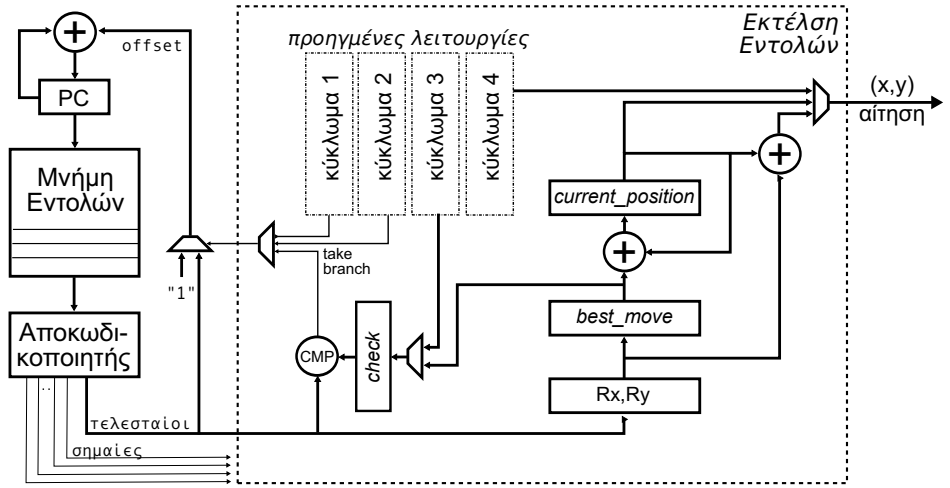
Τελειώνοντας, αναφέρουμε ότι τα παραπάνω ισχύουν κι όταν ο κεντρικός έλεγχος ζητήσει εξέταση περιοχής με οριζόντια ή κατακόρυφη κλασματική μετατόπιση. Μοναδική διαφορά είναι ότι η εσωτερική μηχανή καταστάσεων στέλνει στη μνήμη 64 αιτήσεις της μορφής $(x.z, y)$ ή $(x, y.z)$, όπου $z \in \{\frac{1}{4}, \frac{2}{4}, \frac{3}{4}\}$, κι ότι τα 3/4 του εσωτερικού δέντρου αθροιστών αδρανοποιούνται.

3.1.3 Έλεγχος μονάδας και σύνολο εντολών

Το δομοστοιχείο κεντρικού ελέγχου εκτελεί τρεις λειτουργίες (Εικόνα 3.1). Πρώτον, συλλέγει πληροφορίες σχετικά με τα ανύσματα κίνησης και τα ΑΑΔ των γειτονικών του τρέχοντος πλακιδίων για τις ανάγκες των προηγμένων αλγορίθμων ταιριάσματος περιοχών (αν πρόκειται να εκτελεστεί κάποιος τέτοιος). Δεύτερον, υλοποιεί τον αλγόριθμο ταιριάσματος. Τρίτον, εκτελεί τις χειραψίες έναρξης/τερματισμού με τον ξένιο κωδικοποιητή κι αποστέλλει σειριακά σήματα εκκίνησης στις εξής διεργασίες της ΕΚ: συλλογή πληροφοριών (μηχανή καταστάσεων), ανανέωση δεδομένων της τοπικής μνήμης (μηχανή καταστάσεων), κι εκτέλεση εντολών αλγορίθμου. Η διεργασία συλλογής πληροφοριών συνίσταται σε μια απλή σειρά αιτήσεων στη μνήμη του ξένιου κωδικοποιητή μέσω ειδικών διεπαφών, καθώς και στην αποθήκευση των ανακτηθέντων αριθμών σε εσωτερικούς καταχωρητές για τη μετέπειτα δυνατότητα άμεσης και μαζικής επεξεργασίας τους. Η ανανέωση των εικονοστοιχείων συντελείται στην τοπική μνήμη όπως έχουμε ήδη περιγράψει. Η σημαντικότερη από τις λειτουργίες του δομοστοιχείου είναι η αποκωδικοποίηση κι εκτέλεση του αλγορίθμου.

Η αρχιτεκτονική του προτεινόμενου εφαρμογοείδιου επεξεργαστή ακολουθεί τη λογική με την οποία αναπτύσσεται η πλειοψηφία των ευρετικών αλγορίθμων ταιριάσματος περιοχών. Πιο συγκεκριμένα, παρατηρούμε ότι αυτοί οι αλγόριθμοι αναπτύσσονται αναδρομικά: σε κάθε βήμα βρίσκουν ένα ικανοποιητικό ταίριασμα έτσι, ώστε στο επόμενο βήμα να προσπαθήσουν να το βελτιώσουν εξετάζοντας κάποιες κοντινές σε αυτό θέσεις. Μια τέτοια εξέταση ανάγεται στον υπολογισμό της μετρικής ομοιότητας της υποψήφιας θέσης (εδώ το ΑΑΔ), ενώ οι αποφάσεις βασίζονται στις συγκρίσεις των υπολογισμένων μετρικών τόσο μεταξύ τους, όσο και με τη βέλτιστη μετρική που ανακαλύφθηκε στο προηγούμενο βήμα. Οι αλγόριθμοι διαφοροποιούνται, κυρίως, ως προς τις σχετικές θέσεις των κοντινών υποψηφίων (το μοτίβο ψαξίματος), καθώς και ως προς τους ελέγχους/αποφάσεις που έπονται των συγκρίσεων των μετρικών.

Η Εικόνα 3.5 συνοψίζει την αρχιτεκτονική του επεξεργαστή δείχνοντας τη μνήμη εντολών, τον αποκωδικοποιητή, και τους βασικούς καταχωρητές, οι οποίοι αντικατοπτρίζουν την προαναφερθείσα λογική. Οι εν λόγω καταχωρητές περιέχουν δείκτες (x, y) επάνω στην περιοχή αναζήτησης για την προσωρινή απομνημόνευση σημαντικών περιοχών: ο *current_position* δείχνει στο βέλτιστο ταίριασμα που έχει ανακαλυφθεί έως το τρέχον βήμα της αναδρομής (δηλαδή, στο αποτέλεσμα του προηγούμενου βήματος), ο *best_move* δείχνει στο βέλτιστο ταίριασμα που έχει ανακαλύφθεί έως τον τρέχοντα κύκλο του τρέχοντος αναδρομικού βήματος (σχετική θέση εντός του μοτίβου ψαξίματος), ενώ ο R_xRy



Εικόνα 3.5: Αρχιτεκτονική του εφαρμογοείδιου επεξεργαστή.

περιέχει τη σχετική θέση της υποψήφιας περιοχής που πρόκειται να εξεταστεί άμεσα: η θέση (R_x, Ry) προστίθεται στη θέση *current_position*, ώστε να σχηματιστεί ένα απόλυτο διάνυσμα που θα οδηγηθεί ως αίτηση στο δομοστοιχείο σύγκρισης περιοχών. Τα δεδομένα του R_xRy ανανεώνονται με τελεστέους από τη μνήμη εντολών, η οποία εμμέσως περιέχει το μοτίβο ψαξίματος. Τα δεδομένα του *best_move* αντικαθίστανται με αυτά του R_xRy όταν από την εξέταση της υποψήφιας περιοχής (R_x, Ry) προκύψει μικρότερο ΑΑΔ από το έως τότε βέλτιστο (τότε, η νέα τιμή ΑΑΔ αντικαθιστά την παλαιά στον καταχωρητή *best_sad*). Στο τέλος του αλγοριθμικού βήματος (της εξέτασης του μοτίβου), τα περιεχόμενα του *best_move* προστίθενται σε αυτά του *current_position* ώστε να σχηματιστεί η απόλυτη θέση του ανακαλυφθέντος ταιριάσματος, να ανανεωθεί ο *current_position*, και να προχωρήσουμε στο επόμενο αναδρομικό βήμα.

Η Εικόνα 3.5 δείχνει επιπλέον μια σειρά από ανεξάρτητα κυκλώματα, κάθε ένα από τα οποία μπορεί να παραλειφθεί κατά την τοποθέτηση της μονάδας ΕΚ στο υλισμικό (για λόγους οικονομίας). Κάθε τέτοιο κύκλωμα εκτελεί μια συγκεκριμένη σύνθετη εντολή για λογαριασμό κάποιου προηγμένου αλγορίθμου, η οποία σχετίζεται με τους επιπρόσθετους ελέγχους του και λειτουργεί επικουρικά στην λογική που περιγράψαμε στην προηγούμενη παράγραφο. Η είσοδός τους αφορά, συνήθως, σε πληροφορίες που σχετίζονται με τα γειτονικά πλακίδια (ανύσματα κι ΑΑΔ), ενώ η έξοδός τους επηρεάζει τα περιεχόμενα του καταμετρητή εντολών. Γενικότερα, κατά τη συνήθη πρακτική των αλγορίθμων, ο καταμετρητής εντολών επηρεάζεται στο τέλος κάθε αλγοριθμικού βήματος

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INIT				S_x						S_y					
SAD				R_x						R_y					
CMP				C_x						C_y					
CMPSP				*extop*		—									
JUMPC				*extop*		*offset*									
MBACT				L_1						L_2					
FIXTHR				—											
PREDMV				—											
TSTOP				f	*threshold*										
SUBPIXEL				q	—										
UPDATE				—											
END				—											

Πίνακας 3.1: Σύνολο εντολών του εφαρμογοείδιου επεξεργαστή

μετά από σύγκριση της πλέον ταιριαστής θέσης που βρέθηκε πάνω στο μοτίβο ψαξίματος (τιμή *best_move*, που αντιγράφεται στον καταχωρητή *check* για έλεγχο), με κάποιον τελεστέο που ορίζει ο προγραμματιστής (αναλόγως, προστίθεται στον καταμετρητή κάποιος τελεστέος-μετατόπιση). Η δυνατότητα αυτή επιτρέπει αλλαγές στην σειρά εκτέλεσης του προγράμματος, δηλαδή, επιτρέπει τον έλεγχο της πορείας του αλγορίθμου μέσα στην περιοχή αναζήτησης[3].

Η μνήμη εντολών περιέχει ένα ή περισσότερα προγράμματα, σχετικά μικρά σε μέγεθος (50-150 εντολές), τα οποία περιγράφουν τους αλγορίθμους ταιριάσματος. Ο ξένιος κωδικοποιητής μπορεί να επιλέξει διαφορετικό πρόγραμμα για κάθε τρέχον πλακίδιο αρχικοποιώντας τον καταμετρητή εντολών. Ένα πρόγραμμα απαρτίζεται από εφαρμογοείδιες εντολές που σχεδιάσαμε μετά από τη μελέτη των κοινών χαρακτηριστικών και των απαιτήσεων ενός πλήθους ευρετικών αλγορίθμων. Το προτεινόμενο σύνολο εντολών παρατίθεται στον Πίνακα 3.1. Οι εντολές έχουν πλάτος 16 διφύων, με τα πρώτα 4 να απαρτίζουν τον κωδικό της εντολής (opcode) και τα επόμενα 12 να περιέχουν τους τελεστέους της (operands). Οι πλέον απαραίτητες από τις 12 εντολές είναι οι: *SAD* (εξέταση υποψηφίου), *CMP* και *JUMPC* (έλεγχος πορείας αλγορίθμου), και *UPDATE* (ορισμός τέλους ενός αλγοριθμικού βήματος – ανανέωση καταχωρητών). Με

[3]επιπροσθέτως, επιτρέπει την αποφυγή επανεξέτασης υποψηφίων (σπατάλη χρόνου) προσαρμόζοντας το αναδρομικό βήμα στις θέσεις που ακολουθούν τη φορά κίνησης του μοτίβου. Εναλλακτικά, τοποθετούμε εσωτερική μνήμη για τις εξετασθείσες θέσεις (bitmap).

τις 4 αυτές εντολές μπορούμε να ελέγξουμε τον κορμό της αρχιτεκτονικής της Εικόνας 3.5 και να υλοποιήσουμε οποιονδήποτε απλό αλγόριθμο αναζήτησης, γνωστό ή νέο. Επιπροσθέτως, προτείνονται εντολές για πιο εξειδικευμένες λειτουργίες και προηγμένους αλγόριθμους. Εξηγούμε αναλυτικότερα:

- *INIT*: αρχικοποιεί την αναζήτηση στη θέση *current_position*= (S_x, S_y).
- *SAD*: ορίζει τη σχετική θέση μιας υποψήφιας περιοχής, (R_x, R_y), και ειδοποιεί το συγκριτή περιοχών για την εξέτασή της (υπολογίζει το ΑΑΔ της). Επιπροσθέτως, υποστηρίζει περιορισμένο αριθμό τελεστέων-κωδικολέξεων που σηματοδοτούν την εσωτερική αντικατάσταση των τιμών R_x και R_y με κάποιο από τα διανύσματα κίνησης των γειτονικών πλακιδίων (αριστερά, πάνω, αριστερά-πάνω, δεξιά-πάνω, και μέσος όρος) και την τοποθέτησή του στον R_xR_y.
- *CMP*: Συγκρίνει το διάνυσμα του καταχωρητή *check* με το διάνυσμα (C_x, C_y) και σε περίπτωση ισότητας δίνει τιμή σε μια εσωτερική σημαία ελέγχου (flag), η οποία καθορίζει τη συμπεριφορά της εντολής *JUMPC* που ακολουθεί.
- *CMPSP*: εκτελεί ειδικούς, σύνθετους, ελέγχους που χρησιμεύουν σε προηγμένους αλγορίθμους[4]. Ο τελεστέος *extop* συγκεκριμενοποιεί το είδος του ελέγχου, καθώς και τη σημαία ελέγχου που θα πάρει τιμή από αυτόν.
- *JUMPC*: ανάλογα με το αποτέλεσμα του ελέγχου που προηγήθηκε (από CMP ή CMPSP) προστίθεται ο τελεστέος *offset* στον καταμετρητή εντολών, ώστε να ολοκληρωθεί ένα συνθηκοταγές άλμα (conditional jump). Ο τελεστέος *extop* καθορίζει τη σημαία ελέγχου που εξετάζεται για την εκτέλεση του άλματος (0: ασυνθηκοταγές άλμα, 1: σημαία της CMP, 2-3: σημαίες της CMPSP).
- *MBACT*: σύνθετη εντολή του προηγμένου αλγορίθμου MVFAST, υπολογίζει την κινητικότητα του πλακιδίου. Οι τελεστέοι L_1 και L_2 ορίζουν τιμές (ειδικά κατώφλια) για τον υπολογισμό, ενώ το αποτέλεσμα γράφεται στον καταχωρητή *check* προκειμένου να ελεγχθεί από τον προγραμματιστή (μέσω CMP).
- *FIXTHR*: εκτελεί μια σειρά από βήματα αρχικοποίησης για τον προηγμένο αλγόριθμο PMVFAST και αποθηκεύει τα αποτελέσματα σε αποκλειστικούς καταχωρητές για χρήση στη συνέχεια του προγράμματος (π.χ., μέσω CMPSP).
- *PRDMV*: σύνθετη εντολή, υπολογίζει το μέσο όρο των γειτονικών διανυσμάτων κίνησης για τον προσδιορισμό νέας υποψήφιας περιοχής (π.χ., PMVFAST).
- *TSTOP*: τερματισμός του αλγορίθμου αν το ΑΑΔ του βέλτιστου ταιριάσματος που έχει επιτευχθεί έως τότε είναι μικρότερο του τελεστέου *threshold*. Το f ορίζει το αποτέλεσμα της *FIXTHR* ως κατώφλι για τον ελέγχο τερματισμού.
- *SUBPIXEL*: με διαδικασία παρεμβολής εξετάζονται 4 θέσεις γύρω από τη *current_position* σε απόσταση 1/2 (οριζόντιας και κατακόρυφης μετατόπισης).

[4] π.χ., έλεγχος αν τα γειτονικά πλακίδια εμφανίζουν ίσα διανύσματα κίνησης (PMVFAST).

Όταν $q = 1$ εκτελείται κι ένα επιπρόσθετο αναδρομικό βήμα για να εξεταστούν δυο θέσεις σε απόσταση 1/4 δίπλα από το αποτέλεσμα του πρώτου βήματος.
• *UPDATE*: εντολή που σηματοδοτεί το τέλος ενός αλγοριθμικού βήματος. Η τιμή του *best_move* συσσωρεύεται σε αυτή του *current_position* και, ταυτόχρονα, προωθείται στον *check* για τυχόν έλεγχο από τη CMP (για καθορισμό πορείας).
• *END*: τερματισμός λειτουργίας αλγορίθμου και προώθηση των δεδομένων των καταχωρητών *current_position* και *best_sad* στον ξένιο κωδικοποιητή ως διάνυσμα κίνησης και μετρική ομοιότητας, αντίστοιχα, για το τρέχον πλακίδιο.

Η διάρκεια εκτέλεσης των παραπάνω εντολών[5] είναι 1 κύκλος ρολογιού (συν έναν για ανάκτηση από τη μνήμη), εκτός από την εντολή *SAD*, η οποία εκκινεί το συγκριτή περιοχών και είναι δυνατόν να διαρκέσει έως 22 κύκλους (λόγω του χρόνου σάρωσης της υποψήφιας περιοχής, καθώς και του βάθους της σωλήνωσης ανάμεσα στην τοπική μνήμη και στο συσσωρευτή ΑΑΔ). Προκειμένου να αποφύγουμε μια τόσο μεγάλη καθυστέρηση στην αποκωδικοποίηση του προγράμματος, προτείνουμε τη χρήση μιας ειδικής τεχνικής για υποθετική εκτέλεση εντολών. Πιό συγκεκριμένα, από τα πρώτα κιόλας εικονοστοιχεία που λαμβάνει, ο συγκριτής περιοχών κάνει πρόβλεψη για το αν η τρέχουσα υποψήφια περιοχή θα αποδειχτεί καλύτερη από το βέλτιστο ταίριασμα που έχει έως τώρα ανακαλυφθεί (σελ. 54). Ακολουθώντας αυτή την πρόβλεψη, ο κεντρικός έλεγχος ανανεώνει αμέσως τα δεδομένα του καταχωρητή *best_move* και προχωρά στην εκτέλεση των επόμενων εντολών παράλληλα με τη συσσώρευση που συντελείται στο συγκριτή περιοχών. Η τεχνική αυτή βασίζεται στη γενική παρατήρηση ότι είναι περισσότερες οι κινήσεις των αλγορίθμων που εξαρτώνται από τα περιεχόμενα του *best_move* και πολύ λιγότερες εκείνες που εξαρτώνται από τη τιμή *best_sad* (που ανανεώνεται μόνο στο τέλος της εξέτασης). Η υποθετική εκτέλεση συνεχίζεται μέχρι την αποκωδικοποίηση μιας νέας εντολής *SAD*[6], η οποία οφείλει να αναμένει τον τερματισμό του κατειλημμένου συγκριτή περιοχών. Το κέρδος της τεχνικής είναι η προετοιμασία του επόμενου υποψηφίου έτσι, ώστε τα πρώτα εικονοστοιχεία του να εισέλθουν στο συγκριτή περιοχών αμέσως μετά τα τελευταία του προηγούμενου υποψηφίου, οδηγώντας την εκμετάλλευση της σωληνωτής επεξεργασίας δεδομένων (datapath) σχεδόν στο 100%. Αν κάποια από τις προβλέψεις του συγκριτή περιοχών αποδειχθεί λαν-

[5] η εντολή *SUBPIXEL* δε συγκρίνεται με τις υπόλοιπες, καθώς αποτελεί στην πραγματικότητα ολόκληρη υπορουτίνα: ενεργοποιεί μια αποκλειστική μηχανή καταστάσεων και βάζει τη μονάδα ΕΚ σε λειτουργία υποεικονοστοιχείων. Καλείται –μια μοναδική φορά– μετά το τέλος του αλγορίθμου ακέραιας αναζήτησης και καταναλώνει έως $64 \times 4 + 64 \times 2$ κύκλους.

[6] ή κάποιας εντολής που εξερτάται από την ακριβή τιμή του βέλτιστου ΑΑΔ, π.χ. *TSTOP*

θασμένη (στο τέλος ή κατά τη διάρκεια της συσσώρευσης), τότε εκτελείται μια επαναφορά (rollback) του επεξεργαστή στην κατάσταση που βρέθηκε αμέσως μετά τη τελευταία εκκίνηση του συγκριτή περιοχών (τελευταία *SAD*). Για το σκοπό αυτό αποθηκεύουμε προσωρινά σε καταχωρητές τις τιμές των $R_x R_y$, *current_position*, *best_move* και *program_counter*. Σημειώνεται ότι η εν λόγω διαχείριση (exception handling) δεν επηρεάζει τη λειτουργία του συγκριτή (δεν τον επανεκκινεί) και δεν καθυστερεί την πραγματική εκτέλεση εντολών.

3.2 Ανάπτυξη σε υλισμικό

Η προτεινόμενη μονάδα ΕΚ αναπτύχθηκε σε υλισμικό με στόχο την αξιολόγηση του κόστους και των επιδόσεων λειτουργίας της, καθώς και τη σύγκρισή της με άλλες λύσεις της βιβλιογραφίας. Για την πρωτοτυποποίησή της χρησιμοποιήσαμε διατάξεις προγραμματιζόμενων ολοκληρωμένων κυκλωμάτων (FPGA), και συγκεκριμένα, τη συσκευή Xilinx Virtex-II Pro 40 [62].

Το κόστος της τοποθέτησης της μονάδας στην εν λόγω συσκευή FPGA ανέρχεται σε 2,127 slices (δηλαδή στο 11% του τσιπ), ή 46K ισοδύναμες πύλες, και σε 11 μπλοκ μνήμης (BRAM). Η μέγιστη συχνότητα λειτουργίας της είναι τα 50 MHz. Πιο αναλυτικά, ο Πίνακας 3.2 μοιράζει το κόστος στα 3 δομοστοιχεία[7].

Δομοστοιχείο	*Slices*	*Μνήμη (bits)*
τοπική μνήμη	1,123	20,480
συγκριτής	312	–
έλεγχος	692	2,336
σύνολο	2,127	22,816

Πίνακας 3.2: Κόστος υλοποίησης ανά δομοστοιχείο (σε πόρους του FPGA).

Διαμόρφωση	*Slices*	
πλήρης ΕΚ	2,127	–
απλοί αλγόριθ.	1,724	-19 %
αα +{MVFAST}	1,896	-11 %
αα +{PMVFAST}	2,057	-4 %
ΕΚ – SUBPIXEL	1,837	-14 %

Πίνακας 3.3: Κόστος υποστήριξης μερικών αλγοριθμικών λειτουργιών

Η προτεινόμενη αρχιτεκτονική επιτρέπει συγκεκριμένες επαναδιαρθρώσεις της μονάδος κατά την τοποθέτησή της, ώστε να είναι δυνατή μια ακριβέστερη προσαρμογή στις απαιτήσεις της εκάστοτε εφαρμογής. Η επαναδιάρθρωση αφορά σε πρόσθεση/αφαίρεση αλγοριθμικών δυνατοτήτων από τη μονάδα, ή αλλιώς, σε

[7] τα συγκεκριμένα αποτελέσματα προκύπτουν για περιοχή αναζήτησης 48×48 και συμπεριλαμβάνουν μνήμη ROM με τον αλγόριθμο PMVFAST (προσμετράται στον κεντρικό έλεγχο).

πρόσθεση/αφαίρεση εντολών (βλ. ειδικά κυκλώματα, Εικόνα 3.5, και φίλτρα ψηφιακής παρεμβολής). Φυσικά, οι παρεμβάσεις αυτές μεταφράζονται σε αντίστοιχη αύξηση/μείωση του κόστους υλοποίησης. Στον Πίνακα 3.3 παρατίθεται το κόστος υλοποίησης κάποιων ενδεικτικών διαμορφώσεων της μονάδος ΕΚ και το αντίστοιχο ποσοστό πόρων που αυτές εξοικονομούν: ο περιορισμός σε απλούς αλγορίθμους αναζήτησης μειώνει το κόστος κατά 19%, ενώ ο περιορισμός σε ακέραια αναζήτηση το μειώνει κατά 14%. Επίσης, όπως αναμέναμε, η υποστήριξη των προηγμένων αλγορίθμων προσθέτει σημαντικό κόστος[8].

3.2.1 Λειτουργία

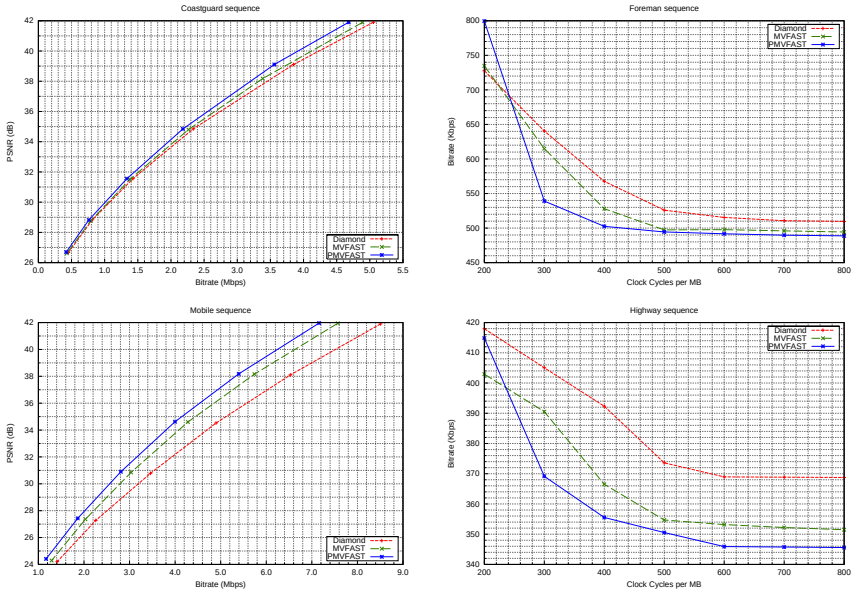
Για τον έλεγχο της λειτουργίας της, ενσωματώνουμε τη μονάδα ΕΚ σε υλισμικό συμπιεστή εικονοροών που ακολουθεί το πρότυπο H.264/AVC[9]. Ως μελέτη περίπτωσης, προγραμματίζουμε κι εκτελούμε ανεξάρτητα τρεις χαρακτηριστικούς αλγορίθμους (τους Diamond, MVFAST, και PMVFAST) για την επεξεργασία τεσσάρων γνωστών δοκιμαστικών βίντεο ανάλυσης 352×288 εικονοστοιχείων (CIF: foreman, highway, coastguard, mobile).

Τα αποτελέσματα των μετρήσεων φαίνονται στην Εικόνα 3.6. Αριστερά, απεικονίζονται οι καμπύλες ποιότητας-διφυορυθμού, PSNR-bitrate, της συμπιεσμένης ροής, οι οποίες προκύπτουν από τη μεταβολή του παράγοντα κβάντισης κατά τις επαναλαμβανόμενες εκτελέσεις του πειράματος. Καταγράφουμε μια καμπύλη για κάθε αλγόριθμο και τις ομαδοποιούμε σε δυο παραστάσεις (πάνω και κάτω πάνελ) που αφορούν σε δυο διαφορετικά βίντεο. Η συνολική λειτουργία του συστήματος (κωδικοποίηση/αποκωδικοποίηση) κρίνεται επιτυχής και, όπως αναμενόταν από τα αποτελέσματα της βιβλιογραφίας, πλέον ποιοτικός αλγόριθμος αποδεικνύεται ο PMVFAST. Μάλιστα, η διαφορά του από τους υπόλοιπους μεγαλώνει στην εικονοροή 'mobile', όπου συναντάμε σαφώς μεγαλύτερη (και δυσκολότερα ανιχνεύσιμη) κινητικότητα των πλακιδίων.

Δεξιά στην Εικόνα 3.6 απεικονίζεται ένα μέτρο για την προσέγγιση της ποιότητας του ταιριάσματος που επιτυγχάνει κάθε αλγόριθμος σε συνάρτηση με το χρόνο που αυτός έχει στη διάθεσή του. Συγκεκριμένα, ορίζουμε χρονικό

[8] σε ορισμένες περιπτώσεις, π.χ., MVFAST και PMVFAST, τα επί μέρους σύνολα των προηγμένων εντολών που απαιτούνται δεν είναι ξένα μεταξύ τους (το κόστος μοιράζεται).

[9] ο συμπιεστής χρησιμοποιεί κωδικοποίηση εντροπίας CAVLC, χωρική πρόβελψη Intra με μπλοκ 16×16 εικονοστοιχείων, μετασχηματισμό και κβαντιστή 52 στάθμεων. Η πρωτότυπη πλακέτα αποτελείται από δυο FPGAs (XCV2P40 και XC2S150), SDRAM 16MB, SRAM 512KB, κάμερα, και διεπαφές για επικοινωνία με PC. Λειτουργεί στα 50 MHz και προσφέρει επεξεργασία πραγματικού χρόνου στα 30 fps με ανάλυση 1024×768 εικονοστοιχεία [63].



Εικόνα 3.6: Χρήση μονάδος στο H.264/AVC. Αριστερά, καμπύλη ποιότητας-διφυορυθμού (3 αλγόριθμοι). Δεξιά, βελτίωση ταιριάσματος με πάροδο χρόνου.

παράθυρο για τη λειτουργία του αλγορίθμου[10] μετά το πέρας του οποίου, ανεξάρτητα από το αν η αναζήτηση έχει τερματίσει, η μονάδα ΕΚ δηλώνει ως αποτέλεσμα το βέλτιστο ταίριασμα που έχει ανακαλυφθεί ως εκείνη τη στιγμή (δηλαδή, το *current_position+best_move*). Φυσικά, όσο καλύτερο είναι το ταίριασμα, τόσο μεγαλύτερη συμπίεση θα επιτύχει το υπόλοιπο σύστημα, άρα τόσο μικρότερο διφυορυθμό θα παρατηρούμε στην έξοδό του. Κρατώντας σταθερό το παράγοντα κβάντισης, μεταβάλουμε το μήκος του παραθύρου κι επαναλαμβάνουμε τη διαδικασία συμπίεσης χρησιμοποιώντας τους τρεις αλγορίθμους σε δυο διαφορετικά βίντεο (πάνω και κάτω πάνελ, Εικόνα 3.6, δεξιά). Τα αποτελέσματα δείχνουν ότι με τη πάροδο του χρόνου οι αλγόριθμοι βελτιώνουν σημαντικά τα ταιριάσματα των πλακιδίων και φθάνουν σε ένα σημείο 'κορεσμού', όπου η περαιτέρω λειτουργία τους δεν προσφέρει κάτι ουσιαστικό στο σύστημα (η πλειοψηφία των αναζητήσεων έχει τερματίσει). Σε αυτό το σημείο φθάνει πρώτος ο PMVFAST (αναμενόμενο), ο οποίος εξάλλου οδηγεί και στο μικρότερο

[10]για τεχνικούς λόγους που σχετίζονται με τη λειτουργία ολόκληρου του κωδικοποιητή, τα χρονικά παράθυρα που χρησιμοποιούνται για την Εικόνα 3.6 συμπεριλαμβάνουν έναν –σχεδόν σταθερό– αριθμό από κύκλους ρολογιού που σχετίζονται με την αντιστάθμιση κίνησης (μετά την εκτίμηση), τη προώθηση του μπλοκ-πρόβλεψη, το συντονισμό των μηχανών Inter/Intra.

διφυορυθμό στην έξοδο του συμπιεστή (άρα παράγει καλύτερα ταιριάσματα, συμπέρασμα συνεπές με τα αποτελέσματα του προηγούμενου πειράματος). Παρατηρούμε ότι στην αρχή του χρόνου λειτουργίας οι προηγμένοι αλγόριθμοι οδηγούν σε χειρότερα αποτελέσματα από τους απλούς, εξαιτίας της συλλογής πληροφοριών και των βημάτων αρχικοποίησης που εκτελούν, διαδικασίες που καθυστερούν τις ουσιαστικές συγκρίσεις περιοχών και τα πραγματικά ταιριάσματα (έπειτα, βέβαια, συγκλίνουν πολύ γρήγορα στο τελικό αποτέλεσμα).

3.2.2 Αξιολόγηση

Προκειμένου να αξιολογήσουμε τη προτεινόμενη μονάδα ΕΚ συγκρίνουμε τα αποτελέσματα της υλοποίησής της με τα αντίστοιχα άλλων αρχιτεκτονικών της βιβλιογραφίας, προγραμματιζόμενων ή μη.

Η προγραμματιζόμενη μονάδα που παρουσιάζεται στην εργασία [34] καταναλώνει παρόμοιο αριθμό από πόρους του FPGA ($2,052$ slices), όμως, προσφέρει πολύ χαμηλότερο βαθμό διεκπεραιωτικότητας: η μονάδα μας υποστηρίζει επεξεργασία βίντεο ανάλυσης $1,024 \times 768$ στα 30 fps, ενώ η [34] υποστηρίζει επεξεργασία πραγματικού χρόνου μόνο για βίντεο χαμηλής ανάλυσης (CIF). Επιπροσθέτως, δεν διαθέτει ψηφιακή παρεμβολή και η περιοχή αναζήτησής της είναι μονο 32×32 εικονοστοιχεία. Οι βασικοί λόγοι για τους οποίους η προτεινόμενη μονάδα ΕΚ εμφανίζει τόσο μεγαλύτερη διεκπεραιωτικότητα είναι η μέθοδος παραλληλοποίησης που εφαρμόσαμε στο συγκριτή περιοχών, η προτεινόμενη τεχνική της υποθετικής εκτέλεσης των εντολών, καθώς και η αποδοτικότητα του εφαρμογοείδιου συνόλου εντολών που σχεδιάσαμε.

Σχετικά με το σύνολο εντολών, παρατηρούμε ότι στην εργασία [34] υλοποιούνται πιο ευέλικτες εντολές, οι οποίες παρέχουν τη δυνατότητα προγραμματισμού γενικότερων αλγορίθμων (ίσως και πέραν του ταιριάσματος περιοχών). Το κόστος της υιοθέτησης μικρών, σχεδόν ανειδίκευτων εντολών, είναι η αύξηση των κύκλων μετάκλησης-αποκωδικοποίησης (fetch-decode) ακόμα και για λειτουργίες που καθίστανται τετριμμένες στα πλαίσια του ταιριάσματος περιοχών. Προκειμένου να ποσοτικοποιήσουμε την εν λόγω διαφορά ανάμεσα στο ευέλικτο σύνολο μικρών εντολών και στο εφαρμογοείδιο σύνολο των πιο σύνθετων εντολών μας, παραθέτουμε τα αποτελέσματα του Πίνακα 3.4. Συγκρίνουμε 3 διαφορετικούς αλγορίθμους αναζήτησης (Four Step, Diamond, και MVFAST) προγραμματισμένους σε 3 διαφορετικές αρχιτεκτονικές (επεξεργαστή γενικής χρήσης 32-bit GPP x86, αρχιτεκτονική [34], και προτεινόμενη) ως προς το μέγεθος του προγράμματος και ως προς τον αριθμό των κύκλων που διαρκεί

Αρχιτεκτον. σετ εντολών	Four Step		Diamond		MVFAST	
	Μέγεθος	*Κύκλοι*	*Μέγεθος*	*Κύκλοι*	*Μέγεθος*	*Κύκλοι*
GPP	1,793	87,219	1,670	73,332	2,493	45,870
Dias et al. [34]	365	5,248	460	4,352	744	2,944
Προτεινόμενη	88	722	84	660	135	511

Πίνακας 3.4: Μέγεθος προγράμματος και κύκλοι εκτέλεσης (ανά πλακίδιο) για διαφορετικούς αλγορίθμους αναζήτησης και για διαφορετικές αρχιτεκτονικές.

η εκτέλεσή του[11]. Όπως γίνεται φανερό, οι κύκλοι της προτεινόμενης αρχιτεκτονικής είναι πολύ λιγότεροι από τους υπόλοιπους (και για τους τρεις αλγορίθμους). Φυσικά, αυτό μεταφράζεται σε πολύ λιγότερο χρόνο εκτέλεσης (ακόμα κι αν λάβουμε υπόψη την υψηλότερη συχνότητα λειτουργίας του GPP).

Επίσης λόγω του αποδοτικότερου συνόλου εντολών, η προτεινόμενη αρχιτεκτονική εμφανίζει γρηγορότερα αποτελέσματα από τη [37], η οποία κάνει χρήση επεξεργαστή RISC και δομοστοιχείων μεγαλύτερου κόστους από αυτά που περιγράψαμε στη παρούσα εργασία. Στη λύση με επεξεργαστή VLIW [36] καταγράφεται πολύ μεγαλύτερη κατανάλωση πόρων (9,000 FPGA slices) με παρόμοιο λόγο διεκπεραιωτικότητας/κόστους, αλλά εξετάζονται πολύ λιγότερες υποψήφιες θέσεις (ταιριάσματα χαμηλότερης ποιότητας) με τοπική μνήμη που εμφανίζει πολύ μεγαλύτερη υποεκμετάλλευση. Η VLSI λύση με επεξεργαστή DSP [43] απαιτεί 8mm^2 με διαδικασία $0.09\mu\text{m}$ (μια ενδεικτική υλοποίηση της προτεινόμενης μονάδας ΕΚ σε VLSI απαιτεί περίπου 1mm^2 στα $0.18\mu\text{m}$).

Ακόμα και σε σύγκριση με μη προγραμματιζόμενες λύσεις, η προτεινόμενη είναι δυνατόν να εμφανίσει σχετική οικονομία πόρων: π.χ., σε σύγκριση με την υλοποίηση του ευρετικού αλγορίθμου FOSS [32]. Φυσικά, η οικονομία αυτή γίνεται πάρα πολύ εμφανής όταν συγκρίνουμε με υλοποιήσεις του αλγορίθμου πλήρους αναζήτησης, π.χ. [22] [24], που βέβαια προσφέρουν αποτελέσματα πολύ καλύτερης ποιότητας (αναμενόμενο tradeoff). Οι συγκρίσεις μπορούν να προχωρήσουν και σε επίπεδο δομοστοιχείων, όπου π.χ., ο προτεινόμενος συγκριτής περιοχών, λόγω των μειωμένων κυκλωμάτων ABS και της σωλήνωσης με αποκλειστικούς πόρους του FPGA (αθροιστών/αφαιρετών), εμφανίζει καλύτερες επιδόσεις από εξειδικευμένες κατασκευές με αθροιστές φύλαξης κρατουμένου [64].

[11] οι κύκλοι του GPP μετρήθηκαν με εργαλεία κατατομής (profiling tools), ενώ οι κύκλοι στην προτεινόμενη αρχιτεκτονική προσεγγίζονται μέσω της χρήσης παραθύρου λειτουργίας για ολόκληρη τη μηχανή Inter μέσα στο συμπιεστή [63] (βλ. προηγούμενη υποενότητα).

Κεφάλαιο 4

Οργάνωση Παράλληλης Μνήμης

Με στόχο την υποστήριξη εφαρμογών γραφικών, η παρούσα διδακτορική διατριβή προτείνει μια οργάνωση της μνήμης με βάση τον ιδανικό αριθμό τραπεζών, $B = E$, για οποιονδήποτε ακέραιο E (το E συμβολίζει το πλήθος των εικονοστοιχείων που μπορούμε να ανακτήσουμε σε έναν κύκλο). Η λύση επιτρέπει την αποφυγή 'δύσκολων' αριθμών –ιδίως των πρώτων– κατά την υλοποίηση. Περαιτέρω, παρακάμπτει τη χρήση περίσσειας τραπεζών συνεισφέροντας έτσι στη σχεδίαση αποδοτικών αρχιτεκτονικών.

Σε αντίθεση με τις υπόλοιπες λύσεις της βιβλιογραφίας, εδώ κάνουμε μια επιπρόσθετη παρατήρηση για τη λειτουργία των αλγορίθμων επεξεργασίας γραφικών, την οποία χρησιμοποιούμε έπειτα ως υπόθεση για την οργάνωση της μνήμης. Όπως περιγράφηκε στην ενότητα 1.2, οι αλγόριθμοι ζητούν σύνολα εικονοστοιχείων από οποιαδήποτε θέση στην εικόνα (απεριόριστη πρόσβαση κατά γραμμές, στήλες, ορθογώνια). Σε αυτή την απαίτηση-περιορισμό αρκείται έως σήμερα η βιβλιογραφία. Οι οργανώσεις που προκύπτουν από αυτή τη μοναδική υπόθεση έχουν τη δυνατότητα εξυπηρέτησης οποιασδήποτε ακολουθίας αιτήσεων, ακόμα κι αν οι θέσεις τους επάνω στην εικόνα εμφανίζουν εντελώς τυχαία σειρά. Όμως οι αλγόριθμοι –στην συντριπτική τους πλειοψηφία– δεν κάνουν εντελώς τυχαίες αιτήσεις, αφού ακολουθούν κάποιο 'σχέδιο' με βάση το οποίο επεξεργάζονται την εικόνα. Με άλλα λόγια, οι θέσεις των διαδοχικών αιτήσεων εμφανίζουν κάποιου είδους συσχέτιση, η οποία καθιστά σχεδόν αχρείαστη την παραπάνω γενικευμένη ικανότητα.

Οι αλγόριθμοι επεξεργάζονται την εικόνα κατά συνεκτικές χωρικά περιοχές. Πολύ σπάνια, αν όχι ποτέ, μεταπηδούν από θέση σε θέση κάνοντας διαδο-

χικές, απομακρυσμένες, ασυσχέτιστες αιτήσεις για εικονοστοιχεία. Άλλωστε, αυτός είναι και ο λόγος για τον οποίο έγινε η αρχική υπόθεση ότι χρειαζόμαστε πρόσβαση κατά 'συμπαγή' σχήματα. Στην εργασία μας επεκτείνουμε αυτήν την ευρέως αποδεκτή υπόθεση και δεχόμαστε το γεγονός ότι η συγκεκριμένη συμπεριφορά των αλγορίθμων παρατηρείται και σε μεγαλύτερη κλίμακα από αυτή του ενός σχήματος. Δηλαδή, τα διαδοχικά σύνολα-αιτήσεις εμφανίζουν μια τοπική συγκέντρωση πάνω στην εικόνα, έστω κι αν ο τόπος αυτός μετακινείται κατά τη διάρκεια της επεξεργασίας. Μερικά αντιπροσωπευτικά παραδείγματα της συγκεκριμένης συμπεριφοράς μπορεί να συναντήσει κανείς σε δισδιάστατους μετασχηματισμούς (π.χ., Fourier), σε ψηφιακά φίλτρα εικόνας, κ.α. Ακόμα και όταν η εξέλιξη του αλγορίθμου δεν είναι απολύτως προβλέψιμη, είναι εξαιρετικά πιθανό να εμφανιστεί κάποια συσχέτιση στις διαδοχικές αιτήσεις του. Ως παράδειγμα αναφέρουμε τους Γρήγορους Αλγόριθμους Ταιριάσματος Περιοχών (εκτίμησης κίνησης), οι οποίοι αλλάζουν πορεία πάνω στην εικόνα παίρνοντας αποφάσεις που εξαρτώνται από τα δεδομένα των πρόσφατων αιτήσεών τους: κατά την απρόβλεπτη πορεία του ο αλγόριθμος σχηματίζει μικρές ομάδες διαδοχικών αιτήσεων, κάθε μια εκ των οποίων καλύπτει –αναγκαστικά– μια μεγάλη τετράγωνη περιοχή της εικόνας, η οποία προορίζεται για σύγκριση με το εκάστοτε πλακίδιο (*Macroblock*) που ορίζει η εφαρμογή.

Με βάση την υπόθεση των συσχετισμένων αιτήσεων, κατασκευάζουμε παράλληλη μνήμη που επιτρέπει την διόρθωση πολλών συγκρούσεων σε σύντομο χρονικό διάστημα. Υπενθυμίζουμε ότι οι συγκρούσεις είναι αναπόφευκτες, αφού στοχεύουμε σε απεριόριστη πρόσβαση κατά {γραμμές,στήλες,ορθογώνια} χρησιμοποιώντας μόνο $B = E$ τράπεζες. Όμως, η κατάλληλη αντιστοίχιση των δεδομένων ελαττώνει την πιθανότητα συγκρούσεων, καθώς και τον χρόνο διόρθωσής τους. Έτσι, το χρονικό επίβαρο κόστος της προτεινόμενης λύσης καθίσταται προτιμότερο από το υλικό επίβαρο κόστος των λύσεων που χρησιμοποιούν πρώτο ή άλλο μεγαλύτερο αριθμό τραπεζών. Επίσης, αποδεικνύεται ότι η νέα λύση οδηγεί σε πιο αποδοτικές αρχιτεκτονικές (με μικρότερη υποεκμετάλλευση) από τις έως σήμερα δημοσιευμένες.

Στο σημείο αυτό πρέπει να τονίσουμε ότι η νέα υπόθεση που εισάγουμε λειτουργεί επικουρικά στην παλαιά. Δηλαδή, η προτεινόμενη οργάνωση μπορεί να εφαρμοστεί και χωρίς τις συσχετίσεις, προσφέροντας μοναδικές δυνατότητες πρόσβασης από μόνη της. Ως εκ τούτου, χωρίζουμε την παρουσίασή της σε δυο κεφάλαια. Στο παρών κεφάλαιο εξετάζουμε τη λύση σα να επρόκειτο για μια ακόμα πρόταση στη στενή βιβλιογραφική προσέγγιση του κεφαλαίου 2, ενώ στο επόμενο κεφάλαιο περιγράφουμε τις πρωτοποριακές ιδιότητες της αντιστοίχισής μας και το πως αυτές συνδυάζονται με την νέα υπόθεση.

4.1 Προτεινόμενη λύση

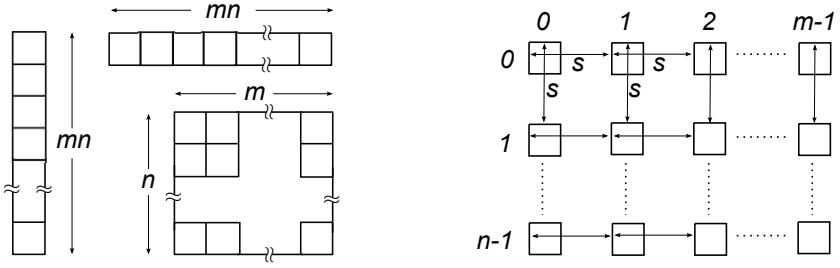
Η προτεινόμενη οργάνωση της παράλληλης μνήμης, όπως και οι προγενέστερές της, πραγματώνεται μέσα από μια συνάρτηση αντιστοίχισης εικονοστοιχείων σε τράπεζες. Πριν από την παράθεση της συνάρτησης και τις αποδείξεις των ιδιοτήτων της, εξηγούμε την ορολογία που χρησιμοποιείται για το σκοπό αυτό στο υπόλοιπο της εργασίας.

4.1.1 Ορολογία/σημειογραφία

Εφεξής, η θέση ενός εικονοστοιχείου πάνω στην εικόνα θα δίνεται με ένα διάνυσμα (x, y), όπου x ακέραιος που σημειώνει τη στήλη και y ακέραιος που σημειώνει τη γραμμή του στοιχείου (οριζόντια και κάθετη απόσταση πάνω στο επίπεδο). Χωρίς βλάβη της γενικότητας, δεχόμαστε ότι το πάνω-αριστερά στοιχείο της εικόνας είναι το $(0, 0)$ και ότι τα x και y αυξάνονται κατά μονάδες καθώς κινούμαστε δεξιά και κάτω στην εικόνα, αντίστοιχα. Ένα σύνολο εικονοστοιχείων, ή αλλιώς ένα σύνολο-αίτηση, θα καθορίζεται από το σχήμα που χρησιμοποιείται για την πρόσβαση και από τη θέση του επάνω στην εικόνα. Συμβιβαστικά, η θέση αυτή θα δίνεται με ένα διάνυσμα (x, y) που προσδιορίζει το πάνω-αριστερά εικονοστοιχείο του εκάστοτε σχήματος.

Θεωρούμε ότι όλα τα σχήματα πρόσβασης έχουν την ίδια επιφάνεια (περιέχουν τον ίδιο αριθμό στοιχείων), ο οποίος παραγοντοποιείται ως $E = m \cdot n$, όπου m και n η οριζόντια και η κάθετη διάσταση των ορθογωνίων σχημάτων. Γενικότερα, τα *βασικά* σχήματα που μας ενδιαφέρουν είναι τα λεγόμενα *συμπαγή*, δηλαδή η ποικιλία {γραμμή,στήλη,ορθογώνιο}, και τα λεγόμενα *αραιά ορθογώνια*. Ένα αραιό ορθογώνιο αποτελείται από ένα πλέγμα εικονοστοιχείων, τα οποία απέχουν μεταξύ τους σταθερή, ακέραιη, απόσταση s επάνω στην πραγματική εικόνα (πιο συγκεκριμένα, το πλέγμα έχει διανύσματα βάσης τα $(0, s)$ και $(s, 0)$). Το πλέγμα είναι πεπερασμένο κι αποτελείται από m στοιχεία στην οριζόντια διεύθυνση και n στην κατακόρυφη· συνολικά, το αραιό ορθογώνιο έχει $m \cdot n$ στοιχεία. Κάθε αραιό ορθογώνιο συμβολίζεται ως *αραιό-s*, όπου s η παράμετρός του. Εμφανώς, ένα αραιό-1 είναι ένα συμπαγές ορθογώνιο. Η ορολογία μπορεί να γενικευτεί για να συμπεριλάβει λιγότερο συμμετρικά αραιά σχήματα, όπως το *αραιό-s_1-s_2*, στο οποίο έχουμε απόσταση μεταξύ των εικονοστοιχείων κατά s_1 στην οριζόντια διεύθυνση και κατά s_2 στην κατακόρυφη. Στην Εικόνα 4.1 φαίνονται τα *βασικά* σχήματα πρόσβασης για $E = m \cdot n$.

Στις μαθηματικές εκφράσεις που ακολουθούν το πηλίκο της ακέραιας διαίρεσης του αριθμού p με τον q θα γράφεται ως $\lfloor \frac{p}{q} \rfloor$, ενώ το υπόλοιπο θα γράφεται



Εικόνα 4.1: *Βασικά* σχήματα πρόσβασης. Αριστερά: *γραμμή, στήλη, ορθογώνιο* (συμπαγή σχήματα). Δεξιά: *αραιό-s*.

ως $(p) \bmod (q)$, ή απλούστερα ως $p \bmod q$. Όταν το υπόλοιπο είναι μηδέν, δηλαδή όταν ο q διαιρεί τον p ακριβώς, θα γράφουμε $q|p$. Τέλος, με $\mathbb{N}_t$ θα συμβολίζουμε το σύνολο των t αρχικών φυσικών αριθμών $\{0, 1, 2, \ldots, t-1\}$.

4.1.2 Η συνάρτηση αντιστοίχισης $\Phi(x, y)$

Όπως τονίσαμε παραπάνω, η προτεινόμενη λύση χρησιμοποιεί πλήθος τραπεζών ίσο με το πλήθος των εικονοστοιχείων στα οποία επιθυμούμε να έχουμε άμεση πρόσβαση, δηλαδή $B = E = mn$. Η αντιστοίχιση που εφαρμόζουμε προκύπτει από τη σύνθεση δυο γραμμικών λοξώσεων. Ειδικότερα, η εικόνα χωρίζεται σε κατακόρυφες λωρίδες πάχους m στηλών. Μέσα σε κάθε λωρίδα οι στήλες υπόκεινται σε γραμμική λόξωση κατά n. Επιπρόσθετα, οι λωρίδες πάχους m υπόκεινται σε γραμμική λόξωση μεταξύ τους κατά το ελάχιστο, δηλαδή κατά 1. Το αποτέλεσμα της σύνθεσης αυτής είναι μια μη-γραμμική αντιστοίχιση, την οποία ονομάζουμε $\Phi(x, y)$ και γράφουμε αναλυτικά ως

$$\Phi(x, y) = \left(x \cdot n + y + \left\lfloor \frac{x}{m} \right\rfloor \right) \bmod (mn) \tag{4.1}$$

Στη συνάρτηση αυτή, ο όρος xn προσδίδει τη λόξωση των στηλών κατά n, ενώ ο προσθετικός όρος $\lfloor \frac{x}{m} \rfloor$ ευθύνεται για τη λόξωση των λωρίδων κατά 1, καθώς και για τις εξαιρετικά ενδιαφέρουσες ιδιότητες της $\Phi(x, y)$ που θα μελετήσουμε παρακάτω. Στην Εικόνα 4.2 φαίνεται ένα παράδειγμα της προτεινόμενης αντιστοίχισης με 16 τράπεζες και $m = n = 4$.

Η επιλογή να αποφύγουμε τη χρήση συναρτήσεων XOR (ή άλλων παρόμοιων αντιστοιχίσεων) και να εργαστούμε με γραμμικές λοξώσεις οφείλεται στην ισοτροπικότητα και τα πλεονεκτήματά της, όπως αυτά εξηγήθηκαν στην ενότητα 2.3.2. Τυπικά, βέβαια, η $\Phi(x, y)$ είναι μια μη-γραμμική λόξωση. Όμως, η

0	4	8	C	1	5	9	D	2	6	A	E	3	7	B	F	4	8	C	0
1	5	9	D	2	6	A	E	3	7	B	F	4	8	C	0	5	9	D	1
2	6	A	E	3	7	B	F	4	8	C	0	5	9	D	1	6	A	E	2
3	7	B	F	4	8	C	0	5	9	D	1	6	A	E	2	7	B	F	3
4	8	C	0	5	9	D	1	6	A	E	2	7	B	F	3	8	C	0	4
5	9	D	1	6	A	E	2	7	B	F	3	8	C	0	4	9	D	1	5
6	A	E	2	7	B	F	3	8	C	0	4	9	D	1	5	A	E	2	6
7	B	F	3	8	C	0	4	9	D	1	5	A	E	2	6	B	F	3	7
8	C	0	4	9	D	1	5	A	E	2	6	B	F	3	7	C	0	4	8
9	D	1	5	A	E	2	6	B	F	3	7	C	0	4	8	D	1	5	9
A	E	2	6	B	F	3	7	C	0	4	8	D	1	5	9	E	2	6	A
B	F	3	7	C	0	4	8	D	1	5	9	E	2	6	A	F	3	7	B
C	0	4	8	D	1	5	9	E	2	6	A	F	3	7	B	0	4	8	C
D	1	5	9	E	2	6	A	F	3	7	B	0	4	8	C	1	5	9	D
E	2	6	A	F	3	7	B	0	4	8	C	1	5	9	D	2	6	A	E
F	3	7	B	0	4	8	C	1	5	9	D	2	6	A	E	3	7	B	F
0	4	8	C	1	5	9	D	2	6	A	E	3	7	B	F	4	8	C	0
1	5	9	D	2	6	A	E	3	7	B	F	4	8	C	0	5	9	D	1

Εικόνα 4.2: Η προτεινόμενη αντιστοίχιση $\Phi(x, y)$ για $B = 16$ και $m = n = 4$. Απεικονίζονται τρία διαφορετικά σύνολα-αιτήσεις ως παραδείγματα πρόσβασης.

προσεκτικά κατασκευασμένη μορφή της οδηγεί σε ισοτροπική οργάνωση εντός των λωρίδων και σε εύκολα υπολογίσιμες ασυνέχειες στα όριά τους. Έτσι, η δρομολόγηση των δεδομένων μπορεί σε αρκετές πρακτικές εφαρμογές να επιτευχθεί μόνο με κυκλικές μεταθέσεις, ενώ στη γενική περίπτωση απαιτείται μια απλή τροποποίηση του κυκλικού ολισθητή.

Πριν προχωρήσουμε στην μελέτη της $\Phi(x, y)$ οφείλουμε να σημειώσουμε ότι η συγκεκριμένη αντιστοίχιση έχει αναφερθεί –μεταξύ πολλών άλλων– στις πρώτες εργασίες της σχετικής βιβλιογραφίας [45], [47]. Όμως, εκεί δεν ερευνήθηκαν ενδελεχώς οι πολύ χρήσιμες ιδιότητες της $\Phi(x, y)$ και δεν προέκυψε ποτέ μια οργάνωση με τις δυνατότητες που προσφέρει η παρούσα (ίσως, λόγω του ελαφρώς διαφορετικού προσανατολισμού των συγκεκριμένων εργασιών).

4.2 Δυνατότητες άμεσης πρόσβασης

Η παρούσα ενότητα μελετά τις δυνατότητες πρόσβασης που προσφέρει η $\Phi(x, y)$ όταν παραβλέψουμε τις πιθανές συσχετίσεις των διαδοχικών αιτήσεων. Δηλαδή, βασίζεται στις κλασσικές υποθέσεις για να μελετήσει την απεριόριστη πρόσβαση ενός κύκλου κατά $\{$γραμμή,στήλη,ορθογώνιο,αραιό-$s\}$.

Δεδομένων των αποτελεσμάτων της ενότητας 2.3.1 και του γεγονότος ότι η προτεινόμενη λύση χρησιμοποιεί τον ιδανικό αριθμό τραπεζών $B = E$, ήδη πριν από την ανάλυση που ακολουθεί, μπορούμε να προβλέψουμε ότι θα εμφανιστούν συγκρούσεις κατά την πρόσβαση στη μνήμη μας. Όμως, θα δείξουμε εδώ ότι η Φ ελαχιστοποιεί τον αριθμό των συγκρούσεων εντός του συνόλου-αίτησης σε μόλις μια. Μάλιστα, η σύγκρουση αυτή δεν εμφανίζεται σε όλες τις περιπτώσεις: υπάρχουν σχήματα που προσφέρουν εντελώς απεριόριστη πρόσβαση (π.χ., στήλη, αραιό-m), ενώ για οποιοδήποτε από τα εναπομείναντα σχήματα υπάρχει τουλάχιστον ένα απλό πλέγμα αιτήσεων το οποίο καλύπτει ολόκληρη την εικόνα επιτρέποντας πλήρη πρόσβαση άνευ συγκρούσεων (γενικότερα, τα εναπομείναντα σχήματα εμφανίζουν πιθανότητα σύγκρουσης το πολύ $\frac{m-1}{m}$).

4.2.1 Κατασκευή και μελέτη εξίσωσης τραπεζών

Στις επόμενες παραγράφους παραθέτουμε τις δυνατότητες πρόσβασης της Φ ξεχωριστά για κάθε σχήμα πρόσβασης υπό τη μορφή θεωρημάτων και πορισμάτων. Οι αποδείξεις των θεωρημάτων χρησιμοποιούν το ακόλουθο Λήμμα 4.2.1, το οποίο απαντά έμμεσα σε μια πολύ βασική ερώτηση: πότε δυο τυχαία εικονοστοιχεία (x_1, y_1) και (x_2, y_2) ΔΕΝ αντιστοιχίζονται από τη Φ στην ίδια τράπεζα μνήμης; Το λήμμα αρκείται στη μελέτη περιπτώσεων όπου τα (x_1, y_1) και (x_2, y_2) εμφανίζουν περιορισμένη απόσταση μεταξύ τους (αφού μας ενδιαφέρουν σχήματα με περιορισμένο εμβαδόν). Επίσης, γενικεύει την περιγραφή της απόστασης των δυο εικονοστοιχείων χρησιμοποιώντας ειδικές παραμέτρους (αφού μας ενδιαφέρει μια ποικιλία σχημάτων). Για να απαντήσουμε στην παραπάνω ερώτηση εξετάζουμε το πότε η Φ δίνει την ίδια έξοδο για δυο διαφορετικές εισόδους. Δηλαδή, εξισώνουμε τις αντιστοιχίσεις $\Phi(x_1, y_1)$ και $\Phi(x_2, y_2)$ κι εξετάζουμε την εξίσωση ισοτιμίας $\Phi(x_1, y_1) - \Phi(x_2, y_2) \equiv 0 \pmod{mn}$. Επιπροσθέτως, εκφράζουμε το δεύτερο εικονοστοιχείο με συντεταγμένες σχετικές ως προς το πρώτο: χρησιμοποιούμε τις συνετατγμένες $(i \cdot s,\ j \cdot s)$ όπου θεωρούμε μεταβλητά τα i και j [1]. Η παράμετρος s υπεισέρχεται εξαιτίας των αραιών ορθογωνίων ($s = 1$ για τη μελέτη συμπαγών σχημάτων).

Λήμμα 4.2.1. *Για οποιουσδήποτε σταθερούς ακεραίους x, $n > 0$, $m > 0$, και*

[1] με άλλα λόγια, επιλέγουμε ένα τυχαίο εικονοστοιχείο (x, y) το οποίο θεωρούμε σταθερό επάνω στην εικόνα κι εκφράζουμε τα κοντινά σε αυτό εικονοστοιχεία με συντεταγμένες σχετικές ως προς (x, y). Στη συνέχεια εξετάζουμε ποια από τα κοντινά εικονοστοιχεία είναι αποθηκευμένα στην ίδια τράπεζα μνήμης με το (x, y).

οποιονδήποτε ακέραιο s με $s|m$, η ακόλουθη εξίσωση ισοτιμίας 4.2

$$i \cdot s \cdot n + j \cdot s + \left\lfloor \frac{x + i \cdot s}{m} \right\rfloor - \left\lfloor \frac{x}{m} \right\rfloor \equiv 0 \pmod{mn} \tag{4.2}$$

έχει το πολύ δυο λύσεις ως προς (i, j) όταν $0 \leq i < m$ και $|j| < n$, τις εξής:

$$\begin{aligned} & (i, j) = (0, 0) \\ \text{και} \quad & (i, j) = (m - 1, n - 1) \end{aligned}$$

Απόδειξη. Χρησιμοποιούμε απλές ιδιότητες και βασικά αποτελέσματα από τη θεωρία αριθμών [65] [66]. Ξεκινάμε παίρνοντας διαδοχικά:

$$\left\lfloor \frac{x + i \cdot s}{m} \right\rfloor - \left\lfloor \frac{x}{m} \right\rfloor = \left\lfloor \frac{i \cdot s}{m} \right\rfloor + \left\lfloor \frac{x \bmod (m) + (i \cdot s) \bmod (m)}{m} \right\rfloor = \left\lfloor \frac{i \cdot s}{m} \right\rfloor + \varepsilon_{xi}$$

με το κλάσμα ε_{xi} να αποτιμάται –ξεκάθαρα– σε 0 ή 1. Περαιτέρω, αφού $s|m$ (δηλαδή, $m = s \cdot m'$) παίρνουμε:

$$\begin{aligned} & \left(i \cdot s \cdot n + j \cdot s + \left\lfloor \tfrac{i \cdot s}{m} \right\rfloor + \varepsilon_{xi}\right) \bmod (mn) \; = \\ & \left(i \cdot s \cdot n + j \cdot s + \left\lfloor \tfrac{i \cdot s}{m} \right\rfloor + \varepsilon_{xi} - \left\lfloor \tfrac{i \cdot s}{m} \right\rfloor \cdot m \cdot n\right) \bmod (mn) \; = \\ & \left(i \cdot s \cdot n + j \cdot s + \left\lfloor \tfrac{i}{m'} \right\rfloor + \varepsilon_{xi} - \left\lfloor \tfrac{i}{m'} \right\rfloor \cdot m' \cdot s \cdot n\right) \bmod (mn) = \\ & \left(s \cdot n \cdot (i - \left\lfloor \tfrac{i}{m'} \right\rfloor \cdot m') + j \cdot s + \left\lfloor \tfrac{i}{m'} \right\rfloor + \varepsilon_{xi}\right) \bmod (mn) \; = \\ & \left(n \cdot s \cdot (i) \bmod (m') + j \cdot s + \left\lfloor \tfrac{i}{m'} \right\rfloor + \varepsilon_{xi}\right) \bmod (mn) \end{aligned}$$

Επομένως, η ισοτιμία 4.2 μπορεί να γραφεί ως

$$\Psi(i, j) + \varepsilon_{xi} \equiv 0 \pmod{mn} \tag{4.3}$$

όπου $\Psi(i, j) = (i) \bmod (m') \cdot \mathbf{ns} + j \cdot \mathbf{s} + \left\lfloor \frac{i}{m'} \right\rfloor$.

Περίπτωση πρώτη: $0 \leq i < m$ και $0 \leq j < n$

Στην περίπτωση αυτή, η αναζήτηση λύσεων (i, j) περιορίζεται στο $\mathbb{N}_m \times \mathbb{N}_n$.

Πρώτα θα αποδείξουμε ότι η συνάρτηση $\Psi(i, j)$: $\mathbb{N}_m \times \mathbb{N}_n \to \mathbb{N}_{mn}$ είναι αμφιρριπτική (1-1 κι επί, bijective). Για να δείξουμε ότι είναι ερριπτική (1-1, injective), παρατηρούμε ότι η $\Psi(i, j)$ εκφράζει ένα πεπερασμένο σύστημα αρίθμησης μικτής βάσης (finite mixed-radix numeral system) της απλής μορφής [67]:

$$\begin{aligned} (\alpha_2 \cdot \beta_1 \beta_0 + \alpha_1 \cdot \beta_0 + \alpha_0) \quad & , \quad 0 \leq \alpha_0 \leq \beta_0 - 1 \\ & , \quad 0 \leq \alpha_1 \leq \beta_1 - 1 \end{aligned}$$

με $\beta_1 = n,\ \ \beta_0 = s,\ \ \alpha_2 = i \bmod (m'),\ \ \alpha_1 = j,\ \ \alpha_0 = \lfloor \frac{i}{m'} \rfloor$. Κάθε αριθμός αναπαρίσταται με μοναδικό τρόπο σε ένα τέτοιο σύστημα αρίθμησης ως $\alpha_2\alpha_1\alpha_0$. Επιπλέον, κάθε (i,j) αντιστοιχεί σε μια μοναδική τριάδα $\alpha_2\alpha_1\alpha_0$ (κάθε j αντιστοιχεί σε ένα μοναδικό α_1 και κάθε i αντιστοιχεί σε ένα μοναδικό ζεύγος $\alpha_2\alpha_0$). Συνδυάζοντας τις δυο αυτές αλήθειες συμπεραίνουμε ότι κάθε (i,j) οδηγεί σε μια ξεχωριστή τιμή την Ψ. Σχετικά με τον επιρριπτικότητα (surjection) της Ψ, είναι εύκολο να δει κανείς ότι $0 \leq \Psi(i,j) \leq mn-1$. Δηλαδή, η Ψ αντιστοιχεί $n \cdot m$ στοιχεία σε ένα σύνολο $n \cdot m$ στοιχείων. Αφού κάθε αντιστοίχιση είναι μοναδική (δείξαμε ότι είναι 1-1) η Ψ καλύπτει όλο το πεδίο τιμών $\mathbb{N}_{mn}$.

Αφού ισχύει ότι $\varepsilon_{xi} \leq 1$, έχουμε ότι $\Psi(i,j) + \varepsilon_{xi} \not> mn$ στο $\mathbb{N}_m \times \mathbb{N}_n$. Επομένως, η εξίσωση 4.3 μπορεί να έχει λύση μόνο όταν $\Psi(i,j) + \varepsilon_{xi} = 0$, ή όταν $\Psi(i,j) + \varepsilon_{xi} = mn$. Αφού η $\Psi(i,j)$ είναι αμφίρριψη, υπάρχει μόνο ένα σημείο (i,j) για το οποίο $\Psi(i,j) = 0$. Εύκολα παρατηρούμε ότι το σημείο αυτό είναι το $(0,0)$ και ότι αποτελεί πάντα λύση της εξίσωσης 4.3. Περαιτέρω, αν και μόνον αν $\varepsilon_{xi} = 1$ όταν η Ψ παίρνει τη μέγιστη τιμή της, τότε υπάρχει και δεύτερη λύση για την 4.3. Η λύση αυτή πρέπει να είναι ένα σημείο (i,j) για το οποίο θα έχουμε $\Psi(i,j) = mn-1$. Αφού η $\Psi(i,j)$ είναι αμφίρριψη, υπάρχει μόνο ένα τέτοιο –υποψήφιο– σημείο, το $(m-1, n-1)$.

Περίπτωση δεύτερη: $0 \leq i < m$ και $-n < j < 0$

Στην περίπτωση αυτή η εξίσωση 4.3 δεν έχει λύσεις. Είναι εύκολο να δούμε ότι $-mn < \Psi(i,j) + \varepsilon_{xi} < mn$. Επίσης, δείχνουμε με απαγωγή σε άτοπο ότι $\Psi(i,j) + \varepsilon_{xi} \neq 0$. Υποθέτουμε ότι υπάρχει σημείο (i_r, j_r) το οποίο αποτιμά την παράσταση σε μηδέν, κι άρα

$$\begin{aligned} ns \cdot (i_r \bmod m') + \lfloor \tfrac{i_r s}{m} \rfloor + \left\lfloor \tfrac{x \bmod m + (i_r s) \bmod m}{m} \right\rfloor = |j_r| s \quad \Rightarrow \\ s \ < \ ns \cdot (i_r \bmod m') + \lfloor \tfrac{x \bmod m + i_r s}{m} \rfloor \ < \ ns \end{aligned}$$

το οποίο είναι άμεση συνέπεια του περιορισμού που έχουμε επιβάλει στο j σε αυτή τη δεύτερη περίπτωση που μελετάμε. Παρατηρούμε ότι η τιμή $i_r \bmod m'$ πρέπει να είναι μηδέν, αλλιώς η παραπάνω έκφραση δεν μπορεί να αποτιμηθεί σε ακέραιο μικρότερο του ns. Δηλαδή, το i_r πρέπει να είναι πολλαπλάσιο του m', συγκεκριμένα $i_r = i'_r \cdot m'$. Με αντικατάσταση αυτού του i_r στην παραπάνω έκφραση συμπεραίνουμε ότι ύπαρξη του υποτιθέμενου (i_r, j_r) συνεπάγεται την αλήθεια της ανισότητας $s < i'_r < ns$. Άτοπο: η ανισότητα αυτή δεν μπορεί να είναι αληθής γιατί από την υπόθεση του λήμματος έχουμε ότι $i_r < m$ και άρα $i'_r < s$. □

Εστιάζουμε τη μελέτη της εξίσωσης του Λήμματος 4.2.1 στην περίπτωση που το σταθερό εικονοστοιχείο (x,y) βρίσκεται σε συγκεκριμένη θέση πάνω

στην εικόνα (κοντά στην αριστερή άκρη οποιασδήποτε κάθετης λωρίδας της διαιρεμένης από το Φ εικόνας). Ο συγκεκριμένος περιορισμός θέσης ελαχιστοποιεί τον αριθμό των λύσεων της εν λόγω εξίσωσης σε μόνο μια. Το ακόλουθο Λήμμα 4.2.2 λειτουργεί συμπληρωματικά στο 4.2.1 και οδηγεί στα πορίσματα των θεωρημάτων της παρούσας υποενότητας.

Λήμμα 4.2.2. *Για οποιουσδήποτε σταθερούς ακεραίους $n > 0$, $m > 0$, s με $s|m$, και x με $x \bmod m < s$, η εξίσωση ισοτιμίας 4.2 έχει ακριβώς μια λύση όταν $0 \leq i < m$ και $|j| < n$, την $(i, j) = (0, 0)$.*

Απόδειξη. Η απόδειξη είναι ίδια με αυτή του Λήμματος 4.2.1. Η μοναδική διαφορά εντοπίζεται στο γεγονός ότι για $i = m-1$ και $x \bmod (m) < s$ παίρνουμε

$$\varepsilon_{xi} = \left\lfloor \frac{x \bmod (m) + (i \cdot s) \bmod (m)}{m} \right\rfloor = \left\lfloor \frac{x \bmod (m) - s + m}{m} \right\rfloor = 0$$

διότι $x \bmod (m) - s + m < m$. Αφού το ε_{xi} γίνεται μηδέν όταν η $\Psi(i, j)$ αποκτά τη μέγιστη τίμη της, $mn-1$, δεν υπάρχει σημείο (i, j) τέτοιο που να οδηγεί στο $\Psi(i, j)+\varepsilon_{xi} = mn$. Άρα, η μοναδική λύση της 4.2 είναι το σημείο (i, j) για το οποίο παίρνουμε $\Psi(i, j) + \varepsilon_{xi} = 0$, δηλαδή το $(i, j) = (0, 0)$. □

4.2.2 Μελέτη πρόσβασης ανά βασικό σχήμα

Παρακάτω καθορίζονται τα βασικά εκείνα σχήματα που επιτρέπουν απεριόριστη πρόσβαση επάνω στην εικόνα: το αραιό-m και η στήλη. Επίσης, αποδεικνύεται ότι το ορθογώνιο, το αραιό-s με $s \neq m$, και η γραμμή, εμφανίζουν το πολύ μια σύγκρουση ανά αίτηση. Μάλιστα, η σύγκρουση αυτή εμφανίζεται πάντα σε ένα συγκεκριμένο εικονοστοιχείο του συνόλου-αίτηση: στο κάτω-δεξιά (δηλαδή, στο τελευταίο). Η συγκεκριμένη ιδιαιτερότητα καθιστά ακόμα ευκολότερη την αναγνώριση και αποκατάσταση του ελλείποντος εικονοστοιχείου όταν εμφανιστεί σύγκρουση σε κάποιο σύνολο-αίτηση. Περαιτέρω, παρατίθενται πορίσματα για τον προσδιορισμό συγκεκριμένων θέσεων επάνω στην εικόνα, οι οποίες επιτρέπουν πρόσβαση άνευ συγκρούσεων ακόμα και για το ορθογώνιο, το αραιό-s, και τη γραμμή.

Το πρώτο από τα θεωρήματα εξετάζει την πρόσβαση κατά ορθογώνια χωρίς να θέτει περιορισμούς στη θέση του συνόλου-αίτηση επάνω στην εικόνα.

Θεώρημα 4.2.3 (Ορθογώνιο). *Η χρήση της αντιστοίχισης Φ επιτρέπει πρόσβαση σε οποιαδήποτε $m \times n$ ορθογώνια περιοχή της εικόνας εμφανίζοντας το πολύ μια σύγκρουση. Αν εμφανιστεί σύγκρουση, τότε αυτή θα είναι στο κάτω-δεξιά εικονοστοιχείο του συνόλου-αίτηση.*

Απόδειξη. Υποθέτουμε ένα $m \times n$ ορθογώνιο A_{xy} της εικόνας με προέλευση το τυχαίο (x, y) πάνω στην εικόνα. Επιλέγουμε τυχαίο εικονοστοιχείο (x_p, y_p) μέσα στο A_{xy} κι εξετάζουμε αν τα λοιπά εικονοστοιχεία του A_{xy} αποθηκεύονται στην ίδια τράπεζα με το (x_p, y_p). Συγκεκριμένα, δυο στοιχεία (x_q, y_q) και (x_p, y_p) αποθηκεύονται στην ίδια τράπεζα όταν

$$\Phi(x_p, y_p) = \Phi(x_q, y_q) \tag{4.4}$$

Γράφουμε κάθε (x_q, y_q) του A_{xy} με σχετικές συντεταγμένες (i, j) ως προς το (x_p, y_p) και το αντικαθιστούμε στην 4.4. Η διαδικασία αυτή οδηγεί στην εξίσωση 4.2 του Λήμματος 4.2.1 με $s = 1$. Δηλαδή, οι λύσεις της 4.2 –εκτός της $(i, j) = (0, 0)$– ορίζουν τα εικονοστοιχεία του A_{xy} που αποθηκεύονται στην ίδια τράπεζα με το (x_p, y_p). Σύμφωνα με το Λήμμα 4.2.1, αν ένα (x_q, y_q) είναι λύση της 4.4, τότε το (x_q, y_q) εντοπίζεται σε απόσταση $(i, j) = (m - 1, n - 1)$ από το (x_p, y_p). Δεδομένων των διαστάσεων του A_{xy}, το μοναδικό (x_p, y_p) για το οποίο η λύση (x_q, y_q) εντοπίζεται εντός του A_{xy} είναι το $(x_p, y_p) = (x, y)$ με $(x_q, y_q) = (x + m - 1, y + n - 1)$. Σημειώνουμε εδώ ότι, τυπικά, εξετάζουμε με αυτόν τον τρόπο όλα τα στοιχεία (x_p, y_p) εντός του A_{xy} και κάθε φορά εκφράζουμε στην 4.4 μόνο τα στοιχεία (x_q, y_q) που βρίσκονται δεξιά, $i \geq 0$, του εκάστοτε (x_p, y_p). Αυτό αρκεί για να έχουμε ελέγξει στο τέλος όλα τα δυνατά ζεύγη εικονοστοιχείων του ορθογωνίου. Συνοψίζοντας, η οργάνωση με χρήση του Φ διασφαλίζει ότι κάθε εικονοστοιχείο του A_{xy}, με εξαίρεση το τελευταίο (στην κάτω δεξιά γωνία του A_{xy}), αποθηκεύεται σε ξεχωριστή τράπεζα μνήμης. □

Επεκτείνοντας τα παραπάνω, διαπιστώνουμε ότι το ορθογώνιο προσφέρει πρόσβαση ενός κύκλου άνευ συγκρούσεων όταν ανακτηθεί από συγκεκριμένες περιοχές της εικόνας. Οι περιοχές αυτές εντοπίζονται αποκλειστικά εντός των κάθετων λωρίδων πάχους m της διαιρεμένης από το Φ εικόνας. Δηλαδή, όταν το ορθογώνιο δεν καταλαμβάνει εικονοστοιχεία από δυο διαφορετικές λωρίδες, τότε δεν εμφανίζει συγκρούσεις (βλ. παράδειγμα σχήματος 4.2). Τυπικά, για οποιοδήποτε από αυτά τα ορθογώνια έχουμε ότι $x \bmod m = 0$ κι επομένως μπορούμε να χρησιμοποιήσουμε το Λήμμα 4.2.2 αντί του Λήμματος 4.2.1 κατά τη διαδικασία εξέτασης της εξίσωσης 4.4. Έτσι, δεν θα εντοπίσουμε κανένα ζεύγος εικονοστοιχείων στην ίδια τράπεζα (δεν υπάρχει δεύτερη λύση για την 4.2) και θα καταλήξουμε ότι

Πόρισμα 4.2.4. *Η χρήση της αντιστοίχισης Φ επιτρέπει πρόσβαση άνευ συγκρούσεων σε οποιαδήποτε $m \times n$ ορθογώνια περιοχή της εικόνας που έχει προέλευση (x, y) με $x \bmod m = 0$.*

Συνεχίζουμε με την εξέταση μιας γενίκευσης του ορθογωνίου: το αραιό-s. Όπως και με το αραιό-1, έχουμε ότι

Θεώρημα 4.2.5 (Αραιό-s). *Όταν $s|m$, η χρήση της αντιστοίχισης Φ επιτρέπει πρόσβαση σε οποιοδήποτε αραιό-s ορθογώνιο της εικόνας εμφανίζοντας το πολύ μια σύγκρουση. Αν εμφανιστεί σύγκρουση, τότε αυτή θα είναι στο κάτω-δεξιά εικονοστοιχείο του συνόλου-αίτηση.*

Απόδειξη. Ακολουθούμε την επιχειρηματολογία που αναπτύξαμε στην απόδειξη του θεωρήματος 4.2.3. Εδώ, τα στοιχεία ενός αραιού-s σχήματος μπορούν να καταγραφούν με σχετικές συντεταγμένες (i', j') μεταξύ τους, όπου $i' = i \cdot s$ και $j' = j \cdot s$, με $i \in \mathbb{N}_m$ και $j \in \mathbb{N}_n$. Έτσι, ελέγχοντας τα εικονοστοιχεία του αραιού-s μέσω της εξίσωσης $\Phi(x_p, y_p) = \Phi(x_p + i', y_p + j')$ παίρνουμε ακριβώς την μορφή της εξίσωσης 4.2. Αφού $s|m$, μπορούμε να κάνουμε χρήση του Λήμματος 4.2.1 (και των υπόλοιπων επιχειρημάτων του Θ. 4.2.3) για να ολοκληρώσουμε την παρούσα απόδειξη. □

Παρόμοια με το απλό ορθογώνιο, όταν το αραιό-s βρίσκεται σε συγκεκριμένες θέσεις της εικόνας, τότε δεν εμφανίζει καθόλου συγκρούσεις. Συγκεκριμένα, αυτό συμβαίνει όταν το αραιό-s βρίσκεται σε απόσταση μικρότερη από s από το αριστερό άκρο μιας οποιασδήποτε κατακόρυφης λωρίδας πάχους m της διαιρεμένης εικόνας. Σε αυτές τις θέσεις ισχύει ότι $x \bmod m < s$ κι επομένως μπορούμε να χρησιμοποιήσουμε το Λήμμα 4.2.2 για να δείξουμε ότι

Πόρισμα 4.2.6. *Όταν $s|m$ και $x \bmod m < s$, η χρήση της αντιστοίχισης Φ επιτρέπει πρόσβαση άνευ συγκρούσεων σε οποιοδήποτε αραιό-s ορθογώνιο με θέση (x, y) πάνω στη εικόνα.*

Παρατηρούμε εδώ ότι, σε σχέση με το απλό ορθογώνιο, το αραιό-s διαθέτει σημαντικά περισσότερες θέσεις πάνω στην εικόνα για πρόσβαση άνευ συγκρούσεων: διαθέτει περισσότερες κατά s. Βέβαια, και για τα δυο σχήματα, οι προνομιακές αυτές θέσεις είναι αρκετές για να κατασκευάσουμε –πολλαπλά– πλέγματα αιτήσεων για την κάλυψη ολόκληρης της εικόνας. Πάραυτα, όσο μεγαλύτερο το s, τόσο μεγαλύτερες 'πιθανότητες' έχουμε να αποφύγουμε τις συγκρούσεις κατά τη πρόσβαση στη μνήμη. Μάλιστα, στην οριακή περίπτωση όπου $s = m$ το αραιό-m δεν εμφανίζει ποτέ συγκρούσεις, αφού το Πόρισμα 4.2.6 ισχύει ανεξαρτήτως του x. Έτσι,

Πόρισμα 4.2.7. *Η χρήση της αντιστοίχισης Φ επιτρέπει απεριόριστη πρόσβαση άνευ συγκρούσεων σε οποιοδήποτε αραιό-m ορθογώνιο της εικόνας.*

Εκτός από το αραιό-m, ένα δεύτερο σχήμα που προσφέρει απεριόριστη πρόσβαση επάνω στη εικόνα είναι η στήλη. Αφού έχουμε εφαρμόσει σε κάθε κολόνα της αρχικής εικόνας μια απλή γραμμική αντιστοίχιση των διαδοχικών εικονοστοιχείων σε τράπεζες, μπορούμε εύκολα να δείξουμε ότι

Θεώρημα 4.2.8 (Στήλη). *Η χρήση της αντιστοίχισης Φ επιτρέπει απεριόριστη πρόσβαση άνευ συγκρούσεων σε οποιαδήποτε στήλη $m \cdot n$ εικονοστοιχείων.*

Απόδειξη. Ελέγχουμε την ισότητα $\Phi(x_p, y_p) = \Phi(x_q, y_q)$ για όλα τα ζεύγη εικονοστοιχείων (x_p, y_p) και (x_q, y_q) εντός της τυχαίας στήλης A_{xy} (όπως κάναμε και στις αποδείξεις των προηγούμενων θεωρημάτων). Χρησιμοποιώντας σχετικές συντεταγμένες για τα (x_q, y_q) προκύπτει η εξίσωση $\Phi(x_p, y_p) = \Phi(x_p, y_p + j)$, όπου (x_p, y_p) τυχαίο στοιχείο της στήλης και $j \in \mathbb{N}_{mn} - \{0\}$. Αν για οποιοδήποτε (x_p, y_p) η εξίσωση αυτή έχει λύση ως προς j, τότε έχουμε σύγκρουση στη μνήμη. Με απλή εφαρμογή του τύπου της Φ στη εξίσωση καταλήγουμε πάντα στην ισοτιμία $j \equiv 0 \pmod{mn}$, η οποία εμφανώς δεν έχει καμμιά λύση όταν $0 < j < mn$. □

Το τελευταίο *βασικό* σχήμα πρόσβασης που απομένει να εξετάσουμε είναι η γραμμή. Πρακτικά, η αντιστοίχιση των εικονοστοιχείων της γραμμής είναι ίδια με αυτήν του ορθογωνίου: εξαιτίας της λόξωσης που εφαρμόσαμε κατά λωρίδες πάχους m της εικόνας, οι n διαδοχικές m-άδες που συναντάμε μέσα σε μια τυχαία γραμμή (x, y) είναι ίδιες με τις n διαδοχικές γραμμές του ορθογωνίου (x, y). Έτσι, οι δυνατότητες πρόσβασης της Φ είναι ίδιες για το ορθογώνιο και για τη γραμμή

Θεώρημα 4.2.9 (Γραμμή). *Η χρήση της αντιστοίχισης Φ επιτρέπει πρόσβαση σε οποιαδήποτε γραμμή $m \cdot n$ εικονοστοιχείων εμφανίζοντας το πολύ μια σύγκρουση. Αν εμφανιστεί σύγκρουση, τότε αυτή θα είναι στο πλέον δεξιό στοιχείο του συνόλου-αίτηση.*

Απόδειξη. Εργαζόμαστε όπως και στην απόδειξη του Θεωρήματος 4.2.8, με τη διαφορά ότι εδώ ελέγχουμε την εξίσωση $\Phi(x_p, y_p) = \Phi(x_p + i', y_p)$ με $i' \in \mathbb{N}_{mn}$. Εφαρμόζοντας το Φ προκύπτει η ισοτιμία

$$i' \cdot n + \left\lfloor \frac{x_p + i'}{m} \right\rfloor - \left\lfloor \frac{x_p}{m} \right\rfloor \equiv 0 \pmod{mn}$$

Αντικαθιστούμε τη μοναδική μεταβλητή i' με δυο ανεξάρτητες μεταβλητές, τις $j \in \mathbb{N}_n$ και $i \in \mathbb{N}_m$, ως $i' = j \cdot m + i$ (είναι εύκολο να δείξουμε ότι αυτή η

απεικόνιση είναι αμφιρριπτική, κι άρα η καταγραφή των εικονοστοιχείων μέσω των i και j είναι ισοδύναμη με την καταγραφή τους μέσω της i'). Από την αντικατάσταση προκύπτει η ισοδύναμη εξίσωση

$$i \cdot n + j + \left\lfloor \frac{x_p + i}{m} \right\rfloor - \left\lfloor \frac{x_p}{m} \right\rfloor \equiv 0 \pmod{mn}$$

η οποία είναι η αυτή που μελετήσαμε στο Λήμμα 4.2.1 για $s = 1$. Δηλαδή, μέσω της αντικατάστασης μεταβλητών επιστρέψαμε στην απόδειξη του Θεωρήματος 4.2.3. Επαναλαμβάνουμε εδώ το συμπέρασμα ότι το μοναδικό ζεύγος εικονοστοιχείων που ενδέχεται να έχουν κοινό Φ είναι το πρώτο στοιχείο (εκείνο που συμπίπτει με την θέση (x, y) του συνόλου-αίτηση) και το στοιχείο που φέρει σχετικές συντεταγμένες ως προς αυτό τις $(i, j) = (m - 1, n - 1)$. Εδώ, το συγκεκριμένο (i, j) αντιστοιχεί στο $i' = mn - 1$, το οποίο είναι το τελευταίο εικονοστοιχείο της γραμμής (το πλέον δεξιό). □

Φυσικά, στην παραπάνω απόδειξη, αντί του Λήμματος 4.2.1, μπορούμε να χρησιμοποιήσουμε το Λήμμα 4.2.2 όταν η γραμμή ξεκινά στο αριστερό άκρο μιας κατακόρυφης λωρίδας και να δείξουμε ότι

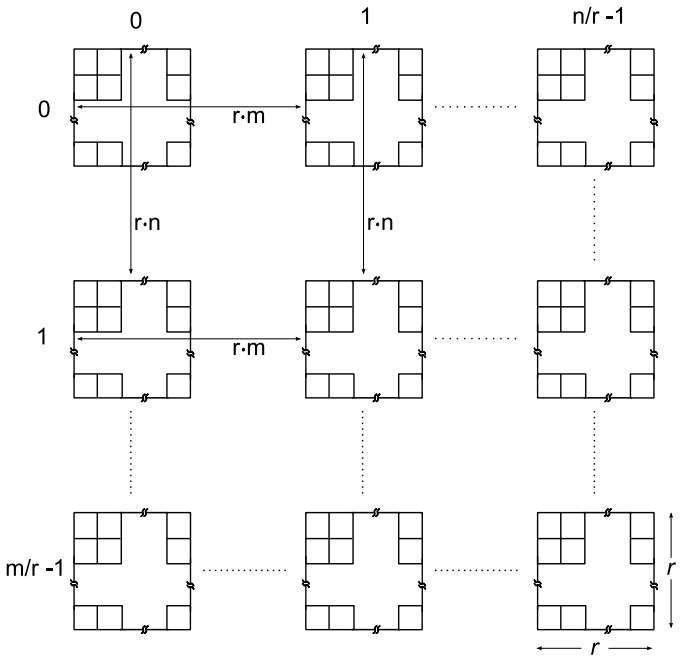
Πόρισμα 4.2.10. *Η χρήση της αντιστοίχισης* Φ *επιτρέπει πρόσβαση άνευ συγκρούσεων σε οποιαδήποτε γραμμή* $m \cdot n$ *εικονοστοιχείων που έχει προέλευση* (x, y) *με* $x \bmod m = 0$.

4.2.3 Πρόσβαση με βοηθητικά σχήματα

Η αποδεικτική μέθοδος που περιγράψαμε παραπάνω βρίσκει εφαρμογή και κατά την εδραίωση δυνατοτήτων πρόσβασης στη μνήμη μέσω πιο περίπλοκων ή ασυνήθιστων σχημάτων, πέραν των βασικών. Η παρούσα εργασία αρκείται στην εξέταση ενός ιδιαίτερου τέτοιου *βοηθητικού* σχήματος που παίζει σημαντικό ρόλο στη διόρθωση ενδεχόμενων συγκρούσεων (όπως αυτή θα παρουσιαστεί στο επόμενο κεφάλαιο). Το σχήμα αυτό το ονομάζουμε *πολυ-τεράγωνο* εξαιτίας της χαρακτηριστικής μορφής που παίρνει στη γενική περίπτωση.

Το πολυ-τετράγωνο

Ένα πολυτετράγωνο-r αποτελείται από πολλαπλές τετράγωνες περιοχές $r \times r$ εικονοστοιχείων η κάθε μία, όπως αυτές απεικονίζονται στην Εικόνα 4.3. Ο ακέραιος r διαιρεί τα m και n ώστε το πολυτετράγωνο-r να αποτελείται από



Εικόνα 4.3: Βοηθητικό σχήμα πρόσβασης: Πολυ-τετράγωνο

$\frac{n}{r} \times \frac{m}{r}$ επί μέρους τετράγωνα και συνολικά να περιέχει $n \times m$ εικονοστοιχεία. Τα τετράγωνα είναι διασκορπισμένα πάνω στην εικόνα σε τακτά διαστήματα μεταξύ τους έτσι, ώστε να σχηματίζουν ένα κανονικό πεπερασμένο πλέγμα με ανύσματα βάσης $\hat{\alpha}_x = (r \cdot m, 0)$ και $\hat{\alpha}_y = (0, r \cdot n)$. Τονίζουμε εδώ το διαφορετικό προσανατολισμό του πολυτετραγώνου, το οποίο αποτελείται από $n \times m$ εικονοστοιχεία, κι όχι $m \times n$ όπως τα βασικά ορθογώνια (αραιά ή μη).

Παρόμοια με το αραιό-s, το πολυτετράγωνο-r προσφέρει πρόσβαση άνευ συγκρούσεων σε πολλές περιοχές της εικόνας. Συγκεκριμένα, όταν η προέλευσή του βρίσκεται αρκετά μακρυά από το δεξί άκρο μιας οποιασδήποτε κατακόρυφης λωρίδας πάχους m της διαιρεμένης εικόνας, έχουμε ότι

Θεώρημα 4.2.11 (Πολυτετράγωνο). *Η χρήση της αντιστοίχισης* Φ *επιτρέπει πρόσβαση άνευ συγκρούσεων σε οποιοδήποτε πολυτετράγωνο-*r *προέλευσης* (x, y) *με* $x \bmod m \leq m - r$.

Απόδειξη. Τα εικονοστοιχεία του τυχαίου πολυτετραγώνου-r με προέλευση (x, y) μπορούν να καταγραφούν με σχετικές συντεταγμένες $(x + i', y + j')$, όπου $i' = \lfloor \frac{i}{r} \rfloor r \cdot m + (i) \bmod (r)$ και $j' = \lfloor \frac{j}{r} \rfloor r \cdot n + (j) \bmod (r)$, με $i \in \mathbb{N}_n$

και $j \in \mathbb{N}_m$. Για να δείξουμε ότι τα εικονοστοιχεία αυτά αποθηκεύονται σε ξεχωριστές τράπεζες όταν $x \bmod m \leq m - r$, δείχνουμε ότι η απεικόνιση $\Phi'(i,j) = \Phi(x+i', y+j')$: $\mathbb{N}_n \times \mathbb{N}_m \to \mathbb{N}_{nm}$ είναι αμφιρριπτική (αντιμετωπίζουμε τα x και y ως σταθερές κατά την απεικόνιση). Αντικαθιστώντας τα i' και j' με τα $i' = i \cdot m - (i) \bmod (r) \cdot (m-1)$ και $j' = j \cdot n - (j) \bmod (r) \cdot (n-1)$ στην έκφραση του Φ, παίρνουμε (με χρήση απλών ιδιοτήτων και βασικών αποτελεσμάτων της θεωρίας αριθμών [65], [66]):

$$\begin{aligned}
&\Phi(x+i',\ y+j') \ = \\
&\Big(\Xi(i,j) + y + xn + \Big\lfloor \tfrac{x+i \bmod (r)}{m} \Big\rfloor\Big) \bmod (mn) \ = \\
&\Big(\Xi(i,j) + \Big\lfloor \tfrac{x \bmod (m)+i \bmod (r)}{m} \Big\rfloor + \text{const}\Big) \bmod (mn)
\end{aligned}$$

όπου

$$\Xi(i,j) = \big(i \bmod (r) + \lfloor \tfrac{j}{r} \rfloor\, r\big) \cdot n + \big(j \bmod (r) + \lfloor \tfrac{i}{r} \rfloor\, r\big)$$

Αφού υποθέσαμε ότι $x \bmod (m) \leq m - r$, έχουμε $\lfloor \frac{x \bmod (m)+i \bmod (r)}{m} \rfloor = 0$. Άρα, για να ολοκληρώσουμε την απόδειξη αρκεί να δείξουμε ότι η απεικόνιση $\Xi(i,j)$: $\mathbb{N}_n \times \mathbb{N}_m \to \mathbb{N}_{nm}$ είναι αμφιρριπτική. Χρησιμοποιούμε ένα παρόμοιο επιχείρημα με εκείνο που είδαμε στην απόδειξη του Λήμματος 4.2.1: παρατηρούμε ότι η $\Xi(i,j)$ δίνει ένα πεπερασμένο σύστημα αρίθμησης της απλής μορφής

$$(\alpha_1 \cdot \beta_0 + \alpha_0) \quad , \quad 0 \leq \alpha_0 \leq \beta_0 - 1$$

με $\alpha_1 = \big(i \bmod (r) + \lfloor \frac{j}{r} \rfloor r\big)$ και $\alpha_0 = \big(j \bmod (r) + \lfloor \frac{i}{r} \rfloor r\big)$. Περαιτέρω, παρατηρούμε ότι κάθε ένα από τα α_1 και α_0 είναι, επίσης, ένα πεπερασμένο σύστημα αρίθμησης της ανωτέρω μορφής. Επομένως, μπορούμε εύκολα να δείξουμε ότι κάθε ζεύγος (i,j) οδηγεί σε μοναδική αναπαράσταση $\alpha_1\alpha_0$, δηλαδή σε μοναδικό $\Xi(i,j)$. Επίσης, βλέπουμε ότι $0 \leq \Xi(i,j) \leq mn-1$. Άρα η $\Xi(i,j)$ είναι 1-1 κι επί, όπως είναι η $\Phi'(i,j)$ για $x \bmod (m) \leq m - r$. Συνεπώς, η Φ δεν αντιστοιχεί ποτέ δυο στοιχεία του πολυτετραγώνου που μελετάμε στην ίδια τράπεζα. □

Παρατηρούμε ότι, όσο μικρότερη είναι η παράμετρος r τόσο μεγαλύτερη είναι η περιοχή της εικόνας στην οποία το πολυτετράγωνο-r επιτρέπει πρόσβαση άνευ συγκρούσεων. Δηλαδή, όσο μικρότερο το r, τόσο μικρότερη η 'πιθανότητα' να εμφανιστεί σύγκρουση κατά την πρόσβαση στη μνήμη μέσω του πολυτετραγώνου. Μάλιστα, η πιθανότητα συγκρούσεων μηδενίζεται όταν $r = 1$ και η Φ προσφέρει ακόμα ένα σχήμα με απεριόριστη πρόσβαση (εκτός της στήλης και

του αραιού-m): το πολυτετράγωνο-1, ή αλλιώς, το αραιό-m-n που υποδειγματοληπτεί μια περιοχή $m \cdot n \times m \cdot n$ της εικόνας (διαφορετικού προσανατολισμού από το βασικό αραιό-m). Στην περίπτωση $r = 1$ ισχύει πάντα η συνθήκη $x \bmod m \leq m - 1$ του θεωρήματος 4.2.11, επομένως

Πόρισμα 4.2.12. *Η χρήση της αντιστοίχισης Φ επιτρέπει απεριόριστη πρόσβαση άνευ συγκρούσεων σε οποιοδήποτε πολυτετράγωνο-1 της εικόνας.*

4.3 Διευθυνσιοδότηση εντός των τραπεζών

Εκτός από την αντιστοίχιση σε τράπεζες μέσω της $\Phi(x, y)$, τυπικά, η αποθήκευση των εικονοστοιχείων στη μνήμη περιλαμβάνει και την αντιστοίχιση τους σε διευθύνσεις εσωτερικά των τραπεζών μέσω κάποιας $\Delta(x, y) : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. Η εν λόγω αντιστοίχιση Δ, αν και άμεσα εξαρτώμενη από τη Φ, δεν είναι μοναδική. Μπορούμε να σχεδιάσουμε πολλές διαφορετικές συναρτήσεις για εσωτερική διευθυνσιοδότηση (πιο εύκολα απ΄ ότι σχεδιάσαμε τη Φ) και να επιλέξουμε την κατάλληλη συνάρτηση ανάλογα με τις απαιτήσεις της εκάστοτε εφαρμογής. Για μια συνολική πρόταση στο πρόβλημα της παράλληλης μνήμης, συμπληρώνουμε το παρόν κεφάλαιο με την παράθεση κάποιων ενδεικτικών αντιστοιχίσεων $\Delta(x, y)$.

Κατά τη σχεδίαση της $\Delta(x, y)$ στοχεύουμε στην πλήρωση τριων βασικών κριτηρίων, τα οποία συμβάλουν στη βελτιστοποίηση της αρχιτεκτονικής. Τα κριτήρια αυτά αναφέρονται στην ερριπτικότητα, στην επιρριπτικότητα, και στην πολυπλοκότητα των αντιστοιχίσεών μας. Όπως αναφέραμε στην ενότητα 2.3.2, μια αποδοτική οργάνωση βασίζεται στην αποθήκευση ενός μόνο εικονσοτοιχείου ανά διεύθυνση μνήμης εντός των τραπεζών (για την αποφυγή υποεκμετάλλευσης της μνήμης). Συνεπώς, μας ενδιαφέρει ο συνδυασμός των αντιστοιχίσεων $\Phi(x, y)$ και $\Delta(x, y)$ να είναι '1-1', δηλαδή να οδηγεί το πολύ ένα εικονοστοιχείο σε κάθε θέση μνήμης (= τράπεζα και διεύθυνση). Επίσης, μας ενδιαφέρει να είναι 'επί', ώστε να εκμεταλλευόμαστε όλους τους υλικούς πόρους που έχουμε χρησιμοποιήσει για την κατασκευή της μνήμης (αποφυγή δημιουργίας κενών θέσεων). Τέλος, παρόμοια με τη Φ, μας ενδιαφέρει η Δ να εμφανίζει μικρή πολυπλοκότητα ώστε να είναι υλοποιήσιμη με οικονομικά κυκλώματα. Οι τρεις αυτοί στόχοι δε είναι πάντα συμβατοί.

Με πρωταρχικό στόχο την πλήρωση του ερριπτικού κριτηρίου (1-1), επιλέγουμε μια απλή στρατηγική σχεδίασης για τη $\Delta(x, y)$, η οποία αποτελείται από τα εξής δυο βήματα:

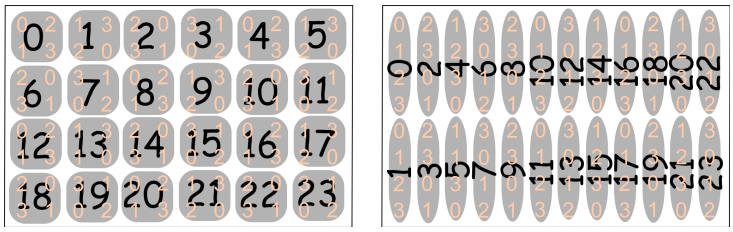
α) κατάτμηση όλης της $W \times H$ εικόνας σε k ξένα μεταξύ τους σύνολα εικονοστοιχείων, U_i, $0 \leq i < k$, έτσι ώστε σε κάθε σύνολο U_i να εμφανίζεται ξεχωριστό $\Phi(x, y)$ ανά στοιχείο $\vec{p}=(x, y)$ του συνόλου. Δηλαδή, $\forall i \neq j, \forall \vec{p} \neq \vec{q} : (U_i \cap U_j = \emptyset) \wedge (\vec{p}, \vec{q} \in U_i \rightarrow \Phi(\vec{p}) \neq \Phi(\vec{q}))$

β) αρίθμηση κάθε συνόλου U_i με έναν μοναδικό αριθμό $\Delta_i \in \mathbb{N}$, ο οποίος χρησιμοποιείται έπειτα ως διεύθυνση για όλα τα εικονοστοιχεία του U_i

Οι περιορισμοί που θέτουμε στα παραπάνω βήματα μας διασφαλίζουν ότι **κάθε** εικονοστοιχείο θα αντιστοιχηθεί σε **ένα** –μόνο– ζεύγος τιμών $\langle \Phi(x, y), \Delta(x, y) \rangle$, καθώς και ότι **κάθε** ζεύγος $\langle \Phi(x, y), \Delta(x, y) \rangle$ χαρακτηρίζει το **πολύ ένα** εικονοστοιχείο. Δηλαδή, οδηγούν στην κατασκευή μιας συνάρτησης '1-1' για την τελική διευθυνσιοδότηση των εικονοστοιχείων στη μνήμη, τη συνάρτηση $\langle \Phi\Delta \rangle$: $W \times H \rightarrow \mathbb{N}_B \times \mathbb{N}_{max\{\Delta_i\}+1}$. Ο πρώτος ισχυρισμός προκύπτει από το ότι η εικόνα χωρίζεται ολόκληρη σε σύνολα, ότι κάθε εικονοστοιχείο (x_p, y_p) τοποθετείται σε ακριβώς ένα από τα σύνολα, έστω U_p, ότι κανένα άλλο στοιχείο του U_p δεν αποθηκεύεται στην ίδια τράπεζα με το (x_p, y_p), και ότι στο U_p αποδίδεται ένα μοναδικό Δ_p. Ο δεύτερος ισχυρισμός προκύπτει από τη μοναδικότητα των Δ_i και από τη μοναδικότητα των $\Phi(x, y)$ μέσα στο σύνολο U_i. Σημειώνουμε ότι από τα βήματα (α) και (β) δεν συνεπάγεται τίποτα σχετικά με την επιρριπτικότητα ή την πολυπλοκότητα της συνάρτησης $\langle \Phi\Delta \rangle$.

Άμεση συνέπεια της προαναφερθείσας στρατηγικής, σύμφωνα με την αρχή του περιστερώνα, είναι ότι κάθε σύνολο U_i έχει πληθικό αριθμό το πολύ B, όπου B ο αριθμός των τραπεζών. Το μέγεθος των συνόλων καθορίζει το πλήθος τους επάνω στην εικόνα, αφού η εικόνα είναι η ένωση όλων των U_i. Άρα, το μέγεθος των U_i καθορίζει και το πλήθος των διευθύνσεων Δ_i, δηλαδή το απαραίτητο βάθος των τραπεζών (κατασκευαστική παράμετρος). Έτσι, σε μεγάλο βαθμό, η ελαχιστοποίηση του υλικού κόστους της μνήμης βασίζεται στην μεγιστοποίηση του μεγέθους κάθε U_i. Εδώ στοχεύουμε, όταν μας επιτρέπουν οι διαστάσεις, σε κατάτμηση εικόνας με $\forall U_i : |U_i| = B$.

Μια πρώτη αποτίμηση των παραπάνω γενικών συμπερασμάτων μας οδηγεί στην ομαδοποίηση γειτονικών εικονοστοιχείων σύμφωνα με κάποιο σχήμα πρόσβασης της Φ που δεν εμφανίζει συγκρούσεις (οπότε περιέχει B στοιχεία με μοναδικό $\Phi(x, y)$ το κάθε ένα). Ως παράδειγμα μπορούμε να χρησιμοποιήσουμε το $m \times n$ ορθογώνιο σε παράθεση (tiling), ή πιο συγκεκριμένα, στις θέσεις (x, y) με $x \bmod m = 0$ και $y \bmod n = 0$. Στην περίπτωση που η οριζόντια και η κάθετη διάσταση του βασικού ορθογωνίου διαιρούν, αντίστοιχα, την οριζόντια και την κάθετη διάσταση της εικόνας, $m|W$, $n|H$, τότε δημιουργούμε ακριβώς $k = WH/mn$ σύνολα U_i πετυχαίνοντας το στόχο $\forall U_i : |U_i| = B$. Η αρίθμηση των συνόλων μπορεί να γίνει, π.χ, από αριστερά προς τα δεξιά και από πάνω



Εικόνα 4.4: Αντιστοίχιση μέσω Φ (m=n=2) και παραδείγματα εσωτερικής διευθυνσιοδότησης (ομαδοποίηση γειτονικών εικονοστοιχείων και αρίθμηση).

προς τα κάτω (raster scan) ξεκινώντας από το $\Delta_0 = 0$ και συνεχίζοντας κατά ένα μέχρι τον ακέραιο $k-1$. Η αρίθμηση αυτή ελαχιστοποιεί τον αριθμό των απαραίτητων διευθύνσεων (βάθος τραπεζών) σε k. Ο συγκεκριμένος τρόπος αντιστοίχισης υπολογίζεται μέσω της

$$\Delta(x,y) = \left\lfloor \frac{x}{m} \right\rfloor + \left\lfloor \frac{y}{n} \right\rfloor \cdot \text{const} \tag{4.5}$$

όπου η const $= \lceil W/m \rceil$ αποτελεί σταθερά της σχεδίασης. Ένα παράδειγμα με διαστάσεις εικόνας που είναι πολλαπλάσιες αυτών του ορθογωνίου φαίνεται αριστερά στην Εικόνα 4.4. Εμφανώς, στην περίπτωση που οι διαστάσεις δεν διαιρούνται ακριβώς, τότε η παραπάνω αντιστοίχιση προκαλεί κενές θέσεις σε ορισμένες από τις τράπεζες της μνήμης.

Η μορφή με την οποία δίνεται η αντιστοίχιση 4.5 φανερώνει την υπολογιστική απαίτηση –στη γενική περίπτωση– για μια πρόσθεση, έναν πολλαπλασιασμό, και δυο διαιρέσεις. Με κριτήριο την περαιτέρω μείωση της πολυπλοκότητας της $\Delta(x,y)$ είναι δυνατόν να αλλάξουμε εντελώς τον τρόπο με τον οποίο αριθμούμε τα σύνολα U_i. Πιο συγκεκριμένα, αντί για την πρόσθεση και τον πολλαπλασιασμό, μπορούμε να χρησιμοποιήσουμε μια απλή συμπαράθεση (concatenation) αριθμών[2] σε κάποιο σύστημα αρίθμησης (εδώ, το δυαδικό) και να κατασκευάσουμε την αντιστοίχιση

$$\Delta(x,y) = \left\lfloor \frac{x}{m} \right\rfloor \,\&\, \left\lfloor \frac{y}{n} \right\rfloor \tag{4.6}$$

η οποία απαιτεί μόνο δυο διαιρέσεις. Φυσικά, όταν εργαζόμαστε στο δυαδικό σύστημα και οι αριθμοί m και n είναι δυνάμεις του 2, τότε οι διαιρέσεις ανάγονται απλά στην επιλογή δυαδικών ψηφίων, οπότε η πολυπλοκότητα υπολογισμού της

[2] $\underline{\&}$: συνένωση αριθμητικών ψηφίων, π.χ., για τους τριψήφιους α,β: $\alpha\&\beta = \alpha_2\alpha_1\alpha_0\beta_2\beta_1\beta_0$

συνάρτησης 4.6 γίνεται μηδαμινή (παρομοίως, η 4.5 μπορεί, κατά περίπτωση, να υλοποιηθεί με μόνο μια πρόσθεση). Το μειονέκτημα της 4.6 είναι τα μεγάλα κενά που ενδέχεται να δημιουργηθούν στη μνήμη όταν $\frac{H}{n} \neq 2^r$, $r \in \mathbb{N}$. Ανάλογα με τις διαστάσεις, είναι δυνατόν να περιορίσουμε το πρόβλημα των κενών παραλλάσσοντας ελαφρώς την 4.6. Για παράδειγμα, μπορούμε να χρησιμοποιήσουμε την μετάθεση $\Delta(x, y) = \lfloor \frac{y}{n} \rfloor \& \lfloor \frac{x}{m} \rfloor$ όταν $\frac{W}{m} = 2^r$, $r \in \mathbb{N}$, ή ακόμα και να βασιστούμε σε διαφορετικό σχήμα ομαδοποίησης προκειμένου να πετύχουμε πιο αποδοτική κάλυψη της εικόνας: αντικαθιστώντας, π.χ., το ορθογώνιο με τη στήλη προκύπτει η $\Delta(x, y) = x \& \lfloor \frac{y}{B} \rfloor$, ένα παράδειγμα της οποίας φαίνεται δεξιά στην Εικόνα 4.4 (το οποίο υλοποιείται χωρίς πράξεις, σε αντίθεση με το παράδειγμα δεξιά, όπου απαιτείται μια πρόσθεση). Οι συγκρίσεις μεταξύ των αντιστοιχίσεων 4.5, 4.6, και των διαφόρων παραλλαγών τους, φανερώνουν τα πλεονεκτήματα/μειονεκτήματα της κάθε υλοποίησης και προσδιορίζουν στην πράξη την κατάλληλη επιλογή για την εκάστοτε εφαρμογή.

Κεφάλαιο 5

Εκμετάλλευση της Συσχέτισης των Αιτήσεων Μνήμης

Οι δυνατότητες πρόσβασης της Φ που αποδείξαμε στο προηγούμενο κεφάλαιο αφορούν στο γενικό πρόβλημα που μελετάται έως σήμερα στη βιβλιογραφία, στο οποίο δε γίνεται καμιά υπόθεση για τη σειρά με την οποία πραγματοποιούνται οι αιτήσεις στη μνήμη. Όπως όμως προαναφέρθηκε, κι όπως θα γίνει ακόμα πιο φανερό μέσα από πρακτικά παραδείγματα στο κεφάλαιο που ακολουθεί, οι συνηθισμένοι αλγόριθμοι δεν εκτελούν εντελώς τυχαίες αιτήσεις στη μνήμη. Αντίθετα, οι διαδοχικές αιτήσεις σχετίζονται μεταξύ τους. Μάλιστα, σε πολλές περιπτώσεις, η σειρά των αιτήσεων (θέσεις και σχήματα) επάνω στην εικόνα είναι απόλυτα τακτική/κανονική και προβλέψιμη. Στο παρόν κεφάλαιο δεχόμαστε μια πολύ γενική υπόθεση για τη συσχέτιση των αιτήσεων και δείχνουμε ότι η προτεινόμενη οργάνωση Φ μπορεί να την εκμεταλλευτεί με επιτυχία προκειμένου να διορθώσει εξαιρετικά γρήγορα τις αναπόφευκτες συγκρούσεις που εξετάσαμε προηγουμένως.

5.1 Περιγραφή συσχετίσεων κι εξειδίκευση του προβλήματος

Η υπόθεσή μας ξεκινά από την γενική παραδοχή ότι οι αλγόριθμοι αιτούνται εικονοστοιχεία κατά συνεκτικές χωρικά περιοχές. Επεκτείνουμε τη παραδοχή αυτή πέρα από το αυτόνομο σύνολο-αίτηση του αλγορίθμου (βλ. ενότητα 1.2), σε ακολουθίες πολλαπλών, διαδοχικών, συνόλων-αιτήσεων. Η προτεινόμενη επέκταση αφορά σε ένα πολύ γενικό μοτίβο χωρικής συνοχής και στηρίζεται

στην εξής παρατήρηση για τις εφαρμογές γραφικών: οι αλγόριθμοι που χρησιμοποιούνται, συνήθως, επεξεργάζονται την εικόνα καλύπτοντας διαδοχικές μεγάλες τετράγωνες περιοχές $K \times K$ εικονοστοιχείων, $K \in \mathbb{N}$. Δηλαδή, αιτούνται διαδοχικά, συνεκτικά, μικρά σύνολα εικονοστοιχείων τα οποία συνθέτουν τελικά ένα μεγαλύτερο τετράγωνο $K \times K$ εικονοστοιχείων. Έπειτα, η διαδικασία συνεχίζεται με κάποια διαφορετική $K \times K$ περιοχή, κ.ο.κ. Παραδείγματα αυτής της συμπεριφοράς μπορεί κανείς να συναντήσει σε πολλές εσωτερικές λειτουργίες της συμπίεσης εικονορροών, αφού η επεξεργασία βασίζεται στην κατάτμηση της εικόνας σε τετράγωνα πλακίδια. Σε άλλες περιπτώσεις, ακόμα κι όταν ο αλγόριθμος δεν ακολουθεί αυστηρά το παραπάνω τετράγωνο μοτίβο, είναι δυνατόν να μεταβάλουμε τη σειρά των αιτήσεών του ώστε να εμφανίζουν την εν λόγω συσχέτιση, χωρίς να επηρεαστεί το τελικό αποτέλεσμα της διαδικασίας. Τέτοια παραδείγματα συναντάμε στα ψηφιακά φίλτρα, όπου η συνέλιξη του μικρού πυρήνα του φίλτρου με τις αντίστοιχες περιοχές εικονοστοιχείων μπορεί να εκτελεστεί ακολουθώντας πολλές διαφορετικές διαδρομές επάνω στην εικόνα.

Τυπικά, η νέα υπόθεση είναι ειδίκευση της κλασσικής. Όμως, συνεχίζει να χαρακτηρίζει την πλειοψηφία των εφαρμογών γραφικών που χρησιμοποιούνται στον πραγματικό κόσμο. Προς την κατεύθυνση αυτή, γενικεύουμε την υπόθεση των συσχετισμένων αιτήσεων θεωρώντας ότι αυτές μπορούν να αποτελούνται από οποιοδήποτε βασικό σχήμα πρόσβασης ενός κύκλου και ότι η $K \times K$ τετράγωνη περιοχή που καλύπτουν μπορεί να έχει οποιαδήποτε προέλευση επάνω στην εικόνα. Η πρώτη γενίκευση στοχεύει στη διατήρηση της ποικιλίας των σχημάτων που πρέπει να διαθέτει μια καλή οργάνωση μνήμης, ενώ η δεύτερη αντικατοπτρίζει την απεριόριστη πρόσβαση που απαιτούν αρκετές εφαρμογές. Επίσης, τονίζουμε ότι δεν υιοθετούμε κάποιον περιορισμό σχετικά με τη διαδρομή που ακολουθούν οι αιτήσεις κατά τη σταδιακή κάλυψη της τετράγωνης περιοχής (υποθέτουμε τυχαία ακολουθία θέσεων εντός τετράγωνης περιοχής).

Το πρόβλημα που μελετάμε μετατρέπεται πλέον στην οργάνωση μιας μνήμης που να προσφέρει πρόσβαση σε τυχαία $K \times K$ τετράγωνη περιοχή της εικόνας μέσω των βασικών σχημάτων {γραμμή,στήλη,ορθογώνιο,αραιό-s}. Διευκρινίζουμε εδώ ότι η απαίτηση για πρόσβαση άνευ συγκρούσεων στην τυχαία περιοχή $K \times K$ συνεπάγεται απεριόριστη πρόσβαση άνευ συγκρούσεων κατά {γραμμή,στήλη,ορθογώνιο} κι επομένως προϋποθέτει αριθμό τραπεζών μεγαλύτερο από το εμβαδόν των σχημάτων[1]. Η παρούσα εργασία επικεντρώνεται

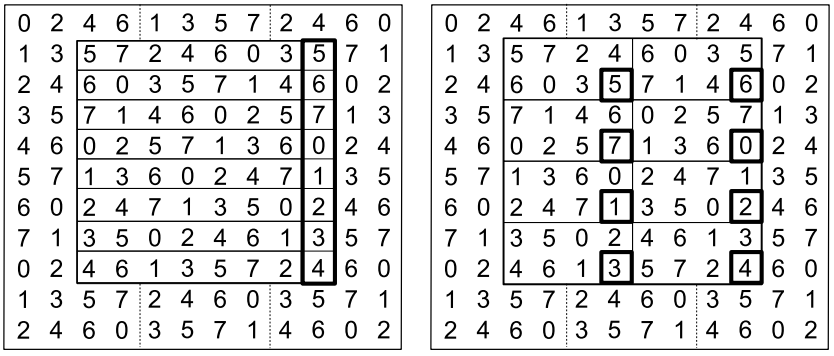
[1]υπό τη συνθήκη αυτή, το νέο πρόβλημα που ορίσαμε είναι απλώς μια ειδική περίπτωση του προβλήματος πρόσβασης άνευ συγκρούσεων που έχει ήδη μελετηθεί στη βιβλιογραφία, κι επομένως, δεν εμφανίζει κάποιο νέο ενδιαφέρον.

στην περίπτωση που χρησιμοποιούμε τον ιδανικό αριθμό τραπεζών, $B = E$, για να αποκτήσουμε πρόσβαση στην περιοχή $K \times K$ σε όσο το δυνατόν λιγότερο χρόνο: συγκεκριμένα, εκτελώντας το πολύ μια επιπρόσθετη αίτηση σε σχέση με αυτές που θα χρειαζόμασταν αν διαθέταμε πρόσβαση άνευ συγκρούσεων[2]. Παρακάτω, θα δείξουμε ότι η προτεινόμενη οργάνωση με χρήση της Φ δίνει μια πολύ αποδοτική λύση στο πρόβλημα αυτό όταν $K = B = m \cdot n$.

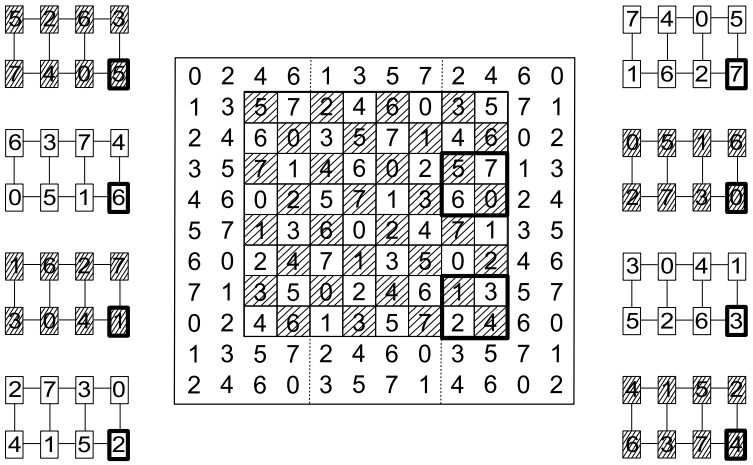
Καλούμε μια $mn \times mn$ τετράγωνη περιοχή της εικόνας ως '*Μακροτετράγωνο*' προέλευσης (x, y), όπου (x, y) η θέση του επάνω-αριστερά εικονοστοιχείου της περιοχής (ίδιος συμβολισμός με αυτόν που χρησιμοποιούμε για την προέλευση ενός συνόλου-αίτησης). Εφόσον μελετάμε απεριόριστη πρόσβαση Μακροτετραγώνων, τα x και y μπορούν να είναι οποιοιδήποτε ακέραιοι αριθμοί. Καλούμε ως '*σάρωση*' μια ακολουθία αιτήσεων που στόχο έχουν να καλύψουν σταδιακά την επιφάνεια ενός ολόκληρου (x, y) Μακροτετραγώνου. Για τη σάρωση ενός Μακροτετραγώνου μπορούμε να χρησιμοποιήσουμε οποιοδήποτε βασικό σχήμα πρόσβασης. Αν κατά τη διάρκεια της σάρωσης καλέσουμε κάποιο σύνολο-αίτηση το οποίο εμφανίσει σύγκρουση, θα αναφερόμαστε σε αυτή ως '*τοπική σύγκρουση*' του Μακροτετραγώνου (x, y). Σε κάθε τοπική σύγκρουση αντιστοιχεί ένα εικονοστοιχείο το οποίο δεν καταφέραμε να ανακτήσουμε (το τελευταίο, κάτω-δεξιά, στοιχείο του συνόλου-αίτηση), καθώς και μια τράπεζα μνήμης που παρέμεινε ανενεργή κατά τη διάρκεια του συγκεκριμένου κύκλου ρολογιού (από τα B δεδομένα, ανακτούμε μόνο $B - 1$). Καλούμε το εικονοστοιχείο που χάνεται ως '*ελλείπον*' και την τράπεζα που δεν χρησιμοποιείται ως '*αδρανή*'. Το ελλείπον εικονοστοιχείο και η αδρανής τράπεζα χαρακτηρίζουν ένα συγκεκριμένο σύνολο-αίτηση που προκαλεί τοπική σύγκρουση και προσδιορίζονται πλήρως από το σχήμα και τη θέση που αυτό φέρει επάνω στην εικόνα. Οι παραπάνω έννοιες γίνονται ευκολότερα αντιληπτές μέσα από το ακόλουθο παράδειγμα.

Έστω οργάνωση μνήμης με 8 τράπεζες κι έστω τυχαίο Μακροτετράγωνο 8×8 εικονοστοιχείων επάνω στην αποθηκευμένη εικόνα. Θεωρούμε εφαρμογή στην οποία ο αλγόριθμος πρέπει να σαρώσει πολλές φορές το Μακροτετράγωνο, χρησιμοποιώντας κάθε φορά ένα διαφορετικό σχήμα πρόσβασης (από αυτά που προσφέρει η οργάνωση μνήμης). Στην καλύτερη περίπτωση, ο αλγόριθμος θα χρειαστεί 8 κύκλους ρολογιού για μια πλήρη σάρωση, αφού υπάρχουν 64 εικονοστοιχεία κι ο μέγιστος ρυθμός μεταφοράς που μπορεί να επιτευχθεί είναι

[2]η συγκεκριμένη διατύπωση καθιστά το νέο πρόβλημα μια γενίκευση του προβλήματος που μελετήσαμε στην προηγουμένη ενότητα, στο οποίο αναζητήσαμε πρόσβαση ενός κύκλου κατά {γραμμή,στήλη,ορθογώνιο} με το πολύ μια σύγκρουση ανά σύνολο-αίτηση.



Εικόνα 5.1: Σάρωση Μακροτετραγώνου με συμπαγή σχήματα πρόσβασης. Αριστερά: γραμμές. Δεξιά: ορθογώνια. Σημειώνονται οι τοπικές συγκρούσεις.



Εικόνα 5.2: Σάρωση Μακροτετραγώνου με αραιό ορθογώνιο (αραιό-2). Καταγράφεται κάθε αίτηση ξεχωριστά. Σημειώνονται οι τοπικές συγκρούσεις.

8 εικονοστοιχεία ανά κύκλο. Αυτό θα συμβεί όταν διαλέξουμε ένα σχήμα που προσφέρει πρόσβαση χωρίς συγκρούσεις (είτε λόγω των θέσεων των αιτήσεων, είτε γιατί το σχήμα παρέχει απεριόριστη πρόσβαση). Στην γενική περίπτωση όμως, αφού έχουμε $B = E$, κατά τη σάρωση θα εμφανιστεί ένας αριθμός από τοπικές συγκρούσεις. Το ερώτημα που τίθεται είναι, πόσες είναι οι συνολικές συγκρούσεις και πόσο γρήγορα μπορούν να διορθωθούν, δηλαδή, πόσους κύκλους χρειαζόμαστε για να ανακτήσουμε τα ελλείποντα εικονοστοιχεία. Στην Εικόνα 5.1 απεικονίζεται η προτεινόμενη οργάνωση μνήμης με χρήση της Φ για $m = 4$, $n = 2$. Επίσης σημειώνεται το Μακροτετράγωνο $(2, 1)$ και δυο παραδείγματα σάρωσης: μέσω γραμμών (αριστερά), και μέσω ορθογωνίων 4×2 (δεξιά). Εξ΄ αιτίας της θέσης του Μακροτετραγώνου, τα συμπαγή αυτά σχήματα οδηγούν σε τοπικές συγκρούσεις κατά τη σάρωση. Ξέρουμε, λόγω των ιδιοτήτων της Φ που έχουμε αποδείξει, ότι θα εμφανιστούν το πολύ 8 συγκρούσεις κατά τη σάρωση. Αυτό όμως που φαίνεται τώρα στο παράδειγμα είναι ότι τα ελλείποντα εικονοστοιχεία των δυο σαρώσεων (σημαδεύονται με έντονες γραμμές) είναι αποθηκευμένα σε 8 διαφορετικές τράπεζες. Επομένως, είναι δυνατόν να ανακτηθούν όλα μαζί σε έναν –μόλις– επιπρόσθετο κύκλο. Τονίζουμε πως, η ιδιότητα αυτή δε χαρακτηρίζει τις υπόλοιπες λύσεις της βιβλιογραφίας, με $B = E$, που περιγράψαμε στην ενότητα 2.3.1. Ακόμα και για περίπλοκες αντιστοιχίσεις, όπως τις τέσσερις που απεικονίζονται στις Εικόνες 2.5 και 2.6, παρατηρούμε ότι χρειαζόμαστε τουλάχιστον δυο επιπρόσθετους κύκλους για να διορθώσουμε τις συγκρούσεις που εμφανίζονται κατά τη σάρωση του Μακροτετραγώνου $(1, 1)$ μέσω, π.χ., ορθογωνίων σχημάτων (συγκεκριμένα, χρησιμοποιούμε ορθογώνιο 3×2, ή 2×3, ή 4×2, ή 3×3, αντίστοιχα, για κάθε μια από τις τέσσερις συναρτήσεις-παραδείγματα των Εικόνων 2.5 και 2.6). Στην Εικόνα 5.2 απεικονίζεται η σάρωση του ίδιου Μακροτετραγώνου μέσω του αραιού-2. Όπως φαίνεται και για την περίπτωση του αραιού, τα 8 ελλείποντα εικονοστοιχεία μπορούν να ανακτηθούν όλα μαζί σε μόλις έναν επιπρόσθετο κύκλο. Φυσικά, το ίδιο Μακροτετράγωνο απαιτεί μηδέν επιπρόσθετους κύκλους αν το σαρώσουμε κατά στήλες ή κατά πολυτετράγωνο-1. Επίσης, αν το Μακροτετράγωνο είχε προέλευση, π.χ., το $(0, 0)$ της εικόνας, τότε θα μπορούσαμε να το σαρώσουμε με οποιοδήποτε σχήμα χωρίς επιπρόσθετους κύκλους. Για να ολοκληρώσουμε το παράδειγμα των Εικόνων 5.1 και 5.2, σημειώνουμε ότι οι αδρανείς τράπεζες κατά τη σάρωση με γραμμές είναι (από πάνω προς τα κάτω) οι {1,2,3,4,5,6,7,0}. Οι ίδιες είναι και κατά τη σάρωση με ορθογώνια (από αριστερά προς δεξιά και από πάνω προς τα κάτω). Για τη σάρωση με αραιό-2 είναι οι {1,3,2,4,5,7,6,0}. Παρατηρούμε ότι, και για τις τρεις περιπτώσεις, κάθε αδρανής τράπεζα της σάρωσης είναι μοναδική. Η συγκεκριμένη ιδιότητα είναι εξαιρετικά

χρήσιμη, όπως θα δείξουμε, όταν η διόρθωση των συγκρούσεων βασίζεται στην επανακατανομή των ελλειπόντων εικονοστοιχείων μέσα στη μνήμη.

Στο παράδειγμα που προηγήθηκε σκιαγραφήθηκε ο στόχος της προτεινόμενης λύσης, καθώς και η σχετική υπεροχή της συγκριτικά με τις προγενέστερες λύσεις της βιβλιογραφίας: διόρθωση των αναπόφευκτων συγκρούσεων σε μνήμες με $B = E$ σε όσο το δυνατόν λιγότερο χρόνο. Στην ενότητα που ακολουθεί εμβαθύνουμε στο στόχο αυτό αποδεικνύοντας ότι

> η προτεινόμενη λύση Φ με $B{=}E{=}\boldsymbol{mn}$ **τράπεζες** προσφέρει απεριόριστη πρόσβαση σε **περιοχή** $\boldsymbol{mn} \times \boldsymbol{mn}$ εικονοστοιχείων μέσω **οποιουδήποτε βασικού σχήματος** πρόσβασης χρησιμοποιώντας το πολύ $\boldsymbol{mn{+}1}$ **κύκλους** ρολογιού.

Σημειώνουμε ότι οι δυνατότητες αυτές δεν απαιτούν τη χρήση ενδιάμεσων, προσωρινών, καταχωρητών (buffer). Η παραπάνω διατύπωση φαίνεται να μεταφέρει με ακρίβεια το επίβαρο κόστος του προβλήματος από το χώρο στο χρόνο[3] αν την αντιπαραβάλουμε με το συμπέρασμα που βγάζει κάποιος μελετώντας τις βέλτιστες λύσεις της βιβλιογραφίας (ενότητα 2.3.1):

> οποιαδήποτε **περιοχή** $\boldsymbol{mn} \times \boldsymbol{mn}$ εικονοστοιχείων είναι δυνατόν να σαρωθεί με **οποιοδήποτε βασικό σχήμα** πρόσβασης σε $\boldsymbol{mn}$ **κύκλους** χρησιμοποιώντας $\boldsymbol{mn{+}1}$ **τράπεζες**.

Με άλλα λόγια, η προτεινόμενη λύση επιλέγει να ξοδέψει έναν παραπάνω κύκλο αντί για μια παραπάνω τράπεζα μνήμης προκειμένου να εξυπηρετήσει τις εφαρμογές γραφικών. Μάλιστα, θα δείξουμε ότι η συγκεκριμένη επιλογή οδηγεί σε πιο αποδοτική λύση (με κριτήριο την εκμετάλλευση του υλικού) συγκριτικά με τις προγενέστερες, ακριβώς γιατί ο επιπρόσθετος κύκλος χρησιμοποιείται υπό προϋποθέσεις κι όχι πάντα, όπως η επιπρόσθετη τράπεζα που ενσωματώνεται υποχρεωτικά/μόνιμα στο υλικό του συστήματος.

5.2 Δυνατότητες σάρωσης Μακροτετραγώνων

Για να προχωρήσουμε στην απόδειξη των νέων δυνατοτήτων που προσφέρει η Φ πρέπει πρώτα να προσδιορίσουμε το ελλείπον εικονοστοιχείο και την αδρανή τράπεζα μιας αίτησης που προκαλεί τοπική σύγκρουση. Από το προηγούμενο

[3] από το κόστος του υλικού στους κύκλους του ρολογιού

κεφάλαιο ξέρουμε ότι το ελλείπον, αν υπάρχει, είναι το τελευταίο στοιχείο του συνόλου (κάτω-δεξιά), το οποίο αποθηκεύεται στην ίδια τράπεζα με το πρώτο στοιχείο, δηλαδή στην τράπεζα $\Phi(x, y)$, όπου (x, y) η θέση της αίτησης. Για τον προσδιορισμό της αδρανούς τράπεζας χρειαζόμαστε μια παρόμοια συνάρτηση του (x, y), καθώς και της αραιής παραμέτρου s (βάζουμε $s = 1$ για τα συμπαγή σχήματα), την οποία δίνουμε στο ακόλουθο Λήμμα

Λήμμα 5.2.1. *Για οποιοδήποτε βασικό σχήμα πρόσβασης, αν ένα σύνολο-αίτηση με προέλευση (x, y) επάνω στην εικόνα οδηγήσει σε τοπική σύγκρουση, τότε η αδρανής του τράπεζα δίνεται από τον τύπο*

$$AT(x, y, s) = \Phi\left(\left\lfloor \frac{x}{m} \right\rfloor m + (x) \bmod (s),\ y\right) \tag{5.1}$$

Απόδειξη. Εφόσον μελετάμε τις περιπτώσεις αιτήσεων που οδηγούν σε τοπική σύγκρουση, υποθέτουμε ότι $\lfloor \frac{x \bmod (m)}{s} \rfloor \geq 1$ και ότι το σχήμα πρόσβασης δεν είναι στήλη ή αραιό-m (βλ. θεωρήματα/πορίσματα της ενότητας 4.2). Επίσης, υποθέτουμε ότι $s = 1$ για τα συμπαγή σχήματα και $s|m$ για τα αραιά.

Για να αποδείξουμε το λήμμα αρκεί να δείξουμε ότι $AT(x, y, s) \neq \Phi(x_q, y_q)$ για κάθε εικονοστοιχείο (x_q, y_q) του σχήματος που εξετάζουμε. Συγκεκριμένα, αρκεί να εκφράσουμε κάθε (x_q, y_q) με σχετικές συντεταγμένες (i, j) ως προς την αρχή (x, y) και να δείξουμε ότι η εξίσωση $\Phi(x + i,\ y + j) = AT(x, y, s)$ δεν έχει λύση (i, j) που να 'δείχνει' εντός του συγκεκριμένου σχήματος.

Τυπικά, η εξίσωση πρέπει να λυθεί για κάθε βασικό σχήμα ξεχωριστά. Όμως, σε όλες τις περιπτώσεις η διαδικασία αυτή καταλήγει στην επίλυση μιας γενικής εξίσωσης ισοτιμίας, την εξής (στην περίπτωση που μελετάμε γραμμές, αρκεί να εφαρμόσουμε την αλλαγή μεταβλητών που χρησιμοποιήσαμε στην απόδειξη του Θεωρήματος 4.2.3):

$$n \cdot \left\lfloor \frac{x \bmod (m)}{s} \right\rfloor \cdot s + \varepsilon_{xi} + \Psi(i, j) \equiv 0 \pmod{mn} \tag{5.2}$$

όπου $i \in \mathbb{N}_m,\ j \in \mathbb{N}_n,\ \varepsilon_{xi}$ το κλάσμα που είδαμε στην απόδειξη του λήμματος 4.2.1, και $\Psi(i, j)$ η αμφιρριπτική απεικόνιση $\mathbb{N}_m \times \mathbb{N}_n \to \mathbb{N}_{mn}$ που ορίσαμε επίσης εκεί.

Για να ολοκληρώσουμε την παρούσα απόδειξη, αρκεί να δείξουμε ότι η 5.2 δεν έχει λύση (i, j) εντός του $\mathbb{N}_m \times \mathbb{N}_n$. Πρώτα απ' όλα, παρατηρούμε ότι ο ακέραιος που αναπαρίσταται στο αριστερό μέλος της εξίσωσης δεν μπορεί να είναι μεγαλύτερος, ή ίσος, από $2mn$ επειδή $\Psi(i, j) < mn,\ \varepsilon_{xi} \leq 1$ και $\lfloor \frac{x \bmod (m)}{s} \rfloor \cdot s \leq m - 1$. Επίσης, δεν μπορεί να είναι μηδέν διότι, εξ' υποθέσεως,

$\lfloor \frac{x \bmod (m)}{s} \rfloor \geq 1$. Επομένως, η εξίσωση 5.2 μπορεί να έχει λύση μόνο όταν το αριστερό της μέλος γίνει ίσο με mn, δηλαδή, όταν $\Psi(i,j) + \varepsilon_{xi} = K$, όπου $K = n \cdot (m - \lfloor \frac{x \bmod (m)}{s} \rfloor s)$. Δείχνουμε ότι πάντα $\Psi(i,j) + \varepsilon_{xi} \neq K$, γιατί όταν $\Psi(i_o, j_o) = K$ έχουμε $\varepsilon_{xi_o} = 1$, και όταν $\Psi(i_u, j_u) = K - 1$ έχουμε $\varepsilon_{xi_u} = 0$ (το ε_{xi} μπορεί να είναι μόνο 0 ή 1). Αφού η $\Psi(i,j)$ είναι αμφιρριπτική, έχουμε ακριβώς ένα ζεύγος (i_o, j_o) κι ένα (i_u, j_u), τα εξής:

$$\begin{aligned} (i_o, j_o) &= \left(m' - \left\lfloor \frac{x \bmod (m)}{s} \right\rfloor,\ 0 \right) \\ (i_u, j_u) &= \left(m - \left\lfloor \frac{x \bmod (m)}{s} \right\rfloor - 1,\ n - 1 \right) \end{aligned}$$

Πράγματι, στην πρώτη περίπτωση παίρνουμε $\varepsilon_{xi_o} = \lfloor \frac{m + (x \bmod m) \bmod (s)}{m} \rfloor = 1$, και στην δεύτερη παίρνουμε $\varepsilon_{xi_u} = \lfloor \frac{m + (x \bmod m) \bmod (s) - s}{m} \rfloor = 0$. □

Έχοντας προσδιορίσει επακριβώς το ελλείπον και την αδρανή τράπεζα για κάθε αίτηση της Φ, μπορούμε να περιγράψουμε συλλογές ελλειπόντων ή αδρανών τραπεζών –που προκύπτουν από μια συγκεκριμένη ακολουθία αιτήσεων– και να αποδείξουμε τη μοναδικότητα κάθε στοιχείου της συλλογής. Αυτή είναι η κεντρική ιδέα στην απόδειξη του επόμενου θεωρήματος. Βέβαια, απαιτείται ακρίβεια και στην διατύπωση-περιγραφή των δυνατών ακολουθιών που πρέπει να μελετήσουμε[4]. Αρχικά μελετάμε απλές σαρώσεις, όπου οι αιτήσεις καλύπτουν ακριβώς ένα Μακροτετράγωνο, χωρίς διπλές αιτήσεις ή παραλείψεις εικονοστοιχείων. Γενικότερα,

Ορισμός. *Έστω K_π τυχαία περιοχή $k{\cdot}m{\cdot}n$ εικονοστοιχείων, $k \in \mathbb{N}$, με τυχαίο σχήμα επάνω στην εικόνα. Καλούμε ως* **τυπική** *τη σάρωση της K_π όταν*

- *Το σχήμα πρόσβασης είναι βασικό και δεν αλλάζει κατά τη σάρωση*
- *Κάθε εικονοστοιχείο της περιοχής K_π ζητείται ακριβώς μια φορά*
- *Δε ζητείται κανένα εικονοστοιχείο εκτός της περιοχής K_π*

Παρατηρούμε ότι μια τυπική σάρωση αποτελείται από ακριβώς k αιτήσεις, ανεξαρτήτως των τοπικών συγκρούσεων (άρα των ελλειπόντων εικονοστοιχείων)

[4] τυπικά, στο συγκεκριμένο θεώρημα, εξετάζουμε μόνο ένα μέρος από όλες τις δυνατές ακολουθίες αιτήσεων που μπορούν να πραγματοποιηθούν στην εικόνα: αυτές που έχουν πρακτικό ενδιαφέρον στις συνηθισμένες εφαρμογές γραφικών. Ο αριθμός όλων των δυνατών ακολουθιών, ανεξαρτήτως συσχετίσεων, είναι εξαιρετικά μεγάλος και αποτρεπτικός για μελέτες σαν αυτή της παρούσας εργασίας. Π.χ., για ασυσχέτιστες mn αιτήσεις σε εικόνα με πλάτος W και ύψος H έχουμε $O\left(\binom{4 \cdot W \cdot H}{m \cdot n} \cdot (mn)!\right)$ δυνατές ακολουθίες.

που θα εμφανιστούν. Επίσης, όταν η K_π είναι ένα Μακροτετράγωνο ($k = mn$), τότε μπορούμε να χρησιμοποιήσουμε οποιοδήποτε συμπαγές σχήμα πρόσβασης για την τυπική του σάρωση, ενώ για να χρησιμοποιήσουμε ένα αραιό-s θα πρέπει να ισχύει ότι $s|m$ και $s|n$. Παραδείγματα τυπικών σαρώσεων Μακροτετραγώνου φαίνονται στις Εικόνες 5.1 και 5.2.

Θεώρημα 5.2.2. *Για οποιαδήποτε τυπική σάρωση ενός τυχαίου Μακροτετραγώνου της εικόνας, αν εμφανιστούν τοπικές συγκρούσεις ισχύουν τα εξής:*

(i) Κάθε ελλείπον εικονοστοιχείο βρίσκεται σε ξεχωριστή τράπεζα.

(ii) Κάθε αδρανής τράπεζα είναι μοναδική.

Απόδειξη. Η σάρωση κατά στήλες και κατά αραιό-m δεν οδηγεί ποτέ σε τοπικές συγκρούσεις, όποια κι αν είναι η θέση του Μακροτετραγώνου πάνω στην εικόνα (Θεώρημα 4.2.8, Πόρισμα 4.2.7). Συνεπώς, θα εξετάσουμε ξεχωριστά τις περιπτώσεις τυπικών σαρώσεων κατά γραμμές, ορθογώνια, και αραιό-s με $s \neq m$. Ξεκινάμε με το πρώτο μέρος του θεωρήματος:

(i) *Ελλείποντα εικονοστοιχεία*

Γραμμές: σύμφωνα με το Θεώρημα 4.2.9, αν η τυπική σάρωση κατά γραμμές δημιουργήσει ένα σύνολο από ελλείποντα εικονοστοιχεία, τότε αυτά θα σχηματίσουν μια στήλη ύψους mn στη δεξιά πλευρά του Μακροτετραγώνου. Σύμφωνα με το Θεώρημα 4.2.8, κάθε εικονοστοιχείο αυτής της στήλης βρίσκεται σε ξεχωριστή τράπεζα, όποια κι αν είναι η θέση της στήλης στην εικόνα.

Ορθογώνια: σύμφωνα με το Θεώρημα 4.2.3, αν η τυπική σάρωση κατά ορθογώνια δημιουργήσει ένα σύνολο από ελλείποντα εικονοστοιχεία, τότε αυτά θα βρίσκονται σε απόσταση (m, n) μεταξύ τους πάνω στην εικόνα. Πιο συγκεκριμένα, θα σχηματίσουν ένα πολυτεράγωνο-1. Συνεπώς, σύμφωνα με το Πόρισμα 4.2.12, κάθε ένα θα βρίσκεται σε ξεχωριστή τράπεζα.

Αραιά-s: Όταν η τυπική σάρωση γίνεται κατά αραιά-s, ο αριθμός των ελλειπόντων στοιχείων δεν περιορίζεται μόνο σε 0 ή σε mn, αλλά μεταβάλλεται ανάλογα με τη θέση (x, y) του Μακροτετραγώνου. Πάντως, σε κάθε περίπτωση, τα εικονοστοιχεία που ενδέχεται να είναι ελλείποντα, είναι εκείνα που βρίσκονται στις κάτω-δεξιά γωνίες των mn συνόλων-αιτήσεων (βλ. Θεώρημα 4.2.5). Παρατηρούμε ότι αυτά τα mn εικονοστοιχεία συνθέτουν ένα πολυτετράγωνο-s. Αν το Θεώρημα 4.2.11 δεν έθετε περιορισμό στη θέση του πολυτετραγώνου, τότε η απόδειξη θα τελείωνε εδώ. Τώρα, όμως, οφείλουμε να διακρίνουμε δυο περιπτώσεις.

Στην πρώτη περίπτωση έχουμε τα πολυτετράγωνα που έχουν θέση (x_{sq}, y_{sq}) με $x_{sq} \bmod m \leq m - s$. Σύμφωνα με το Θεώρημα 4.2.11, τα στοιχεία ενός

τέτοιου πολυτετραγώνου βρίσκονται πάντα σε ξεχωριστές τράπεζες. Δηλαδή, επιβεβαιώνεται το θεώρημα, χωρίς καν να ξέρουμε ποια από αυτά είναι ελλείποντα και ποια όχι. Στην δεύτερη περίπτωση έχουμε τα πολυτετράγωνα που έχουν θέση (x_{sq}, y_{sq}) με $x_{sq} \bmod m > m - s$. Ένα τέτοιο πολυτετράγωνο ξέρουμε ότι περιέχει στοιχεία που μοιράζονται την ίδια τράπεζα. Δεν ξέρουμε όμως αν τα συγκεκριμένα στοιχεία είναι ελλείποντα. Άρα, πρέπει να χωρίσουμε τα στοιχεία του πολυτετραγώνου σε ελλείποντα και μη, ώστε να δείξουμε ότι η 'κοινοκτημοσύνη' τραπεζών δεν αφορά στα ελλείποντα εικονοστοιχεία.

Έστω $(x_{req},\ y_{req})$ η θέση ενός από τα αραιά-s που οδήγησε στο σχηματισμό του πολυτετραγώνου που μελετάμε. Το κάτω-δεξιό στοιχείο του βρίσκεται στη θέση $(x_{pix},\ y_{pix}) = (x_{req} + s \cdot (m-1),\ y_{req} + s \cdot (n-1))$. Από την έκφραση αυτή και από το Πόρισμα 4.2.6 συμπεραίνουμε ότι το εικονοστοιχείο $(x_{pix},\ y_{pix})$ είναι ελλείπον όταν $x_{pix} \bmod m < m - s$. Χωρίζουμε, λοιπόν, τα στοιχεία του πολυτετραγώνου σε δυο υποσύνολα, τα $\mathcal{P}_m$ και $\mathcal{P}_h$. Στο $\mathcal{P}_m$ περιέχονται τα ελλείποντα στοιχεία για τα οποία έχουμε ότι $x_{pix} \bmod m < m - s$, ενώ στο $\mathcal{P}_h$ περιέχονται τα στοιχεία που ανακτώνται κατά τη σάρωση χωρίς σύγκρουση. Για να ολοκληρώσουμε την απόδειξη αρκεί να δείξουμε ότι τα στοιχεία του $\mathcal{P}_m$ αποθηκεύονται σε ξεχωριστές τράπεζες.

Παρατηρούμε ότι, λόγω του περιορισμού $x_{pix} \bmod m < m - s$, το $\mathcal{P}_m$ αποτελείται από τα μικρά τετράγωνα του πολυτετραγώνου-s που μελετάμε, με την απουσία συγκεκριμένων στηλών εικονοστοιχείων: κάθε μικρό τετράγωνο έχει απολέσει την αριστερή πλευρά του με πλάτος $(-x_{sq}) \bmod (m)$, δηλαδή έχει απολέσει τα εικονοστοιχεία για τα οποία ισχύει $x_{pix} \bmod m \geq m - s$. Χωρίς βλάβη, μπορούμε να μελετήσουμε τα στοιχεία του $\mathcal{P}_m$ σα να αποτελούν μέρος ενός άλλου πολυτετραγώνου-s που βρίσκεται δεξιότερα του $(x_{sq},\ y_{sq})$ που μελετάμε. Αυτό το υποτιθέμενο πολυτετράγωνο έχει θέση $(x'_{sq},\ y'_{sq}) = (\lfloor \frac{x_{sq}}{m} \rfloor m + m,\ y_{sq})$, κι άρα, το Θεώρημα 4.2.11 μας διασφαλίζει ότι κανένα ζεύγος των στοιχείων του δεν αποθηκεύεται στην ίδια τράπεζα.

(ii) *Αδρανείς τράπεζες*

Σύμφωνα με το Λήμμα 5.2.1, η αδρανής τράπεζα ενός συνόλου-αίτηση που οδηγεί σε σύγκρουση στη θέση (x_r, y_r) είναι ίδια με την τράπεζα που περιέχει το εικονοστοιχείο $(x_b, y_b) = \big(\lfloor \frac{x_r}{m} \rfloor m + (x_r) \bmod (s),\ y_r\big)$. Στη συνέχεια της απόδειξης αναφερόμαστε στο (x_b, y_b) ως *αντιπρόσωπο* της αδρανούς τράπεζας της θέση (x_r, y_r) και το χρησιμοποιούμε για να προσδιορίσουμε την ίδια την τράπεζα (μονοσήμαντος προσδιορισμός, αφού κάθε δεδομένο βρίσκεται σε μια μόνο τράπεζα). Για να δείξουμε τη μοναδικότητα των αδρανών τραπεζών σε μια τυπική σάρωση, αρκεί να δείξουμε ότι τα εικονοστοιχεία-αντιπρόσωποί τους βρίσκονται σε ξεχωριστές τράπεζες.

Έστω τυχαίο Μακροτετράγωνο με θέση (x, y) πάνω στην εικόνα, το οποίο υπόκειται σε τυπική σάρωση κατά:
Γραμμές: Εκφράζουμε τις θέσεις των mn συνόλων-αιτήσεων με σχετικές συντεταγμένες ως προς το (x, y). Συγκεκριμένα, οι αιτήσεις είναι οι $(x, y + j)$, με $j \in \mathbb{N}_{mn}$. Σύμφωνα με το Λήμμα 5.2.1, οι αδρανείς τράπεζες των αιτήσεων αυτών περιέχουν αντιπροσώπους με θέσεις $(\lfloor \frac{x}{m} \rfloor m,\ y + j)$, με $j \in \mathbb{N}_{mn}$. Τα εικονοστοιχεία αυτά σχηματίζουν μια στήλη με ύψος mn και θέση $(\lfloor \frac{x}{m} \rfloor m,\ y)$. Άρα, σύμφωνα με το Θεώρημα 4.2.8, βρίσκονται σε ξεχωριστές τράπεζες.
Ορθογώνια: Εκφράζουμε τις θέσεις των mn ορθογωνίων ως $(x+i\cdot m,\ y+j\cdot n)$, με $i \in \mathbb{N}_n$ και $j \in \mathbb{N}_m$. Σύμφωνα με το Λήμμα 5.2.1, οι αδρανείς τράπεζες των αιτήσεων αυτών περιέχουν αντιπροσώπους με θέσεις $(\lfloor \frac{x}{m} \rfloor m + i \cdot m,\ y + j \cdot n)$, με $i \in \mathbb{N}_n$ και $j \in \mathbb{N}_m$. Δηλαδή, οι αντιπρόσωποι αυτοι σχηματίζουν ένα πολυτετράγωνο-1 με θέση $(\lfloor \frac{x}{m} \rfloor m,\ y)$ πάνω στην εικόνα. Άρα, σύμφωνα με το Πόρισμα 4.2.12, βρίσκονται σε ξεχωριστές τράπεζες.
Αραιά–s: Εκφράζουμε τις θέσεις των mn αραιών-s ως $(x + i', y + j')$, όπου $i' = \lfloor \frac{i}{s} \rfloor sm + (i) \bmod (s)$ και $j' = \lfloor \frac{j}{s} \rfloor sn + (j) \bmod (s)$, με $i \in \mathbb{N}_n$ και $j \in \mathbb{N}_m$. Όπως αναφέρθηκε και στο πρώτο σκέλος της παρούσας απόδειξης, ο αριθμός των τοπικών συγκρούσεων (κι άρα των αδρανών τραπεζών) εξαρτάται από το (x, y). Εδώ μας ενδιαφέρουν, σύμφωνα με το Πόρισμα 4.2.6, μόνο οι αιτήσεις για τις οποίες ισχύει ότι $\left(x + \lfloor \frac{i}{s} \rfloor sm + (i) \bmod (s)\right) \bmod m \geq s$. Δηλαδή, εκείνες για τις οποίες $(x \bmod m + i \bmod s) \bmod m \geq s$. Άρα εξετάζουμε τα i για τα οποία ισχύει ότι $s \leq x \bmod m + i \bmod s < m$. Χρησιμοποιώντας την ανισότητα αυτή καθώς εφαρμόζουμε την αντιστοίχιση 5.1, συμπεραίνουμε ότι οι αδρανείς τράπεζες που αναζητούμε περιέχουν αντιπροσώπους με θέσεις $\left(\lfloor \frac{x}{m} \rfloor m + \lfloor \frac{i}{s} \rfloor sm + (x + i) \bmod s,\ y + \lfloor \frac{j}{s} \rfloor sn + j \bmod s\right)$, όπου $j \in \mathbb{N}_m$ και $i \in U \subseteq \mathbb{N}_n$. Σημειώνουμε ότι: α) το i δεν καλύπτει υποχρεωτικά όλο το $\mathbb{N}_n$ εξ' αιτίας του περιορισμού που επιβλήθηκε, και β) το σύνολο αντιπροσώπων στο οποίο καταλήξαμε προσδιορίζεται από ένα τύπο που περιγράφει ένα μέρος ενός πολυτετραγώνου-s με θέση $(\lfloor \frac{x}{m} \rfloor m,\ y)$ πάνω στην εικόνα. Το Θεώρημα 4.2.11 μας διασφαλίζει ότι κάθε στοιχείο του συγκεκριμένου συνόλου βρίσκεται σε ξεχωριστή τράπεζα μνήμης. □

Άμεση συνέπεια του Θεωρήματος 5.2.2 είναι ότι οποιοδήποτε Μακροτετράγωνο μπορεί να ανακτηθεί το πολύ σε $mn + 1$ κύκλους. Πιο συγκεκριμένα, χρειαζόμαστε ακριβώς mn κύκλους όταν το Μακροτετράγωνο βρίσκεται στη θέση (x, y) με $x \bmod m = 0$, αφού όποιο βασικό σχήμα κι αν διαλέξουμε δε θα εμφανιστούν τοπικές συγκρούσεις κατά την τυπική του σάρωση. Το ίδιο ισχύει, ανεξαρτήτως θέσης (x, y), όταν διαλέξουμε ως σχήμα πρόσβασης τη στήλη ή

το αραιό-m. Σε όλες τις υπόλοιπες περιπτώσεις μπορούμε να διαβάσουμε το Μακροτετράγωνο από τη μνήμη σε $mn+1$ κύκλους ως εξής: στον πρώτο κύκλο ανακτούμε όλα τα ελλείποντα εικονοστοιχεία που πρόκειται να εμφανιστούν κατά τους mn κύκλους της τυπικής σάρωσης και στη συνέχεια εκτελούμε τη σάρωση διορθώνοντας μια προς μια τις ελλείψεις που εμφανίζονται. Το θεώρημα 5.2.2 εγγυάται ότι τα ελλείποντα στοιχεία (το πολύ mn σε πλήθος, προσδιορίζονται από το σχήμα σάρωσης και τη θέση του Μακροτετραγώνου) είναι αποθηκευμένα σε διαφορετικές τράπεζες κι επομένως ένας κύκλος είναι αρκετός για την ανάκτησή τους. Επίσης, το θεώρημα 5.2.2 εγγυάται ότι οι αδρανείς τράπεζες που πρόκειται να εμφανιστούν κατά την τυπική σάρωση είναι μοναδικές. Το γεγονός αυτό μας επιτρέπει την απομνημόνευση των ελλειπόντων εικονοστοιχείων του πρώτου κύκλου εντός των τραπεζών, κι όχι σε προσωρινούς καταχωρητές (δε χρειαζόμαστε "buffers"). Δηλαδή, στον πρώτο κύκλο μπορούμε να διατάξουμε τις εισόδους/εξόδους των τραπεζών ώστε το ελλείπον εικονοστοιχείο της μετέπειτα αίτησης (x_a, y_a) να αναγνωσθεί από την τράπεζα $\Phi(x_a, y_a)$ στην οποία βρίσκεται μονίμως αποθηκευμένο και να επανεγγραφεί προσωρινά στην τελευταία διεύθυνση της αντίστοιχης αδρανούς τράπεζας $AT(x_a, y_a)$. Ως αποτέλεσμα, όταν πραγματοποιηθεί η αίτηση (x_a, y_a) μπορούμε να ανακτήσουμε το ελλείπον μέσα από την –άλλοτε– αδρανή τράπεζα, ουσιαστικά, καθιστώντας την πρόσβαση άνευ συγκρούσεων. Το θεώρημα 5.2.2 μας διασφαλίζει ότι η προαναφερθείσα προσωρινή επανακατονομή των ελλειπόντων εικονοστοιχείων μπορεί να πραγματοποιηθεί –για όλα μαζί– στον πρώτο από τους $mn+1$ κύκλους ανάγνωσης του Μακροτετραγώνου (λόγω της μοναδικότητας των τραπεζών)[5]. Οι ιδιότητες και η τεχνική διόρθωσης συγκρούσεων που εξηγήσαμε παραπάνω για την ανάγνωση ισχύουν αυτούσιες και κατά την εγγραφή του Μακροτετραγώνου: εκτελούμε μια τυπική σάρωση κι έπειτα, αν υπάρχουν τοπικές συγκρούσεις, γράφουμε σε έναν μοναδικό κύκλο τα εικονοστοιχεία που παραλείψαμε.

[5]η μαθηματική αυτή ιδιότητα είναι ανεξάρτητη της υποκείμενης τεχνολογίας. Μια πιθανή υλοποίηση μέσω τεχνολογίας που διαθέτουμε σήμερα είναι η ευθεία χρήση μνημών διπλής θύρας με προτεραιότητα εγγραφής ("dual port", "read after write" memory) [68], ως εξής: με σωληνωτή αρχιτεκτονική, στον κύκλο c αναρτούμε τις διευθύνσεις των ελλειπόντων εικονοστοιχείων, στον κύκλο $c+1$ αναρτούμε τις διευθύνσεις των εικονοστοιχείων της πρώτης αίτησης, αλλά ταυτόχρονα δρομολογούμε και τις εξόδους των μνημών (τα ελλείποντα που ζητήσαμε προηγουμένως) στις κατάλληλες τράπεζες. Στον κύκλο $c+2$ αναρτούμε τις διευθύνσεις της δεύτερης αίτησης και προωθούμε τα στοιχεία της πρώτης, κ.ο.κ.

5.3 Γενίκευση των αποτελεσμάτων

Τα αποτελέσματα του Θεωρήματος 5.2.2 γενικεύονται μέσω της χαλάρωσης των περιορισμών που θέτει η διατύπωσή του. Ξεκινάμε με τη μελέτη μιας γενίκευσης που σχετίζεται με τον ορισμό της τυπική σάρωσης, και ειδικότερα με την πρώτη απαίτηση του ορισμού για σταθερότητα του σχήματος πρόσβασης. Θα αποδείξουμε ότι οι δυο προτάσεις του 5.2.2 παραμένουν αληθείς ακόμα κι αν το βασικό σχήμα αλλάξει κατά τη σάρωση. Βέβαια, οι αλλαγές στο σχήμα δεν μπορεί να είναι εντελώς αυθαίρετες: οι εσωτερικές αλλαγές πρέπει να σέβονται τις δυο εναπομείνασες απαιτήσεις του ορισμού για πρόσβαση όλων των εικονοστοιχείων του Μακροτετραγώνου –και μόνον αυτών– ακριβώς μια φορά.

Η βασική ιδιότητα της Φ που μας επιτρέπει να σαρώνουμε διαφορετικές υποπεριοχές του Μακροτετραγώνου με διαφορετικά σχήματα πρόσβασης είναι, περίπου, η ανεξαρτησία του συνόλου των ελλειπόντων της υποπεριοχής (και του συνόλου των αδρανών) από το σχήμα που χρησιμοποιήσαμε για να τη σαρώσουμε. Ακριβέστερα,

Πρόταση 5.3.1. *Η τυπική σάρωση μιας τυχαίας περιοχής της εικόνας οδηγεί πάντα στο ίδιο σύνολο αδρανών τραπεζών και στο ίδιο σύνολο τραπεζών ελλειπόντων εικονοστοιχείων, όποιο βασικό σχήμα πρόσβασης κι αν χρησιμοποιηθεί. Στην ειδική περίπτωση που κάποιο σχήμα οδηγεί σε λιγότερες τοπικές συγκρούσεις απ΄ ότι τα υπόλοιπα σχήματα, οδηγεί σε σύνολα τραπεζών που είναι υποσύνολα των αντιστοίχων τους.*

Απόδειξη. Εξετάζουμε τις μικρότερες δυνατές περιοχές για τις οποίες μπορούμε να έχουμε τυπική σάρωση με δυο διαφορετικά σχήματα πρόσβασης (οι μεγαλύτερες περιοχές αποτελούν απλή ένωση αυτών, κι επομένως διατηρούν τις ζητούμενες ιδιότητες). Η τεχνική της απόδειξης είναι παρόμοια με αυτήν που χρησιμοποιήσαμε και στις αποδείξεις που προηγήθηκαν: α) εκφράζουμε τις θέσεις των διαδοχικών αιτήσεων με σχετικές συντεταγμένες ως προς τη αρχή (x, y) της περιοχής που εξετάζουμε, β) υπολογίζουμε στις θέσεις αυτές την αντιστοίχιση Φ (ή την AT), και γ) συγκρίνουμε μεταξύ τους τα σύνολα που προκύπτουν για κάθε διαφορετική σάρωση. Ξεκινάμε με το πρώτο μέρος της πρότασης.

Τράπεζες ελλειπόντων εικονοστοιχείων

Συγκρίνουμε πρώτα τη σάρωση κατά αραιά-s με τη σάρωση κατά γραμμές. Η μικρότερη περιοχή σάρωσης είναι αυτή που προκύπτει καλώντας n διαδοχικά αραιά-s προς τα δεξιά (n γραμμές μήκους mn με κατακόρυφη απόσταση s), δηλαδή ένα σύνολο αιτήσεων $(x + \lfloor \frac{i}{s} \rfloor sm + (i) \bmod s,\ y)$, όπου $i \in \mathbb{N}_n$. Η

ίδια περιοχή καλύπτεται με γραμμές όταν δημιουργήσουμε ένα σύνολο αιτήσεων $(x,\ y + j \cdot s)$, όπου $j \in \mathbb{N}_n$. Σημειώνουμε ότι όταν έχουμε $s = 1$ συγκρίνουμε –ουσιαστικά– τη σάρωση κατά γραμμές με τη σάρωση κατά ορθογώνια.

Εφαρμόζουμε το Φ στα δυο παραπάνω σύνολα αιτήσεων λαμβάνοντας υπόψη ότι όταν έχουμε τοπική σύγκρουση έχουμε $(x \bmod m + i \bmod s) \bmod m \geq s$, άρα $(x \bmod m + i \bmod s) < m$. Από την πρώτη εφαρμογή παίρνουμε το σύνολο των τραπεζών B_α που αποθηκεύουν τα ελλείποντα στοιχεία της σάρωσης κατά αραιό-s, ενώ από τη δεύτερη παίρνουμε το B_γ για τη σάρωση κατά γραμμές:

$$B_\alpha = \left\{ \begin{pmatrix} xn + \lfloor \frac{x}{m} \rfloor + y + & (i \bmod s)n + \lfloor \frac{i}{s} \rfloor\, s \end{pmatrix} \bmod (mn) \mid i \in \mathbb{N}_n \right\}$$
$$B_\gamma = \left\{ \begin{pmatrix} xn + \lfloor \frac{x}{m} \rfloor + y + & js \end{pmatrix} \bmod (mn) \mid j \in \mathbb{N}_n \right\}$$

Για να συγκρίνουμε τα B_α και B_γ συγκρίνουμε τους όρους $(i \bmod s)n + \lfloor \frac{i}{s} \rfloor s$ και js. Υπενθυμίζουμε ότι $s|n$ αφού μελετάμε τυπική σάρωση, κι άρα μπορούμε να συγκρίνουμε τους όρους $(i \bmod s)n' + \lfloor \frac{i}{s} \rfloor$ και j, όπου $n = n's$. Παρατηρούμε ότι η μορφή των δυο όρων παραπέμπει σε πεπερασμένα αριθμητικά συστήματα, κι άρα για κάθε i υπάρχει j ώστε οι δυο όροι να είναι ίσοι. Συνεπώς, $B_\alpha \subseteq B_\gamma$ (δεν είναι αναγκαστικά ίσα, αφού το i ενδέχεται να μην καλύπτει όλο το $\mathbb{N}_n$ λόγω του περιορισμού ύπαρξης τοπικής σύγκρουσης, δηλαδή, η σάρωση κατά αραιά-s ενδέχεται να οδηγεί σε λιγότερες συγκρούσεις απ' ότι οι γραμμές).

Απομένει να συγκρίνουμε τη σάρωση κατά αραιά-s με τη σάρωση κατά ορθογώνια. Η μικρότερη επιτρεπτή περιοχή σάρωσης είναι η συμπαγής περιοχή $ms \times ns$, που προκύπτει από το σύνολο αιτήσεων αραιών-s με θέσεις $(x + i,\ y + j)$, όπου $i, j \in \mathbb{N}_s$. Η ίδια περιοχή καλύπτεται από τα ορθογώνια $(x + km,\ y + ln)$, όπου $k, l \in \mathbb{N}_s$. Εφαρμόζουμε το Φ στα δυο σύνολα και, όπως προηγουμένως, καταλήγουμε στα αντίστοιχα σύνολα τραπεζών ελλειπόντων εικονοστοιχείων των δυο σαρώσεων, B_α και B_o, η διατύπωση των οποίων διαφέρει μόνο στο εξής σημείο: η B_α περιλαμβάνει τον όρο $in + j$, ενώ η B_o τον όρο $ln + k$. Παρατηρούμε ότι για κάθε (i, j) υπάρχει (k, l) τέτοιο που οι δυο όροι να αποκτούν την ίδια τιμή, επομένως κι εδώ έχουμε ότι $B_\alpha \subseteq B_o$.

Αδρανείς τράπεζες

Εργαζόμαστε ακριβώς όπως στο πρώτο σκέλος της απόδειξης, αλλά αντί για το Φ χρησιμοποιούμε την 5.1 που δίνει την αδρανή τράπεζα κάθε αίτησης, δηλαδή την $AT(x_r, y_r, s_r) = \left((x_r \bmod s_r)n + y_r + \lfloor \frac{x_r}{m} \rfloor\right) \bmod (mn)$.

Πρώτα συγκρίνουμε τη σάρωση κατά γραμμές με τη σάρωση κατά αραιά-s (ή κατά ορθογώνια όταν $s = 1$). Χρησιμοποιώντας τις σχετικές συντεταγμένες και τους περιορισμούς που αναφέρθηκαν στο πρώτο σκέλος, υπολογίζουμε τα

σύνολα αδρανών τραπεζών κατά αραιά-s, A_α, και κατά γραμμές[6], A_γ, ως

$$A_\alpha = \left\{ \left(\left\lfloor \frac{x}{m} \right\rfloor + y + \quad ((x+i) \bmod s)n + \left\lfloor \frac{i}{s} \right\rfloor s \right) \bmod (mn) \mid i \in \mathbb{N}_n \right\}$$
$$A_\gamma = \left\{ \left(\left\lfloor \frac{x}{m} \right\rfloor + y + \qquad\qquad js \qquad\qquad \right) \bmod (mn) \mid j \in \mathbb{N}_n \right\}$$

Η σύγκριση των A_α και A_γ ανάγεται τελικά στη σύγκριση των διαφορετικών όρων $((x+i) \bmod s)n + \lfloor \frac{i}{s} \rfloor s$ και js. Με απαλοιφή του κοινού όρου s, όπου $n = n's$, συγκρίνουμε τους $((x+i) \bmod s)n' + \lfloor \frac{i}{s} \rfloor$ και j. Βλέπουμε ότι για οποιοδήποτε i υπάρχει j τέτοιο, ώστε ο δεύτερος όρος να γίνεται ίσος με τον πρώτο (καταλήξαμε πάλι σε πεπερασμένους απαριθμητές). Άρα $A_\alpha \subseteq A_\gamma$ (όχι ίσα, αφού το αραιό-s ενδέχεται να οδηγεί σε λιγότερες συγκρούσεις).

Απομένει να συγκρίνουμε τη σάρωση κατά αραιά-s με τη σάρωση κατά ορθογώνια. Χρησιμοποιούμε την τυχαία περιοχή $ms \times ns$ και τα σύνολα αιτήσεων που περιγράψαμε στο πρώτο σκέλος της απόδειξης. Αντί για το Φ, εφαρμόζουμε το AT στα δυο σύνολα αιτήσεων και προκύπτουν δυο νέα σύνολα, A_α και A_o, η περιγραφή των οποίων διαφέρει μόνο κατά τους όρους $((x+i) \bmod s)n + j$ και $ln + k$, με $i,j,k,l \in \mathbb{N}_s$. Παρατηρούμε ότι για κάθε (i,j) υπάρχει (k,l) ώστε οι δυο όροι να γίνονται ίσοι, άρα έχουμε $A_\alpha \subseteq A_o$. □

Επί της ουσίας, η Πρόταση 5.3.1 μας επιτρέπει να αλλάξουμε σχήμα πρόσβασης σε συγκεκριμένα σημεία της σάρωσης, χωρίς να υπάρχει κίνδυνος να εμφανίσουμε κάποια τράπεζα ελλείποντος εικονοστοιχείου (ή αδρανή) που να ταυτίζεται με κάποια αντίστοιχη που εμφανίσαμε προηγουμένως. Επομένως,

Πόρισμα 5.3.2. *Οι προτάσεις του Θεωρήματος 5.2.2 παραμένουν αληθείς ακόμα κι αν το βασικό σχήμα πρόσβασης αλλάξει κατά τη σύνθετη τυπική σάρωση ενός Μακροτετραγώνου.*

Απόδειξη. Μια σύνθετη τυπική σάρωση Μακροτετραγώνου χαρακτηρίζεται από την αλλαγή του σχήματος πρόσβασης έτσι, ώστε να αποτελεί την ένωση δυο (ή περισσότερων) τυπικών σαρώσεων. Για κάθε τέτοια μερική τυπική σάρωση, το Θεωρήμα 5.2.2 ισχύει. Απομένει να δείξουμε ότι τα σύνολα αδρανών τραπεζών που δημιουργούν οι μερικές αυτές σαρώσεις είναι ξένα μεταξύ τους (έχουν μηδενική τομή ανά δυο). Επίσης, απομένει να δείξουμε ότι και τα σύνολα ελλειπόντων εικονοστοιχείων είναι ξένα μεταξύ τους.

Έστω δυο περιοχές Π_1 και Π_2 του τυχαίου Μακροτετραγώνου, οι οποίες σαρώνονται τυπικά με τα σχήματα F_1 και F_2, αντίστοιχα. Σημειώνουμε πως,

[6] το s που χρησιμοποιούμε για την περιγραφή του συνόλου των γραμμών είναι διαφορετικό από το $s_r = 1$ που χρησιμοποιούμε στον τύπο AT, το οποίο χαρακτηρίζει ένα συμπαγές σχήμα.

ακόμα κι αν αλλάζουμε σχήμα, μελετάμε 'τυπική' σάρωση –συνολικά– για το Μακροτετράγωνο. Με άλλα λόγια, είναι δυνατόν να σαρώσουμε τυπικά την Π_1 με το F_2 και την Π_2 με το F_1. Λόγω της Πρότασης 5.3.1, οι αδρανείς τράπεζες για την Π_1 είναι ίδιες είτε χρησιμοποιούμε το F_1, είτε το F_2 (στη χειρότερη περίπτωση είναι υποσύνολο). Άρα, από το Θεώρημα 5.2.2 συμπεραίνουμε ότι οι αδρανείς τράπεζες της Π_1 είναι εντελώς διαφορετικές από αυτές της Π_2: οι Π_1 και Π_2 αποτελούν κομμάτι μιας τυπικής σάρωσης του Μακροτετραγώνου με χρήση ενός σχήματος, έστω του F_2, κι επομένως δεν υπάρχει αδρανής τράπεζα που να εμφανίζεται ταυτόχρονα στην Π_1 και στην Π_2.

Ακριβώς το ίδιο επιχείρημα χρησιμοποιούμε για να δείξουμε και τη μηδενική τομή των συνόλων ελλειπόντων εικονοστοιχείων των Π_1 και Π_2. □

Μια περαιτέρω γενίκευση των αποτελεσμάτων του Θεωρήματος 5.2.2 αφορά στη συμπεριλήψη περισσοτέρων σχημάτων πρόσβασης κατά τη σάρωση (μαζί με τα βασικά). Ένα παράδειγμα είναι η χρήση του πολυτετραγώνου-1, το οποίο δύναται να χρησιμοποιηθεί οποτεδήποτε κατά την τυπική σάρωση Μακροτετραγώνου (σύνθετη ή μη), χωρίς να δημιουργήσει τοπικές συγκρούσεις (Πόρισμα 4.2.12).

Εκτός από τα παραπάνω αποτελέσματα που σχετίζονται με τη σάρωση Μακροτετραγώνων, οι ιδιότητες της Φ μπορούν να αποδειχτούν χρήσιμες και σε περιπτώσεις όπου η συσχέτιση των αιτήσεων δεν ακολουθεί το τετράγωνο μοτίβο που έχουμε υποθέσει. Γενικότερα, όταν η μοναδικότητα των τραπεζών (ελλειπόντων ή αδρανών) διατηρείται κατά τη σάρωση, μας επιτρέπει τη σπατάλη ενός μόνο κύκλου ανά $mn+1$ αιτήσεις. Ως χαρακτηριστικό παράδειγμα αναφέρουμε την τυπική σάρωση μιας οριζόντιας λωρίδας $m^2n \times n$ εικονοστοιχείων με χρήση γραμμών mn κι ορθογωνίων $m \times n$. Εδώ, είναι τετριμμένο να δείξουμε ότι τα ελλείποντα εικονοστοιχεία της σάρωσης κατά ορθογώνια βρίσκονται ανά δύο σε οριζόντια απόσταση m μεταξύ τους, οπότε, ο προσθετικός παράγοντας $\lfloor \frac{x}{m} \rfloor$ της Φ διασφαλίζει την αποθήκευσή τους σε διαφορετικές τράπεζες. Επίσης, στη σάρωση κατά γραμμές τα ελλείποντα σχηματίζουν μικρές στήλες ύψους n στοιχείων, οι οποίες βρίσκονται σε οριζόντια απόσταση mn μεταξύ τους· δεδομένης της απόστασης mn, ο παράγοντας $\lfloor \frac{x}{m} \rfloor$ της Φ προσδίδει μια γραμμική λόξωση μεταξύ των μικρών στηλών κατά n (όσο και το ύψος τους), δηλαδή, διασφαλίζει ότι τα στοιχεία τους βρίσκονται πάντα σε διαφορετικές τράπεζες. Άρα, για την οριζόντια λωρίδα $m^2n \times n$ καταλήγουμε σε συμπεράσματα ανάλογα με αυτά που είχαμε για το Μακροτετράγωνο. Στα ίδια θα καταλήξουμε κι αν η λωρίδα γίνει κατακόρυφη με $m \times n^2m$ εικονοστοιχεία (τυπική σάρωση κατά στήλες mn και ορθογώνια $m \times n$).

Τέλος, αναφερόμαστε στην περίπτωση όπου δεν ισχύει ο δεύτερος κι ο τρίτος περιορισμός που επιβάλει η τυπική σάρωση, σύμφωνα με τους οποίους πρέπει να συμπεριλάβουμε όλα τα εικονοστοιχεία της περιοχής που σαρώνουμε –και μόνον αυτά– ακριβώς μια φορά. Ο περιορισμός αυτός δεν είναι πάντα απαραίτητη προϋπόθεση για την μοναδικότητα των τραπεζών (ελλειπόντων κι αδρανών) που προσφέρει η Φ. Σημειώνουμε εδώ ένα παράδειγμα από το πεδίο των ψηφιακών φίλτρων εικόνας με χρήση δισδιάστατων πυρήνων: τη σάρωση μιας κατακόρυφης λωρίδας $m \times (mn + n - 1)$ εικονοστοιχείων με mn αιτήσεις του ορθογωνίου $m \times n$ έτσι, ώστε αυτό να σύρεται κατακόρυφα στις θέσεις (x, y), $(x, y + 1)$, ..., $(x, y + mn - 1)$. Στο παράδειγμα αυτό τα δυνητικά ελλείποντα εικονοστοιχεία σχηματίζουν μια συμπαγή στήλη με πληθικό αριθμό mn, την οποία μπορούμε να ανακτήσουμε σε ένα μοναδικό κύκλο, ανεξαρτήτως (x, y).

5.4 Πλεονεκτήματα της προτεινόμενης λύσης

Η οργάνωση παράλληλων μνημών μέσω της αντιστοίχισης Φ προσφέρει ορισμένα βασικά πλεονεκτήματα έναντι των λοιπών λύσεων της βιβλιογραφίας (βλ. ενότητα 2.3.1). Η Φ σχεδιάστηκε με στόχο την υποστήριξη εφαρμογών γραφικών κάνοντας χρήση του ιδανικού αριθμού τραπεζών, $B = E$, για οποιοδήποτε εμβαδόν σχήματος πρόσβασης, $B \in \mathbb{N}$. Ως εκ τούτου, η απουσία περίσσειας τραπεζών σε συνδυασμό με τη δυνατότητα εξειδίκευσης του B σε αριθμούς που εκφράζονται ως δυνάμεις του 2, μας οδηγεί σε λύσεις που υπερτερούν αυτών που βασίζονται σε πρώτους αριθμούς τραπεζών [48] [45], ή στο διπλάσιο από τον επιθυμητό αριθμό [49], ή έστω σε κάποια επιπρόσθετη τράπεζα μνήμης [47] [57][7]. Στις ακόλουθες παραγράφους εντοπίζουμε/συνοψίζουμε τα πλεονεκτήματα της Φ στα εξής σημεία:

[7]Φυσικά, η Φ υπερτερεί και σε σχέση με τις λοιπές λύσεις που θέτουν $B = E$ (όταν πρόκειται για εφαρμογές γραφικών) [56] [46]. Αυτό συμβαίνει γιατί οι συγκεκριμένες λύσεις δεν προσφέρουν απεριόριστη πρόσβαση (δεν έχουν προταθεί για εφαρμογές γραφικών). Επομένως, στην πράξη, η διόρθωση των συγκρούσεων που προκύπτουν σε αυτές τις οργανώσεις μνήμης απαιτεί περισσότερο χρόνο από τον μοναδικό κύκλο της Φ.

α) δυνατότητα σχεδίασης αποδοτικών κυκλωμάτων υποστήριξης της λειτουργίας της μνήμης (αναγνώριση τραπεζών, διευθυνσιοδότηση τραπεζών, και δρομολόγηση δεδομένων),
β) αυξημένη εκμετάλλευση του ζωνικού εύρους της μνήμης[8].

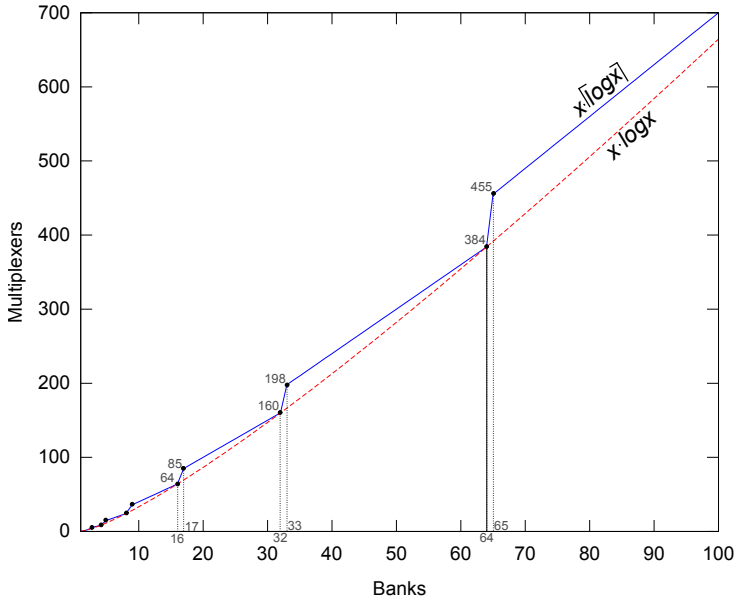
Τα κυκλώματα διευθυνσιοδότησης κι επιλογής τραπεζών μνήμης αποτελούν ένα αναπόσπαστο κομμάτι της αρχιτεκτονικής, καθώς υλοποιούν την μετάφραση της αίτησης $(x, y,$ σχήμα$)$ του αλγορίθμου στις πραγματικές διευθύνσεις μνήμης των αντίστοιχων εικονοστοιχείων. Όπως είναι λογικό, οι πράξεις που απαιτούνται κατά τη μετάφραση είναι άρρηκτα συνδεδεμένες με τον αριθμό B. Ειδικά σε οργανώσεις που χρησιμοποιείται λόξωση, οι πράξεις περιλαμβάνουν ακέραια διαίρεση με το B (modulo και/ή div). Σε όλες αυτές τις περιπτώσεις γίνεται σαφές το πλεονέκτημα που προσφέρει ένας αριθμός δύναμη του 2 σε σχέση με έναν, π.χ., πρώτο: δυνατότητα υπολογισμού με απλή προώθηση διφύων (ή ολίσθηση) κι όχι με διαιρέτες ή άλλες περίπλοκες κατασκευές [69].

Η προτεινόμενη λύση $\Phi(x, y) = (xn + y + \lfloor \frac{x}{m} \rfloor) \bmod (mn)$ επιτρέπει τη χρήση ακριβώς mn τραπεζών, ιδιότητα που αποδεικνύεται εξαιρετικά χρήσιμη στη –συνήθη– περίπτωση που τα m και n είναι δυνάμεις του 2. Η μετάφραση της αίτησης του αλγορίθμου $(x, y,$ σχήμα$)$ περιλαμβάνει τον υπολογισμό της $\Phi(x, y)$ για τον εντοπισμό των τραπεζών που αποθηκεύουν τα ζητούμενα εικονοστοιχεία. Δεδομένου ότι τα m και n είναι σταθερά, ο υπολογισμός της Φ απαιτεί μόνο δυο προσθέσεις μεταξύ των στοιχείων της αίτησης: το y συν δυο συγκεκριμένα μέρη του x, αφού ο πολλαπλασιασμός με n, η διαίρεση με m, και το υπόλοιπο κατά mn υλοποιείται με απλή προώθηση διφύων[9]. Εμφανώς, ο υπολογισμός αυτός είναι σημαντικά οικονομικότερος από αυτόν που απαιτείται όταν έχουμε $B = mn + 1 = 2^k + 1$ ή $B = prime$.

Η μετάφραση της αίτησης $(x, y,$ σχήμα$)$ περιλαμβάνει επίσης και τον υπολογισμό της $\Delta(x, y)$ για τον προσδιορισμό των διευθύνσεων μέσα σε κάθε τράπεζα, έτσι όπως εξηγήθηκε στην ενότητα 4.3. Η $\Delta(x, y)$, όπως και η Φ, εξαρτάται άμεσα από τον αριθμό των τραπεζών (από τα m και n) που χρησιμοποιούμε. Συνεπώς, το γεγονός ότι η προτεινόμενη λύση επιτρέπει τη χρήση δυνάμεων του 2 καθιστά τον υπολογισμό της Δ εξαιρετικά απλό. Ειδικά στην περίπτωση που ταιριάζει στις διαστάσεις της εικόνας η διευθυνσιοδότηση της εξίσωσης 4.6 ή κάποια παραλλαγή της, μπορούμε να υπολογίσουμε τη $\Delta(x, y)$

[8]increased memory bandwtidth utilization

[9]Για xn: προσκόλληση $\log_2 n$ μηδενικών στο x. Για $\lfloor \frac{x}{m} \rfloor$: απόρριψη των $\log_2 m$ διφύων χαμηλής αξίας του x. Για $(z) \bmod (mn)$: επιλογή των $\log_2 mn$ διφύων χαμηλής αξίας του z.



Εικόνα 5.3: Γραφική απεικόνιση του επίβαρου κόστους που εισάγεται στον κυλινδρικό μετατοπιστή από τις λύσεις με επιπρόσθετες τράπεζες όταν $E = 2^k$.

με απλή προώθηση διφύων, χωρίς πράξεις. Σε άλλες περιπτώσεις, αν θέλουμε να αποφύγουμε κενά στη μνήμη, μπορούμε να χρησιμοποιήσουμε μια διευθυνσιοδότηση σαν αυτή της εξίσωσης 4.5, η οποία απαιτεί μόνο πρόσθεση όταν τα m και n είναι δυνάμεις του 2 (ο πολλαπλασιασμός με σταθερά δεν απαιτεί πολλαπλασιαστική μονάδα, ενώ όταν κάποια από τις διαστάσεις της εικόνας είναι επίσης δύναμη του 2, η πράξη υλοποιείται με προώθηση διφύων).

Η προτεινόμενη οργάνωση προσφέρει πλεονεκτήματα και κατά την ανάπτυξη των εσωτερικών κυκλωμάτων δρομολόγησης των δεδομένων. Όπως εξηγήθηκε στην ενότητα 2.3.2, οι εφαρμογές γραφικών απαιτούν την κατάλληλη στοίχιση (μετάθεση) των δεδομένων στην έξοδο της παράλληλης μνήμης έτσι, ώστε αυτά να ακολουθούν κάποια προκαθορισμένη φόρμα (π.χ., raster scan order). Επίσης, η αποθήκευση των δεδομένων απαιτεί την κατάλληλη μετάθεση κατά την είσοδό τους στη μνήμη έτσι, ώστε αυτά να οδηγηθούν στις τράπεζες που τους αντιστοιχούν (εδώ, σύμφωνα με τη Φ). Με άλλα λόγια, μια ολοκληρωμένη αρχιτεκτονική απαιτεί την ύπαρξη μονάδων δρομολόγησης πριν και μετά τις B τράπεζες. Οι μονάδες αυτές εξαρτώνται άμεσα από τη συνάρτηση αντιστοίχισης και σε ορισμένες περιπτώσεις εμφανίζουν σημαντική πολυπλοκότητα και/ή δυσκολία σχεδίασης (π.χ., περίπτωση συναρτήσεων XOR). Προκειμένου

να αποφύγει το αυξημένο κόστος και/ή χρόνο ανάπτυξης των δρομολογητών, η προτεινόμενη Φ βασίζεται στην ισοτροπικότητα που προσφέρει η γραμμική λόξωση (ενότητα 2.3.2). Όπως δείξαμε, η Φ χρησιμοποιεί τη γραμμική λόξωση στο βαθμό που αυτός της επιτρέπει να διατηρεί τις επιθυμητές ιδιότητες: χωρίζει την εικόνα σε λωρίδες διατηρώντας γραμμική λόξωση εντός τους, αλλά και μεταξύ τους (σύνθεση). Με αυτόν τον τρόπο προκύπτει μια σχεδόν ισοτροπική οργάνωση που επιτρέπει τη χρήση ευρέως διαδεδομένων μονάδων μετάθεσης των δεδομένων. Πιο συγκεκριμένα, οι περισσότερες μεταθέσεις που απαιτούνται είναι απλά κυκλικές, οπότε μπορούν να υλοποιηθούν με έναν τετριμμένο 'κυλινδρικό μετατοπιστή' (barrel shifter) [70]. Οι υπόλοιπες απαιτούν κάποια τροποποίηση του ελέγχου των πολυπλεκτών σε ορισμένα επίπεδα του κυλινδρικού μετατοπιστή, ο οποίος συνεχίζει να αποτελεί τη βάση του κυκλώματος δρομολόγησης. Πέρα από τη δυνατότητα χρήσης κυλινδρικών μετατοπιστών, η Φ επιτρέπει και την αποδοτική τους υλοποίηση. Συγκεκριμένα, όταν ο αριθμός B είναι δύναμη του 2, τότε με τη Φ αποφεύγουμε το επίβαρο κόστος δρομολόγησης που εισάγουν οι λύσεις που απαιτούν επιπρόσθετες τράπεζες. Αυτό συμβαίνει, κυρίως, γιατί η προτεινόμενη οργάνωση δεν απαιτεί ένα ολόκληρο επιπρόσθετο επίπεδο στον κυλινδρικό μετατοπιστή προκειμένου να δρομολογήσει την έξοδο της $(2^k{+}1)$-οστής τράπεζας. Φυσικά, η εξοικονόμηση του κόστους γίνεται ακόμα πιο εμφανής όταν έχουμε λύση με πολλαπλές επιπρόσθετες τράπεζες (π.χ., $B =$ πρώτος αριθμός). Η Εικόνα 5.3 προβάλει το κόστος του μετατοπιστή συναρτήσει του πλήθους των εισόδων του, $O(B \cdot \lceil \log_2 B \rceil)$, και τονίζει την απότομη αύξηση του αριθμού των πολυπλεκτών όταν υποχρεωνόμαστε στην δρομολόγηση μιας επιπρόσθετης τράπεζας. Η εξοικονόμηση (σχετική διαφορά) που προσφέρει η προτεινόμενη λύση συγκριτικά με αυτές που φέρουν $B > E$ μεγαλώνει με το E, καθώς και με το $B - E$.

Εκτός από το προαναφερθέν κόστος σχεδίασης και υλοποίησης, μια παράμετρος που επίσης χαρακτηρίζει την αποδοτικότητα των κυκλωμάτων της παράλληλης μνήμης είναι ο βαθμός εκμετάλλευσής τους από τις επεξεργαστικές μονάδες της εφαρμογής (βλ. ενότητα 2.3.2). Γενικότερα, σε επίπεδο εφαρμογής, η βελτιστοποίηση μιας σωληνωτής αρχιτεκτονικής βασίζεται στη μεγιστοποίηση της εκμετάλλευσης των πόρων κατά τη διάρκεια της λειτουργίας: ιδανικά, στοχεύουμε σε 100% εκμετάλλευση του αποδιδόμενου έργου κάθε μικρού ή μεγάλου πόρου. Όταν η γενική αυτή στρατηγική επιμεριστεί στις βασικές μονάδες της αρχιτεκτονικής, όπως στη μνήμη, ένας τρόπος για να ποσοτικοποιήσουμε και να εξετάσουμε το βαθμό εκμετάλλευσης της μονάδας είναι να υπολογίσουμε την εκμετάλλευση του ζωνικού της εύρους. Για το σκοπό αυτό, εδώ, θα υποθέσουμε αίτηση E στοιχείων ανά κύκλο ρολογιού και αριθμό τραπε-

ζών ίσο με B. Δηλαδή, ότι οι επεξεργαστικές μονάδες μπορούν να εκμεταλλευτούν E στοιχεία ανά κύκλο, ενώ η μνήμη μπορεί να διαθέσει (έχει ζωνικό εύρος) B στοιχεία ανά κύκλο. Εμφανώς, όταν $B < E$ εμφανίζεται υπο-εκμετάλλευση των επεξεργαστικών μονάδων (αναμένουν τη συλλογή των E στοιχείων, η οποία απαιτεί περισσότερους από έναν κύκλους). Αντιστοίχως, όταν $B > E$ εμφανίζεται υπο-εκμετάλλευση της μνήμης. Η ιδανική περίπτωση για την οικονομία της αρχιτεκτονικής είναι η ουσιαστική ταύτιση του ζωνικού εύρους της μνήμης με το ζωνικό εύρος των επεξεργαστικών μονάδων, οπότε έχουμε απόδοση κι εκμετάλλευση έργου κατά 100%. Όμως, στις εφαρμογές γραφικών, δεδομένης της απόδειξης $B > E$, διαπιστώνουμε ότι είναι αδύνατο να επιτύχουμε τον ιδανικό λόγο (100%) στη γενική περίπτωση. Οι καλύτερες λύσεις της βιβλιογραφίας (ενότητα 2.3.1) αρκούνται σε $B = E + 1$ κι οδηγούν σε εκμετάλλευση ζωνικού εύρους κατά $\frac{B}{B+1}$. Το ερώτημα που ανακύπτει πλέον είναι αν αυτός ο λόγος είναι ο βέλτιστος που μπορούμε να επιτύχουμε στις εφαρμογές γραφικών. Στη παράγραφο που ακολουθεί υπολογίζουμε την εκμετάλλευση ζωνικού εύρους με την προτεινόμενη οργάνωση[10] και δείχνουμε ότι είναι μεγαλύτερη από το ανωτέρω όριο.

Στην ανάλυσή μας υποθέτουμε συσχέτιση των αιτήσεων κατά Μακροτετράγωνα. Στην περίπτωση της οργάνωσης Φ, η υπό εξέταση εκμετάλλευση εξαρτάται από τις τοπικές συγκρούσεις που οδηγούν στον κύκλο διόρθωσης (ενότητα 5.2), δηλαδή σε ένα χρονικό κενό στο φαινόμενο εύρος ζώνης. Πιο συγκεκριμένα, όταν επιλέξουμε ως βασικό σχήμα πρόσβασης τη στήλη ή το αραιό-m δεν προκύπτουν ποτέ συγκρούσεις, οπότε η εκμετάλλευση είναι πάντα ακριβώς 100%. Για τα υπόλοιπα βασικά σχήματα πρόσβασης η εμφάνιση του επιπρόσθετου κύκλου εξαρτάται από τη θέση (x, y) του Μακροτετραγώνου. Προκειμένου να εκτιμήσουμε το βαθμό εκμετάλλευσης για αυτά τα σχήματα στρεφόμαστε σε μια στατιστική ανάλυση, στην οποία υποθέτουμε ότι οι αιτήσεις του αλγορίθμου για Μακροτετράγωνα ακολουθούν μια ομοιόμορφη κατανομή επάνω στην εικόνα (ισοπίθανες θέσεις). Όπως έχουμε αποδείξει στην ενότητα 5.2, όταν το Μακροτετράγωνο βρίσκεται στη θέση (x, y) με $x \bmod m = 0$ τότε δεν εμφανίζονται συγκρούσεις, ανεξαρτήτως σχήματος πρόσβασης. Τα Μακροτετράγωνα αυτά έχουν πιθανότητα εμφάνισης $\frac{1}{m}$ και οδηγούν σε 100% εκμετάλλευση του ζωνικού εύρους. Για τα υπόλοιπα Μακροτετράγωνα χρειαζόμαστε $mn + 1$ κύκλους σάρωσης, οποιοδήποτε βασικό σχήμα πρόσβασης κι αν

[10]με τη Φ, πρακτικά, οι επεξεργαστικές μονάδες δεν μπορούν να ανακτούν B στοιχεία σε κάθε κύκλο ρολογιού, αφού ανά τακτά διαστήματα υπεισέρχονται κύκλοι διόρθωσης συγκρούσεων. Έτσι, παρότι $B = E$, πρακτικά δεν έχουμε 100% εκμετάλλευση ζωνικού εύρους.

επιλέξουμε (εκτός στήλης και αραιού-m). Τα Μακροτετράγωνα αυτά έχουν πιθανότητα εμφάνισης $\frac{m-1}{m}$ και οδηγούν σε εκμετάλλευση κατά $\frac{mn}{mn+1}$. Συνυπολογίζοντας τα παραπάνω καταλήγουμε ότι η μέση εκμετάλλευση του ζωνικού εύρους με την προτεινόμενη οργάνωση μνήμης είναι

$$\mathrm{E(Util)} \geq \frac{m^2n+1}{m^2n+m} = \frac{B+\frac{1}{m}}{B+1} \tag{5.3}$$

Η τιμή αυτή είναι μεγαλύτερη από τη βέλτιστη που μπορούμε να συμπεράνουμε από τη σχετική βιβλιογραφία (δηλαδή, $\frac{B}{B+1}$) κατά έναν παράγοντα ίσο με $\frac{B+1/m}{B}$.

Κεφάλαιο 6

Εφαρμογές της Προτεινόμενης Οργάνωσης Μνήμης

Στο παρόν κεφάλαιο εξετάζουμε με πιο πρακτικό τρόπο τις επιδόσεις και τα οφέλη της προτεινόμενης οργάνωσης Φ. Χρησιμοποιούμε παραδείγματα κι εφαρμογές από τον πραγματικό κόσμο για να δείξουμε ότι, σε ορισμένες περιπτώσεις, η Φ μπορεί να λειτουργήσει όπως τις βέλτιστες χρονικά λύσεις της βιβλιογραφίας που χρησιμοποιούν επιπρόσθετες τράπεζες και 'δύσκολους' αριθμούς. Φυσικά, η Φ αποφεύγει το επίβαρο κόστος που εισάγουν οι εν λόγω οργανώσεις, γεγονός που την καθιστά προτιμότερη και πιο αποδοτική λύση. Προς την κατεύθυνση αυτή, εξειδικεύουμε τη χρήση της Φ στην μονάδα εκτίμησης κίνησης (ΕΚ) που εισήχθη στο κεφάλαιο 3 και δείχνουμε τη βελτίωση της προτεινόμενης αρχιτεκτονικής μέσα από πειραματικές μετρήσεις. Περαιτέρω, μελετάμε περιπτώσεις όπου η Φ οδηγεί σε τοπικές συγκρούσεις προκειμένου να εκτιμήσουμε τις καθυστερήσεις που εισάγονται στο εκάστοτε σύστημα. Δείχνουμε ότι, σε αυτές τις περιπτώσεις, το χρονικό επίβαρο κόστος της λύσης Φ είναι μικρότερο από το υλικό επίβαρο κόστος των λύσεων επιπρόσθετων τραπεζών (σε απόλυτα νούμερα). Για τους σκοπούς του κεφαλαίου επιλέγουμε να εξετάσουμε χαρακτηριστικά παραδείγματα από δυο πολύ μεγάλα ερευνητικά πεδία που, γενικότερα, εντάσσονται στις εφαρμογές γραφικών: τη συμπίεση εικονοροών και τα ψηφιακά φίλτρα εικόνας.

6.1 Εκτίμηση κίνησης σε πραγματικό χρόνο

Η πρώτη μελέτη περίπτωσης που εξετάζουμε αφορά στην πλέον απαιτητική λειτουργία ενός συμπιεστή εικονοροών πραγματικού χρόνου, δηλαδή στη μηχανή χρονικής πρόβλεψης και στους αλγορίθμους ταιριάσματος περιοχών. Όπως έχουμε ήδη αναφέρει, είναι σημαντικό η συγκεκριμένη μηχανή να διαθέτει τοπική μνήμη προκειμένου να επιταχύνει τις διαδικασίες της και να μην υπερφορτώνει τον ξένιο κωδικοποιητή με αιτήσεις. Η προτεινόμενη οργάνωση Φ μπορεί να υποστηρίξει αποδοτικά αυτή τη λειτουργία. Μάλιστα, είναι δυνατόν να χρησιμοποιήσουμε τη Φ έτσι, ώστε να αποφύγουμε το επίβαρο υλικό κόστος των επιπρόσθετων τραπεζών άλλων λύσεων, χωρίς να επιβαρύνουμε το σύστημα με χρονικό κόστος (χωρίς τοπικές συγκρούσεις).

Έστω εφαρμογή που ακολουθεί το πρότυπο H.264/AVC κι εκμεταλλεύεται τα περισσότερα από τα υπάρχοντα εργαλεία χρονικής πρόβλεψης: πολλαπλά καρέ αναφοράς, κατάτμηση πλακιδίων, ψηφιακή παρεμβολή. Επίσης, έστω ότι οι απαιτήσεις της εφαρμογής και του συστήματος καθορίζουν το βαθμό παραλληλοποίησης της επεξεργασίας, περίπου, στα 8 εικονοστοιχεία ανά κύκλο. Οργανώνουμε τοπική μνήμη 8 τραπεζών χρησιμοποιώντας τη Φ με m=4 και n=2 (Εικόνα 5.1). Η συγκεκριμένη οργάνωση επιτρέπει απεριόριστη πρόσβαση στα σχήματα: αραιό-4, πολυτετράγωνο-1, στήλες και γραμμές μήκους 7 (βλ. ενότητα 4.2). Έτσι, μπορούμε να ενεργοποιήσουμε το αραιό-4 για να σαρώσουμε οποιαδήποτε υποψήφια περιοχή 16×16 εικονοστοιχείων σε 32 κύκλους[1]. Επίσης, μπορούμε να ενεργοποιήσουμε το πολυτετράγωνο-1 για τη σάρωση οποιουδήποτε υποπλακιδίου 8×8 σε 8 κύκλους. Συνεπώς, η εν λόγω οργάνωση μνήμης δύναται να εξυπηρετήσει οποιαδήποτε σειρά αιτήσεων του αλγορίθμου ακέραιας αναζήτησης χωρίς να τον καθυστερήσει, ανεξαρτήτως στρατηγικής ψαξίματος κι ανεξαρτήτως μεθόδου/αποφάσεων κατάτμησης του πλακιδίου.

Όσον αφορά στον αλγόριθμο κλασματικής αναζήτησης, κάνουμε μια γενική –λογική– υπόθεση σχετικά με τη στρατηγική που ακολουθεί η υποκείμενη μηχανή για τη ψηφιακή παρεμβολή μιας περιοχής 16×16 ή 8×8 υποεικονοστοιχείων. Έστω ότι η περιοχή που αιτείται ο αλγόριθμος παράγεται γραμμή προς γραμμή (ή στήλη προς στήλη αν η μετατόπισή της είναι κατακόρυφη). Με τα φίλτρα 6 εισόδων του H.264 [11], η παραγωγή μιας γραμμής (στήλης) 16 υποεικονοστοιχείων απαιτεί ανάγνωση 2+16+3=21 εικονοστοιχείων ακέραιας θέσης. Μπορούμε να συλλέξουμε τα δεδομένα αυτά σε 3 διαδοχικούς κύκλους

[1]εδώ, η χρήση αραιών σχημάτων για τη σάρωση κρίνεται προτιμότερη από τη χρήση συμπαγών, καθώς επιτρέπουν αποδοτικότερη υποδειγματοληψία και υποβοηθούν τις τεχνικές σύγκρισης. Επιβεβαιώνουμε τον ισχυρισμό με πραγματικές μετρήσεις στην υποενότητα 6.1.1

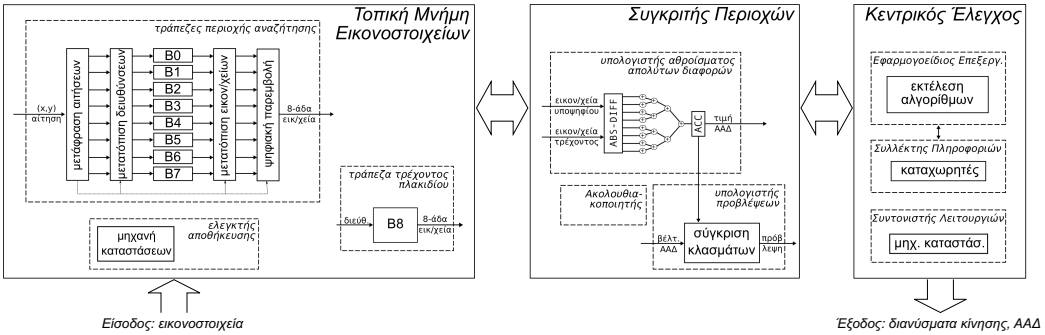
με προσβάσεις σχήματος γραμμής (στήλης) μήκους 7, από οποιαδήποτε θέση επάνω στην περιοχή αναζήτησης. Παρατηρούμε όμως ότι, τον ίδιο αριθμό κύκλων πρόσβασης θα χρειαζόμασταν κι αν είχαμε υλοποιήσει τη χρονικά βέλτιστη λύση της βιβλιογραφίας με μια επιπρόσθετη τράπεζα ($B = 9$, ενότητα 2.3.1). Δηλαδή η Φ, παρότι αποφεύγει το υλικό κόστος της ένατης τράπεζας, δεν εισάγει χρονικό επίβαρο κόστος στο σύστημα. Το ίδιο παρατηρούμε κι όταν η αίτηση του αλγορίθμου αφορά σε υποψήφια περιοχή εμβαδού 8×8: η παραγωγή μιας γραμμής (στήλης) 8 υποεικονοστοιχείων απαιτεί 2+8+3=13 δεδομένα, τα οποία μπορούν να ανακτηθούν σε 2 κύκλους, είτε με τη Φ, είτε με τη λύση επιπρόσθετης τράπεζας.

Συνοψίζοντας, στην παραπάνω μελέτη περίπτωσης είδαμε ότι η χρήση της προτεινόμενης Φ αντί των λύσεων επιπρόσθετης τράπεζας της βιβλιογραφίας (ενότητα 2.3.1) οδηγεί σε πιο αποδοτικές οργανώσεις (με μικρότερο υλικό κόστος) χωρίς να επιβαρύνει το σύστημα ή τον αλγόριθμο με χρονικές καθυστερήσεις (λειτουργεί ως χρονικά βέλτιστη).

6.1.1 Ενσωμάτωση στην προτεινόμενη μονάδα ΕΚ

Συγκεκριμενοποιούμε το παράδειγμα της παρούσας ενότητας στην υλισμική μονάδα ΕΚ που προτείνει το κεφάλαιο 3. Διατηρώντας τη γενική αρχιτεκτονική της εν λόγω μονάδας ΕΚ, οργανώνουμε την τοπική μνήμη σύμφωνα με τη Φ για $B = 8$. Αναπτύσσουμε εκ νέου τη μονάδα ΕΚ σε διατάξεις προγραμματιζόμενων ολοκληρωμένων κυκλωμάτων (FPGA) κι αξιολογούμε το κόστος και τις επιδόσεις της. Συγκρίνουμε τα αποτελέσματα τόσο με άλλες εργασίες της βιβλιογραφίας, όσο και με την έκδοση του κεφαλαίου 3, ώστε να εκτιμήσουμε το κέρδος που αποκομίζουμε κάνοντας χρήση της οργάνωσης Φ.

Στην Εικόνα 6.1 παρουσιάζεται η αρχιτεκτονική της μονάδας ΕΚ, έτσι όπως αυτή τροποποιείται για τις ανάγκες του παραδείγματος. Το δομοστοιχείο κεντρικού ελέγχου παραμένει ίδιο με αυτό που παρουσιάσαμε στην ενότητα 3.1: περιέχει μια μηχανή καταστάσεων για τη σειριακή εκκίνηση των εσωτερικών διεργασιών, μια μηχανή καταστάσεων για τη συλλογή πληροφοριών, και τον εφαρμογοείδιο επεξεργαστή που υλοποιεί το προτεινόμενο σύνολο εντολών για υποστήριξη αλγορίθμων ταιριάσματος περιοχών. Μοναδική αλλαγή αποτελεί η προσθήκη μιας νέας εντολής, της *SUBSAMPLE*, η οποία καθορίζει το βαθμό υποδειγματοληψίας των 16×16 περιοχών που θα εξεταστούν στη συνέχεια του προγράμματος. Οι περιοχές μπορούν να υποδειγματοληπτηθούν κατά $1/1$, $1/2$, ή $1/4$ (256, 128, ή 64 εικονοστοιχεία), ώστε να προκύψει αντίστοιχη μείωση στο χρόνο εξέτασής τους. Η υποδειγματοληψία γίνεται με βάση τα αραιά ορθογώνια



Εικόνα 6.1: Αρχιτεκτονική της εξελιγμένης μονάδας ΕΚ.

που προσφέρει η μνήμη Φ κι ακολουθεί τη φόρμα της 'σκακιέρας' (η πλέον αποδοτική για απλή αποδεκάτιση της εικόνας).

Το δομοστοιχείο σύγκρισης περιοχών μεγέθους 16×16 τροποποιείται έτσι, ώστε να επεξεργάζεται δυο οκτάδες εικονοστοιχείων ανά κύκλο ρολογιού (μια από την περιοχή αναζήτησης και μια από το τρέχον πλακίδιο). Η αλλαγή αυτή συντελείται προκειμένου να βρίσκεται ο ρυθμός διεκπεραιωτικότητάς του σε συμφωνία με το ζωνικό εύρος της νέας τοπικής μνήμης (υποδιπλάσιο εκείνου της ενότητας 3.1). Οπότε, διατηρώντας τη μέθοδο παραλληλοποίησης του αθροίσματος απολύτων διαφορών (ΑΑΔ), μειώνουμε το μέγεθος του δέντρου υπολογισμού στο μισό: από τις 16 εισόδους υλοποιούμε μόνο 8. Η εσωτερική μηχανή καταστάσεων που λαμβάνει την περιοχή-αίτηση (x, y) από τον εφαρμογοείδιο επεξεργαστή ετοιμάζει πλέον έως και 32 αιτήσεις εικονοστοιχείων προς την τοπική μνήμη (αντί για 16) όταν πρόκειται για εξέταση ακέραιας μετατόπισης. Σημειώνουμε ότι όταν εξετάζεται κλασματική μετατόπιση το πλήθος των αιτήσεων δεν αυξάνεται σε σχέση με εκείνο της ενότητας 3.1 γιατί η νέα οργάνωση μνήμης, παρότι διαθέτει μικρότερα σχήματα πρόσβασης, επιτρέπει τον ίδιο ρυθμό παρεμβολής εικονοστοιχείων (=4 ανά κύκλο) χάρη στη δυνατότητα ανάκτησης γραμμών και στηλών. Τέλος, για λόγους βελτιστοποίησης, ελαττώνουμε το μέγεθος του συγκριτή κλασμάτων του ελαχίστου ΑΑΔ για την τεχνική υποθετικής εκτέλεσης εντολών: πλέον πραγματοποιούνται μόνο δυο προβλέψεις[2] καταναλώνοντας ελάχιστους πόρους υλικού.

Οι σημαντικότερες αλλαγές στην αρχιτεκτονική της νέας μονάδας ΕΚ συντελούνται στην τοπική της μνήμη. Τοποθετούμε και πάλι 8 τράπεζες, αλλά τώρα

[2]στο $1/4$ και στο $1/2$ της συσσώρευσης ΑΑΔ. Τα δυο αυτά σημεία υπολογίστηκαν ως βέλτιστα μετά από στατιστική εξέταση πολλών συνδυασμών. Για το σκοπό αυτό αναπτύξαμε κατάλληλο λογισμικό μοντέλο ακρίβειας κύκλων για τη τη μονάδα ΕΚ (cycle accurate model).

κάθε μια έχει πλάτος 8 διφύων κι όχι 32 (ένα εικονοστοιχείο ανά διεύθυνση). Άμεση συνέπεια της μείωσης του πλάτους είναι ότι πλέον δε χρειαζόμαστε τους πολυπλέκτες του πέμπτου σταδίου της σωλήνωσης (Εικόνα 3.3) για την απόρριψη αχρείαστων δεδομένων. Συνεπώς, απαλείφεται ένα σημαντικό ποσό υλικών πόρων κι ένας κύκλος καθυστέρησης (στάδιο σωλήνωσης). Πιο συγκεκριμένα, οργανώνουμε τη μνήμη της 48×48 περιοχής αναζήτησης κατά Φ με m=4 και n=2 (Εικόνα 5.1). Υλοποιούμε κυκλώματα υποστήριξης τριών διαφορετικών σχημάτων με απεριόριστη πρόσβαση: γραμμές μήκους 7, στήλες μήκους 7, και πολυτετράγωνο-1 (με 8 δεδομένα). Το πολυτετράγωνο χρησιμεύει στη μερική ή/και στην ολική σάρωση υποψηφίων ακέραιας μετατόπισης κι όπως θα δείξουμε παρακάτω, είναι πιο αποδοτικό από τα συμπαγή σχήματα. Οι γραμμές και οι στήλες μήκους 7 (=1+4+2) χρησιμεύουν στην παραγωγή 4 υποεικονοστοιχείων με οριζόντια ή κατακόρυφη μετατόπιση, αντίστοιχα, κατά 1/4, 1/2, ή 3/4. Για την παρεμβολή χρησιμοποιούμε τα 4 φίλτρα της ενότ. 3.1, χωρίς να απαιτείται εναλλαγή (transpose) των στοιχείων του σχήματος πρόσβασης ανάλογα με την κατεύθυνση της μετατόπισης (μείωση κόστους). Το κύκλωμα παρεμβολής τοποθετείται πλέον στο πέμπτο στάδιο της σωλήνωσης της τοπικής μνήμης (Εικόνα 6.1). Στο πρώτο στάδιο, το κύκλωμα μετάφρασης της αίτησης (x, y) τροποποιείται ώστε να υπολογίζει τις διευθύνσεις των 8 εικονοστοιχείων και την πρώτη τους τράπεζα ανάλογα με το ζητούμενο σχήμα πρόσβασης· υπολογίζει τη συνάρτηση Φ, καθώς και τη συνάρτηση 4.5 για διευθυνσιοδότηση εντός των τραπεζών. Οι κυλινδρικοί μετατοπιστές του δευτέρου και τετάρτου σταδίου χρησιμοποιούν τον αριθμό Φ που υπολόγισε ο μεταφραστής κι εκτελούν την ίδια λειτουργία με αυτή που περιγράψαμε στην ενότητα 3.1 (με μικρότερο κόστος λόγω του μικρότερου πλάτους εξόδου των νέων τραπεζών). Τα δεδομένα της τοπικής μνήμης ανανεώνονται κατά στήλες πλακιδίων (ενότ. 3.1). Τέλος, το τρέχον πλακίδιο αποθηκεύεται σε ξεχωριστή τράπεζα 32 διευθύνσεων, με την ομαδοποίηση των εικονοστοιχείων του να ακολουθεί τη φόρμα του πολυτετραγώνου-1 (για ανάγνωση όμοια με αυτή της περιοχή αναζήτησης).

Υλοποιούμε τη βελτιωμένη μονάδα ΕΚ στη συσκευή FPGA Xilinx Virtex-II Pro 40 [62]. Η συχνότητα λειτουργίας αυξάνεται στα 80 MHz (από τα 50 MHz της ενότητας 3.2), κυρίως, λόγω της μείωσης των ασύγχρονων μονοπατιών εντός της σωλήνωσης που ενώνει τις τράπεζες με το συσσωρευτή ΑΑΔ (λιγότεροι πολυπλέκτες για τη δρομολόγηση προς το συγκριτή περιοχών και μικρότερο δέντρο υπολογισμού ΑΑΔ, καθώς και καλύτερη εξισορρόπηση των ενδιάμεσων καταχωρητών). Οι καταναλισκόμενοι πόροι υλισμικού για ολόκληρη τη μονάδα ΕΚ είναι 1,579 slices (34Κ ισοδύναμες πύλες) και 9 bram. Αναλυτικότερα, έχουμε 691 slices για το δομοστοιχείο τοπικής μνήμης, 192 slices για τον συγ-

Πίνακας 6.1: Αποτελέσματα εκτέλεσης αλγορίθμων διαφόρων διαμορφώσεων.

		αλγόριθμος: Diamond						αλγόριθμος: PMVFAST						*Όριο*
		ακέραια αναζήτηση			+ κλασματικές θέσεις			ακέραια αναζήτηση			+ κλασματικές θέσεις			@25fps,
Ανάλυση	*Στατιστικά*	ΠΤΔ=1	ΠΤΔ=2	ΠΤΔ=4	ΠΤΔ=1	ΠΤΔ=2	ΠΤΔ=4	ΠΤΔ=1	ΠΤΔ=2	ΠΤΔ=4	ΠΤΔ=1	ΠΤΔ=2	ΠΤΔ=4	80MHz
352×288	PSNR	32.71	32.69	32.34	33.42	33.31	32.87	32.91	32.88	32.60	33.67	33.57	33.18	
	κύκλοι/MB	502	383	341	713	499	410	282	238	220	501	359	295	8080
720×576	PSNR	33.31	33.29	33.13	33.59	33.56	33.33	33.54	33.53	33.41	33.92	33.87	33.67	
	κύκλοι/MB	684	479	410	982	637	498	334	269	242	640	431	335	1975
1280×720	PSNR	33.44	33.42	33.28	33.68	33.64	33.44	34.44	34.40	34.24	34.77	34.70	34.47	
	κύκλοι/MB	786	535	446	1115	708	538	355	281	251	692	459	349	889
1920×1088	PSNR	32.60	32.57	32.45	32.82	32.77	32.58	34.53	34.46	34.32	34.82	34.74	34.52	
	κύκλοι/MB	890	587	481	1246	772	580	408	310	273	772	500	377	392

κριτή περιοχών, και 696 slices για τον κεντρικό έλεγχο. Η επαναδιάρθρωση της μονάδας ΕΚ για την υποστήριξη μόνο απλών αλγορίθμων ταιριάσματος οδηγεί σε υλοποίηση με μόνο 1,176 slices (εξοικονόμηση κόστους 25%). Τα παραπάνω αποτελέσματα καθιστούν εμφανή τη μείωση πόρων που συνεπάγεται η χρήση της Φ στην κατασκευή της τοπικής μνήμης: συγκρίνοντας με τα αποτελέσματα του κεφαλαίου 3 βλέπουμε μείωση 38%. Ίδια ποσοστιαία μείωση έχουμε και στο κόστος του συγκριτή περιοχών (η μείωση περιορίζεται από τους πόρους των μηχανών καταστάσεων που ελέγχουν τη λειτουργία των δυο δομοστοιχείων). Συνολικά για την προτεινόμενη μονάδα ΕΚ πετύχαμε μείωση κόστους 26% κι αυξήσαμε τη διεκπεραιωτικότητά της έως και 33% (σε πλακίδια/δευτερόλεπτο, για αλγοριθμικά αποτελέσματα ίδιας ποιότητας).

Η σύγκριση με τις λοιπές λύσεις της βιβλιογραφίας επιβεβαιώνει τις διαπιστώσεις της ενότητας 3.2. Πιό συγκεκριμένα, βλέπουμε ότι η εργασία [35] βελτιώνει τα αποτελέσματα της προγενέστερης εργασίας [34] (ίδια ερευνητική ομάδα), αλλά παρέχει αρκετά χαμηλότερο ρυθμό διεκπεραιωτικότητας από αυτόν που πετυχαίνει η προτεινόμενη λύση, παρότι απαιτεί 23% περισσότερους πόρους FPGA. Στις εργασίες [41] [42] παρουσιάζεται ένας εφαρμογοείδιος επεξεργαστής με δυνατότητα επαναδιάρθρωσης, με παράλληλους υπολογισμούς, και με σωληνωτή δομή. Όμως, η απλουστευμένη οργάνωση της μνήμης του επιτρέπει ανάγνωση μόνο κατά γραμμές και δεν επιτρέπει τον υπολογισμό των υποεικονοστοιχείων κατά τη μεταφορά (on-the-fly) ούτε την υποδειγματοληψία της εξεταζόμενης περιοχής. Ως εκ τούτου, οι λύσεις [41] [42] εμφανίζουν μεγαλύτερο κόστος από αυτό της προτεινόμενης όταν πρόκειται για εφαρμογές πραγματικού χρόνου και υψηλής ανάλυσης (HDTV) με απαιτήσεις κλασματικής αναζήτησης. Ακόμα και συγκρινόμενη με μη προγραμματιζόμενες λύσεις, η προτεινόμενη μπορεί να αποδειχθεί οικονομικότερη: π.χ., η VLSI λύση [29] χρησιμοποιεί σχεδόν το διπλάσιο αριθμό από λογικές πύλες για να επιτύχει τον ίδιο ρυθμό διεκπεραιωτικότητας με την παρούσα.

Στον Πίνακα 6.1 καταγράφονται τα αποτελέσματα από μια σειρά μετρήσεων που αφορούν στις επιδόσεις χρόνου της μονάδας ΕΚ. Χρησιμοποιούμε έναν αντιπροσωπευτικό απλό αλγόριθμο (τον Diamond) κι έναν χαρακτηριστικό προηγμένο αλγόριθμο (τον PMVFAST), την εκτέλεση των οποίων επαναλαμβάνουμε με τρεις διαφορετικούς παράγοντες υποδειγματοληψίας (ΠΥΔ=1|2|4 για υποδειγματοληψία $\frac{1}{1}|\frac{1}{2}|\frac{1}{4}$), με ή χωρίς την εκτέλεση της *SUBPIXEL*. Χρησιμοποιούμε πολλά γνωστά βίντεο διαφορετικών αναλύσεων (π.χ., foreman, pedestrian, riverbed, κ.α.) και καταγράφουμε το μέσο όρο των κύκλων που διαρκεί η επεξεργασία ενός πλακιδίου, καθώς και τη μέση ποιότητα του αποτελέσματος (το PSNR της εικόνας-πρόβλεψης). Οι μετρήσεις προέρχονται από λογισμικό μοντέλο ακρίβειας κύκλου και διφύου (cycle & bit accurate) της υλισμικής μονάδας ΕΚ. Οι καταγραφόμενοι κύκλοι συμπεριλαμβάνουν μόνο το χρόνο ανανέωσης της τοπικής μνήμης (υποθέτουμε διεπαφή 64-διφύων με τον ξένιο κωδικοποιητή) και το χρόνο εκτέλεσης του αλγορίθμου, συνεπώς, φέρουν μεγαλύτερη ακρίβεια από τη μέθοδο του εσωτερικού παραθύρου που εφαρμόσαμε στον υλισμικό συμπιεστή για τις μετρήσεις της ενότητας 3.2.

Ο Πίνακας 6.1 δείχνει επιπλέον το πλήθος των διαθέσιμων κύκλων ανά πλακίδιο για επεξεργασία πραγματικού χρόνου ('όριο'). Όπως γίνεται φανερό υπό αυτό το πρίσμα, στα βίντεο πολύ υψηλής ανάλυσης η δυνατότητα υποδειγματοληψίας καθίσταται κρίσιμη. Πιο συγκεκριμένα, παρατηρούμε ότι η υποδειγματοληψία κατά $\frac{1}{2}$ κοστίζει μόνο 0.01-0.07 dB ποιότητας, ενώ εξοικονομεί 16-34% του χρόνου. Οι διαφορές αυτές γίνονται ακόμα μεγαλύτερες όταν εφαρμόζουμε υποδειγματοληψία κατά $\frac{1}{4}$. Η εξέταση οριζοντίων και κατακόρυφων κλασματικών μετατοπίσεων προσθέτει 0.2-0.8 dB ποιότητας, αλλά αυξάνει το χρόνο κατά 20-95%. Επίσης, η υποδειγματοληψία στη περίπτωση της *SUBPIXEL* κοστίζει λίγο περισσότερο σε ποιότητα απ΄ ότι στην ακέραια αναζήτηση, επειδή δεν μπορούμε να ακολουθήσουμε τη αποδοτική φόρμα των αραιών ορθογωνίων (χρησιμοποιούμε σκόρπιες γραμμές/στήλες τεσσάρων υποεικονοστοιχείων).

Με μια παρεμφερή σειρά μετρήσεων εξάγουμε συμπεράσματα σχετικά με την αποδοτικότητα των επί μέρους τεχνικών που χρησιμοποιήσαμε στην αρχιτεκτονική μας σχεδίαση. Η προτεινόμενη υποθετική εκτέλεση εντολών εξοικονομεί έως 20% του χρόνου για τους απλούς αλγόριθμους κι έως 10% για τους προηγμένους. Το κέρδος αυτό είναι όμοιο με εκείνο της διαδεδομένης τεχνικής πρόωρου τερματισμού της συσσώρευσης ΑΑΔ σε βίντεο χαμηλής ανάλυσης. Μάλιστα, σε υψηλή ανάλυση ο πρόωρος τερματισμός καθίσταται λιγότερο αποτελεσματικός (μεγαλύτερη χωρική ομοιογένεια των περιοχών), οπότε το κέρδος της υποθετικής εκτέλεσης γίνεται δυο φορές πιο σημαντικό από αυτό του τερματισμού. Όταν οι δυο τεχνικές συνδυαστούν προσφέρουν

κέρδος έως και 34% (Πίνακας 6.1). Η χρήση φίλτρων ψηφιακής παρεμβολής με 4 απομαστευτές αντί για 6 (6-taps, πρόταση H.264/AVC), σε συνδυασμό με τη δυνατότητα ανάκτησης γραμμών και στηλών από την οργάνωση Φ, εξοικονομεί 32% του χρόνου: παράγονται 4 υποεικονοστοιχεία ανά ανακτώμενη 7-άδα, όχι μόνο 2. Απ΄ την άλλη, το κόστος ποιότητας που εισάγουν στο σύστημα κρίνεται αμελητέο, στα -0.004 dB (αφού εξετάζονται μόνο 4+2 θέσεις με οριζόντια ή κατακόρυφη μετατόπιση). Τέλος, η χρήση αραιών ορθογωνίων για την ολική σάρωση των υποψηφίων (ΠΥΔ=1) καταγράφεται ως αποδοτικότερη από τη χρήση συμπαγών σχημάτων, καθώς μειώνει το χρόνο εκτέλεσης του αλγορίθμου έως και 6%: η κάλυψη της περιοχής με σταδιακή ανάκτηση απομακρυσμένων εικονοστοιχείων υποβοηθά, κυρίως, την αποτελεσματικότητα της υποθετικής εκτέλεσης: οι προβλέψεις κατά τη συσσώρευση του ΑΑΔ γίνονται πιο εύστοχες, καθώς η μερική έκταση της υποψήφιας περιοχής που λαμβάνουν υπόψη φαίνεται πιο πλατιά και πιο ομοιόμορφη.

6.2 Συμπίεση εικονοροών πολύ υψηλής ανάλυσης

Σε εφαρμογές επεξεργασίας εικονοροών πολύ υψηλής ανάλυσης και πλαισιορρυθμού, π.χ., 1920×1080 στα 60 πλαίσια το δευτερόλεπτο, είναι λογικό να έχουμε ακόμα μεγαλύτερες απαιτήσεις μνήμης σε σχέση με αυτές που εξετάσαμε στην προηγούμενη ενότητα. Ειδικά όταν θέσουμε αυστηρούς χρονικούς περιορισμούς, πολλές από τις λειτουργίες του συμπιεστή εικονοροών θα πρέπει να τροφοδοτηθούν με, π.χ., διπλάσιο αριθμό εικονοστοιχείων ανά κύκλο προκειμένου να αντεπεξέλθουν στο διπλάσιο αριθμό πλαισίων ανά δευτερόλεπτο (από fps=30 έχουμε fps=60). Στην μελέτη περίπτωσης που ακολουθεί επεκτείνουμε το παράδειγμα H.264/AVC της προηγούμενης ενότητας σε εφαρμογές πολύ υψηλής ανάλυσης και υποθέτουμε απαίτηση για διπλασιασμό του ζωνικού έυρους της μνήμης, δηλαδή, υποθέτουμε 16 εικονοστοιχεία ανά αίτηση.

Οργανώνουμε την παράλληλη μνήμη της υποτιθέμενης μονάδας εκτίμησης κίνησης μέσω της Φ χρησιμοποιώντας $m = n = 4$ και $B = E = 16$ τράπεζες. Σύμφωνα με τα αποτελέσματα της ενότητας 4.2, η συγκεκριμένη οργάνωση επιτρέπει απεριόριστη πρόσβαση κατά στήλες και κατά αραιό-4 (εδώ, το πολυτετράγωνο-1 είναι ίδιο με το αραιό-4). Επομένως, η σάρωση ενός τυχαίου 8×8 υποπλακιδίου δεν μπορεί να επιτευχθεί μέσω κάποιου σχήματος απεριόριστης πρόσβασης. Έτσι, στο συγκεκριμένο παράδειγμα, η χρήση της Φ προκαλεί κα-

θυστέρηση στην διαδικασία αναζήτησης ταιριάσματος περιοχών. Προκειμένου να μετρήσουμε με ακρίβεια το παραπάνω επίβαρο κόστος, προσδιορίζουμε επακριβώς τις φάσεις του αλγορίθμου αναζήτησης, προσομοιώνουμε τη λειτουργία της μονάδας εκτίμησης κίνησης[3], και συγκρίνουμε τον συνολικό χρόνο λειτουργίας με τον χρόνο που θα χρειαζόταν αν είχαμε ενσωματώσει τη χρονικά βέλτιστη λύση της βιβλιογραφίας των $B = 17$ τραπεζών (ενότητα 2.3.1).

Ο αλγόριθμος αναζήτησης χωρίζεται σε τρεις φάσεις. Στην πρώτη φάση, εκτελείται ο ευρετικός *'Three Step Search'* [3] για την ανεύρεση ταιριάσματος ολόκληρου του 16×16 πλακιδίου. Στη δεύτερη φάση, το πλακίδιο χωρίζεται σε τέσσερα 8×8 υποπλακίδια και, για κάθε ένα από αυτά ξεχωριστά, εκτελείται ο ευρετικός *'Diamond Search'* [3] ξεκινώντας από το άνυσμα κίνησης στο οποίο κατέληξε η πρώτη φάση. Στην τρίτη φάση, εξετάζονται οι 8 κλασματικές μετατοπίσεις ($\pm 1/2$) του κάθε υποπλακιδίου γύρω από τη θέση στην οποία σταμάτησε η δεύτερη φάση.

Κατά την πρώτη φάση, η μονάδα ΕΚ χρησιμοποιεί το αραιό-4 για την άνευ συγκρούσεων σάρωση οποιουδήποτε υποψηφίου πλακιδίου ζητάει ο αλγόριθμος. Συνεπώς, η χρήση της Φ δεν καθυστερεί την εκτέλεση της πρώτης φάσης (μηδενικό επίβαρο κόστος). Κατά τη δεύτερη φάση, η σάρωση των υποψηφίων υποπλακιδίων γίνεται με το αραιό-2. Το σχήμα αυτό οδηγεί σε συγκρούσεις όταν το υποπλακίδιο βρίσκεται στη θέση (x, y) με $x \bmod m \neq 0$, με αποτέλεσμα την καθυστέρηση της σάρωσης κατά ένα κύκλο (μέθοδος διόρθωσης, εν. 5.2). Κατά την τρίτη φάση χρησιμοποιεί τα σχήματα της γραμμής και της στήλης για την τροφοδότηση των φίλτρων ψηφιακής παρεμβολής. Συγκεκριμένα, ο υπολογισμός της παρεμβολής μιας ολόκληρης γραμμής (στήλης) του πλακιδίου (16 νέα εικονοστοιχεία) απαιτεί 21 εικονοστοιχεία, τα οποία ανακτώνται σε δυο κύκλους πρόσβασης από τη μνήμη Φ. Δυο κύκλοι πρόσβασης απαιτούνται και όταν ενσωματώσουμε τη χρονικά βέλτιστη μνήμη με $B = 17$. Παρόμοια, οι δυο οργανώσεις οδηγούν στον ίδιο αριθμό προσβάσεων κατά τον υπολογισμό της παρεμβολή μιας γραμμής (στήλης) ενός υποπλακιδίου (1 κύκλος για 8 νέα στοιχεία). Συνεπώς, δεν εισάγεται επίβαρο χρονικό κόστος από τη Φ στην εκτέλεση της τρίτης φάσης.

Συνοψίζοντας τα παραπάνω, το ακριβές επίβαρο κόστος εξαρτάται από το χρόνο που ξοδεύει ο αλγόριθμος στη φάση Β′, καθώς και από τις θέσεις των υποψηφίων 8×8 μπλοκ που αυτός εξετάζει. Για τον υπολογισμό του, προσομοιώνουμε τη λειτουργία του αλγορίθμου χρησιμοποιώντας ως είσοδο μια

[3] αναπτύξαμε λογισμικό μοντέλο της μονάδας με ακρίβεια κύκλου (cycle accurate model).

Πίνακας 6.2: Εκτέλεση αλγορίθμου ταιριάσματος περιοχών με την προτεινόμενη οργάνωση μνήμης (m=n=4): προσδιορισμός του επίβαρου χρονικού κόστους.

	Φάση Β΄ (υποπλακίδια)		**Σύνολική Λειτουργία**	
Ανάλυση	*Αιτήσεις μνήμης ανά υποψήφιο*	*Επίβαρο κόστος*	*Κύκλοι πρόσβασης μνήμης ανά καρέ*	*Επίβαρο κόστος*
720×576	3.53	16.2 %	1.1×10^6	3.83 %
1280×720	3.68	15.9 %	2.7×10^6	3.59 %
1920×1080	3.77	15.6 %	6.4×10^6	3.49 %

ποικιλία 12 διαφορετικών εικονοροών με αντιπροσωπευτικά μεγέθη πλαισίου[4]. Η προσομοίωση εκτελείται δυο φορές, μια υποθέτοντας μνήμη Φ, και μια υποθέτοντας οργάνωση 17 τραπεζών. Καταγράφουμε το χρόνο που ξοδεύεται για πρόσβαση στη μνήμη στην πρώτη περίπτωση, C_Φ, το χρόνο στη δεύτερη περίπτωση, C_O, και υπολογίζουμε το επίβαρο χρονικό κόστος της Φ ως $\frac{C_\Phi - C_O}{C_\Phi}$. Ο Πίνακας 6.2 παραθέτει τα αποτελέσματα για κάθε μέγεθος πλαισίου ξεχωριστά. Δείχνει το συνολικό αριθμό κύκλων που ξοδεύονται στη μνήμη, καθώς και το ακριβές ποσοστό αυτών που οφείλονται στη διόρθωση των συγκρούσεων της Φ (στήλες 4 και 5 του Πίνακα 6.2). Επίσης, δείχνει στατιστικά στοιχεία σχετικά με τη δεύτερη φάση του αλγορίθμου, στην οποία αποκλειστικά παρατηρείται η καθυστέρηση της Φ. Στη δεύτερη στήλη καταγράφεται ο μέσος αριθμός αραιών-2 αιτήσεων που απαιτούνται για την εξέταση ενός 8×8 υποψηφίου μπλοκ. Ο αριθμός αυτός είναι μικρότερος του 4 εξαιτίας του πρόωρου τερματισμού που εφαρμόζεται κατά την εξέταση[5]. Το γεγονός αυτό αυξάνει ελαφρώς το επίβαρο κόστος της Φ (ξοδεύεται ένας κύκλος διόρθωσης ανά λιγότερους κύκλους πραγματικής σάρωσης), το οποίο αντισταθμίζεται μερικώς από τη φύση του αλγορίθμου αναζήτησης που είναι πολωμένος στο μηδέν[6]. Το ακριβές ποσοστό των κύκλων διόρθωσης καταγράφεται στην τρίτη στήλη. Το ποσοστό μειώνεται με την αύξηση της ανάλυσης της εικόνας, αφού αυξάνεται η χωρική συσχέτιση των γειτονικών εικονοστοιχείων και μειώνεται η απόδοση του πρόωρου τερματισμού. Σε κάθε περίπτωση, η προσομοίωση δείχνει ότι το συνολικό επίβαρο χρονικό κόστος της προτεινόμενης Φ είναι μικρότερο από

[4]Βίντεο: Pedestrian, Rush Hour, Blue Sky, River Bed, από 100 πλαίσια το κάθε ένα. Αναλύσεις: 720×576, 1280×720, 1920×1088 εικονοστοιχεία (για όλα τα βίντεο).

[5]τερματισμός εξέτασης αν η σταδιακή συσσώρευση της μετρικής υπερβεί τη βέλτιστη τιμή.

[6]zero biased. Εμφανίζει τάση εξέτασης των θέσεων με $x \bmod 4 = 0$, με αποτέλεσμα να μειώνεται το ποσοστό των μπλοκ που εμφανίζουν συγκρούσεις, από το ισοπίθανο $\frac{3}{4}$ στο 69%.

το επίβαρο κόστος σε υλικό που εισάγει η λύση της επιπρόσθετης τράπεζας, δηλαδή το $1/17 = 5.9\%$, και κατά πολύ μικρότερο του υλικού επίβαρου των λοιπών λύσεων επιπρόσθετων τραπεζών (που φτάνουν έως 50%, εν. 2.3.1).

6.3 Ψηφιακά φίλτρα εικόνας

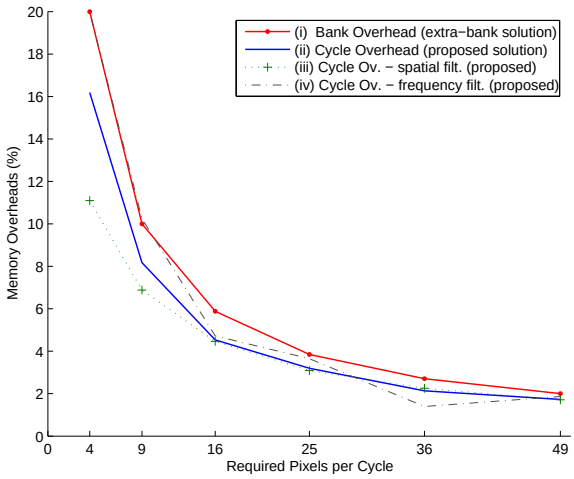
Ένα από τα σημαντικότερα πεδία έρευνας κι ανάπτυξης στην ψηφιακή επεξεργασία εικόνων/εικονοροών έχει ως βάση της τα ψηφιακά φίλτρα. Μια εικόνα μπορεί να φιλτραριστεί είτε στο χωρικό πεδίο (spatial domain), είτε στο πεδίο συχνοτήτων (frequency domain) [71]. Η επιλογή εξαρτάται από το σκοπό κι από τις υπολογιστικές απαιτήσεις της εκάστοτε εφαρμογής. Οι λειτουργίες που εκτελούνται στις δυο αυτές περιπτώσεις εμφανίζουν κάποια κοινά χαρακτηριστικά, τα οποία αναφέρουμε εδώ συνοπτικά.

Το φιλτράρισμα στο χωρικό πεδίο αποτελείται από τη συνέλιξη ενός δισδιάστατου πίνακα $K \times K$ στοιχείων, $K = 2, 3, 4, \ldots$, (πυρήνας του φίλτρου, filter kernel) με την $W \times H$ εικόνα που βρίσκεται υπό επεξεργασία. Πρακτικά, ο πυρήνας του φίλτρου 'ολισθαίνει' επάνω στην εικόνα καλύπτοντας διαδοχικά όλες τις θέσεις της. Σε κάθε θέση πολλαπλασιάζουμε στοιχείο προς στοιχείο τον $K \times K$ πίνακα με το αντίστοιχο $K \times K$ κομμάτι της εικόνας και αθροίζουμε τα αποτελέσματα για να παράξουμε ένα νέο, φιλτραρισμένο, εικονοστοιχείο. Άρα, σε κάθε θέση της παραπάνω ολίσθησης θα προκύπτει μια αίτηση προς τη μνήμη για ένα σύνολο $K \times K$ στοιχείων με αυθαίρετη θέση (x, y) επάνω στην εικόνα (απαίτηση απεριόριστης πρόσβασης).

Το φιλτράρισμα στο πεδίο των συχνοτήτων περιλαμβάνει το δισδιάστατο ευθύ μετασχηματισμό της εικόνας στο πεδίο των συχνοτήτων (π.χ. 2D-FFT), τον πολλαπλασιασμό του με την συχνοτική απόκριση του φίλτρου, και τέλος, τον αντίστροφο μετασχηματισμό του φιλτραρισμένου αποτελέσματος πίσω στο χωρικό πεδίο. Συνήθως, ο δισδιάστατος μετασχηματισμός μιας $W \times H$ εικόνας εκτελείται εφαρμόζοντας $H+W$ διαδοχικούς μονοδιάστατους μετασχηματισμούς σε δυο φάσεις: στην πρώτη, μετασχηματίζουμε μια προς μια τις H γραμμές της εικόνας (ανεξάρτητοι μετασχηματισμοί), ενώ στη δεύτερη φάση εκτελούμε W μονοδιάστατους μετασχηματισμούς στηλών επάνω στο $W \times H$ αποτέλεσμα της πρώτης φάσης. Δηλαδή, απαιτείται η σάρωση της εικόνας πρώτα κατά γραμμές κι έπειτα κατά στήλες. Για λόγους ταχύτητας, η ανάκτηση μιας γραμμής μήκους W (ή στήλης H) μπορεί να πραγματοποιηθεί με διαδοχικές αιτήσεις στη μνήμη για γειτονικές, μη επικαλυπτόμενες, γραμμές B στοιχείων (δεν απαιτείται απεριόριστη πρόσβαση στη μνήμη κατά γραμμές ή κατά στήλες).

Στη μελέτη περίπτωσης που ακολουθεί υποθέτουμε εφαρμογή που χρησιμοποιεί τεχνικές φιλτραρίσματος και στα δυο πεδία –χωρικό και χρονικό. Σύμφωνα με τα παραπάνω, χρειαζόμαστε απεριόριστη πρόσβαση κατά τετράγωνες περιοχές, καθώς και περιορισμένη πρόσβαση κατά γραμμές και κατά στήλες. Σχεδιάζουμε παράλληλη μνήμη μέσω της Φ χρησιμοποιώντας $m = n = K$, όπου K η διάσταση του πυρήνα του υποτιθέμενου χωρικού φίλτρου. Η συγκεκριμένη οργάνωση μας διασφαλίζει πρόσβαση σε συμπαγείς περιοχές $K \times K$, $1 \times K^2$, και $K^2 \times 1$ εικονοστοιχείων. Επομένως, κατά τον μετασχηματισμό της εικόνας μπορούμε να αιτούμαστε διαδοχικές γραμμές (στήλες) K^2 στοιχείων προκειμένου να συλλέξουμε ολόκληρες γραμμές (στήλες) της εικόνας. Οι αιτήσεις-γραμμές εντοπίζονται στις θέσεις (x, y) με $x \bmod K^2 = 0$, κι επομένως δεν οδηγούν σε συγκρούσεις (ενότητα 4.2). Οι αιτήσεις-στήλες εντοπίζονται στις θέσεις (x, y) με $y \bmod K^2 = 0$ και δεν οδηγούν σε συγκρούσεις. Δηλαδή, κατά το φιλτράρισμα στο πεδίο των συχνοτήτων θα εμφανίζεται συνεχώς 100% εκμετάλλευση του ζωνικού εύρους της μνήμης. Κατά το φιλτράρισμα στο χωρικό πεδίο μπορούμε να αιτούμαστε το $K \times K$ τετράγωνο σχήμα πρόσβασης σε συνεχόμενες γειτονικές θέσεις επάνω στην εικόνα (από πάνω προς τα κάτω κι από αριστερά προς τα δεξιά, anti-raster scan). Σε ορισμένα τμήματα της κατακόρυφης σάρωσης προκύπτουν ελλείποντα εικονοστοιχεία τα οποία, συλλογικά, σχηματίζουν κατακόρυφες συμπαγείς στήλες ύψους K^2 (ενότητα 5.3). Οι συγκρούσεις αυτές διορθώνονται κατά ομάδες, ξοδεύοντας 1 κύκλο ανά K^2 κύκλους πραγματικής σάρωσης.

Για την ακριβέστερη εκτίμηση του επίβαρου χρονικού κόστους που εισάγει η προτεινόμενη λύση σε εφαρμογές φίλτρων, προσομοιώνουμε τις παραπάνω λειτουργίες χρησιμοποιώντας δυο διαφορετικές οργανώσεις μνήμης: μια κατά Φ με $B = K^2$, και μια ακολουθώντας τη βέλτιστη χρονικά λύση της επιπρόσθετης τράπεζας, $B = K^2+1$ (ενότητα 2.3.1). Σε κάθε περίπτωση, εκτελούμε διαφορετικούς αλγορίθμους φιλτραρίσματος και μετράμε τους κύκλους που καταναλώνονται για πρόσβαση στη μνήμη. Υπολογίζουμε το επίβαρο κόστος της προτεινόμενης λύσης ως $\frac{C_\Phi - C_O}{C_\Phi}$, όπου C_Φ οι κύκλοι πρόσβασης όταν χρησιμοποιούμε τη Φ και C_O οι κύκλοι της βέλτιστης χρονικά λύσης. Η Εικόνα 6.2 απεικονίζει τα αποτελέσματα λαμβάνοντας υπόψη διαφορετικές απαιτήσεις ζωνικού εύρους για την εφαρμογή, δηλαδή $K = \{2,3,4,5,6,7\}$. Συγκεκριμένα, η καμπύλη (ii) δείχνει το μέσο επίβαρο κόστος της Φ, ενώ οι (iii) και (iv) το επιμερίζουν στα φίλτρα του χωρικού και του συχνοτικού πεδίου, αντίστοιχα. Σημειώνεται ότι το επίβαρο κόστος του συχνοτικού πεδίου δεν οφείλεται σε συγκρούσεις, αλλά στην ικανότητα της λύσης $B = K^2 + 1$ να ανακτά ένα περισσότερο εικονοστοιχείο ανά κύκλο απ΄ ότι η Φ που φέρει $B = K^2$. Επίσης, σημειώνεται ότι



Εικόνα 6.2: Επίβαρο κόστος της λύσης Φ (σε χρόνο) κι επίβαρο κόστος των λύσεων επιπρόσθετης τράπεζας (σε υλικό), για διαφορετικά μεγέθη φίλτρων.

η ανάλυση της εικόνας δεν μετέβαλε ουσιαστικά τα παραπάνω αποτελέσματα (χρησιμοποιήθηκαν εικόνες με ανάλυση από 352×288 έως 1920×1080). Η Εικόνα 6.2 απεικονίζει επίσης, μέσω της καμπύλης (i), το επίβαρο κόστος σε υλικό που εισάγει η λύση της επιπρόσθετης τράπεζας ($= {}^{1}/_{K^2+1}$). Όπως γίνεται φανερό, σε απόλυτη τιμή, το επίβαρο κόστος της προτεινόμενης λύσης είναι πάντα μικρότερο από αυτό της επιπρόσθετης τράπεζας, και κατά πολύ μικρότερο από αυτό που εισάγουν οι λύσεις πρώτου, ή διπλάσιου, αριθμού τραπεζών (ενότητα 2.3.1). Η σύγκριση αυτή είναι ενδεικτική της οικονομίας υλικού που επιτυγχάνεται κατά την ανάπτυξη της παράλληλης μνήμης όταν χρησιμοποιούμε τη Φ.

Κεφάλαιο 7

Επίλογος Διατριβής και Μελλοντική Εργασία

Η παρούσα διατριβή πρότεινε λύσεις σε δυο ξεχωριστά –όχι εντελώς ανεξάρτητα– προβλήματα. Πρώτον, σχεδίασε κι ανέπτυξε σε υλισμικό την αρχιτεκτονική μιας προγραμματιζόμενης μονάδας εκτίμησης κίνησης για εκτέλεση αλγορίθμων ταιριάσματος περιοχών σε πραγματικό χρόνο. Δεύτερον, στοχεύοντας σε ένα ευρύτερο πεδίο εφαρμογών γραφικών, πρότεινε κι απέδειξε τις ιδιότητες μιας βελτιωμένης λύσης στο πρόβλημα της οργάνωσης παράλληλων μνημών με ταυτόχρονη ελαχιστοποίηση πολλαπλών κριτηρίων: τον αριθμό των τραπεζών, το μέγεθός τους, τους απαιτούμενους κύκλους πρόσβασης, καθώς και την πολυπλοκότητα των κυκλωμάτων υποστήριξης της λειτουργίας τους.

Η μονάδα εκτίμησης κίνησης βασίστηκε σε έναν εφαρμογοείδιο επεξεργαστή με ένα μικρό σύνολο σύνθετων εντολών για την υποστήριξη απλών αλγορίθμων ταιριάσματος περιοχών. Η υποστήριξη προηγμένων αλγορίθμων επιτεύχθηκε με επιπρόσθετες εντολές, οι οποίες υλοποιούνται από δομοστοιχειωτά κυκλώματα που ενσωματώνονται στην επαναδιαρθρώσιμη αρχιτεκτονική κατά την τοποθέτησή της στο υλικό. Χάρη στη σωλήνωσή της, στον παραλληλισμό της σε επίπεδο δεδομένων, στην τεχνική της –εξειδικευμένης– υποθετικής εκτέλεσης εντολών που αναπτύξαμε, καθώς και στον μειωμένο αριθμό κύκλων μετάκλησης-αποκωδικοποίησης του συνόλου εντολών που σχεδιάσαμε, η προταθείσα αρχιτεκτονική οδηγεί σε υλοποιήσεις για επεξεργασία πραγματικού χρόνου με μικρότερο κόστος από αυτό που αναφέρουν οι αντίστοιχες λύσεις της βιβλιογραφίας. Η αποδοτικότητα και το κόστος της εν λόγω μονάδας εμφάνισαν σημαντική βελτίωση μετά την ενσωμάτωση της παράλληλης μνήμης που προτάθηκε στο δεύτερο, και μεγαλύτερο, μέρος της διατριβής.

Η οργάνωση παράλληλης μνήμης για αποθήκευση εικόνων βασίστηκε στον ιδανικό αριθμό τραπεζών, $B{=}E$, με στόχο την εξυπηρέτηση οποιασδήποτε αίτησης $E{=}mn$ εικονοστοιχείων κατά {γραμμή,στήλη,ορθογώνιο,αραιό-s} σε έναν κύκλο ρολογιού, όπου m,n,s, αποτελούν ακέραιες παραμέτρους της σχεδίασης. Η προτεινόμενη λύση επέκτεινε τις δυνατότητες πρόσβασης στη μνήμη σε απεριόριστες, ως προς τη θέση, για οποιοδήποτε από τα παραπάνω σχήματα. Προς τη κατεύθυνση αυτή, κατάφερε να αποφύγει την έως σήμερα παγιωμένη τακτική της χρήσης περίσσειας τραπεζών, $B{>}E$, ειδικά όταν αυτή περιελάμβανε πρώτους, ή άλλους, αριθμούς που οδηγούσαν σε πολύπλοκα κυκλώματα διευθυνσιοδότησης και δρομολόγησης των δεδομένων. Η συγκεκριμένη πρόοδος εδράζεται στις ιδιότητες της προτεινόμενης συνάρτησης αντιστοίχισης και σε μια παρατήρηση σχετικά με τις συσχετίσεις που εμφανίζουν οι διαδοχικές αιτήσεις στη μνήμη κατά την εκτέλεση συνήθων αλγορίθμων επεξεργασίας γραφικών. Συνολικά, προέκυψε μια τεχνική διόρθωσης των αναπόφευκτων συγκρούσεων με ένα μικρό χρονικό επίβαρο κόστος, το πολύ έναν κύκλο ανά B αιτήσεις, το οποίο σε απόλυτα νούμερα είναι μικρότερο από το υλικό επίβαρο κόστος των προγενέστερων λύσεων της βιβλιογραφίας. Συνεπώς, η νέα λύση εξυπηρετεί τις ανάγκες των περισσοτέρων εφαρμογών γραφικών παρουσιάζοντας μεγαλύτερη εκμετάλλευση του υλικού της μνήμης με οικονομικότερα κυκλώματα υποστήριξης των λειτουργιών της και με τον ελάχιστο αποθηκευτικό χώρο.

Τα κυριότερα θέματα για μελλοντική έρευνα που απορρέουν από την παρούσα εργασία αφορούν σε: α) προσδιορισμό και μελέτη του ελαχίστου αριθμού συγκρούσεων στη μνήμη δεδομένου του περιορισμού $B{=}E$, των απαιτήσεων των εφαρμογών γραφικών, καθώς και συγκεκριμένων στατιστικών υποθέσεων για την χωρική κατανομή των αιτήσεων των αλγορίθμων, β) προσδιορισμό, μελέτη κι απόδειξη των ιδιοτήτων μιας συνάρτησης αντιστοίχισης σε τράπεζες που να οδηγεί στον ελάχιστο αριθμό συγκρούσεων, γ) κατασκευή συνάρτησης για διευθυνσιοδότηση εντός των τραπεζών, η οποία θα εμφανίζει μικρότερη πολυπλοκότητα και καλύτερη κάλυψη των διευθύνσεων συγκριτικά με τις συναρτήσεις που αναφέρονται στην ενότητα 4.3, δ) μετάβαση της λύσης Φ σε διαφορετικά προβλήματα ή πρακτικές εφαρμογές, όπως π.χ., στη διανομή συχνοτήτων σε δίκτυα τηλεπικοινωνιών με πολλαπλές κεραίες, ή γενικότερα, στο χρωματισμό υπερ-γράφων υπό συγκεκριμένους περιορισμούς, ε) επέκταση της λειτουργικότητας της προταθείσας μονάδας εκτίμησης κίνησης, π.χ., σε αυτόματη επιλογή αλγορίθμου ανάλογα με τα χαρακτηριστικά της εκάστοτε εικονοροής ή σε κατάτμηση του πλακιδίου και υλοποίηση προηγμένων κριτηρίων ταιριάσματος.

Βιβλιογραφία

[1] C. Stiller and J. Konrad. Estimating motion in image sequences. *Signal Processing Magazine, IEEE*, 16:70–91, Jul 1999.

[2] F. Dufaux and F. Moscheni. Motion estimation techniques for digital tv: a review and a new contribution. *Proceedings of the IEEE*, 83:858–876, Jun 1995.

[3] Ian E. G. Richardson. *H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia*. Willey, 2003.

[4] Shan Zhu and Kai-Kuang Ma. A new diamond search algorithm for fast block-matching motion estimation. *IEEE Transactions on Image Processing*, 9(2):287–290, Feb 2000.

[5] Y. Liu and S. Oraintara. Complexity comparison of fast block-matching motion estimation algorithms. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages iii–341–4, 2004.

[6] C. Cheung and L. Po. Novel cross-diamond-hexagonal search algorithms for fast block motion estimation. *IEEE Transactions on Multimedia*, 7(1):16–22, Feb 2005.

[7] A. M. Tourapis, O. C. Au, and M. L. Liou. Highly efficient predictive zonal algorithms for fast block-matching motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(10):934–947, Oct 2002.

[8] A. M. Tourapis, O. C. Au, and M. L. Liou. Predictive motion vector field adaptive search technique (PMVFAST) - enhancing block based motion estimation. In *Proceedings of Visual Communications and Image Processing (VCIP'01)*, 2001.

[9] A. M. Tourapis, O. C. Au, and M. L. Liou. Fast block-matching motion estimation using predictive motion vector field adaptive search technique (PMVFAST), ISO/IEC JTC1/SC29/WG11w MPEG2000/M5866, march, 2000.

[10] I. Ahmad, Weiguo Zheng, Jiancong Luo, and Ming Liou. A fast adaptive motion estimation algorithm. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(3):420–438, Mar 2006.

[11] ITU-T Recommendation H.264. Series H: Audiovisual and multimedia systems, infrastructure of audiovisual services, coding of moving video: Advanced video coding for generic audiovisual services, April, 2003.

[12] B. Kamolrat, W. A. C. Fernando, M. Mrak, and A. Kondoz. 3D motion estimation for depth image coding in 3D video coding. *Consumer Electronics, IEEE Transactions on*, 55:824–830, Aug 2009.

[13] Yen-Kuang Chen. True motion estimation – theory, application, and implementation. PhD disserteation, Princeton University, Nov, 1998.

[14] A. Heinrich, C. Bartels, R. J. van der Vleuten, C. N. Cordes, and G. de Haan. Optimization of hierarchical 3DRS motion estimators for picture rate conversion. *IEEE Journal of Selected Topics in Signal Processing*, 5(2):262–274, Apr 2011.

[15] H. Jung, K. Sung, K. S. Nayak, E. Y. Kim, and J. C. Ye. k-t FOCUSS: A general compressed sensing framework for high resolution dynamic MRI. *Magnetic Resonance in Medicine*, 61:103–116, Jan 2009.

[16] Jing Tian and Kai-Kuang Ma. A survey on super-resolution imaging. *Signal, Image and Video Processing*, Spinger, DOI 10.1007/s11760-010-0204-6, Feb, 2011.

[17] K. Hariharakrishnan and D. Schonfeld. Fast object tracking using adaptive block matching. *IEEE Trans. on Multimedia*, 7:853–859, Oct 2005.

[18] B. S. Manjunath, P. Salembier, and T. Sikora. *Introduction to MPEG-7: Multimedia Content Description Interface*. Willey, 2002.

[19] A. M. Amel, B. A. Abdessalem, and M. Abdellatif. Video shot boundary detection using motion activity descriptor. *Journal of Telecommunications*, 2:54–59, Apr 2010.

[20] Xiaomin Wu, Weizhang Xu, Nanhao Zhu, and Zhanxin Yang. A fast motion estimation algorithm for H.264. In *IEEE Intl. Conference on Signal Acquisition and Processing (ICSAP)*, 2010.

[21] W. He, M. Zhao, C. Tsui, and Z. Mao. A scalable frame-level pipelined architecture for FSBM motion estimation. In *IEEE 20th International Conference on VLSI Design*, 2007.

[22] M. Kim, I. Hwang, and S. Chae. A fast VLSI architecture for full-search variable block size motion estimation in MPEG-4 AVC/H.264. In *IEEE Proceedings of the ASP-DAC*, 2005.

[23] C. A. Rahman and W. Badawy. A quarter pel full search block motion estimation architecture for H.264/AVC. In *IEEE International Conference on Multimedia and Expo*, 2005.

[24] Y. W. Huang, S. Y. Chien, B. Yu Hsieh, and Liang-Gee Chen. Global elimination algorithm and architecture design for fast block matching motion estimation. *IEEE Transactions On Circuits and Systems for Video Technology*, 14(6), June 2004.

[25] T. Moorthy and A. Ye. A scalable architecture for variable block size motion estimation on Field-Programmable Gate Arrays. In *Canadian Conf. on Electrical and Computer Engineering, IEEE*, 2008.

[26] A. M. Campos, F. J. Ballester Merelo, M. A. Martinez Peiro, and J. A. Canals Esteve. Integer-pixel motion estimation H.264/AVC accelerator architecture with optimal memory management. *Microprocessors and Microsystems*, 32(2):68–78, Mar 2008.

[27] Z. Liu, Y. Huang, Y. Song, S. Goto, and T. Ikenaga. Hardware-efficient propagate partial SAD architecture for variable block size motion estimation in H.264/AVC. In *ACM Proceedings of the 17th Great lakes symposium on VLSI*, 2007.

[28] G. J. Sullivan and T. Wiegand. Rate-distortion optimization for video compression. *Signal Processing Magazine, IEEE*, 15:74–90, Nov 1998.

[29] B. F. Wu, H. Y. Peng, and T. L. Yu. Efficient hierarchical motion estimation algorithm and its VLSI architecture. *IEEE Transactions on Very Large Scale Integration Systems*, 16(10):1385–1398, Oct 2008.

[30] Huong Ho. Design and implementation of a fast multi-frame hierarchical motion estimation circuit. In *Consumer Electronics (ICCE), 2011 IEEE International Conference on*, pages 527 – 528, 2011.

[31] Nan Yu, Kichul Kim, and Zoran Salcic. A new motion estimation algorithm for mobile real-time video and its fpga implementation. In *TENCON 2004. IEEE Region 10 Conference*, pages 383 – 386, 2004.

[32] S. Ramachandran and S. Srinivasan. FPGA implementation of a novel, fast motion estimation algorithm for real-time video compression. In *ACM Proc. of the 2001 9th International Symposium on FPGAs*, 2001.

[33] C. A. Rahman and W. Badawy. UMHexagonS algorithm based motion estimation architecture for H.264/AVC. In *Fifth International Workshop on System-on-Chip for Real-Time Applications (IWSOC'05)*, 2005.

[34] T. Dias, S. Momcilovic, N. Roma, and L. Sousa. Adaptive motion estimation processor for autonomous video devices. *EURASIP Journal on Embedded Systems*, 2007(1), Jan 2007.

[35] T. Dias, N. Roma, L. Sousa, and M. Ribeiro. Reconfigurable architectures and processors for real-time video motion estimation. *Journal of Real-Time Image Processing*, 2(4):191–205, Dec 2007.

[36] A. Beric, R. Sethuraman, H. Peters, J. van Meerbergen, G. de Haan, and C. A. Pinto. A 27 mW 1.1 mm^2 motion estimator for picture-rate up-converter. In *Proceedings of the 17th International Conference on VLSI Design, vol.17*, pages 1083–1088, 2004.

[37] T. Li, S. Li, and C. Shen. A novel configurable motion estimation architecture for high-efficiency MPEG-4/H.264 encoding. In *IEEE Proceedings of the ASP-DAC*, 2005.

[38] D. Alfonso, F. Rovati, D. Pau, and L. Celetto. An innovative, programmable architecture for ultra-low power motion estimation in reduced memory MPEG-4 encoder. *IEEE Transactions on Consumer Electronics*, 48(3):702–708, Aug 2002.

[39] S. Dutta and W. Wolf. A flexible parallel architecture adapted to block-matching motion-estimation algorithms. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(1):74–86, Feb 1996.

[40] T. Spiteri, G. Vafiadis, and J. L. Nunez-Yanez. A toolset for the analysis and optimization of motion estimation algorithms and processors. In *IEEE Conf. on Field Programmable Logic & App.*, pages 423–428, 2009.

[41] J. L. Nunez-Yanez, E. Hung, and V. Chouliaras. A configurable and programmable motion estimation processor for the H.264 video codec. In *IEEE Conference on Field Programmable Logic Applications*, pages 149–154, 2008.

[42] J. L. Nunez-Yanez, T. Spiteri, and G. Vafiadis. Multi-standard reconfigurable motion estimation processor for hybrid video codecs. *Computers and Digital Techniques, IET*, 5:73–85, March 2011.

[43] J. Waerdt, G. Slavenburg, J. Itegem, and S. Vassiliadis. Motion estimation performance of the TM3270 processor. In *Proceedings of the 2005 ACM symposium on Applied computing, pp. 850-856*, 2005.

[44] J. Choi, N. Togawa, M. Yanagisawa, and T. Ohtsuki. VLSI architecture for a flexible motion estimation with parameters. In *IEEE Proceedings of the 15th International Conference on VLSI Design*, 2002.

[45] P. Budnick and D.J. Kuck. Organization and use of parallel memories. *IEEE Transactions on Computers*, 20(12):1565–1569, Dec 1971.

[46] J. K. Tanskanen, R. Creutzburg, and J. T. Niittylahti. On design of parallel memory access schemes for video coding. *Journal of VLSI Signal Processing Systems, Springer*, 40(2):215–237, Jun 2005.

[47] D. C. VanVoorhis and T. H. Morrin. Memory systems for image processing. *IEEE Transactions on Computers*, (2):113–125, Feb 1978.

[48] J. W. Park. Multiaccess memory system for attached simd computer. *IEEE Transactions on Computers*, 33(4):439–452, Apr 2004.

[49] C. Liu, X. Yan, and X. Qin. An optimized linear skewing interleave scheme for on-chip multi-access memory systems. In *Proceedings of the 17th Great lakes symposium on VLSI*, pages 8–13, 2007.

[50] Ashoke Deb. Multiskewing–a novel technique for optimal parallel memory access. *IEEE Transactions on Parallel and Distributed Systems*, 7(6):595–604, Jun 1996.

[51] R.F. Sproull, I.E. Sutherland, A. Thompson, S. Gupta, and C. Minter. The 8 by 8 display. *ACM Transactions on Graphics*, 2(1):32–56, Jan 1983.

[52] J. Tanskanen, T. Sihvo, J. Niittylahti, J. Takala, and R. Creutzburg. Parallel memory access schemes for h.263 encoder. In *IEEE International Symposium on Circuits and Systems*, pages 691–694, 2000.

[53] J.M. Frailong W. Jalby and J. Lenfant. Xor-schemes: A flexible data organization in parallel memories. In *International Conference on Parallel Processing*, pages 276–283, 1985.

[54] H. Vandierendonck and K. De Bosschere. Xor-based hash functions. *IEEE Transactions on Computers*, 54(7):800–812, Jul 2005.

[55] Charles F. Laywine and Gary L. Mullen. *Discrete mathematics using Latin squares*. John Wiley and Sons, New York, first edition, 1998.

[56] K. Kim and V.K. Prasanna. Latin squares for parallel array access. *IEEE Transactions on Parallel and Distributed Systems*, 4(4):361–370, Apr 1993.

[57] B. Chor, C. E. Leiserson, R. L. Rivest, and J. B. Shearer. An application of number theory to the organization of raster-graphics memory. *Journal of the ACM*, 33(1):86–104, Jan 1986.

[58] R. Raghavan and J.P. Hayes. On randomly interleaved memories. In *Proc. Supercomputing '90*, pages 49–58, 1990.

[59] N. Topham and A. Gonzalez. Randomized cache placement for eliminating conflicts. *IEEE Transactions on Computers*, 48(2):185–192, Feb 1999.

[60] A. Vitkovski, G. Kuzmanov, and G. Gaydadjiev. Memory organization with multi-pattern parallel accesses. In *Design, Automation and Test in Europe, IEEE*, pages 1414–1419, 2008.

[61] F. Thomson Leighton. *Introduction to Parallel algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann, first edition, 1991.

[62] Xilinx Inc. Virtex-II Pro and virtex-II Pro X FPGA user guide, UG012 (v4.2), http://www.xilinx.com, November 2007.

[63] K. Babionitakis, G. Lentaris, K. Nakos, N. Vlassopoulos, D. Reisis, G. Doumenis, G. Georgakarakos, and I. Sifnaios. A real-time motion estimation FPGA architecture. *Journal of Real-Time Image Processing*, 3(1-2):3–20, Mar 2008.

[64] S. Wong, B. Stougie, and S. Cotofana. Alternatives in FPGA-based SAD implementations. In *IEEE International Conference on Field-Programmable Technology, pp. 449-452*, 2002.

[65] Tom M. Apostol. *Introduction to Analytic Number Theory*. Springer-Verlag, first edition, 1976.

[66] V. Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, first edition, 2005.

[67] D. E. Knuth. *Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Addison-Wesley, 1981.

[68] Application Note: Spartan 3 FPGA Family. Using Block RAM in Spartan-3 FPGAs, XAPP463 (v1.1.2), http://www.xilinx.com, 2003.

[69] M. Kharbutli, K. Irwin, Y. Solihin, and J. Lee. Using prime numbers for cache indexing to eliminate conflict misses. In *10th International Symposium on High Performance Computer Architecture (HPCA'04), IEEE*, pages 288–299, 2004.

[70] J. F. Wakerly. *Digital Design: Principles and Practices*. 3rd Ed., Prentice-Hall, 2000.

[71] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Prentice-Hall, 2002.

Παράρτημα/Appendix
δημοσιεύσεις/publications

{1} *"A Graphics Parallel Memory Organization Exploiting Request Correlations"*, G. Lentaris, D. Reisis., IEEE Transactions on Computers, vol. 59, no. 6, pp. 762-775, June 2010

{2} *"Programmable Motion Estimation Architecture"*, A. Drolapas, G. Lentaris, D. Reisis. Electronics Circuits and Systems, 16th IEEE International Conference on, pp. 323-326, December 2009

{3} *"A Real-Time Motion Estimation FPGA Architecture"*, K. Babionitakis, G. Doumenis, G. Georgakarakos, G. Lentaris, K. Nakos, D. Reisis, J. Sifnaios, N. Vlassopoulos. Special Issue on Field-Programmable Technology, Journal of Real-Time Image Processing, Issues 1-2, Volume 3, Springer, 2008

{4} *"A Real-Time H.264/AVC VLSI Encoder Architecture"*, K. Babionitakis, G. Doumenis, G. Georgakarakos, G. Lentaris, K. Nakos, D. Reisis, J. Sifnaios, N. Vlassopoulos. Journal of Real-Time Image Processing, Issues 1-2, Volume 3, Springer, 2008

{5} *"An Efficient H.264 VLSI Advanced Video Encoder"*, K. Babionitakis, G. Lentaris, K. Nakos, D. Reisis, N. Vlassopoulos, G. Doumenis, G. Georgakarakos, J. Sifnaios. Electronics Circuits and Systems, 13th IEEE International Conference on, pp. 545-548, December 2006

{6} *"Customizing a VLIW Chip Multiprocessor for Motion Estimation Algorithms"*, V. Chouliaras, G. Lentaris, D. Reisis, and D. Stevens. Architecture of Computing Systems (ARCS'11), 24th Intl Conf on, Workshop Proc, pp. 178-184, Como Italy, February 2011

{7} *"A Control-Theoretic Approach for Efficient Design of Filters in DAC and Digital Audio Amplifiers"*, K. Tsakalis, N. Vlassopoulos, G. Lentaris, D. Reisis. Circuits Systems and Signal Processing, Springer-Birkhauser Boston, vol. 30, no. 2, pp. 421-438, April 2011

{8} *"An approach for efficient design of digital amplifers"*, N. Vlassopoulos, D. Reisis, G. Lentaris, G. Tombras, E. Prosalentis, N. Ritas, K. Tsakalis. Circuits and Systems, IEEE International Symposium on, 4 pp., May 2006

{9} *"Design and comparison of FFT VLSI architectures for SoC telecom applications with different flexibility, speed and complexity trade-offs"*, M. Rovini, S. Saponara, L. Fanucci, A. Karachalios, G. Lentaris and D. Reisis, Circuits, Systems, and Signal Processing, Springer-Birkhauser Boston, accepted for publication (June 2011)